

# Développement d'applications sous Android



Arnaud Doniec

IMT Lille Douai

# Objectifs pédagogiques

---

- Maîtriser les outils du kit de développement pour Android
- Être capable de développer une application simple sous Android
- Connaître et prendre en compte les contraintes de développement sur mobiles



# Organisation du cours

---

## Planning:

- 6h de cours / TD
- 3 séances de TP ( $3 \times 4$  heures)

## Ce qui vous est demandé:

- revoir vos cours, TD et TP de programmation objet
- venir en TP avec vos machines perso et le SDK installé

## Évaluation:

- à définir ensemble



# Plan

---

## Introduction

Caractéristiques de l'OS

"L'écosystème" Android

Installation et configuration du SDK

Documentation du SDK

## Premiers pas avec le SDK

Première application sous Android

Utilisation de l'émulateur

Les activités

## Conception d'une interface graphique

Principaux contrôles et boîtes de dialogues

Gestion des évènements

Animation graphique

## Fonctionnalités avancées

Utilisation des ressources

Multi-activités

Utilisation des capteurs du device

## Déploiement d'une application

## Les contraintes liées au développement sur mobile

## Pour aller plus loin...



# Plan

---

## Introduction

### Caractéristiques de l'OS

"L'écosystème" Android

Installation et configuration du SDK

Documentation du SDK

## Premiers pas avec le SDK

Première application sous Android

Utilisation de l'émulateur

Les activités

## Conception d'une interface graphique

Principaux contrôles et boîtes de dialogues

Gestion des évènements

Animation graphique

## Fonctionnalités avancées

Utilisation des ressources

Multi-activités

Utilisation des capteurs du device

## Déploiement d'une application

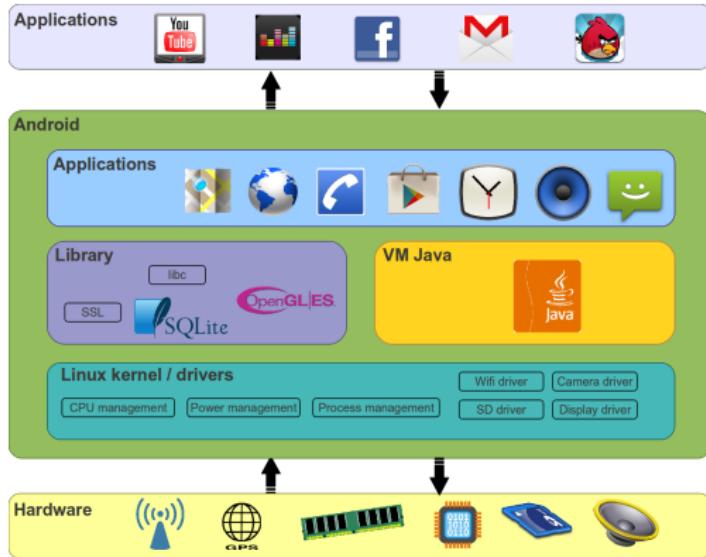
## Les contraintes liées au développement sur mobile

## Pour aller plus loin...



# Qu'est-ce qu'Android ?

*"Android is a software stack for mobile devices that includes an operating system, middleware and key applications." [source developer.android.com]*



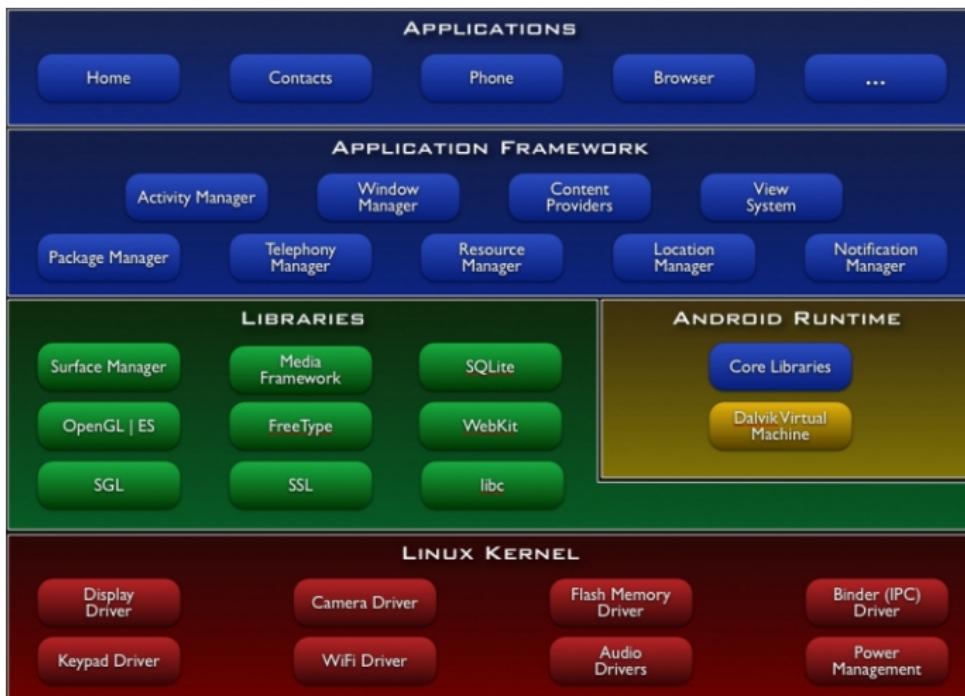
# Origines et choix technologiques

---

- les premiers développements d'Android ont été initiés au sein d'une start-up éponyme, rachetée par Google en 2005.
- Android repose sur un noyau linux
- les API fournies par Android permettent l'accès à un vaste ensemble de fonctionnalités:
  - l'accès aux réseaux GSM, EDGE, 3G, 4G pour la téléphonie et la data ainsi que le support du Wifi, du Bluetooth et du NFC
  - un service de localisation (via une puce GPS ou via une triangulation par le réseau)
  - l'accès aux différents capteurs équipant les smartphones modernes: boussole, accéléromètre, magnétomètre, caméra, ...
  - la manipulation (lecture et enregistrement) d'une grande variété de formats multimédia
  - une bibliothèque graphique complète pour la 2D et la 3D (avec support d'OpenGL)
  - un système de base de données (SQLite) pour le stockage et l'extraction rapide de données enregistrées sur le device
  - ...



# Architecture en couche



# Historique des versions d'Android

---

- 1.0 (*Apple Pie*) : fin 2007, réservée au développeur
- 1.1 (*Banana Bread*) : sortie fin 2008 sur HTC
- 2.3 - 2.3.7 (*Gingerbread*) : 2010
- 4.0.1 - 4.0.4 (*Ice Cream Sandwich*) : 2011
- 4.1 - 4.3.1 (*Jelly Bean*) : 2012
- 4.4.2 - 4.4.4 (*Kitkat*) : 2013
- 5.0 (*Lollipop*) : 2014
- 6.0 (*Marshmallow*) : 2015
- 7.0 (*Nougat*) : 2016
- 8.0 (*Oreo*) : 2017
- 9.0 (*Pie*) : 2018
- 10.0 (*Q*) : 2019



# Plan

---

## Introduction

Caractéristiques de l'OS

"L'écosystème" Android

Installation et configuration du SDK

Documentation du SDK

## Premiers pas avec le SDK

Première application sous Android

Utilisation de l'émulateur

Les activités

## Conception d'une interface graphique

Principaux contrôles et boîtes de dialogues

Gestion des évènements

Animation graphique

## Fonctionnalités avancées

Utilisation des ressources

Multi-activités

Utilisation des capteurs du device

## Déploiement d'une application

## Les contraintes liées au développement sur mobile

## Pour aller plus loin...



# Stratégie commerciale de Google (1/3)

---

- Pour Google, Android n'est juste qu'un service de plus qui vient conforter son "business model" reposant sur la publicité:
  - Android intègre et/ou requiert des services Google (exemple messagerie Gmail), du coup le smartphone devient un canal de diffusion supplémentaire pour la publicité
  - richesse des données fournies par Android pour le ciblage publicitaire (ex: position par GPS)
- Contrairement à Apple, Google a fait le choix de ne pas lier son OS à un matériel spécifique
  - en 2019, plus d'une trentaine de constructeurs de téléphone intègrent Android à leur matériel
  - possibilité de déployer Android sur des terminaux "low-cost"
  - possibilité de personnalisation d'Android par les constructeurs (drivers spécifiques, surcouche graphique, etc)
  - en moyenne il se vend deux fois plus de smartphones sous Android que sous iOS (chiffres au 2nd semestre 2019 pour la france: Android 82% contre 18% pour iOS [*source Kantar*]...)
  - ... mais en 2019, l'App Store a généré 25,5 milliards \$ contre 14,2 pour Google Play



## Stratégie commerciale de Google (2/3)

---

- contrairement à ses autres concurrents (Apple, Blackberry, Microsoft), Google mise sur l'aspect Open Source de son OS:
  - promotion au sein du consortium *Open Handset Alliance* qui a pour objectif de développer des normes ouvertes pour la téléphonie mobile (<https://www.openhandsetalliance.com>)
  - sollicitation de Google de la communauté de développeurs Open Source (<https://source.android.com>)
- le déploiement d'applications est plus souple que sous iOS:
  - vérification des applications plus light dans Google Play
  - présence de "market" alternatifs: F-droid, Samsung Apps, etc.
  - possibilité de déploiement d'applications directement par copie de fichiers .apk



## Stratégie commerciale de Google (3/3)

diversification des terminaux:



# Présentation de Google Play

---

- ensembles de services facilitant la mise en relation des développeurs d'applications et des utilisateurs:
  - plateforme de téléchargements d'apps accessible via une application Android dédiée ou le web (<http://play.google.com>)
  - côté utilisateur:
    - gestion des achats: sauvegarde et synchronisation des apps déjà achetées, facturation
    - gestion des modes de paiements
  - côté développeur:
    - rapports mensuels sur les transactions et paiements effectués par les acheteurs
    - rapport journalier sur les ventes estimées



# Plan

---

## Introduction

Caractéristiques de l'OS

"L'écosystème" Android

## Installation et configuration du SDK

Documentation du SDK

## Premiers pas avec le SDK

Première application sous Android

Utilisation de l'émulateur

Les activités

## Conception d'une interface graphique

Principaux contrôles et boîtes de dialogues

Gestion des évènements

Animation graphique

## Fonctionnalités avancées

Utilisation des ressources

Multi-activités

Utilisation des capteurs du device

## Déploiement d'une application

## Les contraintes liées au développement sur mobile

## Pour aller plus loin...



# Android Studio

---

Google propose un bundle contenant:

- Version d'IntelliJ (IDE semblable à Netbeans et Eclipse) personnalisée
- le SDK (Software Development Kit)

A télécharger ici: <https://developer.android.com/studio>



# Contenu du SDK

---

- les API Android (= les bibliothèques)
- les outils de développements:
  - DDMS (Dalvik Debug Monitoring Service)
  - AAPT (Android Asset Packaging Tool)
  - ADB (Android Debug Bridge)
  - Traceview
  - Dx
  - ...
- un émulateur
- une documentation
- des exemples de code



# SDK Android et Java

---

Le SDK repose sur un sous-ensemble de Java:

- certaines class ont disparu
- certaines class / méthodes n'ont pas le même comportement (ex: `System.out.println()`)
- AWT et Swing (librairies graphiques) n'existent pas sous Android



# Dalvik

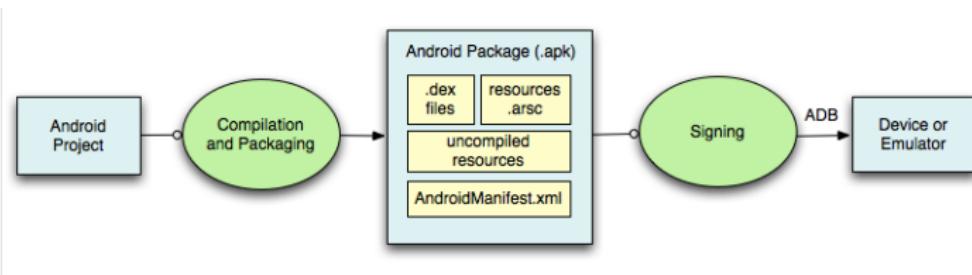
---

Pour ses besoins, Google a redéveloppé sa propre machine virtuelle, appellée Dalvik:

- elle est open-source
- elle n'est pas compatible avec une JVM classique type Java SE
- Dalvik  $\neq$  Java Micro Edition (utilisé sur les mobiles (non-android), PDA, set-top box)
- elle exécute un fichier .dex (équivalent des .class mais avec un bytecode différent, plus light)
- toutes les class d'une application Android sont encapsulées dans un seul fichier .dex



# Génération d'une application Android



# Plan

---

## Introduction

Caractéristiques de l'OS

"L'écosystème" Android

Installation et configuration du SDK

**Documentation du SDK**

## Premiers pas avec le SDK

Première application sous Android

Utilisation de l'émulateur

Les activités

## Conception d'une interface graphique

Principaux contrôles et boîtes de dialogues

Gestion des évènements

Animation graphique

## Fonctionnalités avancées

Utilisation des ressources

Multi-activités

Utilisation des capteurs du device

## Déploiement d'une application

## Les contraintes liées au développement sur mobile

## Pour aller plus loin...



# Doc en ligne

---

Point d'entrée:

<http://developer.android.com/develop/index.html>

- La javadoc:

<http://developer.android.com/reference/packages.html>

- Les guides couvrant différentes thématiques: la création d'interfaces graphiques, l'utilisation des fonctions multimedia, l'accès aux capteurs, etc: <http://developer.android.com/guide/components/index.html>

- Des tutoriaux pour débuter:

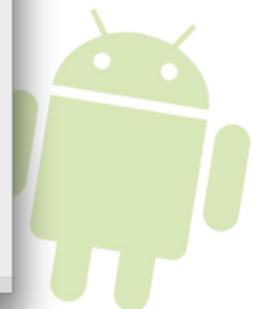
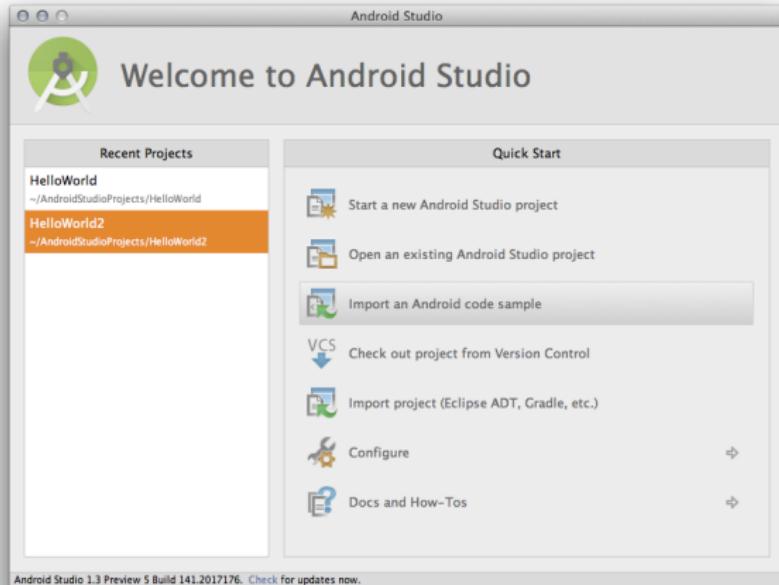
<http://developer.android.com/training/index.html>



# Samples project

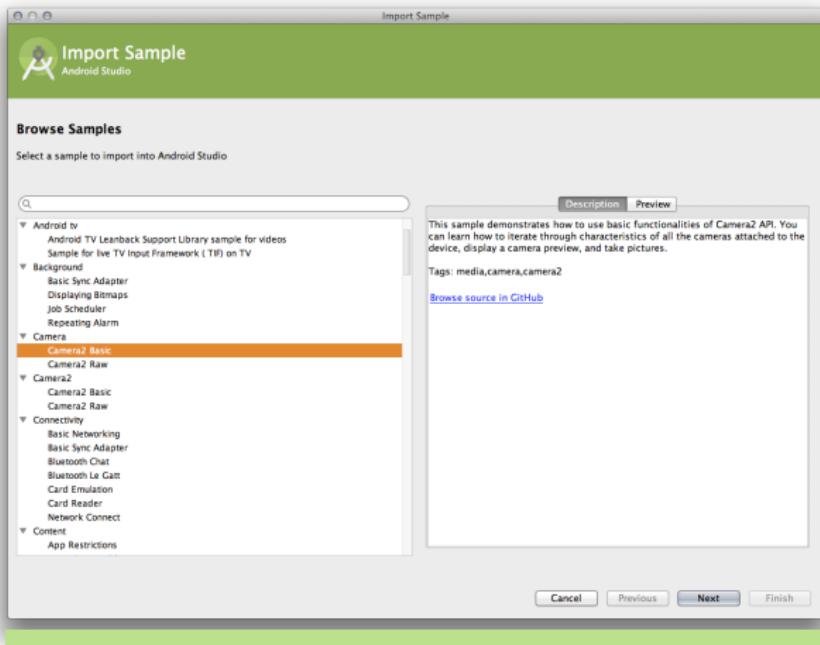
---

Ce sont des mini-projets fournis avec le SDK et classés par thématique: gestion des capteurs, communication,



# Samples project

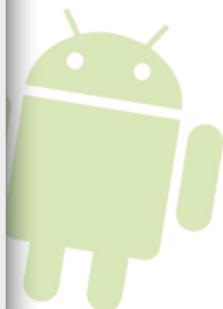
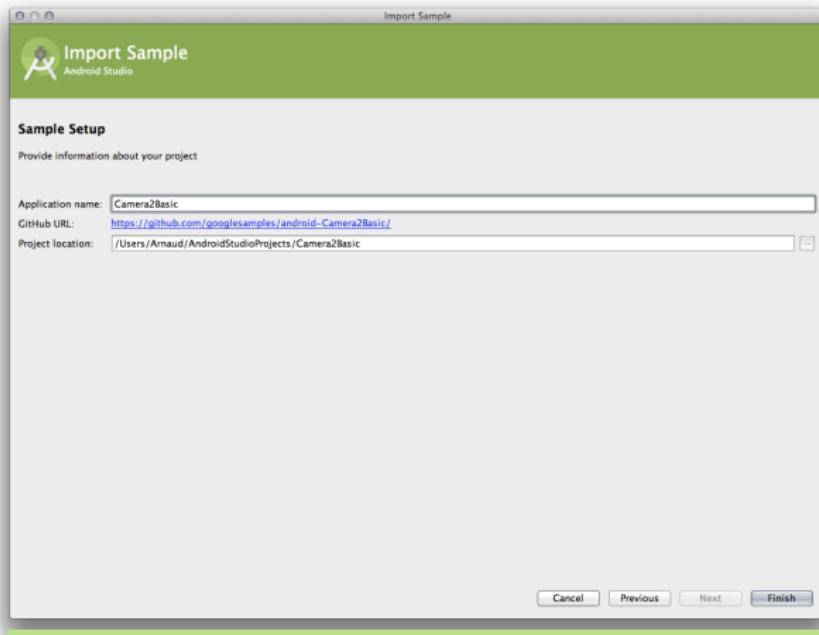
Ce sont des mini-projets fournis avec le SDK et classés par thématique: gestion des capteurs, communication,



# Samples project

---

Ce sont des mini-projets fournis avec le SDK et classés par thématique: gestion des capteurs, communication,



# Plan

---

## Introduction

Caractéristiques de l'OS

"L'écosystème" Android

Installation et configuration du SDK

Documentation du SDK

## Premiers pas avec le SDK

Première application sous Android

Utilisation de l'émulateur

Les activités

## Conception d'une interface graphique

Principaux contrôles et boîtes de dialogues

Gestion des évènements

Animation graphique

## Fonctionnalités avancées

Utilisation des ressources

Multi-activités

Utilisation des capteurs du device

## Déploiement d'une application

## Les contraintes liées au développement sur mobile

## Pour aller plus loin...



# Plan

---

## Introduction

Caractéristiques de l'OS

"L'écosystème" Android

Installation et configuration du SDK

Documentation du SDK

## Premiers pas avec le SDK

**Première application sous Android**

Utilisation de l'émulateur

Les activités

## Conception d'une interface graphique

Principaux contrôles et boîtes de dialogues

Gestion des évènements

Animation graphique

## Fonctionnalités avancées

Utilisation des ressources

Multi-activités

Utilisation des capteurs du device

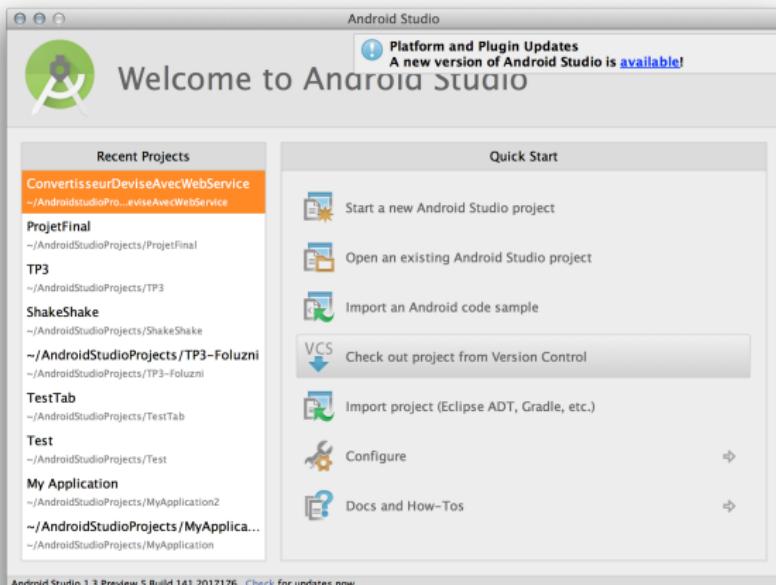
## Déploiement d'une application

## Les contraintes liées au développement sur mobile

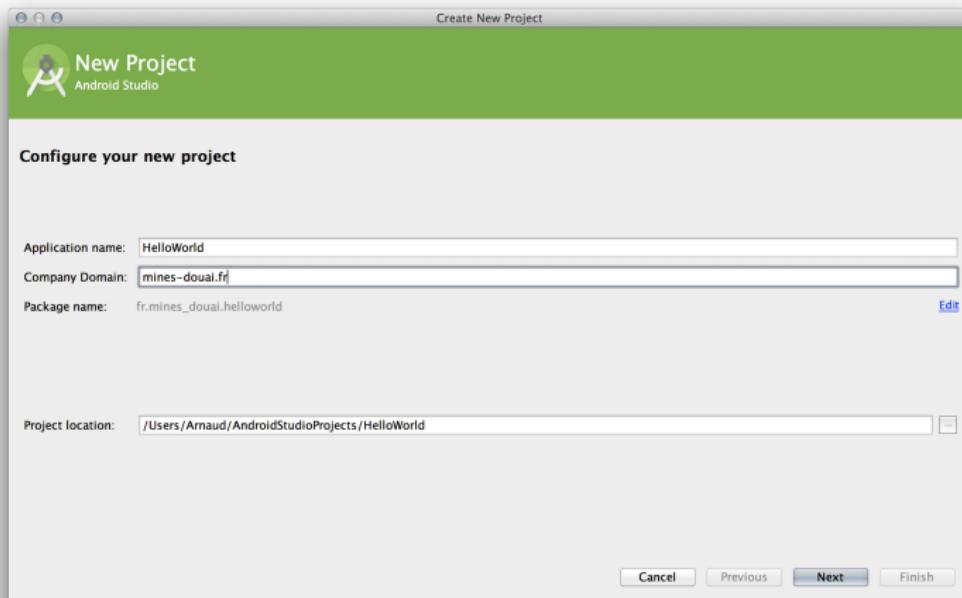
## Pour aller plus loin...



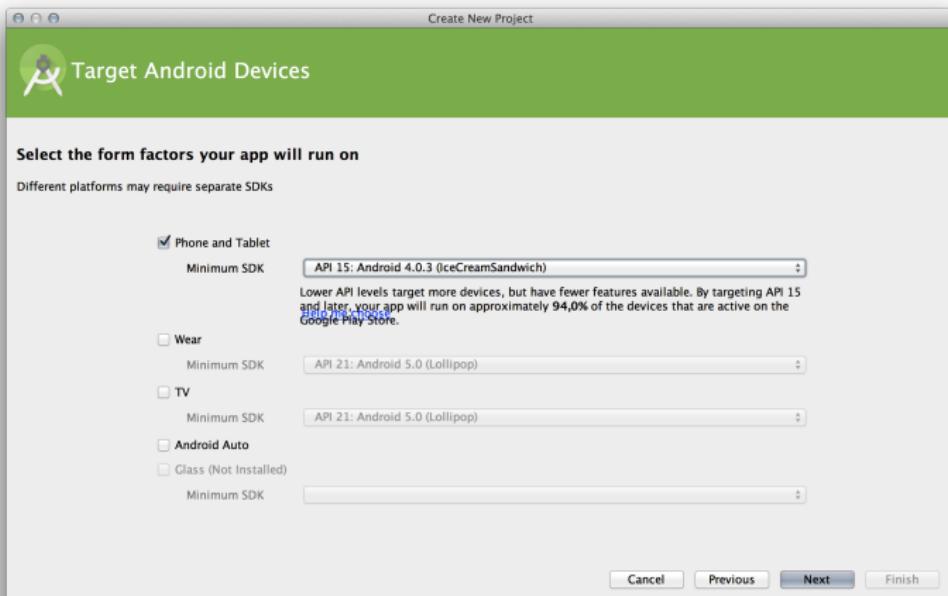
# Hello world: Crédit d'un projet sous Android Studio



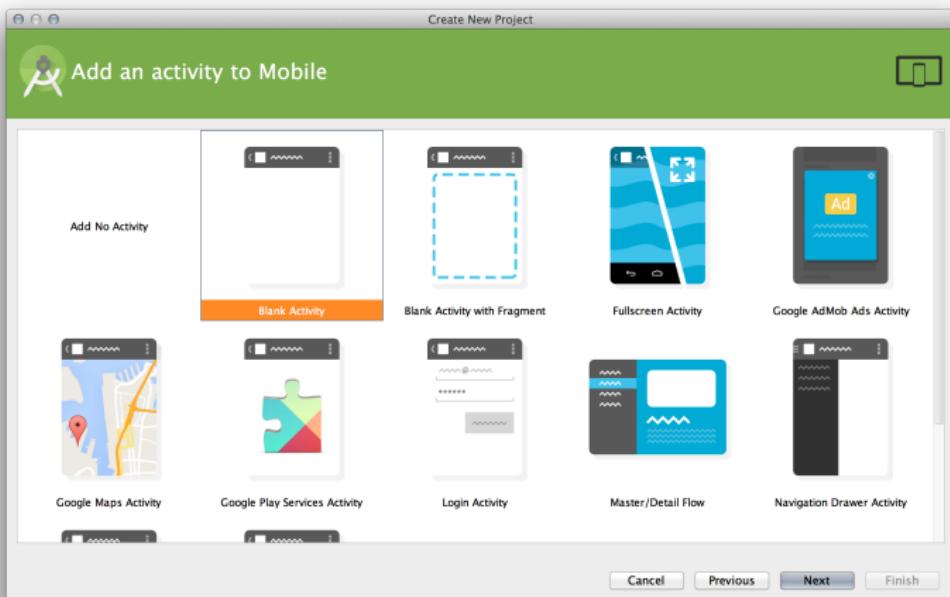
# Hello world: Crédit d'un projet sous Android Studio



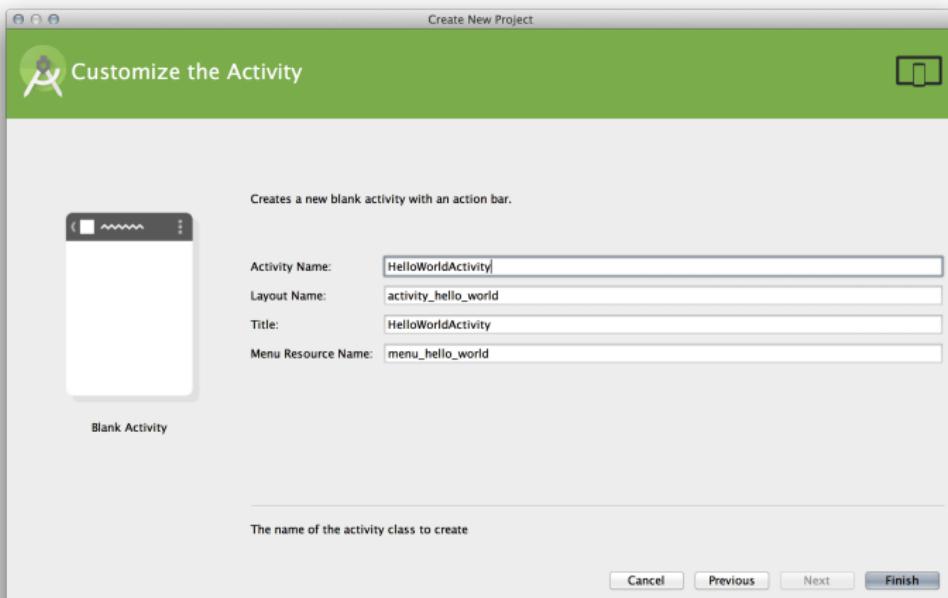
# Hello world: Crédit d'un projet sous Android Studio



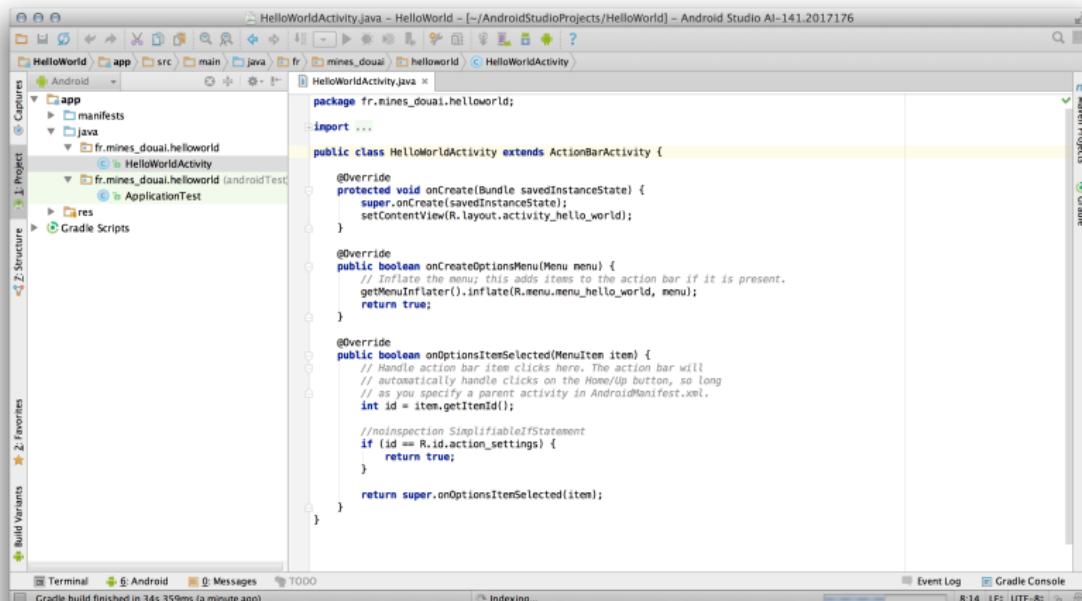
# Hello world: Crédit d'un projet sous Android Studio



# Hello world: Crédit d'un projet sous Android Studio



# Hello world: Crédit d'un projet sous Android Studio



The screenshot shows the Android Studio interface with the following details:

- Project Structure:** The left sidebar shows the project structure for "HelloWorld". It includes the "app" module with its sub-directories: "manifests", "java", "res", and "Gradle Scripts". Inside the "java" directory, there is a package "fr.mines\_douai.helloworld" containing a class "HelloWorldActivity".
- Code Editor:** The main window displays the code for "HelloWorldActivity.java". The code implements the "ActionBarActivity" class and overrides methods like onCreate and onCreateOptionsMenu.
- Bottom Bar:** The bottom bar contains tabs for "Terminal", "Android", "Messages", "Event Log", and "Gradle Console". A message in the Terminal tab states: "Gradle build finished in 34s 359ms (a minute ago)".
- Right Side:** The right side of the interface features a large green Android icon and several status indicators: "8:14", "LF:", and "UTF-8".

# Plan

---

## Introduction

Caractéristiques de l'OS

"L'écosystème" Android

Installation et configuration du SDK

Documentation du SDK

## Premiers pas avec le SDK

Première application sous Android

## Utilisation de l'émulateur

Les activités

## Conception d'une interface graphique

Principaux contrôles et boîtes de dialogues

Gestion des évènements

Animation graphique

## Fonctionnalités avancées

Utilisation des ressources

Multi-activités

Utilisation des capteurs du device

## Déploiement d'une application

## Les contraintes liées au développement sur mobile

## Pour aller plus loin...



# Android Virtual Device

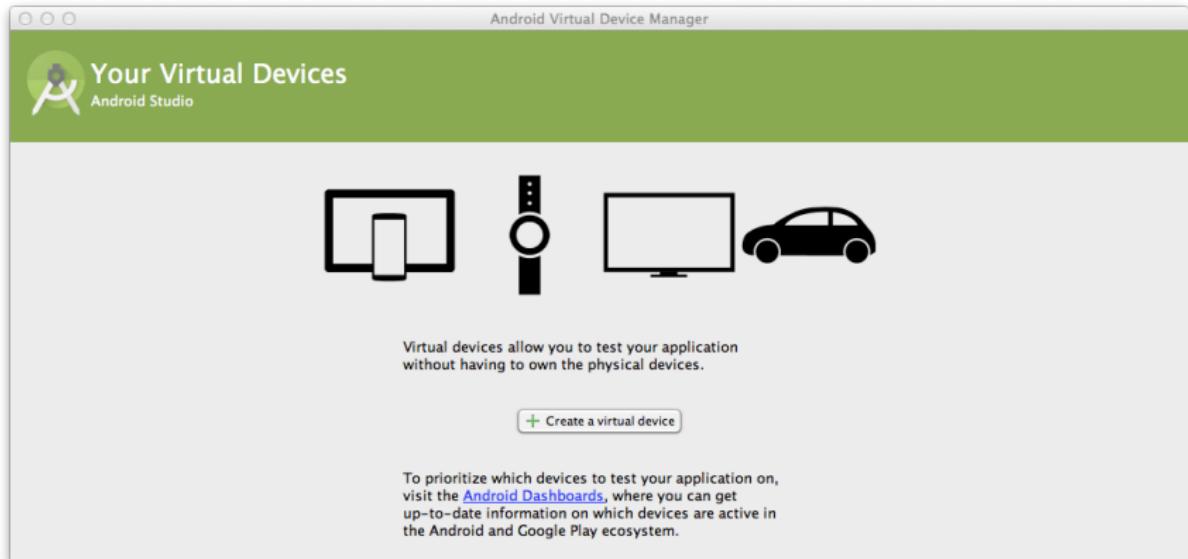
---

Pour tester un programme, deux solutions:

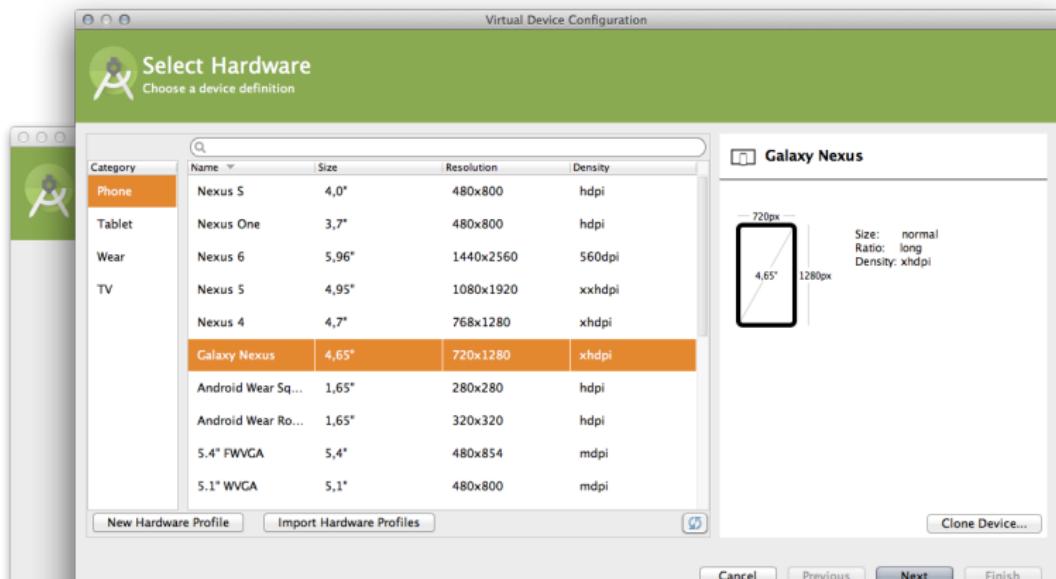
- brancher un device en USB sur le poste de développement
- utiliser un émulateur (Android Virtual Device) qui:
  - reproduit le comportement d'un device (ex: simulation d'appels, envoi de SMS, navigation)
  - peut être créé avec différentes caractéristiques (résolution écran, RAM, CPU, capteurs embarqués...)



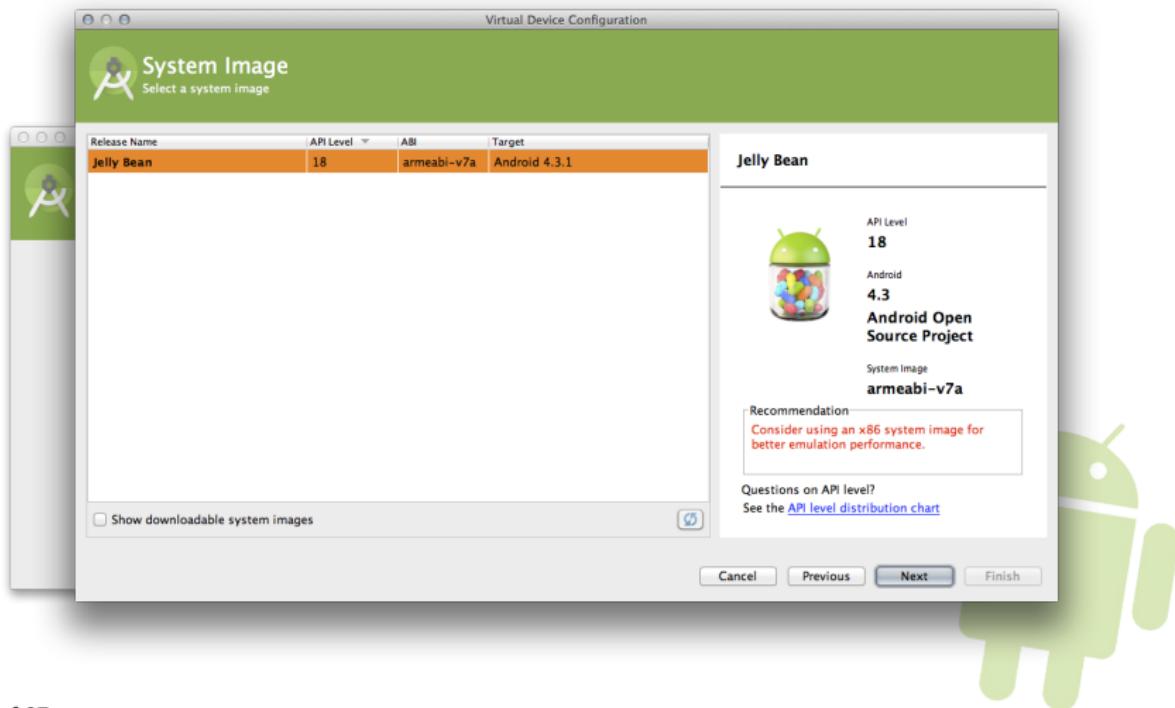
# Création d'un AVD



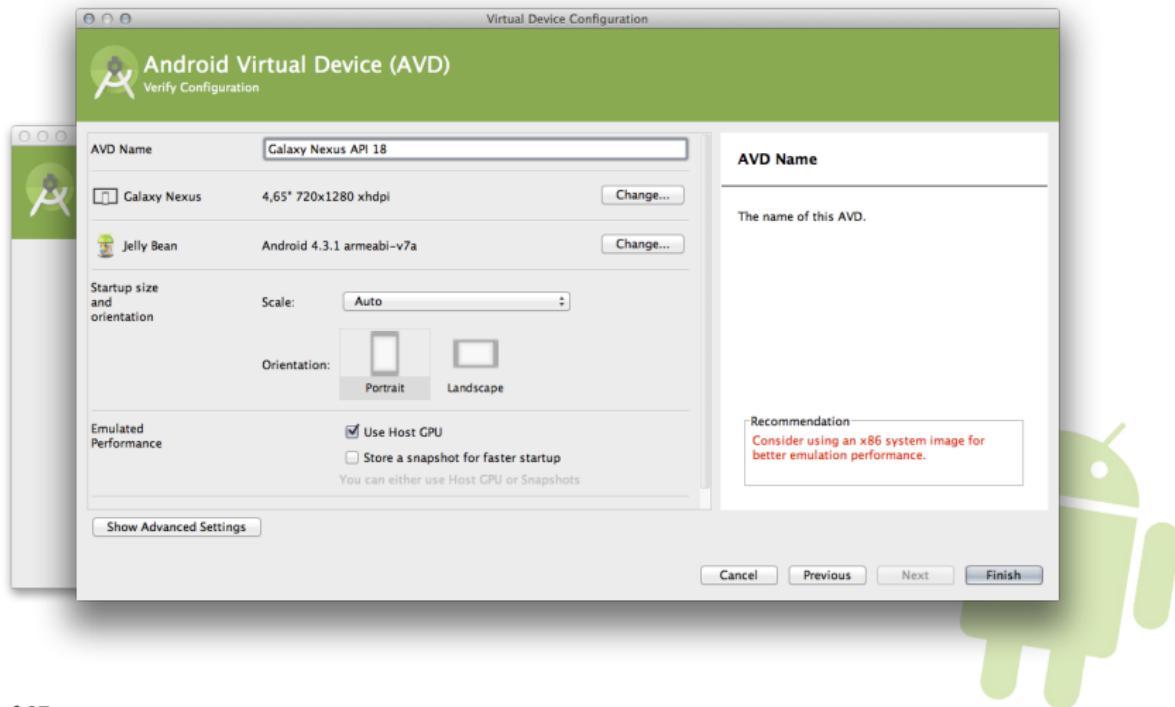
# Création d'un AVD



# Création d'un AVD

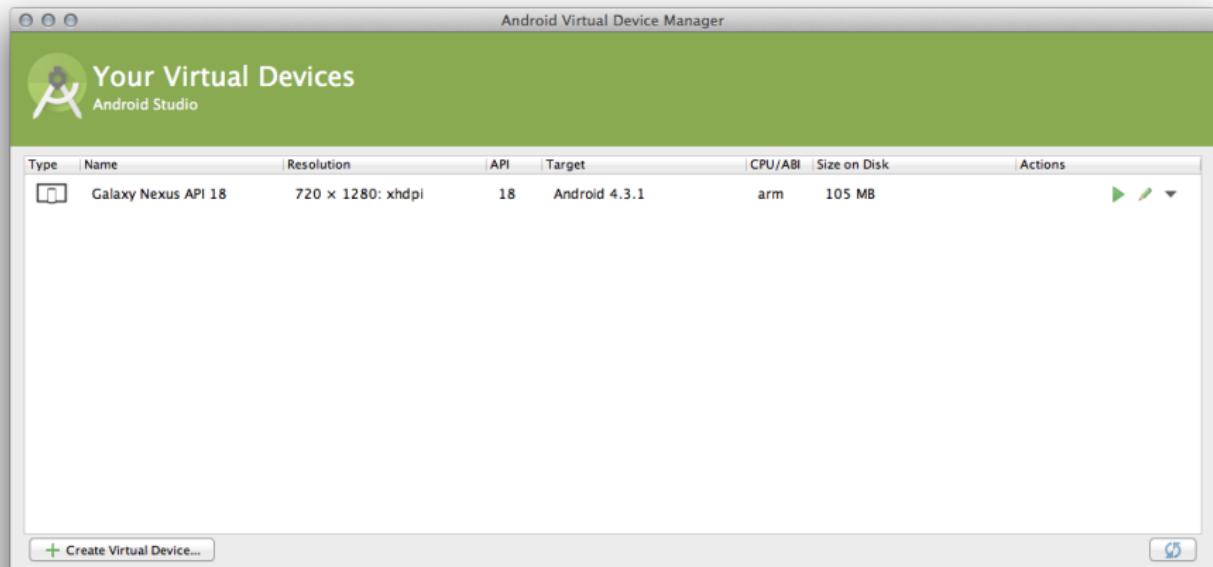


# Création d'un AVD

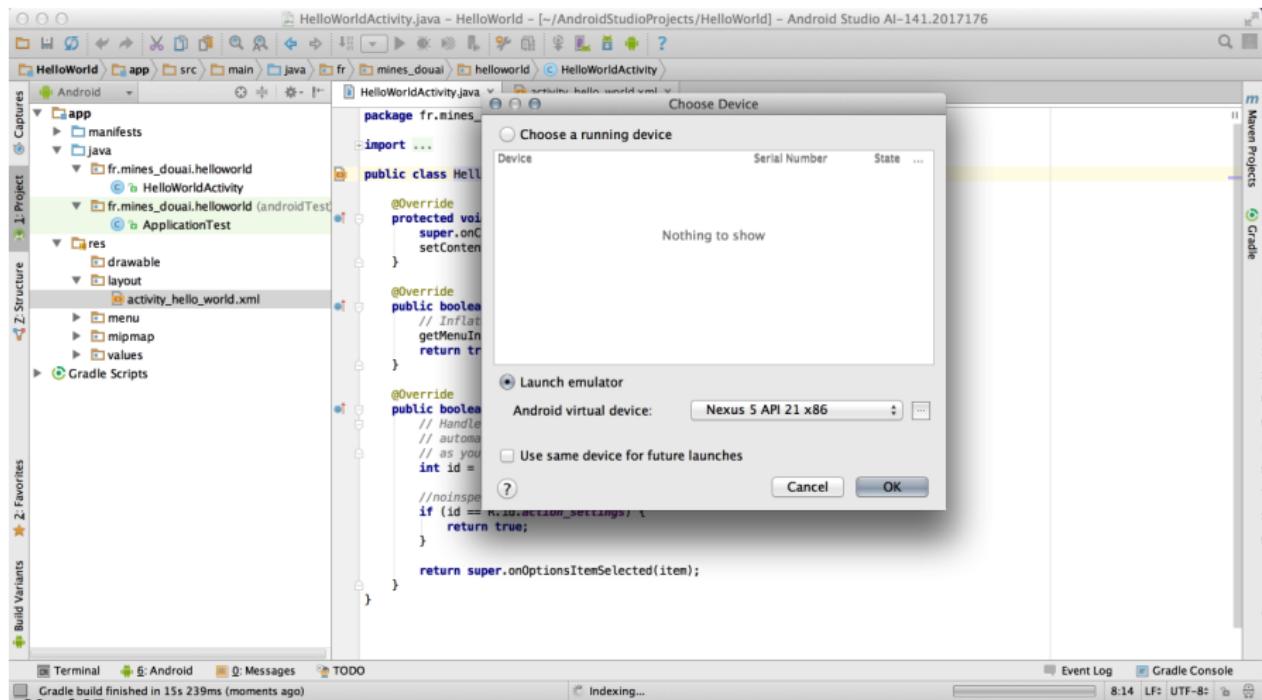


# Création d'un AVD

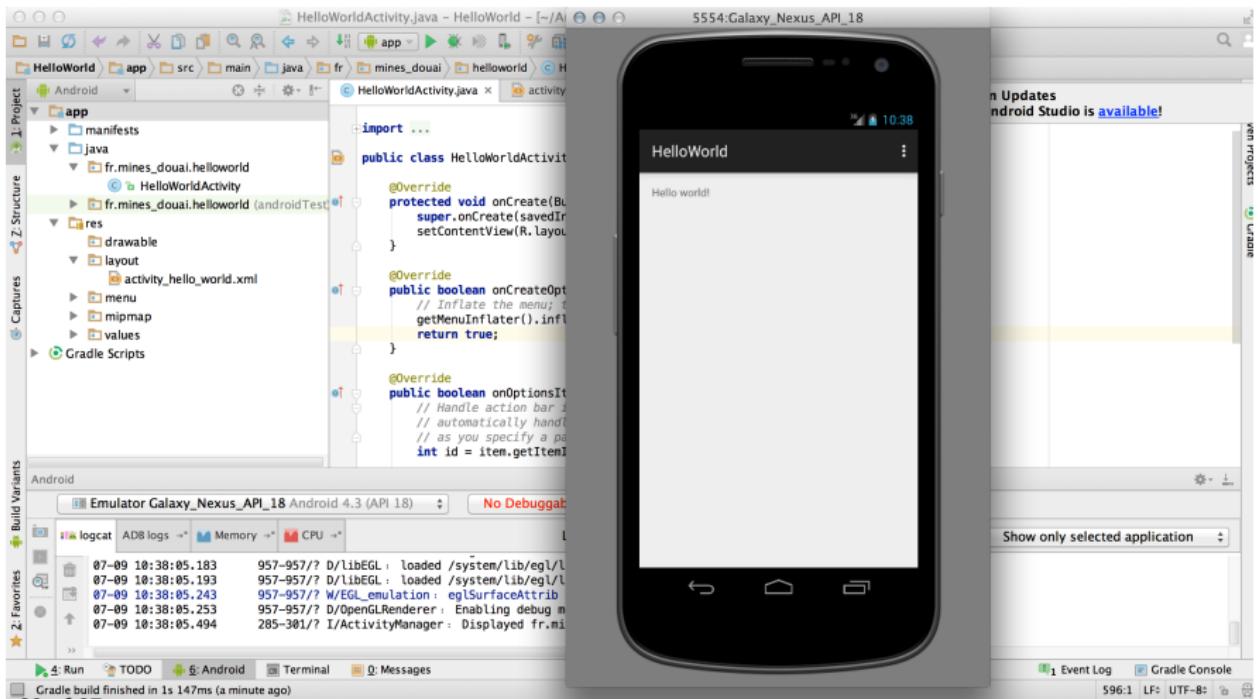
---



# Hello world: Exécution



# Hello world: Exécution



# Plan

---

## Introduction

Caractéristiques de l'OS

"L'écosystème" Android

Installation et configuration du SDK

Documentation du SDK

## Premiers pas avec le SDK

Première application sous Android

Utilisation de l'émulateur

## Les activités

## Conception d'une interface graphique

Principaux contrôles et boîtes de dialogues

Gestion des évènements

Animation graphique

## Fonctionnalités avancées

Utilisation des ressources

Multi-activités

Utilisation des capteurs du device

## Déploiement d'une application

## Les contraintes liées au développement sur mobile

## Pour aller plus loin...



# Hello world: Examinons le code généré

```
package fr.mines.douai.helloworld;

import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;

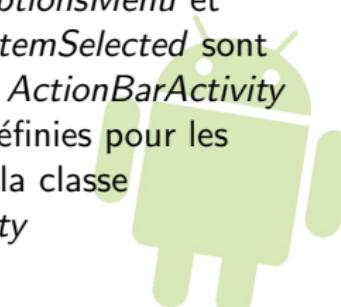
public class MainActivity extends ActionBarActivity {

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

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate (R.menu.menu_main, menu);
        return true;
    }

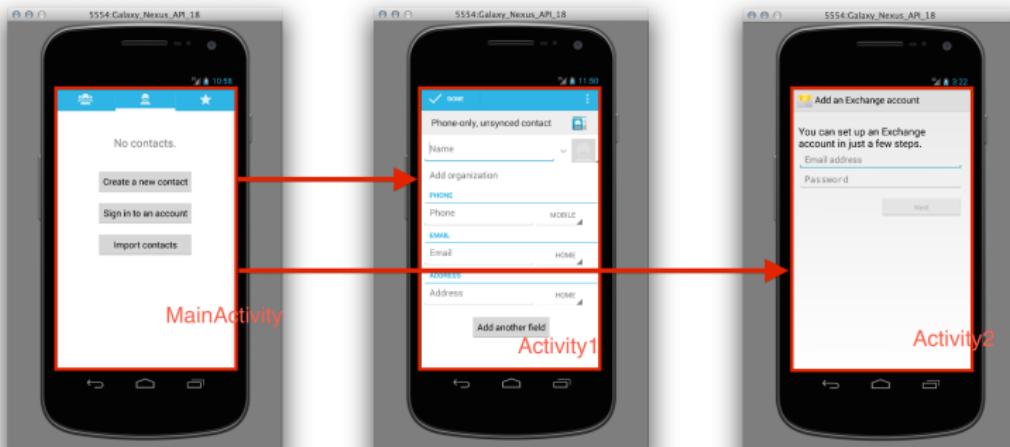
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        int id = item.getItemId();
        if (id == R.id.action_settings) {
            return true;
        }
        return super.onOptionsItemSelected(item);
    }
}
```

- la classe *MainActivity* hérite de la classe *ActionBarActivity* elle-même sous-classe de *Activity*
- les méthodes *onCreate*, *onCreateOptionsMenu* et *onOptionsItemSelected* sont héritées de *ActionBarActivity* et sont redéfinies pour les besoins de la classe *MainActivity*



# Les activités

- La majorité des applications Android utilisent des *Activity*.
- Il s'agit des différents "écrans" de l'application permettant à l'utilisateur d'interagir avec l'application. Une activité comportera des composants graphiques tels que des boutons, zones de texte, zone de dessin, etc.
- Une application possèdera une activité principale à partir de laquelle d'autres activités pourront être chargées:



# Cycle de vie d'un activité

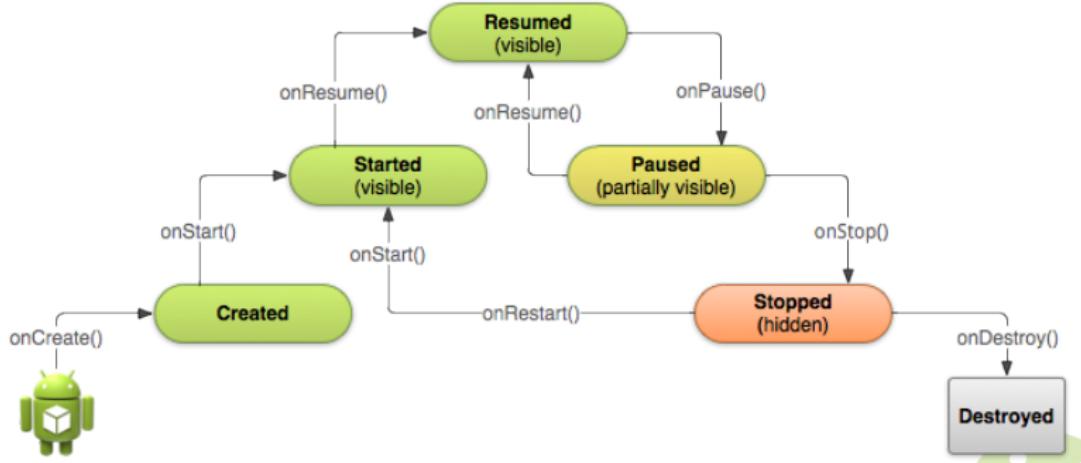


Illustration:

Lorsqu'un smartphone est réorienté par l'utilisateur, l'activité en cours est détruite et recréée (on repasse une deuxième fois dans la méthode `onCreate()`)

# Autres composants d'une application Android

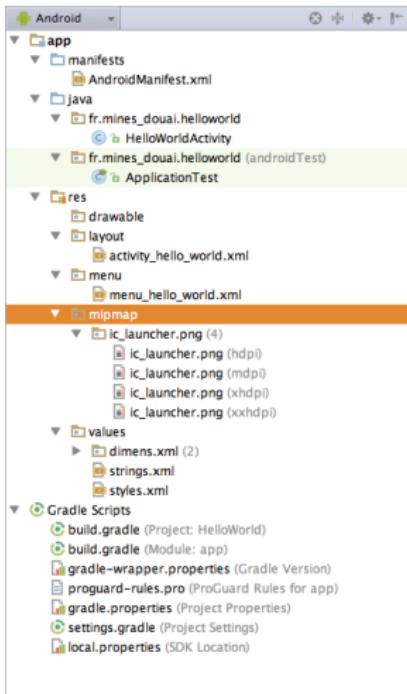
---

D'autres composants peuvent cohabiter aux côtés des activités:

- **les Services:** représentent les composants de l'application qui s'exécutent sans interface utilisateur (UI)
- **les Intentions:** représentent un moyen de communication entre applications via l'envoi de messages.
- **les Récepteurs de diffusion:** sont utilisés conjointement avec les intentions pour récupérer les messages envoyés
- **les Widgets:** composants visuels permettant de récupérer des messages envoyés par les intentions
- **les Notifications:** permettent l'envoi de signaux aux utilisateurs (arrivée d'un mail, mise à jour d'une appli, etc)



# HelloWorld: explorons l'arborescence du projet



Dans l'arborescence d'un projet on retrouve:

- des fichiers Java décrivant le fonctionnement de l'application
- des fichiers xml décrivant les différentes parties de l'interface graphique
- des fichiers image utilisés pour l'interface graphique (icône, fond d'écran, etc)
- des scripts Gradle (outil automatisant la compilation)
- le manifeste de l'application



# Le manifeste d'une application

---

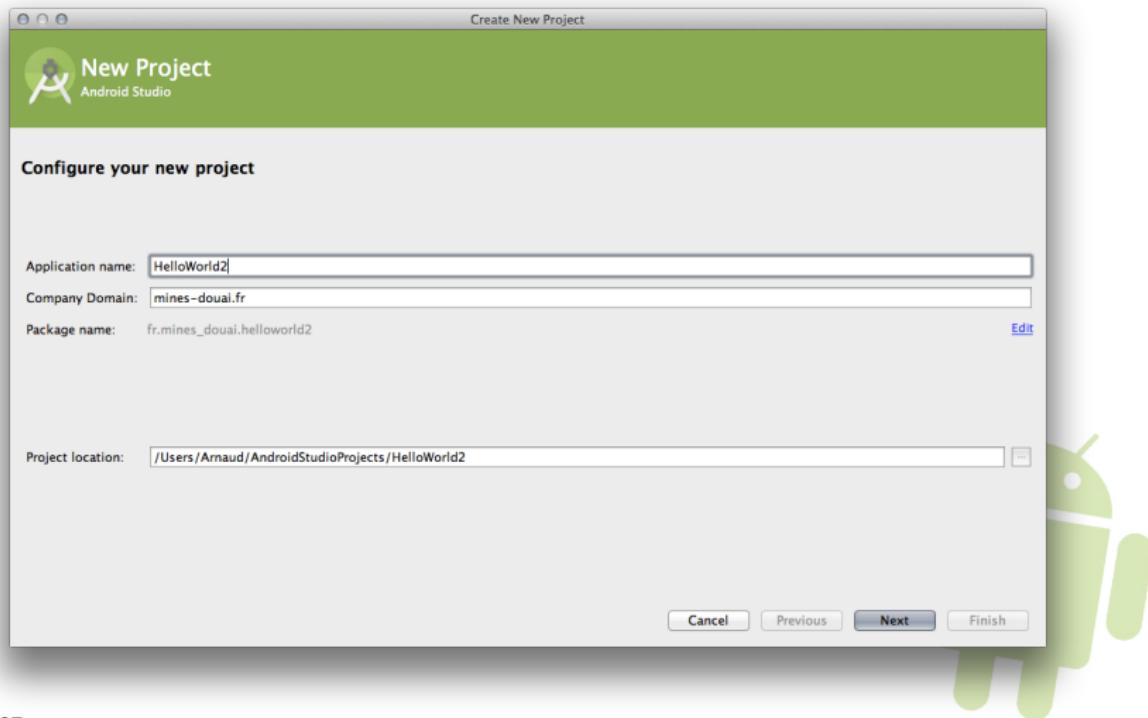
Le manifeste se présente sous la forme d'un fichier XML

*AndroidManifest.xml* incluant:

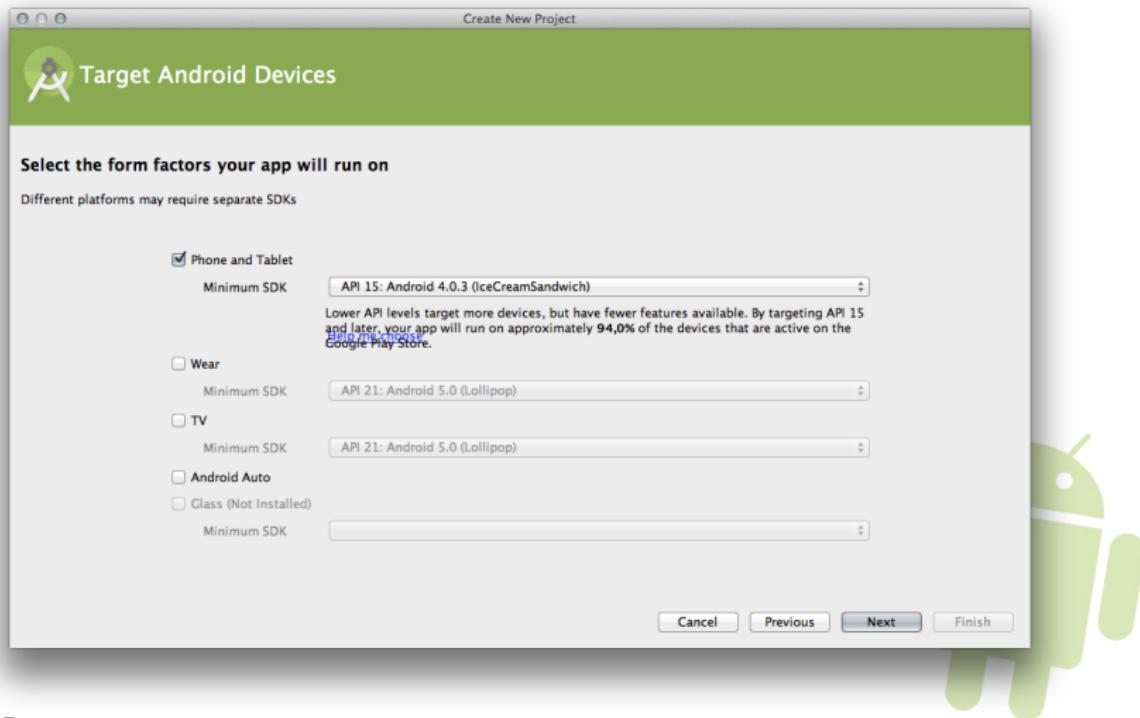
- des noeuds pour chaque composants (activités, services, etc)
- des attributs spécifiant les métadonnées d'une appli (numéro de version, thème, icône)
- l'endroit où l'application sera installée (ex: sur la carte SD ou sur la mémoire interne du téléphone)
- les besoins matériels pour faire tourner l'application (ex: NFC, microphone, bluetooth)
- les permissions utilisateurs (ex: accès à la partie téléphonie, utilisation de la connexion data, possibilité d'envoi de SMS)
- ...



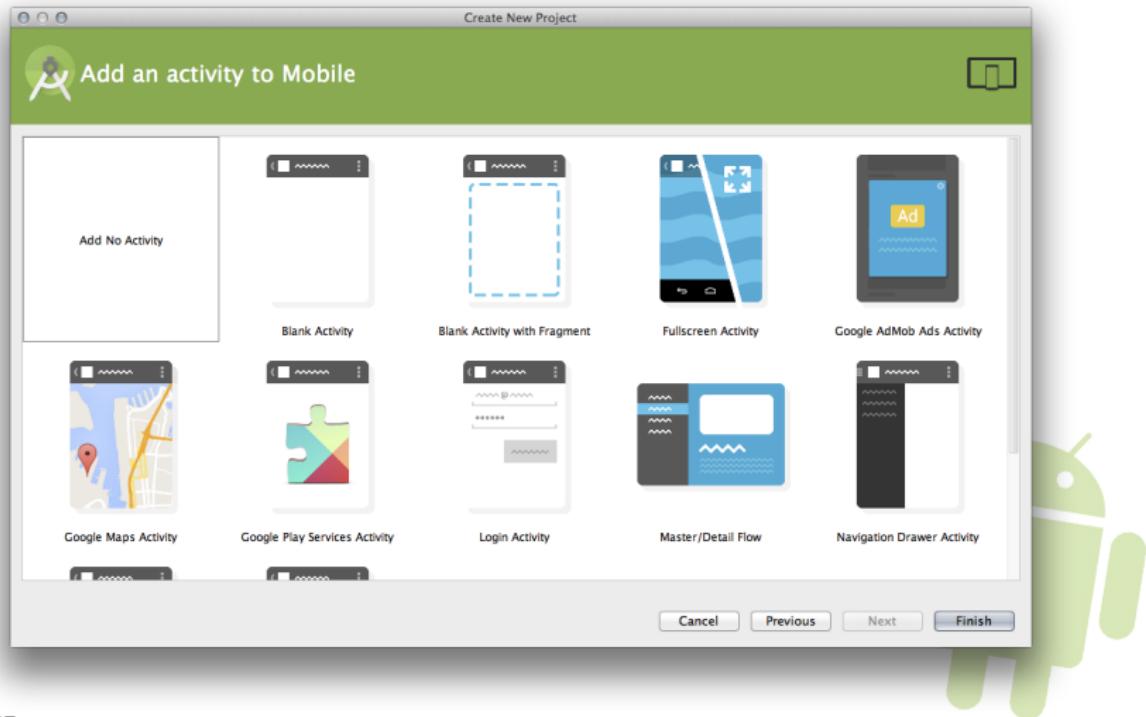
# Hello World 2: créer sa propre activité



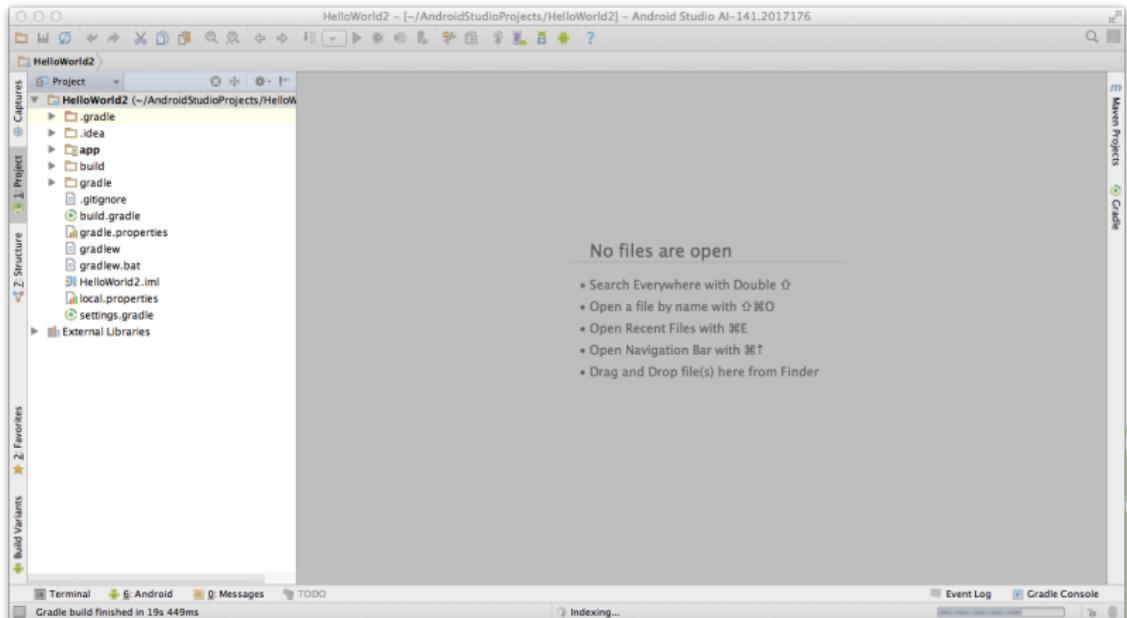
# Hello World 2: créer sa propre activité



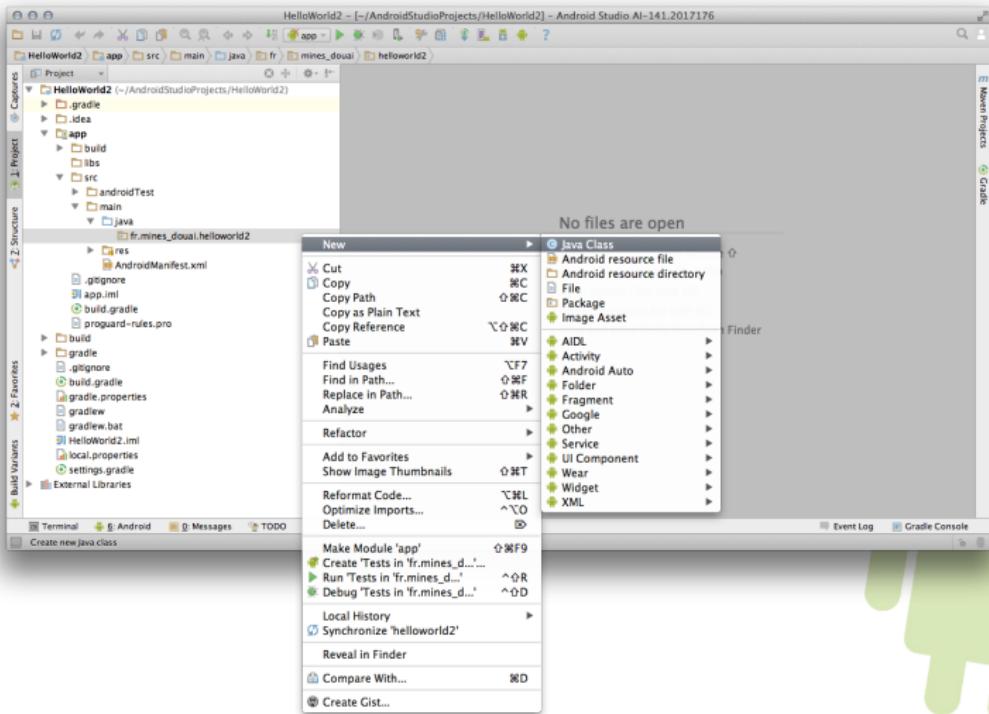
# Hello World 2: créer sa propre activité



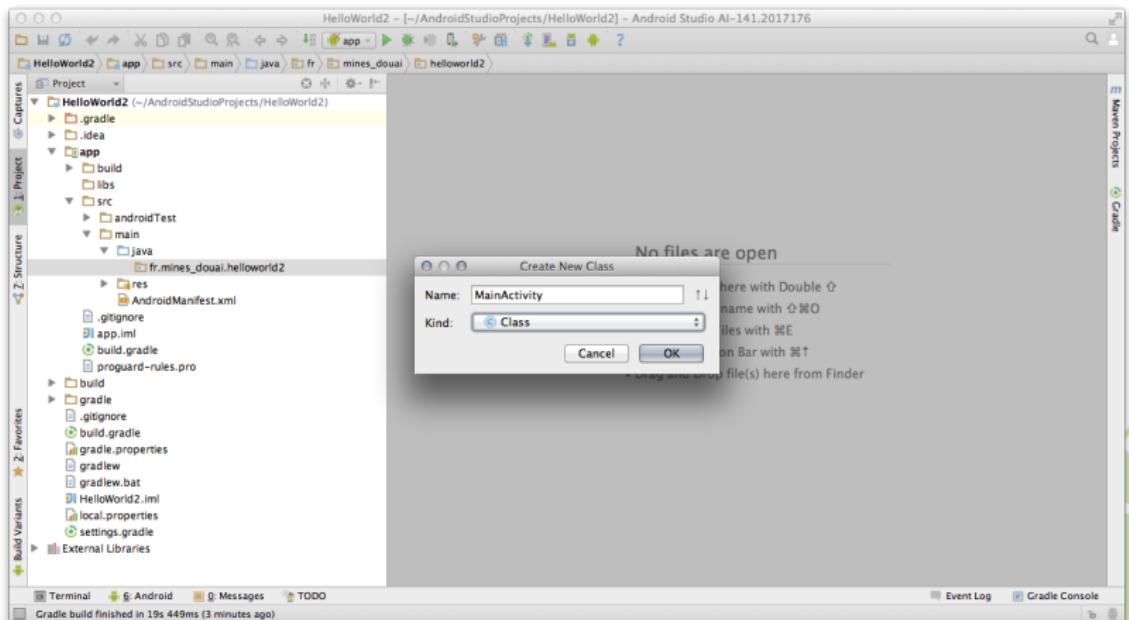
# Hello World 2: créer sa propre activité



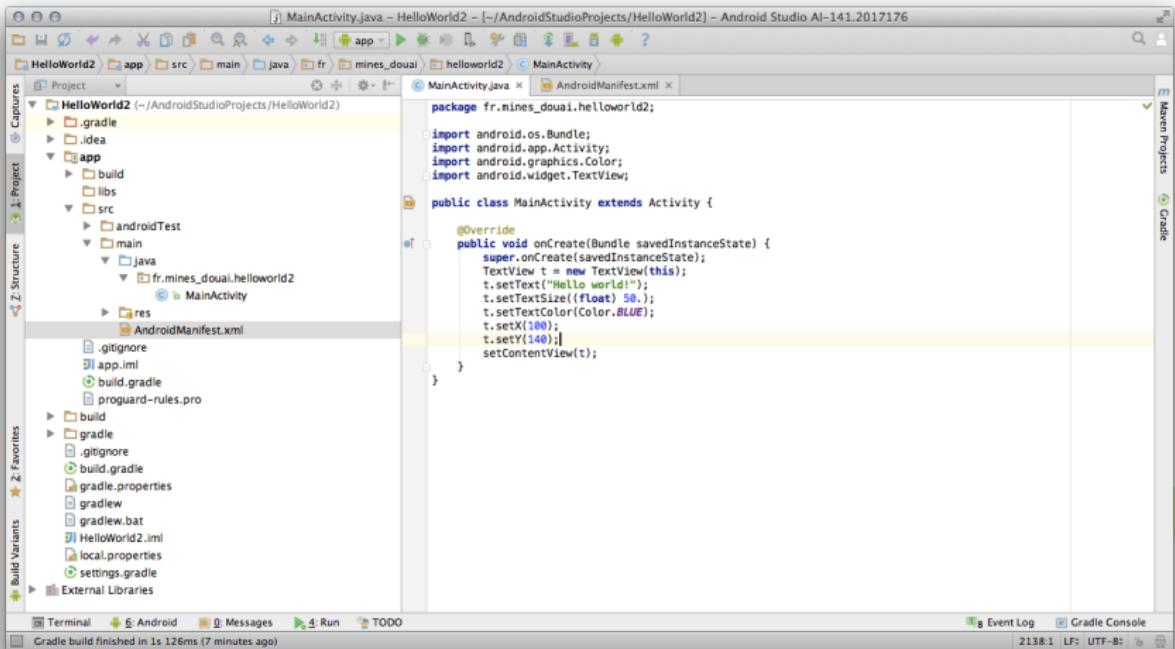
# Hello World 2: créer sa propre activité



# Hello World 2: créer sa propre activité



# Hello World 2: créer sa propre activité



The screenshot shows the Android Studio interface with the project 'HelloWorld2' open. The left sidebar displays the project structure, including the app module with its build, libs, src, and res directories, along with files like .gitignore, app.iml, build.gradle, and proguard-rules.pro. The main editor window shows the Java code for MainActivity.java:

```
package fr.mines_douai.helloworld2;

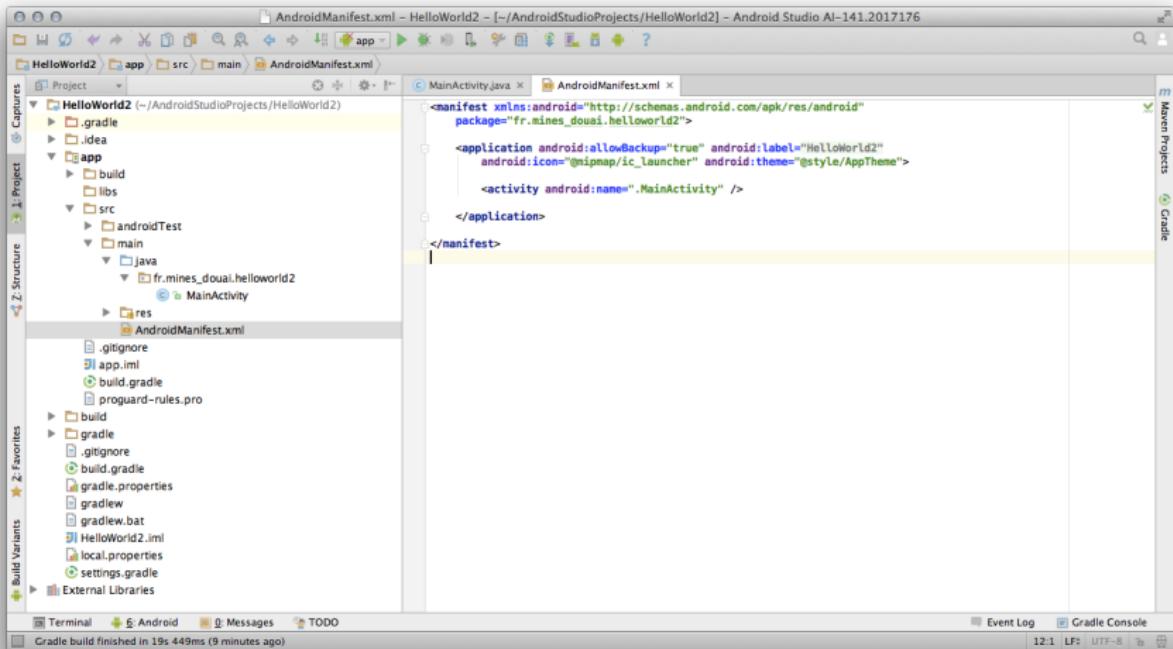
import android.os.Bundle;
import android.app.Activity;
import android.graphics.Color;
import android.widget.TextView;

public class MainActivity extends Activity {

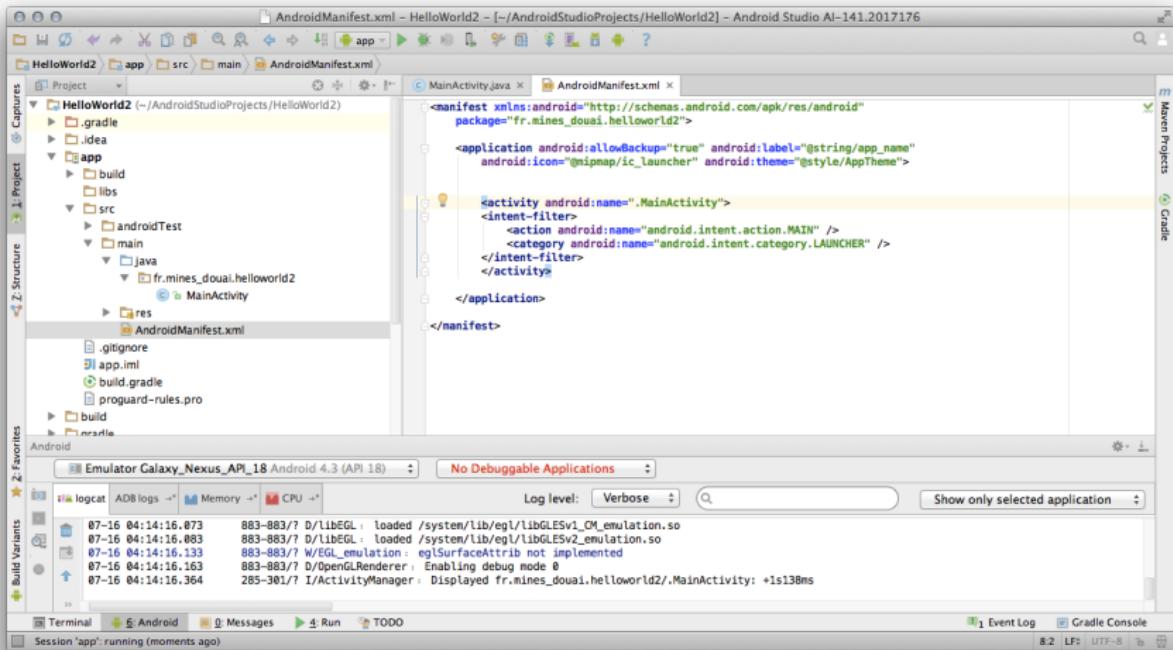
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        TextView t = new TextView(this);
        t.setText("Hello world!");
        t.setTextSize((float) 50.);
        t.setTextColor(Color.BLUE);
        t.setX(100);
        t.setY(140);
        setContentView(t);
    }
}
```

The code defines a MainActivity that extends Activity. It overrides the onCreate method to create a TextView, set its text to "Hello world!", and adjust its size and color. Finally, it sets the view to the current activity.

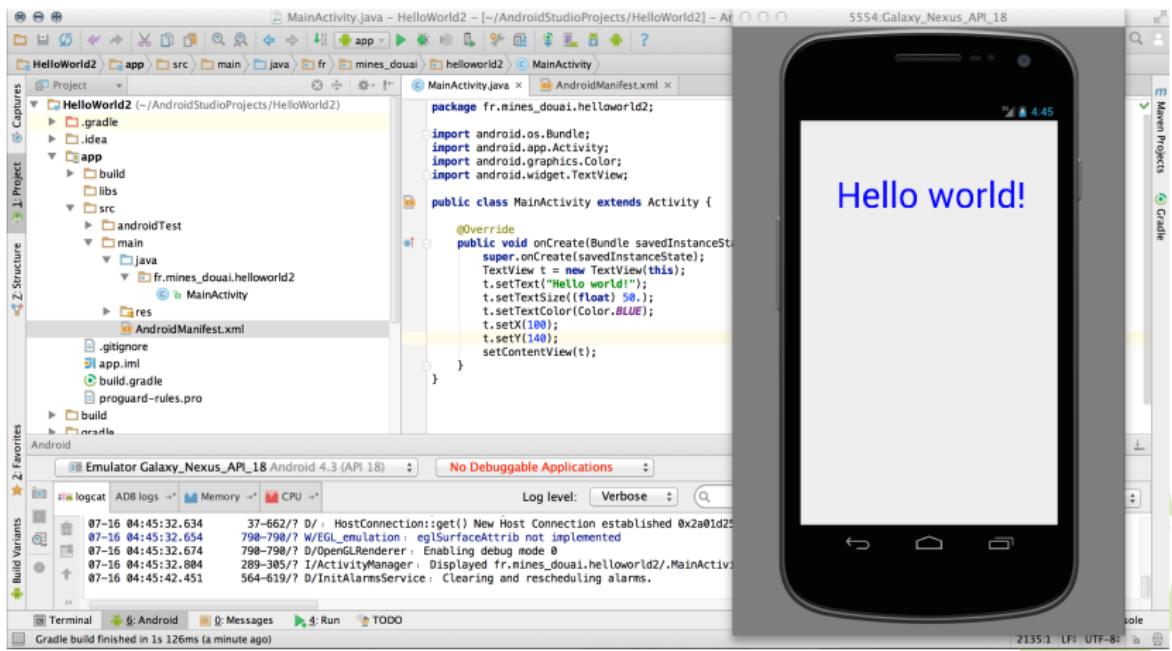
# Hello World 2: créer sa propre activité



# Hello World 2: créer sa propre activité



# Hello World 2: créer sa propre activité



## Création d'une activité

---

Pour créer sa propre activité, il y a deux étapes:

1. On crée une classe Java héritant de la classe *Activity*. Le contenu de cette activité peut être défini dans la classe *onCreate*.
2. On déclare cette nouvelle activité dans le fichier *AndroidManifest.xml* en indiquant comment elle sera utilisée (appelée au démarrage, par une autre activité, etc).



# Création d'une activité - partie Java

---

```
import android.os.Bundle;
import android.app.Activity;
import android.graphics.Color;
import android.widget.TextView;

public class MainActivity extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        TextView t = new TextView(this);
        t.setText("Hello world!");
        t.setTextSize((float) 50.);
        t.setTextColor(Color.BLUE);
        t.setX(100);
        t.setY(140);
        setContentView(t);
    }
}
```

- la classe *MainActivity* hérite directement de la classe *Activity*
- la méthode *onCreate*:
  - est héritée d'*Activity*
  - est complétée pour les besoins de *MainActivity* (mais sa signature ne change pas)
  - n'est pas redéfinie puisque l'on débute en appelant la méthode *onCreate* de la super-classe
  - prend en entrée un objet *Bundle* représentant l'état de l'activité si celle-ci a déjà été créé (*savedInstanceState* vaut *null* la première fois)
  - doit toujours contenir l'appel à *setContentView* qui permet d'associer à l'activité une interface graphique (*View*)
  - ici l'interface graphique se résume à une simple zone de texte (*Textview t*)

# Création d'une activité - partie Xml

---

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="fr.mines_douai.helloworld2" >

    <application android:allowBackup="true" android:label="@string/app_name"
        android:icon="@mipmap/ic_launcher" android:theme="@style/AppTheme" >

        <activity android:name=".MainActivity" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

    </application>

</manifest>
```

- toute activité doit se déclarer à l'intérieur des balises `<application>`  
`</application>`
- une balise `<activity>` doit contenir au minimum l'attribut `android : name` qui fait référence à la classe java de l'activité
- la balise `<intent – filter>` permet d'indiquer quelle action permet de lancer l'activité (ici `android.intent.action.MAIN` indique l'activité est le point d'entrée du programme, l'activité principale)



# Plan

---

## Introduction

Caractéristiques de l'OS

"L'écosystème" Android

Installation et configuration du SDK

Documentation du SDK

## Premiers pas avec le SDK

Première application sous Android

Utilisation de l'émulateur

Les activités

## Conception d'une interface graphique

Principaux contrôles et boîtes de dialogues

Gestion des événements

Animation graphique

## Fonctionnalités avancées

Utilisation des ressources

Multi-activités

Utilisation des capteurs du device

## Déploiement d'une application

## Les contraintes liées au développement sur mobile

## Pour aller plus loin...



# Plan

---

## Introduction

- Caractéristiques de l'OS
- "L'écosystème" Android
- Installation et configuration du SDK
- Documentation du SDK

## Premiers pas avec le SDK

- Première application sous Android
- Utilisation de l'émulateur
- Les activités

## Conception d'une interface graphique

- Principaux contrôles et boîtes de dialogues**
- Gestion des évènements
- Animation graphique

## Fonctionnalités avancées

- Utilisation des ressources
- Multi-activités
- Utilisation des capteurs du device

## Déploiement d'une application

## Les contraintes liées au développement sur mobile

## Pour aller plus loin...



# Notion de View

---

Sous Android, tous les éléments graphiques utilisés pour créer une interface hérite de la class `android.view.View`.

Il est possible:

- de créer son propre élément graphique en créant une nouvelle class qui étend la class `View`
- d'utiliser les nombreux éléments graphiques (zone de texte, contrôleur) fournis par l'API



## Zones de texte

---

Il existe deux types de zone de texte:

- `TextView`: non éditable par l'utilisateur (cf exemple du `HelloWorld`)
- `EditText`: modifiable en cours d'exécution par l'utilisateur



# Exemple de déclaration d'une zone de texte

---

```
package minesdouai.helloworld;

import android.os.Bundle;
import android.app.Activity ;
import android.graphics . Color;
import android.widget.EditText;

public class MainActivity extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EditText t = new EditText(this);
        t.setText("Vous pouvez me modifier!");
        setContentView(t);
    }
}
```



# Les layouts: pourquoi les utiliser ?

---

Que se passe-t-il si on exécute le code ci-dessous ?

```
package minesdouai.helloworld;

import android.os.Bundle;
import android.app.Activity;
import android.graphics.Color;
import android.widget.TextView;

public class MainActivity extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        TextView t1 = new TextView(this);
        t1.setText("Hello world!");
        TextView t2 = new TextView(this);
        t2.setText("Bonjour!");
        setContentView(t1);
        setContentView(t2);
    }
}
```

Comment ajouter les deux zones de texte à notre activité ?

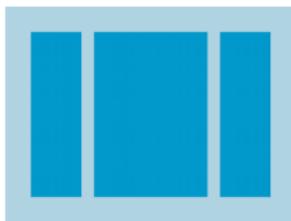


# Les layouts

---

Les layouts héritent de la class `android.view.ViewGroup`. Ils permettent de regrouper différents éléments graphiques ensemble en les positionnant les uns par rapport aux autres.

LinearLayout



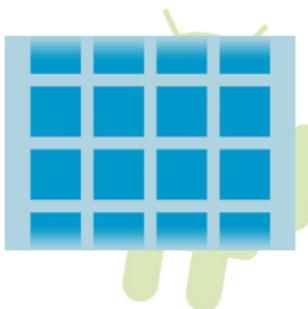
RelativeLayout



ListView



GridView



# Exemple d'utilisation des layouts

---

```
package minesdouai.helloworld;

import android.os.Bundle;
import android.app.Activity ;
import android.graphics.Color;
import android.widget.TextView;

public class MainActivity extends Activity {

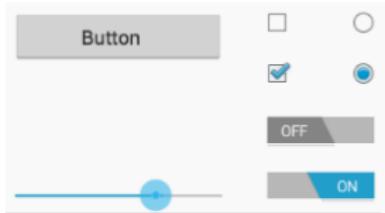
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        LinearLayout l1 = new LinearLayout(this);
        l1.setOrientation (LinearLayout.VERTICAL);
        TextView t1 = new TextView(this);
        t1.setText("Hello world!");
        TextView t2 = new TextView(this);
        t2.setText("Bonjour!");
        l1.addView(t1);
        l1.addView(t2);
        setContentView(l1);
    }
}
```



# Les boutons et sélecteurs

---

- Button: bouton classique
- Checkbox: coche de sélection
- ToggleButton: bouton on/off
- Spinners: liste déroulante



# Exemple de déclaration d'un bouton

---

```
import android.os.Bundle;
import android.app.Activity ;
import android.view.Menu;
import android.widget.Button;
import android.widget.EditText;
import android.widget.LinearLayout;

public class MainActivity extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        LinearLayout l1 = new LinearLayout(this);
        l1.setOrientation(LinearLayout.VERTICAL);
        EditText t = new EditText(this);
        Button b = new Button(this);
        b.setText("Dis bonjour");
        b.setHeight(200);
        l1.addView(t);
        l1.addView(b);
        setContentView(l1);
    }
}
```



# Plan

---

## Introduction

- Caractéristiques de l'OS
- "L'écosystème" Android
- Installation et configuration du SDK
- Documentation du SDK

## Premiers pas avec le SDK

- Première application sous Android
- Utilisation de l'émulateur
- Les activités

## Conception d'une interface graphique

- Principaux contrôles et boîtes de dialogues

## Gestion des événements

- Animation graphique

## Fonctionnalités avancées

- Utilisation des ressources
- Multi-activités
- Utilisation des capteurs du device

## Déploiement d'une application

## Les contraintes liées au développement sur mobile

## Pour aller plus loin...



# Les différents gestes sous Android

---



Touch



Double touch



Long touch



Drag



Swipe



Pinch open



## Ecoute des évènements des utilisateurs

---

Chaque action de l'utilisateur sur l'écran affichant une activité génère un évènement qu'il est possible de capturer dans le code.

Différents écouteurs définis sous Android:

- `View.OnClickListener` pour écouter le 'clic' sur un composant
- `View.OnLongClickListener` pour écouter le 'clic' long sur un composant
- `View.OnFocusChangeListener` pour écouter le changement de focus entre plusieurs composants (ex: différentes zones de texte remplies successivement)
- ...



# Exemple d'utilisation d'un bouton

---

```
import android.os.Bundle;
import android.app.Activity ;
import android.view.Menu;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.LinearLayout;

public class MainActivity extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        LinearLayout l1 = new LinearLayout(this);
        l1.setOrientation (LinearLayout.VERTICAL);
        final EditText t = new EditText(this);
        Button b = new Button(this);
        b.setText("Dis bonjour");
        b.setHeight(200);
        b.setOnClickListener (new View.OnClickListener() {
            public void onClick(View v) {
                t.setText("bonjour");
            }
        });
        l1.addView(t);
        l1.addView(b);
        setContentView(l1);
    }
}
```



# Les boîtes de dialogues

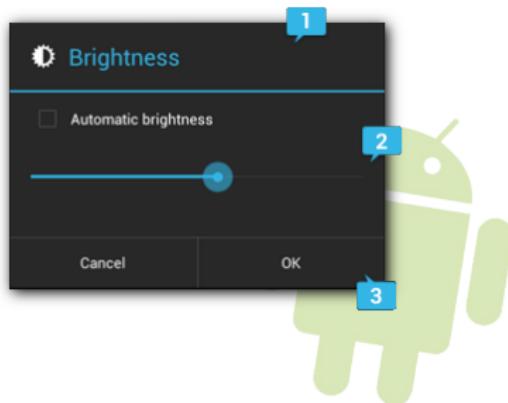
---

Des boîtes de dialogues prêtes à l'emploi sont fournies par le SDK.  
Toutes les boîtes héritent de la class `android.app.Dialog`.

Exemple: `AlertDialog`

3 régions:

- Titre
- Zone de contenu
- Boutons d'action



# Exemple de boîte de dialogue

---

```
...
Boolean rejoue = false;
...
builder = new AlertDialog.Builder(context);
builder .setMessage("Voux-tu encore jouer ?");
builder .setCancelable(false );
builder .setPositiveButton ("Oui" , new DialogInterface.OnClickListener() {
    public void onClick(DialogInterface dialog , int id) {
        rejoue = true;
    }
});
builder .setNegativeButton("Non" , new DialogInterface.OnClickListener() {
    public void onClick(DialogInterface dialog , int id) {
        rejoue=false;
    }
});
...
AlertDialog boite = buider.create();
...
boite .show();
...
if (rejoue){
...
}
```



# Plan

---

## Introduction

Caractéristiques de l'OS

"L'écosystème" Android

Installation et configuration du SDK

Documentation du SDK

## Premiers pas avec le SDK

Première application sous Android

Utilisation de l'émulateur

Les activités

## Conception d'une interface graphique

Principaux contrôles et boîtes de dialogues

Gestion des évènements

## Animation graphique

## Fonctionnalités avancées

Utilisation des ressources

Multi-activités

Utilisation des capteurs du device

## Déploiement d'une application

## Les contraintes liées au développement sur mobile

## Pour aller plus loin...



# Le dessin sous Android

---

- Pour dessiner sous Android, il faut manipuler deux objets: `Paint` (le pinceau) et `Canvas` (la toile).
- La "zone de dessin" est créée par héritage de la class `View`
- La class `Canvas` possède différentes méthodes de dessin:
  - `drawCircle`
  - `drawLine`
  - `drawText`
  - `drawBitmap`
  - ...
- Les options de "tracé" sont accessibles via l'objet `Paint`:
  - `setColor`
  - `setAntiAlias` ...



# Le dessin sous Android: exemple

---

```
...  
  
class ZoneDeDessin extends View {  
  
    Paint paint ;  
  
    public ZoneDeDessin(Context context) {  
        super(context);  
        paint = new Paint();  
        paint.setAntiAlias (true);  
    }  
  
    protected void onDraw(Canvas canvas) {  
        canvas.drawCircle(50,50,10, paint);  
    }  
}  
  
public class MainActivity extends Activity {  
  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(new ZoneDeDessin(this));  
    }  
}  
...  
}
```



## Animation graphique simple

---

- Une manière simple de faire de l'animation graphique (pour un jeu par ex) est de forcer une zone de dessin à se redessiner en permanence (ou du moins lorsqu'un élément de dessin change).
- Pour forcer la View à se redessiner, on peut appeler la méthode `invalidate()`.
- Si "l'animation" est trop rapide, on peut la ralentir en forçant le processus de l'application à "s'endormir" quelques micro-secondes: méthode `Thread.sleep`



# Animation graphique simple: exemple

---

```
class ZoneAnimation extends View {  
  
    Paint paint;  
    int balleX, balleY;  
  
    public ZoneAnimation(Context context) {  
        super(context);  
        paint = new Paint();  
        paint.setAntiAlias (true);  
        balleX = 50;  
        balleY = 50;  
    }  
  
    protected void onDraw(Canvas canvas) {  
  
        Thread.sleep(10); // à entourer d'un try... catch: proposition automatique par eclipse  
        canvas.drawCircle(balleX, balleY,10, paint);  
  
        balleX++;  
        balleY++;  
  
        invalidate ();  
    }  
}
```



## Animation graphique plus complexe

---

L'API Android offre d'autres mécanismes plus riches pour réaliser des animations graphiques:

- Le package Animation qui permet de décomposer une animation dans un fichier XML en utilisant les primitives (balises): <alpha>, <scale>, <translate>, <rotate>  
(<http://developer.android.com/guide/topics/graphics/view-animation.html>)
- Open GL ES qui permet de gérer les animations 3D  
(<http://developer.android.com/guide/topics/graphics/opengl.html>)



# Plan

---

## Introduction

- Caractéristiques de l'OS
- "L'écosystème" Android
- Installation et configuration du SDK
- Documentation du SDK

## Premiers pas avec le SDK

- Première application sous Android
- Utilisation de l'émulateur
- Les activités

## Conception d'une interface graphique

- Principaux contrôles et boîtes de dialogues
- Gestion des évènements
- Animation graphique

## Fonctionnalités avancées

- Utilisation des ressources
- Multi-activités
- Utilisation des capteurs du device

## Déploiement d'une application

## Les contraintes liées au développement sur mobile

## Pour aller plus loin...



# Plan

---

## Introduction

- Caractéristiques de l'OS
- "L'écosystème" Android
- Installation et configuration du SDK
- Documentation du SDK

## Premiers pas avec le SDK

- Première application sous Android
- Utilisation de l'émulateur
- Les activités

## Conception d'une interface graphique

- Principaux contrôles et boîtes de dialogues
- Gestion des évènements
- Animation graphique

## Fonctionnalités avancées

- Utilisation des ressources**
- Multi-activités
- Utilisation des capteurs du device

## Déploiement d'une application

## Les contraintes liées au développement sur mobile

## Pour aller plus loin...



# Qu'est-ce qu'une ressource ?

---

- il peut être intéressant d'externaliser certaines ressources d'une application comme:
  - les images (utile pour faire évoluer le design graphique d'une appli sans toucher au code)
  - les Strings (utile pour traduire rapidement une application sans faire trop de modifs dans le code)
  - les sons, les vidéos
- etc
- l'API Android offre un système de nommage permettant de référencer dans le code des ressources externes et de les instancier sous forme d'objets



# Ressources dans l'arborescence d'un projet

---

Dans un projet:

- les sources (.java) sont localisées dans le dossier `src/`
- les ressources apparaissent dans le dossier `res/`

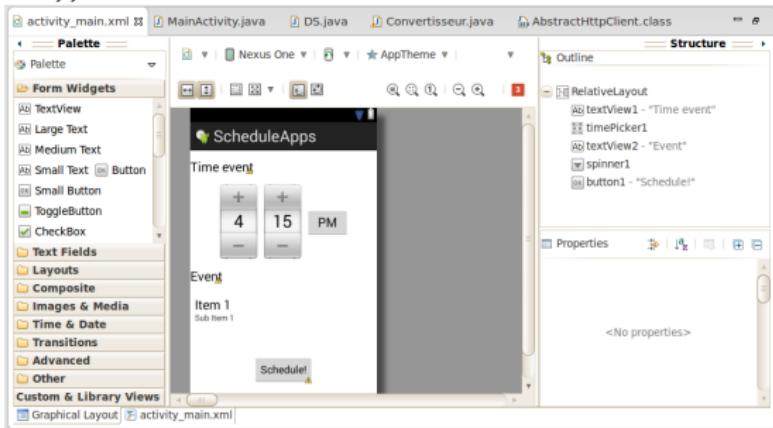
Chaque type de ressources est associé à un sous-répertoire de `res/`:

- `drawable/` contient les ressources de type image (`.png`, `.jpg`, `.gif`, ...)
- `layout/` contient les descriptions xml de l'interface graphique d'une application
- `raw/` contient tout fichier de données brut autre qu'une image, un fichier xml (ex: sons, vidéos)
- `values/` contient des valeurs de type entière, chaîne de caractères



# Ajout d'une ressource

L'ajout d'une ressource peut se faire manuellement (exemple: l'ajout d'une image dans le répertoire `drawable`) ou via ADT (exemple: ajout d'un composant graphique dans l'outil de composition d'interface (onglet `Graphical Layout`)):



## Utilisation d'une ressource

---

Pour permettre au développeur d'accéder à une ressource d'un point de vue programmatique, ADT met à jour automatiquement une class R dans le répertoire gen/. Des sous-class y sont générées automatiquement avec des attributs portant le même nom que les fichiers apparaissant dans res/.

Exemple: accès à un fichier PNG situé dans res/drawable/myimage.png

```
imageView imageView = (ImageView) findViewById(R.id.myimageview);  
imageView.setImageResource(R.drawable.myimage);
```



# Plan

---

## Introduction

- Caractéristiques de l'OS
- "L'écosystème" Android
- Installation et configuration du SDK
- Documentation du SDK

## Premiers pas avec le SDK

- Première application sous Android
- Utilisation de l'émulateur
- Les activités

## Conception d'une interface graphique

- Principaux contrôles et boîtes de dialogues
- Gestion des évènements
- Animation graphique

## Fonctionnalités avancées

- Utilisation des ressources

### Multi-activités

- Utilisation des capteurs du device

## Déploiement d'une application

## Les contraintes liées au développement sur mobile

## Pour aller plus loin...



## Les Intents

---

Il est possible de lancer d'autres activités depuis l'activité principale en utilisant les Intents

- un Intent est objet permettant “la communication” (échange d'info et déclenchement d'actions) entre activités (pas forcément au sein d'une même application)
- il existe deux types d'Intents:
  - les explicites: lorsque le destinataire de l'Intent est renseigné (ex: la class d'une activité)
  - les implicites: lorsque l'Intent contient des données et une action à faire sur ces données sans forcément que l'on sache quelle activité / application / service du système traitera la demande
- les Intents peuvent véhiculer des données entre deux activités en utilisant les méthodes `putExtra` et `get<type>Extra`



# Exemple: code de l'activité principale

---

```
public class MainActivity extends Activity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
  
        setContentView(R.layout.activity_main );  
        Button b = (Button) findViewById(R.id.button);  
        b.setText("Envoyer vers nouvelle activité ");  
        final EditText e = (EditText) findViewById(R.id.editText);  
  
        b.setOnClickListener (new View.OnClickListener() {  
            @Override  
            public void onClick( View view) {  
                Intent lancementSecondeActivite = new Intent(MainActivity.this , SecondActivity.class );  
                lancementSecondeActivite.putExtra("message" , e.getText().toString());  
                startActivity (lancementSecondeActivite);  
            }  
        });  
    }  
}
```



## Exemple: code de l'activité secondaire

---

```
public class SecondActivity extends Activity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_second );  
        TextView e = (TextView) findViewById(R.id.textView);  
        Intent i = getIntent();  
        String m = i.getStringExtra("message");  
        e.setText(m+ "");  
    }  
}
```



# Plan

---

## Introduction

- Caractéristiques de l'OS
- "L'écosystème" Android
- Installation et configuration du SDK
- Documentation du SDK

## Premiers pas avec le SDK

- Première application sous Android
- Utilisation de l'émulateur
- Les activités

## Conception d'une interface graphique

- Principaux contrôles et boîtes de dialogues
- Gestion des évènements
- Animation graphique

## Fonctionnalités avancées

- Utilisation des ressources
- Multi-activités
- Utilisation des capteurs du device**

## Déploiement d'une application

## Les contraintes liées au développement sur mobile

## Pour aller plus loin...



# Les différents types de capteurs

---

- Les smartphones et tablettes sont de plus en plus équipés de capteurs sophistiqués dont les informations recueillies permettent d'enrichir les applications
- On distingue:
  - **les capteurs de mouvements** qui mesurent les forces subies par le device en  $\vec{x}$ , en  $\vec{y}$  et en  $\vec{z}$ : accéléromètres, gyroscopes, magnétomètres, etc
  - **les capteurs d'environnements** qui peuvent mesurer des paramètres tels que la température, la pression de l'air, la luminosité, etc
  - **les capteurs de localisation** qui renvoie une position géographique (coordonnées GPS)



# Identifier les capteurs présents sur un device

---

Pour cela, il faut passer par un objet SensorManager:

```
package com.example.detectioncapteur;

import java.util.List;
import android.hardware.Sensor;
import android.hardware.SensorManager;
import android.os.Bundle;
import android.app.Activity;
import android.content.Context;
import android.util.Log;
import android.view.Menu;

public class MainActivity extends Activity {

    private SensorManager mSensorManager;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main );
        // on cree l'objet SensorManager
        mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
        List <Sensor> deviceSensors = mSensorManager.getSensorList(Sensor.TYPE_ALL);
        for (int i=0;i<deviceSensors.size (); i++) Log.d("Liste capteur", deviceSensors .get(i).getName());
    }

    ...
}
```



# Accès à un capteur

---

```
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;
...
public class MainActivity extends Activity implements SensorEventListener {
    private SensorManager mSensorManager;
    private Sensor accelerometer;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main );
        mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
        // on recupere l accelerometre a partir du SensorManager
        accelerometer = mSensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
        // on associe l ecouteur d evenements au SensorManager
        mSensorManager.registerListener(this , accelerometer , SensorManagerSENSOR_DELAY_NORMAL);
    }

    @Override
    public void onAccuracyChanged(Sensor sensor, int accuracy) { // TODO Auto-generated method stub
    }

    @Override
    public void onSensorChanged(SensorEvent event) { // TODO Auto-generated method stub
        float gammaX = event.values[0], gammaY = event.values[1], gammaZ = event.values[2];
        Log.d("Valeurs accelerometre",gammaX+","+gammaY+","+gammaZ);
    }
}
```



# Plan

---

## Introduction

- Caractéristiques de l'OS
- "L'écosystème" Android
- Installation et configuration du SDK
- Documentation du SDK

## Premiers pas avec le SDK

- Première application sous Android
- Utilisation de l'émulateur
- Les activités

## Conception d'une interface graphique

- Principaux contrôles et boîtes de dialogues
- Gestion des évènements
- Animation graphique

## Fonctionnalités avancées

- Utilisation des ressources
- Multi-activités
- Utilisation des capteurs du device

## Déploiement d'une application

Les contraintes liées au développement sur mobile

Pour aller plus loin...



# Les différents modes de déploiement d'une application

---

- diffusion du .apk par mail
- diffusion du .apk sur support physique (carte mémoire)
- diffusion du .apk via une page web "perso"
- diffusion du .apk sur un des nombreux "market" existant: Slide Me, Yaam, Google Play,



## Utilisation de Google Play pour la distribution d'une application

---

- Google Play est un service payant (25\$ de frais d'inscription en 2012) pour le développeur
- si l'application est destinée à être payante, le développeur doit également s'inscrire en tant que marchand Google Checkout



# Plan

---

## Introduction

- Caractéristiques de l'OS
- "L'écosystème" Android
- Installation et configuration du SDK
- Documentation du SDK

## Premiers pas avec le SDK

- Première application sous Android
- Utilisation de l'émulateur
- Les activités

## Conception d'une interface graphique

- Principaux contrôles et boîtes de dialogues
- Gestion des évènements
- Animation graphique

## Fonctionnalités avancées

- Utilisation des ressources
- Multi-activités
- Utilisation des capteurs du device

## Déploiement d'une application

## Les contraintes liées au développement sur mobile

Pour aller plus loin...



## Considérations matérielles

---

Comparativement à un PC, un terminal mobile dispose:

- puissance de calcul limitée
- mémoire limitée (quantité RAM et vitesse écriture/lecture)
- résolution d'écran 'inférieure'
- connexions lentes, moins fiables et latence élevée
- autonomie limitée



# Ergonomie: les problématiques

Le nombre de résolutions d'écran rencontrées sous Android est élevé (de la TV 42 pouces aux smartwatch à écran 1,5 pouces).



Problématiques:

- résolution / taille des ressources graphiques (images, icônes, ...)
- présentation de l'information en fonction de la taille de l'écran
- multiplication des activités / difficultés de navigations



## Ergonomie: les bonnes pratiques

---

Tiré de <http://developer.android.com/design/index.html>

- ne pas surcharger l'interface, utiliser plusieurs activités
- privilégier images / icônes à du texte long
- n'afficher que les informations nécessaires (et en temps utile)
- respecter les "conventions" de développement de la plateforme



## Applications on-line

---

De nombreuses applications nécessitent l'accès à des ressources en lignes (pub, images, service web, ...). Hélas, les connexions des terminaux mobiles peuvent être:

- intermittentes
- lentes
- coûteuses

À éviter:

- des requêtes réseaux répétées à l'intérieur du thread principal qui ralentissent l'application
- des chargements lourds en cours d'utilisation de l'application
- des requêtes sur des hôtes distants peu réactifs

