Projet de programmation orientée objet

Florent Avellaneda

Département Informatique et Interactions
Aix-Marseille Université

12 janvier 2014

Description de l'option

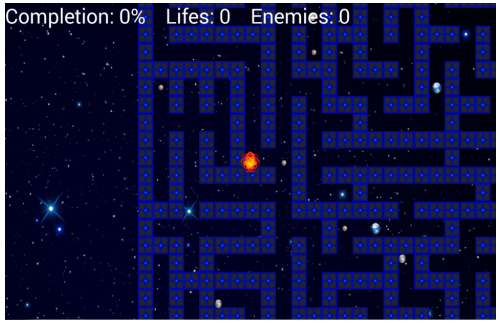
Déroulement :

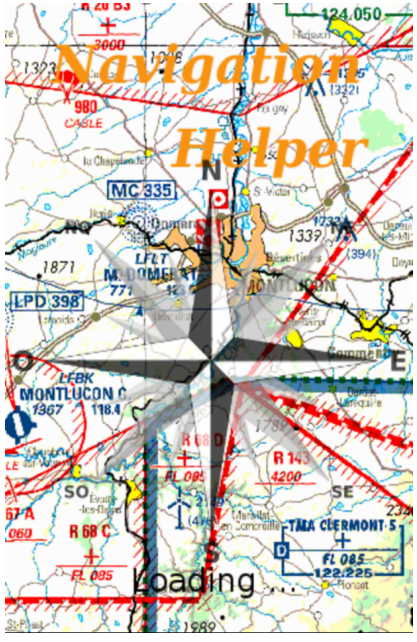
- Nombre d'heures : 60
- Rendu du projet (diagrammes de classe + code) : fin mars
- Dernier TP noté

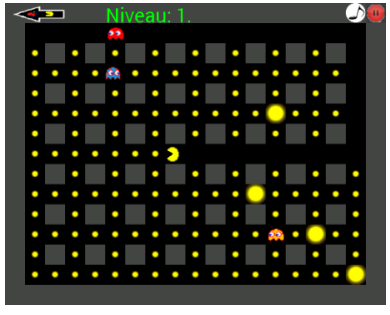
Projet :

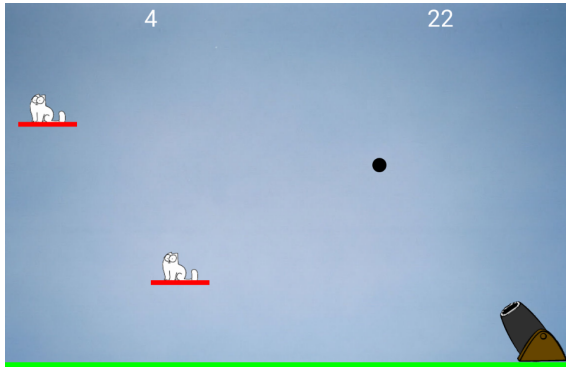
- Projet pour Android
- Maximum 3 personnes par groupe
- Critères d'évaluation :
  - L'architecture
  - La robustesse
  - La qualité de l'implémentation
- Choix du projet libre

Projets 2013



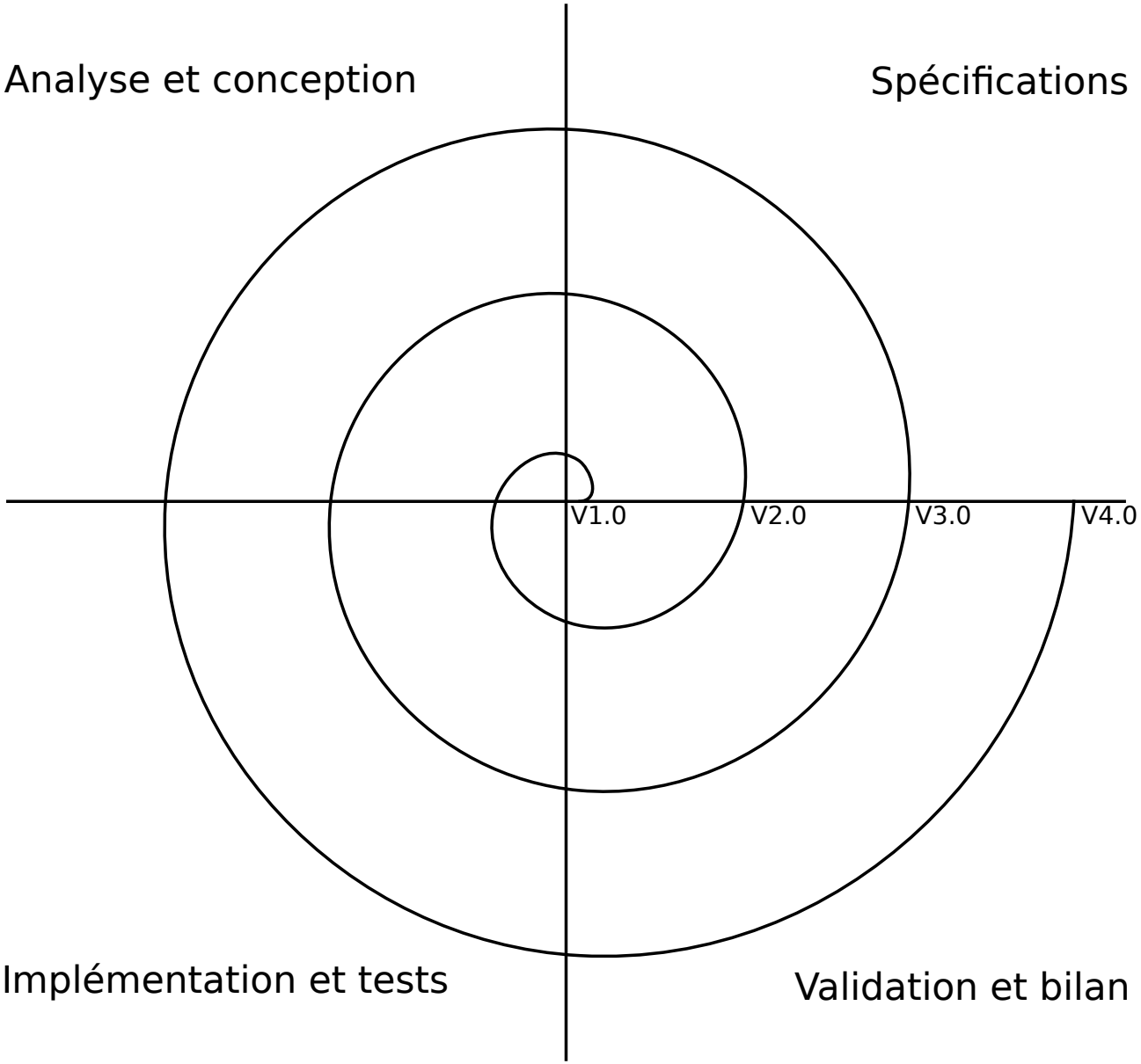






Plan

Démarche incrémentale



Analyse et conception

Quelques principes simples à respecter :

- Accorder une responsabilité cohésive à chaque entité.
- Privilégier exactitude, simplicité et clarté (KISS : Keep It Simple Software).
- Préférer la composition à l'héritage.
- Hériter, non pas pour réutiliser, mais pour être réutilisé.

Les assertions

Utiliser au maximum les assertions.

Exemple 1

```
int foo() {
  for (...)
    if (...)
      return x;
  assert false;
}
```

Exemple 2

```
switch(x) {
  case ...
  ...
  case ...
  default :
  assert false : x;
}
```

Exemple 3

```
public void add(int i) {
  ...
  assert isSorted();
}
```

Ajouter "-prop debug.assert=1" dans les arguments de l'émulateur pour activer la vérification des assertions.

Plan

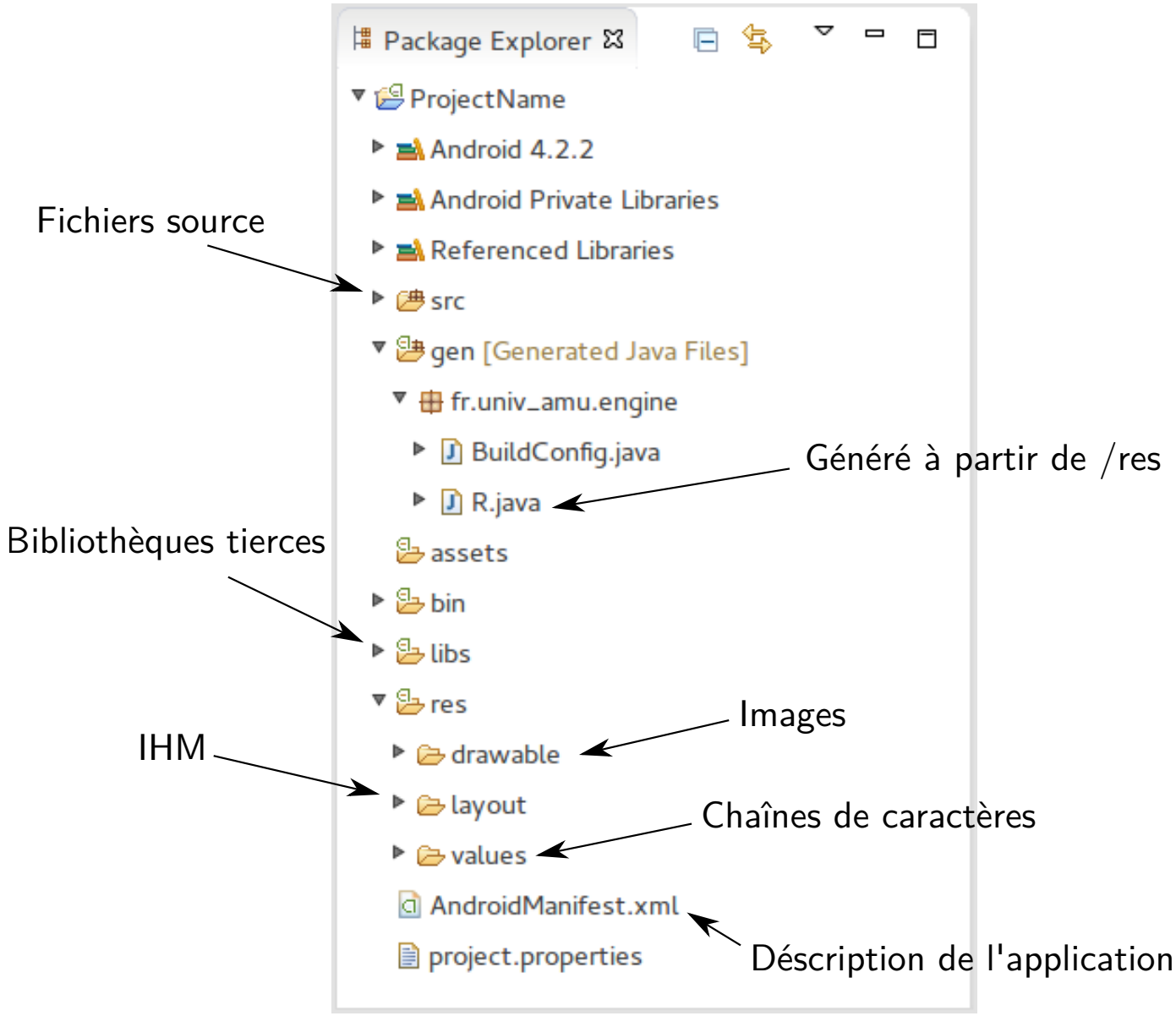
Outils de développement

ADT Bundle :

- Eclipse + ADT plugin
- Android SDK Tools
- Android Platform-tools
- Android platform
- Image Android pour l'émulateur

Lien de téléchargement : `http://developer.android.com/sdk`

Arborescence d'un projet



AndroidManifest.xml

Chaque application doit avoir un fichier *AndroidManifest.xml* dans son répertoire racine. Ce fichier contient :

- Le nom du package de l'application.
- La description de tous les composants de l'application.
- Les permissions.
- La version d'Android à utiliser.
- ...

Les ressources

L'accès aux ressources se fait par l'intermédiaire de la classe statique **R**.

Exemple

```
String s = getResources().getText(R.string.monText)
```

Exemple

```
setContentView(R.layout.mainScreen) ;
```

Exemple

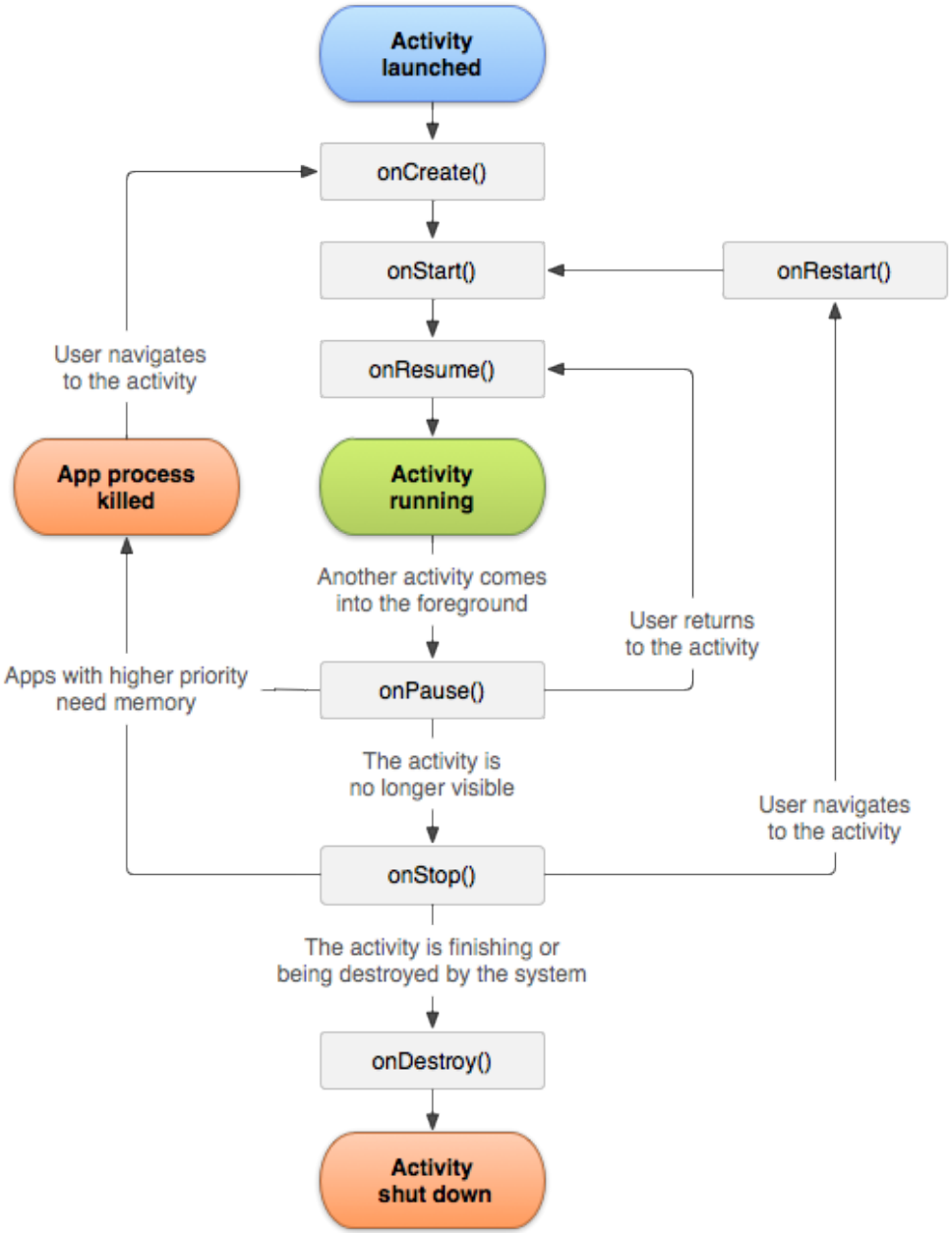
```
getWindow().setBackgroundDrawableResource(R.drawable.monImage) ;
```

Applications Android

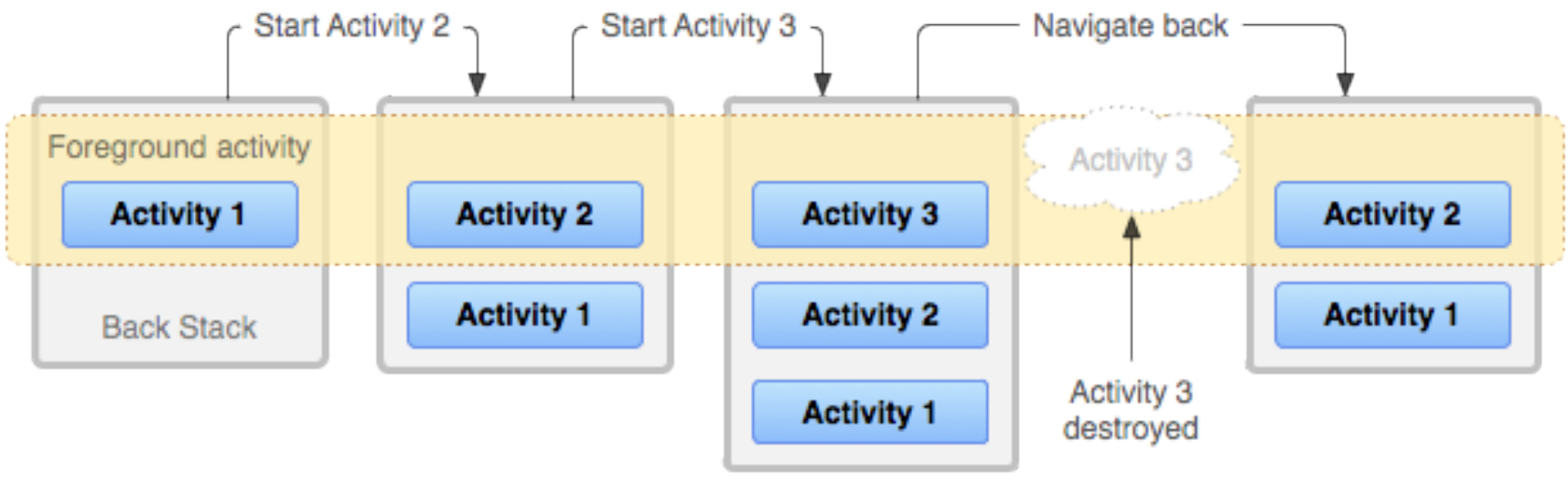
Quatre type de composants :

- *Activities* (applications de premier plan)
- *Services* (applications d'arrière-plan)
- *Content providers* (fournisseurs de contenu)
- *Broadcast receivers* (récepteur d'événements)

Cycle de vie d'une activité



Les activités sont gérées par une pile :



Communication entre composants

Les Intents permettent de gérer l'envoi et la réception de messages afin de faire coopérer les applications.

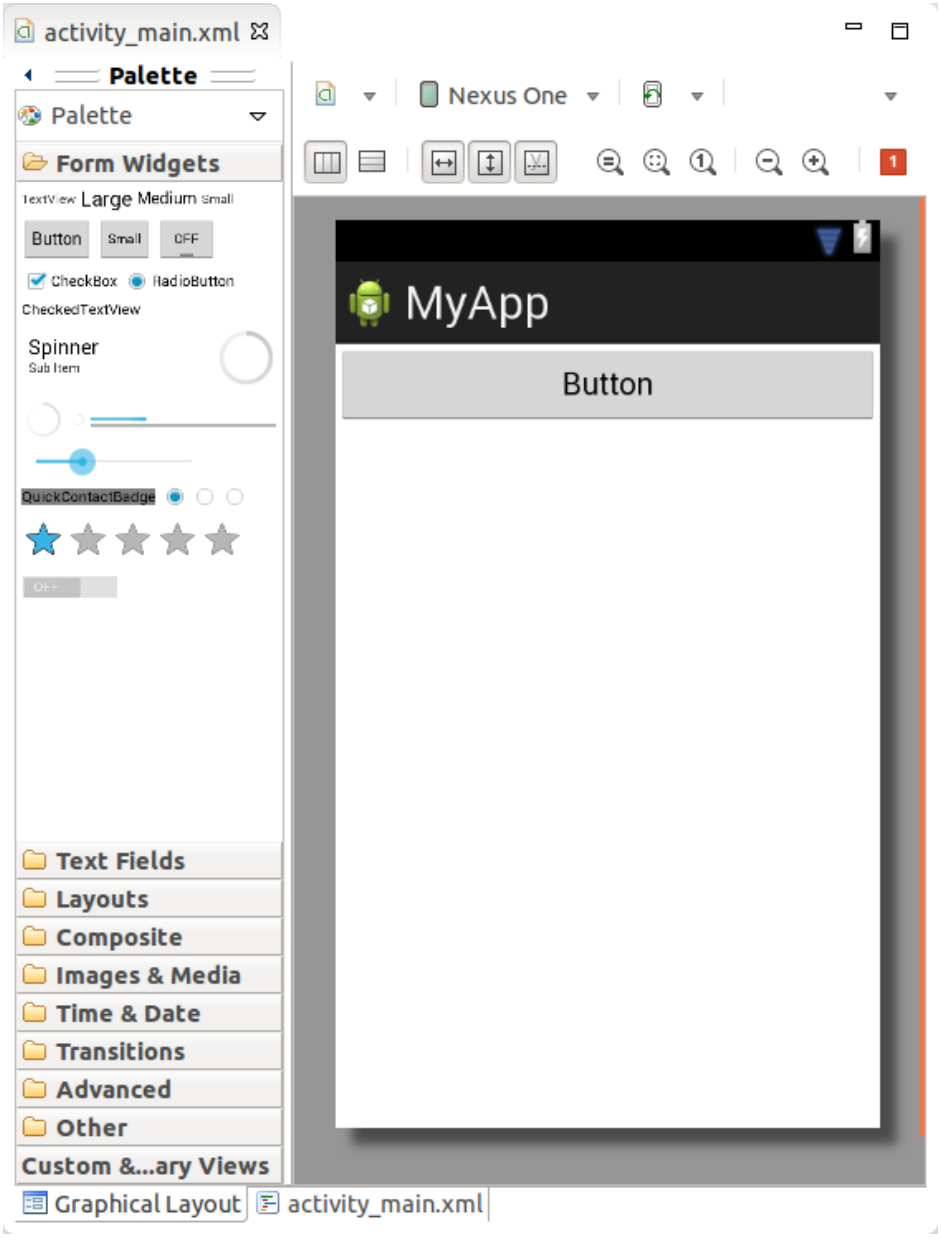
Exemple

```
Intent intent = new Intent(this, UneClass.class);
startActivity(intent);
```

Exemple

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

Les interfaces graphiques



Les interfaces graphiques

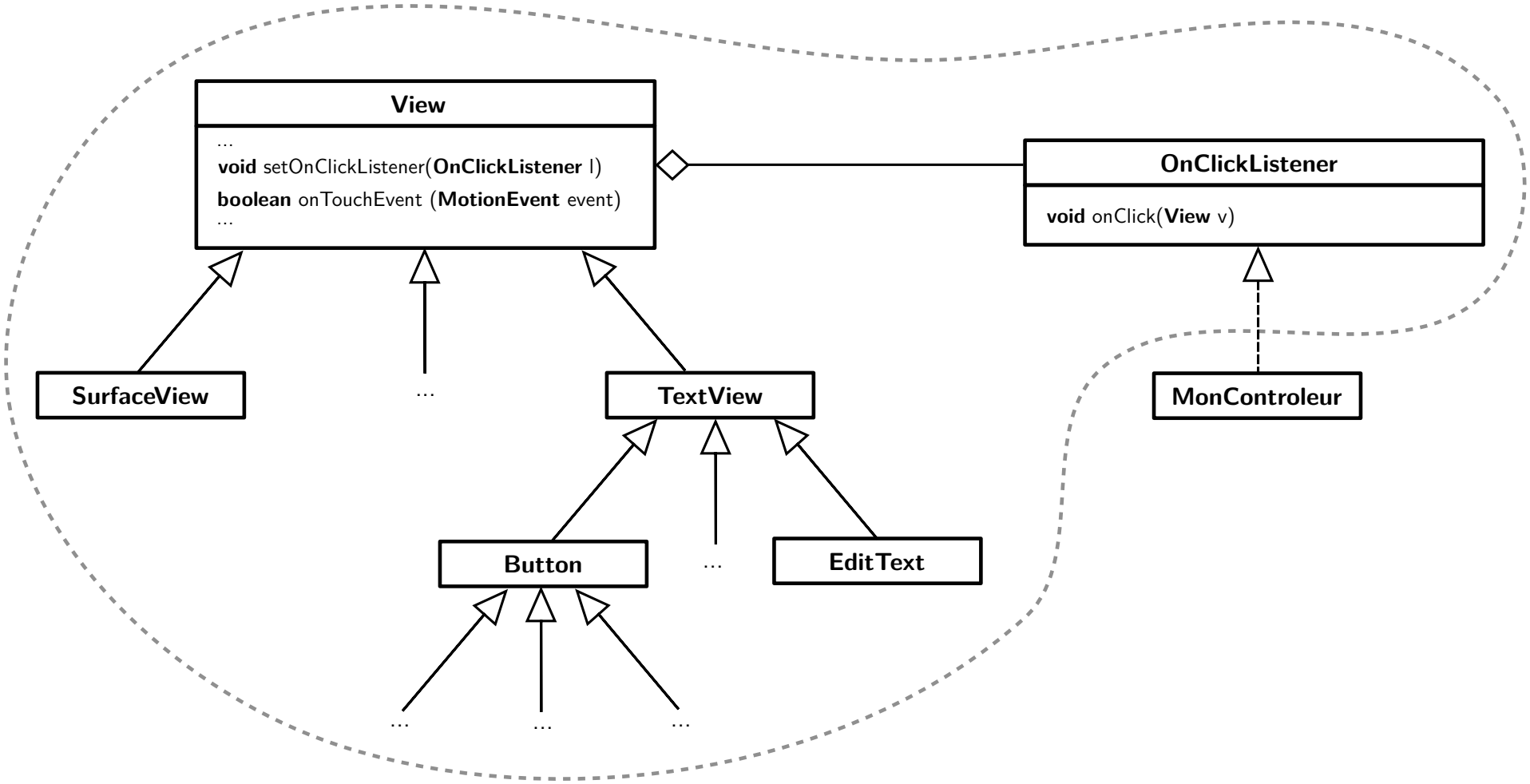
activity_main.xml

```
<LinearLayout
xmlns :android="http ://schemas.android.com/apk/res/android"
android :layout_width="match_parent"
android :layout_height="match_parent" >
  <Button
    android :id="@+id/button1"
    android :layout_width="match_parent"
    android :layout_height="wrap_content"
    android :text="Button" />
</LinearLayout>
```

Importer une IHM

```
setContentView(R.layout.activity_main);
```

Gestion des événements



Architecture souhaitable

Figure : MVP pattern

Trois couches :

- View : interface graphique de l'écran.
- Model : données métier.
- Presenter : s'occuper de la logique métier de l'écran.

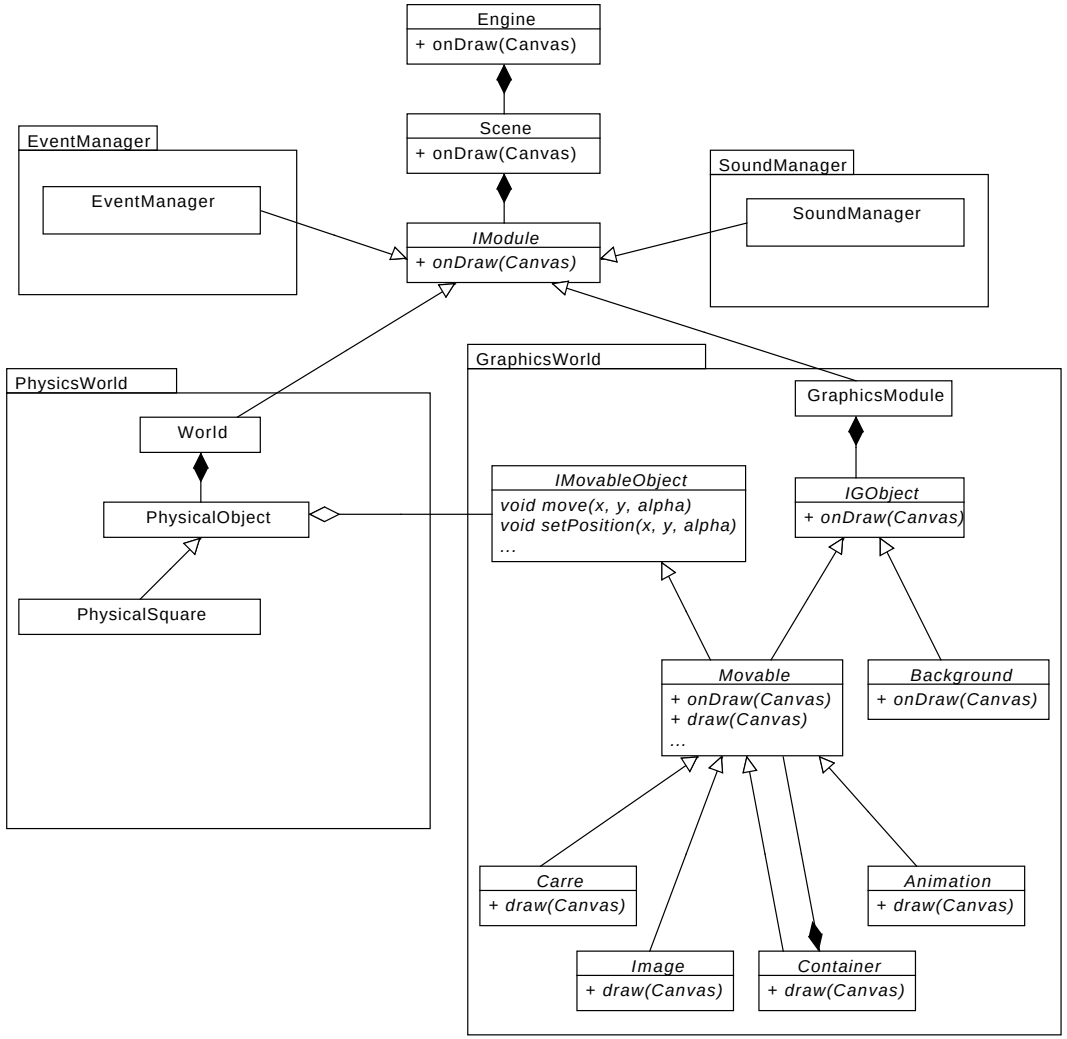
Documentation :
`http://developer.android.com/develop/index.html`

Plan

Deux phases :

1. Concevoir le moteur du jeu
   - affichage
   - physique
   - événements
   - sons
   - ...
2. Concevoir le jeu

Exemple d'architecture



Première itération

Créer une classe ayant la responsabilité de rafraichir la scène.

Cette classe devra :

- Étendre la classe View.
- Rafraichir en boucle la scène.
  - utiliser la méthode **postInvalidateDelayed**
  - limiter le taux de rafraichissement
- Calculer le temps écoulé entre deux rafraîchissements.