Interface graphique

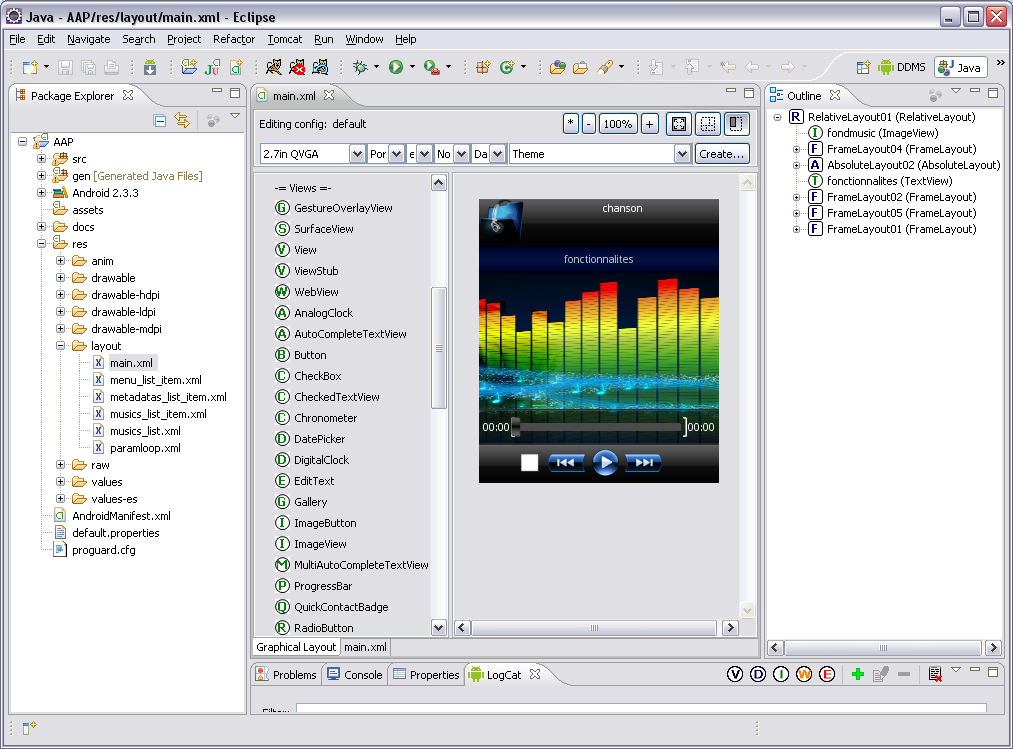
# Construction sous eclipse

Sous Android l'interface graphique est gérée dans des fichiers autres que les fichiers Java, l'interface se construit dans des fichiers XML présents dans le dossier "res/layout".

L’unité de base pour les interfaces graphique sur la plateforme Android est la “View“. C’est une structure de données dont les propriétés stockent la présentation et le contenu d'une zone rectangulaire bien spécifique de l'écran, sa mise en page, dessin, les changements de focus, le défilement... La classe “View“ sert de classe de base pour tous les widgets. On peut donc les utiliser pour construire rapidement notre interface. La liste des widgets disponibles est la suivante : TextView, EditText, Button, RadioButton, Checkbox, ScrollView, ...

Grâce au plugin ADT installé sur Eclipse, nous disposons d’un outil simplifiant la construction de l’interface graphique. En effet, cet outil fournit une liste de “Views“ qu’il suffit de faire glisser sur une zone rectangulaire représentant l’écran du téléphone. L’outil génère ensuite automatiquement le code XML représentant à cette insertion. On peut néanmoins modifier directement le fichier XML.

Certains événements sur les Views peuvent être écrits en XML (changement d’icones quand on clique sur un bouton, animation, …) mais la plupart doivent être écrit en java dans la classe principale de notre projet. L’attribution d’une action sur un évènement se fait de la même manière que pour un projet java classique.



Le projet est également constitué de ressources et d’un fichier R.java qui sont utilisés pour la construction de l’interface graphique :

Les ressources sont des fichiers externes (fichiers non-code) qui sont utilisés par notre code et compilé dans notre application au moment de la compilation. Android supporte un certain nombre de différents types de fichiers de ressources, y compris XML, PNG et JPEG. Les fichiers XML sont des formats très différents en fonction de ce qu'ils décrivent. Les ressources sont externalisées du code source. Les fichiers XML sont compilés dans un format binaire pour des raisons d'efficacité. Les chaînes de caractères sont compressées dans une forme de stockage plus efficace.

## Liste des ressources

* layout-files (fichier plans) 🡪 “/res/layout/”
* images 🡪 “/res/drawable/”
* animations 🡪 “/res/anim/”
* styles, strings and arrays 🡪 “/res/values/”
* les fichiers raw comme mp3s ou videos 🡪 “/res/raw/”

Le fichier R.java est automatiquement générer, il indexe toutes les ressources de notre projet. On utilise cette classe dans notre code source comme une sorte de passage pour référencier toutes les ressources qu’on veut inclure dans notre projet.

Pour tester notre application directement eclipse nous fournit un émulateur qui simuler un véritable téléphone Android.

# Objectifs

Dans un premier temps, l’application étant utilisée sur un téléphone Android tactile, il doit être ergonomique (la taille et la position des touches doivent être adaptées à n’importe quel type de doigt), intuitif (on doit pouvoir, d’un seul coup d’œil, savoir où cliquer en fonction de ce que l’on veut faire) et fluide (l’application doit être la plus rapide possible).

Dans un second temps, sachant que nous développons une application gérant la musique, notre interface graphique doit contenir les éléments suivants : un bouton play/pause de lecture, musique suivante/précédente, une barre de progression synchronisée avec la musique, un accès à la liste des musiques stockées sur la sdcard et enfin tous les boutons des fonctionnalités à réaliser (boucle, equalizer, etc).

# Evolution de l’interface graphique

## Démarrage

N’ayant jamais crée d’application pour téléphone Android auparavant, les premières étapes ont consisté à créer une interface graphique basique, comportant les éléments principaux, tout en découvrant le fonctionnement et les possibilités de l’outil de développement Android de eclipse. L’interface a donc évolué en fonction de notre découverte et de notre perfectionnement à cet outil.

On décomposera l’évolution de l’interface graphique en six grandes étapes majeures.

## première étape

Elle contient tous les boutons des fonctionnalités que nous voulons réaliser. La structure de l’interface a été définie et ne changera plus dans les prochaines versions :

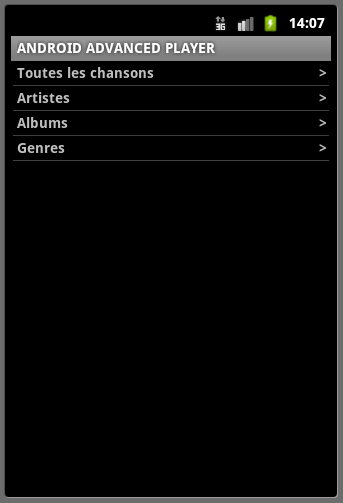
* titre, auteur de la chanson, accès a la liste des musiques en haut de l’écran
* fonction de base d’un lecteur de musique en bas de l’écran (barre de progression, bouton play/pause, musique suivante/précédente)

Aucun bouton n’est encore fonctionnel mais elle dispose déjà de la lecture d’une musique et d’une barre de progression synchronisée avec celle-ci. La musique est stockée en tant que ressource dans le répertoire “rec/raw“ de notre projet.

On a également eu l’idée de rajouter la possibilité d’afficher les paroles et les partitions dans notre application. Le centre de l’écran a donc été réservé à cet effet.



## Deuxième étape

La présentation n’a pas subit de grandes modifications. On a cependant ajouté l’accès à la liste des musiques stockées sur la sdcard. Pour l’instant, les musiques doivent être stockées dans un répertoire nommé “Music“ situé à la racine de la sdcard. Seul le clic sur “Toutes les chansons“ est fonctionnel. Le plugin Android d’eclipse dispose d’un outil permettant de simuler une sdcard.

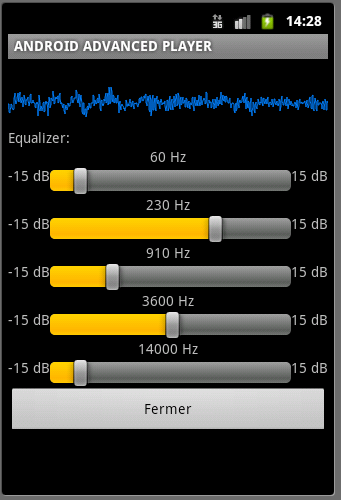
L’affichage des chansons se fait de cette manière :

4.PNG

**Artiste**  Album

Titre Durée

On a également ajouté la fonctionnalité “equalizer“. Elle permet de modifier la fréquence des notes Cette fenêtre est entièrement générée en java en fonction de la fréquence de la musique lancée. ( ? À avoir avec Anthony si ce que je dis est juste)

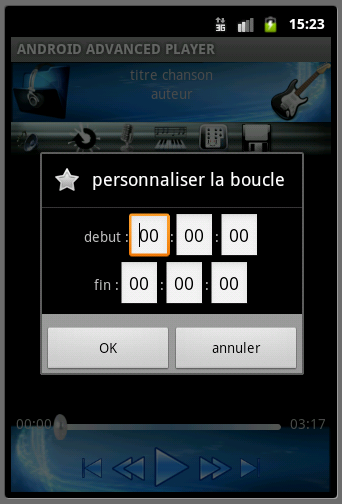


Cette version de projet dispose également d’un premier test de boucle avec des valeurs fixes de début et de fin. Elle ne marche cependant pas correctement car la musique ne retourne pas au début dès la valeur de fin atteinte mais parfois plusieurs secondes après l’avoir dépassé.

## Troisième étape

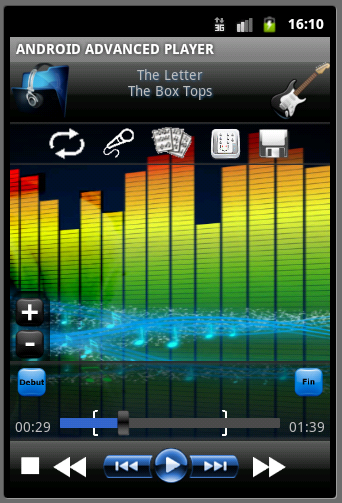
On a commencé à intégrer à l’interface graphique le réglage de début et de fin de boucle.

7.PNGOn voulait ajouter deux curseurs sur la barre de progression de la musique. On pourrait cliquer directement dessus et la faire glisser pour modifier les valeurs de la boucle.

Après de nombreuses recherches, nous n’avons pas réussi à trouver le moyen de rajouter deux nouveaux curseurs que l’on puisse déplacer comme le curseur par défaut de la barre de progression. Nous avons donc commencé, en attendant de trouver une autre solution, par afficher une fenêtre en pop-up.

## Quatrième étape

L’interface a subit un changement complet du graphisme, tout en gardant toujours la même structure définie à la première étape. En effet, après de nombreuses recherches sur la construction d’une interface graphique avec notre outil de développement Android, on n’a été en mesure d’améliorer ce qui avait été fait au départ.

On a également amélioré le réglage des valeurs de la boucle en supprimant le pop-up et en mettant à la place des boutons “+“ et “–“ de réglage. On a également réussi à placer des curseurs au même niveau que la barre de progression. Ces curseurs sont en réalité deux autres barres de progressions placées derrière la barre de progression de la musique et que l’on a personnalisé en leur définissant un style de telle sorte qu’un utilisateur lambda comprenne directement qu’il s’agit des positions de début et de fin de la boucle.

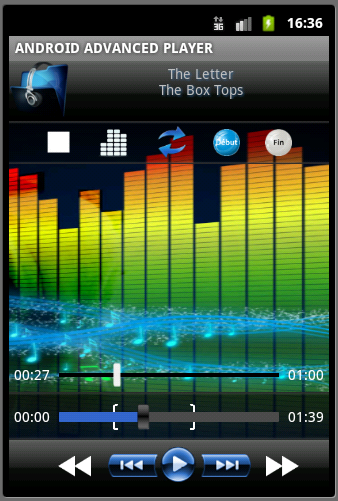
Un bouton stop a également été ajouté. Au premier clic la musique revient à la position correspondant au début de la boucle et au second clic, elle revient au début de la chanson.

On a ensuite affiché le titre et l’artiste de la chanson en fonction de celle qu’on a choisit dans la liste de lecture.

Pour finir, on a maintenant la possibilité de quitter l’application en cliquant sur le bouton “menu“ du téléphone.



## Cinquième étape

L’amélioration principale de cette étape est la mise en œuvre d’une solution finale pour le réglage de la boucle. En effet, le point négatif de la précédente solution était que si on voulait avoir un boucle qui commence vers la fin de la chanson il fallait rester appuyer un certain temps sur le bouton “+“. Ce n’était donc pas une solution optimale. Nous avons donc créé une nouvelle barre de progression juste au dessus de celle de la musique. Celle-ci permet, par un simple glissé, de modifier les valeurs de début et de fin de la boucle. Ces valeurs sont également affichés de chaque coté de la barre.

Cette étape marque également la résolution du problème qu’on avait eu dès la deuxième étape concernant la boucle. Pour synchroniser la position de la musique avec la barre de progression nous avons mis en place un thread qui déplace le curseur de la barre toutes les secondes. L’écart entre chaque mise à jour était auparavant géré par un “sleep“ de la thread. Cependant il semblerait que cette technique ne soit pas adaptée pour le développement sur Android car on a pu remarquer que le téléphone ne répondait pas correctement à ce fonctionnement. La boucle étant gérée dans cette thread, elle ne marchait donc pas correctement. Nous avons donc résolu ce problème en ajoutant à notre projet un objet “Handler“ pointant vers notre thread. Il s’agit d’une valeur numérique identifiant un objet et permettant sa manipulation. Le “Handler“ dispose d’une méthode “[postDelayed](http://developer.android.com/reference/android/os/Handler.html" \l "postDelayed(java.lang.Runnable, long))([Runnable](http://developer.android.com/reference/java/lang/Runnable.html) r, long delayMillis)“ qui permet d’éxecuter notre thread toutes les “delayMillis“ millisecondes. L’utilisation de cette technique marchant parfaitement dans notre projet, nous avons donc gardé cette solution comme solution finale.

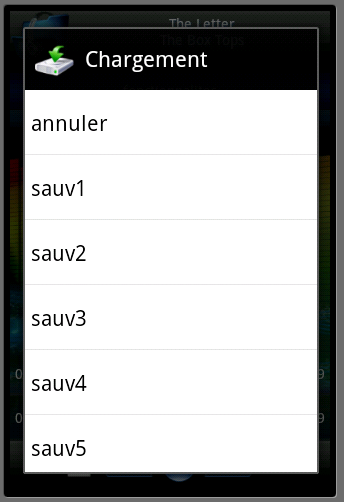
Cette étape dispose également des fonctionnalités “musique suivante/précédente“ et de la sélection d’une musique par genre, album ou artiste.

## Sixième étape

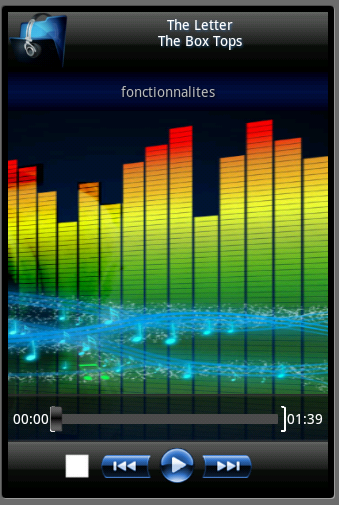
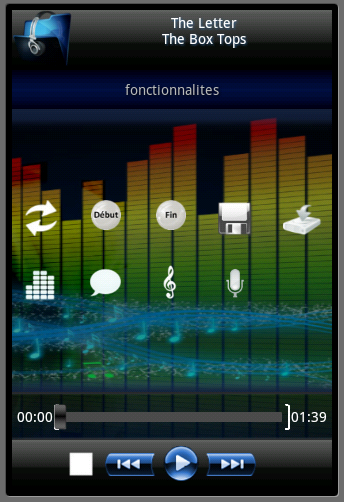
Il s’agit de l’étape finale du projet. Nous avons mit notre application en pleine écran pour récupérer un peu de place.

Liste complète des fonctionnalités réalisées :

* Boucle
* Enregistrement des paramètres de la boucle
* Chargement des paramètres de boucle
* Equalizer
* Affichage des paroles
* Affichage des partitions
* Enregistrement d’un son (pas encore fonctionnel)
* Fonctionnalités de base d’un lecteur de musique (sélection d’une musique dans une liste, musique suivante, musique précédente, play/pause, stop, modification curseur de la musique)

Dans les nouvelles fonctionnalités citées nous avons le chargement des paramètres de la boucle. Lorsqu’on enregistre une boucle, on charge dans un fichier XML le nom de la chanson, la valeur de début et la valeur de fin de la boucle. Lors du chargement, on parcourt ce fichier et on affiche dans un menu contextuel toutes les sauvegardes disponibles correspondant à la chanson en cours. Il suffit alors d’en sélectionner une dans la liste et les curseurs de début et de fin de boucle de l’interface se mettent automatiquement à jour.

Ayant ajoutés de nombreuses fonctionnalités il était impossible de faire tenir tous les icones sur une même ligne comme dans les versions précédentes de notre interface. On a donc placé tous les icones dans une fenêtre que l’on fait apparaître/disparaître (avec une animation : la fenêtre bouge de haut vers le bas et inversement) en cliquant sur “fonctionnalités“.



(+ Détail affichage page web avec paroles et partitions et gestion de l’enregistrement d’un son si on le fait)

# Sources

Pour toute la durée du développement :

<http://android.cyrilmottier.com/>

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

<http://android-france.fr/category/developpement/>

Création des menus de l’interface :

<http://wiki.frandroid.com/wiki/Menus_Android>