

Page de présentation



Sommaire

Introduction	3
Objectifs	3
Contexte	3
Android	3
Qu'est-ce qu'Android ?	3
Caractéristiques	3
Et pour les développeurs ?	4
Organisation et déroulement du projet	4
Outils et technologies utilisés	4
Partage du code	4
Google Code et Project Hosting	4
Subversion	4
Développement	5
Langage Java	5
Eclipse Pulsar	5
Android Software Development Kit (SDK)	5
Android Development Tools (ADT)	5
L'interface graphique	5
Les besoins	6
Conception sous Eclipse	6
Les ressources et le fichier R.java	7
Construction d'une interface	8
Évolution de l'interface graphique	13
Démarrage	13
Première étape	13
Seconde étape	14
Troisième étape	14
Quatrième étape	15
Dernière étape	16
Les fonctionnalités	18
L'equalizer	18
Qu'est-ce qu'un equalizer ?	18
Choix de la base de travail	19
Fonctionnement	19
Implémentation	19
Time Stretching	20
Le concept	20
Fonctionnement	20
Implémentation	20
Rétro-ingénierie	21
Glossaire	23
Références	26



Introduction

Objectifs

Il s'agissait de développement d'un lecteur MP3 pour musicien, pour téléphones Android^[1]. Ce lecteur devait intégrer les fonctionnalités suivantes :

- boucler sur un morceau ;
- ralentir un morceau sans changer la hauteur des notes ;
- changer la tonalité d'un morceau sans le ralentir.

L'interface utilisateur du lecteur devait être ergonomique, les musiciens n'étant pas forcément familier avec les nouvelles technologies.

Contexte

Plusieurs lecteurs MP3 pour téléphones Android existent déjà, le lecteur multimédia par défaut notamment.

En outre, la technologie Android bénéficie d'une très grosse communauté de développeurs, il est donc fort probable que certaines fonctionnalités du lecteur MP3 existent déjà sur la toile, et peuvent être récupérées et intégrées à notre solution.

Android

Qu'est-ce qu'Android ?


Android est un système d'exploitation^[2] open source^[3] pour smartphones^[4], PDA^[5] et terminaux mobiles conçu par Android, une startup rachetée par Google en novembre 2007. Également, on voit apparaître d'autres types d'appareils possédant ce système d'exploitation, tels que les téléviseurs, tablettes et les autoradios.

Pour diffuser en masse son système, Google a fédéré autour d'Android une trentaine de société (dont Samsung, Motorola, Sony Ericsson ou encore LG) à l'intérieur de l'Open Handset Alliance.

Le premier appareil tournant sous Android sorti sur le marché fut le HTC Dream, un smartphone possédant un clavier physique. Le second, le HTC Magic suivi du Samsung Galaxy, et plus tard du HTC Hero.

Caractéristiques

Android est un système d'exploitation fondé sur un noyau Linux^[6], il comporte une interface spécifique développée en Java^[7], les programmes sont exécutés via un interpréteur JIT^[8], toutefois il est possible de passer outre cette interface, en programmant en C^[9], mais le travail de portabilité en sera plus important.



Disponible via une licence Apache ^[10] version 2, le système d'exploitation inclut tous les utilitaires requis par un constructeur ou par un opérateur pour mettre en œuvre un téléphone portable.

Android a été conçu pour intégrer au mieux des applications existantes de Google comme le service de courrier Gmail, celui de cartographie, Google Maps, ou encore Google Agenda, Google Talk, Youtube. Un accent particulier est mis sur la géo-localisation avec Google Latitude et la météo correspondant à la ville la plus proche disponible sur le menu principal.

Et pour les développeurs ?

Android fournit un kit de développement (SDK) permettant de développer des applications spécifiques de la téléphonie mobile à mettre en œuvre sur la plateforme.

La connaissance des technologies Java, idéalement, et XML ^[11] sont nécessaires. En effet, si le code est en Java, toutes les interfaces graphiques et le contenu des fichiers ressources sont en XML ^[11].

Nous vous invitons fortement à consulter le guide du développeur sur le site de la société Android ou sur le wiki du site FrAndroid (*cf: Références*).

Organisation et déroulement du projet

Outils et technologies utilisés

Partage du code

À plusieurs sur un même projet, et parfois une même partie du projet, nous avons dû mettre en place un système pour partager notre code et gérer les versions de nos fichiers.

Google Code et Project Hosting

Google Code est un site web destiné aux développeurs intéressés par le développement relatif à Google. L'entreprise y diffuse des codes sous licence libre ainsi que la liste des API ^[12] utilisables.

Project Hosting est un projet visant à apporter des services gratuits aux développeurs open-source de la même façon que le fait déjà SourceForge.net. Il est intégré au site Google Code et les projets pouvant y être soumis devront être sous une licence open-source.

Subversion

Subversion, ou SVN, est un système de gestion de versions de fichiers, distribué sous licence Apache ^[10] et BSD ^[13]. Il a été conçu pour remplacer CVS, Concurrent Version System. Le projet a été lancé en février 2000 par CollabNet, entreprise informatique qui vend des services et du logiciel, avec l'embauche par Jim Blandy et Karl Fogel, qui travaillait déjà sur un nouveau



gestionnaire de version. En février 2010, SVN est devenu officiellement un projet de la Fondation Apache, sous le nom d'Apache Subversion.

SVN peut s'utiliser en ligne de commande ou à l'aide d'un logiciel intégré tel que TortoiseSVN.

Développement

Langage Java

Nous avons utilisé le langage Java pour développer notre application Android. Nous avons fait ce choix car ce langage nous est familier, et qu'il s'intègre parfaitement au système d'exploitation Android, développé lui aussi en Java.

Eclipse Pulsar

Eclipse est un environnement de développement intégré (IDE) libre, extensible, universel et polyvalent, permettant de créer des projets de développement mettant en œuvre n'importe quel langage de programmation. Il est principalement écrit en Java, et ce langage, grâce à des bibliothèques ^[14] spécifiques, est également utilisé pour écrire des extensions.

Eclipse Pulsar est une variante d'Eclipse. Il s'agit du logiciel Eclipse configuré pour le développement de programmes destinés aux appareils mobiles. Eclipse Pulsar est ainsi une plateforme d'outils intégrés pour le développement sur appareils mobiles.

Nous avons choisi cet IDE car il permet d'intégrer parfaitement SVN et l'utilisation des technologies pour Android.

Android Software Development Kit (SDK)

Il s'agit d'un ensemble d'outils permettant aux développeurs de créer des applications pour Android.

Le SDK est disponible en téléchargement libre sur le site de la société Android.

Android SDK est un outil indispensable pour développer des applications Android.

Android Development Tools (ADT)

ADT est un plugin ^[15] pour Eclipse. Il a été conçu pour simplifier le développement d'application Android avec Eclipse. Il vient donc étendre les fonctionnalités d'Eclipse dans le but de développer rapidement des interfaces graphiques, d'ajouter des composants basés sur l'API ^[12] Android, de faciliter l'utilisation des outils proposés par Android SDK.

Nous avons choisi d'utiliser ce plugin dans le but de nous faciliter le développement et d'éviter au maximum les éventuels problèmes, problèmes qui ne se manifestent généralement pas en utilisant ADT.

L'interface graphique



Les besoins

L'application étant utilisée sur un téléphone Android tactile, l'interface doit être :

- **ergonomique** : la taille et la position des touches doivent être adaptées à n'importe quel type de doigt ;
- **intuitive** : on doit pouvoir, d'un seul coup d'œil, savoir où cliquer en fonction de ce que l'on veut faire ;
- **fluide** : l'application doit être la plus rapide possible.

Sachant que nous développons une application gérant la musique, notre interface graphique doit contenir les déclencheurs des fonctionnalités suivantes :

- lecture d'un morceau ;
- mise en pause d'un morceau ;
- passer au morceau suivant ou précédent ;
- accéder directement à une partie d'un morceau ;
- accéder à la liste des musiques stockées sur la carte SD ^[16] ;
- afficher l'égaliseur ;
- créer une boucle de lecture sur un morceau.

Conception sous Eclipse

Sous Android, l'interface graphique est gérée dans des fichiers autres que les fichiers Java ^[7] : l'interface se construit dans des fichiers XML ^[11] présents dans le dossier "/res " du projet.

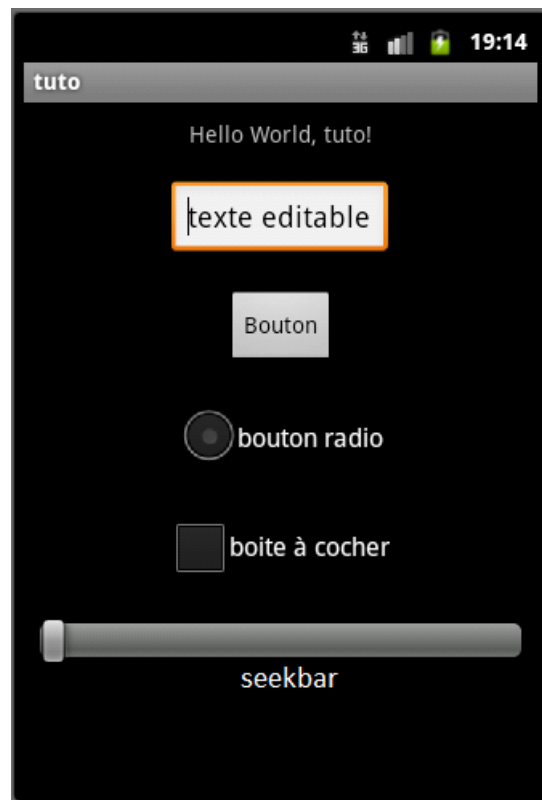
L'unité de base pour les interfaces graphiques sur la plateforme Android est la **View** : composant de l'interface graphique avec lequel un utilisateur peut interagir. C'est une structure de données dont les propriétés stockent la présentation et le contenu d'une zone rectangulaire bien spécifique de l'écran, sa mise en page, les changements de focus (élément affiché qui reçoit les ordres de l'utilisateur), le défilement, etc. Tous les éléments de l'interface héritent de la classe **View**.

Grâce au plugin ^[15] ADT installé sur Eclipse, nous disposons d'un outil simplifiant la construction de l'interface graphique. En effet, cet outil fournit une liste de **View** qu'il suffit de faire glisser sur une zone rectangulaire représentant l'écran du téléphone. L'outil génère ensuite automatiquement le code XML représentant cette insertion. On peut néanmoins éditer ce code XML sans pour autant passer par le plugin ADT.

De nombreux widgets ^[17] sont disponibles par défaut grâce à cet outil (**TextView**, **EditText**, **Button**, **RadioButton**, **Checkbox**, **SeekBar**, etc.).



Quelques widgets



Certains événements, action qui se produit par exemple lorsque l'on clique sur un bouton, peuvent être écrits en XML, mais la plupart doivent être écrits en Java dans la classe principale de notre projet. L'attribution d'une action sur un événement se fait de la même manière que pour un projet Java classique.

Les ressources et le fichier R.java


Le projet est également constitué de ressources et d'un fichier R.java, 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. Elles sont toutes stockées dans le répertoire « /res » du projet. Android supporte un certain nombre de différents types de fichiers de ressources, y compris XML ^[11], PNG ^[19] et JPEG ^[20]. 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.

Le fichier R.java est automatiquement généré : il indexe toutes les ressources de notre projet. On utilise cette classe dans notre code source comme une sorte de passage pour référencer toutes les ressources qu'on veut inclure dans notre projet. Par exemple, pour attribuer une animation à un élément de notre interface, on accèdera à la ressource correspondante de la manière suivante : ***R.anim.monanimation***, « monanimation.xml » étant un fichier se trouvant dans le répertoire « /res/anim ».

Voici une liste exhaustive des ressources existantes au sein du projet :

- Animations : « /res/anim », accessible par ***R.anim*** ;

- 
- Couleurs : « /res/color », accessible par **R.color** ;
 - Images et variations d'images : « /res/drawable », accessible par **R.drawable** ;
 - Mises en page de notre interface : « /res/layout », accessible par **R.layout** ;
 - Menus : « /res/menu », accessible par **R.menu** ;
 - Textes et tableaux de textes : « /res/values », accessible par **R.string**, **R.array** ;
 - Styles : « /res/values », accessible par **R.style** ;
 - Fichiers raw comme mp3s ou videos : « /res/raw », accessible par **R.raw** ;
 - Autres (booléens, entiers, dimensions, etc.) : « /res/values », accessible chacun par **R.bool**, **R.integer**, **R.dimen**, etc.

Construction d'une interface

Une fois le projet de type « Android Application » créé sous Eclipse, un fichier représentant l'interface graphique est automatiquement créé et stocké dans le répertoire « /res/layout » du projet.

Pour que l'application comprenne que ce fichier correspond à celui qu'il faut lancer, la classe java de notre activité principale doit contenir :

```
public class tuto extends Activity {  
    /** Appel quand l'activité est créé pour la première fois */  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        /** « main » est le nom du fichier XML contenant la description de  
            notre interface */  
        setContentView(R.layout.main);  
    }  
}
```

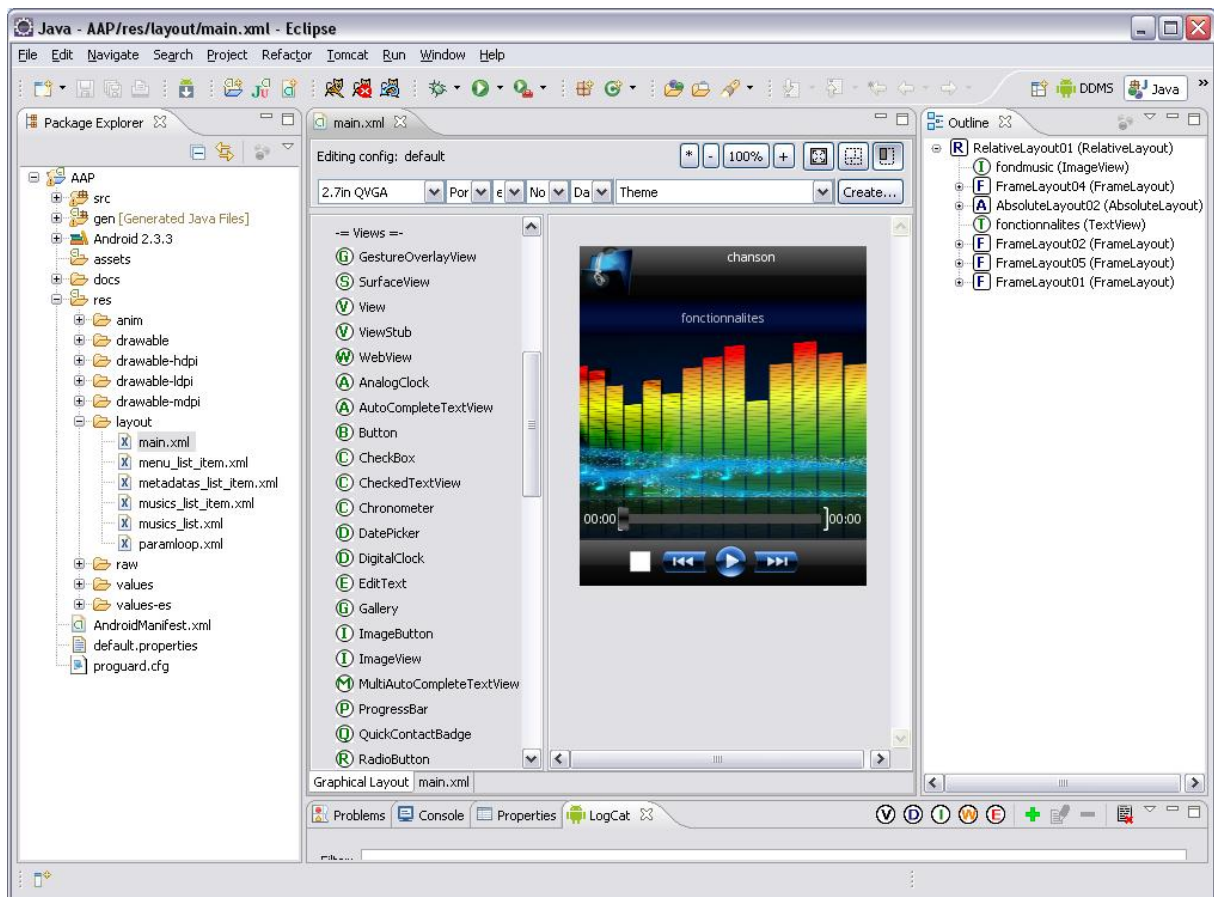
« tuto » est une **Activity** liée à notre interface. Il permet de gérer les événements se produisant sur celle-ci.

Lorsqu'on ouvre le fichier XML ^[11] de notre interface, grâce au plugin ADT, on a :

- au centre, une zone rectangulaire représentant l'écran du téléphone ;
- à gauche, une liste d'éléments que l'on peut placer ;
- à droite, la hiérarchie des éléments présents sur notre interface.

De plus, si on sélectionne un des éléments déjà placé, on peut modifier ses propriétés grâce à l'onglet « Propriétés » situé en bas de la fenêtre.

Création d'une interface graphique avec l'outil ADT dans Eclipse



Les layouts

Avant de commencer à placer des éléments sur notre interface, il faut définir l'allocation de l'espace de l'écran grâce à des **Layout**.

Ces derniers représentent l'agencement des différents éléments graphiques dans notre interface.

Voici quelques exemples de **Layout** :

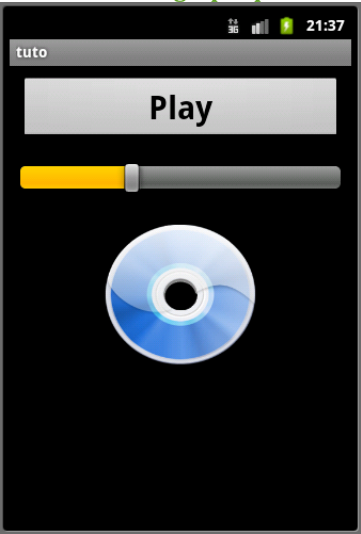
- **LinearLayout** : il organise les différents éléments de l'interface sur une ligne ou sur une colonne ;
- **AbsoluteLayout** : cette mise en page laisse définir les coordonnées exactes des éléments ;
- **RelativeLayout** : il permet de définir la position des éléments en fonction de la position de leurs éléments parents ou d'autres éléments (à droite de, au dessus de, etc.) ;
- **TableLayout** : il range les enfants sous forme de lignes ou de colonnes ; il est composé d'un ensemble de **TableRow**, chacun définissant une ligne ou une colonne ; les éléments sont ensuite insérés dans cet objet.

Une fois que l'on en a choisi un et qu'on l'a fait glisser au centre sur la zone rectangulaire simulant l'écran d'un smartphone, on peut placer nos différents éléments tels que des zones de texte, des boutons, des boutons images, une barre de progression, etc., toujours en faisant glisser l'élément

de la liste vers l'endroit souhaité sur l'écran. Une fois un élément placé, on peut modifier ses propriétés à volonté (exemple pour un bouton : on peut modifier sa taille, le texte, le fond, l'espace avec les autres éléments, etc.). Pour attribuer une image à un élément **ImageButton**, il faudra noter dans la propriété **Background** : **@drawable/img**, notre image ayant pour nom img.PNG ^[19] et étant placée dans le répertoire « /res/drawable » de notre projet.

Le descripteur de notre interface : le fichier XML

Lorsque l'on place les éléments que l'on veut dans la zone rectangulaire représentant l'écran du téléphone, tout ceci est en réalité écrit en XML ^[11] : c'est le plugin ^[15] ADT qui se charge de traduire nos agencements graphiques en XML.

Interface graphique	Équivalent XML
	<pre><?xml version="1.0" encoding="utf-8"?> <LinearLayout xmlns:android=http://schemas.android.com/apk/res/ android android:orientation="vertical" android:layout_width="fill_parent" android:layout_height="fill_parent"> <Button android:id="@+id/Button01" android:layout_height="wrap_content" android:text="Play" android:textStyle="bold" android:textSize="30dp" android:layout_gravity="center_horizontal" android:layout_width="fill_parent" android:layout_margin="10dp"> </Button> <SeekBar android:id="@+id/SeekBar01" android:layout_height="wrap_content" android:layout_width="match_parent" android:layout_margin="10dp"> </SeekBar> <ImageButton android:id="@+id/ImageButton01" android:background="@drawable/image" android:layout_gravity="center_horizontal" android:layout_height="150dp" android:layout_width="150dp" android:layout_margin="10dp"> </ImageButton> </LinearLayout></pre>

Dans cette mise en page, nous avons placé trois éléments : un bouton, une SeekBar (barre de progression) et une image bouton.

Tout ce qui commence par « android : » correspond aux différentes propriétés que l'on affecte à nos widgets ^[17]. On retrouve notamment pour chacun :

- **id** : identifiant unique pour chaque objet ;

- **layout_height** : taille en hauteur (« wrap_content », taille de l'élément par défaut ou « fill_parent », taille du parent) ;
- **layout_width** : taille en largeur (« wrap_content », taille de l'élément par défaut ou « fill_parent », taille du parent) ;
- **layout_margin** : espace libre que l'on veut autour de notre élément.

Les animations

Il est possible d'associer des animations, grâce à d'autres fichiers XML ^[11], sur nos éléments. On détaillera le fonctionnement de deux animations utilisées dans notre projet :

- lorsque l'on clique sur un bouton image, au moment où le bouton est pressé, l'image change, et lorsque l'on relâche la pression, le bouton revient à son état d'origine : pour cela, on crée un nouveau fichier XML que l'on place dans le répertoire « /drawable », ce fichier sera constitué de la manière suivante :

```
<?xml version="1.0" encoding="utf-8"?>
<selector
  xmlns:android="http://schemas.android.com/apk/res/android">
  <item
    android:state_pressed="true"
    android:drawable="@drawable/cdclick" <!-- pressed -->
  />
  <item
    android:drawable="@drawable/cd" <!-- default -->
  />
</selector>
```

Pour affecter cette animation à un élément il suffit de modifier la propriété « Background » et de lui attribuer la valeur **@drawable/nomdufichier**.

- lorsque l'on clique sur un bouton, nous voulons afficher un autre élément par un effet de **slide** du haut vers le bas : comme précédemment, nous allons créer un nouveau fichier XML que l'on placera dans le répertoire « /res/anim/ », ce fichier sera constitué de la manière suivant :

```
<set
  xmlns:android=http://schemas.android.com/apk/res/android
  android:interpolator="@android:anim/accelerate_interpolator">
  <translate
    android:fromYDelta="-100%"
    android:toYDelta="0%"
    android:duration="1000"
  />
</set>
```

- **accelerate_interpolator** : il correspond au type de l'animation. L'animation commence doucement puis s'accélère. Il existe de nombreux autres types d'animation comme par exemple **rotateAnimation**, qui contrôle la rotation de l'objet, **linearInterpolator**, qui rythme le changement d'état comme constant, etc.



- **translate** : animation qui joue sur la position de l'élément. Ici, nous faisons apparaître l'élément en partant du haut vers le bas. « duration » définie la durée de l'animation en millisecondes : ici, nous avons fixé la durée à 1 seconde. Il existe également les animations **alpha**, qui joue sur la transparence d'un élément, **scale**, qui joue sur sa taille, **rotate**, qui donne des effets de rotation sur un élément.

Pour affecter l'animation à un objet et pour qu'elle se lance à un moment précis, cela doit être définie dans la classe Java ^[7] de notre activité :

```
/* On commence par récupérer l'objet que l'on veut animer. Tous les objets de notre
interface sont récupérables par leur id */
FrameLayout frame = (FrameLayout)
findViewById(R.id.fenetreAAfficher);
/* On crée un objet "Animation" en le liant au fichier que l'on vient de créer */
Animation a = AnimationUtils.loadAnimation(AAP.activity,
                                           R.anim.fichiernAnimation);
/* On lui affecte ensuite l'animation et on la lance */
frame.startAnimation(a);
```

Les évènements

Pour un évènement « clic », tous les éléments disposent de la propriété « onclick ». Il suffit d'attribuer à cette propriété une méthode qui devra être lancée lorsque l'utilisateur cliquera sur l'élément. La méthode sera ensuite déclarée dans le fichier Java ^[7] de notre activité sous la forme :

```
public void methode(View v) { ... }
```

Elle doit obligatoirement avoir comme attribut un objet **View** : il doit correspondre à l'élément sur lequel on applique l'évènement.

Pour les autres types d'évènements tels que le « pressé/relevé » sur un bouton, la modification par glissé de la position du curseur d'une barre de progression, etc., ils doivent être déclarés en Java, comme pour un projet classique.

Exemple pour le « pressé/relevé » sur un bouton :

```
/* On récupère l'objet et on lui affecte un écouteur sur un évènement. */
((ImageButton) findViewById(R.id.monBouton)).setOnClickListener( new
OnClickListener() {
    @Override
    public boolean onTouch(View v, MotionEvent event) { ... }
})
```

Évolution 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 d'ADT sous Eclipse. L'interface a donc évolué en fonction de notre découverte et de notre perfectionnement d'utilisation de cet outil.

On décomposera l'évolution de l'interface graphique en cinq grandes étapes majeures.

Première étape

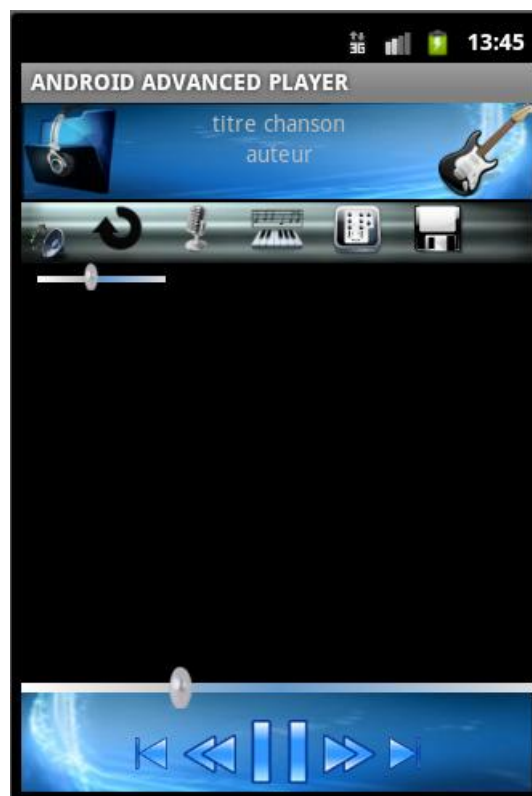
Elle contient tous les boutons des fonctionnalités que nous voulons réaliser. La structure de l'interface a été définie et ne changera plus dans les prochaines versions :

- 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 lecture/pause, bouton suivant/précédent).

Aucun bouton n'est encore fonctionnel, mais l'interface 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 « /res/raw » du projet.

Nous avons également eu l'idée d'implémenter la possibilité d'afficher les paroles et les partitions de la musique en cours de lecture au sein de notre interface. Le centre de l'écran a donc été réservé à cet effet.

Interface version 1



Seconde étape

La présentation n'a pas subi de grandes modifications. Nous avons cependant ajouté l'accès à la liste des musiques stockées sur la carte SD ^[16] et la fonctionnalité **Equalizer**.

En outre, nous avons commencé à implémenter le réglage de début et de fin de boucle : nous voulions ajouter deux curseurs sur la barre de progression de la musique, ainsi, l'utilisateur pourra cliquer directement dessus et la faire glisser pour modifier les valeurs de la boucle.

Objectif final pour la barre de progression



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 (nouvelle fenêtre s'ouvrant au dessus de la fenêtre actuelle). La position de début et de fin de boucle est alors rentrée en saisissant directement les valeurs au clavier.

Réglage de la boucle version 1



Troisième étape

L'interface a subi un « relooking » complet, 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 ADT, nous avons été en mesure d'améliorer ce qui avait été fait au départ.

Nous avons amélioré le réglage des valeurs de la boucle en supprimant le pop-up et en mettant à la place des boutons « + » et « - » de réglage. Nous avons 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 nous avons 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.

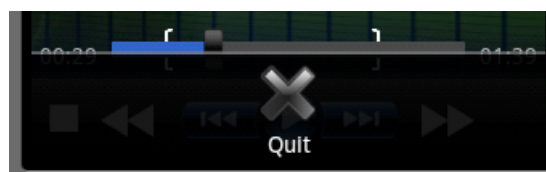
Nous avons ensuite affiché le titre et l'artiste de la chanson correspondant à la musique sélectionnée par l'utilisateur dans la liste de lecture.

Pour finir, nous avons maintenant la possibilité de quitter l'application en cliquant sur le bouton « menu » du téléphone.

Interface version 2



Quitter l'application



Quatrième 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 concerné la boucle : si

nous voulions avoir une boucle qui commençait vers la fin de la chanson, il fallait rester appuyer un certain temps sur le bouton « + », ce qui n'était pas ergonomique. Nous avons donc créé une nouvelle barre de progression juste au dessus de celle de la musique. Celle-ci permet, par un simple glissé/déposé, de modifier les valeurs de début et de fin de la boucle. Ces valeurs sont également affichées de chaque côté de la barre.

Cette étape dispose également des fonctionnalités « suivant/précédent » et de la sélection d'une musique par genre, album ou artiste.

Réglage de la boucle version finale



Dernière étape

Il s'agit de l'étape finale du projet. Nous avons mis notre application en plein-écran pour récupérer un peu de place.

Liste complète des fonctionnalités réalisées :

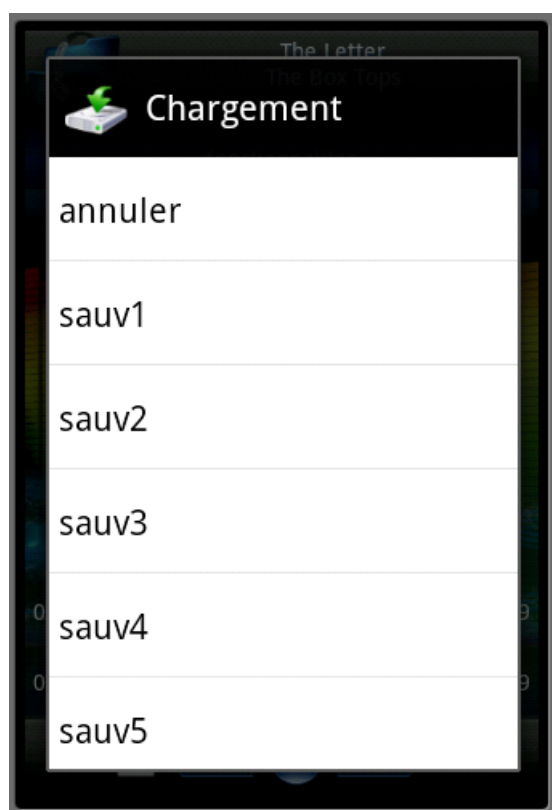
- boucle ;
- enregistrement des paramètres de la boucle ;
- chargement des paramètres de boucle ;
- equalizer ;
- affichage des paroles ;
- affichage des partitions ;
- enregistrement d'un son ;
- sélection d'une musique selon les tris « Toutes les chansons », « Artiste », « Album », « Genre » ;
- fonctionnalités de base d'un lecteur de musique (lecture/pause, suivant/précédent, stop, accès direct à une partie du morceau) ;

Dans les nouvelles fonctionnalités citées nous avons le chargement des paramètres de la boucle. Il est maintenant possible de sauvegarder les paramètres de la boucle pour une chanson et de les charger ensuite. Nous avons donc fait un menu contextuel (un menu qui s'ouvre lorsqu'un utilisateur clique sur un objet de l'interface, offrant ainsi une liste d'options qui varient selon l'objet) qui s'ouvre lorsque l'utilisateur clique sur l'icône de chargement d'une sauvegarde.

Bouton de chargement d'une boucle



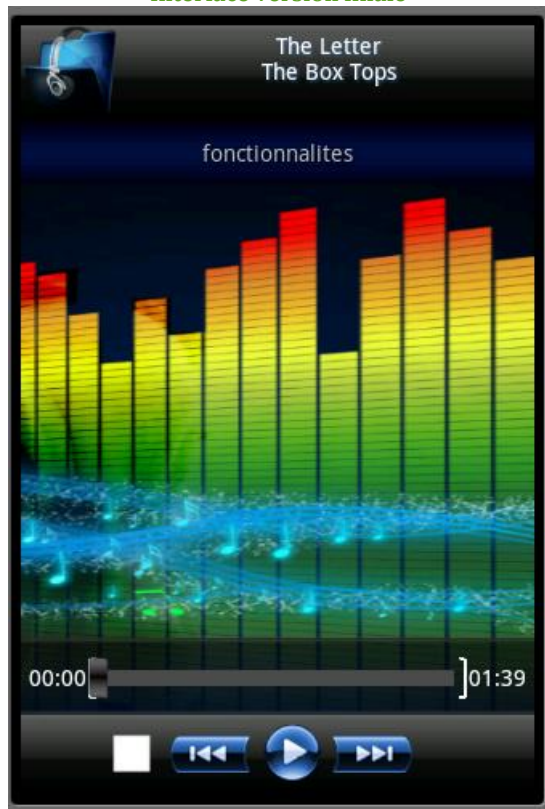
Menu de chargement d'une boucle sauvegardée en mémoire



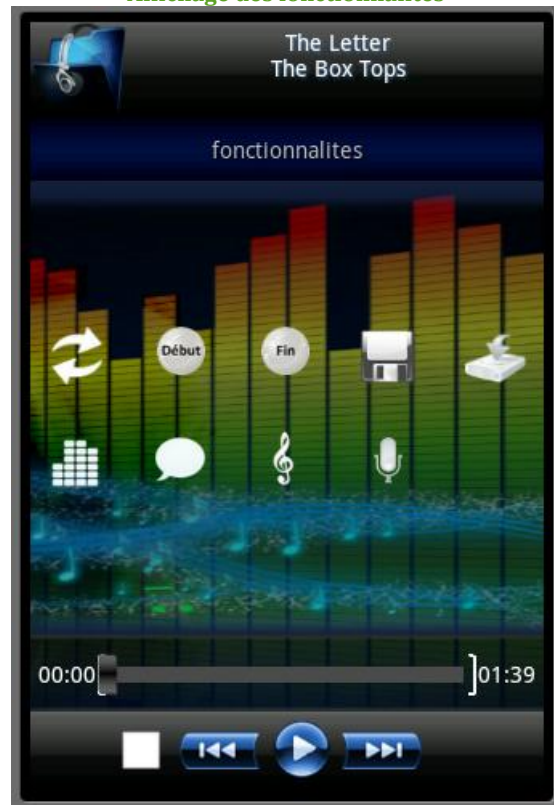
Ce menu contient toutes les sauvegardes disponibles correspondant au morceau en cours de lecture. Il suffit alors d'en sélectionner une dans la liste et les curseurs de début et de fin de boucle de l'interface se mettent automatiquement à jour. Ayant ajoutés de nombreuses fonctionnalités, il nous était impossible de faire tenir toutes les icônes sur une même ligne comme dans les versions précédentes de notre interface. Nous avons donc placé toutes les icônes 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 ».



Interface version finale



Affichage des fonctionnalités




Les fonctionnalités

L'equalizer

Qu'est-ce qu'un equalizer ?

Un égaliseur (equalizer, equaliser ou "EQ" en anglais) ou correcteur de timbre est un appareil ou logiciel de traitement du son. Il permet d'atténuer ou d'accentuer une ou plusieurs bandes de fréquences composant un signal audio.

Une source sonore est composée d'une multitude d'ondes sonores réparties sur un large spectre de fréquences audio. L'être humain adulte perçoit ainsi des ondes sonores de 20 Hz à 20 kHz. Il est possible de distinguer et d'isoler certaines bandes de fréquences afin de leur appliquer un traitement spécifique. En agissant sur une plus ou moins large gamme de fréquences (les graves, les médiums, les aigus), la correction permet d'atténuer ou au contraire d'accentuer le timbre du son. À la différence des correcteurs les plus souvent rencontrés sur les amplificateurs Hifi, autoradios, téléviseurs, etc., les égaliseurs interviennent sur des bandes précises et sont généralement plus performants, en particulier pour renforcer certaines fréquences sans trop générer de nuisances (souffle, bruit de fond, saturation, distorsion, etc.).



Plusieurs types d'égaliseurs sont exploités. Le correcteur le plus simple et le plus répandu exploite 2 ou 3 filtres (grave, médium, aigu) pour traiter sélectivement la bande. Ils sont présents sur les amplificateurs Hifi, les autoradios, les guitares électriques ou encore sur les tables de mixage. Les dispositifs électroniques de correction plus perfectionnés peuvent être assimilés à la gamme des égaliseurs dits « graphiques », adjectif symbolisant l'effet produit lorsqu'on analyse le son. L'impact graphique en bosse ou en creux est révélateur du traitement, chaque bande de fréquence impacte directement la courbe sonore produite.

Choix de la base de travail

Une fois le concept de fonctionnement d'un égaliseur assimilé, nous nous sommes mis à la recherche d'exemples concrets dans le langage de programmation que nous utilisons pour notre projet. Nous sommes finalement tombés sur un site de partage de code japonais. Bien évidemment, Java est un langage universel et nous n'avons donc pas rencontrés de problème pour comprendre le code et tester celui-ci afin de vérifier qu'il correspondait bien à nos attentes. Après des essais en interne, nous avons finalement décidé que le programme était satisfaisant et nous l'avons donc sélectionné comme base de travail.

Fonctionnement

Les principales fonctionnalités de l'égaliseur sont assurées par deux classes de l'API Android.

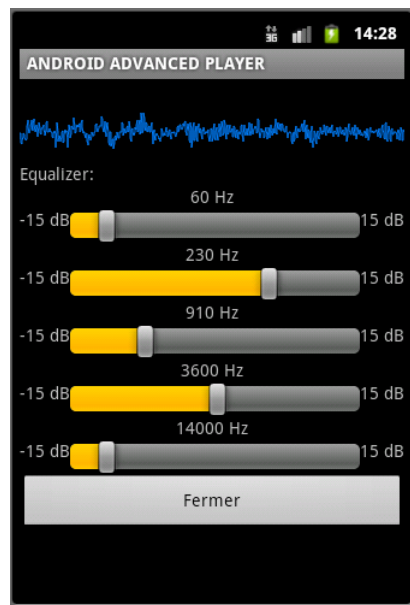
- Le package `AudioEffect` : il permet la manipulation d'effets sonores offerts par le framework ^[22] audio d'Android et propose ainsi plusieurs classes pour y parvenir. Dans le cadre de notre application, on utilisera **`android.media.audiofx.Equalizer`**. Comme son nom le laisse penser, la classe `Equalizer` fournit des méthodes pour agir sur la fréquence d'un signal audio.
Une fois la musique en cours de lecture initialisée avec le signal audio à modifier, on récupère la quantité de bandes de fréquence (plage de fréquence) supportée par le morceau. Pour chacune de ces bandes, on récupère la fréquence centrale et on lui associe une seekbar (un widget ^[17] représentant une barre de modification) variant entre sa valeur minimale et sa valeur maximale en décibels, permettant ainsi de jouer avec le morceau et d'amplifier ou de réduire les aigus ou les graves selon l'envie.
- Le deuxième composant fait ici figure de gadget : il s'agit de la classe **`Visualizer`**. Cette classe permet de récupérer les informations de fréquence d'une bande sonore afin d'y associer un effet visuel simple.

Implémentation

À l'origine, cette application fonctionne de façon indépendante mais dans le cadre de notre projet il a bien évidemment fallu l'inclure et l'associer aux différentes classes existantes. C'est pourquoi, l'activité `Equalizer` n'est lancée que lorsque l'utilisateur le demande par le biais de l'interface. La piste en cours de lecture lui est alors transmise et les modifications apportées par l'utilisateur ont un impact direct sur le morceau en cours d'écoute. Les valeurs de l'`Equalizer` sont ensuite réinitialisées à chaque changement de musique.



Interface de l'Equalizer d'AAP



Time Stretching

Le concept

Le « Time Stretching » est un effet audio numérique qui a pour but de raccourcir ou allonger la durée et/ou le tempo d'un échantillon sonore sans en modifier sa tonalité. C'est le procédé réciproque du « Pitch Shifting » qui consiste à changer la hauteur d'un signal sonore sans en modifier sa longueur.

Fonctionnement

Il existe diverses méthodes pour effectuer du time stretching sur un signal sonore : toutes ces méthodes utilisent des concepts de physique et de manipulation de signal audio extrêmement avancés, c'est pourquoi nous n'allons que brièvement présenter le principe de fonctionnement d'une des méthodes possiblement utilisable : la méthode à base de « Phase Vocoder ».

Phase Vocoder : l'utilisation d'un phase vocoder (vocodeur de phase en français) est une technique complexe faisant appel à des éléments de CPL ^[23]. Elle utilise les transformées de Fourier en continu.

Implémentation

Comme expliqué précédemment, les notions abordées par cette fonctionnalité sont très compliquées et il nous a été impossible d'inclure celle-ci dans notre projet. Le principal problème a été le manque d'exemples concrets disponibles en Java : en effet il ne semble exister aucun code open-source dans ce langage de programmation, les logiciels comme Audacity (logiciel libre pour la manipulation de données) ne sont disponibles qu'en C++.

Il existe néanmoins sur la plate-forme Android une application (Audio Speed Changer) qui inclut cette fonctionnalité. A des fins pédagogiques nous avons donc réalisés de la retro-ingénierie ^[24] sur cette application afin d'en apprendre d'avantage.

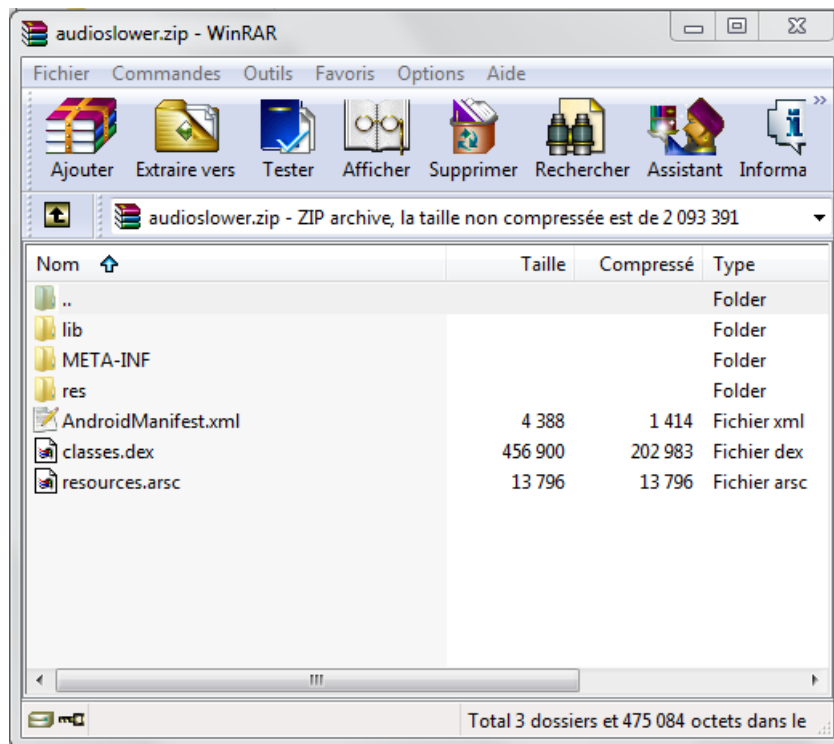
Rétro-ingénierie

Notre but n'était évidemment pas de nuire au développeur et à son application mais bien de comprendre son implémentation.

Notre méthode fut la suivante :

- récupération de la version gratuite de l'application (.apk, format des applications installables Android) sur internet ;
- transformation du fichier .apk en .zip et ouverture de ce dernier : on y trouve tous les fichiers qui composent notre application et en particulier le fichier classes.dex qui représente la version compilée de l'application ;

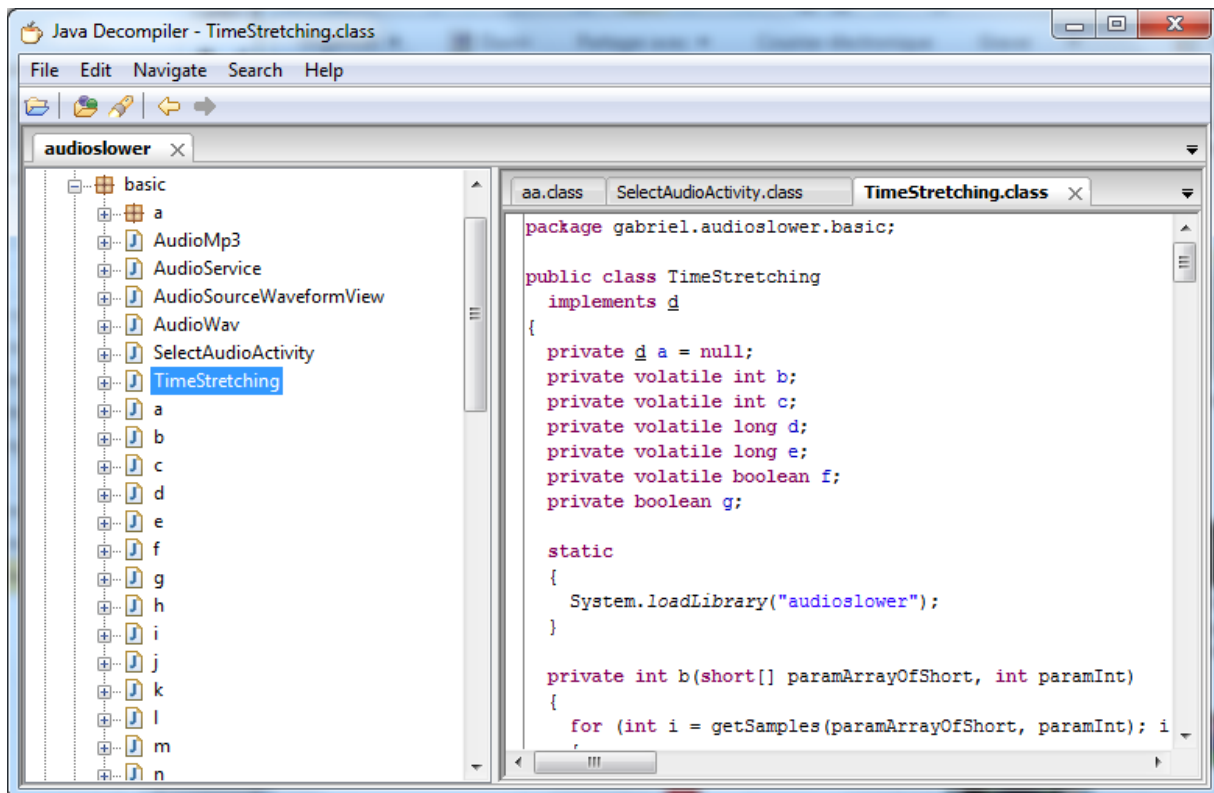
Contenu d'un fichier .apk



- transformation du fichier .dex en .jar grâce au programme dex2jar ;
- le .jar généré contient alors tous les fichiers .class de l'application et il est désormais possible de les lire grâce à un utilitaire tel que JD-GUI qui reconstruit les fichiers .java associés.



Interface une fois le projet ouvert avec JD-GUI



Malheureusement nous nous sommes retrouvés devant un code **obfuscated**, c'est-à-dire qu'il a été rendu illisible à l'œil humain par le développeur lui-même, empêchant ainsi toute analyse.

Après contact avec le développeur, celui-ci nous a conseillé d'abandonner l'idée d'implémenter cette fonction si nos connaissances en manipulation du son n'étaient pas assez importantes, tout en ajoutant que si nous décidions de persister il serait plus aisé de travailler en langage C ou C++, langages que personne dans notre groupe ne maîtrise assez. C'est donc avec regret que nous laissons de côté le time stretching.



Glossaire

- [1] Un téléphone Android est un téléphone dont le système d'exploitation est Android.
- [2] Le système d'exploitation est l'ensemble de programmes central d'un appareil informatique servant d'interface entre le matériel et les logiciels applicatifs.
- [3] Open source est une désignation s'appliquant aux logiciels dont la licence respecte des critères précisément établis par l'Open Source Initiative, organisation dédiée à la promotion du logiciel Open Source, à savoir la possibilité de libre redistribution, d'accès au code source et aux travaux dédiés.
- [4] Appelé aussi téléphone intelligent ou ordiphone, le smartphone est un téléphone mobile disposant aussi des fonctions d'un assistant numérique personnel. Il peut aussi fournir les fonctionnalités d'agenda, de calendrier, de navigation Web, de consultation de courrier électronique, de messagerie instantanée, de GPS, etc.
- [5] Un PDA (Personal Digital Assistant) est un appareil numérique portable servant d'agenda, de carnet d'adresses et de bloc-notes.
- [6] Linux est un logiciel libre créé en 1991 par Linus Torvalds et développé sur Internet par des milliers d'informaticiens bénévoles ou salariés. C'est le noyau de nombreux systèmes d'exploitation.
- [7] Le langage Java est un langage de programmation informatique orienté objet, créé par James Gosling et Patrick Naughton, employés de Sun Microsystems, avec le soutien de Bill Joy, le cofondateur de Sun Microsystems en 1982. Il a été officiellement présenté le 23 Mai 1995 au SunWorld.
- [8] JIT, just-in-time, est une technique de compilation dite compilation à la volée, ou encore traduction dynamique.
- [9] Le C est un langage de programmation impératif conçu pour la programmation système. Inventé au début des années 1970 avec UNIX, un système d'exploitation multitâche et multiutilisateur créé en 1969, C est devenu un des langages les plus utilisés. D'ailleurs, de nombreux langages plus modernes comme C++, Java et PHP, reprennent des aspects du langage C.
- [10] La licence Apache est une licence de logiciel libre et open source. Elle est décrite par l'Apache Software Foundation, une organisation à but non lucratif, qui l'applique à tous les logiciels qu'elle publie. Il existe plusieurs versions de cette licence (1.0, 1.1, 2.0).
- [11] XML, eXtensible Markup Language, est un langage informatique de balisage générique. Cette syntaxe est dite extensible car elle permet de définir différents espaces de noms, c'est-à-dire des langages avec chacun leur vocabulaire et leur grammaire. Cette syntaxe est reconnaissable par son usage de chevrons (< >) encadrant les balises.
- [12] Une API, Application Programming Interface, est une interface fournie par un programme informatique. Elle permet l'interaction des programmes les uns avec les



autres, de manière analogue à une interface homme-machine. Du point de vu technique, une API est un ensemble de fonctions, procédures ou classes, mises à disposition par une bibliothèque logicielle, un système d'exploitation ou un service. La connaissance des API est indispensable à l'interopérabilité entre les composants logiciels.

- [13] La licence BSD, Berkeley Software Distribution, est une licence libre utilisée pour la distribution de logiciels, permettant de récupérer tout ou une partie du logiciel, sans restriction, qu'il soit intégré dans un logiciel libre ou propriétaire.
- [14] Une bibliothèque de programmes est un ensemble de fonctions utilitaires, regroupées et mises à disposition afin de pouvoir être utilisées sans avoir à les réécrire.
- [15] Un plugin, ou plug-in, est un logiciel venant compléter un logiciel hôte dans le but de lui apporter de nouvelles fonctionnalités. Il ne peut généralement pas fonctionner et sont mis au point par des personnes n'ayant pas nécessairement de relation avec les auteurs du logiciel principal.
- [16] Une carte SD (Secure Digital) est une carte mémoire amovible de stockage de données numériques créée en janvier 2000 par une alliance formée par les industriels Panasonic, SanDisk et Toshiba. Elles sont essentiellement utilisées pour le stockage de fichiers dans les appareils photos numériques, les caméscopes numériques, les systèmes de navigation GPS, les consoles de jeux vidéo, les smartphones, etc.
- [17] Un widget, ou composant d'interface graphique, est un élément de base d'une interface graphique avec lequel un utilisateur peut interagir. Ces composants sont généralement regroupés dans des boîtes à outils graphiques (appelés *toolkit* en anglais). Une fois assemblés par un développeur, ces composants forment une interface graphique complète. À noter, l'appellation *control* est connotée Microsoft, l'appellation *widget* étant utilisée dans tous les autres cas.
- [18] La compilation est le travail réalisé par un compilateur. Un compilateur est un programme informatique qui traduit un langage, dit langage source, en un autre langage, appelé langage cible, généralement dans le but de créer un exécutable, un fichier contenant un programme et identifié en tant que tel par le système d'exploitation.
- [19] PNG, Portable Network Graphics, est un format d'images numériques qui a été créé pour remplacer le format GIF.
- [20] La norme JPEG est une norme qui définit le format d'enregistrement et l'algorithme de décodage pour une représentation numérique compressée d'une image fixe.
- [21] Un émulateur est un programme, logiciel, permettant de substituer un élément de matériel informatique, tel qu'un terminal informatique, un ordinateur, une console de jeux vidéo ou encore un smartphone, par un logiciel. La définition du terme « émuler » est « chercher à imiter ».
- [22] En programmation informatique, un framework, ou environnement de travail, est un kit de composants logiciels structurels servant à créer les fondations ainsi que les grandes lignes de tout ou partie d'un logiciel. En programmation orientée objet, un framework est



typiquement composé de classes mères qui seront dérivées et étendues par héritage en fonction des besoins spécifiques à chaque logiciel utilisant le framework.

- [23] Le CPL, Codage Prédicatif Linéaire, est une méthode d'analyse/synthèse développée dans le cadre des techniques d'encodage, de transmission et de réception de la parole.
- [24] La rétro-ingénierie, également appelée rétro-conception, ingénierie inversée ou ingénierie inverse, et reverse engineering en anglais, est l'activité qui consiste à étudier un objet, dans le domaine de la programmation orientée objet, pour en déterminer le fonctionnement interne ou la méthode de fabrication.



Références

Wikipédia

<http://fr.wikipedia.org>

Projet d'encyclopédie collective établie sur Internet, universelle et multilingue.

Android

<http://www.android.com>

Site de la société Android.

Android France

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

Tutoriels sur le développement sous Android.

FrAndroid

<http://www.frandroid.com>

Communauté francophone autour d'Android.

Wiki français d'Android

<http://wiki.frandroid.com>

Wiki du site FrAndroid.

AndroidDevBlog

<http://android.cyrilmottier.com>

Blog francophone sur le développement Android.

Android Developers

<http://developer.android.com>

Site d'aide au développement pour téléphone Android.

Equalizer

<http://www.189works.com/article-9543-1.html>

Site du projet de l'égaliseur utilisé dans AAP.

dex2jar

<http://code.google.com/p/dex2jar>

Repository du logiciel dex2jar, un outil de conversion de fichier .dex en fichier .class.

JD-GUI

<http://java.decompiler.free.fr>

Site du projet « Java decompiler ».

Time Stretching

<http://www.daimi.au.dk/~pmn/sound/timestretch/paper-071002A.doc>

<http://www.engr.uvic.ca/~kzhang/theproject1.html>

<http://my.fit.edu/~vkepuska/ece3551/The%20DSP%20Dimension/Tutorials>

http://en.wikipedia.org/wiki/Audio_timescale-pitch_modification

<http://www.dsppdimension.com/admin/pitch-shifting-using-the-ft>