Projet Android



# Présentation du projet

Age of Empires 2 est un jeu de stratégie en temps réel sortie sur PC en 1999. Dans ce jeu, il est possible de jouer contre des joueurs humains en réseau local, et dans ce type de partie, les joueurs ont alors la possibilité de lancer des sons de provocation en tapant un nombre (l’identifiant du son) dans le chat.

Les sons étants devenus cultes pour les joueurs, j’ai décidé de faire une application reprenant l’ensemble des sons de provocation et dans laquelle il serait possible de jouer d’un simple toucher l’un de ces sons.

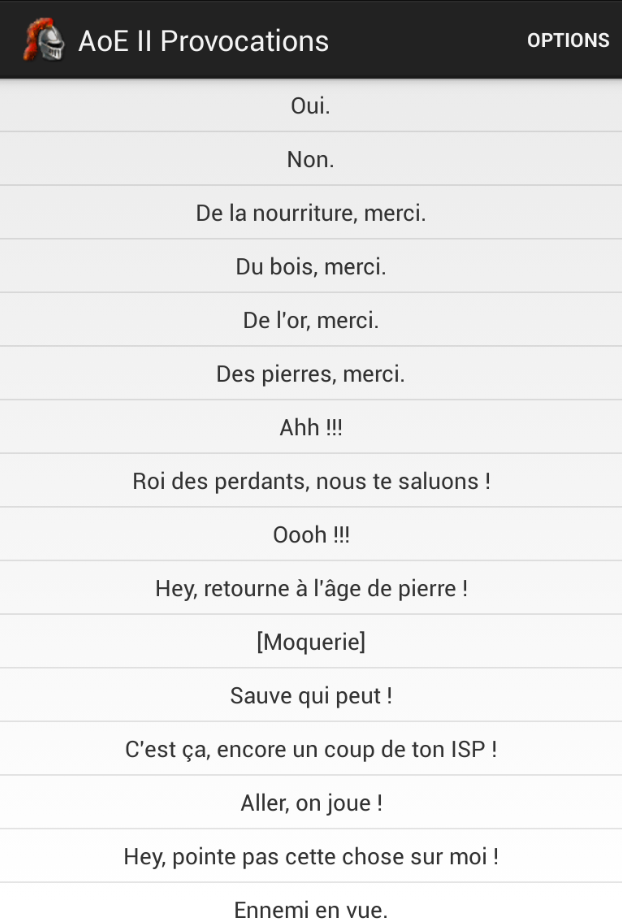
Je me suis également mis en têtre que ce projet pouvait être une manière de tester la publication du Google Play Store, ce que j’ai donc fait. L’application est disponible à cette adresse : <https://play.google.com/store/apps/details?id=com.noxalus.app>.

J’ai aussi rendu le code source disponible publiquement sur Github : <https://github.com/Noxalus/Age-of-Empires-2-Taunts>.

# Fonctionnalités implementées

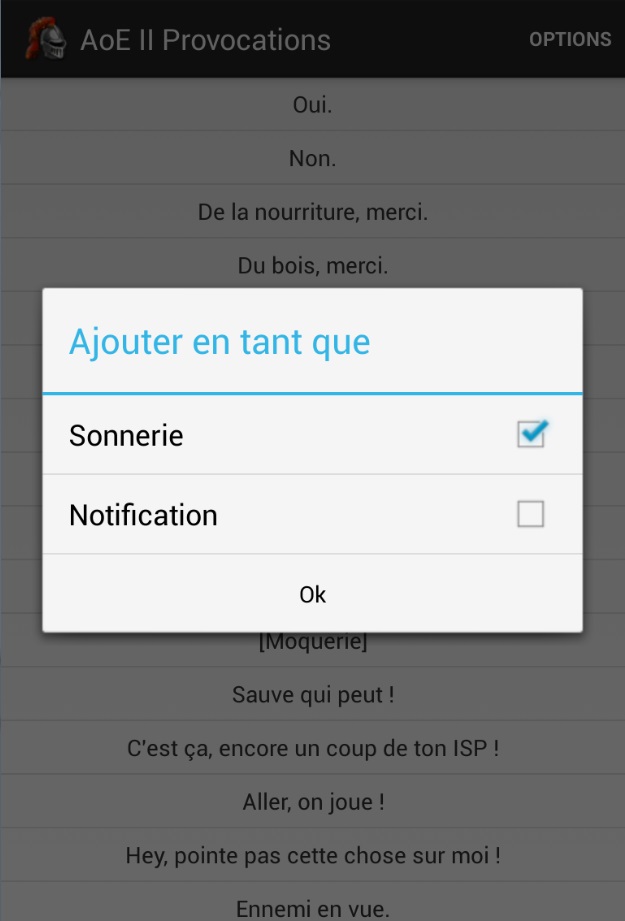
## Lecture de sons

Le premier écran de l’application est une liste de phrases correspondant à des sons issues du jeu Age of Empires 2. Lorsque l’utilisateur clique sur une phrase, le son correspondant est joué.



## Ajout d’un son en sonnerie/notification

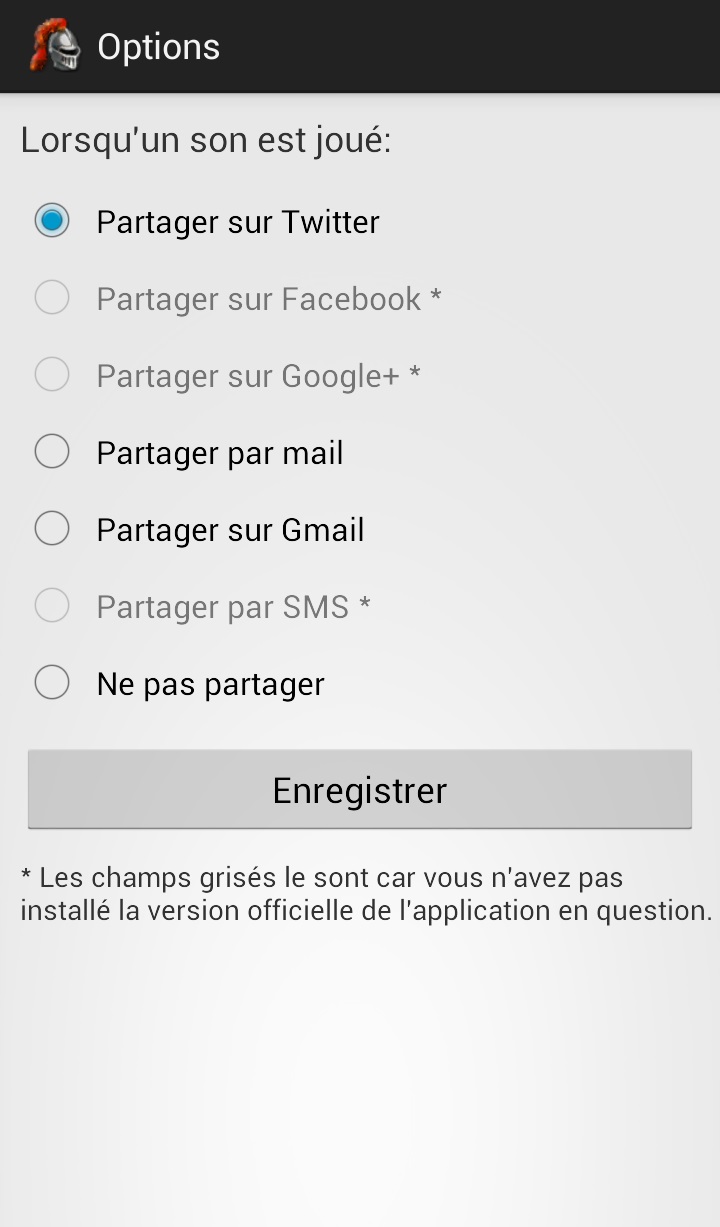
Lors d’un appui long sur une phrase de la liste, une popup (*dialog* en réalité…) s’affiche proposant d’ajouter le son sélectionné en sonnerie ou en notification.



## Partage sur les réseaux sociaux

A chaque fois que vous jouez un son, l’application peut vous proposer de partager le fait que vous avez joué ce son sur les réseaux sociaux. Un écran d’option a d’ailleurs été créé spécialement pour l’occasion auquel vous pouvez accéder en cliquant sur le bouton « OPTIONS » en haut à droite. Vous arriverez alors sur un formulaire vous permettant de choisir sur quel réseau social partager l’information. L’application détecte d’ailleurs les applications officielles installées sur votre appareil et vous empêche de choisir une application qui ne serait pas installées sur votre téléphone. Il est bien évidemment possible de choisir de ne rien partager du tout.

Ce système se sert des préférences partagés et est enregistré au sein de l’application, c’est-à-dire qu’au prochain lancement, votre choix sera toujours le même.



# Problèmes techniques rencontres

## Chargement des sons

La lecture des sons a rapidement été fonctionnelle, tout se fait à base d’instance de **MediaPlayer**. L’idée de base était de faire une liste d’instance de MedialPlayer contenant tous les sons (42 en tout) au chargement de l’application. Au départ je pensais que l’application aurait un peu de mal à tout charger d’un coup, même s’il n’y a pas beaucoup de sons et qu’ils ne pèsent pas lourd, mais il s’est avéré que ça ne fut pas du tout le cas.

Malheureusement, je me suis rendu compte d’un autre problème, une fois 30 sons différents joués, il fut impossible d’en jouer d’autres, comme si le fait d’instancier des MediaPlayer ne suffisait pas à réellement charger les sons en mémoire et qu’Android ne permet pas le chargement de plus de 30 sons simultanément.

Pour résoudre ce problème, il a fallu faire ce que je pensais faire au moment de l’optimisation, c’est-à-dire de ne créer un objet MediaPlayer qu’au moment où l’utilisateur demande de jouer le son et de le libérer après.

## Changement de sonnerie/notification

Tous les sons jouables dans l’application se situent dans le dossier « raw », ils sont donc fatalement sur votre téléphone si l’application y est installée. J’ai donc assez vite trouvé qu’il suffisait d’utiliser la méthode *setActualDefaultRingtoneUri* du **RingtoneManager** en lui passant l’URI du son pour en faire une sonnerie (ou une notification, il suffisait de changer le paramètre qui va bien).

Malheureusement, on se rend assez vite compte que si l’URI donnée en paramètre pointe sur un fichier contenu dans votre dossier « raw », Android ne sera pas capable d’en faire une sonnerie/notification.

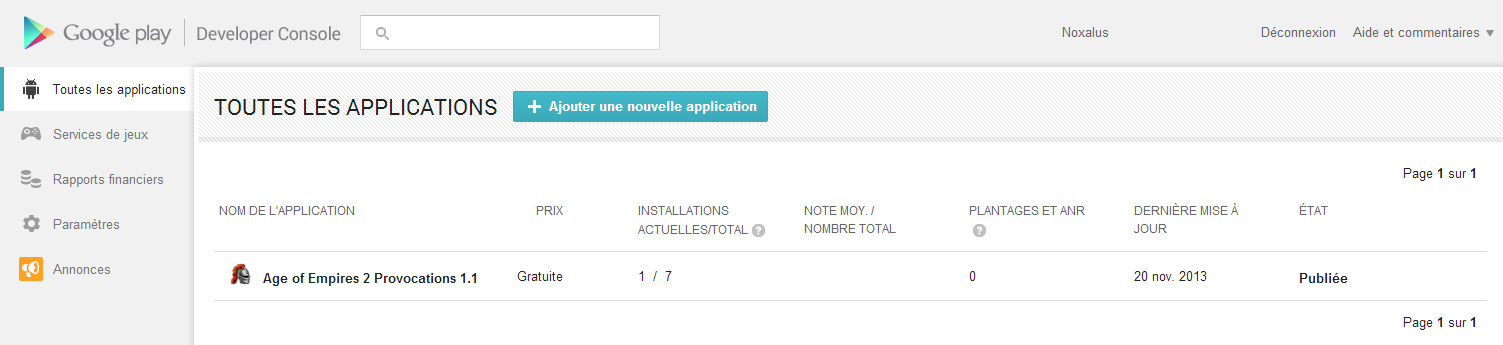
Après quelques recherches sur internet, il s’avère qu’il est obligatoire de copier le fichier son à un autre emplacement de la mémoire du téléphone pour qu’il puisse être affecté en tant que sonnerie/notification. Ce qui, en plus d’être assez laborieux (on doit tout faire à la main avec des FileInputStream/FileOutputStream, je n’ai pas trouvé de méthode magique « FileManager.CopyFile(File file, URI dest) »), nous oblige à rajouter des droits supplémentaire à notre application pour avoir accès en écriture à la mémoire interne du téléphone. Ce qui est vraiment ridicule et peut faire peur aux utilisateurs. Vu les fonctionnalités de l’application, on est en droit de se demander pour quelle raison cette application voudrait écrire dans la mémoire ?!

# Déploiement sur Google Play

Ce projet étant pour moi un réel moyen de tester le développement d’une application Android de A à Z, je me suis dit que l’étape du déploiement sur Google Play était un passage nécessaire et j’ai donc voulu voir comment cela se déroule.

Pour commencer on a, bien évidemment, besoin d’un compte Google+ et il faut s’affranchir des 25$ (18.60€) permettant d’être considéré comme développeurs Android par Google et être à même de publier des applications sur Google Play.

Une fois les frais d’inscription payés, on a accès à ce qu’ils appellent la **Google Play Developer Console**:

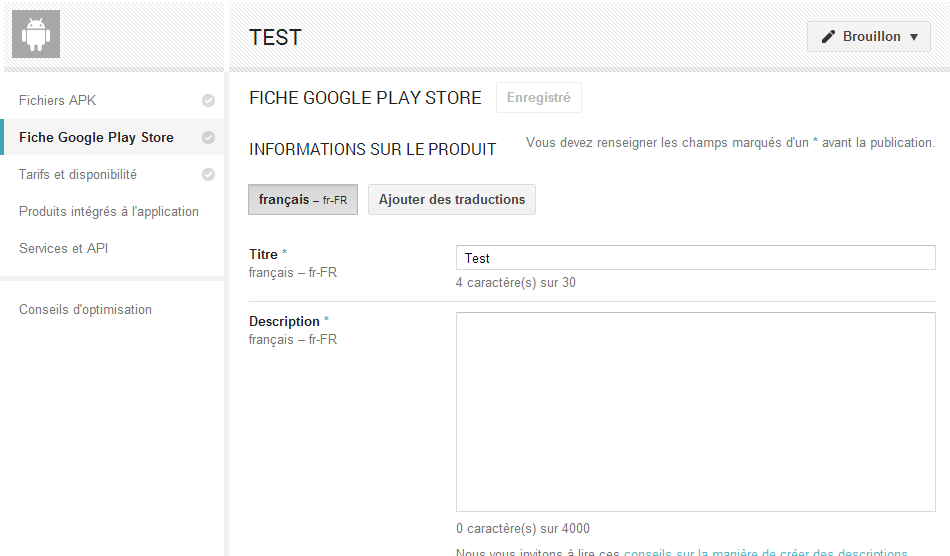


C’est à partir de là qu’il est possible d’ajouter de nouvelles applications, mais c’est aussi là que vous aurez toutes les informations concernant les applications déjà publiées sur le Google Play Store.

A l’ajout d’une nouvelle application, vous aurez à rentrer tout un tas d’informations allant de sa description à sa classification en passant par la possibilité d’ajouté des screenshots de l’application différents en fonction du support (téléphone, tablette 7 pouces et tablette 10 pouces). A noter qu’une icône haute résolution (512x512) est obligatoire pour publier l’application.

Ce qui est intéressant, c’est que la gestion des langues est très bien intégrée et qu’il vous est possible de définir des valeurs différentes pour chaque champ en fonction de la langue choisie.

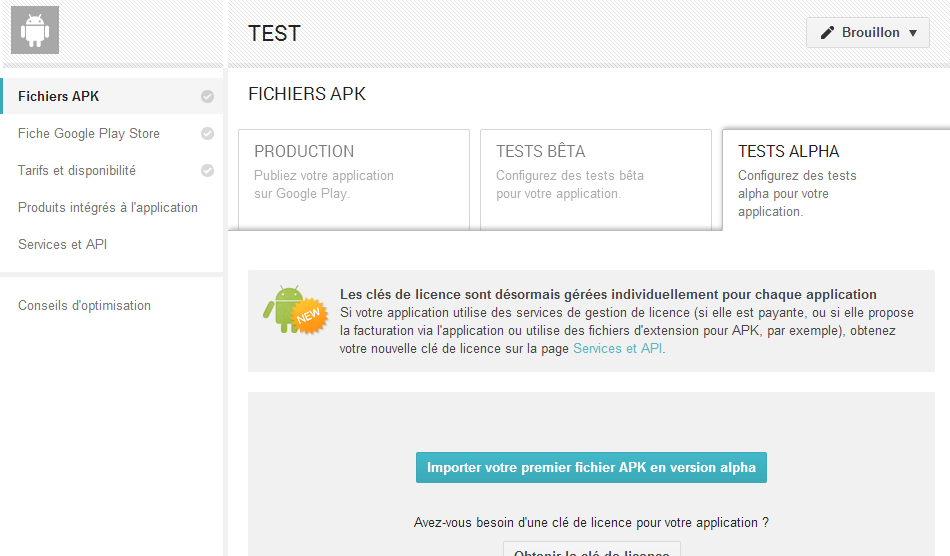
N’ayant pas de compte marchand, je n’ai pas pu tester la gestion des applications payantes, mais il y a toute une partie dédiée à cela, de même pour les services et API.



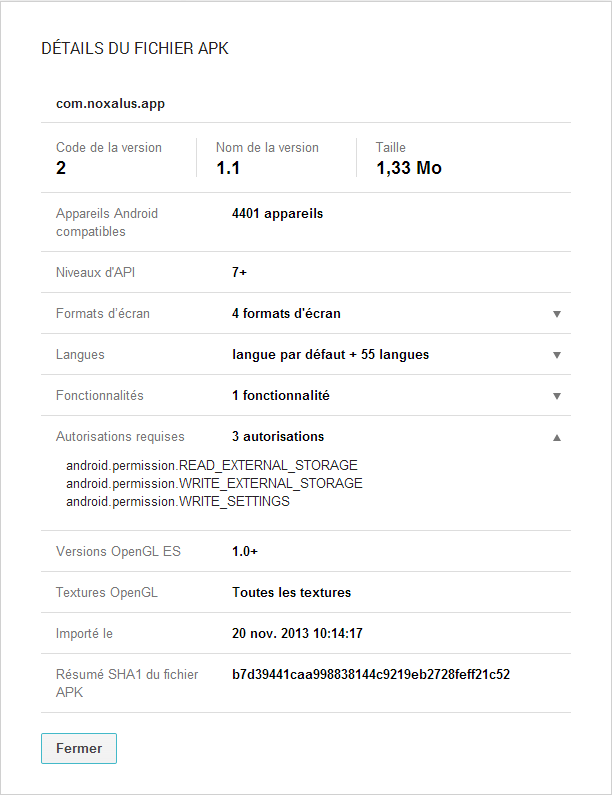
L’étape d’après (ou d’avant en fonction de ce que vous avez décidé), c’est d’envoyer le fichier APK qui est le binaire de votre application. Il est important de savoir que vous devez obligatoirement généré un APK signé pour qu’il puisse être envoyé sur Google Play. J'ai été rejeté plusieurs fois car il me disait que mon APK pouvait être débuggé et qu’il ne fallait pas. Il a fallu que je rajoute cette ligne dans mon **AndroidManifest.xml** pour qu’il accepte:

android:debuggable="false"

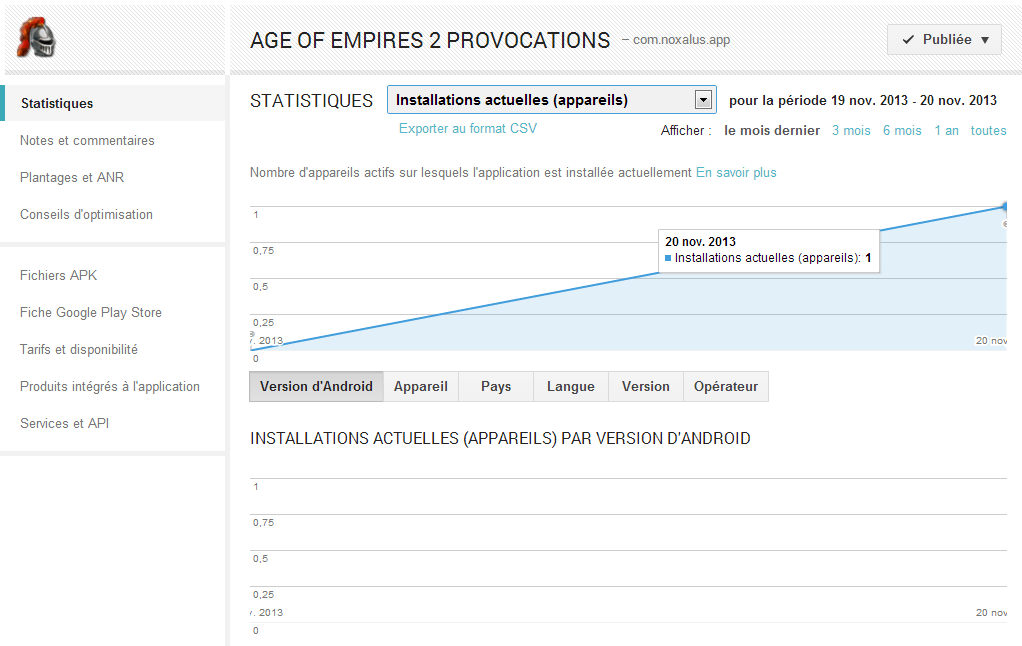
Il est possible de ne pas publier directement votre application une fois votre APK envoyé. En réalité, il existe 3 étapes : alpha, beta et production. Donc si vous souhaitez tester que tout fonctionne bien avant de la rendre visible pour vos potentiels utilisateurs, il est conseillé de passer par l’étape alpha puis beta avant de proposer votre APK en production.



Une fois votre application publiée et visible sur le Google Play Store, vous avez alors accès à des informations sur votre fichier APK comme les autorisations requises :



ou à des statistiques détaillés comme le nombre d’installation actuelles :



Mais également, et là c’est un peu plus intéressant, à des conseils d’optimisation:

