Utiliser Intel® HAXM pour développer des applications Android* Wear et TV

Par Ashok Emani

Date de publication : 9 décembre 2015

I - Introduction

Android* a fait un long chemin en commençant d'abord par les téléphones, puis les tablettes, Google TV*, Android Wear*, Android TV* (remplaçant Google TV), et Android Auto*. Pour les développeurs, compiler et tester leurs applications sur tous ces types d'appareils peut devenir un véritable défi. Ajoutez à cela les différents facteurs de forme et de résolutions d'images, et les opérations de vérification et de test des applications deviennent rapidement un problème. Nous avons Intel® HAXM à la rescousse.

Intel® Hardware Accelerated Execution Manager (HAXM) est un émulateur Android assisté par matériel avec faible charge, excellentes performances, et une latence faible. Vous pouvez en apprendre plus à son propos **ici**.

Avec Intel HAXM, les développeurs peuvent avoir plusieurs instances d'émulation d'Android tournant sur leur système de développement sans avoir à trop s'inquiéter des problèmes de performances, de charge ou de latence. Cela peut s'avérer une aide précieuse dans le processus itératif qu'est le développement et le test d'applications, permettant une productivité conséquente de développement.

Les images d'émulateur Android qui ne sont pas x86 ont un temps de démarrage important, et une réactivité faible de l'interface. Contrairement aux émulateurs Android tiers, avec Intel HAXM vous pouvez utiliser les dernières versions des API et plateformes Android dès qu'elles sont publiées.

Pour des instructions détaillées concernant l'utilisation d'Intel HAXM, veuillez consulter ce **site**.

Dans cet article, nous allons voir comment les développeurs peuvent tirer avantage de l'émulateur Intel HAXM lors du développement d'une application Android universelle qui cible plusieurs plateformes Android telles qu'Android Wear et TV, et les variations des appareils.

II - Utiliser l'application universelle d'exemple Android

Google a récemment publié une application universelle d'exemple pour montrer comment les développeurs peuvent cibler plusieurs facteurs de forme avec la même base de code. Veuillez suivre le lien suivant pour en apprendre plus : sur **github**.

Cette application d'exemple montre quelques-unes des meilleures méthodes pour cibler plusieurs facteurs de forme avec la même base de code. Suivez les instructions du lien ci-dessus pour compiler l'application. Nous l'utiliserons pour charger des instances d'émulation x86 HAXM pour les versions TV, Wear et téléphone dans cet article.

Le projet peut être directement importé dans Android Studio* et les développeurs peuvent profiter des fonctionnalités d'émulation intégrées. Si vous préférez utiliser d'autres environnements de développement, ce qui suit peut vous être utile.

Si vous êtes à l'aise avec la ligne de commande, importez simplement le script de build pour gradle situé dans le répertoire de l'exemple.

```
1. gradlew assembleDebug
```

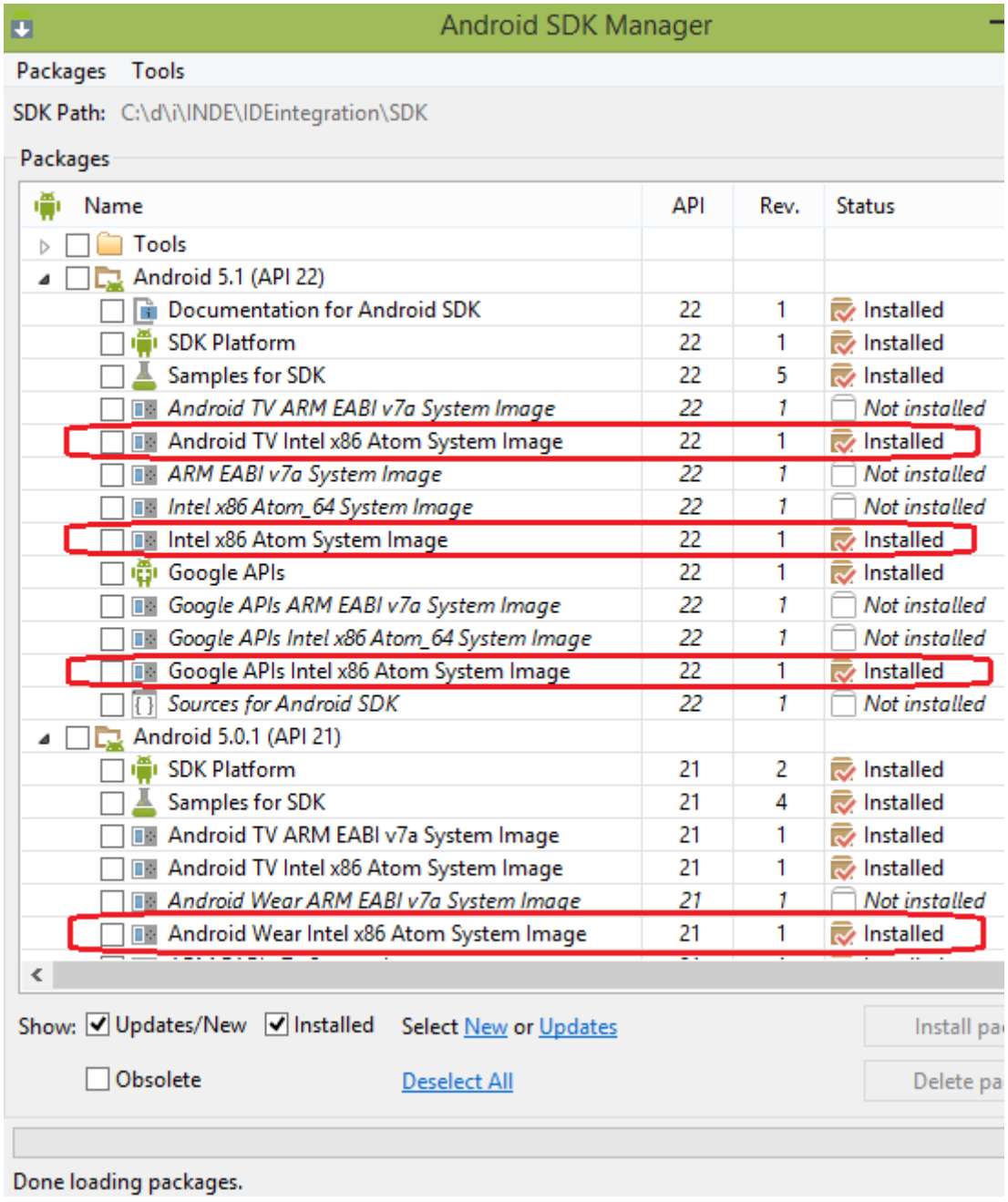
L'application .apk sera disponible dans le répertoire « *mobile/build/outputs/apk/mobile-debug.apk* ».

III - Créez les AVD pour Android TV et Wear

Nous devons nous assurer que nous avons téléchargé les dernières images d'émulation du SDK Android pour TV et Wear, ainsi que l'image standard d'Android pour téléphone et tablette.

Ouvrez le gestionnaire de SDK Android. Vous pouvez le faire depuis la ligne de commande (<Android-SDK>/tools - le dossier devrait s'y trouver) :

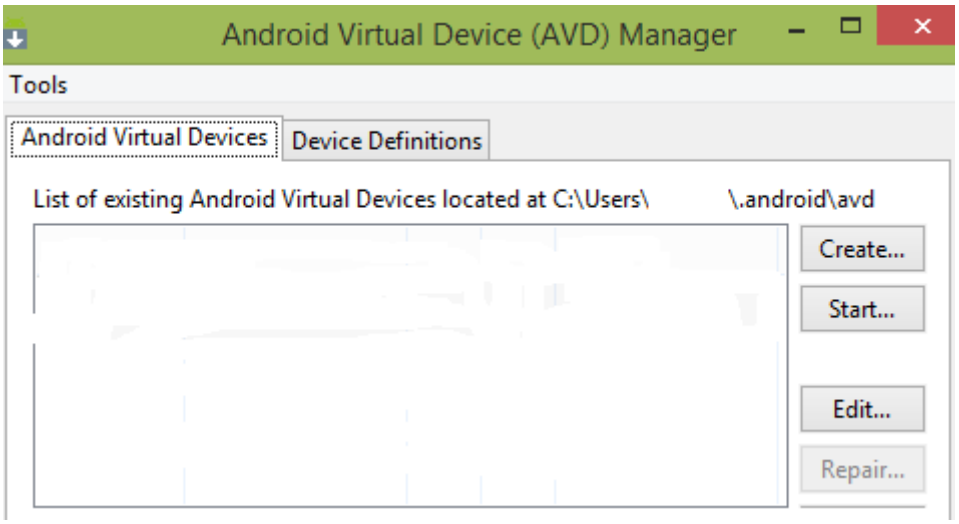
```
> android
```



Ensuite, nous devons créer les configurations de l'émulateur (AVD) pour utiliser les images ci-dessus.

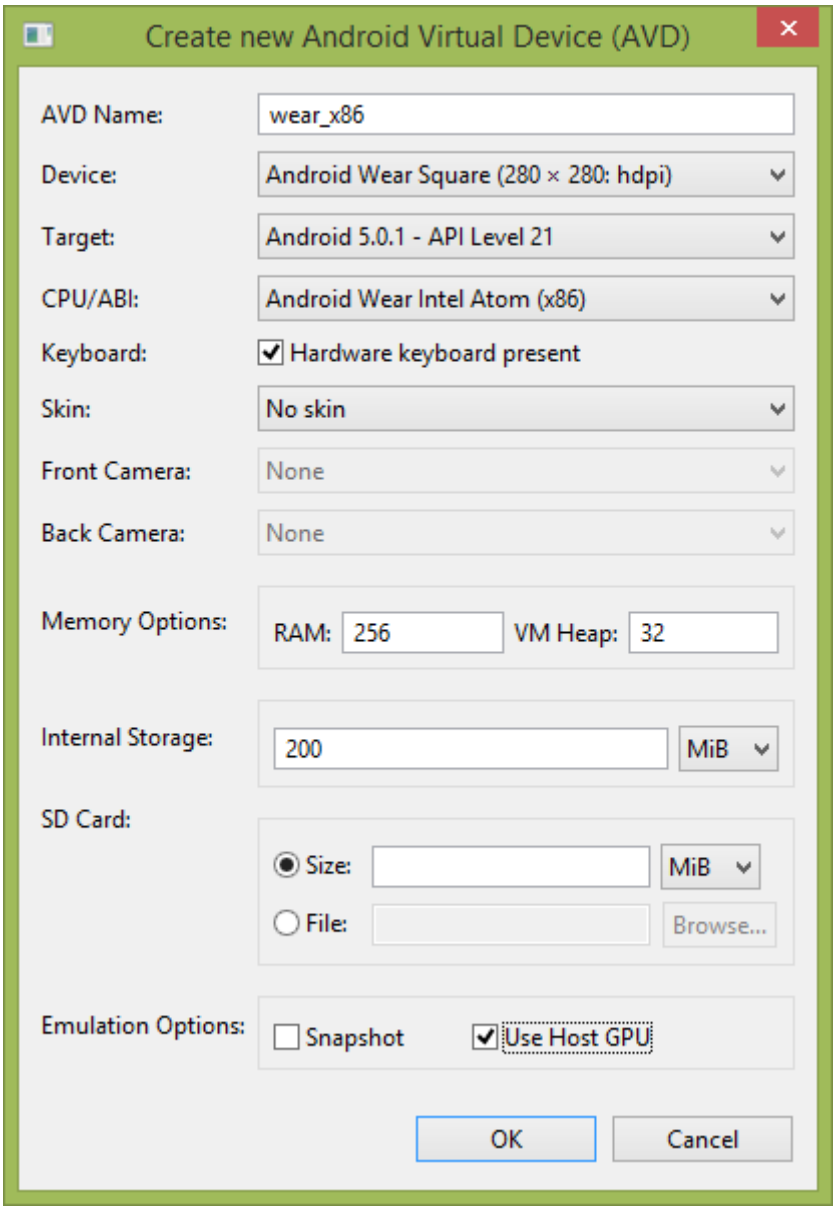
Ouvrez le gestionnaire d'appareils virtuels Android Virtual Device manager. Vous pouvez le faire avec la ligne de commande :

```
> android avd
```

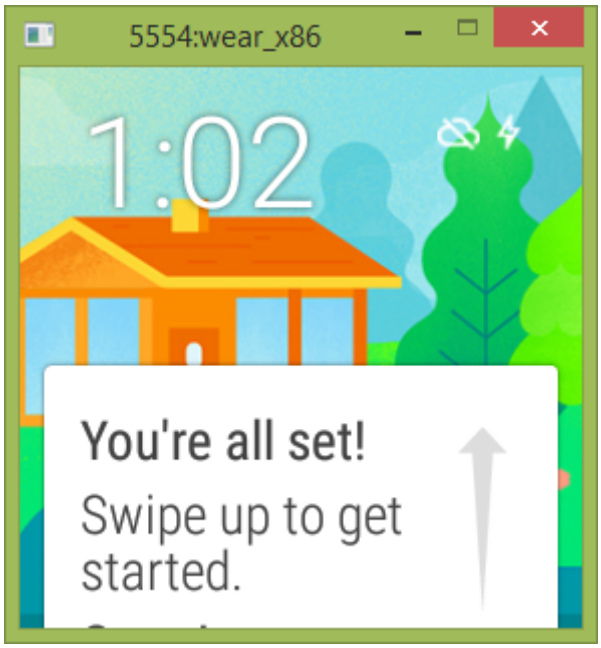


IV - Émulation Android Wear

Créez une configuration AVD pour Android Wear comme indiqué.



Cliquez sur « OK », et lancez l'émulation Wear en cliquant sur « Start… » dans la fenêtre du gestionnaire AVD Manager. Une capture d'écran de l'émulateur Wear après son premier démarrage est montrée ci-dessous :



Android Wear nécessite qu'une application compagnon soit installée sur votre téléphone. L'application est uniquement disponible sur Google Play, donc cela nous oblige à avoir un appareil ayant accès à Google Play : **voir ici**.

Il faut configurer l'appareil avec `adb debugging`, et à la fois l'émulateur Wear et l'appareil apparaissent dans la liste d'appareils adb :

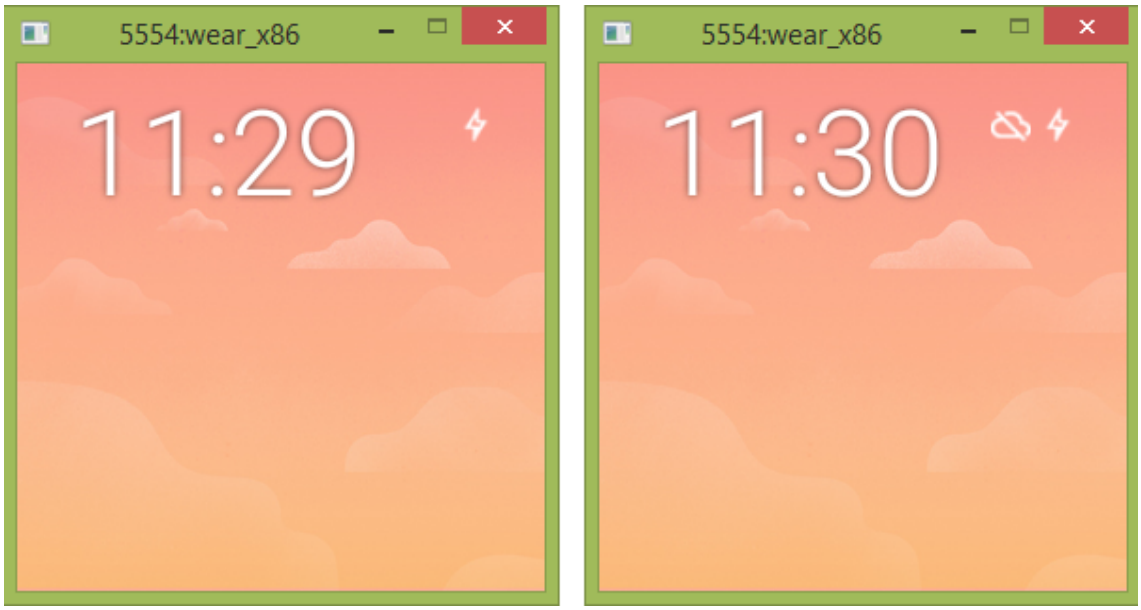
```
D:\android-UniversalMusicPlayer>adb devices
List of devices attached
Z[redacted]       device
emulator-5554   device


D:\android-UniversalMusicPlayer>adb -d forward tcp:5601 tcp:5601
```

Enfin, nous devons rediriger les ports TCP avec :

```
> adb -d forward tcp:5601 tcp:5601
```

L'émulateur Wear devrait être capable de se connecter à votre périphérique. Les copies d'écran ci-dessous montrent les états « connecté » et « déconnecté » de l'émulateur Wear.



Les instructions détaillées pour créer des apps Wear peuvent être consultées à cette **adresse**.

Comme pour toutes les apk android, vous pouvez installer manuellement votre exemple d'apk sur l'émulateur Wear en utilisant l'adb.

```
> adb -s emulator-5554 install -r mobile\build\outputs\apk\mobile-debug.apk
```

Vérifier que l'apk est bien installé sur l'émulateur Wear en utilisant :

```
> adb -s emulator-5554 shell pm list packages | grep example
```

```
D:\android-UniversalMusicPlayer>adb -s emulator-5554 install -r mobile\build\outputs\apk\mobile-debug.apk
1003 KB/s (2825274 bytes in 2.749s)
        pkg: /data/local/tmp/mobile-debug.apk
Success

D:\android-UniversalMusicPlayer>adb -s emulator-5554 shell pm list packages|grep example
package:com.example.android.wearable.photoswatchface
package:com.example.android.uamp
```

Le nom du package pour notre exemple d'app, `com.example.android.uamp` figure bien dans la liste.

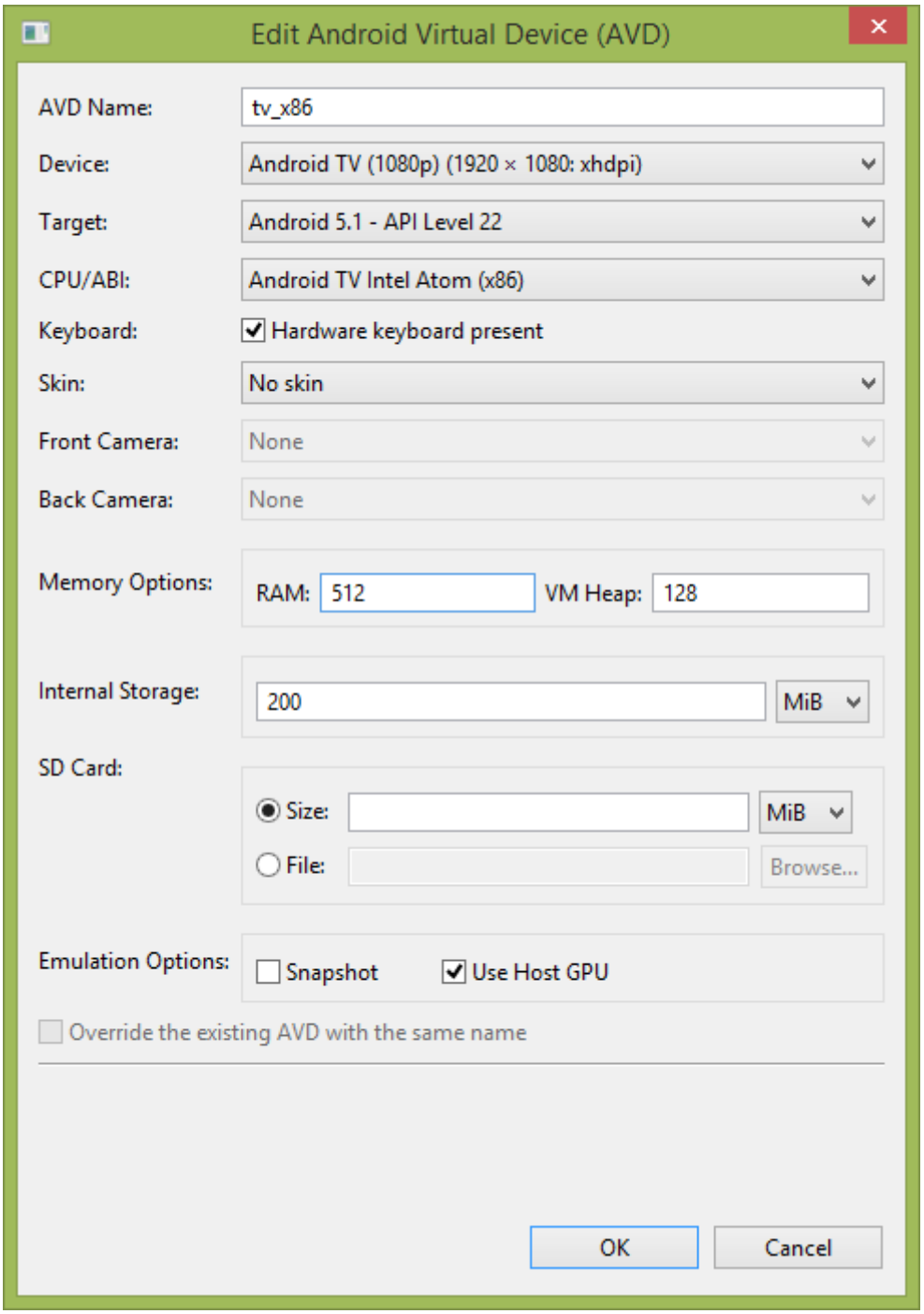
Nous pouvons manuellement lancer notre exemple d'app sur l'émulateur Wear en utilisant :

```
> adb -s emulator-5554 shell monkey -p com.example.android.uamp -c android.intent.category.LAUNCHER 1
```

L'exemple d'app fonctionne désormais sur le périphérique de l'émulateur Wear.

V - Emulation Android TV

Créer une configuration via l'émulateur de l'Android TV (AVD) comme décrit ci-dessous.



Cliquez sur « OK », et démarrez l'émulateur TV en cliquant sur « Start… » dans la fenêtre de gestion de l'AVD.

Nous pouvons vérifier que l'émulateur est accessible de l'adb en tapant :

```
> adb devices
```

Notez l'id de l'émulateur(eg : emulator-55xx) que vous pouvez utiliser comme cible des commandes de l'adb.

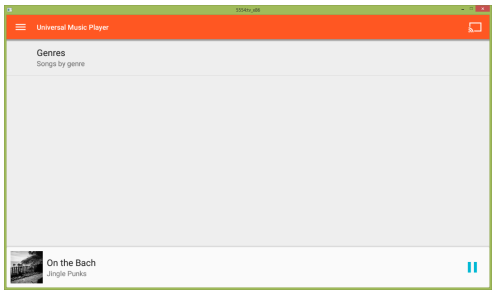
Installez l'apk en utilisant :

```
> adb -s emulator-55xx install -r mobile\build\outputs\apk\mobile-debug.apk
```

Enfin, lancez l'application sur l'instance de l'émulateur Android TV grâce à :

```
> adb -s emulator-55xx shell monkey -p com.example.android.uamp -c android.intent.category.LAUNCHER 1
```

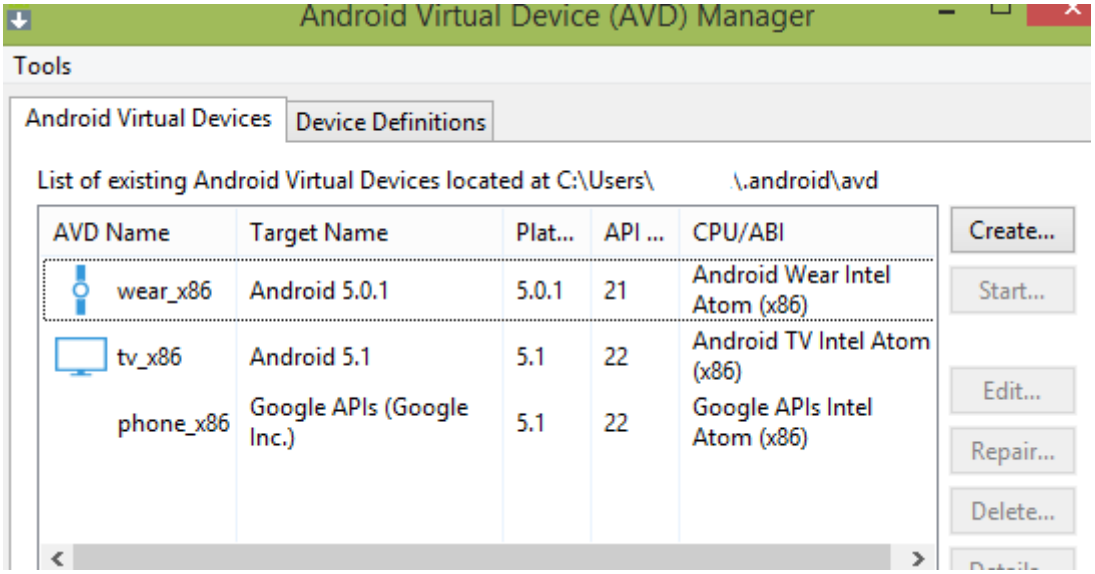
L'exemple d'app fonctionnant sur l'instance de l'émulateur Android TV:



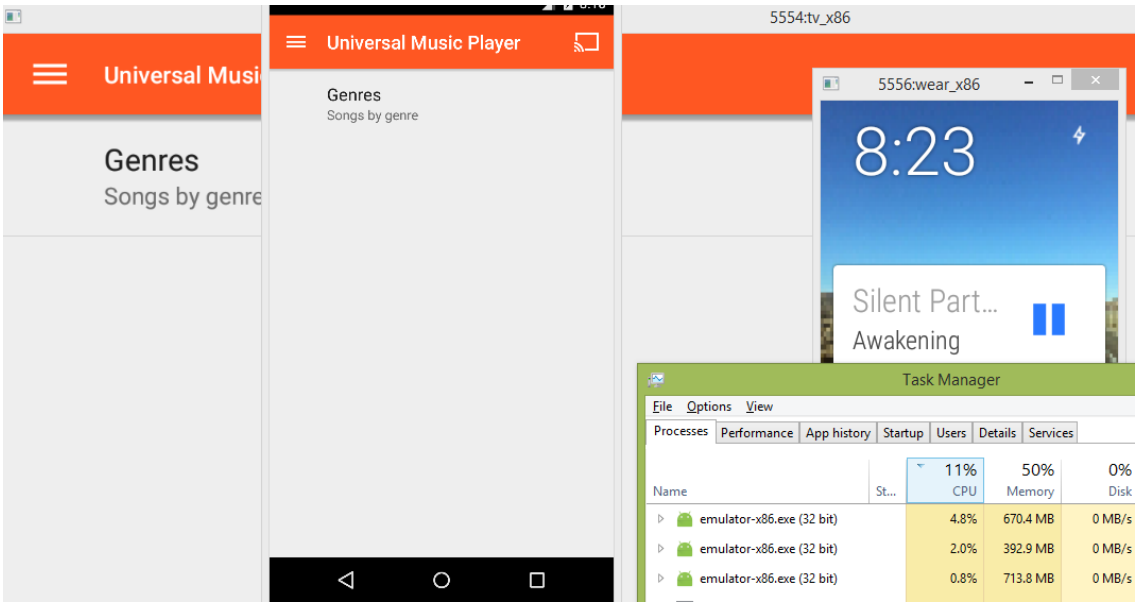
Les développeurs peuvent créer et lancer autant de configurations/instances sur l'émulateur que nécessaire.

La taille de la mémoire appropriée peut être configurée lors de l'installation d'Intel HAXM.

La copie d'écran ci-dessous montre les configurations pour les Wear, TV et téléphone :



Vous pouvez voir que l'exemple d'app universel est lancé sur les trois périphériques (TV, Téléphone et Wear) et voir aussi l'utilisation CPU (notez la faible utilisation CPU) :



Les développeurs peuvent modifier l'allocation mémoire pour plus d'optimisation.

Nous avons à peine effleuré la surface des possibilités liées à l'émulation dans cet article. Merci de vous rendre à cette **adresse** afin de découvrir toutes les options de configuration disponibles.

VI - Références

- Assurez-vous qu'HAXM soit installé correctement en suivant les consignes fournies **ici**

*Tous les noms et les marques cités peuvent être la propriété d'autrui.

Retrouvez toutes les ressources et tous les outils Intel pour les développeurs Android sur la **Zone des Développeurs Intel Android**.

Le **forum Intel Android** pourra également vous aider dans vos questions.

VII - Ressources

- **Outils Intel Android**
- **Intel XDK**
- **Intel INDE**
- **Intel C Compiler**