



Android TP 4 : Un mini-projet

Pour renforcer les compétences acquises jusque-là, nous allons, dans ce TP, développer un mini projet. Celui-ci concerne la gestion de fichiers son.

L'application sera développée sur une page unique (pas de menu), mais elle devra tenir dans un fragment.

1 Mini projet : un peu plus loin avec le son

1.1 1^{ère} version

L'objectif de cet atelier est de lire un fichier son en utilisant un « MediaPlayer ». Un bouton PLAY lance le morceau de musique, un bouton PAUSE l'arrête, un bouton STOP le réinitialise.

Pour rendre les boutons plus "standards" on utilisera des boutons sous forme d'icônes disponibles dans les ressources sur COMMUN. Il faut dans ce cas utiliser un « ImageButton », déposer les icônes dans le dossier « mipmap » (Figures 1, 2, 3)

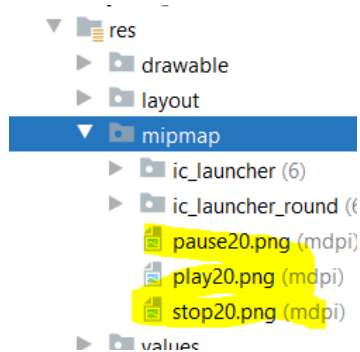


Figure 1 - rangement des icônes

```
<ImageButton  
    android:id="@+id/boutonPlay"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:src="@mipmap/play20"
```

Figure 2 - Utilisation d'un "ImageButton"

On doit obtenir ceci :



Figure 3 - Interface attendue

→ Lire la documentation de référence sur la classe « MediaPlayer » située à cette adresse :

<https://developer.android.com/reference/android/media/MediaPlayer.html>

Etudier notamment la machine à états décrivant le cycle de vie des fichiers media (son et vidéos).

→ Rédiger quelques mots montrant que l'on a compris l'état du media dans les états « préparé », « démarré », « arrêté », « en pause », « fonctionnement terminé »

→ Lire aussi le guide sur la classe « MediaPlayer » situé à l'adresse ci-dessous. On y trouve les outils logiciels pour programmer le « MediaPlayer »

<https://developer.android.com/guide/topics/media/mediaplayer>

Un fichier mp3 est aussi disponible sur COMMUN. Il est bien entendu, possible d'en utiliser d'autres...

Développer le projet. Utiliser la fonction « log » pour faire la mise au point de l'application. Procéder pas à pas.

→ L'émission de son continue-t-elle si on lance une autre application (sans fermer la nôtre évidemment !) ?



→ Lire un fichier mp3 avec le lecteur du téléphone. Répondre à la même question.

1.2 Amélioration

L'utilisateur doit pouvoir maintenant régler le volume sonore avec une barre de progression.

2 Pour aller encore plus loin

Cet autre atelier vise à atteindre des objectifs plus avancés avec le son. **On développera un autre projet.**

1- L'utilisateur pourra

- Régler le volume sonore avec une barre de progression
- Choisir le morceau dans une liste de 6






Soit par une liste déroulante : on peut utiliser le widget « spinner ». Voici par exemple un tutoriel pour le mettre en œuvre : https://www.youtube.com/watch?v=Xi3rdxIVJ_E

Soit par une liste fixe qui s'affiche sur l'écran (ceci correspond davantage aux pratiques des fournisseurs de musique en ligne) : voici un tutoriel pour le mettre en œuvre :

<https://www.youtube.com/watch?v=QtqyuKsPhoQ>

Le programme affichera le volume sonore en %

2- Le logo de l'IUT de Cachan s'affichera en haut à gauche de l'interface utilisateur, sur $\frac{1}{4}$ de la largeur pour un écran vertical. Sa taille devra s'adapter à celle de l'écran de la cible utilisée. Il faudra donc fournir plusieurs images de taille différente, rangées dans le dossier « drawable ». En pratique, la taille des écrans des téléphones, tablettes est toujours liée à leur résolution. C'est pour cela qu'on ne tient compte que de la résolution alors que taille et résolution sont des concepts différents (Figure 1)

| Type | Name | Play Store | Resolution |
|---|-------------------------|---|---------------------|
|  | 7 WSVGA (Tablet) API 25 | | 600 × 1024: mdpi |
|  | Mon_Telephone API 25 | | 1440 × 3200: xxhdpi |
|  | Nexus S API 25 | | 480 × 800: hdpi |
|  | Pixel 2 API 26 |  | 1080 × 1920: 420dpi |


| Verify Configuration | |
|---|----------------------|
| AVD Name | Pixel 2 API 26 |
|  Pixel 2 | 5.0 1080x1920 xxhdpi |

Figure 4 - Quelques exemples

Le fichier « de base » fourni pour cet atelier correspond à la résolution approximative 480dpi (nommée xxhdpi). Le logo occupe environ $\frac{1}{4}$ de la largeur de l'écran vertical d'un téléphone 13cm x 6,5cm/2280x1080/xxhdpi)

Il faudra dans un 1^{er} temps générer les fichiers images correspondants aux résolutions standards (Figure 5). Le logiciel « Paint » avec l'outil « Redimensionner » convient très bien pour générer les différents fichiers.

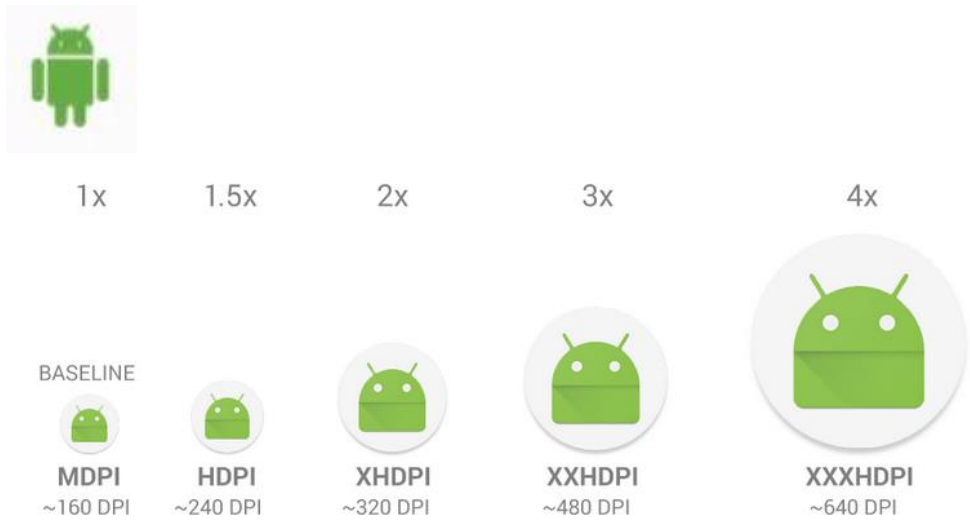


Figure 5 - Formats des images

Il faut ensuite ranger ces fichiers dans 5 dossiers différents car ils doivent porter le même nom.

udiants > Projets_ANDROID > Images_et_Sons > logoiut_tous_les_formats

| <input type="checkbox"/> Nom | Modifié le | Type | Taille |
|------------------------------|------------------|---------------------|--------|
| hdpi | 04/03/2021 05:50 | Dossier de fichiers | |
| mdpi | 04/03/2021 05:50 | Dossier de fichiers | |
| xhdpi | 04/03/2021 05:53 | Dossier de fichiers | |
| xxhdpi | 04/03/2021 05:49 | Dossier de fichiers | |
| xxxhdpi | 04/03/2021 05:52 | Dossier de fichiers | |

Figure 6 - Tous les fichiers

On va ensuite dans le « Ressources Manager » et on ajoute chaque fichier dans le dossier « Drawable », (« Import Drawable ») ANDROID STUDIO reconnaît automatiquement le format (Figure 7)

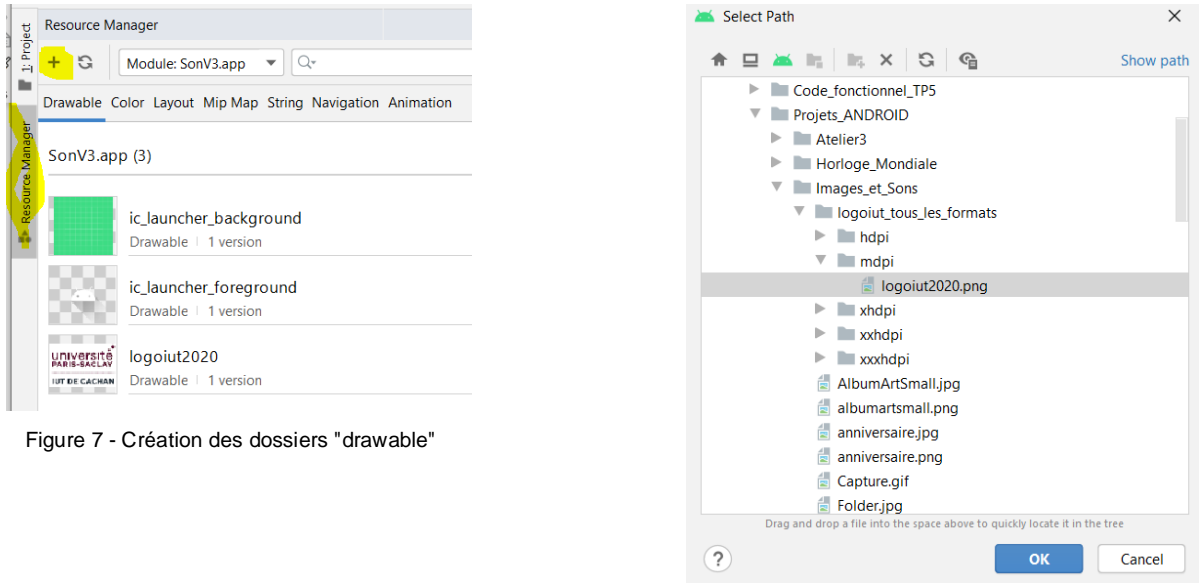


Figure 7 - Création des dossiers "drawable"

On obtient ceci après la création des dossiers (Figure 8) :

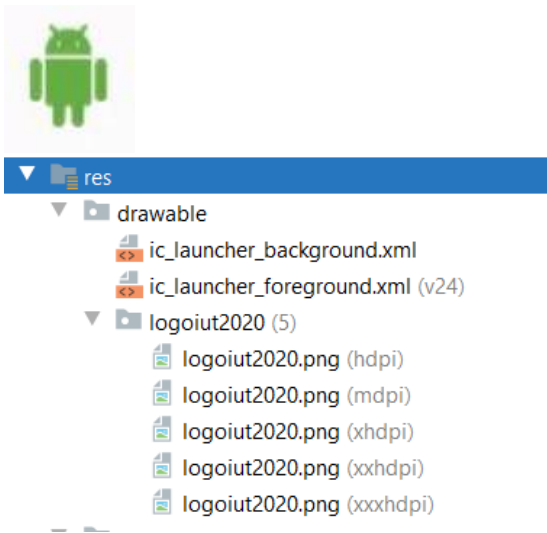


Figure 8 - Résultat de la création des dossiers "drawable"

On utilisera le Widget « ImageView » .

Il est recommandé de tester l'application avec 4 modèles de cibles. On utilise dans ce cas le simulateur (AVD Manager).