

Android Application

Manon Philippot / Ny Andry Raharison / Alexandre Casanova

March 2017

1 Introduction

L'application est basée sur deux grandes activités : MainActivity et SecondActivity. Le MainActivity représente l'écran d'accueil de l'application, sur lequel l'utilisateur peut choisir entre prendre une photo avec la caméra et sélectionner une image dans la gallery. Le SecondActivity correspond au lancement des applications de chargement d'image (caméra et gallery) ainsi qu'à l'activité de traitement de l'image sélectionnée.

2 Besoins fonctionels

2.1 Charger une image

MainActivity lance le processus de chargement de l'image. Il est composée de deux ImageView : le premier qui affiche une icône représentant un appareil photo et qui symbolise l'accès à la caméra, et le second qui affiche une icône représentant un paysage et qui symbolise l'accès à la gallery. Chaque ImageView est attaché à un `setOnClickListener`, qui lancent tous deux le SecondActivity. Ce lancement est accompagné de l'envoi d'un integer à la SecondActivity, symbolisant le choix de l'utilisateur. Si l'utilisateur clique sur l'icône "caméra", l'integer envoyé sera 0, et si l'utilisateur clique sur l'icône "gallery", l'integer envoyé sera 1.

SecondActivity lance la fonction *selectImage* qui s'occupe d'interpréter l'integer reçu. Si elle reçoit 0, alors elle fait appel à la fonction *cameraIntent* qui s'occupe de lancer l'activité correspondant à la caméra, si elle reçoit 1, alors elle fait appel à la fonction *galleryIntent* qui s'occupe de lancer l'activité correspondant au choix d'une image dans la gallery. La fonction *onActivityResult* se charge d'interpréter le résultat de ces activités de chargement d'image. Elle vérifie d'abord si l'activité s'est déroulée normalement (*if (resultCode == RESULT_OK*)). Si c'est le cas, elle interprète le code de la requête : s'il correspond à celui de l'activité de la caméra, elle fait appel à la fonction *onCaptureImageResult* qui se charge d'afficher l'image provenant de la caméra, et s'il correspond à celui de l'activité de la gallery, elle fait appel à la fonction *onSelectFromGalleryResult* qui se charge d'afficher l'image provenant de la gallery.

2.2 Afficher une image

Le `SecondActivity` est composée d'un `ImageView`, initialisé dans la fonction *display*, et occupant tout l'écran du terminal. Cet `ImageView` est actualisé avec une copie de l'image choisie par l'utilisateur par l'intermédiaire de la fonction *onCaptureImageResult* ou *onSelectFromGalleryResult*. Un problème est rencontré pour l'affichage de l'image provenant de la caméra. En effet, l'image affiché est la miniature de la photo prise par la caméra, et non l'image haute résolution. Nous essayons de trouver une solution à ce problème.

2.3 Zoomer & Scroller

Nous avons tenté d'implémenter notre propre classe `MyImageView`, sans succès. Afin de quand même avoir cette fonctionnalité, nous avons importé la library `PhotoView` de `chrisbanes` qui gère le zoom et le scroll d'images. Pour pouvoir importer cette library, nous avons rajouté la ligne (*compile 'com.github.chrisbanes.photoview:library:1.2.3'*) dans le `build.gradle` du projet, puis synchronisé le gradle (*Tools -> Android -> Sync...*). L'utilisation du `PhotoView` consiste à instancier un `PhotoViewAttacher`, en l'attachant à l'`ImageView` pré-existant. Chaque changement de l'image affiché par l'`ImageView` nécessite l'appel à la fonction *update* du `PhotoViewAttacher`. Les fonctionnalités de zoom et le scroll sont ainsi automatiquement prises en compte dans l'`ImageView`.

2.4 Appliquer des filtres

L'ensemble des filtres est implémenté dans la classe `Filters`. Nous avons implémenté les filtres *luminosity*, *contrast*, *histogram equalization*, *grayscale*, *sepia*, *hue selection*, *invert*, *anaglyphe 3D*, *colorize*, *red canal*, *green canal*, *blue canal*. Les filtres de convolution restent à faire. L'appel à ces filtres est réalisé en cliquant sur des boutons situé sur un `ScrollView` horizontal positionné en bas de l'écran, et des boites de dialogue.

2.5 Réinitialiser

Dans le `SecondActivity`, l'image obtenue via la caméra ou la gallery est stockée dans un bitmap *bmp*. Les modifications de l'image par les filtres sont réalisées sur une copie modifiable de ce bitmap, *bmpMod*, qui est chargée dans l'`ImageView` de l'activité. Réinitialiser l'image consiste à charger une copie du bitmap d'origine *bmp* dans l'`ImageView`.

2.6 Sauvegarder une image

Dans le `SecondActivity`, nous avons implémenté une façon de sauvegarder l'image modifiée. Le code semble bon, cependant l'image ne se sauvegarde pas car nous obtenons une erreur "open failed". Nous cherchons une solution à ce problème.

3 Conclusion

L'application est partiellement fonctionnelle car il manque certaines fonctionnalités. Ces fonctionnalités sont en train d'être réfléchies et implémentées.