Introduction au développement sous Android
TP5

Master WeDSci/I2L - 2me année

année 2023-2024

Introduction

L'objectif de ce TP est de poursuivre l'utilisation des intentions implicites et d'expérimenter l'utilisation des menus et des menus contextuels. Il aura pour cadre une l'application permettant de charger et modifier des images.

Exercice 1

L'objectif de cet exercice est d'écrire une application nommée TPimageXXX, où XXX représente votre nom de *login*, permettant de charger une image stockée sur le périphérique et de l'afficher. Une vue de l'application à réaliser est donnée sur la figure 1 :

- l'image à charger se trouve au centre de l'interface ;
- le bouton de chargement se trouve en bas de l'écran (vous utiliserez un composant `ImageButton`) ;
- le texte figurant en haut de l'écran correspond à l'*URI* de l'image chargée.

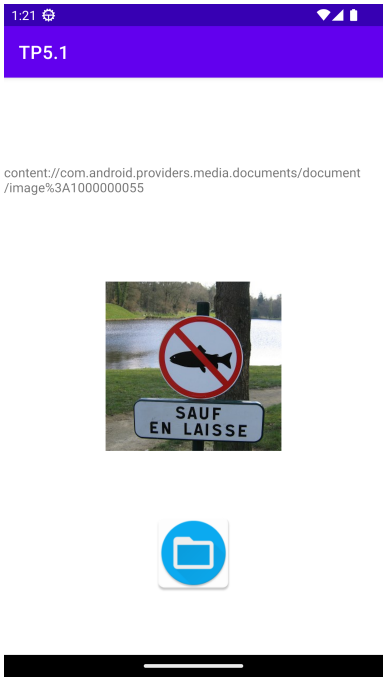


FIGURE 1 – Vue de l'interface graphique de l'exercice 1.

Avant de pouvoir charger une image, il est nécessaire que des images soient présentes sur le périphérique virtuel ... Pour ce faire, la méthode la plus simple est d'utiliser l'appareil de photo embarqué sur votre émulateur :

- Vérifiez, en utilisant le *Android Virtual Device manager*, que la caméra de votre périphérique est bien émulée. Dans le cas contraire, activez cette option ;
- Vous pouvez ensuite utiliser l'appareil photo pour générer quelques images de test ;

— Sur certaines versions d'émulateur se trouve une option *scène virtuelle* qui permet de prendre des images d'une scène un peu plus complexe qu'un simple damier.

Vous pouvez également ajouter des images dans le périphérique virtuel sous Android Studio :

— Vérifiez qu'une SD-Card virtuelle est bien présente sur le périphérique virtuel (via *AVD manager*), puis ouvrez l'explorateur de fichiers du périphérique virtuel (*Device File Explorer*) via le menu **View/Tools/Device File Explorer** ou le bouton correspondant dans le coin inférieur droit de la fenêtre d'Android Studio. À noter qu'il est nécessaire qu'un périphérique virtuel soit en activité pour pouvoir effectuer les actions qui suivent ...
— dans l'explorateur de fichiers qui apparaît, sélectionnez le dossier `mnt/sdcard`. Sélectionnez éventuellement l'un des sous-dossiers `Pictures` ou `DCIM` s'ils sont présents ;
— en vous assurant que le dossier choisi est bien surligné (dans la négative, la copie risque d'être faite à un endroit où vous ne retrouverez pas votre image ...), sélectionnez l'option `upload` pour télécharger une image ;
— Notez qu'il sera nécessaire de redémarrer le périphérique virtuel pour que ces images soient prises en compte ...

étape 1

Dans un premier temps, créez l'interface graphique telle qu'elle apparaît sur la figure 1. On précise que le *widget* à utiliser pour l'image est `ImageView`. Compléter le code de l'application de manière à pouvoir gérer l'appui sur le bouton de chargement, en faisant afficher un message dans la console.

étape 2

Complétez votre application avec une variable de type `ActivityResultLauncher` qui permettra de récupérer l'`Uri` de l'image qui sera choisie. La classe de contrat à utiliser sera alors

`ActivityResultContracts.GetContent`

et le paramètre à utiliser lors du lancement de l'intention sera une chaîne de caractères représentant le type `mime` des images à rechercher. Dans un premier temps, la fonction *callback* qui traitera l'`Uri` de l'image choisie par l'utilisateur se contentera de mettre à jour le champs texte correspondant de l'interface.

étape 3

Complétez votre application en y ajoutant le code de la méthode suivante :

`private void ChargerImage(Uri imageUri)`

qui permettra de charger et afficher l'image dont l'emplacement est indiqué par la paramètre `imageUri`. Cette fonction sera appelée par la fonction `callback` de la question précédente.

On précise les points supplémentaires suivants :

— l'image à récupérer sera stockée dans une instance de la classe `Bitmap`, que vous récupérerez en utilisant le code suivant [1], en supposant que l'`Uri` de l'image se trouve dans la variable `imageUri` :

```
//  ----- préparer les options de chargement de l'image
BitmapFactory.Options option = new BitmapFactory.Options();
option.inMutable = true; // l'image pourra être modifiée
// ------ chargement de l'image - valeur retournée null en cas d'erreur
Bitmap bm = BitmapFactory.decodeStream(getContentResolver().openInputStream(imageUri), null, option)
```

où `imageUri` représentera l'*URI* de l'image retournée par l'application extérieure ;
— la classe à utiliser pour l'affichage de l'image est `ImageView` ;

Exercice 2

Modifiez votre application de chargement d'images en lui ajoutant un menu qui contiendra deux opérations (voir figure 2a) permettant d'effectuer respectivement (i) un miroir horizontal de l'image (figure 2b) et (ii) un miroir vertical (figure 2c) de l'image. Pour récupérer le bitmap mémorisé dans le *ImageView*, vous pourrez utiliser le code suivant :

1. davantage d'informations seront données sur ce code lors de l'étude des `Content Provider`.

```
// image est une instance de ImageView
 Bitmap bitmap = ((BitmapDrawable)image.getDrawable()).getBitmap();
```

(a) Les deux entrées du menu avant de faire un choix.

(b) Résultat après une symétrie horizontale.

(c) Résultat après une symétrie verticale.

FIGURE 2 – Différentes vues de l'utilisation du menu demandé.

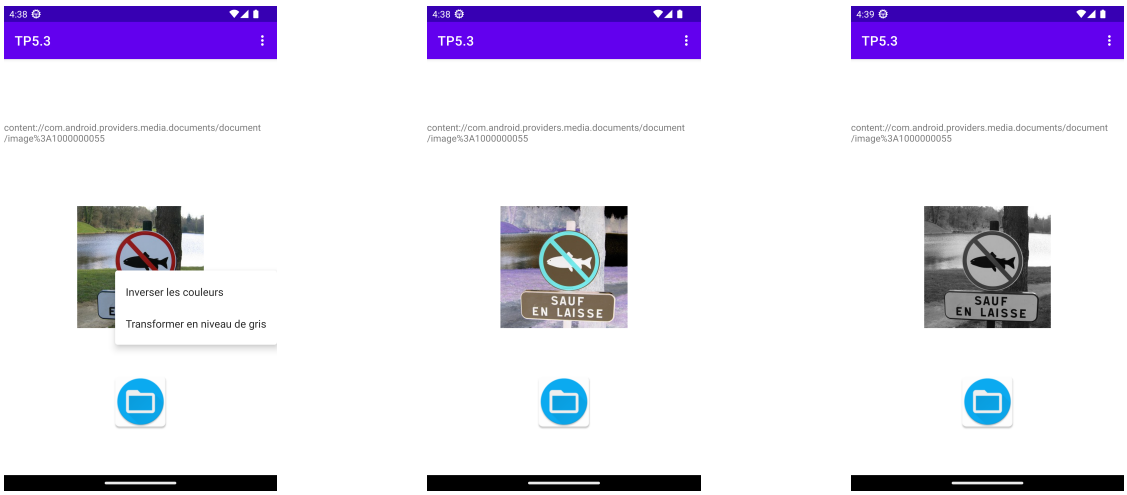
On précise que ces opérations devront se faire pixel par pixel, et non pas en utilisant les opérations géométriques sur les images (solution la plus fréquemment rencontrée sur les forums ...) telles que `setScaleX`, `setScaleY`, utilisation d'une matrice de transformation, etc.

Vous utiliserez donc les méthodes `getPixel` et `setPixel`, qui permettent un accès direct aux pixels d'un bitmap. Vous gérerez le fait que l'utilisateur pourra déclencher les opérations de miroir alors qu'aucune image n'est chargée ...

Exercice 3

Ajoutez un menu contextuel au composant d'affichage de l'image (voir figure 3a), de telle sorte que les deux opérations suivantes puissent être appliquées à l'image en cas de choix :

— Inverser les couleurs de l'image (voir figure 3b) ;
— Transformer l'image en niveaux de gris (voir figure 3c).



(a) Les deux entrées du menu contextuel.

(b) Résultat après une inversion des couleurs.

(c) Résultat après une transformation en niveaux de gris.

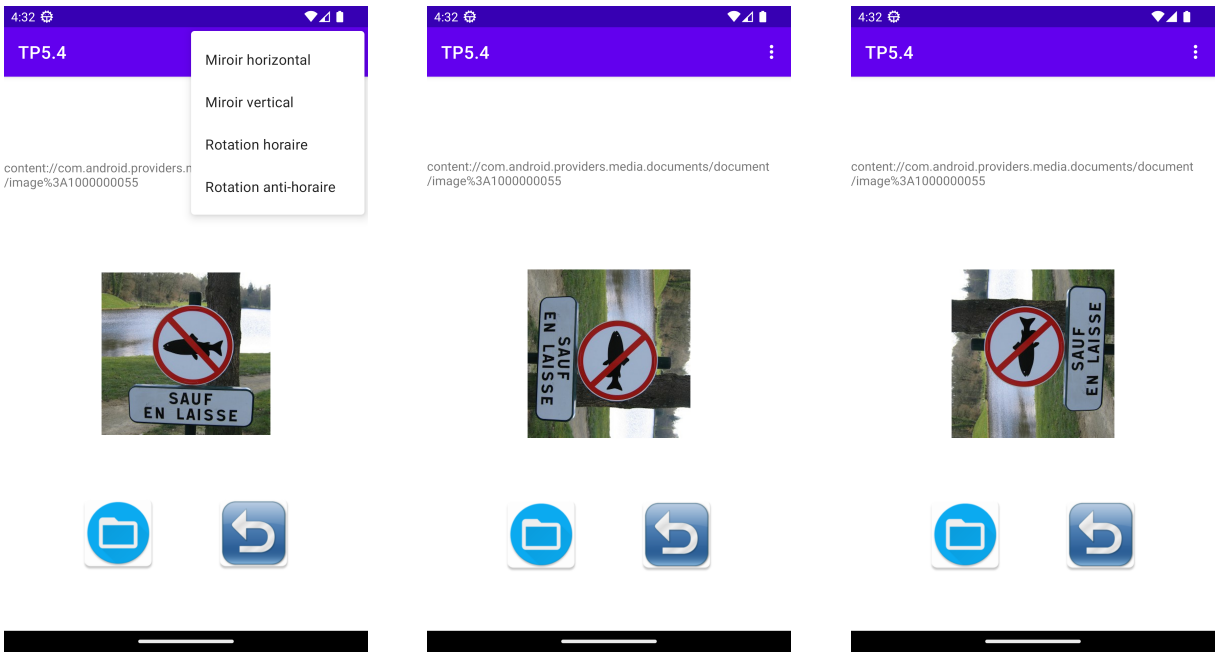
FIGURE 3 – Différentes vues de l'application pour le menu contextuel.

Afin de pouvoir réaliser ces opérations, les informations suivantes vous sont données :

— Une couleur est représentée par un entier 32 bits dont chaque octet représente un canal de couleur. Les canaux sont notés alpha (pour la transparence), rouge, vert et bleu (pour les trois couleurs primaires). Le canal alpha ne sera jamais modifié dans ce TP ;
— Android fourni une classe nommée `Color` permettant de manipuler facilement ces différents canaux ;
— La classe `Bitmap` permet d'affecter une couleur à un pixel ou récupérer la couleur de celui-ci via ses méthodes `setPixel()` et `getPixel()` ;
— Inverser un canal de couleur consiste à transformer sa valeur v en $(255 - v)$;
— Convertir une couleur (a, r, v, b) en un niveau de gris consiste à affecter la même valeur aux trois canaux de couleur primaire, cette dernière pouvant être calculée, dans un premier temps, comme la moyenne m des trois primaires (soit $(a, r, v, b) \rightarrow (a, m, m, m)$).

Exercice 4

Finalisez votre application de telle sorte qu'elle dispose des nouvelles fonctionnalités suivantes :
— Un bouton d'annulation, qui remet l'image dans son état initial, c'est à dire au moment où elle a été chargée ;
— Deux choix supplémentaires dans le menu principal ((voir figure 4a), permettant :
  — d'effectuer une rotation à 90 degrés de l'image dans le sens horaire (figure 4b) ;
  — d'effectuer une rotation à 90 degrés de l'image dans le sens anti-horaire (figure 4c).
  Les opérations concernant ces deux rotations devront être faites pixel par pixel.



(a) Les deux entrées supplémentaires du menu.

(b) Résultat après une rotation horaire.

(c) Résultat après une rotation anti-horaire.

FIGURE 4 – Différentes vues de l'utilisation des nouvelles fonctionnalités.