On cherche à développer une application capable de prendre une photo et d'y appliquer un ou plusieurs filtres.

1 Architecture générale

Question 1 Créez un nouveau projet. Lorsque le type d'activité est demandé, choisir *Empty Views Activity*.

Modifiez le fichier XML spécifiant le layout de `MainActivity` de sorte à obtenir un résultat similaire à celui présenté en Figure 1 : deux `Button` et une `ImageView` (on choisira une image quelconque en tant que `Drawable` de départ pour l'`ImageView`).

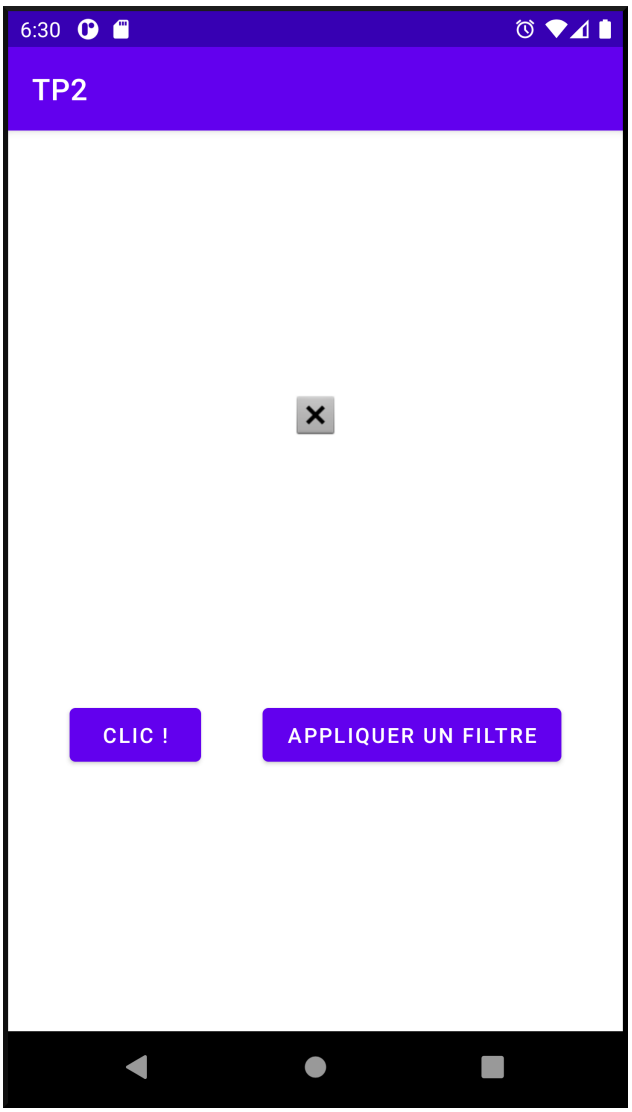

FIGURE 1 – Le layout de `MainActivity`

2 Prendre une photo

On va maintenant se concentrer sur la prise d'une photo. Cela se fera quand l'utilisateur cliquera sur le `Button` "Clic !".

Pour cela, on va utiliser un `Intent` implicite, dans le but de demander à une `Activity` sachant le faire de prendre une photo pour nous.

On va donc faire un appel à `registerForActivityResult()`. On souhaite travailler avec des `Bitmap` : ce sera donc le type `O` d'output. Puisqu'on n'a pas d'instruction particulière à donner pour la prise de photo, le type `I` d'entrée sera `Void`.

Question 2 Implémentez le callback de ce `registerForActivityResult()`. Celui-ci se contentera de copier le `Bitmap` dans une variable globale `bitmap`, et de l'afficher dans l'`ImageView`.

Pour la copie, on utilisera la méthode `Bitmap::copy(Bitmap.Config.ARGB_8888,true)`.

L'affichage se fera grâce à la méthode `ImageView::setImageBitmap()`.

Question 3 Implémentez le contrat de ce `registerForActivityResult()`.

En créant l'`Intent` dans `createIntent()`, on utilisera `MediaStore.ACTION_IMAGE_CAPTURE`, qui permet de demander à ce qu'une photo soit prise.

Dans `parseResult()`, le `Bitmap` à récupérer est associé à la clef `"data"` :

```
public Bitmap parseResult(int resultCode, Intent result) {
  return (Bitmap) result.getExtras().get("data");
}
```

N.B. *À noter qu'une `Activity` lancée par un `Intent` implicite de type `ACTION_IMAGE_CAPTURE`, en plus de renvoyer un `Bitmap` dans l'`Intent` de retour, stocke une image de plus haute résolution dans le système de fichiers de l'appareil. On ignorera cette image pour l'instant, mais on verra plus tard dans le semestre comment la récupérer.*

Question 4 Implémentez la méthode `setOnClickListener()` du `Button` "Clic !" pour qu'un clic déclenche la prise d'une photo et son affichage dans l'`ImageView`.

Ajoutez la ligne

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

au fichier Android Manifest, et testez votre code.

3 Appliquer un filtre de permutation des couleurs

On veut que chaque clic sur le `Button` "Appliquer un filtre" permute les couleurs de la photo. Un `Bitmap` est une matrice de dimension `Bitmap::getWidth()` × `Bitmap::getHeight()` de pixels, qui ne sont rien de plus que des `int`.

Chaque pixel se décompose en rouge, vert et bleu (RGB).

Question 5 Implémentez une méthode `permute()`, qui parcourt `bitmap`, et pour chaque pixel `p` qui le compose, récupéré via la méthode `Bitmap::getPixel(int x, int y)`,

— récupère la teneur en rouge de `p` grâce à la méthode statique `Color.red` :

```
int red(int pixel)
```

— fait de même pour les teneurs en vert et en bleu
— utilise la méthode `Bitmap::setPixel(int x, int y, int col)` pour remplacer le pixel `p`, en permutant ses teneurs en rouge, vert et bleu. On utilisera la méthode statique `Color.rgb`

int rgb(**int** r, **int** g, **int** b)

pour créer le nouveau pixel.

La méthode `permute()` modifie donc `bitmap` en place.

Question 6 Implémentez le callback `setOnClickListener()` du `Button` "Appliquer un filtre", et testez le résultat.

4 Pour celles et ceux qui vont plus vite que la musique

Question 7 (☠) Complétez selon vos envies l'interface de `PhotoFiltre` pour qu'elle permette davantage de fonctionnalités. À titre d'exemple, on pourra proposer à l'utilisateur de flouter l'image en faisant des moyennes locales sur les pixels, de renverser l'image, de lui faire effectuer des rotations... On créera autant de `Button` et de `Spinner` que nécessaire.

En particulier, on abandonnera la modification de `bitmap` en place, pour avoir plus de libertés. On pourra alors créer un nouveau `Bitmap` grâce à la méthode statique `Bitmap.createBitmap`

Bitmap createBitmap(**int** width, **int** height, Bitmap.Config c)

Pour le troisième argument, on dupliquera la configuration de `bitmap` via `Bitmap::getConfig()`.