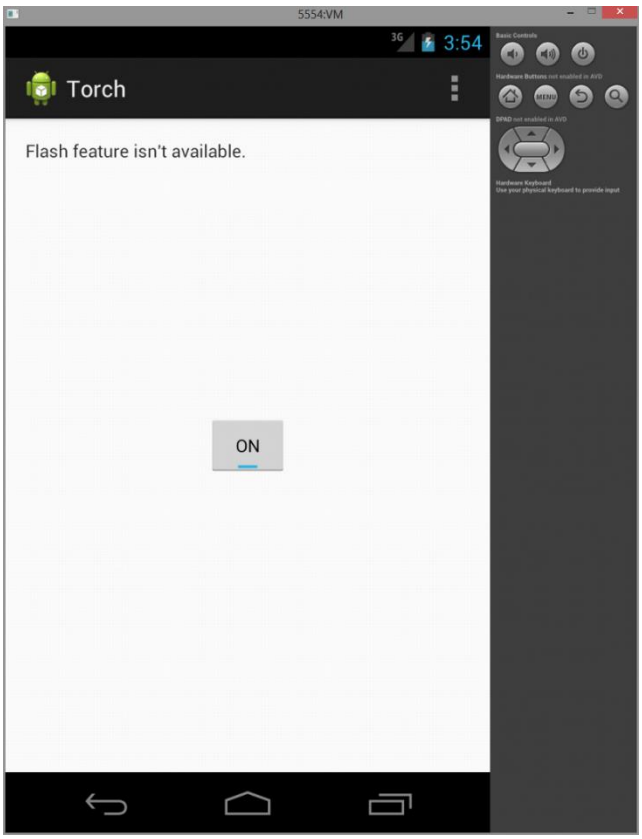
Mon application : Lampe torche

Le but de ce projet est de développer une petite application, votre véritable première application, ayant pour fonctionnalité d'utiliser le flash de votre téléphone comme lampe torche. Cette fonctionnalité n'étant pas présente sur tous les téléphones, veuillez vérifier sa présence avant de commencer. A défaut, travaillez sur la possibilité d'utiliser l'écran comme « lampe » en augmentant la luminosité et en affichant un layout entièrement blanc.

Explications



Les étapes permettant d'arriver à la fin du TP sont :

- Modifier les permissions de l'application afin d'autoriser l'utilisation de l'appareil photo (« CAMERA ») et du flash (« FLASHLIGHT »).
- Modifier le layout afin d'ajouter un bouton (ici un « ToggleButton ») pour contrôler l'allumage et l'extinction du Flash.
- Créer deux fonctions permettant l'allumage et l'extinction du flash d'une manière sécurisée.

Les permissions

Par défaut, l'application n'a aucune permission. Ainsi, dès que vous souhaitez utiliser une fonctionnalité de l'appareil, vous devez le spécifier dans l'« AndroidManifest.xml ». Il en existe plus de 130, en voici quelques-unes :

- ACCESS_NETWORK_STATE : Autorise l'accès aux informations réseau
- ACCESS_WIFI_STATE : Autorise l'accès aux informations
- BATTERY_STATS : Autorise l'application à récupérer le statut de la batterie
- CAMERA : Autorise l'utilisation de l'appareil photo
- FLASHLIGHT : Autorise l'accès au flash
- INTERNET : Autorise l'accès à Internet
- MOUNT_FORMAT_FILESYSTEMS : Autorise le formatage de la carte SD
- NFC : Autorise l'utilisation du capteur NFC
- READ_CONTACTS : Autorise l'accès en lecture des contacts
- WRITE_EXTERNAL_STORAGE : Autorise l'écriture sur la carte SD.

Pour cette application, nous allons ajouter ces deux dans le manifeste :

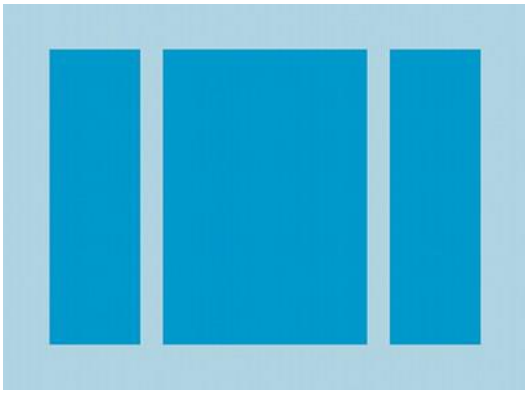
```
<uses-permission android:name="android.permission.FLASHLIGHT"/>
<uses-permission android:name="android.permission.CAMERA"/>
```

Layout(s)

Un layout est un moyen de mettre en page de votre vue. Il contient un ensemble d'objets héritant soit de « ViewGroup », soit de « View » comme pour « ToggleButton » ou bien « Textview » (comme vu dans le précédent TP).Un layout peut contenir d'autres layout, etc. …

Il en existe plusieurs fourni par le SDK, les plus connus sont: « LinearLayout », « RelativeLayout », « ListView ».

LinearLayout



Il permet d'aligner tous les enfants dans la même direction (soit à la verticale, soit à l'horizontal).

Il est très pratique à utiliser. Pensez à des formulaires, etc. …

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

</LinearLayout>
```

RelativeLayout

Il permet d'effectuer des positionnements relatifs.

La position de chaque élément peut-être spécifiée en fonction des éléments voisins : « à la gauche de », « avant », « aligner avec », « au milieu de », etc. …

Nous allons utiliser ce layout afin de placer un bouton au milieu de la vue.

http://developer.android.com/guide/topics/ui/declaring-layout.html

ToggleButton

Il s'agit d'un bouton à deux états. Il va nous permettre d'allumer ou d'éteindre le flash en fonction de son statut. Pour récupérer le statut du bouton, vous devez utiliser la méthode « .isChecked() », elle retourne un booléen.

Ajoutez-en un dans votre xml, et vous devriez vous approcher de ceci :

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity" >

    <TextView
        android:id="@+id/message"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_world" />

    <ToggleButton
        android:id="@+id/toggleButton1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        android:text="ToggleButton" />

</RelativeLayout>
```

Afin de manipuler ce bouton, vous devez le déclarer à l'intérieur de votre activité :

```
public class MainActivity extends Activity {
      ToggleButton button = null;
      Camera cam = null;

[…]
```

Dans la méthode « onCreate() », appelé lors de la construction de l'activité, nous allons initialiser notre objet en utilisant la méthode « findViewById() ». Cette méthode permet de rechercher la référence d'un objet à l'intérieur d'une vue à partir d'un identifiant (identifiant déclaré à l'intérieur votre fichier xml « @+id/MyId »). Cela permet de créer un lien entre votre vue et votre code java.

Afin de détecter le changement d'état de notre bouton, nous allons lui assigner un « OnClickListener » :

```
button = (ToggleButton)findViewById(R.id.toggleButton1);
button.setChecked(true);
button.setOnClickListener(new OnClickListener() {
	@Override
	public void onClick(View v) {
		if (button.isChecked()) {
			startFlash();
		} else {
			stopFlash();
		}
	}
});
```

Et en fonction du statut du bouton, nous allons activer ou non le flash.

Si vous souhaitez avoir plus d'informations sur les « ToggleButton » :
http://developer.android.com/guide/topics/ui/controls/togglebutton.html

Objet caméra

L'objet « Camera » est une abstraction permettant de manipuler complètement l'appareil photo ainsi que le flash. Je vous invite à consulter cette page afin de prendre connaissance de toutes les possibilités de cet objet : http://developer.android.com/reference/android/hardware/Camera.html .

Faites attention lors de l'utilisation de cet objet : Ce dernier contient un verrou permettant à une seule instance d'application d'accéder simultanément au périphérique. Si vous ne vous déconnectez par correctement de la caméra lors de la sortie de votre application, le verrou ne sera pas libéré et aucune autre application ne pourra l'utiliser. Vous seriez ainsi obligés de redémarrer le téléphone.

Vérifier si la fonctionnalité est présente

Il est important de tester la présence de la fonctionnalité, dans le cas contraire vous risquez un plantage de l'application.

Une méthode de l'objet « Activity » permet de le faire :

```
this.getPackageManager().hasSystemFeature(PackageManager.FEATURE_CAMERA_FLASH)
```

Allumer le flash

Pour allumer le flash, il vous suffit de modifier les paramètres de l'appareil photo comme indiqué ci-dessous puis lancer la prévisualisation. Vous devez au préalable récupérer l'instance de l'objet « Camera » en appelant la méthode static « open() ». Cette dernière peut ne pas réussir et générer une exception, pensez à ajouter des vérifications afin d'éviter tout crash.

```
void startFlash() {
	if (cam != null)
		stopFlash();
	cam = Camera.open();
	Parameters p = cam.getParameters();
	p.setFlashMode(Parameters.FLASH_MODE_TORCH);
	cam.setParameters(p);
	cam.startPreview();
}
```

Eteindre le flash

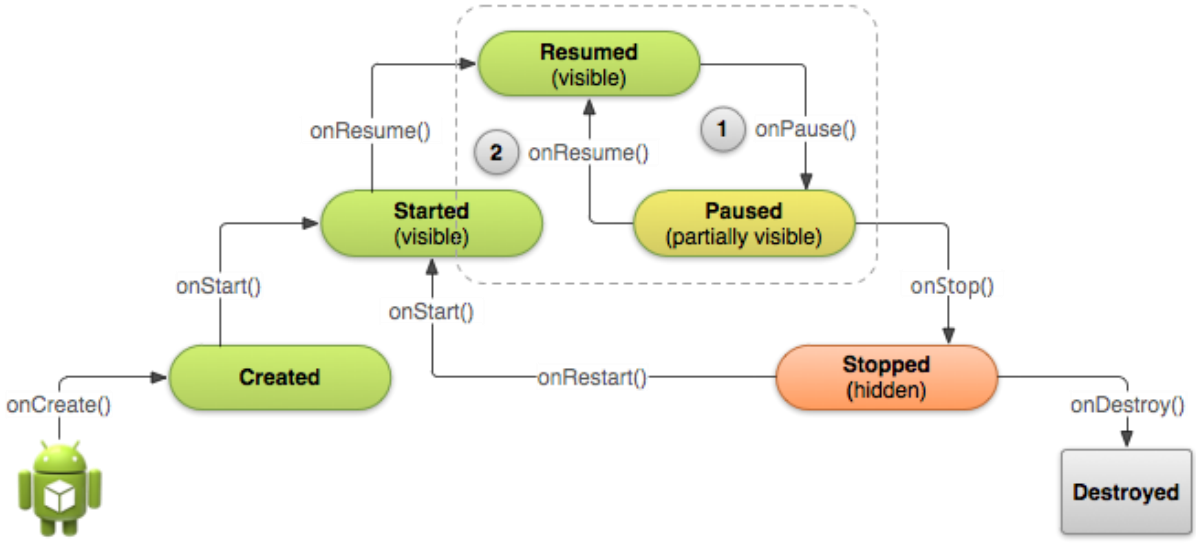
Dans le même principe, il suffit d'arrêter la prévisualisation et de libérer l'objet.

```
void stopFlash() {
	if (cam != null) {
		cam.stopPreview();
		cam.release();
		cam = null;
	}
}
```

Cycle de vie de l'application

Afin d'éviter tout blocage de l'appareil photo, il est vivement conseillé d'éteindre le flash quand l'utilisateur sort de l'application, et de le rallumer quand il la relance grâce aux méthodes « OnClose() » et « OnResume() ».

Voici un diagramme[1] d'état représentant les différentes étapes lorsque qu'une application est arrêtée et/ou lancée :



Résultat

Le projet « Torch » est aussi disponible sur le dépôt suivant : https://github.com/steven-martins/tp-android

[1] http://developer.android.com/training/basics/activity-lifecycle/pausing.html