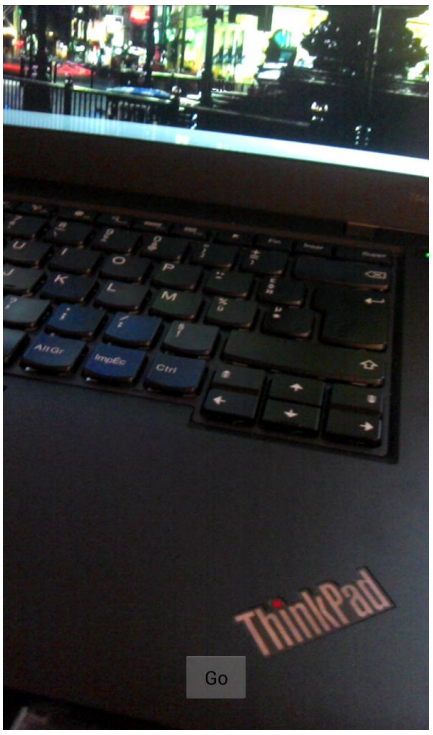
Mon application : Capture Photo



Explications

Ce TP a pour but de vous faire manipuler un capteur : l'appareil photo.

Pour complètement y arriver, vous devrez :

- Manipuler correctement l'objet Camera en respectant le cycle de vie de l'activité (comme pour le TP lampe torche).
- Avoir un layout permettant de prévisualiser la caméra (SurfaceView)
- Injecter un second layout comportant un bouton permettant le déclenchement de l'enregistrement d'une photo
- Détecter l'orientation du périphérique afin d'avoir les meilleurs paramètres de prévisualisation

Nous vous conseillons de lire le contenu de ce lien pour résoudre ce TP :
http://developer.android.com/training/camera/photobasics.html

Ainsi que :
http://developer.android.com/training/camera/cameradirect.html

Ajout des permissions

Cette application nécessite deux autorisations : la première pour manipuler l'appareil photo, la seconde afin de pouvoir écrire nos photos sur le périphérique.

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

Layout

Comme indiqué précédemment, le layout ne contient qu'un objet « SurfaceView ».

Fullscreen

Afin d'avoir un rendu plein écran, il suffit d'appliquer le flag « fullscreen » à la fenêtre ainsi que demander une fenêtre sans titre :

```
requestWindowFeature(Window.FEATURE_NO_TITLE);
getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
                     WindowManager.LayoutParams.FLAG_FULLSCREEN);
```

SurfaceView

C'est un moyen rapide et abstrait de manipuler une vue pixel par pixel.

Dans un premier temps, l'objet doit être initialisé ainsi que son « holder » :

```
preview = (SurfaceView) findViewById(R.id.surfaceView1);
previewHolder = preview.getHolder();
previewHolder.addCallback(surfaceCallback);
previewHolder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
```

L'objet précédemment initialisé nécessite la création du callback « SurfaceHolder.Callback », celui-ci va se charger de l'initialisation de la prévisualisation :

```
SurfaceHolder.Callback surfaceCallback = new SurfaceHolder.Callback() {
	public void surfaceCreated(SurfaceHolder holder) {
	}

	public void surfaceChanged(SurfaceHolder holder, int format, int
width, int height) {
		initPreview(width, height);
		startPreview();
	}

	public void surfaceDestroyed(SurfaceHolder holder) {
	}
};
```

« Overlay » : Injecter une vue

Afficher une vue au-dessus d'une autre, dans notre cas au-dessus de « SurfaceView », il suffit d'appeler la méthode « addContentView » de l'activité courante en précisant l'objet « View » préalablement généré à partir du fichier xml :

```
overlay = getLayoutInflater().inflate(R.layout.overlay, null);
infos = (TextView)overlay.findViewById(R.id.textView1);
infos.setText("Hello");
addContentView(overlay, new
LayoutParams(LayoutParams.FILL_PARENT,LayoutParams.FILL_PARENT));

Button button = (Button)overlay.findViewById(R.id.button1);
button.setOnClickListener(new OnClickListener() {

	[..]

});
```

OrientationEventListener

Afin de gérer correctement la prévisualisation de l'appareil photo, il est important de prendre en compte l'orientation du périphérique grâce à un « OrientationEventListener » :

```
OrientationEventListener orientationListener = new
OrientationEventListener(this) {

      @Override
      public void onOrientationChanged(int orientation) {
            if (orientation == ORIENTATION_UNKNOWN) return;
            android.hardware.Camera.CameraInfo info =
                  new android.hardware.Camera.CameraInfo();
            android.hardware.Camera.getCameraInfo(0, info);
            orientation = (orientation + 45) / 90 * 90;
            int rotation = 0;
            if (info.facing == CameraInfo.CAMERA_FACING_FRONT) {
                rotation = (info.orientation - orientation + 360) % 360;
            } else {  // back-facing camera
                rotation = (info.orientation + orientation) % 360;
            }
            orient = rotation;
            if (camera != null) {
              Camera.Parameters parameters = camera.getParameters();
              Log.d("onOrientationChanged", "Rotation: " + orient);
              parameters.setRotation(rotation);
            }

      }
};
orientationListener.enable();
```

Tachez de mettre à disposition l'orientation du périphérique aux autres méthodes (dans notre cas « orient »).

Camera

La surface de prévisualisation est liée à l'appareil photo. Il est ensuite configuré sur deux aspects :

- L'autofocus, si cette fonctionnalité est disponible, nous l'activons
- La sélection de la taille du flux de prévisualisation. Il existe un nombre important de résolutions différentes. La fonction suivante permet de trouver la surface la plus importante en prenant en compte l'orientation du téléphone :

```
	private Camera.Size getBestPreviewSize(int width, int height,
			Camera.Parameters parameters) {
		Camera.Size result = null;
		if (orient == 90) {
			int tmp = width;
			width = height;
			height = width;
		}
		for (Camera.Size size : parameters.getSupportedPreviewSizes()) {
			if (size.width <= width && size.height <= height) {
				if (result == null) {
					result = size;
				} else {
					int resultArea = result.width * result.height;
					if (size.width * size.height > resultArea) {
						result = size;
					}
				}
			}
		}
		return (result);
	}
```

Code pour initialiser la prévisualisation :

```
private void initPreview(int width, int height) {
	if (camera != null && previewHolder.getSurface() != null) {
		try {
			camera.setPreviewDisplay(previewHolder);
		} catch (Throwable t) {
			// TODO Auto-generated catch block
			t.printStackTrace();
		}

		if (!cameraConfigured) {
			Camera.Parameters parameters = camera.getParameters();
			Camera.Size size = getBestPreviewSize(width, height,
parameters);

			List<String> focusModes = parameters.getSupportedFocusModes();
			if
(focusModes.contains(Camera.Parameters.FOCUS_MODE_CONTINUOUS_PICTURE)) {

parameters.setFocusMode(Camera.Parameters.FOCUS_MODE_CONTINUOUS_PICTURE);
			}
			if (size != null) {
				parameters.setPreviewSize(size.width, size.height);
				camera.setParameters(parameters);
				cameraConfigured = true;
			}
			camera.setDisplayOrientation(orient);
		}
	}
}

private void startPreview() {
	if (cameraConfigured && camera != null) {
		camera.startPreview();
		inPreview = true;
	}
}
```

Prendre une photo

Il existe plusieurs méthodes pour prendre une photo, comme l'indique la documentation Android.

Dans notre cas, il est préférable d'utiliser la version avec le callback permettant d'avoir une meilleure maîtrise et pourquoi pas appliquer des filtres sur l'image avant sa sauvegarde.

```
camera.takePicture(null, null, new PictureCallback() {

	@Override
	public void onPictureTaken(byte[] arg0, Camera arg1) {
		arg1.startPreview();
		try {
			File file = new File(createImageFileName());
			FileOutputStream outputStream = new
FileOutputStream(file);
			outputStream.write(arg0);
			outputStream.close();
			updateGallery(file);
		} catch (IOException e) {
			e.printStackTrace();
		}

	}}
);
```

Une fois la photo prise, la prévisualisation est automatiquement coupée. Vous devez la relancer manuellement.

« createImageFileName() » est une méthode permettant de générer un nom de fichier *unique* pour votre photo. (Il s'agit d'une version dérivée du portail développeur d'Android) :

```
private String createImageFileName() {
    String timeStamp = new SimpleDateFormat("yyyyMMdd_HHmmss").format(new
Date());
    String imageFileName = "JPEG_" + timeStamp + "_";
    File storageDir = Environment.getExternalStoragePublicDirectory(
            Environment.DIRECTORY_PICTURES);
    return storageDir.getAbsolutePath() + "/" + imageFileName + ".jpg";
}
```

Mettre à jour la bibliothèque

Voici un petit morceau de code permettant de faire prendre en compte votre nouvelle photo par le système :

```
private void updateGallery(File f) {
	Intent mediaScanIntent = new Intent(Intent.ACTION_MEDIA_SCANNER_SCAN_FILE);
	Uri contentUri = Uri.fromFile(f);
	mediaScanIntent.setData(contentUri);
	this.sendBroadcast(mediaScanIntent);
}
```

Résultat

L'ensemble du projet « PhotoCapture » est disponible sur le dépôt github :
https://github.com/steven-martins/tp-android

Une très grande partie du code contenu dans ce TP est issu de la page :
http://developer.android.com/training/camera/photobasics.html