Multimédia

sous android

Plan

I. Vibreur

```
<uses-permission android:name="android.permission.VIBRATE" />
Vibrator v;
v = (Vibrator) getSystemService(Context.VIBRATOR_SERVICE);
int duree=100;   //en milliseconde
v.vibrate(duree);
```

II. Capture photo

II.1. Lancer l'intent de capture de la photo

```
<uses-feature android:name="android.hardware.camera"
                  android:required="true" />
static final int REQUEST_IMAGE_CAPTURE = 1;

private void dispatchTakePictureIntent() {
    Intent takePictureIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
    if (takePictureIntent.resolveActivity(getPackageManager()) != null) {
        startActivityForResult(takePictureIntent, REQUEST_IMAGE_CAPTURE);
    }
}
```

II.2. Récupérer la photo capturée

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (requestCode == REQUEST_IMAGE_CAPTURE && resultCode == RESULT_OK) {
        Bundle extras = data.getExtras();
        Bitmap imageBitmap = (Bitmap) extras.get("data");
        mImageView.setImageBitmap(imageBitmap);
    }
}
```

III. Capture Video

III.1. Lancer l'intent de capture de la vidéo

```
<uses-feature android:name="android.hardware.camera"
                  android:required="true" />
static final int REQUEST_VIDEO_CAPTURE = 2;
private void dispatchTakeVideoIntent() {
    Intent takeVideoIntent = new Intent(MediaStore.ACTION_VIDEO_CAPTURE);
    if (takeVideoIntent.resolveActivity(getPackageManager()) != null) {
        startActivityForResult(takeVideoIntent, REQUEST_VIDEO_CAPTURE);
    }
}
```

III.2. Récupérer la vidéo capturée

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent intent) {
    if (requestCode == REQUEST_VIDEO_CAPTURE && resultCode == RESULT_OK) {
        Uri videoUri = intent.getData();
        mVideoView.setVideoURI(videoUri);
        mVideoView.start();
    }
}
```

IV. Détection de visage

IV.1. Mobile Vision API

La version 7.8 des Google Play Services a vu l'ajout des API Mobile Vision en remplacement de l'API Android FaceDetector.Face. Elle apporte de grosses améliorations de détection en image fixe ou en mouvement.

Ces API ne reconnaissent pas le visage d'une personne, mais permettent de les détecter en image ou en vidéo. Si un visage quitte le champ de vue d'une vidéo, mais réapparait plus tard, il sera détecté comme un nouveau visage [4].

Pour utiliser cette API ajouter dans le gradule :

implementation "com.google.android.gms:play-services-vision:20.1.3"

IV.2. La détection

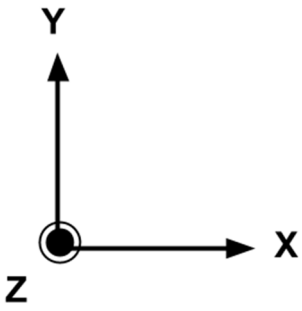
Si un objet visage est détecté, les coordonnées de son placement sur l'image seront retournées :

- getPosition() : retourne la coordonnée la plus en haut et à gauche du visage sur l'image.
- getWidth() : retourne la largeur de la zone où le visage a été détecté
- getHeight() : retourne la hauteur de la zone où le visage a été détecté
- getId() : retourne un ID que le système a associé au visage [4]

IV.3. L'orientation

L'API Face est capable de détecter un visage dans de multiple orientation. L'API est donc capable de détecter un visage même si la moitié du visage est manquant.

- getEulerY() : retourne la rotation du visage selon l'axe vertical. (y==Euler Y)
- getEulerZ() : retourne la rotation sur l'axe Z [4]. (r==Euler Z)



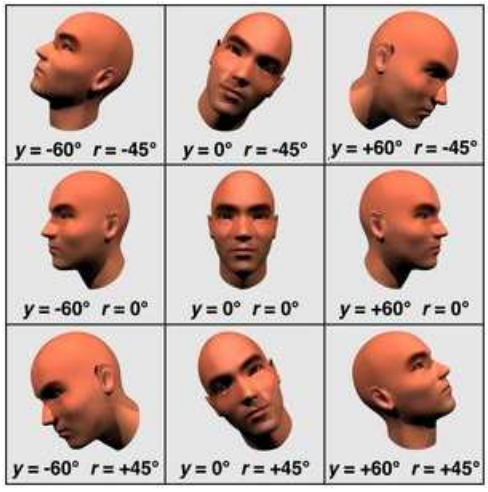


Figure 1 : Angles d'inclinaison du visage [5]

IV.4. Les points d'intérêts

L'API fournit la méthode getLandmarks() qui retourne une List d'objet important du visage avec leur coordonnée. Est détecté : Bas et haut de la bouche, joue gauche et droite, les deux oreilles, les deux pointes d'oreilles, les deux yeux, la gauche et la droite de la bouche et la base du nez.

Figure 2 : Points d'intérêt d'un visage [5]

IV.5. L'état du visage

L'API permet aussi une détection intelligente d'état du visage :

- getIsLeftEyeOpenProbability() : Retourne une valeur entre 0 et 1, de la probabilité que l'oeil gauche est ouvert
- getIsRighteyeOpenProbability() : Retourne une valeur entre 0 et 1, de la probabilité que l'oeil droit est ouvert
- getIsSmilingProbability() : Retourne une valeur entre 0 et 1, de la probabilité que le visage sourit [4]

IV.6. Classes à utiliser

Bitmap	//Pour transformer l'image d'un ImageView en bitmap Bitmap bitmap = ((BitmapDrawable)imgView.getDrawable()).**getBitmap**();
FaceDetector.Builder	FaceDetector.**Builder**(Context context); FaceDetector.Builder **setTrackingEnabled** (boolean trackingEnabled) FaceDetector.Builder **setLandmarkType** (int landmarkType) FaceDetector.Builder **setClassificationType** (int classificationType) FaceDetector **build**();
Frame	Frame frame = new Frame.Builder().setBitmap(bitmap).**build**();
FaceDetector	SparseArray<Barcode> **detect**(Frame frame); **ALL_LANDMARKS** **ALL_CLASSIFICATIONS**
SparseArray<Face>	int **size**() Face **valueAt**(int index);
Face	float **getEulerY**() float **getEulerZ**() float **getHeight**() int **getId**() float **getIsLeftEyeOpenProbability**() float **getIsRightEyeOpenProbability**() float **getIsSmilingProbability**() List<Landmark> **getLandmarks**() PointF **getPosition**() float **getWidth**()
Landmark	PointF **getPosition**() Retourne les coordonnées (x, y) du landmark avec (0, 0) est le point supérieur gauche de l'image. int **getType**() Retourne le type du landmark (BOTTOM_MOUTH LEFT_CHEEK LEFT_EAR LEFT_EAR_TIP LEFT_EYE LEFT_MOUTH, etc.).
PointF	float **x** float **y**

V. Reconnaissance des codes-barres

La nouvelle version des Google Play Services, portant le numéro de version 7.8, inclut diverses nouveautés, à commencer par une API dédiée à la reconnaissance des codes-barres.

Pour faire la reconnaissance des codes-barres d'une image, utiliser les classes suivantes :

Bitmap	//Pour transformer l'image d'un ImageView en bitmap Bitmap bitmap = ((BitmapDrawable)imgView.getDrawable()).**getBitmap**();
BarcodeDetector.Builder	BarcodeDetector.**Builder**(Context context); BarcodeDetector.Builder **setBarcodeFormats**(int format); BarcodeDetector **build**();
Frame	Frame frame = new Frame.Builder().setBitmap(bitmap).**build**();
BarcodeDetector	SparseArray<Barcode> **detect**(Frame frame);
SparseArray<Barcode>	int **size**() Barcode **valueAt**(int index);
Barcode	**ALL_FORMATS** EAN_13 EAN_8 QR_CODE URL … String **rawValue** : Valeur du code à barres telle qu'elle a été encodée dans le code à barres. int **format** : Format du code à barres, par exemple EAN_13. public Point[] **cornerPoints** : 4 points d'angle dans le sens des aiguilles d'une montre en commençant par le coin supérieur gauche. Rect **getBoundingBox**() : retourne la boîte englobante alignée sur l'axe du code à barres.
Rect	int **bottom** int **left** int **right** int **top**

Références :

[1] http://android-france.fr/2009/09/utiliser-en-developpement-le-service-vibreur-de-votre-telephone-android-dans-vos-applications/
[2] https://developer.android.com/training/camera/photobasics.html
[3] https://developer.android.com/training/camera/videobasics.html
[4] http://www.android-dev.fr/la_detection_de_visage_avec_les_mobilevision_api
[5] https://developers.google.com/vision/face-detection-concepts
[6] https://developers.google.com/vision/android/multi-tracker-tutorial
[7] http://www.frandroid.com/android/303792_google-play-services-7-8-api-dechiffrer-codes-barres
[8] https://developers.google.com/android/reference/com/google/android/gms/vision/face/package-summary
[9] https://developers.google.com/android/reference/com/google/android/gms/vision/barcode/package-summary