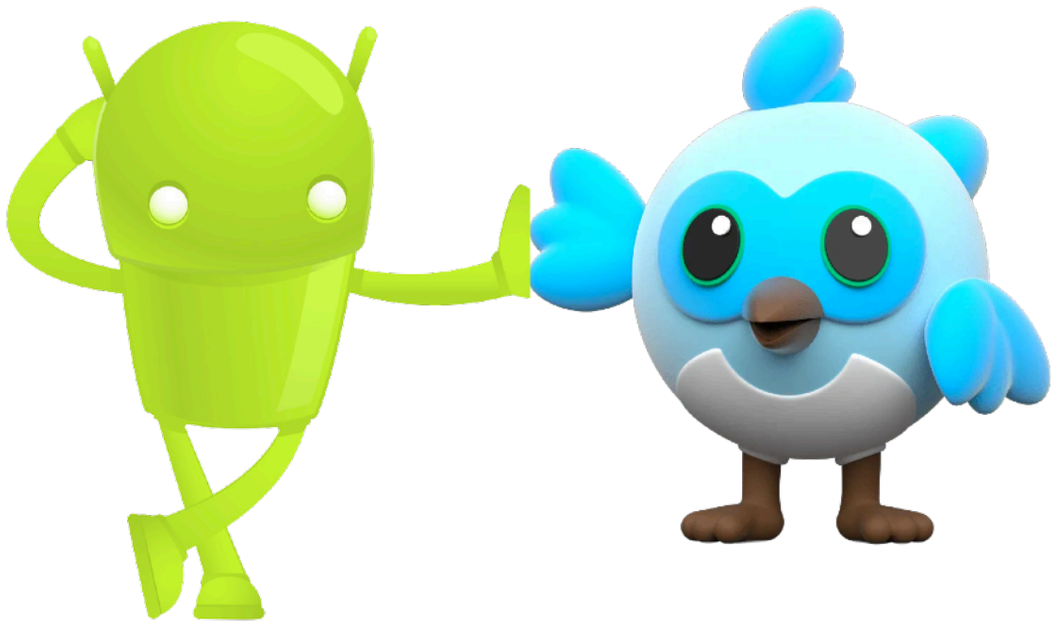
Technologies et Développement Mobile

Nadia Elouali
n.elouali@esi-sba.dz

2023-2024



Chapitre IV : Concepts avancés

- Programmation concurrente/Arrière plan
  - Services
  - Threads
- Fonctionnalités Android
  - Les capteurs
  - Téléphonie & SMS
  - Audio, vidéo & caméra

Concepts avancés

- Fonctionnalités Android
  - Les capteurs

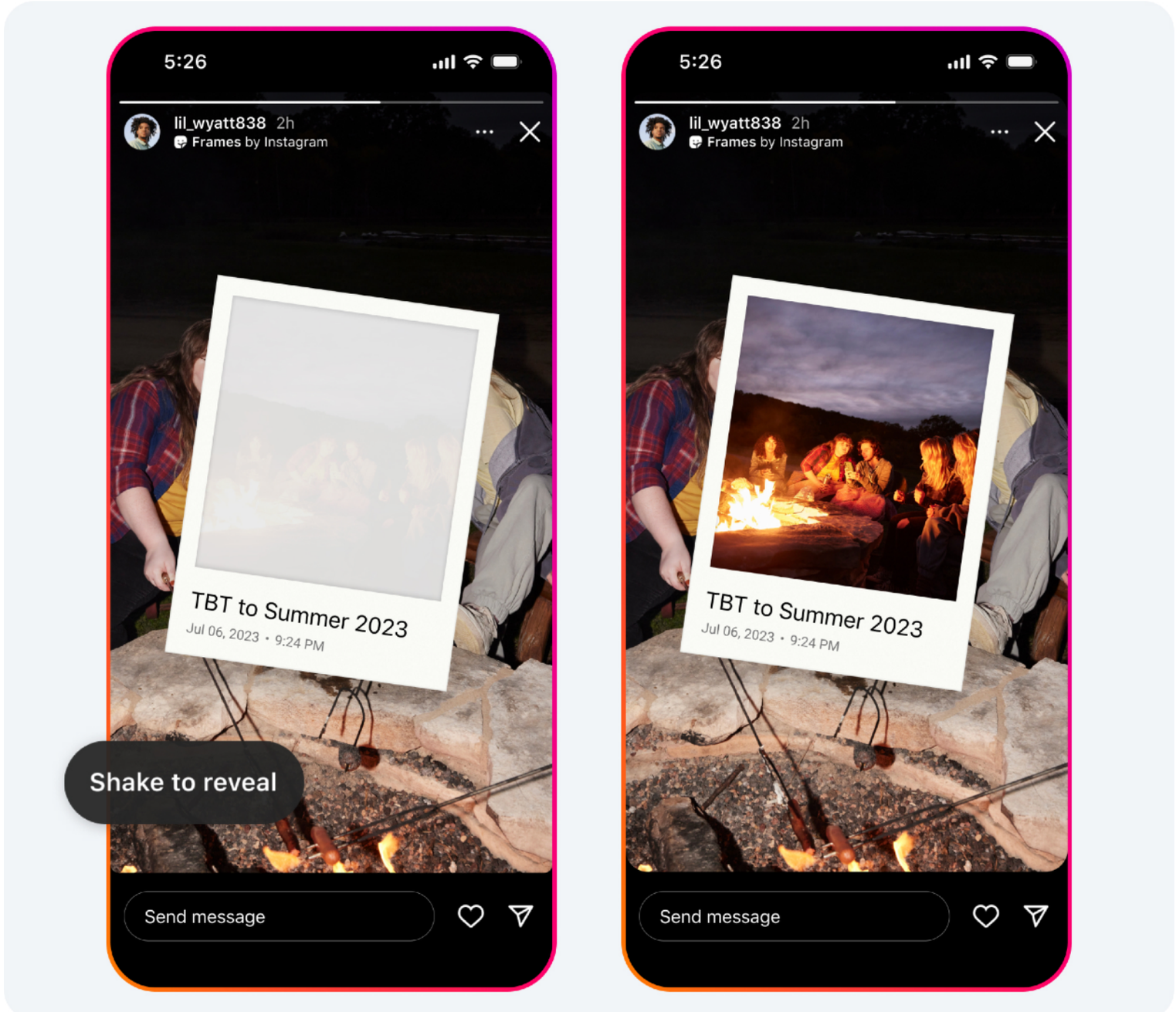
- La plupart des appareils sous Android disposent de capteurs intégrés qui mesurent le mouvement, l'orientation et les diverses conditions environnementales.

- Ces capteurs sont capables de fournir des données brutes avec une précision élevée, et sont utiles si on souhaite surveiller le mouvement ou le positionnement d'un appareil tridimensionnel, ou si on souhaite surveiller les changements de l'environnement ambiant à proximité d'un appareil.

- Par exemple, un jeu peut utiliser les capteurs pour déduire des gestes et des mouvements complexes de l'utilisateur, tels que l'inclinaison, le tremblement, la rotation ou le pivotement.

Concepts avancés

- Fonctionnalités Android
  - Les capteurs



Concepts avancés

- Fonctionnalités Android
  - Les capteurs

- D'un point de vue technique, on trouve deux types de capteurs :
  - Certains sont des composants matériels, c'est-à-dire qu'il y a un composant physique présent sur le terminal. Ils fournissent des données en prenant des mesures (l'accéléromètre par exemple).
  - Autres capteurs sont uniquement présents d'une manière logicielle. Ils se basent sur des données fournies par des capteurs physiques pour calculer des données nouvelles (le capteur d'orientation, par exemple).

Concepts avancés

- Fonctionnalités Android
  - Les capteurs

- Les différents capteurs sont représentés par une valeur dans la classe android.hardware.Sensor.

Nom du capteur	Valeur système	Type	Description	Utilisation typique
Accéléromètre	TYPE_ACCELEROMETER	Matériel	Mesure la force d'accélération appliquée au terminal sur les trois axes (x, y et z) (m/s^2).	Détecter les mouvements (secouer/ shake).
Gyroscope	TYPE_GYROSCOPE	Matériel	Mesure la vitesse de rotation sur chacun des trois axes en radian par seconde	Détecter l'orientation de l'appareil (vidéos à 360°)

Concepts avancés

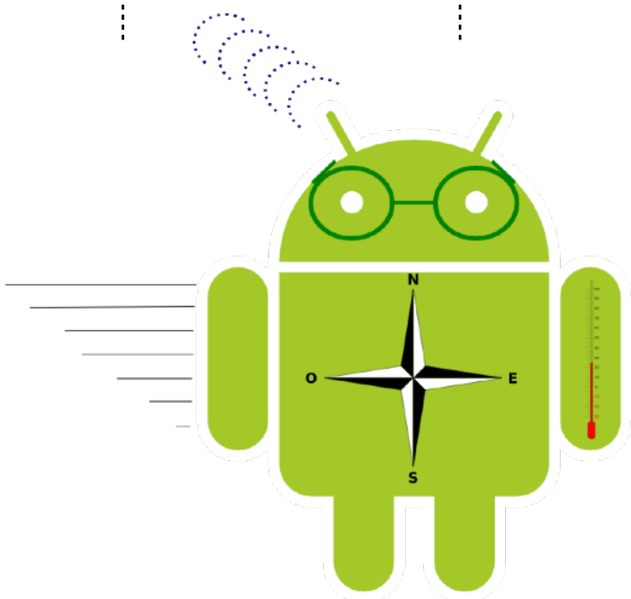
- Les capteurs

Photomètre	TYPE_LIGHT	Matériel	Mesure le niveau de lumière ambiante en lux (lx).	Détecter la luminosité pour adapter celle de l'écran de l'appareil.
Magnétomètre	TYPE_MAGNETIC_FIELD	Matériel	Mesure le champ géomagnétique sur les trois axes en microtesla (µT).	Créer une boussole.
Orientation	TYPE_ORIENTATION	Logiciel	Mesure le degré de rotation que l'appareil effectue sur les trois axes	Déterminer la position de l'appareil.
Baromètre	TYPE_PRESSURE	Matériel	Mesure la pression ambiante en hectopascal (hPa) ou millibar (mbar).	Surveiller les changements de pression de l'air ambiant.

Concepts avancés

- Fonctionnalités Android
  - Les capteurs

Capteur de proximité	TYPE_PROXIMITY	Matériel	Mesure la proximité d'un objet en centimètres (cm).	Détecter si l'utilisateur porte le téléphone à son oreille pendant un appel.
Thermomètre	TYPE_TEMPERATURE	Matériel	Mesure la température de l'appareil en degrés Celsius (°C).	Surveiller la température.



Concepts avancés

- Fonctionnalités Android
  - Les capteurs
- Utilisation des capteurs :

- Le Manifest : Demander la présence d'un capteur

- Si l'application n'ait aucun sens sans un capteur particulier, il faut indiquer dans le Manifest qu'un utilisateur sans ce capteur ne peut pas effectuer le téléchargement. Il faut ajouter une ligne de type <uses-feature> :

```
<uses-feature android:name="android.hardware.sensor.accelerometer"
                              android:required="true" />

<uses-feature android:name="android.hardware.sensor.proximity"
                              android:required="true" />
```

- android:required="true" sert à préciser que la présence de l'accéléromètre est absolument indispensable. S'il est possible d'utiliser l'application sans l'accéléromètre, mais qu'il est fortement recommandé, alors il suffit de mettre à la place android:required="false".

Concepts avancés

- Fonctionnalités Android
  - Les capteurs

- Uses-Permission vs Uses-Feature :

- <uses-permission> demande la permission de l'utilisateur pour faire une action (Localiser, etc.).

- <uses-feature> déclare la nécessité d'utiliser un matériel (camera, accéléromètre, etc.)/logiciel (bluetooth) particulier.

- Google Play utilise les éléments <uses-feature> pour filtrer les appareils qui ne répondent pas à ses exigences en termes de fonctionnalités matérielles et logicielles. Il présentera l'application uniquement aux utilisateurs dont les périphériques répondent aux exigences, plutôt que de la présenter à tous les utilisateurs.

Concepts avancés

- Fonctionnalités Android
  - Les capteurs

- Identifier les capteurs :

- La classe qui permet d'accéder aux capteurs est SensorManager. Pour en obtenir une instance, il suffit de faire :

```
SensorManager  sensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);
```

- Si on veut connaître la liste de tous les capteurs existants sur l'appareil, il faut utiliser la méthode List<Sensor> getSensorList(int type) avec type qui vaut Sensor.TYPE_ALL.

```
List<Sensor> liste = sensorManager.getSensorList(Sensor.TYPE_ALL);
for(int i=0;i<liste.size();i++) {Log.i("capteur "+i, " "+liste.get(i).getName());}
```

Concepts avancés

- Fonctionnalités Android
  - Les capteurs

- Identifier les capteurs (Galaxy S5) :

```
I/capteur 0:  MPU6500 Acceleration Sensor
I/capteur 1:  MPU6500 Gyroscope Sensor
I/capteur 2:  MPU6500 Uncalibrated Gyroscope Sensor
I/capteur 3:  AK09911C Magnetic field Sensor
I/capteur 4:  AK09911C Magnetic Sensor UnCalibrated
I/capteur 5:  Barometer Sensor
I/capteur 6:  TMG399X Proximity Sensor
I/capteur 7:  TMG399X RGB Sensor
I/capteur 8:  MPL Rotation Vector
I/capteur 9:  MPL Game Rotation Vector
I/capteur 10:  SAMSUNG Step Detector Sensor
I/capteur 11:  SAMSUNG Step Counter Sensor
I/capteur 12:  SAMSUNG Significant Motion Sensor
I/capteur 13:  Screen Orientation Sensor
I/capteur 14:  Orientation Sensor
I/capteur 15:  Gravity Sensor
I/capteur 16:  Linear Acceleration Sensor
```

Concepts avancés

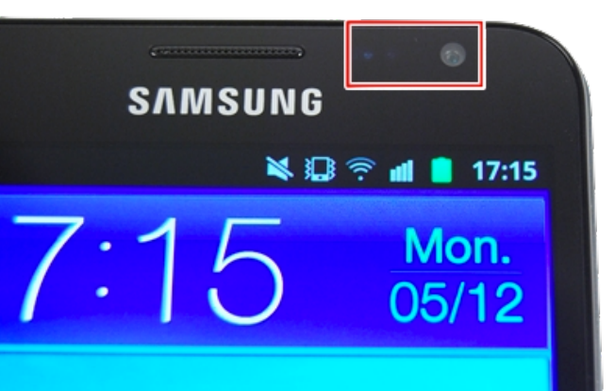
- Fonctionnalités Android
  - Les capteurs

- Identifier les capteurs (Huawei Y9) :

```
I/capteur 0:  accelerometer-kx023
I/capteur 1:  mag-akm09918
I/capteur 2:  orientation
I/capteur 3:  light-rpr531
I/capteur 4:  proximity-rpr531
I/capteur 5:  gyroscope
I/capteur 6:  gravity
I/capteur 7:  linear Acceleration
I/capteur 8:  rotation Vector
I/capteur 9:  significant Motion
I/capteur 10:  step counter
I/capteur 11:  geomagnetic Rotation Vector
I/capteur 12:  CAP_PROX sensor
I/capteur 13:  Game Rotation Vector Sensor
```

Concepts avancés

- Fonctionnalités Android
  - Les capteurs



DU SCREEN RECORDER

Concepts avancés

- Fonctionnalités Android
  - Les capteurs

```java
public class MainActivity extends AppCompatActivity implements SensorEventListener {
    SensorManager sensorManager;
    Sensor lightCap;
    ImageView visage;
    LinearLayout l1;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        sensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);
        lightCap = sensorManager.getDefaultSensor(Sensor.TYPE_LIGHT);
        l1=(LinearLayout)findViewById(R.id.l1);
        visage=(ImageView) findViewById(R.id.visage);
        visage.setImageResource(R.drawable.blanc);

    }
```

```xml
<ImageView
    android:id="@+id/visage"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:gravity="center"/>
```

Concepts avancés

- Fonctionnalités Android
  - Les capteurs

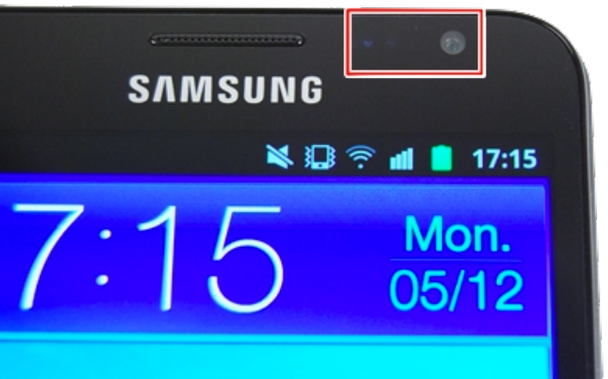
```java
@Override
public void onPause() {
    sensorManager.unregisterListener(this, lightCap);
    super.onPause();
}

@Override
public void onResume() {
    sensorManager.registerListener(this, lightCap,
                    sensorManager.SENSOR_DELAY_NORMAL);
    super.onResume();
}
```

```java
@Override
public void onSensorChanged(SensorEvent sensorEvent) {
    if (sensorEvent.sensor.getType() == Sensor.TYPE_LIGHT) {

        float x=sensorEvent.values[0];
        if(x==0) {visage.setImageResource(R.drawable.noir);
                l1.setBackgroundColor(Color.BLACK);}
        else {visage.setImageResource(R.drawable.blanc);
              l1.setBackgroundColor(Color.WHITE);}
    }
}
@Override
public void onAccuracyChanged(Sensor sensor, int i) {}
```

Concepts avancés

- Fonctionnalités Android
  - Les capteurs

```
<uses-feature
android:name="android.hardware.sensor.light"
    android:required="true"
    />
```



DU SCREEN RECORDER

Concepts avancés

- Fonctionnalités Android
  - Les capteurs

- Détection des changements des capteurs :

1

- Si un capteur a calculé une nouvelle valeur, la méthode onSensorChanged(SensorEvent event) est lancée :

  - SensorEvent indique à chaque fois quatre informations contenues dans quatre attributs :

    - l'attribut accuracy indique la précision de cette mesure (il peut avoir les mêmes valeurs que précédemment),
    - l’attribut sensor contient une référence au capteur qui a fait la mesure,
    - l'attribut timestamp est l'instant en nanosecondes où la valeur a été prise,
    - et enfin les valeurs sont contenues dans l'attribut values.

Concepts avancés

- Fonctionnalités Android
  - Les capteurs

- Détection des changements des capteurs :

1

- values est un tableau d'entiers.
- Si les valeurs correspondaient aux trois axes, alors le tableau a trois valeurs : values[0] est la valeur sur l'axe x, values[1] la valeur sur l'axe y et values[2] la valeur sur l'axe z.
- Si le calcul ne se fait pas sur trois axes, alors il n'y aura que values[0] qui contiendra la valeur.

Attention, cette méthode sera appelée très souvent, il ne faut donc pas effectuer d'opérations bloquantes à l'intérieur. Si on effectue des opérations longues, alors il se peut que la méthode soit à nouveau lancée alors que l'ancienne exécution n'avait pas fini ses calculs, ce qui va encombrer le processeur au fur et à mesure.

Concepts avancés

- Fonctionnalités Android
  - Les capteurs

- Détection des changements des capteurs :

2

- Un changement de précision du capteur avec la méthode onAccuracyChanged(Sensor sensor, int accuracy) avec :
  - sensor le capteur dont la précision a changé et
  - accuracy la nouvelle précision.

- La méthode OnAccuracyChanged est invoquée chaque fois qu'un capteur rapporte avec une précision différente. C’est utile lorsqu’on souhaite obtenir les données du capteur uniquement lorsque la précision est élevée, par exemple.

Concepts avancés

- Fonctionnalités Android
  - Les capteurs

- Détection des changements des capteurs :

2

- accuracy peut valoir
  - SensorManager.SENSOR_STATUS_ACCURACY_LOW pour une faible précision,
  - SensorManager.SENSOR_STATUS_ACCURACY_MEDIUM pour une précision moyenne,
  - SensorManager.SENSOR_STATUS_ACCURACY_HIGH pour une précision maximale,
  - SensorManager.SENSOR_STATUS_ACCURACY_UNRELIABLE s'il ne faut pas faire confiance à ce capteur.

Concepts avancés

- Fonctionnalités Android
  - Les capteurs

- Écouter les capteurs :

- Une fois l'interface écrite, il faut déclarer au capteur que nous sommes à son écoute.

- Pour cela, on va utiliser la méthode boolean registerListener(SensorEventListener listener, Sensor sensor, int rate) de SensorManager, avec
  - le listener,
  - le capteur dans sensor et
  - la fréquence de mise à jour dans rate.

- Il est possible de donner à rate les valeurs suivantes, de la fréquence la moins élevée à la plus élevée :

Concepts avancés

- Fonctionnalités Android
  - Les capteurs

- Écouter les capteurs :

- SensorManager.SENSOR_DELAY_NORMAL (0,2 seconde entre chaque prise);
- SensorManager.SENSOR_DELAY_UI (0,06 seconde entre chaque mise à jour, convient aux interfaces graphiques);
- SensorManager.SENSOR_DELAY_GAME (0,02 seconde entre chaque prise, convient aux jeux);
- SensorManager.SENSOR_DELAY_FASTEST (0 seconde entre les prises).

De manière générale, la meilleure pratique est d'avoir la valeur la plus lente possible, puisque c'est elle qui permet d'économiser le plus le processeur et donc la batterie. On peut même (à partir d'Android 9 de définir notre propre valeur en microsecond (2000000 microsecond ≈ 2s))

Concepts avancés

- Fonctionnalités Android
  - Les capteurs

- Désactiver l'écoute :

- Enfin, on peut désactiver l'écoute d'un capteur avec unregisterListener(SensorEventListener listener, Sensor sensor).

- Il ne faut pas oublier de désactiver les capteurs pendant que l'activité n'est pas au premier plan (donc il faut le désactiver pendant onPause() et le réactiver pendant onResume()), car le système ne le fera pas pour nous.

- De manière générale, désactivez les capteurs dès qu'on ne les utilisez plus.

Concepts avancés

- Fonctionnalités Android
  - Les capteurs

- Exemple plus significatif : Envoyer un SMS aux proches en cas de chute libre (pour les personnes âgées, ayant une épilepsie, etc.).

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

```
@Override
public void onSensorChanged(SensorEvent event) {
    if (event.sensor.getType() == Sensor.TYPE_ACCELEROMETER)
    {
        float x = event.values[0];
        float y = event.values[1];
        float z = event.values[2];

        double accelationVector = Math.sqrt(x*x + y*y + z*z);
        if (accelationVector < 1.2)
        {
            SmsManager manager = SmsManager.getDefault();
            manager.sendTextMessage("05554", null, "Aide-moi !", null, null);
        }
    }}
```

sentIntent (PendingIntent)

num d'envoi (null to use the current default)

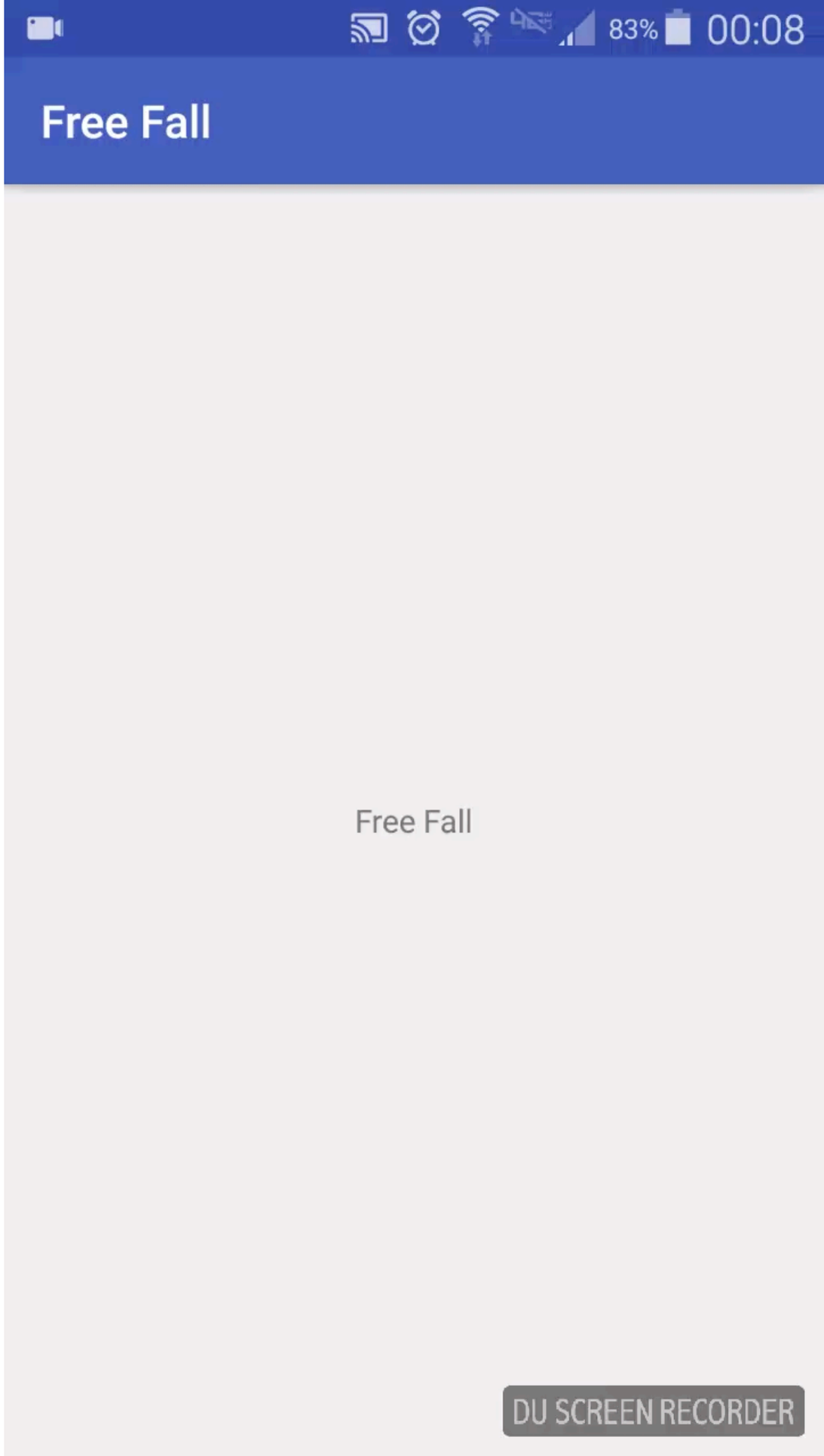
deliveryIntent

http://www.ejerm.com/vol2_jun_2015/img/pdf/cs013%20AUTOMATIC%20HUMAN%20FREE%20FALL%20DETECTION%20USING%20ANDROID.pdf

Concepts avancés

- Fonctionnalités Android
  - Les capteurs

« L'appareil est en chute libre si la longueur du vecteur donnée par l'accéléromètre est approximativement nulle. En théorie, il devrait être exactement égal à zéro, en pratique, il ne devrait être que proche de zéro »



Flutter

- Les capteurs sous Flutter :

- Comme « MethodChannel », « EventChannel » permet de communiquer avec l'application Flutter à l'aide de flux d'événements asynchrones.

- Donc, MethodChannel est utilisé pour « communiquer avec le code native en utilisant des appels de méthode asynchrones » alors que EventChannel est utilisé pour diffuser des données.

Flutter

- Les capteurs sous Flutter :

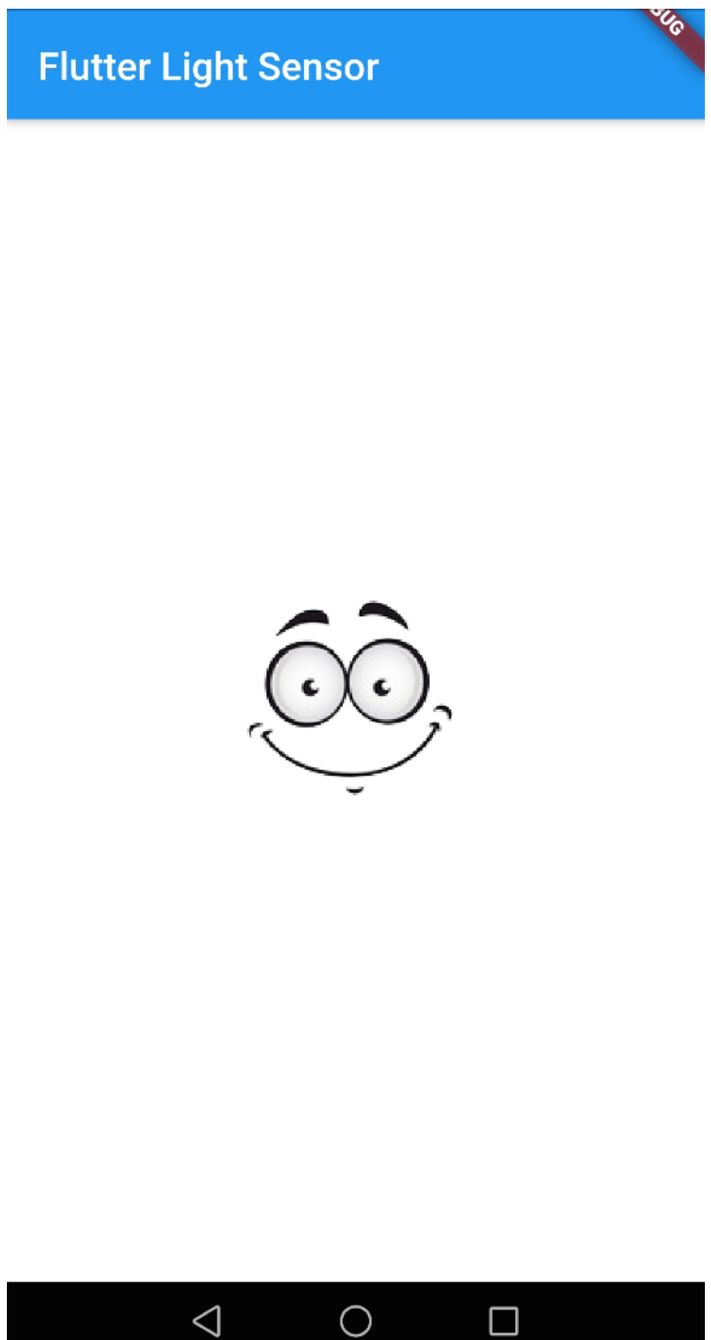
- Dans la partie Android native, on doit définir la méthode « configureFlutterEngine » afin de spécifier l'EventChannel ainsi que les deux méthodes : onListen et onCancel.

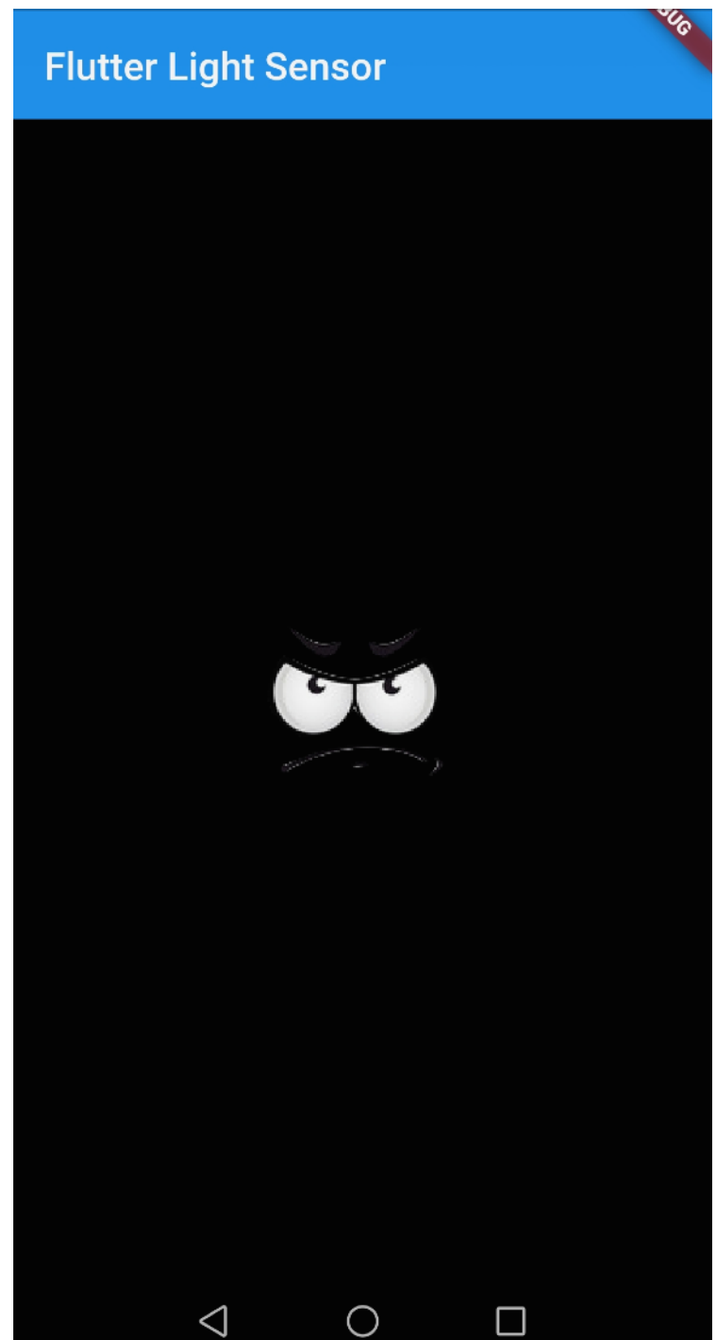
- Après, dans la partie Flutter, on doit s'abonner à cet « EventChannel » et l'écouter.

Flutter

- Les capteurs sous Flutter :

- L'exemple de **lightCap**





```kotlin
override fun configureFlutterEngine(@NonNull flutterEngine: FlutterEngine) {

    val sensorManager = context.getSystemService(Context.SENSOR_SERVICE) as
                                                                    SensorManager
    val sensor = sensorManager.getDefaultSensor(Sensor.TYPE_LIGHT)

    var listener: SensorEventListener? = null

   GeneratedPluginRegistrant.registerWith(flutterEngine)

   EventChannel(flutterEngine.dartExecutor.binaryMessenger,
      "lightnotif").setStreamHandler(object : EventChannel.StreamHandler {
       override fun onListen(args: Any?, events: EventChannel.EventSink) {
           listener = createSensorListener(events)
           sensorManager.registerListener(listener, sensor,
                                          SensorManager.SENSOR_DELAY_NORMAL)
       }

       fun createSensorListener(events: EventChannel.EventSink): SensorEventListener {
           return object : SensorEventListener {
               override fun onAccuracyChanged(sensor: Sensor, accuracy: Int) = Unit

               override fun onSensorChanged(event: SensorEvent) {
                   events.success(event.values[0])
               }
           }
       }

       override fun onCancel(args: Any?) {

       }})
}
```

Le listener du capteur

un objet qui permet d'envoyer des événements à la partie Flutter

in MainActivity (kotlin)

```
class _MyHomePageState extends State<MyHomePage> {
Color couleur=Colors.white;
String img="assets/blanc.png";

  //de Android vers Flutter : event avec le nom lightnotif qui écoute
  static const event = const EventChannel("lightnotif");
  void getLightValue () {
    event.receiveBroadcastStream().listen((data1) {
      setState(() {
        if(data1<20) {couleur=Colors.black;img="assets/noir.png";}
        else {couleur=Colors.white;img="assets/blanc.png";}
      });});};}
 @override
  void initState() {
    super.initState();
    getLightValue();}
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: couleur,
      appBar: AppBar(
        title: Text(widget.title),),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: <Widget>[
            Image.asset(img),],),),);}}
```

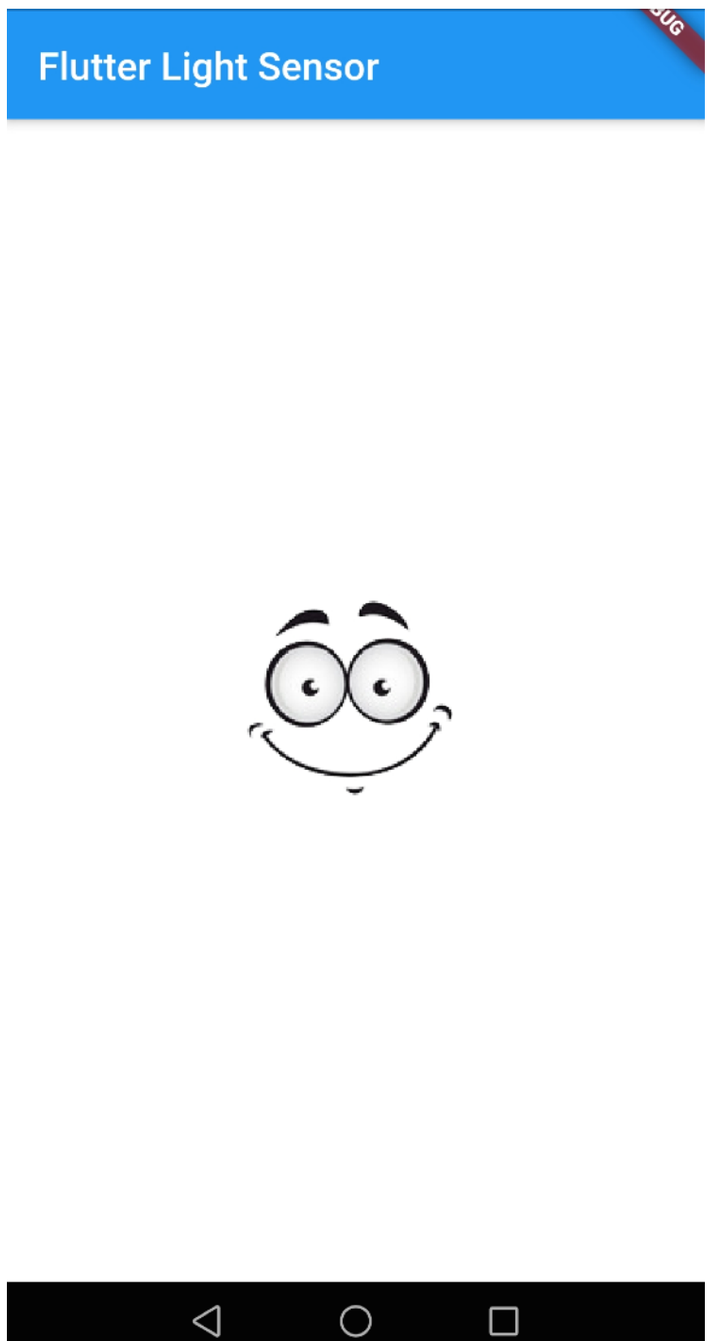
ou bien =0

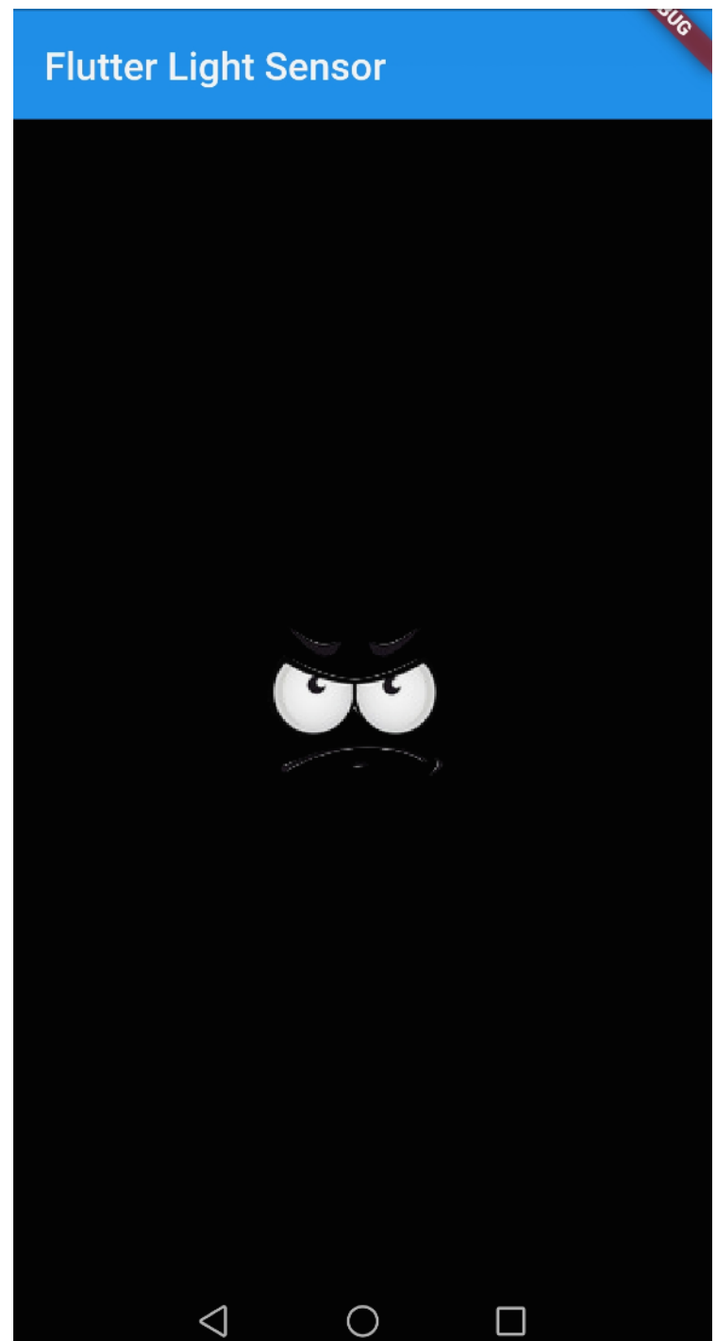
in main (dart)

Flutter

- Les capteurs sous Flutter :

- L'exemple de **lightCap**





Chapitre IV : Concepts avancés

- Programmation concurrente/Arrière plan
  - Threads
  - Services
- Fonctionnalités Android
  - Les capteurs
  - Téléphonie & SMS
  - Audio, vidéo & caméra

Concepts avancés

- Fonctionnalités Android
  - Téléphonie & SMS

- Les fonctions de téléphonie sont relativement simples à utiliser.

- L'état de la téléphonie est géré par la classe TelephonyManager qui permet de récupérer des informations concernant l'état du téléphone.

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

Concepts avancés

- Fonctionnalités Android
  - Téléphonie & SMS

Passer un appel

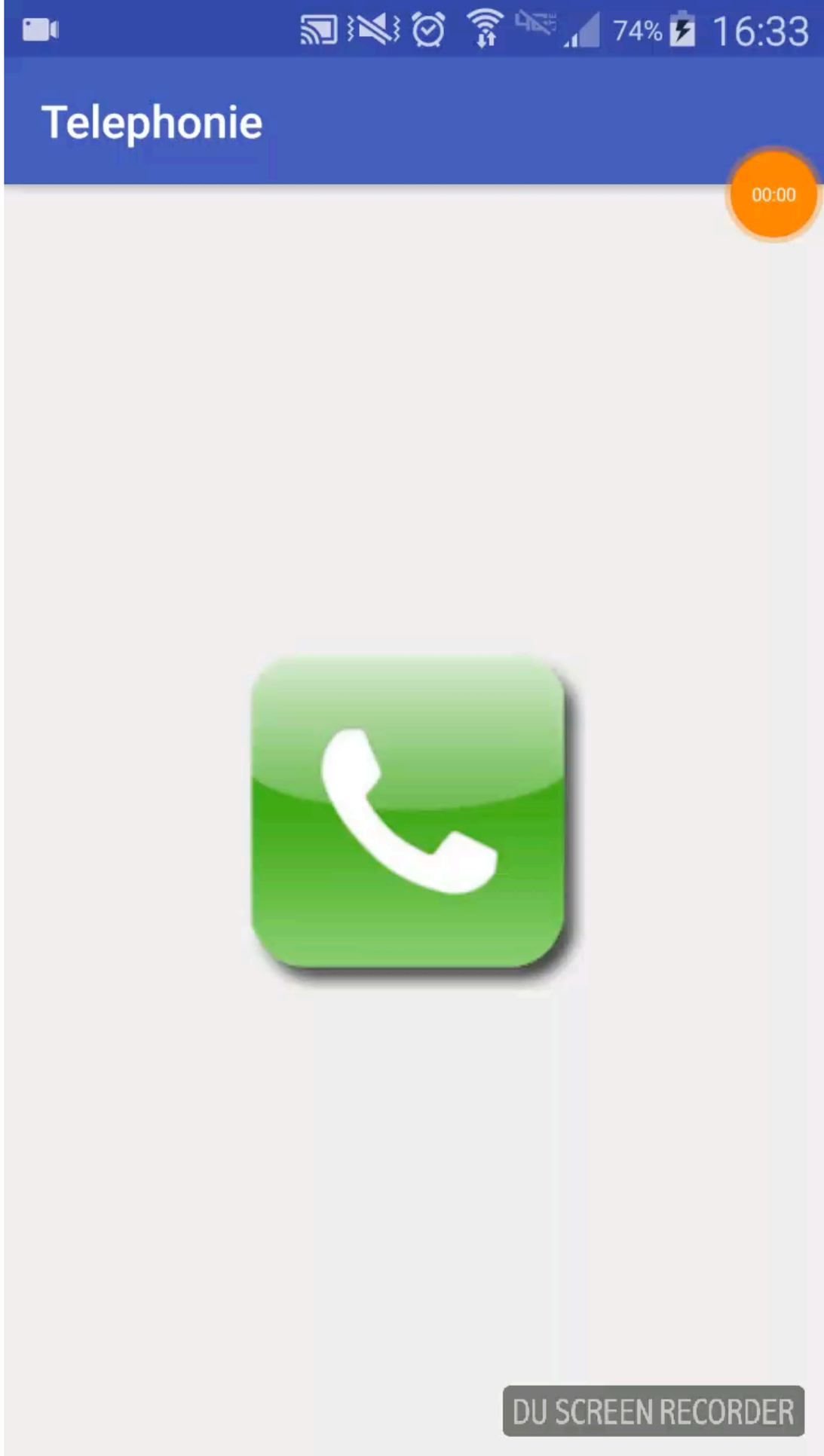
- Il est possible de passer un appel ou de déléguer l'appel, ces deux actions étant réalisées avec un Intent.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    Button c=(Button) findViewById(R.id.call);
    c.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Uri telnumber = Uri.parse("tel:0248484000");
            Intent call = new Intent(Intent.ACTION_CALL, telnumber);
            if (ActivityCompat.checkSelfPermission(getApplicationContext(),
            Manifest.permission.CALL_PHONE) != PackageManager.PERMISSION_GRANTED) {
            ActivityCompat.requestPermissions(MainActivity.this, new String[]
                                              {Manifest.permission.CALL_PHONE},1);

            }
            else {startActivity(call);}
        }
    });
}
```

Concepts avancés

- Fonctionnalités Android
  - Téléphonie & SMS

Passer un appel

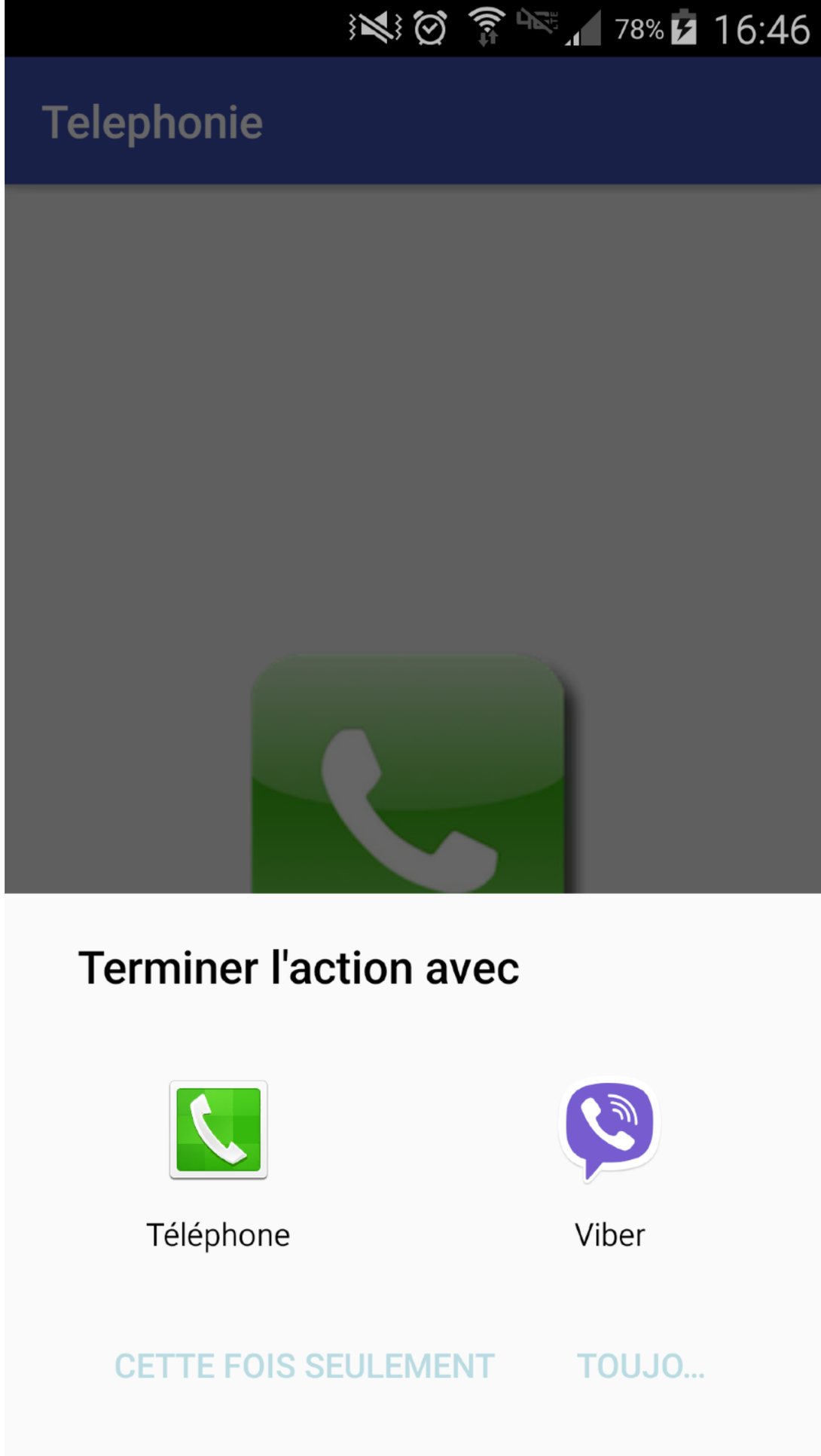


Concepts avancés

- Fonctionnalités Android
  - Téléphonie & SMS

Déléguer un appel

```
Uri telnumber =
Uri.parse("tel:0248484000");
Intent call = new
Intent(Intent.ACTION_DIAL, telnumber);
. . .
startActivity(call);
```



Concepts avancés

- Fonctionnalités Android
  - Téléphonie & SMS

- Même chose pour les SMSs :

Si la permission android.permission.SEND_SMS est disponible, il est possible d'envoyer des SMS au travers de SmsManager :

```
SmsManager manager = SmsManager.getDefault();
            manager.sendTextMessage("+5554", null, "Coucou !", null, null);
```

- Il est possible de créer un filtre d'Intent pour recevoir un SMS qui sera géré par un broadcast receiver. L'action à préciser dans le filtre d'Intent du receveur est android.provider.Telephony.SMS_RECEIVED

Chapitre IV : Concepts avancés

- Programmation concurrente/Arrière plan
  - Threads
  - Services
- Fonctionnalités Android
  - Les capteurs
  - Téléphonie & SMS
  - Audio, vidéo & caméra

Concepts avancés

- Fonctionnalités Android
  - Audio, vidéo & caméra

- Android donne la possibilité d'enregistrer des audios en utilisant MediaRecorder.

```
MediaRecorder recorder = new MediaRecorder();
 recorder.setAudioSource(MediaRecorder.AudioSource.MIC);
recorder.setOutputFormat(MediaRecorder.OutputFormat.THREE_GPP);
recorder.setAudioEncoder(MediaRecorder.AudioEncoder.AMR_NB);
recorder.setOutputFile(PATH_NAME);
recorder.prepare();
recorder.start();   // Recording is now started
...
recorder.stop();
recorder.reset();   // You can reuse the object by going back to setAudioSource() step
recorder.release(); // Now the object cannot be reused
```

★ **Note:** The Android Emulator cannot record audio. Be sure to test your code on a real device that can record.

Concepts avancés

- Fonctionnalités Android
  - Audio, vidéo & caméra

- Pour pouvoir enregistrer, l'application doit indiquer à l'utilisateur qu'elle accèdera à l'entrée audio de l'appareil. Une permission est donc nécessaire :

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

- Une autre permission est nécessaire pour l'enregistrement des records :

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

Concepts avancés

- Fonctionnalités Android
  - Audio, vidéo & caméra

```
public class MainActivity extends AppCompatActivity {
    private Button play, stop, record;
    private MediaRecorder myAudioRecorder;
    private String outputFile;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        play = (Button) findViewById(R.id.play);
        stop = (Button) findViewById(R.id.stop);
        record = (Button) findViewById(R.id.record);
        stop.setEnabled(false);
        play.setEnabled(false);
        outputFile = Environment.getExternalStorageDirectory().getAbsolutePath() + "/
                                    recording.3gp";
        myAudioRecorder = new MediaRecorder();
        myAudioRecorder.setAudioSource(MediaRecorder.AudioSource.MIC);
        myAudioRecorder.setOutputFormat(MediaRecorder.OutputFormat.THREE_GPP);
        myAudioRecorder.setAudioEncoder(MediaRecorder.OutputFormat.AMR_NB);
        myAudioRecorder.setOutputFile(outputFile);
```

Renvoie le répertoire de stockage principal (ou bien getExternalFilesDir)

Concepts avancés

- Fonctionnalités Android
  - Audio, vidéo & caméra

```
myAudioRecorder.setAudioSource(MediaRecorder.AudioSource.MIC);
```

Règle la source audio à utiliser pour l'enregistrement (le microphone).

```
myAudioRecorder.setOutputFormat(MediaRecorder.OutputFormat.THREE_GPP);
```

Définit le format du fichier de sortie produit pendant l'enregistrement (format de fichier multimédia 3GPP).

```
myAudioRecorder.setAudioEncoder(MediaRecorder.OutputFormat.AMR_NB);
```

Définit l'encodeur audio à utiliser pour l'enregistrement (format de fichier AMR NB -système de compression de données audio optimisée pour le codage de la parole-)

```
myAudioRecorder.setOutputFile(outputFile);
```

Pour enregistrer le fichier en sortie dans le bon endroit

Concepts avancés

- Fonctionnalités Android
  - Audio, vidéo & caméra

```
record.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        try {
            myAudioRecorder.prepare();
            myAudioRecorder.start();
        } catch (IllegalStateException ise) {
            // faire qlq chose
        } catch (IOException ioe) {
            // faire qlq chose
        }
        record.setEnabled(false);
        stop.setEnabled(true);
        Toast.makeText(getApplicationContext(), "Recording started",
                       Toast.LENGTH_LONG).show();
    }
});
```

Concepts avancés

- Fonctionnalités Android
  - Audio, vidéo & caméra

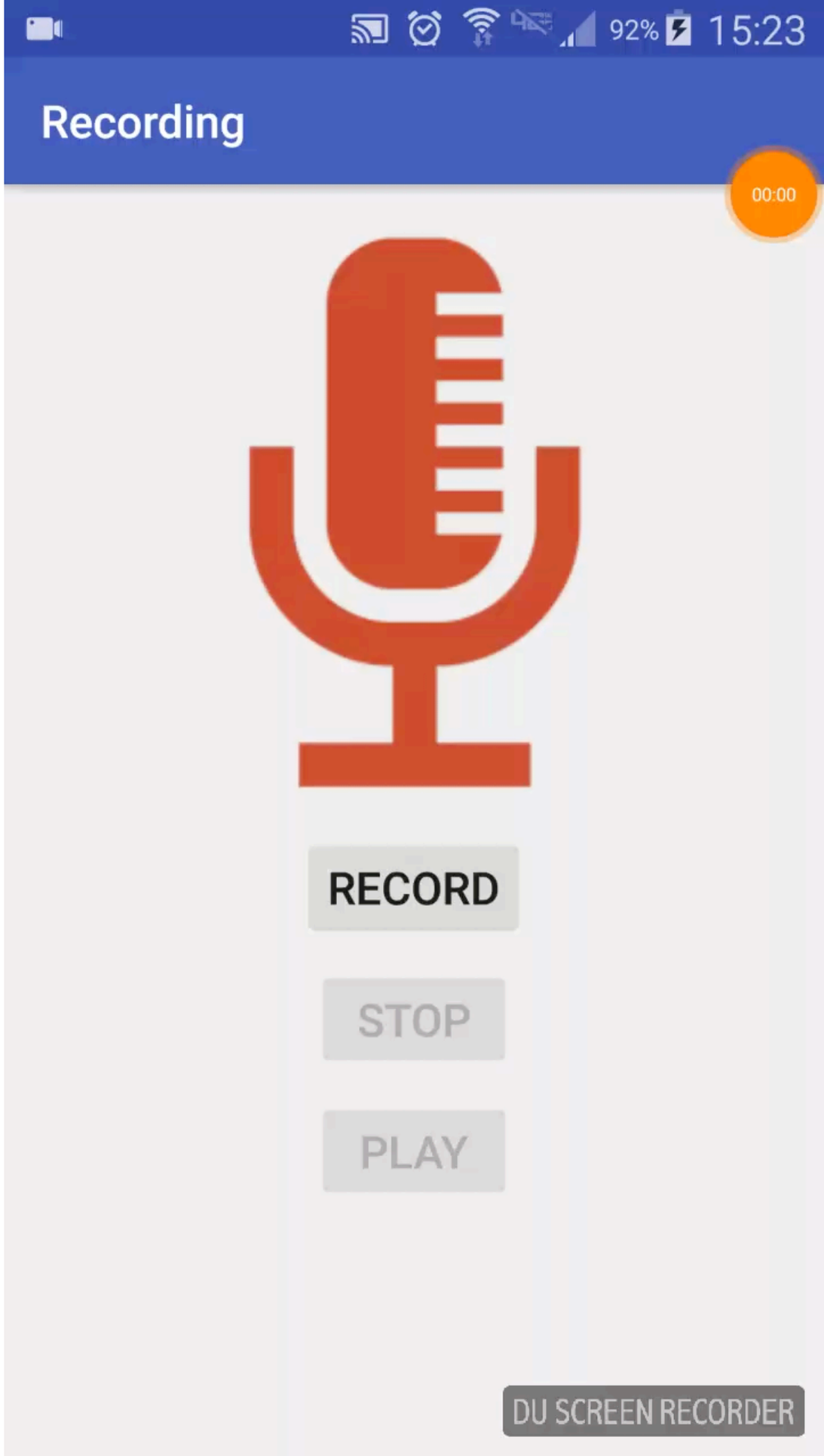
```
stop.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        myAudioRecorder.stop();
        myAudioRecorder.release();
        myAudioRecorder = null;
        record.setEnabled(true);
        stop.setEnabled(false);
        play.setEnabled(true);
        Toast.makeText(getApplicationContext(), "Audio Recorder successfully",
                                    Toast.LENGTH_LONG).show();
    }});
```

Libère les ressources associées à cet objet

```
play.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        MediaPlayer mediaPlayer = new MediaPlayer();
        try {mediaPlayer.setDataSource(outputFile);
            mediaPlayer.start();
            Toast.makeText(getApplicationContext(), "Playing Audio",
                Toast.LENGTH_LONG).show();
            } catch (Exception e) {}}});
```

Concepts avancés

- Fonctionnalités Android
  - Audio, vidéo & caméra



Concepts avancés

- Fonctionnalités Android
  - Audio, vidéo & caméra

```
if (ActivityCompat.checkSelfPermission(getApplicationContext(),
        Manifest.permission.RECORD_AUDIO) != PackageManager.PERMISSION_GRANTED) {

    ActivityCompat.requestPermissions(MainActivity.this, new String[]
            {Manifest.permission.RECORD_AUDIO,
                    Manifest.permission.WRITE_EXTERNAL_STORAGE},1);

}
```

Il ne faut pas oublier de vérifier les permissions dans l'activité aussi.

Concepts avancés

- Fonctionnalités Android
  - Audio, vidéo & caméra

Layout

Créer une application qui prendra une photo.

```
<RelativeLayout. . .>

    <ImageView…/>

    <Button
        android:id="@+id/button_image"
        android:layout_width="wrap_content"
        android:layout_height="48dp"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true"
        android:onClick="takePicture"
        android:text="Take a picture!"/>

</RelativeLayout>
```

Concepts avancés

- Fonctionnalités Android
  - Audio, vidéo & caméra

Activité

```
public void takePicture(View view) {

   Intent takePictureIntent = new
                    Intent(MediaStore.ACTION_IMAGE_CAPTURE);
    file = Uri.fromFile(getOutputMediaFile());
     takePictureIntent.putExtra(MediaStore.EXTRA_OUTPUT, file);

try {
    startActivityForResult(takePictureIntent, 1);

   } catch (ActivityNotFoundException e) {
    // display error state to the user
   }

}
```

En cas ou la caméra n'est pas disponible

C'est le code de la sous-activité

Concepts avancés

- Fonctionnalités Android
  - Audio, vidéo & caméra

Activité

```
private static File getOutputMediaFile(){
    File mediaStorageDir = new
File(Environment.getExternalStoragePublicDirectory(
            Environment.DIRECTORY_PICTURES), "CameraDemo");

    if (!mediaStorageDir.exists()){
        if (!mediaStorageDir.mkdirs()){
            return null;
        }
    }

    String timeStamp = new
SimpleDateFormat("yyyyMMdd_HHmmss").format(new Date());
    return new File(mediaStorageDir.getPath() + File.separator +
            "IMG_"+ timeStamp + ".jpg");

}
```

Concepts avancés

- Fonctionnalités Android
  - Audio, vidéo & caméra

```
<uses-feature android:name="android.hardware.camera"
    android:required="true" />
```

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

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

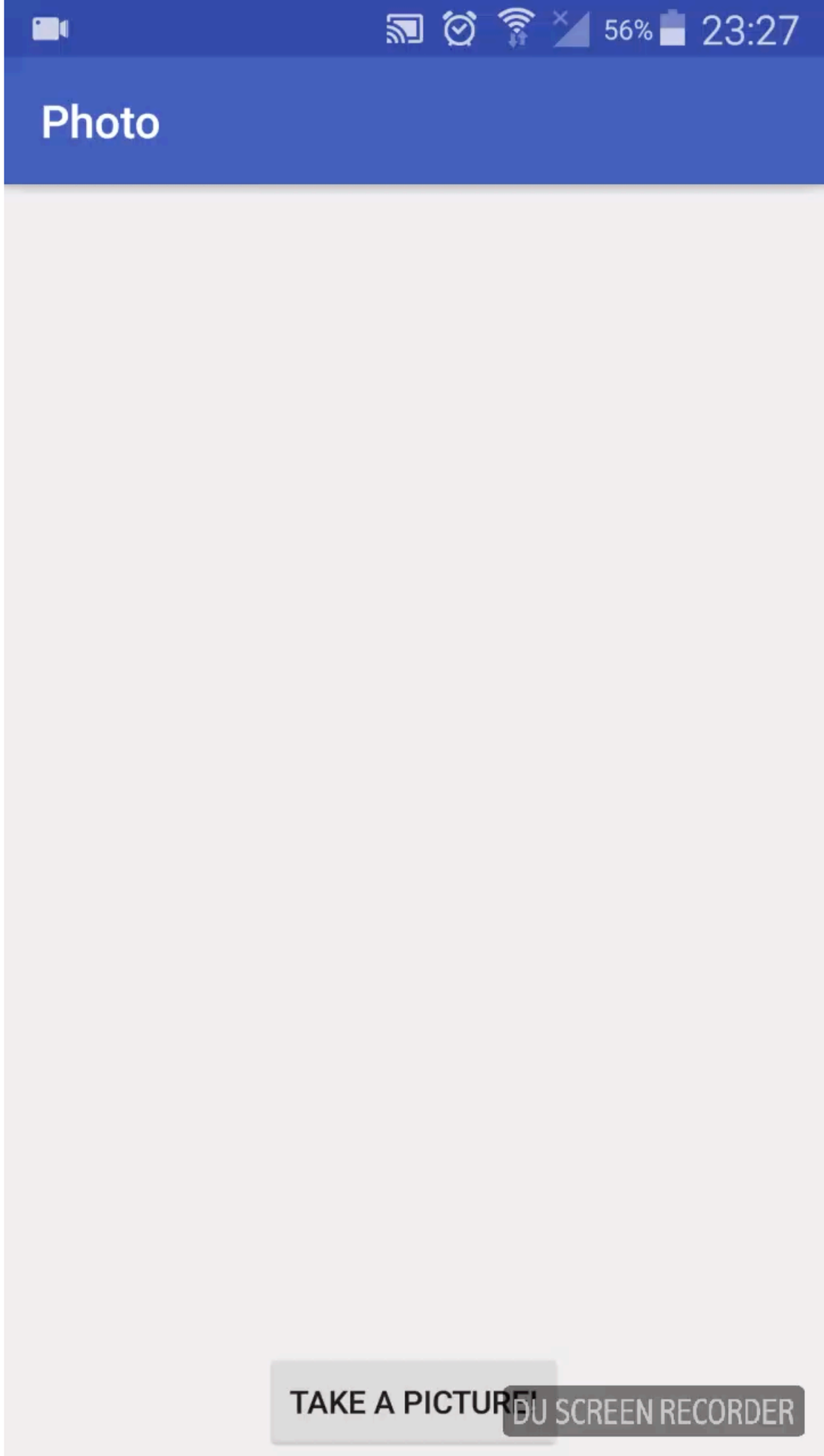
```
if ((ActivityCompat.checkSelfPermission(getApplicationContext(),
                Manifest.permission.CAMERA) !=
                        PackageManager.PERMISSION_GRANTED) &&

   (ActivityCompat.checkSelfPermission(getApplicationContext(),
                Manifest.permission.WRITE_EXTERNAL_STORAGE) !=
                        PackageManager.PERMISSION_GRANTED)) {

    ActivityCompat.requestPermissions(MainActivity.this, new String[]


{Manifest.permission.CAMERA,Manifest.permission.WRITE_EXTERNAL_STORAGE},200);}
```

Concepts avancés

- Fonctionnalités Android
  - Audio, vidéo & caméra

- Il faut utiliser le « FileProvider » pour récupérer le lien vers la photo prise (https://developer.android.com/reference/androidx/core/content/FileProvider)

- Même principe pour les vidéos : MediaStore.ACTION_VIDEO_CAPTURE



Flutter

Pour SmsManager, TelephonyManager, etc. il y a toujours des plugins correspondants. Sinon, on peut passer par « MethodChannel ». Pour prendre photos/vidéos flotter propose CameraController.

Chapitre IV : Concepts avancés

- Programmation concurrente/Arrière plan
  - Threads
  - Services
- Fonctionnalités Android
  - Les capteurs
  - Téléphonie & SMS
  - Audio, vidéo & caméra

THE END