Rapport du TP2

Introduction

Le projet a été fait sous Android Studio en utilisant mon Honor 9 comme appareil Android pour exécuter les projets.

Description du tp

Ce tp est composé de 7 exercices différents. Tous les 7 projets sont contenus dans un seul dossier compressé pour faciliter le rendu. Nous ne parlerons pas de tous les projets ici car certains sont assez similaires au niveau du fonctionnement donc inutile d'expliquer plusieurs fois la meme chose, et de plus, les premiers exercices sont déjà dans le cours de l'UE.

Les projets sur les capteurs

Il s'agit des 6 premiers exercices. Ces exercices consiste à manipuler plusieurs capteurs différents et à en manipuler les données. La structure du code en est donc plutôt similaire. En effet, la seule partie qui change vraiment est la méthode héritée de l'interface SensorEventListener : onSensorChanged().

```
1  @Override
2      public void onSensorChanged(SensorEvent sensorEvent) {
3          float dist;
4          if (sensorEvent.sensor.getType() == proxySensor.getType()) {
5              dist = sensorEvent.values[0];
6              if (dist>0.1) {
7                  near.setVisibility(View.INVISIBLE);
8                  away.setVisibility(View.VISIBLE);
9              }
10             else {
11                 near.setVisibility(View.VISIBLE);
12                 away.setVisibility(View.INVISIBLE);
13             }
14         }
15     }
```

On peut voir ci-dessus la méthode onSensorChanged() de l'application de proximité. Elle va, en fonction de la valeur récupérée par le capteur de proximité, afficher une image ou une autre. View.VISIBLE et View.INVISIBLE servent à justement rendre visible ou invisible l'image.

Maintenant, voici la même méthode mais pour l'application de la lampe torche. Le principe est que lorsque l'utilisateur secoue l'appareil, la lampe torche s'allume et il faut qu'il le secoue une fois de plus pour l'éteindre.

```
1  @Override
2      public void onSensorChanged(SensorEvent sensorEvent) {
3          float x, y, z;
4          if (sensorEvent.sensor.getType() == gyroSensor.getType()) {
5              x = sensorEvent.values[0];
6              y = sensorEvent.values[1];
7              z = sensorEvent.values[2];
8
9              if (x>2 || y>2 || z>2){
10                 if (on_off)
11                     on_off = turnOff(on_off);
12                 else
13                     on_off = turnOn(on_off);
14             }
15         }
16     }
```

On peut remarquer les fontions turnOn() et turnOff(). Ces fonctions vont nous servir à savoir si on doit allumer ou éteindre la lampe en leur donnant un booléen (on_off ici). Ce booléen permet de savoir si la lampe est actuellement allumée ou éteinte. Une fois que l'une des deux fonctions s'est exécutée, ce booléen change de valeur.

```
public boolean turnOn(boolean on_off){
    String cameraId = null;
    try {
        cameraId = torch.getCameraIdList()[0];
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
            torch.setTorchMode(cameraId, true);
        }
        on_off = true;
    } catch (CameraAccessException e) {
        e.printStackTrace();
    }
    return on_off;
}

public boolean turnOff(boolean on_off){
    String cameraId = null;
    try {
        cameraId = torch.getCameraIdList()[0];
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
            torch.setTorchMode(cameraId, false);
        }
        on_off = false;
    } catch (CameraAccessException e) {
        e.printStackTrace();
    }
    return on_off;
}
```

Application de Géolocalisation

Cette application nous permet de savoir nos coordonnées GPS, c'est à dire la latitude et la longitude. Comme nous n'utilisons pas un capteur natif de l'appareil comme l'accélerometre par exemple, mais plutôt la connection avec un GPS ou internet, son fonctionnement diffère quelques peu des autres applications de ce tp. Ici on utilise une classe LocationManager qui nous permet d'avoir accès à un système de localisation pour avoir la localisation de l'appareil, mais aussi d'être informé des changements dans la localisation si l'appareil s'est déplacé entre temps. Pour cela, on utilise la méthode onLocationChanged()

```
1       @Override
2       public void onLocationChanged(@NonNull Location location) {
3           double lat = location.getLatitude() ;
4           double lon = location.getLongitude();
5           System.out.println(lat + " / " + lon);
6           latitude.setText("latitude: " + lat + "");
7           longitude.setText("longitude: " + lon + "");
8       }
```

A Noter

Afin que l'application de localisation fonctionne, il faut au préalable donner la permission à celle-ci, dans les paramètres du téléphone, d'accéder à la localisation de l'appareil.