Introduction to App Development for Android

Kompass App

Prof. Dr. Paul Rawiel

An App that points to north

- New Project with an empty Activity
- Projektname: Kompass
- Packagename: de.cursoufpr.kompass
- select minimum SDK API

→ create the project

Project structur as shown

Important nodes

- java for Code
- res for Layout

Sensors

- Which sensor are available in your smartphone?
  - inclination?
  - camera
  - Accelerations?
  - Rotations?
  - …
- What kind of sensor is needed for a Kompass?
  - How can one get data from a sensor?
  - A Listener is needed?

SensorManager

- Central Android service to administer all sensors
- → we need a reference to that service
- (SensorManager)getSystemService(Context.SENSOR_SERVICE)
- Then we need special sensors
  - sensorManager.getDefaultSensor(Sensor.TYPE_MAGNETIC_FIELD);
  - sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
- Why these two sensors?

Read sensor data

- There is no method sensor.getData()
- such a method would have to be called in regular time periods
  - Not efficient
  - Energy intensive
- instead: the sensor sends a notification if there is something new
- What can be followed from this?

→We need a Listener that registers this

→SensorEventListener

Sensordaten auslesen

- If the App is in the background → no sensor data needed
- The correct way:
  - ➢Subscribe to sensor events, when Activity is activated
  - ➢Sensor event happens, draw the compass needle in the correct angle
  - ➢When the Activity is stopped, the sensor event subscription can be stopped too

SensorEventListener

- Musst be registered
- Only when the App is in the foreground
- always when an App is pushed to the foreground, the method `onResume()is` called
- in this method the Listener is linked to the two sensors
  - `sensorManager.registerListener(this, accelerometer,SensorManager.SENSOR_DELAY_GAME);`
  - respectively for the magnetic field sensor

SensorEventListener

- Can be unregisterd when the App goes to the background
- allways when an App leaves the foreground, the method `onPause()` is called
- in this method the Listener for the two sensors is unregistered
  - `sensorManager.unregisterListener(this);`
  - `super.onPause();`
- The key method: onSensorChanged(Sensor event){ … }

The Compass needle

- Many possibilities
- We draw one
- Define an own public View that can be rotated

```
public class CompassneedleView extends View { … }
```

- What does the class need?
  - a Constructor
  - an attribute for the rotation angle and the possibility to set it
  - a `Canvas` on which the needle can be drawn

The class CompassneedleView

```
private float angle = 0;
private Paint paintcolor = new Paint();

public CompassneedleView(Context context) {
        super(context);

         paintcolor.setAntiAlias(true);
         paintcolor.setColor(Color.WHITE);
         paintcolor.setStyle(Paint.Style.FILL);
    }
```

Die Klasse KompassnadelView

```
public void setAngle(float angle) {
        this.angle = angle;
        invalidate();
    }
```

this method setAngle sets the angle in which later, the View will be drawn

The line invalidate() tells Android, that the latest drawn content of the View is outdated and the View is drawn again

The onDraw method

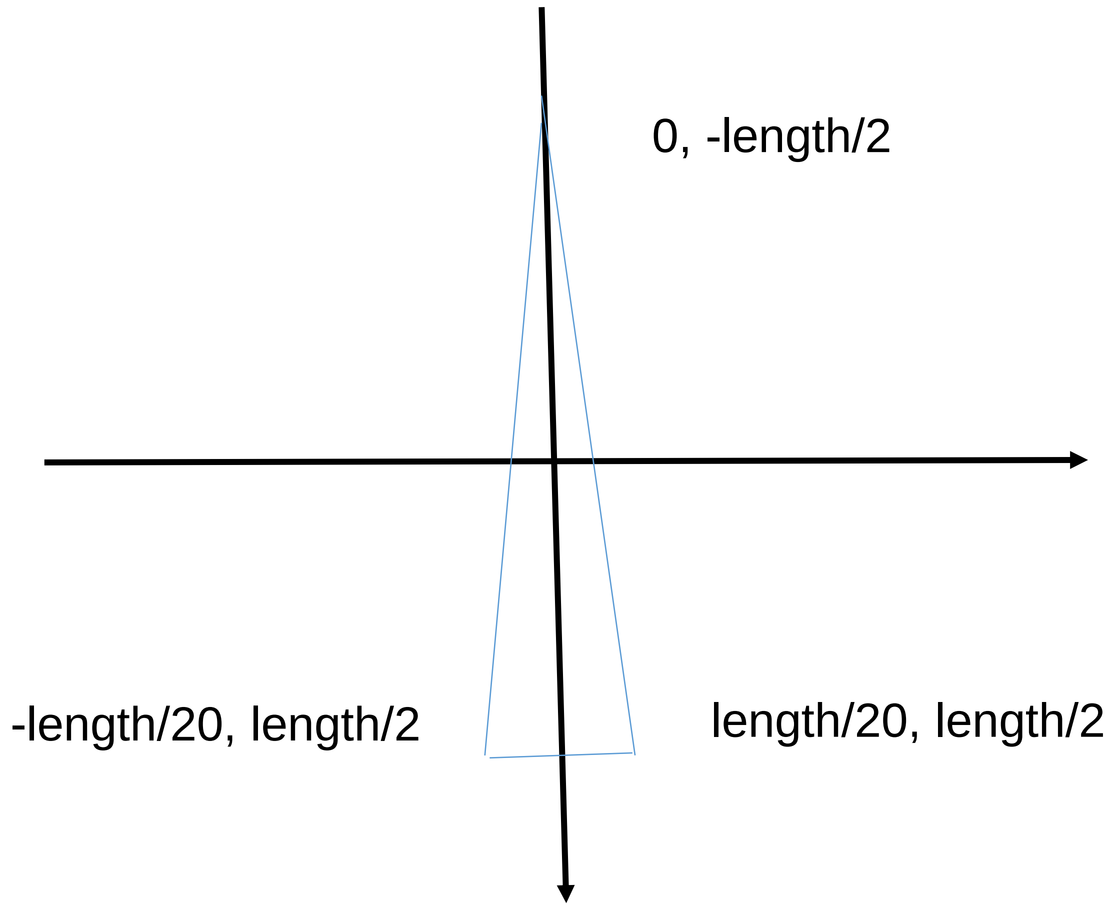
in this Methode width and height of the canvas are determined

```
    int width = canvas.getWidth();
    int height = canvas.getHeight();
```

The length of the Compass needle shall be equal to the lower of these two values

Then, a `Path`-Object has to be constructed, for the path of the virtual paint brush

Path – the Compass needle



Path – the Compass needle

```
Path pfad = new Path();
pfad.moveTo(0,-length/2);
pfad.lineTo(length/20,length/2);
pfad.lineTo(-length/20,length/2);
pfad.close();
```

Still before drawing the View

By default the center of rotation is the lower left corner of the canvas

therefore:

```
canvas.translate(width/2,height/2);
```

Rotate the canvas with

```
canvas.rotate(angle);
```

Then

```
canvas.drawPath(pfad, paintcolor);
```

Connect View and Activity

- In the onCreate method of the Activity

```
    view = new CompassneedleView(this);
```

- Set this View as content of the Activity

```
    setContentView(view);
```

- The correct angle of the View is set in the onSensorChanged-Methode

The onSensorChanged Method

```
public void onSensorChanged(SensorEvent event) {
switch(event.sensor.getType()) {
        case Sensor.TYPE_MAGNETIC_FIELD:
                gm = event.values.clone();
                break;
        case Sensor.TYPE_ACCELEROMETER:
                gr = event.values.clone();
                break;
            }
        if (gm!=null && gr!=null) {
                SensorManager.getRotationMatrix(rot, ink, gr, gm);
                SensorManager.getOrientation(rot, orientationValues);
                view.setWinkel(-orientationValues[0]*RHO_DEG);
            }
        }
```

That was it …

- the Compass can be tested
- Annything annoying?
- What can be improved?