App Quest 2019

Morse Encoder

In dieser Aufgabe werden wir eine App schreiben, welche mit Hilfe eines Lichtsignals einen Morse-Code verschickt. Am Treasure Hunt wird es ein Gerät mit einer Empfänger-App geben, welches die Lösung für das Logbuch anzeigt, sobald es den richtigen Notruf als Morse-Nachricht empfangen hat. Den richtigen Notruf findet ihr jeweils an den Posten.

Als erstes müssen wir uns aber mit dem Morse-Code vertraut machen. Hier ist das Wichtigste für diese Aufgabe zusammengefasst: Es gibt 5 Elemente, woraus der Morse-Code zusammengesetzt ist:

- Ein kurzes helles "Dit", welches hier als "·" repräsentiert wird.
- Ein langes helles "Dah" der Länge von drei Dits, welches hier als "-" repräsentiert wird.
- Eine kurze dunkle Pause zwischen Dits und Dahs von der Länge eines Dits, welche hier mit "" (leerer String) repräsentiert wird.
- Eine mittlere dunkle Pause zwischen Buchstaben bzw. Morsezeichen der Länge von drei Dits, welche hier mit " " (Leerzeichen) repräsentiert wird.
- Eine lange dunkle Pause zwischen Wörtern der Länge von sieben Dits, welche hier mit " / " repräsentiert wird.

Aus Dits, Dahs und kurzen Pausen werden Morsezeichen zusammengesetzt (wir verwenden ausschliesslich Buchstaben und KEINE Zahlen und Sonderzeichen):

Buchstabe	Morsezeichen
A	·-
B	-···
C	-·-·
D	-··

E	·
F	··-·
G	--·
H	····
I	··
J	·---
K	-·-
L	·-··
M	--
N	-·
O	---
P	·--·
Q	--·-
R	·-·
S	···
T	-
U	··-
V	···-
W	·--
X	-··-
Y	-·--
Z	--··

Aus diesen Morsezeichen und der mittleren und langen Pause werden Wörter und schliesslich ganze Sätze gebildet. Als Beispiel nehmen wir die beiden Wörter

```
APP QUEST
```

Diese würden wie folgt repräsentiert:

```
·– ·–– ·–– / ––·– ··– · ··· –
```

Bei uns soll ein Dit die Länge von einer halben Sekunde (500ms) haben. Vor und nach der Übertragung soll es jeweils dunkel sein, damit die Empfänger-App die Nachricht erkennen kann.

Da wir nun verstehen, wie der Code funktioniert, brauchen wir also nur noch eine Lichtquelle und ein gutes Timing! Als Lichtquelle können wir einfach eine rechteckige View verwenden, die wir abwechslungsweise auf schwarz (dunkel) oder weiss (hell) setzen. Ein gutes Timing ist etwas komplizierter: Weder iOS noch Android sind real-time Operating Systems, d.h. es gibt keine Garantie, dass ein Timer wirklich genau zu der gewünschten Zeit ausgeführt wird. Viele kleine Fehler im Timing können sich rasch addieren und dazu führen, dass Sender und Empfänger nicht mehr synchron arbeiten. Deshalb ist es bei dieser Aufgabe wichtig, den Zeitpunkt beim Start der Übertragung festzuhalten und anschliessend die ganze Übertragung relativ zu diesem Zeitpunkt zu takten. Der Ausführungszeitpunkt jedes einzelnen Signals bleibt zwar weiterhin ungenau, diese Ungenauigkeiten können sich so aber nicht addieren.

App testen

Weil wir euch die Notrufe für den Treasure Hunt noch nicht verraten wollen und ihr trotzdem in der Lage sein sollt, eure App zu testen, haben wir sowohl für iOS wie auch für Android einen Morse-Decoder implementiert, der die folgende Notruf-Nachricht akzeptiert:

```
THIS IS A TEST
```

Ihr braucht zum Testen also zwei Mobiltelefone. Die Encoder und Decoder von iOS und Android sind aber kompatibel, so dass ihr z.B. nicht zwingend zwei iPhones oder zwei Android Geräte braucht.

Hier könnt ihr die Morse-Decoder Projekte herunterladen:
Morse-Decoder Android
Morse-Decoder iOS

Merke: Ebenfalls sehr nützlich zum Testen ist es, wenn wir die Zeit seit Beginn der Übertragung als Log-Nachrichten ausgeben und überprüfen, ob diese bis auf kleine Ungenauigkeiten stimmen.

Android

Auf Android können wir als Lichtquelle mit einem einfachen View Element arbeiten. Die (Hintergrund-)Farbe kann man dann mit folgendem Code setzten:

```
view.setBackgroundColor(ContextCompat.getColor(this, R.color.white));
```

Für das Timing unter Android werft ihr am Besten einen Blick auf diese Funktion.

Alternativ könnte man auch das Flashlight von eurem Smartphone verwenden. Das ist aber deutlich aufwändiger zu programmieren und zu timen, da man das über die komplizierte Kamera-Schnittstelle machen muss.

Hier findet ihr zwei Klassen die das Codieren des Morsecodes übernehmen. Diese könnt ihr zum Beispiel wie folgt verwenden (der Code ist unvollständig und kann, muss aber nicht, übernommen werden):

```
public static final int DIT_DURATION_MILLISECONDS = 500;

private void sendNextPrimitive() {
    MorseEncoder morseEncoder = new MorseEncoder();
    List<Primitive> code = morseEncoder.textToCode(message);

    // transmission finished?
    if (code.size() == 0) {
        // lights should be off after the transmission
        lightOff();
        return;
    }

    Primitive nextPrimitive = code.remove(0);

    if (nextPrimitive.isLightOn()) {
        lightOn();
    } else {
        lightOff();
    }

    setTimer(nextPrimitive.getSignalLengthInDits() * DIT_DURATION_MILLISECONDS);
}
```

iOS

Bei iOS könnt ihr als Lichtquelle einfach eine normale View verwenden, die ihr abwechslungsweise auf schwarz (dunkel) oder weiss (hell) setzt. Wir haben wieder eine Projekt-Vorlage vorbereitet, die ihr hier herunterladen könnt. Diese Vorlage ist unvollständig und enthält diverse TODOs, die von euch erledigt werden müssen, damit die App korrekt funktioniert.