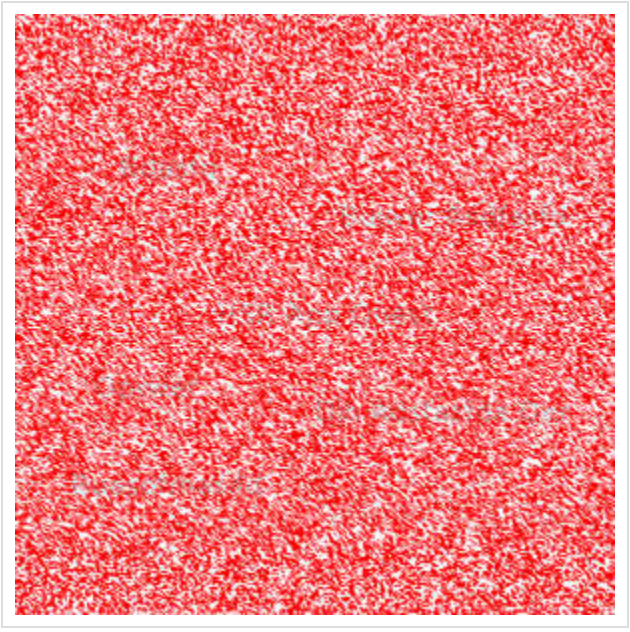
App Quest 2019

Dechiffrierer

Nach der eher einfachen Metall Detektor Aufgabe wird es nun ein wenig anspruchsvoller: wir erstellen eine App um speziell chiffrierte Bilder lesbar zu machen.



Welches Wort ist im Bild versteckt?

Klicke hier um das Lösungsbild anzuzeigen

Wie man sieht haben wir noch zusätzliches unsinniges Kauderwelsch eingebaut um die Aufgabe zu erschweren. Das wird auch am Treasure Hunt der Fall sein!

Eine Möglichkeit, das Bild lesbar zu machen wäre, sich einfach eine Brille mit roten Gläsern oder Folie aufzusetzen. Durch die rote Farbe wird das störende rot-weiss Muster sozusagen entfernt. Natürlich wollen wir keine Folienbrille basteln, sondern eine entsprechende App schreiben. Wir benötigen dazu eine sogenannte Multiplikation jedes Pixels des Fotos mit der Farbe Rot.

$$f(a, b) = ab$$

(Quelle: Wikipedia)

Multipliziert wird dabei jede Rot-, Grün- und Blau-Komponente einzeln, wobei jede Komponente nicht etwa einen Wert zwischen 0 und 255 hat, wie wir das zum Beispiel aus der Webentwicklung gewohnt sind, sondern nur zwischen 0 und 1. Im Code müssen die Werte also noch entsprechend skaliert werden. Die „Farbe Rot“ mit der wir multiplizieren ist nichts anderes als 100% Rot und je 0% Grün und Blau.

(Apropos Webentwicklung: In CSS könnten wir denselben Effekt mit background-blend-mode: multiply erreichen. Unter Android und iOS programmieren wir das aber selbst.)

Android

Die Android App besteht im wesentlichen aus einer einzigen Activity, die ein Foto aufnimmt und das anschliessend nachbearbeitet und darstellt.

Foto Aufnehmen

Als erstes muss ein Foto aufgenommen werden. Dabei müssen verschiedenste Dinge beachtet werden (Permissions, Speicherort). Der ganze Ablauf ist in diesem Tutorial gut erklärt. Das Bild kann dann mit einer kleinen Hilfsfunktion als Bitmap eingelesen werden:

```java
private Bitmap readImageFile(Uri imageUri) {
    File file = new File(imageUri.getPath());
    InputStream is = null;
    try {
        is = new FileInputStream(file);
        Bitmap bitmap = BitmapFactory.decodeStream(is);
        return bitmap;
    } catch (FileNotFoundException e) {
        Log.e("DECODER", "Could not find image file", e);
        return null;
    } finally {
        if(is != null) {
            try {
                is.close();
            } catch (IOException e) {
            }
        }
    }
}
```

readImageFile.java hosted with ❤ by **GitHub** **view raw**

Das Bitmap kann dann zum Beispiel in einer ImageView mit setImageBitmap angezeigt oder weiterverarbeitet werden.

Bitmap Bearbeiten

Um das Bitmap zu bearbeiten müssen wir die einzelnen Pixel manipulieren können:

```java
private Bitmap applyFilter(Bitmap bitmap) {
    int width = bitmap.getWidth();
    int height = bitmap.getHeight();
    int[] data = new int[width * height];

    bitmap.getPixels(data, 0, width, 0, 0, width, height);

    // Hier können die Pixel im data-array bearbeitet und
    // anschliessend damit ein neues Bitmap erstellt werden

    return Bitmap.createBitmap(data, width, height, Bitmap.Config.ARGB_8888);
}
```

applyFilter.java hosted with ❤ by **GitHub** **view raw**

Pixel werden einfach als Integer dargestellt, wobei die 32 Bits des Integers in 4 Mal 8 Bit aufgeteilt werden (daher auch die ARGB_8888 beim createBitmap Aufruf) und folgende Bedeutung haben:

ALPHA								RED								GREEN								BLUE							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0

Quelle: DYClassroom

Tipp: Die Color-Klasse enthält Hilfsmethoden um die einzelnen Farbanteile zu extrahieren.

So können wir nun also jeden Pixel mit einem Pixel der Farbe Rot multiplizieren und können dann hoffentlich den chiffrierten Begriff entziffern. Dieses Wort muss dann nur noch im Logbuch eingetragen werden (es gibt bei dieser Aufgabe keinen QR-Code zu scannen). Das Lösungswort kann zum Beispiel über einen AlertDialog (Tutorial) eingegeben werden.

iOS

Die iOS App kann aus einem einzelnen View Controller bestehen, über den wir ein Bild aufnehmen, nachbearbeiten und darstellen können.

Foto aufnehmen

Um ein Foto aufzunehmen, können wir die Klasse UIImagePickerController verwenden. Hier gibt es ein gutes Video Tutorial, welches den ganzen Ablauf im Detail erklärt: https://www.youtube.com/watch?v=4CbcMZOSmEk

Pixel bearbeiten

Vom UIImagePickerController bekommen wir ein UIImage. Dieses müssen wir zuerst noch in einen Pixel-Buffer umwandeln. Dafür habe ich euch in der Projektvorlage (siehe unten) die Hilfsmethoden prepareContext() und getPixelBuffer() zur Verfügung gestellt. Eure Aufgabe ist es dann, in der applyFilter() Methode mit einem for-Loop jedes einzelne Pixel mit der Farbe Rot zu multiplizieren.

```swift
func applyFilter(to image: UIImage) -> UIImage? {
  guard let context = prepareContext(for: image),
        let pixelBuffer = getPixelBuffer(from: context) else {
        return nil
  }

  // TODO: Hier müssen wir die einzelnen Pixel im pixelBuffer modifizieren und wieder
  // zurück in den pixelBuffer schreiben.

  guard let outputCGImage = context.makeImage() else {
    return nil
  }
  return UIImage(cgImage: outputCGImage, scale: image.scale, orientation: image.imageOrientati
}
```

applyfilter.swift hosted with ❤ by **GitHub** **view raw**

In der App werden einzelne Pixel mit dem struct RGBA32 dargestellt. Dieser ist auch in der Projektvorlage vorhanden.

Eingabe des Lösungswortes

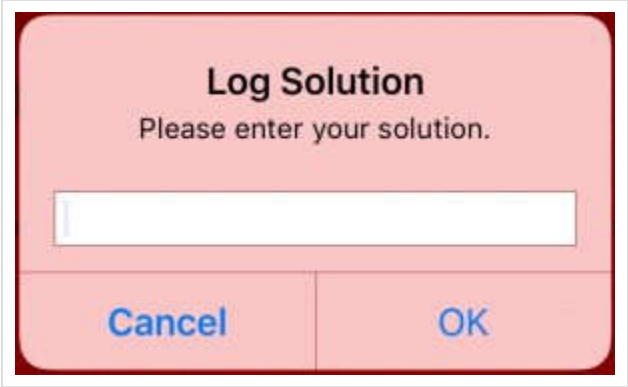
Nachdem wir das Bild dechiffriert haben, brauchen wir noch eine Möglichkeit, die Lösung im Logbuch einzutragen. Dazu können wir einen UIAlertController mit einem Eingabefeld benutzen, wie das im folgenden Code-Beispiel gezeigt wird:

```swift
@IBAction func logSolution(sender: UIBarButtonItem) {
  let alertController = UIAlertController(title: "Log Solution",
                                          message: "Please enter your solution.",
                                          preferredStyle: .alert)
  alertController.addTextField(configurationHandler: { textField in
    textField.autocorrectionType = .no
  })
```

```
8     alertController.addAction(UIAlertAction(title: "Cancel", style: .cancel))
9     alertController.addAction(UIAlertAction(title: "OK", style: .default, handler: { _ in
10      guard let solution = alertController.textFields?[0].text else {
11        return
12      }
13
14      // In der Konstante solution steht das Lösungswort, das vom Benutzer eingegeben wurde.
15      // TODO: Dieses Wort muss jetzt ähnlich wie beim Metalldetektor im Logbuch eingetragen wer
16    }))
17
18    present(alertController, animated: true)
19  }
```

UIAlertController.swift hosted with ❤ by **GitHub** **view raw**

Das sieht dann ungefähr so aus:



Projektvorlage

Für die iOS Teams habe ich eine einfache Projektvorlage vorbereitet. Diese Vorlage könnt ihr hier herunterladen. Die Vorlage ist an den mit // TODO gekennzeichneten Stellen zu ergänzen.