ToiletPaper #136

Android CameraX: Let’s take a selfie

Autor: Kevin Stieglitz / Software Engineer / Business Division Automotive World

✕ Problem

Die Benutzung der Kamera war schon immer eine kleine Herausforderung bei Android. Nachdem Google die alte android.hardware.camera-API mit Android 5 (Lollipop) aufgrund zu geringen Funktionsumfangs als „deprecated“ markiert hat, wurde im Gegenzug die Camera2-API eingeführt. Diese ermöglicht es Hersteller und Entwickler, auch komplexe Funktionen in die Kamera zu integrieren. Im Gegenzug hat sich aber auch der Aufwand und die Komplexität der Implementierung stark erhöht.

✔ Lösung

Um der erhöhten Komplexität entgegen zu wirken, wurde auf der Google IO 2019 die CameraX-Bibliothek vorgestellt, welche Teil des Android Jetpack Komponentensets ist. CameraX setzt auf die Camera2-API auf und kann ab Android 5 verwendet werden. Dabei wird die Idee von Use-Cases umgesetzt, um eine einfache API zu gewährleisten. Derzeit werden drei Use Cases bereitgestellt:

- Preview: Zeigt eine Vorschau der Bilder an.
- Image Analysis: Bietet einen Zugang zu einem Stream von Bildern zur anschließenden Weiterverarbeitung.
- Image Capture: Speichert Bilder in hoher Auflösung.

CameraX bietet aber nicht nur verschiedene Use Cases an, sondern übernimmt auch die Konfiguration von gerätespezifischen Einstellungen. Google stellt hierfür ein eigenes Test Lab bereit, worin eine Vielzahl unterschiedlicher Geräte automatisiert getestet werden.
Außerdem wird ein optionales Add-on mit den Namen „extensions“ bereitgestellt, welches den Zugriff auf gerätespezifische Effekte (wie beispielsweise HDR-, Nacht- oder Potrait-Modus) ermöglicht.

➔ Beispiel

Folgendes Snippet zeigt, mit wie wenig Aufwand der Preview-Use-Case in das System eingebunden werden kann. Das Ergebnis ist im Screenshot rechts oben zu sehen.

```
1  private fun bindPreviewUseCase() {
2      // Get screen metrics used to setup camera for full screen resolution
3      val metrics = DisplayMetrics().also { textureView.display.getRealMetrics(it) }
4      val screenAspectRatio = Rational(metrics.widthPixels, metrics.heightPixels)
5
6      // Set up the view finder use case to display camera preview
7      val viewFinderConfig = PreviewConfig.Builder().apply {
8          setLensFacing(CameraX.LensFacing.FRONT)
9          setTargetAspectRatio(screenAspectRatio)
10         setTargetRotation(textureView.display.rotation)
11     }.build()
12
13     // Use the auto-fit preview builder to automatically handle size/orientation changes
14     preview = AutoFitPreviewBuilder.build(viewFinderConfig, textureView)
15
16     // Apply declared configs to CameraX using the same lifecycle owner
17     CameraX.bindToLifecycle(viewLifecycleOwner, preview)
18 }
```

✚ Weiterführende Aspekte

- https://developer.android.com/training/camerax