Agenda

• Why mobile is important for us

• ARToolKit6 for Android
  • Get familiar with ARTK6 on Android
  • Tracking example
  • Getting started

• ARToolKit6 for iOS
  • Tracking example
  • Structure of ARTK6
  • Getting started

• Questions
Augmented Reality feels like magic
Why mobile

• People love looking at their phone

• We enable them to see beyond their screen

• Mobile devices are the start to get people in touch with augmented reality

Step 1: Developer uses "ARToolKit Camera Calibrator" app to calibrate his or her devices. Results are pushed to calibration server.

Step 2: User runs an ARToolKit-based app. App fetches calibration data for his or her device from server.

Step 3: On subsequent runs, app uses locally cached calibration data.
STEPS TO BE AWESOME

got cape.  
wear cape.  
fly.
Architecture on Android

[Diagram showing the architecture of ARToolKit on Android, with various components and their relationships depicted through arrows and labels.]

Key:
- C++ class
- External library
- Internal library
- Hybrid system
- Binding

ARToolKit 6
Architecture on Android

Key:
- **C++ class**
- **Binding**
- **Java class**

ARController

- **<interface> CameraEventListener**
  - instantiates **CameraEventListenerImpl**

- **<interface> ARToolkitCallback**
  - uses **ARToolkitCallbackImpl**

- **<abstract> ARActivity**
  - uses **CameraEventListenerImpl**
  - uses **ARRenderer**
  - uses **ShaderProgram**

- **<abstract> ARRenderer**
- **<abstract> ShaderProgram**

- **<singleton> ARToolkit**
  - uses **NativeInterface**

- **NativeInterface**
  - Java API
  - JNI
  - C API

- **C API**

- **Java API**

- **JNI**

- **C++ class**

- **Binding**

- **Java class**
Tracking example

```
ARTrackingActivity -> ARTrackingRenderer

ARActivity
ARRenderer
ShaderProgram
SimpleShaderProgram

<interface> CameraEventListener
<interface> ARToolKitCallback
<interface> ARToolKitCallbackImpl
<abstract> ARActivity
<abstract> ARRenderer
<abstract> ShaderProgram

<interface> CameraEventListenerImpl
<interface> ARToolKitCallbackImpl
<interface> ARActivity
<abstract> ARRenderer
<abstract> ShaderProgram

<interface> CameraEventListener
<interface> ARToolKitCallback
<interface> ARToolKitCallbackImpl
<abstract> ARActivity
<abstract> ARRenderer
<abstract> ShaderProgram

<interface> CameraEventListenerImpl
<interface> ARToolKitCallbackImpl
<interface> ARActivity
<abstract> ARRenderer
<abstract> ShaderProgram

<interface> SimpleShaderProgram
<interface> VertexShader
<interface> FragmentShader
<interface> ShaderProgram
<interface> ARDrawable

<interface> ARDrawable
<interface> SimpleShaderProgram
<interface> VertexShader
<interface> FragmentShader
<interface> ShaderProgram
<interface> ARDrawable

<interface> NativeInterface
Java API
JNI
C API

Key:
- C++ class
- Binding
- Java class
- App class
- instantiates
- uses
- implements
```
The actual code

- ARTrackingActivity
- ARTrackingRenderer
Getting started

- ARToolKit6 SDK download

- Setup Android Studio and the NDK (Native Development Kit)

- Install the JDK
  - For Android SDK manager

- Open the SimpleTracking project

- Documentation
Questions regarding Android

Contact: thorsten.bux_c@daqri.com
Virtual Environment Generation

- Construct a VE containing the digital dimensions of the experience
- Align the VE with the actual environment

• OpenGL projection matrix to match camera optical properties
• Video background rendering
• Basic model loading and rendering
• Too low-level for most users but maximally customisable

• High-performance cross-platform game engine
• Camera projection and position represented in scene graph
• Video background rendering
• Large ecosystem of solutions and vendors

OpenGL

OpenGL ES

Unity
## Development platforms

<table>
<thead>
<tr>
<th>Platform</th>
<th>Developer tools</th>
<th>APIs</th>
</tr>
</thead>
<tbody>
<tr>
<td>iOS</td>
<td>Xcode</td>
<td>iOS-specific classes for AR</td>
</tr>
<tr>
<td>Android</td>
<td>Android Studio</td>
<td>Choice of Java / Java+native, full native with C++ framework</td>
</tr>
<tr>
<td>macOS</td>
<td>Xcode</td>
<td>Examples in C++</td>
</tr>
<tr>
<td>Linux</td>
<td>standard tools</td>
<td>Examples in C and C++</td>
</tr>
<tr>
<td>Windows Desktop</td>
<td>Visual Studio 2013</td>
<td>Examples in C and C++</td>
</tr>
<tr>
<td>Windows Mobile</td>
<td>Visual Studio 2013</td>
<td>Native C# component and examples, low-level C/C++ framework</td>
</tr>
</tbody>
</table>
Prerequisites

• A Mac

• iPhone or iPad running iOS 7.0 or later (can’t use simulator for ARToolKit)

• An Apple ID to associate with your developer account (no cost to deploy to your own device)

• Xcode
• Open Examples/Source/iOS/ARToolKit6 Tracking Example.xcodeproj

• Choose target: ARToolKit6 Tracking Example/Device

• Build
Running on a device

• Connect device, and check it is enabled in Organizer

• Choose an app target and build

• Select team. Also choose your own unique app bundle ID

• If security warning: enable developer account in device settings