Testing solution for Virtual Reality applications
DEKRA Performance Test Solutions
Motivation facts

- Mobile apps consume data differently depending on various network conditions.
- Carriers need to understand how the most popular Android and iOS apps consume data from a network.
- Carriers need to understand the network conditions that drive poor/excellent User Experiences.
- Carriers need to test an app in the exact same manner that customers use apps. No simulations... just real apps consuming real data.
Starting Point: DEKRA’s Current solution for non VR apps

- Non VR Apps:
  - Downlink Intensive Video Streaming (including 4k)
  - Uplink Intensive Video Streaming
  - Two-way Video Streaming
  - Social Media
Starting Point: Current solution for non VR apps

Testing Topology: High Scalability: M x N devices can be automated simultaneously

- **Test Controller**
- **Automation Agents**
- **Automated Devices**
- **Internet**
- **App Server**

**Network Link**
Automation Agents can be deployed in remote sites

**ADB Link**
ADB (Android Device Bridge) can use USB or Wi-Fi. In either case, Automated Devices must be in locally attached to the Automated Agent.
Starting Point: Current solution for non VR apps

Automation Test Flow: Example – Video Streaming App

Test Controller

Automation Agents

Automated Devices

Internet

List of scripts

Step 1: Open App

Step 2: Select Video

Step N: Start Play Back

Script Phase 1: Browsing through the app menu until starting video streaming

KPI: Initial Buffering Time

Spinner Appear Detection

Spinner Dissapear Detection

Script Phase 2: QoE KPI deduction based on UI object events

© 2018 DEKRA Testing and Certification S.A.U.
Starting Point: Current solution for non VR apps

- We use the following technologies:
  - **Appium** [open source test automation framework for use with native, hybrid and mobile web apps] for
    - Browsing through the App menu
    - Recognize UI objects (e.g., spinner, progression bar)
  - **ADB** [Android Device Bridge] for device file size usage reporting.
  - **OCR** [Optical Character Recognition] for extracting App information
Starting Point: Current solution for non VR apps

The following KPIs have been proved:

<table>
<thead>
<tr>
<th>Mobile Apps</th>
<th>KPIs</th>
</tr>
</thead>
<tbody>
<tr>
<td>All (App Agnostic)</td>
<td>Battery, Data Usage, Throughput</td>
</tr>
<tr>
<td>Netflix</td>
<td>Initial Buffering, Re-bufferings</td>
</tr>
<tr>
<td>YouTube</td>
<td>Initial Buffering, Re-bufferings, Video Resolution</td>
</tr>
<tr>
<td>Instagram</td>
<td>Access Time, Initial Buffering, Re-bufferings</td>
</tr>
<tr>
<td>Periscope</td>
<td>Initial Buffering, Re-bufferings</td>
</tr>
<tr>
<td>Skype Video Call</td>
<td>Call Setup Time, Call Result</td>
</tr>
<tr>
<td>WhatsApp</td>
<td>Sharing Time</td>
</tr>
<tr>
<td>Facebook</td>
<td>Access Time, Initial Buffering, Re-bufferings, Sharing Time</td>
</tr>
<tr>
<td>Snapchat</td>
<td>Send Content Time, Access Time</td>
</tr>
<tr>
<td>Livestream</td>
<td>Initial Buffering, Re-bufferings</td>
</tr>
</tbody>
</table>
Starting Point: Current solution for non VR apps

- **Limitations** of this approach for testing VR/Gaming apps:
  - **Performing movement**
    - VR and gaming apps require physical movement of the hosting device. As the gyroscope and the accelerometer cannot be mocked, a hardware platform is required.
  - **Retrieving App State**
    - Unlike other apps, VR and gaming apps are programmed in an Android UI Canvas where the graphical engine works (e.g., Open GL). Appium (or similar) cannot recognize the UI objects inside an App Canvas.
Starting Point: Current solution for non VR apps

- Limitations of this approach for testing VR/Gaming apps:

Actual App UI

What Appium sees….
In order to overcome those limitations we have upgraded the architecture:

- Robotic Platform

**Four Servo-Motors:**

1. Pitch  (-90°, +90°)
2. Roll 1  (-180°, 0°)
3. Roll 2  (0°, +180°)
4. Yaw    (-90°, +90°)
Testing Solution for VR/Gaming apps

System Requirements

- VR and Gaming Testing Requirements:
  - Ability to measure the KPI “time to load a virtual scene” \((t_2 - t_1)\), where
    - \(t_1\) = user clicks on “start scene” button
    - \(t_2\) = the scene is totally rendered in device screen

- Minimize reaction time \((t_3 - t_4)\) where
  - \(t_3\) = target appears on the screen
  - \(t_4\) = tap on that target

- VR/Gaming QoE KPI Deduction
  - VR/Gaming: Automate the browsing through the app where some UI could be moving objects.
  - Gaming: Shoot at moving target
Testing Solution for VR/Gaming apps

Implementation Decisions

- High performance screen capture
  - Requirement: Higher than 24 frames per second

- Low delay screen touch
  - Requirement: Lower than 10 ms response time

- IR (Image Recognition)
  - Requirement: High pattern matching accuracy and high performance
  - Technology used: OpenCV: “Open Source Computer Vision Library”
Testing Solution for VR/Gaming apps

Closed-Loop Implementation

**Automation Agents**

- \( \Delta t_1 \)
- \( \Delta t_2 \)
- \( \Delta t_3 \)
- \( \Delta t_4 \)
- \( \Delta t_5 \)

**Automated Device / Robotic Platform**

- \( t_3 \)

**Goal**

Average Human Reaction Time: **284 ms**


**Implemented Reaction Time:**

- \( \Delta t_1 \sim 1-10 \text{ ms} \rightarrow \) TCP socket latency
- \( \Delta t_2 \sim 100-200 \text{ ms} \rightarrow \) Image Recognition performance
- \( \Delta t_3 \sim 1-10 \text{ ms} \rightarrow \) Serial COMM latency
- \( \Delta t_4 \sim 100 \text{ ms} \rightarrow \) Time to aim at object
- \( \Delta t_5 \sim 1-10 \text{ ms} \rightarrow \) TCP socket latency

**DEKRA’s solution Reaction Time = (230, 330) ms**
## Testing Solution for VR/Gaming apps

### KPIs Implemented

<table>
<thead>
<tr>
<th>KPI</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Network Resources Usage</strong></td>
<td>Data Usage, Throughput</td>
</tr>
<tr>
<td><strong>Device Resources Usage</strong></td>
<td>Battery, CPU, GPU</td>
</tr>
<tr>
<td><strong>Time to load the virtual world</strong></td>
<td>Time elapsed from selecting a scenario (world, experience, etc.) to loading the 3D visual context</td>
</tr>
<tr>
<td><strong>Immersion Cut-off</strong></td>
<td>Probability that successfully started immersion is ended by a cause other than the intentional termination by the user</td>
</tr>
<tr>
<td><strong>Time to load the virtual scene</strong></td>
<td>Time elapsed from selecting a scene to reloading the 3D visual context</td>
</tr>
</tbody>
</table>
Showcase: Testing Google Cardboard App

- VR experience, e.g., for Google Earth
- Replacing the mouse by the head movement
Showcase: Testing Google Cardboard App

Testing Setup

Automatic test cycles: **40 repetitions** / **BW configuration**
Showcase: Testing Google Cardboard App

Test repetition flow

Open App

Navigate through the app
until click “start experience”

Measurement

Automatic test cycles: **40 repetitions / BW configuration**
Showcase: Testing Google Cardboard App

Test Results

KPI: Average Time To Load Scenario (s)

9.57 s (best scenario)

KPI: Network Data Usage

8 MB (all scenarios)
Conclusions

Key-takeaways

▪ The “time to load scenario” KPI is severely impacted by the quality of the network access, mainly the available bandwidth (Mbit/s).

▪ Online Virtual Reality apps consumes huge amount of network data, which has impact on network planning and deployments.

▪ Online Virtual Reality apps requires high device GPU performance.

Lesson-learnt

▪ Thanks to the fast closed-loop response time of the solution on Android, the solution can be also used to measure online games apps.

▪ The image recognition library matching score parameter has impact on the accuracy of the “time to load scenario” measurement.

▪ Nowadays Online VR apps do not implement video quality adaptation in order to adapt the app experience to the network conditions.