Remote Android Rendering Summary

Joel Isaacson

Ascender Technologies Ltd.

The Problem

- There are just too many pixels to simply transmit over a long-haul network.
- There are a number of techniques that have been attempted.
- They all entail some compromises:
  - Resolution
  - Accuracy
  - Frame Rate
  - Latency

The Solution

- Our approach is to export graphics at the rendering level.
  - Resolution – Resolution independent.
  - Accuracy – No loss of accuracy.
  - Frame Rate – Extreme compression, typically 500 bytes per frame. 60 fps can be achieved with a 30 Kbtye/sec bandwidth.
  - Latency – No buffering or round-trip latencies in the graphic codec. Simplex streaming.

A Perfect Storm

- It seems that a technological cosmic alignment has happened:
  - Fast, low-power 64 bit ARM multi-processors (Cortex A50) with virtualization extensions.
  - Adoption of Android apps in a broad gamut of use cases, including the enterprise.
  - Ever increasing adoption of cloud based solutions.
  - Possibility of efficiently transporting Android graphics via a long haul network.

Remote Android Rendering Summary

Joel Isaacson

Ascender Technologies Ltd.

For many years it has been possible to access graphical application via remote desktop software. In recent years Cloud computing has become more prominent and is a crucial computing paradigm.

Android has captured a large market share. The challenge addressed in this talk is to efficiently export Android graphics so as to support standard Android apps remotely.

The Problem

- There are just too many pixels to simply transmit over a long-haul network.
- There are a number of techniques that have been attempted.
- They all entail some compromises:
  - Resolution
  - Accuracy
  - Frame Rate
  - Latency

Current techniques to provide remote graphic access are pixel based.
An example of pixel-based remote Android graphics is: Amazon's test drive, which allows remote demos of Android apps before purchase. Pixel based solutions force compromise on all four performance properties:

● Resolution ● Accuracy ● Frame Rate ● Latency

Our techniques allow un-compromised performance coupled with very low network bandwidth.

The Solution

- Our approach is to export graphics at the rendering level.
  - Resolution – Resolution independent.
  - Accuracy – No loss of accuracy.
  - Frame Rate – Extreme compression, typically 500 bytes per frame. 60 fps can be achieved with a 30 Kbtye/sec bandwidth.
  - Latency – No buffering or round-trip latencies in the graphic codec. Simplex streaming.

The data compression algorithm reduces the volume of data to less than the toolkit level. The rendering stream is scanned for sequences of commands that are reversed engineered into both application and toolkit level routines. These routines are entered into dictionaries shared by both the encoding (server) and decoding (client) ends. Long sequences of rendering commands are sent by simple reference to the dictionary entries from the server to the client.
More details can be found on the URL:

http://www.ascender.com/remote-graphics

A Perfect Storm

- It seems that a technological cosmic alignment has happened:
  - Fast, low-power 64 bit ARM multi-processors (Cortex A50) with virtualization extensions.
  - Adoption of Android apps in a broad gamut of use cases, including the enterprise.
  - Ever increasing adoption of cloud based solutions.
  - Possibility of efficiently transporting Android graphics via a long haul network.

The enabling technologies that allow for Remote Android Graphics have many uses:

- Cloud computing, remote app server
- App library, subscription model
- App demos
- Remote enterprise applications
- Set-top boxes
- Cloud Gaming