

Project Number 32: Android Emulator Extension

Project Sponsors: Jim Butler, Steven Labelle

Faculty Advisor: Tom Schubert

Team Members: Solomon Habtemariam, Jeremiah Franke, Omar Saadoun

Project Report

June 8 2015

Version: 1.1

**Table of contents:**

Executive Summary

Motivation

Problem Statement

Methods

Conclusions

Project managment

Testing

Ethics

Discussion

Appendix

**Executive Summary:**

Currently Android App developers lack a fast emulator at a low price to perform testing for development across multiple target devices. The market for a software emulator is in high demand driven by the fragmentation of the android market across device manufacturers, API levels, screen sizes and available sensors. The goal of this project is to create a fast and sensor rich android emulator that can be made available to small app development companies at a low cost and be able to work across Windows, Macintosh and Linux host machines.

To that end Oracle’s Virtualbox was used to create snapshots of several target devices to cover a reasonable representation of the Android ecosystem. These virtual machines are built from an android kernel specific to the device’s manufacturer combined with the target device’s hardware binary file.

Python scripts were written to manage the startup and shutdown of these snapshots to automate the use of the emulator for any target device. These scripts were developed to work across Linux, Windows and Macintosh file systems and command line interfaces.

A networking plug-in for Virtualbox was used to allow port forwarding between the host machine and the virtual device. The Android Debugging Bridge was installed to communicate between the host machine and the virtual device. Python scripts for operating the Android Debugging Bridge were written and developed across Linux, Windows and Macintosh file systems and command line interfaces.

An open source program for sending Android sensor data through the Android Debugging Bridge to a device was modified to work with Virtualbox. A test harness was written to take a test bench of sensor events and a sampling rate then forward that data to the emulated device in real time.

The result is a free, open source, fast, reliable and fully featured emulator that runs on Linux, Windows and Macintosh host machines.

**Motivation:**

Fragmentation is both a strength and weakness of the Android ecosystem, a headache for developers that also provides the basis for Android’s global reach. Android devices come in all shapes and sizes, with vastly different performance levels and screen sizes. Furthermore, there are many different versions of Android that are concurrently active at any one time, adding another level of fragmentation. What this means is that developing apps that work across the whole range of Android devices can be extremely challenging and time-consuming. See the graphs on fragmentation in the Appendix from opensignal.

Despite the problems, fragmentation also has a great number of benefits – for both developers and users. The availability of cheap Android phones (rarely running the most recent version) means that they have a much greater global reach than iOS, so app developers have a wider audience to build for.

In today's Android App development community, the typical app developer must choose between:

1. Purchasing and maintaining a growing collection of physical mobile devices.
2. Outsourcing testing to a testing lab with a large collection of physical mobile devices.
3. Utilizing a web/cloud testing service who have & maintain a large collection of physical mobile devices.

These 3 solutions are very expensive especially for small app development companies. Compounding the expense is the fragmentation of the android marketplace. The top 20 devices change considerably year to year, some recieve API updates while others don't. Each might have special features unique to the manufacturer such as facial recognition scanners or finger print scanners that are not available on devices from other vendors. Further, screen resolution and size makes developing a UI across multiple platforms that looks good very difficult.

For these reasons it is necessary to develop more powerful android emulators that can do more of the development testing and troubleshooting. Currently there are two types of emulators. Slow emulators (running on ARM machine code require a translation layer on x86 machines) like the Android emulator provided in Android Studio by Google. Android Studio is a free piece of software but it is bloated, slow and does not have an interface for automating testing. Fast emulators like Genymotion exist. They run virtual devices through Oracle's Virtualbox and device images provided by the open source Android x86 project. However, the 40$ per month per person price tag keeps this solution out of the hands of most developers.

There is a clear and highly in demand market for an open source and free android emulator that runs on x86 machines with an interface for automated regression testing.

**Problem Statement:**

In today's Android App development community, the typical app developer must choose between:

1. Purchasing and maintaining a growing collection of physical mobile devices.
2. Outsourcing testing to a testing lab with a large collection of physical mobile devices.
3. Utilizing a web/cloud testing service who have & maintain a large collection of physical mobile devices.

All 3 of these alternatives can quickly become expensive, especially for the smaller app development companies (12 people or less). This project is an exploration into the possibilities of utilizing the Software Android emulators to perform more work than they do today but in an automated fashion. If this is possible, the smaller app development companies may be able to reduce their costs associated with maintaining an up to date library of physical devices.

Goals:

* Research and select a virtual machine platform for each host OS.
* Create a reliably functioning Android VM on each hose using either a public x86 Android distribution (e.g. www.android-x86.org/or an Intel Android distribution).
* Add SW to the VM (e.g. services, apps, drivers, etc.) as needed to emulate sensor chips and the data that they return (at least GPA and accelerometer). In this case, emulation means to insert SW between the HW sensor and the OS. This SW should intercept the HW data from the sensor and replace it with data from another source (e.g. SD card).
* Add SW to the system to allow programmatic control of the fake sensor data (e.g. on/off).
* The project goals are to create 3 prototypes (1 each for Windows, Mac, and Ubuntu):
* The successful prototype would be able to fool an app that uses GPS data into thinking it’s in Cairo, then Tokyo, then Paris without leaving town. With programmatic controls, a test developer could create an automated test that uses SD data to test how an app responds to sensor data.
* Other sensors can be prototyped as well. This includes Hall, accelerometers, heat, fingerprint, camera, etc. The ability to fool multiple sensors at one time is highly desired but not required.

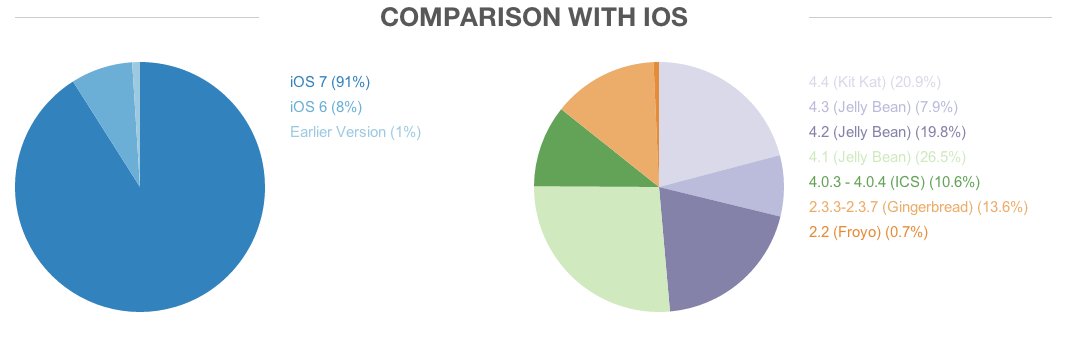
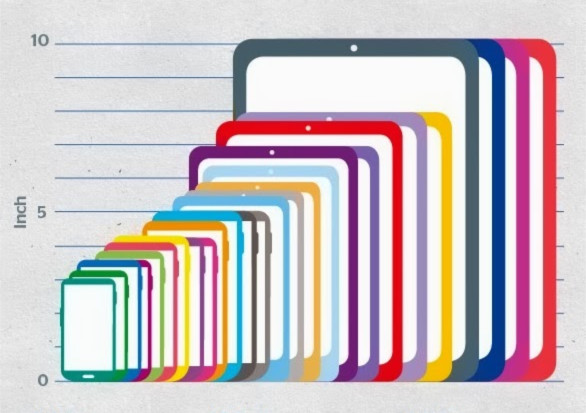
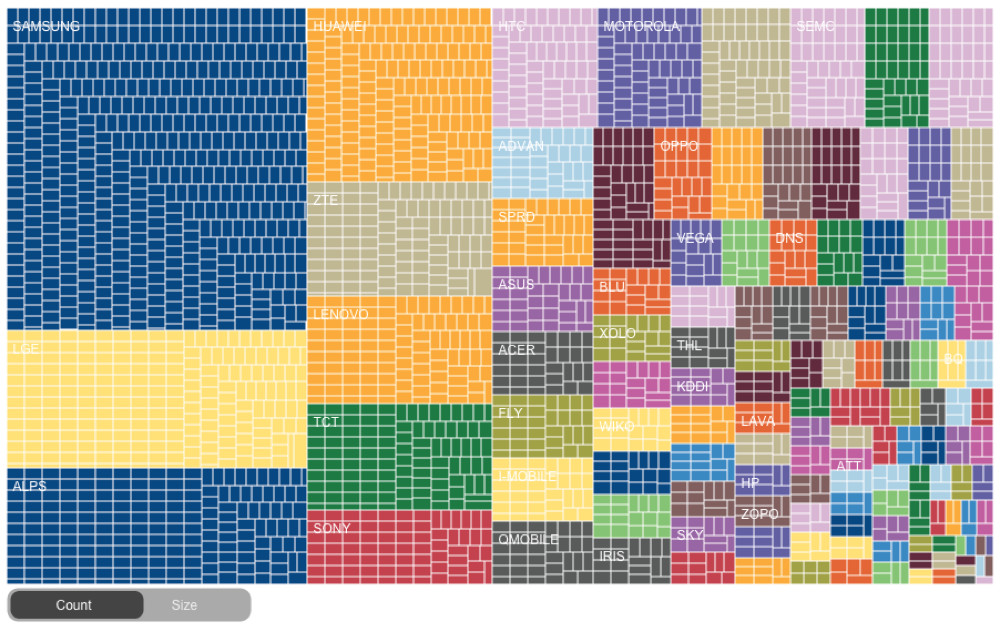
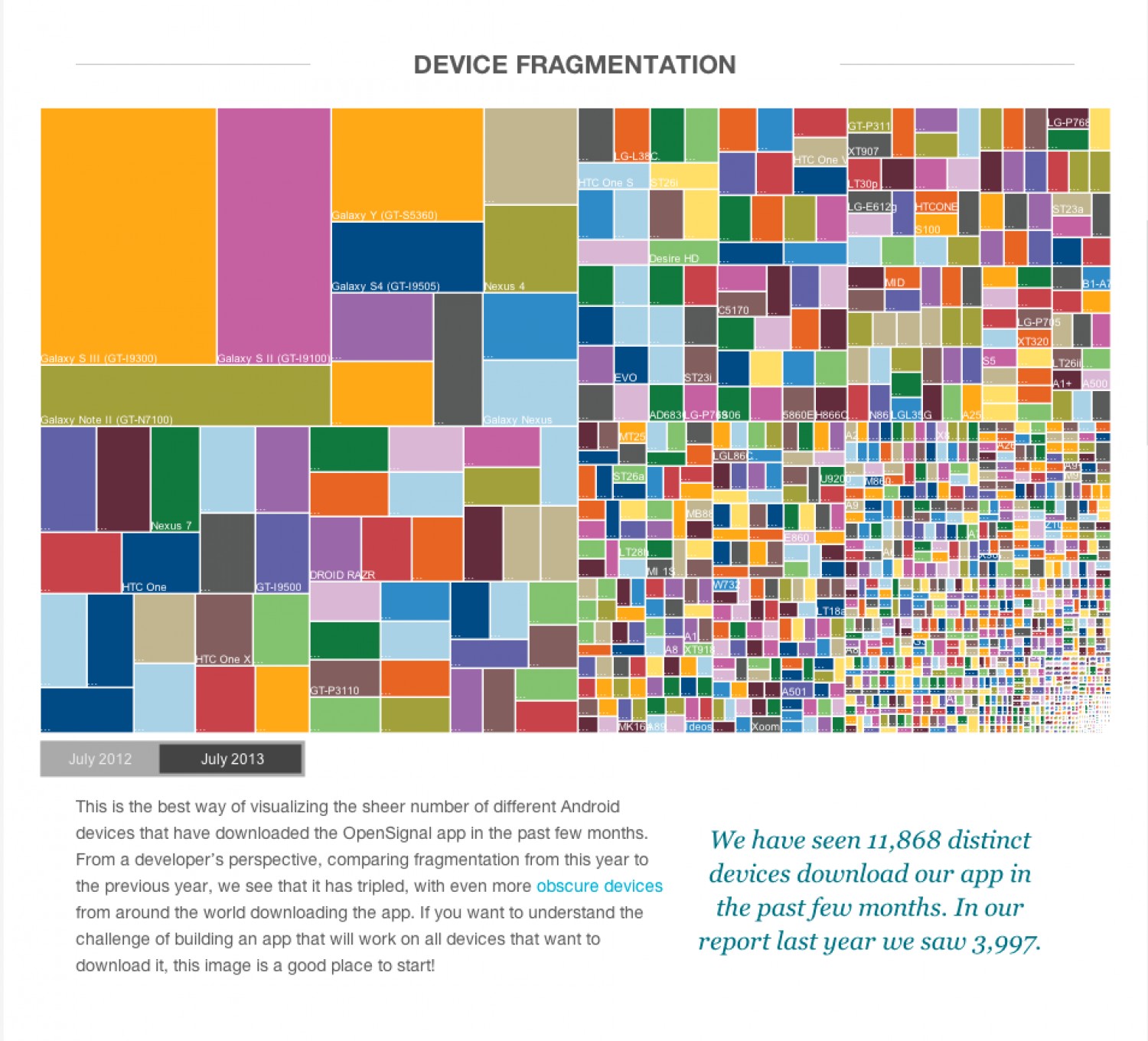
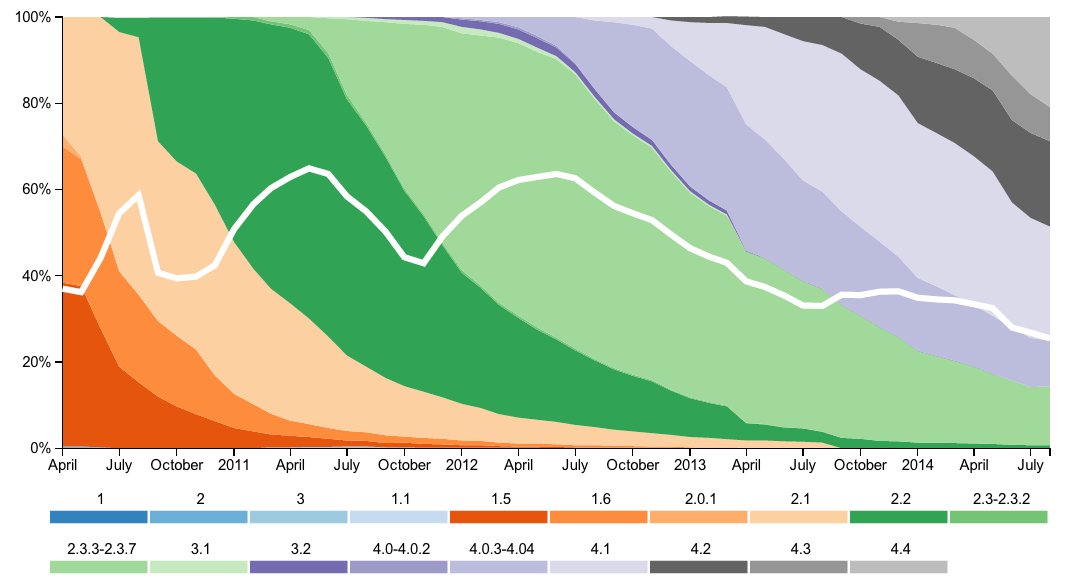
**Project Timetable:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Milestone | Task Name | Duration | Start | Finish |
| **1** | **Product Design Specification** | **11 days** | **Mon 2/16/15** | **Fri 2/27/15** |
|  | Product Description | 6 days | Mon 2/16/15 | Sat 2/21/15 |
|  | Project Schedule & Milestones | 5 days | Sun 2/22/15 | Thu 2/26/15 |
| **2** | **Phase 1: Virtual Device Setup** | **30 days** | **Fri 2/27/15** | **Thu 4/9/15** |
|  | Manual Target Device Setup | 3 days | Fri 2/27/15 | Tue 3/3/15 |
|  | Target Device Snapshots | 7 days | Wed 3/4/15 | Thu 3/12/15 |
|  | Boot Target Device From Snapshots | 7 days | Fri 3/13/15 | Mon 3/23/15 |
|  | Automate Boot From Snapshot | 7 days | Tue 3/24/15 | Wed 4/1/15 |
|  | Extend Boot From Snapshot To Second API | 6 days | Thu 4/2/15 | Thu 4/9/15 |
| **3** | **Phase 2: Device Sensors Simulation** | **32 days** | **Fri 4/10/15** | **Sun 5/24/15** |
|  | Sensor Monitor APP | 2 days | Fri 4/10/15 | Mon 4/13/15 |
|  | Create Sensor Events | 7 days | Tue 4/14/15 | Wed 4/22/15 |
|  | Override Sensor Events Data | 7 days | Thu 4/23/15 | Fri 5/1/15 |
|  | Adjust Events Sample Rate | 5 days | Sat 5/2/15 | Thu 5/7/15 |
|  | Extend Types Of Sensors | 7 days | Fri 5/8/15 | Sun 5/17/15 |
|  | Create Automated Regression Testing Platform | 6 days | Mon 5/18/15 | Sun 5/24/15 |
| **4** | **Project Demo** | **10 days** | **Mon 5/25/15** | **Fri 6/5/15** |
|  | Presentation Slot | 10 days | Mon 5/25/15 | Fri 6/5/15 |

**Product Design Specification:**

The goal of this project is to produce Android device snapshots for emulation using Virtualbox. Design and build a system for automating the startup of different Android virtual devices in Virtualbox. Minimize the boot time of these images to a maximum of 15 seconds. Ensure cross platform compatibility of the product across Windows, Mac and Ubuntu Operating Systems. Add to the product the ability to monitor Android device sensors (GPS, Accelerometer). Add to the product the ability to fake Android device sensor data from the user. Create the test harness framework for regression testing of the emulated device's sensors.

|  |  |  |
| --- | --- | --- |
| Marketing  Requirements | Engineering Requirements | Justification |
| 1,2,3 | 1. Compatibility across Windows, Mac and Ubuntu operating systems. | Accommodate as many app developers as possible on as many host machines as possible. |
| 1,2,3 | 1. Multiple target android device. | Provide meaningful coverage of the wide and fragmented android device market. |
| 1,2,3 | 1. Multiple target android APIs. | Provide meaningful coverage of the wide and fragmented android device market. Support API 19 and 21. |
| 2,3 | 1. Automated interface for app development and testing. | Test applications across multiple target devices and APIs. |
| 2,3 | 1. Minimize boot time for any target android device and API combination. | Customer feels that boot time of android virtual devices greater than 15 seconds is too long for automated test platforms. |
| 1,2,3 | 1. Provide control over emulated devices sensor data via software hooks. | Robust interface for automated sensor data. |
| **Marketing Requirements:**   1. Wide range of app developers. 2. Minimize cost to the app developer. 3. Minimize setup and test time for the app developer. | | |

Appendix: