

Dealing with Device Fragmentation in Mobile Games Testing

Ru Cindrea - Altom Consulting

About me and Altom

@ru_altom

- started as a tester in 2002
- partner and software tester at Altom since 2008
 - software testing services
 - testing training - BBST series online
- into mobile testing, mobile automation and mobile app development
- lately worked with Bitbar on using Testdroid Cloud for mobile games testing using image recognition

Testdroid Cloud from Bitbar



- real devices in the cloud
 - support for most common platforms
 - over **500 unique** devices
 - remote access as well as running scripts
-
- working with mobile games companies
 - helping them develop a test framework
that allows for fast checking of new builds

Mobile Games: The Context and The Challenges



Fragmentation

- Lessons learned from Web and Mobile App testing to deal with fragmentation:
 - ◆ use scripts to automate repetitive checks
 - ◆ choose most common combinations

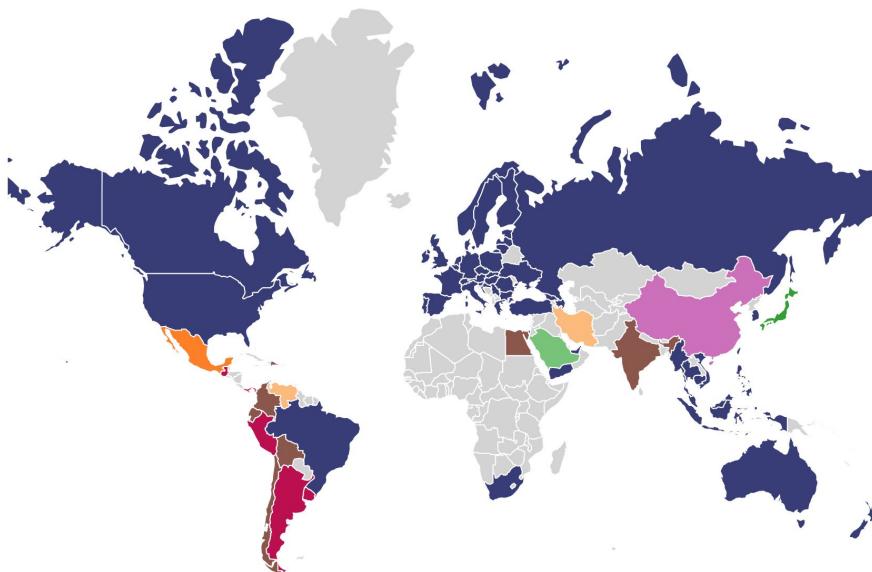
A Definition



WIKIPEDIA
The Free Encyclopedia

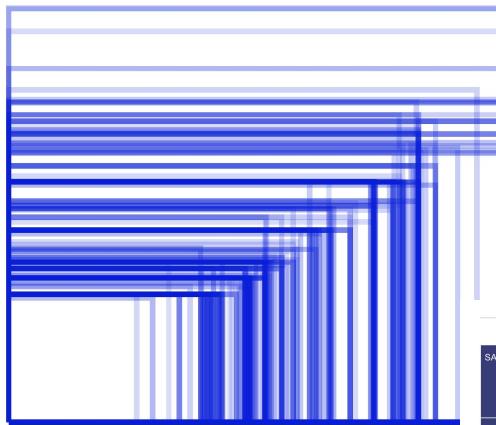
“In computer programming, fragmentation is when a combination of software and hardware do not provide a consistent, top-level experience for the vast majority of its user-base.”

Most Common Where?

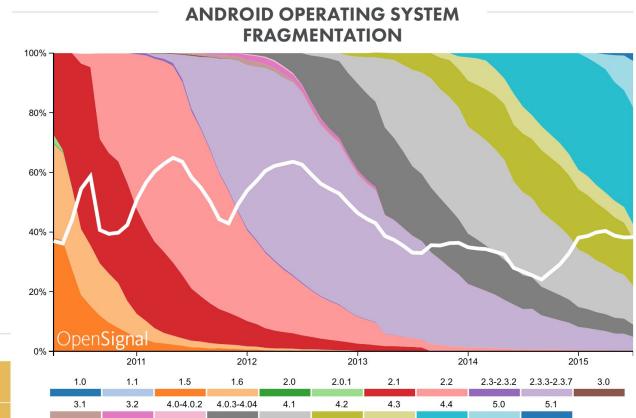
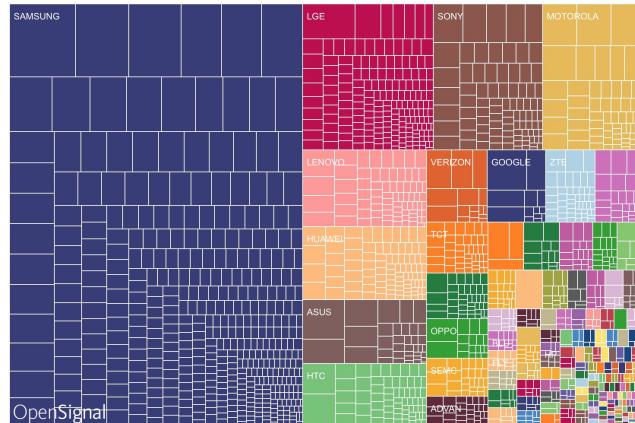


No data HTC TCT Docomo Huawei Xiaomi Sony LG Samsung

Most Common Based on What?



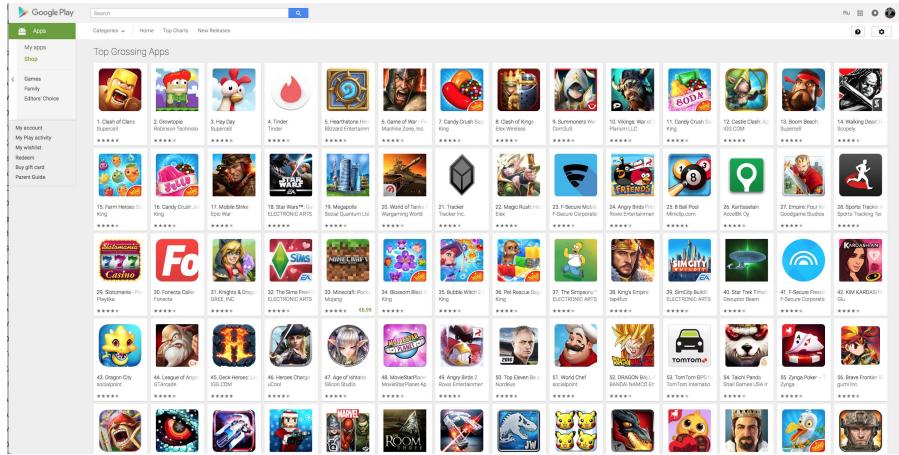
BRAND FRAGMENTATION



Most Common Based on What?

- Graphics/performance can make or break a game
- Factors:
 - ◆ OS Versions
 - ◆ Memory
 - ◆ Resolution
 - ◆ Chipsets
 - ◆ OEM
- Low and mid range devices are important

Incredibly Competitive Market



Over 6 billion USD revenue for the Top 10 Mobile Games in 2015 (<https://www.superdataresearch.com/blog/us-digital-games-market>)



Ilkka Paananen - Supercell CEO:

- **Top 10 companies make 50% of all mobile games revenue**
- **10000 new games are submitted monthly**

Fragmentation

- Lessons learned from Web and Mobile App testing to deal with fragmentation:
 - ◆ use scripts to automate repetitive checks
 - ◆ ~~choose most common combinations~~

no longer restricted to X most common devices, we can test on over 400 of them

Automation Challenges in Mobile Games

Automated UI Scripts Difficult

Game Engines like Unity

- exported binaries for iOS, Android, etc.
=> game is one big canvas
- game engine tools are mostly focused on unit testing and require instrumentation
- “click at coordinates” not feasible

Automated UI Scripts Difficult

Game Engines like Unity

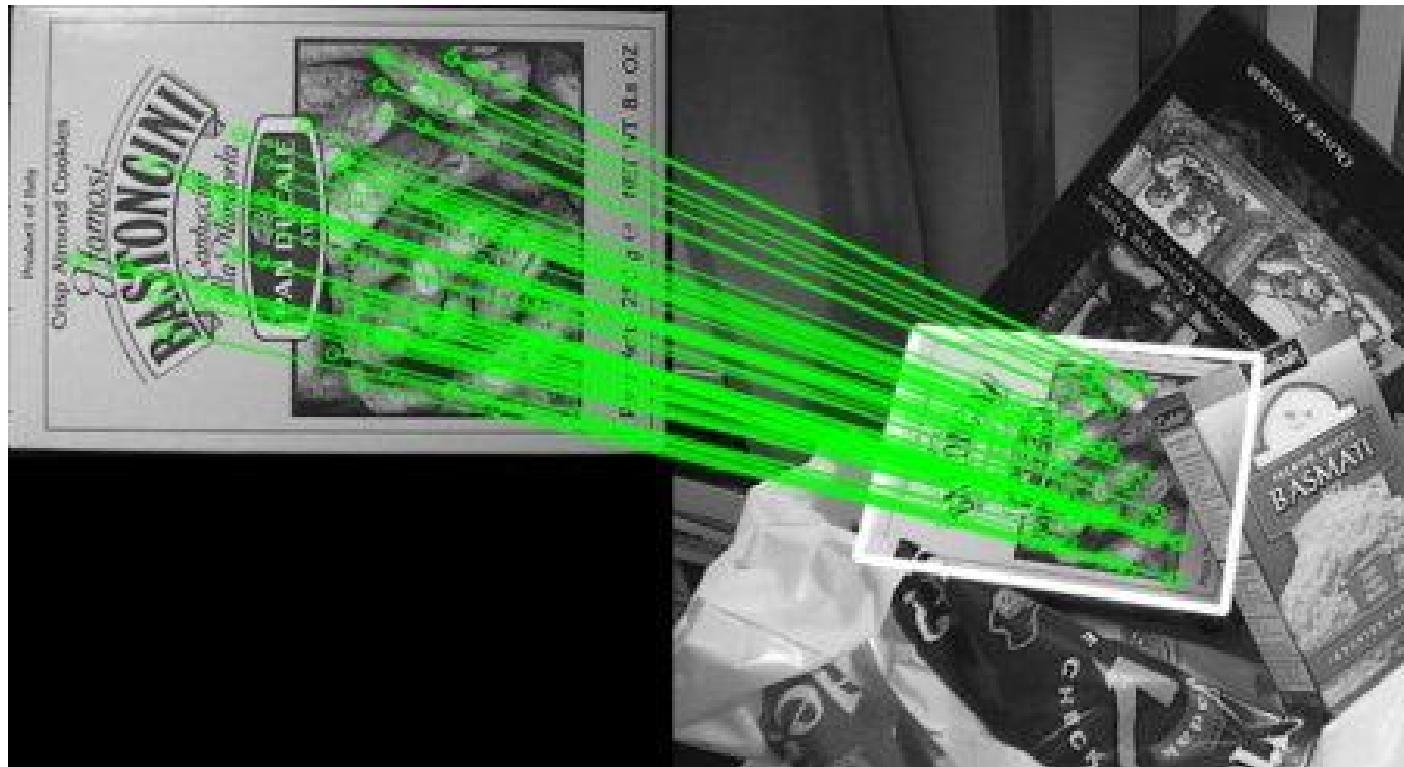
- exported binaries for iOS, Android, etc.
=> game is one big canvas
=> image processing/recognition?
- game engine tools mostly focused on unit testing and require instrumentation
- “click at coordinates” not feasible
=> unless we know the exact coordinates all the time?

Framework Using OpenCV & Appium

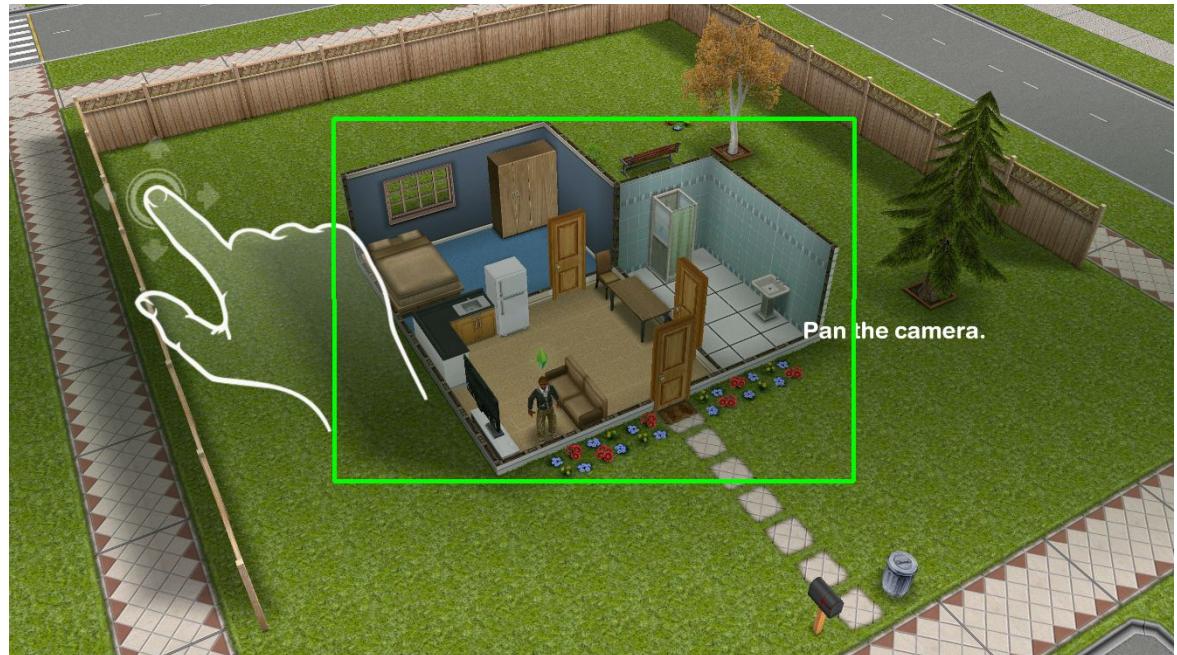
- use OpenCV to find objects on screen
- create a test framework around it that allows clients to easily develop their own scripts
- worked with Bitbar team on developing this framework
- have scripts runnable in Testdroid Cloud

We don't want pixel perfect or exact matches

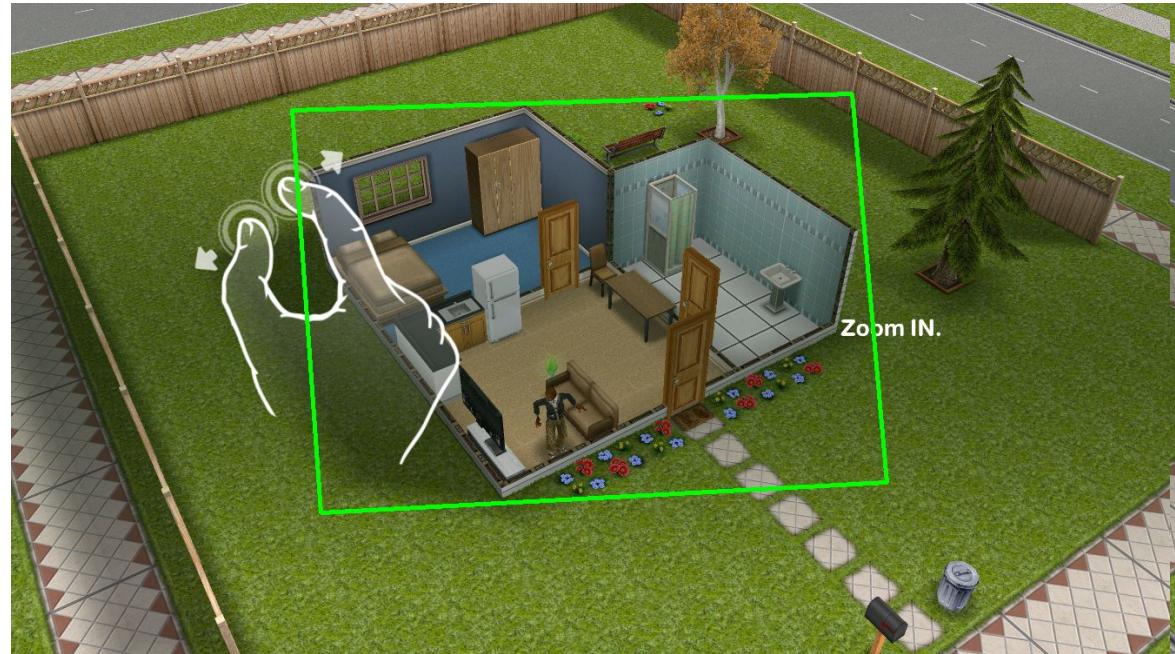
OpenCV: Feature Matching + Homography to find Objects with Akaze Algorithm



OpenCV Akaze Algorithm: Resolution Agnostic, works rotated and stretched items



OpenCV Akaze Algorithm: Resolution Agnostic, works rotated and stretched items



OpenCV Akaze Algorithm: Resolution Agnostic, works rotated and stretched items



OpenCV Akaze Algorithm: Resolution Agnostic, works rotated and stretched items



Set of Query Images = Mobile Elements



SELECT VEHICLE



JEEP



MOTOCROSS BIKE



COST: 75 000 COINS



MORE



GET COINS

NEXT



Mobile Strike

INSTALL

Mobile Elements defined with query images

```
@Test  
public void test02_CheckMenu() throws Exception {  
    findImageOnScreen("car");  
    tapImageOnScreen("more");  
    tapImageOnScreen("back");  
    findImageOnScreen("car");  
    log("PASS -> Main Menu Displayed");  
}
```

Mobile Elements defined with query images

```
@Test  
public void test02_CheckMenu() throws Exception {  
    findImageOnScreen("car");  
    tapImageOnScreen("more");  
    tapImageOnScreen("back");  
    findImageOnScreen("car");  
    log("PASS -> Main Menu Displayed");  
}
```



Mobile Elements defined with query images

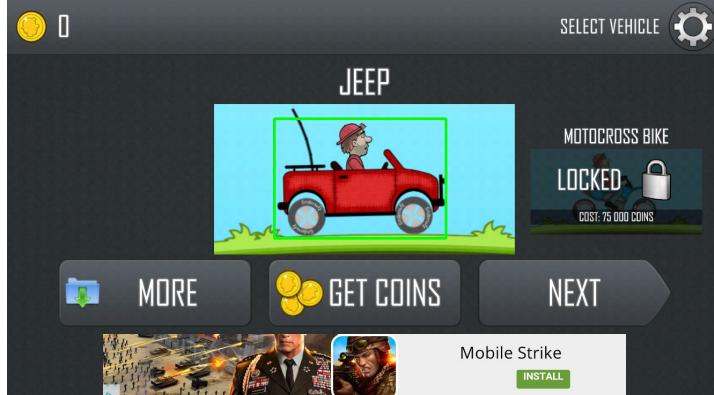
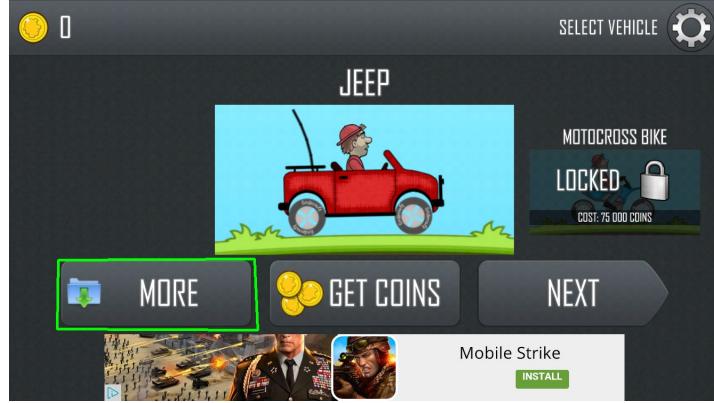
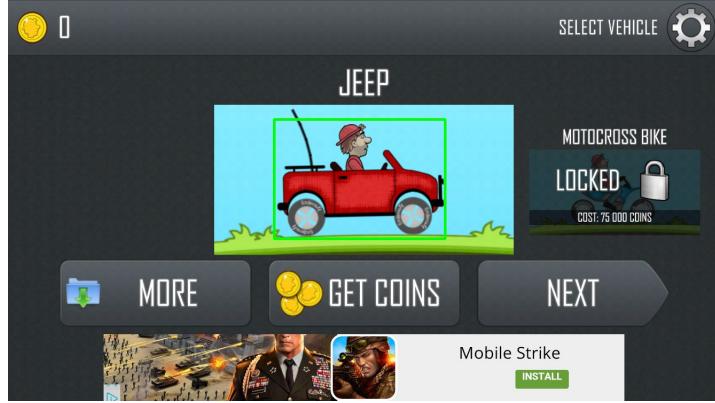
```
@Test  
public void test02_CheckMenu() throws Exception {  
    findImageOnScreen("car");  
    tapImageOnScreen("more");  
    tapImageOnScreen("back");  
    findImageOnScreen("car");  
    log("PASS -> Main Menu Displayed");  
}
```



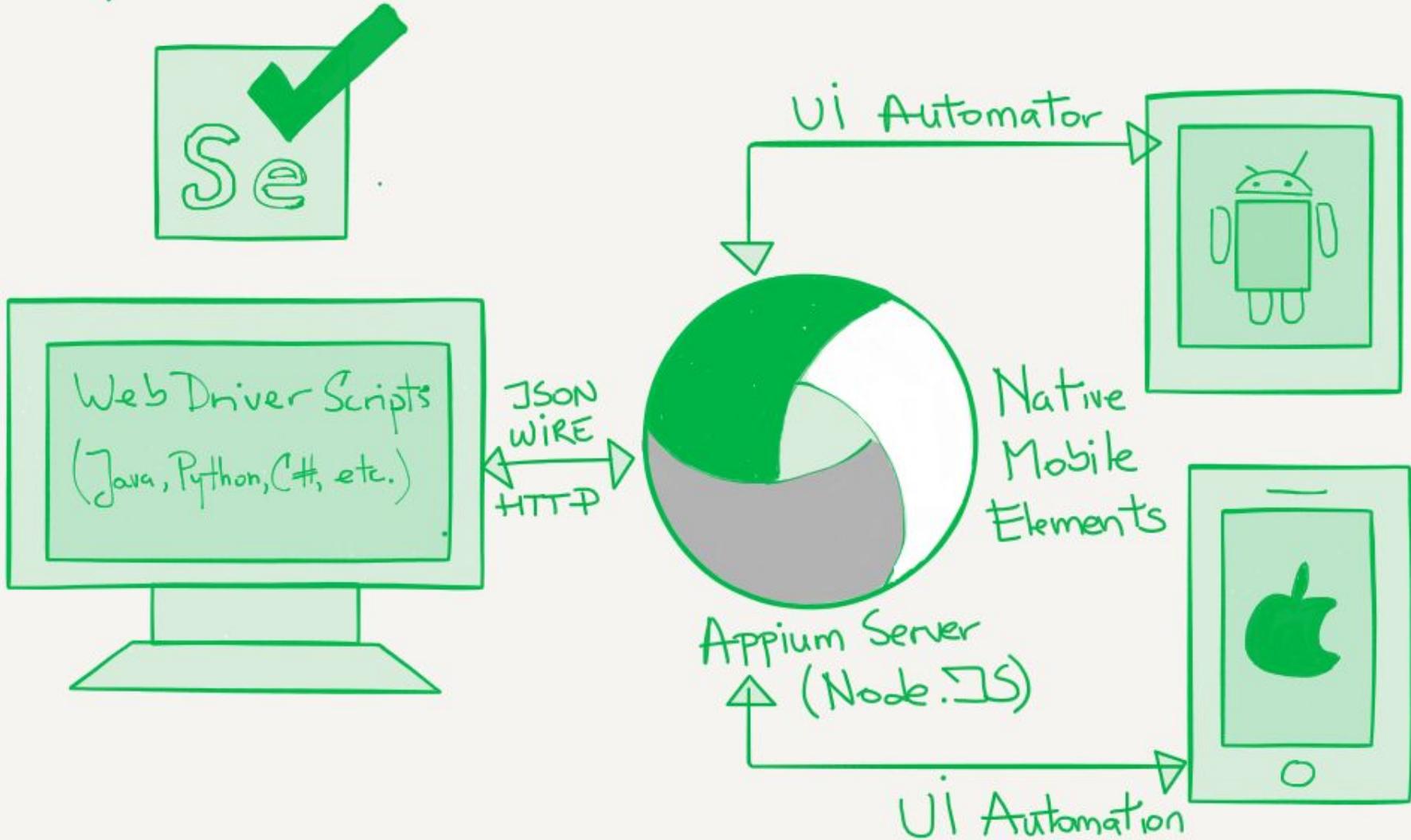
Mobile Elements defined with query images

```
@Test  
public void test02_CheckMenu() throws Exception {  
    findImageOnScreen("car");  
    tapImageOnScreen("more");  
    tapImageOnScreen("back");  
    findImageOnScreen("car");  
    log("PASS -> Main Menu Displayed");  
}
```





Appium



Now the same thing in the Testdroid Cloud



Create new testrun

1. Application
2. Upload test file
3. Select devices
4. Advanced options



1. Application: hcr.apk

file size: 37.3 MB file upload date: 10.01.2016 10:46:18

Update application file:

*Use this field to upload or update your application file. Please use the *.apk files only.*

no file selected

3. Select devices:



Browse devices (28)

Search by device name

Show by: [@clear all](#)

iOS VERSIONS

API Level

- 1
- 10
- 12
- 13
- 15
- 16
- 17
- 18
- 19
- 21
- 22
- 23
- 7
- 8
- 9

CPU

	Asus Google Nexus 7 (201... ANDROID 5.1.1 Quad-core 1.5 GHz Krait 2048 MB WUXGA (1920 x 1200) <input type="checkbox"/>		Asus Google Nexus 7 ME... ANDROID 5.1.1 Quad-core 1.3 GHz / ARMv7-A 1024 MB WXGA (1280 x 800) <input type="checkbox"/>
	Dell Venue 8 7840 -US ANDROID 5.1 Dual-core 2.1 GHz Intel Silvermont / IA-32 (x86), IA-64 (x64), SSE4, SSE 4.1, SSE 4.2 2048 MB WUXGA (1920 x 1200) <input type="checkbox"/>		HTC One M9 ANDROID 5.1 Quad-core 1.5 GHz Cortex-A53 ~ Quad-core 2 GHz Cortex-A57 / ARMv8-A 3072 MB Full HD (1920 x 1080) <input type="checkbox"/>
	LG G4 ANDROID 5.1 Quad-core 1.44 GHz Cortex A57 ~ Dual-core 1.82 GHz Cortex A57 ARMv8-A 3072 MB QHD (2560 x 1440) <input type="checkbox"/>		LG G4 F500L ANDROID 5.1 Quad-core 1.44 GHz Cortex A57 ~ Dual-core 1.82 GHz Cortex A57 ARMv8-A 3072 MB QHD (2560 x 1440) <input type="checkbox"/>
	LG Google Nexus 5 D820 ... ANDROID 5.1.1 Quad-core 2.3 GHz Krait 400 / ARMv7 2048 MB Full HD (1920 x 1080) <input type="checkbox"/>		Micromax Canvas A1 AQ4... ANDROID 5.1.1 Quad-core 1.3 GHz Cortex-A7 / ARMv7 1024 MB FWVGA (854 x 480) <input type="checkbox"/>

- Create new groups of devices
 - filters
 - platform
 - API Levels
 - resolutions
 - etc

Device statuses

Search devices



Device	Status	Installing application	Launching application	Test execution	Test cases passed
Acer Iconia Tab A1-810	succeeded ✓	N/A ✓	N/A ✓	4m 12s ✓	3/3 ✓
Asus Google Nexus 7 ME370T	succeeded ✓	N/A ✓	N/A ✓	5m 41s ✓	3/3 ✓
HTC Desire 510	succeeded ✓	N/A ✓	N/A ✓	3m 43s ✓	3/3 ✓
HTC Desire 516	succeeded ✓	N/A ✓	N/A ✓	3m 45s ✓	3/3 ✓
HTC Google Nexus 9 5.0.1 - ...	succeeded ✓	N/A ✓	N/A ✓	4m 24s ✓	3/3 ✓
HTC Google Nexus 9 6.0 EU	tests failed ✘	N/A ✓	N/A ✓	4m 46s ✓	2/3 ✘
HTC One M7 4.3	succeeded ✓	N/A ✓	N/A ✓	5m 0s ✓	3/3 ✓
HTC One Mini 2	succeeded ✓	N/A ✓	N/A ✓	6m 34s ✓	3/3 ✓
Lenovo Lemon K3 K30-T	succeeded ✓	N/A ✓	N/A ✓	4m 29s ✓	3/3 ✓
LG G Flex 2 H955	succeeded ✓	N/A ✓	N/A ✓	4m 40s ✓	3/3 ✓

Show all 20 devices ▾



Execution status: Succeeded ✓

Test cases passed: 3/3



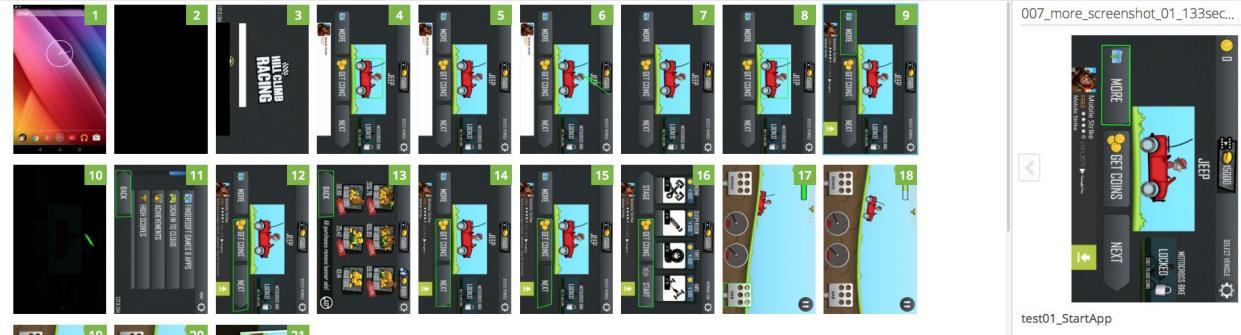
test01_StartApp

test02_CheckMenu

test03_StartDriving

Steps in test01_StartApp method
No steps information available.

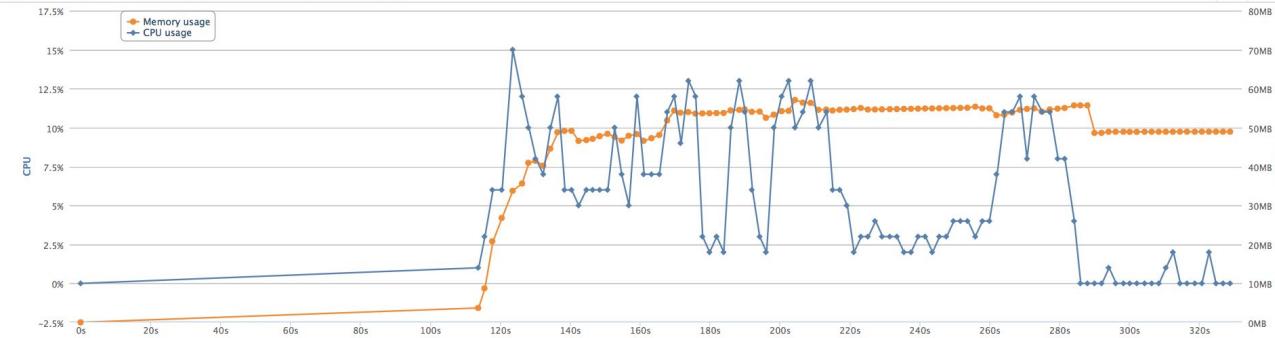
Screenshots by Test Steps



007_more_screenshot_01_133sec...

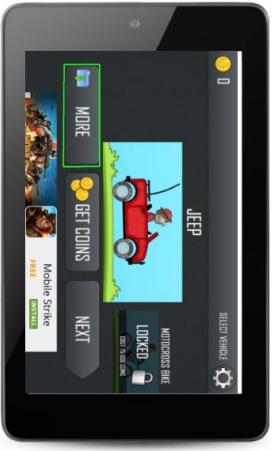
test01_StartApp

Performance for Asus Google Nexus 7 ME370T 5.1.1





Acer Iconia Tab A1-810



Asus Nexus 7 5.1.1



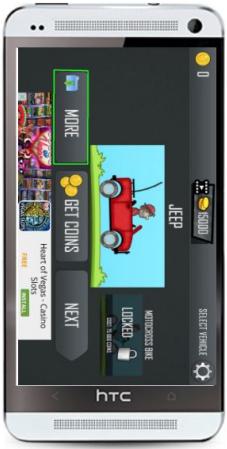
HTC Desire 516 dual sim



HTC Google Nexus 9 - US



HTC Google Nexus 9 6.0 EU



HTC One M7 4.3



HTC One mini 2



Lenovo Lemon K3 K30-T

Conclusions

Some conclusions

- very reliable
- types of problems found:
 - out of memory
 - crashes
 - graphics missing/not displayed correctly
- start with simple scenarios
- allow for fast checking by a person after each run rather than trying to verify everything automatically

Thank you!

@ru_altom

ru.cindrea@altom.fi

QUESTIONS?