Smart Android GUI Testing Approaches

Yavuz Koroglu Alper Sen

Department of Computer Engineering
Bogazici University, Istanbul/Turkey
yavuz.koroglu@boun.edu.tr
depend.cmpe.boun.edu.tr

November 6, 2017

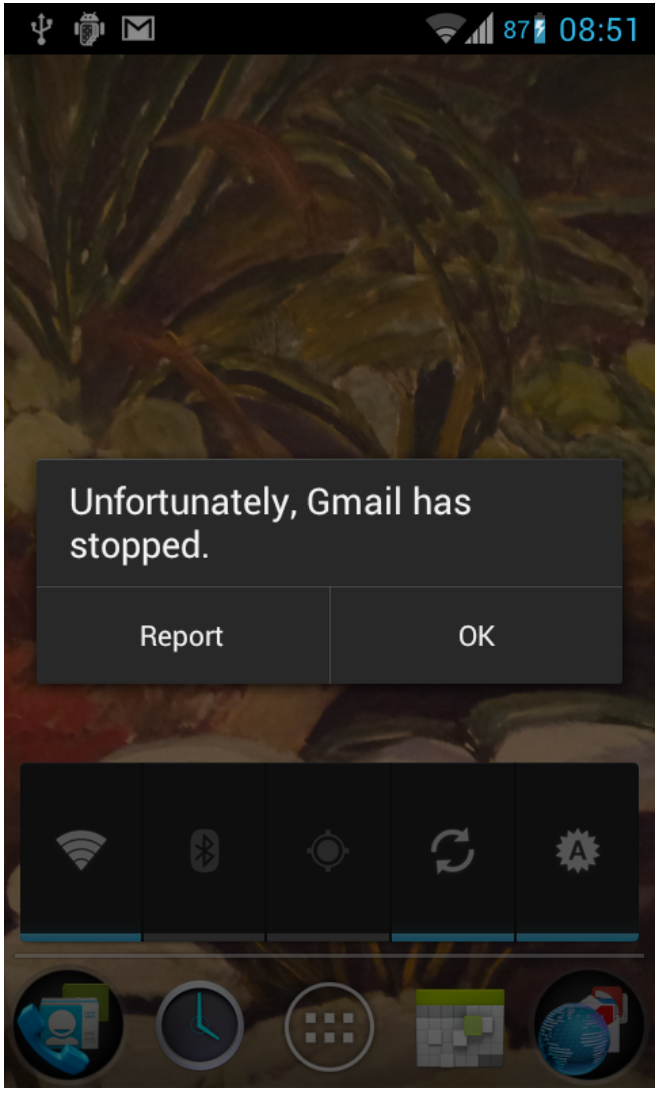
Overview

Android Usage Today

- We use phones 3 hours/day.

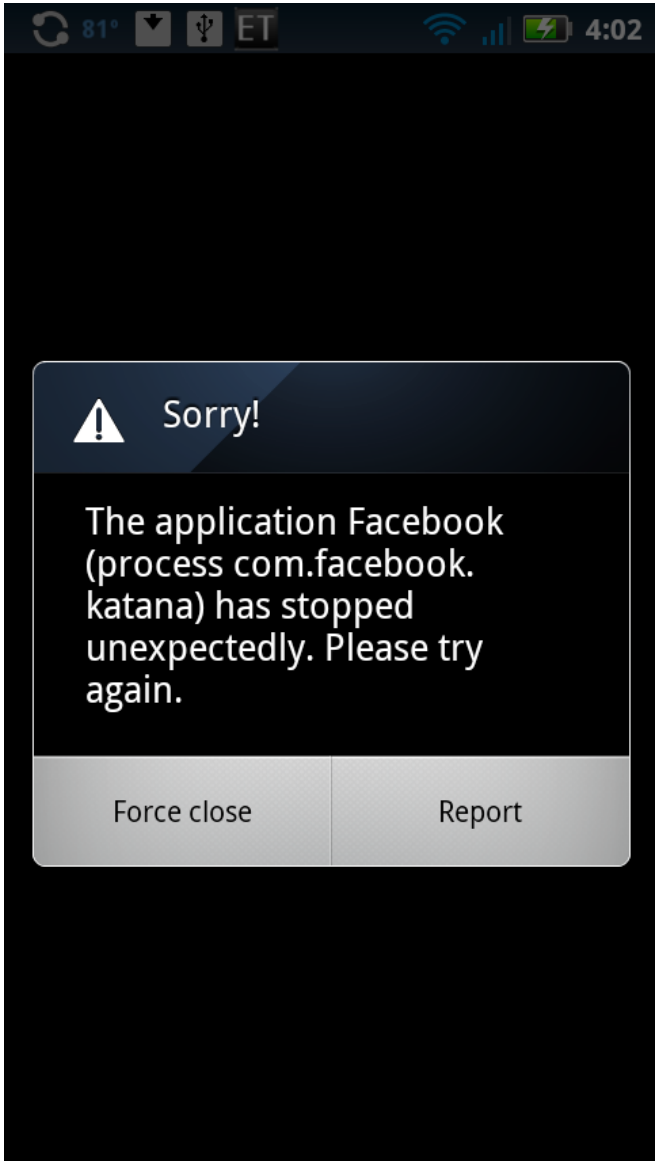
Android Usage Today

- We use phones 3 hours/day.
- We constantly get error messages.

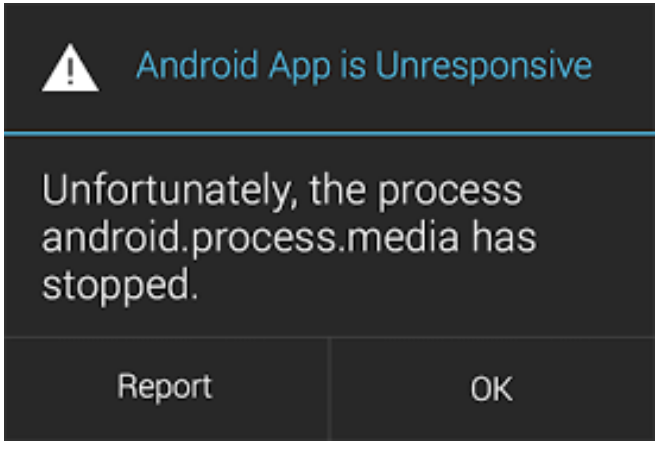


Android Usage Today

- We use phones 3 hours/day.
- We constantly get error messages.



Android Usage Today



- We use phones 3 hours/day.
- We constantly get error messages.

Android Usage Today

- We use phones 3 hours/day.
- We constantly get error messages.

Fully-Automated Android GUI Testing

GUI Testing

- Click buttons,
- Fill textboxes,
- Drag & drop,
- Swipe,
- Toggle WiFi etc.

Automation is a MUST

$>$ 2.2M Applications in the Android market.

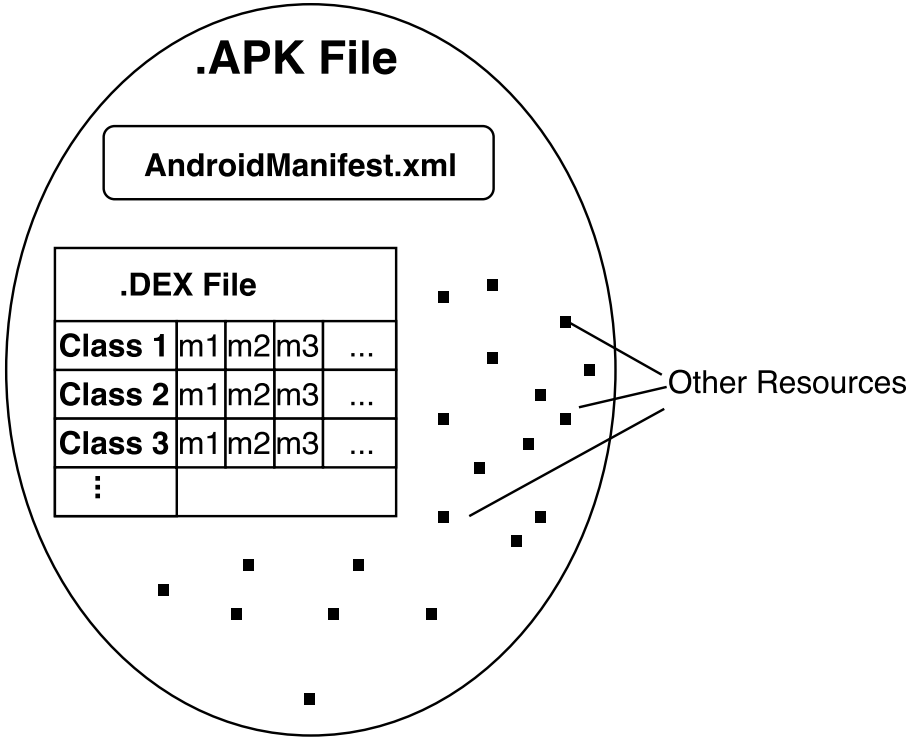
Internal Structure of an Android Application (.APK File)

Structure Overview

1. Executable .DEX file,
2. AndroidManifest.xml, and
3. Other resources - pictures, sounds etc.

AndroidManifest.xml

- Activity Names,
- Launchable Activities, and
- Permissions.



.DEX File

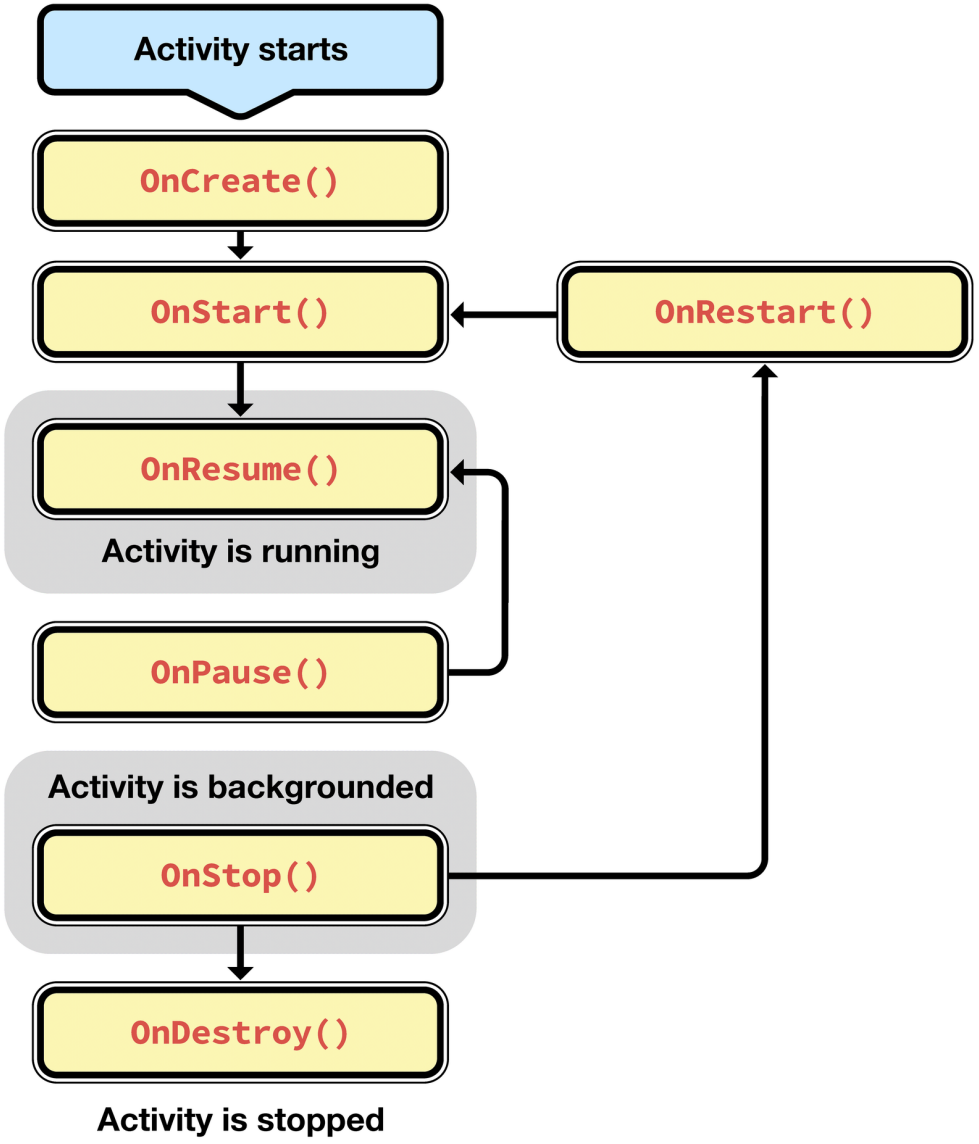
- Formed by **Java classes**.
- Each class has methods.

Java Classes for Android

Class Categories

1. **Activity:** Represents **different screens** of the application.
   - **Launchable Activity:** The **first** activity of the application.
2. **Service:** Represents tasks that runs in **background**. Started and stopped from activities.
3. **Content Provider:** Dynamically presents the information provided by various services to the activity.
4. **Broadcast Reciever:** Triggered by **external events** (SMS, GPS, clock timeout etc.) and activates specific code segments. Activities do NOT trigger them.
5. **Other Classes:** All other classes that inherit `java.lang.Object`.

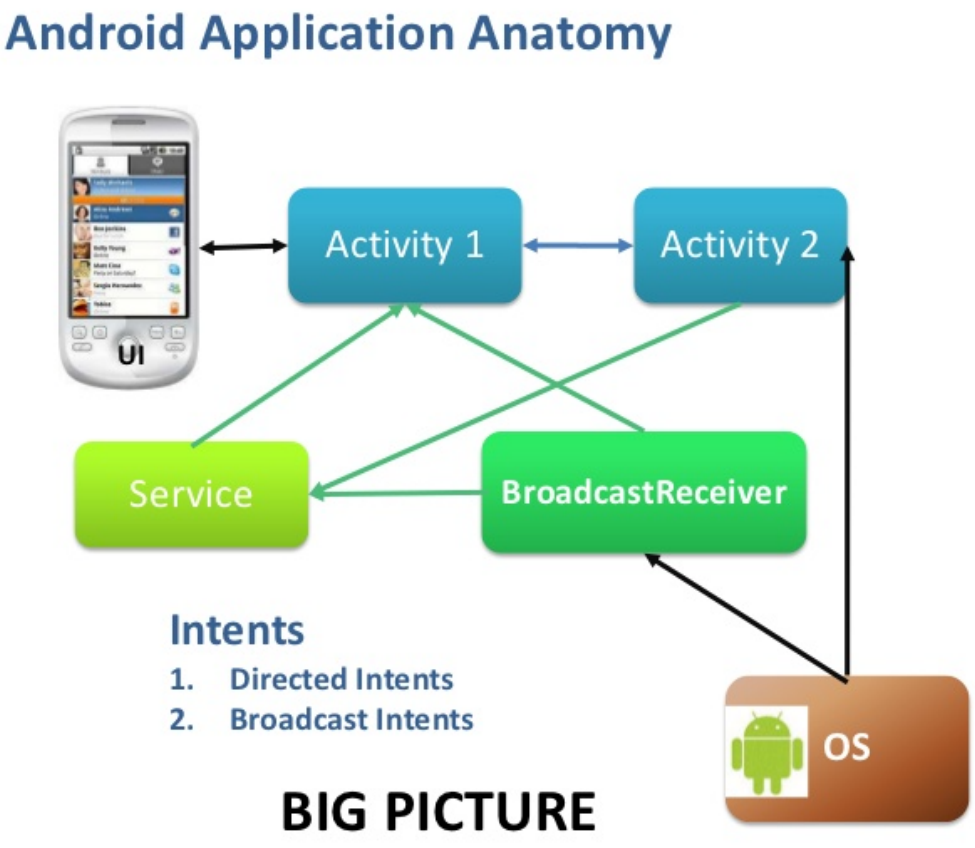
Activity Life-Cycle



Properties of Activities

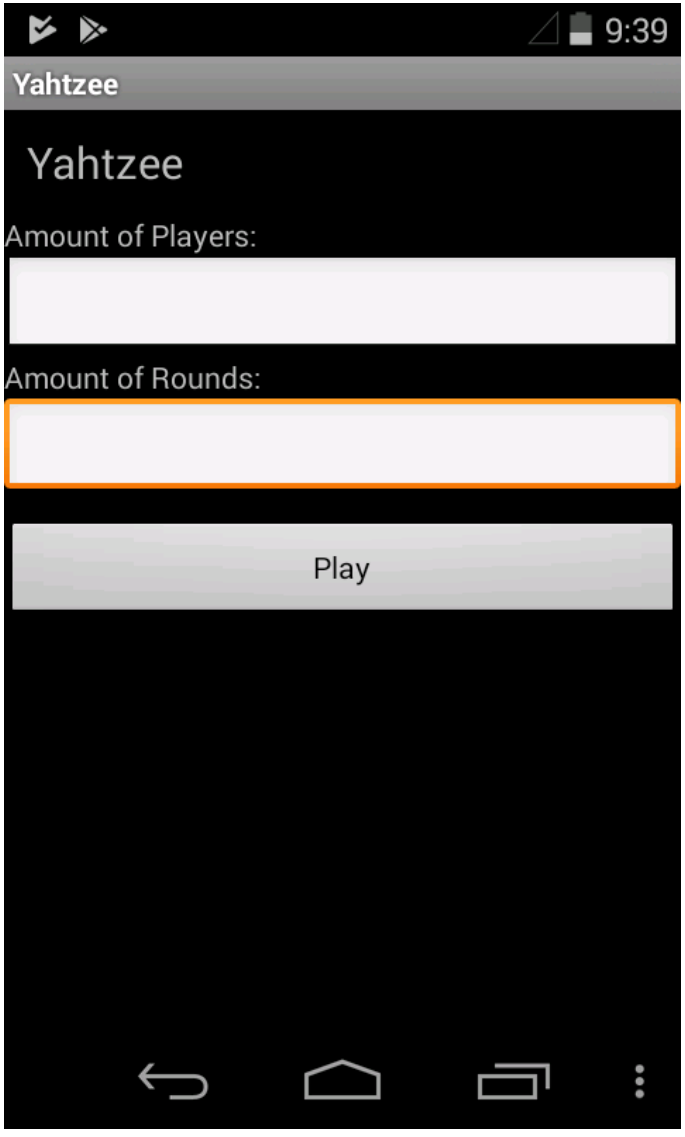
- Defaults expected to be **overwritten**.
- Developers **depend on defaults**.
- **Error-prone**.

Services, Content Providers, and Broadcast Recievers



- OS fires **events**.
- BroadCast recievers and the target activity **recieve** the event.
- Event recievers **trigger** services and other activities.
- Content providers are **intermediaries** between services and activities.

Execution of an Android Application



GUI State

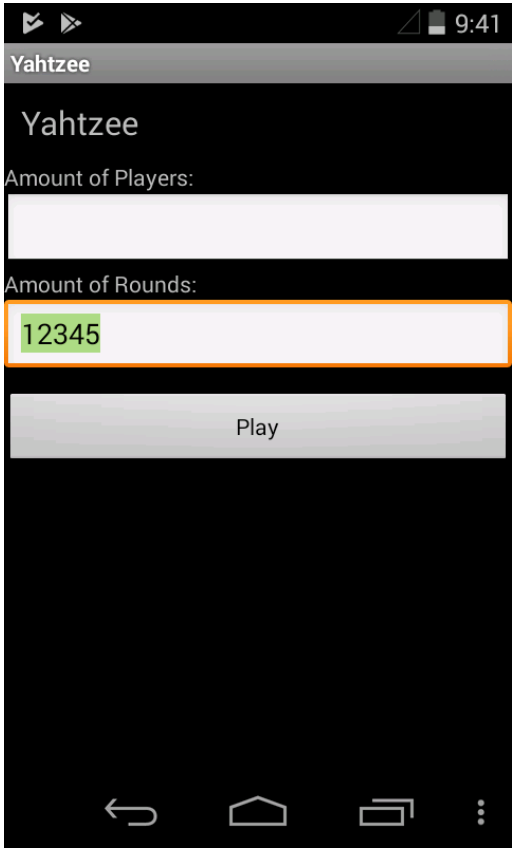
Concatenation of the following:

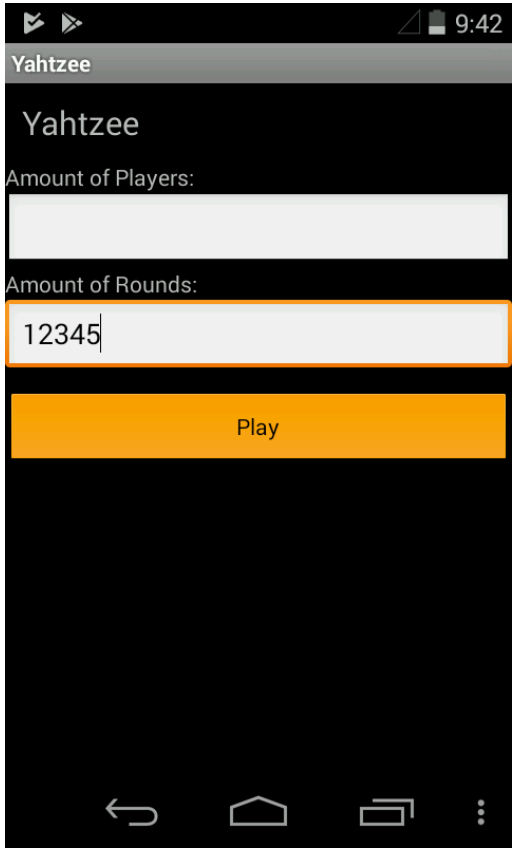
1. Java Package Name,
2. Activity Name,
3. Contextual States (WiFi, Orientation etc.),
4. GUI Components - Thier sizes, labels, and accesibility.

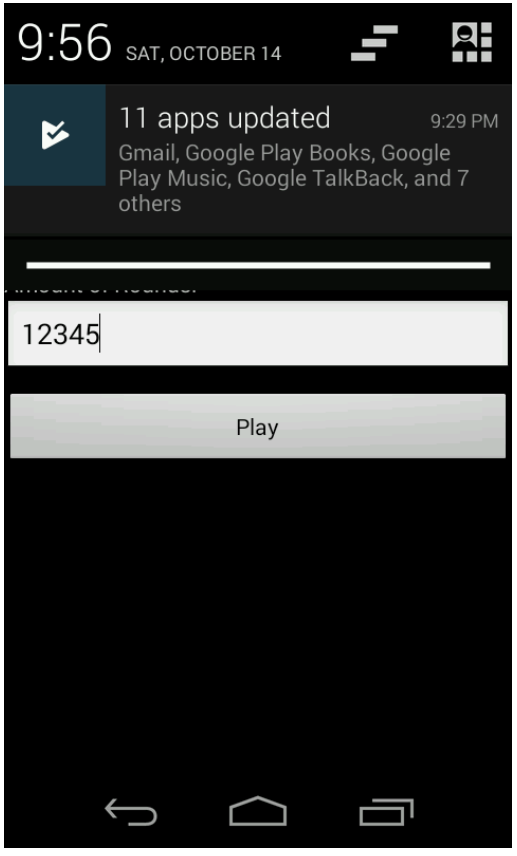
Execution of an Android Application

GUI Action

Actions performed by a user: **text, click, swipe** etc.







List of GUI Actions

Tablo: List of all GUI Actions

Non-Contextual	Param1	Param2	Param3	Param4	Param5
click	x	y	-	-	-
longclick	x	y	-	-	-
text	x	y	string	-	-
swipe	x1	y1	x2	y2	duration
menu	-	-	-	-	-
back	-	-	-	-	-

Contextual	Parameter
connectivity	on/off/toggle
bluetooth	on/off/toggle
location	gps/gps&network/off/toggle
planemode	on/off/toggle
doze	on/off/toggle

Special	Param1	Param2	Param3	Param4	Param5
reinitialize	package	activity	-	-	-

Automatic Remote Control of Android

Proceed to DEMO.

Monkey

What does Monkey do?

- Randomly generates
  1. **System events** and
  2. **GUI actions**.
- Comes with the Android OS.
- **Very fast**, thousands of actions in a second.

Monkey Pros/Cons

Pros

- Speed
- Many kinds of actions

Cons

- Unrealistic input
- Can't go to deep into the application.

Monkey Pros/Cons

→ **menu**

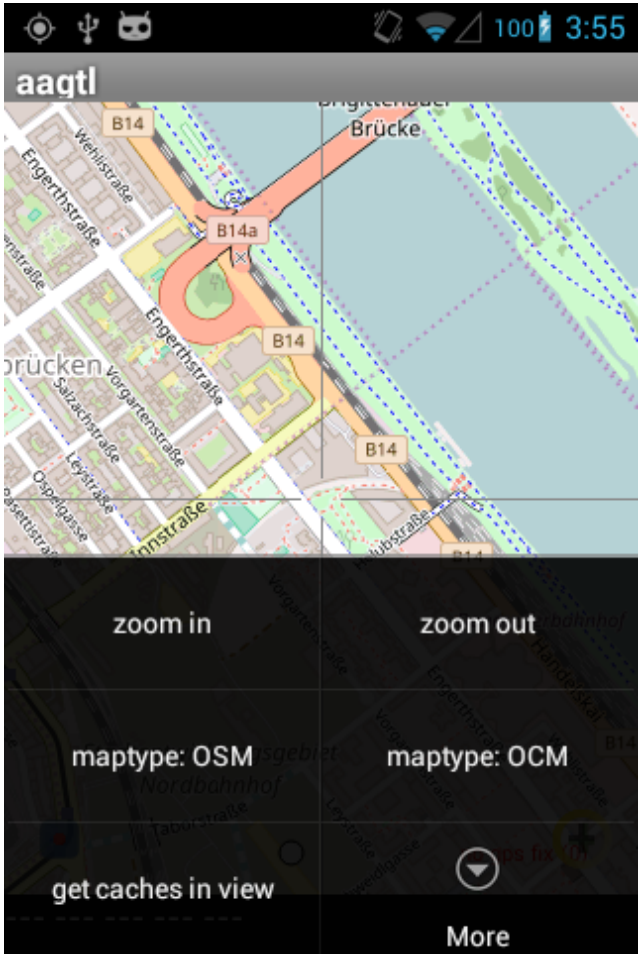
Pros

- Speed
- Many kinds of actions

Cons

- Unrealistic input
- Can't go to deep into the application.

Monkey Pros/Cons



→ **click** More

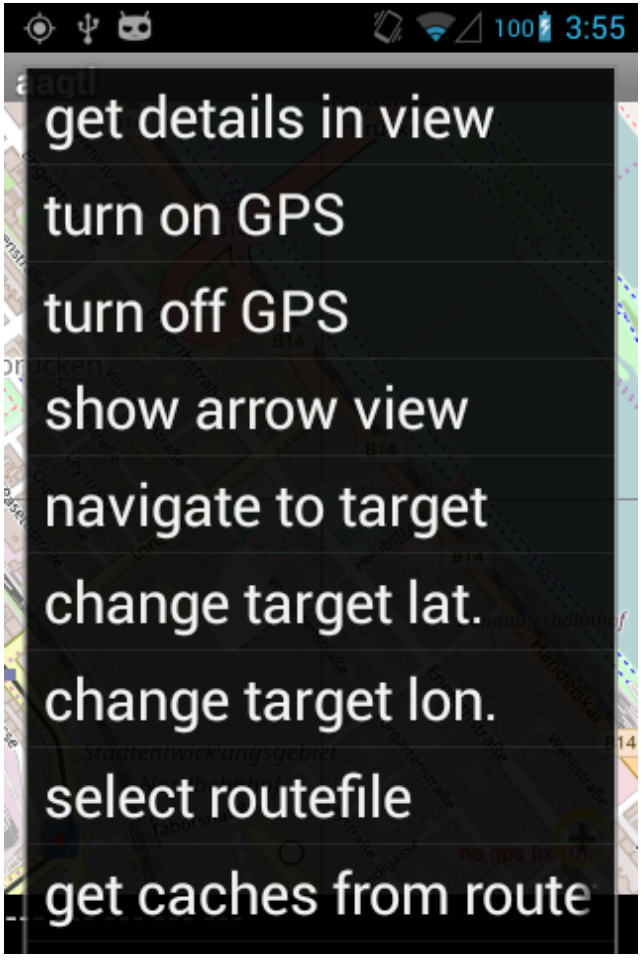
Pros

- Speed
- Many kinds of actions

Cons

- Unrealistic input
- Can't go to deep into the application.

Monkey Pros/Cons



→ **click** show arrow view

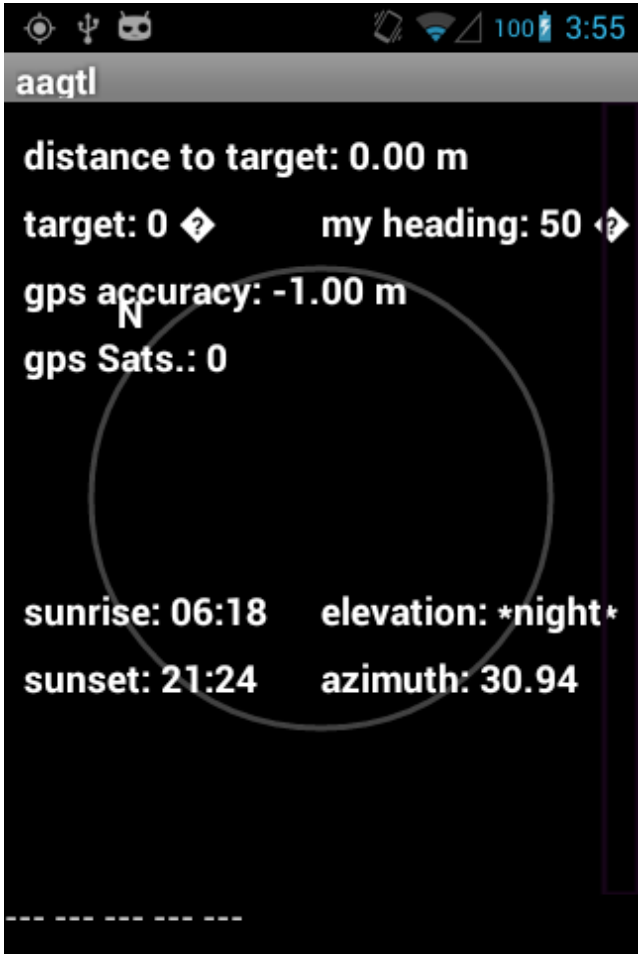
Pros

- Speed
- Many kinds of actions

Cons

- Unrealistic input
- Can't go to deep into the application.

Monkey Pros/Cons



$\rightarrow$ **menu**

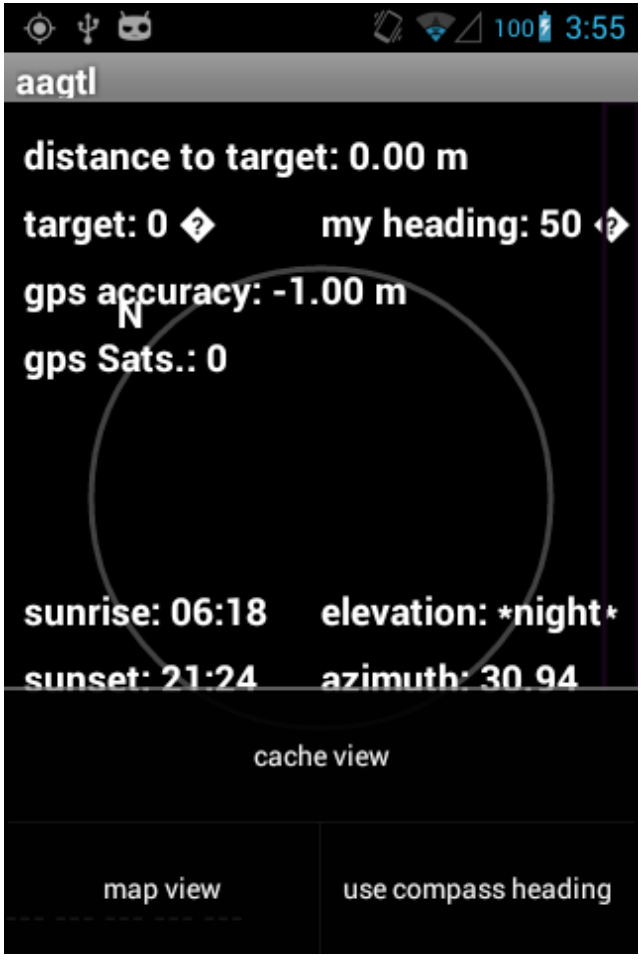
Pros

- Speed
- Many kinds of actions

Cons

- Unrealistic input
- Can't go to deep into the application.

Monkey Pros/Cons



→ **click** cache view

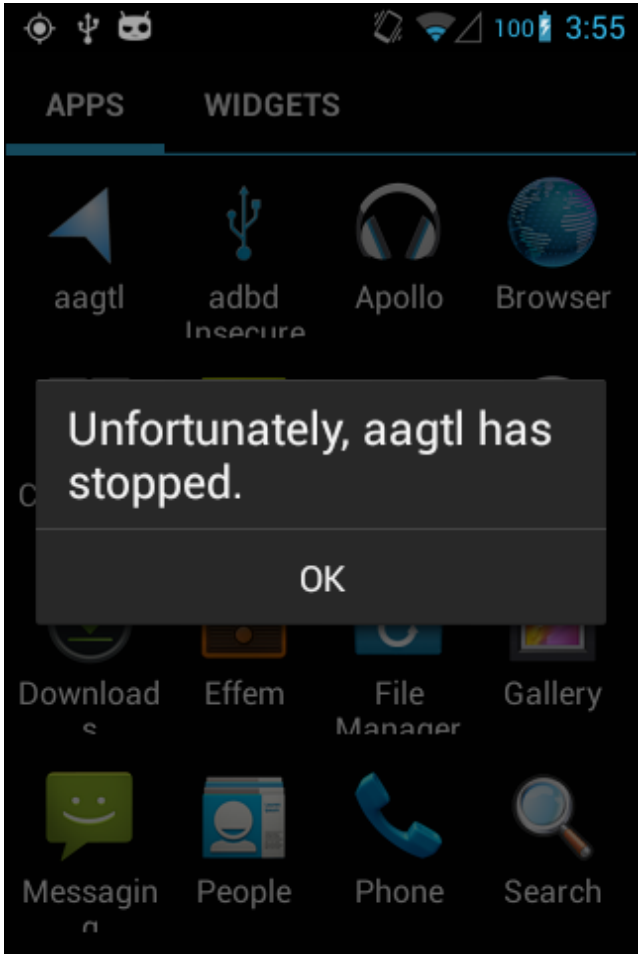
Pros

- Speed
- Many kinds of actions

Cons

- Unrealistic input
- Can't go to deep into the application.

Monkey Pros/Cons



→ Monkey CAN'T detect the crash. Too deep in the app.

Pros

- Speed
- Many kinds of actions

Cons

- Unrealistic input
- Can't go to deep into the application.

Monkey DEMO

Proceed to DEMO.

Other Tools in the Literature

Publicly Available Tools

1. **A^{3}E :** Targeted Exploration. Uses **Static Activity Transition Graph (SATG)** to test yet unexplored activities or test the activities that have a transition to unexplored activities.
2. **DynoDroid :** A random tester that gives bias towards relevant events that **trigger relevant methods**.
3. **SwiftHand :** Learns a **finite-transition model** of the Application Under Test (AUT) to **minimize restarts**.
4. **PUMA :** Introduces **cosine similarity** between GUI states.
5. **Sapienz :** Uses **evolutionary algorithms** to generate test cases.

!! None of the tools detect as many crashes as **Monkey**.

Measures of Testing Tool Performance

Crashes

- **Number of Crashes Detected :** The main goal of testing is to detect as **many crashes** as possible.
- **Number of Distinct Crashes :** Testing tools may **abuse** the same crash for performance increase. Must count **each crash once**.

How to compare crashes?

- Get related Android system logs via built-in **LogCat**.
- **Assumption:** Similar stack traces correspond to the same crash.

Measures of Testing Tool Performance

Coverage

What if,

- There is **no crash**, or
- Testing tools have the **same crash performance**?

Then, measure how much of the application is covered.

High-Level Coverage

- Activity Coverage (used)
- Widget Coverage (not used)
- Event Coverage (not used)
- State Coverage (not used)

Low-Level Coverage

- Class Coverage (not used)
- Method Coverage (used)
- Branch Coverage (used)
- Statement Coverage (used)

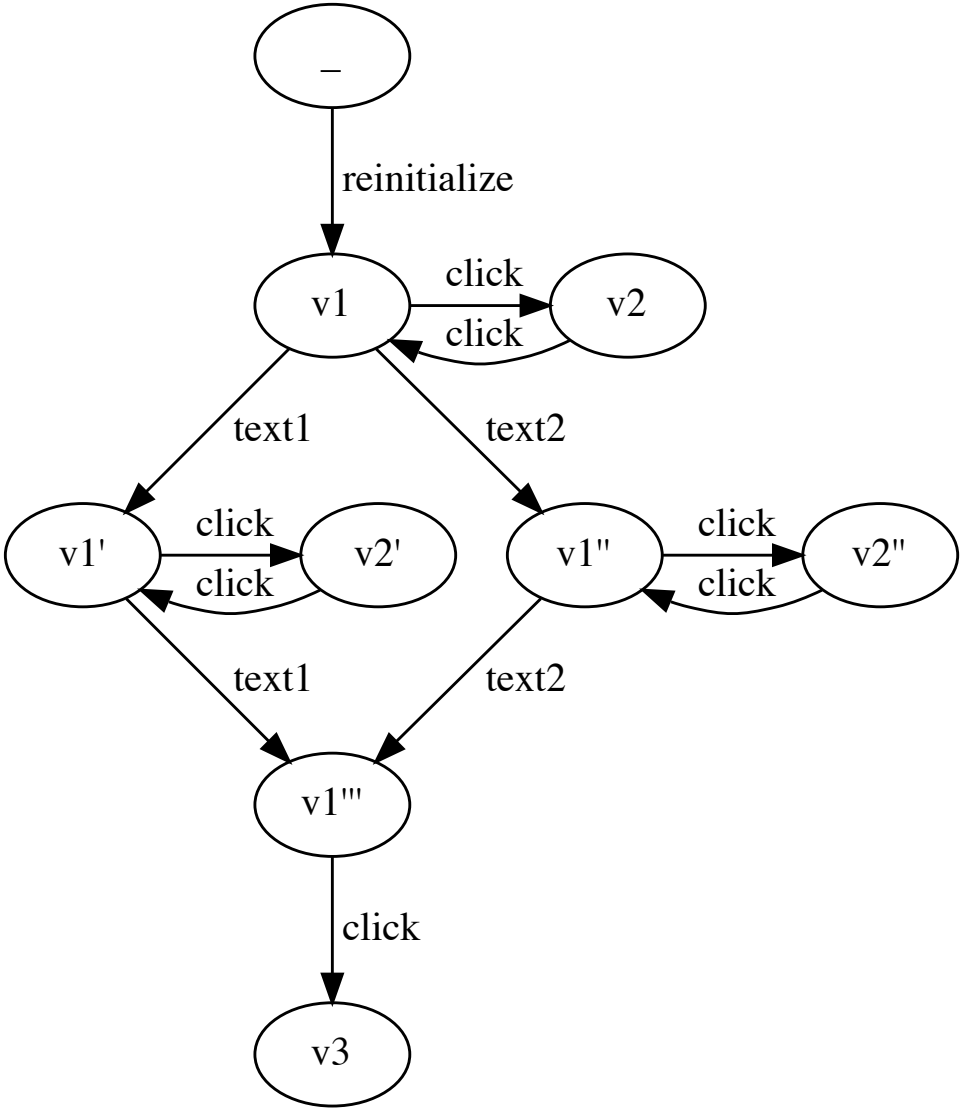
AndroFrame

What is AndroFrame?

- **Fully automated**,
- **Model learning**, and
- **Black-box**

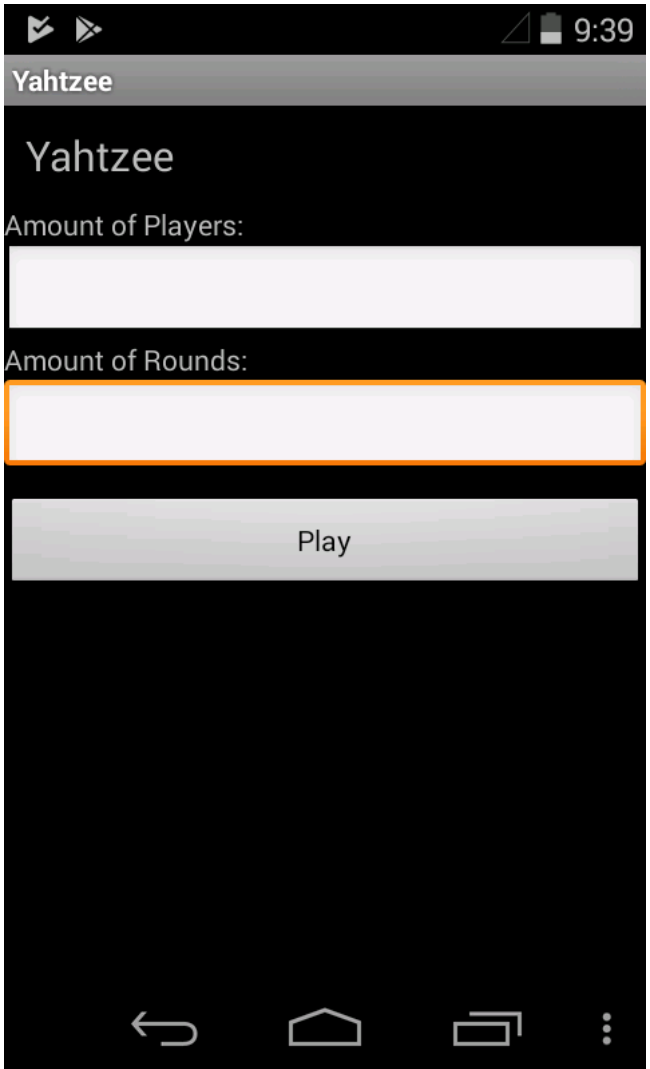
Features

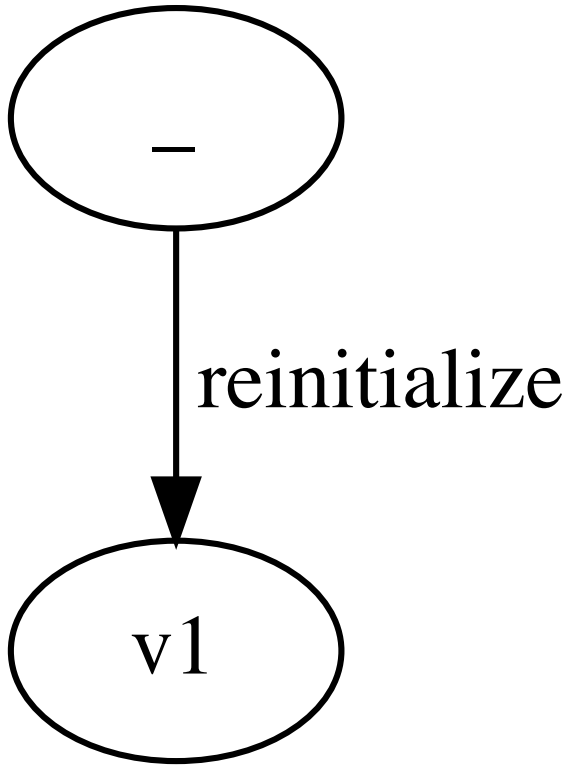
- Extended Labeled Transition System (ELTS).
- **Action Decisions:** Machine-Learning Based.



AndroFrame Example

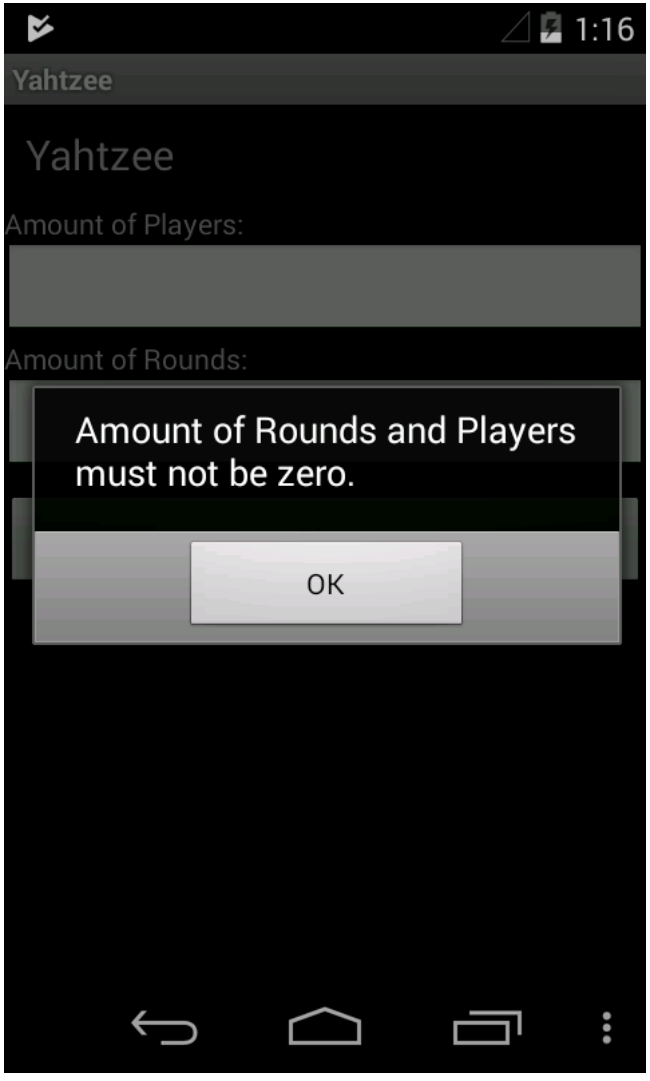
Action: reinitialize com.tum.yahtzee MainActivity

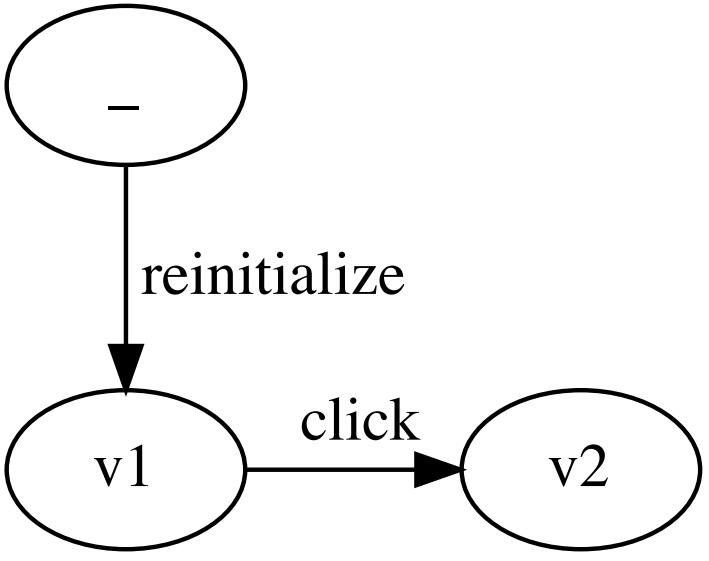




AndroFrame Example

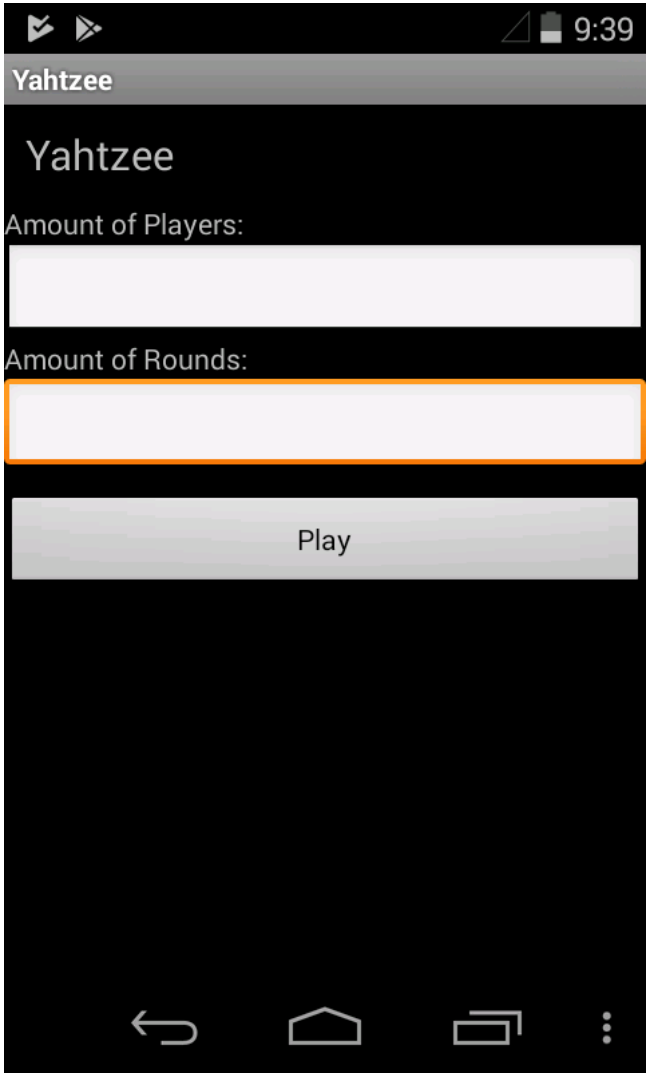
Action: click 200 390 (click play)

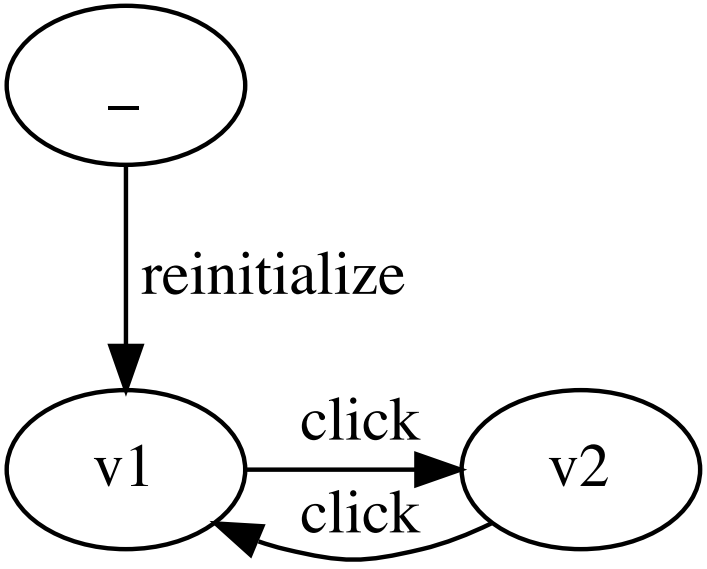




AndroFrame Example

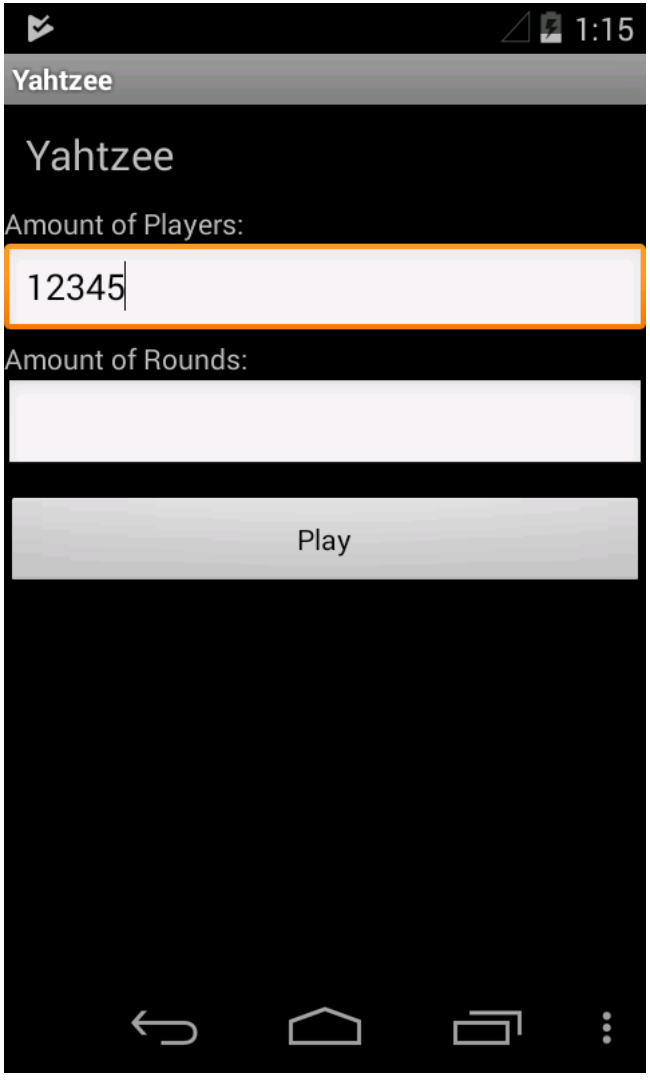
Action: click 200 410 (click ok)

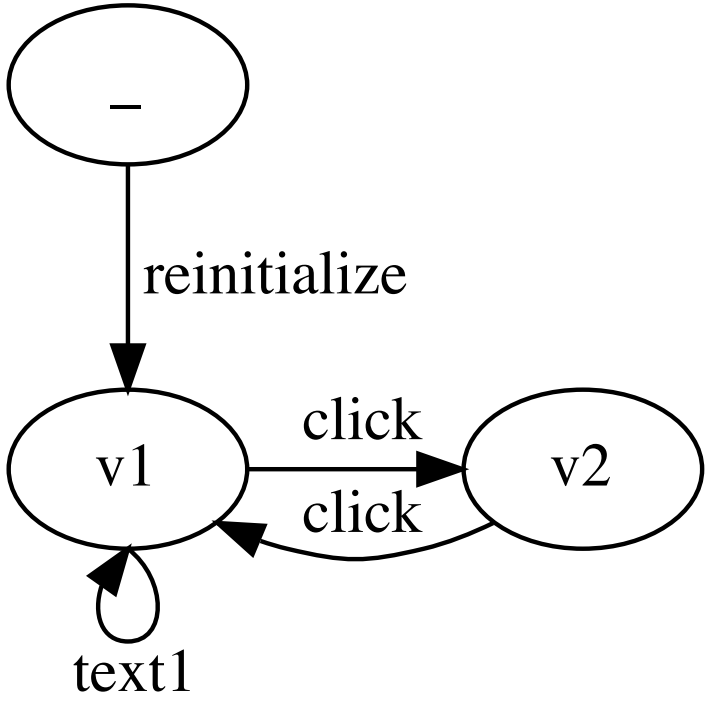




AndroFrame Example

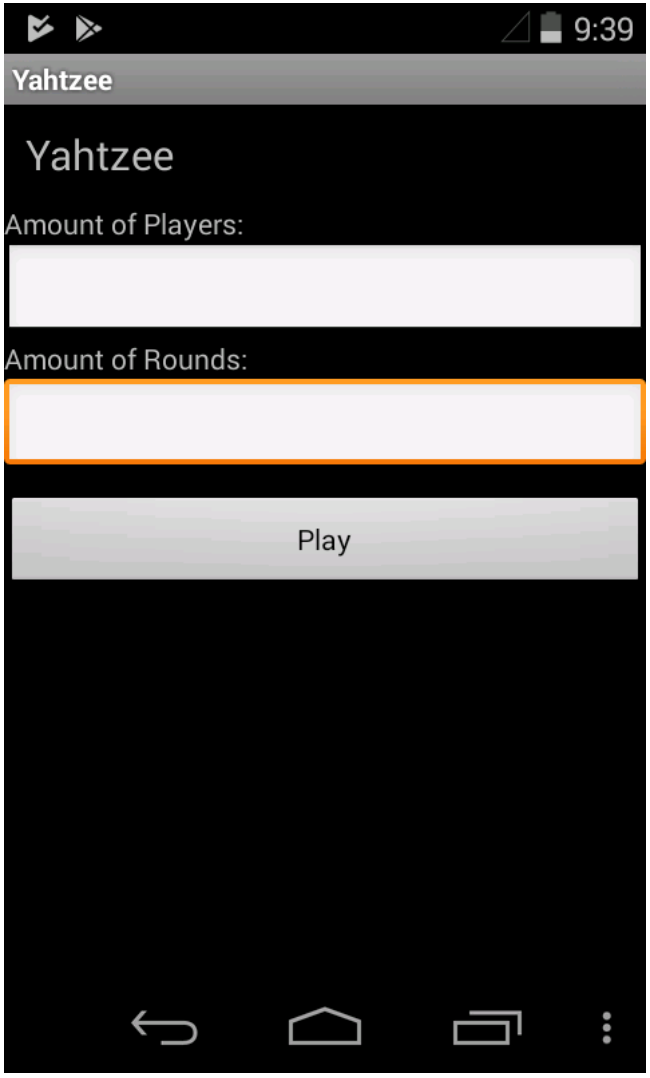
Action: text 200 270 12345 (text1)

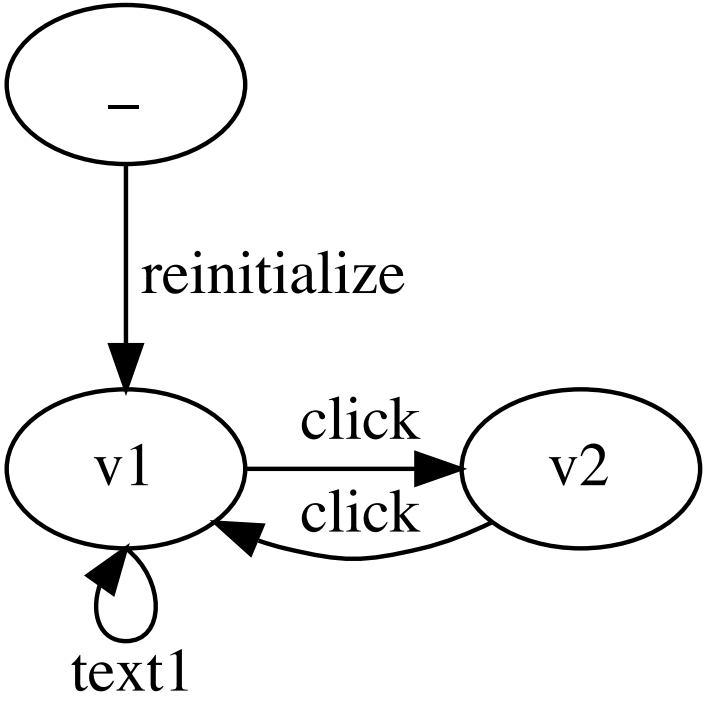




AndroFrame Example

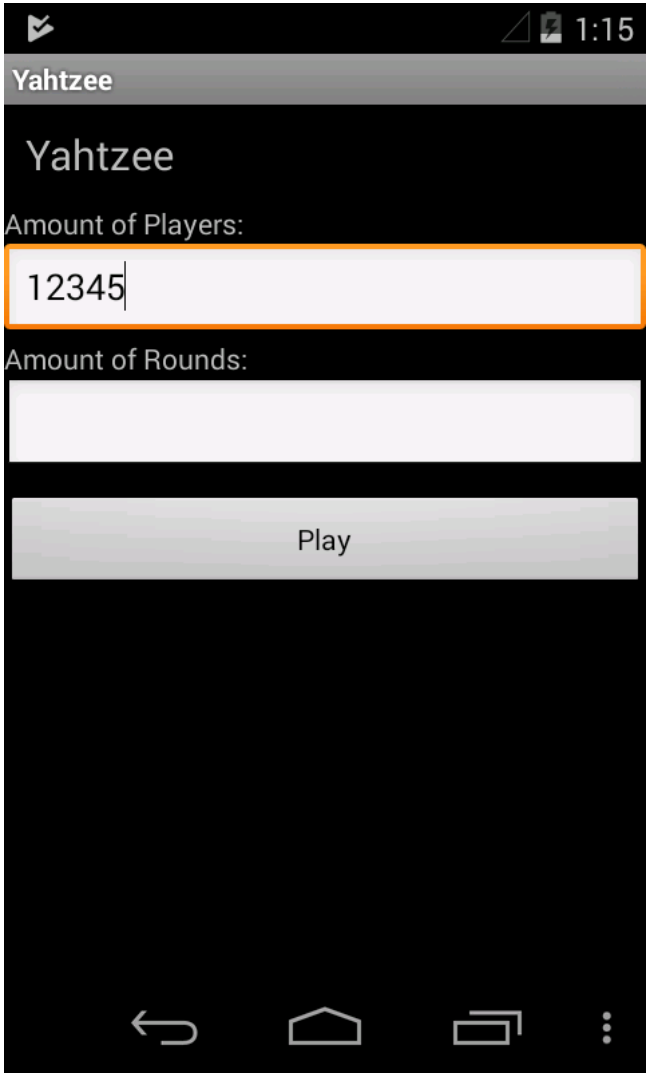
Action: reinitialize com.tum.yahtzee MainActivity

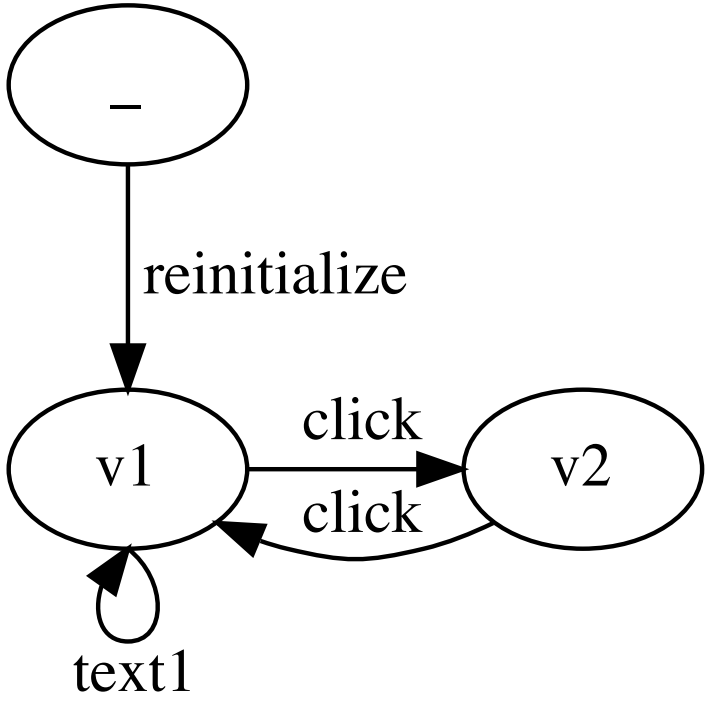




AndroFrame Example

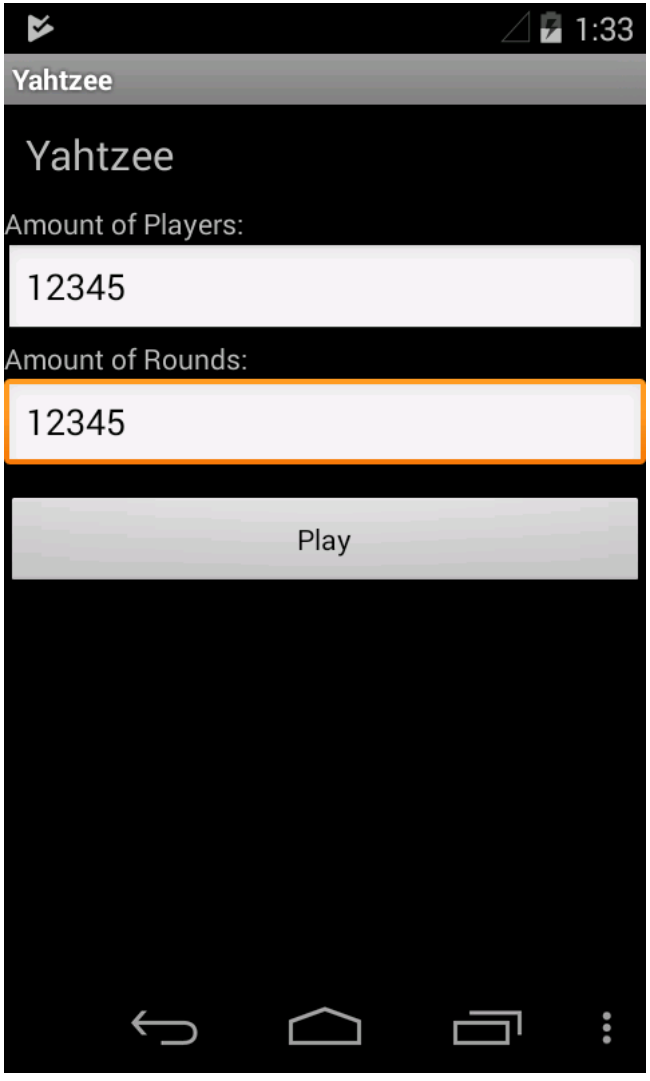
Action: text 200 270 12345 (text1)

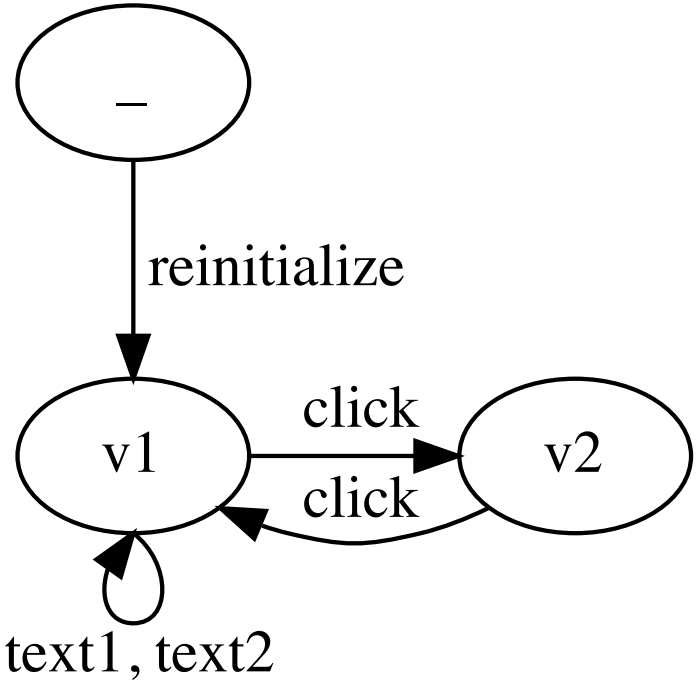




AndroFrame Example

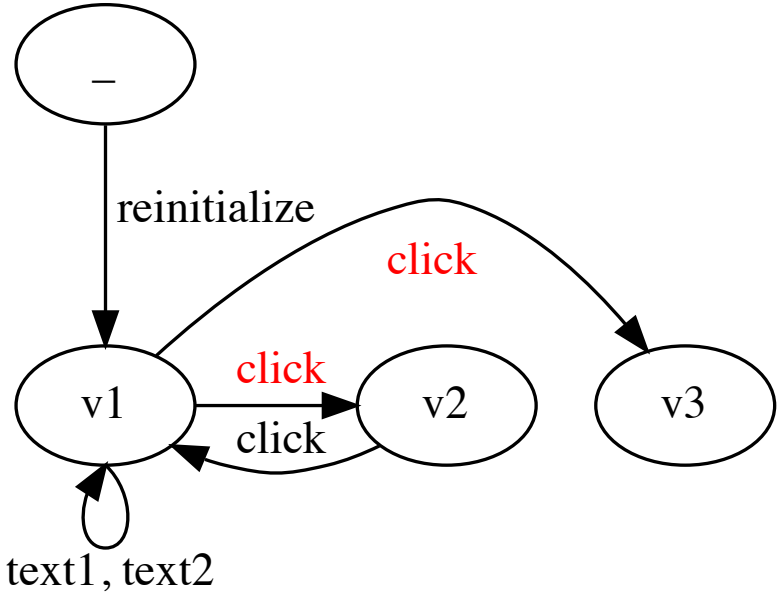
Action: text 200 330 12345 (text2)





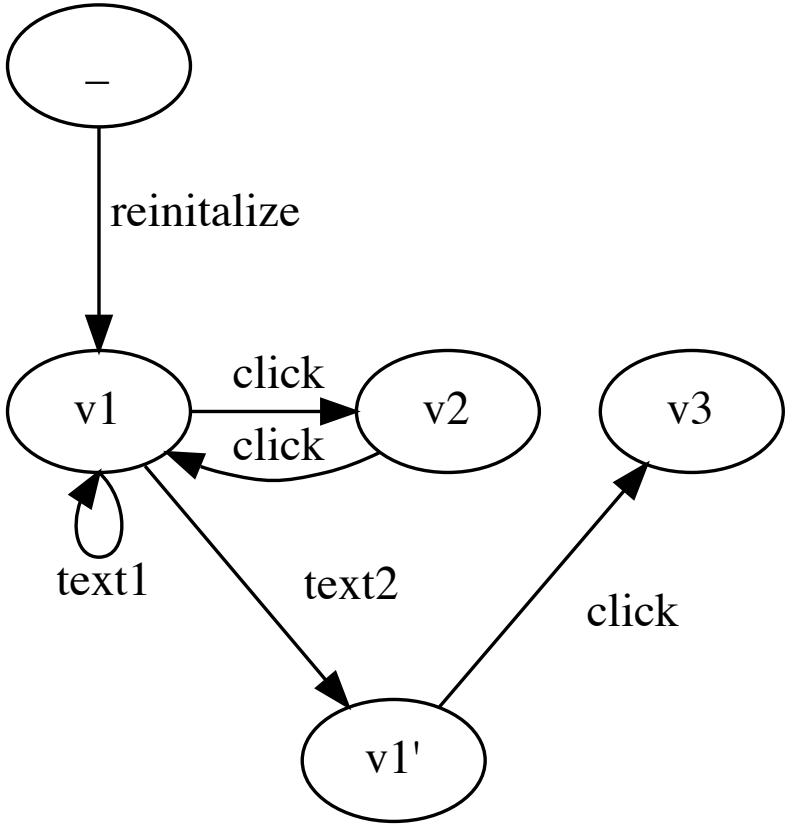
AndroFrame Example

Action: click 200 390 (click play)



AndroFrame Example

Action: click 200 390 (click play)



Reinforcement Learning

State and Action Abstractions

$$\beta(v) = \begin{cases} 1, & |\lambda(v)| \leq 1 \\ 2, & |\lambda(v)| \leq 3 \\ 3, & |\lambda(v)| \leq 8 \\ 4, & |\lambda(v)| \leq 15 \\ 5, & |\lambda(v)| > 15 \end{cases} \qquad \alpha(z) = \begin{cases} 1, & z \text{ is a } \textit{menu} \\ 2, & z \text{ is a } \textit{back} \\ 3, & z \text{ is a } \textit{click} \\ 4, & z \text{ is a } \textit{longclick} \\ 5, & z \text{ is a } \textit{text} \\ 6, & z \text{ is a } \textit{swipe} \\ 7, & z \text{ is a } \textit{contextual} \end{cases} \tag{2}$$

Reinforcement Learning

Q-Matrices as Expectation Distributions for Multiple Objectives

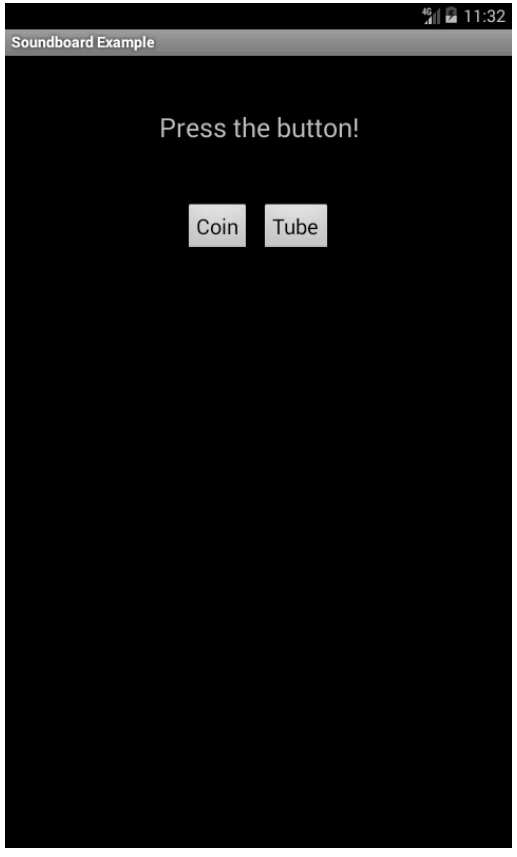
$$\vec{Q}_a = \begin{bmatrix} 0.11 & 0.09 & 0.40 & 0 & 0.10 & 0.30 & 0 \\ 0.13 & 0.44 & 0.26 & 0 & 0.12 & 0.05 & 0 \\ 0.06 & 0.66 & 0.16 & 0 & 0.13 & 0 & 0 \\ 0.17 & 0.25 & 0.40 & 0 & 0.09 & 0.09 & 0 \\ 0.06 & 0.28 & 0.52 & 0 & 0.09 & 0.05 & 0 \end{bmatrix} \quad (3)$$

$$\vec{Q}_c = \begin{bmatrix} 0.04 & 0.18 & 0.33 & 0 & 0.12 & 0.33 & 0 \\ 0.19 & 0.18 & 0.12 & 0 & 0.44 & 0.07 & 0 \\ 0.13 & 0.43 & 0.15 & 0 & 0.07 & 0.23 & 0 \\ 0.17 & 0.18 & 0.48 & 0 & 0.18 & 0 & 0 \\ 0.33 & 0.26 & 0.13 & 0 & 0.23 & 0.04 & 0 \end{bmatrix} \quad (4)$$

Test Case Mutation (TCM)

Case Study 1: Loop-Stressing

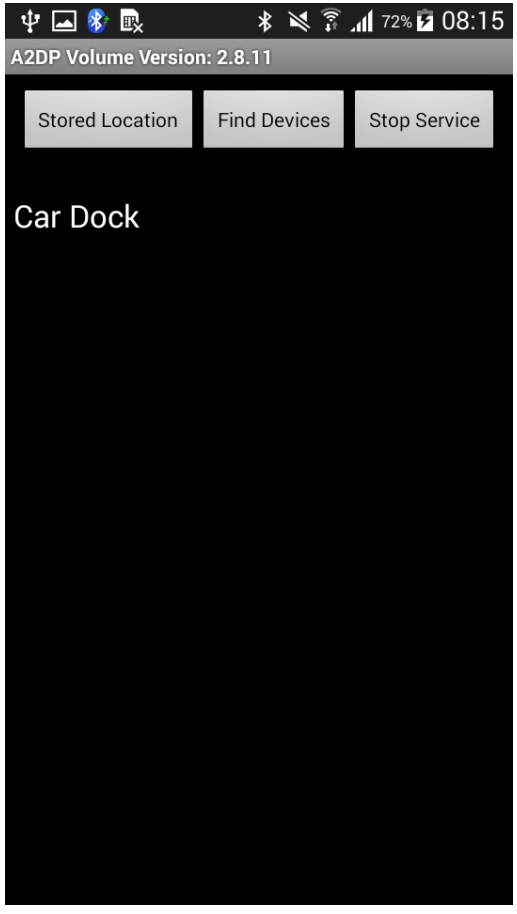
Pressing **Coin** button multiple times results in crash.

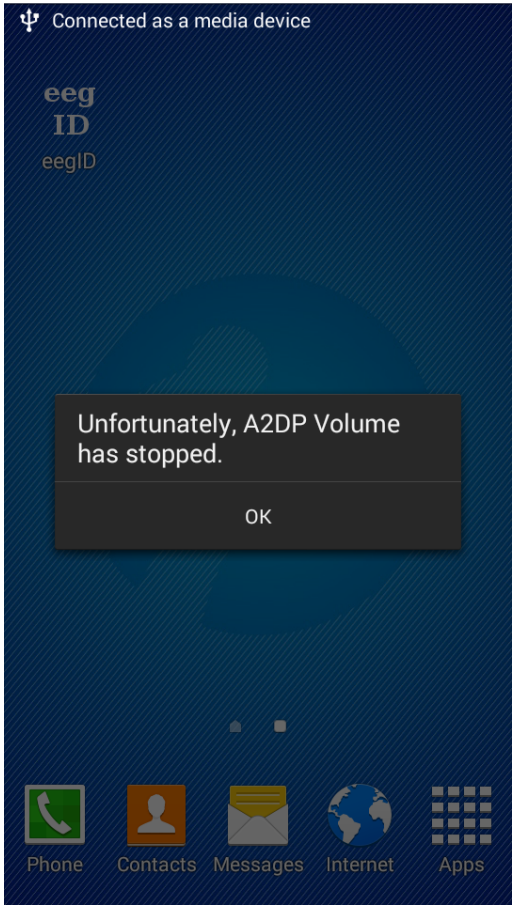


Test Case Mutation (TCM)

Case Study 2: Contextual-State Toggling

Turning **bluetooth** on and then clicking **Find Devices** reeulsts in crash.

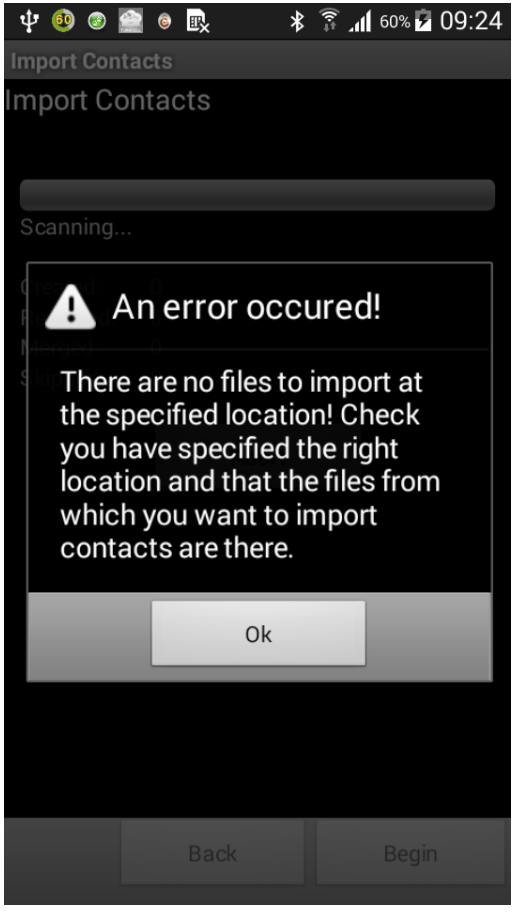


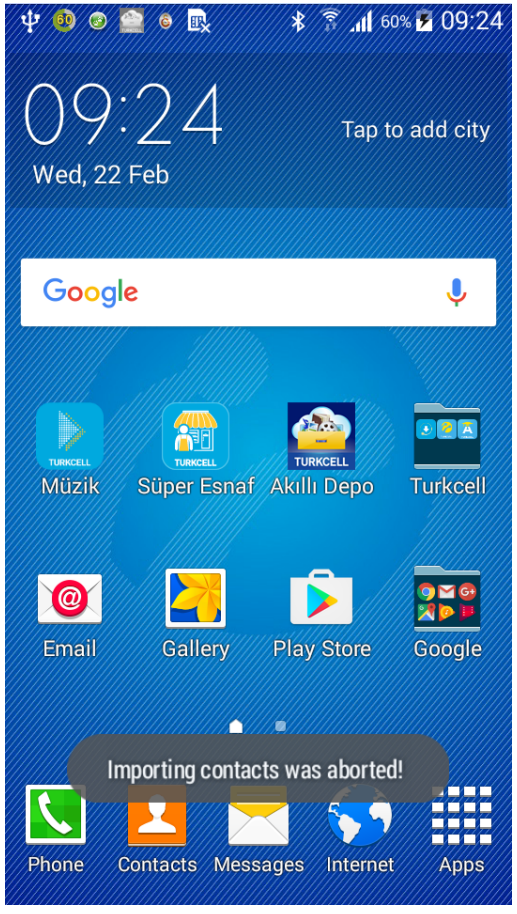


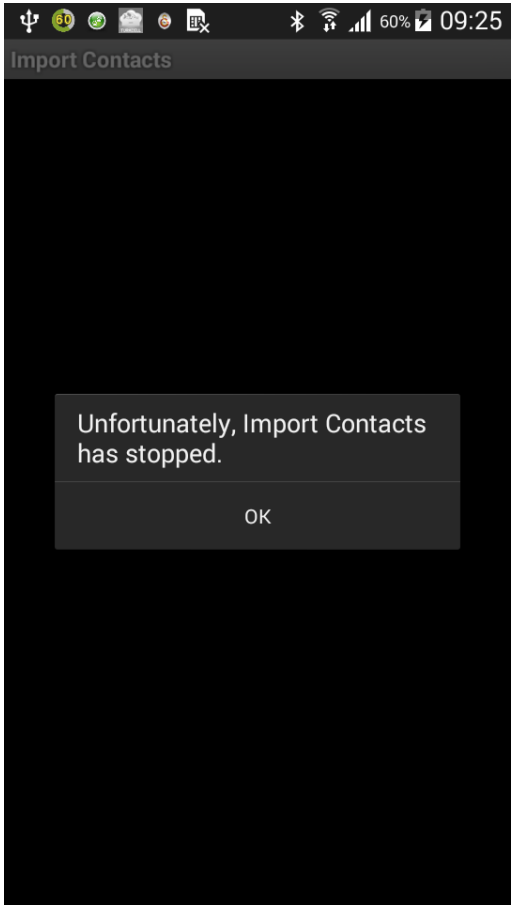
Test Case Mutation (TCM)

Case Study 3: Pause-Resume

Pausing and then resuming results in a crash.



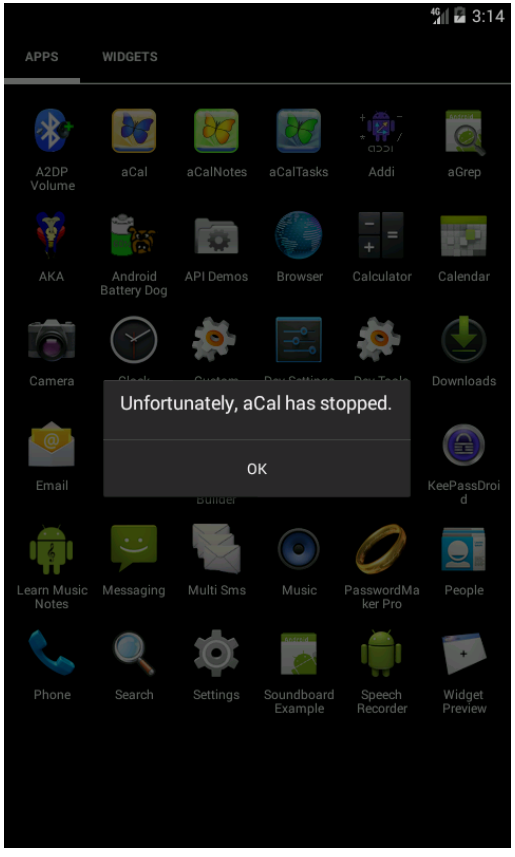




Test Case Mutation (TCM)

Case Study 4: Change Text

Changing text results in a crash.



Test Case Mutation (TCM) Example

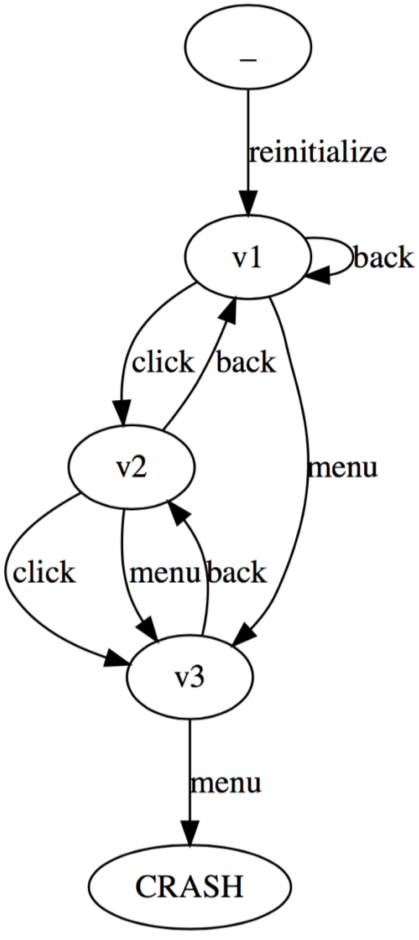
	Test Case A			
1	_	v1	reinit	10
2	v1	v2	click	1
3	v2	v1	back	1
4	v1	v2	click	1
5	v2	v1	back	1

	Test Case B			
1	_	v1	reinit	8
2	v1	v3	menu	2
3	v3	CRASH	menu	1

	Test Case C			
1	_	v1	reinit	9
2	v1	v1	back	0
3	v1	v2	click	1
4	v2	v3	click	2
5	v3	CRASH	menu	2

	Test Case D			
1	_	v1	reinit	15
2	v1	v1	back	0
3	v1	v2	click	2
4	v2	v1	back	1
5	v1	v3	menu	3

(a) Test Cases



(b) AUT Model

	Minimized			
1	_	v1	reinit	15
2	v1	v1	back	0
3	v1	v2	click	2
4	v2	v1	back	1

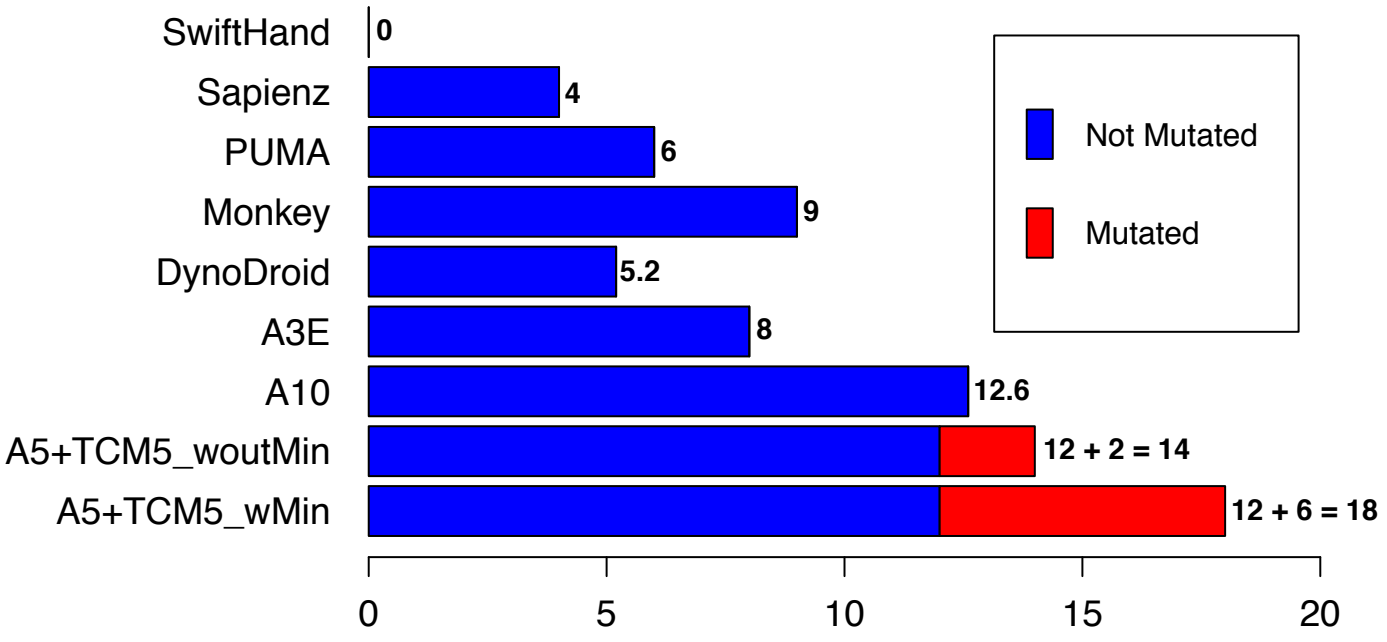
	Mutated 1			
1	_	v1	reinit	15
2	**v1**	**v1**	**back**	**1**
3	**v1**	**v1**	**back**	**1**
4	**v1**	**v1**	**back**	**1**
5	**v1**	**v1**	**back**	**1**
6	**v1**	**v1**	**back**	**1**
7	**v1**	**v1**	**back**	**1**
8	**v1**	**v1**	**back**	**1**
9	**v1**	**v1**	**back**	**1**
10	**v1**	**v1**	**back**	**1**
11	v1	v1	back	0
12	v1	v2	click	2
13	v2	v1	back	1

	Mutated 2			
1	_	v1	reinit	15
2	**v1**	_	**doze off**	**2**
3	_	**v1**	**doze on**	**2**
4	v1	v1	back	0
5	**v1**	_	**doze off**	**2**
6	_	**v1**	**doze on**	**2**
7	v1	v2	click	2
8	**v1**	_	**doze off**	**2**
9	_	**v1**	**doze on**	**2**
10	v2	v1	back	1
11	**v1**	_	**doze off**	**2**
12	_	**v1**	**doze on**	**2**

(c) Minimization and Mutation

Fig. 2: Motivating Example (mutations are denoted as bold)

Some Results



Experimental Set

100 Applications from known F-Droid benchmarks.

Future Work

App-Agnostic Oracles

Automated oracles that **find non-crashing problems** in Android.

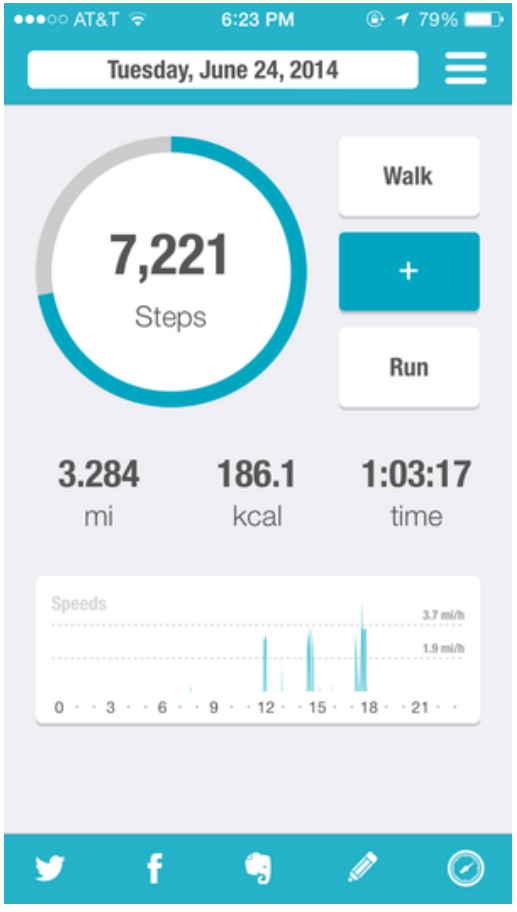
- Pausing-Resuming not returning the same state.
- Broken layout after double rotation.
- Broken back button not going to previous state.

Feedback-Directed Monkey (FDMonkey) Testing

- Monkey can't go **deep into the application**.
- **Guide Monkey parameters** using the coverage, crash and other info.

Specification-Based Testing

It is not interesting to test some applications for crash.



Specification-Based Testing

Test for the output correctness via specifications

Thank You. Any Questions?