Global Journal of Computer Science and Technology
Volume 12 Issue 8 Version 1.0 April 2012
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals Inc. (USA)
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

Rubik's Cube Application for Android Mobile Phones with Addressing Android Operating System Problems

By Amar Changdeo Gadekar, Amit Uttamrao Jagtap, Amit Jugmandar Kunchalas & Arpita Jadhav

Pune University Dehu Phata, Kelgaon- Alandi(D), Maharashtra India

Abstract - Android Smart phones enable a new, rich user experience in pervasive computing, but their hardware is still very limited in terms of computation, memory, and energy reserves, thus limiting potential applications. In this paper, we are dealing with battery drain problems and stack overflow problems of android mobile phones. For that we are going to develop Rubik's cube game application. We will test this app on several mobile phones of same configuration and of different manufacturer i.e. Samsung, Motorola, HTC, Sony Ericson, Micromax. In this research we will measure the performance of our app and will conclude about some parameters of Android Operating System i.e. heap utilization, power consumption, smooth and faster execution which will give meaningful information to reveal or solve or address Android Operating System problems in a detailed manner.

Keywords : *Rubik's Cube, Android Operating System, Problems.*

GJCST Classification: *D.m*

Strictly as per the compliance and regulations of:

Rubik's Cube Application for Android Mobile Phones with Addressing Android Operating System Problems

Amar Changdeo Gadekar [α], Amit Uttamrao Jagtap [σ], Amit Jugmandar Kunchalas [ρ] & Arpita Jadhav [ω]

Abstract - Android Smart phones enable a new, rich user experience in pervasive computing, but their hardware is still very limited in terms of computation, memory, and energy reserves, thus limiting potential applications. In this paper, we are dealing with battery drain problems and stack overflow problems of android mobile phones. For that we are going to develop Rubik's cube game application. We will test this app on several mobile phones of same configuration and of different manufacturer i.e. Samsung, Motorola, HTC, Sony Ericson, Micromax. In this research we will measure the performance of our app and will conclude about some parameters of Android Operating System i.e. heap utilization, power consumption, smooth and faster execution which will give meaningful information to reveal or solve or address Android Operating System problems in a detailed manner.

Keywords : Rubik's Cube, Android Operating System, Problems.

I. Introduction

Smart phones with Internet access, GPS, sensors, and various applications are recently seeing explosive adoption. The Apple iPhone, Blackberry smart phones, and the Google Android phone are a few prominent examples. In a slightly more advanced capability bracket also lie mobile Internet devices (MIDs) such as the Nokia N8 and Moblin-based devices that provide a richer untethered Internet experience. With popularity, such devices also see new applications by a broader set of developers, beyond the mobile staples of personal information management and music playback. Now mobile users play games; capture, edit, annotate and upload video; handle their finances; manage their personal health and "wellness" (e.g., iPhone Heart Monitor and Diamedic). However, with greater application power comes greater responsibility for the mobile execution platform: it is now important to track memory leaks and runaway processes sucking up power, to avoid or detect malicious intrusions and private data disclosure, and to manage applications with expensive tastes for high volume data or advanced computational capabilities such as floating-point or vector operations.

a) Mobile Performance Parameters

Android Phones has limited processing capabilities. To maximize it following parameters are considered while developing an Android application.

a) Processor
b) RAM
c) Heap Utilization
d) Power Consumption
e) Type of Touch Screen

b) Rubik's Cube

The Rubik's Cube is a 3-D mechanical puzzle invented by a Hungarian sculptor Erno Rubik. In a classic Rubik's Cube, each of the six faces is covered by nine stickers, among six solid colors (traditionally white, red, blue, orange, green, and yellow). For the puzzle to be solved, each face must be a solid color.

Our aim is to develop a Rubik's cube game application for touch screen mobile phones on the Android operating system platform using the Android OpenGL graphics library. In our project, we can control the vector cube in a 3D virtual environment (panning/tilting) to solve the timed puzzle. This is implemented in three modules:

I. Memory representation of the cube-It deals with a standard 3×3×3 cube, exploring the group of available moves and its action on sets of the cube's constituent parts ultimately showing the group of moves to be better-understood.
II. Visualization- In the actual implementation, the observer is moved in a spherical shape around the origin. To the user, the cube appears to be rotating because the background always remains same.
III. Designing of the touch UI- action-reaction of the touch user interface is developed. Concept of Group Theory is used as the fundamental platform for implementing these. Rubik's Cube is an age old and popular game. Through make this

Author α : Department of Computer Engineering, Maharashtra Academy of Engineering, Pune University Dehu Phata, Kelgaon-Alandi(D), Maharashtra-412105, India.
E-mail: amargadekar36@gmail.com
Author σ : Department of Computer Engineering, Maharashtra Academy of Engineering, Pune University Dehu Phata, Kelgaon-Alandi(D), Maharashtra-412105, India.
E-mail: amit.scorpian@gmail.com
Author ρ : Department of Computer Engineering, Maharashtra Academy of Engineering, Pune University Dehu Phata, Kelgaon-Alandi(D), Maharashtra-412105, India.
E-mail: amitkunchalas1090@gmail.com

timeless game available on the mobile platform, for the user who is always on the go.

II. Proposed Research Work

Gaming application utilizes maximum processor power capabilities, and maximum utilization of heap. Sometimes app requires memory beyond the heap size, at that time there is possibility of application crashing i.e. Force Close message in Android Phones. So basically we are tend to solve these problems where application which has to be developed should be able to run without crashing on minimal configuration smart phone. We know that there are app store from where we can buy, purchase or can download free apps i.e. Android Market. Free application's license fee is paid by advertisement sponsors and user is able to play the game with advertisements appearing on the screen. Billion users mostly download the free applications. They purchased it until and unless it is required. So this advertisement requires internet connection while playing the games and also uses heap which affects the performance of game app.

So we are dealing with following android phone problems:

a) Android Battery Drain Problem
b) Stack Overflow
c) Advertisement overhead for free apps

a) Battery Drain Problem

An Android application will run on a mobile device with limited computing power and storage, and constrained battery life. Because of this, it should be *efficient.* Battery life is one reason you might want to optimize your app even if it already seems to run "fast enough". Battery life is important to users, and Android's battery usage breakdown means users will know if your app is responsible draining their battery. There are two basic rules for writing efficient code considering battery: Don't do work that you don't need to do. Don't allocate memory if you can avoid it.

Smartphone's always tries to access for the internet for the services like GPS, phone log updating, messenger services, software upgradation, wifi-hotspot detection, GPRS-EDGE connectivity, 3G network searching etc. While playing the game, most of the app requires internet connections though they are not necessary.

b) Stack Overflow

In a Smartphone several processes are running in background. It means it supports multitasking. So in multitasking operation stack or heap plays important role for the main app. If heap size gets fully occupied then there are chances of app crashing. So to avoid this we are developing algorithm in which the app will automatically kill the processes which are idle and not required to stay in the processes queue. Mostly Priorities are assigned to processes on which we can sort out which processes has to be killed.

Restricted access to network:

We are solving this problem by building an advertisement container for free app.

c) Advertisement Overhead

Free app's license fees are paid by sponsors. User is able to play the game with little bit interference of advertisements. This advertisements are get downloaded when user runs the app. For this app always requires internet connection. And which consumes more power. So we will develop advertisement container in which we will install the ads, this container will be updated once in a day. So that user doesn't require internet access when he wishes to play the app. So this internet overhead can be reduced by this advertisement container.

As we are building advertisement container, in this container when we are downloading or updating the stack of ads, these ads get compressed and we can put them into low resolution. So when user plays the game these add will access minimum area of heap without affecting the performance of game.

III. Mathematical Model

Data:

$$Cube = \{ANDR, MAT, PHY_{MEM}, HEAP, TI, Pow, APKO, E|\emptyset Cube\}$$

Where,

ANDR=Android OS,
MAT= Six 2D Matrices for each face of cube,
PHYMEM= Physical Memory,
HEAP= Shared memory,
TI= Touch Interface,
Pow= Power consumed per unit time,
APKO= Other applications overhead on the system,
E=Exceptions,
ØCube= Rules.

Success : Successful execution of Rubik's Cube application with minimum power consumption per unit time.

Failure : Memory or power not available.

Input State Validation:

Input State Validation
{PHYMEMAVAIL > PHYMEMREQ}

Objective Functions:

1. To match the pattern:

The initial input to the pattern matching function:
IP= {Six 2D matrices, Goal state patterns}
Output states after execution of pattern matching function:

Success: If input matrices matches with the pattern.
Failure: If input matrices do not match with the pattern.

2. To consume lesser power:
The initial power requirement can be stated as below:

$$\mathrm{Pow_{AVAIL}} > \mathrm{Pow_{REQ}} + \mathrm{Pow_{OVRHD}} > 0$$

In general, the power consumed by the app can be calculated as:

$$\mathrm{POW_{REQ}} = \sum \mathrm{POW_{GAME}}, \mathrm{POW_{ADDS}}, \mathrm{POW_{OVRHD}}, \mathrm{POW_{NW}}, \mathrm{POW_{TI}}$$

Function 'battery consumption':

$$\mathrm{POW_{REQ}} = \{\mathrm{POW_{GAME}}, \mathrm{POW_{ADDS}}, \mathrm{POW_{OVRHD}}, \mathrm{POW_{NW}}, \mathrm{POW_{TI}}\}$$

Where,

POWREQ= Total power consumed by app
POWGAME= Power required by game app
POWOVRHD= Power required by other apps
POWNW= Power required by network
POWTI= Power required by touch interface
Power consumption after function execution:

$$\mathrm{POW_{REQ}} = \sum \mathrm{POW_{GAME}}, \mathrm{POW_{TI}} - \sum \mathrm{POW_{ADDS}}, \mathrm{POW_{NW}}, \mathrm{POW_{OVRHD}}$$

3. Avoid stack overflow:

$$\mathrm{FREE_{STACK}} = \{\mathrm{MEM_{TOTAL}}, \mathrm{MEM_{APP}}, \mathrm{MEM_{OTHRAPP}}, \mathrm{APP_{ID}}, \mathrm{APP_{PRIO}}\}$$

Where,

MEMTOTAL= Total memory of system
MEMAPP= Memory being consumed by game app
MEMOTHRAPP= Memory being consumed by other apps
APPID= Application or process IDs of current apps
APPPRIO= Application or process priorities of current apps

Function:
If,

$$\sum \mathrm{MEM_{APP}}, \mathrm{MEM_{OTHRAPP}} > \mathrm{MEM_{TOTAL}}$$

Then,

KILLAPP= MINPRIO(APPPRIO) && APPIDLE

Table.1: Android Smart Phone Comparison

	RAM (MB)	ROM (MB)	Processor (MHz)	Touch Screen Type	Battery (Talk time in hrs)
Samsung Galaxy Fit S5670	280	180	600	Capacitive	12 hrs 2G 9 hrs 3G
DELL XCD35	256	512	600	Capacitive	9 hrs 2G 8 hrs 3G
Samsung Galaxy POP	280	180	600	Capacitive	10 hrs 2G 9 hrs 3G

Above table gives comparison of several android smart phones with respect to their hardware configuration.

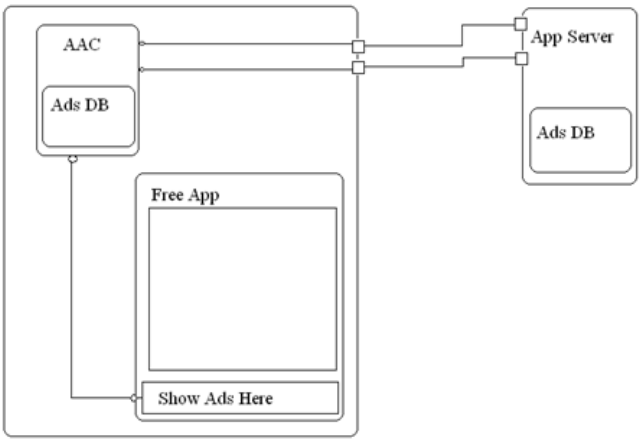

Figure.1: Application Advertisement Container (AAC) Framework

Application Advertisement Container (AAC) will connect to application server to fetch several new ads from advertise database once a week when user have an active internet connection.

Application (while running) need not to be connected to the internet in order to show ads. Ads are fetched from application advertisement container (AAC).

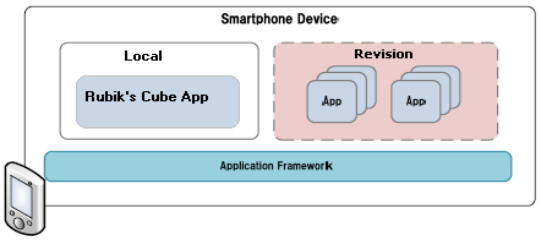

Figure.2 : handling heap & crashing problem framework

Local area contains active foreground application i.e. current running application.

Revision area contains background running applications. These applications are revised. It means to improve processor power we have to kill processes which are not required. These applications are put up in the revision area and by analysis it processes which are not required are killed. Active flag is marked for system and antivirus applications. Active flag tasks are not killed.

IV. Conclusion

The Rubik's Cube game is being brought to the virtual world of mobility using the popular operating system Android. Android is an operating system for mobile devices such as cellular phones, tablet computers etc. There are currently over 70,000 apps available for Android, which makes it the second most popular mobile development target.

Popular with a huge mass of people with varied age-groups having competitions held world wide for the person solves in the minimum time span.

In this project, we propose the modules for Rubik's cube application viz. Memory representation of the cube, visualization and designing of the touch UI. The memory implementation comprises of the representation of a 3D cube in 2D, whereas, in visualization the actual appearance of the cube and its functions are implemented and in the last module, the function of the touch interface is designed.

There are various options available for the user to change the settings according to naive and advanced users for adjusting the speed, vibration, sound options, brightness etc. The Goal for the user is to make each face of a solid color, initial state of the Rubik's cube will be in scrambled state. Born with Google integration, open source, constantly improved- courtesy the big brain at Google. More options available to manage your phone and Gmail account. An intuitive User interface packed with options and flexibility, smoother, more efficient aesthetically pleasing display technically superior fast and powerful. Thus our application of Rubik's cube on android for touch screen mobile phones keeps the adrenaline of user's pumping high. It works great in development of brain in small children helping to develop logic and analytical skills at a tender age. It's not just a game but a life time experience, so come, play and experience.

References Références Referencias

1. "Device Requirements" Android Source Code <git://android.git.kernel.org/platform/development/pdk/docs/guide/system_requirements.jd>
2. "Dalvik" Android Source Code <git://android.git.kernel.org/platform/development/pdk/docs/guide/dalvik.jd>
3. L. Xie, X. Zhang, A. Chaugule, T. Jaeger, and S. Zhu, "Designing System-Level Defenses against Cell phone Malware," *Reliable Distributed Systems, 2009.* SRDS '09. 28th IEEE
4. International Symposium on, 2009, pp.83 -90.
5. Kunkle, D.; Cooperman, G. "Twenty-six Moves Suffice for Rubik's Cube." Proceedings of the International Symposium on Symbolic and Algebraic Computation (ISSAC '07), ACM Press.
6. McPhail, B. (2003). The Complexity of Puzzles: NP-Completeness Results for Rubrics cube and Minesweeper. Thesis for BA, Division of Mathematics and Natural Sciences, Reed College.
7. A. Rudenko, P. Reiher, G. J. Popek, and G. H. Kuenning. Saving portable computer battery power through remote process execution. *MCCR.*
8. H. Kim et al., "Detecting Energy-Greedy Anomalies and Mobile Malware Variants," in International Conference on Mobile Systems, Applications, and Services (MobiSys), 2008.
9. A. P. Fuchs et al., "SCanDroid: Automated Security Certification of Android Applications," 2010.
10. Http://portal.acm.org/citation.cfm?id=801054&dl=GUIDE&coll=GUIDE&CFID=10178237&CFTOKEN=61248052
11. Http://www.arm.com/files/pdf/26379.pdf
12. Developer.android.com/guide/tutorials/hello-world.html
13. Group Theory and the Rubik's Cube. Texas State Honors Math Camp.
14. Adventures of Group Theory: Rubik's Cube, Merlin's Machine, and Other Mathematical Toy.