Research Journal of Applied Sciences, Engineering and Technology 5(18): 4438-4443, 2013
ISSN: 2040-7459; e-ISSN: 2040-7467

Submitted: July 27, 2012 Accepted: September 17, 2012 Published: May 05, 2013

Android Graphic System Acceleration Based on DirectFB

[1]Jiang Zhong-Qiu, [2]Zhang Min and [1]Liu Chang-Rong
[1]Huaian College of Information Technology, Huai'an Jiangsu, 213003, China
[2]National ASIC System Engineering Research Center, Southeast University, Nanjing 210096, China

Abstract: In this study, based on analyzing the hardware abstraction layer and native graphics libraries of Android graphics system, the drawback of Skia library which could only support software rendering is pointed out. And then the third-party open graphics library DirectFB which supports 2D hardware acceleration is introduced, the architecture and interface of DirectFB and Skia are analyzed and compared with each other in detail. After DirectFB being ported into Android system, a novel hardware acceleration layer with double-buffer technology is designed and implemented, which will make Skia and DirectFB coexist and complement with each other and ultimately implement the 2D hardware acceleration in Android system. A JNI interface is designed for Java programs. The optimization scheme is verified by the specialized test benchmarks df-dok, the experimental results indicated that the performance of Android graphics system in layer blending operations is accelerated by an average of 5.58x as well as 2.18x speedup on average in bitblit operations when processing complex graphics operations such as layer blending and bitblit etc.

Keywords: Adaptation layer, android graphics system, DirectFB, hardware acceleration, Skia

INTRODUCTION

With the rapid development of Mobile Internet, a variety of mobile devices, such as net-book, intelligent cell phone and tablets etc., emerge to our life and they are becoming indispensable partitions of our daily work and life. The hardware configuration is continual enhanced and updated, while user feeling is not only decided by the hardware configuration, but also seriously lays on the software system coupled with the hardware platforms. Up to Now, the state-of-art public intelligent OS (Operating System) is mainly composed of Windows mobile series of Microsoft Corp android of Google and iOS of Apple Corp etc.

Google's Android system is firstly released in Nov. 5th, 2007 (Android (Operating System), 2010) formally. It is mainly constructed by operation system, middle firmware, user interface and applications. It is the newest OS, but the most public one to engineers. Compared to other closed OS, it adopts the Apache license, its source code is open and free, which makes it to be the most popular one for many corporations home and abroad, such as Motorola, Samsung, Sony Ericsson, China mobile and Huawei etc., most of them have selected it as their based developing platforms. Android has become the hot-spot in industry and academia scopes.

The graphics system and UI (User Interface) are the key-partition of the Android system, the performance and even the success or failure of one mobile device mostly relies on them. As the core engine of 2D graphics sub-system, Skia only supports software rendering acceleration, which has made the graphic system to be bottle-neck of system performance, especially in sceneries involving numerous image drawing and complex UI displaying. There are few literatures in this scope, the major optimization approaches is to adopt the hardware accelerating port of Copybit module to improve the 2D graphic processing, but it could only issue a few simple operations and couldn't cover the whole 2D operations. From this viewpoint, the Android graphic system is paid emphasis on in this study to find other hardware scheme to accelerate 2D graphic rendering. It is difficult to modify the Skia directly to make it support hardware acceleration taking large risk. DirectFB with the characteristic of open source is introduced and ported to Android system, which will make it cooperation with the native Skia and eventually accelerating 2D graphics rendering by hardware.

The relation Research works on Android could be classified into three levels. The first one is Android applications. Ping-Xin (2009) implemented input method into the input frame of Android. Ughetti *et al*. (2008) designed a P2P mobile application program based on the agent mechanism and this communication model of JADE frame will make developers to design P2P software with high efficiency. The second one is focused on the Android frame, which lies in the cross-platform porting and optimization of Dalvik VM

Corresponding Author: Jiang Zhong-Qiu, Huaian College of Information Technology, Huai'an Jiangsu, 213003, China

(Virtual Machine), including design and optimization of the application frame and local frame. Tapia (2008) improved the performance of open core in Android for Audio codec and the related operations. Cheung (2009) adopted a novel Markov decision process to optimize software in Android. The third class is about the driver level and system porting. The drivers in low level are important partitions of Android, which includes Linux kernel, drivers and Android HAL (Hardware Abstract Level) etc. Wen-Chang Chung (Xuguang, 2009) discussed the Android system porting for PX270 platform of Marvell. There are some research works at this point. Yonggang *et al*. (2010) studied the application of 2D hardware graphics acceleration in the embedded multimedia system based on Linux, a self-adaption and seamless architecture of software and hardware graphics acceleration has been designed and a scheme for buffer submitting based on computing workloads has been introduced. Compared to the previous researches, the hardware adaption level of Android is focused on and optimized in ours, DirectFB is discussed in detail and 2D hardware acceleration units is utilized to improve the performance of graphics system as well as the whole system.

In this study, the drawback of Skia library which could only support software rendering is pointed out. And then the third-party open graphics library DirectFB which supports 2D hardware acceleration is introduced, the architecture and interface of DirectFB and Skia are analyzed and compared with each other in detail. After DirectFB being ported into Android system, a novel hardware acceleration layer with double-buffer technology is designed and implemented, which will make Skia and DirectFB coexist and complement with each other and ultimately implement the 2D hardware acceleration in Android system. A JNI interface is designed for Java programs. The optimization scheme is verified by the specialized test benchmarks df-dok, the experimental results indicated that the performance of Android graphics system in layer blending operations is accelerated by an average of 5.58x as well as 2.18x speedup on average in bit lit operations when processing complex graphics operations such as layer blending and bitblit etc.

ANDROID GRAPHIC SYSTEM

Android is a large and complex system, it includes Linux kernel, hardware adaption level, system frame level and Java application level. Linux kernel is the core and fundamental partition, the schedule algorithm is optimized for Android based on the standard Linux kernel and some other drivers are added into it. Drivers operate the hardware in low level directly, therefore designing efficient drivers for one fixed hardware platform is very important. The hardware adaption level is the interface package between kernel and application level. The next level is the Android system frame layer, which includes the application frame layer to provide API for Java applications. And some other open source libs such as Bionic, C lib and Skia etc., are added into it. Dalvik VM (Yi-Min and Rong, 2010) compiles the Java code to class files and transforms it into .dex files, the compiled Java program will ignore the platforms' differences and run cross-platform.

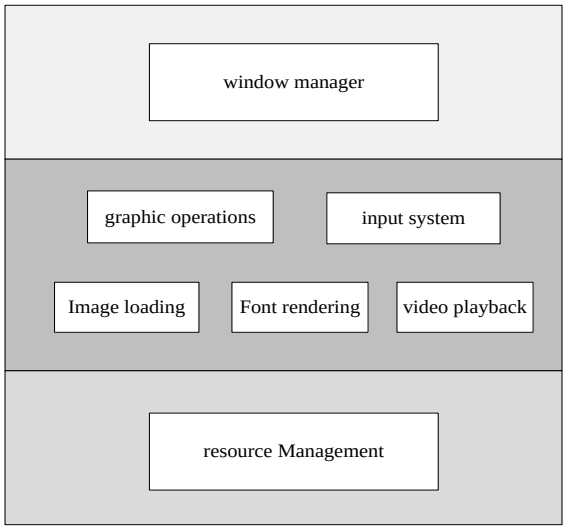

Fig. 1: Graphic sub-system of DirectFB

The graphics system adaption layer is analyzed and improved in our work, which includes graphics hardware adaption layer android graphics lib and application frame layer related to graphics. The graphics hardware adaption layer is constructed by PMEM, Framebuffer driver module and Gralloc graphics memory allocation module; they are the basement for system running. Graphics frame includes Libui lib, SurfaceFlinger lib, OpenGL ES lib and Skia lib. Skia is the native 2D graphics rendering engine lib of Android which is our focus point.

ANALYSIS OF DIRECTFB GRAPHICS LIB FRAME

As the graphics lib of the embedded system, DirectFB is a complete system, as showing in Fig. 1, it composed three layers. The first one is Window Manager (2011), which is utilized to control the layout and display of windows. The second one is the implementation of DirectFB, it could be classified into graphic operation, input system, image loading, Font rendering and video playback according to functions. The window manager is not isolated, but tightly coupled to the middle layer. The reason lies in that the operations such as window rendering, click on, image display and video playback in those windows are based on the corresponding sub-system in this layer. The third one is the resource management layer, which is basement for the middle layer. Its functions includes two sides, one

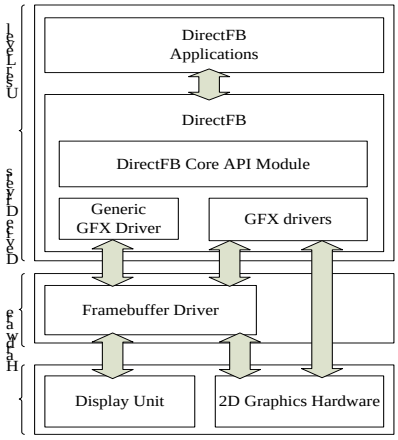

Fig. 2: Architecture of DirectFB

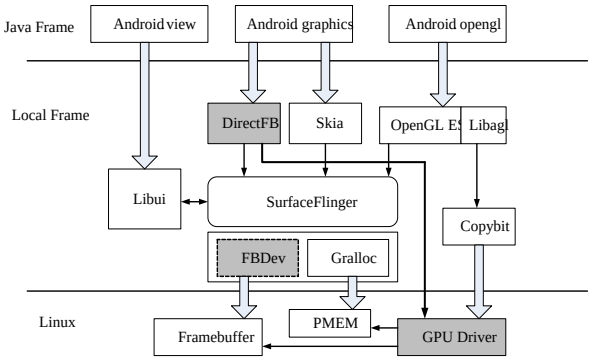

Fig. 3: Integration of Skia and DirectFB in android

side is in charge of memory allocation and release for the sub-systems in middle layer and the other side is for access to the lock and signal transmitting etc. They are the fundamental for the intra-system of sub-systems and inter-system to share resource of the whole platform.

This study focuses on performance of the graphics rendering and the design architecture of DirectFB graphics operations is paid emphasis on.

As showed in Fig. 2, the architecture of DirectFB (Renesas Solution Corp, 2005) composes user application layer, kernel driver layer and hardware layer. The user application layer could be divided into DirectFB application and lib layer. The DirectFB applications utilize the standard API (Application Interface) to finish the related operations such as rectangle filling/drawing and block transmitting etc. And those API will call down for the general GFX drivers which lies in user state, it will check whether the system supports hardware acceleration or not. If the hardware layer supports graphics acceleration modules with the correspondent kernel drivers, GFX will utilize the hardware acceleration module with kernel driver for graphics rendering, the native software API of DirectFB will be used for rendering on the other side. Whether hardware acceleration or software rendering, the final graphics data will be transmitted to Framebuffer for display.

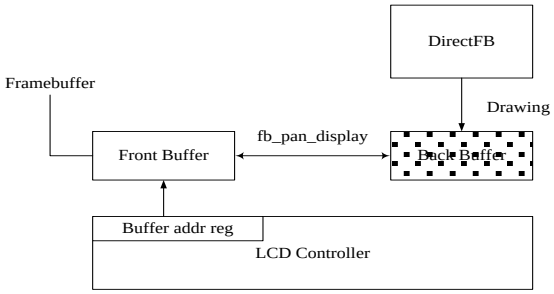

Fig. 4: Switch process of double buffer

Design and implementation of DirectFB: As showed in Fig. 3, the white rectangles construct the Skia graphics system. Skia has a hardware acceleration module and the correspondent driver, but all of the 2D graphics operations could only be issued by software due to lack of hardware calling interface in its engine. This mechanism will degrade the performance of Android graphics system significantly.

Those gray rectangles outline the DirectFB's layout in Android graphics system. DirectFB and Skia are in the same layer of Android and provide 2D graphics drawing API for the upper applications, while DirectFB can call for the 2D hardware acceleration module in GFX driver frame for hardware acceleration. The upper applications will own two schemes for graphics rendering, Skia will be selected in sceneries of low performance or without hardware acceleration and DirectFB should be chose for the sceneries needing high performance and with hardware acceleration interface.

Hardware acceleration for DirectFB: Frame buffer driver is utilized to control the hardware display unit which is the final step in graphics system. User applications will call for the operation interface of graphics lib firstly, select hardware or software rendering interface according to the hardware functions and system configurations and eventually transmit the graphics data into Framebuffer to display on screen. Either for the Gralloc module of Skia or the FBDev module of DirectFB, they are all based on standard Framebuffer driver. Thus optimization for the Framebuffer driver is very important in integrating DirectFB into the Android system.

The Framebuffer driver could be optimized from two sides. The first one is to allocate more than one cache buffer and control them to output graphics data to display unit alternatively. While the second one lies in that designing multi-layer Framebuffer and composed them by hardware to output graphics data. These two sides will be discussed in detail as following.

Adopting double buffer: As showed in Fig. 4, double buffer transmitting scheme executes in the exchange mechanism. The standard Framebuffer driver uses fb-

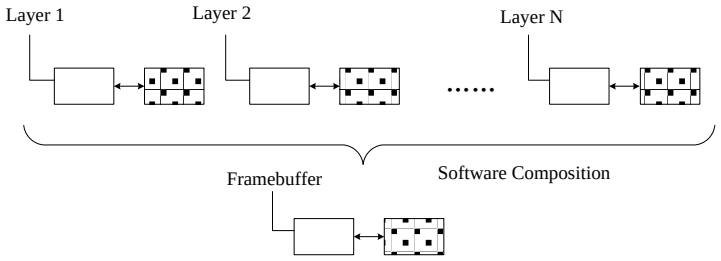

Fig. 5: Graphic layers combined by software

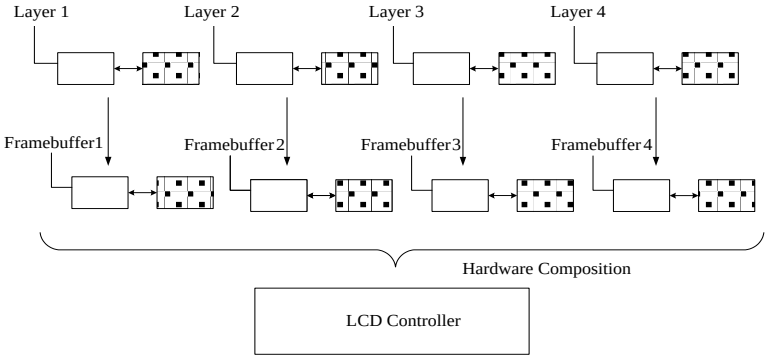

Fig. 6: Graphic layers combined by hardware

set-par( ) to complete displaying alternatively between those two buffers, which needs to probe and configure some parameters such as LCD horizontal and vertical timing, pixel format and alpha composition etc. And these operations will cause significant system overhead, even lead fuzzy and flicker in large-resolution display and high refresh rate sceneries. A dedicated interface called fb-pan-display( ) is designed for the Framebuffer driver, it is utilized to issue the exchange operation which will reduce the calling and configuration time significantly and improve the display quality.

Adopting multi-physical layer for display output: Display multi-layer needs the support of the hardware display module. The kernel allocates multi-layer Framebuffer, each layer provides consistent upper API, while the composition of them is completed by hardware.

In normal applications, the multi-layer composition is issued by software. As showed in Fig. 5, the kernel provides Framebuffer driver for the LCD controller. Although double buffers have been adopted, the composition of upper multi-layer is issued by software and then written to the Framebuffer. While the multi-layer composition will involve alpha hybrid computation (Pulli *et al*., 2005) and this will occupy a significant of CPU resource. If the composition could be satisfied by hardware, the graphics performance will be improved dramatically.

As showed in Fig. 6, the processor in our scheme support four hardware display layers, each physical layer adopts double buffer, the upper graphics layer will output data to the Frame buffer and those four Framebuffer will be composed by hardware to display eventually.

Adaptation layer of hardware acceleration module: The adaptation layer (Takanari *et al*., 2007) hides the hardware detail downside and provide unified interface upside. The related source code could be found in gfxdrivers/sep0611, which will be compiled and exist as dynamic lib in system.

The hardware acceleration adaptation layer of DirectFB needs to implement two major data structures of GraphicsDriverFuncs and GraphicsDeviceFuncs, which definitions locate in src/core/gfxcard.h. The adaptation layer (Xiao-Xue *et al*., 2010) includes six interface functions for GraphicsDriverFuncs, such as Probe( ), GetDriverInfo( ), InitDriver( ), InitDevice( ), CloseDriver( ), sep0611-setup-driver( ). Probe( ) is used to probe whether the system support special graphics acceleration module or not. GetDriverInfo( ) is utilized to get the driver information of hardware acceleration module. InitDriver( ) and CloseDriver( ) will initialize and close the driver. Sep0611-setup-driver( ) opens the device node of hardware acceleration module, get the display memory and IO memory and then remap them into user state for the upper applications to access display memory and hardware acceleration unit directly. Thereafter, the structure of GraphicsDeviceFuncs, which includes the control port of hardware acceleration module and the entire 2D graphics hardware

acceleration interface, will be instantiated (Enhua and Youquan, 2004).

After GraphicsDriverFuncs and Graphics DeviceFuncs being implemented, the hardware acceleration system exists as dynamic lib in system and it will be loaded in system initialization, GraphicsDriverFuncs will be registered to system. After this step, DirectFB could call the hardware interface of GraphicsDeviceFuncs for graphics rendering.

JNI upper interface design: The upper applications of Android system are developed by Java language and the application frame of Android provides Java API for developers. All of the local libs are implemented in C or C++ language and this need to utilize JNI to provide interface, which will make Java to call the local lib. Therefore, after porting DirectFB to Android and designing the hardware acceleration adaptation layer, we implement a JNI interface in DirectFB for Java applications.

EXPERIMENTS AND ANALYSIS

Experimental platform: We choose SEP6200 (ASIC Center, 2010) processor as our experimental platform. It adopts TSMC65LP CMOS process and centered in handheld mobile communication markets with high performance. It is composed of five major function parts, which are system clk control, interface and port, multi-media system, GPS navigation system and memory system. The platform configured with 4.3 Inch TFT LCD screen with 800*480 display resolution, 24 bit pixel data format and resistor touch panel. In software side, we select Uboot as the system boot tools android 2.3 based on Linux 2.6.32 kernel as OS. LCD Framebuffer driver has been designed and optimized. 2D hardware acceleration driver and touch panel driver have been written and added into the base system and system image files are compiled on 64 bit ubuntu server. Direct FB is downloaded from its official website of www.directfb.org with the version to be 1.4.13.

DirectFB test results and analysis: DirectFB provides hardware acceleration interface, while software rendering is the default way. After porting DirectFB to Android system, the df-dok test bench is used to verify the performance of graphics rendering in DirectFB and Skia engine and the results are compared with each other quantitatively. The test results are showed in Table 1 and Fig. 7.

It could be concluded from Table 1 and Fig. 7 that the performance of software rendering for Skia and DirectFB are almost at the same level, while compared to the performance of 2D hardware acceleration scheme in DirectFB, large differences will exist and the detail analysis as following:

- The performance of Skia is perfect when its basic drawing interface is called and utilized for simple operations, such as draw straight line and rectangle etc. While if these operations involve composition of multiple graphics layers, the rendering performance will degraded dramatically.
- The software rendering performance of Skia and DirectFB are basically the same. The performance improvement by DirectFB software rendering lies in 1-2% and almost has no performance gains, thus hardware acceleration rendering scheme by DirectFB is the only way to optimize the Android graphics system.
- After implementing hardware rendering of DirectFB, for simple drawing operations such as drawing straight line and rectangle etc., compared to Skia, it might decrease some percentage of rendering performance. The main reason lies in that it will lead to numerous system states switching when adopting the hardware acceleration scheme. This additional overhead will increase the system burden and it could neutralize or even overpass the performance gains due to the hardware acceleration module. When it is utilized to tackle with complex graphics operations which involve numerous computations, such as composition of multiple graphics layers and Bitmap blitting, it could bring about significant performance gains for graphics system. Compared to Skia lib, the performance enhancement for the composition of multiple graphics layer is 558% on average and Bitmap

Table 1: Test results of Skia and DirectFB

Bench no	Bench name	Statics unit	Skia software rendering	DirectFB software rendering	DirectFB hardware rendering
1	Draw text	KChars/sec	32.965	33.027	19.700
2	Draw text2	KChars/sec	17.025	17.153	19.712
3	Draw rect	MPixel/sec	37.012	37.216	71.234
4	Draw rect2	MPixel/sec	3.310	3.353	61.826
5	Draw polygon	MPixel/sec	27.247	27.430	8.947
6	Draw polygon2	MPixel/sec	3.118	3.225	8.745
7	Draw lines	KLines/sec	18.160	18.163	6.976
8	Draw lines2	KLines/sec	4.950	5.095	6.851
9	Draw image	MPixel/sec	17.608	17.873	19.100
10	Draw image2	MPixel/sec	22.557	22.760	24.617
11	Draw image3	MPixel/sec	11.508	11.619	19.376
12	Draw image4	MPixel/sec	2.043	2.053	18.137

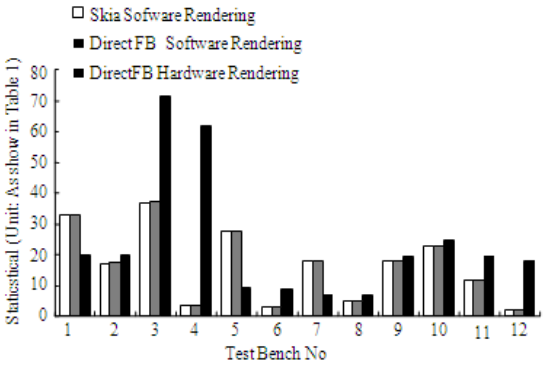

Fig. 7: Test results comparison between Skia and DirectFB

blitting could obtain 218% performance gains on average.

In summary, it needs to operate multiple graphics layers, which involves graphics layer composition and Bitmap blitting frequently in actual applications. These operations are complex and compute-intensive and could be accelerated by hardware rendering of DirectFB for performance improvement. There is some performance loss in simple graphics operations, but this kind of operations will not be utilized frequently. Thus our hardware acceleration scheme by DirectFB will bring about significant application performance gains.

CONCLUSION

The Android graphics system is studied in detail in this study. From the adaptation layer, after the base modules of Android graphics system such as PMEM etc., being analyzed, the architecture and interface of Skia and DirectFB are analyzed deeply and compared to each other in detail. DirectFB is ported to the Android system and made to be coexisted and complemented with the native Skia lib. Double buffer technique is adopted into graphics system. An adaptation layer for hardware acceleration module is designed and the hardware acceleration for 2D graphics rendering is implemented eventually. A local JNI interface is designed for Java applications in DirectFB. The acceleration scheme is verified by df-dok test bench. The testing results indicated that, after adopting our hardware acceleration scheme by DirectFB, except for a few simple graphics operations, whether vertical compared to DirectFB software scheme, or longitudinal compared to Skia software scheme, our scheme could improve the performance of Android graphics system significantly as well as the overall system performance.

ACKNOWLEDGMENT

This study is supported by Innovation Fund project in Science and technology enterprises of Jiangsu Province (No. BC2009207).

REFERENCES

Android (Operating System), 2010. Retrieved from: http:// en. wikipedia. org/wiki/ Android_ (operating_ system).

ASIC Center, 2010. SEUIC SEP6200 Development Document. Southeast University, Nanjing, pp: 5-22.

Cheung, L.T., 2009. Markov Decision Process (MDP) framework for optimizing software on mobile phones. International Conference on Compilers, Architecture and Synthesis for Embedded Systems of the 7th ACM International Conference, ACM, New York, pp: 11-20.

Enhua, W. and L. Youquan, 2004. General purpose computation on GPU. J. Comp. Aid. Design Comp. Graph., 16(5): 601-612.

Ping-Xin, L., 2009. Android plug-in input method programs design. J. Comp. Knowl. Technol., 5(35): 9979-9981.

Pulli, K., T. Aarnio, K. Roimela and J. Vaarala, 2005. Designing graphics programming interfaces for mobile devices. J. IEEE Comp. Graph. Appl., 25(6): 66-75.

Renesas Solution Corp, 2005. Writing Custom GFX Drivers for Direct FB.

Takanari, H., M. Hisao and O.K. Denis, 2007. How to write Direct FB Gfxdrivers for your Embedded Platform. Technology Consulting Company IGEL Corp.

Tapia, J., 2008. Introduction to the opencore audio components used in the android platform. New Trends in Audio for Mobile and Handheld Devices of the 34th AES International Conference, pp: 48-71.

Ughetti, M., T. Trucco and D. Gotta, 2008. Development of agent-based, peer-to-peer mobile applications on ANDROID with JADE. Proceedings of UBICOMM, Valencia, pp: 287-294.

Window Manager, 2011. Retrieved from: http:// en. wikipedia. org/wiki/Window_manager.

Xiao-Xue, Y., W. Li-Hu, Y. Jia-Ning and N. Li-ping, 2010. Direct FB applied in embedded remote desk control system. J. Comp. Eng. Design, 31(9): 2127-2130.

Xuguang, H., 2009. An Introduction to Android [DB/OL].. Inha Univeristy, Dababase Lab.

Yi-Min, Z. and CH. Rong, 2010. Analysis about process in dalvik virtual machine. J. Comp. Technol. Dev., 20(2): 83-86.

Yonggang, J., Q. Zhengwei, P. Juanchun and Z. Quan, 2010. Performance test and analysis for 2d graphic accelerator on pxa300 platform. J. Comp. Appl. Software, 27(5): 86-88.