# Demo: Mobile Gaming on Personal Computers with Direct Android Emulation

Qifan Yang[1,2], Xinlei Yang[1], Zhenhua Li[1], Yunhao Liu[1,3],
Rui Zhou[2], Guoyang Du[2], Ziwen Wu[2], Tianyin Xu[4], Ennan Zhai[5]

[1]Tsinghua University [2]Tencent Co. Ltd. [3]Michigan State University [4]UIUC [5]Alibaba Group

## ABSTRACT

Playing Android games with Windows x86 PCs is now popular, and the common solution is to use mobile emulators built with the AOVB (Android-x86 On VirtualBox) architecture. Nevertheless, running heavy 3D Android games on AOVB incurs considerable overhead of full virtualization, thus often leading to unsatisfactory smoothness. To tackle this issue, we present DAOW, a commercial game-oriented Android emulator implementing the idea of *direct Android emulation*, which eliminates the overhead of full virtualization by providing foreign Android binaries with direct access to the domestic PC hardware through Windows kernel interfaces. In this demo, we will demonstrate that DAOW essentially outperforms traditional AOVB-based emulators in terms of running smoothness, game startup time, and memory usage.

## 1 INTRODUCTION

Computer games, as one killer application of PCs and mobile devices, own a huge market of billion dollars [5]. The rapid evolution of computer games contributes to numerous technical innovations regarding both hardware (larger memories, faster CPUs, and graphics cards) and software (e.g., multimedia support and OS kernel improvements) [1]. In recent years, as mobile gaming is becoming the largest segment of the game market [5], many game vendors are inclined to implementing mobile games over their PC or console versions. While since porting mobile-based implementation onto PC platforms with different OSes and architectures requires immense efforts, only a few mobile games have corresponding PC versions. Despite tool support (*e.g.,* Unity and Unreal), such porting is still far from trivial as the existing tools provide neither correctness guarantee nor usability control.

With the boom in mobile-first gaming, there exist pressing demands for supporting mobile games on PC platforms, owing to the better QoE (*e.g.,* visual experience and operating experience) that PC-based gaming can provide. The *de facto* solution for playing mobile games on PCs usually depends on *mobile emulators*, such as Bluestacks, Genymotion, KoPlayer, Nox, and MEmu. All these game-oriented emulators employ a full virtualization architecture called AOVB (Android-x86 On VirtualBox), namely running Android-x86 on top of a VirtualBox VM. Android-x86 is an x86 porting of the Android OS, and VirtualBox bridges Android-x86 (the guest OS) to the host OS (*e.g.,* Windows). The AOVB architecture obtains high popularity as it is free, open-source, and fully transparent to unmodified mobile game binaries.

While AOVB-based emulators can run most mobile games, they are only capable of providing desired gaming experiences for 2D games as well as less interactive 3D games. As to *heavy* 3D games like Vainglory and PUBG Mobile, however, AOVB-based emulators yield substantially degraded gaming experiences (measured by *smoothness*, which is detailed in our full paper [2]). A *heavy* Android game is empirically considered to invoke 2000+ rendering instructions on average to display a graphic frame. Note that even millisecond-level stagnation can detract the overall experience in gaming, which differs from many other applications.

To demystify the above issue, we built and maintained an AOVB-based emulator (referred to as AOVB-EMU), which possesses more than 30 million users running over 40,000 Android game apps. Based on our measurement of its user experiences, the performance bottleneck stems from the considerable overhead of full virtualization. Aiming at supporting heavy mobile games, a series of para-virtualization and hardware-assisted optimizations are applied to AOVB-EMU, including GPU acceleration for graphic processing, VirtIO [3] for increasing the bandwidth of rendering pipelines, and Intel VT [4]. Such optimizations indeed substantially increase the smoothness when running heavy 3D games, yet the gaming experiences are still far from satisfactory. To handle this, the boundary of virtualization needs to be broken.



*MobiCom '19, October 21–25, 2019, Los Cabos, Mexico*



ACM ISBN 978-1-4503-6169-9/19/10.
https://doi.org/10.1145/3300061.3343372

We develop DAOW which, to the best of our knowledge, is the first and only emulator that can achieve the same level of smoothness when running heavy 3D Android games on Windows PCs, as being played on Android phones natively. At its heart lies the idea of *direct Android emulation*, which directly executes Android app binaries on top of x86-based Windows. Specifically, DAOW provides foreign Android binaries with direct access to the domestic PC hardware through Windows kernel interfaces, thus achieving nearly native hardware performance. Full technical details of DAOW [2] will be presented at the main conference of ACM MobiCom 2019.

## 2 THE DESIGN OF DAOW

In order to implement direct Android emulation, a series of challenges from the distinctions at different levels have to be dealt with, including ISA (ARM *vs.* x86), OS (Android *vs.* Windows), and device control (touch screen *vs.* physical keyboard and mouse). *First,* there exist significant distinctions in data structures and execution behavior of binaries between Android and Windows. One possible solution is to conduct instruction-level rewriting, yet it changes the layout of the original binaries and complicate the implementation. *Second,* Android/Linux and Windows have different sets of system calls (syscalls), which requires great engineering efforts to translate Linux syscalls to Windows, as well as incurs large runtime overhead if not appropriately implemented. *Third,* the interaction gap between mobile and PC-based gaming, which is rooted in the intrinsic hardware differences between mobile devices and PCs, also requires judicious consideration. For instance, PC games use physical keyboards and mouses for inputs while mobile games define a variety of buttons in different contexts. Furthermore, PCs' large screens could aggravate the subtle rendering issues of mobile games, thus leading to uncomfortable aliasing effect.

To address the above challenges, we make the following endeavors in the design and implementation of DAOW:

- *A data-driven, pragmatic approach is employed to fulfill cost-efficient instruction rewriting and syscall emulation.* We first comprehensively profile the instructions and syscalls involved in a wide variety of Android game apps. Based on this, we prune different types of instructions which need rewriting by reducing them to a few "patterns". For each pattern, we utilize trampolines and write native Windows utility functions so as to minimize the changes in binary structures during instruction rewriting. Besides, we prioritize the support for the popular syscalls while treating the rarely used ones as exceptions; we also leverage the "common divisors" among the syscalls to significantly reduce the engineering efforts.
- *We leverage a series of graphics techniques to bridge the interaction gap between mobile and PC-based gaming.* An intelligent mapping technique is introduced for dynamically detecting on-screen buttons and mapping them to appropriate keys of the physical keyboards. Moreover,a progressive anti-aliasing method that assembles multiple existing techniques is adopted with low overhead to smoothen rendering distortion and eliminate aliasing.
- *To further enhance the performance and gaming experiences, we make a number of optimizations in DAOW.* We improve the efficiency of syscall emulation through extensive resource sharing, early preparation, and delayed execution. We also employ shared memory for direct bulk data transfer between the app instances and the media component for real-time user interactions. In addition, we utilize security approaches to prevent external cheating programs from modifying Android game app instances, thus further enhancing users' gaming experiences.

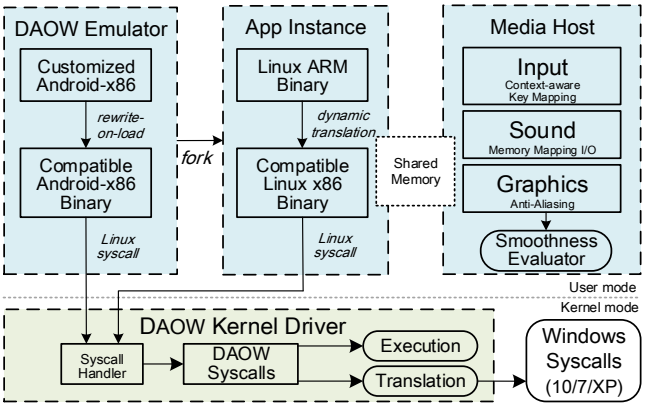


**Figure 1: Architectural overview of DAOW.**

As depicted in figure 1, our design of DAOW contains three components: 1) Emulator, 2) Kernel Driver, and 3) Media Host. The Emulator `inits` a customized Android framework which is decoupled from the original Android-x86 distribution (by removing the built-in Linux kernel and the unused services), and rewrites its binaries while loading them into memory. Then in order to allow dynamic translation from ARM binaries to x86 binaries, the Emulator forks an extra Windows process for running an Android game app. The Kernel Driver disposes Linux syscalls through a series of DAOW syscalls (*i.e.,* our refined "common divisors" among Linux syscalls)—they are either directly executed or translated into Windows syscalls for execution. Additionally, Media Host copes with user input, sound, and graphics issues, as well as measure the smoothness of the game.

**Implementation and Evaluation.** DAOW is implemented in ~500K lines of C++ code. Since its first launch in Sep. 2017, DAOW has been used by 50+ million users to run ~8000 heavy Android games on Windows PCs. Compared with

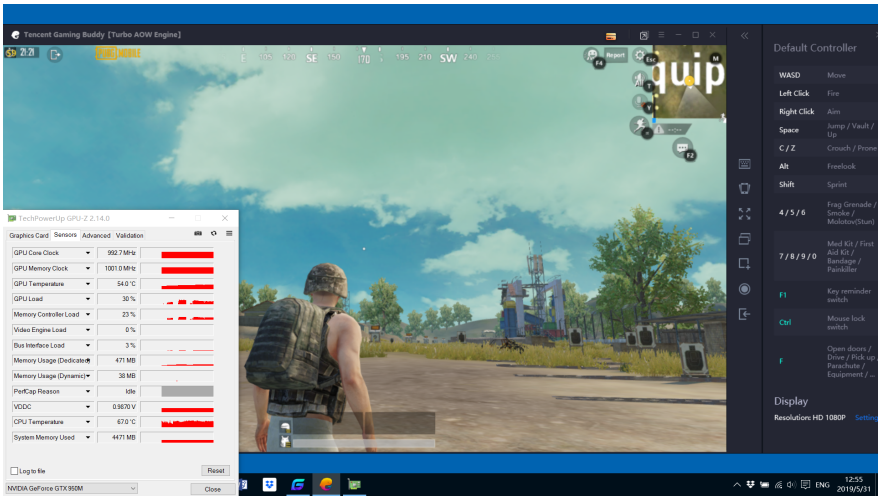


**Figure 2: A screenshot of DAOW (together with GPU-Z) when a 3D FPS game is running.**

AOVB-EMU, DAOW improves the smoothness by an average of 21%, from 0.76 ("rarely smooth") to 0.92 ("mostly smooth"), for millions of users when playing heavy 3D games. Besides, it decreases near half of (48%) the the game startup time and the memory usage by 22% on average. Please refer to our full paper [2] for details.

## 3 DEMONSTRATION PLAN

Our demonstration will use two commodity laptops (labeled as A and B) with the same hardware configurations, BIOS options (VT off by default), operating system, drivers, and Internet access. We will install an AOVB-based Android emulator and our developed DAOW on both laptops. Several popular heavy Android games and Android benchmarks will be installed in every emulator. Figure 2 shows PUBGM (a 3D first-person shooter game) running with DAOW on a laptop. The main panel of DAOW holds the virtual screen of an emulated Android instance. To create a more immersive experience, the mouse cursor is hidden by default. On the left of the main panel, there is a toolbar with useful features, such as full-screen mode and screen recording. The default controller panel shows the default key mapping of the mobile game. Specially, the "F" key is a context-aware multi-functional key that dynamically changes its key mapping to reuse available keys and resolve possible conflicts [2].

To monitor the utilization of system resources, we will install several third-party benchmark utilities such as CPU-Z and GPU-Z. The demonstration consists of two components: (1) comparing the smoothness of the graphics by running the same graphics benchmarks or game playbacks, and (2) letting the audience experience DAOW in person.

**Smoothness Comparison.** We run the graphics benchmarks and game playbacks (which reproduce the same gaming scenario without human intervention) on both laptops with different emulators. The audience will be able to tell the smoothness of the graphics different between the AOVB-based emulator and DAOW, with DAOW running on a much higher frame rate. Besides, from the statistics shown on third-party utilities, the audience can also quantitatively compare their resource usage including CPU and GPU usage.

**In-person Experience.** We will invite the audience to play the same heavy Android games on both types of Android emulators. Besides smoothness and details of graphics, the audience can also tell the difference of keyboard and mouse support, since DAOW provides a more user-friendly key mapping. Our demo requires the default space (one $6 \times 2.5$ ft table) and two power outlets. Wireless access point is needed as most popular Android games require Internet connectivity. The expected setup time of our demo is less than 10 minutes.

## REFERENCES


[1] Riad Chikhani. 2015. The History of Gaming. https://techcrunch.com/2015/10/31/the-history-of-gaming-an-evolving-community/.

[2] Qifan Yang, Zhenhua Li, Yunhao Liu, Hai Long, Yuanchao Huang, Jiaming He, Tianyin Xu, and Ennan Zhai. 2019. Mobile Gaming on Personal Computers with Direct Android Emulation. In *Proceedings of ACM MobiCom*.

[3] Rusty Russell. 2008. Virtio: Towards a De-facto Standard for Virtual I/O Devices. *ACM Operating Systems Review* 42, 5 (2008), 95–103.

[4] Rich Uhlig, Gil Neiger, Dion Rodgers, Amy L. Santoni, Fernando C.M. Martins, Andrew V. Anderson, Steven M. Bennett, Alain Kagi, Felix H. Leung, and Larry Smith. 2005. Intel Virtualization Technology. *IEEE Computer* 38, 5 (2005), 48–56.

[5] Tom Wijman. 2018. Mobile Revenues Account for More Than 50% of the Global Games Market in 2018. https://newzoo.com/insights/articles/global-games-market-reaches-137-9-billion-in-2018-mobile-games-take-half/.