Quantifying User Satisfaction in Mobile Cloud Games

Chun-Ying Huang[1], Cheng-Hsin Hsu[2], De-Yu Chen[3], and Kuan-Ta Chen[3]

[1]Department of Computer Science and Engineering, National Taiwan Ocean University, Taiwan
[2]Department of Computer Science, National Tsing Hua University, Taiwan
[3]Institute of Information Science, Academia Sinica, Taiwan

ABSTRACT

We conduct real experiments to quantify user satisfaction in mobile cloud games using a real cloud gaming system built on the open-sourced GamingAnywhere. We share our experiences in porting GamingAnywhere client to Android OS and perform extensive experiments on both the mobile and desktop clients. The experiment results reveal several new insights: (1) gamers are more satisfied with the graphics quality on mobile devices, while they are more satisfied with the control quality on desktops, (2) the bitrate, frame rate, and network delay significantly affect the graphics and smoothness quality, and (3) the control quality only depends on the client type (mobile versus desktop). To the best of our knowledge, such user studies have never been done in the literature.

Categories and Subject Descriptors: H.5[Information Systems Applications]: Multimedia Information Systems

General Terms: Measurement

Keywords: Cloud games, mobile games, performance evaluation, user studies

1. INTRODUCTION

Increasingly more cellular users own smartphones and tablets, e.g., majority ($> 50\%$) of cellular users in U.S., U.K., China, Australia, South Korea, and Italy have switched from feature phones to smartphones [11]. Many of these mobile users play mobile games, e.g., 59% of the mobile users have played games on their mobile devices in 2011 [12]. Moreover, the mobile gaming market share is expected to grow to 16 billion USD by 2016 [10]. Compared to console and desktop games, mobile games are often less visually appealing due to the limited CPU/GPU power, memory space/speed, network bandwidth, and battery capacity, which may drive *serious gamers* away from mobile games.

Cloud gaming renders, captures, and encodes game scenes on powerful cloud servers, and streams the encoded game scenes in real time over the Internet to less capable client devices. The client devices collect gamers' inputs and send them back to the cloud servers, in order to interact with cloud games [13]. Although cloud gaming appears to be very suitable for resource-constrained mobile devices, commercial cloud gaming solutions, from OnLive, GaiKai, G-Cluster, OTOY, Spoon, T5-Labs, and Ubitus, are mostly accessible on: (i) PCs using native or web applications or (ii) TVs using set-top boxes. We believe that *mobile cloud gaming* has not been widely deployed at least due to the following two serious challenges:

- **Steep development cost.** Traditionally, the game industry is conservative and careful about adapting to new consoles and development environments, for the sake of cost control. However, most of the cloud gaming platforms [1, 4, 14] dictate proprietary SDKs, which in turn discourages the game industry from adopting cloud gaming. Thus far, GamingAnywhere [6] is the only cloud gaming platform in the literature, which is fully *transparent* to games. That is, GamingAnywhere frees the game industry from porting their games to new SDKs.
- **High bars on user satisfaction.** Gamers demand for high-quality game scenes *and* low response delay, which are inherently difficult, especially on resource-constrained mobile devices. Existing measurement studies [9] concentrate on low-level objective metrics, and how these measurement results affect user satisfaction is not well understood.

In this paper, we addressed the two challenges in two steps. First, we optimize GamingAnywhere [6] client for mobile devices. GamingAnywhere is the first open-source cloud gaming platform, designed for high extensibility, portability, and reconfigurability. Optimizing GamingAnywhere client on mobile devices is non-trivial due to tight resource constraints, and specialized CPU/GPU architectures and mobile OS's. In this paper, we share our experiences in porting GamingAnywhere client to Android, while porting to other mobile OS's is also possible. To the best of our knowledge, our mobile client is the first mobile cloud gaming client that is transparent to games, i.e., gamers can use our client to play *any* PC games on their mobile devices. Details on our porting experiences are given in Section 3.

Second, we conduct user studies using the real mobile cloud gaming system to analyze how different configurations affect the gaming experience. We consider configurations with several parameters: resolution, frame rate, bitrate, and network delay. Both resolution and frame rate affect the visual quality under a given bitrate; however, the precise impacts depend on many other factors including game genres and device types. For example, car racing games may need higher frame rates, while strategy games need higher resolutions. Moreover, response delay greatly affects user experience [3], and network delay is a component of response delay [2]. The network delay does not seem to be controllable at first glance. This observation however is not always correct, for example: (1) cloud gaming platforms may place games in different data centers to control network latency [5] and (2) mobile clients may choose different access networks (such as 4G/LTE over 3G networks) for lower network delay. Our extensive user studies lead to several key observations:

- **Overall.** Gamers demonstrate diverse user satisfaction levels on desktops and mobile devices. Generally, gamers are more satisfied with: (1) the graphics quality on mobile devices and (2) the control quality on desktops.
- **Impacts of configurations on user satisfaction.** Encoding bitrate, frame rate, and network delay are the three most critical system parameters affecting the graphics and smoothness quality.

The rest of this paper is organized as follows. Section 2 surveys the current mobile cloud gaming research. Section 3 presents our experiences in porting GamingAnywhere client to Android OS. This is followed by the detailed user studies in Section 4. Section 5 concludes the paper.

2. RELATED WORK

All commercial cloud gaming platforms are closed and proprietary, and thus cannot be used in our user studies. GamingAnywhere [6] is an open-source cloud gaming platform consisting of three entities: the server, client, and portal. To use GamingAnywhere, a gamer first logs into the portal and selects a desired game. The portal then launches the chosen game and server on the same (virtual) machine in the cloud. The portal also notifies the client to set up connections to the server, which starts a game session. In the current paper, we port the GamingAnywhere client to mobile devices, and use it to conduct user studies, which is not possible on commercial cloud gaming platforms.

Several research projects [1, 4, 14] attempt to enhance the low-level performance of mobile cloud gaming. Shu et al. [14] propose to employ 3D warping technique to perform lightweight image processing at mobile clients, so as to increase coding efficiency and mitigate network delay. Hemmati et al. [4] propose to selectively encode game objects to reduce the required network bandwidth and processing power without affecting gaming quality. Cai et al. [1] propose to dynamically allocate cloud resources to meet the needs of mobile gamers, who often move across diverse contexts. These approaches [1, 4, 14] are not transparent to games, requiring game developers to adopt proprietary SDKs. In contrast, GamingAnywhere supports all PC games as-is.

Most existing measurement studies on cloud gaming are done using desktop computers [2, 7, 8, 13]. Chen et al. [2] and Shea et al. [13] concentrate on objective quality metrics, while Jarschel et al. [7, 8] consider subjective quality metrics. Different from these studies [2, 7, 8, 13], we take mobile cloud gaming into consideration. The performance of mobile cloud gaming has only been measured recently [9]. Lampe et al. [9] consider three low-level performance metrics: latency, energy, and cost, trying to demonstrate the feasibility of mobile cloud gaming. In contrast, we study how different configurations affect user satisfaction via extensive user studies, which is a key research problem to optimize mobile cloud gaming.

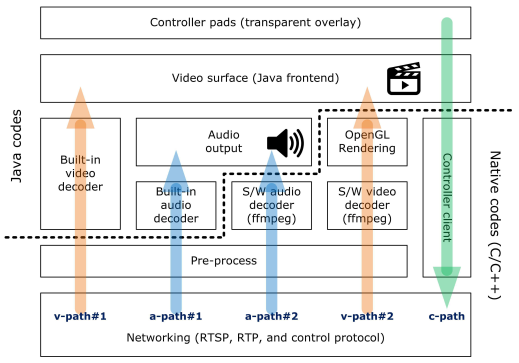


Figure 1: The architecture of the mobile client.

We emphasize that our GamingAnywhere client is a *transparent* and *open* mobile cloud gaming platform, which allows us (and other researchers) to use off-the-shelf PC games for experiments.

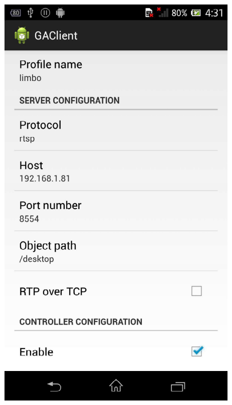
3. PORTING CLIENT TO ANDROID

Challenges. Mobile devices are connected to the Internet via WiFi or cellular networks, which incur long network latency and dictate an extremely efficient GamingAnywhere mobile client. Furthermore, computational power and battery life are two critical constraints on mobile devices. Therefore, modern mobile devices often come with built-in GPUs and audio/video hardware codecs, which are different from general-purpose CPUs. Last, due to small screen sizes, designing a unified game controller for all game genres is quite difficult. Details on how we address these challenges are given below.

Architecture. We implement the mobile GamingAnywhere client on Android OS. Figure 1 reveals the software architecture of the Android client. The skeleton of our mobile client is written in Java, but some components are implemented as loadable shared objects in native C and C++ codes. The native codes are mainly used for two purposes: (1) to bridge the codes between Java and GamingAnywhere library, and (2) to minimize maintenance overhead by sharing existing desktop client codes.

Networking component. This component is responsible for several operations, such as setting up RTSP connections, receiving RTP packets, and transmitting control packets. The same software component also manages audio/video buffers, parses the audio/video packet headers, extracts the MIME-type, and configures the decoders. The networking component is implemented in the native code, and integrated with the Java skeleton via Java Native Interface (JNI).

Decoding. The mobile GamingAnywhere client supports two types of codecs: (1) software codecs and (2) Android built-in codecs. The audio and video codecs are configured independently. Hence, there are four paths in Figure 1, i.e., `a-path#1` and `a-path#2` for audio and `v-path#1` and `v-path#2` for video. Software codecs are the same as those of GamingAnywhere desktop client. In contrast, the built-in codecs are provided by the Android `MediaCodec` frame-



80% 4:31

GAClient

Profile name
limbo

SERVER CONFIGURATION

Protocol
rtsp

Host
192.168.1.81

Port number
8554

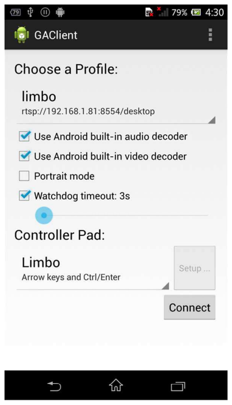
Object path
/desktop

RTP over TCP ☐

CONTROLLER CONFIGURATION

Enable ✓

(a) Edit profile for Limbo.



79% 4:30

GAClient

Choose a Profile:

limbo
rtsp://192.168.1.81:8554/desktop

✓ Use Android built-in audio decoder

✓ Use Android built-in video decoder

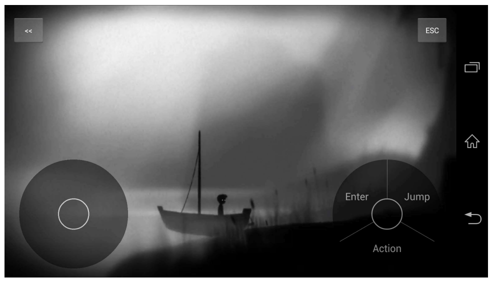
☐ Portrait mode

✓ Watchdog timeout: 3s

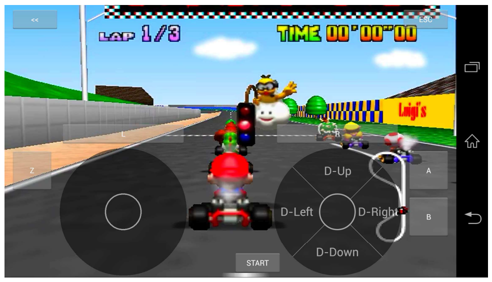
Controller Pad:

Limbo
Arrow keys and Ctrl/Enter

(b) Choose profile and controller pad for Limbo.



(c) Play Limbo with Limbo's controller pad.



(d) Play Mario Kart with N64 controller pad.

Figure 2: Screenshots of the mobile client.

work, which is available on Android 4.1 or later. `MediaCodec` framework is only accessible from Java side. Therefore, on receipt of an audio or video packet from the network, the packet is parsed in native codes and then passed to Java for decoding. For audio packets, decoded raw audio frames can be retrieved from the framework and then used for playback. For video packets, decoded video frames are directly presented to the user through a pre-created surface object. That is, the decoded raw video frames are not accessible to applications, and are automatically resized to fit the resolution of the surface object.

Rendering. The raw audio frames are always rendered with the `AudioTrack` framework in Java, no matter whether built-in or software codecs are used. When software codecs are used, the decoded video frames have to be first converted from YUV420P to RGB565 format. We then employ OpenGL ES library to resize and render decoded video frames. More specifically, each decoded frame is treated as a texture and drawn on an OpenGL surface. The `MediaCodec` framework and OpenGL ES adopt different surface objects: `SurfaceView` and `GLSurfaceView`, respectively.

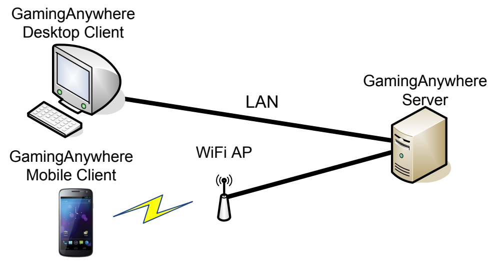


Figure 3: Our experiment setup.

Controllers. The controller pads are transparent overlays on top of the video surface. Designing a unified controller to support all game genres is out of the scope of this paper. Instead, we implement three representative controllers for gamers to choose from. The implemented controllers are designed for: (1) Nintendo 64 emulator, (2) Nintendo DS emulator, and (3) Limbo.

Practical concerns. Our mobile GamingAnywhere client is affected by two practical limitations. First, the H.264 *intra-refresh* option is not supported by some built-in codecs, especially those hardware-accelerated codecs. This option increases the robustness of video streaming under lossy wireless channels. We have tested this on several Android devices and found that at least the first generation Nexus 7 tablet does not support the option: the built-in codec freezes shortly after the video playout starts. A workaround is to disable *intra-refresh* on the GamingAnywhere server at the expense of degraded robustness. Second, some built-in codecs are not included in system images built from Android Open Source Project (AOSP). This is because these codecs require drivers that are not open-source. In that case, the mobile GamingAnywhere client only shows a blank screen.

Prototype implementation. Upon installing the client, a user first creates a profile using the user interface shown in Figure 2(a). Each profile consists of configurations, like server address, server ports, and codec parameters. Then, the user selects the desired profile and controller using the interface shown in Figure 2(b). Once the *Connect* button is pressed, the client connects to the server. Figures 2(c) and 2(d) show the rendered game screens with user-specified controllers for Limbo and Mario Kart, respectively.

4. EXPERIMENTS

In this section, we present our experiments that were designed to evaluate the user satisfaction in mobile cloud games. We focus on the differences in user satisfaction introduced by the mobile devices (i.e., compared with desktop cloud gaming), and the effect of various system parameters on the perception of mobile cloud gamers.

4.1 Experiment Setup

The experiment environment consists of three hosts: a server, a desktop client, and a mobile client. We set up our GamingAnywhere server on a Windows 7 desktop with an Intel Core i7-870 Processor (8 MB cache and 2.93 GHz) and

Table 1: Selected Games

Game Title	Genre	Platform
Limbo (limbo)	2D platform	Windows
Mario Kart 64 (kart)	Racing	Nintendo 64
Super Mario 64 (mario)	3D platform	Nintendo 64
Super Smash Bros. (smash)	Fighting	Nintendo 64

Table 2: A Summary of Experiment Settings

Period	2013/11/28–2013/12/19
Subjects	15 (5 females, 10 males)
Age	21–34 years old (mean 26.2 years old)
Total game sessions	1,020
Total gameplay time	1,020 minutes
Parameters†	
Resolution‡	640x480, **960x720**, 1280x960
Bitrate	1 Mbps, **3 Mbps**, 5 Mbps
Frame rate	5 fps, 20 fps, **50 fps**
Delay	**0 ms**, 150 ms, 300 ms

† Default values are highlighted in boldface.
‡ Screen resolution for Limbo is fixed at 1280x720.

8 GB main memory. The desktop client runs on a Windows 7 desktop with an Intel Core2 Quad Processor Q9400 (6 MB Cache and 2.66 GHz) and 4 GB memory, and the mobile client runs on a Samsung Galaxy Nexus (1.2 GHz dual-core CPU, 1 GB memory, AMOLED 4.65-inch screen, and 720p resolution) with Android 4.2.1. The desktop client and the mobile client were connected to the server via a Gigabit Ethernet LAN and an 802.11 wireless LAN, respectively. We ensured that both LANs are under-utilized during our experiments for fair comparisons. The experiment environment is shown in Figure 3.

4.2 Experiment Design and Data Collection

We select four games, as listed in Table 1, in different genres for this study. Three of the four games are from the Nintendo 64 platform, and we use the `mupen64plus` emulator to run those games. To study the user satisfaction under different configurations, we vary four system parameters: video resolution (resolution), encoding bitrate (bitrate), frame rate, and network delay (delay). In the experiments, we vary each parameter with three levels while keeping the other parameters to their default values. For resolution settings, we change the game resolution options in the `mupen64plus` emulator for the three Nintendo 64 games; unfortunately, Limbo does not allow resolution setting, so we can only run Limbo with 1280x720. We configure the frame rate and encoding bitrate at the GamingAnywhere server. As to the network delay, we employ `ipfw`[1] to increase the network delay of the traffic between the server and the clients. Detailed settings for each factor are listed in Table 2 with the default values highlighted in boldface.

We conduct two user studies: (1) desktop cloud gaming and (2) mobile cloud gaming. Each subject participates in both studies in random order, and each study consists of the four games in random order. Every subject plays each game under various configurations: 9 configurations for the Nintendo 64 games and 7 for Limbo, as Limbo disallows resolution changes. Therefore, each player was asked to play a total of $(9 \times 3 + 7) \times 2 = 68$ game sessions in our experiment. We require each game session to last for a minute, and the game is automatically terminated. Each subject is then prompted to evaluate their gaming experience in terms of the following three aspects on a five-level MOS scale:

- **Graphics**: The visual quality of the game screens.
- **Smoothness**: The negative impact due to delay, lag, or unstable frame rate observed during game play.
- **Control**: The quality of the control mechanism (i.e., keyboard for desktops and touch screen for mobile devices).

We recruited a total of 15 subjects to participate in our experiments and summarize the collected dataset in Table 2.

[1]`ipfirewall` (or `ipfw` in short) is a FreeBSD IP packet filter and traffic accounting facility. There is a port of `ipfw` and the `dummynet` traffic shaper available on Linux, OpenWrt and Microsoft Windows.

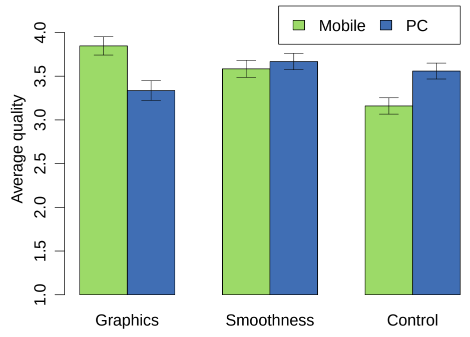


Figure 4: Overall MOS scores under different system parameters and client devices.

4.3 Mobile versus PC: Which One Is More Satisfying?

To compare the overall user satisfaction of GamingAnywhere clients on different devices, we first compute the overall MOS scores across all games and configurations on each device. Figure 4 gives the average results with 95% confidence levels. We make the following observations.

Graphics. Interestingly, the mobile client leads to significantly higher scores, although the mobile display is much smaller. This observation may be attributed to two reasons. Firstly, the subjects may have *lower expectation* on graphics quality in mobile games. Generally, mobile devices have relatively low computing and graphics rendering power compared to high-end desktops, as a result, most mobile games do not even try to compete with PC games in terms of graphics quality. While GamingAnywhere provides basically identical graphics qualities on both mobile and desktop client, they are evaluated with different standards. Secondly, as the physical dimensions of desktop and mobile devices are quite different (i.e., 27 versus 4.65 inches), the graphics imperfectness due to video encoding/decoding and network loss, such as blur, blocking effects, and mosquito noise, tends to be more easily spotted by subjects on desktop screens. It appears that the subjects rate the graphics quality based on a "minus principle." In other words, the satisfaction levels are rated based on the flaws observed rather than on the absolute quality of a stimulus.

Smoothness. Overall, the smoothness of game play on desktop and mobile clients is fairly close. This demonstrates that our GamingAnywhere client is well-tuned on both desk-

Table 3: Student's One-tailed t-tests for the Differences Between Desktop and Mobile Cloud Gaming

		overall	kart	limbo	mario	smash
Graphics	Mobile is better	6.4e-11***	0.011*	2.1e-06***	6.2e-04***	6.4e-04***
	PC is better	1.000	0.989	1.000	0.999	0.999
Smoothness	Mobile is better	0.889	0.764	0.769	0.186△	0.963
	PC is better	0.111△	0.236	0.231	0.814	0.037*
Control	Mobile is better	1.000	1.000	0.984	0.988	1.000
	PC is better	1.4e-09***	5.7e-05***	0.016*	0.012*	2.5e-04***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '△' 0.2 ' ' 1

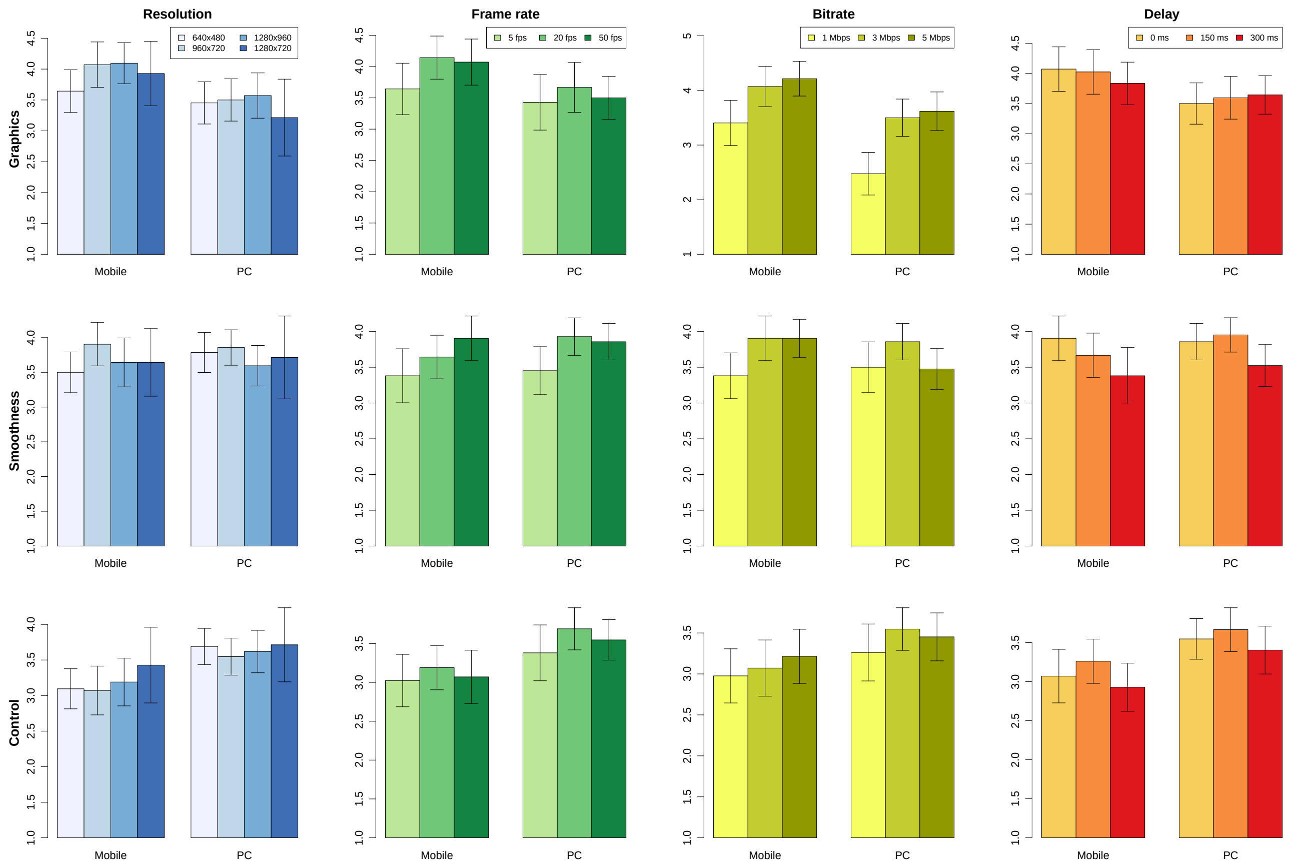


Figure 5: MOS scores associated with each factor setting and client device.

tops and mobile devices. Therefore, the software implementations do not bias the experiment results on the graphics and control quality.

Control. The desktop client performs better than the mobile version in terms of control. It is less surprising as the selected games are not specifically designed for mobile devices. On desktop computers, keyboards are capable of complicated key-stroke combinations and can be easily used to simulate controls of other platforms such as N64. In contrast, touch screens are much more restricted.

Next, we perform hypothesis tests to study how the device differences affect user satisfaction while the subjects play games in different genres. Table 3 lists the one-tailed t-tests for the scores from different client devices and games. The table reveals that the differences in graphics and control between the two clients are highly significant, with p-values less than 0.001. It is also shown that the subjects are much more sensitive to the graphics difference in Limbo than in Mario Kart. This can be attributed to the nature of the two games: Mario Kart is a fast-paced racing game, hence gamers may not pay too much attention on the degradation in graphics quality; in contrast, Limbo is relatively static, and gamers are sensitive to the graphics quality. The other observation is that the subjects are less sensitive to the difference in control in platform games (Limbo and Super Mario) than in fighting (Super Smash Bros.) and racing (Mario Kart) games. We believe this is because fighting and racing games are faster-paced and have AI opponents directly competing with you, therefore precise control is crucial for gamers to prevail in these games; while in platform games, you have time to prepare for your every action and some failed attempts are tolerable.

4.4 Impact of Various System Parameters

We report the average MOS scores (with 95% confidence levels) under different system parameters and client devices in Figure 5. Since there are three levels for each system pa-

Table 4: ANOVA Tests for the Impact of Individual System Parameters on User Satisfaction

		Res.	Frame rate	Bitrate	Delay
Graphics	Mobile	0.134△	0.030*	0.010*	0.404
	PC	0.894	0.169△	1.6e-04***	0.738
Smoothness	Mobile	0.210	0.013*	0.041*	0.042*
	PC	0.406	0.102△	0.929	0.085△
Control	Mobile	0.863	0.922	0.542	0.471
	PC	0.767	0.411	0.460	0.494

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '△' 0.2 ' ' 1

rameter in our experiments (except the 1280x720 resolution for Limbo), we use one-way analysis of variance (ANOVA) model to test whether each factor introduces significant effect on the user satisfaction in gaming. We had to exclude some data from Limbo, as Limbo disallows resolution changes. Table 4 gives the ANOVA results. This table shows that *bitrate* is the most important parameter that affects the graphics quality on both client devices. A deeper look reveals that the bandwidth needed to stream the game screen under the default resolution and frame rate is about 3.5 Mbps. When the bandwidth is not enough, data will be dropped and the graphic quality will severely decrease. The drop in frame rate also degrades subjects' perceived quality of graphics, although the impact is much weaker. One may expect the resolution to be a significant factor to graphic quality, however, the results show that it has only little if any effect on graphic quality. It is probably because the selected games have substandard graphics details compared to today's standards, therefore when the screens are upscaled to the same resolution on the client, the differences are hardly noticeable. We will look into more details on this observation in the future.

The smoothness of the games is affected by many system parameters. Not surprisingly, *network delay and frame rate* impose significant impact on smoothness, because lag and unstable frame rate directly result in low smoothness. In fact, high network delay and low frame rate both lead to the same negative impacts on gamers' reactions, which increases the gamers' levels of frustration. In addition to delay and frame rate, our analysis indicates that low bitrate may also affect the games' smoothness on mobile devices. We suspect that subjects may give low MOS scores on smoothness when the graphics quality is extremely low, as they can not play the game anyway. More detailed user studies to verify our hypothesis are among our future tasks.

Last, the control quality is not affected by any of the system parameters. Rather, it is affected by the client device types (desktops versus mobile devices) more.

5. CONCLUSION

In this paper, we have presented a mobile cloud gaming system built upon GamingAnywhere. We shared our experiences in porting a cloud gaming client to Android, which are also applicable to other mobile OS's. We used the mobile and desktop clients to conduct extensive user studies, so as to understand the implications of different system parameters (resolution, frame rate, bitrate, and network delay) on user experience (graphics, smoothness, and control). The experiment results reveal that: (1) gamers are more satisfied with the graphics quality on mobile devices and the control quality on desktops, (2) the bitrate, frame rate, and delay affect the graphics and smoothness quality the most, and (3) the control quality is mainly affected by the device type. Several future research directions are possible. For example, we have not observed the impact of resolution on user experience, which may be shown in larger-scaled user studies.

References

[1] W. Cai, C. Zhou, V. Leung, and M. Chen. A cognitive platform for mobile cloud gaming. In *Proc. of the IEEE International Conference on Cloud Computing Technology and Science (CloudCom'13)*, pages 72–79, Bristol, UK, December 2013.

[2] K. Chen, Y. Chang, H. Hsu, D. Chen, C. Huang, and C. Hsu. On the quality of service of cloud gaming systems. *IEEE Transactions on Multimedia*, 16(2), Feb 2014.

[3] M. Claypool and K. Claypool. Latency can kill: Precision and deadline in online games. In *Proc. of ACM SIGMM Conference on Multimedia Systems (MMSys'10)*, pages 215–222, Phoenix, Arizona, February 2010.

[4] M. Hemmati, A. Javadtalab, A. Shirehjini, S. Shirmohammadi, and T. Arici. Game as video: Bit rate reduction through adaptive object encoding. In *Proc. of ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'13)*, pages 7–12, Oslo, Norway, February 2013.

[5] H. Hong, D. Chen, C. Huang, K. Chen, and C. Hsu. QoE-aware virtual machine placement for cloud games. In *Proc. of IEEE Workshop on Network and Systems Support for Games (NetGames'13)*, Denver, CO, December 2013.

[6] C.-Y. Huang, K.-T. Chen, D.-Y. Chen, H.-J. Hsu, and C.-H. Hsu. Gaminganywhere: The first open source cloud gaming system. *ACM Transactions on Multimedia Computing Communications and Applications*, pages 1–25, Jan 2014.

[7] M. Jarschel, D. Schlosser, S. Scheuring, and T. Hobfeld. An evaluation of QoE in cloud gaming based on subjective tests. In *Proc. of International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS'11)*, pages 330–335, Seoul, Korea, June 2011.

[8] M. Jarschel, D. Schlosser, S. Scheuring, and T. Hobfeld. Gaming in the clouds: QoE and the users' perspective. *Mathematical and Computer Modelling*, 11-12(57):2883–2894, June 2013.

[9] U. Lampe, R. Hans, and R. Steinmetz. Will mobile cloud gaming work? Findings on latency, energy, and cost. In *Proc. of IEEE International Conference on Mobile Services (MS'13)*, pages 960–961, Santa Clara, CA, June 2013.

[10] Mobile gaming, July 2011. `https://www.abiresearch.com/research/product/1006313-mobile-gaming/`.

[11] The mobile consumer: A global snapshot, February 2013. `http://www.nielsen.com/content/dam/corporate/uk/en/documents/Mobile-Consumer-Report-2013.pdf`.

[12] PopCap games mobile gaming research, June 2012. `http://www.infosolutionsgroup.com/popcapmobile2012.pdf`.

[13] R. Shea, J. Liu, E. Ngai, and Y. Cui. Cloud gaming: Architecture and performance. *IEEE Network Magazine*, 27(4):16–21, July/August 2013.

[14] S. Shi, C. Hsu, K. Nahrstedt, and R. Campbell. Using graphics rendering contexts to enhance the real-time video coding for mobile cloud gaming. In *Proc. of ACM Multimedia (MM'11)*, pages 103–112, Scottsdale, AZ, November 2011.