A Hybrid-Streaming Method for Cloud Gaming: To Improve the Graphics Quality delivered on Highly Accessible Game Contents

Kar-Long Chan[1], Kohei Ichikawa[1], Yasuhiro Watashiba[1], Uthayopas Putchong[2], Hajimu Iida[1]

[1]*Nara Institute of Science and Technology, Ikoma, Japan*
{kar_long.chan.jr2, ichikawa, watashiba}@is.naist.jp
iida@itc.naist.jp
[2]*Kasetsart University, Bangkok Thailand*
pu@ku.ac.th

Abstract

The emerging Cloud Gaming Service provides a highly accessible video gaming experience. With Cloud Gaming, potential players without enough local resource can access high-quality gaming using low-spec devices. With advancing technology, we consider that if the processing power at low-spec devices can be well harvested, the quality delivered on Cloud Gaming can be further improved. Therefore, we propose a Hybrid-Streaming System that aimed at improving the graphic quality delivered by Cloud Gaming. By utilizing the available rendering power from both the Cloud Server and client PC, the system distributes rendering operations to both sides to achieve the desired improvement. Quantitative results show the proposed method improves graphics quality, as well as reducing the server's workload while attaining acceptable network bandwidth consumption levels.

Keywords: *Cloud Gaming, Hybrid-Streaming, Graphics Quality;*

1. Introduction

The emerging Cloud Gaming Service envisions an intriguing future providing millions of clients with novel and highly accessible gaming experiences. By leveraging reliable, elastic and high-performance computing resources, Cloud Gaming shifts the intensive workload of game processing from the client device to a powerful Cloud Server. In its simplest form, shown in Figure 1, the actual interactive gaming application is stored at the Cloud Server, where it is executed when requested. The rendered game scenes are then streamed back to the client device as an encoded video sequence over a network. At the client side, the input of control events from devices such as a mouse, keyboard, or joystick are recorded and sent back to the Cloud Server for the manipulation of game logics.

The on-demand feature of Cloud Gaming benefits both clients and game developers by easing possible incompatibility issues between the gaming software and the hardware environment. Therefore, recently it has been not only an active topic both in industrial [6] and research fields [7], but also a new area which possesses tremendous market value [5]. With Cloud Gaming, even potential customers without enough resources or those who cannot afford high-spec machines can access high-quality gaming using low-spec devices such as smartphones or tablets.

Ideally, Cloud Gaming is a powerful solution that lowers the technical boundary between players and high-quality game contents. However, a challenging objective when developing a sustainable Cloud Gaming service is to maintain and improve the client Quality of Experience (QoE), because network constraints play critical roles in affecting the system performance [3][4]. In general, interaction delay and graphics quality are two significant criteria that determine the client QoE. Cloud Gaming demands rigid real-time responsiveness to achieve a satisfying QoE [2][7], so currently most related research focuses on alleviating interaction delay [9][1]. On the other hand, findings of a conducted subject test show that clients are sensitive to changes in the graphics quality and smoothness during gameplay, which implies that the graphics quality also notably affects the QoE of Cloud Gaming [8].

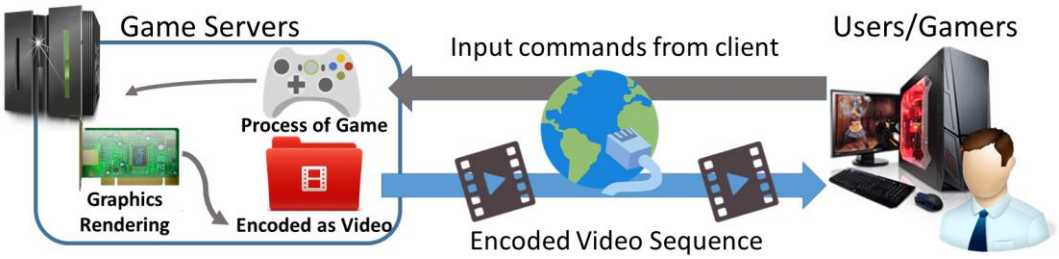


Figure 1. Brief Structure of Cloud Gaming

Furthermore, with advancing in-game visual effects and high resolution displays, the client demands for gaming with more realistic graphics on their devices are rising. However, in term of graphics quality, there is an obvious gap between the traditional local rendering and Cloud Gaming's streaming encoded video, which results in graphics quality significantly degraded from the original.

Furthermore, with the advancing technology, even resource-constrained devices such as smartphones or bare-bone computers are often equipped with built-in graphics processing units. Therefore, this study aims at enhancing the graphics quality delivered on existing Cloud Gaming system by utilizing this available processing power. Currently we address the use case of a PC, which is not necessary a significant powerhouse, but is capable of rendering to a certain extent. Based on the two existing streaming methods introduced in the next section, our approach allocates the available rendering power on the client side to achieve improved graphics quality. Furthermore, distributing rendering tasks to the client PC also mitigates the workload on the server.

2. Existing techniques

In general, two major streaming methods are used in Cloud Gaming: Image-based streaming and Instruction-based streaming. These two methods differ from each other in how the game contents are delivered from the server to the client.

2.1 Image-based Streaming

In a Cloud Gaming environment using Image-based streaming, game logics, which drive the progression of gameplay, are manipulated according to the client inputs received by the server CPU. Afterwards, graphics are rendered through a dedicated graphics processing unit. The rendered game contents are then encoded as video and streamed back to the client device. Up on receiving the encoded video, the client device decodes the contents and finally shows the corresponding frame on the display.

One advantage of Image-based streaming is that the encoded video, in terms of bandwidth consumption, is suitable for streaming in a general network environment. Normally, game contents are streamed as 720p video, while the quality can be raised to 1080p in a faster network environment. Another advantage is that since decoding can be processed using the low-cost decoder chips which are commonly embedded in client devices, this approach is ideal for running on resource-constrained devices. Therefore, given the wide availability of this method, most commercial Cloud Gaming service providers apply Image-based streaming to deliver game contents.

For example, GamingAnyWhere is the first and only available open source Cloud Gaming platform [7], which is based on the structure of Image-based streaming. It is designed to be highly extensible and customizable, allowing users to extend the capabilities of the platform with ease. In this research, Cloud Gaming, in general, refers to a system utilizing the Image-based streaming.

2.2 Instruction-based Streaming

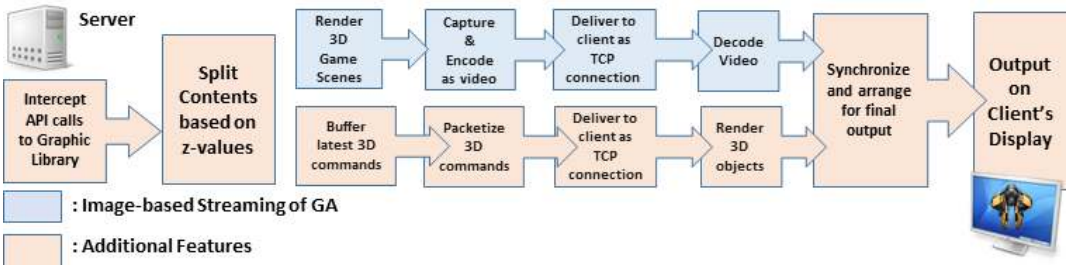


Figure 2. Data Flow of Proposed Hybrid-Streaming

In the work by Eiscert et al's [10], in a Cloud Gaming system using Instruction-based streaming, after the server processes the game logic, every API call to perform the corresponding rendering task is intercepted. The intercepted API functions, referred to the graphics commands, are compressed and sent to the client device. Together with the graphics command, related 3D data such as geometry mesh and texture are also streamed to the client device. After the arrival of this data, the client device processes the game rendering based on the received graphic commands.

The biggest advantage of Instruction-based Streaming over Image-based Streaming is the preserved original graphics quality, as the actual rendering is performed at the client device. Furthermore, without the heavy burden of 3D graphics processing, the Cloud Server is more effective at simultaneously handling more clients' requests. However, it requires the client device to be not only compatible with the delivered 3D graphics command, but also powerful enough to process high quality rendering, thus indicating less availability than Image-based streaming.

3. Hybrid-Streaming system

The goal of this research is to enhance the graphics quality above the level of Image-based streaming. As such, our proposed solution is to integrate the mechanism of Instruction-based streaming with Image-based streaming to achieve the desired improvement.

The Hybrid-Streaming System, which adopts both streaming methods, distributes partial game contents streamed as graphic commands to be rendered locally at the client PC. Simultaneously, it maintains partial rendering tasks at the Cloud Server and streams the corresponding contents as a video sequence. The improvement of the graphics quality is mainly contributed by the portion of the game contents rendered at the client PC. In addition, by offloading partial rendering tasks to the client device, the GPU workload at the server is reduced as well.

3.1 System structure overview

Figure 2 presents the overall structure of the Hybrid-Streaming System. The blue boxes indicate the original data flow of the Image-based streaming in GamingAnyWhere (GA) which our Hybrid-Streaming System is based on, while the orange boxes refer to the additional features for achieving the whole system.

Within the system, an important objective is how the products of two different streams are composed into the final representation of the game contents. This largely depends on the way of splitting the game contents at the server. In our system, all the objects are separated into two groups by comparing the depth value. The upper layer contains shallower objects (closer to the client view). The lower layer contains deeper objects (further away from the client view). Since the contents delivered through Image-based streaming result in a video frame without any depth factor, the game contents represented in this form should be treated as the background. For this reason, the objects belonging to the lower layer are streamed as a video sequence, which undergoes the original process of Image-based streaming. On the other hands, objects belonging to the upper layer are streamed as graphics commands. Therefore, soon as the client PC completes the rendering operations for the graphics commands, the products can be overlaid on top of the background provided by the contents from Image-based streaming. Figure. 3 provides a graphical representation of this process for producing the final product.

ISSN: 2384-8766 http://dx.doi.org/10.17083/ijsg.v4i2.163

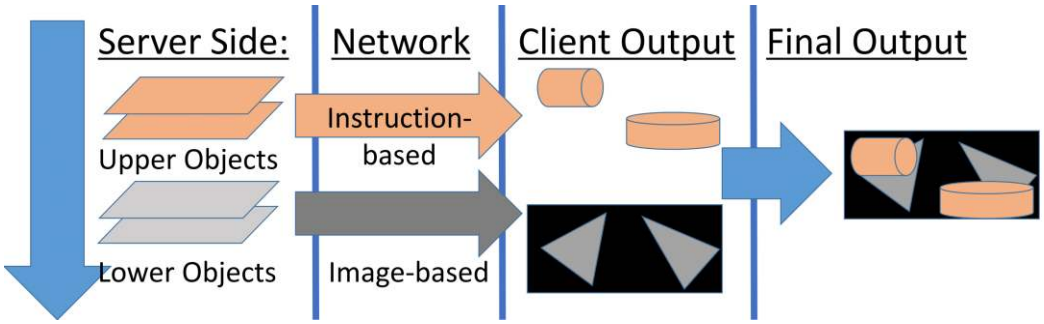


Fig. 3. Graphical Representation of Constructing the Final Output from Two Streaming Methods

In the following subsections, the data flow of Image-based Streaming is explained by exploring GamingAnyWhere (GA), which is the open-source Cloud Gaming platform that our work is based on. Furthermore, the additional features for achieving the Hybrid-Streaming System are introduced as well.

3.2 Structure of GamingAnyWhere's image-based streaming

In GA, the data flow refers to streaming audio and video frames from the server to the client. For our system, we focus on the graphics data manipulation in the data flow, indicated by the blue boxes in Figure. 2. After the server processes and renders the game, a designated video capturer implemented in GA is triggered to capture the contents in a polling manner. The captured data is then buffered and encoded by a customizable encoder module. Afterwards, the encoded video frame is delivered to the client through network. In GA's client module, the decoder only buffers the packets representing the most currently encoded video frame. Once the video is decoded, the buffer is cleared for the next frame. On the other hand, the audio streams and the control inputs from the client's device are the other two main streams that affect the overall gaming experience, but they are not the focus of this research. Compared with the graphics data, these two streams consume significantly less network bandwidth in the GA structure.

3.3 Additional features

To achieve the Hybrid-Streaming System shown in Figure 2, the necessary additional features include those for splitting the game contents, Instruction-based streaming, synchronizing the data from the two streams and arranging the final output.

Splitting the Game Contents -- To distinguish whether an object belongs to the upper layer or the lower layer during game processing, it is necessary to dynamically intercept the API calls to the graphics library and looks for the most updated z-value of each object's center point. A threshold z-value is defined for classifying which layer each object belongs to. Developers can define different thresholds to handle different in-game scenarios.

Contents' Processing Prior to Streaming -- After the lower layer objects are rendered at the server, the background undergoes the normal process for Image-based streaming of GamingAnyWhere, as described in the last subsection.

For Instruction-based streaming, the intercepted OpenGL instructions that correspond to rendering the upper-layer objects are not sent to the GPU at the server, but instead packetized for streaming to the client PC. Because usually the same sequence of graphic commands is routinely called to perform a particular task, we encode the sequence as a fixed code which indicates the corresponding task. As such, rather than delivering every native OpenGL instruction individually to the client, our system stream a light-weight code with other necessary parameters, maintaining a smaller overhead.

Furthermore, for other graphic data such as objects vertices and textures that potentially incur a heavy network load, prior to the actual gaming a copy of such data is streamed and saved at the client PC. Therefore, during the actual gameplay, only the graphics commands, which include the lightweight code and parameters described above, are streamed to the client device and hence, reducing the overhead. The Instruction-based streaming through network is based on a normal TCP connection, which is separated from the one of Image-based Streaming connection.

Data Synchronization -- Synchronizing the data of the two streams is another critical design objective. In our Hybrid-Streaming System, the latest graphics command for rendering the upper layer objects is buffered and updated in every game processing cycle. Whenever GA's source capturing is triggered, the most currently buffered graphics command and related parameters are packetized and streamed to the client. After the graphics commands are sent to client, the buffer is cleared and the system continues with the next iteration of API interception. Since the contents of the two streams traverse separated TCP connections, a precise global time stamp is appended to the respective headers during packetization. The global timestamp is used by the client module to identify the same-frame contents from the two streams.

Final Arrangement at Client Device -- After the video is decoded from Image-based streaming, an additional step assigns a suitable depth value to the video frame to ensure the background contents always behind locally rendered objects.

For on-site rendering, a parallel running thread is responsible for receiving the inputs from the Instruction-based streaming and decoding the contents afterwards. Based on the decoded lightweight code which represents a particular rendering task, the corresponding sequence of graphic commands is invoked with the received parameters. This allows the client PC to perform local rendering.

Finally, the embedded global timestamp is checked to confirm the same-frame contents before updating the next game frame. When confirmed, the game frame, composed of the locally rendered objects overlaid on top of the video frame, is updated on the client display.

4. Implementation

Based on the Image-based Streaming structure of GA, we implemented a prototype to simulate the output of our system. It allows us to achieve the current main goal of evaluating the graphics quality. At this stage, graphics commands together with related parameters are streamed directly from the source of our demo application. The client side then renders the corresponding 3D objects rendered based on the received instructions together with the necessary texture data saved in advance on the client device.

4.1 Instruction-based streaming

The current Instruction-based Streaming sends 3D graphics commands directly from our created demo application which is OpenGL-based. In the client module, a separate thread was developed for requesting graphic commands. This thread, which operates in parallel with the original GA's thread for decoding video sequence, periodically sends requests to server for the latest graphic commands. In addition, a lock is used to synchronize the contents from the two streams.

As for the construction of 3D graphics commands, a sequence of OpenGL instructions which are responsible for a particular function is represented by an ID. Functions here refer to changing the object's location, rotation, scale, environment lighting effects or camera positions. For the corresponding rendering operation to be performed properly on the client side, the data that should be packetized includes the Object ID which indicates the object to be rendered, the Function ID which refers to the function to be performed, and the set of related parameters. In addition, multiple functions can be aggregated into a single packet by simply putting the function set, which includes the parameters, one after another. The current packet layout is designed only for the minimum required data in this preliminary implementation, so it can be adjusted to adopt more patterns for more complex gaming applications later.

ISSN: 2384-8766 http://dx.doi.org/10.17083/ijsg.v4i2.163

Fig. 4. Demo Application Output

4.2 Arrangement in the client module

After decoding the received video frame, the contents of the lower layer are mapped as a texture on an OpenGL-based rectangular-shaped polygon, which represents the background objects of the final output. By setting a suitable depth value for this polygon object, the background contents are always further away than any locally rendered 3D objects.

As for the contents of the upper layer, the client module decodes the corresponding packetized graphic command sets. Afterwards, OpenGL functions are called to render the objects into a buffer, which also contains the polygon object representing the background contents.

Finally, when the buffered products are requested for updating the next frame on the display, the locally rendered 3D objects correspondingly overlay on top of the background contents.

5. Measurement

For our current primary goal of improved graphics quality, we compared the graphics quality of our prototype Hybrid-Streaming System with the original Image-based streaming system of GamingAnyWhere. We also evaluated the graphics card usage and the network load of both systems. Since using existing game software as test case is overly sophisticated for the prototype stage of the system, we instead a simple interactive demo application for the evaluations.

5.1 The demo application for evaluation

Figure 4 shows the output of the OpenGL-based demo application, in which three jet-fighters are presented. The main background, which is a mountain landscape scene, is a base element that is always classified as a lower layer object. Therefore, it is always delivered to the client by Image-based streaming.

In addition, the location of all the jet fighter objects can be explicitly defined in the demo, so we can also conveniently classify whether the object belongs to the lower layer or the upper layer. For example, in Figure 4, the two closer jet fighters can be classified as upper layer objects, while the further one together with the landscape background are classified as lower layer objects and streamed as encoded video. Moreover, more 3D objects can easily be added to the scene, which enables us to perform testing with more complicated scenarios.

5.2 System environment

As for the evaluation setup, the server machine is equipped with an Intel Core i7 4770K 3.5GHz CPU, a Nvidia GeForce GTX780 graphic card and 16GB system memory, running a 64-bit CentOS Linux system. The client machines include one Laptop A (2.6GHz Intel Core i7 CPU, Nvidia GeForce GT750M with 16GB system memory) and four Laptop B (2.1GHz Intel Core i7, Intel HD Graphics 4000 with 8GB system memory). Each machine had a pre-compiled GA module installed. The Image-based streaming of GA is set to stream the demo application with a resolution of 1280 x 720. The testing was conducted in a campus network environment.

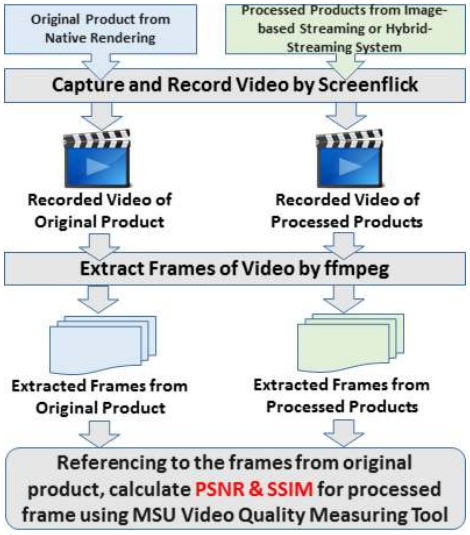


Figure 5. Process of Measuring PSNR and SSIM

5.3 Procedure for measuring graphics quality

Referring to previous researches related to Cloud Gaming [11], Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity (SSIM) are measured to evaluate the graphics quality. PSNR refers to the ratio between the signal and the noise introduced by compression, with a typical range between 30 and 50 decibel (dB). On the other hand, SSIM is an index for determining the similarity between two images, with a value that ranges from -1 to 1. For both metrics, the higher the value is, the better the graphics quality it indicates.

As shown in Figure 5, to take the measurement of PSNR and SSIM, a short video was captured from the output of the client PC with the contents delivered either by Image-based Streaming or Hybrid-Streaming. For the reference source, we captured a short video of the demo natively running on our Linux server. After acquiring a video source for each mode, we extracted every frame from the videos. Then, a chosen frame from the Image-based streaming and the Hybrid-Streaming videos was used to calculate the PSNR and the SSIM with reference to the original quality frame. We measured the PSNR and the SSIM under three scenarios.

5.4 Measuring GPU usage and network traffic load

The Hybrid-Streaming System is also expected to reduce the server workload as well, since partial rendering tasks are offloaded to the client PC. Therefore, we measured the respective GPU usages of the server and all the clients' devices by utilizing tools including nvidia-smi [12] and Intel-gpu-tools [13].

In addition, developing a Cloud Gaming system that is viable in common broadband network environment is another critical design objective. Since the implemented Instruction-based streaming in our proposed system may incur additional overhead, to evaluate the availability of the system in a broadband network, it is necessary to measure the traffic load. For this reason, we used IpTraf [14], a network monitoring system, to investigate the network bandwidth consumption during streaming of the demon application contents.

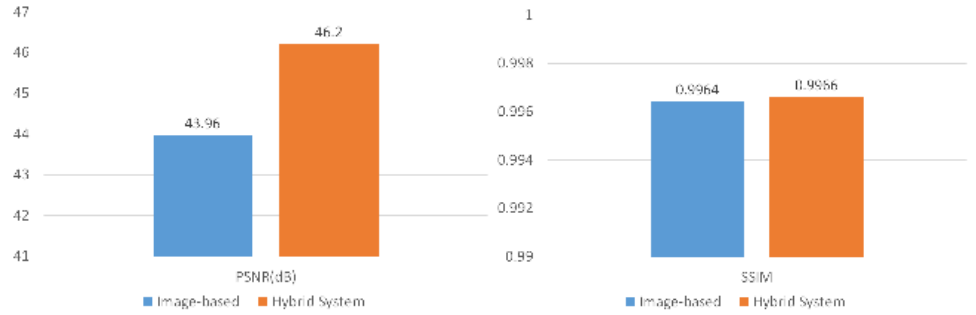
Both the measurements were taken with the scenario presenting 30 jet fighters in the scene. At the Cloud Server, five instances of the demo applications were executed and streamed to five client devices. In addition, equal numbers of jet fighters were distributed to each client device to be rendered locally, where the measured cases include 0 jet fighter (Image-based Streaming), 6 jet fighters, 12 jet fighters, 18 jet fighters, 24 jet fighters and 30 jet fighters.

6. Results of Evaluation

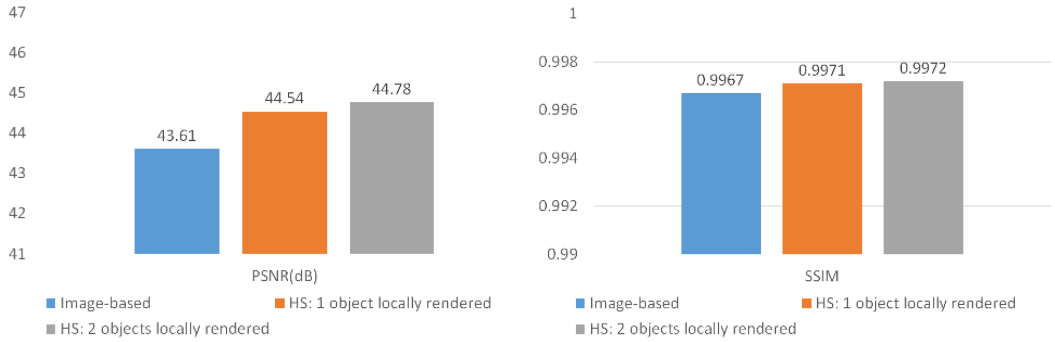
6.1 PSNR and SSIM results

Figure 6(a) shows the PSNR and the SSIM of the test case with one jet fighter displayed by the demo application. The Hybrid-Streaming System, referred to as HS in the table, with the jet fighter object is locally rendered on the client PC, achieved 46.20dB compared to 43.96dB for Image-based streaming. For SSIM, the Hybrid-Streaming System also achieved better results than Image-based streaming, but the difference is not very significant. This maybe because SSIM is sensitive to a different aspect from PSNR.

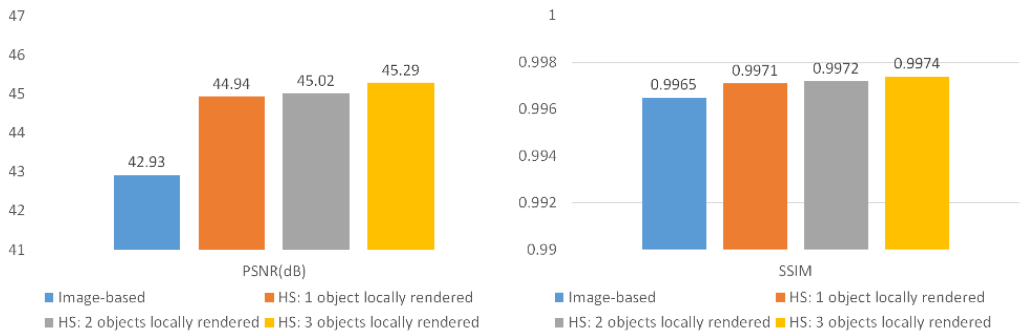
For the other two testing scenarios, overall the Hybrid-Streaming System achieved better results than Image-based streaming for both PSNR and SSIM. Furthermore, among scenarios with the Hybrid-Streaming System, when more 3D jet fighters are streamed as graphics commands, increasingly, better graphics quality is indicated. This is reasonable because local rendering preserves the original quality of the 3D objects.



(a) With 1 jet fighter



(b) With 2 jet fighters

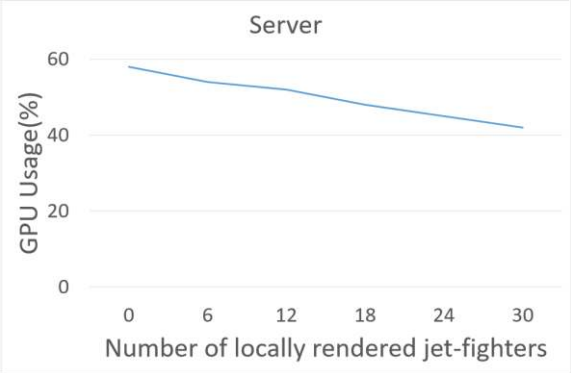


(c) With 3 jet fighters

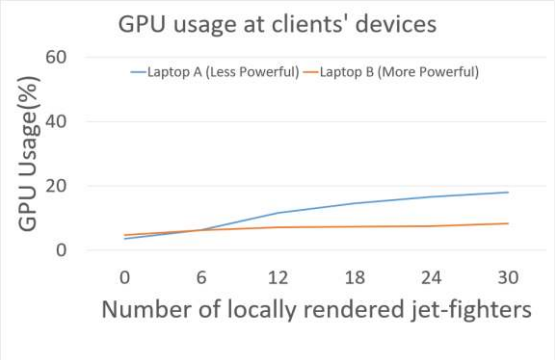
Figure 6 Evaluation of the Graphics Quality

6.2 GPU workload at server and client

As shown in Figure 7(a), the measured GPU usage at the server and the client PCs demonstrated expected behavior. When more objects are distributed to the client PCs, the GPU usage at the server, which is mainly due to object rendering and video encoding, decreases gradually. Simultaneously, as more objects are assigned to be rendered locally, the GPU usage of each client device, which mainly includes video decoding and object rendering, increases. Figure 7(b) shows the respective GPU usages of Laptop A and Laptop B (Average value of the four laptops). Since Laptop A is equipped with a more powerful GPU, the utilization is lower than the less capable built-in GPU of Laptop B.



(a) GPU usage at the server



(b) GPU usages at the client devices

(c) Consumed network bandwidth

Figure 7 Evaluation Results of GPU usage and Consumed Bandwidth

6.3 Network bandwidth consumption

The results of the measured network bandwidth, presented in Fig. 7(c), do not indicate an increasing trend. As such, the overhead incurred by the additional Instruction-based streaming mechanism is relatively trivial. We believe that the bandwidth is mostly consumed by delivering the background contents as encoded video. We expect that more sophisticated graphics commands in actual gaming applications will likely lead to a larger overhead. Even so, as the proposed Hybrid-Streaming System saves related graphical data such as textures and geometry meshes at client devices prior to actual gaming, the size of the overhead should not be dramatically increased by streaming only the light-weight graphics command set. Furthermore, since the the global average connection speed is estimated to be around 7Mbps [15], a Hybrid-Streaming system should be viable in most environment since the average consumed bandwidth shown in Fig. 7(c) is approximately 3.2Mbps (3200 kbps).

7. Discussion

7.1 Determination of the z-value threshold

The depth threshold for determining whether an object belongs to the upper layer or the lower layer is currently specified by game developers. However, when an object crosses the depth boundary, to align correctly, the same object must be rendered both at the Cloud Server and the client PC. This leads to redundant rendering. As a solution, we consider to use a different set of zNear and zFar at the server side to clip out corresponding contents, thus reducing the redundancy.

7.2 Compatibility with other genres of games

The developed demo application provides a perfect situation for evaluating the Hybrid-Streaming system. However, further in-depth testing with different genres of gaming applications is necessary. Especially, we need to determine whether the difference in terms of graphics quality is significant in real gaming application. We consider that our system should benefit game genres such as 2D scrolling, fighting and FPS, since these games usually have clearly distinguished background and foreground objects.

7.3 Usage with more complicated gaming applications

At the current prototype stage, we have only applied our demo application for evaluation. Based on our findings and observations, we have gained some insights and expectations about usage with more complicated gaming applications. First, in a more complicated gaming application, it is likely that advanced visual-effects such as fog, lens flare and motion blur will be applied. Therefore, our system must apply corresponding effects in renderings both at the client device and the cloud server. In addition, since visual effects offloaded to client device may additionally increase the corresponding GPU workload, it is critical for our system to balance the delivered graphics quality and client GPU workload, including processing visual-effects.

Furthermore, to implement the Instruction-based streaming mechanism in our system, special modifications were required in the source code of the demo application. However, we believe that it would be difficult to make direct modification in the source code of more complicated gaming applications. To avoid this, one of our future works is to include all the necessary functions in a library which can easily be adapted by Cloud Gaming frameworks without any necessity to modify the application source code.

7.4 Significance of the improvement

Based on the demo application, the current quantitative evaluation shows the improvement in the graphics quality, but the difference is not significant. Therefore, it is too early to be conclusive about our work. We expect that with more complicated visual-effects, our system will demonstrate more obvious advantages. Furthermore, we also plan to conduct qualitative evaluations to comprehensively measure the actual improvement from the player's perspective.

7.5 Accessibility from resource-constrained devices

The main goal of the Hybrid-Streaming system is to improve the delivered graphics quality for Cloud Gaming, while maintaining high accessibility to game contents by users with resource-constrained PCs. Based on the evaluation results shown in Figure. 7(b), graphics processing power on the client machine can be utilized to achieve improved quality. Furthermore, compared to the more capable laptop A, laptop B's built-in graphics processing unit had higher utilization, but maintained a reasonable range. We believe that in a scenario where a more complicated game scene occurs, a resource-constrained PC such as laptop B could be over-utilized. Therefore, in our future work, we will optimize the system by balancing the tradeoff between graphics quality improvement and the workload at the client device. In addition, we will expand the system compatibility with other resource-constrained devices such as smartphones or tablets.

8. Conclusion and future works

In this paper, we have presented a Hybrid-Streaming System for the purpose of enhancing graphics quality delivered in Cloud Gaming by utilizing the available graphics processing power on the client device. To construct the proposed system, we integrated the mechanism of the Instruction-based streaming with the more prevalent structure of Image-based streaming. By distributing rendering tasks and utilizing the graphics processing resource of both the client device and the server, enhanced graphics quality in Cloud Gaming can be delivered. Quantitative measurement shows that an implemented prototype of the Hybrid-Streaming System achieved better graphics quality than GamingAnyWhere's Image-based streaming, while maintaining insignificant overhead during network streaming. The evaluation also showed that the workload on the client devices increased but maintained a reasonable level, which implies Cloud Gaming's high accessibility was preserved in our experiment. Furthermore, the Hybrid-Streaming System also alleviated the graphics processing workload at the server by offloading rendering tasks to the client device.

Currently, the Hybrid-Streaming System is still at the prototype stage, and more thorough investigations are needed to evaluate the system as a prospective Cloud Gaming platform. In future works, we plan to qualitatively evaluate the QoE of the system, as well as applying actual gaming software to conduct further testing. In addition, we will continue to improve the applicability of the Hybrid-Streaming System by developing a more comprehensive framework. Under different gaming scenarios, it is also critical to keep a reasonable workload at the client device to maintain high accessibility of video game contents for Cloud Gaming. Therefore, optimizing the tradeoff between the graphics quality improvement and the workload at the client device is a critical part in our future work. At the same time, we will expand compatibility of the system with other resource-constrained devices such as smartphones and tablets.

References

[1] W. Cai, C. Zhou, V. Leung, and M. Chen. A cognitive platform for mobile cloud gaming. In 2013 IEEE 5th International Conference on Cloud Computing Technology and Science (CloudCom), volume 1, pages 72–79. IEEE, 2013"H3DAPI Datasheet," https://doi.org/10.1109/cloudcom.2013.17

[2] K.-T. Chen, Y.-C. Chang, P.-H. Tseng, C.-Y. Huang, and C.-L. Lei. Measuring the latency of cloud gaming systems. In Proceedings of the 19th ACM international conference on Multimedia, pages 1269–1272. ACM, 2011. https://doi.org/10.1145/2072298.2071991

[3] K.-T. Chen, C.-Y. Huang, and C.-H. Hsu. Cloud gaming onward: research opportunities and outlook. In 2014 IEEE International Conference on Multimedia and Expo Workshops (ICMEW), pages 1–4. IEEE, 2014. https://doi.org/10.1109/icmew.2014.6890683

[4] V. Clincy and B. Wilgor. Subjective evaluation of latency and packet loss in a cloudbased game. In 2013 Tenth International Conference on Information Technology: New Generations (ITNG), pages 473–476. IEEE, 2013. https://doi.org/10.1109/ITNG.2013.79

[5] "Cloud Gaming Report 2012. Distribution and monetization strategies to increase revenues from cloud gaming," http://www.cgconfusa.com/report/documents/Content-5minCloudGamingReportHighlights.pdf

[6] T. Geron. Sony to acquire cloud gaming startup gaikai for $380 million. http://www.forbes.com/sites/tomiogeron/2012/07/02/ sony-to-acquire-cloud-gaming-startup-gaikai-for-380-million/, 2012.
[7] C.-Y. Huang, C.-H. Hsu, Y.-C. Chang, and K.-T. Chen. Gaminganywhere: An open cloud gaming system. In Proceedings of the 4th ACM multimedia systems conference, pages 36–47. ACM, 2013. https://doi.org/10.1145/2483977.2483981
[8] M. Jarschel, D. Schlosser, S. Scheuring, and T. Hoßfeld. An evaluation of qoe in cloud gaming based on subjective tests. In 2011 Fifth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), pages 330–335. IEEE, 2011. https://doi.org/10.1109/IMIS.2011.92
[9] L. Xu, X. Guo, Y. Lu, S. Li, O. C. Au, and L. Fang. A low latency cloud gaming system using edge preserved image homography. In 2014 IEEE International Conference on Multimedia and Expo (ICME), pages 1–6. IEEE, 2014. https://doi.org/10.1109/icme.2014.6890279
[10] Eisert, P. and Fechteler, P.: Low delay streaming of computer graphics, 15th IEEE International Conference on Image Processing, 2008, IEEE, pp. 2704--2707 (2008). https://doi.org/10.1109/icip.2008.4712352
[11] Jurgelionis, A., Bellotti, F., De Gloria, A., Eisert, P., Laulajainen, J. and Shani, A.: Distributed video game streaming system for pervasive gaming, STreaming Day 2009 (2009).
[12] "nvidia-smi," https://developer.nvidia.com/nvidia-system-management-interface
[13] "Intel-gpu-tools," https://01.org/linuxgraphics
[14] "IpTraf," http://iptraf.seul.org/
[15] Akamai. (2016). Akamai's [state of the internet] Q4 2016 report. Retrieved from https://www.akamai.com/us/en/multimedia/documents/state-of-the-internet/q4-2016-state-of-the-internet-connectivity-report.pdf