Master’s Thesis

A Hybrid Game Contents Streaming Method to Improve Graphic Quality Delivered by Cloud Gaming

Kar Long Chan

February 5, 2015

Department of Information Systems
Graduate School of Information Science
Nara Institute of Science and Technology
A Master's Thesis
submitted to Graduate School of Information Science,
Nara Institute of Science and Technology
in partial fulfillment of the requirements for the degree of
MASTER of ENGINEERING

Kar Long Chan

Thesis Committee:
Professor Hajimu Iida (Supervisor)
Professor Hirokazu Kato (Co-supervisor)
Associate Professor Kohei Ichikawa (Co-supervisor)
A Hybrid Game Contents Streaming Method to Improve Graphic Quality Delivered by Cloud Gaming*

Kar Long Chan

Abstract

In gaming industry, Cloud Gaming is a new form of gaming service on trend trying to serve millions of players around the world with novel gaming experience. It aims to provide high-quality gaming service at any device, including thin clients which are incapable of handling high-definition gaming softwares. Players are only required to use any device that can connect to cloud servers for receiving streaming data through network and display game contents. Ideally Cloud Gaming is a promising service of providing novel gaming experience but in reality, various technological barriers make Cloud Gaming not comprehensively feasible for every type of game, such as first person shooting game which requires fast responsiveness. In addition, most existing cloud service, which streams encoded video sequence back to the client, is difficult to catch up with the rising demands for graphic quality. In this thesis, the proposed hybrid game contents streaming method aims at improving graphic quality delivered from Cloud Gaming. Game contents streamed from cloud server are split into two parts, as one part is streamed as video sequence. Another part, which is streamed as graphic commands, is for rendering job to be performed at the client side. The locally rendered game contents are then displayed accordingly together with the contents streamed as video sequence. By taking the advantage of distributing rendering operation, the combined computing power is expected to produce better graphic quality.

Keywords:

Cloud Gaming, Hybrid-Streaming Method, Graphic Quality
Acknowledgements

Being able to conduct my Master study in NAIST is considered to be one of the most precious experiences in my life. Not only I could live in the well-equipped, friendly environment of NAIST, and also I have been able to study and research under valuable guidances of many bright professors. Receiving generous helps, advices and caring from many great friends and great people around me drive me this far to achieve my master thesis. Time I have stayed in NAIST is so meaningful and I would like to sincerely express my gratitude to many helpful ones.

First and foremost, I would like to show my deep appreciation to Professor Hajimu Iida, who grants me the priceless opportunity to study in Laboratory for Software Design and Analysis. His enthusiasm to educate students and his very bright attitudes create a very positive, healthy research environment of SDLab, in which I have been able to passionately conduct my research. His advices and guidances on my research also give me confidence to continue in the field of Cloud Gaming.

To Professor Hirokazu Kato, I have been feeling so grateful and fortunate to learn essential knowledge of Computer Graphic in his lecture, because such knowledge is largely related to my research on Cloud Gaming. As my thesis committees, his suggestions given in my Seminar presentation also benefits me with improving some critical areas of my research.

Next I would like to also have my deepest thanks to Associate Professor, Kohei Ichikawa. Professor Ichikawa has been providing guidance since 3 years ago, as I was an internship student in Osaka University. By that time he suggested me to
research on visualisation, which eventually drives me to conduct my main study in Cloud Gaming. His continuous helps for my research and his supportive attitude to push me forward help me to successfully accomplish the master course.

I would like to thank my thesis committee. Thank you so much for reviewing my thesis and for many insightful comments and suggestions, which help me to improve the thesis.

Also I want to specially thank Professor Shinji Shimojo and Associate Professor Susumu Date in Cyber Media Centre of Osaka University for providing a precious opportunity of internship at their lab three years ago. Experience from my internship is a deciding factor for me to pursue my Master study in Japan, which leads me to having Master study in NAIST at present. Furthermore, under their guidances, I obtained important skills and knowledges of Computer Science, through which I became more passionated to continue my study in this field.

In the summer of year 2014, I was offered a great chance to having a one-and-half-month internship at Kasetsart University in Thailand. Through this one month, I was able to receive a lot of significant advices from assistant professor Putchong Uthayopas and associate professor Chanting Chantrapornchai. Their concerns on network environment inspire me with a lot of ideas that could be shaped up for future research. Furthermore, Thai students and friends that I met are very friendly, and their hospitalities make my stay in Thailand so memorable.

Being able to study with many bright students in SDLab makes my Master course better than the best. I want to thank to all SDLab members for providing helps to live in japan. Their passions in study and their generous advices on my research encourage me to do better and work harder. Especially I want to express my appreciation to Xin Yang for his advices and inspirations of expanding my research interest into gamification, which I find myself deeply interested in. Furthermore, I also want to specially thank to Naoki Nakayama and Erina Makihara for their guidances on Japanese languages, as I become more proficient to communicate with Japanese people around.

Finally and the most importantly, from the bottom of my heart, I want to have my most sincere thank you to my beloved parents and younger sister. For
their unconditional love and endless supports since everything began, I am able to overcome difficulties and achieve what I have now. It is wordless to express my greatest gratitude to them, who let me to chase for my dreams. By completing this Master course, I am able to fulfil my promise with them and I want to show them even brighter achievement by pursuing for a PhD. I hope to dedicate this thesis to them.
Contents

Abstract i

Acknowledgements iii

1 Introduction 1
  1.1. Concept of Cloud Gaming ................................. 2
  1.2. Conducted Evaluations on Cloud Gaming .................. 3
  1.3. Research Motivation ...................................... 4
  1.4. Research goal ............................................. 6
  1.5. Organization of Thesis ..................................... 7

2 Related Work 8
  2.1. Image-based streaming .................................. 8
  2.2. Instruction-based streaming .............................. 10
  2.3. Other Approaches ...................................... 12

3 Background of Proposed Method 13
  3.1. Comparison between Image-based Streaming and Instruction-based Streaming ........................................... 14
  3.2. Position of Our System .................................. 17

4 The Approach 19
  4.1. Process of Image-based Streaming ....................... 19
    4.1.1 Capturing Game Source ............................. 20
List of Figures

1.1 A brief structure of Cloud Gaming ............................................. 3
2.1 An overview of GamingAnyWhere’s image-based streaming structure 10
2.2 An overview of instruction-based streaming structure ...................... 11
4.1 The video data control structure of GamingAnyWhere ...................... 20
4.2 The overall structure of proposed hybrid streaming system .............. 24
4.3 Current method of splitting objects at server side and arranging
    for the final output at client’s side ......................................... 25
5.1 Configuration for Applied Encoder ........................................... 28
5.2 Configuration for Applied Encoder ........................................... 28
5.3 Encoder Parameter .................................................................... 28
5.4 Demo with 1 jet-fighter object .................................................. 30
5.5 Demo with 3 jet-fighter objects ................................................... 30
5.6 Contents inside Demo being split for server side and client’s side .... 31
5.7 Matrix for converting YUV420 to RGB ........................................ 33
5.8 Flow of Processing in our client module ..................................... 33
7.1 Run charts of PSNR in the case of 2 jet-fighter objects and 3 jet-
    fighter objects ........................................................................ 40
7.2 Run charts of SSIM in the case of 2 jet-fighter objects and 3 jet-
    fighter objects ........................................................................ 40
8.1 Demo when displaying 1 jet-fighter object ..................................... 51
8.2 Demo when displaying 3 jet-fighter object................. 52
List of Tables

3.1 Comparison between Image-based streaming and Instruction-based streaming ................................. 14

6.1 Results of PSNR and SSIM: Scenario of 1 jet-fighter on display . . 37
6.2 Results of PSNR and SSIM: Scenario of 2 jet-fighter on display . . 37
6.3 Results of PSNR and SSIM: Scenario of 3 jet-fighter on display . . 38
6.4 Graphic Card workload at server ............................... 38
Chapter 1

Introduction

Video Game industry has been considered as an essential sector in media industry, joining with the other two sectors including movie industry and music industry. The global market of video game is having rapid revenue growth, with an expectation of rising from $67 billions in 2013 to $82 billions [1] in 2017. In order to dynamically adjust to the complex and highly competitive business environment, video game industry is innovating its business model with the help of cloud technology. Efficiently utilising large amount of data through widely deployed data centre helps to form the diversity of video game industry nowadays. Forms of video game business utilising cloud may include model of social usage, which can be recognised by many existing Massively Multiplayer Online Role-Playing Games (MMORPG) such as World of Warcraft 1 and Finally Fantasy XIV 2, together with games provided by social network based services. Another cloud-based video game business can refer to game contents delivery or distribution service, as Steam 3 operated by Valve Corporation4 is among the most famous platforms in this segment of video game business. Cloud Gaming is a relatively new business on trend having rapid development in term of technology and scale, and a forecast to reach 8 billion US dollars by 2017 [2] shows the evidence of

1World of Warcraft: http://us.battle.net/en/int?r=wow
2Final Fantasy XIV: http://jp.finalfantasyxiv.com/lodestone/
3Steam: http://store.steampowered.com/
4Valve Corporation: http://www.valvesoftware.com/
tremendous market potential of cloud gaming.

1.1. Concept of Cloud Gaming

Cloud Gaming provides gaming services on demand, implying high accessibility to game contents from clients. System of Cloud Gaming handles rendering, which is the most complicated part in contents processing, at the cloud servers and stream the encoded game scenes as video sequences, to the client over a network with sufficient bandwidth, such as connection speed of 5 Mbits/s is recommended by OnLive [3]. The method of streaming game contents as video sequences is called image-based streaming. Another method, which is called instruction-based streaming, delivers game contents by streaming 3D commands to the client’s side, as such the game contents are locally rendered and shown on client’s display. At the client’s side, control events from mice, keyboards, joysticks and other types of input devices are accurately recorded and transmitted from the client’s machine back to the cloud servers for corresponding manipulation of game logics. A brief structure of Cloud Gaming is shown at Figure 1.1.

The On demand nature of Cloud Gaming draws significant attention from both clients and game developers for various reasons. As for the benefits of clients, Cloud Gaming frees clients from the complication of game software installation and management of compatibility with hardwares. With the mere requirement of thin client, which is a device being able to connect to network and display the received game scenes, clients are granted with more different choices of platforms including PCs, laptops, tablets and smart phones to play games. In addition, clients could pay less costs for more game choices. In term of advantages gained by game developers, Cloud Gaming eases the possible incompatibility issues between hardwares and softwares. Therefore, developers are easier to adapt their game softwares to more different platforms, thus decreasing the production costs and increasing the net revenues. Features of Cloud Gaming envision a promising future of providing million clients with novel gaming experience, as it has been an active topics both in industries and research fields recently. OnLive [4] is a
leading Cloud Gaming service provider based in San Francisco USA providing gaming softwares from over 50 publishers. Sony’s acquisition of Cloud Gaming company Gaikai [5] in the year of 2012 shows great interests toward Cloud Gaming from leading media enterprises. GamingAnyWhere [6], developed by Chen et al, is an open-source Cloud Gaming platform available for the use of research.

1.2. Conducted Evaluations on Cloud Gaming

However, maintaining and improving quality of experience (QoE) is considered to be very challenging in Cloud Gaming services because networking constraints become majorly significant. Design of Cloud Gaming system must be able to dynamically adapt to varied network conditions in order to preserve stable quality. Previous researches performed excessive performance evaluation [6, 7] for multiple existing Cloud Gaming platforms including OnLive, StreamMyGames [8] and GamingAnyWhere, based on output frame rate and video quality metrics including PSNR and SSIM. Network emulation tool such as dummynet [9] has been applied for injecting delay, packet loss and bandwidth to simulate different network conditions. Results of evaluations show that Frame Rate Per-Second in
most platforms degrade in unstable network environment such as long network delay, high packet loss and limited bandwidth. In addition, similar degrades can be found in Graphic Quality, as it is also vulnerable to bad network connection. Among the tested platforms, Cloud Gaming service provider OnLive maintains relatively stable performance.

In another research, similar evaluation has been conducted, but from the user’s perspective on different types of games [10]. By taking QoE measurement using subjective tests, the result suggests that most players have different tolerance based on the type of the game, such as players of cloud-based First Person Shooting (FPS) games are less tolerant of network latency and packet loss than players of other game genres. The result also agrees with previous research showing that response delay as low as 100ms is required to maintain good QoE while playing FPS game. However, due to the inevitable network delay, achieving good QoE of playing FPS game on Cloud Gaming system is not very viable yet. This can also explains the lack of new FPS games being provided by existing cloud gaming company such as OnLive. Better support for games required with intensive responses is of critical consideration for designing future Cloud Gaming system.

1.3. Research Motivation

Another similar subject test being conducted finds that players are sensitive to changes in graphic quality and smoothness during gameplay [11]. This implies that graphic quality plays an important role in Cloud Gaming experience, especially for those games which rely on impressive visuals. Nowadays many game softwares, particularly for those major contenders, largely depend on realistic visual effects to keep up with their competitiveness in the market. Alongside with the rapid development of new gaming softwares, in-game visual effects tend to be more complex and require intensive calculation of hardware. In addition, players’ being able to smoothly experience their gamings at Full HD resolution (1920 x 1080) is considered to be the current standard, while such standard is
even shifting to higher resolution such as 4K (3840 x 2160). The rising demand of gaming with more realistic graphics on high resolution display not only puts burden on traditional local rendering, but even introduces more technical difficulties for Cloud Gaming.

Existing Cloud Gaming service providers OnLive deploy their powerful data centres across multiple areas for providing players with low latency services. However, in order to maintain fairness and sufficient frame rate per second at client side, an up-to-date graphic quality is sacrificed. A survey [12] being conducted on current leading cloud gaming system shows that each streaming from OnLive, which is encoded as 720p (1280 x 720) video sequence, is processed by distributed computing power approximately equivalent to a dual-core specification with Nvidia 9800GTX\(^5\) graphic card. Compared to traditional desktop gaming and current mainstream game consoles such as PS4, the obviously downgraded graphic quality from OnLive becomes marginal to match up with nowadays standard. The same survey also explores Gaikai and discovers that it provides significantly better graphic quality compared to the one from OnLive. As each game is handled by a more capable Nvidia GeForce GTX 560Ti graphic card, certain advanced graphics settings such as texture filtering and multi-sampling anti-aliasing become customisable for players as well. In addition, the CPU-dirven x264 video encoder utilised by Gaikai takes clear advantage at streaming quality over hardware encoders, which are exploited in the OnLive system. However, the improvement of graphic quality results in lower frame rate per second at client side, as it is halved compared to 60fps delivered by OnLive.

Generally, balancing the delivered graphic quality and frame rate per second at the client is a major objective of providing Cloud Gaming services. Compared to traditional gaming platform such as game consoles or PC machine, high accessibility to game contents is undoubtedly one of the main advantages of Cloud Gaming, but the delivered graphic quality is not as impressive as the traditional local rendering. In order to keep high competitiveness of cloud gaming platform, it is necessary to research for improving graphic quality while maintaining the

\(^5\)Nvidia: http://www.nvidia.com/page/home.html
original advantages.

1.4. Research goal

In this paper, a hybrid game contents streaming method is proposed aiming at improving graphic quality delivered from Cloud Gaming to PC players, and such method is based on a more practical usage scenario which will be discussed in later session. In order to maintain availability to many devices, the method is mainly developed upon an image-based streaming structure, which is a preferable solution used in most existing service providers. The concept of hybrid streaming here refers to the method constructed from both image-based streaming and instruction-based streaming, as these two terms will be explained shortly. By using the proposed method, game contents streamed from cloud server are split into two parts, as one part is streamed as video sequence. Another part, which is streamed as graphic commands, is for rendering job to be performed at the client side. The rendered game contents are then displayed accordingly with the contents streamed as video sequence. By taking the advantage of distributing rendering operation, the combined computing power is expected to produce better graphic quality. Better graphic quality here refers to improvement over image-based streaming, which the output quality is considered to be degraded from original even though it is more utilised method. Contents streamed as graphic instruction in the proposed method could prevent portion of contents as original quality, thus achieving improvement. Furthermore, since portion of processes is offloaded to client’s device for local rendering, lower server workload is mitigated as well.

At the present stage of work, a basic prototype has been constructed based on the only available open-source Cloud Gaming platform, GamingAnyWhere for the purpose of proof of concept. In addition, a custom OpenGL ⁶ demo was used for evaluating the graphic quality using the metric of PSNR and SSIM.

⁶OpenGL: https://www.opengl.org/
1.5. Organization of Thesis

The remainder of the paper is structured as follows. In Chapter 2, two important terms related Cloud Gaming are explained, together with some existing Cloud Gaming platforms and involved technologies will be briefly introduced. Chapter 3 discusses the position of our system by establishing an usage assumption. Chapter 4 explains the proposed method in details. Chapter 5 shows the implementation of our prototype, which is built upon GamingAnyWhere. Chapter 6 presents the conducted evaluations and achieved results. In Chapter 7, we will discuss the achievement of the proposed system based on the evaluations, novelty of the system and threats to validity. Finally, in Chapter 8, we conclude our works and provide an outlook for future works.
Chapter 2

Related Work

Cloud gaming system can be regarded as a type of real-time remote rendering system, as the technology is similar to application such as Remote Desktop. However, dedicated video player and encoder are usually specifically designed for the use of cloud gaming, which guarantee an environment to handle more rigid real-time response. In general, the streaming approaches of cloud gaming system can be categorised into two types, image-based streaming and instruction-based streaming. These two methods differ from each other in how the game contents are streamed from the server to the client.

2.1. Image-based streaming

In the cloud gaming system with image-based streaming approach, game logics are calculated at the server CPU and the 3D graphics renderings are processed through the dedicated Graphic Processing Unit at the server. The rendered contents are then compressed into 2D video and streamed to the client. At the client side, the received streaming of video is decoded and the corresponding game contents are shown on the client’s display. Since the decoding can be done by using low-cost decoder chips which are massively embedded in consumer’s electronics, this approach is ideal for thin client running on source-constrained devices. In addition, this approach also suits well with the concept of Cloud
Gaming, by freeing client from complicated 3D operations. Most of the existing commercial cloud gaming platforms such as OnLive, Gaikai and StreamMyGame are using this approach in their systems. OnLive is one of the leading cloud gaming service providers in the market, as it is famous for providing fair and stable streaming of game contents by widely deploying dedicated servers and highly specialised encoding hardwares and algorithms. All games provided by OnLive are streamed at the quality of 720p. Similar to Onlive, Gaikai, which is now acquired by Sony, provides cloud gaming with better and customisable graphic quality by utilising more powerful datacenter and CPU-driven encoder for the video streaming. StreamMyGame is a game streaming solution that allows players to setup their own cloud gaming environment, by which players are able to stream their own games through network. However, these commercial cloud gaming systems are closed-source with proprietary control, as such detailed techniques and algorithms could not be thoroughly explored.

On the other hand, GamingAnywhere is the first and the only available open source cloud gaming platforms, which is developed based on image-based streaming structure, as shown in Fig 2.1. For capturing the game screens rendered from the graphic processing units at the server, two capture modules are implemented in GamingAnywhere: the hooking mode and desktop streaming mode, which will be explained more specifically in later section. Moreover, GamingAnywhere is designed to be highly extensible, which allows developers to easily follow the programming interfaces of the modules and extend the capabilities of the system. The system is also portable to different environment such as Windows, Mac OS X and Linux while the client module could run on Android device. A large number of built-in audio and video codecs are supported, and customisable by simply editing a text-based configuration file to adapt the system into a demanded usage scenario. According to the evaluation done in previous research [6], GamingAnywhere outperforms the other two cloud gaming platforms including OnLive and StreamMyGame in term of response and graphic quality under a good network environment. Another advantage demonstrated by GamingAnywhere is that the system introduces less network traffic compared to the other two. However, it is
Figure 2.1: An overview of GamingAnyWhere’s image-based streaming structure

not as stable as OnLive if the environment becomes varied. Our proposed hybrid system is implemented based on GamingAnyWhere.

2.2. Instruction-based streaming

Significantly different from Image-based streaming, instruction-based streaming represents another mechanism of streaming gaming contents in cloud gaming environment. Instead of rendering the game contents using the graphic processing units at the server side, the graphics commands are intercepted, compressed, and then streamed to the clients. Soon after the arrival of the data, the clients render the game contents by its local graphic processing units based on the received commands. The brief structure of instruction-based streaming is shown in Figure 2.2.

Eisert et al implemented a low-latency instruction-based streaming [13] in the framework of Game@Large [14, 15]. Beside intercepting every API calls to OpenGL library as well as the SDL library, commands that relate to screen update
Figure 2.2: An overview of instruction-based streaming structure

such as glXSwapBuffers and SDL_GL_SwapBuffers are modified to determine if the frame is ready for display. In order to stream OpenGL commands with low latency, the memories of the buffer residing at the server’s graphic card are transmitted to the client as well, which can be cached for future use. Furthermore, since most of the games require feedbacks from the graphic card, large amount of overhead could be introduced in the network environment. In order to address this issue, the proposed system simulates the client’s local OpenGL State at the server, by which feedback could be achieved within the server. Another main feature is tokenising the graphics commands to reduce the overhead during streaming. Representing argument such as texture as short code word, or grouping multiple commands as a single token efficiently reduce the overhead. The corresponding texture data or image is also encoded as a format similar to H.264. The conducted evaluation shows that the proposed method could save up to 85% of data bit during the transmission, thus achieving low latency.

The approach of Instruction-based streaming approach requires graphic processing unit at the client side not only compatible with the streamed graphics commands such as OpenGL and Direct3D\(^1\), but also powerful enough to render

\(^1\)Microsoft DirectX: http://www.microsoft.com/ja-jp/directx/default.aspx
the game contents in real time and high quality. Since graphics unit at the server side is not used in this approach, multiple games can be executed in one PC simultaneously and it is suitable for providing services in local network environment. For example, since it does not require very powerful server to process multiple games together, instruction-based streaming is a viable option for hotel to serve its customer with gaming services. However, imposing more workload on clients makes this approach not a viable option for source-constrained devices.

2.3. Other Approaches

Beside the previous two major streaming methods, an approach of video streaming with post-rendering operation is also proposed [16]. This method is majorly designed for cloud-based mobile game usage. Compared to broadband network, wifi connection of mobile device implies much smaller bandwidth and possible longer latency. Therefore, this approach aims to enhance video encoding for cloud-based mobile gaming service by taking the advantage of the run-time graphics rendering contexts form the 3D game engine. In this method, the modified encoder selects a set of key frames in the video sequence, uses a 3D warping algorithm based on the context data from the game engine to interpolate other non-key frames. The interpolation allows encoding warping residues with much lower bit rate, while maintaining or even improving video quality.

Another proposed method aims at streaming high quality graphics to mobile devices with low latency in the in-home environment, by exploiting a distributed rendering setup at the server [17]. The interactive multimedia application is processed in the distributed environment, while the output of the contents is compressed as ETC1 coding. Since ETC1 coding is natively supported by OpenGL ES which is the main 3D graphics library in mobile device, the application contents can be conveniently represented as OpenGL ES texture on the mobile display. Due to the powerful computing offered by the distributed setup, this streaming solution is also well suited for real-time visualisation of data intensive computations, which belong to the aspect of big data and HPC.
Chapter 3

Background of Proposed Method

Considering that majority of cloud gaming services are based on the approach of image-based streaming, the goal of this research is to enhance the delivered graphics quality, by which the mechanism of instruction-based streaming is exploited. In this paper, we propose a hybrid streaming approach which distributes part of the game data into the processing of instruction-based streaming while keeping another part of contents as image-based streaming, aiming to achieve better graphic quality at the client’s side by combining outputs from these two mechanisms, and simultaneously reduce workloads at the server. The idea of this approach is from Scalable Link Interface (SLI), a multi-GPU technology developed by NVIDIA for linking two or more Graphic Cards together to increase the processing power available for a single graphic output. Certainly the transfer rate in the network environment of Cloud Gaming cannot match with the local PCI-E speed of graphic Card, but the concept of distributing 3D processing workload into different graphics processing units can be taken as reference to manipulate data in Cloud Gaming system.
Table 3.1: Comparison between Image-based streaming and Instruction-based streaming

<table>
<thead>
<tr>
<th></th>
<th>Image-Based</th>
<th>Instruction-Based</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server Workload</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>Requirement at Client</td>
<td>Low</td>
<td>High</td>
</tr>
<tr>
<td>Incurred Network Load</td>
<td>Low</td>
<td>Varied depending on the gaming software</td>
</tr>
<tr>
<td>Graphic Quality at Client’s Output</td>
<td>Degraded(Encoded Video)</td>
<td>High (Original Quality)</td>
</tr>
<tr>
<td>Practical Usage</td>
<td>Commercial Service Providers</td>
<td>Small Network Environment</td>
</tr>
</tbody>
</table>

3.1. Comparison between Image-based Streaming and Instruction-based Streaming

For traditional platforms such as game consoles or PC machines, the major criteria of judging gaming quality falls on the rendering power of the player’s chosen graphics processing unit. By measuring Frame Per Second and benchmark system such as 3D marks under different graphics setting (anti-aliasing, resolution, etc), the capability of the graphics card could be known. On the other hand, gaming provided by Cloud Gaming is in the form of on-demand service, which is usually fair and not customisable from player, the quality of service, known as QoS, becomes the main criteria for judgement. In order to establish a clear base point of our proposal, approaches of image-based streaming and instruction-based streaming are compared in the QoS area of server workload, requirement at the client, delivered graphic quality, network load and usage in reality. The comparison is shown in Table 3.1.

In the structure of image-based streaming, server side is required to process a sequence of task including manipulation of in-game logics, graphics render-
ing, data capturing and video encoding, which introduces heavier workload. As for instruction-based streaming, server simply intercepts every API calls to 3D graphics library and streams the commands to the client, so comparatively it requires lighter workload. Previous study shows that multiple different games could be executed simultaneously in the same machine under the implementation of image-based streaming.

With the image-based streaming, client is merely required with a device that is able to connect to the streaming server, receive the game contents which are represented as video sequence, decode the contents and show it on player’s display. As the decoding can be processed by low-cost decoder chips which are vastly embedded in many consumer’s electronics, the image-based streaming becomes a viable approach for client to play games on source-constrained devices. On the other hand, instruction-based streaming requires user’s device not only compatible with the 3D commands, but also powerful enough to render the contents in real-time, which significantly means a more demanding, capable equipment.

Since Cloud Gaming is running in a network environment, network load incurred by the system is another essential criteria to be considered. Previous research [6] conducted measurement on uplink and downlink traffic such as bit rate per second for multiple image-based streaming platform including GamingAnywhere and OnLive. According to the measurement, for frame rate per second at 40 - 60 fps, OnLive achieves an average of downlink bit rates at 3-5 Mbps while the open source GamingAnywhere achieves even lighter network load at around 1.5-3Mbps. The evaluation shows that by utilising efficient encoding mechanism at the server, both platforms achieve reasonable network load, which is manageable for normal broadband network environment. As for the instruction-based streaming, the network load is varied depending on the gaming software being processed. An experiment [18] performed on Games@Large, which is a cloud gaming system that is able to stream game as either image-based or instruction based, shows that instruction-based streaming requires up to 6-8Mbps for certain game running at 25fps while some games may require up to 80Mbps for running at 2fps.
Another major criteria is the delivered graphic quality. As being discussed in previous session, image-based streaming provides on-demand services with fairness, as each streaming is handled by a portion of computing power equally distributed from data centre. The rendered data is then encoded as video sequence, as the quality of the video largely depends on the encoding mechanism. OnLive delivers game contents as 720p video by utilising X.264 hardware encoder, as the output at the client side is considered to be significantly downgraded from original quality. The more capable system of Gaikai applies more powerful CPU-driven encoder, which provides users with better graphic quality at the output. Generally, when compared to traditional local rendering, the video quality from image-based streaming is acceptable, but not very surprising. Even though at the server side, powerful grid of graphic processing units are possibly utilised for producing very high-quality image, such image is eventually encoded as lossy format for streaming to the client. Standing from the position of service provider, it is reasonable because not only they need to deliver fair service to each client, but also the system is required to be adaptive to common network environment.

In contrast with the lossy output of image-based streaming, instruction-based streaming, which is based on the usage scenario that player’s device is capable of rendering the 3D contents, provides original and lossless gaming graphics at the client side. As our system is built up on an image-based streaming structure, we aim at improving graphic quality by reproducing the original quality of the image at the client side, through the mechanism of instruction-based streaming.

In term of the usage in reality, image-based streaming is the major solution for most Cloud Gaming service provider, since it is a viable option for source-constrained device and frees the player from the possible incompatibility between gaming software and hardware. Instruction-based is reported to be applied for smaller network environment such as providing gaming service at the hotel.

In this research, based on the image-based streaming approach, we aim at improving the delivered graphic quality by applying mechanism of instruction-based streaming as well.
3.2. Position of Our System

For validating the proposed hybrid streaming approach, our assumption of usage scenario is explained in this section.

Highly accessible to game contents is the core idea of Cloud Gaming, as most existing service providers emphasise on the point that high quality games can be available for playing on any device. It frees users from the complication of installing the gaming softwares and upgrading their systems for newer games. As such, the approach of image-based streaming which represents game contents as video sequence becomes the most used solution. Therefore, the requirement for user’s device is very low.

However, our assumption addresses the actual use case by standing from the position of game players. Most likely, the devices that they own usually equip with graphic processing unit such as graphic cards in PC, or specialised graphics chips such as Nvidia Targa embedded in many present mobile devices. This implies that the device is possibly not powerful, but at least more capable than just connecting to internet and decoding the video sequence. To this group of potential users of Cloud Gaming, Cloud Gaming is attractive in the way of highly accessible to game contents, but not the graphic quality which is represented as encoded video sequence.

Another stand point is based on our observation that in current Cloud Gaming service, many popular game titles being offered including Assassin’s Creed and BioShock are actually PC based. Therefore, we consider that potential players likely play games under a PC environment at the local side. However, compared to running a copy of game software at local side for smoother gameplay with better graphic quality, the advantage of Cloud Gaming becomes marginal.

Previous work showed that nowadays each US household on average owns at least one dedicated game console or PC for game playing, while the global market growth of these game systems is expected to expand from 58.7 billion in 2011 to 83 billion in 2016 [19]. Such data proves that our assumption is viable for current situation.
Therefore, increasing the value of Cloud Gaming from game player’s perspective becomes our motivation to develop this hybrid streaming model. Based on our assumption that most devices at client’s side are not totally source-constrained, our hybrid system effectively utilizes the computing power at local side to process some contents, which are offset from the server side. The local rendering at client’s side is expected to preserve the original quality of the distributed contents. While keeping the implementation completely upon the image-based streaming structure, we expect the extra advantages brought by instruction-based streaming could help to enhance graphics quality, and simultaneously reduce the workload at the server.
Chapter 4

The Approach

Since the proposed Hybrid Streaming method utilises instruction-based streaming on top of the main structure of image-based streaming, processing steps including splitting the gaming contents into these two mechanisms and arranging the respective outputs accordingly at the client’s display are critical and fundamental. In this section, we explain our approaches of applying the two major streaming mechanisms, the way of splitting the source contents and representation of the final output at the client.

4.1. Process of Image-based Streaming

Our hybrid approach is implemented upon the existing open source Cloud Gaming platform GamingAnywhere, which uses image-based streaming approach to deliver game contents. In our model, instead of having every game contents to be streamed as image-based, we offset a portion of the game contents to instruction-based streaming for local rendering, while keeping the remaining contents streamed as video sequence. The well implemented mechanism of handling video source achieves satisfying results in previous evaluations, which gives us confidence that no significant modification for the process of image-based streaming is needed in our system.

In the structure of GamingAnywhere, two types of network flows are defined,
the data flow and the control flow, as shown in Figure 4.1. Data flow is used to stream audio and video frames from the server to the client while in reverse direction, control flow is used to send user’s actions including inputs from keyboard, mouse or joysticks back to the server. In this research, we will focus on the graphics data manipulation throughout the data flow from server to client.

### 4.1.1 Capturing Game Source

After the game source is processed and rendered at the server, a designated video capturer implemented in GamingAnyWhere captures the source. Two types of video source capture modules are presented in GamingAnyWhere: the Desktop Capture mode and the API Intercept mode. Desktop Capture mode is triggered in a polling manner, which is taking a screenshot of the desktop at a specified rate. For example, if the desired frame rate is set to 30fps at the client side, the capture interval will be 1/30 seconds. In addition, a desired region of the captured screens can be extracted when necessary. As for the API Intercept module, whenever a game completes the rendering of an updated screen in the
back buffer, the module has a chance to intercept graphics drawing function calls and captures the screen directly from the back buffer. Since it processes in an opportunistic manner, a token bucket rate controller is used to decide whether the module should capture a screen in order to meet the desired fps.

4.1.2 Encoding

By using the either capturer, the captured raw data is stored in a shared buffer owned by the capturer module and shared with the encoder. The encoder, which is permitted to be read-only, encodes the raw data. Since the encoding module used in GamingAnyWhere is from libavcode library, which is part of ffmpeg\(^1\) project, any codec supported by libavcodec could be used in the system because of the library’s high extensibility.

4.1.3 Network Streaming

The encoded frame is delivered to client through network connection with either TCP or UDP protocol, which can be specified based on the preference of the user. In order to handle the possible situation of packet loss, TCP protocol is being set in our system. The encoded frame is delivered as interleaved binary data in RTSP while RTP/RTCP packets which represents the statistics information of the streaming quality are achieved by libavformat library. All these data streams through one TCP-based network connection.

At the client, only packets representing the most current encoded frame delivered from server is buffered at the decoder. For detecting whether the consecutive packet correspond to the same video frame, marker bit in each packet is examined. If the received packet has a zero marker bit, which represents that it is not the last packet associative with the current video frame, it will be appended to the buffer. Otherwise, the decoder will decode a video frame based on all the packets currently buffered, and then clear the buffer for placing packets representing the next frame. The decoded frame is shown on client’s display based

\(^1\)ffmpeg: https://www.ffmpeg.org/
on the SDL library context in original GamingAnyWhere implementation, but in order to adopt the mechanism of local rendering, we have transformed the graphics context to OpenGL based, which will be explained in later section.

Since no significant modification is needed for the mechanism of image-based streaming, we focus on the customisable feature provided by GamingAnyWhere, which we could adjust the encoding parameters that affect the video quality delivered by this streaming. The setting of the encoding parameters will be introduced in the next section.

4.2. Process of Instruction-based Streaming

Based on the established assumption in previous section, the main purpose of instruction-based streaming in our proposed method is to keep a portion of game contents as lossless, original graphics quality by utilising client’s local rendering. As for this portion of game contents, the data is represented as 3D graphics commands and streamed through the internet to the client. At the client’s end, game contents are rendered based on the received 3D commands and presented on client’s display.

4.2.1 3D Commands Intercepting

Intercepting 3D graphics commands is the fundamental step in instruction-based streaming. Instruction-based cloud gaming systems proposed in previous researches intercept all API calls to graphics libraries such as OpenGL, DirectX as well as SDL, for the reason that many gaming softwares are developed based on these libraries. Due to the complexity of supporting all graphics library, our current system only considers the case of OpenGL. In order to capture the OpenGL commands, OpenGL Stream Codec (GLS) is planned to be applied. GLS is a facility for encoding and decoding streams of 8-bit values that represent sequences of OpenGL calls invoked by the program. The encoded commands, which are represented as either human-readable text or binary, are then packe-
tised for transmission through a normal TCP connection.

4.2.2 Frequency of Capturing

Frequency of capturing the commands is also critical, since it is necessary to synchronise with the contents of image-based streaming. Compared to the mechanism of screen capturing in image-based streaming, the screen updates of running game software at local side is significantly more frequent. Considering that our system is mainly based on GamingAnywhere, the interval between each 3D commands capturing should be correspondingly matched with the two capture modes mentioned in last section. In order to approximately synchronise with either video capture modes, two most currently captured 3D commands should be buffered. Whenever the video capturing is triggered by GamingAnywhere, we could check if the captured 3D commands are complete or not. If it is complete, it will be sent to the client. Otherwise, the last captured 3D commands will be used instead. Furthermore, for notifying the client’s device that contents streamed as 3D commands or video sequence are representing the same frame, an extra marker bit will be used in packetization.

4.2.3 Process of related data

Regarding other corresponding data such as objects vertices and textures being stored in buffers, we are currently considering two approaches to handle such data. One is based on Eisert et al’s proposal, by efficiently compressing all these data with the 3D graphics commands set for streaming. However, it may introduce too much complexity in our system. Therefore, we are considering another approach, which is based on our usage assumption. Since the client’s device is not source-constrained, a copy of texture and objects data could be streamed in advance and saved at the client side from the beginning. Therefore, only the graphics commands set is necessary to be streamed to the client. Then at the client side, the rendering operation can be performed by locally loading corresponding data from the client’s device. Since only graphics commands set is delivered to the
client, this approach could significantly reduce the overhead in the transmission.

4.3. Final Product of the Hybrid Streaming

In Figure 4.2, we show the overall structure of our hybrid streaming system. Within the system, how the final representation of game contents is correspondingly formed from the products of two different streams is another important design objective, which should largely depend on the way of splitting game contents at the server. At present, we are considering to compare the depth value of each object in one frame. Based on the depth value, all the objects are separated into two groups, the upper layer which contains shallower objects and the lower layer which contains deeper objects. Considering that the contents delivered through video-based streaming result in video frame without any depth factor, the game contents represented as this form should be treated as background. For this purpose, objects belonging to lower layer are streamed as video sequence. On the other hands, objects belonging to upper layer are streamed as 3D graphic
command sets. Therefore, as soon as the rendering operation is completed at the client side, the resulted contents can be overlaid on top of the background represented by the contents streamed as video sequence. The process is indicated by Figure 4.3.

In current implementation, the demo we have created has every object strongly defined as foreground and background, which we can explicitly manipulate the depth location of each object. However, for dynamically adapt to more complicated usage, we are considering to take the centre point of each object as reference to define the depth location of corresponding object. As the object may transform from time to time through shader, this particular reference point is also explicitly calculated for acknowledging the system of the object’s depth. By knowing the depth, we could define if the object belongs to upper layer or lower layer.
Chapter 5

Implementation

In order to construct the proposed hybrid streaming model, we implement the system based on the open source image-based streaming structure of GamingAnywhere. The majority of the implementation is about the mechanism of instruction-based streaming, which includes steps of intercepting 3D graphics commands, splitting the game contents, packetising the 3D commands, streaming the packets to the client and rendering the contents at the client side. Considering that completing the whole system requires significant amount of works and subsidiary evaluations, we first created a prototype to simulate the outputs of our system because our current main goal is to evaluate the graphics quality. By simulating the expected output of our system and conducting a preliminary evaluation on the quality, we could compare the result with the image-based approach and eventually prove our concept. For this reason, instead of constructing a complete instruction-based mechanism, the current prototype implementation directly loads and renders an OpenGL object, which is pre-installed at the client side, and overlay the rendered objects on the other contents streamed as video sequence from the server side. Afterwards, based on the archived outputs, we conducted the evaluation by using the metrics of PSNR and SSIM.
5.1. Setting Image-based streaming in GamingAnywhere

We installed the GamingAnywhere pre-compiled package, which comes with both client module and server module, on two machines, as one is for server and another one is for client. The server machine is equipped with an Intel Core i7 4770K 3.5GHz CPU, a Nvidia GeForce GTX 780 graphic card and 16GB system memory, running as a 64-bit CentOS Linux system. As for the client machine, it is a MacBook Pro equipped with a 2.6 GHz Intel Core i7 CPU, a Nvidia GeForce GT 750M with 16GB system memory. After successfully installing GamingAnywhere, we had our system operate in the campus network environment, which is considered to be sufficient in our implementation.

In order to stream a game in GamingAnywhere, the platform provides a convenient way of executing either in Desktop Capture mode or API Intercept Mode (Hooking Mode) by inputting a simple command line in CLI. The command takes a configuration file as parameter, in which we specify the name of the gaming window we want to capture if using the Desktop Capture mode. Otherwise, we specify the gaming software execution file we want to intercept if using the Hooking Mode. Many other parameters such as encoding bits, rate of capturing and pixel formats can be set within the configuration file.

For achieving the best graphic quality, the hooking mode is ideally the best option since it takes more original raw data directly from the back buffer. However, the hooking-mode is not supported in a 64-bit Linux system and hence, the desktop capturing mode is the only option in current implementation. The encoder is x264, which is the default configuration come with GamingAnywhere, and the data is streamed as H.264/MPEG-4 AVC format. The following figures show the setting of our configuration files. Figure 5.1 is the basic setting of the Desktop Capturing Mode, which includes the name of the window we want to capture. Figure 5.2 shows the configuration for the encoder to be used, which is x264 and the desired frame rate at 24fps for the output video. Specific parameters such as encoding bitrate at 3000kbps can be set as well, which is shown in
5.2. Demo application for current evaluation

At current stage of the implementation, taking the use of existing game software as test case is difficult because the data in the software can be large and complicated. Instead, we decided to create our own simple demo application for the current evaluation by using graphics library OpenGL. Before creating this demo application, we established two design objectives to base on. First, this application should have clearly distinguished front objects and background objects, which allows us to easily split the contents into image-based streaming and local
rendering. Secondly, the objects are static so synchronising the contents will not affect the evaluation.

Based on these two design objectives, we created a demo application which is shown in Figure 5.4, in which one jet-fighter object is presented, and in Figure 5.5 where there are three jet-fighter objects. The main background, which includes the scene of mountain landscape, is created by applying the concept of skybox. Skybox is a method to compensate for the limited bit-depth of depth buffers, which put a limit on the amount of details that can be rendered at a distance. Usually, the skybox is a simple cube with 6 different textures placed on the faces, which perceives a illusion of real 3D world around the viewer. In addition, since skybox is always aligned to remain stationary with respect to the viewer, it gives the viewer a feeling that it is infinitely far and not reachable, while all the other objects must be at least closer than the skybox to the viewer. Because of the characteristic of skybox, it is the base element that is always classified as lower layer object, which represents the background contents at client side. Therefore, it is transferred to the client by image-based streaming.

In the foreground, there is a jet-fighter model statically showing on the screen. This jet-fighter 3D model is created by using Blender, a free and open-source tools for creating 3D graphics. 3D object created in this tool can be exported to multiple files which contain essential information such as coordinates of vertices, texture image and mapping location. For loading all the required data into our demo, we have utilised Assimp\(^1\), a Open Source Library to import 3D models formats. This library allows us to conveniently load all the necessary 3D data and save it into the vertex buffers for constructing the desired 3D object. We could also add more 3D objects into the demo with ease. In addition, the location, rotation and scale of every object can be adjusted together with the camera position and certain lighting effects.

Since the location of every 3D object could be explicitly defined in the demo, we could also easily classify if the 3D object is lower object, which is streamed from the server as video sequence, or upper object, which is rendered at the local

\(^1\)Assimp: http://assimp.sourceforge.net/
Figure 5.4: Demo with 1 jet-fighter object

Figure 5.5: Demo with 3 jet-fighter objects
Figure 5.6: Contents inside Demo being split for server side and client’s side.

For example, in Figure 5.5, there are three jet-fighter objects being shown. The two closer jet-fighters could be classified as contents rendered at the local side, while the further one could be contents streamed from server together with the skybox. Currently the classification is done manually. After classifying all the 3D objects, we directly load the 3D objects at the local side. Figure 5.6 shows the contents that we split for two streaming.

Furthermore, since the foreground object is static, the influence of synchronisation is considered to be not significant, which allows us to confidently capture frame from the demo and perform comparison with reference model in the evaluations.

5.3. Modification in Client Module

Since the instruction-based streaming has not been implemented in the current prototype, there is no modification done at the server module of GamingAnywhere, as such the image-based streaming is the only streaming mechanism presented. In our prototype implementation, most of the source code modification
was done at the client module. The main goal of the modification is to add the mechanism of rendering 3D objects, and also make the original video data streamed from the server compatible with the OpenGL context. Figure 5.8 shows the workflow of our implementation.

The original client module of GamingAnywhere applies SDL library for both taking inputs from client, and also presenting the video contents on the display. Under the Desktop Capturing mode, the video streamed from the server is encoded in a YUV420 colour space. This format is natively supported by SDL so it can be converted to SDL_texture, a representation of pixel data, directly at the client. The game contents, which are in the pixel form of SDL_texture, could be shown on client’s display by using SDL-based renderer. However, the whole progress is achieved within SDL-based context, which is not compatible with the rendering process of OpenGL.

For implementing the mechanism of rendering 3D object, it is necessary to switch every graphics-related processing to OpenGL-context based. Therefore, handling the video contents streamed from the server is different from the original way as well. In OpenGL, YUV420 is not a natively supported pixel format, but the incompatibility can be easily solved by converting YUV420 to RGB colour space through GLSL shader program. As a significant part in our implementation, we have made a GLSL shader that converts the values of Y, U, V, from the received video to the respective values of R, G, B through a simple matrix calculation. A script of our shader is shown in Figure ?? . As the video data now is in RGB colour space, we could simply map the contents as texture on a pre-made rectangular-shaped polygon, which represents the background objects of our output. By setting a suitable depth value for this polygon object, we could guarantee that background contents are always behind of the locally rendered 3D objects.

As for the mechanism of rendering 3D object, because we have changed the context to OpenGL, similar flow of implementation done in the demo can be applied in the modification. After classifying the upper layer objects in the demo for being rendered at the client, we use Assimp as well to load the corresponding
rgb = mat3 (1.164, 1.164, 1.164, 
0.000, −0.391, 2.018, 
1.596, −0.813, 0.000) * yuv;

Figure 5.7: Matrix for converting YUV420 to RGB

Figure 5.8: Flow of Processing in our client module

3D data directly from the client’s storage. In addition, in order to be consistent with the demo, the shader specified for manipulating the 3D object is the same as the one used in the demo. By knowing the parameters such as coordinate of location, angle of rotation, camera position and lighting factors, the 3D objects should be aligned exactly as the representation in the demo.

Finally, the final output of our system is represented by accordingly overlaying the 3D objects on top of the background contents streamed from the server.
Chapter 6

Evaluation and Results

The implemented prototype successfully overlays the client’s locally rendered static jet-fighter object upon the contents streamed from the server, as shown in Figure A. After successfully achieving the desired output, we conducted the main evaluation on graphics quality by using metrics of Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity (SSIM). We have evaluated these metrics on multiple situations such as loading multiple numbers of 3D objects into the demo, and also distribute different portion of objects for being rendered at client’s side. Furthermore, we also perform a simple investigation on the work load of graphic processing in different situation.

6.1. Evaluation Method for Graphics Quality

6.1.1 Metrics for Measuring Graphic Quality

Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity (SSIM) are two metrics for measuring graphic quality, as both are commonly used in previously conducted researches related to Cloud Gaming.

PSNR is used to measure the quality of reconstruction of lossy compression, which refers to the output from either image-based streaming or proposed method in this research. It measures the ratio between the signal, which is the original...
quality produced by traditional graphic processing, and the noise introduced by the compression. The higher PSNR value is the better the graphic quality it indicates. Typically the range of PSNR is between 30 and 50 dB.

SSIM is an index for measuring the similarity between two images. It refers to measuring the image quality based on an initial uncompressed image as reference. Therefore, in this research, the comparison between image-based streaming and proposed hybrid method is referencing to the original quality. The range of SSIM is from -1 to 1 while value closer to 1 indicates higher similarity between the processed image (image-based streaming or proposed hybrid method) and the original reference (produced by traditional method).

6.1.2 Process of Evaluation

In order to take the measurement of PSNR and SSIM, we used a window capturing software Screenflick\(^1\) to capture a short video at 60fps for the demo application running in image-based streaming mode and hybrid mode on our client’s machine, which is a MacBook Pro. As for the reference source that both PSNR and SSIM measurements need to base on, we capture a short video of the demo natively running on our Linux server as well. The setting of window capturing is configured to be as high quality as possible for the purpose of retaining the original quality. After acquiring a video source for each mode, we used a command of ffmpeg to extract every frame from each video source. After picking a frame from each video source of image-based streaming mode and our hybrid mode, we measured the PSNR and SSIM by comparing to the reference frame. The tool we used for measuring the metrics is MSU Video Quality Measurement Tool, which allows us to easily retrieve the values of both SSIM and PSNR.

\(^1\)Screenflick: [http://www.araelium.com/screenflick](http://www.araelium.com/screenflick)
6.2. Evaluating workload of graphic processing unit at the server

Beside the goal of improving graphics quality, the proposed hybrid model should also reduce the server workload because portion of the contents are offset to client’s side for rendering. Since we are using Nvidia graphic card, we could utilise nvidia-smi, which is a specialised monitoring tool that can show the usage of the equipped graphic card. We have investigated the usage of the graphic card under different situations of loading various numbers of 3D objects into the demo.

6.3. Evaluation Environment

The system setup has been stated in last section. As for the OpenGL demo, it is set to run at the resolution of 1280 x 720. The campus internet is considered to be stable and fast, which does not negatively affect our evaluation.

Furthermore, we have performed measurement of graphic quality respectively on our demo with multiple number of jet-fighter objects being loaded: With one jet-fighter, with two jet-fighters and with three jet-fighters.

6.4. Results

6.4.1 Measurement of PSNR and SSIM

Table 6.1 shows the results of PSNR and SSIM in the case of displaying one jet-fighter in the demo. Our proposed hybrid model, in which the jet-fighter object is rendered at the client side, achieves PSNR of 46.20 dB, compared to 43.96 dB from the traditional image-based streaming. The higher PSNR we achieve in our hybrid system implies better graphic quality over the traditional method. As for the SSIM, our system also achieves better, but not very significant result than image-based streaming.
Table 6.1: Results of PSNR and SSIM: Scenario of 1 jet-fighter on display

<table>
<thead>
<tr>
<th></th>
<th>PSNR</th>
<th>SSIM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Image-based Streaming</td>
<td>43.96</td>
<td>0.9964</td>
</tr>
<tr>
<td>Hybrid System</td>
<td>46.20</td>
<td>0.9966</td>
</tr>
</tbody>
</table>

Table 6.2: Results of PSNR and SSIM: Scenario of 2 jet-fighter on display

<table>
<thead>
<tr>
<th></th>
<th>PSNR</th>
<th>SSIM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Image-based Streaming</td>
<td>43.61</td>
<td>0.9967</td>
</tr>
<tr>
<td>Hybrid System: 1 jet-fighter locally rendered</td>
<td>44.54</td>
<td>0.9971</td>
</tr>
<tr>
<td>Hybrid System: 2 jet-fighters locally rendered</td>
<td>44.78</td>
<td>0.9972</td>
</tr>
</tbody>
</table>

The second case displays two jet-fighter objects in the demo, and we have conducted evaluations on two scenarios for our hybrid model. The first scenario is having one jet-fighter object rendered at the client side and another streamed from the server. The second scenario is to have both two jet-fighters rendered at the client side. As shown in Table 6.2, the first scenario achieves PSNR of 44.54 dB, while the second scenario achieves slightly better result at 44.78 dB. Both hybrid scenarios are better than 43.61 dB achieved by traditional image-based streaming. SSIM results from three scenarios are close, but our hybrid-model are still slightly better.

In the last case, three jet-fighter objects are displayed in the demo. Similar to the second case, we have evaluated graphic quality based on various scenarios of rendering different number of objects at the client side. As shown in Table 6.3, all the scenarios of using hybrid model achieves overall better results than image-based streaming, in terms of both PSNR and SSIM. Furthermore, in the scenarios of our hybrid system, higher PSNR was achieved by rendering more jet-fighter objects locally at the client.
Table 6.3: Results of PSNR and SSIM: Scenario of 3 jet-fighter on display

<table>
<thead>
<tr>
<th></th>
<th>PSNR</th>
<th>SSIM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Image-based Streaming</td>
<td>42.93</td>
<td>0.9965</td>
</tr>
<tr>
<td>Hybrid System: 1 jet-fighter locally rendered</td>
<td>44.94</td>
<td>0.9971</td>
</tr>
<tr>
<td>Hybrid System: 2 jet-fighters locally rendered</td>
<td>45.02</td>
<td>0.9972</td>
</tr>
<tr>
<td>Hybrid System: 3 jet-fighters locally rendered</td>
<td>45.29</td>
<td>0.9974</td>
</tr>
</tbody>
</table>

Table 6.4: Graphic Card workload at server

<table>
<thead>
<tr>
<th>Number of Objects</th>
<th>Graphic Card Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Without any jet-fighter object</td>
<td>13%</td>
</tr>
<tr>
<td>1 jet-fighters processed at server</td>
<td>14%</td>
</tr>
<tr>
<td>2 jet-fighters processed at server</td>
<td>14%</td>
</tr>
<tr>
<td>3 jet-fighters processed at server</td>
<td>15%</td>
</tr>
</tbody>
</table>

6.4.2 Usage of Graphic Card at the Server

We have also investigated the Graphic Card usage at the server, based on the three scenarios of displaying different number of jet-fighter objects in the demo. The usage was measured under the execution of the demo and also encoding the contents for streaming. The results, which are shown in Table 6.4, imply an increase of usage while loading more jet-fighter objects, but the difference is not very significant.
Chapter 7

Discussion

In this section, the findings from conducted evaluations will be investigated. In addition, we will also talk about the novelty of the proposed hybrid-streaming, and also some potential threats to validation.

7.1. Investigation on Evaluations

Evaluation on graphic quality from previous section shows that our hybrid model outperforms the traditional image-based streaming, as shown in Figure 7.1 and Figure 7.2. When measured based on PSNR, our model demonstrates a clear advantage over image-based streaming. As shown in first scenario when one jet-fighter was displayed, our model achieves up to 2.24dB better. In the case of SSIM, our model retains slightly better result than image-based streaming, but the difference is not very significant. This can be due to the reason that SSIM is more sensitive to different aspect when compared to PSNR [20].

Furthermore, the evaluation also shows that when applying our hybrid model, by having more contents rendered locally at the client side, higher PSNR value could be achieved. For example, in the scenario of three jet fighters, the achieved PSNR is 45.29dB while having all 3 jet-fighter objects locally rendered at the client, compared to 44.94 dB while having only one locally rendered. This is reasonable because rendering at the local side could help to preserve the original
Figure 7.1: Run charts of PSNR in the case of 2 jet-fighter objects and 3 jet-fighter objects

Figure 7.2: Run charts of SSIM in the case of 2 jet-fighter objects and 3 jet-fighter objects
quality of the object, compared to encoded video streaming from the server.

We have also investigated the graphic card usage at the server by loading different number of objects, but the achieved result is considered to be not very obvious. Without loading any jet-fighter object, the usage of graphic card is 13%, compared to 15% when having three jet-fighter objects loaded. The increase of graphic card usage while loading more objects is indicated, but not very significantly. The main reason is considered to be that our used demo is simple and without too many complicated effects. However, if the proposed method is applied in an environment with larger scale, the non-significant difference will be accumulated, which eventually leads to total workload saving. Further evaluation on larger environment is within our future plan.

Numerical evaluation indicates desired improvement in term of graphic quality from our hybrid model, but the difference is not very obvious by subjectively viewing the overall output, possibly due to the used simple demo. By zooming into the frame of the output, we could find that our hybrid model achieves better detailed graphics such as texture representation and clearer edges of the object. Some examples of cropped images from tested frames are shown in Appendix. As part of our future work, we will apply our system on more practical target such as open source gaming softwares, and we expect that our hybrid model will demonstrate more significant advantage over traditional image-based streaming.

7.2. Novelty of our system

The original image-based streaming method processes most of the complicated tasks at the server and stream the completed contents to the client, so it can adapt to more devices, but at the same time good graphic quality is not well achieved. On the other hand, instruction-based streaming transfers the original 3D commands to the client, which keeps impressive graphics quality but introduces too much workloads at the client. Our proposed hybrid model is novel in the way of applying distributed computing. After splitting the game contents, the hybrid model effectively utilises computing resources available both from client
and server for processing the product. Not only we could keep the advantage of availability to many devices, but also retaining good graphic quality.

A similar research conducted by Nan et al [21] has proposed a method of jointly using video and 3D graphics streaming to construct a cloud gaming framework, aiming at achieving low-bit rate transmission. In this proposed method, two stages of processing can be defined. As for the first stage, game contents are streamed as normal video streaming, but simultaneously 3D data such as geometry mesh and textures are transmitted to the client and saved at a buffer. As time goes by, more 3D data is accumulated at the client’s buffer and the frame rendered from this local buffers tends to be closer to the original data. Soon as all graphics data is received at the client, the system could shift to the second stage of only streaming 3D commands for the graphics to be rendered locally at the client’s device, thus achieving low-bit rate eventually. Comparing to this method of progressively shifting from image-based streaming to instruction-based streaming, our hybrid model utilises both streaming in parallel. The progressive method may still introduce too much workload at the client’s side, while our proposed model tries to effectively allocate computing power from both sides to retain availability and good graphics quality at the same time.

7.3. Threats to Validation

7.3.1 Is the system available to every device?

One of the most essential characteristics of Cloud Gaming is high availability to different kinds of devices. Therefore, image-based streaming, which requires only minimal tasks at the client device, is a more ideal solution to provide cloud gaming service. However, we consider that this usage scenario is conservative, as we have established an use case based on more actual scenario. We consider that nowadays, most of the clients’ devices, including those regarded as source-constrained, are at least equipped with certain basic graphic processing unit that is more capable than just decoding. This is reasonable because in order to meet
client’s rising demands for good visuals, graphic processing unit is rapidly developed and commonly embedded in most of the electronic devices such as PC, smartphones and tablets. Even though this basic graphic processing unit may not be powerful enough to process every game content smoothly, but it should be capable of processing a portion of it. Based on this point, the usage of our system is validated and also available to many common devices.

7.3.2 Is the system going to incur heavy network load

Since the mechanism of instruction-based streaming has not been implemented yet, statistical evaluation of network load incurred by the whole system cannot be conducted at current stage. However, based on previous evaluations on GamingAnywhere, the image-based streaming structure incurs reasonably low traffic load, which is manageable for most broadband network. As for the instruction-based streaming, we are planning to stream all 3D data such as meshes and textures to client’s side in advance before starting the game. Therefore, during the execution of gameplay, only 3D commands are required to stream to client. We expect that in such way, the overhead adding to the image-based streaming which we base on should be light, making our system available in most broadband network environment.

7.3.3 Is the system compatible with every game?

The implemented demo clearly defines foreground object and background object, as we could also explicitly control every elements within and split the contents based on the depth value. However, in order to generalise the usage, we need to apply real gaming software as case studies but the investigation can be challenging, considering that most gaming softwares are closed source and interactions between objects could be varied and complicated. We expect that splitting contents based on depth value is suitable for certain game genres such as fighting game, which can be represented by the famous Street Fighter\textsuperscript{1}, and side scrolling

\textsuperscript{1}Street Fighter: www.capcom.co.jp/streetfighter/
game such as Oboromuramasa\textsuperscript{2} because foreground objects and background objects are usually clearly distinguished. Currently we could not conclude if every game is compatible with our system because more investigation is necessary, but improving the mechanism of contents manipulation at the server is one main future plan.

\textsuperscript{2}www.marv.jp/special/game/wii/oboromuramasa/
Chapter 8

Conclusion & Future Work

In this thesis, we present our idea of implementing a hybrid streaming method to enhance the graphic quality offered by Cloud Gaming. This method is based on image-based streaming and instruction-based streaming, which are the two major approaches used in existing Cloud Gaming platforms. Our goal is to use this hybrid streaming to utilise graphics processing power from both client’s side and server side to eventually achieve better graphic quality. We expect that this hybrid-streaming method can help to increase the value of Cloud Gaming service from user’s perspective.

In order to validate our system, we have established an usage scenario, which we believe to reasonably conform to the actual situation. We have also implemented a simple prototype based on GamingAnyWhere for preliminary evaluation on graphics quality and server workload. The results from PSNR and SSIM show that our hybrid model achieves better graphic quality compared to image-based streaming. In addition, by processing fewer 3D objects at the server side, the server work load is reduced.

As for the future work, the most current goal is to implement the mechanism of instruction-based streaming. Afterwards, we will conduct more practical investigations by applying our method on some available open source gaming softwares. If chance is granted, we may also apply NVIDIA GRID cloud gaming technology [22], which is a toolkit to perform efficient image capture and remote
processing for the NVIDIA GPU. This toolkit allows us to make full advantage of GPU processing power to provide GPU accelerated applications and games through network to the client. We could integrate our hybrid system with this toolkit for manipulating the data more directly and efficiently.

Evaluating user’s Quality of Experience (QoE) when using our system is also critical, because ultimately Cloud Gaming service is user-centred. Subjective test involved with actual users is an essential step for Cloud Gaming developer to know about users’ standard and expectation. Designing a QoE measurement based on subjective mean opinion score of users [23] is on our future plans as well.

Cloud Gaming system is a metropolitan consisting of technologies from different fields. Chen at el recently has shared his prospects of future Cloud Gaming Development [24], from which several significant research problems spanning over a wide spectrum of different directions have been discussed: The distributed systems, video codecs, virtualisation, human-computer interaction, quality of experience, resource allocation, and dynamic adaption. Addressing these issues should help to make Cloud Gaming system more profitable, and more successful to provide high quality services to clients. Considering that Cloud Gaming is still a relatively new research field and there are lot of works needed to be done, we are dedicated to keep bringing up novel ideas and contributing to this field in years to come.
List of Publications

Technical Paper

References


Appendix

A. Cropped Images from Demo

(a) Original Quality
(b) Image-Based Streaming
(c) Hybrid Model

Figure 8.1: Demo when displaying 1 jet-fighter object
Figure 8.2: Demo when displaying 3 jet-fighter object