Action-aware Combat Model for Efficient Video Compression of Massively Multiplayer Online Role-playing Games on Cloud Gaming Platforms

by

Sardar BASIRI

THESIS PRESENTED TO ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
IN PARTIAL FULFILLMENT OF A MASTER'S DEGREE
WITH THESIS IN SOFTWARE ENGINEERING
M.A.Sc.

MONTREAL, JULY 7, 2021

ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

BOARD OF EXAMINERS

THIS THESIS HAS BEEN EVALUATED

BY THE FOLLOWING BOARD OF EXAMINERS

Mr. Kaiwen Zhang, Thesis supervisor
Department of Software and IT Engineering, École de technologie supérieure

Mr. Stéphane Coulombe, Thesis co-supervisor
Department of Software and IT Engineering, École de technologie supérieure

Mr. Sheldon Andrews, President of the Board of Examiners
Department of Software and IT Engineering, École de technologie supérieure

Mr. Carlos Vázquez, Member of the jury
Department of Software and IT Engineering, École de technologie supérieure

THIS THESIS WAS PRESENTED AND DEFENDED

IN THE PRESENCE OF A BOARD OF EXAMINERS AND THE PUBLIC

ON "JUNE 21, 2021"

AT ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

DEDICATION

I dedicate this thesis to my dear parents for their unyielding love and support from the beginning.

ACKNOWLEDGEMENTS

I would like to express my deep and sincere gratitude to my research supervisor, Prof. Kaiwen Zhang, for his guidance and support during my study and research. His exemplary up-to-dateness in research and teaching was always encouraging and helpful.

It is a genuine pleasure to express my deep and sincere thanks to my research co-supervisor, Prof. Stéphane Coulombe, for his kind suggestions, guidance, and support. His enthusiasm and help have enabled me to complete my thesis.

I would also like to thank the team members who have worked on MMOnkey framework.

Modèle de combat conscient de l'action pour des jeux de rôle en ligne massivement multijoueurs sur des plates-formes de jeux en nuage

Sardar BASIRI

RÉSUMÉ

Les jeux en nuage permettent aux utilisateurs de jouer à de nouveaux jeux de haute qualité à distance sans avoir besoin d'un appareil puissant. Les utilisateurs peuvent se connecter à distance à leurs appareils clients légers (par exemple, tablettes électroniques et téléphone intelligent) pour envoyer leur entrée de commande et, en réponse, recevoir une vidéo de la plate-forme de jeu en nuage.

Le codage vidéo et la transmission de la vidéo compressée sur Internet sont les principaux facteurs affectant la latence de bout en bout à partir de l'entrée de l'utilisateur jusqu'à la réception de la vidéo résultante. Cette latence doit être maintenue minimale afin d'améliorer la réactivité du jeu et la qualité de l'expérience du joueur. L'autre facteur important affectant l'expérience du joueur est la qualité visuelle du flux vidéo. Étant donné qu'une meilleure qualité vidéo nécessite un plus grand débit binaire pour le joueur, le défi abordé dans ce mémoire est de maximiser la qualité du flux vidéo avec le même débit binaire vidéo ou, de manière équivalente, de réduire le débit binaire vidéo pour la même qualité vidéo.

Notre objectif principal est étudié dans le contexte du jeu de rôle en ligne massivement multijoueur fonctionnant sur des platesformes de jeux en nuage. Les jeux de rôles en ligne massivement multijoueurs sont le genre de jeux en ligne où un grand nombre de joueurs peuvent interagir dans un vaste environnement virtuel. La zone la plus bondée est la zone de combat, où les joueurs utilisent différentes armes et compétences associées à leurs personnages de jeu pour interagir les uns avec les autres.

Nous présentons un modèle de combat conscient de l'action pour des jeux de rôle en ligne massivement multijoueurs sur les plates-formes de jeux en nuage. Le modèle détermine l'importance de chaque objet dans chaque état du jeu, en tenant compte des propriétés des diverses actions qui pourraient être effectuées par n'importe quel objet dans la zone de combat. En fonction de l'importance de chaque objet dans chaque état du jeu, le modèle décide à quelle fréquence la position de l'objet dans la scène doit être mise à jour. En réduisant la fluidité du mouvement des objets en fonction de leur importance pour le joueur dans chaque état du jeu, moins de bits sont requis par l'encodeur pour coder les images vidéo avec la même qualité visuelle que l'approche traditionnelle.

Le modèle peut être utilisé avec la plupart des jeux de rôle en ligne massivement multijoueurs afin d'améliorer la qualité de la vidéo compressée en utilisant la même quantité ou moins de bits. L'avantage de notre approche est que le codeur-décodeur (CODEC) utilisé pour générer le flux vidéo du jeu en nuage est considéré comme une boîte noire. Aucun changement n'est appliqué à l'encodeur lui-même ou aux scripts ni aux entrées de l'encodeur, ce qui améliore la portabilité et la compatibilité de notre solution avec les plates-formes de jeux en nuage.

Une large sélection de scénarios de test a été considérée pour l'évaluation et pour montrer les excellentes performances de notre modèle. Nos résultats expérimentaux indiquent que pour la même qualité visuelle que l'approche traditionnelle, Action-awaRe COmbat moDEl (ARCODE) réduit le débit binaire vidéo entre 10% et 41%.

Mots-clés: jeux en nuage, jeux multijoueurs en nuage, jeux mobiles en nuage, jeux en ligne massivement multijoueurs, jeux de rôle en ligne massivement multijoueurs

Action-aware Combat Model for Efficient Video Compression of Massively Multiplayer Online Role-playing Games on Cloud Gaming Platforms

Sardar BASIRI

ABSTRACT

Cloud gaming allows users to play new, high-quality games remotely without needing a powerful device. Users can connect remotely with their thin-client devices (e.g., tablets and smartphones) to send their control input and, in response, to receive a video from the cloud gaming platform.

The video encoding and the transmission of the compressed video over the Internet are the main factors affecting the end-to-end latency, starting from the user's input to the reception of the resulting video. This latency should be kept minimal in order to improve the game's responsiveness and the quality of experience (QoE) of the player. The other important factor affecting the player's experience is the quality of the video stream. Since better video quality requires more bandwidth (controlled by the bit rate) from the player, the challenge addressed in this thesis is to maximize the quality of the video stream with the same video bit rate or, equivalently, reducing the video bit rate for the same video quality.

Our main focus is on massively multiplayer online role-playing games (MMORPGs) running on cloud gaming platforms. MMORPGs are the genre of online games where a huge number of players can interact in a vast virtual environment. The most crowded area is the combat area, where players use different weapons and skills associated with their game characters to interact with each other.

We present an ARCODE for MMORPGs on cloud gaming platforms. The model determines the importance of each object in each state of the game, considering the properties of various actions that could be performed by any object in the combat area. Based on the importance of each object in each state of the game, the model decides how frequently the position of the object in the scene should be updated. By reducing the smoothness of movement of objects based on their significance to the player in each state of the game, fewer bits are required by the encoder to code the video frames with the same visual quality as the traditional approach.

The model can be used with most MMORPGs in order to improve the quality of the compressed video when using the same amount or fewer bits. The advantage of our approach is that the coder-decoder (CODEC) used for the cloud gaming stream is considered as a black box. No change is applied to the encoder itself or to the scripts and the inputs of the encoder, which improves the portability and compatibility of our solution with established cloud gaming platforms.

A wide selection of test scenarios was considered for the evaluation and to show the excellent performance of our model. Our experimental results indicate that for the same visual quality as the traditional approach, ARCODE reduces the video bit rate between 10% and 41%.

Keywords: cloud gaming, multiplayer cloud gaming, cloud mobile gaming, massively multiplayer online games, massively multiplayer online role-playing games, MMOG, MMORPG

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

LIST OF ABBREVIATIONS AND ACRONYMS

AI	Artificial Intelligence
AoE	Area of Effect
AoI	Area of Interest
ARCODE	Action awaRe COmbat moDEl
AV	AudioVisual
AVC	Advanced Video Coding
BDBR	Bjontegaard Delta bit rate
CMG	Cloud mobile Gaming
CODEC	Coder-Decoder
CPU	Central Processing Unit
CSAIM	Combat State Aware Interest Management
CTU	Coding Tree Unit
FEC	Forward Error Correction
FPS	Frames Per Second
GPU	Graphics Processing Unit
HDR	High Dynamic Range
HEVC	High Efficiency Video Coding
IM	Interest Management

MB	Macroblock
MCG	Multiplayer Cloud Gaming
MMOG	Massively Multiplayer Online Game
MMORPG	Massively Multiplayer Online Role-Playing Game
MSE	Mean Squared Error
MV	Motion Vector
NPC	Non-Playable Character
NVENC	NVIDIA Video ENCoding
P2P	Peer-to-Peer
PC	Personal Computer
PSNR	Peak Signal-to-Noise Ratio
QoE	Quality of Experience
QP	Quantization Parameter
ROI	Region of Interest
RPG	Role Playing Game
RTS	Real-Time Strategy
RTSP	Real Time Streaming Protocol
SSIM	Structural Similarity Index
UI	User Interface

VM	Virtual Machine
VQM	Video Quality Metric

INTRODUCTION

Cloud gaming (Shea, Liu, Ngai & Cui, 2013), also known as gaming on-demand or gaming-as-a-service, allows users to play high-quality games remotely without the necessity of having a powerful device to run the game. The first major cloud gaming emerged with the announcement of OnLive (Mangalindan, 2020), which has officially launched in 2010. The most recent cloud gaming platform was announced by Amazon in 2020 as Amazon Luna (Businesswire, 2020). Cloud gaming obtains more and more attention, and this is proven by the fact that the major companies in the game industry have launched their own cloud gaming services. PlayStation Now announced by Sony (Hollister, 2014), Xbox game pass cloud gaming announced by Microsoft (Choudhry, 2018), GeForce Now[1] announced by Nvidia, and Stadia announced by Google (Hollister & Statt, 2019) are examples of cloud gaming platforms.

Many people see cloud gaming as the future of gaming. New technologies have fundamentally changed the experience of playing computer games. In essence, streaming video games is similar to streaming TV shows or music. However, the major difference from these other non-interactive forms of media is that user inputs (gamepads, keyboard, mouse, ...) are constantly sent from the user to the cloud gaming platform over the Internet. In return, audiovisual (AV) data is sent back to the player from the cloud gaming platform. The only requirements are to have a device (e.g., mobile terminal, tablet) and Internet connection to send the inputs and receive the AV data.

The significant benefits of cloud gaming are presented in Cai, Shea, Huang, Chen, Liu, Leung & Hsu (2016). The advantages for gamers are the availability of the service on different devices with a wide range of games available to be played at any time. Gamers can play games remotely by connecting to the Internet without downloading the game on their devices locally. Moreover, they can play different games on a broader range of devices with limited hardware

[1] https://www.nvidia.com/en-us/geforce-now/

requirements. Cloud gaming can offer features such as client migration during game sessions across different computers, and also allows players to share the game replays with others.

From the perspective of game developers, cloud gaming provides the vital advantage of narrowing development down to a single (cloud) platform. If the game consoles and personal computers (PCs) are entirely replaced by cloud gaming platforms for playing video games, cross-platform design is no longer a concern. Game development companies can easily focus on designing the game itself, and accordingly, it can significantly reduce the time and the costs of making and testing games. In addition, they can make more profit as they do not need retailers, and also, more gamers can play their games.

With all the benefits that this technology brings, it also imposes limitations on gamers. One of the most important limitations is the necessity of having a reliable Internet connection to be able to play games. However, since having an Internet connection is mandatory for online games, cloud gaming is naturally a suitable platform to play online games as it does not impose any additional requirements beyond those required to play such games.

In this thesis, we focus on massively multiplayer online role-playing games (MMORPGs) on cloud gaming platforms. The emergence of this popular genre of online games occurs with the announcement of Ultima Online in 1997 (Zhang, 2010). World of Warcraft and Final Fantasy XIV: A Realm Reborn are two examples of the most popular MMORPGs with 2,540,956 and 1,544,458 active players in 2020, respectively[2]. The Elder Scrolls Online is one example of MMORPG with 578,074 active players in 2020[2] which is officially released on Stadia in June, 2020[3]. Figure 0.1[4] shows a screenshot from the combat area of the Elder Scrolls Online.

---

[2] https://mmo-population.com/top/2020

[3] https://www.elderscrollsonline.com/en-us/news/post/58303

[4] https://www.ign.com/wikis/elder-scrolls-online/PvP

Figure 0.1 Screenshot from the combat area of the Elder Scrolls Online

In MMORPGs, many players interact in a persistent game world. Each player controls an avatar to interact with other players in an immense virtual environment. MMORPGs have their unique aspects that need to be considered and satisfied by the game companies to offer players a fair and enjoyable gaming experience. They need to be consistent, scalable, and cheat-resistant with acceptable performance to be valuable to players. A cloud gaming platform with a good infrastructure can meet these challenges. In Section 1.2.3, two different network architectures commonly used for massively multiplayer online games (MMOGs) and MMORPGs with their pros and cons are briefly presented.

Motivation:

The encoder is a major part of the cloud gaming platform. The task of the video encoder is to convert the raw video content (comprised of pixels) into a compressed format allowing its efficient transfer over the Internet. H.264/MPEG-4 advanced video coding (AVC) (ITU-T, 2003), and H.265/high efficiency video coding (HEVC) (ITU-T, 2013) are two examples of video

compression standards. Video games usually have complex scenes with many objects and many motions within the scene. This leads to high bandwidth requirements to provide the number of bits needed by the encoder to encode the video with good perceptual quality. Therefore, players need to have a good Internet connection with high bandwidth for a good quality of experience (QoE). The visual quality of the video will decline significantly if the bit rate budget is low, which is a scenario that commonly takes place in mobile devices due to the lack of a good and reliable Internet connection. The motivation is to provide a better visual quality for the users with bandwidth constraints imposed by low-speed Internet connections.

Overview:

Since video compression removes redundancies in successive frames which allows to only transmit the differences, the more changes in a video, the more bits are needed to code the video at a certain quality. In videos with higher and less predictable motions in consecutive frames, more bits are needed to code the video frames, and the encoder must sacrifice quality not to exceed a determined bit rate budget.

In this thesis, we present an Action-awaRe COmbat moDEl (ARCODE) for MMORPGs running on cloud gaming platforms. The model is aware of the game semantics and considers the importance of each object by taking into account different action data. The position update intervals are adjusted considering the importance of each object in each state of the game. Reducing the fluidity of the movement of the objects in the scene by reducing the frequency of their visual updates based on their importance, fewer bits can be allocated to code the video frames with higher visual quality. We present how using our proposed model improves the visual quality of the video stream, for each player, with comparable bandwidth.

Contributions:

The main contributions of this thesis are:

- Proposing a model to reduce the frequency of motion updates of in-game objects considering their significance to a player in the combat area of MMORPGs running on cloud gaming platforms, in order to increase the visual quality of the transmitted video to players with bandwidth constraints.

- A detailed definition of a model for MMORPGs where the importance of in-game objects to each player is determined considering the state and actions of all objects in the scene to adjust the position update rates of each object. To the best of our knowledge, our model is the first model that considers different object types and their actions in each game state to determine their significance to the player.

- Integration of our proposed model into MMOnkey (Wang, Zhang & Jacobsen, 2017), the adopted framework for MMORPGs and integration of the framework comprises our proposed model into a game stream protocol, using Moonlight[5] and the GeForce Experience software [6] to evaluate the performance of our model.

The remainder of this thesis is organized as follows. In Chapter 1, we present the background on cloud gaming, MMORPGs, and the adopted framework for this research. Chapter 2 is devoted to some of the related works performed in cloud gaming, followed by the differences between our approach and the state-of-the-art methods. In Chapter 3, we provide with details the different aspects and factors of our model. In Chapter 4, we present our experimental setup and performance methodology. We then present and discuss the experimental results and finally conclude the work.

---

[5] https://moonlight-stream.org/

[6] https://www.nvidia.com/en-us/geforce/geforce-experience/

CHAPTER 1

BACKGROUND

In this chapter, we present background information related to our research. The first section is dedicated to cloud gaming. It is followed by the explanation of different aspects of MMORPGs in the second section. Finally, in the third section, we end this chapter with an in-depth explanation of the adopted MMORPG research framework, MMOnkey.

1.1 Cloud Gaming

Cloud gaming offers a new way of playing computer games without having a powerful system. The game is executed on cloud servers, and players do not need to install games on their PCs or to download them on their consoles. They can simply connect to a cloud gaming platform over the Internet, using their thin-client devices such as their smartphone, and play a variety of games remotely. Cloud gaming takes the inputs sent from the thin-client device and sends a video sequence back to the player. In the following section, we present the architecture and framework of the cloud gaming platform.

1.1.1 Architecture and framework

As depicted in Figure 1.1, in cloud gaming, the inputs from the player are sent using a thin-client device to the cloud gaming platform. After the changes are made in the game world, considering the inputs, the player receives AV data sent from the cloud gaming platform. There exist some steps in between from when the player sends the inputs and the time the cloud gaming platform transmits AV data back to the player.

Figure 1.2 (Shea *et al.*, 2013) illustrates the framework of a cloud gaming platform. After the cloud gaming platform collects the inputs sent from the thin-client device and the actions are processed, they should be applied to the game world, and the updated scene should be rendered by a graphics processing unit (GPU). The rendered scene will then be compressed by the video

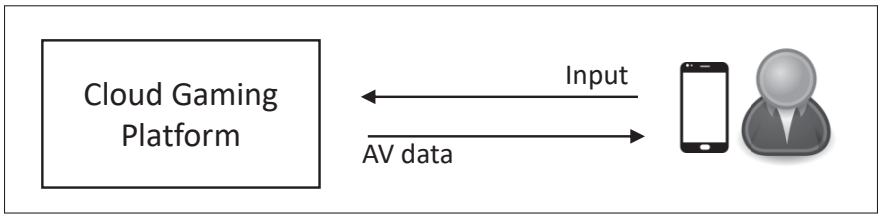

Figure 1.1 Cloud gaming architecture

encoder to make it suitable for transfer over the Internet, considering the bandwidth limitation. After the video compression step is completed, the video will be streamed to the user. The user receives the video on the thin-client device, where it should be decoded to be displayed on the screen of the device.

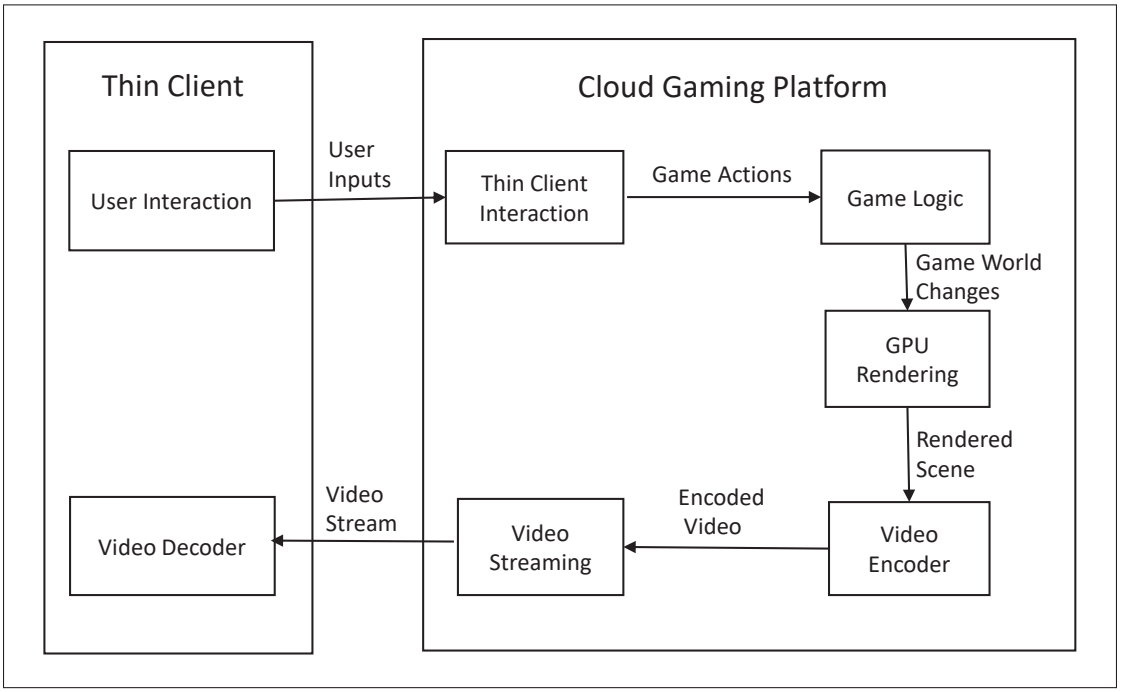

Figure 1.2 Framework of a cloud gaming platform

1.1.2 Multiplayer cloud gaming

As illustrated in Figure 1.3 (Deng, Li, Seet, Tang & Cai, 2018), in multiplayer cloud gaming (MCG), each player connects to a rendering server that acts as the client and can be seen as a representation of the player on the cloud system. Each rendering server hosts an instance of the game application for the connected player and processes the graphics and logic of the game. All the rendering servers are connected to a remote game server that acts similar to the traditional server in a client-server architecture (Figure 1.6). The remote server maintains the game session and manages the consistency of the game states among multiple connected clients. Thus, the rendering servers are servers from the perspective of the players, and they are clients of the remote game server.

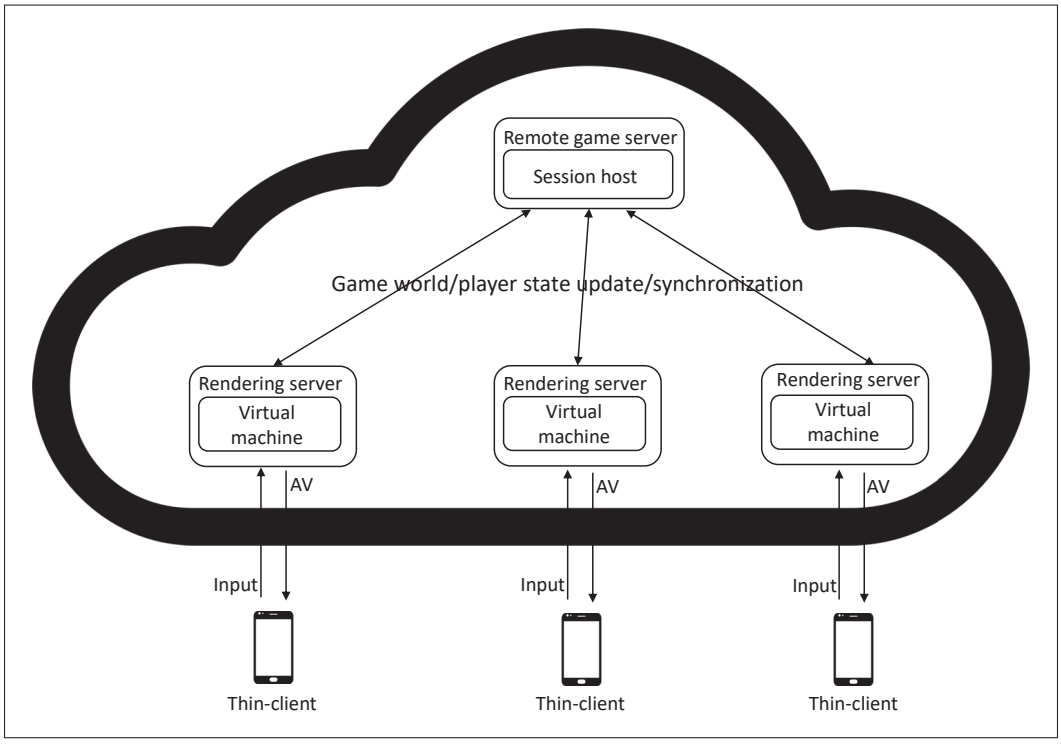

Figure 1.3 Typical MCG architecture

1.1.3 Video encoding

The encoder is doing the process of compressing AV content in the cloud gaming platform. Each frame of a video is processed into units of a macroblock (MB) or coding tree unit (CTU) on which a prediction is formed. This prediction can be performed by taking into account either the information of the current frame where the coding is done within the frame, or it could be made from the information of the previously coded frames. The former is referred as *Intra coding* while the latter is referred as *Inter coding*. Both predictions are based on the pixels that have already been coded.

As depicted in Figure 1.4[7], in Intra prediction, the encoder is coding the frame, block by block, line by line from left to right and top to bottom. For a block in a frame, we can use pixels from already coded blocks to make the prediction. Since pixels in a video are highly correlated, the prediction process is very efficient.

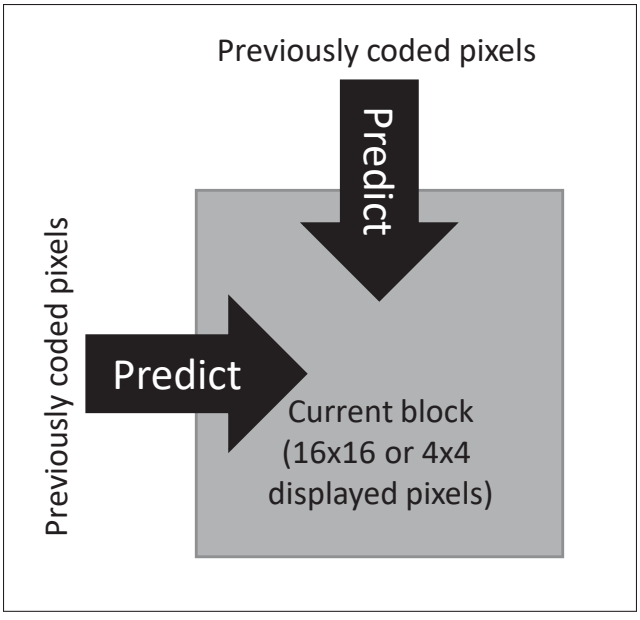

Figure 1.4 Intra prediction. The previously coded pixels of a block in a frame are used to make the prediction

[7] https://www.vcodex.com/an-overview-of-h264-advanced-video-coding/

Inter prediction is performed between the frames of a video (Figure1.5[7]), and exploits the fact that blocks of pixels in a frame to encode are frequently present in a past frame and their displacement (motion vector (MV)) permits to construct a highly efficient prediction.

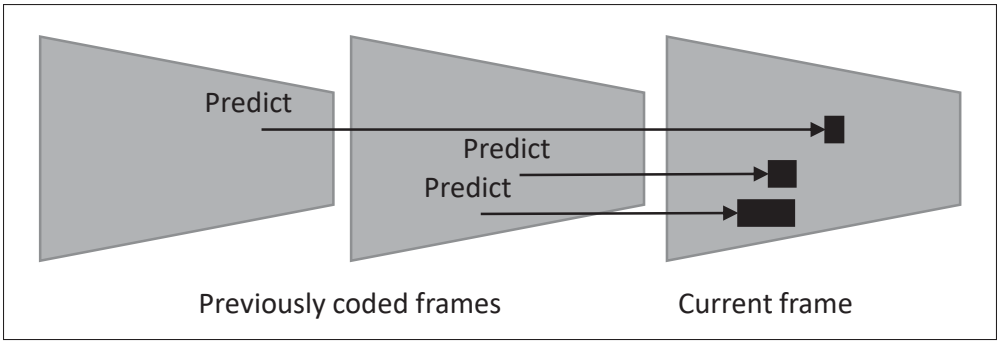

Figure 1.5 Inter prediction - The blocks of the current frame are predicted from the blocks of the past frames

The compression is applied individually to each frame of a video. Nonetheless, in Intra or Inter coding, we only send the prediction model parameters (e.g. motion parameters) and prediction error (the difference) between the information to code and its prediction, making the process very efficient particularly for inter coding. Furthermore, when the video does not exhibit significant changes, the number of bits to signal them is very small. The bit rate depends on the amount of video changes and how efficiently the video encoder can signal them for the desired video quality. For instance, HEVC (ITU-T, 2013) comprises more efficient coding tools to perform this task than H.264 (ITU-T, 2003).

1.1.4 Cloud gaming challenges

Two of the most critical challenges identified for cloud gaming are the interaction delay and the image quality (Shea *et al.*, 2013). These challenges should be addressed by cloud gaming platforms to provide the best possible QoE.

- Interaction delay: In Claypool & Claypool (2006), the authors present different threshold for the tolerable latency for different genres of online games. The maximum threshold for

first-person games is 100 milliseconds. In fast-paced first-person shooter games, the latency should be much lower in comparison to other genres of online games, since who first pulled the trigger decides who will win the game.

For third-person role playing games (RPGs), MMOGs, and MMORPGs, the maximum threshold is 500 milliseconds. In these genres of games, players do not expect to see the result of the actions executed by their avatar instantaneously. The higher delay in these genres than the first-person shooter games does not affect who will win and who will lose.

In the games with the omnipresent view (top-down view of many controllable characters by each player), like real-time strategy (RTS) games, the threshold for delays could be up to 1000 milliseconds. Here, the commands of players for different groups of characters in order to do something in the game can take seconds to minutes to complete. Therefore the sensitivity is low, and the threshold can be much higher compared to other genres of games.

As discussed earlier, the end-to-end latency in cloud gaming is from the time the players send their inputs until they receive the AV data from the cloud gaming platform. Although the presented threshold of delay for online games is based on traditional gaming systems; nevertheless, the latency should be set in a way that the QoE of players does not degrade in cloud gaming platforms. The difference in cloud gaming is that the interaction delay is also a challenge for single-player games since cloud gaming adds overhead which must be minimized to maintain good QoE.

- Image quality: The process of compressing the video and delivering it to the user should be done with acceptable quality. The higher the video quality is, the more bandwidth is required by players. This can be an issue for players with lower bandwidth, especially for those who want to play the games on their smartphones. In cloud mobile gaming (CMG) (Wang & Dey, 2009), connecting to a high-speed Internet is not always possible, and the quality of the video will suffer.

1.2 Massively Multiplayer Online Role-playing Games

MMORPGs are networked games with a massive virtual game world and a large number of players using their avatar to interact with each other in the virtual environment. There exist different kinds of objects in the game. Each player controls a character known as an avatar. Non-playable characters (NPCs) are controlled by artificial intelligence (AI). Based on their AI behaviour, these characters can interact with other objects in the game. Non-interactive objects (e.g., trees, stones, buildings) are considered obstacles and contribute to making the game scene more alive and realistic. They should be taken into account when implementing the path-finding in AI behaviour.

The *actions* in the game are defined as any kind of interactions the objects have in the game. Actions can be simple (e.g., movements, picking up or dropping different objects), or they can be complex (e.g., using skills in the combat area).

Two types of weapons are commonly used in many MMORPGs, *melee weapons* for short-range attacks and *long-range weapons* for attacking the distance targets. Axe and bow are examples of a melee weapon and a long-range weapon, respectively.

Besides the weapons, players can usually use different *skills* associated with their avatar. The effect of some skills can only be applied to the source of the action. For instance, *dash* is a skill applied to the action source to increase its speed for a limited time. Some skills, for instance, a fireball skill, are usually cast from the action source to a specific target, and their effect is applied to the target. The other common skills are the area of effect (AoE) skills. The AoE skills can be used to affect not only one object at a time but several objects in a specific area. The orison of healing and firestorm are two examples of AoE skills. Orison of healing can be used to heal other friendly characters within its target area. The firestorm skill is used to damage any opponent character within its target area.

In many MMORPGs, performing some actions, especially using skills during the combat, have a cooldown for a specific period of time. It means that when an action is performed, a considered amount of time should pass for the action to be available again.

In the following subsections, we present several aspects of MMORPGs. We first present their challenges. We then explain various concepts: replicas in online games, network architecture, interest management technique, and area of effect (AoE) mechanism (Heger, Schiele, Süselbeck & Becker, 2009). Finally, we give a brief overview of combat state aware interest management (CSAIM) (Wang *et al.*, 2017). This model is implemented by taking into account the concepts such as interest management and AoE.

1.2.1 MMORPGs challenges

In this subsection, we first enumerate and then explain various challenges of MMORPGs. The following challenges should be addressed to offer a fair and enjoyable gameplay experience to players:

- Scalability: A large number of players can interact in a persistent virtual world.
- Cheat resistance: The game should be able to provide a fair gameplay experience to players.
- Consistency: All the in game performed actions should be reliable, mentioning the game semantics.
- Performance: All the state changes in the game should be delivered to players at the right time.

As the name suggests, MMORPGs are a genre of games that allows a massive number of players to play and experience the game world together. Scalability is a major aspect of these games, and the network architecture should be able to handle a large number of players. The more players in the game, the more bandwidth is required to propagate the updates.

These games prepare a competitive environment for players to interact and compete with each other. Players expect the game to grant a fair gameplay experience where the only factor that determines who wins and who loses is the skills of the players. Thus, the network architecture should be cheat-resistant to prevent cheaters from doing malicious acts in the game.

Consistency and performance are other concerns in online games. Usually, for better performance, we should sacrifice consistency and vice versa. Most of the time, consistency is being neglected to have better performance. Nonetheless, it is essential to control consistency. To better understand the importance of consistency, imagine two players, Bob and Alice, in a game state where they can interact with each other through their avatars in the game. Bob sees Alice's avatar and wants to shoot it. He aims at the target and shoots. He expects to kill or damage the health of Alice's avatar. The action is missed since Bob has a stale view of the game state and the position of Alice's avatar on his machine is inconsistent with its real position on the server. The example illustrates the importance of addressing consistency issues in the game.

Controlling the consistency increases the latency and reduces the performance since the server should control every state update, and also, the order of the updates should be respected. Here, the main point that needs to be considered is the trade-off between consistency and performance and how to do it in a manner that does not negatively affect the experience of the player. In order to choose the right trade-off between consistency and performance, Zhang & Kemme (2011) proposed allocating different consistency levels based on the importance of each action in the game. Therefore, in each state of the game, high consistency levels are allocated to the essential objects and low consistency levels to the less important ones.

1.2.2 Replicas in online games

The main difference between online games and single-player games is that online games suffer from network latencies, whereas single-player games are playable locally with no additional network latency. In online games, two different copies of each object are considered: the primary or master copy and the secondary copies, also known as replicas (Yahyavi & Kemme, 2013).

Players can decide to change the state of each object in the game. For that reason, first, they need to send a request to the server by making changes on replicas that are stored on their local machines, and then the server will decide if they are authorized to change the state of the objects they want. If they were eligible to do so, the update will be performed on the primary copy of the objects, and then all other players will be notified that the state of the game has changed. Changing the state of an object can refer to many things, such as picking up or dropping an object or affecting the health of an object. The distribution of updates is done in a way similar to publish/subscribe systems. In Cañas, Zhang, Kemme, Kienzle & Jacobsen (2014), different publish/subscribe designs are presented for multiplayer games.

1.2.3 Network architecture

Client-server and peer-to-peer (P2P) are two main architectures for online games (Yahyavi & Kemme, 2013). Even after the announcement of cloud platforms for games, these two architectures are the most utilized architectural paradigms for online games since, so far, the main focus of cloud platforms is on single-player games. Each of these two architectures has its pros and cons.

In a client-server architecture such as the one illustrated in Figure 1.6, the game world state and all the master copies are stored on the server, and each client must connect to the server to play the game. Every interaction made by each player is sent to the server, and the state of the game is updated based on the interactions and then propagated to the other players.

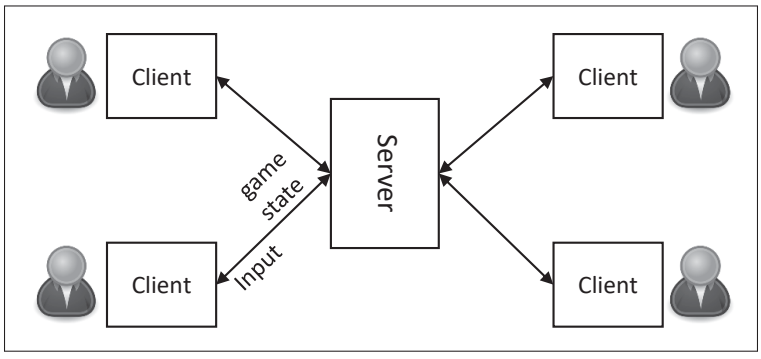

Figure 1.6 Client-server architecture

In a P2P architecture such as the one illustrated in Figure 1.7, no central server exists to hold the master copies and being responsible for the game state. Instead, each client machine (node) maintains a part of the game world. Each node stores some of the master copies and the replicas of other objects. Therefore, each client machine partly acts as a server and sends the updates to the other machines holding the replicas. Scalability is the major advantage of these architectures since more connected clients to the network mean more nodes to be used to disseminate the updates. Low cost is another advantage of P2P architectures since the game companies do not need to spend much money on the server part for providing and maintaining the infrastructure.

However, considering the fact that the world and the master copies are on the clients' machines, cheating is common and easy under these architectures. This major issue forces game companies to prefer the use of the client-server architecture to offer a fair experience to their customers. In Yahyavi & Kemme (2013), the authors present a thorough explanation of P2P architectures for MMOGs. In Yahyavi, Huguenin, Gascon-Samson, Kienzle & Kemme (2013), a scalable cheat-resistance support for distributed MMOGs is proposed.

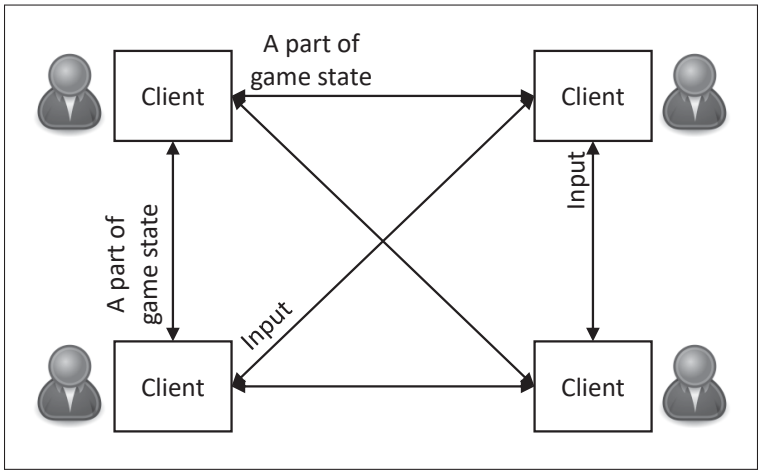

Figure 1.7 Peer-to-peer architecture

1.2.4 Interest management

Interest management (IM) is a popular technique used in many online games to increase scalability (Boulanger, Kienzle & Verbrugge, 2006). On an abstract level, IM is working like a publish/subscribe system. As the name suggests, we have publishers publishing events and then the subscribers subscribing to those events.

In online games, we have a sequence of events. Objects can be both publishers and subscribers. Everything that happens in the game world can be seen as an event. These events are constantly changing. IM can decide when an object should subscribe to a published event to receive the updates. IM benefits from the limited awareness of players about the virtual environment of the game. In each state of the game, players focus on the objects that they can interact with, and most of the time, those are the objects in the vicinity of their avatar. Therefore, in each state, each player is aware of a small portion of the game world, known as the interest area or the area of interest (AoI). The player only receives the updates from the objects within the AoI. This technique significantly reduces the number of updates exchanged by players, causing a considerable reduction in bandwidth demand and making the game more scalable.

One of the most popular IM is the space-based interest management, which is based on proximity and is presentable by referring to the aura-nimbus model (Benford & Fahlén, 1993). The aura is the area around an object. Nimbus or the AoI is the area with the player's avatar in the center. When the nimbus of the player's avatar intersects the aura of an object, the player will be aware of the presence of the object in the space.

It is common to partition the game world into smaller regions to help the spaced-based interest management and reduce the computational cost of the pure aura-nimbus model. This technique is known as region-based interest management. Here, the game world is divided into many small zones. Whenever the boundary of the area around the player's avatar overlaps with the boundary of these regions, the player will receive updates from the objects within these regions. Figure 1.8 illustrates the region-based interest management and how zone division and the area around the player can determine the AoI of the player. As shown, the game world is divided into equal

size grids. The objects in the orange zones are visible to the player, and the objects in the blue zones are not. The square shape area around the avatar determines which zones are interesting and visible to the player. Each sub-figure illustrates how the area around the avatar is moving as the avatar moves and makes new zones visible to the player by subscribing to new zones and unsubscribing from old zones, which makes them invisible to the player. The game world can be divided into different region shapes (e.g., squares, hexagons, triangles). In Boulanger *et al.* (2006), different game world divisions and different interest management techniques are thoroughly compared.

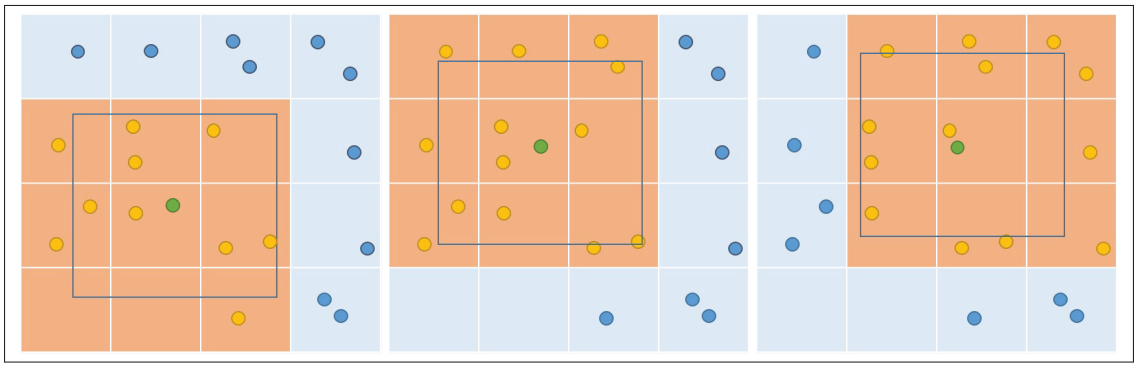

Figure 1.8 Region-based interest management specifies the portion of the game world that is interesting to the player (the orange grids)

1.2.5 Area of effect

AoE proposed in Heger *et al.* (2009), is a mechanism that can be built on top of the space-based interest management. It specifies an area, within the space-based interest area, where the objects it contains can be affected by an action. We can associate an AoE for each action in the game. The shape, size, and the lifetime of the AoE is based on the characteristics of each action. While the AoI is defined as the area interesting to the player, the AoE is an area within which a specific action changes the state of some objects. Figure 1.9 illustrates a circle-shape AoE around the avatar.

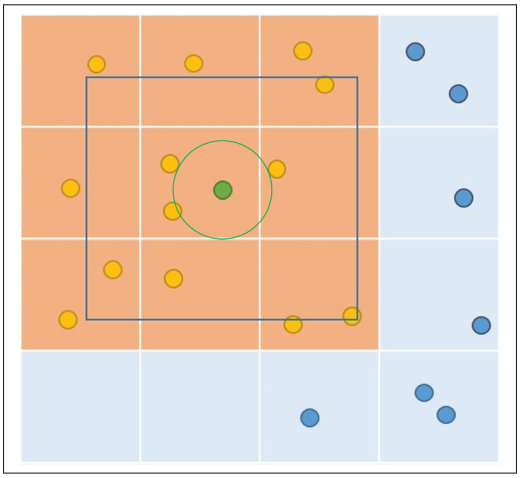

Figure 1.9 Area of effect specifies an area in which the objects within can be affected by an action

1.2.6 Combat state aware interest management

In the model proposed in Wang *et al.* (2017), the main focus is on performing the IM in the combat area of the MMORPGs. CSAIM reduces the number of transmitted updates to players during combats. Depending on the state of the game, the position updates will be adjusted. Players receive updates from the objects within the AoI of their avatar. The AoI size is considered as the size of the screen. The model uses two fixed frequency updates within the AoI. The client receives high-frequency position updates from the objects within the AoE around the player's avatar, and low-frequency position updates are considered for the objects outside the AoE. The AoE size is decided considering the maximum range of actions the player can perform in the virtual environment, which forms a circle-shape area around the player's avatar. The model can reduce the network load of the client-server architectures in MMORPGs.

1.3 MMOnkey

In this section, we describe MMOnkey (Wang *et al.*, 2017) because it is the MMORPG research framework that we have adopted for our research. It is developed in C#[8]. It is designed based on a single server architecture with Photon network engine[9] for the server-side and Unity game engine[10] for the client-side.

1.3.1 Architecture overview

MMOnkey has a client-server architecture (Figure 1.10), using peers for the communication part. Each decision made by the player is based on the local replica on the client-side. The state change requests are then sent to the server, which maintains the master state of each object. Based on the game semantics and prerequisites of each action in the game, the server can decide to accept or reject the request to change the state of an object. On acceptance, after the master state is updated, the updated state is disseminated to the interest management to decide which client has the right to receive the update.

1.3.2 Entity system

Figure 1.11 depicts the entity system of MMOnkey. Each object in the game scene is an entity. The EntityFactory is responsible for creating the entities. The subclasses are decided based on the input parameters passed to the functions, each responsible for creating different entity types. After the entities are created, they are added to the game world. The attributes of entities inherited from the Attribute class and stored in a dictionary. The state of the entities can be modified by the Modifier class.

---

[8] https://docs.microsoft.com/en-us/dotnet/csharp/

[9] https://www.photonengine.com/en-US/Photon

[10] https://unity.com/

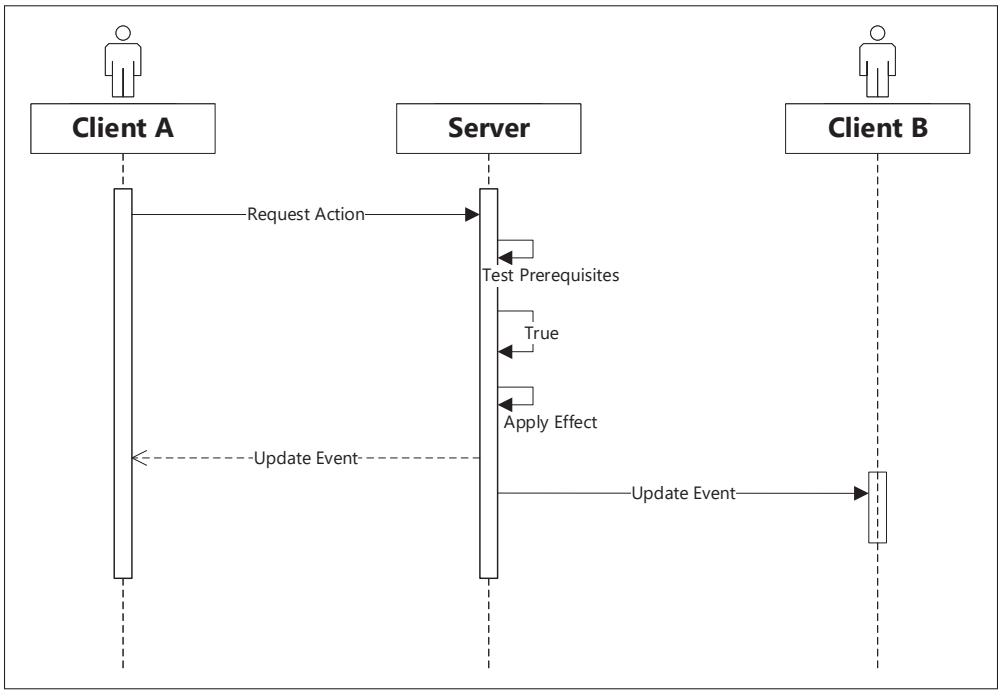

Figure 1.10 Simplified MMOnkey Architecture. Clients send the action request to the server. Server checks the prerequisites of the action, and if the prerequisites are passed, it applies the effect of the action and sends the update event back to all the eligible clients

1.3.3 Action system

An action object holds all the information about the actions requested by the client. It inherits from the data contract implemented by Photon. It is used to check the validity of the request. After a request for action arrives at the server, the action object factory is responsible for creating an instance of that object.

In Figure 1.12, two examples of different action object implementations are depicted. Dash is not a complex action; therefore, it proceeds straight to the action object. On the other hand, the orison of healing is a complex action with the cast time. Cast time means that if every prerequisite of action is passed, the modifier will be applied to the game world after the cast time has passed. Complex behaviours such as casting are realized by ActionContinueCondition. TimedContinueCondition is used to decide the right time to fire a continue event, and the

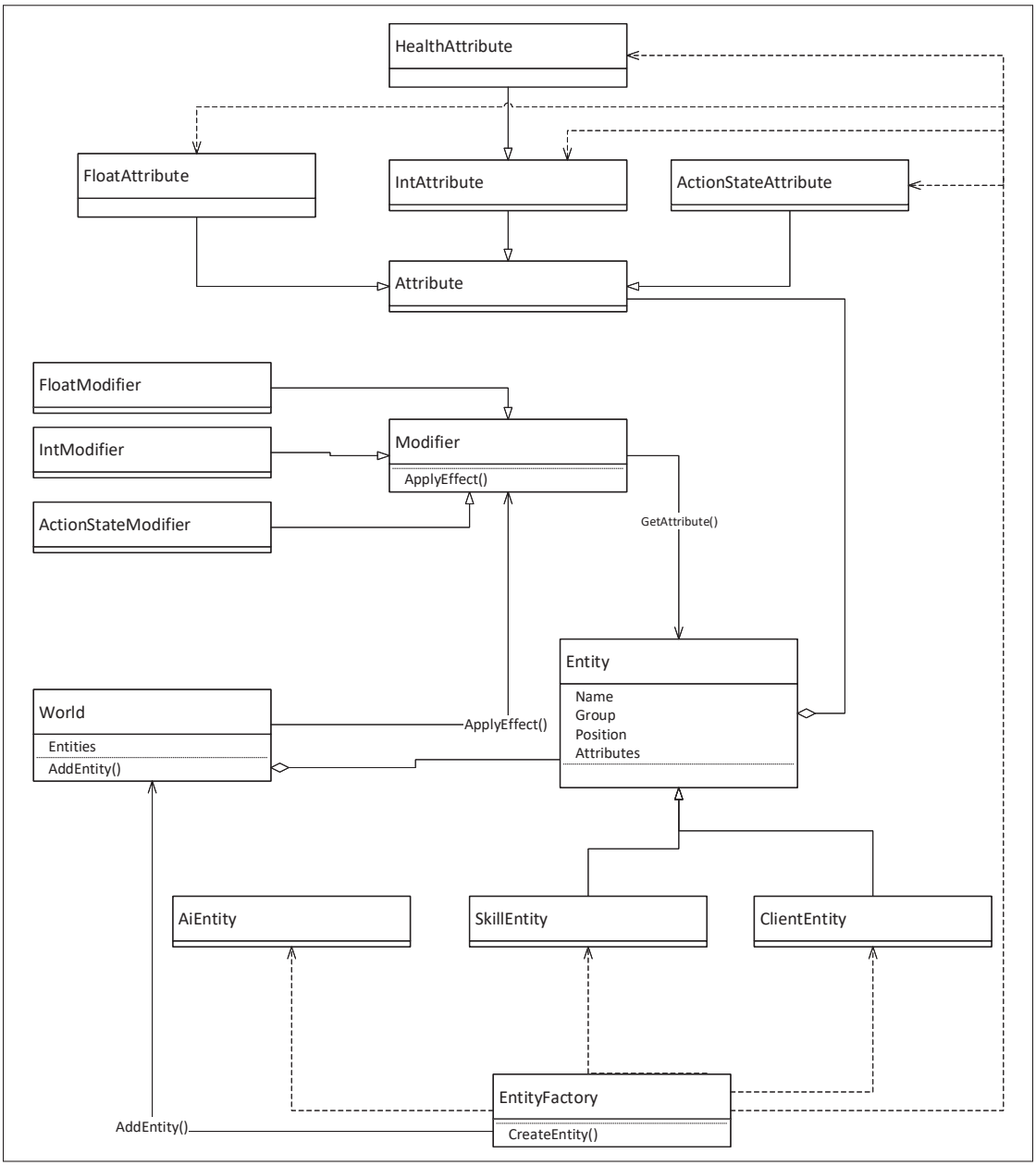

Figure 1.11 MMOnkey entity system

CastActionObject is using it for the casting behaviour. Actions can be interrupted during the casting process; thus, the interruption events are defined as an effect for some actions. Interrupt events are considered by an InterruptContinueCondition. The action objects manage them by considering the priority of the actions based on the time the server receives the request. After

the first action is handled, the other continue conditions will be disposed. The interrupt effects are handled by the InteruptionHandle and are not applied to the game world.

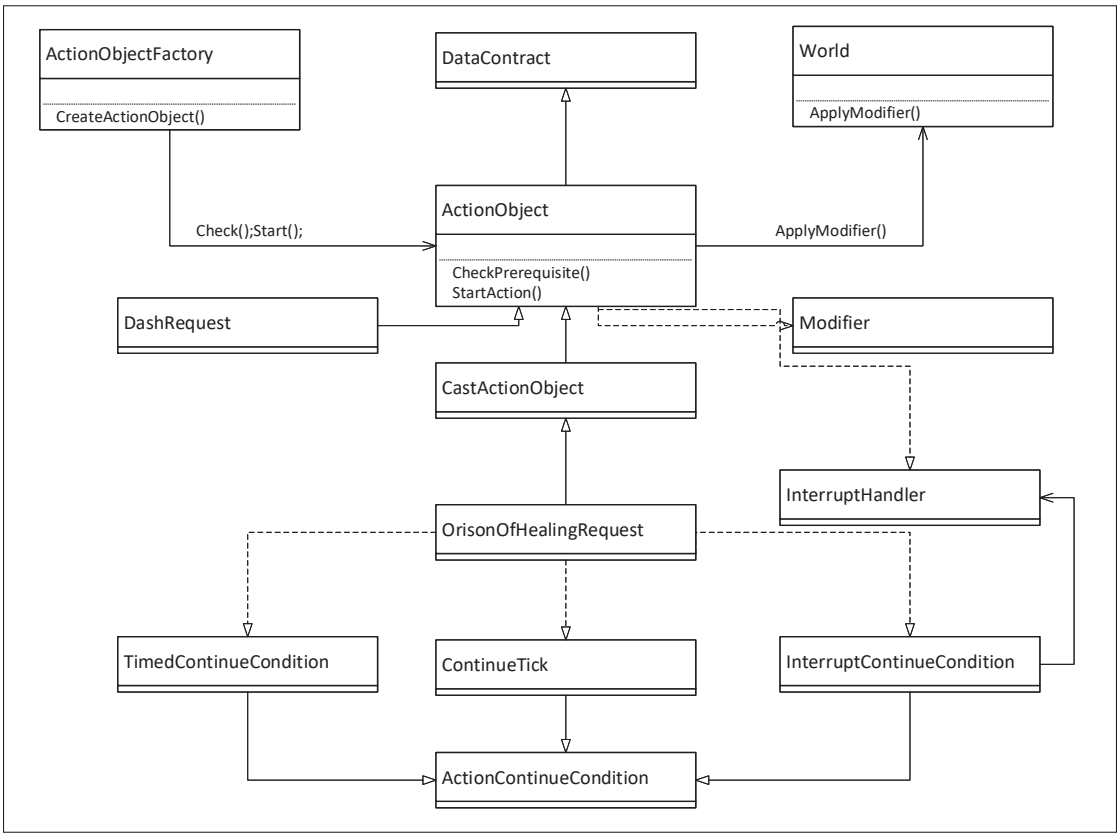

Figure 1.12 MMOnkey action system

1.3.4 Client and server communication

Different types of message exist in our framework:

- Commands: Action requests sent from client to server for authorization
- Response: After a command is received, server can directly respond to it
- Event: After a command is received, server can send an event at any time

Different message codes are defined for each request and used to recognize the attached parameters to each message (e.g., 0: Enter World, 1: Move, 2: Action). When a client requests

to enter the game world, the parameters will be sent to the server, and then the data contract defines the parameters of each message (e.g., Move data contract needs a position parameter in a message). Each action object in MMOnkey is an action data contract, and based on its action code, it will be sent to the proper subclass.

1.3.5 Peers and operation handlers

Each time a client is connected to the server, an instance of the peer class will be created by the Photon application. Anytime the client sends a request to the server, it will be passed on to the specific client's peer, and there it will be handled by different operation handlers (Figure 1.13).

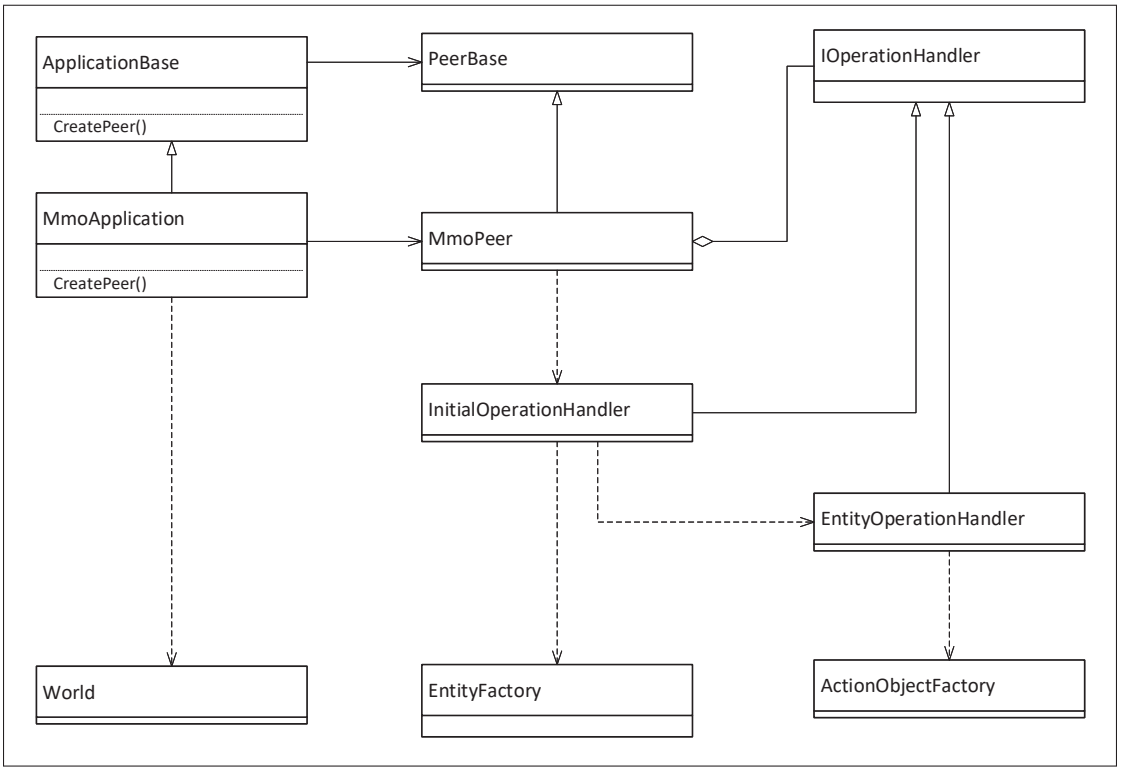

Figure 1.13 Peer and operation handler

1.3.6 Interest management

Square tile interest management is one way of implementing region-based interest management. This IM is part of the Photon application to determine which client should receive the updates. Different Retlang (CSharp threading library) message channels are used for different message types. In MMOnkey, five different message channels are on each region:

- RegionEventChannel
- PositionEventChannel
- EntityRegionChangedChannel
- RequestInfoInRegionChannel
- RequestRegionExitInfoChannel

The first two channels are used to publish the updates in the region. The others are for the moment an entity moves from the current region to another. EntityRegionChangedChannel is used to notify other entities on the entrance of the entity to another region. RequestInfoInRegionChannel is used so that the entity receives the information of other entities in the new region it just entered. RequestRegionExitInfoChannel is used to get the information of the entities outside the interest area.

1.3.7 Client network layer

The client network layer of MMOnkey is illustrated in Figure 1.14. Photon library comprises PhotonPeer and IphotonPeerListener, which are responsible for the communication between client and server. ServerPeerListener is the implementation of this interface for MMOnkey. Instead of the Photon library, the RequestOperations, Game, EventOperations, and ResponseOperations are responsible for the communication in MMOnkey. Game class is used to establish the client connections to the server. RequestOperations class makes and forward an object with the specific

action parameters to the peer, and from there, it is sent to the server. Events are handled by EventOperations and response messages by ResponseOperations.

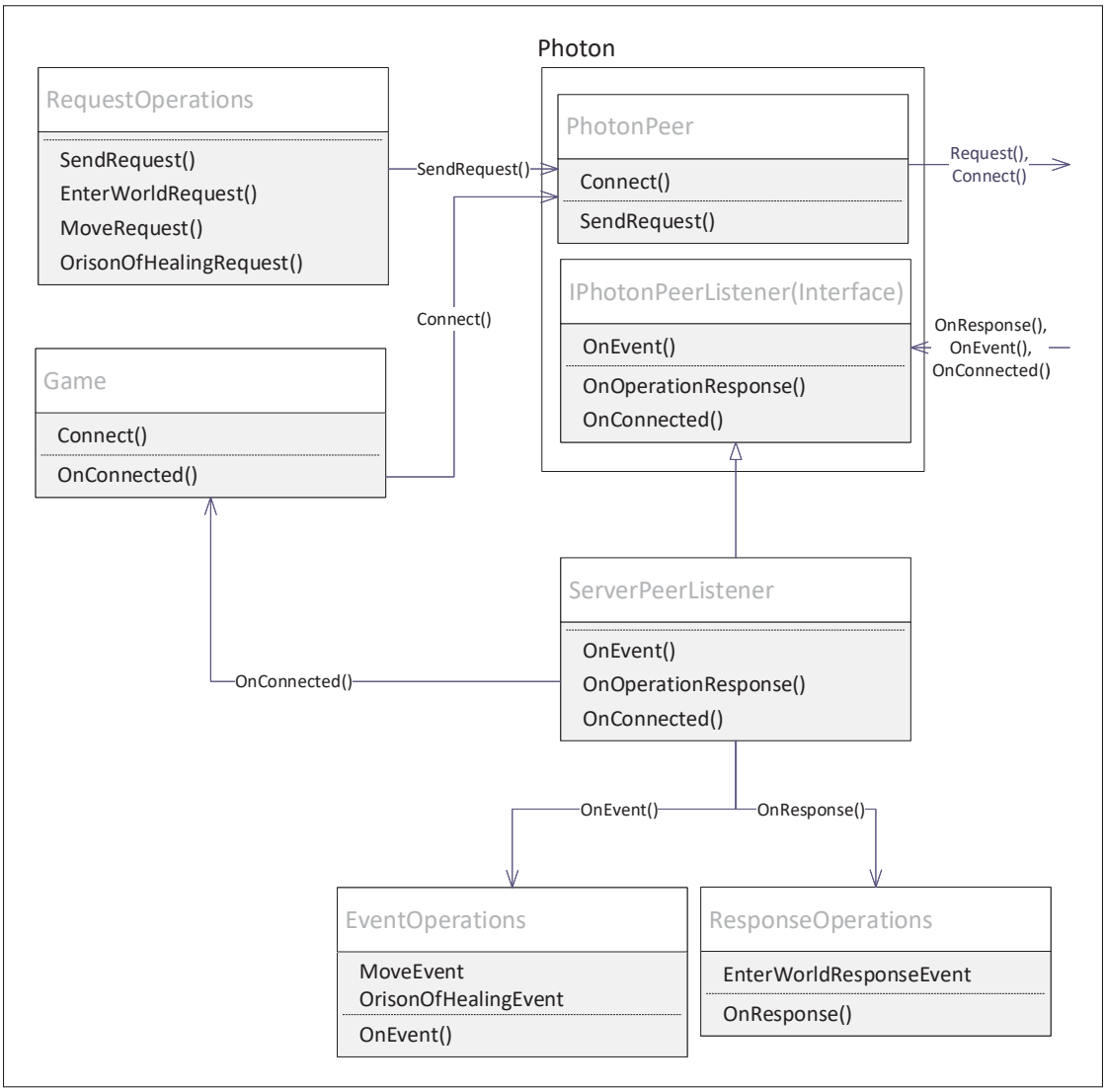

Figure 1.14 MMOnkey client network layer

1.3.8 Combat system

Each player controls an avatar to interact with the game world. W-A-S-D keys or arrow keys on the keyboard are used as the movement keys. Manual targeting is implemented to be used

during combats. Players need to aim at the target (can be an area or an entity) using the mouse pointer. The left mouse button is for using axe auto-attack or bow auto-attack, and 1-2-3 keys on the keyboard are for using three different skills during the gameplay. When players' avatars and NPCs use their weapons to affect other entities, in order for their action to be effective, the target must be within the target area of the action. Figure 1.15 demonstrates the target area of axe-auto attack and bow-auto attack in the framework implemented as a cone and a rectangle, respectively.

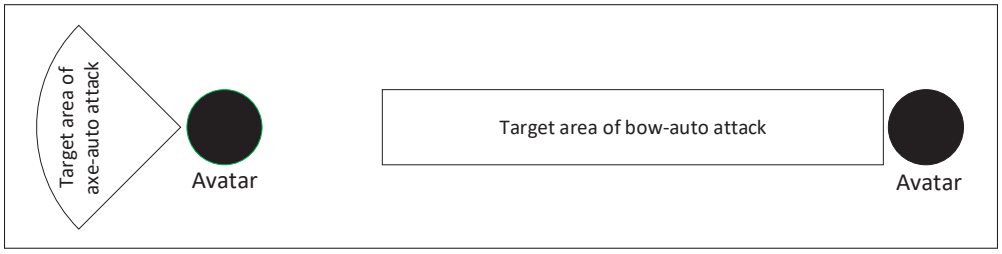

Figure 1.15 Target area of the melee attack (axe auto-attack) and long range attack (bow auto-attack) in MMOnkey

1.3.9 Similar works

Similar works to MMOnkey are Colyseus (Bharambe, Pang & Seshan, 2006), and Mammoth (Kienzle, Verbrugge, Kemme, Denault & Hawker, 2009). Colyseus is a distributed architecture for multiplayer games with a single-copy consistency model and primary-copy replication. Mammoth is another research framework for MMOGs with a replication engine using interest management. A modular infrastructure is offered for the NPCs which are controlled by the AI. The reason behind choosing MMOnkey is that it is explicitly implemented as a research framework for MMORPGs with a fast-paced combat system.

In this chapter, we presented cloud gaming platforms and gave a thorough overview of MMORPGs. We also provided a detailed review of the MMOnkey framework for MMORPGs. In the next chapter, we discuss the related works performed to improve the QoE in cloud gaming.

CHAPTER 2

RELATED WORKS

In this chapter, we present some of the related works performed by researchers to optimize cloud gaming systems. Referring to the proposed classification system of cloud gaming papers in Cai *et al.* (2016), the cloud gaming systems optimization is split into two groups. The first group is cloud server infrastructure that is further divided into two areas of resource allocation and distributed architectures. The second group is content and communications, which is divided into two areas of data compression and adaptive transmission. Since our proposed model can be included in the second group, we conclude the chapter by highlighting the differences between our approach and the similar state-of-the-art methods covered in the content and communications section.

2.1 Cloud Server Infrastructure

In the cloud server infrastructure class, the problem of resource allocation among multiple data centers and clients is studied to maximize the cloud gaming experience for users. The second area includes the works proposing new architectures and optimizing the distributed architectures of cloud gaming platforms.

2.1.1 Resource allocation

Wang, Liu & Dey (2012) proposed a wireless network-aware cloud scheduler for CMG to make it scalable and also to decrease the costs of cloud platforms. Their approach takes into account the wireless network constraints and the cost of cloud resources to schedule the cloud instances for different users. Their algorithm comprises different functions: cost-based utility function, mobile gaming user experience-based utility function, and a mobile gaming user experience/cost-based utility function for the resources allocation process. They also proposed an algorithm that considers the dynamic changes of the wireless network to adjust

the communication and computation requirements of users. Their experimental results show a better QoE and lower cost for cloud services than the original CMG approach.

A dynamic resource provisioning for cloud-based gaming infrastructures is proposed in Marzolla, Ferretti & D'angelo (2012). Their proposed model estimates the response time of the system for different configurations. A greedy algorithm is used in their model to determine the minimum number of computing nodes that can be allocated to satisfy the response time required by MMOGs.

The resource allocation strategies based on the predicted session length are presented by Li, Tang & Cai (2015) to reduce the cost of the cloud gaming systems by deploying the virtual machines (VMs) more efficiently. The authors proposed an efficient request dispatching algorithm to be used instead of classical bin packing algorithms (e.g., First Fit and Best Fit) to assign the play requests based on the duration of game sessions. The length of each session is predicted using a neural network-based approach. Their experimental results show the reduction of the cloud server resources using the proposed dispatching algorithm.

In Dhib, Boussetta, Zangar & Tabbane (2017), the authors proposed a model to improve the resource allocation in cloud gaming services focusing on MMOGs. Their model considers a trade-off between the allocated resources costs and the response delay proportionate to the allocated resources.

Various heuristics are proposed by Deng *et al.* (2018) to address the sever allocation problem in MCG. For instance, the lowest-combined-price algorithm considers a data center with the lowest combined price of all its permitted data centers to assign the client to the selected data center. The lowest-capacity-wastage avoids the capacity wastage of each data center in order to enhance the cost-effectiveness of the server allocation. The lowest-amortized-cost heuristic is used in their proposed hill-climbing algorithm to consider the lowest amortized cost for each client and assign a group of clients to the chosen data center until all the clients are assigned to data centers.

A cloud gaming architecture is proposed for MMORPGs in Jaya, Cai & Li (2020) for a flexible resource allocation. Their proposed architecture separates the game logic and players' interactions management from the rendering process. Any rendering server is then possible to serve the players in any virtual location. In order to minimize the rental cost of rendering servers for MMORPGs, different heuristic algorithms in two categories of online algorithms and offline algorithms for rendering server allocation are proposed by the authors. In the category of online algorithms, the lowest price allocation algorithm selects a rendering server with the lowest cost. The lowest waste resource allocation minimizes the waste resources on active rendering servers. In order o minimize the additional workload of the central processing unit (CPU) for the incoming requests, the highest workload share algorithm is used. Finally, as for the lowest waste price allocation, different factors are considered. The local search and mathematical lower bound algorithms are proposed as the offline rendering server allocation algorithms. A local search algorithm is used to get the offline solution in a short time, and a mathematical lower bound algorithm is for an optimal solution but in more time.

2.1.2 Distributed architectures

Latency in cloud gaming is an issue needed to be addressed. In the case of MMOGs, this genre of games can suffer from the increased latency introduced by cloud gaming platforms. Süselbeck, Schiele & Becker (2009) propose to benefit the P2P techniques to distribute the functionality of MMOG server between multiple servers. These servers can be co-located with the cloud servers to reduce the additional delay of playing MMOGs on cloud systems.

In order to decrease the recognized latency by the clients, deploying edge servers near the end-users is proposed by Choy, Wong, Simon & Rosenberg (2012). Experimentally, they show that the user coverage can be improved by more than 28.0% by adding a few servers to the network edge. In Choy, Wong, Simon & Rosenberg (2014), the authors investigate that serving 90% of end-users requires a large number of servers in an edge-only deployment. They proposed a hybrid edge-cloud architecture in order to reduce the latency of on-demand gaming. Different approaches to deploying a hybrid infrastructure using the edge servers and the cloud resources

are explored in their work. They discovered that a hybrid infrastructure using a voting-based strategy for smart edge selection and game placement can serve 90% of end-users with 80 ms latency.

The deployment of the game services on a distributed cloud infrastructure is proposed by Kämäräinen, Siekkinen, Xiao & Ylä-Jääski (2014) in order to reduce the latency of cloud gaming. The authors show that a hybrid decentralized cloud infrastructure allows the deployment of game servers near the local clients in a local network. The proximity of the cloud server can be considered based on the network connection quality of the user and the required latency of a specific game. Their prototype distributed cloud gaming platform can achieve a shorter response time compared to the centralized deployment of the cloud gaming server.

2.2 Content and Communications

The related works discussed in this section are mainly proposed for optimizing the cloud gaming platforms in the area of content and communications. in the first subsection we present some of the performed works proposed in order to optimize the compression methods of cloud gaming platforms. The second subsection comprises some works that are more similar to our approach. Their approach is to manipulate the graphical content of the game considering the network dynamics to reduce the video bit rate and make it suitable for video streaming purposes.

2.2.1 Data compression

In Shi, Hsu, Nahrstedt & Campbell (2011), the authors introduced the use of graphics rendering contexts, taking into account the pixel depth, camera motion pattern, and the auxiliary frames to assist the video encoding. They proposed a 3D image warping to improve the video encoding performance and reduce the interaction latency. They also proposed using auxiliary frames in double warping, which improves the 3D image warping performance. Auxiliary frames are the rendered frames by the game engine considering the other viewpoint of the camera that does not exist in the video sequence. In their proposed video coder, which is built on top of the H.264

coder-decoder (CODEC), the keyframes are first selected in the video, and other intermediate frames are interpolated with the 3D image warping algorithm. Finally, H.264/AVC encodes the warping residues using a lower bit rate to leave more bits to be assigned to encode keyframes.

In Semsarzadeh, Hemmati, Javadtalab, Yassine & Shirmohammadi (2014), the authors proposed the use of objects' information in game engines to accelerate the video encoding process. An interface is presented to be used between the game engine and the video encoder to collect some information about the motion of the objects from the game engine and pre-process them in a way that is understandable for the video encoder. The information about the motions of the objects within the scene can be used to skip the motion estimation procedure in the video encoder. They demonstrate the possibility of saving 14.32% of the motion estimation time and 8.86% of the encoding time by using their model.

In Xu, Guo, Lu, Li, Au & Fang (2014), a technique based on the motion estimation strategy to rectify the camera rotation is proposed to have a reference view parallel to the current view for a more motion estimation friendly process and fewer bits to compress the residues. They proposed that the camera motion between the adjacent views be broken up into three different types: horizontal and vertical displacement, perpendicular displacement, and rotation. The translation-based motion estimation model is less suitable for camera rotation compared to other types and requires more bits. In their proposed algorithm, the camera rotation is rectified to make the reference view parallel to the current view, which produces motion estimation-friendly video frames. After this technique is performed to the game scene, they use their proposed edge preserved interpolation algorithm to have a better edge prediction than the conventional image interpolation algorithms. Their method achieved the 18.0% BD-rate reduction compared to the algorithm of x264.

Liu, Dey & Lu (2015) proposed to use the rendering information for generating a MB level saliency map for all the video frames. The saliency map and scene composition information are then used to prioritize the regions of a video frame based on their importance. Based on the importance of each MB and the bit rate budget, the quantization parameter (QP) values

are adopted. They also used the rendering information and proposed a method for the MVs calculation. They used the computed MVs in a fast mode selection algorithm to reduce the number of candidate modes of each MB. Their second approach compared with regular H.264/AVC can save 42.0% of the encoding time but with a little deterioration in video quality and decrease in peak signal-to-noise ratio (PSNR).

Sun & Wu (2015) proposed a MB level rate control scheme for H.264/AVC based on the region of interest and keyframes from the video and scene-change detection. They proposed two ways to get region of interest (ROI). It should be provided by the game developer or extracted by the video encoder. The ROI information is then translated to its importance at the level of MB. The bigger is the ROI value, the more bit rates are allocated to this area. The rate control-aware cloud system is proposed to produce real-time ROI information for the video encoder.

In Hegazy, Diab, Saeedi, Ivanovic, Amer, Liu, Sines & Hefeeda (2019), the authors used the same technique as in (Sun & Wu, 2015) with a different weight assignment method. Their target video encoder is HEVC. Similarly, they used the ROIs to optimize the quality by allocating a different number of bits to different regions based on their importance. ROIs are defined using the information revealed by the game developer about different objects in the game. Their model is a software component implemented between the game process and the encoder. It controls the encoding bit rate. Based on the ROIs in the game, it computes different encoding parameters to be used by the encoder. After the ROIs are defined, different weights are calculated for blocks inside and outside the ROIs. The bit rate savings in ROIs are between 21.0% and 46.0% compared to the baseline HEVC encoder.

Lu, Wang & Chien (2019) proposed a solution to reduce the time of video encoding dedicated to compute in-game MVs. Their method considers the information of the objects in the game application to preprocess the objects' MVs. Pre-processing of the MVs is done in three steps: a coordinate system transformation, MV determination, and finally, the suitable MVs are selected for the prediction units in the encoder. After the pre-processing is done, the pre-processed MVs of objects are passed to the HEVC encoder to be used as the value of motion estimation. The

encoding performance of the HEVC is increased by bypassing the traditional diamond search. Using their solution can reduce the coding time up to 49%, and the bit rate reduction is up to -17.0% Bjontegaard delta bit rate (BDBR) with the HEVC encoder.

Ahmadi, Zadtootaghaj, Pakdaman, Hashemi & Shirmohammadi (2021) proposed a skill-based visual attention model which is used by the video encoder to allocate bit rate based on the skill of players clustered in beginner, intermediate and expert skill levels. An eye-tracking experiment is performed to collect the gaze data, and further identify the attention pattern similarities among the players of each group. Due to the impossibility of collecting eye-tracking data for different players, they used the score of the players to predict their attention clusters. Finally, a saliency map is generated for each cluster to be used by the encoder. The parameters of the encoder, such as QP values, are set based on the importance of each region. Their experimental results show the video bit rate reduction in H.264/AVC by an average percentage of 13.0, 5.0, 15.0, for the beginner, intermediate, and expert skill levels, respectively.

2.2.2 Adaptive transmission

In Wang & Dey (2010a), different optimization techniques are proposed for the application layer to guarantee an acceptable response time and video quality in cloud gaming systems. The developed optimization techniques comprise downlink gaming video-rate adaption, uplink delay optimization, and client play-out delay adaptation. The knowledge of the game type is used in the video rate adaption strategy to find the best encoding settings for a particular game. A rate-selection algorithm is also used to reduce the downlink delay to less than an acceptable response time threshold by the end-user. This algorithm is used so that the bit rate is dynamically adjusted based on the network condition.

In Wang & Dey (2010b), the same authors proposed a technique to address the constraints of the communication and computation in CMG. They considered different parameters such as realistic effect (color depth, anti-aliasing, texture filtering, and lighting mode), view distance, texture, and environment details and identified the costs of the communication and computation

of each parameter. Two steps of optimal adaptive rendering settings and the level-selection algorithm of their technique change these rendering parameters and can satisfy the constraints of the communication and computation due to the network bandwidth fluctuation and the available capacity of the cloud server. Their technique reduces the communication cost (video bit rate) and the computation cost (GPU utilization) of the cloud server.

In Hemmati, Javadtalab, Nazari Shirehjini, Shirmohammadi & Arici (2013), a selective object encoding is proposed to achieve a lower video bit rate by excluding the less important objects in the game scene and help the process of frame encoding on the server. The importance of each object in the game should be provided by the game designer for each object in each activity in the game, and the model selects the important objects considering the activities in the game and includes them in the game scene. Their approach achieves a lower bit rate of 2.2% to 8.8% for video streaming.

Inspired from proposed solutions in (Wang & Dey, 2010b) and (Hemmati *et al.*, 2013), in Lu, Liu & Dey (2017), the authors proposed a model to manipulate the visual content by considering the importance of in-game objects. Based on the importance of each object, it is either decided not to render it in the scene or to manipulate its textures to reduce the video bit rate.

Chuah & Cheung (2014) introduced a game image coding framework with two layers. The first layer is the base layer, which comprises the low-quality images sent to the thin-client device to be rendered locally, and the second one is the enhancement layer, which is sent by the cloud servers instead of high-quality images to the clients to be used for the base layer quality improvement. They showed that parameterization of polygon modeling and illumination for the base layer leads to a small computation at the mobile devices. Also, the transmission of the enhancement layer requires a lower bit rate compared to the transmission of a high-quality image. Their proposed layered coding reduces the bit rate up to 65.0% compared to H.264/AVC inter-frame coding to compress the images.

In Wu, Yuen, Cheung, Chen & Chen (2015), a novel transmission scheduling framework is proposed to deliver high frame rate video in CMG. The model is capable of adjusting the video

traffic load dynamically and perform forward error correction coding. To achieve their goal, they first proposed an online video frame selection algorithm in order to minimize the total distortion based on different factors such as network status, input video data, and delay constraint. An unequal forward error correction coding scheme is introduced to layout differentiated protection for Intra frames and predicted frames with a low-latency cost. Their proposed model optimizes the quality of high frame rate video by filtering video frames and adjusting data protection levels. Their experimental results show that the model outperforms the existing transmission schemes in video quality improvement measured in PSNR, and end-to-end latency reduction. By profiling the execution time, the result illustrates that their model has better performance, up to 20.0% compared with the reference transmission schemes in most emulation scenarios.

2.3 Discussion

In the previous sections, we surveyed some of the related works with their achievements in two main categories of cloud server infrastructure, and content and communications. As our work can be placed in the second class, in this section, we identify the possible weaknesses of the approaches in this group and also discuss the differences between our model and these performed works.

In works such as Sun & Wu (2015) and Hegazy *et al.* (2019), the authors proposed the use of ROI extracted using the information revealed by the game developers for different objects in the game. The ROI then could be used by the video encoder for rate control purposes. In works such as Liu *et al.* (2015), the authors proposed using a saliency map and scene composition information for the ROI identification and bit rate allocation. In Ahmadi *et al.* (2021), a saliency map is generated considering the score of the players in the game. The saliency map is then used by the encoder to set the QP values based on the importance of each region. Generally speaking, these tasks are difficult to implement in practice and require access to the video encoder's code and deep knowledge of such code. To use a saliency map for the encoder, substantial image processing operations are needed to be done, which is not very befitting. Changing the QP in an encoder based on factors such as ROI or attention maps, or modifying the motion estimation

process as proposed in Semsarzadeh *et al.* (2014); Xu *et al.* (2014), and Lu *et al.* (2019) is very challenging and codec-specific. Furthermore, using a non-standard video coding process seriously limits the deployment of the game.

As discussed in Shi *et al.* (2011), only the cloud gaming platforms in which the video encoder can extract graphics rendering contexts in real-time from the rendering engine can benefit from their methods.

The rendering adaptation technique proposed in Wang & Dey (2010b) is based on the communication and computation constraints and does not consider the ROI of the player. They proposed a technique to change different rendering parameters, as explained earlier. However, considering the critical regions to the player to change these parameters could be interesting and helpful to enhance the QoE of the user.

Determining the importance of in-game objects in each activity in the game using the information provided by the game designers as proposed in Hemmati *et al.* (2013) is not always possible. All the activities in the game, besides all the in-game objects, should be listed by game designers. The importance of each object should then determined by them for each possible activity in the game. The other important case that needs to be considered is that excluding fixed and immovable objects from the scene do not always significantly reduce the video bit rate. Moreover, sometimes removing an object can expose other objects that need more bits to be coded by the encoder.

2.3.1 The differences between our approach and related works

Our goal is to develop a model that can be used for most MMORPGs on cloud gaming platforms to improve the video quality with the same bit rate or to reduce the video bit rate for the same video quality, regardless of which encoder is used by cloud systems and without any required changes to the encoder's code and its inputs.

We focus on the game, so that it generates content that is easier to encode. We plan to use the same concept of distance as proposed in Wang & Dey (2010b), and the importance of each object in the game as proposed in Hemmati *et al.* (2013) in order to apply changes in the game scene. The difference is that our model will not consider these concepts to decide what objects should be included in the scene or to manipulate the textures and effects of the objects in the scene, as these tasks require extra work to be performed on the cloud server.

The ARCODE we propose for MMORPGs on cloud platforms, will simply consider the importance of each movable object in the scene and determine different position update rates for different objects in the game to reduce the number of motion updates and changes in the scene. Accordingly, it makes the video file more encoder friendly since fewer bits are consumed by the encoder to code the video frames at a given quality level.

Many proposed solutions require a significant amount of information to be provided by game designers or players to determine the importance of each object and each region in the game. Our model is action-aware and determines the importance of each object based on the actions it can perform in each state of the game. Only a few pieces of information, such as the range of each action, should be provided by the game designer. The proposed technique in Wang *et al.* (2017), and briefly explained in Section 1.2.6, is taken into account to implement our model. In the following subsection, we identify the main differences between our model and CSAIM.

2.3.2 The differences between ARCODE and CSAIM

The major difference between the ARCODE and CSAIM model is the challenges considered to be addressed by implementing each one. CSAIM is implemented in order to reduce the updates propagated in a client-server architecture and increase the scalability of MMORPGs in this traditional network architecture. The proposed ARCODE is implemented to reduce the number of changes and motions in consecutive video frames in cloud gaming platforms which results in fewer bits consumption by the encoder to code the video. Therefore, the parameters each model takes into account to determine the position update frequencies are different.

Technically speaking, CSAIM only considers the AoE of the player's avatar. Thus, each player receives updates with short intervals between each update based on the AoE of its own avatar, and the AoE of other objects are not considered in the model. Here, the issue is when there is no object within the AoE of the player's avatar, but the player's avatar is within the AoE of another object. In this situation, the player who controls the avatar does not receive enough updates from the position of the other object, and making an accurate decision is not possible.

Each player selects different skills and weapons for their character, and different weapons and skills could be determined for the NPCs. Accordingly, the size of AoE may be different for each object in the game, which makes the situation asymmetric. Figure 2.1 illustrates this through three possible scenarios. Thus, considering the AoE of other objects in the model should not be neglected. In each state of the game, it is common to be in the AoE of different objects simultaneously.

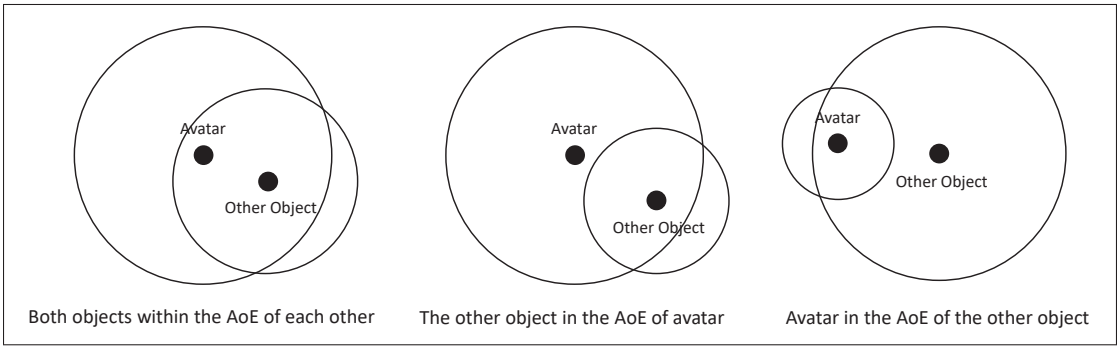

Figure 2.1 Different possible situations in a combat area which requires to consider the AoE of different objects

The proposed ARCODE takes into account every movable object type in the combat area (the avatar of all players, NPCs, and skill objects) and form an AoE for each object considering the characteristics of the object and actions it can perform in the game.

Apart from considering the AoE of all the objects in the combat area, our model takes into account different frequency areas (compared to the CSAIM which only considers two frequency areas) based on the distance of the player's avatar to other objects, the size of each object, and

the player's available bit rate for the game streaming. Based on all these factors, ARCODE considers different position update rates for each object in the game.

In this chapter, we surveyed the works performed in order to improve the experience of playing video games on cloud systems. We finalized the chapter with a comparison between our proposed model and similar state-of-the-art methods. In Chapter 3, we provide a detailed presentation of our proposed model, ARCODE, for MMORPGs on cloud gaming platforms.

CHAPTER 3

PROPOSED SOLUTION: ACTION AWARE COMBAT MODEL

In this chapter, we present our proposed solution called ARCODE, implemented for MMORPGs running on cloud gaming platforms. Our objective is to reduce the number of bits consumed by the encoder while maintaining the same visual quality as the traditional approach. In other words, ARCODE seeks to achieve better video quality for the same number of bits.

Compressing the video frames using lossy compression algorithms will directly affect the quality of the video. The vital part to consider is that we allocate a constant bit rate for video streaming in cloud gaming and the encoder can therefore only use a limited number of bits on each frame.

Different video resolutions and frame rates require different bit rates for the video to be compressed at a certain quality. It further depends on the nature and the properties of the video to encode. Videos with more complex motions and textures require more bits to encode at a certain fidelity.

We investigate the relationship between video game graphics and Intra and Inter prediction, which are used in video compression to transmit video information efficiently (Section 1.1.3). We implemented our model based on the assumption that less frequent changes in consecutive frames result in a smaller bit rate for the same quality or equivalently better quality for the same bit rate.

In cloud gaming, each client connects to the cloud gaming platform with a different available bit rate. If the bit rate is low, fewer bits can be transferred, and the video quality will suffer, especially in the case where many movements and changes are happening in the game scene.

ARCODE takes into account the importance of each object considering the actions it can perform during combat. The model captures different action data (e.g., the maximum range of action and its target area) to form an area of effect (AoE) around the object. As discussed in Section 1.2.5,

the AoE is an area where the state of the objects within the area can be affected by the performed actions.

In addition to the AoE mechanism, the size of each object and its distance to the player's avatar is also considered by the model, along with the bit rate and resolution chosen by the player before the game streaming is started. Based on all these factors, the model determines different position update rates for each object in each state of the game. Reducing the rate of the updates in the states of the game where an object is considered less crucial to the player reduces the number of changes in the video frame sequence. Accordingly, it results in fewer bits consumption by the encoder to code the video frames.

3.1 AoE for Different Object Types in ARCODE

ARCODE captures different action data for different types of movable objects in the combat area and forms an AoE for each object. One type of movable object in MMORPGs is the character, which can be an avatar controlled by a player or an NPC controlled by AI. We refer to them as character entities. The other type is a skill object spawned on the request of character entities. We refer to them as skill entities. Figure 3.1 demonstrates different entity types in our exemplar game. In the following subsections, we explain the forming of an AoE for each of the mentioned objects in the game.

3.1.1 AoE of character entity

The importance of a character entity in each state of the game is determined by the *actions* (e.g., using a weapon, casting a skill) it can perform during the gameplay to change the state of other entities. Considering the properties of the skills and weapons a character entity owns, different circle-shape areas are formed as the AoE of the character entity. The size of each area is based on the range in which the character entity can perform a specific action.

As illustrated in Figure 3.2, if a character entity has three skills with the range of 9 for the first skill, 11 for the second one, and 20 for the third one, and it also owns a weapon like an axe which

Figure 3.1 Different entities in our exemplar game

allows the character to perform the axe-auto attack action with a radius of 4, then four different size areas are formed as the AoE of the character entity with the character entity in the center.

3.1.2 AoE of skill entity

In our model, the skill objects are considered independent entities with their own behaviour after being spawned into the game world. In each state of the game, the target area of each skill specifies the area that the skill can impact. As explained in Section 1.3.8, for the executed

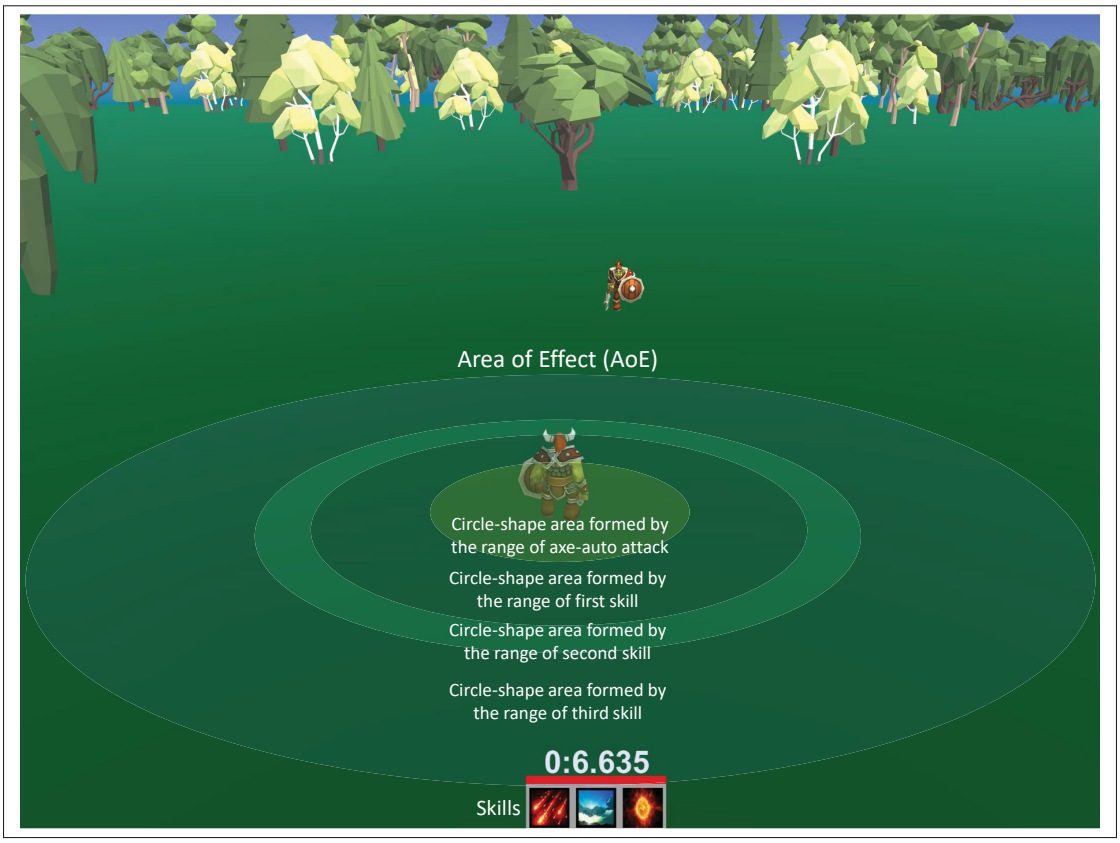

Figure 3.2 AoE of a character entity

actions to be effective, not only the target object should be within the range, its position should be within the target area of the performed action. The target area of skill is considered as the AoE of its skill entity. The target area of two skills in our exemplar game is shown in Figure 3.3. After a skill is cast on the request of a character entity, it only affects the entities within its target area. The skill entity is just visual feedback to show that the skill can affect this particular part of the game world. The AoE radius is equal to the radius of the target area, which is considered according to the size of the skill entity.

To wrap up this section, we demonstrate in Figure 3.4 a state in which the player has performed two different actions to damage the health of the opponent entities within the AoE of his or her character entity. Imagine that the player used a skill to damage the health of the opponent entity #1 and #2. The player then performed the axe auto-attack to damage the health of the opponent

Figure 3.3 Target area of skill entities

entity #3. Here, all these three opponent entities are affected by the skill and the axe auto-attack as their positions are within the target area of these actions. The opponent entity #4 and #5 are safe in this state. Although they are within the AoE of the character entity, their position is not within the target area of the performed actions. Therefore, they are not affected in this state by the current performed actions.

Figure 3.4 A combat state where a character entity has used an axe-auto attack and a skill to attack the opponent entities within its AoE. Only the opponent entities within the target area of the performed actions are affected

3.2 Types of Situations Handled by ARCODE

A variety of weapons and skills are available for different players and NPCs in MMORPGs. Depending on what weapons and skills each character entity has, the size of the AoE can be different. A player's avatar may be within the AoE of different entities simultaneously without these entities being within the AoE of the player's avatar. This asymmetric situation is addressed in our model by considering the AoE of all entities in each game state. ARCODE considers different situations in a battle area to decide the update rates each player needs to receive from

different entities in each game state. In the following subsections, we explain each situation. To explain each situation, we refer to the player's avatar who owns the area and from whose point of view the game is going on as the client entity.

3.2.1 Updates based on the AoE of client entity

The client entity receives updates based on its AoE within which one or more character entities can be. Each circle-shaped area of the AoE has its associated update frequency.

If the circle-shape area around the client entity is formed based on the range of the weapons, the model considers the shortest interval between updates from the position of the character entity within the AoE of the client entity. Since the target area of the weapons are usually small, and their effect is applied instantaneously, the player needs to have an accurate approximation of the position of the target to attack it with weapons his or her avatar owns. If the position of the target is not accurate enough, and the player has a stale view (the outdated view) of the target, the attack can be missed and not be successful.

Figure 3.5 demonstrates how the attack of the client entity on the target can be missed. The real position of the target is maintained on the server. The outdated position is at the client-side. The figure illustrates that if the player receives the outdated position of the target, an undesirable inconsistency happens. It is quite apparent why the position of the target should be updated frequently in this highly interactive combat state.

On the other hand, the target area of skill is usually big, and most of the time, the effect of skill is not applied instantaneously to the game world. Therefore, If a character entity is within an area formed based on a skill the client entity owns, the position of the character entity can be updated with lower frequency.

We consider fixed intervals between each position update for different areas of the client entity's AoE. We do not want these intervals to change under any circumstances and want to be sure that the client entity always receives the most appropriate update frequencies based on its own AoE.

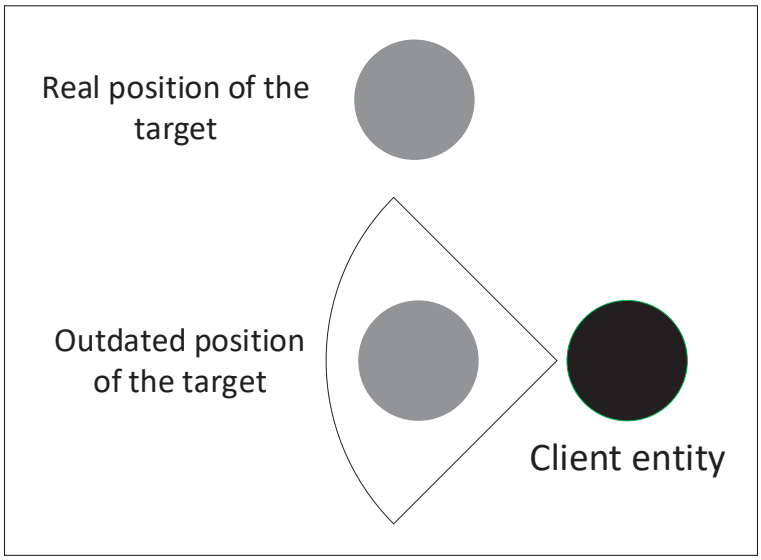

Figure 3.5 Undesirable inconsistency happens when the position of the target is updated with low frequency

In each state of the game, depending on the circle-shape area in which the character entity is located, if the corresponding action to that area is not on cooldown, the client entity receives updates from the position of the character entity. In each state, our model checks if the corresponding action to the formed area is on cooldown or not. If it is on cooldown, the area is not considered by the model until the action becomes available again.

The position of a character entity may be within different frequency areas of the client entity's AoE simultaneously. If all the corresponding actions to these areas are available in the current state and the areas overlap, the model always considers the area with the highest frequency. For instance, if a character entity is within the range of the axe-auto attack and skill with a bigger range, the model considers the associated frequency of the area formed by the axe-auto attack.

3.2.2 Updates based on the AoE of other entities

In our model, the AoE of the client entity always takes precedence over the AoE of other character entities if their position is within the AoE of the client entity. Therefore, the client

entity receives updates based on the AoE of other character entities if their position is not within the AoE of the client entity.

The actions performed by the client entity do not affect the skill entities. Therefore, the client entity always receives updates from a skill entity based on the AoE of the skill entity, regardless of the skill entity's position, if it is within the AoE of the client entity or outside it.

The model considers different parameters to determine the intervals between the updates the client entity should receive from the position of other entities based on their AoE. It takes into account the distance of the client entity to the edge of the other entity's AoE, the size of the other entity, the bit rate, and the resolution chosen by the player on the cloud gaming platform. The following formula is used to calculate the intervals in milliseconds between each update for a given entity:

$$\text{Interval}(\text{E}, \text{CE}) = Min\left(\left\lfloor 1+\left[(\text{w}_{\text{DI}} \times \text{DI}(\text{E}, \text{CE}))+(\text{w}_{\text{PI}} \times \text{PI})\right] \times \text{RBD} \right\rfloor \times \text{Interval}_{\text{min}}, \text{Interval}_{\text{max}}\right) \tag{3.1}$$

where CE denotes the client entity, E denotes the other entity under consideration. The other parameters are described in the following paragraphs:

1. DI(E, CE): The *distance impact* parameter is dynamically calculated based on the position of the client entity relative to the AoE of the other entity considered in each state of the game. The *distance impact* is calculated as follows:

$$\text{DI}(\text{E}, \text{CE}) = \left| \text{Distance}(\text{E}, \text{CE}) - \text{Radius}(\text{E}) \right| \tag{3.2}$$

   where Radius(E) is the radius of the other entity's AoE. If the other entity is a character entity, the radius of the AoE is equal to the radius of action with the biggest range among all the actions that the character entity can perform. If this action is on cooldown and not available, the next biggest range among all the actions is taken into account by the model.

Moreover, the distance between the client entity and the other entity is calculated as follows:

$$\text{Distance}(\text{E}, \text{CE}) = \sqrt{(\text{E}_\text{x} - \text{CE}_\text{x})^2 + (\text{E}_\text{y} - \text{CE}_\text{y})^2 + (\text{E}_\text{z} - \text{CE}_\text{z})^2} \tag{3.3}$$

where $\text{E}_\text{x}, \text{E}_\text{y}, \text{E}_\text{z}$ ($\text{CE}_\text{x}, \text{CE}_\text{y}, \text{CE}_\text{z}$) are the positions in x, y, and z, respectively, of entity E (and client entity (CE)). As the distance between two entities is getting smaller, the chance that they start interacting and be affected by each other is raised. Thus, distance is a critical factor to consider in our model.

The most critical situation occurs when the position of the client entity is near or on the edge of the other entity's AoE. In this situation, the client entity needs to receive enough updates so that the player can make a decision and start to react instantaneously. On the contrary, the client entity receives fewer updates whenever it moves away from the edge of the AoE of the other entity, no matter if it is outside the AoE or inside it. This is the reason why DI increases when the distance decreases within the radius.

Figures 3.6 and 3.7 show the impact of the distance on how frequently the client entity receives updates considering its distance to the AoE of a character entity and a skill entity respectively. The refresh interval is proportional to DI, and thus the frequency of updates is inversely proportional to DI. Client entity receives updates with short intervals and high frequency from the position of the other entity when its position is near or on the edge of the other entity's AoE.

2. PI: The *pixel impact* is another parameter in the model. From the visual effect point of view, the larger entities cover more space of the game scene. Therefore, updating their position could be done at a lower frequency than smaller entities since precision is not very important due to their big size. Thereby, it makes sense to emphasize small entities more.

   In our implementation, pixel impact is a constant value read from an XML file, an excerpt of which can be seen in Figure 3.8. Each entity has a specific pixel impact. We determined these values based on the size of each entity and how they fit in our formula to get the desired update intervals. As we are using a weighted sum of the *distance impact* and the *pixel*

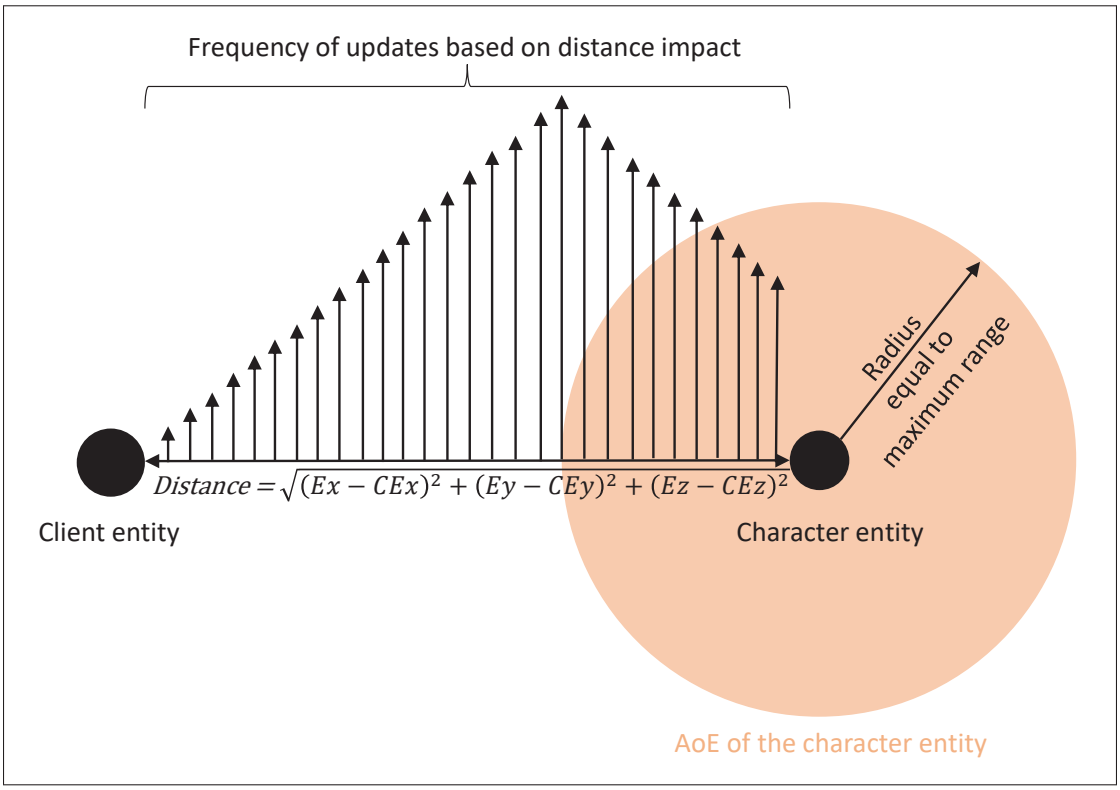

Figure 3.6 Update frequencies based on the distance of the client entity to the edge of the AoE of a character entity

impact in Equation 3.1, this is very important that two factors (PI and DI) have comparable scales. Since the value of DI is calculated automatically, we need to decide the PI's value in a way that respects the scale comparability of the two factors.

3. w_{DI} and w_{PI}: Our formula is a weighted sum of pixel impact and distance impact in which *distance impact weight* and *pixel impact weight* are the weights assigned to them accordingly. We have:

$$0 < w_{DI} < 1 \quad \text{and} \quad w_{PI} = (1 - w_{DI}) \tag{3.4}$$

w_{DI} value is read from the XML file, and w_{PI} is calculated as the 1 - w_{DI} since they are the weights for pixel impact and distance impact, and the weighted sum should be equal to 1. By changing the w_{DI}'s value, we specify the effect of each pixel impact and distance impact on the intervals between each update. Based on the decision of which factor has the

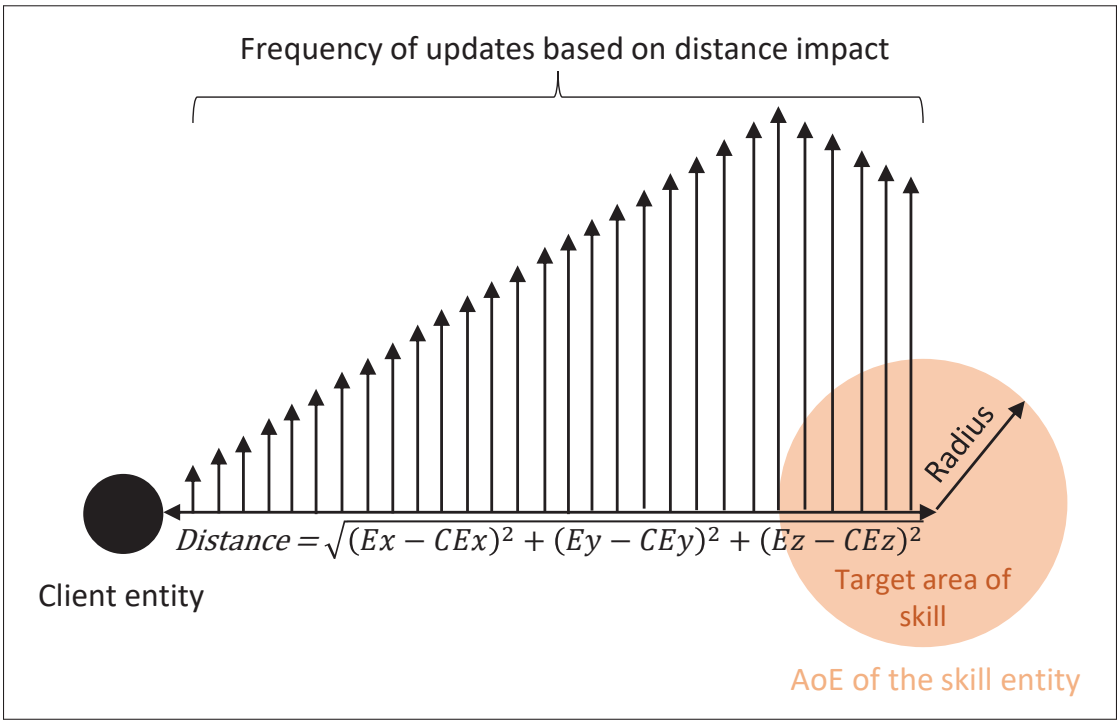

Figure 3.7 Update frequencies based on the distance of the client entity to the edge of the AoE of a skill entity

most impact on determining the intervals between updates, w_{DI} can get different values, and accordingly, the value of w_{PI} will be changed.

The values assigned for PI, w_{DI}, and accordingly w_{PI} (Figure 3.8), are the first intuitive proposal of values for these parameters. In the next chapter, we use the current values to show that the model can give us very positive results. However, they are not necessarily the best values to be used in equation 3.1. To get the most suited values, more simulations and optimizations should be performed.

4. RBD: For the *relative bit rate difference*, we first calculate the ratio of the player's bit rate budget (BB) and the standard bit rate (SB) in the cloud gaming platform for the best video quality. The smaller the bit rate budget value compared to the standard bit rate, the smaller the ratio of them would be. Therefore, we subtract the obtained ratio from 1 to calculate the relative bit rate difference. The more significant the relative bit rate difference, the lower the

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <userSettings>
    <setting name="AvatarPixelImpact" serializeAs="String">
      <value>1</value>
    </setting>
    <setting name="FirestormPixelImpact" serializeAs="String">
      <value>3</value>
    </setting>
    <setting name="OrisonOfHealingPixelImpact" serializeAs="String">
      <value>3</value>
    </setting>
    <setting name="FireballPixelImpact" serializeAs="String">
      <value>1</value>
    </setting>
    <setting name="DistanceImpactWeight" serializeAs="String">
      <value>0.6</value>
    </setting>
    <setting name="StandardBitrateBudget_720" serializeAs="String">
      <value>10</value>
    </setting>
    <setting name="StandardBitrateBudget_1080" serializeAs="String">
      <value>20</value>
    </setting>
  </userSettings>
</configuration>
```

Figure 3.8 An excerpt of the XML file with the determined pixel impact value for different objects in our exemplar game

frequency of position updates the client entity receives.

$$\mathrm{RBD} = 1 - \left(\frac{\mathrm{BB}}{\mathrm{SB}}\right) = \left(\frac{\mathrm{SB} - \mathrm{BB}}{\mathrm{SB}}\right) \tag{3.5}$$

Different players connect to the cloud gaming platform with different connection speeds leading to different bit rates. The encoder generates a higher data rate for better video quality and higher resolution. The higher the generated data rate, the higher is the required bit rate to play the game. For instance, if a player wants to play a game with the highest

quality in 1080P resolution, he needs a bit rate of nearly 20 Mbps. If the connection can provide the required bit rate, ARCODE does not need to perform any action. However such bit rate is quite high and not all players can support it.

Nevertheless, if the player connects with a bit rate of 5 Mbps, ARCODE will perform optimizations so that the player can play the game with the best possible video quality at this lower video bit rate. Our model considers the bit rate budget and the resolution chosen by player to manage the frequency of updates and provide a better visual quality without compromising the gaming experience compared to the traditional approach.

5. $\text{Interval}_{\text{min}}$: This is the shortest interval between each update in our model. If the relative bit rate difference is equal to 0, equation (3.1) provides the shortest interval between each update. For this reason, in equation (3.1), the number 1 is added to the result obtained from calculating the effect of pixel impact, distance impact, and relative bit rate difference, and then multiplied by $\text{Interval}_{\text{min}}$, measured in milliseconds.

6. $\text{Interval}_{\text{max}}$: This is the longest interval between each update to be received by the client entity from the position of other entities based on their AoE. This parameter is taken into account by the model so that if the client entity is too far away from any other entity of any size, it does not receive updates with intervals longer than $\text{Interval}_{\text{max}}$.

In this chapter, we thoroughly explained how our proposed model adjusts the updates in the combat area of MMORPGs to reduce the amount of motion and changes in the scene to assist the process of video encoding by the encoder in cloud gaming platforms.

In the next chapter, we first present our experimental setup for the evaluation. We then explain our evaluation methodology followed by the experimental results based on different evaluation scenarios in our exemplar game. We finally conclude the next chapter with the conclusion and future work.

CHAPTER 4

PERFORMANCE EVALUATION

In this chapter, we present our experimental setup in the first section. In the second section, we explain our validation methodology. The third section is devoted to our experimental parameters. Finally, in the forth and final section, we present our experimental results, followed by analyzing and discussing the results, conclusions, and future work.

4.1 Experimental Setup

4.1.1 MMOnkey framework

The adopted framework was presented with details in Section 1.3. The framework has two main parts. The MMOnkey server comprises our proposed model and is responsible for managing the updates. Our model is built on top of the square tile interest management that currently exists on the MMOnkey server (Section 1.3.6). The functionality of interest management was explained in Section 1.2.4. We increase the size of the interest area considered for each client entity to the size of the whole game world. This way, players can see all the entities in the game scene. Our model then controls the updates from the interest management before they are sent to the client entity.

The client-side comprises our exemplar game developed in Unity 3D. The game has three scenes: MainMenu, CharacterCreation, and GameWorld. The game starts from the MainMenu scene, where the player can connect to the MMOnkey server on the host machine, which is also our game stream PC. The second scene is CharacterCreation, where the player can build his avatar using the client user interface (UI) shown in Figure 4.1.

In MMOnkey, before each player's avatar is spawned into the game world, the player needs to build his or her character in terms of the name, team, weapons, and skills the character can use during gameplay. The client UI was designed previously in the framework for this purpose. As

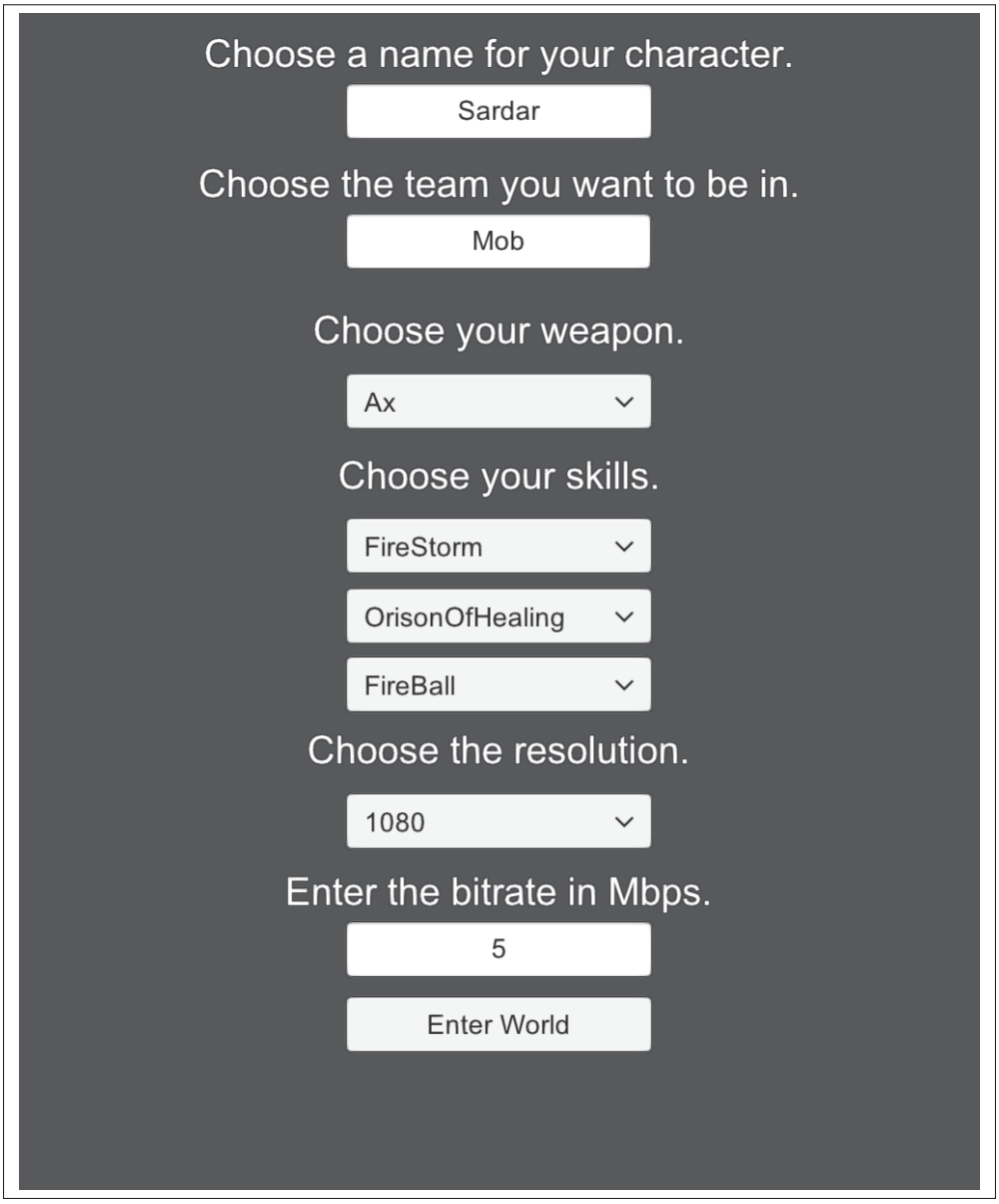

Figure 4.1 Client UI in MMOnkey showing the CharacterCreation scene

illustrated in Figure 4.1, we added two new fields (Choose the resolution, Enter the bitrate) to the UI to get the information about the bit rate and the resolution chosen by the player for the game streaming. This information is sent to the MMOnkey's server application to be considered by our model to adjust the update rates. Finally, after the character creation is done, the player can enter the last scene and interact with the game world through his or her avatar.

4.1.2 Game stream setup

To integrate the framework into a game stream protocol, we chose Moonlight for the client. Moonlight is an open-source version of the NVIDIA GAMESTREAM[11] client, which allows players to play remotely on compatible devices. Moonlight is implemented in order to be able to run on different operating systems such as Windows, Mac, and Linux. Figure 4.2 illustrates the setup and components we used for the game stream purpose.

The PC which is considered as the game stream PC server, needs to have the GeForce Experience software installed. In the GeForce Experience software, the shield option allows us to add our game and play it remotely on the shield devices or the devices where have the Moonlight client installed. To take advantage of this feature, the game stream PC requires to be equipped with an NVIDIA GeForce GTX/RTX 600+ series GPU, or NVIDIA Quadro GPU[12]. The encoding is performed by an independent section called NVIDIA video encoding (NVENC) which is a feature of the graphics card that performs video encoding, and it supports both H.264 and HEVC CODECs. Hardware encoding removes the load of video encoding from CPU, and it is done directly by the GPU.

In order to make our exemplar game ready for streaming, after building our game application and getting the executable file from Unity 3D, we need to add it to the shield window in the GeForce Experience software. To do this, we open the GeForce Experience software on the game stream PC, we go into settings, choose the shield tab, enable the GameStream option, and finally add our game to the launcher window.

Our game stream PC has a GeForce RTX graphic card. According to Burnes (2019), GeForce RTX GPUs require 15 percent less bit rate for the same video quality in comparison to the previous generation of Pascal GPUs for video streaming, using H.264. GeForce RTX GPUs can stream with better image quality compared to x264 Fast, and much the same as x264 Medium preset.

---

[11] https://www.nvidia.com/en-us/shield/support/shield-tv/gamestream/

[12] https://github.com/moonlight-stream/moonlight-docs/wiki/Setup-Guide

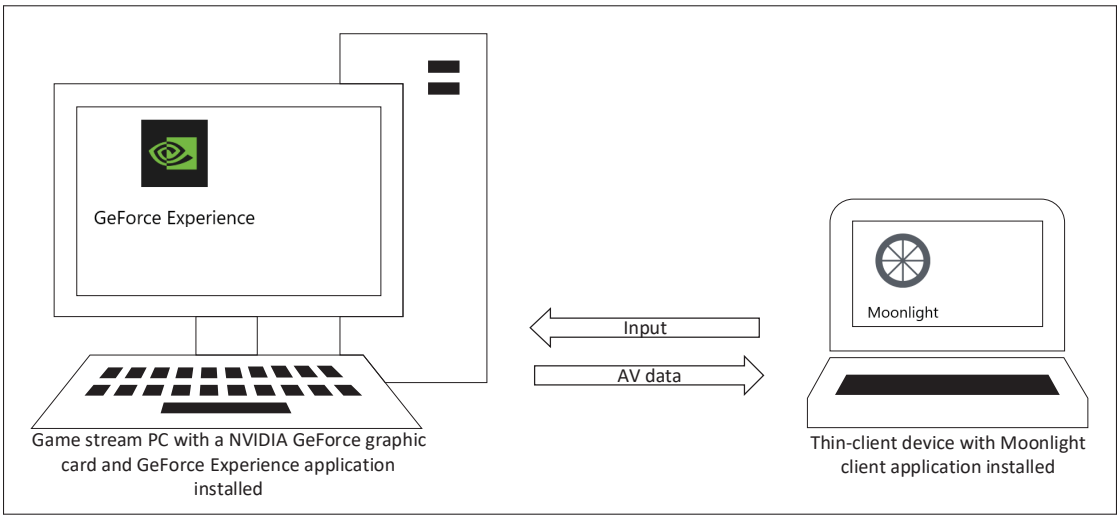

Figure 4.2 The game stream setup with GeForce Experience application installed on the host machine and Moonlight application installed on the client device

Moonlight itself is only the client, but it can control several encoding parameters such as bit rate, resolution, frame rate, and video CODEC via a real time streaming protocol (RTSP) handshake. Figure 4.3 shows a part of the Moonlight client settings panel and the options we control for the evaluation[13]. The parameters of interest in this work are the video resolution, the frame rate, the video bit rate, and finally, the video CODEC option that allows us to choose between H.264 and H.265 (aka HEVC) as the target video encoder for our evaluation.

When we open the Moonlight application on the thin-client device, the game stream PC on the same network usually appears automatically in the PC list. If it does not show up after a few seconds, we can add it manually by entering the IP address of the game stream PC. The complete setup guide can be found on Moonlight's setup guide page[14].

[13] The image capture has altered to make it clearer and more readable.

[14] https://github.com/moonlight-stream/moonlight-docs/wiki/Setup-Guide

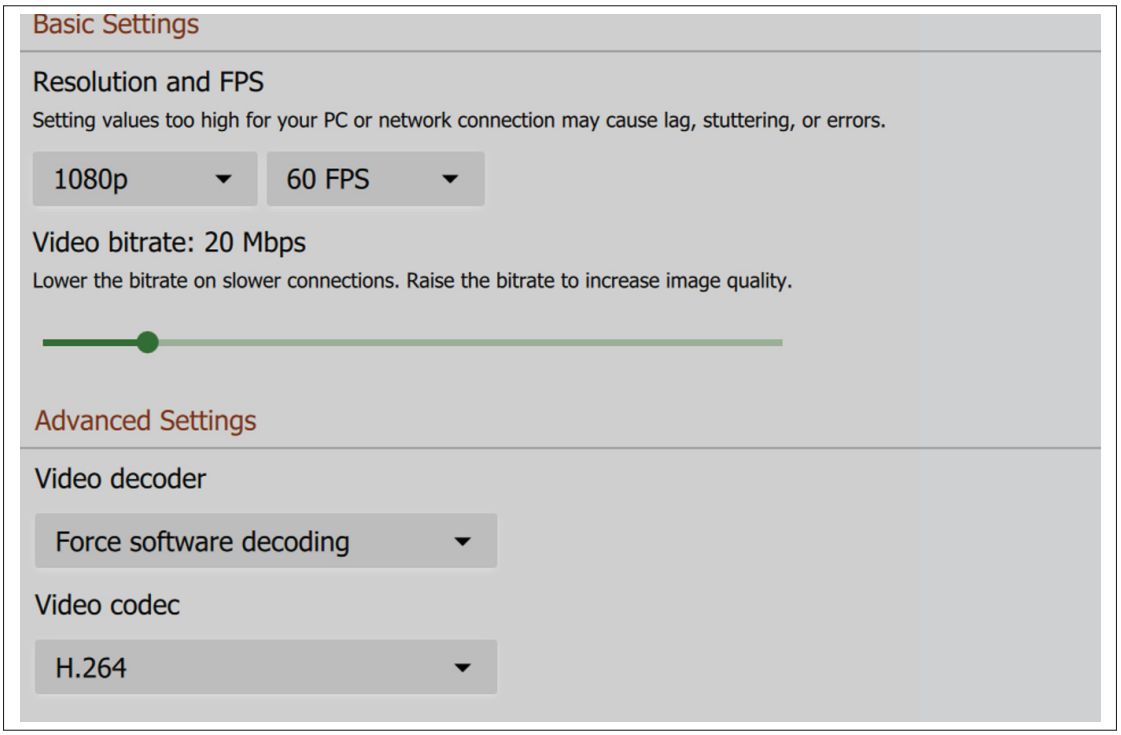

Figure 4.3 A part of the Moonlight client settings that shows the encoding parameters we can change for our evaluation

4.2 Performance Evaluation Methodology

To see if our model provides superior performance compared to the baseline[15], we need to measure the bit rate the proposed ARCODE saves compared to the baseline for the same video quality. For this purpose, we use a metric inspired from the Bjøntegaard model (Bjøntegaard, 2001). The Bjøntegaard model compares two methods by either measuring the bit rate savings for the same quality (BD-Rate) or the video quality improvement, in our case measured in PSNR, for the same bit rate (BD-PSNR).

As it is discussed in Tan, Weerakkody, Mrak, Ramzan, Baroncini, Ohm & Sullivan (2016), different objective video quality metrics such as PSNR, structural similarity index (SSIM) and video quality metric (VQM) are proposed. Nonetheless, the most popular one for evaluating coding performance remains the PSNR.

[15] The unaltered game is referred as the baseline approach.

The PSNR is the ratio between the maximum possible power of the signal and the power of the distortion, or the reconstruction error. It is commonly used to measure the quality of the compressed video compared to the original one.

As illustrated in Figure 4.4, in the video coding evaluation methodology, researchers compare the performance of various video CODECs by encoding the same content and comparing the reconstructed video by each to the original content. For each of the CODECs taken into account for the comparison, the Bjøntegaard model requires various PSNR and bit rate data points (at least four) to evaluate the coding efficiency of the CODECs. Therefore, the same original video is encoded with various bit rates (or quantization paremeter values), and the PSNR is computed by taking into account the original video frames and the reconstructed ones.

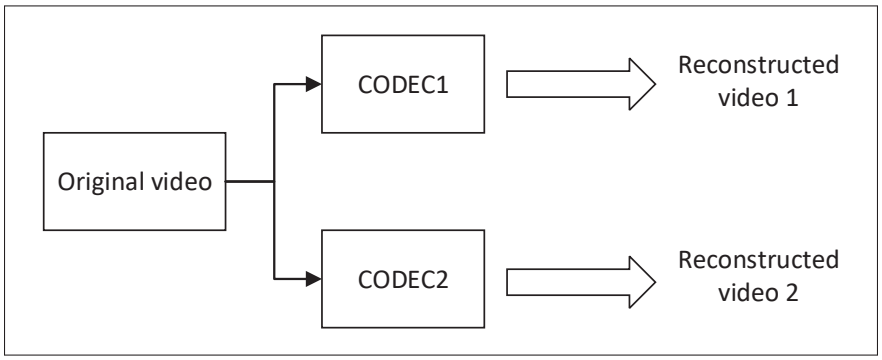

Figure 4.4 In the common video evaluation methodology the same video is encoded by different CODECs to compare the coding performance of the CODECs

In our case (see Figure 4.5), there is not a single original content that is encoded with various video encoders, but variations of the same content encoded with the same encoder. We thus compare how well different original contents, which are assumed to be visually equivalent in terms of quality and usefulness, are coded using the same CODEC. Assuming that the contents generated by the baseline and the proposed model are of equivalent visual quality, we evaluate which is more efficiently encoded under four target bit rates. This is a departure of the traditional use of the Bjøntegaard model, but it is no less relevant. For this reason we call our evaluation methods *similar reference* (SR)-BD-Rate and SR-BD-PSNR.

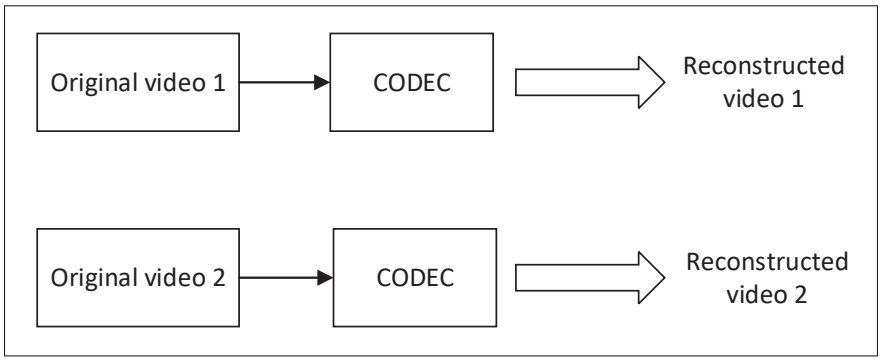

Figure 4.5 In our evaluation methodology, two different videos are encoded using the same CODEC to see which content is encoded more efficiently

Each video pair consists of an original and a reconstructed video. The original video is recorded from the game stream PC. The reconstructed video sequence comprises the video frames extracted from Moonlight after the video contents are sent in a compressed format over the Internet from the game stream PC to the thin-client device. The original video is recorded using OBS studio[16]. OBS is open-source software for video recording and video streaming. We only use the video recording option since the video streaming is done using the game stream setup discussed in 4.1.2. The reconstructed video is the video frames of the raw H.264 or HEVC elementary stream extracted from Moonlight. We use Qt Creator IDE[17] to run Moonlight's source code as an application and modify its script to extract the video frames. In Figure 4.5, the first and the second video pairs are the contents generated by the baseline, and our proposed model, respectively.

After capturing each video pair, we have to align in time the two videos in order to be able to compute the PSNR by taking into account the frames of the reconstructed video with respect to those of the original one. We use FFmpeg[18] to extract the raw YUV video frames of each video segment. FFmpeg is a multimedia framework and a command-line tool that offers many options to convert multimedia files. We wrote a program in C that starts from a desirable extracted frame

[16] https://obsproject.com/

[17] https://www.qt.io/download

[18] https://www.ffmpeg.org/

of the reconstructed video and finds the similar one in the original video. After the alignment part is done, the PSNR is computed, as we explain in the following subsection.

4.2.1 PSNR calculation

To compute the PSNR, first we need to compute the mean squared error (MSE). MSE is computed between the reconstructed video and the original one as follows:

$$\text{MSE} = \frac{\sum_{i=0}^{M-1} \sum_{j=0}^{N-1} (I(i,j) - J(i,j))^2}{M \cdot N}. \tag{4.1}$$

In Equation 4.1, M is the width of an image or a frame of a video, and N is the height. MSE is computed using the pixels of the reconstructed frame J with respect to those of the original frame I. For a three-color components video frame, if we want to calculate the PSNR_Y, we use only the Luma pixels and the product of the frame's width and height at the denominator. If we want both Luma and Chroma components of a video frame to calculate the PSNR_YCbCr, the sum is performed over all pixels of each component, and the denominator is $1.5(M \cdot N)$ because the chroma components are subsampled horizontally and vertically by a factor of 2 in the 4:2:0 sampling format used. We consider both Luma and Chroma components in our evaluation. After computing the MSE, PSNR is computed as:

$$\text{PSNR} = 10 \cdot \log_{10} \frac{(2^B - 1)^2}{\text{MSE}} \tag{4.2}$$

where B is the bit depth, which is typically 8 for non high dynamic range (HDR) content:

$$(2^B - 1) = 255 \tag{4.3}$$

It is first calculated for each video frame separately and then averaged over all frames of a video sequence.

4.2.2 Evaluation Metrics

Two different modifications of the Bjøntegaard metrics are computed to evaluate the coding performance of a system. Although the formula is the same as the original metrics, it is not performed on the same original content while it used with the same codec. We can evaluate a difference in bit rate (SR-BD-Rate) for the same video quality or a difference in quality (SR-BD-PSNR) for the same bit rate:

1. SR-BD-Rate: The average bit rate difference in percent (%) over the range of all PSNR values. The difference in bit rate is calculated based on two interpolation curves for both reconstructed videos, each one coded with the same encoder. The interpolation can be done to find bit as a function of PSNR. Referring to Figure 4.5, for the first reconstructed video (1):

$$\mathrm{bit}_1(\mathrm{PSNR}) = a_1 + b_1 \times \mathrm{PSNR} + c_1 \times \mathrm{PSNR}^2 + d_1 \times \mathrm{PSNR}^3 \tag{4.4}$$

and for the second reconstructed video (2):

$$\mathrm{bit}_2(\mathrm{PSNR}) = a_2 + b_2 \times \mathrm{PSNR} + c_2 \times \mathrm{PSNR}^2 + d_2 \times \mathrm{PSNR}^3 \tag{4.5}$$

The polynomial coefficients (a, b, c, d) are determined from a curve that passes through all data points.

The difference of two polynomial integrals in a given interval (from minimum PSNR to maximum PSNR) is divided by the interval to calculate the SR-BD-Rate. Due to the large scale of the bit rate, the logarithmic bit rate scale is used.

$$\left[\exp\left(\frac{1}{\mathrm{PSNR}_{\max} - \mathrm{PSNR}_{\min}} \int_{\mathrm{PSNR}_{\min}}^{\mathrm{PSNR}_{\max}} \mathrm{bit}_2(\mathrm{PSNR})\, d\,\mathrm{PSNR} - \int_{\mathrm{PSNR}_{\min}}^{\mathrm{PSNR}_{\max}} \mathrm{bit}_1(\mathrm{PSNR})\, d\,\mathrm{PSNR}\right) - 1\right] \times 100 \tag{4.6}$$

2. SR-BD-PSNR: The average quality difference for PSNR in dB over the range of all bit rates. The polynomial for the first and second reconstructed videos are considered as:

$$\mathrm{PSNR}_1(\mathrm{bit}) = a_1 + b_1 \times \mathrm{bit} + c_1 \times \mathrm{bit}^2 + d_1 \times \mathrm{bit}^3 \tag{4.7}$$

$$\mathrm{PSNR}_2(\mathrm{bit}) = a_2 + b_2 \times \mathrm{bit} + c_2 \times \mathrm{bit}^2 + d_2 \times \mathrm{bit}^3 \tag{4.8}$$

To calculate the SR-BD-PSNR, the difference of two polynomial integrals in given interval (from minimum bit rate to maximum bit rate) is calculated as:

$$\int_{\mathrm{bit}_{\min}}^{\mathrm{bit}_{\max}} \mathrm{PSNR}_2(\mathrm{bit})\, d\,\mathrm{bit} - \int_{\mathrm{bit}_{\min}}^{\mathrm{bit}_{\max}} \mathrm{PSNR}_1(\mathrm{bit})\, d\,\mathrm{bit} \tag{4.9}$$

In this section, we explained our validation methodology. In the following section, we present our experimental parameters.

4.3 Experimental Parameters

Different tests are performed to evaluate the performance of our model. Each test, alongside different chosen settings on Moonlight, such as the bit rate, resolution, and CODECs, are presented in the next section. The general details of the tests and the information related to the evaluation are provided in the following paragraphs.

NPCs. Because having real players using the game and control their avatars is not always possible, and producing consistent gameplay in different rounds is not an easy task, we implemented an interactive behaviour for the NPCs to simulate the combat area. Using the NPCs with interactive behaviour is one logical way of producing consistent gameplay in each round. In our exemplar game, NPCs are able to move in different directions. If they do not have a target within their AoE, they find a target, move to the target, and when the target is within the range, they use their weapons and skills to interact with the target.

Skills. For the evaluation, we consider three different skills to be used by NPCs during the gameplay: orison of healing, firestorm, and fireball. Firestorm and fireball are used to damage the health of any opponent entity within their target area. The orison of healing is a skill that is used to heal any friendly entity inside its target area. The implemented functionality for each skill is inspired by real MMORPGs. Based on the functionality of each skill, the corresponding skill entity has its movement implemented and its visual effect designed by us. In the 3D scene, the firestorm movement starts from the top of the target area and moves vertically, downwards to the target's position. The fireball movement starts from the position of the action source (the entity that cast the fireball skill) and moves horizontally towards the target. Finally, the orison of healing movement starts from the ground of a target area and moves back and forth vertically.

Update Intervals. For the baseline, the interval between each update is 30 ms (the update rate of 33 Hz). Therefore, each 30 ms, the client entity receives one update from other entities in the game scene. In ARCODE, the interval between each update the client entity receives from each entity type during gameplay is different. As an example, the interval and the frequency of updates are presented in Table 4.1, for 1080P resolution with 2 Mbps bit rate as the available bit rate chosen by the player on Moonlight. The required bit rate by Moonlight for the best visual quality in 1080P resolution is 20 Mbps. The update intervals are the same for the 720P resolution with 1 Mbps available bit rate chosen by the player when a 10 Mbps bit rate is required for the best visual quality.

Please note that we only illustrate the intervals between each update the client entity receives from other entities, based on their AoE, when the position of the client entity is near or on the edge of the other entities AoE. The interval between each update increases as the distance of the client entity to the edge of the other entities AoE increases. In ARCODE, the shortest and the longest interval between each update the client entity receives from other entities are 30 ms and 600 ms, respectively.

Video Sequence Duration. Each video sequence captured from Moonlight comprises about 3600 frames for 60 seconds gameplay in 60 frames per second (FPS). As we mentioned earlier,

Table 4.1 Update rates the client entity receives using our model (Resolution and bit rate chosen by player are 1080P and 2 Mbps, respectively)

Updates from character entities based on the client entity's AoE			
Action type	Distance range	Update intervals	Update rates
AxeAutoAttack	0 – 4 (radius of the area)	30 ms	33 Hz
Firestorm	0 – 11 (radius of the area)	60 ms	16 Hz
Fireball	0 – 20 (radius of the area)	60 ms	16 Hz
OrisonOfHealing	0 – 9 (radius of the area)	60 ms	16 Hz
Updates from character entities based on the character entities' AoE			
Action type	Distance range	Update intervals	Update rates
AxeAutoAttack	3 – 4 (Within AoE)	57 ms	17 Hz
	4 – 5 (On the edge of AoE)	40 ms	25 Hz
	5 – 6 (Outside AoE)	57 ms	17 Hz
BowAutoAttack	8 – 9 (Within AoE)	57 ms	17 Hz
	9 – 10 (On the edge of AoE)	40 ms	25 Hz
	10 – 11 (Outside AoE)	57 ms	17 Hz
Firestorm	10 – 11 (Within AoE)	57 ms	17 Hz
	11 – 12 (On the edge of AoE)	40 ms	25 Hz
	12 – 13 (Outside AoE)	57 ms	17 Hz
Fireball	19 – 20 (Within AoE)	57 ms	17 Hz
	20 – 21 (On the edge of AoE)	40 ms	25 Hz
	21 – 22 (Outside AoE)	57 ms	17 Hz
OrisonOfHealing	8 – 9 (Within AoE)	57 ms	17 Hz
	9 – 10 (On the edge of AoE)	40 ms	25 Hz
	10 – 11 (Outside AoE)	57 ms	17 Hz
Updates from skill entities based on the skill entities' AoE			
Skill entity	Distance range	Update intervals	Update rates
Firestorm	4 – 5 (Within AoE)	78 ms	12 Hz
	5 – 6 (On the edge of AoE)	62 ms	16 Hz
	6 – 7 (Outside AoE)	78 ms	12 Hz
Fireball	19 – 20 (Within AoE)	57 ms	17 Hz
	20 – 21 (On the edge of AoE)	40 ms	25 Hz
	21 – 22 (Outside AoE)	57 ms	17 Hz
OrisonOfHealing	4 – 5 (Within AoE)	78 ms	12 Hz
	5 – 6 (On the edge of AoE)	62 ms	16 Hz
	6 – 7 (Outside AoE)	78 ms	12 Hz

the game has three scenes. To avoid redundant frames from the first and second scenes, we start to stream the game from the third scene (GameWorld). Therefore, the captured video from Moonlight only comprises the contents generated during the gameplay.

4.4 Experimental Results

Our experimental results are presented in this section. We consider various scenarios in our exemplar game to show the superiority of our model compared to the baseline. Specific details of each test are explained before the test results are presented. Please note that the images of each test taken from our exemplar game are presented in Appendix I.

4.4.1 Main scenario

Our main scenario is the simulation of the common combat area of most MMORPGs where a large number of players interact through their avatars in a huge combat area. In this scenario, the battle is concentrated in most regions of the combat area. Accordingly, changes in the scene are distributed in different parts of the scene.

For this scenario, we present the results by changing different in-game properties such as the camera rotation and translation. In terms of video settings chosen on Moonlight, we take into account different resolutions and different target video CODECs for streaming.

We first compare the performance of our proposed model with the baseline when the client entity is idle and without any camera rotation and translation in the scene. In each round, the client entity is spawned in a fixed position, and the battle takes place in front of it. The test is done in 1080P and 720P resolutions with both H.264 and H.265 video CODECs as the target CODEC for game streaming chosen on Moonlight.

4.4.1.1 Test results for H.264 at 1080P and 720P

Referring to Figures 4.6 and 4.7, the results show a significant bit rate reduction in ARCODE compared to the baseline. The information in Table 4.2, and 4.3 show that for the same visual quality in a 1080P and a 720P resolution, having H.264 as the target CODEC, ARCODE can reduce the video bit rate by about 37.0% compared to the baseline. In other words, with the same video bit rate, ARCODE improves the visual quality of the video by almost 2.5 dB. It

means that when the encoder wants to encode the video contents generated by our model, it produces less distortion than encoding the generated contents in the baseline. Accordingly, the player receives a video sequence with better visual quality in our model compared to the baseline with comparable bandwidth. A comparison of the image quality in baseline and ARCODE is demonstrated in Figure-A I-1 and Figure-A I-2, for the resolution of 1080P and 720p respectively.

Please note that the bit rate of the transferred video is always lower than the target video bit rate chosen on Moonlight in all cases (see Table 4.2 and 4.3). This is partly due to the reservation of part of the bit rate for controlling stream data and audio data. The bit rate also includes forward error correction (FEC) overhead which is not video data. The other reason is that the encoder does not always consume all of the allowed bit rate to encode the video frames.

Table 4.2 The calculated SR-BD-Rate and SR-BD-PSNR under various target bit rates chosen on Moonlight for our proposed model and the baseline (H.264 - 1080P)

Baseline (H.264 - 1080P)		
Target bit rate on Moonlight (bps)	Bit rate (bps)	PSNR (dB)
2,000,000	1,260,060	31.60
4,000,000	2,465,296	36.78
6,000,000	3,716,505	38.75
8,000,000	5,373,133	40.31
ARCODE (H.264 - 1080P)		
Target bit rate on Moonlight (bps)	Bit rate (bps)	PSNR (dB)
2,000,000	1,117,303	33.24
4,000,000	2,339,456	38.95
6,000,000	3,464,753	40.49
8,000,000	5,013,130	41.78
	SR-BD-Rate (%)	**SR-BD-PSNR (dB)**
	-36.2	2.38

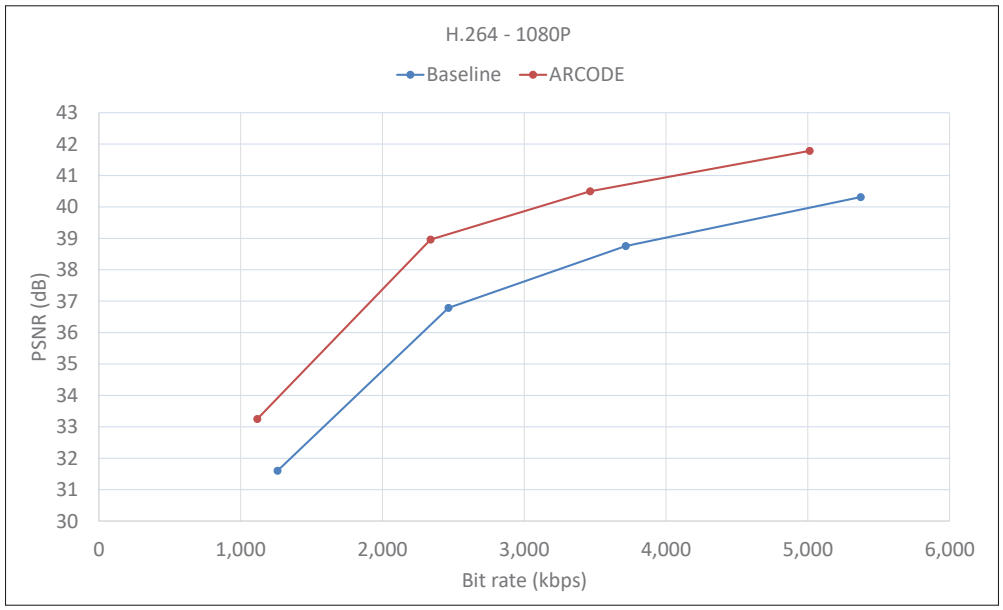

Figure 4.6 Comparison of the proposed model with the baseline for the first scenario when the client entity is idle in the scene. Without camera rotation and translation in the scene (H.264 - 1080P)

Table 4.3 The calculated SR-BD-Rate and SR-BD-PSNR under various target bit rates chosen on Moonlight for our proposed model and the baseline (H.264 - 720P)

Baseline (H.264 - 720P)		
Target bit rate on Moonlight (bps)	Bit rate (bps)	PSNR (dB)
1,000,000	606,140	30.67
2,000,000	1,345,719	35.37
3,000,000	2,133,963	37.68
4,000,000	2,711,185	38.77
ARCODE (H.264 - 720P)		
Target bit rate on Moonlight (bps)	Bit rate (bps)	PSNR (dB)
1,000,000	565,883	32.27
2,000,000	1,296,053	37.71
3,000,000	2,040,045	39.74
4,000,000	2,575,955	40.76
	SR-BD-Rate (%)	**SR-BD-PSNR (dB)**
	-37.5	2.44

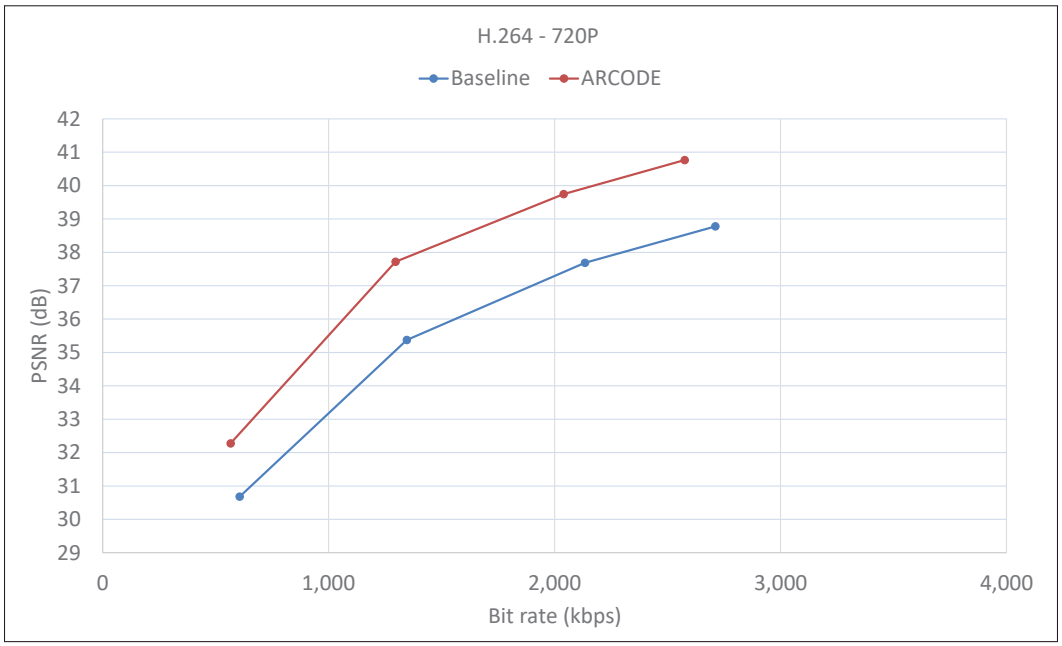

Figure 4.7 Comparison of the proposed model with the baseline for the first scenario when the client entity is idle in the scene. Without camera rotation and translation in the scene (H.264 - 720P)

4.4.1.2 Test results for H.265 at 1080P and 720P

Based on the results shown in Figures 4.8 and 4.9, it is quite obvious that our model leads to a significant bit rate saving compared to the baseline for the same visual quality, having H.265 as the target encoder. The information in Tables 4.4, and 4.5 shows that for the same visual quality in 1080P and 720P resolution, ARCODE can reduce the video bit rate by about 36.0% compared to the baseline. A comparison of the image quality in baseline and ARCODE is demonstrated in Figure-A I-3 and Figure-A I-4, for the resolution of 1080P and 720p respectively.

Table 4.4 The calculated SR-BD-Rate and SR-BD-PSNR under various target bit rates chosen on Moonlight for our proposed model and the baseline (H.265 - 1080P)

Baseline (H.265 - 1080P)		
Target bit rate on Moonlight (bps)	Bit rate (bps)	PSNR (dB)
2,000,000	1,429,479	34.07
4,000,000	2,704,546	37.29
6,000,000	3,899,796	38.97
8,000,000	5,587,132	40.52
ARCODE (H.265 - 1080P)		
Target bit rate on Moonlight (bps)	Bit rate (bps)	PSNR (dB)
2,000,000	1,399,762	36.11
4,000,000	2,609,504	39.27
6,000,000	3,656,600	40.62
8,000,000	5,189,492	41.95
	SR-BD-Rate (%)	**SR-BD-PSNR (dB)**
	-36.8	2.06

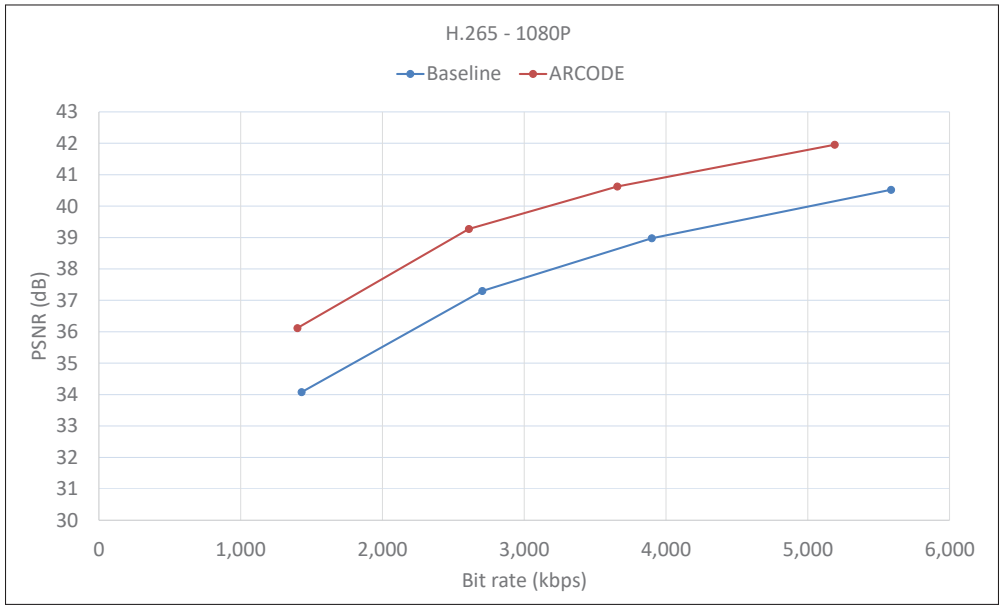

Figure 4.8 Comparison of the proposed model with the baseline for the first scenario when the client entity is idle in the scene without camera rotation and translation in the scene (H.265 - 1080P)

Table 4.5 The calculated SR-BD-Rate and SR-BD-PSNR under various target bit rates chosen on Moonlight for our proposed model and the baseline (H.265 - 720P)

Baseline (H.265 - 720P)		
Target bit rate on Moonlight (bps)	Bit rate (bps)	PSNR (dB)
1,000,000	666,031	31.81
2,000,000	1,436,486	36.30
3,000,000	2,033,138	37.99
4,000,000	2,817,334	39.53
ARCODE (H.265 - 720P)		
Target bit rate on Moonlight (bps)	Bit rate (bps)	PSNR (dB)
1,000,000	645,967	33.79
2,000,000	1,381,059	38.43
3,000,000	1,942,634	39.99
4,000,000	2,661,753	41.49
	SR-BD-Rate (%)	**SR-BD-PSNR (dB)**
	-36.3	2.30

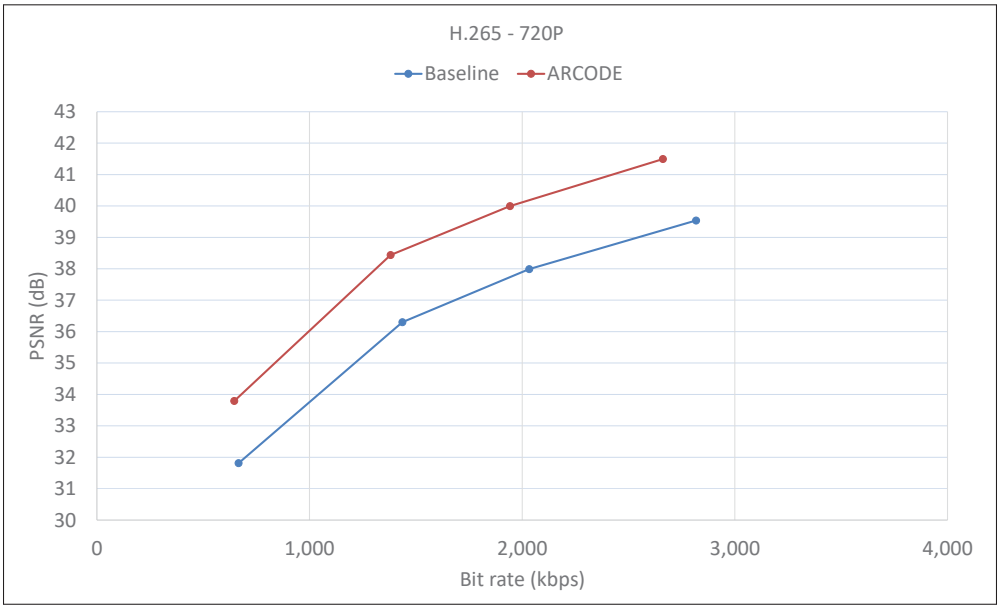

Figure 4.9 Comparison of the proposed model with the baseline for the first scenario when the client entity is idle in the scene without camera rotation and translation in the scene (H.265 - 720P)

4.4.1.3 Test results for camera rotation using H.264 at 1080P

For this test, the camera rotation is performed with the client entity in the center. The client entity is idle in the scene. The rotation of the camera according to the position of the client entity is considered in such a way that the elements of the combat area are always in the camera frame. The camera continuously rotates from the right side of the client entity to the left side and vice versa. This test is done in 1080P resolution and the H.264 video CODEC for game streaming.

The results in Figure 4.10 show the performance of our model compared to the baseline when we have the camera rotation in the scene. The information in Table 4.6 indicates that for the same video quality, our model, compared to the baseline, reduces the video bit rate by almost 10%. A comparison of the image quality in baseline and ARCODE is demonstrated in Figure-A I-5.

Table 4.6 The calculated SR-BD-Rate and SR-BD-PSNR under various target bit rates chosen on Moonlight for our proposed model and the baseline with camera rotation (H.264 - 1080P)

Baseline (H.264 - 1080P)		
Target bit rate on Moonlight (bps)	Bit rate (bps)	PSNR (dB)
2,000,000	1,715,084	29.75
4,000,000	2,547,378	32.77
6,000,000	3,858,145	36.57
8,000,000	5,665,088	38.55
ARCODE (H.264 - 1080P)		
Target bit rate on Moonlight (bps)	Bit rate (bps)	PSNR (dB)
2,000,000	1,626,629	29.88
4,000,000	2,523,332	33.74
6,000,000	3,837,499	37.13
8,000,000	5,632,676	38.97
	SR-BD-Rate (%)	**SR-BD-PSNR (dB)**
	-9.2	0.748

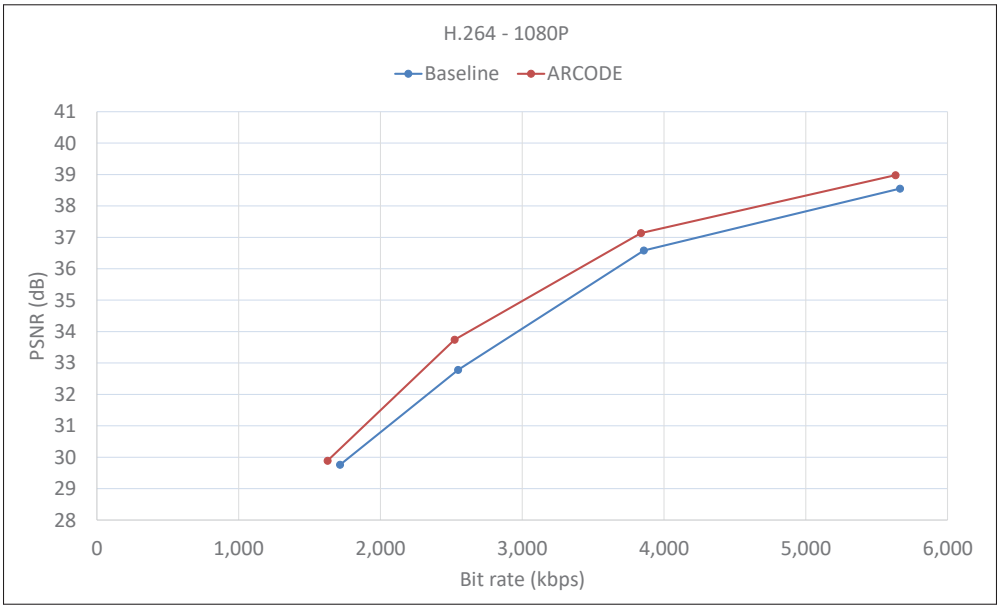

Figure 4.10 Comparison of the proposed model with the baseline for the first scenario when the client entity is idle in the scene, with camera rotation (H.264 - 1080P)

4.4.1.4 Test results for camera translation using H.264 at 1080P

For this test, the client entity is not idle in the scene. It moves horizontally and vertically in two opposite directions within a region in the combat area. The camera follows the position of the client entity in the scene. This test is done in 1080P resolution and H.264 video CODEC for game streaming.

Based on the results shown in Figures 4.11, our proposed model performs well compared to the baseline in the situation that the client entity is continuously moving in a region of the combat area with the camera following its position. Referring to Table 4.7, our model reduces about 14% of the bit rate for the same visual quality compared to the baseline. A comparison of the image quality in baseline and ARCODE is demonstrated in Figure-A I-6.

Table 4.7 The calculated SR-BD-Rate and SR-BD-PSNR under various target bit rates chosen on Moonlight for our proposed model and the baseline with camera translation (H.264 - 1080P)

Baseline (H.264 - 1080P)		
Target bit rate on Moonlight (bps)	Bit rate (bps)	PSNR (dB)
2,000,000	1,359,619	30.32
4,000,000	2,598,380	35.04
6,000,000	3,851,754	37.12
8,000,000	5,646,171	38.85
ARCODE (H.264 - 1080P)		
Target bit rate on Moonlight (bps)	Bit rate (bps)	PSNR (dB)
2,000,000	1,263,650	30.60
4,000,000	2,507,259	35.82
6,000,000	3,807,593	37.84
8,000,000	5,596,229	39.31
	SR-BD-Rate (%)	**SR-BD-PSNR (dB)**
	-14	0.885

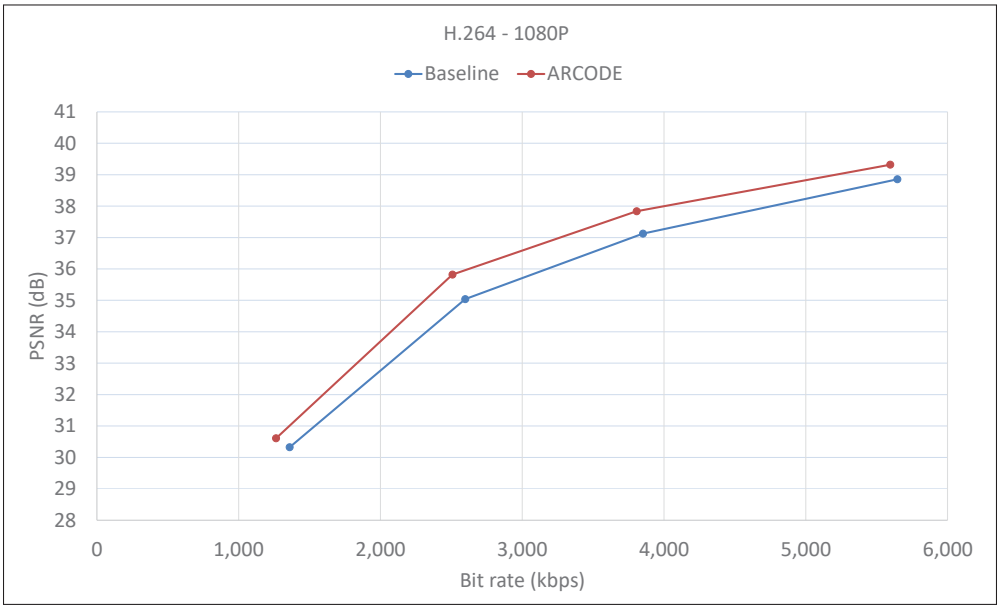

Figure 4.11 Comparison of the proposed model with the baseline for the first scenario when the client entity is moving in the scene with camera translation (H.264 - 1080P)

4.4.2 Boss fight scenario

In most MMORPGs it is a common scenario where a large number of players, using their avatars, gather around one or a few bigger and stronger enemies to fight them. Here the battle only occurs in relatively small regions of the scene. Accordingly, the changes only happen within these regions, and the other areas of the combat area remain unchanged.

In our main scenario, we considered various properties for our evaluation and compared the performance of our model to the baseline by changing each of these properties. For this scenario, we only show the results when the client entity is idle in the scene, without the camera rotation and translation. The target resolution is 1080P and the target video CODEC is H.264.

Figure 4.12 illustrates the performance of our model compared to the baseline in the boss fight scenario. The information in Table 4.8 indicates that for the same visual quality, ARCODE saves about 16% of the bit rate. A comparison of the image quality in baseline and ARCODE is demonstrated in Figure-A I-7.

Table 4.8 The calculated SR-BD-Rate and SR-BD-PSNR under various target bit rates chosen on Moonlight for our proposed model and the baseline for the boss fight scenario (H.264 - 1080P)

Baseline (H.264 - 1080P)		
Target bit rate on Moonlight (bps)	Bit rate (bps)	PSNR (dB)
2,000,000	1,281,728	31.29
4,000,000	2,444,565	36.52
6,000,000	3,762,945	38.70
8,000,000	5,496,938	40.28
ARCODE (H.264 - 1080P)		
Target bit rate on Moonlight (bps)	Bit rate (bps)	PSNR (dB)
2,000,000	1,198,875	31.90
4,000,000	2,394,505	37.36
6,000,000	3,698,110	39.38
8,000,000	5,345,811	40.97
	SR-BD-Rate (%)	**SR-BD-PSNR (dB)**
	-16.1	0.96

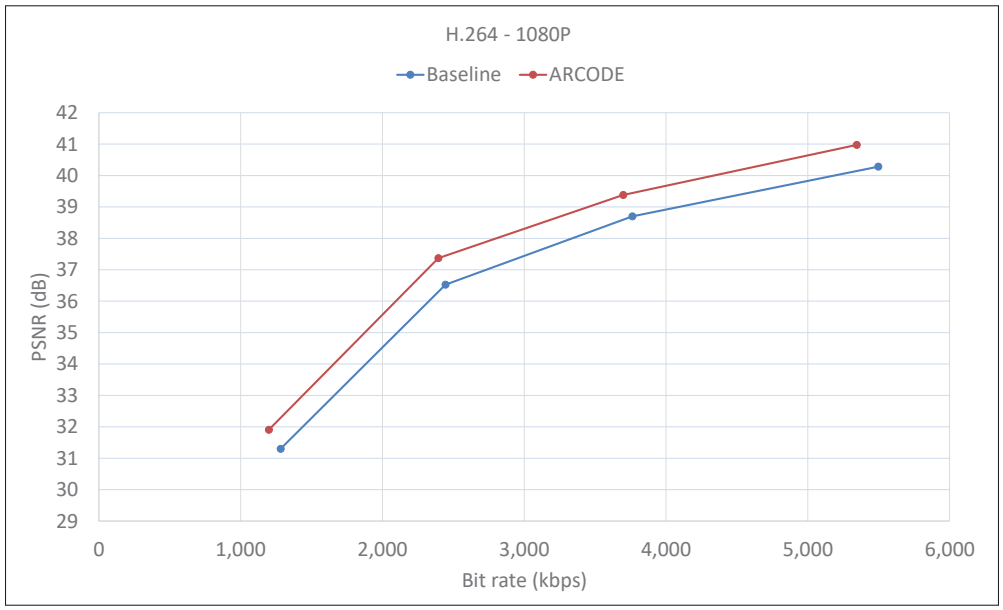

Figure 4.12 Comparison of the proposed model with the baseline for the boss fight scenario (H.264 - 1080P)

4.4.3 Sensitivity test

For the sensitivity test, we consider two scenarios. In the first scenario, we only take into account the movement of NPCs within the scene. NPCs are moving in the scene horizontally and vertically in two opposite directions without using their weapons and skills. In the second scenario, NPCs are continuously using their skills during the battle. Therefore in each state of the game, many skill entities are spawned in the scene. In both scenarios, the client entity is idle, and there is no camera rotation and translation in the scene. In both scenarios, the NPCs are positioned in various parts of the game scene in order to have motions in different regions. The test is done in 1080P resolution with H.264 as the target CODEC.

Figure 4.13 illustrates the performance of our model compared to the baseline when we only have the movement of the NPCs in the game scene. Based on the information in Table 4.9, in this scenario, ARCODE can save about 11% of the bit rate for the same quality, compared to the baseline.

Table 4.9 The calculated SR-BD-Rate and SR-BD-PSNR under various target bit rates chosen on Moonlight for our proposed model and the baseline when NPCs are moving in the scene without using their weapons and skills (H.264 - 1080P)

Baseline (H.264 - 1080P)		
Target bit rate on Moonlight (bps)	Bit rate (bps)	PSNR (dB)
2,000,000	977,288	36.74
4,000,000	2,182,546	42.21
6,000,000	3,211,311	44.08
8,000,000	4,746,644	45.82
ARCODE (H.264 - 1080P)		
Target bit rate on Moonlight (bps)	Bit rate (bps)	PSNR (dB)
2,000,000	922,571	37.18
4,000,000	2,120,275	42.89
6,000,000	3,209,810	44.87
8,000,000	4,677,164	46.21
	SR-BD-Rate (%)	**SR-BD-PSNR (dB)**
	-11.0	0.78

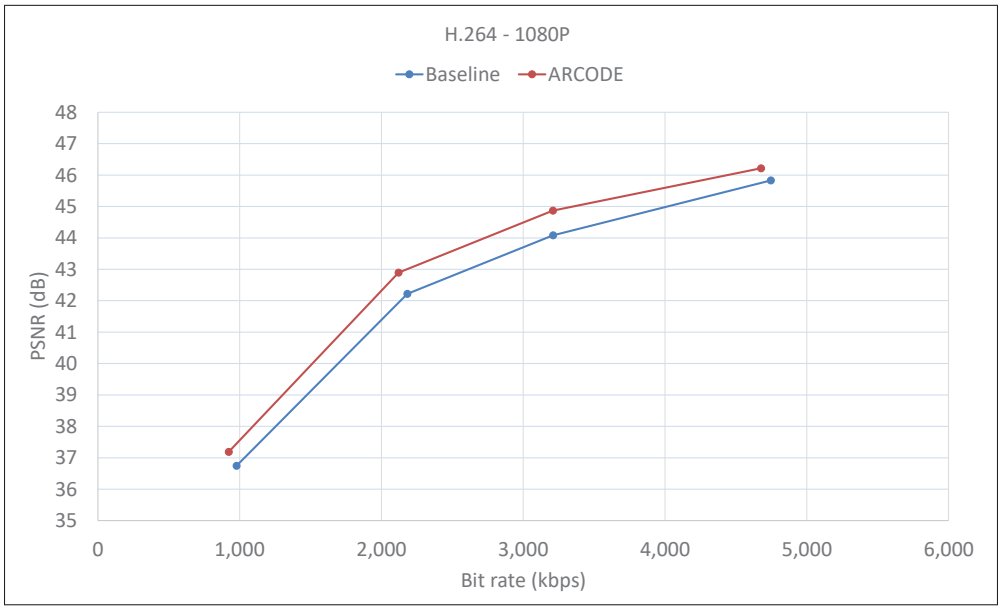

Figure 4.13 Comparison of the proposed model with the baseline when NPCs are moving in the scene without using their weapons and skills (H.264 - 1080P)

Referring to Figure 4.14, it is a significant difference between the performance of ARCODE and the baseline when we have many skill entities with their motions in the game scene. The information in Table 4.10 shows that for the same video quality, using our proposed model leads to about 41% bit rate saving compared to the baseline. A comparison of the image quality in baseline and ARCODE for both tests is demonstrated in Figure-A I-8 and Figure-A I-9.

Table 4.10 The calculated SR-BD-Rate and SR-BD-PSNR under various target bit rates chosen on Moonlight for our proposed model and the baseline when NPCs continuously using their skills (H.264 - 1080P)

Baseline (H.264 - 1080P)		
Target bit rate on Moonlight (bps)	Bit rate (bps)	PSNR (dB)
2,000,000	1,466,605	29.48
4,000,000	2,637,395	33.35
6,000,000	3,930,123	35.56
8,000,000	5,669,344	37.25
ARCODE (H.264 - 1080P)		
Target bit rate on Moonlight (bps)	Bit rate (bps)	PSNR (dB)
2,000,000	1,250,225	30.82
4,000,000	2,426,315	36.10
6,000,000	3,704,641	37.95
8,000,000	5,352,089	39.33
	SR-BD-Rate (%)	**SR-BD-PSNR (dB)**
	-41.3	2.95

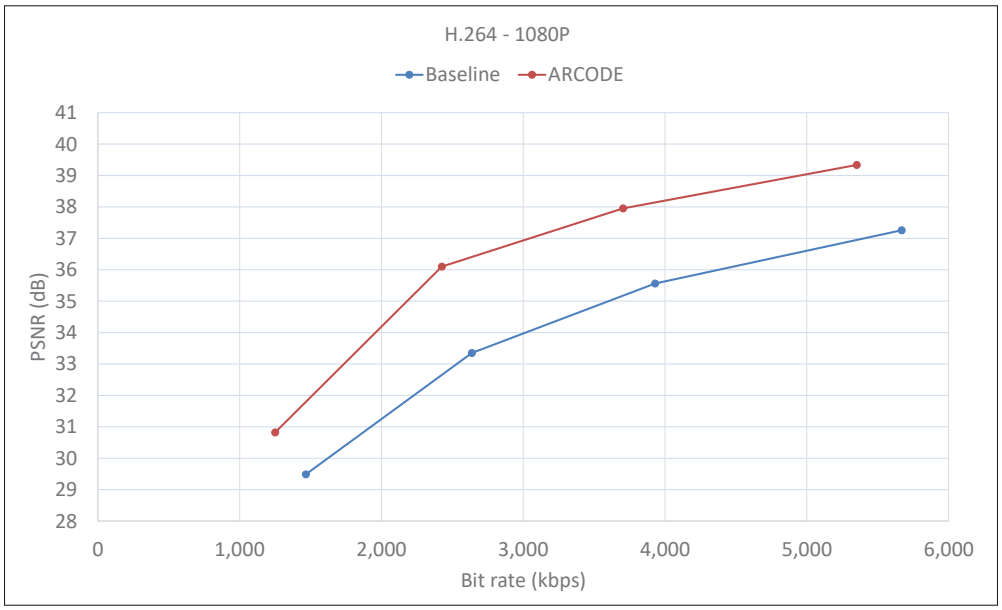

Figure 4.14 Comparison of the proposed model with the baseline when NPCs continuously using their skills (H.264 - 1080P)

4.4.4 Results analysis and discussion

The results show the superiority of our model compared to the baseline in all cases. Regarding the usability of our model for different target video CODECs, the results obtained from having both CODECs as our target video CODEC for game streaming support the comparability and portability of our model. It is quite apparent that our proposed model works well for both video CODECs.

The performance of our model is significant in cases where we do not have any camera rotation and translation in the game scene. The movement of the client entity in the scene always causes the camera movements. It can be camera translation, camera rotation, or both at the same time. Camera rotation can also happen when the client entity is idle in the scene. The camera rotation and translation apply changes to the video frames, and our model has no control over it. The test results of the camera rotation and translation in our main scenario show that the performance of our model declines in these cases. However, ARCODE still has a better performance compared to the baseline.

Considering the results of the boss fight scenario, generally, if the changes and motions in the game scene are concentrated only in some relatively small regions, since the position of the client entity is near to at least one of these regions to interact, it receives updates with short intervals from many entities in the game scene. Here, only a few or no region is left from which the client entity can receive position updates with long intervals. As a result, the performance of our model declines compared to the main scenario in which we have the battle distributed in many regions.

Considering the results obtained from the sensitivity tests, when the number of objects in the scene is small, the amount of movement and changes in the scene decreases accordingly. Here, the encoder can encode video frames with a limited bit rate allocated for the game streaming. Conversely, when the number of objects and amount of motion in the scene are large, the encoder needs to spend more bits to encode video frames. If the player does not provide the required bit rate, the visual quality of the video is sacrificed so that the encoder does not exceed the allocated bit rate. Here, ARCODE significantly outperforms the baseline, and by reducing the amount of motion in objects in the scene, it makes the encoder require fewer bits to encode video frames with the same visual quality as the baseline. As a result, the greater the number of objects and the greater amount of motion, the more significant is the difference in the performance of ARCODE compared to the baseline.

The update intervals and update rates shown in Table 4.1 are based on the determined values for the properties that are shown in Figure 3.8 and used in Equation 3.1. As we mentioned earlier, the values for each parameter can change, and to get the optimum values, more simulations should be performed. Apart from more objective tests, the subjective tests can be performed having many real players using the game to find the best trade-off between the visual quality of the video and the frequency of the updates player receives, which leads to the best QoE.

Regardless of having our proposed model and the baseline into consideration, with a comparison between the performance of H.264 and H.265 video CODECs based on the results we have from the experiment performed with the two video CODECs in our main scenario, we can see that

H.265 performs better than H.264. However, contrary to the expectations, the difference between their performance is not very significant. It prompted us to compare the performance of the two CODECs with a new video sequence with more motion in the scene and more complex textures than our exemplar game. This video sequence is streamed from our game stream PC server to the thin-client device. The results showed that using H.265 can reduce the video bit rate by almost 27% for the same visual quality compared to H.264. It is also investigated that NVENC is using H.264 high profile, which is quite efficient. According to the results obtained from this experiment, it can be concluded that the reason behind not seeing a very significant difference between the performance of the two CODECs, having our exemplar game into consideration, is: the use of H.264 High profile by NVENC, which is superior to H.264 baseline, and so closer to H.265; the nature of the visual content in our exemplar game which is quite different from movies or real life visual content for which these video CODECs are primarily designed.

CONCLUSION

With the increasing popularity of cloud gaming, the need for players to have powerful devices to run games with complex graphics decreases, and instead, the need for high-speed Internet connections increases. The expensive process of graphics rendering is directly performed on cloud gaming platforms, and a compressed video is transmitted over the Internet to the end-users. The Internet connection of players should provide the required bit rate to be used by the encoder to encode the video frames without degrading the visual quality of the video. If the Internet connection cannot provide the required bit rate, the visual quality of the video decreases. A video with higher motions and more complex textures adds more complexity to the encoding process, and the encoder needs to consume more bits to encode the video at a certain fidelity.

We implemented a model for MMORPGs on cloud gaming platforms to reduce the frequency of motion updates and changes in consecutive frames of a video and reduce the number of bits needed by the encoder to encode the video with the same visual quality as the baseline. ARCODE provides a better visual quality with comparable bandwidth in MCG. Players with lower bandwidth imposed by a low-speed Internet connection, which is very common in CMG, can experience playing MMORPGs on their thin-client devices with better visual quality compared to the baseline.

ARCODE is action-aware, and by capturing the properties of different actions, adjust the update rates the player receives from the position of other objects in the game. The model considers the importance of each object to the player in each state of the game and adjusts the frequency of updates based on their importance. The importance of other objects to a player, is not only determined by the actions that the avatar of the player can perform during gameplay but also is determined by the actions each object can perform in the combat area in each state of the game. By reducing the number of motion updates of each object's visual in the scene based on their significance in each state of the game, the amount of motion and number of changes in the scene

is reduced accordingly. Less motion in the scene leads to fewer required bits to encode each video frame.

One of the advantages of our model is that it is aware of the game semantics, and to determine the importance of the objects in each state of the game, it only requires little information about the in-game actions to be provided by the game developers. ARCODE adjusts the position updates in each state of the game in a way such that it does not affect the playability and the fairness of the game to players with bandwidth constraints.

The other advantage of ARCODE is its usability for different video CODECs in cloud gaming platforms. The model does not need to modify the internal behaviour (encoding algorithms), the inputs or the scripts of the CODECs to optimize the performance of the encoder. ARCODE generates contents that are easier to encode to achieve this goal.

A wide selection of test scenarios was considered for the evaluation and to show the excellent performance of our model. Our experimental results indicate that for the same visual quality as the baseline, ARCODE reduces the video bit rate between 10% and 41%.

5.1 Future Work

Several future works that can improve our model or expand upon it can be conducted. Here are a few suggestions.

Different game assets. Different game assets with different textures and complexity in design can be considered for the evaluation. The terrain, the characters, skill objects, and generally speaking, the whole game world with different objects inside it can be redesigned in various ways. The evaluation can be done with various game scenes and objects to see how the proposed model performs in each case.

Different genres of online games. We offered the concept of reducing the motion of in-game objects in order to deliver a video with better visual quality to players in MMORPGs. Other genres of online games with different characteristics can use this idea and change the implementation of the model based on the characteristics of the game.

Different properties of in-game actions and objects. Various action data and characteristics of different in-game objects can be used to adjust the update rates. For instance, the amount of impact that performing an action can have on other objects and the speed of each object in the game can also be taken into account to adjust the update rates.

Adaptive bit rate budget model. Right now, the information about the bit rate budget and the target resolution is requested from the player on the client UI. An adaptive bit rate budget model can be used in order to calculate the relative bit rate difference on the fly, considering the bit rate fluctuations during the game stream.

Optimization of update rates. For the various game genres, assets, objects, and actions, optimize the update frequencies in order to reduce the bit rate at a certain visual quality without sacrificing playability. Investigating an automated manner to perform such optimization would be of great interest.

Subjective tests. An experiment with a group of real players playing the game can help find the best trade-off between the perceptual quality and the frequency of updates each player receives considering their bit rate budget.

APPENDIX I

IMAGES FROM THE GAME

This appendix demonstrates the perceptual quality of the content generated in baseline and our proposed model, using frames taken from our exemplar game. The frames are captured from Moonlight after the video contents are encoded. The first frame in each figure is an image of the content generated in the baseline. The second frame is an image of the content generated in ARCODE.

(a) Baseline

(b) ARCODE

Figure-A I-1 The image quality of baseline compared to ARCODE when the client entity is idle without camera rotation and translation (video CODEC: H.264, resolution: 1080P, bit rate: 2Mbps)

(a) Baseline

(b) ARCODE

Figure-A I-2 The image quality of baseline compared to ARCODE when the client entity is idle without camera rotation and translation (video CODEC: H.264, resolution: 720P, bit rate: 1Mbps)

(a) Baseline

(b) ARCODE

Figure-A I-3 The image quality of baseline compared to ARCODE when the client entity is idle without camera rotation and translation (video CODEC: H.265, resolution: 1080P, bit rate: 2Mbps)

(a) Baseline

(b) ARCODE

Figure-A I-4 The image quality of baseline compared to ARCODE when the client entity is idle without camera rotation and translation (video CODEC: H.265, resolution: 720P, bit rate: 1Mbps)

(a) Baseline

(b) ARCODE

Figure-A I-5 The image quality of baseline compared to ARCODE when the client entity is idle, having the camera rotation in the scene (video CODEC: H.264, resolution: 1080P, bit rate: 4Mbps)

(a) Baseline

(b) ARCODE

Figure-A I-6 The image quality of baseline compared to ARCODE when the client entity is moving, having the camera translation in the scene (video CODEC: H.264, resolution: 1080P, bit rate: 4Mbps)

(a) Baseline

(b) ARCODE

Figure-A I-7 The image quality of baseline compared to ARCODE in boss fight scenario (video CODEC: H.264, resolution: 1080P, bit rate: 4Mbps)

(a) Baseline

(b) ARCODE

Figure-A I-8 The image quality of baseline compared to ARCODE when the NPCs are moving in the scene without using their skills (video CODEC: H.264, resolution: 1080P, bit rate: 2Mbps)

(a) Baseline

(b) ARCODE

Figure-A I-9 The image quality of baseline compared to ARCODE when the NPCs continuously using their skills (video CODEC: H.264, resolution: 1080P, bit rate: 4Mbps)

REFERENCES

Ahmadi, H., Zadtootaghaj, S., Pakdaman, F., Hashemi, M. R. & Shirmohammadi, S. (2021). A Skill-Based Visual Attention Model for Cloud Gaming. *IEEE Access*, 9, 12332-12347. doi: 10.1109/ACCESS.2021.3050489.

Benford, S. & Fahlén, L. (1993). A spatial model of interaction in large virtual environments. In *Proceedings of the Third European Conference on Computer-Supported Cooperative Work 13–17 September 1993, Milan, Italy ECSCW'93* (pp. 109–124).

Bharambe, A. R., Pang, J. & Seshan, S. (2006). Colyseus: A Distributed Architecture for Online Multiplayer Games. *NSDI*, 6, 12–12.

Bjøntegaard, G. (2001). Calculation of average PSNR differences between RD-curves. *VCEG-M33*.

Boulanger, J.-S., Kienzle, J. & Verbrugge, C. (2006). Comparing interest management algorithms for massively multiplayer games. *Proceedings of 5th ACM SIGCOMM workshop on Network and system support for games*, pp. 6–es.

Burnes, A. (2019). Available Now: New GeForce-Optimized OBS and RTX Encoder Enables Pro-Quality Broadcasting on a Single PC. Retrieved from https://www.nvidia.com/en-us/geforce/news/geforce-rtx-streaming/.

Businesswire. (2020). Introducing Luna—Amazon's New Cloud Gaming Service Where it's Easy to Play on the Devices You Already Own. Retrieved from https://www.businesswire.com/news/home/20200924005810/en/.

Cai, W., Shea, R., Huang, C., Chen, K., Liu, J., Leung, V. C. M. & Hsu, C. (2016). A Survey on Cloud Gaming: Future of Computer Games. *IEEE Access*, 4, 7605-7620. doi: 10.1109/ACCESS.2016.2590500.

Cañas, C., Zhang, K., Kemme, B., Kienzle, J. & Jacobsen, H.-A. (2014). Publish/subscribe network designs for multiplayer games. *Proceedings of the 15th International Middleware Conference*, pp. 241–252.

Choudhry, K. (2018). Project xCloud: Gaming with you at the center. Retrieved from https://blogs.microsoft.com/blog/2018/10/08/project-xcloud-gaming-with-you-at-the-center/.

Choy, S., Wong, B., Simon, G. & Rosenberg, C. (2012). The brewing storm in cloud gaming: A measurement study on cloud to end-user latency. *2012 11th Annual Workshop on Network and Systems Support for Games (NetGames)*, pp. 1–6.

Choy, S., Wong, B., Simon, G. & Rosenberg, C. (2014). A hybrid edge-cloud architecture for reducing on-demand gaming latency. *Multimedia systems*, 20(5), 503–519.

Chuah, S.-P. & Cheung, N.-M. (2014). Layered coding for mobile cloud gaming. *Proceedings of International Workshop on Massively Multiuser Virtual Environments*, pp. 1–6.

Claypool, M. & Claypool, K. (2006). Latency and player actions in online games. *Communications of the ACM*, 49(11), 40–45.

Deng, Y., Li, Y., Seet, R., Tang, X. & Cai, W. (2018). The Server Allocation Problem for Session-Based Multiplayer Cloud Gaming. *IEEE Transactions on Multimedia*, 20(5), 1233-1245. doi: 10.1109/TMM.2017.2760621.

Dhib, E., Boussetta, K., Zangar, N. & Tabbane, N. (2017). Cost-aware virtual machines placement problem under constraints over a distributed cloud infrastructure. *2017 sixth international conference on communications and networking (ComNet)*, pp. 1–5.

Hegazy, M., Diab, K., Saeedi, M., Ivanovic, B., Amer, I., Liu, Y., Sines, G. & Hefeeda, M. (2019). Content-aware video encoding for cloud gaming. *Proceedings of the 10th ACM multimedia systems conference*, pp. 60–73.

Heger, F., Schiele, G., Süselbeck, R. & Becker, C. (2009). Towards an interest management scheme for peer-based virtual environments. *Electronic Communications of the EASST*, 17.

Hemmati, M., Javadtalab, A., Nazari Shirehjini, A. A., Shirmohammadi, S. & Arici, T. (2013). Game as video: Bit rate reduction through adaptive object encoding. *Proceeding of the 23rd ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*, pp. 7–12.

Hollister, S. & Statt, N. (2019). Google's Stadia Cloud Gaming Service is Coming November 19th: Everything You Need to Know. Retrieved from https://www.theverge.com/2019/6/6/18654632/google-stadia-price-release-date-games-bethesda-ea-doom-ubisoft-e3-2019.

Hollister, S. (2014). Sony announces PlayStation Now, its cloud gaming service for TVs, consoles, and phones. Retrieved from https://www.theverge.com/2014/1/7/5284294/sony-announces-playstation-now-cloud-gaming.

ITU-T. (2003). ITU-T Recommendation H.264: Advanced video coding for generic audiovisual services.

ITU-T. (2013). ITU-T Recommendation H.265: High Efficiency Video Coding.

Jaya, I., Cai, W. & Li, Y. (2020). Rendering server allocation for mmorpg players in cloud gaming. *49th International Conference on Parallel Processing-ICPP*, pp. 1–11.

Kämäräinen, T., Siekkinen, M., Xiao, Y. & Ylä-Jääski, A. (2014). Towards pervasive and mobile gaming with distributed cloud infrastructure. *2014 13th Annual Workshop on Network and Systems Support for Games*, pp. 1–6.

Kienzle, J., Verbrugge, C., Kemme, B., Denault, A. & Hawker, M. (2009). Mammoth: a massively multiplayer game research framework. *Proceedings of the 4th International Conference on Foundations of Digital Games*, pp. 308–315.

Li, Y., Tang, X. & Cai, W. (2015). Play Request Dispatching for Efficient Virtual Machine Usage in Cloud Gaming. *IEEE Transactions on Circuits and Systems for Video Technology*, 25(12), 2052-2063. doi: 10.1109/TCSVT.2015.2450152.

Liu, Y., Dey, S. & Lu, Y. (2015). Enhancing video encoding for cloud gaming using rendering information. *IEEE Transactions on Circuits and Systems for Video Technology*, 25(12), 1960–1974.

Lu, C.-W., Wang, S.-D. & Chien, S.-Y. (2019). A Novel Gaming Video Encoding Process Using In-Game Motion Vectors. *IEEE Transactions on Circuits and Systems for Video Technology*, 30(7), 2207–2214.

Lu, Y., Liu, Y. & Dey, S. (2017). Asymmetric and selective object rendering for optimized Cloud Mobile 3D Display Gaming user experience. *Multimedia Tools and Applications*, 76(18), 18291–18320.

Mangalindan, J. (2020). Cloud gaming's history of false starts and promising reboots. Retrieved from https://www.polygon.com/features/2020/10/15/21499273/cloud-gaming-history-onlive-stadia-google.

Marzolla, M., Ferretti, S. & D'angelo, G. (2012). Dynamic resource provisioning for cloud-based gaming infrastructures. *Computers in Entertainment (CIE)*, 10(1), 1–20.

Semsarzadeh, M., Hemmati, M., Javadtalab, A., Yassine, A. & Shirmohammadi, S. (2014). A video encoding speed-up architecture for cloud gaming. *2014 IEEE International Conference on Multimedia and Expo Workshops (ICMEW)*, pp. 1–6.

Shea, R., Liu, J., Ngai, E. C. . & Cui, Y. (2013). Cloud gaming: architecture and performance. *IEEE Network*, 27(4), 16-21. doi: 10.1109/MNET.2013.6574660.

Shi, S., Hsu, C.-H., Nahrstedt, K. & Campbell, R. (2011). Using graphics rendering contexts to enhance the real-time video coding for mobile cloud gaming. *Proceedings of the 19th ACM international conference on Multimedia*, pp. 103–112.

Sun, K. & Wu, D. (2015). Video rate control strategies for cloud gaming. *Journal of Visual Communication and Image Representation*, 30, 234–241.

Süselbeck, R., Schiele, G. & Becker, C. (2009). Peer-to-peer support for low-latency massively multiplayer online games in the cloud. *2009 8th Annual Workshop on Network and Systems Support for Games (NetGames)*, pp. 1–2.

Tan, T. K., Weerakkody, R., Mrak, M., Ramzan, N., Baroncini, V., Ohm, J. & Sullivan, G. J. (2016). Video Quality Evaluation Methodology and Verification Testing of

HEVC Compression Performance. *IEEE Transactions on Circuits and Systems for Video Technology*, 26(1), 76-90. doi: 10.1109/TCSVT.2015.2477916.

Wang, J. Y., Zhang, K. & Jacobsen, H.-A. (2017). Combat state-aware interest management for online games. *Proceedings of the 18th ACM/IFIP/USENIX Middleware Conference: Posters and Demos*, pp. 17–18.

Wang, S. & Dey, S. (2009). Modeling and Characterizing User Experience in a Cloud Server Based Mobile Gaming Approach. *GLOBECOM 2009 - 2009 IEEE Global Telecommunications Conference*, pp. 1-7. doi: 10.1109/GLOCOM.2009.5425784.

Wang, S. & Dey, S. (2010a). Addressing Response Time and Video Quality in Remote Server Based Internet Mobile Gaming. *2010 IEEE Wireless Communication and Networking Conference*, pp. 1-6. doi: 10.1109/WCNC.2010.5506572.

Wang, S. & Dey, S. (2010b). Rendering Adaptation to Address Communication and Computation Constraints in Cloud Mobile Gaming. *2010 IEEE Global Telecommunications Conference GLOBECOM 2010*, pp. 1-6. doi: 10.1109/GLOCOM.2010.5684144.

Wang, S., Liu, Y. & Dey, S. (2012). Wireless network aware cloud scheduler for scalable cloud mobile gaming. *2012 IEEE International Conference on Communications (ICC)*, pp. 2081–2086.

Wu, J., Yuen, C., Cheung, N., Chen, J. & Chen, C. W. (2015). Enabling Adaptive High-Frame-Rate Video Streaming in Mobile Cloud Gaming Applications. *IEEE Transactions on Circuits and Systems for Video Technology*, 25(12), 1988-2001. doi: 10.1109/TCSVT.2015.2441412.

Xu, L., Guo, X., Lu, Y., Li, S., Au, O. C. & Fang, L. (2014). A low latency cloud gaming system using edge preserved image homography. *2014 IEEE International Conference on Multimedia and Expo (ICME)*, pp. 1–6.

Yahyavi, A., Huguenin, K., Gascon-Samson, J., Kienzle, J. & Kemme, B. (2013). Watchmen: Scalable Cheat-Resistant Support for Distributed Multi-player Online Games. *2013 IEEE 33rd International Conference on Distributed Computing Systems*, pp. 134-144.

Yahyavi, A. & Kemme, B. (2013). Peer-to-peer architectures for massively multiplayer online games: A survey. *ACM Computing Surveys (CSUR)*, 46(1), 1–51.

Zhang, K. & Kemme, B. (2011). Transaction Models for Massively Multiplayer Online Games. *2011 IEEE 30th International Symposium on Reliable Distributed Systems*, pp. 31-40.

Zhang, K. (2010). *Persistent transaction models for massively multiplayer online games*. (Ph.D. thesis, McGill University Library).