The University of Hong Kong

Faculty of Engineering

Department of Computer Science

COMP7702

Project Title

Cross Platform Game Development with Unity

Submitted in partial fulfillment of the requirements for the admission
to the degree of Master of Science in Computer Science

By
Student name: YANG Zhou
HKU student no.: 2013950527

Supervisor's title and name: Dr. T. W. Chim
Date of submission: 30/11/2014

Declaration of the Candidate

I, the undersigned, hereby declare that the work contained in this thesis is my own original work, and has not previously in its entirely or in part been submitted at any university for a degree.

Only the source cited in the document has been used in this draft. Parts that are direct quotes or paraphrases are identified as such.

I agree that my work is published, in particular that the word is presented to third parties for inspection or copies of the work are made to pass on to third parties.

The University of Hong Kong,

Signature

The University of Hong Kong

Master Thesis

Cross Platform Game Development with Unity

YANG Zhou

2013950527

Abstract

As a lightweight open source game engine, Unity have become the most popular game engine in recent years. With its powerful functions we can design and build a cross-platform game on our own. By using the Unity engine, we manage to design and successfully implement an ARPG (Action Role Playing Game) and make it a multiplayer game. As part of the team I am mainly responsible for the construction of skill system and the whole network infrastructure. Appling the mechanism of P2P (Peer-to-Peer) network, the network system can provide support for both LAN (Local Area Network) and Internet connection, which means you can play the game with your friend in the same local network or on the Internet. Some special Network Optimization Skill also be applied to resolve some network problems exist in Unity. By now our game can be fully functional on both single and multiplayer mode. The game quality had achieved a high level of stability and fluency with all the related function in normal network. In this article, I will present the network mechanism in Unity and how we integrated the network functions into our single game system to make it work online. And the network mechanism can be applied to any Unity-base game with proper implementation.

Key Word: Unity, multiplayer, cross-platform, network mechanism, LAN, Internet, P2P network, network optimization.

Table of Contents

List of Tables

List of Figures

1 Introduction

Driven by the development of Internet and the promotion of smart digital devices, nowadays the electronic games have become a very common entertainment form for people. Various game engines attract many developers to join the development of electronic games. With the cross-platform feature, Unity soon becomes the most popular game engine in the game industry for indie game developers. Unity allows the developer to create various games and help them deploy the game to multiple platforms easily, which includes PC, Web, iOS, Android and some home game console such as Xbox and some other platforms. As a result, the game developer can focus more on the content of game and the making of more interesting game.

In the history of computer game, the multiplayer game mode has always been one of the most important factors for both the users and the developers. Technically speaking, the multiplayer game mode refers to a game pattern in which more than one player can play in the same game scene. In multiplayer games players can cooperate in order to finish some tasks, which cannot be achieved by single person. Or they can also fight against each other, to determine who is the better or more skilled player. Although the game AI today have been greatly improved to make the interaction between player and game objects more real and interesting, the fact still proves that playing with another player will bring a much better experience for the game players.

In this article, I am going to introduce how I design and build the networked part for an RPG game, which is implemented with my partner. The contents include the structure and mechanism of networking we build for our game system. Since the game is implemented using the Unity Game Engine, the technologies part covered will also be based on the Unity Game Development.

This article can be mainly divided into two parts. In the first part, I will introduce how to build a skill system and implement network support for an RPG game using the Unity Engine. The first part will mainly focus on the skill system. The process of design and implementation will be presented in detailed. Then the second part will be about the network section. In network section I am going to present some network function provided by the Unity Engine. And I will also give a detailed description on the mechanism of the network we are going to build, as well as how we actually integrate the network function in the current game system to make it a multiplayer game. Hopefully this article can help you gain a deeper understanding about the network mechanism in Unity game development.

2 Background

The RPG Game, which refers to the “Role Playing Game”, is a very classic game type applied in many electronic games. In RPG games, players are required to play the game as a role with specific characteristics. And the player can select different roles to enjoy different gaming experience. What my partner built in this project is a simple RPG game system. And my part of work is to integrate some network functions in the developed game system to make it playable online. So it is necessary for me to make a brief introduction on the game system before I describe how to apply network mechanism on it. In fact the whole game system consists of several parts, and each of them will be describe briefly in the following paragraph.

2.1 Camera System

The camera system manages the view of the players. Camera is a special game object, which provides the basic vision of the game scene for the users. In other words, the users see the game’s world from the camera’s angle. So by changing the position and rotation of the camera object the player can see the world in different ways, which is the key function for the camera system. The camera system defines the ways users control their game view. The camera will be focus on the player game object controlled by the user. The distance between camera and player can be adjusted using the mouse scroll wheel. Users can also rotate the view by holding the right mouse button with rotating the mouse.

2.2 Player Controller

The player controller system is mainly responsible for the movement of the player game object. It defines the way in which the users can controller his player’s movement in the game scene. Its job is to translate users’ input into the actual movement in the game world. However, the player controller system can’t work without the camera system. Since the movement output would be related to the player’s view angle, the player controller require the information from camera system to help the translation.

2.3 Player Animation

The player animation system serves as an agent of the animator controller. As we all know, the animation solution provided by Unity is the Mecanim, which will control the animation using a state machine system. However, to switch between different states we need to change the value of some pre-set parameters and send it to the animator controller. And we can simple do that with the player animations system, which include a series of functions to send different parameters. By communicating with the player animation system we can easily change the state of animation.

2.4 Character Status

The character status system is in charge of a game object’s life status. It applies several parameters to describe a game object’s current status. Generally speaking, it is a system that stores and keeps updating a series of game-related data. Although it doesn’t have direct impact on the game world, it actually plays the most important role in the game system. By providing or modifying the latest data of a game object, it defines the way game object interact with each other.

2.5 Warrior Combat

The warrior combat system is part of the combat system, which includes the warrior combat system and DPS combat system. The warrior system describes the way in which the warrior player engage in basic attack functions. By cooperating with other systems, the warrior combat system can make the player perform attack actions and actually cast damage to the nearby enemies. By setting a specific range, the warrior combat system can create an array, get the enemies within the range in the array, and then apply damage to all the enemies game objects in the array when the attack is finished. The warrior combat system also has a combo system, which can provide fast and continuous attack with rapid input.

2.6 DPS Combat

The DPS combat system is also part of the combat system. Different from the warrior combat system, the DPS combat system is specially designed for long-range attacker, which will hurt the enemy by launching a projectile. When the projectile hit the enemy objects, it will find the enemy’s character status system and apply pre-set damage to the object. The major features of the DPS combat system are its function to find the closest enemy and the auto-aim function. Whenever an attack is launched, the system will automatically find the closest enemy game object within the attack range and make the enemy face it automatically until a projectile is shoot to the enemy.

3 The Skill System

In this part, I am going to introduce how I design and implement the skill system base on the basic game system developed by my partner. First I will present you the different definitions on the three different characters: Warrior, Mage and Archer. And then I will tell you how I design the skill base on different characteristics of the characters. In the final part I will describe the structure and mechanism of the skill system, which involves the original game system, the visual effect elements and the actually effect on different game objects.

3.1 Different Roles of Characters

The skill system is actually an extension for the combat system. It aims to provide various ways for the players to engage in battle. When facing with different enemies or different situations the player should uses the skill in different way. These skills might improve the ability of the player, or weaken the enemies' abilities. They might be used to heal your teammates in multiplayer mode, and might also used to cast damage on the enemies. Anyway, each skill will be a special way to assist your player in the combat.

Generally, the game system will make the game more interesting. However, to differ the roles of different characters, we should design the skills according to the different characteristics of them. The following form shows the differences in roles and characteristics for different character.

Character	Role and Characteristic
	Warrior **Role:** Tank, Damage Taker. **Characteristic:** Have admirable health point while small volume of magic point. Low attack power but can attack nearby enemies in groups.
	Mage **Role:** Supporter, Assistance Provider. **Characteristic:** Opposite from the warrior, mage have small volume in health point but strong in magic point. Also low attack power, major responsibility is to provide support for other roles.
	Archer **Role:** Attacker, Damage Caster. **Characteristic:** Both health point and magic point are insignificant. Have excellent attack power. Performance will be better with the help of other two roles.

Table 1: Character Category

3.2 The Design of Skills

In our design, the skills can be divided into two main types. The first type is the common skills. The common skills are some simple skills that can be used no matter which character you use. The design of common skills intends to provide all the character the basic ability for them to escape battle and conduct a fast recovery. In this way the users can get a better game experience. After all nobody like to wait a long time for the character to recover to the best status after a deadly battle.

So here are some descriptions for the common skills:

Skill	Descriptions
	Name: Healing **Description:** Improve the player's recover rate on health point
	Name: Speed Up **Description:** Improve the player's moving speed shortly

Table 2: Common Skills

We also design and implement some useful items, which can have similar effect like skills. They can provide recovery or immediate transport for the player, which will be of great help to the RPG game player. So we also integrate these magical items as a part of the skills. The detailed information is as followed:

Skill	Descriptions
	Name: Transport Scroll **Description:** Transport the player to starting position immediately
	Name: Green Gem **Description:** Immediate recovery of health point
	Name: Blue Gem **Description:** Immediate recovery of magic point

Table 3: Common Items

The common skills are available for every character, to provide supportive functions for the players. However, as for the second part of the skills, they are totally different. The second part of the skills is the special skills. Different characters have different special skills. And I will introduce those skills from one character after another.

We will first introduce the special skills of warrior，which are as followed:

Skill	Descriptions
	Name: Shield Up **Description:** Improve the player’s amour for a while
	Name: Demon’s Power **Description:** Improve damage by decrease health point

Table 4: Warrior Special Skills

The two special skills of warrior suits the role of warrior very well. The first one will increase warrior’s amour by improve his amour. In this way, the warrior’ ability to take damage will be improved and make it a better tank. The second one is a skill which increase warrior’s power by decrease his health point. When facing large group of enemies, the best way to reduce damage is to kill all the enemies quickly. Since the warrior character have a combo system. This skill can help it to wipe out large group of enemies quickly while only need to undertake the damage brought by the skill.

And then here are the special skills for mage:

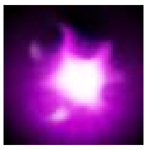
Skill	Descriptions
	Name: Group Healing **Description:** Heal all the players within a certain range
	**Name:** Magical Explosion **Description:** Cast huge damage to all the enemies nearby

Table 5: Mage Special Skills

The role of mage is an assistant, so as its skills. It can heal all the friendly players nearby by using the first special skills. And it can also cast great damage on all the enemy game objects nearby. Since mage is a character with small amount of health point, getting too close to a group of enemies might kill him. However, when cooperating with the other player, mage can provide the best assistance than any other role.

The last one would be the archer and its special skills are:

Skill	Descriptions
	Name: Magical Arrow **Description:** Change the normal arrow to a deadly magic arrow
	Name: Ice Explosion **Description:** Slow down all the enemies nearby

Table 6: Archer Special Skills

Archer plays a role of attacker. As a result, the first special skill of Archer is about damage improvement. The magical arrow will do a great more damage than the normal arrow does. And the second special skill can slow down all the enemies, which allow the archer to attack the slow down enemies from far away.

3.3 The Implementation of Skill System

Now all the skills have been designed, let me introduce how we build the whole skill system step by step. In our design, there are two stages for every skill, which includes:

- Skill Ready
- Skill Execution

Normally the skill will be in ready stage. If you want to invoke a skill, the system will check whether the player have enough magic point to do that. And the skill will only be invoked when the player have enough magic point.

After a player successful invoke a skill, the skill being invoked will enter the Skill Execution stage. The magic point cost will be updated and a cool down clock will be set to show how long the player have to wait before the skill is ready again. When the cool down process is over, the skill will enter the ready stage.

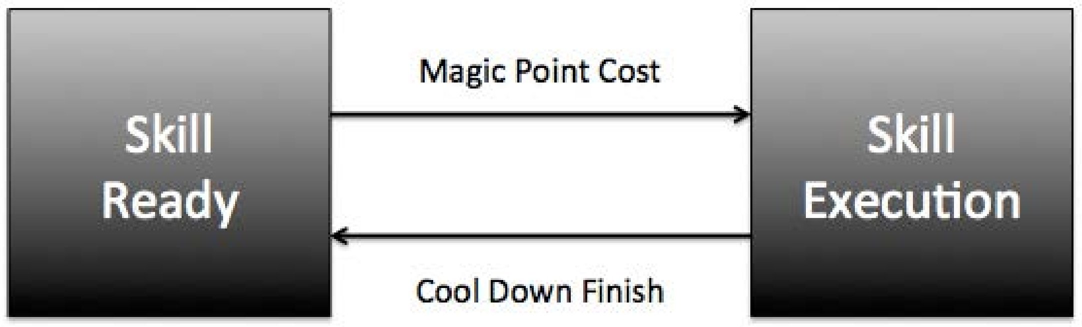


Figure 1: The State of Skill

The whole skill system consists of three main components. The most important is the skill manager, which manage all the information about every skill. The player status refers to the character status system on a specific player, and it provides the information of current magic point (mana). The action bar is part of the UI system. It manages the way in which player send the order of invoking a skill.

So the basic mechanism and event flow is as followed: First the player will invoke a skill by pressing a skill hot key (which can be set at the skill bar) or directly click on the skill icon. Then the action bar will check whether a skill is ready. If the skill is in cool down stage the input will be ignored. Otherwise, it will send the request to the skill manager.

The skill manager receive the request, and then it will find the skill's magic point cost and ask the player status whether the player have enough magic point to launch the skill. If the answer is yes, the skill will enter an execution stage. In this stage, the skill manager will ask the player status to reduce the magic point as much as the skill takes. Then it will also inform the action bar to start a cool down count down. And finally the skill manager will perform the skill on the game world by deploying some visual effects and modify some game data.

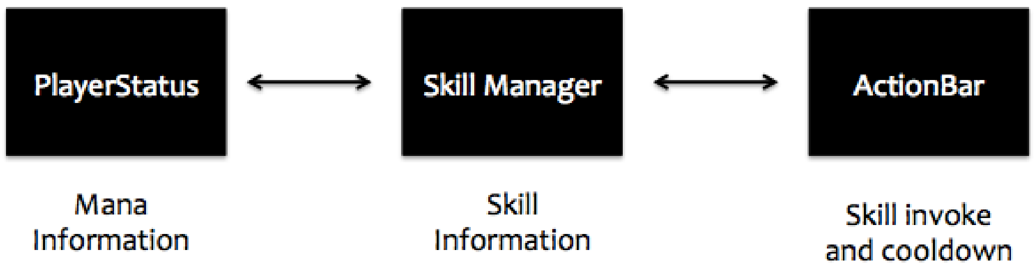


Figure 2: The Structure of Skill System

The content of the skills might differ, but for all the skills they all follow the same mechanism as above. Since some skills are quite powerful, the mana cost and cool down time are necessary for keeping the user from over using them. Generally speaking, some skill with small impact will cost less mana and have less cool down time. However, for some powerful skills (such as the special skills of mage) the price will be more expensive. But in this way, the game will have a better balance and become more interesting.

4 The Network System

Network is a larger and complex topic in the filed of game development. Since we want to make our game a multiplayer game, it is necessary for us to figure out what a network system will be like and how should we implement it. In the project the network part is my responsibility and I manage to design and build the networked game system for both LAN and Internet. The network is not like a plug-and-use section, so as a matter of fact I need to integrate the network in the current game system. Although both the LAN and Internet connection is based on the same type of network, their API and actual functions varies from each other.

In this article the network we refers to will be the network solutions provided by Unity. In Unity, it is not very difficult to set up a network. However, the point is how you can implement the network function in your game to make it workable online. In order to do that, we must gain a full understanding about the network mechanism we are handling and what tools do we have to help us out. They might have a huge difference from what you thought, but I will explain all these in detail in the following section.

In this case, I would like to start with the network functions provided by Unity.

4.1 Network Elements in Unity

4.1.1 Network Initialization

Network is the communication between two computers Basically the network's concept will include the role of a server and a client. The client will keep requesting information from the server, while the server will respond to the request from the client. Since the network functions of Unity are based on p2p (peer-to-peer) network, any machine can initialize as a server, or connect to an existing server as client. Once the two parties join the same network, like a computer initialize as server and another computer connect to it as client. Once the connection is successfully established, the two computers will be able to exchange the game data during game play. The Unity’s network API include some quick method to help establish a network, it would be easy to set up but to maintain and manage the network will require careful implementation and the cooperation of the other network elements.

4.1.2 Network View

Network Views are the main component involved in sharing data across the network in Unity. It can be attached on particular game objects, which might be active or need to send related data on the network. Technically speaking, it is identified across the network by its NetworkViewID, which is basically just an identifier that is negotiated to be unique among the networked machines. Normally it is presented in the form of a 128-bit number but will be compressed down to 16 bits when transferring through the network. Every packet arrives on the client side will be applied to a specific Network View according to the specific NetworkViewID. In this way, the information will be sent to the correct game object, the message will also be unpack and applied in the way we implement.

The Network View has two ways to send data. The first way will be the RPC (Remote Procedure Calls), which is a way to invoke the function remotely. In the project I apply this feature to implement the global invoked functions. And the second way will be State Synchronization, which can be used to send special data in the way you want it or deal with the income packages. They are all some basic functions and we need to build the self-defined network functions base on them. The technical details will be introduced right below.

4.1.3 RPC (Remote Procedure Calls)

The Remote Procedure Calls is a mechanism, which allows you to invoke a function remotely. It is usually used to execute some event on all the clients in the game, and you can also apply it to pass event information between two specific parties. Generally, you can create any RPC method however you like. But you need to pay attention to the following details:

1. The RPC call do not have a limit on the number of parameters, but the network traffic will increase as the number of parameter. Some the less parameter an RPC have, the better the network performance will be.
2. Only certain types of parameter will be accepted in RPC, which includes int, float, string, Vector3 and Quaternion.
3. You can also use the internal struct NetworkMessageInfo which holds additional

information.

4. The script including RPC call should be attached to a game object, which have a Network View component on it. If the RPC is used exclusively then the Network View's State Synchronization can be set to off.

4.1.4 State Synchronization

The State Synchronization is another way of send data across the network by Network View. In fact, you can select a certain mode for the State Synchronization on the Network View Component in Inspection. The modes includes Reliable Delta Compressed, Unreliable and Off. The off mode, as mentioned above, will turn off the function of State Synchronization when you only need to use exclusive RPC. Here we will focus on the other two modes.

Reliable Delta Compressed mode, as the first word in its name, is a reliable way of sending and receiving data. During the data transmission, it will automatically compare the data that was last received by the client to the current state. By keep sending the same UDP packet until the receipt is confirmed, the Unity make sure every packet arrives destination reliably. If a packet is lost, the later packet will be kept in a buffer until the lost packet is re-sent and received.

In Unreliable mode, the sender will pay no attention to whether the data has been received. In this case, it might cause error when you are sending the data only if it is changed. However, if you keep sending the state all the way, the Unreliable mode will have less delay and require less bandwidth to do the job.

In Network View, you can set an observed object as the target of State Synchronization. For example, if the observed object is a transform, then the Network View will automatically send and synchronize the position, rotation and scale of the transform object. You can also set the observed object to a specific script. However, you need to implement a method called OnSerializeNetworkView, in which the way of sending and receiving date should be defined. In this way, you can send any specific data as you want or do anything you would like.

The State Synchronization seems to be a good way to synchronize data through the network, however it actually has some impact on the fluency of the game and we need to make some improvement on the script. The problem and solution will be discussed in detail in the section of network optimization.

4.2 Network Solutions

In this project we intends to implement the network support for two different types of network connection. The first one will be the LAN (Local Area Network) connection, which will provide the multiplayer game experience for the people in the same local network. And the second one will be the Internet connection, anyone with Internet access can player together no matter which network they are in. And as for these two different network game modes, we have select different network solution to build the network.

4.2.1 Unity Standard Networking

For the network support for LAN, we use the native network API provided by Unity. Since the standard networking elements have already be introduced in detail in the above section, I will skip the description part for it. The main reason we select Unity standard networking to be our network solution is that it only acquire the IP address and Port number, which is quite easy for the people within the same local network to tell each other. Some people might think that if we can connect through the Internet we no longer need LAN connection. However, the LAN connection has its own advantage. Independent from the Internet will keep the game from being disturbed by the network condition on the Internet.

4.2.2 Photon Unity Networking

In the part of Internet connection, we select Photon Unity Networking as the solution for network support. It is a free solution available on the official Unity Asset Store. And now let me show you some features provided by this network solution.

- The Photon Unity Networking provides cloud solutions, we can use room name to identify different game and find our own game. The network address stuff will no longer be bothering.
- Although both Unity standard networking and Photon Unity networking have similar networking mechanism to p2p network in the network communication

section. There is still some difference between them. In Unity standard network, all clients are connecting to the same server. So if the server player leave the game the connection might be dropped. However, in Photon Unity networking, the hosting is handled on the cloud by dedicated Photon server. The connection will carry on as long as the room exists.

- The Photon Unity Networking has developed a different but more abundant API base on the Unity standard networking. It can provide a better connectivity as well as performance than the Unity's own native API.
- Although the network mechanism used by both Unity and Photon are quite similar, you need to develop with different API. So in fact I need to implement two different version of game system, one works with Unity standard networking on LAN and the other one will be working with Photon API on Internet.

So we have seen different network solutions for Unity on networking. However, before we actually start the process of implementation, we need to figure out how the network mechanism works.

4.3 Network Mechanism

In the section I will introduction the main network mechanism applied in our Network System, which will be the p2p (peer-to-peer) network. The p2p network is a decentralized communication models in which each parties has identical capabilities in network communication sessions. In the model of server and client, the two different parties have clearly different role in communication, the client will be sending request and the server will fulfill the request. However, since our game is a small real-time action RPG game, the server and client mode is a little too much. What we intend to build is a lightweight p2p network, with this mode anyone can set up a server and the others can connect to start a multiplayer game. Although the both the Unity and Photon claim that their network model is not p2p network, they are refer to the hosting features. In Unity the network connection is host by the server player, in Photon they have offered dedicated server to host the connection. However, apart from that, the network communication of both solutions is following the p2p network mechanism. So how do different game instances communicate with each other? I will describe the network communication model in detail in the following section.

4.3.1 Network Communication Model

Since the WOW (World of Warcraft) hit the game market in 2003, the MMORPG have become more and more popular as time pass by. In recent year some real time multiplayer game such as DOTA (Defense of the Ancients) and LOL (League of Legends) also rise and attract many teenager players. Although many people have seen or even played the networked game, they do not have a full understanding about the network model behind.

To many people, they might think that online multiplayer online game is like creating a huge “room” on the Internet, and then when new player join the game, a new character will be assigned to him. And everybody just play in the same world. That is the point of view from normal people who do not have related knowledge or experience in network communication.

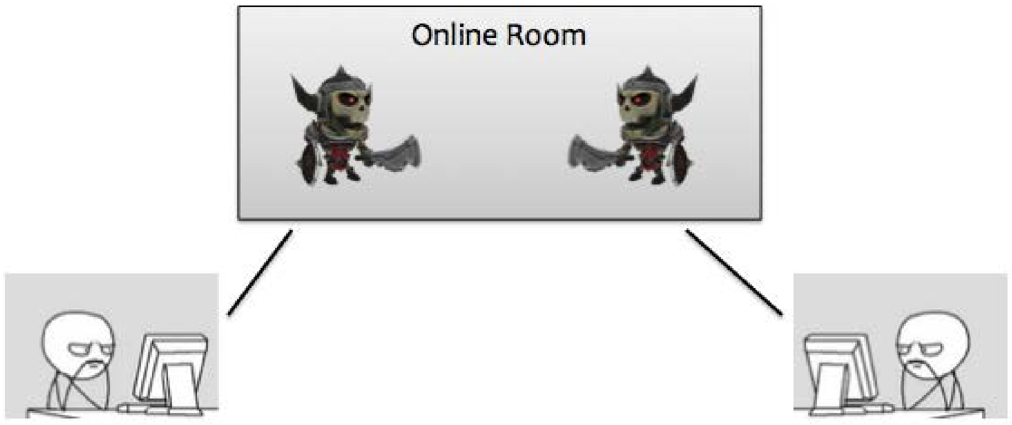


Figure 3: People’s opinion on network game

However, what would actually be like when we are playing multiplayer game with the other player online? Let’s check it out.

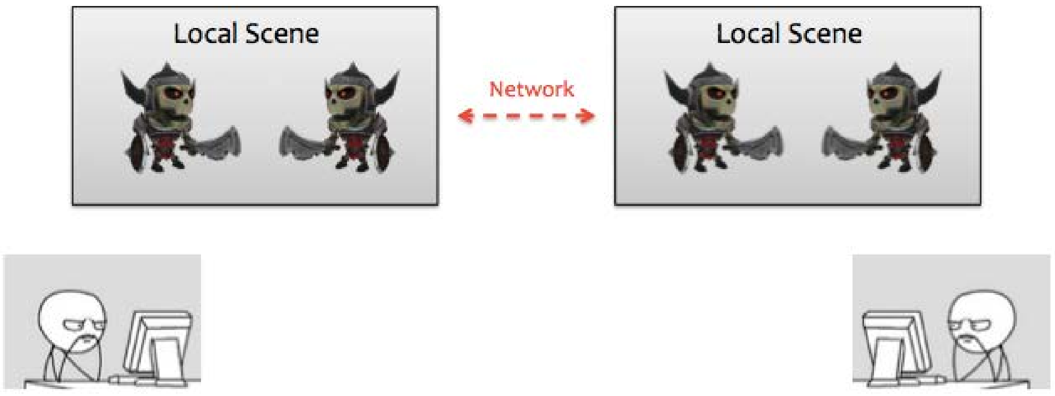


Figure 4: The actual situation of network game

What actually happen when we are playing network game might be different from what people thought they would be. As a matter of face, when we are playing online game, the game is not online. Actually the game is still a local instance running is your own computer. When you are playing you are always playing in your own local scene in the game. You might ask: so what is the network going to do? What the network actually does is to send the information about the network object or some game data, which will update your local game to the latest situation.

For example, in the picture above, when a player join a network game, the network will automatically generated the game objects (in this case, another player) on the network in his local game scene. All the players besides the local player will be the network mirrors of the original players in other player's local scene. As the game proceed, the network will keep updating game information between the two game scenes to make sure the two players are having the best real-time game experience.

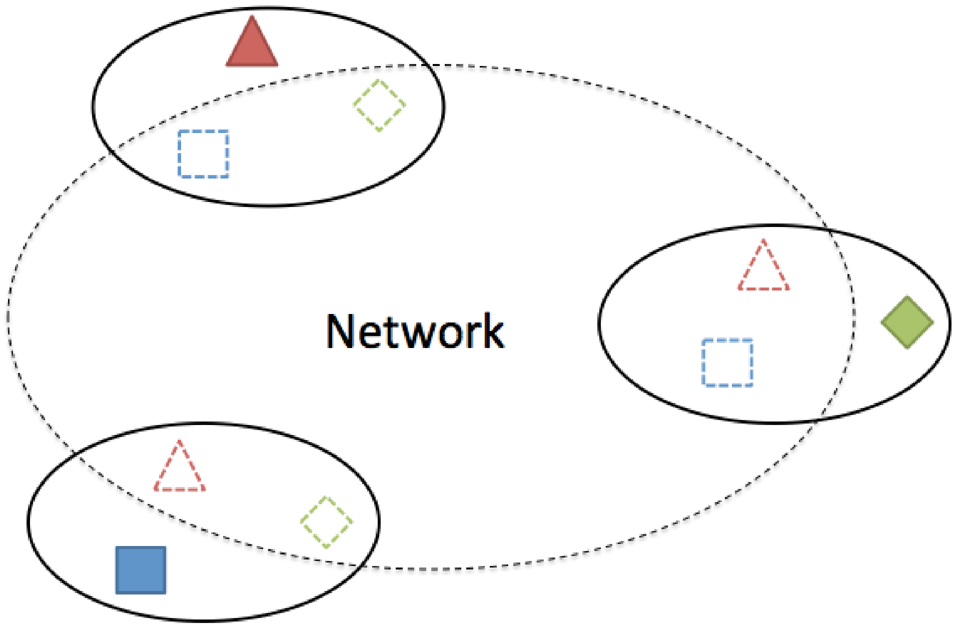


Figure 5: Network Communication Model on Network Game

4.3.2 Network Communication Procedures

Before we start the implementation of network system, we should also clarify the whole procedures when we are playing the networked game. Generally the game flow for the single mode game will be like:

1. Start the game.
2. Play the game.
3. Quit the game.

According to the game flow above, we can also design a general event flow for the network system we are going to build, just as followed:

1. Establish the network connection when the user selects a certain game mode.
2. Manage the communication of different game objects and handle network events during the game play.
3. When a user quits the game, destroy all the game objects related to him and disconnect from the network.

As a result, I divide the functions into three major parts. According to the requirement, I will set up a network manager to mange the network connection and watch for the network objects. This network manager will be responsible for the network establishment and the network disconnection part. The detailed process will be like:

1. Implement a network manager.
2. Set up the manager in the demo scene.
3. Start connection according to user’s input.
4. Save the customized player information from demo manager.

5. Establish the network and load game scene.
6. Generate networked objects.
7. Find all the networked objects with the same network identifier.
8. Execute the functions on networked objects that can destroy the related local game objects (such as the UI elements).
9. Destroy all the networked objects with the same network identifier.
10. Disconnect from the current network.

However, the second part of the network system is not something individual from the game system. In fact the second part is about modifying the original game system. By defining the way to generate network player, the way to send and receive data and some other aspects, we will change the way that game objects communicate with each other and finally make the game system fully function online as it is in single game mode. There will be a lot of problems in network communication without a proper configuration. And that's what I am going to introduce in the next part.

4.4 Network Communication Problems

To make the original game system works on network, I need to implement the original function in a totally different way, because the data communication in network environment has huge difference with the single game environment. Special attention should be paid to the scope of the function, is it a local function or a global function?

We also need to select the data we need to synchronize in a careful manner, if we cannot handle them in the right way they might become huge problems. And here I will introduce some problem I often come across in the implementation of network game system. By analyzing these problems, you will also gain a deeper understanding upon the network mechanism we are handling.

Before I state the problems I would like to explain some terminologies to avoid any confusion that might be caused:

- Real Player: Refers to the local player object generated by the local scene.
- Shadow Player: Refers to the network player object generated by the network on the scene of the other clients in the network.
- Example: B join the a network game created by A, then the local game scene generate a local player (real player) in the local scene of B. At the same time the network will create a player object (shadow player) just like the local player on the local scene of A. Since there will be a network identifier which will link the real player to the shadow player, you can implement different system on them to make sure the game function is OK in both scene.

Problem 1: Duplicate Input

When we create a player on the network, if we do not make any specification on the generating method of the shadow player. We will probably meet the problem of duplicate input. The Duplicate Input Problem refers to the problem that a game object might receive the message from both the network and the local scene. And it will happen if you don't disable the function of reading input from local scene. For example, if B joined the game of A, there will be a shadow player of B in the scene of A. Since the shadow player is still sensitive to user input, its behavior will be affect by A's input. However, since the shadow player also needs to follow the data from the real player, it receives the input from both parties. In this case, the action of the shadow player will become unpredictable and will probably goes quite differently from the real player. When it happen the two scenes will no longer be synchronized and the game system will be in chaos. To avoid such situation, we should pay more attention to the network generation part.

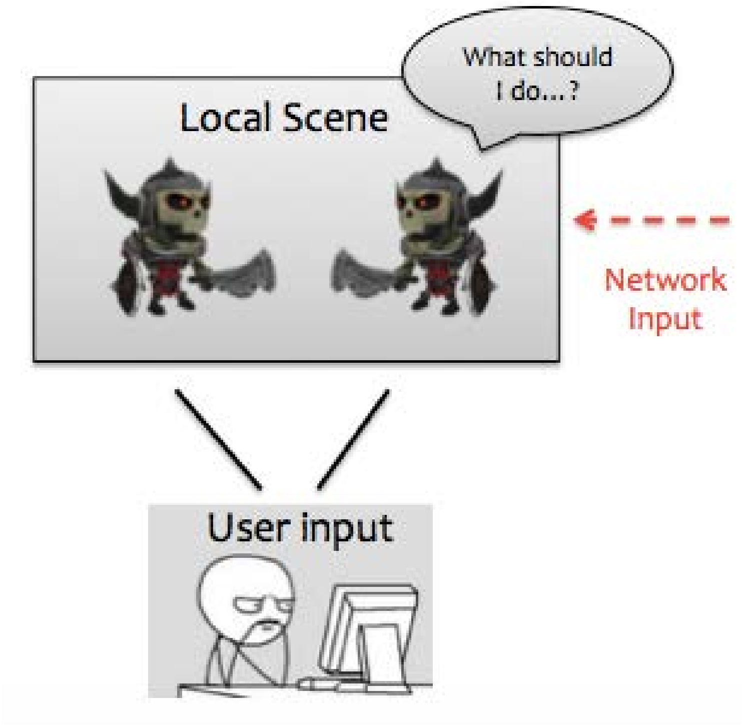


Figure 6: Duplicate Input Problem

Solution: When a player is generated by the network, disable some functions of the player and make it just follow the behavior of the real player. In this project, I disable the player movement system and the player combat system on the shadow players, and fixed many input functions to avoid such problem.

Problem 2: Interaction Failure

If all the system on the shadow player is active there will be huge problems. However, if all the system on the shadow player is inactive we would have an interaction failure problem. What is an interaction failure problem? I will go through this with a simple example.

In the network model we can see that besides the local player, any other player is generated shadow player created by the network. That means when you are interacting with the other player, you are not interacting with the real player. Actually you are interacting with the shadow player created on your local by the network. It goes in the same way in the local scene of other player, what they have in their local scene is your shadow player created by the network. So in fact the shadow player plays an important role in interacting. It serves as an intermediate in the interaction between real players. So to make the interaction works we need to make special implementation on the shadow player to make sure it can receive and react to certain events.

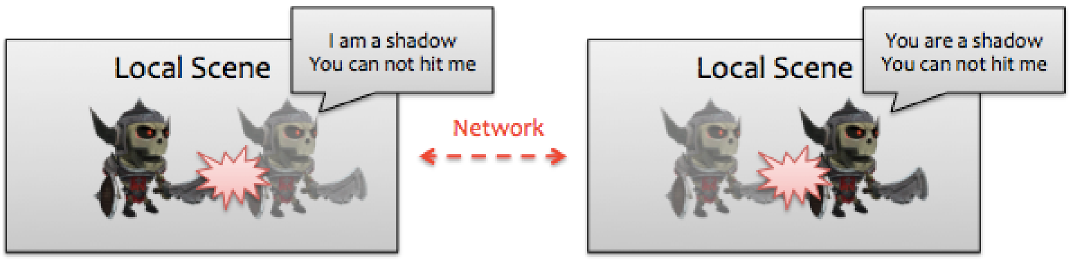


Figure 7: Interaction Failure Problem

Solution: By analyzing the game system, find out all the event-driven functions in different game systems. Find out the functions that are essential to the interaction between different game objects, such as attack and heal. Make sure the event can be received by the shadow player and be sent back to the real player.

Problem 3: Duplicate Update

For the functions driven by the events, we should pay extra attention to them. Or the problem of Duplicate Update will be caused.

Generally, both the real player and the shadow player are able to respond to certain events and make change. However, we should make sure the same thing only happen once on all the clients' network. For example, both the real player and shadow player can receive damage events and update their current HP. However, if we do not apply proper implementation on the attack function of the enemy, then enemy will be following the original attack function, which will call an attack event locally. As we all know, when there is one enemy in the network scene, the actual situation is that there will be one enemy on each client's local scene. If the enemy attacks the player, all the enemies in all scenes will attack the same player (no matter real player or shadow player). Since all of them will receive attack event and update the damage to HP, the damage will be update for more than once.

Or we can make another simple example. In single game, the player will recover 1 HP per second. If this time-driven function is effective on all the shadow players, then suppose we have five clients, the player will be recover 5 HP per second because the update will be called on each player once per second. In this case, we need to make the update function only available on one player.

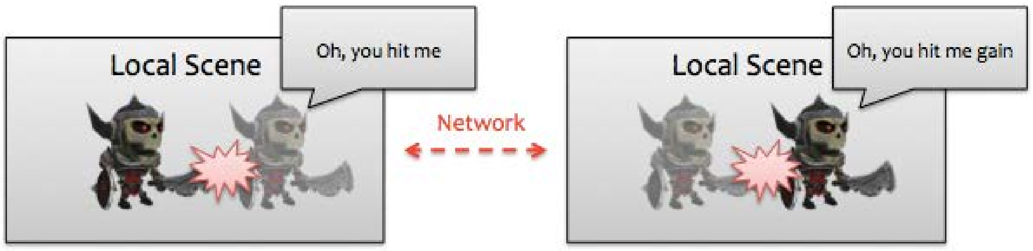


Figure 8: Duplicate Update Problem

Solution: Pay more attention to the event-driven functions and make sure the events only trigger once in all the game scenes connected to the network.

4.5 Network Communication Patterns

In order to avoid the problems mentioned above when we reconstruct the game system with network functions, I design several different network interaction patterns, which can be applied to the original game systems. And I would introduce them one by one in the up coming paragraph. By matching these different patterns on different functions or parameter in the game system, we can have a better view on the complex game system.

Events: Local vs. Global

Events can be refers to the functions in the game systems. Here I design two kinds of events, which are the local events and the global events.

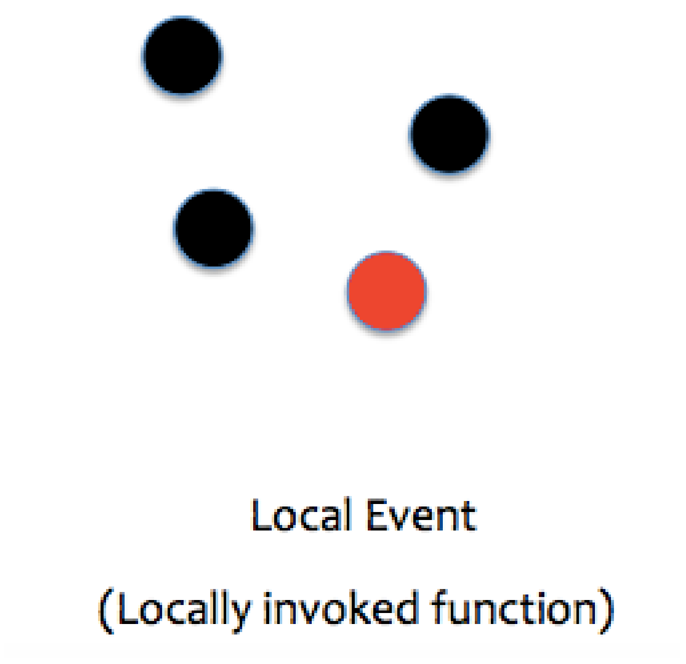


Figure 9: Local Event

First I will introduce the local event. The local event refers to the events which can only be invoked by the real player locally. In game system, this refers to some functions that can only be called locally. For example, "move" can be defined as a local event, because I can only control the real player and I hope the shadow player just simple follow the movement of real-player. In this case, the move function is set to be only available locally.

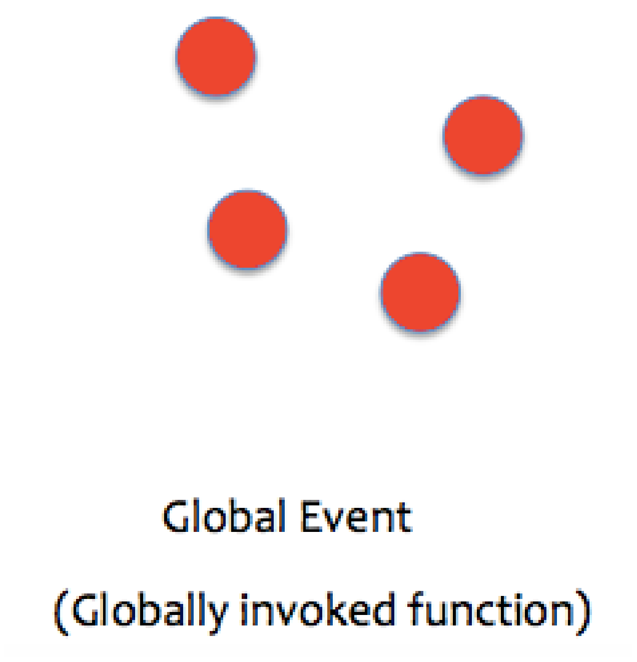


Figure 10: Global Event

As for the global events, it refers to the event that once invoked then will be perform on all the clients. In other words, it refers to a global invoked function. This event is also very important because it will have less delay than transmitting data. It is very useful for creating related visual effect. For example, when the player is moving, you can invoke a global function to perform walking animation on all the players. By invoke this function globally we do not need to send extra data.

Data Transmission: Single Broadcast vs. On Change Broadcast

In the network game environment, the different approaches of invoking functions are useful. However, to ensure the better accuracy and fluency, only modify the functions is far from enough. Beside some functions include random factors. If it is called globally it will probably have different result. To keep a high level of synchronization, we need to make sure some key data can be properly transmitted and applied. However, similar to the network events, there are also two different pattern to send data.

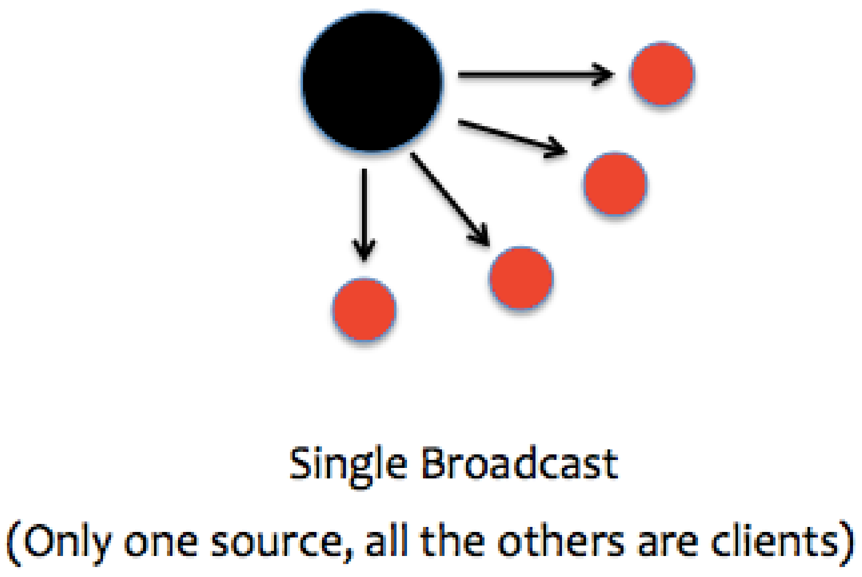


Figure 11: Single Broadcast

The first way is the single broadcast pattern. In single broadcast pattern these will only be one broadcast source, and all the other will be the receiver. Normally the source will be the real player. It is usually used combining the local event pattern, so as to sent some data as the reference for the other shadow player. For example, “move” is a local event which can only be locally invoked. However, by keep sending the position data to all the shadow players, the shadow player will act highly precise as the real player do.

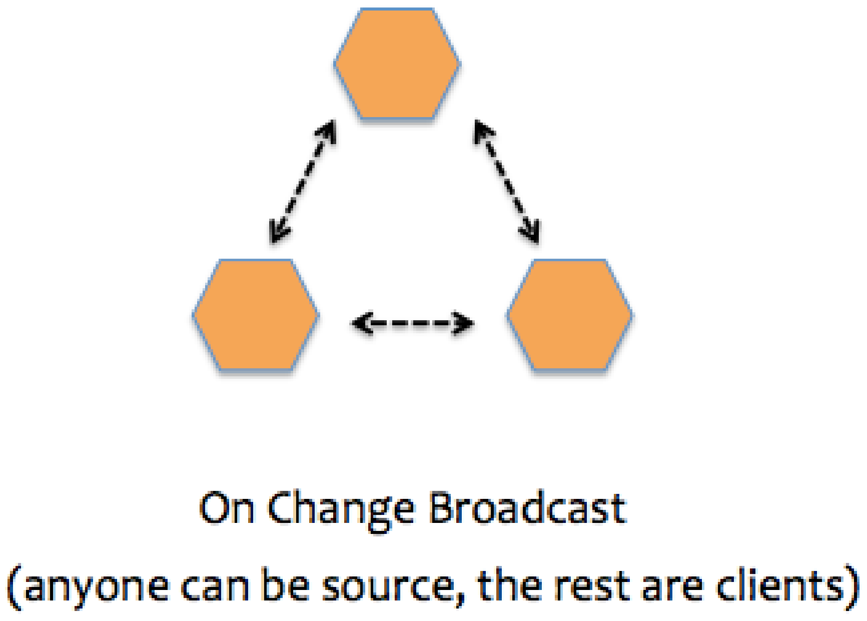


Figure 12: On Change Broadcast

As for the second pattern, which is the on change broadcast. All the nodes can be a source only if a change happens. For example, if a shadow player is attacked, it will get damage and the HP will be reduce. Then the attacked shadow player will ask all the other game objects with same identifier to update their HP to the latest. It works the same if the damage happens on real player. In this way, the shadow player can receive interaction event and update some key data with all the other network objects on the network.

4.6 Network Communication Example

Since I have designed several different patterns which can be applied on data or functions, what I need to do is to analyze all the functions and data in different game system, define a pattern and implement the network functions according to the pattern.

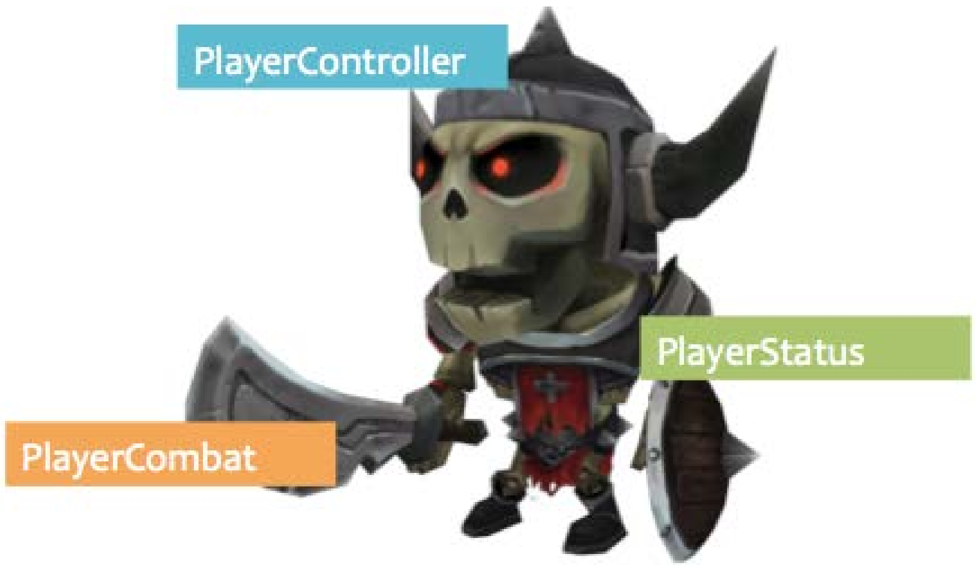


Figure 13: Different game systems to be modified

It is a huge job because the functions, data and communication in the original game system are already complex enough. And here I will show you how a simple attack event is operated over the network.

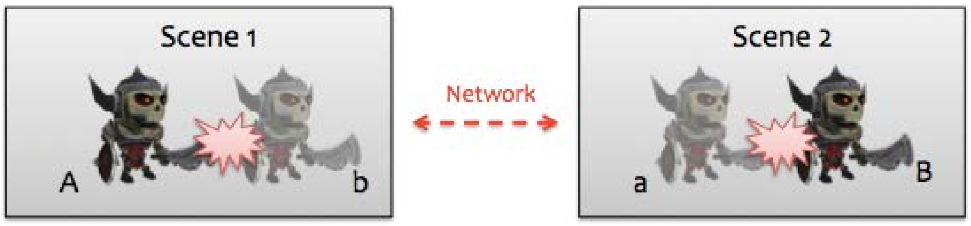


Figure 14: A successful attack action on network

The event flow of an attack action on the network is as followed:

1. A move to b. (User invoke <u>local event</u> “move” to move player A to the position

of shadow player b. By sending the position data with single broadcast, the shadow player a also move to the same place in the other scene).

2. A attack b. (User invoke local event "attack". As a result, the real player A cast attack on the shadow player b only in Scene 1. By sending the animation state with single broadcast, the shadow player a in Scene 2 also perform an attack animation to real player B).
3. b gets damage. (The local event "dodamage" is invoked by the local event "attack". However, after apply the damage on shadow player b locally in Scene 1, the latest HP is broadcast with on change broadcast).
4. Hit effect generated on both B and b. (The "dodamage" also invoke a global event "GetHit", which will generate hit effect. Since it is a global event, it is executed on both Scene 1 and Scene 2).

The above is the network activities involved in a simple attack action. It is a small example but can generally explain what I mainly do for my project. To make the new game system works on the network, I need to apply network patterns to every single function and data in this complex game system. If the functions do not suit any of the network pattern and then I need to rewrite and match again.

4.7 Network Optimization

At last I want to talk about the network optimization solution included in my network system. I have mentioned that the Unity's State Synchronization might have bad impact on the fluency and performance of the network in the section which describes the Unity network elements. Now let me explain what impact it would have and how I deal with it.

As we all know, one way to synchronize the game object's position and rotation is to set the game object's transform as the target observed by the Network View component. But this is not a good way. Because when the Network View synchronizes the position information, it actually just receives the latest position and makes directly change. It might looks on problem when the network condition is fine, when the update frequency is high and there is only little difference between the new position and the last position. However, when the network condition is bad, the position information will have larger difference, and you will found the movement of the game object will become highly inconsistent. Under such situation, the way of directly modification will bring an experience of "jump" rather than "walk".

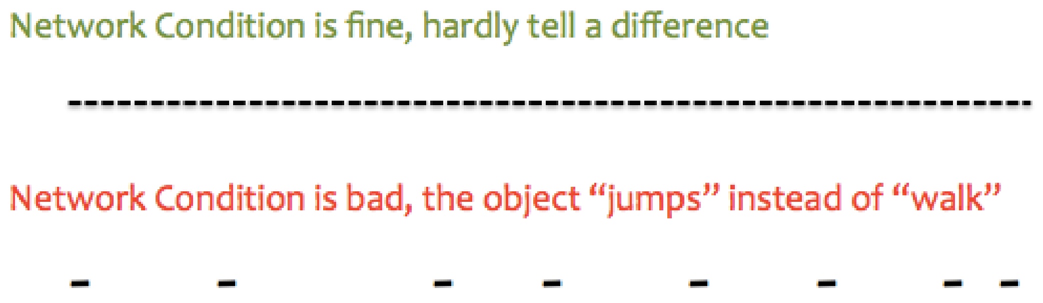

Network Condition is fine, hardly tell a difference
Network Condition is bad, the object "jumps" instead of "walk"

Figure 15: Network condition's effect on position data synchronization

So how can we just smooth the motion despite the terrible network condition? I just come up with an idea and implement it in the network system. The details of the solution is as followed:

1. Create a buffer on the receiving side of the game object.
2. Receive and store the position information from sending side in the buffer.
3. Use a Lerp() function with the position information in the buffer one by one.

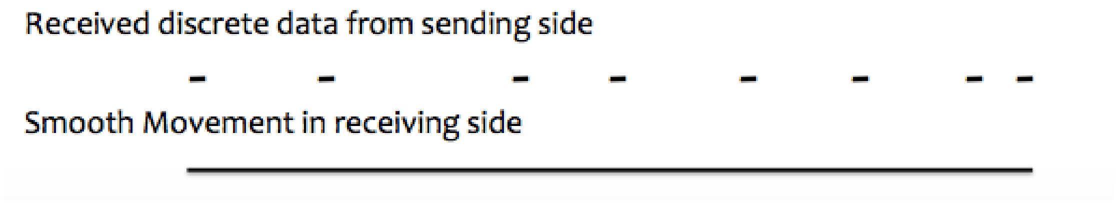

Received discrete data from sending side
Smooth Movement in receiving side

Figure 16: Apply Lerp to transfer the discrete data into smooth movement

The Lerp() function is a function provided by Unity general API, it takes two positions and a time as input, then perform smooth transfer between the two position within the time. With this method, the game object can move smoothly despite the discrete data received.

4.8 Network System Overview

After months of hard work, the network system is able to support the game in different network modes finally. Now Let’s check out what is achieved by the network system.

- ✓ Multiplayer Game Support for LAN Connection
- ✓ Multiplayer Game Support for Internet Connection

- ✓ Any game instance can serve as server or connect as client
- ✓ Successful network optimization, high-level of fluency is achieved
- ✓ Consistent and stable gaming experience

5 Conclusion

In this article I have gave a detailed description on the design and implementation of the skill system and network system. In the part of skill system, I show how we design special skills according to the difference between the three characters. The detailed structures as well as the communicating information are given to gain a full understanding on the working mechanism of skill system. In the second part, which is about the network system. I introduce some basic network elements that might be used to achieve network functions base on the original game system. Then I describe the basic network communication model in multiplayer game, providing a full view on the actually way that different game instance communicate with each other in the network. Some common problems and solutions are also mentioned to indicate the potential errors that might occur during Unity networking development, as well as the way to avoid them. By matching the original game system functions and data into some network patterns, we can perform the implementation with better efficiency. In the end, an optimization method is introduced to smooth the network synchronization on the position of game object.

As the outcome of this project, I implement two different network game systems. One of them supports the LAN game while the other one support the Internet game. As the network systems works perfectly fine under different network circumstances, our project finally ends in a satisfactory way. From this project I have learnt a lot about the network functions in game development. And I also gain the experience on how to design and implement network system in Unity. As we all know, there is no standard solution on networking for every Unity game. What you can do is to understand the network mechanism, and then try to apply them in your game system. Finally I hope this article can be enlightening and helpful for your game development in Unity!

Reference

[1]. http://docs.unity3d.com/Manual/net-HighLevelOverview.html

[2]. http://www.paladinstudios.com/2013/07/10/how-to-create-an-online-multiplayer-game-with-unity/

[3]. http://searchnetworking.techtarget.com/definition/peer-to-peer

[4]. http://doc.exitgames.com/en/pun/current/reference/pun-and-un

[5]. http://docs.unity3d.com/ScriptReference/Network.html

Acknowledgement

First of all, I want to express my sincerest gratitude to my project supervisor Dr. T. W. Chim. He had been providing solid support from day one and all the way along through the project. His instructions are always enlightening and his assistance is very supportive. He is just like a friend of mine and I really enjoy working under his guidance.

And I also want to give my credit to my teammate Sun Yining, who design and development the basic RPG game system. She is a reliable teammate and I am happy to work with her.