Network Game Development Assessment: Technical Report

2103587 Zihang Jiao

I. Introduction

My first intent was to build a moving and shooting game when starting this project. Players shoot bullets at each other, the position information of bullets and players are sent through the network, and synchronized between different machines, so that all machines are capable of showing a consistent view of the game world;

In this report, I aim to provide the reader with an understanding and justification for making decisions. This report should also act as a starting point for further development of the game and reflect on the achievements and difficulties throughout the game's development.

II. Game Control

After compiling the game, the user will perform as server or client. Clients can connect to the server to transfer information to each other. If the user chooses to play as a client, and there is no valid server to connect to, the client will be lost connections, and an error message will be printed. In the game I made, only one server is permitted to exist simultaneously. If the user wants to be a server when there is already one existing, the user will be regarded as a new client.

The game compiles at 60 frames per second. The players can use "WASD" key buttons to move the player in four directions and press "J" to shoot. There is a concept of "face direction," representing the player's last direction, which is also the direction of the bullet when the player press"J." Every time the player presses the "J" button, if the time difference between the last shoot is more than 0.1 seconds, a bullet will be shooted. If the Player is hit by the bullet, the HP will be deduced, if the HP of the player is drop to 0, the player is regarded as dead. Dead player can press "R" to resurrect from death.

III. API

I am using SMFL to build the network. The function I use is sf::TcpSocket from SFML2.5.1. SFML function is object-oriented, and the library is well written, which highly increases the software-development productivity. I am using non-blocking IO to build the the network, which allows a single process to serve multiple requests at the same time. Instead of the process being blocked and waiting for I/O operations to complete, this increases the responsiveness of the game.

IV. Architecture

The network structure I use for this game is a client-server network, which clients can connect and transfer data to the server. The server receives the data and sends it to all the players. After compiling the game, the user will perform as server or client. Two threads will be compiled if the user chooses to play as the server. One thread is used to build the server, while the other runs the client code. If the user chooses to play as a client, the client code will be executed and connect with the server. If there is no valid server to connect to, the client will lose connections, and an error message will be printed. In the game I made, only one server is permitted to exist simultaneously. If the user wants to be a server when there is already one existing, the user will be regarded as a new client.

One advantage of using a client-server network is that it can efficiently solve the data conflict. The player that chooses to be a server will become the server host. The server host determines the player's HP, which can be regarded as the most critical data through the game.

V. Protocols and Integration

A. Application layer Protocol

There are three types of information that need to be transported through the network. They are host, player, and bullet information, respectively. The format of data sent is an int, representing the type of the data, followed by the content of the data, which is usually int or float.

1) host-server information: When the server is established, a list called NetworkPlayerList will be created. When a new client is connected to the server, it will be pushed to the end of the list. The client at the head of the list is regarded as the host-server. When the first host-server is selected, The server will send a package in the form of int to the client that selected as host-server, If the host-server leaves the game, it will be deleted from the list, and the server will send confirming data to the new host-server.

2) player information: The player's information is contained in three parts: the player's position, moving direction, and HP.

The player's position is sent to the server every 1 second. The data format is two floats representing the player's x and y coordinate. The server broadcasts the data to all other clients after receiving the packet. After other clients receive the data, that player's position is updated by its new position on their client.

The player's direction is used to indicate which direction the player is facing. The data is sent to the server every time the direction of the player changes. The data format is two floats, representing the direction in x and t coordinate, respectively. After knowing the player's direction, other clients can change the animation of which direction the character is facing. The direction will also be sent when the new client is connected to the server to generate the facing animation properly.

The HP information can only be sent by the server host . If the bullet hits any player at the server host's client, the server host will send a float package, which indicates the player's updated HP to the server, then the server will broadcast the information to other clients. When a new client is connected to the server, all players' HP information will be sent to the new client to acknowledge the current situation.

3) bullet information: Sending the direction and position of every single bullet through the network is a heavy job. In this case, we only send a int to represent a bullet has been shoot by the player. The bullet position is determined by the local client. One drawback of this technique is that, when a new player is connected to the server, the existing bullet will not be shown properly, if the player is hit by the bullet, the HP will not change, until the server-host synchronise the HP information through the server. However, if the shoot rate of the bullet is not very large, and the speed of the bullet is fast enough, this problem is negligible as the time interval is short.

Package type	**package data type**	**send approach**
0(Player ID)	A packet contains int 0 represents the datatype, followed by a int represents the ID of the player	When the client is successfully connected to the server, this packet is sent from the server to the client to acknowledge the player ID
1(Disconnect)	A packet contains int 1 represents the datatype, followed by an int represents the ID of the player	If a client leaves the game, it will send a disconnect confirmation packet to the server. The server will broadcast the packet to all other clients.
2(Full server)	a packet contains int 2 represents the datatype	If a client tries to connect to the server and reaches the maximum limitation, a packet is sent to the player to acknowledge the connection has failed.
3(Player direction)	a packet contains int 3 represents the datatype, followed by a player ID, the moving direction and facing direction in x, y-axis respectively	When the direction of the player changes or a new client has joined the game, the packet will be sent to the server.
4(Player position)	a packet contains int 4 represents the datatype, followed by player ID, followed by the x,y coordinates of position respectively	The packet is sent every 1 second.
5(Server-host confirmation)	a packet contains int 5 represents the datatype	When a client is connected to the server, it will send this packet to the server. If that client is assigned as host, the server will send a packet back to the client. If the previous host left the game, the server would send the packet to the new host.
6(Player name)	A packet contains 6 represents the datatype, the player ID and name respectively	When the client is connected to the server, it sends the player ID and player name to the server.

Table I: Packets information table

Package type	package data type	send approach
7(Player list)	a packet containing int 7 represents the data type and all players' names and id.	When a new client is connected to the server, it will send a packet containing int 7. Then the server will then send the name and id of all other players back to the client.
8(Bullet event)	a packet containing int 8 represents the data type, an int represents the player ID, and an int represents the shooting event.	when a player shoots a bullet, it will send a packet to the server. The server will broadcast the packet to all other players.
9(Player HP)	a packet containing int 9 represents the data type, followed by the player's ID who has a change in HP, and a float represents changed HP.	The packet can only be sent from the server host. The server will broadcast to all other clients.

Table II: Packets information table

B. Transport layer Protocol

In this project, my choice was to use the transmission control protocol. Compared to UDP, TCP is more reliable. It confirms that the data can be sent to the target client. As we use the server-host to be the truth value when handling conflicts, TCP makes the data transmission between server-host and server more reliable.

VI. Prediction

Linear prediction is used to predict the position of the player. As the player's velocity is constant, the displacement of the player is constant for every frame. The position of the player is predicted with the following functions:

$pos_{x(}next) = pos_{x(}prev) + displacement_x$

$pos_{y(}next) = pos_{y(}prev) + displacement_y$

where $(pos(next)$ represent the position of the player at next frame, $(pos(prev)$ represent the position of the player at the previous frame.

VII. Testing

The play may not have a good experience under poor network conditions. With the network lagging, it takes longer for the player to receive the packet, so there will be a noticeable latency in other players' movement. Because he HP data, the HP is determined by the host server player. In the perspective of the player with a poor network, the HP of the player might decrease then restored to maximum. Because no one has been hit in the perspective of the host server. The problem might be more severe with packet dropping. The HP data will not be updated to other player. If Player A thinks he killed another player B, but player B is still alive in the server host's perspective. If the HP confirmation packet is dropped and fails to be received by A, A's client will regard B as Dead, and the player B figure will become invisible. An invisible player will shoot Player A until the next HP packet is successfully received.

References

[1] [C++ SFML - Simple 2D Games] - GAME 1 / PART 5 - Killing enemies! [video]. Available at: <https://www.youtube.com/watch?v=Sil75qOxCW0>
[2] [How To Make A Multiplayer Game In Unity 2021.1 [video]. Available at: <https://www.youtube.com/watch?v=4Mf81GdEDU8> .
[3] [Let's Code A Multiplayer Voxel Game in C++ - The Engine [video]. Available at: <https://www.youtube.com/watch?v=4Rg1RriQZ9Q> .
[4] [Lag Compensation for Unity Multiplayer Games | MMAG 12 [video]. Available at: <https://www.youtube.com/watch?v=1-zX0E1YBH8> .