Research on Frame Synchronization Optimization Measures in Fighting Games

Yi Qi, Wen-Yuan Hao* and
Kai Gao
Xiamen Academy of Arts & Design, Fuzhou University, Xiamen, China

***Abstract*--With the continuous development of communication technology, most of user latency happens within 150ms. Thus, some fast-paced frame sync network games begin to appear in large quantities. As fighting games are featured by a large number of quick operations in a short period of time, the requirements on actual operation and screen synchronization become higher. This paper attempts to put forward some optimization measures to solve the problems of frame-synch fighting games.**

I. INTRODUCTION

In the virtual game world, guaranteeing the consistency of the game is a basic premise. In addition to the basic function of communication, the synchronization mechanism has the most important task, that is to ensure the consistency of the game. The constantly optimized frame synch technology based on the Lockstep synchronization mechanism is an important measure to maintain the game consistency, ensure the game fairness and improve the experience of the player.

II. LOCKSTEP SYNCHRONIZATION MECHANISM PRINCIPLE

Lockstep is an important synchronization mechanism in the Peer-to-Peer architecture. Figure 1 gives a timeline when the player A, B, and C are online. The time slice separated by the dotted lines is called as turn. The arrow indicates that the player broadcasts his own operation instructions to other players. The initial state S0 is the same. At the end of the first turn, each player receives a set of operation instructions "I" from all players in the current game. At t1, all players' computers will automatically calculate the results. Since both I and S0 are identical, the next state S1 calculated on each player's computer must be the same. So is the principle for the second turn.

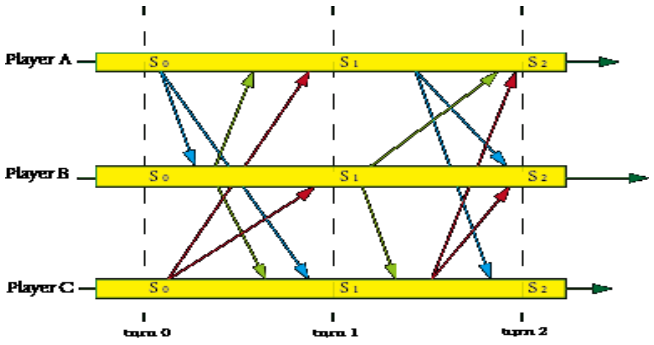

Fig. 1

However, as shown in Figure 2, Lockstep sets very high requirements on network latency. Although the final calculation results are the same, the screen of the game that uses Lockstep synchronization mechanism responds more slowly than the direct local rendering because of the network latency. So, there are games that use frame synch technology to make sure that the computer can instantly respond to the player's input.

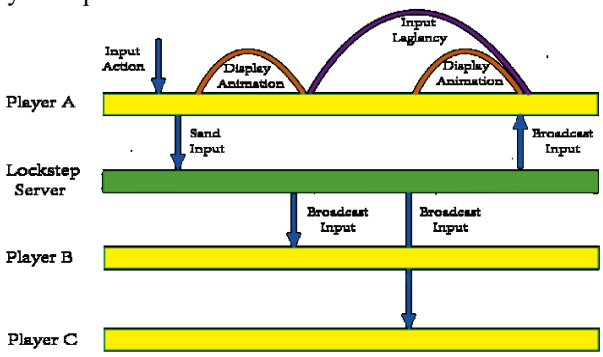

Fig. 2

III. GENERAL MEASURES OF FRAME SYNC

The first is to drive the rendering module and the logic module separately. The so-called rendering module, which is used to render the image to the local client, uses a native rendering driver and belongs to the Update () event in the Unity engine. The so-called logic module is used to run changes of all the attack determinations (character position, attack and attacked area). It is driven with network packets sent by the server, that is, to get the packets from the synchronization server before running.

The second is to correct the rendering error. Due to the discrepancy between local client-driven result and the network packet-driven result, some mechanisms are required to correct such discrepancy. There are two commonly used methods: First, timing synchronization. Pre-rendered characters stop running every 200ms (can be set as other values), and wait for status synchronization with the logic role. This method is suitable for stop-and-go characters; second, for certain inputs, such as release of attack skills, or attacked, the logic role is enabled to correct pre-rendered characters. Corrected contents include the position dragging and the corresponding animation displayed. In Update () event, 60 calls happen per second. However, usually 15-30 calls per second are needed for the synchronization server packet receipt, and thus, updating of "logic role" animation or location should be carried out at a speed that is 2-4 times faster than the normal speed.

Fig. 3

As shown in Figure 3, "logical role" and "display role" do not overlap in many cases, and there may be the discrepancy in position and the displayed animation. The Figure above is what happens within 15ms since the player presses the button "Move to Right". This will take some time to correct the "display role "according to the "logic role."

IV. RENDERING OPTIMIZATION MEASURES

First of all, it's necessary to correct the Move behavior and Fight behavior. Correction of Move behavior means to stop movement of rendered character and send movement instructions to the network at a fixed time (recommended 200ms), and wait for overlap of the logic role and the rendered character. If the waiting time exceeds the fixed interval (1000ms recommended), the rendered character will be dragged directly to the position of the logic role to guarantee the consistency. Correction of Fight behavior refers to each release of skills (local rendering). If the direct position difference between the rendered and logic roles is greater than a fixed value (100 pixels recommended), the rendered character is immediately dragged to the position of the logic role (see figure 4).

Fig. 4

After each release of skills (local rendering), wait for a fixed interval (100ms recommended) before accepting new input instructions (as shown in figure. 5).

Fig. 5

Immediately after each hit (logical rendering), the attacked party is dragged to assure the position consistency between the rendered and logic characters.see (see figure 6).

Fig. 6

V. CONCLUSION

The most important way to use pre-rendering technology is to stick with the principle of frame synchronization: the same input leads to the same output, and the rigorously synchronized game logic is separated from its presentation, so that the screen feels smoother and "consistency" of the frame sync game will not be affected. Because there is an "invisible" but correct logical model, you only need to find ways to "appropriately" correct the presentation to the correct position or state. This correction can be optimized with the "inoperable" timing of fighting games, enhancing the user's gaming experience.

REFERENCES

[1] L. A. Petrosyan ”About new strongly time-consistency solutions in cooperative differential games”, Proceedings of the Steklov Institute of Mathematics, no. 211, pp. 335–340, 1995.

[2] F. W. Li, L. W. Li, and R. W. Lau, “Supporting continuous consistency in multiplayer online games,” in ACM International Conferencon Multimedia, pp. 388–391, 2004.

[3] T. Hampel, T. Bopp, and R. Hinn, “A peer-to-peer architecture for massive multiplayer online games,” in ACM SIGCOMM workshop on network and system support for games, 2006.

[4] L. Pantel and L. C.Wolf, “On the impact of delay on real-time multiplayer games,” in Proc. ACM NetGames’02, pp. 23-29,Apr. 2002.

[5] J. Vogel and M. Mauve, “Consistency control for distributed interactive media,” in Proc. ACM Multimedia’01, pp. 221-230, Sep./Oct. 2001.