33rd European Modeling & Simulation Symposium
18th International Multidisciplinary Modeling & Simulation Multiconference

ISSN 2724-0029 ISBN 978-88-85741-57-7 © 2021 The Authors.
doi: 10.46354/i3m.2021.emss.017

A Real-Time Streaming Based VR Scenario Construction Method for the Simulation of Complex Products

Yi Lv, Yufen Wu, Xue Yang, Shuhong Xu

Beijing Aircraft Technology Research Institute,Comac,Beijing,102211,China

Abstract

During the process of simulation scene construction and visualization, a series of multidisciplinary calculation results are need to be integrated into the same complex products, and all the intricate dates should be update in real time, especially function simulation, which involving the simulation of different instruments and meters. If only one host is used for rendering, it will be taken a lot of time for the large rendering tasks, at the same time, network resources are not used sufficiently. In the paper,a server-client architecture is built through using the object-oriented modular design method, the rendering tasks that the multi-disciplinary's graphics are set by sever, and all these tasks are obtained by the clients separately, when the rendering task of client is finished and transmit to the sever in real time through video stream to integration,the method can realize the rapid construction of simulation scenarios, and provide technical support for the rapid simulation evaluation of complex products in the early stage of product's design.

Keywords:combining simulation; Model Integration; Cloud rendering;Server-client architecture; Virtual reality.

1. Introduction

With the development of the computer technology, Virtual Reality(VR) has become a novel simulation method, and begin to be widely applied in the design and manufacture of complex product, like commercial airplane, ship and High-Speed train. The VR simulation is capable to quickly build a life-like 3D immersive scenario, show the design information in a visualized way and let the designers walk into their design in a first-person view, which can meet the requirement of cross-disciplinary field evaluation and fast scheme iteration in the early development, increase the efficiency and lower the cost. However, building the VR scenario for the simulation of complex product meets the challenges of massive computer aided design (CAD) data processing, Complex data structure analysis and professional function simulation. Where the powerful computing performance and the corporation of professional visualizing software are inevitable. On the state of the art, all the works shall be done on a single PC, which may result in heavy computational load. Furthermore, due to the lack of interface between professional visualizing software and the VR scenario developing platform, the simulation results of existing software can' t be directly integrated, which leads to the duplication works and low efficiency.

This manuscript based on the features of complex product simulation, presents a Client-Server remote rendering method to improve the current VR Scenario construction mode. By distributing the works to the rendering nodes in the network, and using the real-time streaming to integrate the rendering result in the master node, the entire work can be done through the corporation of multiple PC and variety of professional software. The proposed method can greatly relieved the work load of single PC and avoid the duplication works of reproducing the simulation that existing software have already done.In the paper, section 2 introduces the processing methods and difficult points of building complex virtual scene,section 3 presents the method and detailed steps of building virtual scene of the paper,and section 4 establishes a simulation scene of virtual cockpit to verify the proposed method. Section 5 gives the conclusion and future research plan.

2. State of the art

On the state of the art, the related works of real-time streaming or the remote rendering focus mainly on the rendering method, whose target is to realize displaying the local rendering result to a remote mobile device in an efficient way.

Beiwu Liu[1] proposed a real-time streaming based massive model web browser display method, using the Hoops graphic engine and massive model browser interactive rendering tech, realized the major CAD software's model in practical engineering by simplify the complex model.For the problem of energy consumption waste caused by mismatched scheduling of rendering nodes and tasks in cloud rendering system, Qian Li[2] et al established the energy consumption model of rendering tasks,considered the energy consumption of idle nodes and running tasks, split the tasks according to the characteristics of rendering tasks, took reducing the overall energy consumption of the system as the optimization goal, and used matrix to store the energy consumption of subsequent tasks,Subsequent task scheduling improved node utilization, reduced idle energy consumption of nodes and time complexity by exchanging space for time, and improved the performance of cloud rendering system.

Khatri et.al[3]proposed a method that loaded the texture information of the model during the process of transmitting and rendering, until all the models were rendered, then all the vertex information or outline information of the cloud model was sent to the display terminal. In the course of transmission, the model was divided several regions, and used the loading mode that the visible part of the model was loaded first in the specified viewpoint position, which can reduce the rendering delay to the maximum extent under the limited bandwidth condition and improve the user experience.

Another way to improve the rate of cloud rendering is to transfer the complex rendering tasks to the server which used high-performance computer to solve, and the perspective information of the client side was fed to the server at the same time. A camera of server with the same perspective as the client in the three-dimensional scene, and transmits the rendering results to the client for display through the network in the way of video stream. Thomas, G.[5] proposed a client- server method to achieve remote rendering on mobile terminals. Vieira.et[6]presented an optimization model,which according to some constrains produces a optimum allocation of tasks to workstations and worker,and the other apporaches can not to estimate performance indicatores,but the model cannnot visualized. This method in this paper used image-based rendering (IBR) method, which made it possible to render complex scenes on mobile devices. However, when using IBR, in order to avoid exposure and occlusion, the camera need to be placed in a specific location, so its application was limited.

3. Real-Time Streaming based VR Scenario construction method

The proposed method applied the object oriented modular design. With the Server-Client architecture, cross-disciplinary visual simulation result, which delivered by the rendering node in the cloud, can be integrated in the local VR scenario through the real-time streaming.

3.1. Structure

There are two principle parts to realize the function. The client and the server.

Client deploys in the remote rendering node, works on capturing the visual simulation result given by an arbitrary third party software and sending the result to the servers.In this part, we work directly on the rendering result of the third party software, using DirectX to capture the frame in the GPU buffer, and encoding the frame to the H264 stream in real-time, finally, package it to a socket and send to one or multiple server with UDP protocol.

Server deploys in the local VR scenario building platform, which is Unreal Engine 4(UE4) in this manuscript. Server in charge of receiving the streaming from client. Server listens to the specific port and receive the original picture through decoding. The picture shall be transferred to the map of UE4 material and updated in real-time.Finally, by applying the dynamic material to a static mesh and setting the location and orientation in the 3D VR scenario, the intergrating of the remote rendering result can be accomplished.

3.2. Workflow of the method

The workflow of the proposed method is shown in the Fig.1.

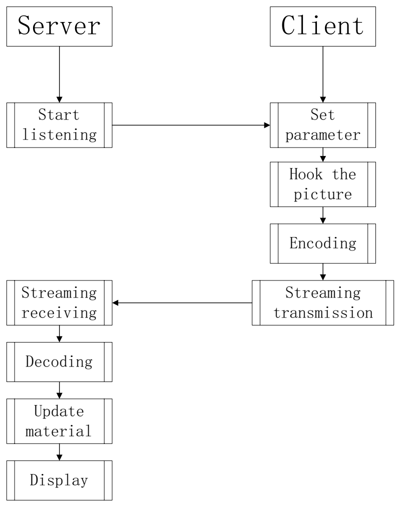
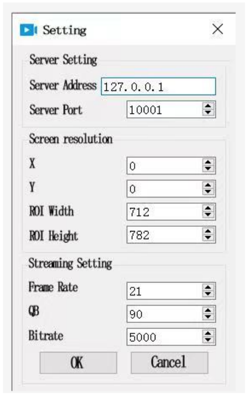


Figure 1. Workflow of the method

The workflow including：

(1) Launch the UE4 simulation project on the server, listening to the given port and waiting for the streaming data from client。

(2) Launch the client to set the parameter and begin capturing ,the UI designed is shown in the Fig.2. The parameters includes the server address, server port, capture start position, capture frame size, and some streaming quality control value. The address setting makes it possible to let multiple clients send streaming data to one server, and the port allows the server to receive streaming data from multiple clients.



Setting

Server Setting

Server Address	127.0.0.1
Server Port	10001

Screen resolution

X	0
Y	0
ROI Width	712
ROI Height	782

Streaming Setting

Frame Rate	21
QB	90
Bitrate	5000

Figure 2. Client parameters setting

(3) During the capturing loop, the ScreenCapture class shall launch the frame capture process, using DirectX to hook the current frame in the GPU and save it to the m_frames.

(4) ScreenCapture class launches the encoding process, keep encoding the frames buffered in the m_frames to H264 streaming. By separating the frame capture and encoding into two process, the flicker problem caused by waiting for capturing can be resolved.

(5) NetManager package the encoding data to socket and transfer to the specific server with UDP protocol.

(6) Server receives data and decode to the original frame, and update the dynamic material.

3.3. Synchronization mechanism

To improve the display quality a synchronization work between server and client shall be done. In the local rendering process, the vertical synchronization signal is usually used to synchronize the frequency of the rendering frame and the refresh rate of the display device. In order to avoid frame skipping and stalling, double buffering technology is introduced, and the display terminal draws the rendered frame image from the buffer. At the same time, the GPU will pre-render the next frame and put it into another buffer. When the vertical synchronization signal is received, the system exchanges two buffers to display the latest rendered frame, but if the rendering frame takes too long to miss the next vertical synchronization signal, it must wait for the next vertical synchronization signal to be displayed. In our case, we consider the static mesh to show the dynamic material as a remote virtual display terminal and create a virtual swap chain for it. The refresh rate of the display terminal is driven by the vertical synchronization signal on the server side, and the rendering frame is driven by the vertical synchronization signal on the client side. Due to asynchronous frame rendering and network bandwidth delay, the server side cannot accurately obtain the vertical synchronization signal of the client side. The rendering on the server side and the client side are not synchronized.

Using the vertical synchronization signal on the server side to drive the frame rendering on the client side can effectively solve the remote rendering synchronization problem [7]. Setting the frame rate frames per second (FPS) of the server side and the client side to be the same. Assuming that the client side renders the nth frame time is T_n^r. The display time and waiting time of the nth frame are calculated as the benchmark, then dynamically adjust the $(n+1)th$ frame rendering time of client side, as shown in the formula:

$$\mathrm{T}_{n+1}^r = T_n^r + \frac{1}{FPS} s + T_s \tag{1}$$

$$T_s = a\left(T_n^v - T_n^o - T_c - T_m\right) \tag{2}$$

$$T_m = k\Delta\theta_n \tag{3}$$

Among the above formulas, T_s is the dynamic migration time of the start time of rendering the

$(n+1)th$ frame, T_n^v is the start time of the vertical synchronization of the nth frame, T_n^o is the rendering completion time of the nth frame, $T_n^v - T_{n-1}^o$ is the synchronization waiting time of the nth frame. Assuming that the rendering time of the nth frame and the $(n+1)th$ frame are the same, which means $T_{n+1}^o - T_{n+1}^r = T_n^o - T_n^r$. In order to minimize the waiting time of the $(n+1)th$ frame, making the time for the $(n+1)th$ frame to be put into the buffer the same as the vertical synchronization time, $T_{n+1}^o = T_{n+1}^v$. But if the $(n+1)th$ frame doesn' t finish its rendering at the specified time, the n+1th frame will miss the vertical synchronization time. Therefore, T_c is introduced as the offset of T_{n+1}^o to adaptively adjust the rendering time of continuous frames, which can alleviate the problem of missing vertical synchronizing time. If the view angle of the server side is moving quickly, the content of the next frame will change significantly, so that its rendering time may be longer than the previous frame. The introduction of T_m represents the time difference caused by the rapid change of the view angle. However, the absolute distance that the viewing angle moves in one direction within one frame is very small, so in practice, the impact on the content change of the next frame is limited. However, the rotation change of the viewing angle affects the content change of the next frame very significantly. Therefore, we only consider the rotation change of the viewing angle. As shown in formula (3), T_m is determined by the change of the viewing angle $\Delta\theta_n$ and the constant scaling parameter k. We set the value of k to 100 based on experience. As a parameter of the low-pass filter, with a value of 0.1, the value of T_s in formula (2) can be then calculated.

The purpose of the synchronization using this remote vertical synchronization drive is to render the frame buffer before the next vertical synchronization signal, so as to reduce the delay while not dropping frames as much as possible. Even if some rendering frames are lost, while delaying the rendering of the frames, users can still get the latest perspective of the server to continue rendering, which maximizes the user experience.

4. Result and discussion

To evaluate the performance of the proposed method, we applied our method in a practical complex product simulation by building the simulation scenario of the commercial airplane cockpit.

During the construction of the cockpit scenario, one of the most challenge works is to draw the flight instruments dynamically. Due to the lack of the interface between UE4 and professional flight instrument rendering software, all the rendering work shall be duplicated in the UE4 with the traditional workflow. However,with the proposed method, this work can be quickly done. During simulation, the flight instruments are rendered with a third party software in a remote rendering node in the local network, as shown in Fig.3.

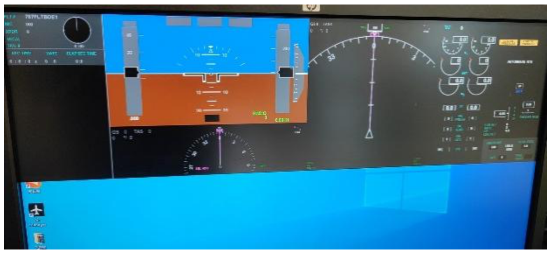


Figure 3. Rendering result of flight instrument

And the intergration result in the master node is shown in Fig3, the remote rendering result is displayed in the area identified by the red block. The testing shows that the current tool can support at least four 1920*1080 streaming simultaneously with delay up to 150ms. which proves the applicable of the method,and all the results were integrated in the virtual cockpit, as shown in the Fig.4.

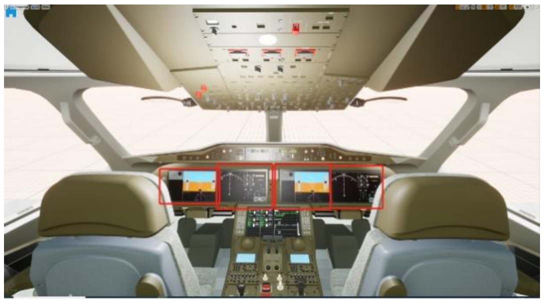


Figure 4. Intergration of the result

5. Conclusion

The current research of remote rendering technology is mainly focus on optimizing the streaming performance and reduce the interaction delay. In this manuscript we change the thought, apply the remote streaming technology to the construction of simulation scenario, to solve the problem of the integration of multi-specialty software visualization results, and meet the requirement of complex product simulation like commercial airplane. Compared with the traditional technical means, this manuscript solves the problem of interconnection and integration of the rendering results of multiple visualization software products, which can greatly relieve the workload of building the VR scenario improve the efficiency.

Funding

This work was supported by the National Key R&D Program of China (No. 2018YFB1701602).

Reference

[1] Liu Beisheng Research on cloud rendering based 3D BIM model visualization techniques, Journal of Beijing Jiaotong University,41（6）:1-6；2017.
[2] Li Qian et al, A task scheduling strategy with energy optimization for cloud rendering systems,Journal of XI ’ AN Jiaotong university,50(2):1-6;2015
[3] Khatri, Anita(1); Rishi, O.P.(1), Augmenting Cloud Service Discovery Using Ontology, Springer Science and Business Media Deutschland GmbH, 1187,193-201, 2021
[4] S. Eilemann, M. Makhinya, and R. Pajarola. 2009. Equalizer: A scalable parallel rendering framework. IEEE Trans. Visual. Comput. Graphics 15（3）：436-452，2009
[5] Thomas, G., Point, G., Bouatouch, K. A client-server approach to image-based rendering on mobile terminals. Technical
[6] Vieira Antonio A.C. et al.Combining simulation and optimization models on a production line problem:A case study.31st European modeling and simulation symposium,EMSS 2019:174-181.
[7] Luyang Liu et al,Cutting the Cord: Designing a High-quality Untethered VR System with Low Latency Remote Rendering. MobiSys’18, June 10-15, 2018, Munich, Germany