Hierarchical Rendering System Based on Viewpoint Prediction in Virtual Reality

Ping Lu[1], Fang Zhu[1], Ping Li[2], Jinman Kim[3], Bin Sheng[4], and Lijuan Mao[5]

[1] ZTE Corporarion, Nanjing, People's Republic of China
[2] The Hong Kong Polytechnic University, Hong Kong, People's Republic of China
[3] The University of Sydney, Sydney, Australia
[4] Shanghai Jiao Tong University, Shanghai, People's Republic of China
shengbin@sjtu.edu.cn
[5] Shanghai University of Sport, Shanghai, People's Republic of China
maolijuan@sus.edu.cn

Abstract. Virtual reality (VR) systems use multi-modal interfaces to explore three-dimensional virtual worlds. During exploration, the user may look at different objects of interest or in different directions. The field of view of human vision is $135^\circ \times 160^\circ$, but the one requiring the highest resolution is only in $1.5^\circ \times 2^\circ$. It is estimated that in modern VR, only 4% of the pixel resources of the head-mounted display are mapped to the visual center. Therefore, allocating more computing resources to the visual center and allocating fewer viewpoint prediction rendering techniques elsewhere can greatly speed up the rendering of the scene, especially for VR devices equipped with eye trackers. However, eye trackers as additional equipment may be relatively expensive and be harder to use, at the same time, there is considerable work to be done in the development of eye trackers and their integration with commercial head-mounted equipment. Therefore, this article uses an eye-head coordination model combined with the saliencey of the scene to predict the gaze position, and then uses a hybrid method of Level of Detail (LOD) and grid degeneration to reduce rendering time as much as possible without losing the perceived details and required calculations.

Keywords: VR · Hierarchical rendering · LOD · Hand track · Eye-head coordination

1 Introduction

1.1 Significance

With the development of new technologies such as ray tracing, the fidelity and fineness of computer graphics are increasing day by day, but at the same time, the pres-sure of graphics rendering on graphics hardware is also increasing. The

emergence of VR technology has further increased the computing power requirements of graphics rendering, which makes it paramount that we optimize our rendering technologies as much as possible. On this issue, our viewpoint prediction driven hierarchical rendering technology is based on the principal that human vision has a focused range.

Considering the real-time requirements of VR applications [6,8], the efficiency of the viewpoint position prediction method also needs to be carefully considered. The focus of conventional visual saliency prediction work [1] is mainly on the accuracy of the algorithm, but it does not provide real-time performance guarantee for VR applications. Hardware-based eye tracking is better suited to meet real-time requirements, but this method is limited by the cost of hardware, delay and other factors, in comparison, the model based on eye-head coordination, has fewer limitations and can be better to meet the real-time viewpoint rendering requirements of VR applications.

In terms of hierarchical rendering, this article adopted a hybrid method of LOD and mesh degeneracy: after degenerating the model once, using the retained vertices as the skeleton, the simplified model and the original model are saved separately, and then during the rendering process, the mesh near the viewpoint is merged back with the original model. This method can achieve grid-level precision adjustment. At the same time, no new grid simplification calculation is required during the movement of the viewpoint, which greatly reduces the calculation resource occupation of the GPU part.

1.2 Article Structure

The main operation processes of this method are loading the model and performing simplified model calculations. In this step, one or more simplified models are generated for each object, and the incremental data between the models at various levels is saved, as well as the rendering cycle. The viewpoint prediction algorithm and the model stitching step based on the predicted viewpoint are performed in this step. In the last step, the observer's current visual attention point position is predicted, and the model grid near the viewpoint is restored to a high-precision state.

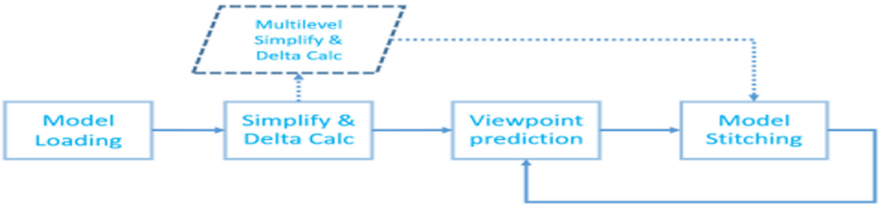

Fig. 1. Software-based predictive viewpoint following rendering framework in VR

2 Related Works

2.1 Visual Saliency Prediction

Visual saliency prediction is a research hotspot in computer vision. Inspired by the neuron structure of the primate visual system, in their paper Laurent Itti, proposed a classic visual attention model [4], which uses multi-scale image features to calculate saliency maps. After that, many models were proposed, such as the use of graph theory, the introduction of Markov absorption chain; layer extraction and training SVM for detection; the image is divided into local, regional, and global, and the center is obtained after multi-scale comparison peripheral histogram. Along with the progress of deep learning, the detection of saliency areas based on convolutional neural networks also appeared [5,10,12]. Sitzmann. [11] discussed the saliency of VR in static images. Xu, Proposed a gaze prediction model in dynamic360° immersion video [13].

2.2 Eye-Head Coordination

The eye tracker can measure eye movement and give the eye fixation position, but the cost of using the eye tracker is too high, plus it also has a certain delay, making it difficult to calculate the real-time eye focus position. Thus, software based real-time prediction is extremely important. Eye-head coordination refers to the coordinated movement between the eyes and head, which has been studied in the fields of cognitive science and Neuroscience. Yarbus [14] found that the eyes and head are in coordinated movement during gaze transfer, and there is a connection between eye-head coordination and visual cognition. In some studies [15], it was found that there is often a delay between eye movement and head movement, and eye movement usu-ally occurs before head movement. Most studies on eye-head coordination have focused on the relationship between the magnitude of head movement and eye movement, and found that the two are closely related. In VR systems, head motion information appears to be extra critical and important, based on a large number of head-eye movement data sets, Hu [3] obtained the correlation between head movement and eye position.

2.3 Model Simplification Algorithm and Hierarchical Rendering Method

Model reduction algorithm. The edge collapse algorithm was proposed by Hugues Hoppe [2] as a progressive mesh implementation in 1996. The basic operation process of edge collapse is to select an edge in the grid and delete it, and merge the vertices at both ends into one point. Each time the edge collapses, two triangles are reduced from the mesh. After this process is repeated, the number of triangles in the grid can be continuously reduced to simplify the model. Low Kok Lim [7] proposed an algorithm that uses clustering algorithm for model simplification in 1997, directly interpolating and merging the set of vertices considered to be

close to each other in the grid, means that we can greatly reduce the number of vertices and faces.

Hierarchical rendering method. Face culling algorithms are widely used in modern rendering systems. The most basic method for face culling is back culling, further algorithms are occlusion culling and frustum culling, that is, objects and triangles that are not currently in the view cone are not rendered. In the level of detail (LOD) method, we set several models with different levels of fineness for the same object, or even in some cases not render the object, and the LOD method fits well with viewpoint prediction.

3 Content and Methods

The algorithm in this article is mainly comprised of two parts: the first part is view-point prediction, and the other part is to perform hierarchical rendering optimization based on the results of viewpoint prediction. The following text will explain these two aspects in detail.

Algorithm 1. Gaze Prediction

Input: $angleX, angleY$(set of head rotation degree), $time$(set), $velX, velY$(set of head move speed), $accX, accY$(set of head move acceleration), $meanGazeXStatic$, $HeadVelXThresMin$, $HeadVelXThresMax$

Output: $resultX$,$resultY$(degree of gaze line)

cal $accMeanX, accMeanY$;
cal $velStdX, velStdY$;
for x in $angleX$ **do**
  if $velX$ in $StaticRegion$ **then**
    $resultX$=$meanX$
  else if $velX$ in $IntentionalMoveRegion$ **then**
    $resultX$=$calX()$
  else if $velX$ in $SuddenMoveRegion$ **then**
    $resultX$=$meanX$
  end if
end for
for y in $angleY$ **do**
  if $velY$ in $StaticRegion$ **then**
    $resultY$=$meanY$
  else if $velY$ in $IntentionalMoveRegion$ **then**
    $resultY$=$calY()$
  else if vel in $SuddenMoveRegion$ **then**
    $resultY$=$meanY$
  end if
end for

3.1 Viewpoint Prediction

The eye-head coordination model is predicted, mainly based on head movement information. In addition, the gaze behavior in VR is a complicated pattern, which is affected by various factors such as content, task, delay and so on. The eye-head coordination model used in this article can explain the real-time gaze behavior of users when exploring the virtual environment in chronological order. The specific formula [3] is as follows:

$$x_g(t) = \alpha_x \cdot v_{hx}(t + \Delta t_{x1}) + \beta_x \cdot a_{hx(t)} + F_x(t + \Delta t_{x2}) + G_x(t) + H_x(t) \quad (1)$$

$$y_g(x) = \alpha_y \cdot v_{hy}(t + \Delta t_{y1}) + F_y(t + \Delta t_{y2}) + G_y(t) + H_y(t) \quad (2)$$

The model separates horizontal and vertical calculations, and combines the angular velocity and angular acceleration of head movement, delay of eye movement and head movement, content, tasks and other factors. α_x,α_y,β_x are the influencing coefficients of various factors, which are solved by further fitting the data. The model is divided into three independent areas of static, intentional movement and sudden movement according to the angular velocity of the head movement. When in the static area or sudden movement area, the user's gaze position is less relevant to the head movement, and is mainly affected by the scene content, the user's psychology, and the influence of factors such as state, emergencies, etc., while in the intentional movement area, the position of the user's gaze on the screen is strongly linearly related to the rotation speed of the head. Algorithm 1 shows the pseudo code of the final prediction algorithm.

3.2 Hierarchical Rendering

In order to cooperate with the viewpoint prediction algorithm, the graded rendering algorithm of this article should meet the following requirements:

1. The rendering fineness can be determined according to the distance of the object from the viewpoint.
2. The rendering fineness of only part of the object can be reduced.
3. Do not add too much computing burden to the CPU.

To this end, the following article designed a multi-precision stitching algorithm, which is based on real-time grid degeneration, and drawing on the idea of LOD. This algorithm will simplify the model mesh to different levels when the model is first loaded, and save the simplified model and the vertices and faces that have changed during the simplification of each level, so that any part of the model mesh can be restored to high precision when necessary. This algorithm consists of simplification and splicing.

The simplification algorithm is similar to the edge collapse algorithm, but vertices are deleted instead of edges. The algorithm first scans all the vertices of the model, if the vertices are on a sharp edge or corner, they are important for maintaining the outline of the model and cannot be deleted, otherwise, check if

Algorithm 2. Simplify

Input: V(set of vertices), T(set of triangles),$threshold$
Output: V'(new set), T'(new set), $delta$(dealt data needed by the recovery algorithm)

```
set C={}; set D={}; set A={};
for v in V do
    bool canCull=true;
    for t1,t2 in t ∈ T | t contains v do
        if angle(t1,t2) <threshold then
            canCull=false;
            break;
        end if
    end for
    if canCull then
        for v2 in x ∈ V | x and v are connected by an edge do
            if v2 in C then
                canCull=false;
                break;
            end if
        end for
    end if
    if canCull then
        C.add(v);
    end if
end for
```

any vertex is directly connected to it, if none of them are marked deleted, the vertex is marked as remove-able. After a round of scanning, some vertices will be marked showing that they can be deleted. Traverse this part again and delete all the triangular faces containing these vertices. For each deleted vertex and the surrounding triangular faces, you will get a closed polygon hole. For this hole, choose a vertex, and divide the polygon into a series of triangles through the diagonal line from that point, and add it to the model mesh as a new triangle. The combination of the deleted vertex, the deleted face around the vertex, and the newly added triangular face that fills the hole of the vertex is saved for use in stitching. This simplification algorithm can be run repeatedly to generate multiple levels of simplification, the pseudo code is in Algorithm 2.

Another important part is the stitching algorithm. Compared with the simplified algorithm, the stitching algorithm is relatively simple. After the viewpoint prediction algorithm gives the observer's current viewpoint position, scan the delta set of objects near the viewpoint, find the vertices within a certain distance from the viewpoint, and restore them and the triangles around them to the simplified model according to the stored data.

4 Experiment Results and Analysis

This article implements the model in Unity, in order to compare the rendering performance before and after enabling hierarchical rendering, we recorded the CPU usage, GPU usage, video memory usage and memory usage under two conditions.

As shown in Fig. 2, the model on the screen will be significantly simplified after hierarchical rendering is enabled. Since the line of sight is on the roof of the cabin, there is no obvious change in the details of the roof, but the model of the tower on the left is obviously simplified.

Fig. 2. (a) Without hierarchical rendering, (b)With hierarchical rendering

After separately running the program with hierarchical rendering enabled and following the same route in the scene, we recorded the following data in Table 1 (the system rest load has been removed). It can be seen from the table that the GPU occupancy rate has dropped significantly after the hierarchical rendering is turned on, indicating that our algorithm can indeed further reduce the number of rendered triangles on the existing rendering optimization technology to reduce the GPU load. But it is also worth noting that the CPU usage, memory usage and video memory usage have all increased. After further testing, we found that the CPU-intensive operation is not viewpoint prediction or model stitching, but the process of returning new triangle data to the Unity engine after model stitching is completed. This process involves not only copying an array, but also recalculating normal and texture coordinates. It also takes CPU cycles to transfer vertex data from memory to video memory. It can be seen that frequent and large-scale modification of model data in 3D rendering is very expensive behavior.

The increase in memory usage is because when the simplified algorithm saves incremental data, it must write some auxiliary data, such as indexes, for subsequent correct stitching, which causes the model to occupy more memory. As for the increase in the amount of video memory, after analysis, it may be triggering a certain cache mechanism of the GPU, which saved some of the transition model data to reduce the amount of memory and video memory data transfer, but because of the fact that our model changes are too many, this mechanism fails to effectively reduce our vertex transmission overhead.

Table 1. System load changes when hierarchical rendering On and Off

Data	Off	On	Rate
CPU	17%	28%	+62%
GPU	80%	50%	−37%
Memory	2 GB	3.8 GB	+90%
Video memory	1.1 GB	1.7 GB	+54%

5 Conclusion

The hierarchical rendering method in this article achieves automatic grid-level accuracy adjustment by mixing LOD and model degeneracy methods. At the same time, with the help of the viewpoint detection algorithm, we can use a more aggressive optimization algorithm to further reduce the number of triangles that the GPU needs to render based on the existing optimization algorithm. In the case of highly complex scenes or heavy rendering loads such as VR, GPU performance becomes a bottleneck, the group's hierarchical rendering method can effectively reduce the GPU load at the cost of CPU load, thereby achieving a higher display frame rate and a smoother visual experience.

Acknowledgement. This work was supported in part by the National Key Research and Development Program of China under Grant 2018YFF0300903, in part by the National Natural Science Foundation of China under Grant 61872241 and Grant 61572316, and in part by the Science and Technology Commission of Shanghai Municipality under Grant 15490503200, Grant 18410750700, Grant 17411952600, and Grant 16DZ0501100.

References

1. Borji, A., Itti, L.: State-of-the-art in visual attention modeling. IEEE Trans. Pattern Anal. Mach. Intell. **35**(1), 185–207 (2013)
2. Hoppe, H.: Progressive meshes. In: International Conference on Computer Graphics and Interactive Techniques, pp. 99–108 (1996)
3. Hu, Z., Zhang, C., Li, S., Wang, G., Manocha, D.: SGaze: a data-driven eye-head coordination model for realtime gaze prediction. IEEE Trans. Vis. Comput. Graph. **25**(5), 2002–2010 (2019)
4. Itti, L., Koch, C., Niebur, E.: A model of saliency-based visual attention for rapid scene analysis. IEEE Trans. Pattern Anal. Mach. Intell. **20**(11), 1254–1259 (1998)
5. Kamel, A., Sheng, B., Yang, P., Li, P., Shen, R., Feng, D.D.: Deep convolutional neural networks for human action recognition using depth maps and postures. IEEE Trans. Syst. Man Cybern. Syst. **49**(9), 1806–1819 (2019)
6. Karambakhsh, A., Kamel, A., Sheng, B., Li, P., Yang, P., Feng, D.D.: Deep gesture interaction for augmented anatomy learning. Int. J. Inform. Manag. **45**, 328–336 (2019)
7. Low, K., Tan, T.: Model simplification using vertex-clustering. In: Interactive 3d Graphics and Games, pp. 75–82 (1997)

8. Lu, P., Sheng, B., Luo, S., Jia, X., Wu, W.: Image-based non-photorealistic rendering for realtime virtual sculpting. Multimedia Tools Appl. **74**(21), 9697–9714 (2014). https://doi.org/10.1007/s11042-014-2146-4
9. Meng, X., et al.: A video information driven football recommendation system. Comput. Electr. Eng. **85**, 106699 (2020)
10. Pan, J., Sayrol, E., Giroinieto, X., Mcguinness, K., Oconnor, N.E.: Shallow and deep convolutional networks for saliency prediction. In: Computer Vision and Pattern Recognition, pp. 598–606 (2016)
11. Sitzmann, V., et al.: Saliency in vr: how do people explore virtual environments. IEEE Trans. Vis. Comput. Graph. **24**(4), 1633–1642 (2018)
12. Sheng, B., Li, P., Zhang, Y., Mao, L.: Greensea: visual soccer analysis using broad learning system. IEEE Trans. Cybern., pp. 1–15, May 2020
13. Xu, Y., et al.: Gaze prediction in dynamic 360 immersive videos. In: Computer Vision and Pattern Recognition, pp. 5333–5342 (2018)
14. Yarbus, A.: Eye Movements and Vision. New York (1967)
15. Zangemeister, W.H., Stark, L.: Active head rotations and eye-head coordination. Ann. N.Y. Acad. Sci. **374**(1), 540–559 (1981)
16. Zhang, P., Zheng, L., Jiang, Y., Mao, L., Li, Z., Sheng, B.: Tracking soccer players using spatio-temporal context learning under multiple views. Multimedia Tools Appl. **77**(15), 18935–18955 (2017). https://doi.org/10.1007/s11042-017-5316-3