2019 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation

Optimization Strategies for Real-Time Rendering of Virtual Scenes on Heterogeneous Mobile Devices

Wei Gai[1], Xiyu Bao[1], Meng Qi[2*], Yafang Wang[1], Juan Liu[1], Gerard de Melo[3],
Lu Wang[1], Lizhen Cui[1], Chenglei Yang[1,4], and Xiangxu Meng[1,4]
[1]School of Software, Shandong University, Jinan, China
Email: gw@sdu.edu.cn
[2]Shandong Normal University, Jinan, China
Email: qimeng@sdnu.edu.cn,
[3]Rutgers University, New Brunswick, USA
[4]Engineering Research Center of Digital Media Technology, MOE, Jinan, China

***Abstract*—Increasing numbers of VR applications are being developed for mobile devices. Due to the various hardware limitations of different mobile devices, the amounts of data for virtual scenes need to be restricted if one seeks to still obtain fast rendering and a real-time interaction experience. To alleviate this problem, this paper proposes novel optimization strategies for real-time rendering of virtual scenes on heterogeneous kinds of mobile devices with diverging processing abilities. In particular, we deploy a virtual scene around an editable roaming path so as to reduce the arrangement of invisible virtual objects, and optimize the scene layout in the design stage. The virtual objects are bounded with respect to the roaming path, such that the system can render the virtual scenes in the roaming stage in real-time. Our experiments show that these strategies can be adaptively applied for different smartphone devices, operating in larger-scale virtual scenes constructed in advance.**

***Index Terms*—Optimization strategy; real-time interaction; virtual reality**

I. Introduction

In the past decade, mobile devices have become truly ubiquitous across all continents, providing computing, storage, as well as a number of input and output modalities. Based on this pervasiveness, low-cost VR mounts for smartphones, such as Google's Cardboard viewer [1] have emerged as a particularly widespread type of VR HMD, with an ever-increasing number of applications [2].

A challenge with smartphone-based VR systems is that their hardware performance is limited, such that the rendering speed must be taken into consideration as an important constraint in designing VR content. In order to ascertain real-time rendering of 3D scenes, some popular rendering acceleration techniques from computer graphics such as level-of-detail (LOD) [3] and visibility culling [4] have also been adopted in many mobile VR apps. Another approach is to pre-compute all possible images that a VR user might encounter and construct a storage cache in advance, so that the mobile VR systems can quickly obtain the images to be presented to the user while roaming in the scene [5]. Real-time rendering for mobile VR systems can also be facilitated by constructing a rendering service platform on higher-performance servers [6]. However, remote rendering VR contents can also have certain downsides (e.g., interaction latency). Another research direction is to study how to reduce the rendering burden at runtime by already taking the deployment of the 3D scenes into consideration during the design stage [4].

However, one important issue that has been neglected is how to adapt to the different performance profiles of heterogeneous mobile devices in the design of VR scenes. While designers may, for instance, develop various virtual museum scenes, these may not be able to run in real time on each user's smartphone, since the latter may have inadequate performance characteristics. This is akin to how different printers come with different printer drivers that may render the same print job with a different quality, in accordance with the capabilities of the printer.

In order to address this gap, this paper proposes novel optimization strategies for real-time rendering of virtual scenes on heterogeneous kinds of mobile devices with diverging processing power. In particular, we present a method of deploying a virtual scene around an editable roaming path with an optimized arrangement of virtual objects in the design stage. This enables the system to later render the virtual scenes in real-time during the roaming stage, by quickly retrieving only the pertinent virtual objects, which are bounded with respect to the roaming path in the design stage. This strategy can be applied for different mobile devices, operating in larger-scale virtual scenes constructed in advance.

II. Related Work

Real-time rendering is crucial for ensuring real-time interaction. In order to enable real-time rendering of VR scenes, there are numerous techniques to accelerate the rendering of 3D contents. In this paper, we mainly consider methods targeting mobile devices.

OpenGL ES is a well-known API for real-time rendering on mobile systems [7], but relies on developers to optimize the rendering quality. He et al. proposed an accelerated rendering method to alleviate the low performance of vector graphics rendering on mobile devices [8]. Some rendering platforms have been built to provide remote real-time rendering services for mobile VR systems. Several studies [9], [10] proposed

978-1-7281-4034-6/19/$31.00 ©2019 IEEE
DOI 10.1109/SmartWorld-UIC-ATC-SCALCOM-IOP-SCI.2019.00111

remote rendering frameworks that offload the burden of rendering 3D contents from mobile devices to more powerful servers. However, remote rendering of 3D contents on mobile devices may be impractical due to network delays.

In order to ensure real-time rendering of virtual reality scenes, there are many techniques to accelerate rendering by pre-processing. Besides LOD [3] and visibility culling [4], potentially visible sets (PVS) are often used to accelerate the rendering of 3D scenes. The latter pre-compute a candidate set of potentially visible polygons in advance and an index is used at run-time to quickly obtain an estimate of the visible geometry [11].

Finally, some studies consider how to reduce the rendering burden at runtime by reasonably considering the deployment of the 3D scenes as early as during the design stage. Kin et al. proposed an animated virtual scene layout system, which can support the user in arranging the scene in accordance with the needs of the shooting path [4]. Sun et al. proposed a genre of participatory design for 3D virtual scenes on mobile devices [12], in which designer adjusts the layout of objects according to the user's VR experience feedback. However, these studies have not considered how to adapt VR scene design to the varying levels of performance of different mobile phones.

III. DESIGN AND IMPLEMENTATION

For mobile device-based virtual reality systems, the motivation for our approach derives principally from the following considerations:

1) Despite the diversity of mobile devices, the models and performance characteristics of prominent mainstream smartphones are known. Thus, when designing a scene, one can account for this information and build an optimized virtual scene custom-tailored for particular types of devices to reduce the burden of rendering during the experience.

2) Although the performance on a particular device varies depending on which other programs are running simultaneously, this sort of information can be procured. As a result, we can specifically factor in this information in order to dynamically adjust the rendered content based on the actual available computing power. The goal is to ensure real-time rendering of the scene despite the device being in different states at different points in time.

3) Considering that for any roaming path in a virtual scene, it is easy for users to get lost, many application systems provide a set of fixed camera paths for users to navigate. Hence, we can consider laying out a virtual scene around a given roaming path, reducing the need to consider occluded or otherwise invisible virtual objects, optimizing the scene layout, and organizing the scene data so as to speed up the rendering.

Based on the above analysis, this paper proposes a virtual scene optimization and acceleration rendering method that can adapt to the performance of different mobile devices for indoor scenes such as virtual museums. In order to speed up the rendering, two strategies proposed in this paper are considered:

1) *Optimizing the layout of the virtual scene in the design stage.* The designer chooses the recommended roaming paths, and the visible region of each path can be computed and visualized automatically, i.e., the PVS based on the preset roaming path are pre-computed. This information then guides the designer in devising a reasonable layout. The goal is to ensure that the designed scene is supported on different kinds of devices. Moreover, users should be able to experience scenes with similar content on devices with different performance properties. Although the layout of the scene may be different, the number of virtual objects viewed will almost be the same. In other words, high-end mobile devices will see smooth and fine-grained scenes, while low-end users can also see reasonably complete content smoothly, albeit with a diverging layout and a different degree of fine-grainedness.

2) *Further selecting the rendered contents by calculating the visible region of the user's current viewpoint in the running phase based on the PVS obtained in the design strategy.* The goal is to run virtual scenes in real-time for users.

In the following, we will describe these two phases in detail.

A. Virtual Scene Optimization Layout based on Mobile Device Performance

Before the scene is rendered in real-time, the content that will not be in view can be eliminated, thus reducing the rendering burden. A common method is to preset a possible camera path and pre-compute the contents that might be rendered, i.e., a potentially visible set. Based on such a strategy, we design a virtual scene optimization layout method accounting for the performance of the mobile devices. The steps are as follows:

Step 1: Obtain device profile. The user determines the mobile device model to which the designed scene may be applied. The system will retrieve the corresponding device performance profile from a pre-established database to guide the designer's design.

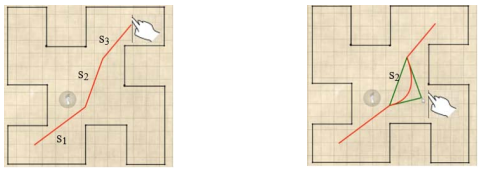
Step 2: Coarse-grained design. The designer defines a recommended roaming path in the scene subject to any pertinent requirements and design goals. At this point, the system interactively displays the visible region along each path, which is the effective area that the designer can deploy. The designer adjusts and deploys the scenes in the effective area.

Step 3: Fine-grained design. The designer arranges virtual objects (such as artifacts) in each visible region. In the process of arranging objects, the system compares the rendering load of the object model in the current scene with the maximum capacity that can be rendered by the mobile device obtained in the first step, and guides the designer towards making a layout consistent with it. In the event that the object rendering load exceeds the performance constraints of the mobile device, the system prompts the designer to make suitable adjustments, such as placing the current model elsewhere, dispersing the layout, or replacing other models.

The details of Steps 2 and 3 are as follows.

1) Coarse-grained Design: The designer first interactively designates the possible roaming path, consisting of a number of segments, which is the path that the user is expected to follow. The designer can set the path via touch operations or by clicking on the action. For instance, in Fig. 1, the path is

defined by the designer, composed of the segments s_1, s_2, and s_3. The designer can modify the segment types via drag-and-drop operations. In Fig. 1(b), the designer modifies the shape of s_2 by means of curve generation until satisfied.



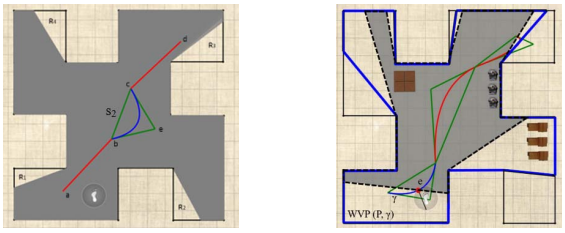
(a) (b)

Fig. 1. The interactive design for designers. (a) Designing the path via simple touch operations, and (b) modifying the shape of a path.

Visibility computation plays an important role in virtual architectural buildings. When walking along the designer-given paths, a weak visibility polygon can be computed in advance to speed up exact visibility queries of moving points along the trajectories. At the same time, this method can also avoid the arrangement of objects in the invisible region and improve the logic and effectiveness of the design. Weak visibility polygons are defined as follows.

Definition *Weak Visibility Polygons (WVP).* For a curve or region γ and a point q in a polygon P, if there exists a point p in γ such that p and q are visible, we say that q is weakly visible from γ, and the set of points inside P that are weakly visible from γ form the weak visibility polygon of γ, denoted as $WVP(P, \gamma)$ [13].

When the path, composed of line segments or curves, has been completed, the WVP of each segment is automatically computed by the method [13]. The WVP of the segment s_2 is shown in Fig. 2(a): The gray polygon is the visible region of s_2, while the non-gray areas (i.e., R_1, R_2, R_3, and R_4) are not visible for s_2. When designers arrange the object on the path s_2, they only need to arrange objects in the gray visible region, i.e., the arrangement in the invisible regions R_1, R_2, R_3 and R_4 can effectively be avoided.



(a) (b)

Fig. 2. The weak visible region. (a) The weak visibility polygon (gray region) of s_2 (blue line), and (b) The visible region of the viewpoint e in $WVP(P, \gamma)$. The blue border polygon represents $WVP(P, \gamma)$ and the gray area marks the visible region of the current viewpoint e based on the orientation of e.

2) Fine-grained design: Based on the performance tolerance of the current device and complexity of the virtual scene, the designer is guided to ensure the usability of the design.

We define a data structure to record the information between the path and virtual objects:

$$< P_{\mathrm{id}}, B(\gamma), WVP(P, \gamma), M_{\mathrm{V}}(\gamma), D >$$

Here, P_{id} represents the given roaming path, $B(\gamma)$ represents the curve segment on P_{id}. $WVP(P, \gamma)$ represents the weak visibility polygon of the curve segment, $M_{\mathrm{V}}(\gamma)$ represents visible models in the visible region of the curve segment $WVP(P, \gamma)$, including the attributes of the model: model number, model capacity, model position. D represents the processing power of the mobile device.

When the designer places an object, we need to update the above data structure, which is used to capture pre-processed input information, specifically the PVS, to improve the rendering speed during the user experience. In particular, when the designer positions an object in the weak visibility polygon of a segment, there are two things for the system to do.

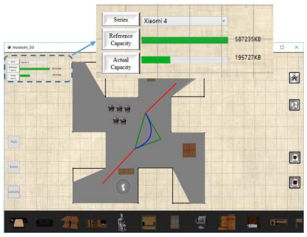


Fig. 3. The interactive interface for the designer's fine-grained design. The enlargened part shows the performance information of a mobile device.

First, it computes the capacity of the objects in the current visible region and determines whether this capacity exceeds the maximum capacity of the mobile device. The interface depicted in Fig. 3 show how the capacity of the objects in the current visible region are presented to the designer. If the capacity does exceed the maximum capacity of the device, the system can interactively prompt the designer to modify the design, such as by adapting the layout, or replacing an object with one that satisfies the capacity constraints.

Second, the system computes the association of the object with each segment of the roaming path. The object may be visible in one or more segments of the path. For this, the system traverses each segment along the path, and determines whether the object belongs to the visible polygon corresponding to each segment. If it belongs to the visible polygon, the capacity of the objects in the visible region will be calculated and the data structure will be updated.

Finally, after the presentation, the designer can further make some adjustments based on the above data structure.

B. Real-time Rendering of Virtual Scenes

Deploying this idea, we found that even for the same model of mobile device, different users exhibit different usage patterns, which moreover vary over time, resulting in different current processing capabilities of the mobile device. For example, if there are too many apps installed on a mobile phone, and there are many background processes, the available compute power will be limited.

Our approach thus takes the current runtime state of the mobile device as a reference, and adjusts the maximum rendering ability of the device in each visible region. When the user is in the immersive experience phase, if the rendering capacity in the current field of view is too large, the virtual content can be optimized by means of LOD. That is, according to the distance relationship between the virtual object and the viewpoint, objects at a farther level are replaced with a model for a different level of detail, until the scene complies with the current maximum rendering capability of the device, such that a real-time rendering of the scene is ensured. Specifically, our approach consists of the following steps.

Step 1: Estimate the actual current processing power of the device, denoted as D_r. If $D_r < D$, update D as $D = D_r$.

Step 2: Determine which segment of the path the user's current viewpoint e belongs to, denoting it as γ. Determine the weak visibility polygon $WVP(P, \gamma)$ (See Fig. 2(b)).

Step 3: Compute the visible region of the current viewpoint e in $WVP(P, \gamma)$ based on the orientation of the viewpoint, denoted as V_e. In Fig. 2(b), the visible region of the current viewpoint e is depicted as a gray polygon.

Step 4: Traverse all models in $M_{\mathrm{V}}(\gamma)$ and select models belonging to V_e.

Step 5: Determine whether the capacity of the objects in V_e exceeds the maximum capacity D of the device. If the rendering load of the objects in V_e exceeds D, we invoke the LOD technique to speed up the scene rendering. This entails that the objects in V_e are sorted with respect to their distance from the current viewpoint e, and the corresponding simplified models of objects are selected by proceeding from farther to nearer ones, in each case, choosing a model of lower granularity for objects that are farther from the viewpoint, until the total rendering load in V_e no longer exceeds D.

IV. EXPERIMENTS AND RESULTS

A. Experimental Design

We have developed a system supporting the design of virtual scene layouts using multi-touch technology to quickly arrange virtual objects on a 2D canvas, while a 3D window displays the corresponding virtual scenes in real time, enabling the designer to flexibly and rapidly construct 3D virtual scenes [12].

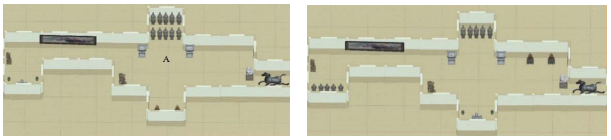
We further implemented the virtual scene created by the designers as an Android app that can operate on a mobile device running Android OS. The phones feature a touch screen, an accelerometer, and Wi-Fi capabilities. Users can interact with the virtual world via Google Cardboard-style 3D glasses and a motion-tracking sensor. The former, composed of a cardboard and a smartphone, can present stereoscopic images. The motion-tracking sensor we select is a Kinect, which can capture the position of users.

In order to verify the effectiveness of our proposed strategies, we designed two experiments to assess our method. In the first experiment, we evaluate the design effectiveness across heterogeneous mobile devices. In the second experiment, we evaluate the effectiveness of our strategy on a single series of mobile phones but across different use states.

B. Experiment 1

The participants are requested to design a virtual museum with 30 artifact models using our design system. The resulting designs are later run on phones with different characteristics, which, in this experiment, are chosen as the Xiaomi 5 series and the Xiaomi 4 series. The maximum rendering load in the same viewpoint of these two mobile phones (number of patches) is $6.9 * 10^7$ and $3.5 * 10^7$ respectively.

Although the performance of the mobile phones may diverge, it is necessary to ensure that the artifacts seen by the user are the same, that is, the number of models to be rendered in the entire scene is the same. The designed virtual museum using 30 cultural relics models is shown in Fig. 4. In region A in Fig. 4(a), the rendering load exceeds the maximum capacity of the Xiaomi 4 in the weak visibility polygon of the segment ab (see Fig. 5), and the designer resorts to redistributing cultural relics to ensure the real-time operation of the scene. Fig. 4(b) is the adjusted scene that can be run on the less powerful Xiaomi 4.



(a) (b)

Fig. 4. Museum layout designed on two mobile phones: (a) the virtual scene on the Xiaomi 5 series, (b) the virtual scene on the Xiaomi 4 series.

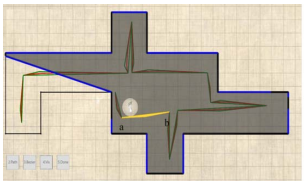


Fig. 5. Roaming path and its weak visibility polygon. Herein, ab is one of the roaming paths chosen by the designer, and the gray region is the weak visibility polygon of ab.

In the design process, we regard the constraint on the mobile device performance as one of the primary design factors, as it can ensure that the user can later attain a frame rate of at least 15 frames per second within the same viewpoint region. This entails that the design adapt to different performance profiles of different mobile devices, thus improving its effectiveness and availability.

In order to more validate our method, we also design various virtual scenes with different spatial structures. For example, we design a virtual maze learning scenes in Fig. 6, using 20 dinosaur models for immersive experiences of users. Similar to the above, the layout of the virtual maze for the Xiaomi 5 series is shown in Fig. 6(a), while the adjusted scene that can be run on the Xiaomi 4 is shown in Fig. 6(b).

The above experimental results show that the proposed design strategy can effectively guide designers in laying out

(a) (b)

Fig. 6. Maze layout designed on two mobile phones: (a) the virtual scene on the Xiaomi 5 series, (b) the virtual scene on the Xiaomi 4 series.

the virtual scenes such that rendering efficiency is considered and an adaptation to different mobile devices is possible.

C. Experiment 2

Whereas in Experiment 1, the designed virtual museum and maze was run across different models of mobile phones, in Experiment 2, the performance of the rendered scene is tested on the same model of a mobile phone but in different states, such as with more startup processes or more apps running in the background. Before each run, we fetch information regarding the current operating status of the mobile phone. Specifically, we rely on the percentage of available RAM of the mobile phone as an indicator. In the designed scene, the position of the light source is fixed; the materials of the virtual objects are simple ones such as a solid wood, and reflection is not considered. When the user roams around the virtual scene, the number of frames rendered by different users at each instant is counted, thus providing an assessment of the effectiveness of the method.

In the experiment, we compared the attained frame rate for the virtual scene across 4 different cases: the designed virtual scene is Museum, and the available percentages of RAM on the mobile phone are 50% and 30% respectively; the designed virtual scene is Maze, and the available percentages of RAM on the mobile phone are 50% and 30% respectively. We consider two human participants, each experiencing each scenario 10 times. We first consider the Xiaomi 5 series for these experiments, and subsequently repeat the experiments with Xiaomi 4 devices.

The experimental results for users roaming in Museum on Xiaomi 5 are given in Fig. 7. Here, the blue lines represent the results when not relying on our strategy, while the green lines consider the case of using our strategy. Each line represents an average over a user experiencing the museum 10 times.

During the process of roaming, the capacity of the virtual objects to be rendered varies depending on the user's vantage point. For instance, the number of rendered frames between points 1 and 25 is very different to the interval from 30 to 40, which suggests that the capacity of the virtual scene that the user sees is different. In addition, the method proposed in this paper compares the load of the objects rendered at the current instant with the mobile phone's current maximum rendering capabilities. If the rendering capacity in the current field of view is too large, our method will adjust the scene rendering content. According to the distance relationship between a virtual object and the viewpoint, the objects at a far distance

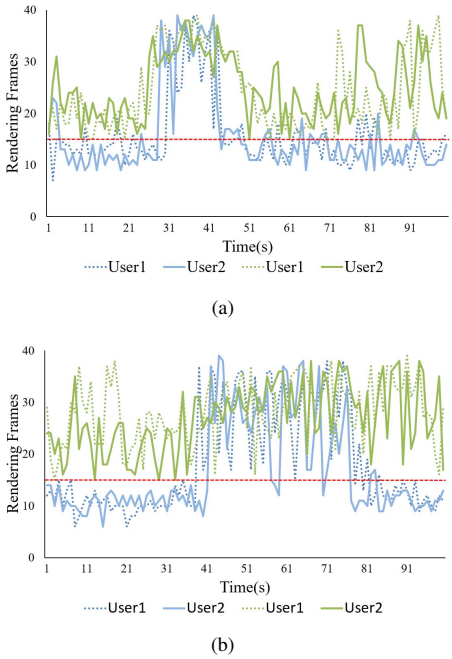


Fig. 7. Comparison between the number of rendered frames obtained using our method (green lines) and not using our method (blue lines) for Museum when the available percentage of RAM is (a) 50% and (b) 30% on Xiaomi 5.

will be replaced with models with different levels of detail until the mobile phone in its current state can cope with the rendering load.

Hence, as can be seen in Fig. 7(a), the number of rendered frames for the same user is different at different time points. Moreover, since the viewing angles of different users at each time point may be different, the number of rendered frames of different users at the same point in time may diverge.

The experimental results show that our method can guarantee that the number of frames rendered per second is at least 15, ensuring the user's real-time experience. When our method is not invoked, the rendering capacity in the current field of view may easily become too large, making it difficult to achieve real-time effects.

In order to further evaluate the effectiveness of this across different phones, we repeated the experiments for the 4 different cases on Xiaomi 4. We also get similar results that our method can achieve real-time rendering (see Fig. 8).

When the same mobile phone series is in different states of usage, we can estimate the current computing power of the phone. When the available percentage of RAM in the phone is 50%, the maximum rendering capacity of the phone is about 50% of the original rendering capacity. When the available percentage RAM is 30%, the maximum rendering capacity tends to be about 40% of the original rendering capacity. This paper takes the current state of the phone as a reference, and adjusts the maximum rendering ability of the phone in each visible region to ensure the real-time operation of the scene by reducing the number of patches.

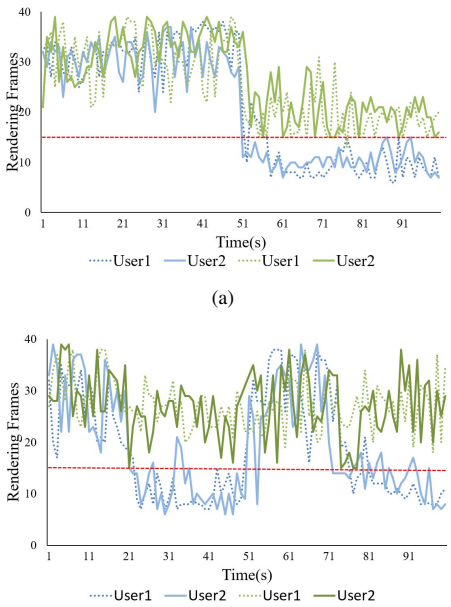


Fig. 8. Comparison between the number of rendered frames obtained using our method (green lines) and not using our method (blue lines) for Maze when the available percentage of RAM is (a) 50% and (b) 30% on Xiaomi 4.

The above results show that for phones with different performance characteristics, the virtual scene can be rendered by our method to ensure the real-time experience of the user. Moreover, even when the same type of phone is used but in different usage states, our method can dynamically adapt the rendering of the virtual scene to ensure it is in real time. Thus, the method proposed in this paper can flexibly adapt to different performance characteristics of phones, robustly enabling real-time rendering of scenes of different scales in accordance with the actual rendering abilities of the device.

V. Conclusion

Mobile device-based virtual reality systems may face many challenges in enabling real-time interaction, due to factors such as the limited computing performance and storage capacity. In this paper, a virtual scene optimization layout strategy is proposed to adapt to different mobile devices for real-time interactive display. In the scene design phase, the performance of the device for the user's immersive experience is regarded as one of the critical factors. We consider how to optimize the virtual scene to adapt to the performance profiles of different mobile devices and propose an accelerated rendering method based on a custom proposed data structure to enable the ability of rendering large-scale scenes in real time on different mobile devices. The experimental results show that this strategy can effectively support real-time rendering of virtual scenes designed by designers for devices with different performance profiles.

As an initial exploration, the current usage scenarios considered in this paper are mainly virtual indoor scenes such as virtual museums and labyrinths. The characteristics of such scenes are that the wall structure can allow for a Sweep operation with a two-dimensional view to obtain a three-dimensional structure.The next step is to explore the expansion to more types of scenes. In addition, our work can also be adapted by invoking visibility algorithms that extend beyond the path into the region so that participants can move more freely. Overall, we believe that this framework has great potential in providing a more robust real-time VR experience to the millions of mobile device users across the globe wishing to explore the potential of low-cost VR mounts.

Acknowledgment

This work is supported by the National Key Research and Development Program of China (2018YFC0831003), Shandong Key Research and Development Program (2017CXGC0606), and the National Natural Science Foundation of China (61802232).

References

[1] Google. (2015) Google cardboard. [Online]. Available: https://www.google.com/get/cardboard/

[2] W. Gai, C. L. Yang, Y. L. Bian, C. Shen, X. X. Meng, L. Wang, J. Liu, M. D. Dong, C. J. Niu, and C. Lin, "Supporting easy physical-to-virtual creation of mobile vr maze games: A new genre," in *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems(CHI'17)*, New York, USA, May 2017, pp. 5016–5028.

[3] F. Biljecki, H. Ledoux, and J. Stoter, "An improved lod specification for 3d building models," *Computers, Environment and Urban Systems*, vol. 59, pp. 25–37, 2016.

[4] K. Kin, T. Miller, B. Bollensdorff, T. DeRose, B. Hartmann, and M. Agrawala, "Eden: a professional multitouch tool for constructing virtual organic environments," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'11)*, New York, USA, May 2011, pp. 1343–1352.

[5] K. Boos, D. Chu, and E. Cuervo, "Flashback: Immersive virtual reality on mobile devices via rendering memoization," in *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services*, New York, USA, 2016, pp. 291–304.

[6] W. J. Lee, Y. Shin, J. Lee, S. Lee, S. Ryu, and J. Kim, "Real-time ray tracing on future mobile computing platform," in *SIGGRAPH Asia 2013 Symposium on Mobile Graphics and Interactive Applications*, New York, USA, Nov. 2013, p. 56.

[7] OpenGL. (2018) The standard for embedded accelerated 3d graphics. [Online]. Available: https://www.khronos.org/opengles/

[8] B. S. He, X. L. Xu, and T. Zheng, "Vector graphics rendering on mobile device," in *Conference on Communications and Mobile Computing*, New Jersey, USA, Mar. 2009, pp. 8–11.

[9] S. Shi, "Building low-latency remote rendering systems for interactive 3d graphics rendering on mobile devices," in *Proceedings of the 19th ACM international conference on Multimedia*, New York, USA, Nov. 2011, pp. 859–860.

[10] C. Crassin, D. Luebke, M. Mara, M. Mcguire, B. Oster, P. Shirley, P. P. Sloan, and C. Wyman, "Cloudlight: A system for amortizing indirect lighting in real-time rendering," *Ppsloan Org*, 2013.

[11] L. Lu, C. L. Yang, W. Z. Wang, and J. Q. Zhang, "Voronoi-based potentially visible set and visibility query algorithms," in *2011 Eighth International Symposium on Voronoi Diagrams in Science and Engineering*, New Jersey, USA, 2011, pp. 234–240.

[12] X. W. Sun, Y. F. Wang, G. de Melo, W. Gai, Y. L. Shi, L. Zhao, Y. L. Bian, J. Liu, C. L. Yang, and X. X. Meng, "Enabling participatory design of 3d virtual scenes on mobile devices," in *Proceedings of the 26th International Conference on World Wide Web Companion (WWW'17)*, New Jersey, USA, 2017, pp. 473–482.

[13] C. L. Yang, W. Z. Wang, Y. J. Yang, L. Lu, Z. J. Zhu, B. H. Zhu, and W. Zeng, "Weak visibility polygons of nurbs curves inside simple polygons," *Journal of Computational and Applied Mathematics*, vol. 256, no. 1, pp. 1–15, 2014.