Web3D-based Online Walkthrough of Large-scale Underground Scenes

Xiaojun Liu, Ning Xie, Jinyuan Jia*
Department of Software engineering, School of Software Engineering
Tongji University
Shanghai 201804, China
xjliu@126.com, ningxie@tongji.edu.cn, yjia@tongji.edu.cn

Abstract—**Large-scale scenes' processing has become the major trend today. We mainly address the online walkthrough of Large-scale underground (UG) scenes in this paper. Taking into account the characteristics of UG scene, we first propose a lightweight preprocessing to optimize the raw UG scene and unify the raw data with scene, sub-scene and simple model. Then we generate a three-layered grid structure for organizing the scene to facilitate the visibility culling and data accessing. Finally, we design two scene management strategies, named SOI-ExteriorShell and Portal-InteriorShell, and integrate our methods in an experimental prototype. The experimental result shows that our method can remove a large amount of redundancies from the raw data, reduce resource consumption greatly and make it possible to walkthrough in large-scale UG scenes online without any web browsers plugins.**

Keywords- lightweight; large-scane; underground scene; Web3D

I. INTRODUCTION

Nowadays, web page has been regarded as the most popular way for information sharing. The number of Web3D based application is growing rapidly and it is also urgently required to display large-scale 3D scenes over the internet, especially with web pages. However, the web browsers, not like standalone software, cannot afford the resource consumption for rendering and processing large-scale 3D scenes in real-time. Moreover, though the bandwidth of the internet connection is increasing rapidly, it is still a great challenge to transfer such a tremendous amount of data. In this paper, we present an online walkthrough solution for large-scale Underground (UG) scenes.

UG scenes, including UG buildings with complex indoor and outdoor structures, are mainly the man-made constructions located below the ground surface, such as UG station, UG supermarket etc. In UG scenes, there are simple objects like UG pipelines and complex buildings which consist of a great many polygon data that demonstrate the buildings in three kinds of shells: interior shell, exterior shell and connecting shell. Interior shells depict the room boundaries respectively inside the exterior shell which stands for the boundary surface of an entire building. The connecting shell is the bounded connecting parts between shells or buildings. Generally, these three kinds of shells are primary structured with ceiling, floor and wall by sets of polygons. The relations between two shells, if exists, are also depicted by shared polygons. Therefore, the raw UG scenes have some defective characteristics:

(1) **Redundant**. Many duplicated edge, vertex and geometrically reused model always exist in the raw scenes.

(2) **Unstructured.** Logically the scene primary contains three kinds of shells, but actually are just polygons. Shells cannot be extracted explicitly from unstructured polygons.

(3) **Inefficient**. It is very inefficient to use raw UG model in scene management, e.g. just locating a shell needs to search thousands of polygons.

To walkthrough the UG scenes, we must optimize them in advance. In this optimization, not only data structure but also scene management should be considered.

II. RELATED WORK

There are a great many studies related to 3D scenes rendering. Scene graph has been widely used in e.g. OpenSG, Blue-c etc [1-2]. To improve their performance, Song et al. [3] extended OpenSG for processing continuous simulation data. Open Scene Graph (OSG) also extended them by integrating plugins for file operation [4]. Li et al. [5] optimized the current grid organization algorithm with a new structure named Optimized Recursive Grids (ORG). Stein et al. [6] discussed the spatial data structures in 3D web environment and showed the importance of combination of data structure and visibility for web3D.

Meanwhile, some researches focus on reducing the data amount of scenes. Cohen-Or, Kasik etc. mainly related this work on visibility analysis [7-10]. They improved rendering by scene culling, graphics hardware accelerating, area of interest, etc. Fisher and Wen et al. addressed the similarities between models [11-12]. Their methods can be used to identify the recurred object so as to remove redundancies.

Many other related studies are also conducted. Zheng et al. [13], Deng et al. [14] and Limper et al. [15] investigated their work mainly about network direction, from ray tracing, demand loading strategy, a streamable format for 3D data transmission etc. [16-19] mainly focused on data caching, distributed scene oriented service development and efficient data exchange architecture.

In spite of the considerable related researches, many of them lack in considering the characteristics of UG scenes or remote visualization over the internet.

III. OVERVIEW

In this work, on considering the characteristic of the raw UG scene data, we primarily introduce our approach in 3 parts (the first 3 steps as shown in Fig. 1).

a) Lightweighted preprocessing. To remove the redundancies and reconstruct the relations of shells, we unify the raw data with a shell-based format. (Section IV)

b) Scene graph generation. With the shell-based data, we generate a three-layered grid structure to organize the whole scene. (Section V)

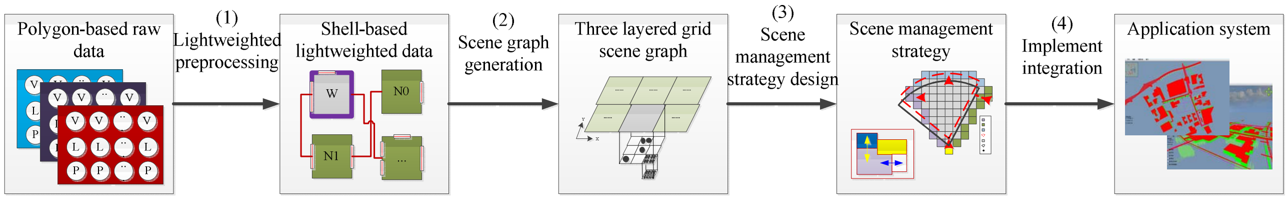

Polygon-based raw data
(1) Lightweighted preprocessing
Shell-based lightweighted data
(2) Scene graph generation
Three layered grid scene graph
(3) Scene management strategy design
Scene management strategy
(4) Implement integration
Application system

Figure 1. Pipeline of our paper

c) Scene management strategy design. We design two scene management strategies to respectively correspond to outdoor and indoor scenes. (Section VI)

IV. LIGHTWEIGHTED PREPROCESSING

There are two steps in this section: shell extraction and redundancies removal.

A. *Shell extraction*

Because of their similar internal structure, we consider every UG building as a sub-scene, which contains many shells. Shell extraction is a way to reconstruct the input polygons into shells and their topological relations. These shells are therefore deemed to be child models of UG buildings. Fig. 2 shows a building and its extracted shells with 1 exterior shell (W1), 7 interior shells (N1 to N7), and 10 portals (C1 to C10).

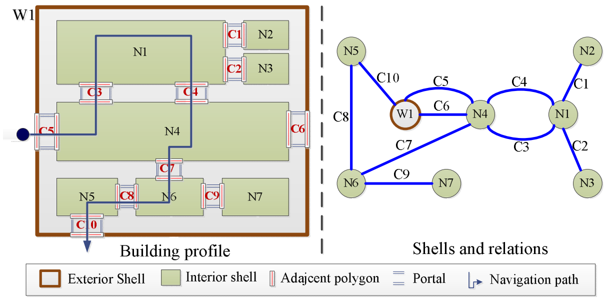


Figure 2. A shell extraction example

For UG pipelines, due to their linear spatial shape and the fact that they are always related to road sections, we hereby handle them as the child of corresponding road section and regard the road section as sub-scenes.

B. *Redundancies removal*

There are many redundant data which needs to be removed, e.g. some reused standard models in UG scenes, as illustrated in Fig. 3 and duplicated edge/polygons.

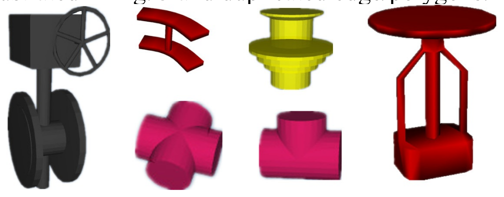


Figure 3. Standard model examples in UG scene

In this section, we mainly focus on these two kinds of redundancies: reused model and extruded vertex. For reused model, such as two shells or two standard models in Fig. 3, we directly use the voxel shape description (VSD) [12] to detect the repeated models. For extruded vertex, a simplification of geometric operations is employed.

With the preprocessing, the two kinds of sub-scenes: building and road sections, are uniquely identified by their IDs. Their child models are identified by names and connected with relations. Though the redundant data has been removed, the structure is preserved.

V. SCENE STRUCTURE GENERATION

In large-scale 3D scenes, the amount of data that needs to be processed is greatly depended on viewpoint position, sight range and other factors. Considering their different characteristics, we classify our scene into indoor and outdoor scenes. While roaming, we can switch between these two scenes upon the current viewpoint position.

A. *Three-layered Grid structure for Outdoor Scene*

We organized outdoor 3D scenes with a three-layered 2D uniform grid structure, because the space of third dimension Z is so small comparing with the other two dimensions. Many researches in grid generation have existed, and we employ and improve the method [5] in our method because of its efficiency.

(1) We decrease the involved polygons from all to the bottom ones for reducing the computational cost;

(2) We only record the description file (DF) names or the model IDs to reduce the amount of loaded DF;

(3) Our method is extended to support appending, deletion and updating models on the grid layer.

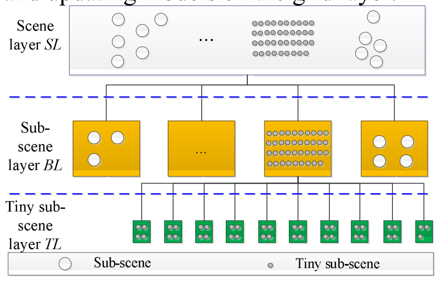


Figure 4. Three-layered grid structure for outdoor scene

In the three-layered 2D uniform grid (Fig. 4), three layers are designed for different situation. The first layer *SL* is to control the amount of information currently loaded, so as to avoid the loaded information far beyond sight range. On *SL*, the names of DF from *BL* are recorded by *SL*'s unit cell. *BL* is the layer on which the Sector of interest (SOI) is implemented, and it is important for determining the range of scene scheduling and collision detection. The third layer *TL* is a supplement layer to deal with the extreme case in which too many (tiny) sub-scenes are in the same *BL*'s cell and will lead to a sharp increase of the DF size. By *TL*, we can process this independently.

The pseudo-code of our method for sub-scene deletion on the existed scene structure is as follows:

```
Algorithm: Sub-scene deletion from the existed scene structure
Input: Sub-scene ID, Scene structure DF
Output: Scene structure DF after deletion process
1.  begin
2.    flag = FALSE;
3.    for each DF on SL do
4.      flag ←Locate the DF information on BL;
5.      if flag is TRUE then
6.        goto line 10;
7.    end for
8.    if flag == FALSE then
9.      goto line 23;
10.   Locate the BL's cell of the sub-scene to list LC;
11.   for each element of LC do
12.     Search the sub-scene ID;
13.     if searched then
14.       Delete the ID number;
15.       goto line 23;
16.     if searched the DF name on TL then
17.       goto line 21;
18.   end for
19.   if searched nothing then
20.     goto line 23;
21.   if searched DF on TL then;
22.     execute as line 10 to line 18;
23. end
```

B. Shell-based Structure for Indoor Scene

The strategy for indoor scene is Portal-InteriorShell detailed in VI-B. The goal is just to process the shells related with current shell. In this way, all the invisible shells can be culled away and the shell-based structure can be directly used to generate corresponding scene structure.

VI. Scene Management Strategy Design

To support the smooth walkthrough, we introduce two online management strategies in our methods.

A. SOI-ExteriorShell Outdoor Scene Strategy

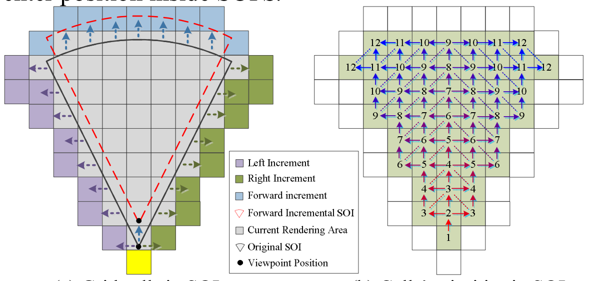
When in large-scale outdoor scenes, because of the limit sight range, we are just interested in the visible area around our viewpoint. An appropriate strategy for managing interest area could filter out the invisible scene models [7-9] efficiently. Based on the scene structure, we build a sector region as our interest area (SOI, Fig. 5 (a)) which can be calculated based on the Field Of View (FOV) to minimize the resource. In SOI, all the IDs of sub-scenes are firstly loaded from DF and then the corresponding sub-scene data followed. The cell of SOI is coded as following:

```
struct Grid_Cell {
        long              x, y;      //cell coordinate
        vector<string>    vID;       //IDs included
}
```

In the real world, the nearest object should be seen first. In our method, we resolve it by evaluation a priority by their position. In a simple manner, the cell's priority *w* in our method is formulated as (1):

$$w = | s.x - p.x | + | s.y - p.y |, \qquad (1)$$

where *p* is the position of the viewpoint, and *s* is the cell's center position inside SOI *S*.



(a) Grid cells in SOI (b) Cells' priorities in SOI

Figure 5. Sector of interest

While navigating, the SOI is updated while the viewpoint translating (Fig. 5). Firstly, the incremental cells in the near future are automatically forecasted, and pre-loaded according to their priority value from (1). Secondly, the cells no longer in SOI will be released from memory to guarantee the constant memory footprint.

B. Portal-InteriorShell Indoor Scene Strategy

With the occlusion of roofs, walls, and floors, it is very different to navigate in indoor scenes with in outdoor ones. Because the visible spaces are always connected by "openings" or "portals"[7], we adopt a Portal-InteriorShell indoor scene strategy.

In this strategy, the data scheduling and rendering is driven by portals. For an actual simulation, we also introduce the distances between viewpoint and portals of current shell to determine the appropriate time to load the necessary data. The distances are detected in real-time and can be presented as a brief formula (2):

$$D = v * (t + \Delta t), \qquad (2)$$

where *v* is the velocity of navigating. *t* is the elapsed time within data loading, and the Δt depicts the time delay.

Fig. 6 shows the progressive data load corresponding with the path in Fig.2. We can see that the amount of processing data is just the shells connected with the current shell directly.

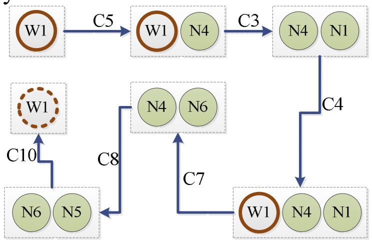
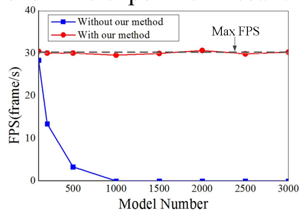


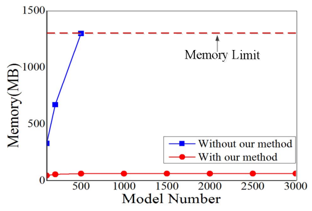
Figure 6. Progressive load along the path in Fig. 2

VII. Experiment Result and Discussion

To analyze our method, we implement a prototype in which we respectively select 100, 200, 500, 1000, 1500, 2000, 2500, and 3000 models. And the sizes of the models range from 50K to 7.1M. The hardware configuration is: Intel i5-3330 CPU, 2.0 GB RAM, NVIDIA GeForce 610M and Windows 8 (64 bit). Initial settings are listed as follow: 8 cells for SOI radius, 6 models per cell, 90 degrees for FOV, and 30 for max FPS. Fig. 7 and Fig. 8 show the experiment result.



(a) FPS comparison



(b) Memory comparison

Figure 7. Performance comparison

From Fig. 7, we can see that the FPS maintains around the max refresh rate with our method but drops sharply without our method. Especially when the number of models increases to 500, the FPS drops close to 0. And the similar improvements can also be found in memory footprint. We can conclude that the scene models have been sufficiently culled and the memory consumption can remain stable even though the complexity of scene grows.

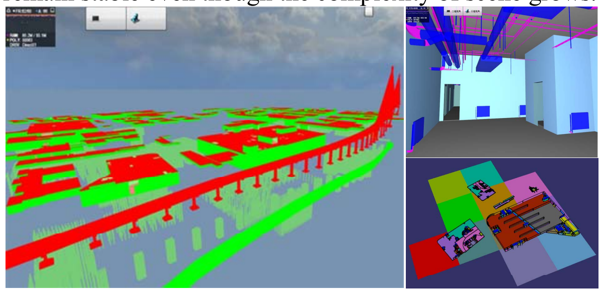


Figure 8. Snapshots from our experiment

VIII. CONCLUSION

Scene graph and its management play an important role in walkthrough of large-scale scenes. Based on the characteristics of UG scenes, we proposed a lightweight scene processing method which is composed of 3 main procedures: raw data lightweighting, scene structure reconstruction and scene management integration. Some other simplifications and improvements are made for improving efficiency. The experiment showed that our method can achieve both low resource consumption and high efficiency and can be easily extended to other scenarios similar to UG buildings. Finally, there are also some shortcomings that need further efforts, e.g. the method used for simplifying extruded shell just works in limit cases. To have a better lightweighting, we will have to analyze the shape and distribution of models further.

ACKNOWLEDGMENT

We thank the anonymous reviewers for their valuable comments firstly. Meanwhile, this work was also supported by Projects of National Natural Science Foundation (No.61272276), National 12th five years Plan for Science & Technology Support (No.2012BAC11B00-04-03), The Research Fund for the Doctoral Program of Higher Education (No.20130072110035), Changbai Valley Talent Plan of Changchun National Hi-Tech Industrial Development Zone (No.3-2013006), and Key project in scientific and technological of Jilin province (No.20140204088GX).

REFERENCES

[1] D. Reiners, "OpenSG: A scene graph system for flexible and efficient realtime rendering for virtual and augmented reality applications", PhD theses, Darmstadt, TU, Germany, 2002.

[2] M. Naef, E. Lamboray, O. Staadt, and M. Gross, "The blue-c distributed scene graph", Proc. of the workshop on Virtual environments, Zurich, Switzerland, 2003, pp. 125-133.

[3] I. Song, and J. Yang, "A scene graph based visualization method for representing continuous simulation data", Comput. in Industry, 2011, 62(3): 301-310.

[4] D. Burns, and R. Osfield, "Open scene graph a: Introduction, b: Examples and applications", Proc. of IEEE Virtual Reality, Washington, DC, 2004, pp. 265-265.

[5] J. Li, and W. Wang, "Optimizing Grid Construction in Linear Complexity", J. of Software, 2011, 22 (10): 2488-2496 (in Chinese).

[6] C. Stein, M. Limper, and A. Kuijper. "Spatial data structures for accelerated 3D visibility computation to enable large model visualization on the web", Proc. of the 19th International ACM Con. on 3D Web Technol. 2014.

[7] D. Cohen-Or, Y. L. Chrysanthou, C. T. Silva, and F. Durand, "A Survey of Visibility for Walkthrough Applications", IEEE Trans. Vis. Comput. Graph., 2003, 9(3):412-431.

[8] D. Kasik. "Visibility-guided rendering to accelerate 3D graphics hardware performance", *ACM SIGGRAPH 2007 courses*. 2007.

[9] J. Jia, W. Wang, M. Wang, C. Fan, C. Zhang, and Y. Yu, "Multi-layered Incremental & Scalable Sector of Interest(MISSOI) Based Efficient Progressive Transmission of Large-scale DVE Scenes", Chin. J. of Comput., 2014, 37(6):1324-1334 (in Chinese).

[10] C. Stein, et al. "hare3d: Rendering Large Models in the Browser" , *WebGL Insights* (2015): 317.

[11] M. Fisher, and P. Hanrahan, "Context-based search for 3D models", ACM Trans. on Graph., 2010, 29(6):182.

[12] L. Wen, N. Xie, J. Jia, "Fast accessing Web3D contents using lightweight progressive meshes", Computer Animation and Virtual Worlds In press, 2015.

[13] Z. Zheng, E. Prakash, and T. K. Y. Chan, "Interactive View-Dependent Rendering over Networks", IEEE Trans. Vis. Comput. Graph., 2008, 14(3):576-589.

[14] Y. Deng, and R. W. H. Lau, "On Delay Adjustment for Dynamic Load Balancing in Distributed Virtual Environments", IEEE Trans. Vis. Comput. Graph., 2012, 18(4): 529-537.

[15] M. Limper, et al. "SRC-a streamable format for generalized web-based 3D data transmission", Proc. of the 19th International ACM Con. on 3D Web Technol. 2014.

[16] M. Limper, et al. "The POP buffer: Rapid progressive clustering by geometry quantization", Computer Graphics Forum. Vol. 32. No. 7. 2013.

[17] A. Schiefer, et al. "Service-oriented scene graph manipulation", Proc of the 15th International ACM Con. on 3D Web Technol. 2010.

[18] Y. Jung, et al. "Declarative 3D Approaches for Distributed Web-based Scientific Visualization Services", Dec3D. 2012.

[19] T. Franke, et al. "VCoRE: a web resource oriented architecture for efficient data exchange", Proc. of the 18th International ACM Con. on 3D Web Technol. 2013.