Rendering Optimization of Building Information Models Based on Occlusion Culling

Taiping Jiang [1,a], Shixiang Huang [1,b*], Zhilian Yan [2,c]

[1]School of Computer Science and Technology, Anhui University of Technology, Ma'anshan, China
[2]School of Electrical and Information Engineering, Anhui University of Technology, Ma'anshan, China

[a]jtp2008@ahut.edu.cn, [b]3014973038@qq.com, [c]zlyan@ahut.edu.cn

Abstract. The application of Building Information Modeling (BIM) technology in the field of construction engineering has become increasingly mature, and its practical application on construction sites is closely related to lightweight models. However, BIM involves high complexity and contains large amounts of geometric data, leading to inefficient rendering on the web. To address this issue, this paper proposes an occlusion culling algorithm based on scene management. The algorithm employs an octree spatial data structure to manage the model, then creates bounding boxes according to the constructed octree. A ray-casting algorithm is used to analyze all the bounding boxes, ultimately identifying and culling the occluded regions. Experimental results show that the algorithm effectively reduces the geometric data to be rendered without affecting the model's accuracy, thereby improving the scene rendering speed and demonstrating the algorithm's effectiveness.

Keywords: Building Information Model (BIM); Occlusion Culling; Spatial Mesh Division; Model Rendering

1 Introduction

Building Information Modeling (BIM) is a digital representation of a physical building, used to carry the data assets throughout the entire lifecycle of a building product. It serves as the foundation for the digital transformation of the construction industry and promotes high-quality development of the industry [1]. In recent years, under the government's advocacy, BIM technology has been introduced and actively applied in China's construction industry. However, in practical engineering, BIM models are large in size and difficult to transfer or download, leading to challenges such as inefficiency in usage and communication delays, which can cause project schedule setbacks. When using BIM technology, engineering projects need to store a large amount of design drawings and models. Participants must rely on powerful computer hardware and software to view the models, which has caused various issues and lim-

S. M. Chung et al. (eds.), *Proceedings of 2024 4th International Conference on Public Management and Big Data Analysis (PMBDA 2024)*, Advances in Economics, Business and Management Research 318,
https://doi.org/10.2991/978-94-6463-656-7_35

ited the full potential of BIM technology. Therefore, achieving BIM lightweighting has become a key step in applying BIM technology. Currently, the primary method for achieving BIM lightweighting is to load models onto the web[2-11]. However, with the rapid development of the construction industry, the data volume and complexity of building models are growing exponentially. The large amount of data leads to reduced rendering efficiency on the web, affecting the smoothness of model display.

Both domestic and international research departments and scholars have conducted studies on lightweight BIM model display and scene rendering optimization on the web. Xu Zhao et al.[12] proposed a model visualization method based on WebGL and IFC, which allows the interaction of BIM model geometric information and OBJ file information, achieving BIM model display on the web and providing a solution for lightweight modeling. However, model data optimization and scene rendering effects still need improvement. Li Shaoqing, Huo Liang, et al.[13] focused on the triangular mesh of the model and proposed an edge folding algorithm for model simplification. This algorithm simplifies the triangular mesh and reduces the complexity of the mesh, but it also affects the appearance of the model. Wang Qi et al.[14] proposed an adaptive rendering control scheme, using an LOD-AD optimization algorithm to cull distant and smaller triangular meshes. They also prioritized rendering higher-level components when the frame rate is low, achieving adaptive control during the rendering process. This method improved the rendering smoothness while maintaining model accuracy but had some limitations in optimizing models outside the view frustum.

To address these issues, this paper proposes an occlusion culling algorithm based on scene management. The algorithm uses an octree spatial partitioning structure to manage the model, constructs bounding boxes based on the octree, and then employs a ray-casting algorithm to analyze all bounding boxes. The algorithm ultimately identifies and culls occluded regions, thus improving rendering smoothness without affecting the model's appearance. This achieves lightweight display and rendering optimization of Building Information Models on the web.

2 BIM Lightweighting

2.1 Model Data Conversion

Building Information Models (BIM) typically contain large amounts of architectural elements and their associated property information, such as walls, floors, windows, and pipelines. The diversity and complexity of this information significantly increase the complexity of the model. Additionally, BIM models are often stored in Revit format, which imposes certain limitations on viewing and usage. Therefore, it is necessary to convert the model format.

To convert a Revit-based BIM model to the GLTF format, the model usually needs to be exported to an intermediate format first. Initially, the model can be exported from Revit using its built-in plugin, either to Navisworks (NWC) or IFC (Industry Foundation Classes) format, both of which are widely supported. Next, 3D modeling tools such as Blender can be used to import the IFC or NWC files via the appropriate plugin. Finally, the "GLTF 2.0" export function in Blender can be used to convert the

model into the GLTF format. GLTF is supported by many 3D engines and platforms, such as WebGL, Three.js, Babylon.js, and Unity, offering excellent cross-platform compatibility. It has become one of the standard formats for Web 3D content.

2.2 Model Web Reconstruction and Rendering

Three.js is an open-source JavaScript library used for creating and displaying 3D graphics on the web. Based on WebGL, it allows developers to easily build complex 3D scenes and has been widely used by developers worldwide in recent years. Therefore, in this paper, the Three.js engine is used to reconstruct and render Building Information Models on the web. The rendering process is shown in Figure 1.

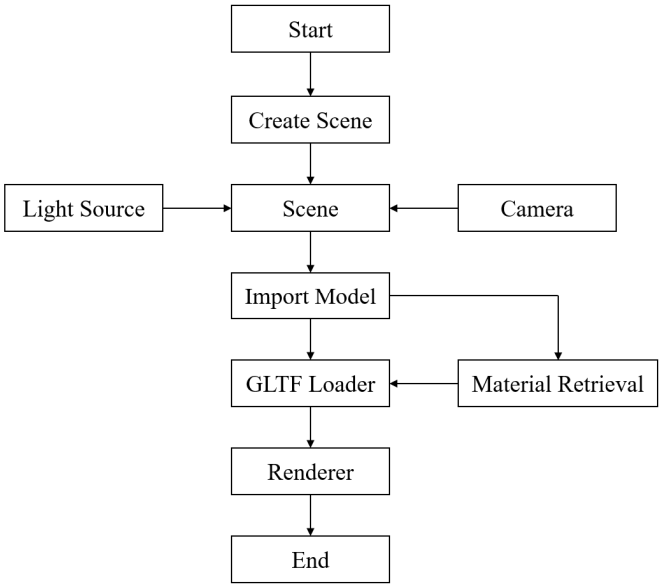

Fig. 1. Three.js Rendering Process

First, the basic elements of a 3D scene are set up using the Three.js framework, including the scene, camera, and renderer. Then, the Building Information Model (BIM) is imported into the scene, and the built-in gltfLoader and dracoLoader in Three.js are used to parse the data within the model. Finally, the model reconstruction is completed in the browser using appropriate methods. To improve the viewing experience, geometric transformations of the model can be implemented using OrbitControls.

2.2.1 Geometric Transformation.

Translation. In a translation operation, matrix transformation is required. Let there be a point A in the scene with coordinates (X, Y, Z, 1). After translation, the point becomes A_1 with coordinates (X_1, Y_1, Z_1, 1). During this process, point A is moved by

the translation matrix T, with the translation components along the three axes being X_t, Y_t, and Z_t. Therefore, the relationship between point A and the translated point A_1 is as follows:

$$\begin{bmatrix} X_1 & Y_1 & Z_1 & 1 \end{bmatrix} = \begin{bmatrix} X & Y & Z & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ X_t & Y_t & Z_t & 1 \end{bmatrix} \tag{1}$$

The matrix T is as follows:

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ X_t & Y_t & Z_t & 1 \end{bmatrix} \tag{2}$$

Rotation. In a rotation operation, there are three key elements: the axis of rotation, the direction of rotation, and the angle of rotation. When rotating counterclockwise by an angle of θ° around the Z-axis, the relationship between the point B(X, Y, Z) before rotation and the point $B1(X_1, Y_1, Z_1)$ after rotation is as follows:

$$\left.\begin{aligned} X_1 &= X\cos\theta + Y\sin\theta \\ Y_1 &= Y\cos\theta + X\sin\theta \\ Z_1 &= Z \end{aligned}\right\} \tag{3}$$

Scaling. Let Z_x, Z_y, and Z_z be the scaling factors along the three coordinate axes, and let the scaling matrix be S. The relationship between point C and the scaled point C_1 is as follows:

$$\begin{bmatrix} X_1 \\ Y_1 \\ Z_1 \\ 1 \end{bmatrix} = \begin{bmatrix} Z_x & 0 & 0 & 0 \\ 0 & Z_y & 0 & 0 \\ 0 & 0 & Z_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{4}$$

The matrix S is as follows:

$$S = \begin{bmatrix} Z_x & 0 & 0 & 0 \\ 0 & Z_y & 0 & 0 \\ 0 & 0 & Z_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5}$$

3 Occlusion Culling

3.1 Related Research on Occlusion Culling

Determining the visibility of polygons relative to dynamic and continuously changing viewpoints is a complex challenge. In large-scale scenes, there are typically a large number of geometric models, and precise visibility calculations require substantial time, making real-time processing difficult. Therefore, visibility problems in scene models are often addressed by using conservative or approximate visibility determination based on occlusion regions formed by other objects[15-17]. Many scholars have conducted extensive research in the fields of visibility and occlusion culling. Research on occlusion primarily falls into two categories: static (preprocessing) occlusion culling algorithms and dynamic (real-time) occlusion culling algorithms.

Static occlusion culling algorithms are techniques that preprocess the scene before rendering. These algorithms first partition the entire scene spatially and calculate potential visibility for each sub-region. During real-time rendering, the algorithm can quickly identify the visible parts from the current viewpoint. This method is especially suitable for static scenes, as it can significantly reduce drawing time during rendering. However, its limitations include being applicable only to static scenes, requiring a large amount of storage space to store visibility information, and having a relatively high time overhead during preprocessing.

Dynamic occlusion culling algorithms aim to address the challenges posed by the dynamic changes of objects in a scene. These algorithms compute visibility information in real-time for each frame, dynamically determining which objects are visible and culling the occluded parts. This method is highly flexible and can adapt to complex and changing scenes. However, dynamic occlusion culling algorithms typically require more computational resources to maintain good performance during real-time rendering.

Therefore, selecting an appropriate occlusion culling technique requires a comprehensive consideration of both the scene characteristics and performance requirements. The occlusion culling method proposed in this paper consists of two main parts: scene preprocessing and CPU-based occlusion information calculation. First, the scene is optimized through simple preprocessing steps, such as using spatial partitioning techniques to decompose complex geometric structures into smaller regions, thereby reducing data complexity and improving the efficiency of subsequent computations. Then, the CPU is used to calculate the occlusion information for the preprocessed scene data, determining the visibility of each region or object using techniques such as ray casting.

3.2 Scene Partitioning

In occlusion culling algorithms, calculating whether an object is occluded is a necessary and time-consuming process. To improve computational efficiency, spatial partitioning data structures are employed to organize the scene. Uniform grid partitioning is a method that divides a spatial region into regular grids. This method is simple to understand and easy to implement; however, when dealing with complex geographic or architectural distributions, it may fail to reflect the actual situation effectively, leading to the generation of redundant data. Standard binary trees or quadtree structures are suitable for handling types of scenes like maps or images, but when faced with the complex topological and geometric structures of 3D city buildings, they may divide models at incorrect nodes along the edges. Therefore, this paper adopts the octree structure, which is more appropriate for scene management and dynamic culling.

An Octree is a tree-based data structure used to represent three-dimensional space, primarily for spatial partitioning and nearest neighbor searching. The partitioning process is as follows: Initially, all objects are contained within an axis-aligned bounding box. The bounding box is then recursively divided into eight sub-bounding boxes, each storing objects that lie within or intersect the boundaries of the sub-box. This process continues until the maximum recursion depth is reached or the bounding box no longer contains any objects. Eventually, each leaf node of the octree stores a certain number of objects. The octree's partitioning method is adaptive and can adjust based on the distribution of objects in the environment. Furthermore, octree traversal is more efficient because many adjacent, evenly distributed grids are merged into a single octree node. Since octree nodes are axis-aligned bounding boxes, intersection tests with objects are relatively fast. The octree generation process is shown in Figure 2.

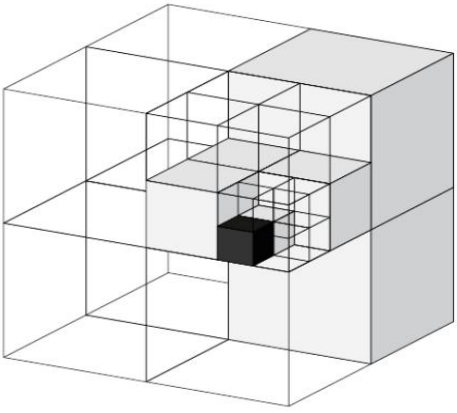

Fig. 2. Octree Generation Process

Before partitioning the scene of a Building Information Model (BIM), it is necessary to construct the AABB (Axis-Aligned Bounding Box) for the spatial region of

the scene. The AABB is a minimal six-sided box that encloses objects in three-dimensional space, with all its edges aligned parallel to the coordinate axes. The advantage of the AABB is its simplicity in computation—only the minimum and maximum values of the object along each axis need to be found to quickly construct the bounding box. After obtaining the AABB for the entire BIM, the octree partitioning can proceed. The specific process is as follows:

1. Divide the AABB bounding box into eight subtrees, aligning them along the xyz axes of the box's center.
2. Check the tree depth of the octree and the number of triangles in each subtree. If the maximum tree depth has not been reached and the number of triangles exceeds a specified threshold, return to step 1.
3. When further subdivision is not possible, store the triangles within the subtree.

Based on the constructed octree, a simple mesh for the bounding model is created. The scene model units within the subtree are then bound to the mesh, and unbound model units are filtered out to prevent empty meshes from affecting occlusion culling. The number of mesh units created through the octree partitioning is much smaller than the number of scene units, and these mesh units can be used to assist in ray detection. If ray detection is performed directly on the scene units of the model, the large number of scene units would require many rays, placing a heavy computational load on the CPU. Therefore, using mesh units generated by the octree to assist in ray detection can effectively reduce the number of rays and lessen the computational burden on the CPU.

3.3 Visibility Analysis

Visibility computation is a critical part of the occlusion culling algorithm. In this paper, the visibility determination is performed using a ray-casting algorithm. The process is as follows:

1. Traverse all the mesh units generated from the octree partitioning of the scene.
2. Use the camera's position as the starting point of the ray, and cast rays towards the positions of each mesh unit as the endpoint for the ray-casting calculation.
3. Based on the ray-casting results, sequentially determine the visibility of the selected mesh units. If a unit is visible, the scene model unit bound to this mesh is marked as visible; otherwise, it is culled.

By using ray-casting technology to calculate the visibility of mesh units in real-time, the efficiency of occlusion culling can be significantly improved. When the camera observes the scene in the view space, only the visible objects in the current scene units are rendered, while the invisible objects are ignored, thus improving the real-time rendering frame rate.

4 Experiment and Result Analysis

To verify the feasibility of the proposed method, the scene management-based occlusion culling algorithm was implemented on a standard computer. The experiment was conducted on the Edge browser, with the machine configured as a laptop with a 13th Gen Intel(R) Core(TM) i7-13700H 2.40 GHz processor, 16 GB of memory, and a 64-bit Windows 11 operating system.

A library model was selected for the occlusion culling experiment. The model data is shown in Table 1.

Table 1. Model Data

Model name	mesh count	triangle face count
library	45608	2662236

To analyze the performance improvement brought by the occlusion culling algorithm, the camera was set to move around the scene from multiple angles. The number of model meshes rendered and the frame rate were statistically compared between the cases with and without occlusion culling, as shown in Figures 3 and 4.

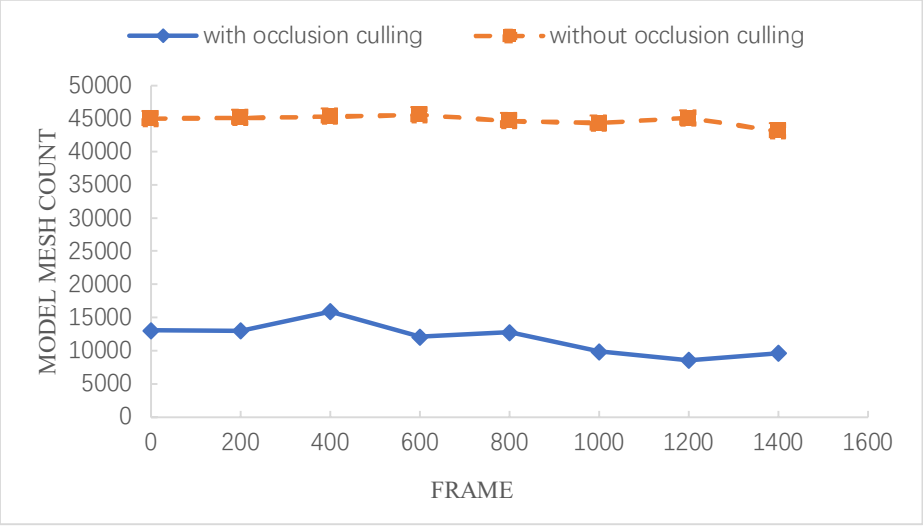

Fig. 3. Comparison of Model Mesh Count for Rendering

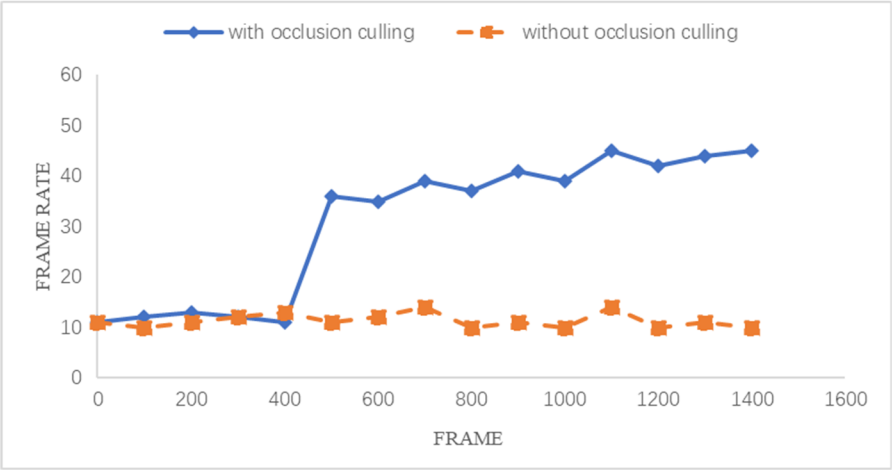

Fig. 4. Frame Rate Comparison Chart

From the above experimental results, we can conclude that after applying the occlusion culling algorithm, the number of rendered model meshes is significantly reduced. The average culling rate is approximately 67%, and the rendering frame rate has increased from the original stable 10 frames per second to around 40 frames per second, demonstrating a clear improvement in rendering efficiency.

5 Conclusion

To address the issue of low rendering efficiency on the web due to the complexity and large geometric data contained in Building Information Models (BIM), this paper proposes an occlusion culling algorithm based on scene management. The algorithm divides the scene into an octree and establishes bounding boxes, then uses ray casting to determine the visibility of a specific region and identifies whether it is occluded based on the results. The experimental results demonstrate that the proposed algorithm effectively improves rendering performance during model navigation, alleviating the issue of low rendering smoothness on the web and providing strong technical support for BIM lightweighting. However, despite the notable progress achieved in optimizing rendering processes, challenges related to model loading have emerged. The octree construction process is time-consuming, resulting in longer-than-expected model loading times. A potential solution lies in storing the constructed octree for reuse during subsequent model loading, thereby reducing initialization time. This approach represents a promising direction for future research.

References

1. Liu, Z., Xiong, C., Ding, Z. (2023). Research on BIM Data Extraction Method Based on IFC and Ontology Interaction. In *Proceedings of the 9th National BIM Academic Confer-*

ence (pp. 7). China Society of Cartography and GIS, Shenzhen University, China-Australia BIM and Smart Construction Research Center; Artificial Intelligence and Digital Economy Guangdong Provincial Laboratory; Key Laboratory of Resilient Infrastructure in Coastal Cities, Ministry of Education; Shenzhen Subway Underground Station Green, Efficient and Intelligent Construction Key Laboratory. DOI: 10.26914/c.cnkihy.2023.048037.

2. Chen, Q., Wang, W. (2022). Research on BIM Model Lightweighting and Key Technologies for 3D Display. *Guangdong Civil and Architecture*, 29(05), 1-5. DOI: 10.19731/j.gdtmyjz.2022.05.001.
3. Wang, X., Dong, Z., Chen, B., et al. (2024). Research on Web-based Revit Model Lightweighting and Large Screen Display. *Journal of Qingdao University of Technology*, 45(01), 73-81.
4. Chao, Y., Niu, Z., Qi, H. (2020). Research on BIM Model Visualization Based on WebGL. *Hydropower Energy Science*, 38(09), 79-82.
5. Liu, G., Chen, G., Chen, Q., et al. (2023). Key Technologies for Efficient Visualization of Large-scale 3D Spatial Models Based on the Web. *Geospatial Information*, 21(12), 8-13.
6. MIAOR, SONG J, ZHU Y. 3d geographic scenes visualization based on webgl[C]// 2017 6th Inter-national Conference on Agro-Geoinformatics. IEEE, 2017: 1-6.
7. Guo, J., Cheng, P., Yan, Q., et al. (2019). Research on Simplification and Visualization Methods of 3D Complex Models in Web Environment. *Surveying and Mapping Engineering*, 28(02), 45-51. DOI: 10.19349/j.cnki.issn1006-7949.2019.02.009.
8. Bian, G., Chen, W. (2021). Research on Web-based Visualization Methods for Building 3D Models. *Journal of Graphics*, 42(05), 823-832.
9. Junhwi C ,Chaehyeon K ,Jae K L , et al.Web-based agricultural infrastructure digital twin system integrated with GIS and BIM concepts[J].Computers and Electronics in Agriculture,2023,215
10. Mohamed G A ,Abdallah R M ,Marzouk M .BIM and semantic web-based maintenance information for existing buildings[J].Automation in Construction,2020,116
11. Calin B ,José Á M H ,Antonino M , et al.A framework using BIM and digital twins in facilitating LCSA for buildings[J].Journal of Building Engineering,2023,76
12. Xu, Z., Zhang, L., Suo, H., et al. 3D Tiles Data Visualization of Buildings Based on Industrial Foundation Classes. *Journal of Zhejiang University (Engineering Edition)*, 2019, 53(06): 1047-1056.
13. Li, S., Huo, L., Shen, T., et al. 3D Building Model Edge Folding Simplification Algorithm Considering Angle Errors. *Journal of Wuhan University (Information Science Edition)*, 2021, 46(08): 1209-1215. DOI: 10.13203/j.whugis20190269.
14. Wang, Q., Li, Z., Li, C., et al. Research on Web-Based Lightweight BIM Model Display and Rendering Control. *Computer Measurement and Control*, 2022, 30(09): 177-183. DOI: 10.16526/j.cnki.11-4762/tp.2022.09.027.
15. Xu, H., He, B., Zhang, C., et al. Real-Time Rendering of Urban 3D Scenes Based on Potential Visible Set. *Geodesy and Geographical Information Science*, 2023, 48(02): 80-85. DOI: 10.14188/j.2095-6045.2022376.
16. Jin, H., Liu, H., Miao, B. Horizon Occlusion Culling Algorithm in Real-Time Rendering of Large-Scale Terrain. *Surveying Science*, 2010, 35(06): 84-86. DOI: 10.16251/j.cnki.1009-2307.2010.06.065.
17. Xu, H., He, B., Kuai, X., et al. Efficient Visualization of 3D Urban Scenes by Integrating Hierarchical Level of Detail and Potential Visible Set. *Surveying Bulletin*, 2024, (01): 102-108. DOI: 10.13474/j.cnki.11-2246.2024.0117.