Research on Loading 3D Model Based on World Wind

Haowei Zhang, Xingdong Wang, Huilin Niu

College of Information Science and Engineering, Henan University of Technology, Zhengzhou, Henan 450001 China

Abstract: As an open source 3D engine, World Wind does not directly provide interfaces for loading 3D model, which greatly limits its application scenarios. To solve this problem, this paper proposes an improved BSP (Binary Space Partitioning) algorithm to load 3D models. First, use the improved BSP algorithm to preprocess the 3D models, thereby increasing the speed of loading 3D models. Second, based on the World Wind open source software, the model loading algorithm is implemented and a single model is loaded. Finally, use the JSON files as the configuration information for loading 3D models, read the files to load multiple models and complete the construction of the 3D scene. Experimental results show that the improved BSP algorithm improves the loading and rendering speed of the 3D model.

Keywords: World Wind, BSP algorithm, loading 3D models

1. Introduction

With the rapid development of geographic information technology, more and more scientific research institutions and companies begin to develop 3D geographic information systems based on the open source 3D engine. Among the many open source 3D engines, NASA's World Wind is favored by developers because of its stable performance. However, World Wind does not directly provide interfaces for 3D model loading, and problems such as slow loading speed, high memory usage, and poor display effects often occur during the loading process, which brings great inconvenience to subsequent research. In response to this problem, scholars did a good deal of research and proposed many solutions. Li et al. select the appropriate size after dynamically storing the block data through the database and use LOD (Levels of Detail) technology to speed up model generation [1]. Li et al. use a real-time rendering algorithm based linear quadtree to reduce the amount of data through downsampling [2]. Sun et al. realize rendering performance optimization by establishing multi-layer LOD simplification [3]. Wang et al. use quadtree to simplify terrain and combine multi-core CPU parallel computing to improve loading efficiency [4]. Olanda et al. propose a wavelet pyramid for compressing terrain data, which is conducive to data transmission [5]. Zhang et al. use the BSP tree in the binary space to complete the loading of the model in the Vega software and realize the interaction of the characters in the virtual scene [6]. However, the computational complexity of the BSP tree algorithm is relatively large, and the efficiency will be reduced when dealing models in complex scenes.

Based on the above research, this paper proposes an improved BSP algorithm to load 3D models. First, use the improved BSP algorithm to preprocess the 3D models, thereby increasing the speed of loading 3D models. Second, based on the World Wind open source software, the model loading algorithm is implemented and a single model is loaded. Finally, use the JSON files as the configuration information for loading 3D models, read the files to load multiple models and complete the construction of the 3D scene. Experimental results show that the improved BSP algorithm improves the loading and rendering speed of the 3D model.

2. BSP algorithm

2.1. Principle of BSP Algorithm

The basic idea of the algorithm is as follows: It tries to organize all the boards (They are called planes in BSP) into a tree. Each plane divides the space it is in into two parts, which are divided into smaller spaces by other planes until the end, determine the occlusion order of each room (It is called leaf in BSP) relative to the eyes.

The expression of the BSP algorithm is shown below:

$$k' = [(PM)^{-1}]^T k \tag{1}$$

Where K represents the plane of the coordinate system in the 3D scene, P represents the projection matrix, and M represents the change matrix from the initial coordinate system of the 3D scene to the target coordinate system. The calculation formulas for the maximum dot product d_{max} and the minimum dot product d_{min} of the frustum are as follows:

$$d_{\max} = |k'_X| + |k'_y| + |k'_z| \tag{2}$$

$$d_{\min} = |k'_X| - |k'_y| - |k'_z| \tag{3}$$

By calculating whether d_{min} is less than zero, it can be used to determine the negative half space of a 3D object.

In a 3D scene, if you want to perform cross-domain clipping, set the vertical vertex of the region ($V=\{V_1,V_2,V_3,\ldots,V_n\}$) as the vertex of a convex polygon. If the formulas (4) and (5) are satisfied, the node w is added.

$$LV_i > 0 \tag{4}$$

$$L\,V_{i+1} < -\ s \tag{5}$$

Where L is the cutting surface, and s is the experimental parameter.

In order to reduce the complexity of the BSP algorithm, a fuzzy space segmentation method is constructed using the principle of the sphere tree, and the original fine segmentation is replaced with a rough space segmentation. The key is to divide the boundary sphere into all polygon sets, and all the objects in the set do not need to be subdivided again. The calculation formula for the new node w is as follows:

$$w = V_i + t(V_{i+1} - V_i) \tag{6}$$

$$t = \frac{LV_i}{L(V_{i+1} - V_i)} \tag{7}$$

The calculation formula of the new cutting surface generated by V_i and V_{i+1} is as follows:

$$L_i = \langle \frac{V_{i+1}V_i}{\| V_{i+1}V \|}, 0 \rangle \tag{8}$$

2.2. BSP Algorithm Implementation

2.2.1. Single model loading

The prepared 3D model data processed into dae format, and the model is loaded based on the improved BSP algorithm. The specific algorithm flow is shown in Figure 1.

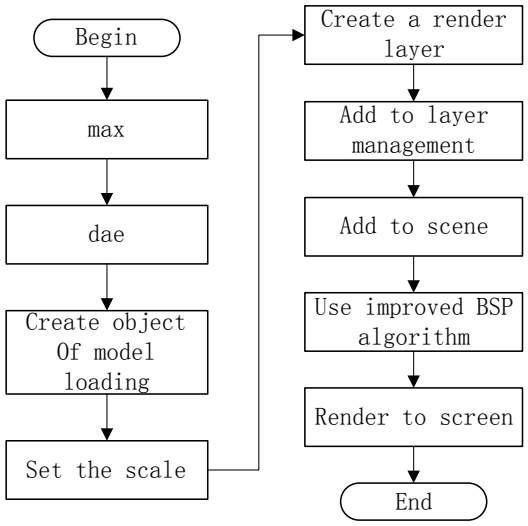

Figure 1. Single model loading

First, the data format of the 3D model is converted to dae, and the preprocessing of the model data is completed based on the improved BSP algorithm. The WorldWind.ColladaLoader class instantiates to create a model loading object colladaLoader, and transfers it to the current model to load the position information in the scene. Second, pass the local disk storage path of the current model file as a parameter to the init method of the colladaLoader object, and set the scale and opacity of the loaded object. Then use the WorldWind.RenderableLayer class to create the rendering layer object SModelLayer of the current model, and add the rendering layer to the layer manager through the addLayer method of the wwd object, thereby adding it to the scene. Finally, the redraw method of the wwd object is called to render the current model to the screen.

The main code is as follows：

```
function AddModel(renderLayerName, arg_position, sDirPath, docName, nScale, opacity) {
    var SModelLayer = new WorldWind.RenderableLayer(renderLayerName);
    wwd.addLayer(SModelLayer);
    var position = arg_position;
    colladaLoader = new WorldWind.ColladaLoader(position);
    colladaLoader.init({dirPath: sDirPath});
    colladaLoader.load(docName, function (scene) {
            scene.scale = nScale;
        SModelLayer.addRenderable(scene);
    });
    SModelLayer.opacity = opacity;
    wwd.redraw();
    }
```

2.2.2. Multiple model loading

The multiple model loading is implemented on the basis of a single model. The configuration information of multiple models is stored in a file, and then the model loading method is called to shorten the loading time and improve the loading speed.

As a lightweight data format, JSON files are convenient for transmission, analysis and generation in the network. Therefore, the JSON files are used as the parameter information of the scene model, and the batch loading of multiple models is completed through the analysis of JavaScript, and at the same time, it is convenient for users to configure the parameter information of the model externally. The corresponding code of multiple model loading are as follows:

```
$.ajax({
url : "model/Models.txt",
datatype: "json",
async : true,
  data :{},
      success : function(result) {
var dataArray = result;
show(dataArray);
function show(dataArray){
for (var i = 0; i < dataArray.models.length; i++) {
var layerName = dataArray.models[i].renderLayerName;
var latitude = dataArray.models[i].arg_position[0].lat;
var longitude = dataArray.models[i].arg_position[0].lng;
var height = dataArray.models[i].arg_position[0].height;
var position = new WorldWind.Position(latitude,longitude,height);
var path = dataArray.models[i].sDirPath;
var name = dataArray.models[i].docName;
var scale = dataArray.models[i].nScale;
var opacity = dataArray.models[i].opacity;
```

```
            addAnyModel(layerName, position, path, name, scale, opacity);
                        }
                    }
                }
            });
```

2.2.3. Result analysis

In the same operating environment, the model loading method based on the improved BSP algorithm proposed in this paper is compared with the unimproved model loading method. Table 1 shows the comparison of the running time between the method in this paper and the unimproved model loading method.

Table 1. Parameter comparision

Times	BSP algorithm	Improved BSP algorithm
1	35′32″	23′56″
2	36′51″	24′06″
3	35′83″	24′13″
4	36′48″	23′81″
5	34′68″	25′15″

It can be seen from Table 1 that the improved BSP algorithm improves the loading and rendering speed of the 3D model.

3. Conclusion

This paper proposes an improved BSP algorithm to load 3D models. First, use the improved BSP algorithm to preprocess the 3D models, thereby increasing the speed of loading 3D models. Second, based on the World Wind open source software, the model loading algorithm is implemented and a single model is loaded. Finally, use the JSON files as the configuration information for loading 3D models, read the files to load multiple models and complete the construction of the 3D scene. Experimental results show that the improved BSP algorithm improves the loading and rendering speed of the 3D model.

Acknowledgment

This research was supported by the Key Scientific Research Project of Colleges and Universities in Henan Province (No. 19B420001).

References

[1] Li, Y.J.; Tan, T.D. The generation of large-scale terrain and topography in 3D scenes. *Computer Applications and Software* **2013**, 30(11), 131-135.

[2] Li, Q.; Dai, S.L.; Zhao, Y.J.; et al. The block LOD real-time rendering algorithm for large-scale terrain. *Journal of Computer-Aided Design & Computer Graphics* **2013**, 25(5), 708-713.

[3] Sun, Y.L.; Song, G.F.; Feng, Z.H. A method for rendering vector data over global 3D regular terran grid. *Geography and Geo-Information Science* **2013**, 29(6), 14-17+25.

[4] Wang, Q.Y.; Luo, Z. Parallel rendering of massive terrain data based on LOD. *Computer Systems & Application*s **2017**, 26(12), 200-206.

[5] Olanda, R.; Perez, M.; Orduna, J.M.; et al. Improving hybrid distributed architectures for interactive terrain visualization. *Journal of Supercomputing* **2017**, 73(1), 17-28.

[6] Zhang, K. Research on roaming technology of immersing 3D virtual. *Journal of Changchun University of Science and Technology (Natural Science Edition)* **2016**, 39(01), 104-106.