/\* I declare that this work is my own. Author: Bingxun Yang, xavierybx@gmail.com\*/

**3D assignment report**

**Introduction**

A scene graph, an important data structure and consisting of nodes, is commomly used in building the 3D models and it based on the oriented-object program concept which regard every node as an object. A good scene graph could simplifying application development and make the program use the graphic hardware with the optimal.(Hinrichs, n.d.; Reiners, 2002) In the report, it would be around the scene graph to show how the assignment could be finished. Then the effect of each node’s transformation and the APIs for the scene graph would be mentioned and make some comparison between the scene graph used in the project with other open source scene graph. Because the importance of scene graph could not be regardless, the advance approaches of using scene graph would be mentioned. More specially, the occlusion culling approach and the texture-based direct volume rendering, both based on the OpenSG, would be mentioned as the representation of advance approach.

**Scene Graphs for Geometry and Transformation**

In the project, the scene graphs had been basically used. There is a quickly view image as the image 1 showing. The scenes could be dived into six parts which are the desk, the lamp, the pen container, the books, the showing space and the spotlight source. When creating the model in the MyEventListener.java, it only needs to build these six independent objects. (Code showing in image 2) Except the spotlight, building the model of each part needs to build a scene graph at beginning.

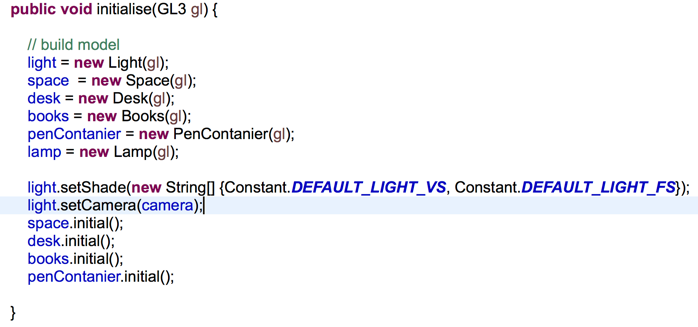


Image 1. The Scene Image 2. The Building Code

Firstly, the space object consists of 10 planes with 4 cubes after transforming. The scene graph is complex and would be showing in below image 3. (“T” for Translate, “S” for Scale and “R” for Rotate) If only render the space model, it would be look like below image 4.

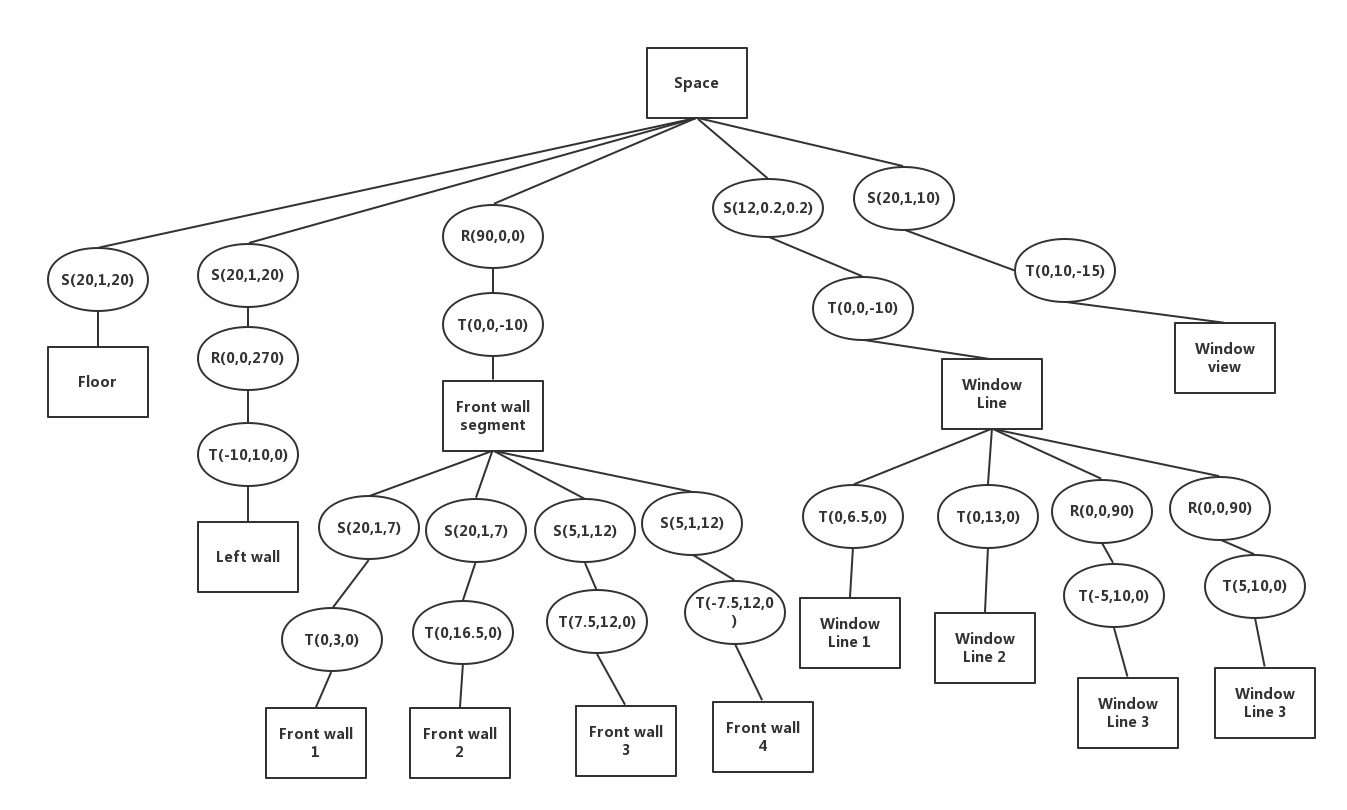


Image 3. Space Scene Graph



Image 4. The Space

Secondly, the desk consists of a platform and four legs and all of the component are transforming from the basis object which is the cube. The scene graph would be shown in below image 5. If only render the space model, it would be look like below image 6.

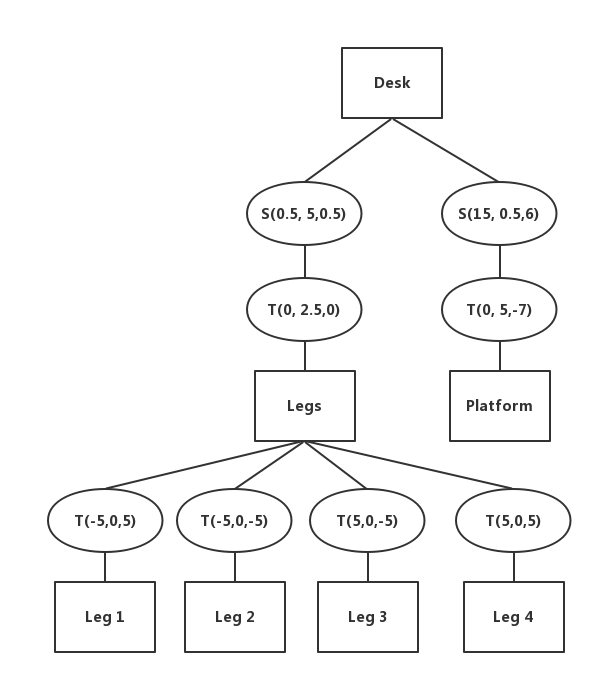
 

Image 5. The Desk Scene Graph Image 6. The Desk

Thirdly, there some stuff had been placed on the desktop. The pen container and the pen are whole object. Same to the books, all books are regarded as one object. The scene graph of these two objects could be represented in one image (Image 7), because it is not complex. The render scene of these stuff could be showing in Image 8.

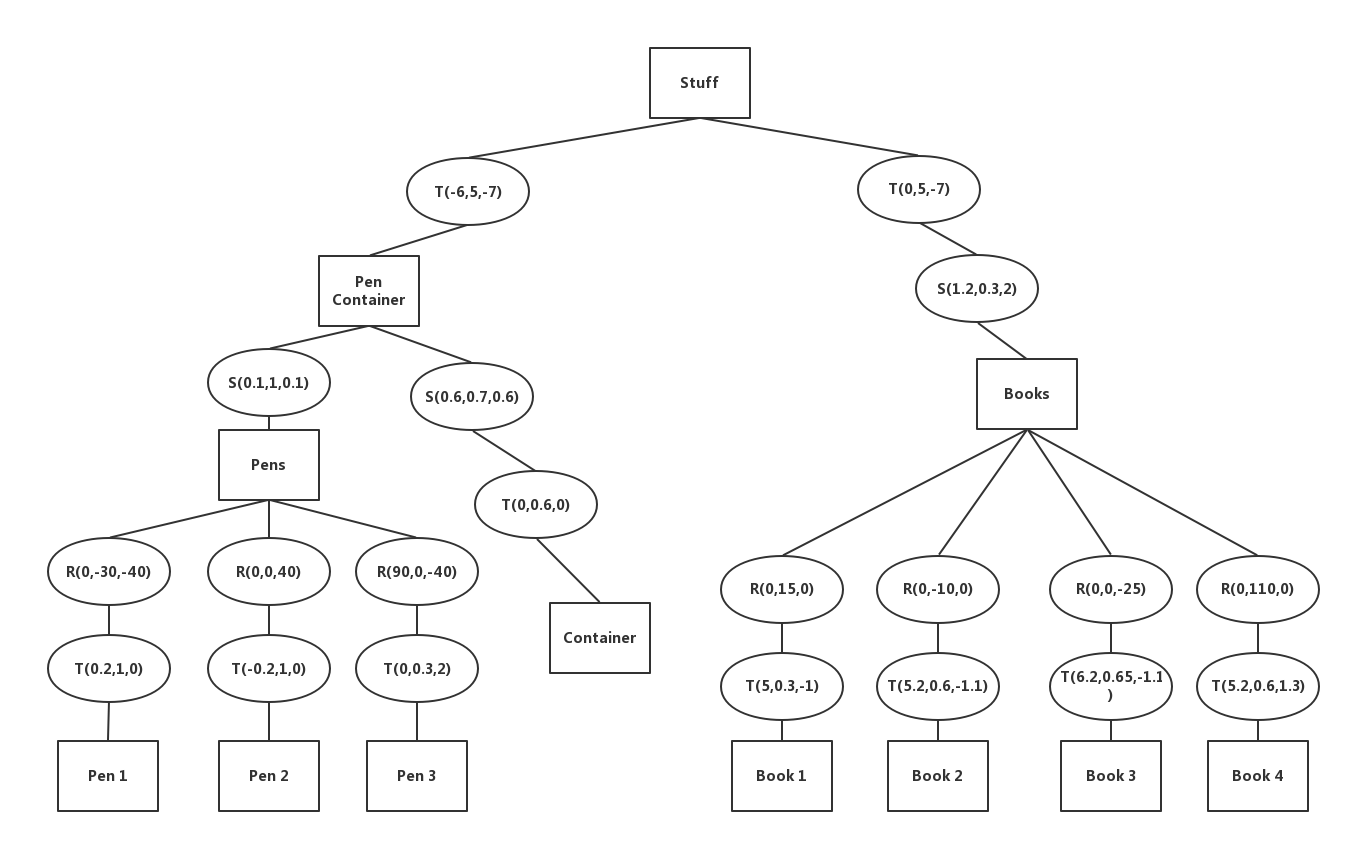


Image 7. The Stuff Scene Graph



Image 8. The Stuff on The Desktop

Finally, the lamp is the most complex model in this project, because the animation would be applied on the lamp and the lamp should be flexible enough so that could do transform easily. The lamp contains a few basis models which are the cylinder, the cone, the sphere and the hemisphere. The scene graph of lamp is in the below image 9. In the image 9, the lamp scene graph, would not specify some parameters in transformation because these parameters are to control the pose when the lamp move and they would be represented in x, y or z. Also, only the lamp rendering scene would show in the image 10.

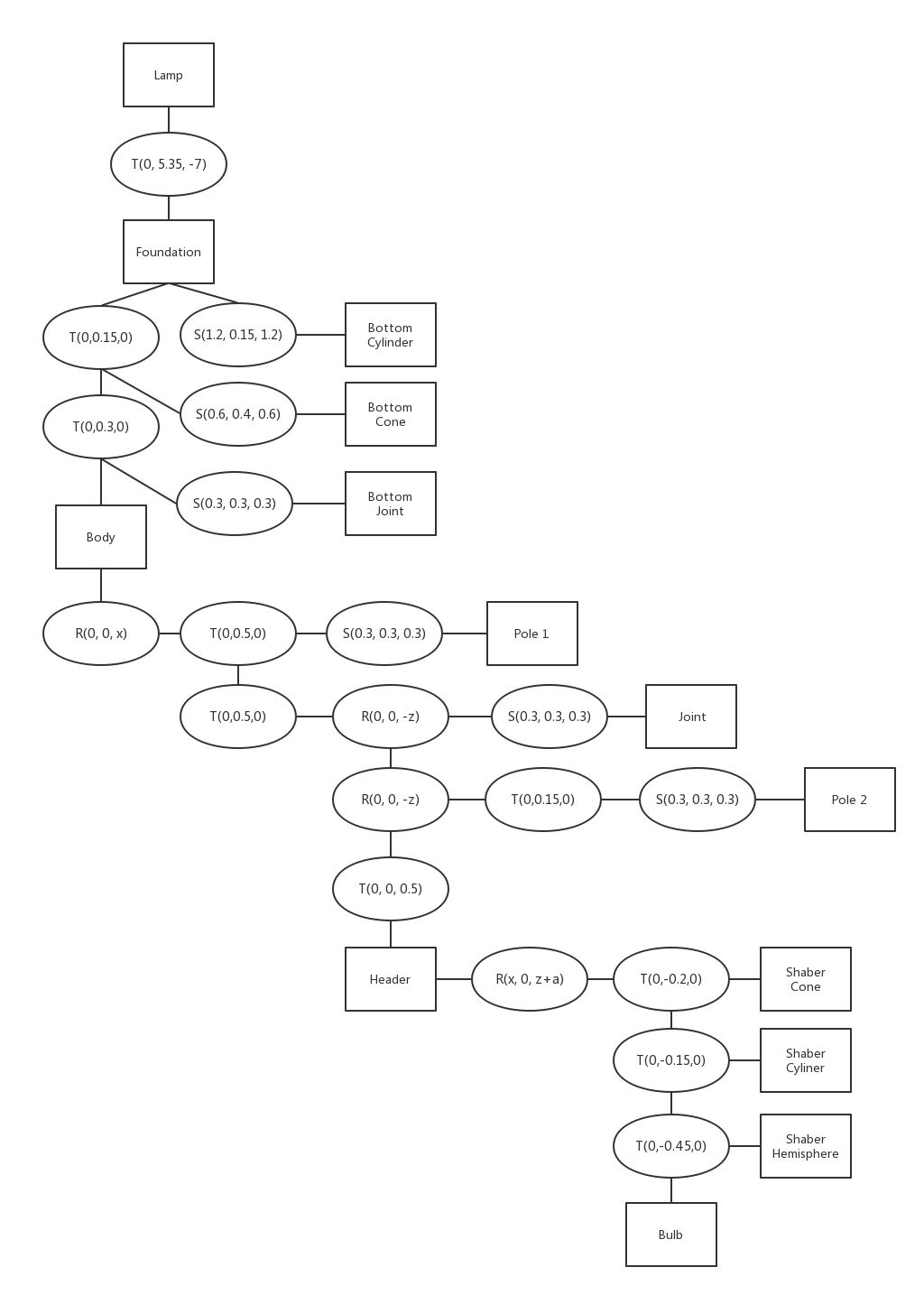


Image 9. The lamp Scene Graph



Image 10. The Lamp

According to the figures shown in above, it could be noticed that is all the children nodes’ transformation is based on the last node’s transformation so that if one node has been changed, then the nodes in following would be changed as well. So, that could connect the component nodes as a whole object and move simultaneously.

**Scene Graph and APIs**

In the implementation, there is a basis class named “BaseNode” which also is a superclass and inhered by component nodes such as pole, bulb, joint, etc. But actually these component nodes are geometry nodes, because these nodes had been created based on the basis geometry such as the cylinder, sphere, cube etc. The class “NodesContainer”, also inhere the class “BaseNode”, is mainly used to collect the component nodes and constitute the higher lever components nodes. Taking the lamp as an example, there are three “NodesContainer”s which are separately “Foundation”, “BodyPole” and “Header”. The lamp consists of these three containers and the containers represent the different parts of structure of the lamp. In each container, it contains the basic component nodes such as “BottomCone”, “JointNode”, “PoleNode”, etc. There is a figure which could show the relationship among “BaseNode”, “NodesContainer” and parts of other geometry nodes. Seeing in following image 11. Expending the class “BaseNode”, some basis attributes and functions contained inside and the following image 12 would show the inside construction.

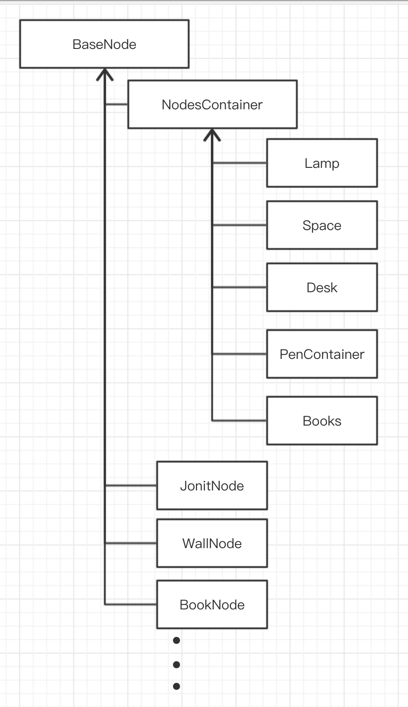


Image 11. Inhered Diagram



Image 12 Class BaseNode

The important functions in “BaseNode” are “addChild()”, “update()” and “render()”. In the program, every node could be the parent node of a new node which means that could add the any new node as the children nodes. Then every node has its own state when call “render()” function, all the transformation matrix would be calculated and pass to the children nodes. The transformation matrix updating is from calling the function “update()”.

The main concept of the scene graph in the project is that separate the state data from the scene graph, it could reduce complexity of the scene graph and also could make the code looks more logical and clear. That is a little similar with the MVC mode (known as Model-View-Control)(Curry, n.d.) which a program mode that broadly used in the developing the visualization program. The SceniX, a scene graph system, is also using the main design concept that separate the data from the scene graph.(Tobler, 2011)

**Advance use of Scene Graphs**

With the 3D graphics hardware rapid development, the computer, already, ships with the 3D accelerators. Almost, the majority of 3D program will use the scene graph, except a few graphic program will directly use immediate mode for some special effect. The existing scene graph make it easier to do 3D graphical program because it shift the programmers’ attention from triangles, vertices and thinking about controlling the render pipeline to think about how to present the object perfectly.(Sowizral, 2000)

Since the oldest scene graph Inventor was developed, there have created a lot of advance scene graph system. But the Inventor is not obsoleted because it is still used in some significant applications. Y and Performer are nearly as old as Inventor, but they have the different mindset with Inventor which is meant to object-oriented and easy to use. Performer focus on the speed and it was the first system to use multiple processes to do rendering in the parallel pipeline that is famous APP-CULL-DRAW division. Y is a little younger and similar with Performer, but it is good at sounds process and supports projected texture and shadow.(Informatik, 2002) So, there are still some very useful characteristics in the old systems.

Of course, in recent years, there is an increasing number of brilliant scene graph systems. These systems are in order to create much easier environment for the programmers paying more attention on the model and scene.

OpenSG, a scene graph system developed by Reiners, mainly focus on the performance.(Staneker et al., 2004) It has some good extension approach that make the performance better when rendering the 3D scene. The reason why it could perform so well is that the main concept is use the traversal to visit the nodes needed for a certain action.

A famous culling approach named occlusion culling is the characteristic of the OpenSG, that means the approach would not be included in the most scene graph such as Open Inventor, IRIS Performer, Cosmo3D.(Staneker et al., 2004) Basically, the occlusion culling is an approach that only rendering the object that could be seen by camera. There are two image (Image 13 and Image 14) which are screen shot from the YouTube video that is an experiment of using the occlusion culling. The experiment is based on the Unity 3D software. When not applying the occlusion culling, 73.2k triangles has been rendered, but when applying the occlusion culling, only 2.3k triangles has been rendered. So, by using occlusion culling, the cost of rendering has been reduced and the performance has been improved. In the real-time rendering, rendering useful and less triangles are the crucial point of time consuming, especially when the situation comes to the large and complex scene, if the occlusion culling could be applied very well, not only the scene experience would not be affected, but also the consumption of the computer would not increase.

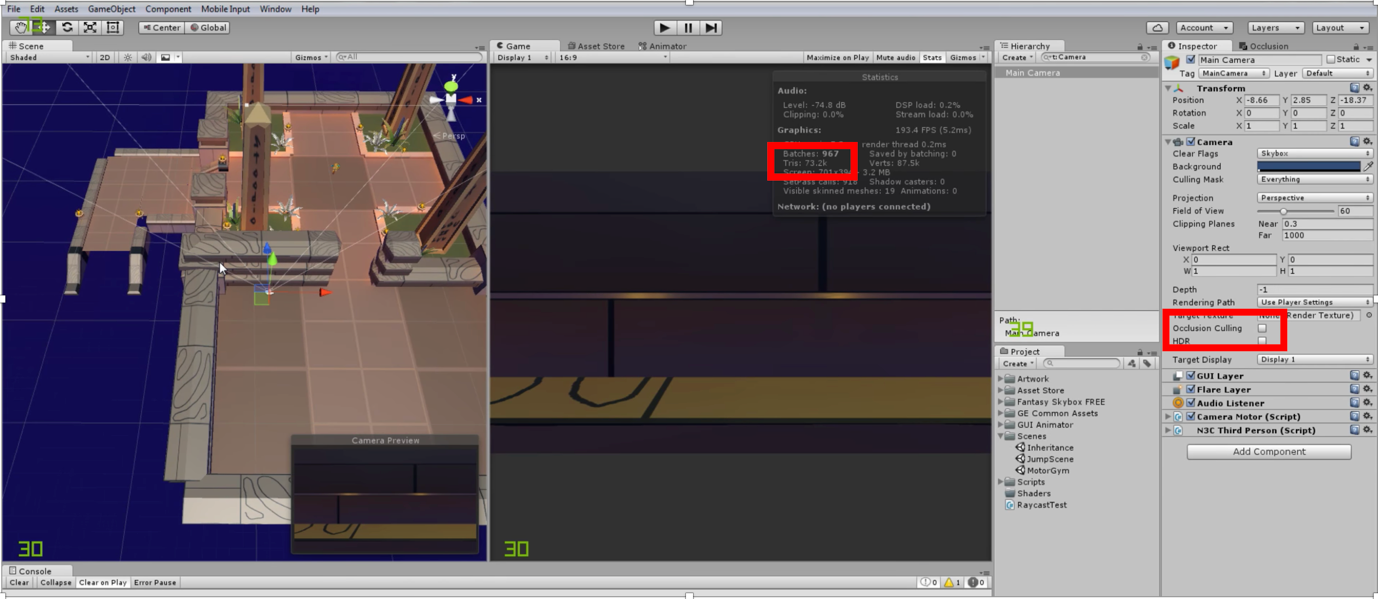


Image 13. Not applying the occlusion culling, 73.2k triangles has been rendered

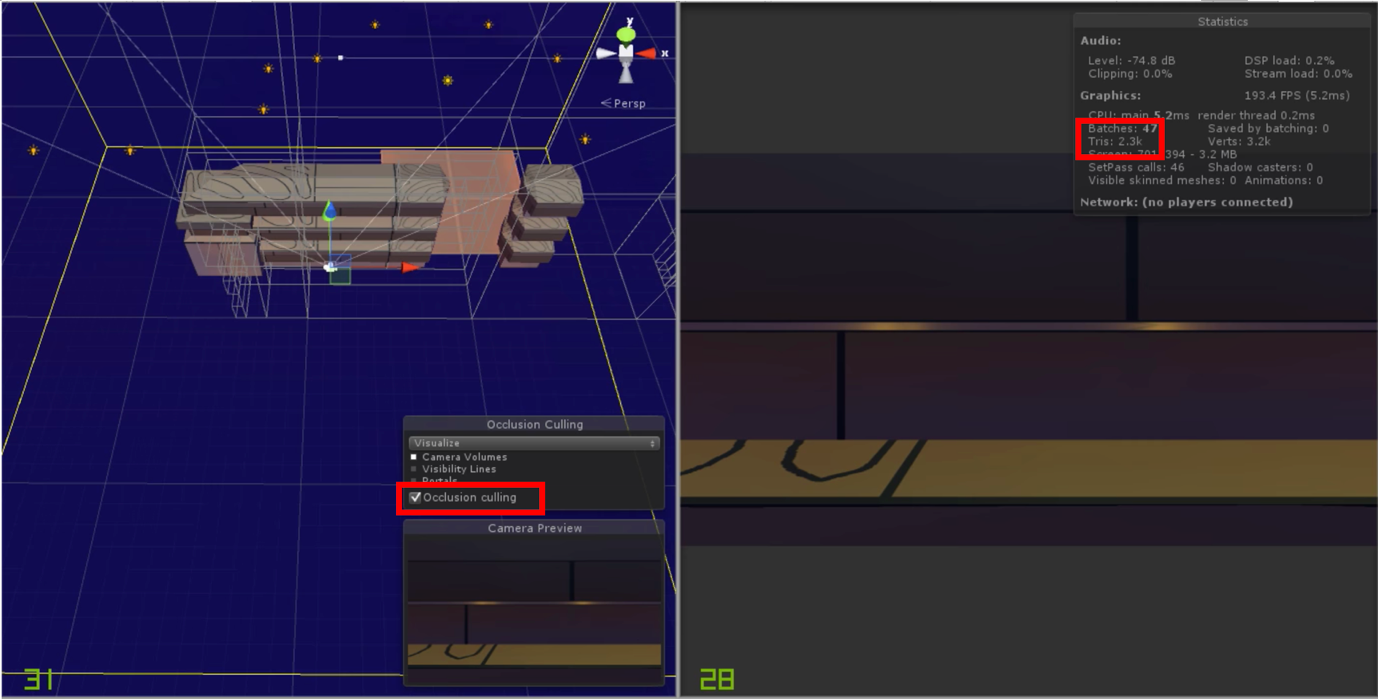


Image 14. Applying the occlusion culling, 2.3k triangles has been rendered

In addition to the occlusion culling based on the OpenSG, there is a 3D volume rendering approach also based on the OpenSG. The importance of 3D volume visualization has dramatically increased in recent years because of the advanced technology’s motivation. It has been broadly used in the many field, such as an important field which is medical ray casting.

OpenSG provide a free, available and portable scene graph system(Weiler et al., 2004) that could be easily extended. There is a textured-based direct volume rendering approach extending the OpenSG by the volume rendering library. The approach is based on the texture-based rendering. The texture-based direct volume rendering is the most important algorithm for rendering volumetric data because it achieves high rates by exploiting the 3D texture hardware of modern graphics adapters.(Weiler et al., 2004) It could adopt two types textures map which are 3D texture map and 2D texture mas and the algorithm only be change a little when rendering 2D texture maps.

The framework consists of four components which are slicer, shader, texture manager and renderer which is a controlling instance. (See the structure in the image 15.) Making the programming become simple is the major goal for this direct volume rendering approach, which also could be named as a framework. It also should be flexible enough so that could adopt numerous graphic chip. So, the shader object is a customized object that implement the callback function as a plug and it could be made some special changes to fit the hardware. In the initialization, the shader component decides the number and the format of the texture maps according to ideal visualization mode. The texture manager could give the registration to the normal texture maps even the multiple textures. Then the slicer using the algorithm to process the texture maps as intensity texture with scalar data. The algorithm is performed by the renderer.

In a word, this framework could be very easy for programmer to use and it allows for the integration of volumetric effects into any OpenSG scene. (Weiler et al., 2004)

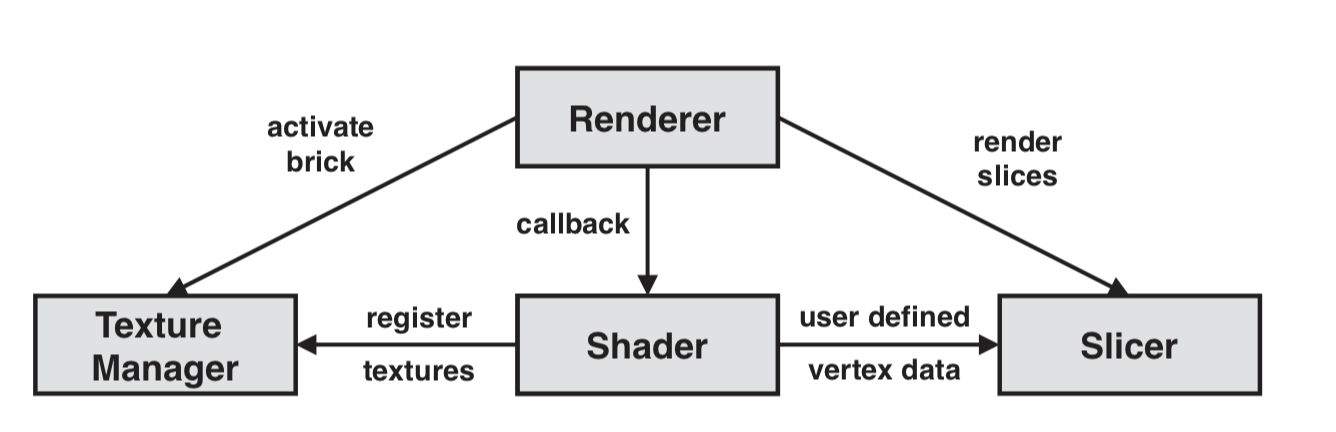


Image 15. Direct volume rendering framework

**Conclusion**

In this report, it mainly talks about the scene graph applied in the lamp project and comparing the approach used in the project with the traditional approach. In this project, it adopts a method that is a little similar with the MVC mode which separates the state data from the scene graph and updates the data outside out the scene graph tree. Using this method could make the code looks more logical and the structure to be clearer. Some advance approaches of using scene graph also had been mentioned in the report and it mainly based on the OpenSG system. The occlusion culling is approach that could improve the rendering performance when rendering the large and complex scene. The direct volume rendering is an approach that optimize the traditional volume rendering approach and improve the texture-base 3D volume rendering performance, meanwhile the new approach has good compatibility with the different graphic chip.

In a word, choosing a good or an appropriate scene graph is the key point to make a 3D computer graphic program. Good scene graph could not only make the programming process become easy and clear but also could improve the program performance when running the program.

**Reference**

[1] Curry, E. (n.d.) ‘Flexible Self-Management Using the Model-View- Controller Pattern’ pp. 84–90.

[2] Hinrichs, K. (n.d.) ‘A Generalized Scene Graph.’

[3] Informatik, V. F. (2002) ‘OpenSG : A Scene Graph System for Flexible and [4] Efficient Realtime Rendering for Virtual.’

[5] Reiners, D. (2002) ‘Scene Graph Rendering’ pp. 1–18.

[6] Sowizral, H. (2000) ‘Scene Graphs in the New Millennium’ (February) pp. 56–57.

[7] Staneker, D., Bartz, D. and Stra, W. (2004) ‘Occlusion Culling in OpenSG PLUS,’ 28 pp. 87–92.

[8] Tobler, R. F. (2011) ‘Separating semantics from rendering : a scene graph based architecture for graphics applications’ pp. 687–695.

[9] Weiler, M., Klein, T. and Ertl, T. (2004) ‘Direct volume rendering in OpenSG,’ 28 pp. 93–98.

[10] N3K EN (2016). Simple occlusion culling - Unity 3D. [Viewed 12 December 2018]. Available from: https://www.youtube.com/watch?v=OmuQmydipGg