**学　修　報　告　書**

（特別学修生様式）

**Study Report**

平成　　　2017 　　3　　　31

北陸先端科学技術大学院大学長　殿

To: President of JAIST

1616080

私は、このたび学修が終了しましたので、下記のとおり報告します。

I hereby report the completion of my study described as follows:

記

|  |  |
| --- | --- |
| 在学期間  Period of Stay | 平成　　29 　1 　9 　～平成　29 　4　　7 |
| 研究題目  Research Title | Real-time rendering engine |
| 学　修　概　要Study Summary  Real-time rendering is one important branch in computer graphics, it is commonly used in game industry and also animation industry. Global illumination in real-time rendering is one method of light simulation to achieve both performance and speed. Since traditional ray-tracing is very expensive, it is almost impossible to render the complex scene in real-time. I chose voxel-based cone tracing as my research theme during the period of stay. This algorithm introduces some advanced rendering technique to speed it up.    **Algorithm overview**  **1. Voxelization**  Voxelization is one of the method which can store the information of every voxel in 3d texture. The information can be vertex normal, lighting information and anything else. To do the voxelization, I follow the paper on OpenGL Insight.   * Vertex shader: transform the original vertex information to vertex shader. * Geometry shader: calculate every face normal and choose one axis that this normal dominate to. Project each triangle along its dominate axis of normal, transform the world position to the 3d voxel coordinate (every axis is form 0 to texture size). * Pixel shader: do the light calculation in Pixel shader. If the position is in the 3d texture range, for example 512x512 viewport use 512x512x512 3d texture. Store the voxel information in this 3d texture which is as big as the viewport.   What we get on the screen now are pictures (models in 2d dimension) that parallel project to the screen.    *Figure1: The projected image of bunny in different view*    *Figure2: The different stage of voxelization*   1. **Visualization**   This is a step we visualize the result of voxelization. We do not necessarily need visualization in the voxel cone tracing algorithm, but visualization can help us to check whether the lighting calculation and information saved in the texture is correct or not. The result of visualization also can be used as the rendering style in games such as Minecraft.   * Vertex shader: I send sv\_vertex id to the vertex shader, the id is from 0-256x256x256 automatically by directx11. So I can get the position same to the 3D texture coordinate by the sv\_vertex id. Get the data from 3D texture use this position. * Geometry shader: We first check this point saves voxel data or not. Then to the voxels, we send the box offset to it to make it “box voxelization”, to be notice the box offset should be considered relative to the voxel size which is boundingbox/voxel dimension. * Pixel shader: get the color saved in the voxel texture, render the scene with it.     *Figure3: Visualization of voxelized Cornell box in 256^3 dimension(left), 128^3(middle), 64^3(right)*    *Figure4: Visualization of voxelized bunny in 256^3 dimension(left), 128^3(middle), 64^3(right)*   1. **Lighting**   There are two parts of lighting, one is the directing lighting calculate in the voxelization part. The other is calculate in the true shading part.   * In voxelization: we need the light information to send into 3d texture. The light information is use to approximate the lighting performance, since we use this light mainly to check the visibility, we don’t need very accurate lighting result. What we do is trying to get the approximately right lighting but faster speed. So we use Blinn-phong lighting model here. * In conde tracing: in direct lighting, we use the BRDF lighting model to simulate the approximately lighting and material. Here I use Cook-Torrance lighting model. In indirect lighting part, I trace 7 diffuse cones and 1 specular cone.     *Figure5: direct lighting using BRDF model*   1. **Cone tracing**   Cone tracing is one kind of ray-tracing method uses to compute indirect lighting. In this algorithm, we don’t use the initial complex cone tracing. Here we use ray marching but opposite to the “ray” we use “cone” to trace and get the opposite mipmap level.  For every point, we decide the direction, max distance and the sample step, if the ray get some point in this tracing process in 3d texture, we add color to this point. Finally, we gather the direct light and indirect.  But there’s still problem as the following picture, it is the color of diffuse cone tracing, I can get part of the right result after I tracing the 7 diffuse cones along the circle of its normal. See the color respond to its direct lighting scene, the indirect color has bounced, it means the pixel color is right, but it isn’t continuously, it is discrete. I think it is may be the problem of the sample step. I didn’t figure it out until now, but I will still work on it.    *Figure6: direct lighting channel(left),indirect lighting channel(middle),final gathering(right)* | |

上記学修が終了したことを認めます。

I hereby confirm that the student has completed his/her study described above.

平成 29 4 7

　宮田一乘