Assignment 4. Slice-Based Volume Rendering

## Total of Points for this Assignment: 15

In this assignment, you will explore the slice-based volume rendering technique that can be used for visualizing three-dimensional volumetric datasets.

Like previous tasks, you are required to write your program using OpenGL shading language (GLSL) to do volume rendering. Please read GPU Gems Ch39: Volume Rendering Techniques (See the suggested link in Appendix). It contains the detailed implementation process needed for this assignment. Please make sure you read and understand the material, otherwise you won’t be able to finish this assignment correctly.

There are two ways of rendering a volume: slice-based rendering and volume ray-casting. In this assignment, you are required to use slice-based rendering (Please do not use ray-casting to do this assignment). You do not need to implement lighting & illumination part and advanced techniques in Chapter 39.5. The detailed requirements are listed as below:

1. Load 3D Texture objects (\*.raw) and bind its texture uvw onto a cube. The 3D texture coordinates for a cube is: vec3(1.0,1.0,1.0) - cube.position. We provide the cube data that includes edges (no diagonal lines) for volume rendering (see attachment). Please do not use previous cube.obj for this purpose. (3 points)
2. Implement view slicing algorithm shown in GPU Gems Chapter 39.3.1 (Example 39.1). Make sure you understand this algorithm so that you can finish this tessellation algorithm. The users can change sampling rate (number of slices per unit length) on GUI. (4points)
3. Implement a slider function to view your tessellation triangles (in lines) of slices based on current view direction. (2points)
4. Implement blending function and shader programs to visualize volume object. (2points)
5. Implement GUI of transfer function graph. Your graph needs to be able to visualize the transfer function of Alpha value w.r.t. intensity value. You are required to add multiple slider controller to control transfer function mapping of alpha values. Note that the Alpha value needs to be corrected based on the current sampling rate of slices, as shown in equation (3) of Chapter 39.4.3. (2points)
6. Load a colorbar texture and implement transfer function mapping the intensity values to RGB values provided by the colorbar. (2points)

## Appendix:

1. Online link for GPU Gems Ch39: <http://developer.download.nvidia.com/books/HTML/gpugems/gpugems_ch39.html>
2. Cube data, \*.raw texture data, and the code to load \*.raw arrays are provided in the attachment.
3. Hint: when binding 3Dtexture raw data using glTexImage3D, please use GL\_RED as ‘internalFormat’ and ‘format’ property as our raw data is a single channel data. (See [https://www.khronos.org/registry/OpenGL-Refpages/gl4/html/glTexImage3D.xhtml)](https://www.khronos.org/registry/OpenGL-Refpages/gl4/html/glTexImage3D.xhtml)%20%204.The)

https://www.khronos.org/registry/OpenGL-Refpages/gl4/html/glTexImage3D.xhtml

1. The colorbar image is a 10\*256 color image. You only need to use the first row of 1\*256 to get a mapping from intensity value to color RGB.
2. Intensity range retrieved from fragment shader (using texture() function) is [0,1]. You can get a mapping from this intensity value to both Alpha value and RGB value needed for fragment shader. Eg. This value can be mapped to [0,255] for color/alpha transfer function.
3. For opacity correction, the default reference sampling rate s0 can be set to a value in the range of 5-20. Your sampling\_rate, provided by users on GUI, can be set in the range of 100-2000 (See Chapter39.4.3 equation (3) on how to correct the Alpha value based on the current sampling rate).
4. Your compositing equation is dependent on the rendering order of your tessellation sequence (back to front/ front to back). For front to back rendering order, your source blending factor is One, and destination blending factor is OneMinusSRC. For back to front rendering order, your source blending factor is OneMinusDST, and destination blending factor is One. (See Chapter39.4.3 equation (5) and glBlendFunc at <https://www.khronos.org/registry/OpenGL-Refpages/gl4/html/glBlendFunc.xhtml>)