AAA633 Visual Computing (Spring 2021)
Instructor: Prof. Won-Ki Jeong
Due date: May 2, 2021, 11:59 pm.

Assignment 1: Scalar visualization using volume rendering (ver 1.1)

In this assignment, you will implement direct volume rendering for visualizing scalar values defined on a 3D volume grid. Specifically, you are required to implement a ray casting algorithm for three different volume rendering methods; maximum intensity projection, alpha compositing, and iso-surface rendering a ray casting algorithm using alpha compositing. The goal of this assignment is to learn how to use OpenGL to make 3D renderings. An example of expected volume rendering result (alpha compositing) is as follows:
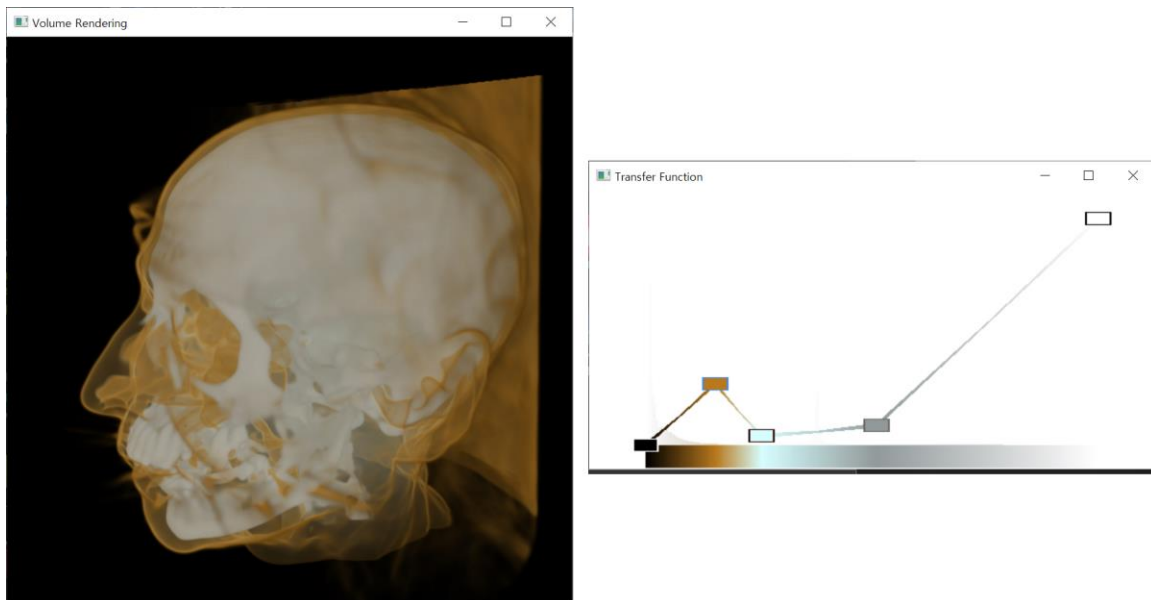

Figure 1. Left : Volume Rendering Window, Right: 1D Transfer Function Editor

The provided source code already contains a partial implementation of the OpenGL viewer and a transfer function editor. The provided skeleton code is tested on a Windows PC and Microsoft Visual Studio 2019 (community version). Use CMake to generate a visual studio solution/project file. CMake is a platform-independent project generation tool (http://www.cmake.org/).

You are required to <u>modify and submit</u> the following three files only; `main.cpp`, and `volumeRendering.frag/vert`. Note that you should not change the other files because we will use those unmodified files when we check your code for grading.

1. Overview of the skeleton code

The provided code generates two viewing windows; one is the transfer function editor, and the other is the main volume rendering window. The volume rendering code is only a skeleton and renders the color of the center voxel value mapped by the transfer function. However, the transfer function editor is a fully working code and you do not need to change that.

The below is the instruction of transfer function editor:
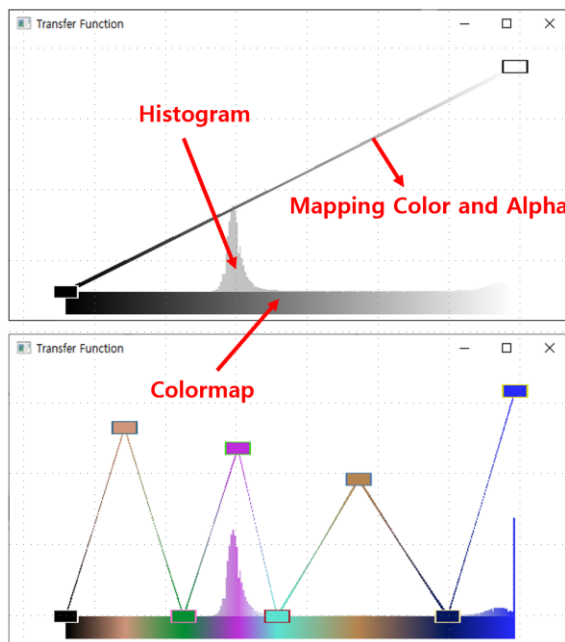
Transfer Function axis

- X: voxel data value (0~255, 8bit)
- Y: alpha
- Node color: R,G,B
- Edge color: color interpolation of end nodes

Transfer Function control

- Mouse left button drag: change node position
- Mouse Right button click: change node color (randomly assign color when it clicks)
- Shift key + Mouse left button click: create new node
- Shift key + Mouse right button click: delete existing node

The provided skeleton shader code (`volumeRendering.frag`) shows an example of how to access 1D and 3D textures in the shader to read the transfer function table and voxel intensity value from the input 3D volume.

2. Volume rendering via ray casting (90 points)

To make the volume renderer working, you need to modify/complete the main OpenGL code and fragment shader. There are two main parts you need to implement:

2.1 Virtual trackball and ray casting framework

In this part, you need to generate an OpenGL viewer with virtual trackball functions (rotate/zoom using mouse interaction). Make sure the left-mouse

button is rotation and the right-mouse button is zooming. You can change the eye position and up vector in the OpenGL viewer using mouse event handlers (mouse click/move), and pass these parameters to the fragment shader to finally generate the ray for sampling.
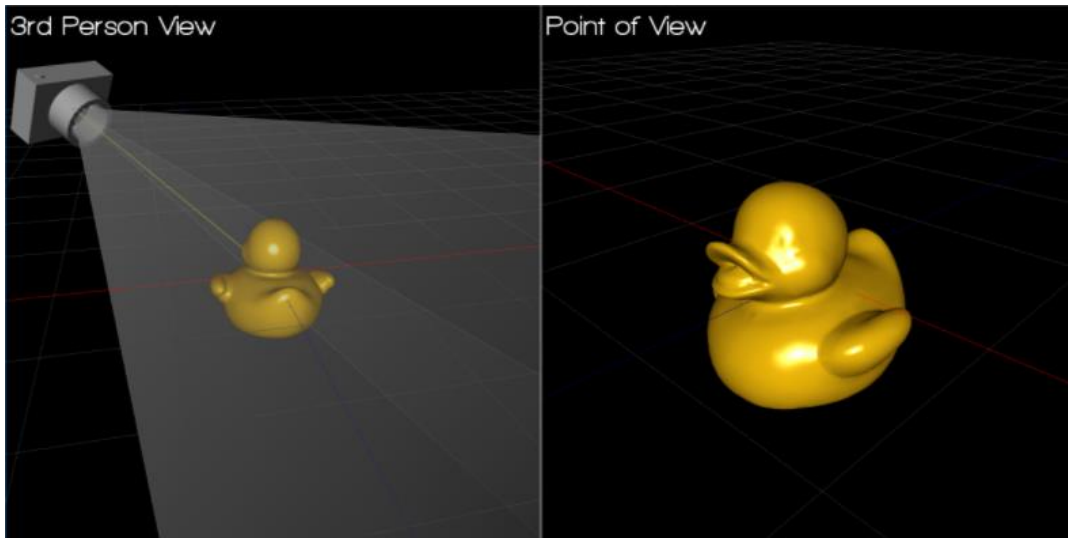

Figure 2. Example of viewpoint and rendered result,
http://www.songho.ca/opengl/gl_camera.html

Once the eye position and per-pixel location are passed to the fragment shader, then the viewing ray can be generated, and you should calculate the intersection between the viewing ray and six sides of the cube to find the two endpoints, one is the entry and the other is the exit point. Then, ray casting is simply walking on the viewing ray between these two end points, collects voxel values, computes the color value using a transfer function table and blend them together based on the rendering algorithm.
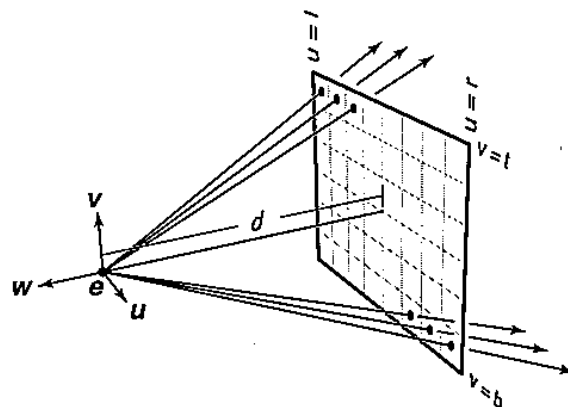

Figure 3. Ray setup for perspective viewing

2.2 Rendering algorithms

Once the entry and exit points are found, then you can walk on the viewing ray, starting from entry to exit end points. The number of samples (or the step size `dt`) can be defined on your own. You should implement three different rendering algorithms: maximum intensity projection (MIP), alpha compositing, and iso-surface rendering.

For maximum intensity projection, you need to pick the maximum intensity among the sampled values.

For alpha compositing, first you should read the volume intensity at the given sampling location using a 3D texture (`sampler3D tex`), and then you should use the transfer function texture (`sampler1D transferFunction`) to convert the voxel intensity to RGBA value. Using RGBA, you should apply alpha blending. You should use front-to-back compositing and early termination (if alpha value is close to 1.0 then terminate). You also need to adjust alpha to allow easy control of transparency (alpha^x will make alpha value smaller).

For iso-surface rendering, first you should find the location where zero-crossing occurs. This can be detected as comparing two consecutive sampling points (say x1 and x2) and check whether the given iso-value is between these two sample values. Once you detected the zero-crossing, you may need to refine the location (e.g., starting from x1 but reduce the step size by half and find a zero-crossing location, repeat this until you find a sufficiently small range of (x1,x2) that includes the iso-value). Then, you should compute the surface normal at the zero-crossing point by calculating gradient vector (dx, dy, dz). Last, you should compute the color on the sampling point using the Phong illumination model (diffuse, specular, ambient).

Switching between rendering modes should be done by pressing 0, 1, or 2 in the keyboard.

Grading:
- OpenGL viewer using virtual trackball (20 pts)
- MIP (20 pts)
- Alpha blending (20 pts)
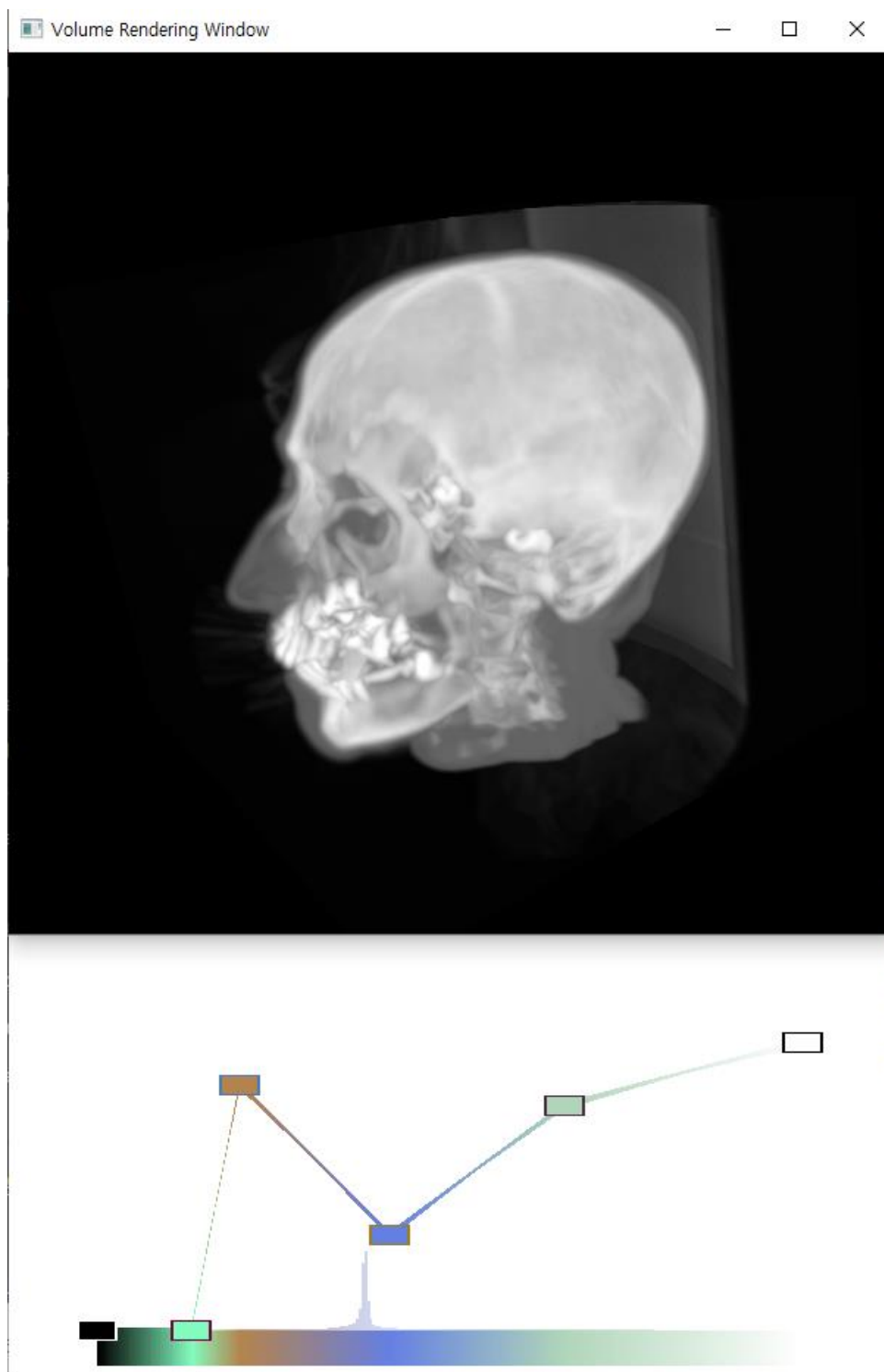- Iso-surface rendering (30 pts)

3. Report (10 pts)

Submit a report describing your code, implementation details, and rendering results. There are 5 volume data (.raw files) in the provided code. Make the most beautiful rendering images (one per volume data) and include them in the report.
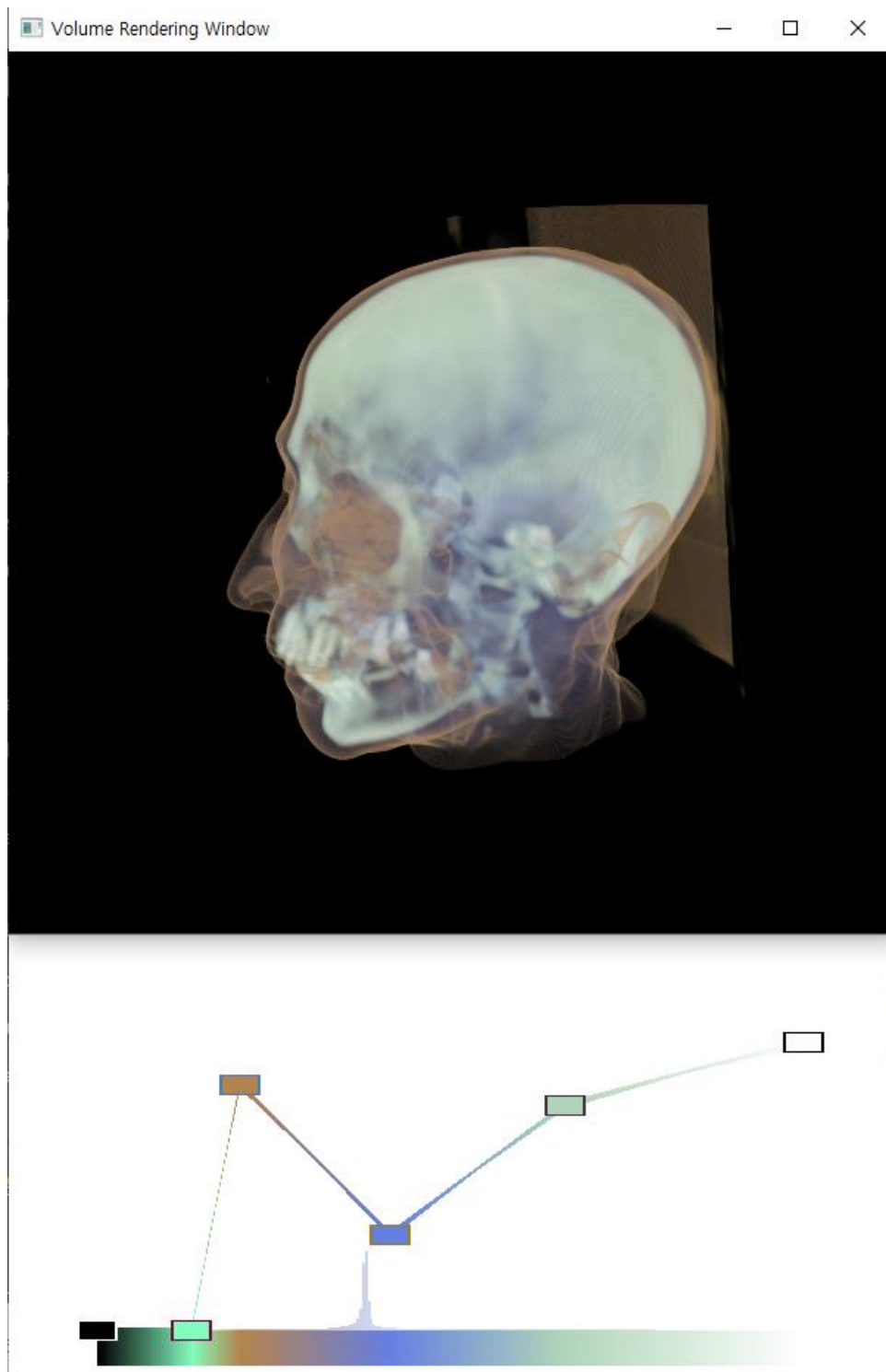
What to submit:
- `volumeRendering.frag,` `volumeRendering.vert` and `main.cpp` files (do not submit other files)
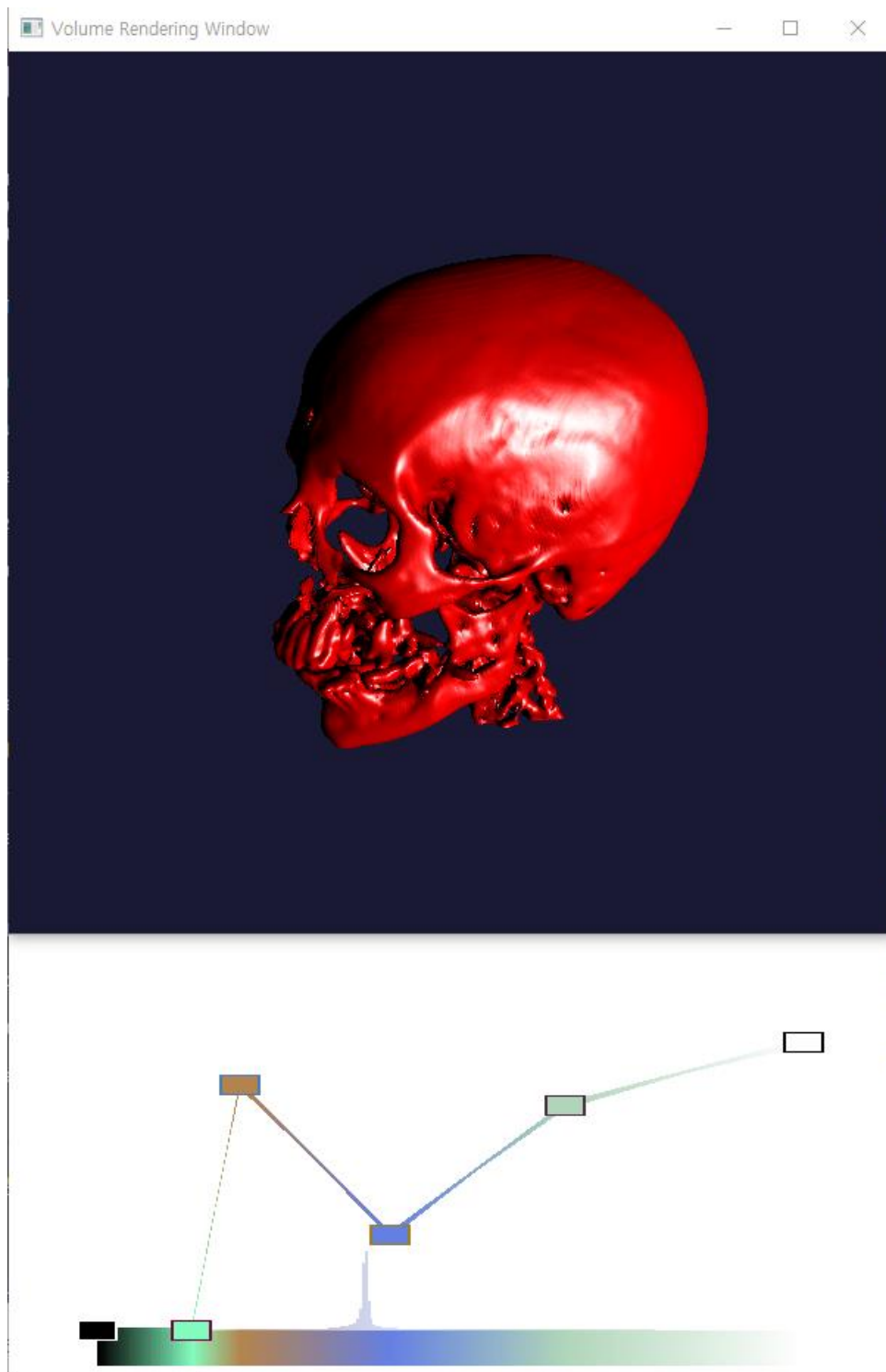- A report (pdf format)

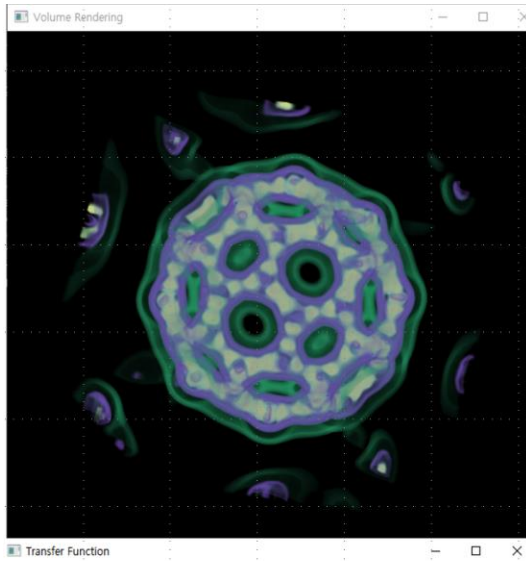Good luck!!!

Examples:

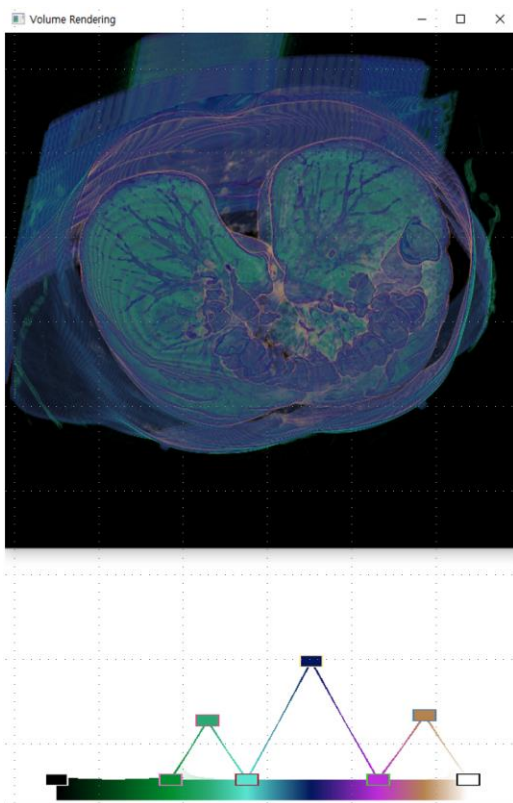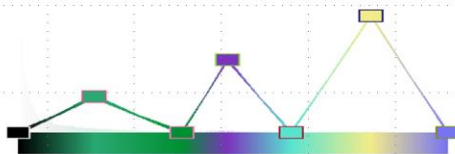Maximum Intensity Projection 예시

Alpha compositing 예시

Iso-surface rendering, iso-value: 0.52

Bucky_32_32_32.raw

Dimension (x, y, z): 32, 32, 32

lung_256_256_128.raw

Dimension (x, y, z): 256, 256, 128