COSE436 Interactive Visualization (Fall 2024)
Instructor: Prof. Won-Ki Jeong
Due date: Dec 15, 2024, 11:59 pm.

Assignment 4: Volume Rendering

In this assignment, you will implement direct volume rendering. Specifically, you are required to implement a ray casting algorithm using three different rendering methods, such as 1) maximum intensity projection, 2) isosurface rendering, and 3) alpha compositing. An example of alpha blending volume rendering result is as follows:

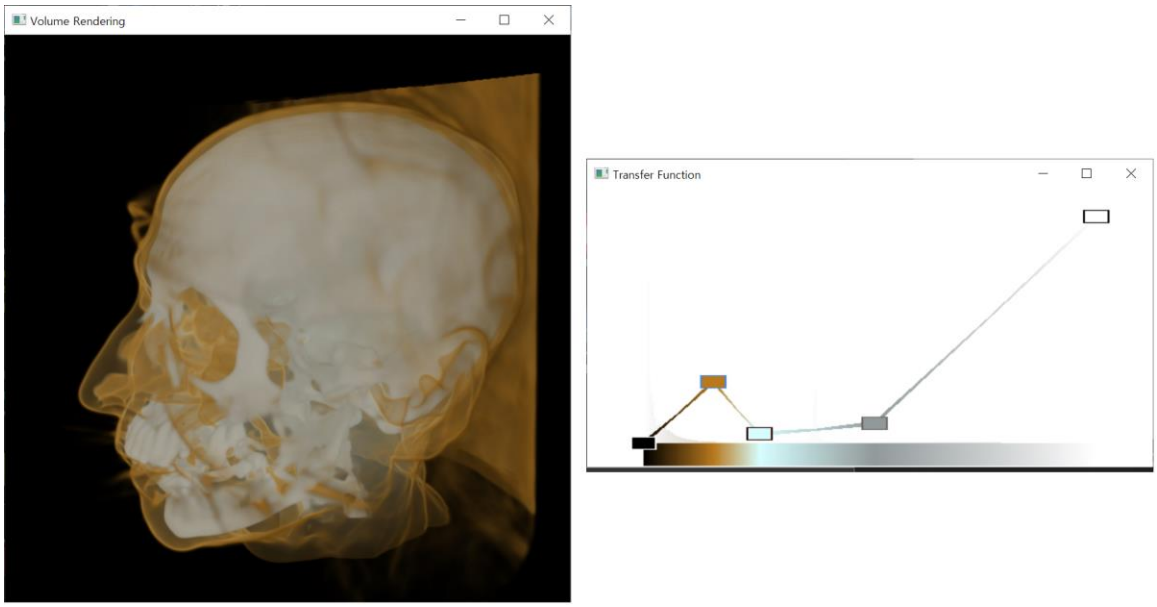


Figure 1. Left : Volume Rendering Window, Right: 1D Transfer Function Editor

The skeleton code includes the implementaiton of transfer function editor (tfeditor.h), so you only need to implement the viewer and volume rendering code. You are allowed to modify main.cpp and shader codes only (.vert and .frag).

1. Overview of the skeleton code

1.1 Viewer

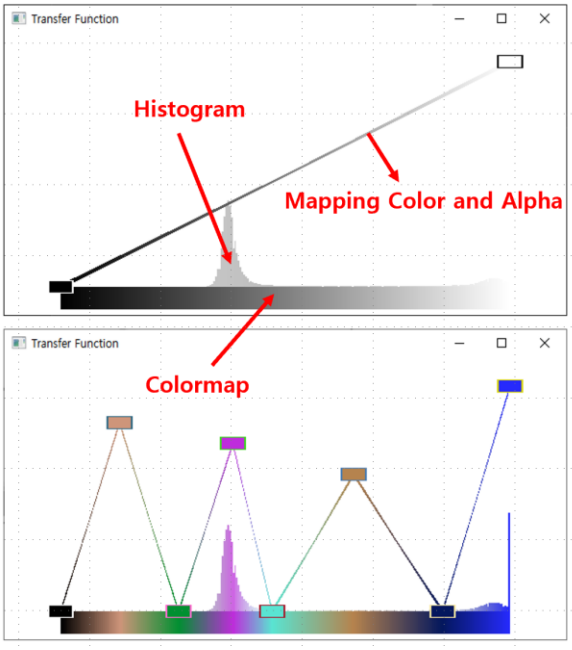
You are required to implement virtual trackball functions for rotation and zooming of the 3D volume (Left-click is for rotation and right-click is for zooming). You need to implement a proper mouse callback function to handle

user input and pass the volume and viewer parameters to the shader. The following viewer-related uniform variables are pre-defined in the fragment shader:

`Vec3 eyePosition` : location of the viewer
`Vec3 up` : Up vector of the viewer
`Vec3 objectMin` : the smallest x/y/z of the input volume
`Vec3 objectMax` : the largest x/y/z of the input volume

1.2 Trasfer function editor

The below is the usage instruction of the provided 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

2. Ray casting (90 points)

To make the volume renderer working, you need to complete the fragment shader. There are two parts you need to implement as follows:

2.1 Ray-cube intersection

In this part, you need to compute intersection between the viewing ray and six sides of the cube to find two endpoints, one is the *entry* and the other is the *exit* point.

2.2 Sampling and compositing

Once the entry and exit points are found, then we need to walk on the viewing ray, starting from the entry point to the exit point. The number of samples (or the step *dt*) can be defined on your own. You should implement three volume rendering methods: maximum intensity projection (MIP), isosurface rendering, and alpha compositing.

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

For isisurface rendering, you need to stop at the zero-crossing location, and compute surface normal at the intersection point. Then you can apply Phong illumination for shading. More details can be found in the lecture notes.

For alpha compositing, you should use the transfer function texture to convert the sampled voxel intensity to RGBA value. It is recommended to use 1D texture to pass the trasfer function table from your C++ code to the shader. Note that 1D texture sampler is already defined in the fragment shader (`sampler1D transferFunction`). 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 (pow(alpha,x) with a proper choice of x will make the alpha value smaller).

You also need to implement keybard callback to switch between three different rendering methods, for example, press "1" is MIP, press "2" is iso surfacing, and press "3" is alphs compositing.

3. Report (10 pts)

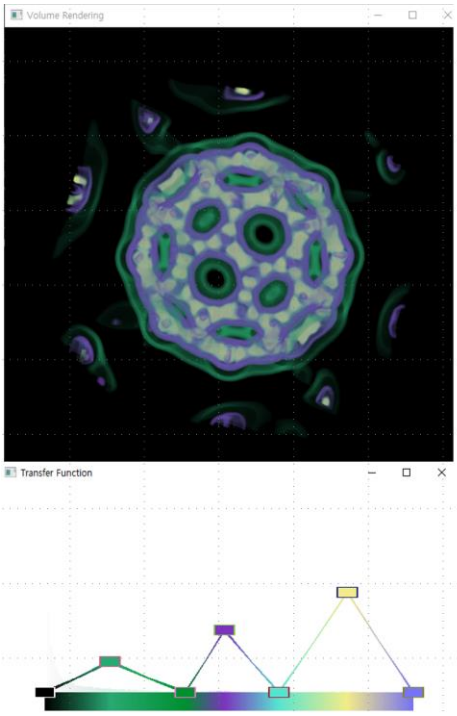
Submit a report describing your code, implementation details, and rendering results. There are 5 volume data (.raw files) in the provided code. Make the most beautiful rendering images (one per volume data) and include them in the report.

What to submit:

- Main.cpp and shader (.vert and .frag) files
- Report (pdf format)

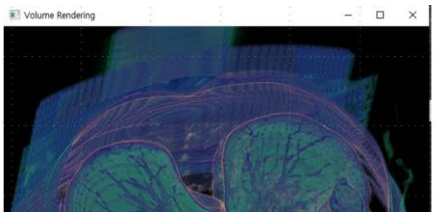
Good luck!!!

Rendering Examples:



Bucky_32_32_32.raw

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



lung_256_256_128.raw

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