Visualization HW2
Seren Lowy – 2023/10/09

1.1 Marching Cubes: Limitations

The marching cubes algorithm is easy to understand, but it has several limitations. First, it is not computationally optimal in either time or space, so it wastes CPU cycles and memory resources. It wastes memory and time by generating many examples of redundant output data, such as meaningless triangles of zero area sharing the same edge and vertex space as existing nonzero triangles. [1]

Second, isosurfaces produced by the marching cubes algorithm can sometimes be shaped badly so that they appear to have topological holes in regions that were smooth in the original object the data was sampled from, or other kinds of deviations that do not reflect the meaning of the data. This is related to how the algorithm resolves ambiguous cubes containing multiple disconnected regions of intersection with the isosurface. Since the marching cubes algorithm only directly observes edges of the polygonal isosurface geometry, it has to guess at the faces connecting the edges. For cases where multiple, disjointed sets of edges appear on the same cell, multiple ways of spreading facets between the edges are possible. The algorithm only works with one cell at a time, so it ignores information that could help it choose a good pattern of facets. The result is that some cells may contain an unexpected break in the surface, or join regions of the surface where they should be separate or vice versa (similar to the straits/isthmus problem in the 2D version, marching squares). [1]

Finally, the marching cubes algorithm causes information loss by producing output artifacts. In marching cubes isosurfaces, small details in the data may be smoothed over, because a limited resolution of cells are used to capture the input, and features which occur and stabilize wholly within the space of a single cell are usually indistinguishable to the capture process of the algorithm. Sharp edge and corner-like features can be incorrectly captured as smooth or wide angles (“aliasing artifact”), even if they cover more than one cell. Gradient maps of isosurfaces that use marching cubes algorithm output can also be incorrect, in such a way that accidental spots of shadow or bright light can appear when the gradient map is visualized as a shading map, among other consequences. [1]

Sources: [1] Newman, T. and Yi, H. (2006). “A survey of the marching cubes algorithm.”

1.2 Raycasting and Splatting: limitations

Raycasting is a method for volume rendering that uses approximately linear paths through the voxels of a volume to quickly sum color contributions of the included voxels to each pixel of the screen. The rays are chosen to align exactly with pixels and voxels (they have slightly irregular shapes like snakes of connected cubes), and be completely tessellating/non-overlapping, so computation is not redundant and can be very fast. But the disadvantages of raycasting include limited ability to display lighting, transparency and occlusion, depending on the compositing algorithm used. To get the best image quality, we need to choose the alpha compositing method, which is also the slowest.

The disadvantages of splatting include slow performance, as well as visual artifacts such as color bleeding and popping. The performance is slow because the splatting algorithm iterates over every sample, in 3D volume space, rather than just every pixel in 2D screen space. It performs redundant computation by painting color over many pixels around each sample, and repainting pixels in multiple passes, often with very minimal contributions from a range of samples that may not be relevant to the appearance of the pixel; in contrast with raycasting, which paints each pixel in exactly one pass and only considers one contribution from each voxel that is strictly touching the ray aligned with the pixel.

Visualization HW2
Seren Lowy – 2023/10/09

Color bleeding is when parts of neighboring voxels appear in the reverse order of the occlusion order of the voxel centers. This happens during compositing due to a compositing technique called sheet compositing that is designed to improve performance. The compositing order of sheets, or flat planes containing layers of voxels, is fixed where all voxels in the same sheet get composited before any voxels in the next sheet. But the isotropic shapes of the footprints of voxels may extend into the geometric volumes of past sheets, especially when the sheets are out of alignment with the screen plane, causing colors to overlap in the wrong order at the peripheries of footprints for some voxels, and the result is inconsistent and/or incorrect color displayed on the image.

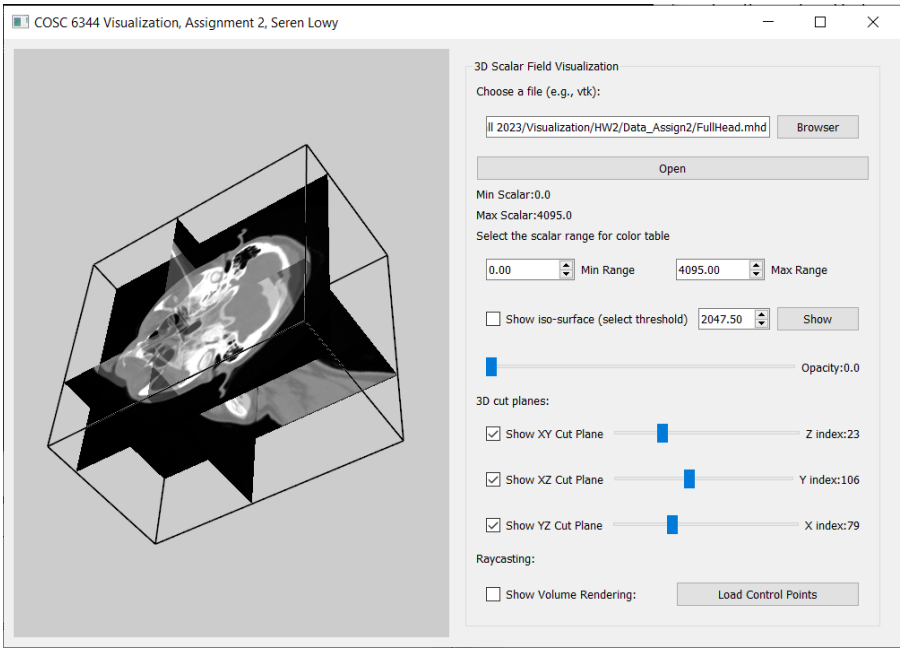
Color popping is when colors and lighting suddenly change when a volume is viewed from minutely different angles. This happens because of a conditional choice in the splatting algorithm which constructs sheets of voxels for compositing based on the relative orientation of the voxels and the image plane. When the screen orientation changes, causing changes with respect to which voxels get grouped into the same sheet and which ones are separated into different sheets, the algorithm renders pixels all over a large area of the volume (such as a planar face) with a very different compositing result all at once. In a moving visualization where an object is rotating, this occurs as a flash of light disappearing from one area of the volume and appearing suddenly on a different area.

A solution to both of the artifact problems is an image-aligned sheet buffer. This is an additional computation step where there is an intermediate layer of color values that collects color contributions from parts of voxel footprints, cutting along a plane which is adaptively aligned with the image plane. This helps to ensure that every pixel gets composited in the right order of voxels.

Raycasting with alpha compositing and voxelized ray templates is faster than any kind of splatting, but it produces good quality images, so I would recommend using it over any other method.

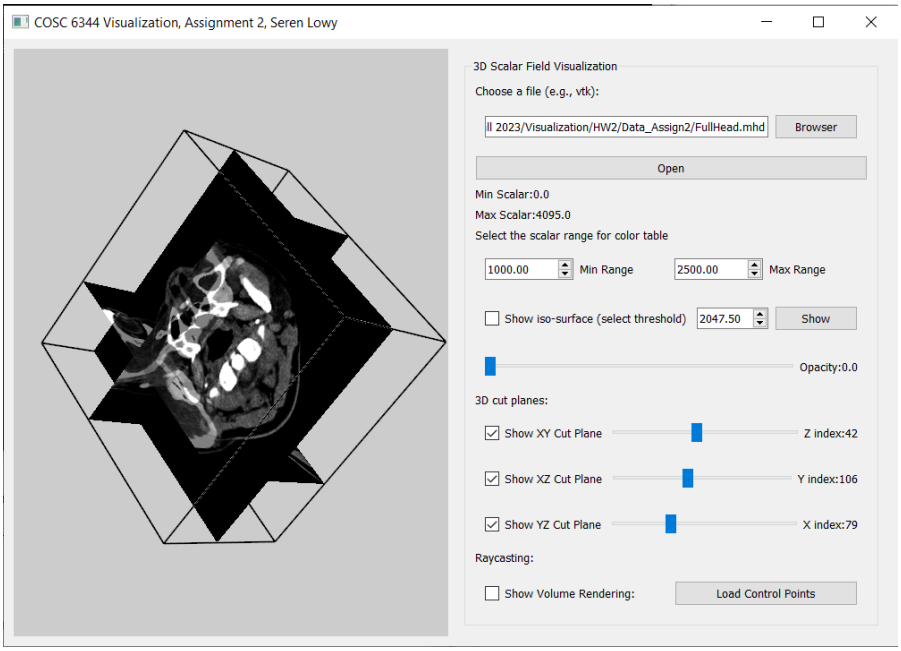
Source: lectures

2. Cut Planes



Visualization HW2
Seren Lowy – 2023/10/09

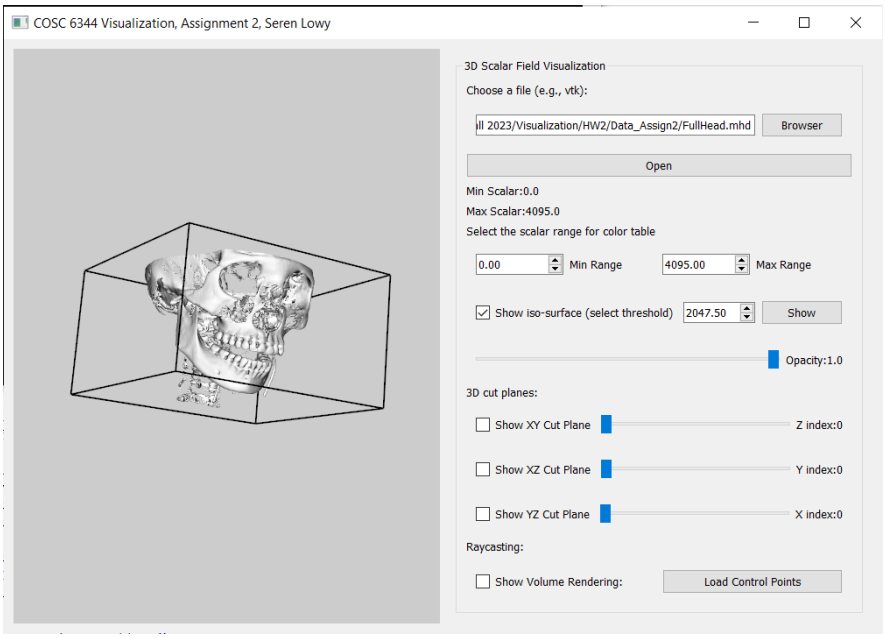
Above: screenshot of program window, showing 3 cut planes (YZ, XZ, XY), dataset “FullHead.mhd”, default values for dynamic scalar range.



Screenshot of cut planes with narrower dynamic range. Some bone structures are more visible.

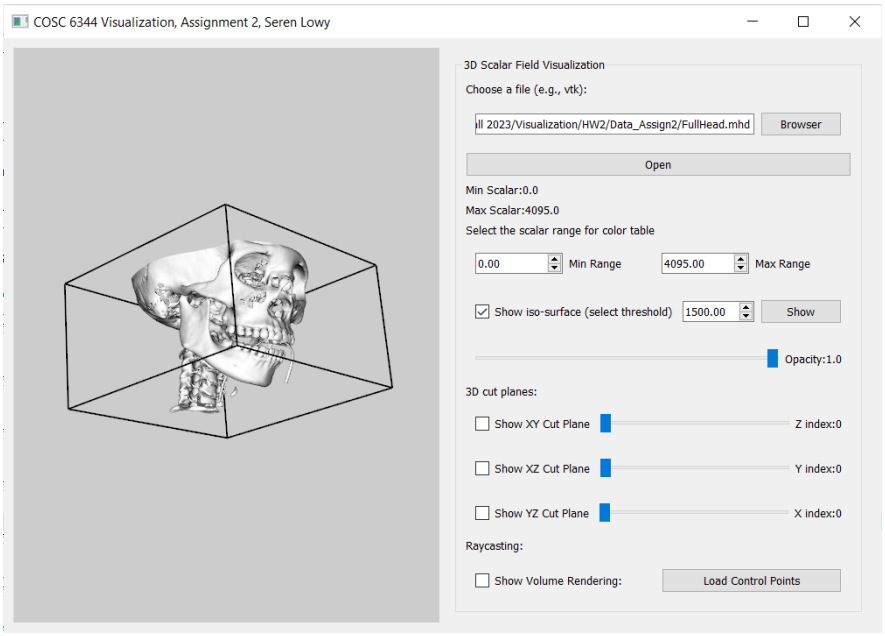
Please test my program UI to verify that checkboxes correctly show and hide the cut planes, Min Range and Max Range spinboxes interactively modify the colors visible in the cut planes, and Min range and Max range are not allowed to be in decreasing order (min > max, out of order).

3. Iso-Surface Visualization

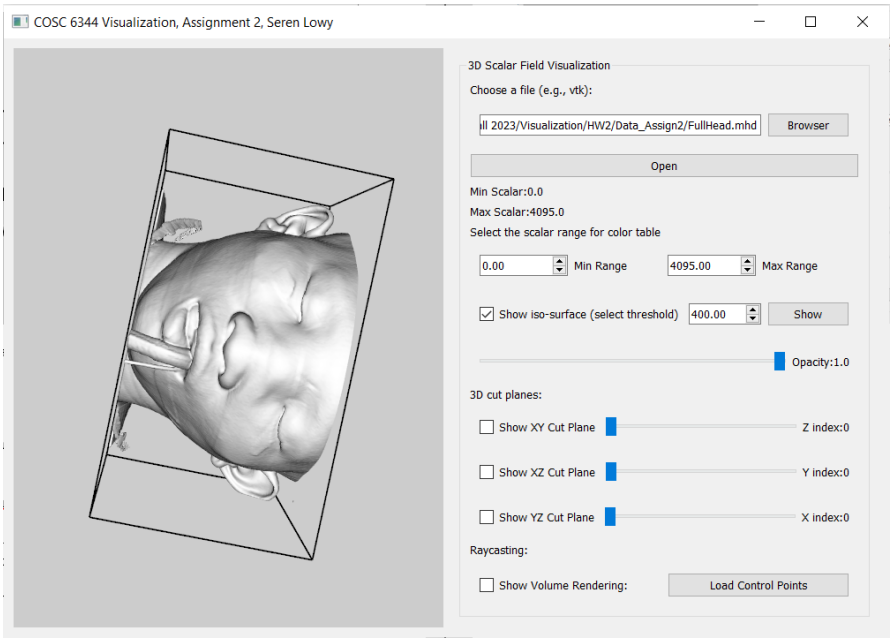


Visualization HW2
Seren Lowy – 2023/10/09

Above: The program window with an iso-surface of dataset “FullHead.mhd” at the default value (half the scalar range, 2047.50) and full opacity. The result looks like a skeleton skull face.

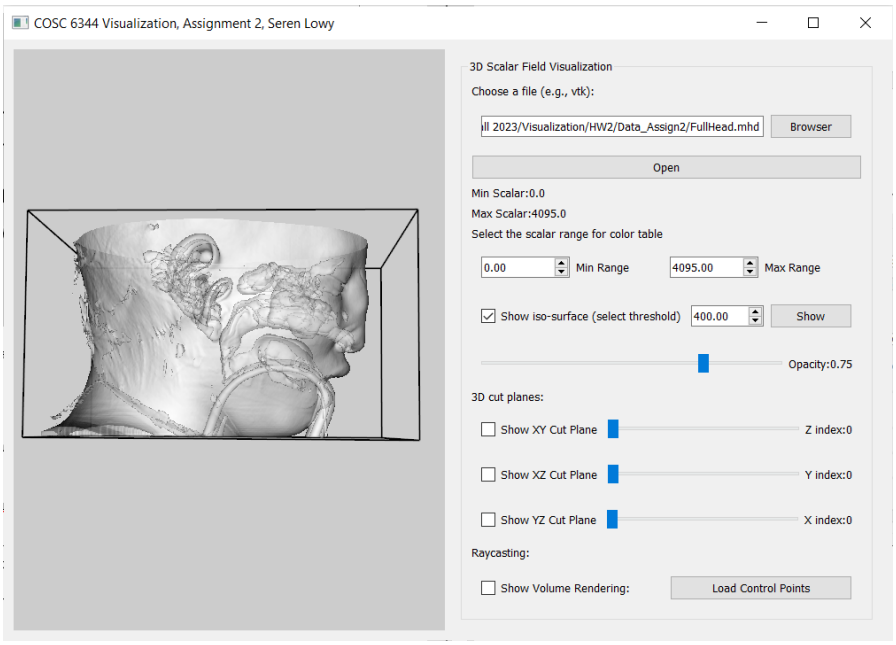


The iso-surface of 1500.00 value and full opacity, which looks like a skull with less of the bone worn out.



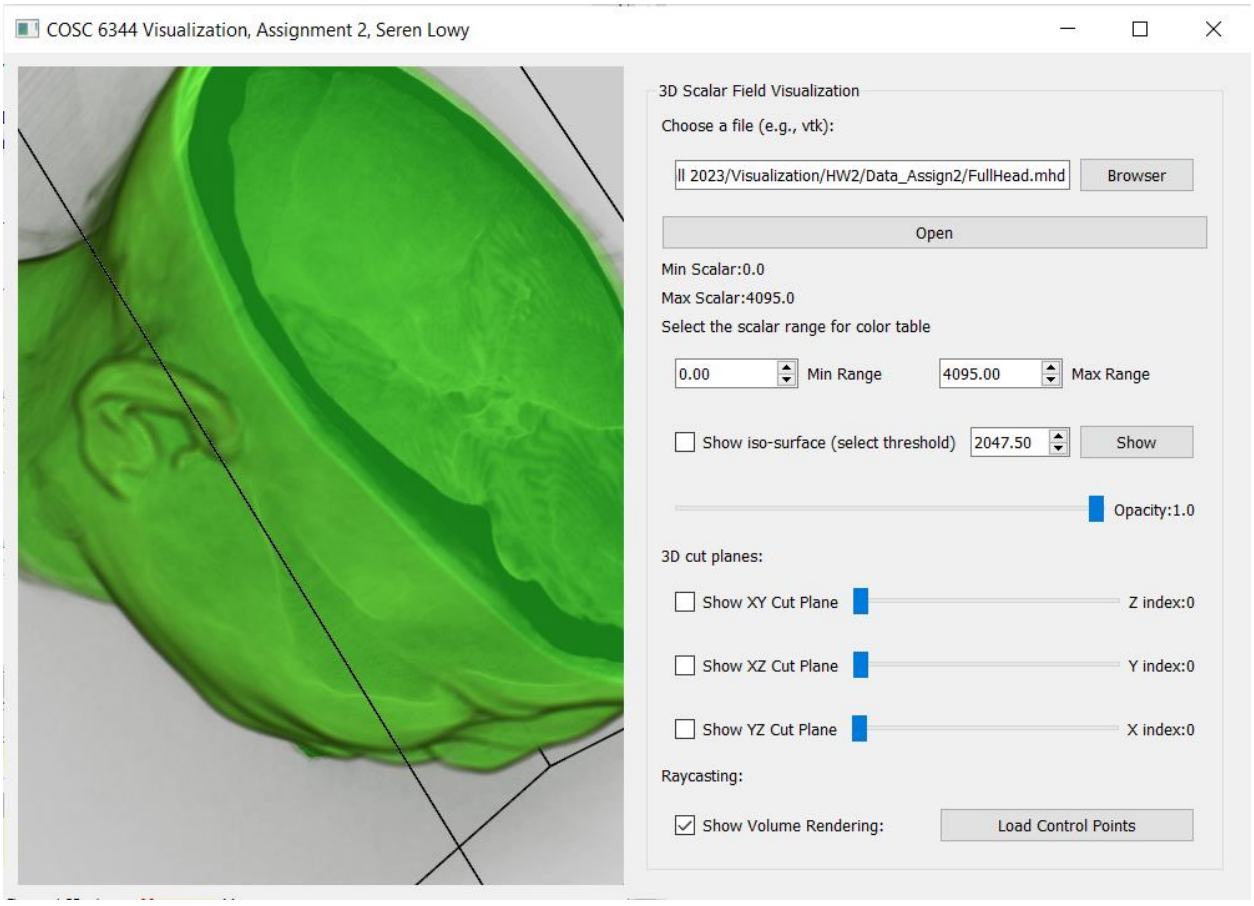
The iso-surface of value 400.00 and full opacity, which looks like an unconscious face of a hospital patient with a breathing tube.

Visualization HW2
Seren Lowy – 2023/10/09



When opacity is changed to 0.75, we can see the sinus system and inner ears inside the patient's face.

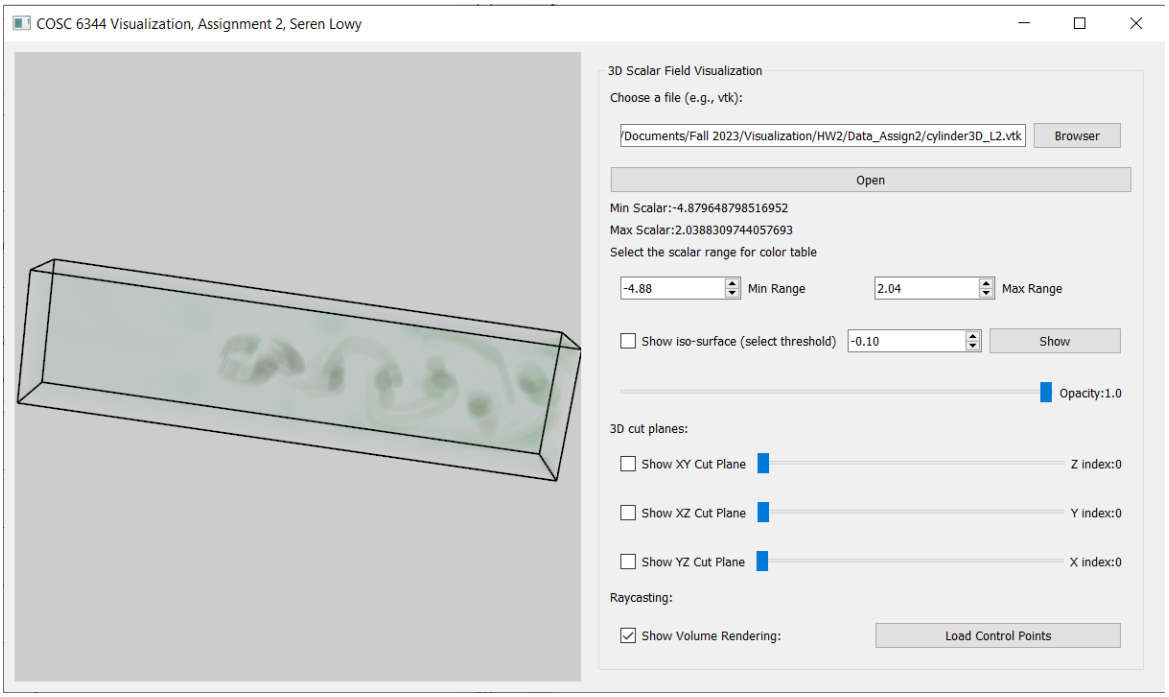
4.0 Raycast Volume Rendering



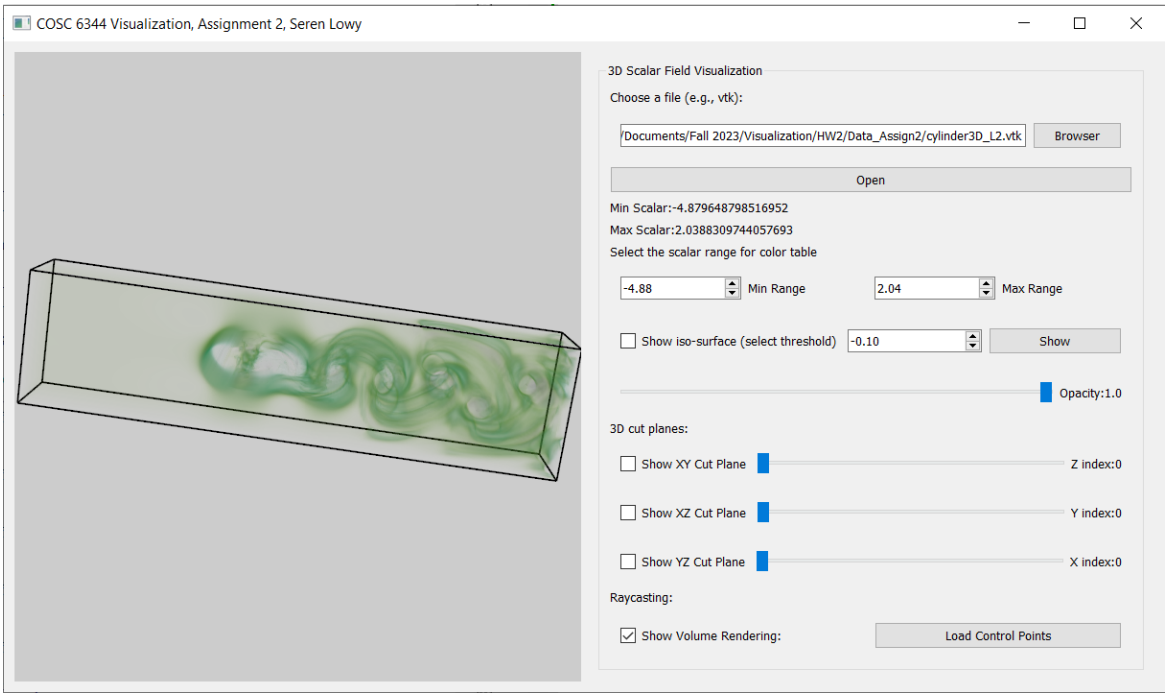
Visualization HW2
Seren Lowy – 2023/10/09

Above: Volume rendering of dataset “FullHead.mhd” in customized color. Regions of the patient’s brain are visible along with their face.

4.1 Raycast rendering, exploring Cylinder dataset



Above: Smaller tube-like structures (vortex cores)



Larger structures (flow regions surrounding the vortex cores)