Joseph Shumway
830003458

CSCE 181
Spring 2022

Transfer-Function-Independent Acceleration Structure for Volume Rendering in Virtual Reality
Balazs Faludi at University of Basel
https://www.youtube.com/watch?v=eGfX1iWzkh0

For CT Scans to render a volume using Direct Volume Rendering, they start with a large set of single channel 16 bit images in a stack that represents the slices of the 3D volume they want to represent. The next step is to use a ramp gradient to map the greyscale image to colors typical to the body. These include colors representing Air, Soft Tissue, and Bone. Once that is complete, they pass those values into a ray marching function that outputs a 3d model. With different transfer functions, there are many ways to represent the body. For example, moving the slider inwards with more transparency reveals more of the bone structure and hides soft tissue. Pulling that value out slightly shows muscle tissue. Pulling out nearly all the way allows for a view of the whole body from the outside with skin over everything.

The ray marching process begins with a ray that is cast through the bounding box of the image. Once it hits something that is marked in the ramp gradient as visible, it sets that voxel as that color and moves on until it exits the bounding box. This is done many many times until the voxel data is complete. Of course, this has its drawbacks. Wherever there is only empty space, there is still computation work being done. This is fixed with distance maps, where every voxel stores the distance to the next voxel. This allows the ray marching algorithm to use less steps for areas with low concentrations of detail. This still requires referencing the other image and recalculating values so there is a caveat.

Another alternative is to use Octrees. This is done by dividing the data set into small blocks and computing a Boolean value of whether there are visible voxels or not. From there, you compute higher levels by computing binary OR operations on each group of 4 blocks to create the new level. This is done over and over until the number of blocks reaches one. Using this data structure, you can then use ray marching to create the model. There is still more options such as Min-Max Octrees, Bitfield Transfer functions, and more.

Balazs then proceeded to compare different benchmarks and their values done using a high-end Intel CPU and AMD RX 6800XT GPU based machine. They used a 2160x2160 resolution, 6 different datasets, and 2 full rotations in 10 seconds. Quite the beefy setup for a very complex job. With different block sizes for the Octree, you can see that the result is much clearer with a smaller block size, which is expected. With higher block sizes, the first dataset's frame time increased significantly, but only as the size reached around 32^3. With the other datasets, the difference was less dramatic and scaled more linearly between each block size. As far as the performance in ms, the Bitfield Octree performed the best out of all the methods, only taking 7ms per generation. The others ranged from 7.1-7.7 ms, with the Distance Map taking as long as 54.6 ms. You could say their method is a massive improvement to the original functions. This allows much better performance in rendering the volumes, especially in VR where performance is critical to user experience.