Attribution/License

- Original Materials developed by Mike Shah, Ph.D. (www.mshah.io)
- This work was initially developed for the 2020 PetaScale Blue Waters Institute
- Funding for the development of this work came from http://shodor.org/
- This slideset and associated source code may be used freely
  - Attribution is appreciated but not necessary

Volume Rendering in CUDA

CUDA Lesson 4

Applying CUDA to Volume Rendering

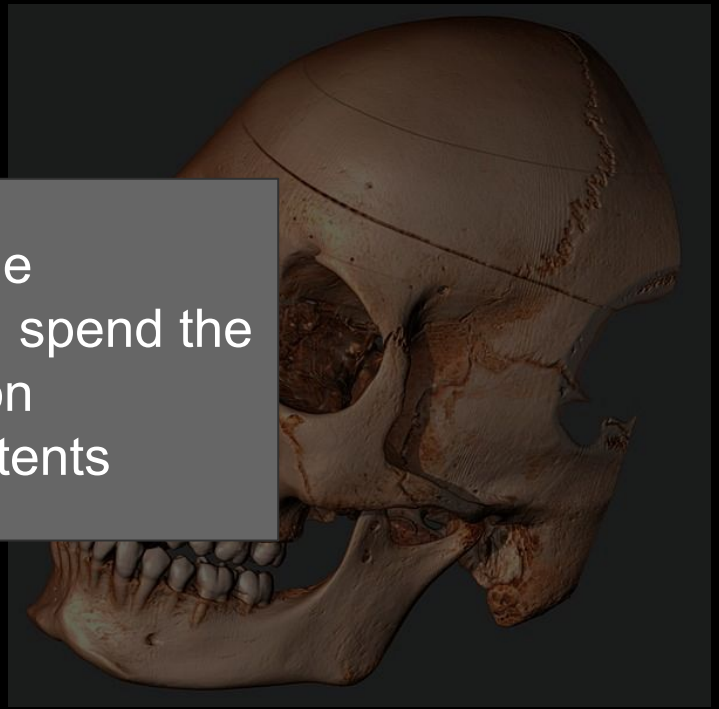
- We have previous looked at many applications of CUDA
- One popular example of using CUDA is in Volume Rendering applications
  - CUDA can be used to accelerate the rendering of 3D data so better images can be generated
  - This has applications in biomedical, geological, simulation, and gaming applications to name a few.

https://upload.wikimedia.org/wikipedia/commons/thumb/e/ee/High_Definition_Volume_Rendering.JPG/600px-High_Definition_Volume_Rendering.JPG

Applying CUDA to Volume Rendering

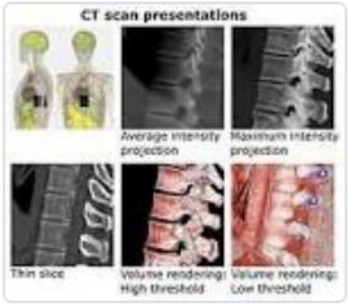
- We have previous looked at many applications of CUDA
- One popular exa… of using CUDA is in Volum… applications
  - CUDA can be us… rendering of 3D … can be generate…
  - This has applicat… geological, simulation, and gaming applications to name a few.

First let's define Volume Rendering, and we will spend the remainder of this lesson understanding the contents



https://upload.wikimedia.org/wikipedia/commons/thumb/e/ee/High_Definition_Volume_Rendering.JPG/600px-High_Definition_Volume_Rendering.JPG

Volume Rendering



In scientific visualization and computer graphics, **volume rendering** is a set of techniques used to display a 2D projection of a 3D discretely sampled data set, typically a 3D scalar field. A typical 3D data set is a group of 2D slice images acquired by a CT, MRI, or MicroCT scanner.

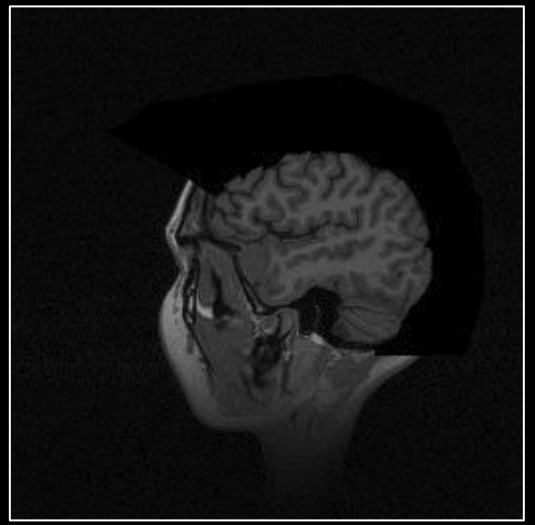
en.wikipedia.org › wiki › Volume_rendering

Volume rendering - Wikipedia

https://en.wikipedia.org/wiki/Volume_rendering#:~:text=In%20scientific%20visualization%20and%20computer,%2C%20MRI%2C%20or%20MicroCT%20scanner.

Volume Rendering in Practice

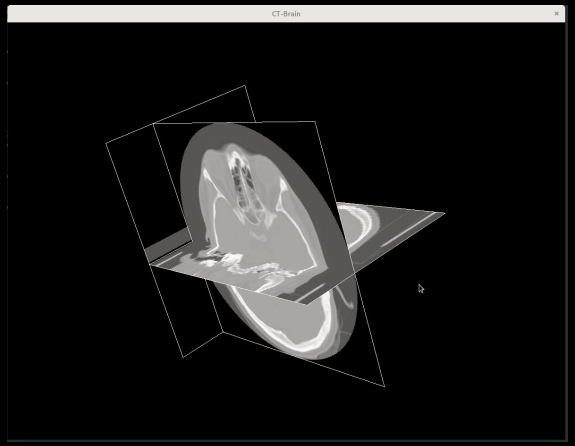
- There are many ways to implement volume rendering
- One such method is to scan a 3D model, and then take many cross-sectional ‘slices’ (saved as images) of that model at different points.
    - An example slice is shown to the right of a brain from a data set

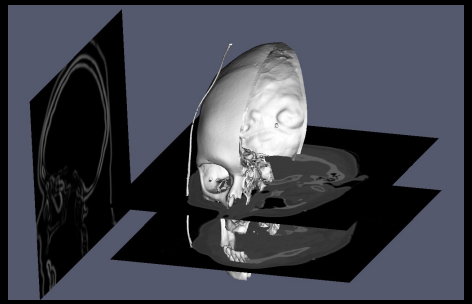


Mr. Brain data set from: https://graphics.stanford.edu/data/voldata/

Volume Rendering in Practice

- Here is another example from the popular VTK (Visualization Tool kit) from Kitware.
- Observe the idea of displaying many images as the mouse cursor moves around.
- If we ‘render’ each of these images (i.e. where there are light colored pixels and not black pixels), we can create a 3D figure like below





Cross section, notice the ‘slicing’ through the cross section to show different pieces: https://www.vtk.org/wp-content/uploads/2018/09/itkViewever_gif_brain-1.gif

2D Texture-Based Volume Rendering

- The previously described technique is known as 2D Texture-Based Volume rendering.
  - Observe again the 2D textures being arranged one after the other in thin slices.
  - With enough 3D slices closely packed together, we can ‘see’ a 3D figure.

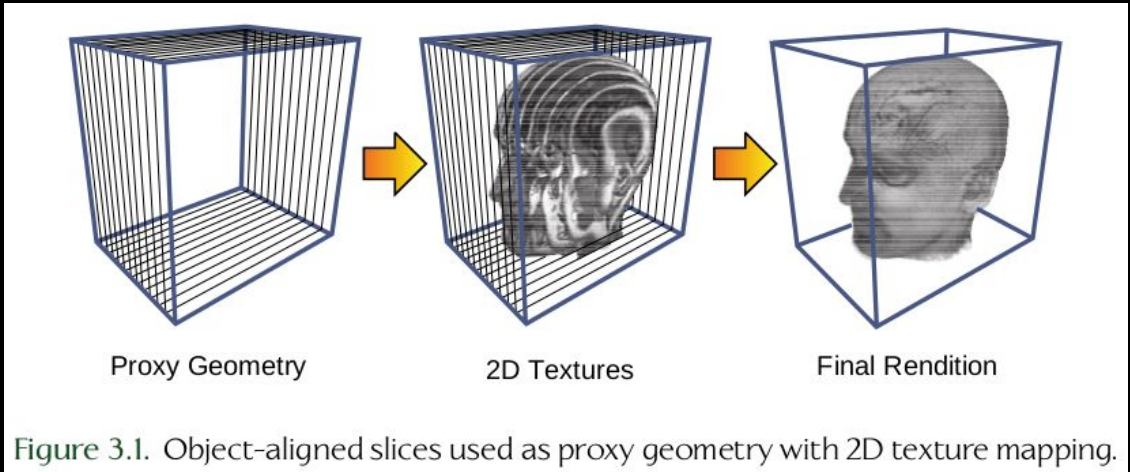


Figure 3.1. Object-aligned slices used as proxy geometry with 2D texture mapping.

Figure from ‘Real-Time Volume Graphics’

2D Texture-Based Volume Rendering

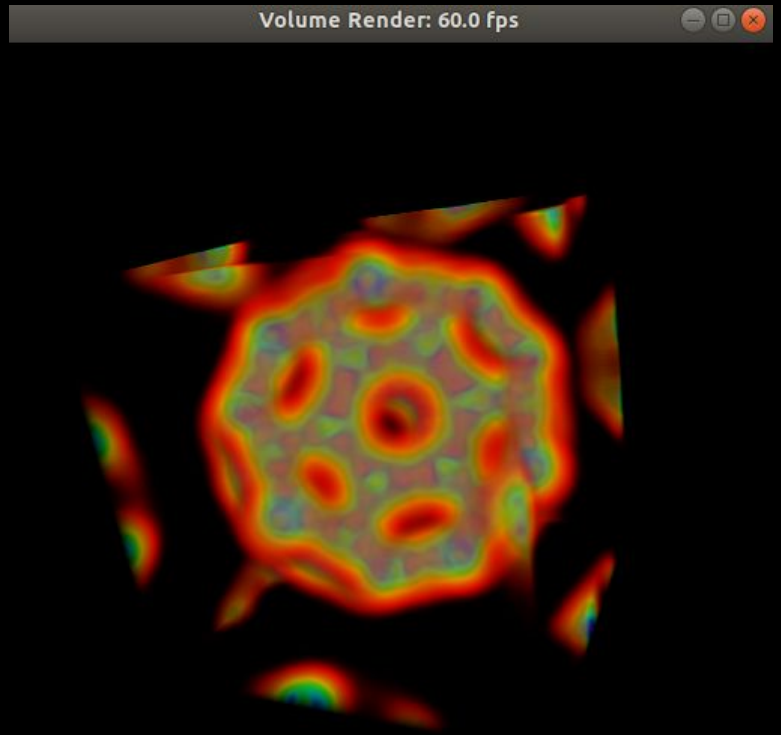
- The rendering portion of the textures can be done using quads in a library like OpenGL.
- You can use other graphics libraries like DirectX, or even other tools (e.g. processing) to draw textures on quads and send that data to the GPU.
  - (To make the volume look better, developers will use different blending and transparency techniques as well)

```
// draw slices perpendicular to x-axis
// in back-to-front order
void DrawSliceStack_NegativeX() {

   double dXPos = -1.0;
   double dXStep = 2.0/double(XDIM);

   for(int slice = 0; slice < XDIM; ++slice) {
      // select the texture image corresponding to the slice
      glBindTexture(GL_TEXTURE_2D, textureNamesStackX[slice]);

      // draw the slice polygon
      glBegin(GL_QUADS);
         glTexCoord2d(0.0, 0.0); glVertex3d(dXPos,-1.0,-1.0);
         glTexCoord2d(0.0, 1.0); glVertex3d(dXPos,-1.0, 1.0);
         glTexCoord2d(1.0, 1.0); glVertex3d(dXPos, 1.0, 1.0);
         glTexCoord2d(1.0, 0.0); glVertex3d(dXPos, 1.0,-1.0);
      glEnd();

      dXPos += dXStep;
   }
}
```

Code Listing from Chapter 3 of ‘Real-Time Volume Graphics’

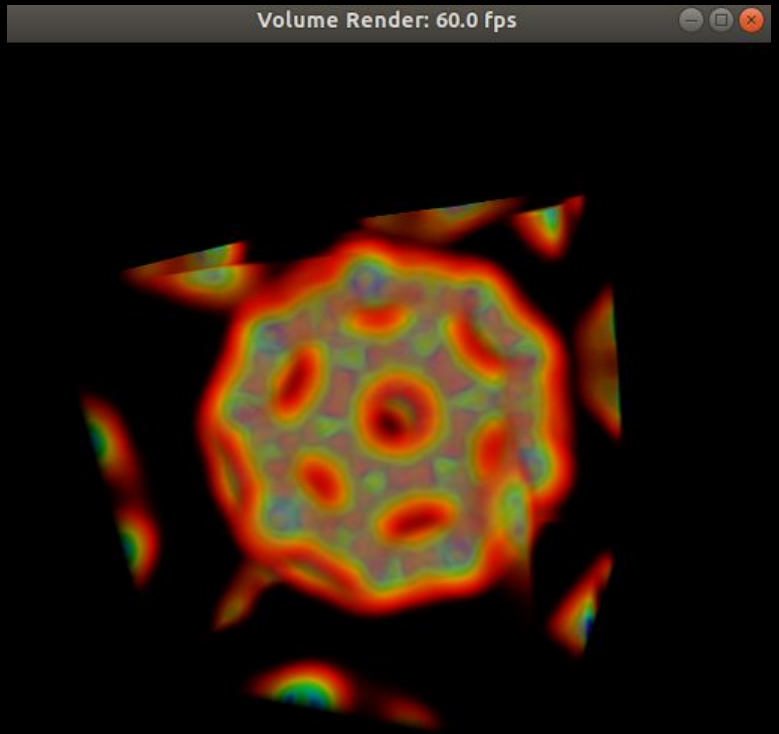
CUDA Acceleration

- CUDA threads can be used to accelerate the computation of rendering data
- For example
  - An important part of volume rendering is figuring out the ‘transfer’ function when volume rendering.
  - This is essentially the function that is used to ‘filter’, show, or otherwise alter the final 3d volume that is shown to the user.
    - The transfer function ultimately is what gives optical properties to a 3d volume (color or opacity)
      - Note the ‘blank’ spaces in the image in the right for instance



Volume Render Toolkit Example

- A sample volume renderer is available in this modules code to generate the image to the right
- It is provided by folks at NVidia at this location:
  - http://developer.download.nvidia.com/compute/DevZone/C/html_x64/Volume_Processing.html



Student Exercise

- Your exercise will be to run the example, and take some time to learn about how a simple volume render works.
- You may experiment with the transfer function to see how it affects the overall rendering
- Students should also try uploading their own image in the volume render