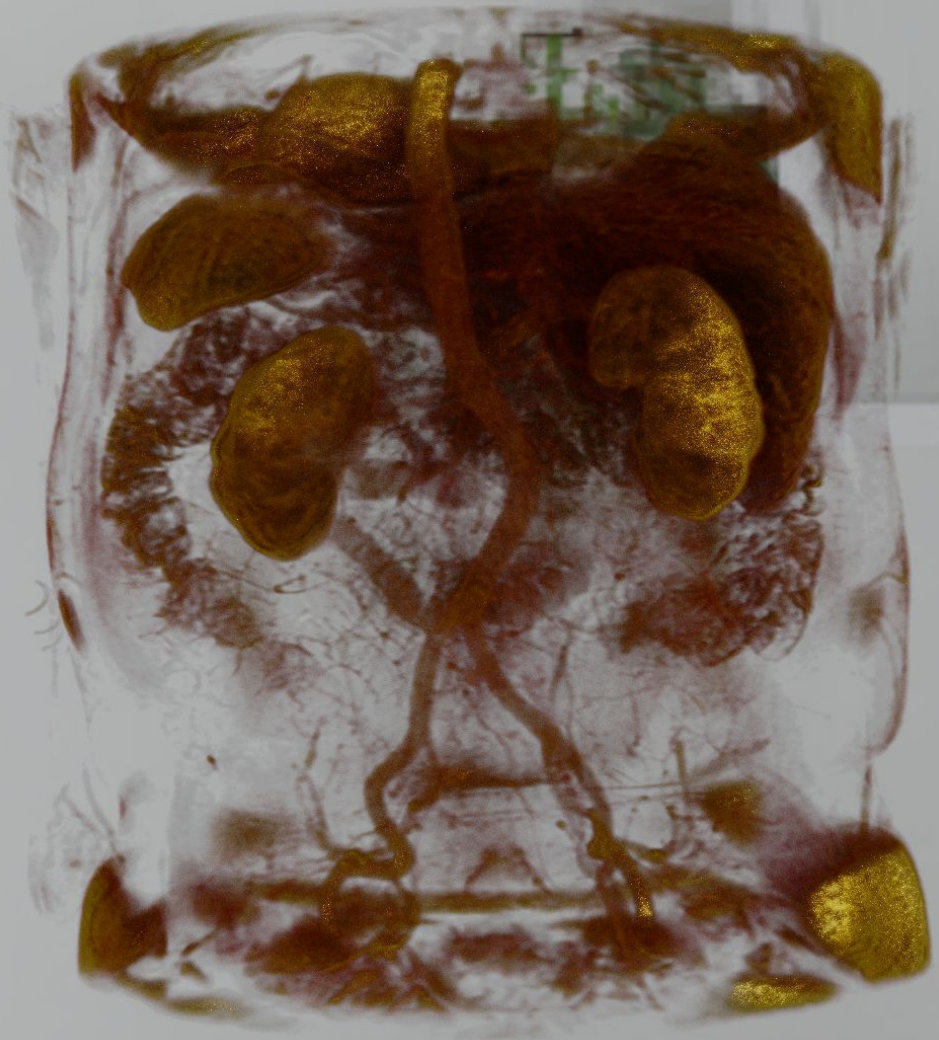
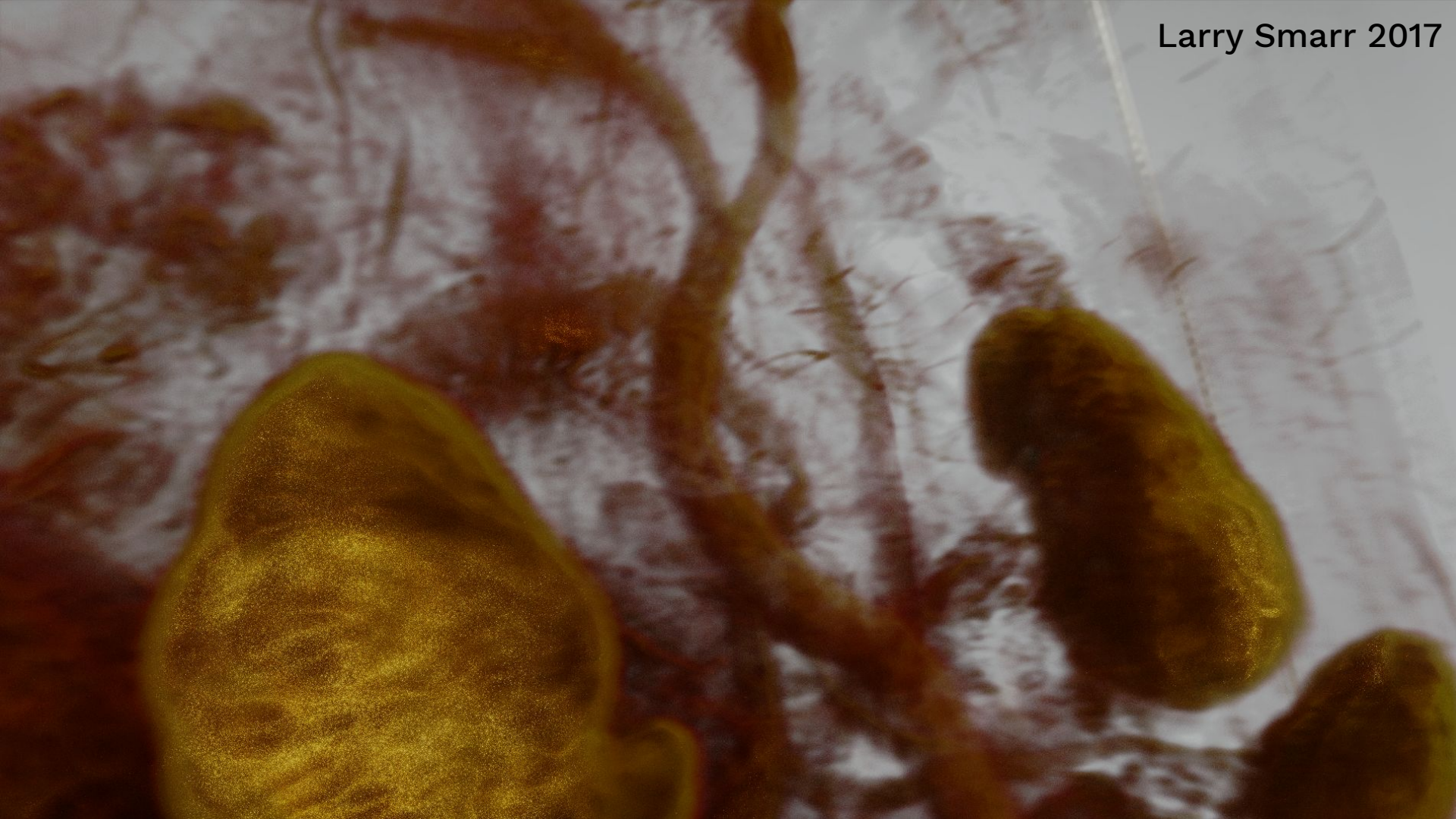
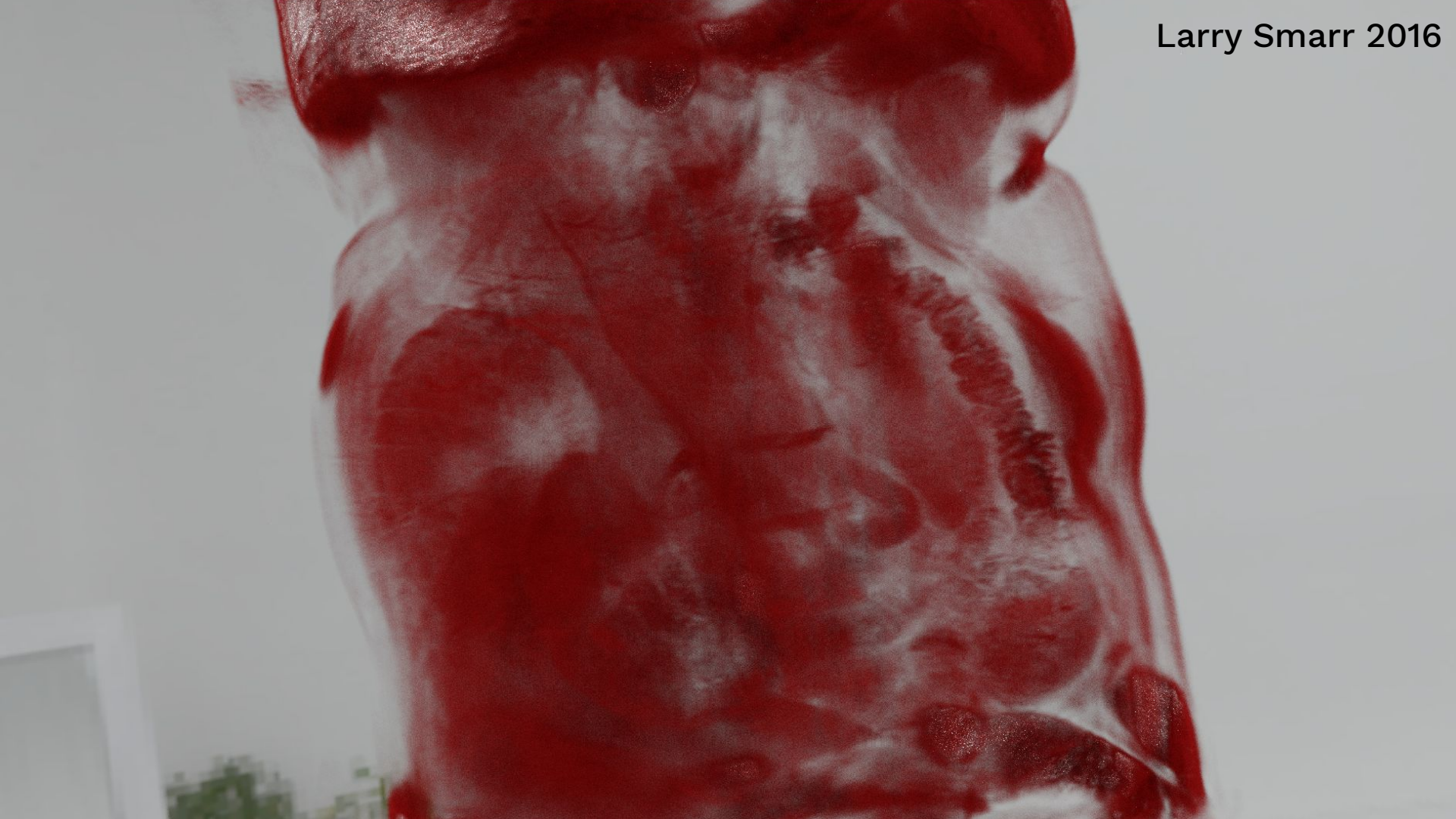
Cinematic Dicom Renderer

CSE 199 | Nicolas Nebel



Larry Smarr 2017


Larry Smarr 2017


Larry Smarr 2016


Some Low-Res lungs I found online
(banding artifacts caused by high step count)

Outline

First, it dispatches a program to generate camera rays and initialize the ray textures.

The main tracing program uses woodcock tracking to march each ray out and determine its termination point. It then determines the next bounce ray if there is a hit or shades the respective pixel with the cubemap color if there is a miss.

Lastly, a simple variance-guided denoising program is run to smooth out areas of high variance.

Features

Hybrid scattering (from *Exposure Render*), using either a volume phase function or surface BRDF based on gradient

Diffuse + clearcoat BRDF model, using the clearcoat lobe from Disney's principled BSDF and a Lambertian diffuse lobe.

Linear and constant piecewise functions for color, opacity, and clearcoat strength.

Variance-guided denoiser to improve quickly rendered images with a low sample count and HDR tone mapping using the ACES filmic tone mapping curve.

Blazing fast(~6 FPS on my GTX 960M, 640 CUDA cores, 1080p)

Merging guide

Code is located here: *https://github.com/nickwn/ivl-cr*

To merge with the current software:

- Basically copy the glsl files over and set up the pipeline to initialize the texture inputs and uniforms properly
- I set up the LUTs and cubemap in main.cpp and the raytracing textures for ray state in RaytracePass.cpp
- I set up a simple include system for my shaders so I don’t need to duplicate common code for random number generation, worst case the common code can be pasted into the files