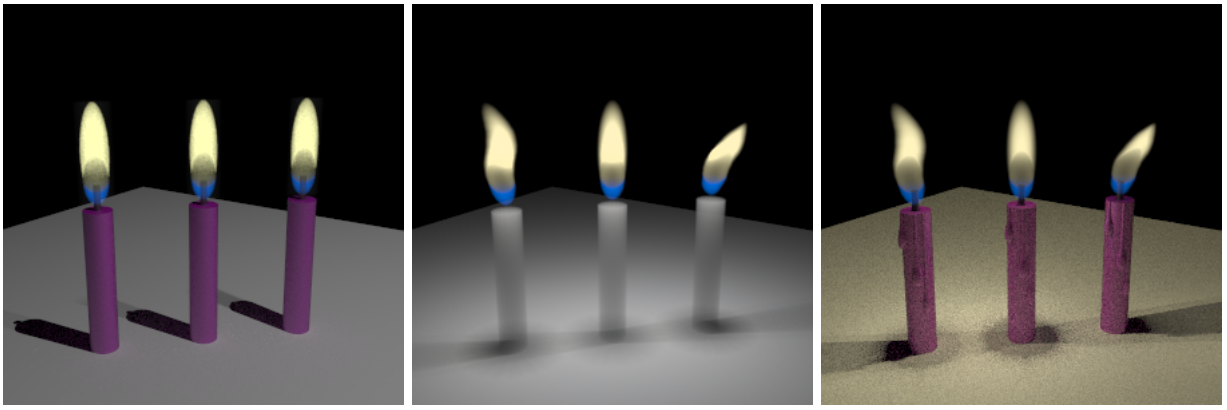
CS348B Final Project: Rendering Candles
June 12, 2013

Ben Mildenhall

For my final project, I wanted to render lit candles. To allow the scene to be entirely candlelit, I had to extend pbrt to support direct and indirect illumination created by VolumeRegions with nonzero emitted light.



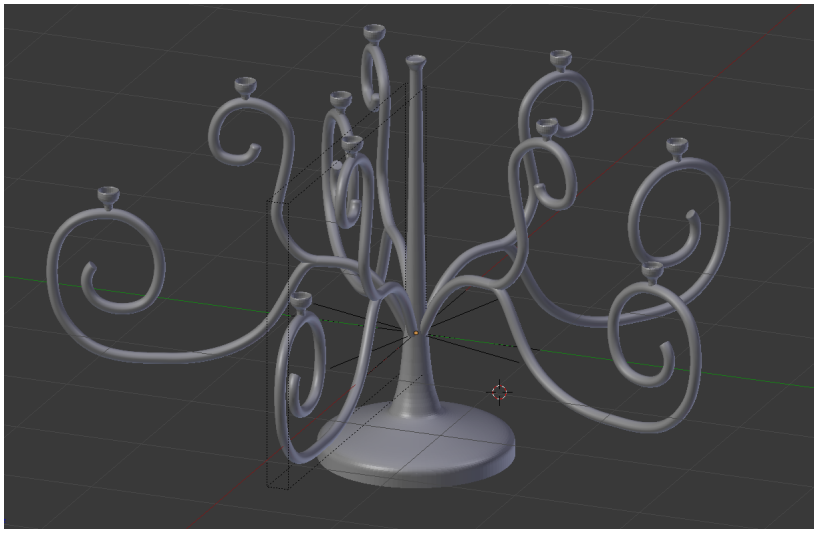
My first task was to add a new subclass of VolumeRegion called "datavolume." A datavolume could read in lists of RGB spectra (3 floats each) from files to acquire an RGB value at each point in a three dimensional grid for the attenuation constant, scattering constant, and light emitted by some region of participating media. In the case of flames, the volume has a very low albedo so scattering is negligible. I rendered all images with scattering constant 0 and used the "emission" volume integrator.

The first image on the left demonstrates the ability of a datavolume to create multicolored regions of "smoke," something that a normal DensityRegion cannot. I wrote a program to generate the flame data using nested and distorted ellipsoids of different colors. The center image demonstrates an approximation of how the scene might look with light emanating from the candle flames by using point lights centered within each flame. In addition, it approximates the subsurface scattering of wax with dense cylindrical datavolumes. (Note that the light cast is white rather than the yellow color of the flames.)

In the image on the right, all light comes from the candle flames. To achieve illumination coming from the flame datavolumes, I modified the EstimateDirect lighting function. Every time a ray is traced to compute direct illumination, the transmission and light emitted along that ray are also calculated and used to modify the value returned by EstimateDirect. (So for the PathIntegrator, we still do not get attenuation along segments in the path that pass through volumes, but this is very unlikely when the volumes are as small as the flames in the picture, particularly when we have an empty sky and background.)

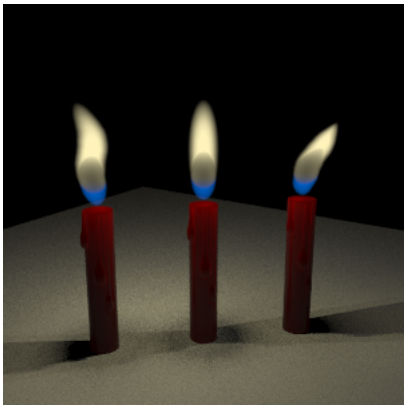
To get the direct lighting estimate to include reasonable amounts of light from the volumes, I had to somehow indicate to the renderer that the flames were important regions to sample. I created a new Light subclass called "DummyLight." DummyLight has start and end points. It always returns zero when it is sampled as a light, but it returns ω_i directions that point to some random point uniformly interpolated between its start and end points. To get the flames to light the scene, I placed DummyLights with lighting segments through the middle of each flame (so they start at the bottom of the flame and end at the top). Since most of the light emanates from the center of the flames, and the flames are rather skinny, this seemed like a reasonable approximation rather than sampling a random point from the whole three dimensional bounding box of the volume.

The third picture also demonstrates improved candle shapes. I wrote a program to generate cylindrical triangle meshes and then extrude drip-like shapes out of their sides. The drip shapes are piecewise smooth functions rotated around an axis that lies on the edge of the cylinder.



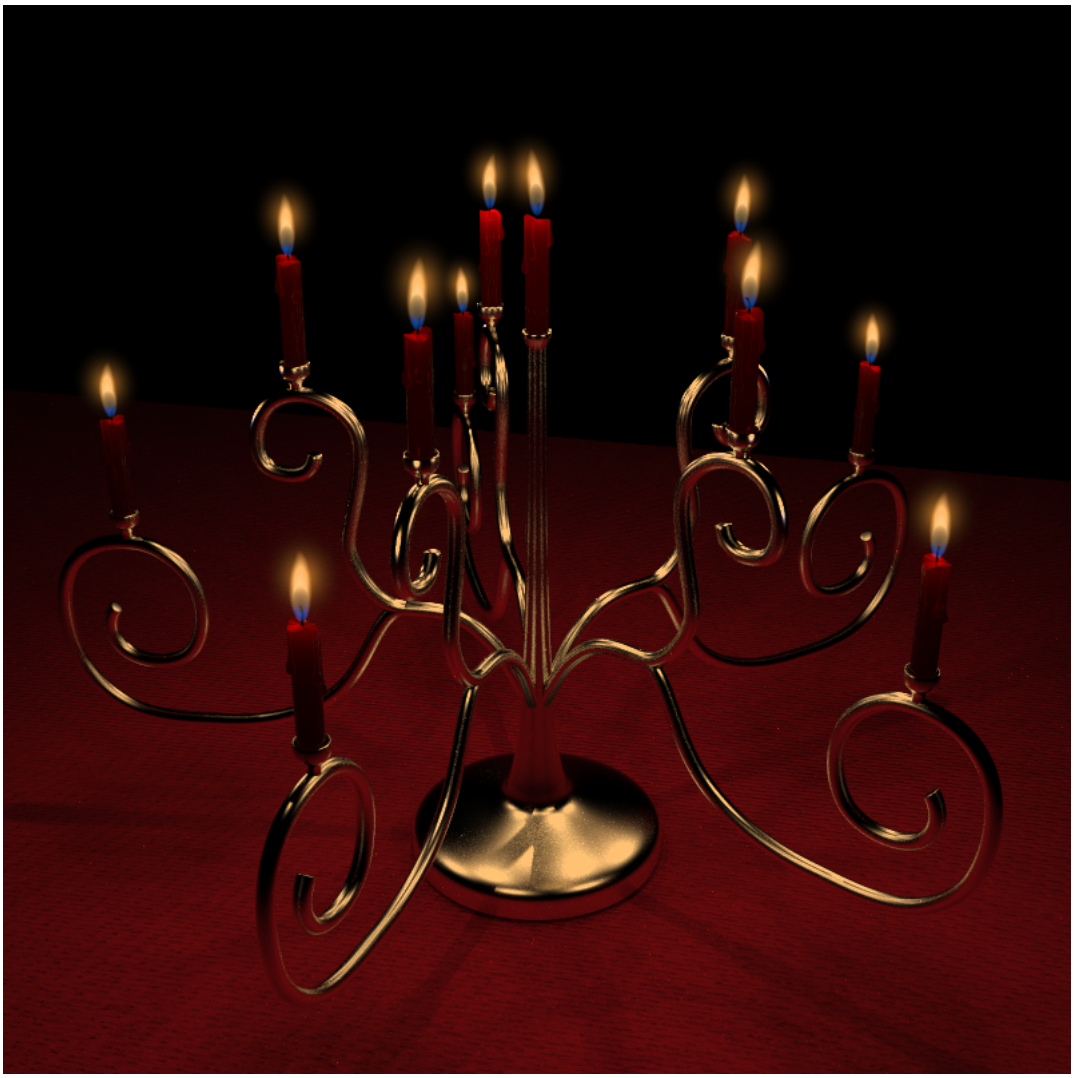

After spending some time on the candle and flame modeling, I decided to model a candelabra for the final scene. I used Blender for this. The central column of the model is a distorted cylinder, and the arms were modeled using Bezier curves. The whole mesh was subdivided three times for extra smoothness (using Catmull-Clark). The left-hand image is the mesh as displayed in Blender, and the right-hand image is the mesh when rendered using the "metal" material with the silver parameters in part.

This image is an early mockup of how the candelabra might look with my candle and flame models. All illumination comes from the candles in this scene.




My next task was to add subsurface scattering to the candle wax. I used pbrt's existing implementation of Jensen's dipole approximation. I had to add a few lines to the method computing irradiance at all the sample points on the candles that would also account for light emanating from the scene's aggregate VolumeRegion. Once that worked, I grafted together the dipole subsurface integrator and the path integrator to produce an integrator that would both approximate subsurface scattering and use path tracing for global illumination. This integrator was used to produce the image on the left.

In addition, the image on the left has a much improved flame model and a candle model with an uneven top. The top of the candle is some random section of a function of the form $k \sin(ax + by + \theta)$. I also began experimenting with different textures for the base that the candles were resting on.



For the final image, I used a cloth texture as a bump map for the diffuse red base (from http://mifti-stock.deviantart.com/art/White-Fabric-Texture-132192381). I also added a "glow" around the flames, meant to mimic the glow seen around candle flames viewed through the human eye. In this case, the glow is approximated by a fading orange ellipse centered in the flame. The flames are randomly perturbed, and the wicks also bend to follow the bottom portions of the flames. The final image was rendered using 4096 samples per pixel and is 800 pixels wide and 800 pixels tall. I had to use so many samples per pixel since I used path tracing without much importance sampling, or Metropolis sampling, so bright spots and noise were hard to eliminate. Even in the final image, a few pesky bright spots persist. Given more time, I would have tried to implement a volumetric Metropolis sampling algorithm.