Questions

1. Suppose we have a function $g(x, y, z)$ that is defined at the eight corners of a unit cube. Give a formula for interpolating these corner values to some interior value at (x, y, z).
   (This comes up in volume rendering when you sample points along a ray within a volume of data.)

2. We saw in the lecture a volume rendering method that used a light source and shading. However this method did not allow for shadows. Why is it meaningful to consider shadows in the problem of volume rendering ? The idea here is that if the volume can absorb light (and diminish intensity) from layer to layer toward the eye, then the light from the light source to the volume should also be diminished as it goes from layer to layer. Indeed the reasons that clouds have both white and grey parts is that the grey parts are shadows (not floating carbon particles!).

   How could one render shadows in volume rendering? Assume for simplicity that the light source is at infinity and the direction is different from the line of sight, say it is 45 degrees above. Also assume the camera projection is orthographic.

3. **I may set more problems. Check back.**

Answers

1. $$\begin{aligned} g(x,y,z) = {} & (1-x)\ g(0,y,z) + x\ g(1,y,z) \\ = {} & (1-x)\ (\ (1-y)\ g(0,0,z) + y\ g(0,1,z)\ ) \\ & +\quad x\ (\ (1-y)\ g(1,0,z) + y\ g(1,1,z)\ ) \\ = {} & (1-x)\ (1-y)\ (1-z)\ g(0,0,0)\ + \\ & (1-x)\ (1-y)\ z\ g(0,0,1)\ + \\ & (1-x)\ y\ (1-z)\ g(0,1,0)\ + \\ & (1-x)\ y\ z\ g(0,1,1)\ + \\ & x\ (1-y)\ (1-z)\ g(1,0,0)\ + \\ & x\ (1-y)\ z\ g(1,0,1)\ + \\ & x\ y\ (1-z)\ g(1,1,0)\ + \\ & x\ y\ z\ g(1,1,1) \end{aligned}$$

   Notice that there is one term in the sum for each of the eight corners of the cube, and the weight factor for each corner is determined by whether the parameter (x, y, z) is 0 or 1.

2. Here I will sketch out a possible approach that has been proposed in the literature. As usual, define a set of layers in the volume – the "proxy geometry". Let the orientation of (i.e. normal to) these layers be in the direction of the halfway vector H, the same H used in Blinn-Phong.

   Similar to shadow mapping, the solution consists of two passes.

   In the first pass, compute a 3D texture that represents the RGB value of the light that is reflected from each layer toward the camera. This depends on the light that reaches the point (and hence on shadows). To compute this, proceed from the front layer to the back layer and keep track of the illumination strength reaching each point. Compute the illumination that passes through that point to the next layer, based on the alpha value. As mentioned above, using the illumination that reaches the point, compute the reflected RGB value toward the viewer.

   After the first pass, for each point in the volume, we now have an RGB value for that voxel. In the second pass, composite layers from back to front, using the RGB values that were computed in the first pass. Use alpha blending to accumulate (and absorb) RGB values along lines of sight toward the viewer.

   For a few examples, see `http://www.cs.utah.edu/~jmk/simian/`