a. Test for intersections with the edges of the bounding box. Add each intersection point to a temporary vertex list. Up to six intersections are generated, so the maximum size of the list is fixed.
b. Compute the center of the proxy polygon by averaging the intersection points. Sort the polygon vertices clockwise or counterclockwise by projecting them onto the *x*-*y* plane and computing their angle around the center, with the first vertex or the *x* axis as the reference. Note that to avoid trigonometric computations, the tangent of the angle and the sign of the coordinates, combined into a single scalar value called the *pseudo-angle*, can be used for sorting the vertices (Moret and Shapiro 1991).
c. Tessellate the proxy polygon into triangles and add the resulting vertices to the output vertex array. The slice polygon can be tessellated into a triangle strip or a triangle fan using the center. Depending on the rendering algorithm, the vertices may need to be transformed back to object space during this step.

39.4.3 Rendering

Transfer Functions

The role of the transfer function is to emphasize features in the data by mapping values and other data measures to optical properties. The simplest and most widely used transfer functions are one dimensional, and they map the range of data values to color and opacity. Typically, these transfer functions are implemented with 1D texture lookup tables. When the lookup table is built, color and opacity are usually assigned separately by the transfer function. For correct rendering, the color components need to be multiplied by the opacity, because the color approximates both the emission and the absorption within a ray segment (*opacity-weighted color*)(Wittenbrink et al. 1998).

Example 39-2. The Fragment Program for 1D Transfer Function Dependent Textures

```
void main(uniform sampler3D dataTex,
          uniform sampler1D tfTex,
          float3 texCoord : TEXCOORD0,
          float4 color : COLOR)
{
  float v = tex3d(texCoord, dataTex);  // Read 3D data texture and
  color = tex1d(v, tfTex);             // transfer function texture
}
```