### 39.3.1 A Simple Example

The following example is intended as a starting point for understanding the implementation details of texture-based volume rendering. In this example, the transfer function is fixed, the data set represents the opacity directly, and the emissive color is set to constant gray. In addition, the viewing direction points along the *z* axis of the data coordinate frame; therefore, the proxy geometry consists of rectangles in the *x*-*y* plane placed uniformly along the *z* axis. The algorithm consists of the steps shown in Algorithm 39-1.

**Example 39-1. The Steps of the Simple Volume Rendering Application**

1. Create and download the data set as a 3D alpha texture.
2. Load the fragment program shown in Listing 39-1.
3. Load the modelview and projection matrices.
4. Enable alpha blending using 1 for the source fragment and (1 – source alpha) for the destination fragment.
5. Disable lighting and depth testing (there is no opaque geometry in this example).
6. Bind the data texture to texture unit 0.
7. Enable and bind the fragment program and specify its input.
8. Draw textured quads along the *z* axis. The *x*-*y* vertex coordinates are (–1, –1), (1, –1), (1, 1), (–1, 1). The corresponding *x*-*y* texture coordinates are (0, 0), (1, 0), (1, 1), (0, 1). The *z* vertex and texture coordinates increase uniformly from –1 to 1 and 0 to 1, respectively.

**Example 39-1. The Fragment Program for the Simple Volume Renderer**

```
void main(uniform float3 emissiveColor,
          uniform sampler3D dataTex,
          float3 texCoord : TEXCOORD0,
          float4 color : COLOR)
{
  float a = tex3D(texCoord, dataTex);  // Read 3D data texture
  color = a * emissiveColor;           // Multiply color by opacity
}
```

Figure 39-5 shows the image generated by the simple volume renderer. The following sections demonstrate how to make each stage of the example more general and useful