Texture-based volume rendering techniques perform the sampling and compositing steps by rendering a set of 2D geometric primitives inside the volume, as shown in Figure 39-3. Each primitive is assigned texture coordinates for sampling the volume texture. The *proxy geometry* is rasterized and blended into the frame buffer in back-to-front or front-to-back order. In the fragment shading stage, the interpolated texture coordinates are used for a data texture lookup. Next, the interpolated data values act as texture coordinates for a *dependent lookup* into the transfer function textures. Illumination techniques may modify the resulting color before it is sent to the compositing stage of the pipeline.

39.3 Texture-Based Volume Rendering

In general, as shown in Figure 39-4, texture-based volume rendering algorithms can be divided into three stages: (1) Initialize, (2) Update, and (3) Draw. The Initialize stage is usually performed only once. The Update and Draw stages are executed whenever the application receives user input—for example, when viewing or rendering parameters change.

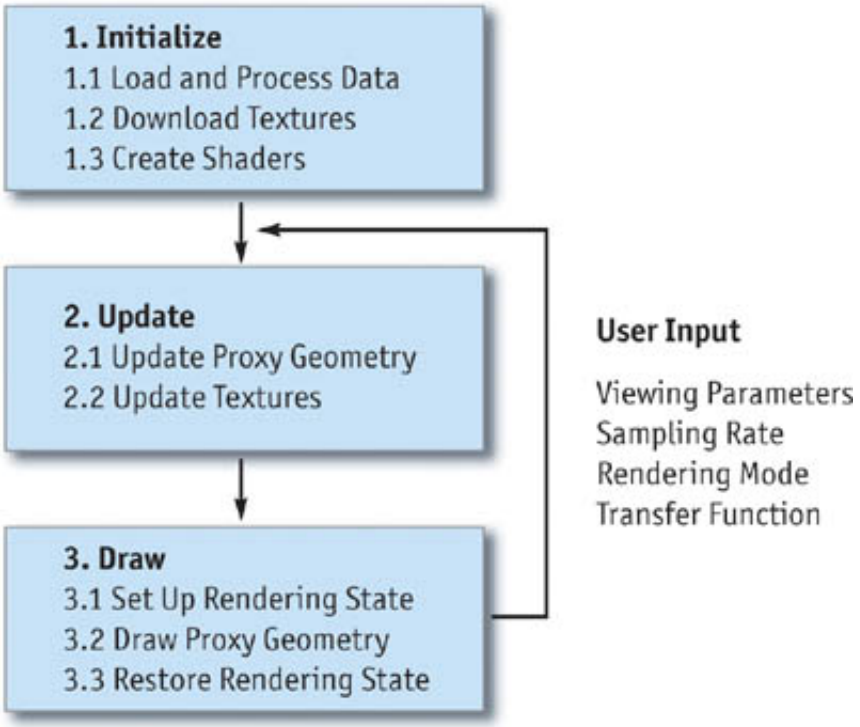

Figure 39-4 The Steps of a Typical Texture-Based Volume Rendering Implementation

At the beginning of the application, data volumes are loaded into CPU memory. In certain cases, the data sets also need to be processed before packing and downloading them to texture memory. For example, one may choose to compute gradients or down-sample the