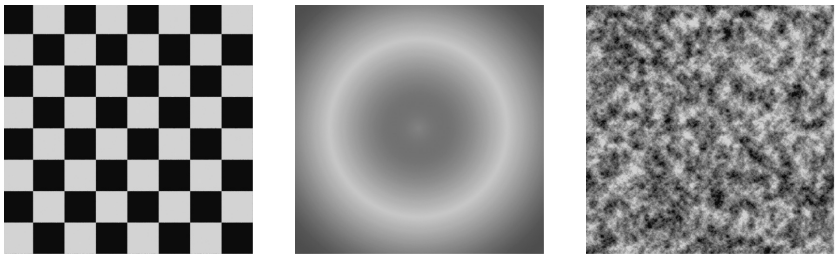


Figure 3-85. Procedures, or miniature subprograms within the software, can be used to generate a texture image.

sional texture images is **procedural mapping**, in which the software system itself creates the image by means of a collection of procedures, or subprograms (Figure 3-85). When generating a texture procedurally, you normally begin by specifying which procedure you wish to use. Your choices may range from a relatively simple procedure that calculates a grid pattern to a very sophisticated procedure that calculates a fractally based pattern of cloudlike elements. Some procedures are quite specific. A procedure might be designed, for example, to create an image that then creates the effect of water ripples when used as a bump map.

In selecting a procedure, you normally indicate values for a number of different variables, or parameters, that control the look of the procedurally generated two-dimensional image. For a grid-generating procedure, for example, you might be asked to indicate the number of horizontal bars, the number of vertical bars, the color of the bars, and the color of the background. For more complicated procedures, the number of parameters can increase dramatically. The aforementioned water-ripple procedure might have twenty or more variables that allow you to control various characteristics of the water image.

Surface texture mapping is an extremely powerful technique, as you can see by studying any of the color plates in this book, all of which make extensive use of it. In this section we have presented the fundamentals of this technique. Sections 6.5, 6.6, and 6.7 present more advanced details on this same rich topic.

3.9 Solid Texture Mapping

All of the mapping techniques discussed in the previous section involve the mapping of a flat two-dimensional pattern onto a surface. A great many interesting effects can be created with the different varieties of this kind of texture mapping. Some rendering effects, however, do not lend themselves to this two-dimensionally oriented mapping approach, but require instead a more

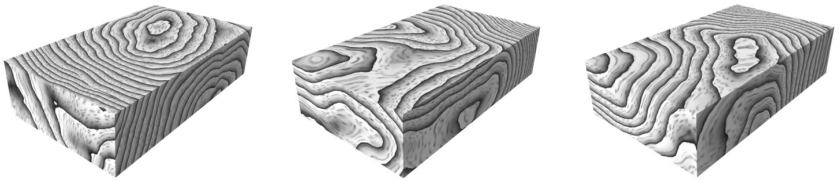


Figure 3-86. Wood-grain patterns are difficult to render effectively with two-dimensional mapping techniques.

fully three-dimensional method. A good example of this is wood, which has a grain. When you cut or carve something out of wood, the grain of the wood is visible on all the exterior surfaces of the object. The wood-grain patterns vary, however, depending on the angle of each surface relative to the direction of the grain (Figure 3-86).

Surface mapping techniques (see the previous section), which either project a pattern onto a surface or wrap a pattern across the surface of an object, do not handle this kind of effect very successfully. To continue with the example of wood, the result of parameterized mapping would look more like a block wrapped in wood-grain-patterned paper than like a block made of wood. Nor would projection mapping work any better, since it would produce a streaking effect along some of the surfaces. Neither of these approaches produces the effect of an object made *of* the material. Surface mapping techniques, whether projected or parameterized, tend to produce the effect of a pattern placed *on* the object.

The technique developed to solve this problem is known as **solid texture mapping**. The idea behind this technique is that you create a virtual *volume* of texture and then immerse your object in that volume. Depending on how the object "floats" in the volume, it picks up different colors of the volumetric texture.

The simplified volumetric texture in Figure 3-87a consists of a few wavy lines projected back (with some irregularity) through space. In Figure 3-87b a block-shaped object has been dropped into this texture volume. Figure 3-87c shows the patterns on the surface of the object that result from this "immersion." Wherever one of the wavy lines went through the block, a pat-

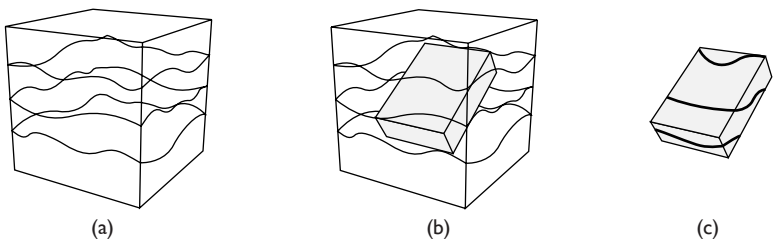


Figure 3-87. In solid texture mapping, the model is immersed in a volume of texture.

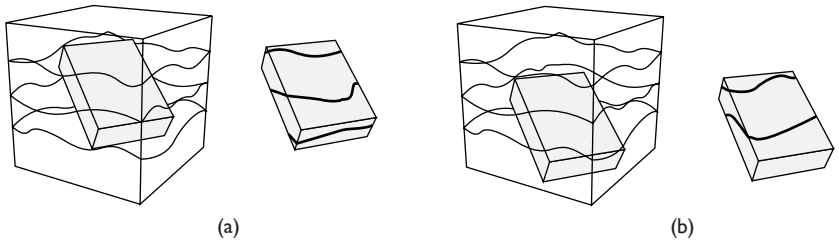


Figure 3-88. Transforming the model within the texture volume changes the patterns which appear on the surface of the model.

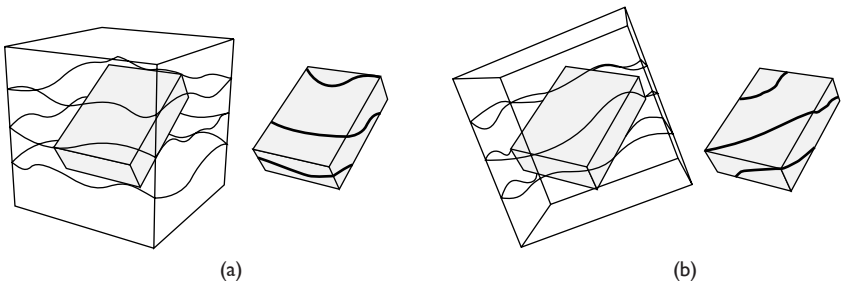


Figure 3-89. The texture volume itself can be transformed to produce a different pattern on the model.

tern is left on the block.

Since the pattern of texture is a three-dimensional volume, the color patterns that end up on the surface of the mapped object can be altered simply by transforming the object within this volume in some way. In Figure 3-88a the block of the previous illustration has been given a rotation, which causes it to pick up a different pattern of lines. In Figure 3-88b, the block has been translated downward slightly, which produces yet another pattern of lines.

Just as the object can be transformed within the texture volume to produce a different pattern of marks, the texture volume itself can be transformed to produce different marks on the object. Thus, in Figure 3-89b the block has not moved from the position it occupies in Figure 3-89a, but the entire texture volume has been rotated. This rotation causes a different pattern of lines to cross the block immersed in that volume. The different patterns on the rendered blocks are the result solely of this rotation of the texture volume.

Software packages that permit this kind of transformation of the texture volume also make it possible (usually through a menu selection) to link the transformations of the texture volume to the transformations of the object being textured. This is done by making the texture volume a hierarchical child of the object being textured (see section 2.9). Thus, if the object moves to the right, the texture volume moves exactly the same amount to the right. If the object rotates, the texture volume rotates exactly the same amount and in the same direction. This linking of transformations is done to prevent the texture pattern from moving about on the surface of the mapped object

whenever the mapped object moves. If the two sets of transformations are not linked, the pattern on the object shifts, as you saw in Figures 3-88 and 3-89. In an animation, this kind of shifting causes the patterns on the surface of an object to swirl and slide across the object as it moves about in space.

With solid texturing as with surface texturing, you can adjust parameters to control the result of the procedure. A "wood" procedure, for example, includes a number of parameters for controlling the density of the wood grain, the color of the grain, the color of the wood pulp, and so on. A procedure that simulates the effect of marble includes various parameters for controlling the veining of the marble, the colors of the marble patterns, and so on. Notice how the vein patterns seem to run "through" the urn in Figure 3-90. This effect is especially apparent around the rim of the urn.

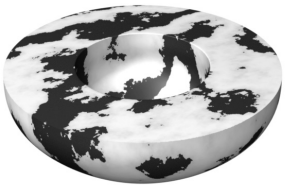


Figure 3-90. A procedurally generated solid texture that simulates marble. Notice how the texture passes through the form at the lip of the urn.

Along with using solid texturing to generate patterns of color on the surfaces of objects, you can use solid texturing in numerous ways similar to the different interpretations of surface texture mapping. For example, the volume of texture might be interpreted as a pattern of transparency rather than of color. The result of this interpretation would be to make transparent holes in the surface of the rendered object (Figure 3-91). This technique can be especially useful, for example, when trying to create a smoke effect with either volumetric fog lights (section 3.3) or particles (section 7.8). Bump mapping is another common application of solid texturing.

So far you have looked at the general principle of the solid-texturing process and the general effects of the process on an object, but you have not seen how you actually go about generating the volume of texture at the heart of the whole process. There are several ways that you normally can do this.

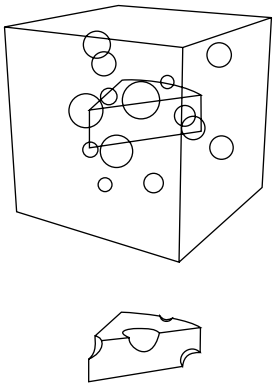


Figure 3-91. Solid texturing can also be used for transparency mapping.

One simple way to create a volume of texture is to create a flat two-dimensional image and to extrude that image straight back through space along the Z axis. In Figure 3-92a the original image is a pattern of three wavy lines. In Figure 3-92b you see this two-dimensional image pushed straight back through space to produce a cubic volume of three-dimensional pattern.

Notice that this is the same process used by the planar-projection-mapping technique discussed in the previous section. In fact, planar projection mapping, as well as all the other projection mapping techniques, create simplified volumes of texture, and are therefore in a certain sense "solid." The simplified volumes created by projection mapping, however, are so regular that there is no sense of solidity or true volume when they map to objects. What you see with these techniques has very much the feel of an *image* on a *surface*.

To make the patterns of texture volume more interesting and more useful, a **noise** parameter is added to the process in order to generate randomness as the image extrudes back through space. This parameter permits you to control how regularly or irregularly the extrusion takes place. As the noise is increased, the extruding pattern fluctuates more and more as it

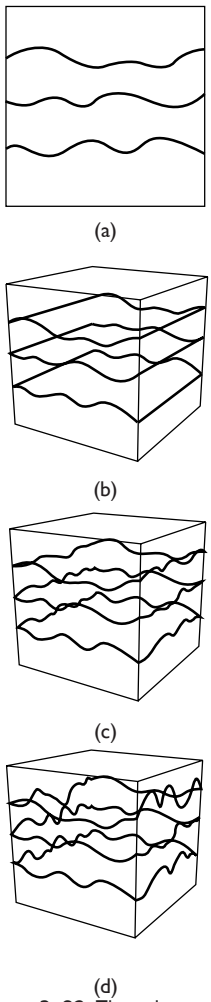


Figure 3-92. The volume of texture is generated by pushing a two-dimensional pattern through space. Randomness is added to create irregularity in the texture volume.

moves back through space. In Figure 3-92c the noise parameter has been given a low value, resulting in a slight wobbling of the image as it extrudes back along Z. In Figure 3-92d the noise parameter has been increased to produce a more pronounced wobbling and a much more irregular volume of texture.

It is this irregularity of the volume texture that gives it the appearance of solidity. Because the texture is everywhere different within the volume, no part of a textured object receives the same texture. This avoids the streaking problem of projection mapping. At the same time, because the texture maps to the object according to the object's immersion in the volume, the stretching problems of parameterized mapping are also avoided.

So far, we have been discussing solid textures created by projecting and distorting a two-dimensional image. In practice, it is more common to create your solid texture without explicitly providing a two-dimensional image. To do this, the software generates the texture volume procedurally, in much the same way that two-dimensional textures are generated procedurally. Systems that offer this kind of solid texturing usually offer within menus several specific procedures to achieve specific effects. For example, a "marble" procedure might procedurally generate a volume of marblelike texture; a "cloud" procedure might procedurally generate a volume of wispy cloudlike texture. Many of these three-dimensional solid-texturing procedures make use of noise and randomness to achieve the desired irregularity in the texture. Many of them also make use of fractal mathematics (see section 6.3) so that the irregularity and randomness of the solid texture is maintained no matter how much detail is seen in the final rendering. Figures 3-86 and 3-90 are examples of procedurally generated solid textures.

The immersion approach of solid texture mapping, discussed above, solves another problem encountered with surface mapping techniques, both parameterized and projected. With surface mapping, it can be extremely difficult to get a single texture pattern to map smoothly across the adjacent surfaces of two objects without a visible seam between the surfaces. In Figure 3-93a there is a noticeable discontinuity in the texture pattern where the top cylinder touches the bottom cylinder.

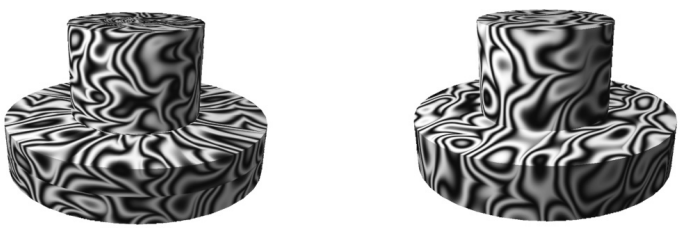


Figure 3-93. Solid texture mapping can produce continuous texture patterns across the seam between two objects.

It is possible, however, to apply a single solid texture to an entire hierarchy of objects. Since the entire hierarchical object is immersed within one texture volume, the solid-texture-mapping technique now creates a smooth, continuous texture pattern across the seam between the two cylinders (Figure 3-93b).

3.10 Final Rendering

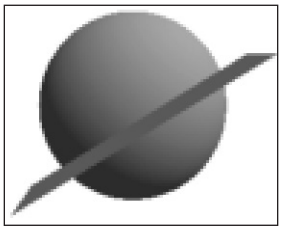
RESOLUTION

When it comes time to render the final frames of an animation, you must think about several factors that affect the look and quality of the final images. One of the most basic of these factors is **resolution**. As you have seen, any digital picture is composed of a grid of tiny rectangular picture elements, or pixels. The density of this grid is called the "resolution" and is measured as the number of pixels in the horizontal, or X, direction by the number of pixels in the vertical, or Y, direction. Thus, an image with a resolution of 300 × 200 has 300 pixels in the X direction and 200 pixels in the Y direction. In terms of columns and rows, this image has 300 columns in the X direction and 200 rows in the Y direction.

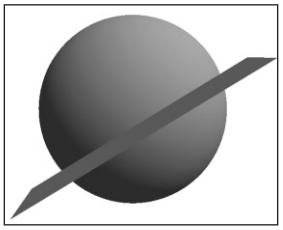
Generally, higher resolutions result in better-quality renderings. An image rendered at 300 × 200 (Figure 3-94a) looks considerably cruder than the same image rendered at 600 × 400 (Figure 3-94b), and that image looks smoother yet if rendered at 1,200 × 800. The cruder, more "pixelized" look of the lower-resolution image is especially apparent around the edges of the objects.

In some cases, however, you accomplish nothing by increasing the resolution of a rendering. If the final image is displayed very small, a low-resolution rendering may do perfectly well. An animation produced for a four-inch screen at a bank automated teller machine, for example, gains nothing by being rendered at a very high resolution. The additional detail gained by a higher resolution rendering would be imperceptible to the human eye. The same is frequently true in the game industry, where because of both screen size and the need to play the animations in real-time, rendering resolution is often deliberately kept low. On the other hand, if the image is displayed very large—for example, as a large-screen projection in a movie theater—then a very high-resolution rendering is critical. In this case, if the resolution is too low, the individual pixels become visible when the image is blown up to full size. A typical resolution for this sort of application might be a few thousand pixels in each direction.

Unfortunately, as in life, nothing in computer graphics comes free. The higher the resolution of an image, the longer the rendering takes. Remember



(a)



(b)

Figure 3-94. The resolution at which an image is rendered affects the final image quality. (For the sake of comparison, the lower-resolution image has been enlarged so that it is the same size as the higher-resolution image.)