# Solid Texturing of Complex Surfaces

*Darwyn R. Peachey*

Department of Computational Science
University of Saskatchewan
Saskatoon, Canada

*ABSTRACT*

Texturing is an effective method of simulating surface detail at relatively low cost. Traditionally, texture functions have been defined on the two-dimensional surface coordinate systems of individual surface patches. This paper introduces the notion of "solid texturing". Solid texturing uses texture functions defined throughout a region of three-dimensional space. Many nonhomogeneous materials, including wood and stone, may be more realistically rendered using solid texture functions. In addition, solid texturing can easily be applied to complex surfaces which are difficult to texture using two-dimensional texture functions. The paper gives examples of solid texture functions based on Fourier synthesis, stochastic texture models, projections of two-dimensional textures, and combinations of other solid textures.

CR Categories and Subject Descriptors: I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling; I.3.7 [**Computer Graphics**]: Three-Dimensional Graphics and Realism.

Additional Key Words and Phrases: anti-aliasing, image synthesis, shading, texturing.

## 1. Introduction

The most realistic and attractive computer generated images are usually those that contain a large amount of visual complexity and detail. Unfortunately, the amount of complexity which can be directly represented using geometric models is limited by computational considerations. Surface texturing (introduced by Catmull [4]) is an effective method of simulating surface detail at relatively low cost.

In computer graphics, objects are commonly modeled using surface representations. The surfaces of most objects are *complex surfaces* consisting of a collection of individual *surface patches*, which may abut or intersect. Each patch is a simple mathematical entity, for example, a planar polygon, a quadric surface, or a parametric patch. Texturing consists of computing the value of some shading parameter $\rho$ for specified points or areas on the patch.

Traditionally, texturing has been done by means of functions defined on a two-dimensional texture space, usually the unit square.

$$\rho = \rho(u,v)$$

A two-dimensional surface coordinate system is defined for each surface patch, along with a "natural" mapping $N$ from any point $(X,Y,Z)$ on the patch to a corresponding point $(u,v)$ in the unit square.

$$(u,v) = N(X,Y,Z)$$

$$\rho = \rho(N(X,Y,Z))$$

Originally, the shading parameter $\rho$ was used to determine the reflectance or color of a surface patch at various points. Blinn later used texture functions to control environmental reflections [1], shininess and roughness [2], and surface normal direction [3]. The latter technique is particularly significant because it permits the simulation of bumps and wrinkles on geometrically smooth surfaces. Gardner [8] used texture functions to control transparency. Cook [5] introduced the use of generalized shading expressions which allowed a different shading model for each object or surface. In Cook's system, the $(u,v)$ arguments to a texture function may be computed by arbitrary expressions, and the value produced by the texture function may be used in arbitrary expressions.

Many methods of defining texture functions have been used. Most common is the use of a stored table of texture values. The summed-area table, proposed by Crow [7], is a means of representing tabular texture data for easier anti-aliasing calculations. The data in a texture table may come from a digitized photograph, or may be generated by any of several texture synthesis techniques. Synthetic textures can also be represented procedurally, by a function that directly computes a value when called, rather than obtaining the value from a table.

In this paper, we introduce a different approach to texturing called *solid texturing*. Solid texturing is shown to have advantages for rendering objects whose surface texture arises from their internal structure. Solid texturing also provides a way of easily texturing complex surfaces. (The SIGGRAPH reviewers have indicated that the concept of solid texturing has been independently proposed and implemented by Ken Perlin of MAGI.)

## 2. Solid Texturing

### 2.1 Solid Texture Functions

A solid texture function for a shading parameter $\rho$ is simply a texture function defined at the points on a surface in terms of their 3-space coordinates rather than their surface coordinates:

$$\rho = \rho(X,Y,Z)$$

In fact, it is usually more convenient to define a solid texture function throughout a given volume of space, not just at the points on a surface. This generalization makes it unnecessary to be concerned about the shape of the surface being textured, as long as it lies within the volume where the solid texture function is defined. As with two-dimensional texture functions, it is quite easy to make a solid texture function periodic so that it is defined at all points in 3-space, though care may be required to avoid discontinuities in the function at the boundary between one period and the next.

The simplest way of applying a solid texture function is to use the three-dimensional scene coordinates of the point being rendered as the argument of the texture function. It is often more convenient to use the local coordinate system of some object, rather than the scene coordinate system, to determine the coordinates for solid texturing. Applying a three-dimensional coordinate transformation (matrix multiplication) to the coordinates before passing them to the texture function allows more flexibility. For special applications, a solid texture function may be evaluated based on another three-dimensional argument such as the surface normal vector, the direction of the observer, the direction of a light source, etc.

### 2.2 Advantages of Solid Texturing

Objects which are made by carving or machining a chunk of a nonhomogeneous material exhibit a surface texture due to variations in the internal composition of the material. Many natural materials are nonhomogeneous and have complex internal structures which result from growth, accretion, cracking, or mixing while in a liquid state. Natural materials such as plant and animal derivatives and veined or patterned minerals can be more realistically modeled using appropriate solid texture functions.

For example, objects machined from solid wood exhibit different grain textures depending on the orientation of the surface with respect to the longitudinal growth axis of the original tree. Traditional graphics techniques of applying wood grain textures typically result in a veneer or "plastic wood" effect. Figure 1 was produced by applying a simple two-dimensional simulated woodgrain texture to the surfaces of a block and a sphere. Although each surface of the block resembles wood, the relationship between the textures on the adjacent surfaces destroys the illusion. The distortion of the woodgrain texture when applied to the surface of the sphere results in unrealistic spiral curves and discontinuities. Some improvement might be produced by scaling, rotating, or pre-stretching the textures [7], but it would be difficult to give a convincing impression of solid wood without adding more textural information.

A more realistic solid wood effect may be produced by using a solid texture function that can determine which type of wood (light or dark) exists at a given point in space (Figure 2). The function used here is simply a collection of coaxial cylinders alternating between light and dark wood. The central axis of the cylinders is oriented approximately from right to left in the figure, with a slight tilt upward and away from the camera. Although this cylindrical function is an excessively regular model of the structure of wood, it results in a surprisingly realistic woodgrain texture.

Comparing Figures 1 and 2, a number of advantages of solid texturing are evident. The grain texture seen on the right end of the block in Figure 1 is realistic and is completely different from the grain on the sides of the block. The relationship between the textures on the various surfaces of the block is correct for solid wood. This is particularly clear from the way in which the textures match up at the edges between surfaces. The sphere in the lower part of Figure 2 also shows a convincing solid wood texture, without the unreasonable spirals and discontinuities seen in the sphere in Figure 1. All of these improvements were achieved without manipulating the texture function by scaling, rotating, or pre-stretching to suit the geometric models.

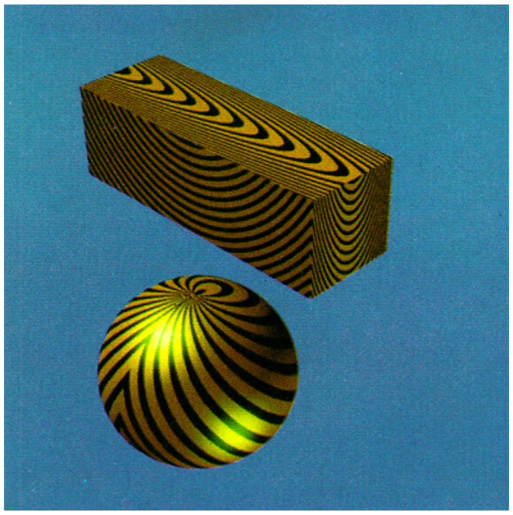
Figure 1

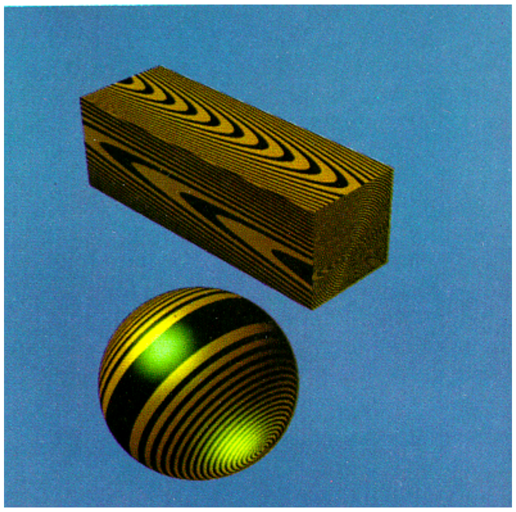
Figure 2

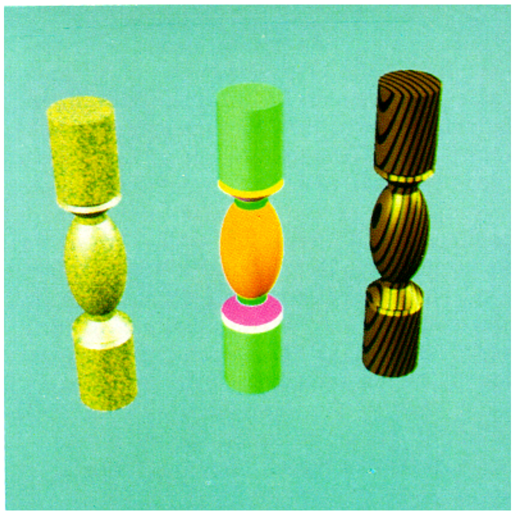
Figure 3

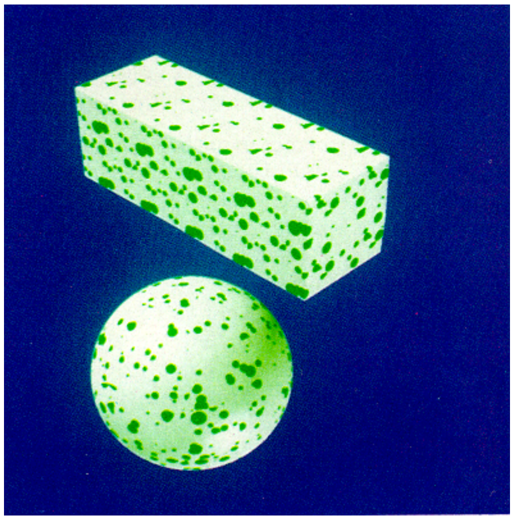
Figure 4

It should also be noted that solid texturing eliminates the aliasing problems that arise from the highly compressed surface coordinate system near the poles of a sphere or in regions of tight curvature on some parametric surfaces. The compressed surface coordinates in these cases result in a large area of the texture space appearing in a small area of the image. Since the solid texture function is defined independently of any surface coordinate system, there is no such compression of textural detail when solid texturing is used. Of course, aliasing still can be a problem when the texture is compressed by perspective, for example, when a surface is distant or is nearly parallel to the viewing direction.

## 3. Texturing Complex Surfaces

An interesting and valuable property of solid textures is their straightforward applicability to arbitrarily complex surfaces. Using traditional texturing methods, it is easy to separately texture each patch of a complex surface. However, it is much harder to map a single texture over the entire complex surface in a coherent fashion without introducing discontinuities. To texture map a complex surface, the texture space must be partitioned into regions to be applied to the various patches that make up the complex surface. The partitioning must take into account the connectivity of the patches, so that textural discontinuities do not appear at the boundaries between adjacent patches. Methods have been devised to do this partitioning for groups of patches that form a sufficiently regular mesh. For example, Crow's rendering system [6] includes a program which assigns texture space coordinates to the vertices of quadrilateral patches that approximate a surface of revolution. Each quadrilateral is mapped to a rectangular cell of a regular grid that covers the texture space. The program assigns adjacent quadrilaterals to adjacent cells of the grid. Within a single quadrilateral, bilinear interpolation is used to compute the texture coordinates of an arbitrary point from the texture coordinates of the four vertices.

Use of surface texturing becomes more awkward and *ad hoc* as the number of patches grows and as their arrangement becomes less regular. When we consider the case of patches which do not neatly meet at the edges, but which may intersect one another, the problem of mapping a texture onto the complex surface is even more difficult. In order to properly assign areas of the texture space to patches, it is necessary to determine the exact curve along which two or more patches intersect, a task which may be mathematically or computationally intractable.

Since adjacent patches may be of different types, different shapes, and different sizes, the surface coordinate systems on opposite sides of the boundary between two patches can be radically different. For example, a sharply curved patch may twist and distort a texture pattern which is undistorted on an adjacent flat patch. Even two adjacent flat patches may have greatly different scales, so that the size of textural details is much larger on one patch than on the other. Because of these surface coordinate system effects, it is very difficult to ensure apparent continuity of two-dimensional textures applied to complex surfaces.

Solid texturing can be directly applied to arbitrarily complex surfaces, without encountering the problems discussed in the preceding paragraphs. All of these problems result from the geometric relationships between patches or from the surface coordinate systems of the patches. Solid texture functions are independent of the surface geometry and the surface coordinate systems. Since every point on a complex surface has a unique position in three-dimensional space, a solid texture function can be directly evaluated at the point without any additional information.

Figure 3 shows three instances of a spindle-shaped object that might form part of a piece of furniture. The center spindle is painted in several different colors to distinguish its various surface patches. The other two spindles are rendered with solid textures that approximate granite and wood, to demonstrate the straightforward applicability of solid texturing to complex surfaces.

## 4. Generating Solid Textures

In principle, solid texture functions can be defined and evaluated in most of the ways which are popular for two-dimensional texture functions. Texture functions may be divided into digitized textures and synthetic textures. The advantage of digitized textures is their potential for realism. Digitized textures are quite popular in two-dimensional texturing, because it is relatively easy to digitize a photograph. Digitizing solid textures is far less convenient, since it involves the two-dimensional digitization of a large number of cross-sectional slices through some material. It is often preferable to use a synthetic solid texture to avoid this awkward digitization process. Synthetic textures are more flexible than digitized textures, in that synthetic textures can be designed to have certain desirable properties or to meet certain constraints; for example, a synthetic texture can often be made smoothly periodic, so that it can be used to fill an infinite texture space without visible discontinuities at the boundary between one period and the next.

While a digitized texture must be stored in a tabular form and evaluated by table lookup, a synthetic texture may be evaluated directly in procedural form, or may be pre-evaluated and stored in tabular form. Tabular storage of synthetic texture functions is a way of improving execution speed and code simplicity at the expense of storage space. Three-dimensional texture tables pose greater problems than two-dimensional tables in this regard. High-resolution texture tables for a three-dimensional texture contain very large amounts of data (for example, $512 \times 512 \times 512$ resolution requires 134 million bytes, assuming one byte per texel). Thus, synthetic texture functions that can be evaluated procedurally are often preferable to three-dimensional texture tables.

### 4.1 Solid Texture Models

Synthetic textures are generated from a mathematical texture model which may be designed to approximate some natural texture. A great variety of texture models have been successfully used in two-dimensional texturing, and it is possible to construct three-dimensional versions of many of these models.

Bombing [11] is a stochastic texture model which has proven useful in two-dimensional texturing. Bombing consists of randomly dropping bombs of various shapes, sizes, and orientations onto the texture space. An analogous procedure may be performed in a three-dimensional texture space. Figure 4 shows a synthetic "bubble" texture which might result naturally from bubbles or droplets of one substance being captured within another substance during solidification from a molten or liquid state. The texture was generated by placing

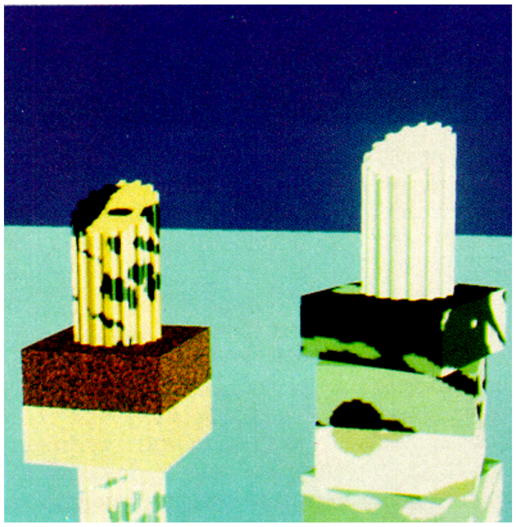

Figure 5

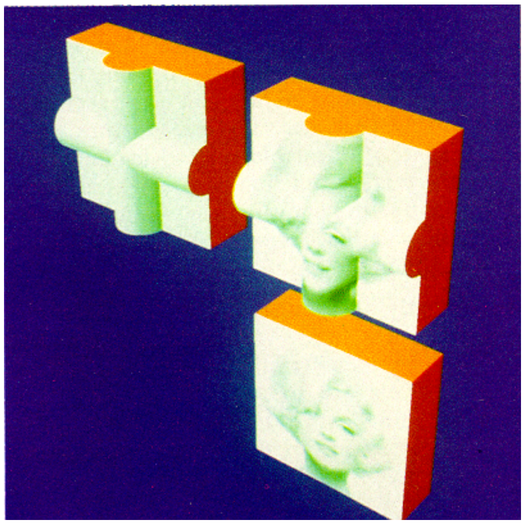

Figure 6

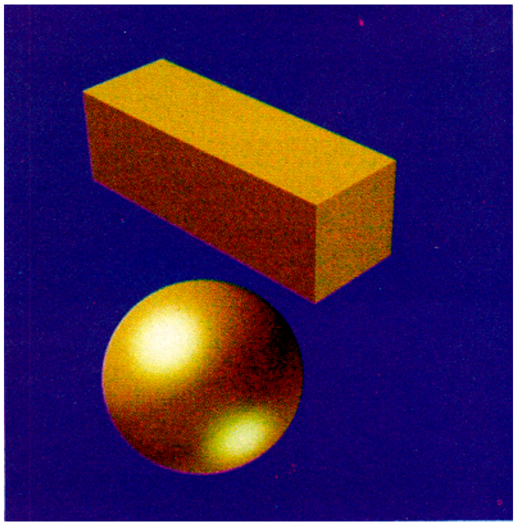

Figure 7

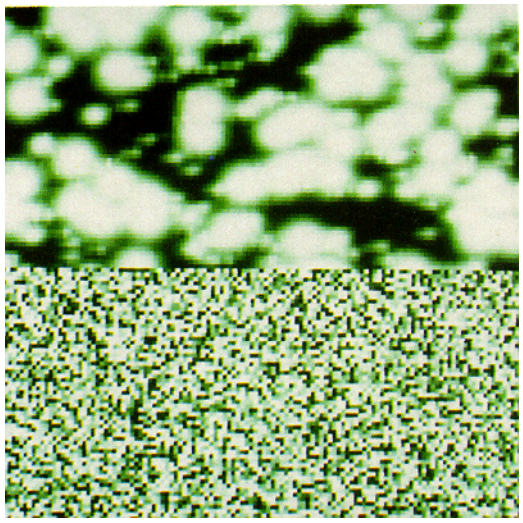

Figure 8

green spherical bubbles of randomly chosen sizes at randomly chosen locations within a 128×128×128 cube. The cube was then replicated to fill the three-dimensional texture space. The locations and sizes of the 200 bubbles were stored in a small table. To evaluate the texture function at a given point in space, the bubble table was searched to determine whether or not the given point was within at least one bubble. The same technique can be applied to bombs with a more complex shape by storing the shapes of the bombs as small three-dimensional texture tables which are used only when the input point is near one or more of the bombs.

Fourier synthesis [1][8][12] also extends to the three-dimensional case. Solid texture functions whose value consists of a sum of sinusoidal functions of $(x,y,z)$ may easily be formulated and evaluated. The "marble" solid texture function used for the column on the left and the blocks on the right in Figure 5 makes use of Fourier synthesis in a less straightforward way. The marble texture consists of dozens of "veins" which pass through the material in randomly chosen directions starting from randomly chosen locations. Each vein deviates from its basic direction according to two functions each of which is the sum of three sinusoids. Each vein has a circular cross-section whose radius is derived from another sum of sinusoids. The phases of all of these functions are determined randomly for each vein. The vein growth process is used to generate a three-dimensional texture table with a resolution of $128\times128\times128$ and one bit per texture element. During rendering, the table is used to determine the values of the texture function at the eight vertices of a cube enclosing the point being rendered. Then trilinear interpolation is applied to calculate the value of the function at the point.

Undoubtedly many other solid texture models will be developed in future to generate solid texture functions. The most fruitful sources of inspiration for these models are the well-established literature concerning synthetic two-dimensional textures, along with an understanding of the structure and causes of natural textures.

## 4.2 Projection Functions

Projection functions are a class of solid texture functions based on two-dimensional textures which are projected through three-dimensional space. For example, a two-dimensional texture $\rho(u,v)$ can be applied to a complex surface by means of the orthogonal projection function $R$:

$$R(X,Y,Z) = \rho(X,Y) \text{ for } X \text{ and } Y \in [0,1)$$

$$R(X,Y,Z) = 0, \text{ otherwise.}$$

$R$ simply projects the texture $\rho$ along the $Z$ axis. Each texture element of $\rho$ generates a rectangular parallelepiped that extends infinitely in both directions parallel to the $Z$ axis. Of course, $R$ can be rotated, translated, and scaled with respect to the object being textured, so the projection direction need not ultimately coincide with the $Z$ axis.

Figure 6 shows an example of the orthogonal projection of a digitized image, shown at the lower right, onto a complex surface consisting of polygonal and cylindrical patches, shown at the upper left. The resulting textured object is shown at the upper right of Figure 6. Note that there are no discontinuities in the mapping of the digitized image onto the complex surface, even along the intersection curves between the various surface patches.

The solid texture function used to approximate wood grain in Figures 2 and 3 is actually an orthogonal projection of a set of concentric circles. In this case the concentric circles are generated procedurally. By using a two-dimensional texture table it would be quite easy to improve the realism of the solid texture by varying the thickness and spacing of the circles and allowing them to deviate from a strictly circular shape. However, a truly convincing wood texture would require a fully three-dimensional texture table to represent knots and other features that cut across the longitudinal growth axis of the tree.

The synthetic textures used by Gardner [8] are also examples of orthogonal projection functions; they are defined on a three-dimensional scene coordinate system, but their values are determined entirely by the $X$ and $Y$ coordinates. This amounts to a projection of the texture along the $Z$ axis. (The reviewers have noted that Alan Barr presented a texture projection scheme called "decals" at the SIGGRAPH '83 State-of-the-Art in Image Synthesis Course.)

Orthogonal projection functions certainly are not the only types of projection functions that may prove useful. Cylindrical projections, where the two-dimensional texture table is indexed by, for example, the $Y$ coordinate and the angle of rotation around the $Y$ axis in the $X-Z$ plane, form another interesting class of projection functions. Other projection functions can be designed to preserve particular properties of the two-dimensional texture. Cartographers have long used the properties of conformality, equivalence, and equidistance [9] to classify projection functions for the restricted case of mapping from a spherical surface to a planar one.

## 4.3 Combination Functions

Interesting solid textures can be produced by combining other solid textures in various ways. A combination function $C(X,Y,Z)$ may be defined by

$$C = \square(A_1, A_2, \cdots, A_m)$$

where $A_i(X,Y,Z)$ are solid texture functions and $\square$ is an m-ary operation used to combine the values of the $A_i$ functions.

The "granite" texture shown in Figure 7 is a combination of three solid texture functions. Each of the three functions is an orthogonal projection of a two-dimensional texture. The texture shown in the upper part of Figure 8 is the basis of two of the projections, while the texture shown in the lower part of Figure 8 is used for the third projection. The three projections are combined by eight bit unsigned addition with wraparound. Even though each of the three projections is limited to two dimensions of textural richness, the combination function is a true three-dimensional texture. (The granite texture is also used in Figures 3 and 5.)

## 5. Texturing Costs

Although solid texturing may intuitively seem like a costly technique, it can be quite inexpensive relative to other parts of the rendering process. The coordinates used as arguments to the solid texture functions are usually easily available as a by-product of other calculations (although a matrix multiplication may be required to transform the coordinates to the texture space). The cost of solid texturing depends primarily on the cost of evaluating the solid texture functions themselves. This cost varies widely with the nature of the function.

Figures 1 through 7 were produced by a ray-tracing system called PORTRAY [10]. Figure 1, which uses a single two-dimensional texture table, took 18% less CPU time to generate than Figure 2, which uses a procedural solid texture function representing concentric cylinders. If the time needed to synthesize the texture table for Figure 1 is included, the difference drops to 12%. The solid texture function makes heavy use of a square root library function, and could probably be made much faster. Figure 6, which uses a projection function to map a digitized image onto a complex surface, was produced in less than 50% of the CPU time used by either of Figures 1 and 2, suggesting that other aspects of the rendering process have more impact on total cost than does the texturing method. Little can be said in general based on these cost comparisons, except that solid texturing need not be prohibitively expensive.

If solid texturing is likely to be much more costly than two-dimensional texturing in a given case, it is possible to use the solid texture itself to produce two-dimensional texture tables. Patch texture functions can be generated for each patch of a given object by evaluating the solid texture function at selected points on the patch and storing them in a two-dimensional table (possibly a summed area table [7]). The solid texture must be sampled at an adequate frequency; it may be necessary to sample at a high resolution and filter the samples with a weighted averaging technique to obtain a lower resolution anti-aliased texture.

Note that the generation of the patch texture function must be done separately for each patch, and requires that the position and orientation of the patch with respect to the entire object (and with respect to the solid texture function) be known. It is not sufficient to perform this process once to produce a two-dimensional texture for use on several patches or objects. The texture is inherently three-dimensional; the patch texture function is merely a computational shorthand used for greater efficiency in rendering a particular patch of a particular object.

Generating patch texture functions from the solid texture may be desirable and cost effective if the same object model is to be reused for many images (e.g., a sequence of frames for animation). It may not be cost effective if only a single image is produced, because the solid texture function will probably be evaluated at points that are not visible in the image. Patch texture functions may be generated for convenience rather than efficiency, because the patch functions can be used with existing graphics systems without modifications to support solid texturing directly.

## 6. Conclusions

Solid texturing extends the notion of texturing from a surface coordinate basis to a solid coordinate basis. Solid textures are particularly suited to realistic texturing of objects which have been machined or carved from natural materials. Solid textures are also convenient for texturing complex surfaces without producing unpleasant discontinuities at the boundaries between surface patches. Mapping a two-dimensional texture onto an arbitarily complex surface by normal means is very difficult and requires a great deal of geometric information and geometric computation. Solid texturing of complex surfaces uses no geometric information directly, and leaves all geometric computations to other parts of the rendering system. This makes the solid texturing approach a simple, flexible, and powerful means of texturing complex surfaces.

Solid textures can be generated by three-dimensional functions of various types, many of which have two-dimensional analogs. Solid textures also can be generated from two-dimensional textures as a means of applying such textures to complex surfaces. If desired, two-dimensional texture functions for individual patches may be automatically produced from a solid texture function, and applied to surface patches in the traditional way.

Clearly the notion of solid texturing can be explored further. We have only begun the investigation of interesting three-dimensional functions and ways in which they may be used to achieve particular effects.

Solid texturing is not intended to replace traditional surface texturing, but rather to provide an additional technique in the image synthesis toolkit. It can achieve effects which are difficult or impractical to achieve directly using two-dimensional texture functions. However, solid texturing may be used to generate two-dimensional texture functions which achieve such effects.

## Acknowledgements

This research was supported by the Natural Sciences and Engineering Research Council of Canada through infra-structure grant no. A2527. Larry Custead provided assistance in photographing images, while Judy Peachey and Derek Andrew offered valuable encouragement during the writing of the paper. The work could not have been done without the support and facilities of the Department of Computational Science and the University of Saskatchewan.

**References**

[1] BLINN, J. F. and NEWELL, M. E. Texture and reflection in computer generated images. *Commun. ACM 19*,10(Oct. 1976), 542-547.

[2] BLINN, J. F. Models of light reflection for computer synthesized pictures. *Comput. Gr. 11*,2(Summer 1977), 192-198.

[3] BLINN, J. F. Simulation of wrinkled surfaces. *Comput. Gr. 12*,3(Aug. 1978), 286-292.

[4] CATMULL, E. *A Subdivision Algorithm for Computer Display of Curved Surfaces.* Ph.D. dissertation, University of Utah, 1974.

[5] COOK, R. L. Shade trees. *Comput. Gr. 18*,3(July 1984), 223-231.

[6] CROW, F. C. A more flexible image generation environment. *Comput. Gr. 16*,3(July 1982), 9-18.

[7] CROW, F. C. Summed-area tables for texture mapping. *Comput. Gr. 18*,3(July 1984), 207-212.

[8] GARDNER, G. Y. Simulation of natural scenes using textured quadric surfaces. *Comput. Gr. 18*,3(July 1984), 11-20.

[9] LORD, E. A. and WILSON, C. B. *The Mathematical Description of Shape and Form.* Ellis Horwood Limited, 1984.

[10] PEACHEY, D. R. *Portray - An Image Synthesis System for Realistic Computer Graphics.* Research Report 84-18, Dept. of Comp. Science, Univ. of Saskatchewan, 1984.

[11] SCHACHTER, B. J. and AHUJA, N. Random pattern generation processes. *Comput. Gr. Image Process. 10*(1979), 95-114.

[12] SCHACHTER, B. J. Long-crested wave models. *Comput. Gr. Image Process. 12*(1980), 187-201.