Procedural Shape Synthesis on Subdivision Surfaces

LUIZ VELHO[1], KEN PERLIN[2], LEXING YING[2], HENNING BIERMANN[2],

[1]IMPA - Instituto de Matematica Pura e Aplicada
[2]Media Research Lab / NYU

Abstract. We present methods for synthesizing 3D shape features on subdivision surfaces using multiscale procedural techniques. Multiscale synthesis is a powerful approach for creating surfaces with different levels of detail. Our methods can also blend multiple example multiresolution surfaces, including procedurally-defined surfaces as well as captured models.

1 Introduction

Synthetic surface representations can be created by data capture, interactive shape modeling, or procedural synthesis. Each has advantages. Procedural synthesis, however, can also automatically generate surface details to modify an arbitrary base shape. Multiresolution procedural models add the capability to handle shapes that span a large range of scales, since they can produce more detail where needed.

This paper describes a framework to integrate procedural shape synthesis on a modeling system. We use multiresolution subdivision surfaces as a basis to do multiscale surface operations. This framework allows us to mix together various techniques of interaction, procedural synthesis and deformation. We show how these combined techniques can be used at interactive rates, locally and globally, to define surface deformations as well as to seamlessly fuse together and reconcile models with different shape and textural characteristics. A key benefit of this approach is the ability it affords to work within different levels of a multiscale representation. This provides the computational basis that allows designers to work across many levels of scale.

1.1 Previous Work

Previous work on procedural shape synthesis is closely related to texture generation.

Procedural texture generation is a powerful method for designing realistic textured image and volumes [5]. Perlin [10] showed that an expression language combined with a few primitive functions can produce high-quality textures with very little memory overhead. These techniques are beginning to appear in commodity graphics hardware. Perlin and Hoffert [11] extend these techniques to create volumetric textures; procedural shapes can then be defined as a level set or high-frequency transition within the volume. However, it is often more convenient and efficient to deal with surface shape in terms of a local parameterization, rather than as a function in 3D.

Worley [17] demonstrates a cellular texture basis function that divides space into cells in a manner similar to our use of "seed" points. Perlin and Velho [13] apply procedural textures at different levels of a multiscale domain in order to create infinitely zoomable 2D texture painting. The current work applies this multiscale notion to 3D surface deformation.

The interactive techniques for shape feature specification have many aspects in common with paint systems.

Digital paint programs (e.g. [1]) are a mainstay of 2D image generation. Multiresolution image painting supports arbitrary resolution images [12, 2]; Perlin and Velho [13] provide the ability to paint with multiscale procedural textures. Haeberli and Hanrahan [7] introduced a paint program for painting textures directly onto 3D surfaces (descendents of this algorithms are available in many commerical packages and are commonly used for feature film production). We use similar techniques to interactively define operations on surfaces.

1.2 Outline

The structure of the paper is as follows: First, we review the principles of multiresolution surfaces that we use for shape representation. Next, we show how to procedurally define multiscale shape details, such as textures that resemble rock, berries, animated tentacles, and mushrooms. Then, we discuss the various ways that multiscale details can be applied to surfaces. Features may be placed at one level or simultaneously at different scales. Finally, we show how different shape details can be combined and blended together, through a series of examples that include rust upon a metal machine part, "mummification" of a human skull, and a seamless blending of two types of planets. This last is interesting because it involves a post-processing shader; the multi-scale blending is done not as a final texture, but as one intermediate step in a sequence of shape texture operations, as will be shown in the planet example of Section 6.2.

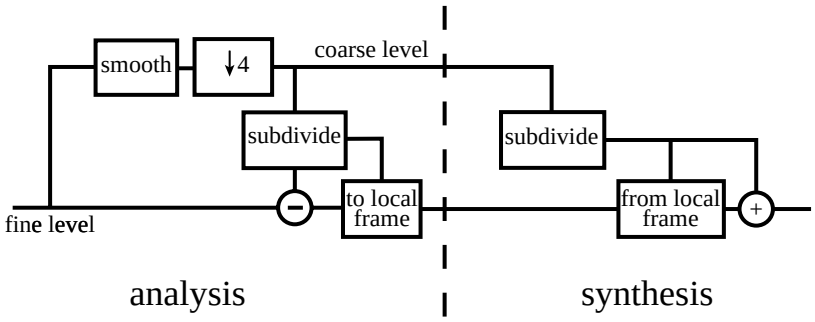


Figure 1: Synthesis and analysis diagrams for multiresolution surfaces.

2 Multiresolution Surfaces

Here we briefly review subdivision-based multiresolution surfaces; details can be found in [9, 14, 18].

Subdivision surfaces can be viewed as generalization of splines to arbitrary control meshes. Subdivision defines a smooth surface recursively, as a limit of a sequence of meshes.[1] Each finer mesh is obtained from the coarser mesh by using a set of fixed refinement rules e.g. Loop [8] or Catmull-Clark [4] subdivision rules. In our work, we use Catmull-Clark subdivision.

Multiresolution surfaces extend subdivision surfaces by introducing *details* at each level. Each time a finer mesh is computed, it is obtained by adding detail offsets to the subdivided coarser mesh. As details can be specified only at a finite number of levels, the process reduces to standard subdivision once we run out of details. The process of reconstructing a surface from the coarse mesh and details is called *synthesis* (Figure 1). The inverse process of converting the data specified on a fine resolution level to the sequence of detail sets and the coarsest level mesh is called *analysis*. For analysis, we need a way of obtaining the coarse mesh from the fine mesh. This can be done in a number of ways: simple Laplacian smoothing or Taubin's smoothing [15], quasi-interpolation or fitting. For our purposes, quasi-interpolation appears to be the most suitable approach.

An aspect of multiresolution surfaces important for modification operations is that details are represented in local frames, which are computed from the coarser level; this is analogous to representing detail surface in the frame computed from the base surface. Note also, that when we include procedural shape synthesis into this model, it is possible to generate an arbitrary amount of detail on a smooth surface.

[1] More precisely, the limit surface is the pointwise limit of a sequence of piecewise linear functions defined on the initial control mesh.

3 Computational Framework for Multiscale Synthesis

In this section we describe the computational framework behind multiscale procedural shape synthesis. We exploit the fact that subdivision surfaces make shape information available for display and for editing as a sequence of separate differently scaled level-of-detail components. This structure gives us the opportunity to mix data with procedurally generated synthetic deformation textures. The basic paradigm is to express a procedural displacement as a sum of scale-limited components. Then each component can be used to modify the equivalent level of detail of a subdivision surface.

There are two spatial domains in which the procedural deformation data can be defined: (i) In the underlying 3D Euclidean volume (as in [10]) and (ii) over a parametric coordinate system imposed within the surface manifold. We will demonstrate the ability to mix these two together in useful ways.

The computational framework (Figure 2) starts with a set of shape definitions. Each of these is either an acquired and stored shape description (e.g. a digitized skull mesh), or a synthesized shape signal (e.g. a torus).

Each shape definition takes as its domain either an (x, y, z) coordinate location, or a (u, v) parametric location on a base surface. The output of each shape definition is a set of displacement control points at each scale, The constructed shape is defined as a smooth reconstruction of the displacement control points at every scale, followed by a sum over all scales of the reconstructed signals.

These shape definitions are blended together via an alpha signal. The alpha signal may either be interactively "painted" by a user, or defined procedurally. The result of the blending is a detailed surface definition. This can be post-processed via a shader, to produce a final detailed surface definition, which is then rendered.

A animation time parameter can feed into: (i) any synthesized shape definition, (ii) the synthesized alpha, and (iii) the post-processing shader.

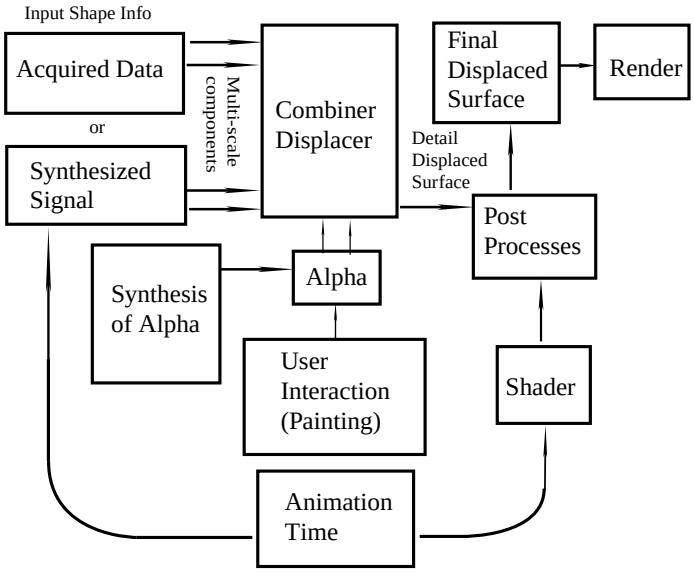


Figure 2: Diagram of computational framework.

4 Defining Multiscale Shape Detail

In this section we describe the principles of procedural generation of shape features on surfaces and give examples of procedural shape models.

4.1 Basic Principles

Our multiscale procedural shape synthesis is accomplished through the addition of geometric details at the various levels of the multiscale shape model.

For this, we design a procedural definition of the basic shape feature that we want to paste on a surface at some scale level. This procedure is a function F that synthesizes the difference between the feature at two successive scale levels. The input of the function is a point p on the surface and a scale level l. The output is a displacement d to be added as a detail at that level, $d = F(p, l)$, where p is a point given either in local intrinsic surface coordinates (u, v) or in global extrinsic coordinates (x, y, z); and d is a displacement relative to the surface at level l.

We have designed and experimented with a few shape detail procedural models. These models exploit the two basic characteristics of the procedural definition: (i) the type of coordinates and; (ii) the magnitude of displacements relative to the level.

Models based on global coordinates lead to *volumetric shape definitions*, i.e. features are taken from the three dimensional space in which the surface is embeded. Models based on local coordinates lead to *surface shape definitions*, i.e. features are "grown" on the surface. Some of our models are based on intrinsic coordinates and some on extrinsic coordinates.

The magnitude of the displacements is usually related to the scale level. Models in which displacement is inversely proportional to scale lead to *fractal-like features*. Models in which displacement is directly proportional to scale lead to *morphogenic-like features*. When the magnitude of the displacement is independent of scale, the features are essentially arbitrary. This is appropriate for man-made shapes or even physical phenomena, such as small waves. We experimented with all these kinds of displacement.

Below we present a chart relating the above classification with some procedural models that we created, and are illustrated by examples in the next section.

	Fractal	Morphogenic
Global (3D)	"rock"	"mushroom cloud"
Local (2D)	"berry"	"tentacle"

4.2 Examples

Here we show some results of using our multiscale procedural shape synthesis.

Rock is an example of a volumetric–fractal shape model. The spatial coordinates of the reference surface are used as the input of a noise function generator. The displacements are given in a $1/f$ fashion, where f is related to the scale level.

Traditionally a procedural "rock" shader is defined as a sum of Perlin Noise functions [10]. However, if one works

within a multiscale framework that c ontains a B-spline reconstruction filter at every successive scale level, it was demonstrated in [13] that it is only necessary to specify a random value at every control point.

The algorithm is then done in two successive passes, in the first pass, the surface points at each level are given a random perturbation value, based either on the (x, y, z) location or on the $(u, v, level)$ coordinates of the base surface (which acts a reference shape).

```
init_rock()
  for (level = 0 ; level < nLevels ; level++) do
    for all (u, v) on this level do
      P(u, v,level) = random()
```

In the second pass the stored displacement values are retrieved from the parametric domain:

```
Vector rock_detail(u, v, level)
  value = 0
  for (l = 0 ; l < level ; l++) do
    value += reconstruct_level(2^l u, 2^l v, l)
  return value
```

In practice we set the details for a surface point at all levels in one pass. This makes the evaluation of this procedure $O(N)$, where N is the number of mesh vertices.

The rock texture is used in some of the examples in Section 6, and shown in Figure 3.

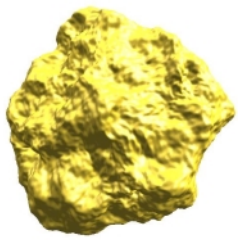


Figure 3: Fractal Rock.

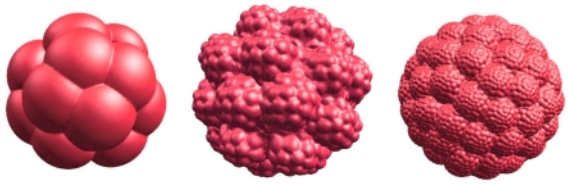
Berry is an example of a hybrid surface/volume–fractal shape model. The initial seed to the cell features is a set of points placed on the surface according to a Poisson-disk distribution. From these initial points at a base level, spherical domes are grown recursively on the surface at each level of detail.

In this example, we define a coherent procedural texture within a surface, by spreading a set of equally spaced seed points, as in [16]. This allows us to define a base level texture. Then we create each successive recursive detail level by defining a volume texture around each seed point from the previous level, to define the positions of a cluster of seed points.

We use this structure to modify control points on the surface. At every level, each control point on the surface will be closest to one seed point. We use the Euclidean distance from the control point to that seed point to weight a perturbation of the control point into the surface normal direction. In order to shape the detail into a section of a sphere (to create the bulging "berry" feature), we use the seed's radius of influence R. Given that the surface point is a distance r from the seed point, we define a perturbation into the surface normal direction of magnitude:

$$R * (1 - r^2/R^2)^{1/2}$$

The surface normal direction is redefined at each scale level, based on the perturbation that had been applied on the previous scale level. For this reason, the cluster features at each level grow outward, not from the original surface, but perpendicularly to the evolving detail surface. Figure 4 shows the construction process for the berry shape.



(a) (b) (c)

Figure 4: Berry; (a) Base domes from initial seed points – level 1 of detail; (b) first recursion added to domes – level 2 of detail; (c) final berry – 3 levels of detail.

Tentacles is an example of surface–morphogenic shape model. From initial seed points on the surface tentacles are grown outward. The direction and lenght of the displacements vary at each level. The displacements are directly proportional to scale.

Below we give pseudo-code of the shape detail procedure.

```
Vector tentacle_detail(Point2 seed, int level)
  Scalar magnitude = reference_lenght * level
  Scalar p = (PI/3) * level
  Vector displacement=(sin(θ + p), cos(θ + p), 1)
  if (is_even(level)) then
    displacement *= -1
  return displacement * magnitude
```

(Note that the intrinsic coordinates of the seed point must correspond at all levels.)

Figure 5 illustrates the growth process of the tentacle for 4 levels of detail, as the feature grows from a seed point.

Figure 5: Growth process of the tentacle. Levels 1 to 4.

The shape feature model has two parameters: *reference length* and *rotation angle* θ. These two parameters can be used for modeling purposes. The parameters can be time-varying and be used for animation

We remark that the basic structure of the tentacle shape detail model can be used as the basis to create many types of models such as the submarine explosive mine shown in Figure 6.

Figure 6: Submarine explosive mine.

Other variations of the growth model are possible. One ideia is to use L-Systems to create branching structures. For this type of model, in addition to the feature grown from the inital seed point, branching features are grown at higher levels of detail.

Mushroom Cloud is an example of hybrid surface/volume–morphogenic shape model.

The features grow from seed points on the surface, but are based on the 3D coordinates in the neighborhood of each seed point. The displacement is directly proportional to scale. Figure 7 shows an example of mushroom cloud features placed on a spherical shape.

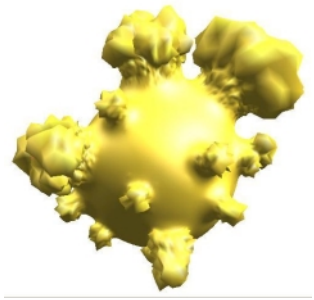


Figure 7: Mushroom planet (inspired on the planet from *The Little Prince* of Saint-Exupery).

5 Applying Multiscale Detail to the Surface

Once we have defined a repertoire of multiscale shape detail procedures, we can use them to create new shapes from base shapes. These procedures can be applied globally to a surface, as shown in the examples of the previous section.[2]

We can also apply the shape detail procedures as a local operation to construct a single feature at a given seed point of the surface. This can be a very powerful modeling tool if applied interactively. Our software implementation is fast enough to enable interactive modeling on a Pentium III 800Mhz class machine with 512Mbytes of memory and an OpenGL graphics card.

In this section, we describe some results of interactive modeling using local multiscale detail operations. We have experimented with two kinds of operations: feature placement and local shape modification.

5.1 Feature Placement

A local feature placement operation consists of the application of the shape detail procedure at a single seed point of the surface.

There are two ways to implement the local feature placement operation: subordinate or independent of the parametrization of the subdivision surface.

In this work we adopted the first option, where features are placed only at the control points of the subdivision surface. This option relies only on the basic subdivision surface structure. For this reason, it is simpler and more efficient, but has the disadvantage that when the parametrization is not uniform some distortions could happen.

Note that features can be placed at any arbitrary intermediate level of scale of the subdivision surface. Below we give some examples of applying the feature placement at a single level and at multiple levels.

[2]Note that, in some cases, this uniform global placement relies on the a set of seed points evenly distributed on the surface.

Placement at the Same Level

When features are placed at the same level, they all have the same size and usually they do not interfere with each other.

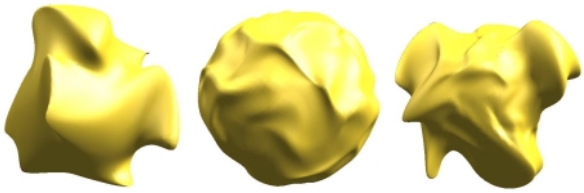
Figure 8 shows three examples of feature placement at the same level of the manequin head and skull models.

Figure 8: Local feature placement at the same level.

Placement at Different Levels

When features are placed at different levels, they have different sizes and usually interfere with each other. This enables a very powerful modeling framework.

Figure 9 shows one example of feature placement at different levels. In Figure 9(a), we applied a "spur" shape detail procedure to several points only at level 1 of a spherical surface. In Figure 9(b), we applied the same local feature operations only at level 3 of the surface. In Figure 9(c), we applied the local feature operations at both levels. Note that we obtained a combination of features at different scales.



(a) (b) (c)

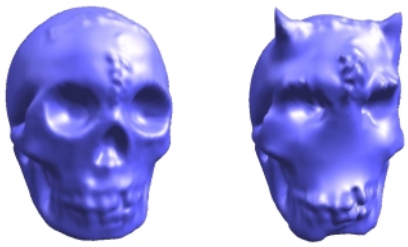
Figure 9: Combination of features at different levels.

5.2 Local Detail Modification

A local detail modification operation consists of the application of a signal processing operation to details in a small neighborhood of a point of the surface. The operation can attenuate or enhance the details as some levels. This is very much in the spirit of [6], where a range "frequency bands" of the surface features are modified. The local method has the advantage that, if applied interactively gives a much finer control of this technique to the user as a modeling tool.

To implement this operation is important to have two components: a distance function from a point on the surface that extends over the neighborhood where the modification is applied; and a smooth drop-off function of distance. These components together provide a way to apply the modification without creating discontinuities on the surface. In our implementation we currently employ a topological distance function with a cubic drop-off kernel.

Figure 10 shows an example of surface local signal processing applied to the skull model. In Figure 10(a) we present the original skull model and in Figure 10(b) we present the modified result. We smoothed the nose area and enhanced the jaw and details on top of the head to create a horny carnival mask. The interactive edit session took less than 5 minutes.



(a) (b)

Figure 10: Local signal processing for meshes.

6 Combining Multiscale Details

In this section, we describe mustiscale shape blending. This is a powerful shape combination operation, that can be either applied locally or globally. We remark that there are other multiscale combination operations for shapes, which we didn't consider, and remain as a topic for further research.

The blending operation between two multiscale shapes is done in the same way that Burt [3] defined a multi-scale blending between two images: at each scale level, a transition occurs which is proportional to the size of one sample at that scale level. As Burt demonstrated for images, the result is a surface definition that does not have an explicit visual transition. Instead, the effect is that one type of surface is gradually and naturally transformed into the other. This step involves only a linear combination of the two signals.

6.1 Blending of Different Procedural Models

The rusted fuse is an example of a multiscale blending of two shapes generated by different procedural models. One model is a rock and the other is a fuse.

The fuse shape is a surface of revolution whose profile is defined by $\sin(\sum 1/2^i \sin(t^{2^i}))$.

Figure 11 shows the result of blending between these two shapes. The blending is specified by a plane oriented in the $(1, 2, 0)$ direction. In order to avoid aliasing a "soft" transition region is used to blend between the the detail coefficients of the two models. This region changes from level to level according to $c * 1/2^{l-1}$, where c is a constant that depends on the size of the object. Note the while the transition is sharp at the finest level, the coarse level features of one shape influences the other beyond the dividing blending plane.

Figure 11: Blending between two procedural shapes.

6.2 Combining Instances of a Procedural Model

The planet is an example of combining the same procedural multiscale shape model with different parameters.

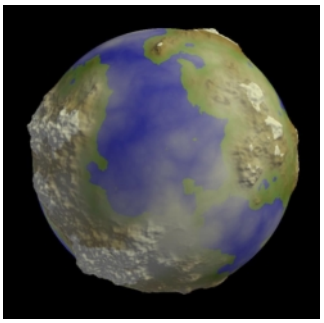
The procedural model used for the planet is the one defined for the rock. The difference is that instead of generating a displacement of the surface we generate an scalar function that is fed into a post-processing operation.

The result of shape blending is that two values are defined at every detail point: (i) a multi-scale blended scalar function value, and (ii) a blend parameter, between 0.0 and 1.0, which indicated the relative influence of each sub-planet on the final scalar value.

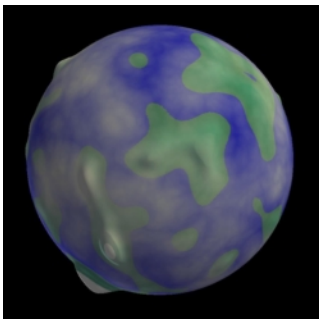
Post-processing

Once the blending is complete, there will be a single scalar value defined at every detail point on the surface mesh. This scalar value represents information from all scales that can be used to generate features requiring non-linear shader operations, such as snowcaps, mountains, lowlands, lakes, oceans. The post-blending procedural shader uses this scalar value, together with the blend variable, in arbitrary ways to generate the various terrestrial features. The blend parameter is used to influence the color produced by this procedural shader.

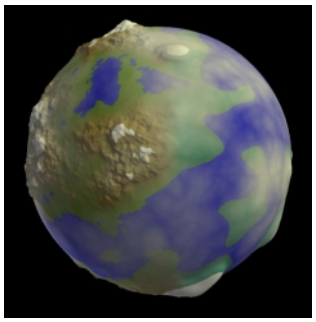
Figure 12(c) shows a synthetic planet which is the result blending in multiscale an "earth-like" planet, shown in Figure 12(a), with an "alien" planet, shown in Figure 12(b). Note how the different characteristics of the coastlines and topography blend seamlessly. One can see, scanning across individual features which straddle the transition region, that they gradually change their (statistically defined) appearance. For example, a single lake that appears jagged, with high fractal dimension, on one side of the transition, gradually turns into a smoothly contoured lake.



(a)



(b)



(c)

Figure 12: Planet.

Importance of post-processing: It is important that some portion of the procedural shading can be done after the multiscale blending has occurred. This allows the features created by that shader, which may involve non-linear operations, to be visually coherent across the transition created by the linear multi-scale blending operation.

7 Conclusions and Future Work

We have demonstrated how multi-scale representations can enable acquired shape data and synthetic procedural texture generators to be used together as a powerful and general shape modeling paradigm. These techniques can be applied locally and interactively to parts of a model, and can be used to seamlessly fuse together and reconcile models which have different shape and textural characteristics. The ability to work within different levels of a multi-scale representation allows a designer to interactively make changes at very different levels of scale, as well as to rapidly shift between large scale and detail work.

In future work, we plan to use these techniques to build a fully-featured procedural shape painting and editing system. We plan to incorporate infinitely zoomable surface representations, in a extension of the representation schemes that were presented for zoomable textural painting in [13]. This will allow designers to create procedurally-enhanced details of arbitrary scale. These surface representations can rely on lazy evaluation, so that the finest visible details of procedurally enhanced multiscale shape and displacement textures need ever be evaluated only when closely viewed.

References

[1] Adobe Systems. Adobe Photoshop. Software package.

[2] Deborah F. Berman, Jason T. Bartell, and David H. Salesin. Multiresolution Painting and Compositing. In *Proceedings of SIGGRAPH '94*, pages 85–90, July 1994.

[3] P. J. Burt and E. H. Adelson. A multiresolution spline with application to image mosaics. 2(4):217–236, October 1983.

[4] Ed Catmull and James Clark. Recursively generated B-spline surfaces on arbitrary topological meshes. 10(6):350–355, 1978.

[5] David S. Ebert, F. Kenton Musgrave, Darwyn Peachey, Steven Worley, and Ken Perlin. *Texturing and Modeling*. Morgan Kaufmann Publishers, July 1998.

[6] Igor Guskov, Wim Sweldens, and Peter Schröder. Multiresolution signal processing for meshes. *Proceedings of SIGGRAPH 99*, pages 325–334, August 1999.

[7] Paul E. Haeberli. Paint by numbers: Abstract image representations. *Computer Graphics (Proceedings of SIGGRAPH 90)*, 24(4):207–214, August 1990.

[8] Charles Loop. Smooth subdivision surfaces based on triangles. Master's thesis, University of Utah, Department of Mathematics, 1987.

[9] Michael Lounsbery, Tony DeRose, and Joe Warren. Multiresolution analysis for surfaces of arbitrary topological type. *Transactions on Graphics*, 16(1):34–73, January 1997.

[10] Ken Perlin. An image synthesizer. *Computer Graphics (Proceedings of SIGGRAPH 85)*, 19(3):287–296, July 1985. Held in San Francisco, California.

[11] Ken Perlin and Eric M. Hoffert. Hypertexture. *Computer Graphics (Proceedings of SIGGRAPH 89)*, 23(3):253–262, July 1989.

[12] Ken Perlin and Luiz Velho. A wavelet representation for unbounded resolution painting. Technical report, New York University, New York, 1992.

[13] Ken Perlin and Luiz Velho. Live Paint: Painting With Procedural Multiscale Textures. In *SIGGRAPH 95 Conference Proceedings*, pages 153–160, August 1995.

[14] K. Pulli and M. Lounsbery. Hierarchical editing and rendering of subdivision surfaces. Technical Report UW-CSE-97-04-07, Dept. of CS&E, University of Washington, Seattle, WA, 1997.

[15] Gabriel Taubin. A signal processing approach to fair surface design. In *SIGGRAPH 95 Conference Proceedings*, pages 351–358, 1995.

[16] Greg Turk. Generating textures for arbitrary surfaces using reaction-diffusion. *Computer Graphics (Proceedings of SIGGRAPH 91)*, 25(4):289–298, July 1991.

[17] Steven P. Worley. A cellular texture basis function. *Proceedings of SIGGRAPH 96*, pages 291–294, August 1996.

[18] Denis Zorin, Peter Schröder, and Wim Sweldens. Interactive multiresolution mesh editing. *Proceedings of SIGGRAPH 97*, pages 259–268, August 1997.