PERGAMON

Computers & Graphics 26 (2002) 865–875

COMPUTERS & GRAPHICS

www.elsevier.com/locate/cag

Algorithmic shape modeling with subdivision surfaces☆

Luiz Velho[a,*], Ken Perlin[b], Henning Biermann[b], Lexing Ying[b]

[a] IMPA—Instituto de Matemática Pura e Aplicada, Estrada Dona Castorina 110, 22461-320, Rio de Janeiro, Brazil
[b] Courant Institute of Mathematical Sciences, New York University, 719 Broadway, New York, NY 10003, USA

Abstract

We present methods for synthesizing 3D shape features on subdivision surfaces using multi-scale procedural techniques. Multi-scale synthesis is a powerful approach for creating surfaces with different levels of detail. Our methods can also blend multiple example multi-resolution surfaces, including procedurally defined surfaces as well as captured models.
© 2002 Elsevier Science Ltd. All rights reserved.

Keywords: Procedural models; Multi-scale surfaces; Catmull–Clark subdivision; Shape blending; Warping and morphing; Fractals; Signal mesh processing

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. Multi-resolution 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 multi-resolution subdivision surfaces as a basis to do multi-scale 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 fully exploit the multi-scale representation. Therefore, it gives a computational basis that allows designers to work across many levels of scale. The main contribution of our work is the adaptation of algorithmic synthesis and multi-scale operations to the context of shape modeling using subdivision surfaces.

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. Perlin [1] 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 [2] 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 [3] demonstrates a cellular texture basis function that divides space into cells in a manner similar

☆Based on "Procedural Shape Synthesis on Subdivision Surfaces" by L. Velho, K. Perlin, H. Biermann and L. Ying, which appeared in Proceedings of SIBGRAPI 2001 (XIV Brazilian Symposium on Computer Graphics and Image Processing) pp. 146–153 © 2001 IEEE.

*Corresponding author. Tel.: +55-21-529-5126; fax: +55-21-512-4115.

E-mail addresses: lvelho@impa.br (L. Velho), perlin@mrl.nyu.edu (K. Perlin), biermann@mrl.nyu.edu (H. Biermann), lexing@mrl.nyu.edu (L. Ying).

to our use of "seed" points. Perlin and Velho [4] apply procedural textures at different levels of a multi-scale domain in order to create infinitely zoomable 2D texture painting. The current work applies this multi-scale notion to 3D surface deformation.

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

Digital paint programs are a mainstay of 2D image generation. Multi-resolution image painting supports arbitrary resolution images [5]; Perlin and Velho [4] provide the ability to paint with multi-scale procedural textures. Haeberli and Hanrahan [6] introduced a paint program for painting textures directly onto 3D surfaces; descendants of this algorithms are available in many commercial 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 this paper is as follows: First, we introduce the concept of multi-resolution surface. Next, we show how to procedurally define multi-scale shape details, such as textures that resemble rock, berries, animated tentacles, and mushrooms. Then, we discuss the various ways that multi-scale 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 6.3.

2. Subdivision-based multi-resolution surfaces

Here we briefly review subdivision-based multi-resolution surfaces; details can be found in [7–9].

2.1. Subdivision surfaces

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. More precisely, the limit surface is the pointwise limit of a sequence of piecewise linear functions defined on the initial control mesh. Each finer mesh is obtained from the coarser mesh by using a set of fixed refinement rules, e.g. Loop [10] or Catmull–Clark [11] subdivision rules. In our work, we use Catmull–Clark subdivision surfaces.

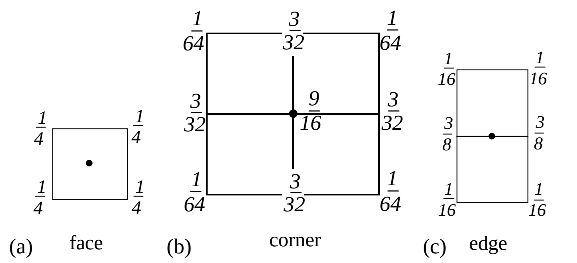

Fig. 1. Catmull–Clark rules for ordinary vertices.

Refinement rules can be specified by diagrams called *templates*. These templates indicate the weights that are used to express vertices of the refined mesh as a linear combination of their neighbor vertices in the unrefined mesh.

The Catmull–Clark subdivision scheme generalizes subdivision of bicubic tensor product B-splines. It produces surfaces that are C^2 everywhere, except at extraordinary vertices where they are C^1.

This scheme is composed of three refinement rules: a *face rule*, applied to new vertices at the center of current faces; an *edge rule*, applied to new vertices at edge midpoints of current faces; and a *corner rule*, for updating old vertices at the corners of current faces.

The Catmull–Clark subdivision rules for ordinary vertices (i.e. with valence $= 4$) are shown in Fig. 1. Extraordinary vertices (i.e. with valence$\neq 4$) occur only at the corners of existing faces. Therefore, the rule for extraordinary vertices generalizes the corner rule (see [11]).

2.2. Multi-resolution surfaces

Multi-resolution 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*. 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*.

An aspect of multi-resolution 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.

These two processes are illustrated in Fig. 2. Each block corresponds to an operation that is applied to the mesh. Such operations can change either the geometry

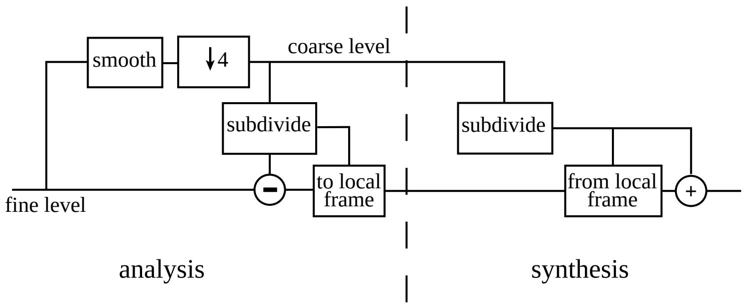

Fig. 2. Analysis and synthesis diagrams for multi-resolution surfaces. The analysis process decomposes a fine resolution mesh into a coarse base mesh and displacement details. The synthesis process adds details to a subdivided coarse mesh producing a finer resolution mesh.

(e.g., coordinate values of vertices) or the topology (e.g., face and edge connectivity) of the mesh.

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 [12], or quasi-interpolation [13]. For our purposes, quasi-interpolation appears to be the most suitable approach. After smoothing is applied the mesh is decimated (block $\uparrow 4$) to produce a coarse mesh. The details are generated by refining the mesh using subdivision and computing the difference between the smooth subdivided mesh and the fine mesh. These detail offsets are then converted to the local frame of their corresponding vertices.

The synthesis process reconstructs the fine mesh by subdividing the coarse mesh and adding details after they have been converted from relative offsets in the local frame to absolute offsets in global coordinates.

2.3. Atlases and charts

We use a variant of *characteristic maps* [14] to construct an atlas for the surface M. The atlas consists of overlapping chart maps mapping parts of the surface to the plane; the image of each chart map, the chart is homeomorphic to a disk. The domains of the chart maps cover the whole surface. We use this atlas to be able to make seamless transitions between different patches of the multi-resolution surface.

More precisely, a *chart* is a homeomorphism onto an open subset of the plane, $\Phi : M \to \mathbf{R}^2$. An *atlas* for M is a finite collection of charts

$$\{\Phi_i : U_i \to \mathbf{R}^2;\ i = 1, \dots, m\},$$

such that $\{U_i\}$ is an open cover of M. The maps

$$\Phi_{i,j} = \Phi_j \circ \Phi_i^{-1} : \Phi_i(U_i \cup U_j) \to \Phi_j(U_i \cup U_j)$$

are called *transition functions*. We say that the atlas is of class C^k if the transition functions are all of class C^k.

We define an atlas for a subdivision surface by associating charts with control vertices. Therefore, a single chart map is defined for each vertex of the control mesh. In the case of Catmull–Clark subdivision surfaces, this map takes a single ring of quadrilaterals around the vertex and maps it into the plane (note that, after the first subdivision level the Catmull–Clark mesh is formed only by quadrilateral faces). Thus, the overlap $(U_{v1} \cup U_{v2})$ of two adjacent charts Φ_{v_1} and Φ_{v_2} corresponds to a quadrilateral. Fig. 3 illustrates this setting. It shows a single chart map for a vertex v, the ring of quadrilaterals U_v on the surface (top) and the corresponding planar domain of Φ_v (bottom).

The details of the definition of the characteristic map are irrelevant for our purposes; only several important properties as well as the efficient way of computing the chart map are of importance to us.

Here we describe an efficient way to compute the chart map using subdivision. The values of the map for mesh vertices are computed by subdivision in the plane. We start with a double ring (e.g., the set of vertices in the 2-neighborhood of a vertex) of control points.[1] The positions of the control points in this ring depend only on the valence because they are in canonical position relative to the 2D parameter domain. We apply subdivision to refine the planar mesh, dropping all vertices of the mesh for which the control points are outside the domain of the characteristic map (i.e., points on the outer ring). As we proceed with refinement, we get a finer and finer mesh, which is in one-to-one correspondence with the part of the control mesh of our surface obtained by refinement of the single ring of quadrilaterals around the vertex of interest. In this way, we compute numerically a smooth parametrization given by the basis functions associated with the

[1] We need to use a double ring of control vertices because the values of the map are computed from both the inner and outer ring of control vertices surrounding the central vertex of the chart map.

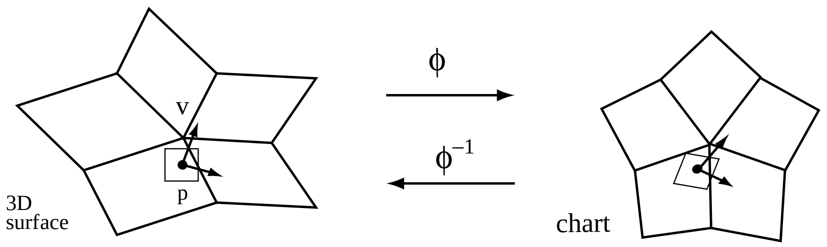

Fig. 3. Mapping Φ_v from the 3D surface to the chart corresponding to control point v, and its inverse.

subdivision scheme (e.g., bicubic B-Spline for regular vertices and Catmull–Clark for extraordinary vertices).

The chart map Φ_v, where v is a vertex, has the following important properties:

- All maps Φ_v are one-to-one;
- If v_1 and v_2 are two adjacent control vertices, then the composition $\Phi_{v_2} \circ \Phi_{v_1}^{-1}$ is C^2 continuous and regular.

It is worth noting that even if a scheme is only C^1 at extraordinary vertices, but C^2 everywhere else (e.g., the standard Catmull–Clark scheme), the transition functions $\Phi_{v_2} \circ \Phi_{v_1}^{-1}$ are still C^2.

3. Computational framework for multi-scale synthesis

In this section we describe the computational framework behind multi-scale 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 corresponding 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 [1]) 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 (Fig. 4) 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.

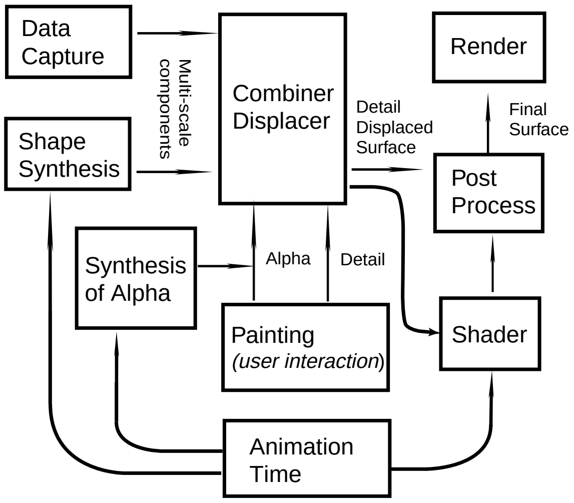

Fig. 4. Diagram of computational framework containing the different modules used to synthesize, modify and combine procedural and sampled detail components of a multi-resolution surface.

These shape definitions are blended together via an alpha signal [15]. The alpha signal defines the weight used by compositing operators to combine different shape details. This 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.

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

4. Defining multi-scale 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 multi-scale procedural shape synthesis is accomplished through the addition of geometric details at the various levels of the multi-scale 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 3D space in which the surface is embedded. 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 have experimented with all these kinds of displacement.

Below we present a chart relating this classification with the shape detail procedural models that we created, and will be illustrated in examples of the next section.

	Fractal	Morphogenic
Global (3D)	"Rock"	"Mushroom cloud"
Local (2D)	"Berry"	"Tentacle"

We remark that the relationship between displacements and scale is usually valid only within a limited range of scales. This observation is important for both theoretical and practical reasons. On one hand, in order to produce a well-defined surface the interpolated displacements from all scales must have a convergent sum. Therefore, this range needs to be controlled, specially when the displacements are increasing with scale. In this way, we define morphogenic features by displacements that increase only over a *limited range* of scales. On the other hand, it is only possible to compute a finite number of resolution levels when synthesizing a surface. Thus, the goal is to provide a good approximation to the limit surface.

4.2. Examples

Here we show some results of using our multi-scale procedural shape synthesis.

4.2.1. Rock

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 [1]. However, if one works within a multi-scale framework that contains a B-spline reconstruction filter at every successive scale level, it was demonstrated in [4] 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 P 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 < $n$Levels; level++) do
        for all $(u, v)$ on this level do $P$ = random()
```

In the second pass the stored values are retrieved from the parametric domain as displacements along the normal surface direction.

```
Vector rock_detail $(u, v, \text{level})$
  value = 0
  for ($k = 0; k <$ level; $k$++) do
    value += reconstruct_level $(2^k u, 2^k v, k)$
  return (0, 0, value)
```

In practice, we set the details for a surface point at all levels in one pass using the Perlin noise function. This makes the evaluation of this procedure $O(N)$ not $O(N^2)$.

At first glance, it would seem too restrictive the use of displacements along the normal direction only. However, this is not the case here because the details are relative to the local frame, and as the mesh is subdivided the surface can be displaced in arbitrary directions. In fact, displacements along the normal direction are highly desirable since they provide more control of the changes (see, for example, [16]).

The rock texture is used in some of the examples in Section 6.

4.2.2. Berry

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 [17]. This allows us to define a base level texture. Then we define 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. Fig. 5 shows the construction process for the berry shape.

4.2.3. Tentacles

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

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

```
Vector tentacle_detail (seed, level)
  Scalar magnitude = reference_length * level
  Scalar ph = (PI/3) * level
  Vector displace = (sin($\theta$ + ph), cos($\theta$ + ph), 1)
  if (is_even(level)) then
        displace *= -1
  return displace * magnitude
```

(Note that the intrinsic coordinates of the seed point must correspond at all levels. Also, observe that the **if** statement makes the tentacle grow in alternating directions as the mesh is subdivided.)

Fig. 6 illustrates the growth process of the tentacle for 4 levels of detail, as the feature grows from a seed point.

The shape feature model has two parameters: *reference length* and *rotation angle* θ. These two parameters determine the degrees of freedom of the articulated structure of the tentacle (e.g., orientation and relative size of links). The parameters can be used for modeling purposes. They can be time-varying also and be used for animation purposes.

Fig. 7 shows a tentacle creature produced by applying uniform tentacles to a spherical subdivision surface. The tentacles grow and rotate.

We remark that the basic structure of the tentacle shape detail model can be used as the basis to create other types of models such as the one shown in Fig. 8(a).

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

4.2.4. Mushroom cloud

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 within a range of scales and controlled in

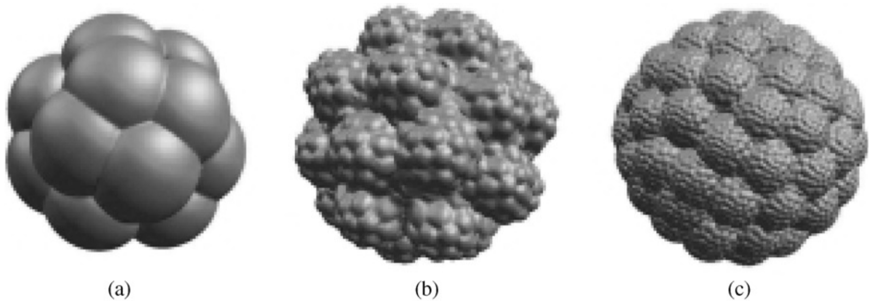

Fig. 5. Berry; (a) Base domes from initial seed points—level 1 of detail; (b) first recursion added to domes—level 2 of detail; and (c) final berry—3 levels of detail.

this particular procedural model to avoid self-intersection.

The procedural shape function (not shown because of space limitations) was designed to produce a mushroom feature. The displacements are such that when this basic feature is generated on a planar surface, the refined mesh does not self-intersects. This does not guarantees that self-intersections will not occur in general, but indicates that model will behaved well when applied to low curvature areas of a surface.

Fig. 8(b) shows an example of mushroom cloud features placed on a spherical shape. This shape was modeled interactively with the intent to create a ‘‘mushroom planet’’.

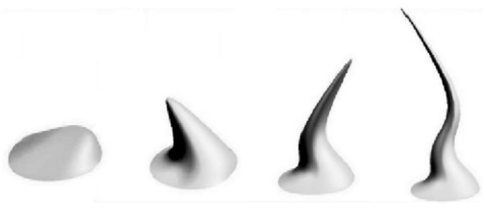
Fig. 6. Growth process of tentacle. Levels 1–4.

5. Applying multi-scale detail to the surface

Once we have defined a repertoire of multi-scale 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.

Note that, in some cases, this uniform global placement relies on a set of seed points evenly distributed on the surface. One example is the berry model.

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 800 Mhz class machine with 512 Mbytes of memory and an OpenGL graphics card.

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

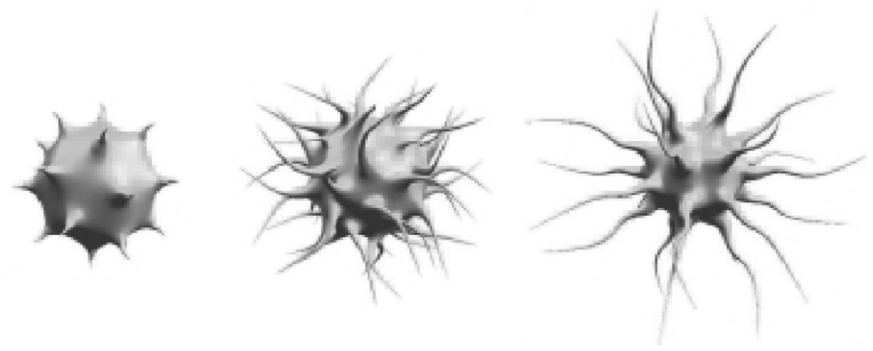
Fig. 7. Tentacle creature. Tentacles grow and rotate.

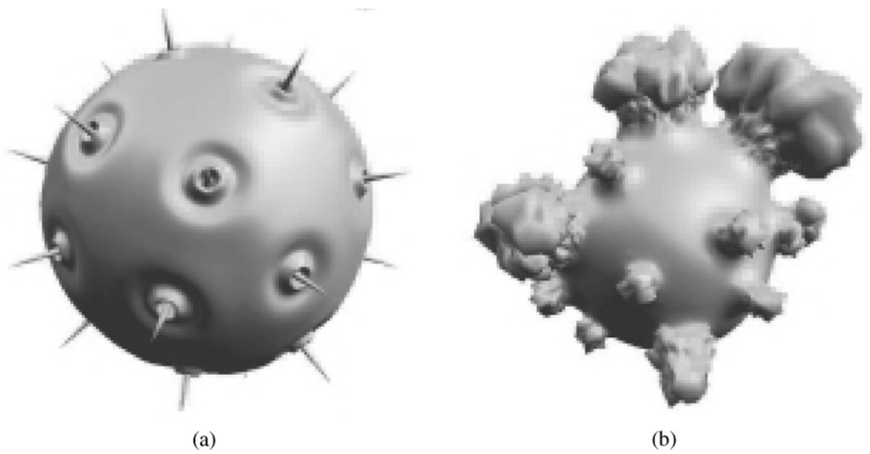

Fig. 8. (a) Submarine explosive mine; and (b) mushroom planet (inspired on the planet from *The Little Prince* of Saint-Exupery).

5.1. Local 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, i.e., the displacements are applied either at discrete locations of the basis functions that define the multi-resolution surface, or at arbitrary locations of the continuous surface domain.

The first option is simpler and more efficient, but has the disadvantage that when the parametrization is not uniform some distortions could happen. This option relies only on the basic subdivision surface structure.

The second option is more complex and less efficient, but has the advantage of being adapted to the surface metric. This option relies on the machinery for surface charts described in Section 2.3.

We have implemented both options, and some of the examples were generated using the first implementation while others the second. (We remark that parametrization independent feature placement employs a different definition of the surface detail procedures—essentially the detail coefficients have to be re-synthesized from the original shape detail procedure, by performing a multi-scale analysis on the tangent plane of the surface at the seed point.)

The 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 the same level and at multiple levels.

5.1.1. 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. Fig. 9 shows examples of feature placement at the same level of the mannequin head and skull models.

5.1.2. 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. Fig. 10 shows one example of feature placement at different levels. In Fig. 10(a), we applied a "spur" shape detail procedure to several points only at level 1 of a spherical surface. In Fig. 10(b), we applied the same local feature operations only at level 3 of the surface. In Fig. 10(c), we applied the local feature operations at both levels. Note that we obtained a combination of features at different scales.

5.2. Local detail modification

A local detail modification operation consists of the application of a signal processing operation in a small neighborhood of a point of the surface. The operation can attenuate or enhance the details at some levels. This is very much in the spirit of [18], where a range of "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 it 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.

Fig. 11 shows an example of surface local signal processing applied to the skull model. In Fig. 11(a) we present the original skull model and in Fig. 11(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 user did not have any particular artistic skills. Nonetheless, the system

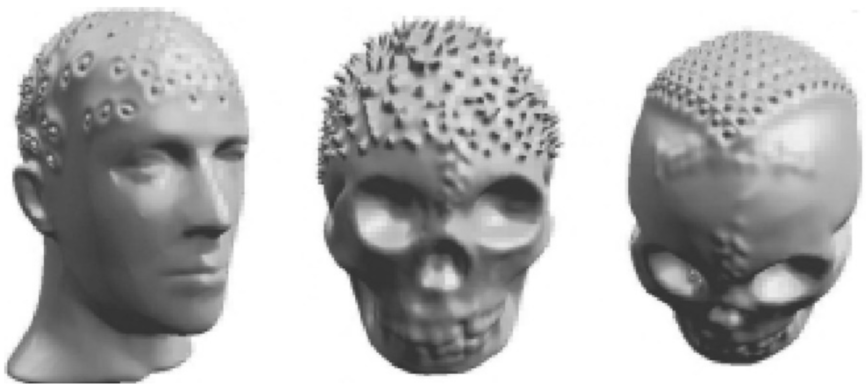
Fig. 9. Local feature placement at the same level.

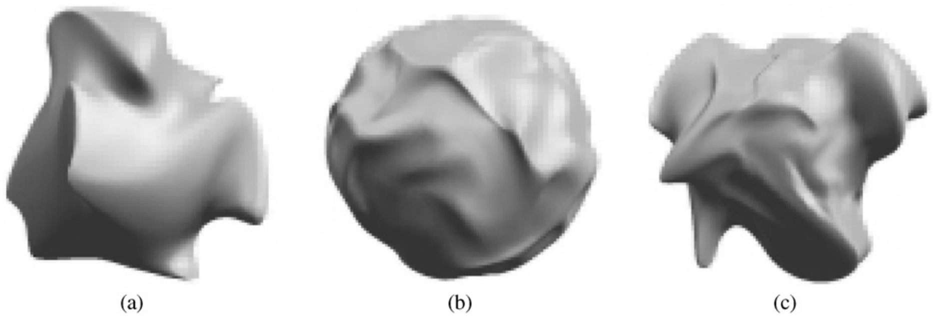
Fig. 10. Combination of features at different levels.

revealed itself to be intuitive and responsive. The interactive edit session took less than 5 min.

6. Combining/blending different multi-scale details

In this section, we describe multi-scale shape blending. This is a powerful shape combination operation, that can be applied either locally or globally. We remark that there are other multi-scale combination operations for shapes, which we did not consider, and remains as a topic for further research.

The blending operation between two multi-scale shapes is done in the same way that Burt [19] defined a multi-scale blending between two images: at each scale level, a transition occurs whose width 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 two procedural shapes

The rusted fuse is an example of a multi-scale blending of two shapes generated by different procedural models. One model is a rock and the other is a fuse-like shape.

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

Fig. 12 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 detail coefficients of the two models. This region changes from level-to-level according to $c1/2^{l-1}$, where c is a constant that depends on the size of the object. Note that while the transition is sharp at the finest level, the coarse level features of one shape influences the other beyond the dividing blending plane.

6.2. Blending procedural and sampled shapes

The mummified skull is an example of blending between a synthetic multi-scale shape and a real-world object. One shape is the rock model and the other is the skull model. The skull is a multi-scale surface generated by performing multi-scale analysis to a digitized polygonal mesh. Note that in the case of the multi-resolution subdivision surface the detail coefficients come from the analysis pipeline.

Fig. 13 shows the blending of these two shapes to create the mummified skull. In Fig. 13(a) we used the same transition regions as in the previous example to produce a sharp blend. In Fig. 13(b) we used a transition region that has the same extent at all levels to create a "soft" blend between the two shapes.

6.3. Blending instances of a procedural model

The planet is an example of combining the same procedural multi-scale 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 a scalar function that is fed into a post-processing operation.

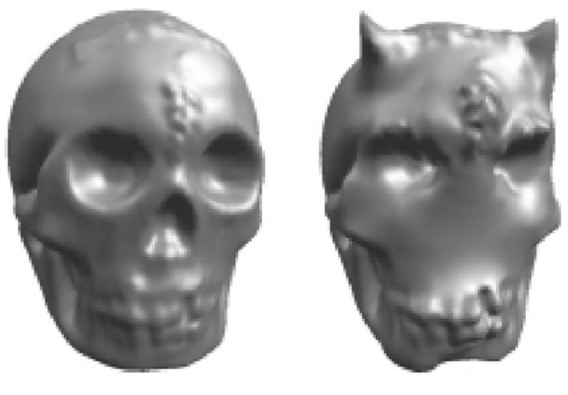
Fig. 11. Local signal processing for meshes.

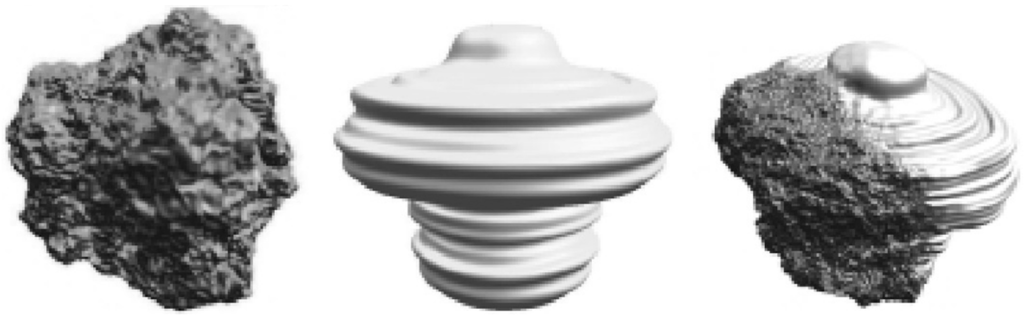
Fig. 12. Rusted fuse: blending between two procedural shapes.

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.

6.3.1. 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.

Fig. 14(c) shows a synthetic planet which is the result of blending in multi-scale an "earth-like" planet, shown in Fig. 14(a), with an "alien" planet, shown in Fig. 14(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.

It is important that some portion of the procedural shading can be done after the multi-scale 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 using 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

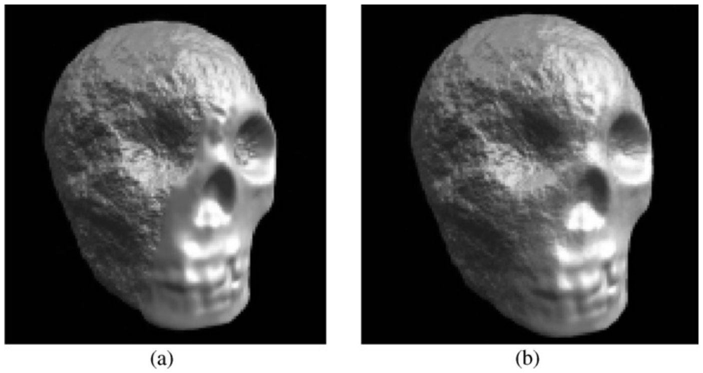
Fig. 13. Skull: (a) sharp transition; and (b) soft transition.

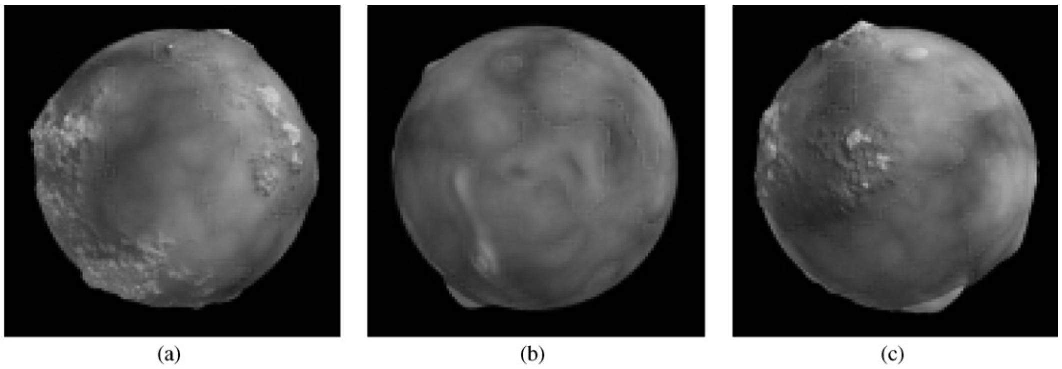

Fig. 14. Planet.

editing system. We plan to incorporate infinitely zoomable surface representations, in an extension of the representation schemes that were presented for zoomable textural painting in [4]. 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 multi-scale shape and displacement textures need ever be evaluated only when closely viewed.

Acknowledgements

This work is part of a CNPq/NSF international cooperation agreement. Luiz Velho is a member of Visgraf and is partially supported by a CNPq research grant. Visgraf is sponsored by CNPq, FAPERJ, FINEP, and IBM Brasil.

References

[1] Perlin K. An image synthesizer. Computer Graphics 1985;19(3):287–96.

[2] Perlin K, Hoffert EM. Hypertexture. Computer Graphics 1989;23(3):253–62.

[3] Worley SP. A cellular texture basis function. Proceedings of SIGGRAPH 96, August 1996. p. 291–4.

[4] Perlin K, Velho L. Live paint: painting with procedural multiscale textures. In: SIGGRAPH 95 Conference Proceedings, 1995. p. 153–60.

[5] Berman DF, Bartell JT, Salesin DH. Multiresolution painting and compositing. In: Proceedings of SIGGRAPH '94, 1994. p. 85–90.

[6] Haeberli PE. Paint by numbers: abstract image representations. Computer Graphics 1990;24(4):207–14.

[7] Lounsbery M, DeRose T, Warren J. Multiresolution analysis for surfaces of arbitrary topological type. Transactions on Graphics 1997;16(1):34–73.

[8] Pulli K, Lounsbery M. Hierarchical editing and rendering of subdivision surfaces. Technical Report, 04-07, University of Washington, 1997.

[9] Zorin D, Schröder P, Sweldens W. Interactive multiresolution mesh editing. In: Proceedings of SIGGRAPH 97, 1997. p. 259–68.

[10] Loop C. Smooth subdivision surfaces based on triangles. Master's thesis, University of Utah, 1987.

[11] Catmull E, Clark J. Recursively generated B-spline surfaces on arbitrary topological meshes. Computer Aided Design 1978;10(6):350–5.

[12] Taubin G. A signal processing approach to fair surface design. In: SIGGRAPH 95 Conference Proceedings, 1995. p. 351–8.

[13] Biermann H, Martin IM, Zorin D, Bernardini F. Sharp features on multiresolution subdivision surfaces. Pacific Graphics, 2001. p. 140–9.

[14] Reif U. A unified approach to subdivision algorithms near extraordinary points. Computer Aided Geometric Design 1995;12:153–74.

[15] Porter T, Duff T. Compositing digital images. Computer Graphics 1984;18(3):253–9 (held in Minneapolis, Minnesota).

[16] Guskov I, Vidimce K, Sweldens W, Schröder P. Normal meshes. In: Proceedings of ACM SIGGRAPH 2000, Computer Graphics Proceedings, Annual conference series, ACM Press/ACM SIGGRAPH/Addison-Wesley Longman, 2000. p. 95–102. ISBN 1-58113-208-5.

[17] Turk G. Generating textures for arbitrary surfaces using reaction–diffusion. Computer Graphics (Proceedings of SIGGRAPH 91) 1991;25(4):289–98.

[18] Guskov I, Sweldens W, Schröder P. Multiresolution signal processing for meshes. Proceedings of SIGGRAPH 99, August 1999. p. 325–34.

[19] Burt PJ, Adelson EH. A multiresolution spline with application to image mosaics. ACM Transactions on Graphics 1983;2(4):217–36.