Generating Organic Textures with Controlled Anisotropy and Directionality

Takayuki Itoh
IBM Research, Tokyo Research Lab

Kazunori Miyata
Japan Advanced Institute of Science and Technology

Kenji Shimada
Carnegie Mellon University

This article presents a method for generating organic textures by tessellating a region into a set of pseudo-Voronoi polygons using a particle model and then generating the detailed geometry of each of the polygons with fractal noise.

As one of the main attributes of an object's surface, a quality texture is often the key to realistic computer graphics. Textures represent various aspects of an object's surface properties, including optical properties, such as colors and glossiness, and geometric properties, such as bumps and dents. Adding texture to an image produces richer and more realistic surfaces. However, it's tedious and impractical for graphic designers to draw organic textures manually.

One way to avoid manual drawing is to scan a photograph of a real organic texture and map it onto surface geometry. However, when the surface of an organic texture has small geometric features, such as bumps and dents, its shades and shadows do not match the lighting conditions used for rendering the rest of the scene. And when the aspect ratio of the real photograph differs from that of the target surface, the texture becomes distorted. Our method solves these problems by tessellating an object into a set of polygons, generating a cell geometry for each polygon with fractal noise, and rendering the image onto the object's surface.

Because we use a particle system and physically based simulation to tessellate a region into a set of pseudo-Voronoi polygons, our method might seem similar to some of the previous work in texture generation—especially the cellular texture method.[1] However, this particle method, originally proposed for mesh generation, is well suited to organic texture generation because it lets you specify an anisotropic force field to generate texture cells. You can also specify rectangular arrangements with controlled directionality. In our method, after we tessellate a region into a set of pseudo-Voronoi polygons, we define an initial polyhedral shape for each of the polygons by sweeping each polygon with tapering and skewing effects. We then refine the polyhedral geometry and round it while using fractal noise to add fine geometric features.

With our method, you can automatically create an organic texture consisting not only of hexagonally arranged cells, but also of orthogonally arranged cells. Because our method first tessellates a region into a set of polygonal cells, it generates a texture for an arbitrarily shaped surface by simply specifying the geometry, cell directionality, and a few rendering parameters.

Characteristics of organic textures

By observing real organic textures, such as the ones shown in Figure 1, we observed the following characteristics: An organic texture consists mostly of cells in hexagonal or rectangular arrangements. And each cell typically takes the shape of a hexagon or a rectangle. In orthogonal cell arrangements, organic textures often show distinct directionality by forming streamlines.

In organic textures, cell geometry is often stretched in a certain direction. The aspect ratio of the anisotropy and its orientation might vary over the entire surface domain. Two constraints make generating such organic textures difficult: the rectangular arrangement of the cells and their anisotropic geometry. These two issues closely relate to how a region is tessellated. Once you locate a set of points or generators in a 2D domain, you can use the popular Voronoi method to tessellate the domain into a set of polygons.[2]

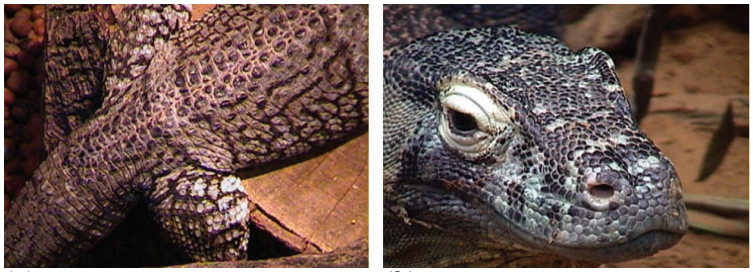
The larger challenge, however, is to find a set of points with a rectangular arrangement in which the rectangles' orientations are aligned with specified streamlines. Such point locations cannot be created with the physically based particle systems previously proposed in computer graphics research, such as the techniques that include retiling polygon models or representing implicit surfaces. This is because each of these particle systems creates a hexagonal arrangement of points. You can apply a cellular texture method[1] to a rectangular and an anisotropic tessellation, but because the method is an

 Published by the IEEE Computer Society

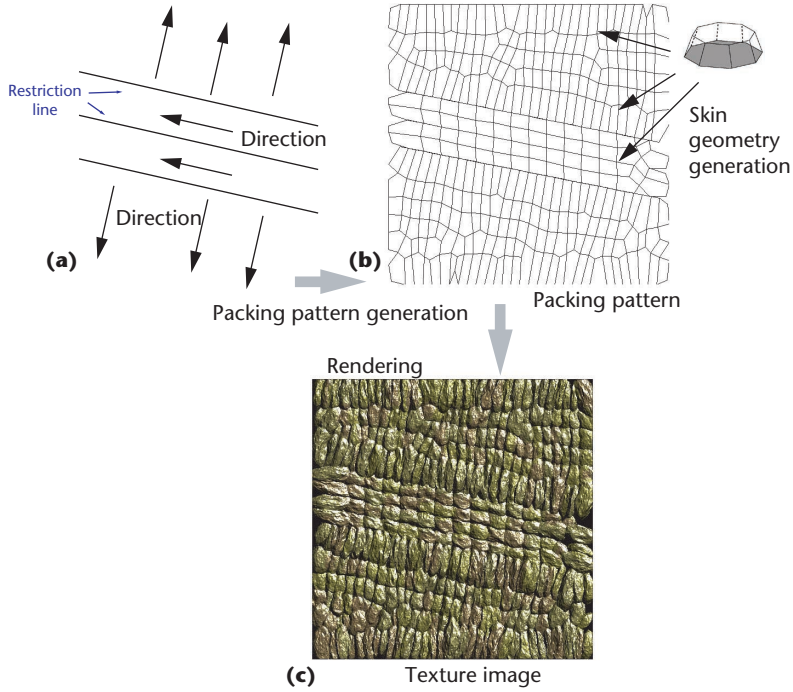
energy-optimization approach rather than a physically based approach, it requires expensive computations.

We use a different physically based particle system, called *bubble mesh*, which was originally devised for solving various meshing problems in FEM analysis. The bubble mesh system can help create different types of meshes by packing cells of various shapes—circles for isotropic triangular meshes,[3] ellipses for anisotropic triangular meshes,[4] squares for isotropic quadrilateral meshes,[5] and rectangles for anisotropic quadrilateral meshes.[6]

As Figure 2 shows, our proposed method consists of three steps: making a packing pattern of skin cells, generating each skin cell's 3D geometry with fractal noise, and rendering a final image based on the skin geometry and lighting parameters.



1 Examples of organic textures: (a) crocodile and (b) lizard.

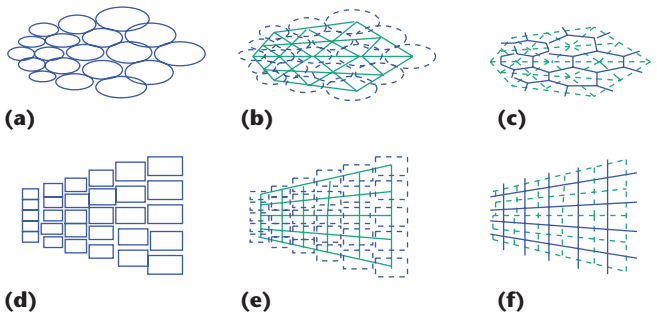


2 Method overview: After we generate (a) the packing pattern, we generate (b) the skin geometry. We then obtain (c) a textured image by rendering the texture with specified lighting conditions.

Pseudo-Voronoi tessellation

Given a 2D geometric domain, a desired size distribution of cells, a desired directionality, and a desired anisotropy, we can generate a polygonal tessellation consisting primarily of quadrilateral or hexahedral polygons compatible with the specified cell sizes, directionality, and anisotropy. You can specify the input size distribution or automatically calculate it according to the curvature of the input domain or other metrics. Additionally, you can specify the input directionality or automatically calculate it by interpolating from the direction of the input boundary edges.

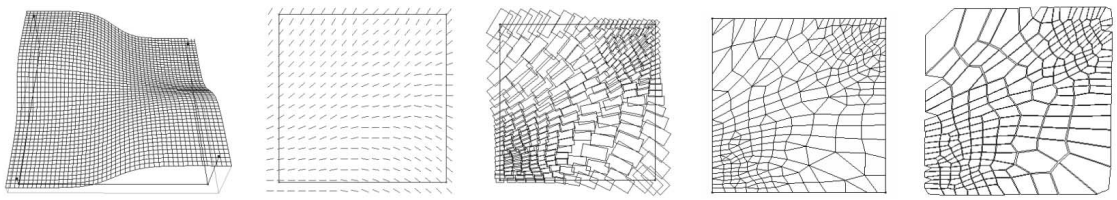
This approach is based on the observation that natural-looking hexahedral or rectangular tessellations are geometric duals of well-shaped triangular or quadrilateral meshes, as Figure 3 shows. Our algorithm performs pseudo-Voronoi tessellation in three steps: packing elliptic or rectangular cells closely in the domain (Figures 3a and 3d), generating a triangular or quadrilateral mesh by connecting the centers of the ellipses or rectangles (Figures 3b and 3e), and tessellating the domain into mostly hexagonal and rectangular polygonal cells (Figures 3c and 3f). Figure 4 (next page) shows an example of a pseudo-Voronoi tessellation.



3 Pseudo-Voronoi tessellation: (a) elliptic cells, (b) a triangular mesh, (c) hexahedral Voronoi polygons, (d) rectangular cells, (e) a quadrilateral mesh, and (f) quadrilateral Voronoi polygons.

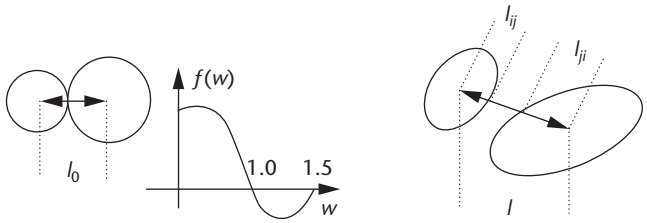
Close cell packing

We designed the particle model implemented here to obtain a closely packed arrangement of elliptic or rectangular cells. We defined a proximity-based force field between pairs of cells so that the force field exerts a repelling force when the two cells are too close together. Our system exerts an attracting force when the two cells are separated by more than a specified distance. This method—based on the bubble mesh method—



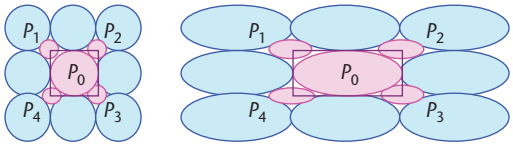
(a) (b) (c) (d) (e)

4 Substeps of packing pattern generation: (a) heights of grid points representing preferred cell sizes, (b) preferred directionality of rectangular packing, (c) closely packed rectangles, (d) primarily quadrilateral mesh, and (e) pseudo-Voronoi tessellation.



(a) (b)

5 Circular cell packing and elliptic cell packing: (a) stable distance and force function of circular cells and (b) stable distance of elliptic cells.



(a) (b)

6 Square cell packing and rectangular cell packing: (a) a square cell realized using five circular force fields and (b) a rectangular cell realized using five elliptic force fields.

tightly packs a set of circles using a force field similar to the van der Waals force.

In the original bubble mesh method, you calculate the stable distance between the centers of two adjacent circular cells as the sum of the desired radii of the two cells. A user-given scalar field specifies the radii of the cells. Here, we denote the radii of the two cells as r_i and r_j; the distance between the centers of the cells as l; the stable distance between the cells as $l_0 = r_i + r_j$; the ratio of the current distance and l_0 as $w = l/l_0$; and the corresponding linear spring constant at the target distance as k_0. We describe the force model used in the bubble mesh method as a function of w. The essential characteristic of this approach is that we apply a repulsive force when $w < 1$ and an attractive force when $w > 1$. As Figure 5 shows, the force $f(w)$ satisfies the following conditions: $f(1) = f(1.5) = 0, f'(0) = 0, f'(1) = -k_0$.

By solving for these conditions, we can write the function as follows:

$$f(w) = \begin{cases} \dfrac{k_0}{l_0}\left(\dfrac{5}{4}w^3 - \dfrac{19}{8}w^2 + \dfrac{9}{8}\right) & 0 \le w \le 1.5 \\ 0 & 1.5 \le w \end{cases}$$

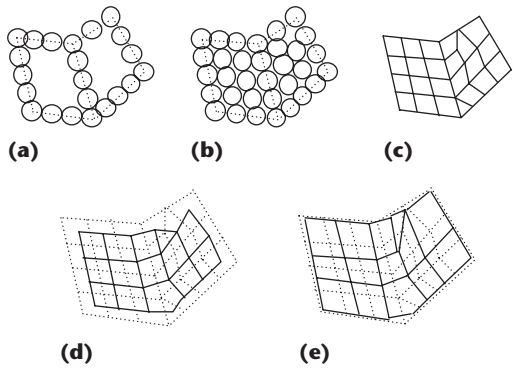
We extended the bubble mesh method to provide close packing of elliptic cells. This approach calculates the directions of the major and minor axes of a cell from the given direction and calculates the radii along these axes from the given cell sizes and anisotropy. We calculate the effective distance between two cells as the sum of the two lengths measured along the line segment that connects the centers of the two cells to their boundaries. If the effective distance is less than the desired stable distance, then the force is negative and the cells are repelled from each other. If these two lengths are l_{ij} and l_{ji}, as shown in Figure 5, then we can write the effective distance as $l_0 = l_{ij} + l_{ji}$.

We also extended the bubble mesh method to achieve close packing of square cells. We denote a potential field around a cell as Ψ_{p0}. For square cells to align orthogonally, we add four subpotential fields Ψ_{p1}, Ψ_{p2}, Ψ_{p3}, and Ψ_{p4} at the four corners of the square cell P_1, P_2, P_3, and P_4 to the original potential field Ψ, as Figure 6 shows. If the desired cell size is locally uniform, the radii of the four subpotential fields should be $(\sqrt{2}-1)r_0$, where r_0 is the radius of the central potential field Ψ_{p0}. If you specify graded cell sizes, however, the radii of the subpotential fields should be adjusted accordingly. We express the potential field shown in Figure 6 as a weighted linear combination of the central potential field and the four subpotential fields:

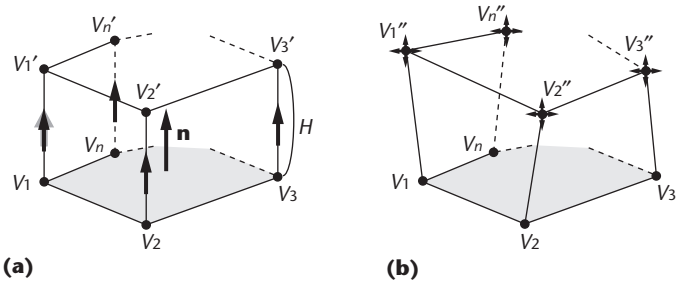
$$\Psi = \Psi_{P_0} + (\sqrt{2}-1)(\Psi_{P_1} + \Psi_{P_2} + \Psi_{P_3} + \Psi_{P_4})$$

We extended this square cell packing approach to rectangular cell packing, as Figure 6 shows. In all cases, we assume both the point mass at the center of each cell and the effect of viscous damping. We then solve the motion equation numerically to find a tightly packed configuration of cells using a standard numerical integration scheme. We terminate the integration process when the displacements of all cells in an iteration become small. While solving the equation, we adjust the number of cells in the domain by checking the population density. We add one cell around another cell when the smallest w value between the cell and its adjacent cells is larger than a user-specified value. We delete a cell when the smallest w value between that cell and its adjacent cells is smaller than a user-specified value.

The geometric input models consist of a set of 3D polygons. As Figure 7 shows, we first pack cells along the boundaries of the polygons, then pack them inside the polygons. If the input model is too fine, we pack cells



7 Cell packing and Voronoi tessellation processes: (a) cells on the domain boundary, (b) cells inside the domain, (c) mesh connecting the centers of the cells, (d) Voronoi tessellation, and (e) boundary treatment.



8 Initial skin shape: (a) sweeping base polygon and (b) random displacement.

using a simplified model, then project them onto the original input model. We express the complexity of the cell-packing process as $O(mn)$, where O denotes the amount of complexity, n denotes the total number of square cells and m denotes the average number of cells that suffer attractive or repulsive forces from another cell. To accomplish this, we store cells in a grid surrounding the entire input domain to search for the adjacent cells surrounding a given cell. Because our system extracts an almost constant number of adjacent cells, we treat m as a constant. The complexity of the process increases according to the total number of cells, which means that our method should be faster than optimization-based methods such as the cellular texture method.

Pseudo-Voronoi tessellation from packed cells

Given closely packed elliptic or rectangular cells, our approach generates anisotropic triangular or quadrilateral meshes by connecting the centers of the cells. To obtain the configuration of an anisotropic triangular mesh from the centers of elliptic cells, we applied an anisotropic Delaunay triangulation algorithm.[4,6] While the original Delaunay triangulation algorithm satisfies the condition that no other vertices lie inside a triangular element, this anisotropic triangulation uses circumellipses defined by the user-given directionality and anisotropy, instead of using circumcircles.

To obtain an anisotropic quadrilateral mesh by connecting the centers of a set of packed rectangular cells, we used a triangular-to-quadrilateral mesh-conversion algorithm. The mesh-conversion algorithm first tests all the possible quadrilaterals formed by coupling two adjacent triangular elements and then computes a score that measures the quality of each resultant quadrilateral element. The algorithm then converts the pairs of triangles to a set of quadrilaterals in the order of their scores. Our method then generates a pseudo-Voronoi tessellation from these anisotropic triangular, quadrilateral, or mixed meshes. We obtain this tessellation by connecting the center of each mesh element to the centers of all adjacent mesh elements. As Figure 7 shows, a polygon encloses each internal mesh node.

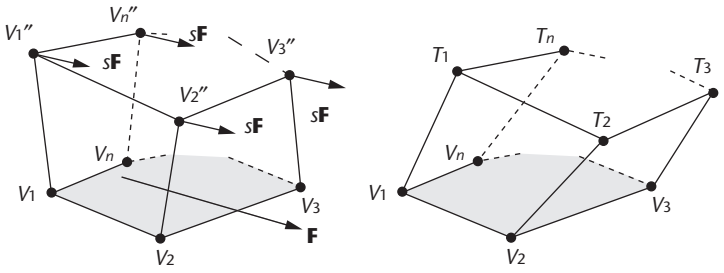
Our method does not fill the entire given domain with the Voronoi polygons. It requires special treatment around the domain boundary. We call the edge of the Voronoi polygon a *boundary edge*. Boundary edges can run along a given domain's perimeter or at a set of user-specified line segments inside the domain. When a node connects to such boundary edges, our approach connects the center of the boundary edges instead of connecting the centers of the mesh elements. Also, our algorithm does not form pseudo-Voronoi polygons for a vertex on a boundary edge. Our approach therefore generates pseudo-Voronoi polygons that are well aligned along the mesh boundary and along other user-specified line segments.

Organic texture generation

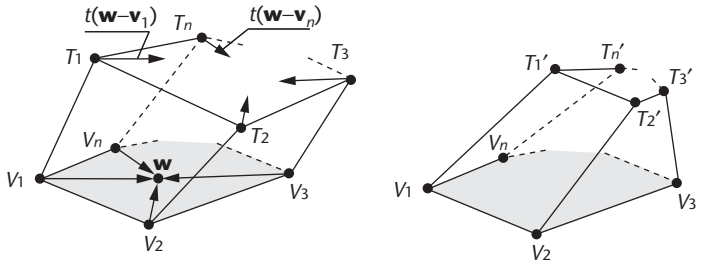
We obtain the organic texture by generating a skin texture for each of the pseudo-Voronoi cells created by the method described previously. We create each skin texture by generating the initial skin mesh for each pseudo-Voronoi polygon, smoothing the mesh by a subdivision surface method, and creating small geometric features with fractal noise.

Each generated pseudo-Voronoi polygon must be transformed into a realistic skin surface geometry. First, depending on the type of target organic texture, we might add a gap between the cells or pack the cells more tightly by scaling them. The amount of scaling is specified by a scaling parameter (Sv). Next, we obtain an initial 3D skin mesh by sweeping the polygon of each cell in the normal vector direction. We calculate the height of this sweeping operation by multiplying the skin size (the average distance from the center of gravity of a cell polygon to the corners of the polygon) with a user-specified sweeping parameter. The sweeping operation defines a prism. We then deform this prism by displacing the top corners randomly within a specified displacement range, as Figure 8 shows. We can adjust the distortion parameter to create various skin shapes.

After we deform the initial skin shape, we can skew the prism, as Figure 9 (next page) shows, according to a specified flow vector **F**. In Figure 9, we displace each of the top corners (V_1'', V_2'', and so forth) by vector $s\mathbf{F}$, where s is a user-defined skewing parameter. This process adds overlapping layers of skin cells. Choosing a larger skewing parameter can generate scale-like skins. We can deform the prism further with a tapering operation, as Figure 10 shows, by displacing each of the top corners of



9 Skewing operation: (a) flow vector that defines the direction of skewing and (b) skewed prism.



10 Tapering operation: (a) centripetal displacement and (b) tapered prism.

the prism (T_1, T_2, and so forth) with a centripetal vector $t(\mathbf{w}-\mathbf{v}_i)$, where t is the user-defined tapering parameter and $\mathbf{w}$ is the center of gravity of the base polygon. Setting a larger tapering parameter yields thorny skins.

After we apply all of these deformations, we represent the shape of each skin cell as a coarse triangular mesh, called the initial skin mesh. We obtain the final skin mesh by refining and smoothing the initial skin mesh. After we generate the initial skin meshes, or initial control meshes, we have to smooth their sharp corners. We achieve this smoothing by using a surface subdivision method. Other researchers have proposed several surface subdivision methods,[7,8] but we use Loop's method[9] because it generates triangular patches and works well with our implementation.

With Loop's method, we insert new vertices at the midpoint of each edge of a given control mesh. We then connect the vertices to divide an initial triangular element into four smaller triangles. In parallel with the mesh-smoothing process, we can further displace skin mesh nodes with fractal noise to add small geometric features, such as bumps and dents, to a surface.

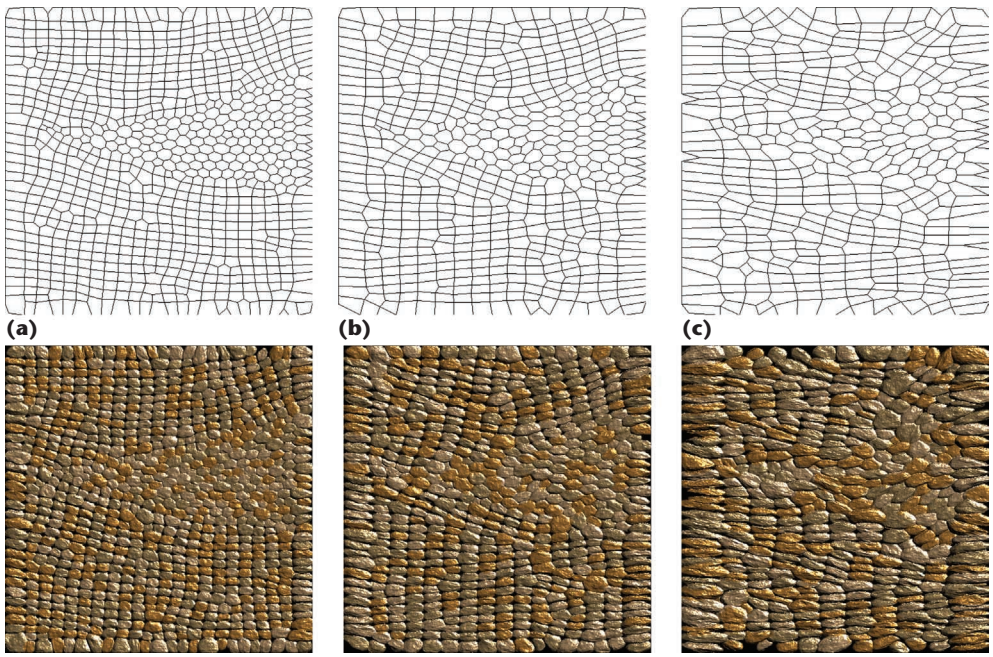
We generate 3D fractal noise by recursively subdividing a triangular mesh element into smaller triangles and then adding a new node at the midpoint of each side of the triangle. The nodal point is then displaced vertically.[10] We store and merge the displacement vectors for nodal points at each subdivision level into the subdivision surface geometry.

Results

Our system gives us the ability to adjust many parameters to create different organic textures:

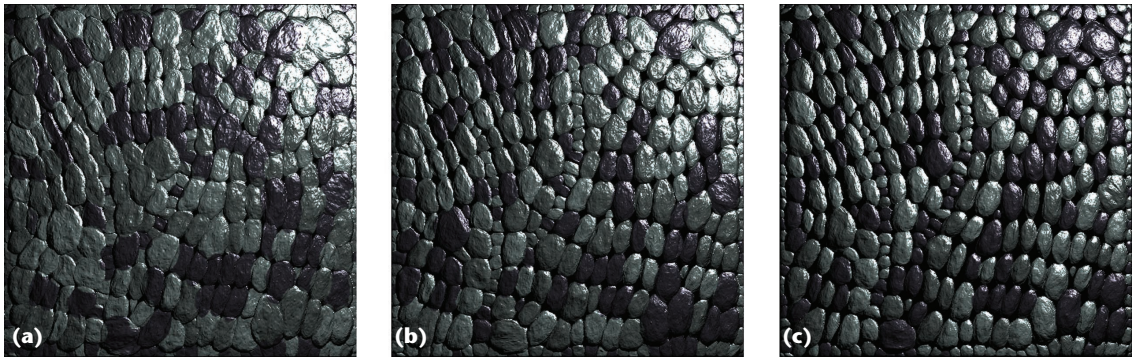
- Degree of anisotropy, or aspect ratio of cell geometry
- Scaling parameter of cell polygon
- Sweeping parameter of initial skin mesh
- Skewing parameter of initial skin mesh
- Tapering parameter of initial skin mesh
- Distortion parameter of initial skin mesh
- Fractal dimension of fractal noise controlling skin's surface roughness
- Amplitude of fractal noise

One unique capability of our texture-generation method is the ability to control the anisotropy of the tex-



11 Textures with different degrees of anisotropy, or aspect ratios: (a) 1.5, (b) 2.0, and (c) 3.0.

12 Textures with different values of the skewing parameter: (a) 0.0, (b) 2.0, and (c) 3.0.



13 Textures with different values of the tapering parameter: (a) 0.0, (b) 0.2, and (c) 0.5.

ture cells. Figure 11 shows how texture appearances change with different aspect ratios. By setting a high aspect ratio, we can stretch the textures in specific flow directions. Figure 12 shows the effect of changing the skewing parameter. Specifying a larger skewing parameter leads to some overlapping layers of texture cells, which can yield scale-like skins. The textures shown in Figure 12, in contrast to those shown in Figure 11, have bumpier surfaces because of the larger amplitude of fractal noise. Figure 13 shows the effect of changing the tapering parameter. The skins bulge when we use a larger taper parameter.

One of the advantages of our method is the ability to create cells packed in both hexagonal arrangements and rectangular arrangements. If you use a triangular mesh for generating the pseudo-Voronoi tessellation, the resultant cell arrangement becomes hexagonal. And if you use a quadrilateral mesh, then the resultant cell arrangement becomes rectangular. Our method can generate organic textures and map them onto a 3D object. Figure 14 shows an example of this technique. In the example, we packed cells in a rectangular arrangement and specified their directionality so that the texture cells align well along the longitudinal direction of the leg.

Computational time changes according to the number of texture cells to be generated. In the examples shown in Figures 11, 12, and 13, it takes 10 to 40 seconds to generate pseudo-Voronoi tessellations; it takes approximately 3 to 6 additional seconds to generate detailed skin geometry on an Intel Pentium III 933-MHz processor. The size of the generated texture images in all the examples is 512 × 512 pixels. For the examples in Figures 11, 12, and 13, we laid the 3D meshes on a plane and rendered them in a parameter space using in-house software. For the example in Figure 14, we imported the 3D meshes into 3D Studio Max and rendered them. The number of generated triangles depends on the number of cells and their shapes. Typically, the number runs to around 0.5 million.

14 A textured leg illustrating the technique of generating organic textures and mapping them onto 3D objects.

The methods we use do not rely on texture maps. The visual quality of our results might be lower than those techniques that do rely on texture maps, but the geometrical richness of our approach is competitive with other methods. We intend to continue extending this method to generate surface attributes, such as colors and optical features, procedurally for each skin cell.

Related Work

Researchers working in generating organic textures have typically published findings in three areas: tiling textures, which subdivide a surface into subregions and generate procedural textures for each subregion; modeling and rendering of organic materials, which increase scene richness; and cellular textures, which apply particle systems for modeling surface details. Among these, the cellular texture approach is most closely related to our method because our method also uses a particle system.

Grünbaum and Shephard's reference book on visual geometry surveys various aspects of patterns and tiling.[1] Yessios presents methods to generate common materials, such as stones and wood, with 2D line patterns.[2] Miyata proposed an enhanced method for automatically generating 3D stone wall patterns.[3] A limitation of such tiling textures is that regions must be aligned carefully, or visible discontinuities—such as seams and gaps—might be apparent.

Researchers have proposed various methods for modeling and rendering organic materials. In particular, they have applied reaction-diffusion equations—originally proposed as a model of morphogenesis by Turing—to the texture-synthesis problem.[4,5] Fowler et al. have reported a method of generating seashell patterns using reaction-diffusion equations.[6] Worley reported a cellular texture basis function[7] that complements Perlin's noise function.[8]

Fleischer et al. proposed a cellular texture method[9] that can model surface details such as scales, feathers, and thorns. Their method computes the locations, orientations, and other properties associated with cellular particles. After you obtain a hexagonal arrangement of a system of particles by optimization, you convert each particle to a geometric unit with user-defined appearance parameters and then render the detail of each texture.

There are several similarities between these works and ours. These other methods use particle systems, and each particle has energy potential. The energy potential is calculated based on particle-to-particle distance and direction, while the total energy potential is minimized iteratively. By defining interparticle or intercellular forces explicitly, we speed up the cell tessellation process significantly. In Fleischer's cellular texture method, cells are located using an energy-optimization approach, and the energy of each cell is calculated by a cost function consisting of several energy terms—which requires extensive computations.

Our approach, on the other hand, calculates the intercellular force explicitly and solves the equation of motion, a second-order ordinary differential equation, using the fourth-order Runge-Kutta method. Consequently, our method converges very quickly. Most examples using our method require only 10 to 20 seconds to find the cell arrangements with an Intel Pentium III PC.

References

1. B. Grünbaum and G.C. Shephard, *Tiling and Patterns*, W.H. Freeman and Co., 1987
2. C.I. Yessios, "Computer Drafting of Stones, Wood, Plant and Ground Materials," *Computer Graphics*, vol.13, no. 2, 1979, pp. 190-198.
3. K. Miyata, "A Method of Generating Stone Wall Patterns," *Proc. Siggraph*, ACM Press, 1990, pp. 387-394.
4. G. Turk, "Generating Textures for Arbitrary Surfaces Using Reaction-Diffusion," *Proc. Siggraph*, ACM Press, 1991, pp. 289-298.
5. A. Witkin and M. Kass, "Reaction-Diffusion Textures," *Proc. Siggraph*, ACM Press, 1991, pp. 299-308.
6. D.R. Fowler, H. Meinhardt, and P. Prusinkiewicz, "Modeling Seashells," *Proc. Siggraph*, ACM Press, 1992, pp. 379-388.
7. S. Worley, "A Cellular Texture Basis Function," *Proc. Siggraph*, ACM Press, 1996, pp. 191-294.
8. K. Perlin, "An Image Synthesizer," *Proc. Siggraph*, ACM Press, 1985, pp. 287-296.
9. K.W. Fleischer et al., "Cellular Texture Generation," *Proc. Siggraph*, ACM Press, pp. 239-248.

Conclusions

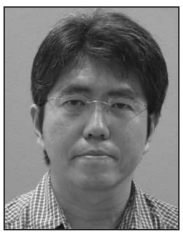
We plan to extend our work into weathering and varied skin conditions. The organic textures presented here are pristine and immutable, even though real ones are not. Some weathering effects[11,12] such as wear, abrasion, and wounds are also important factors in improving the reality of a computer-rendered organic texture. Our method generates a cell's shape with a procedural approach. An interesting consideration for future work is to generate an organic texture from real sample images or to simulate various situations such as wet and dirty skin.[13] ■

References

1. K.W. Fleischer, et al., "Cellular Texture Generation," *Proc. Siggraph*, ACM Press, 1995, pp. 239-248.
2. K. Mehlhorn and S. Näher, *LEDA: A Platform for Combinatorial and Geometric Computing*, Cambridge University Press, 1999, pp. 686-707.
3. K. Shimada and D.C. Gossard, "Bubble Mesh: Automated Triangular Meshing of Non-Manifold Geometry by Sphere Packing," *Proc. 3rd Symp. Solid Modeling and Applications*, ACM Press, 1995, pp. 409-419.
4. K. Shimada, A. Yamada, and T. Itoh, "Anisotropic Triangulation of Parametric Surfaces via Close Packing of Ellipses," *Int'l J. Computational Geometry and Applications*, vol. 10, no. 4, 2000, pp. 417-440.
5. K. Shimada, J. Liao, and T. Itoh, "Quadrilateral Meshing with Directionality Control through the Packing of Square Cells," *Proc. 7th Int'l Meshing Roundtable*, Sandia National Laboratory, 1998, pp. 61-76.
6. N. Viswanath, K. Shimada, and T. Itoh, "Quadrilateral Meshing with Anisotropy and Directionality Control via Close Packing of Rectangular Cells," *Proc. 9th Int'l Meshing Roundtable*, Sandia National Laboratory, 2000, pp. 227-238.
7. D. Doo and M. Sabin, "Analysis of the Behavior of Recursive Division Surfaces Near Extraordinary Points," *Computer Aided Design*, vol.10, no.6, 1978, pp. 356-360.

8. E. Catmull and J. Clark, "Recursively Generated B-Spline Surfaces on Arbitrary Topological Meshes," *Computer Aided Design*, vol. 10, no. 6, 1978, pp. 350-355.
9. C. Loop, *Smooth Subdivision Surfaces Based on Triangles*, master's thesis, Dept. of Mathematics, Univ. of Utah, 1987
10. A. Fournier, D. Fussell, and L. Carpenter, "Computer Rendering of Stochastic Models," *Comm. ACM*, vol. 25, no. 6, 1982, pp. 371-384.
11. J. Dorsey and P. Hanrahan, "Modeling and Rendering of Metallic Patinas," *Proc. Siggraph*, ACM Press, 1996, pp. 387–396.
12. J. Dorsey et al., "Modeling and Rendering of Weathered Stone," *Proc. Siggraph,* ACM Press, 1999, pp. 225-234.
13. H.W. Jensen, J. Legakis, and J. Dorsey, "Rendering Wet Materials," *Proc. 10th Eurographics Workshop on Rendering*, Springer Verlag, 1999, pp. 273-282.

Takayuki Itoh *is a research staff member at IBM Research, Tokyo Research Laboratory. His research interests include mesh generation, surface reconstruction, photorealistic rendering, scientific visualization, and information visualization. Itoh received a PhD in electronics and communications from Waseda University.*



Kazunori Miyata *is a professor in the Center for Knowledge Science at the Japan Advanced Institute of Science and Technology in Ishikawa, Japan. His research interests include texture synthesis, simulation of natural phenomena, computer graphics, multimedia applications, and media art. Miyata received a PhD in computer science from the Tokyo Institute of Technology.*

Kenji Shimada *is a professor in the Department of Mechanical Engineering, the Department of Biomedical Engineering, and the Robotics Institute at Carnegie Mellon University. His research interests include geometric modeling, computational geometry, computer graphics, and computer-assisted orthopedic surgery. Shimada received a PhD from the Massachusetts Institute of Technology.*

Readers may contact Takayuki Itoh at IBM Research, Tokyo Research Laboratory, 1623-14, Shimotsuruma, Yamato-shi, Kanagawa, 242-8502 Japan; itot@computer.org.