Several Approaches to Texture Mapping

Jesse Reiter

September 12, 1995

1 Introduction

Texture mapping is one of the most successful techniques used in computer graphics. It was first introduced by Catmull ([Cat74]) in 1974 as a means of enhancing the realism of rendered images and boosting their richness without affecting the complexity of their underlying geometry. Texture mapping has since become so prevalent that untextured images seem plastic and boring.

The general idea behind texture mapping is based on a simple principle. Rather than model the microfacet geometry of the texture on a surface in all its minute detail, *shrink-wrap* a planar picture of the texture around the object. Though simple in principle, this shrink wrapping process is unfortunately complicated in practice. It invariably introduces spatial distortions to the texture patterns, which may considerably alter its visual appearance on the surface.

Following a somewhat more detailed description of the general problem, this paper will discuss two methods of overcoming the difficulties. The methods approach different aspects of the mapping process, and hence are not closely related. They are complementary, however, and together offer a cohesive solution. The first method discussed deals with the acquisition of textures, and introduces a special class of them known as *Reaction-Diffusion Textures.* These textures are procedurally synthesized, and thus afford a means to pre-compensate for surface distortion. The second method deals with the actual fitting of a texture onto a surface. It utilizes differential geometric methods to reduce the distortion incurred by preserving geodesic curvature on the surface through the mapping.

2 General Problem

An abstract definition of textures is provided in [FFC87]. Briefly, a *texture* is a correspondence $\tau : \mathbf{R}^2 \rightarrow C$ between a region in the plane and a finite set of colors (or intensities) C, whereby each point in the region is assigned a color value. In practice, a texture is just a discrete image, which amounts to a two-dimensional array of color values. The object of texture mapping is to assign these values to every point on the surface in a convincing manner. Toward that end a correspondence must be found between the surface and the region on the plane, whereby the region to be wrapped nicely around the surface.

Traditional texture mapping is performed in three stages. In the first, the texture is procured. This is usually done by either optically scanning a photograph, by manufacturing it manually using some drawing program, or by synthesizing the image procedurally. In the second stage, the correspondence between the 3D surface and the 2D texture is established, assigning a texture coordinate (u, v) to each surface point (x, y, z). This correspondence is called the *texture map*. Finally, the third stage assigns a color value to every surface point depending on the surrounding neighborhood of the corresponding texture point. This paper will assume that the third stage is straightforward and direct, even though in practice it is not. The color sampling generally produces undesired and inconsistent artifacts, and must be subjected to various antialiasing filtering techniques.

The texture map is nontrivial, as mapping a planar texture onto a 3-dimensional surface without distortion is essentially equivalent to a non-warped flattening of the surface. In most cases doing so is impossible. In fact, the small class of surfaces which can undergo a non-warped flattening has received a special name: such surfaces are called *developable surfaces*. By definition, then, fully spreading out a non-developable surface will necessarily introduce distortions. The ultimate goal of texture mapping is to minimize these.

More formally, this goal may be stated as follows. Given a surface $S \in \mathbf{R}^3$ with an area metric $\mathcal{A}_3 : S \rightarrow \mathbf{R}$ and a planar area metric $\mathcal{A}_2 : \mathbf{R}^2 \rightarrow \mathbf{R}$, we seek the following goal:

Goal. *Find a mapping $m : S \rightarrow \mathbf{R}^2$ s.t. for any small region $R \subset S$, $\mathcal{A}_3(R)/\mathcal{A}_2(m(R)) \approx k$ for some constant k.*

In other words, the goal is to find a map that for any small region on the surface with constant surface area A, its image A' on the plane is approximately constant.

3 Reaction Diffusion Textures

In 1952, Alan Turing ([Tur52]) proposed a chemical reaction he dubbed *Reaction-Diffusion* to account for pattern formation in biological morphogenesis. Turing postulated that specific chemicals – the so called *morphogens* – diffuse through the tissue and react with each other to produce other morphogens. This interaction process, he claimed, eventually settles down to form stable morphogen patterns which, in turn, excite and inhibit specific cell development in the embryo.

The Reaction-Diffusion process can be modeled with a system of non-linear partial differential equations which describe the rate of change of a chemical's concentration at one given point as a function of its current concentration there and that of others in neighboring points. This non-linear model, however, is extremely difficult to analyze, and thus the behavior of such systems has been studied empirically using numerical simulation. Representing different chemicals as different colors, and concentrations as intensities, these simulations produce lively images on standard graphics hardware.

3.1 Basic Form

The basic form of the RD model involves three processes: reaction, diffusion, and dissipation. Reaction controls the rate of morphogen creation. Diffusion controls the transport of morphogens from points of higher concentration to points of lower concentration. Dissipation governs the breakdown of morphogens causing them to decay exponentially to zero in the absence of other influences. The dissipation term is provided mainly as a control term to ensure reasonable behavior of the system.

As mentioned above, the concentration of each morphogen is a function of position and time. The variation in concentration of a given morphogen C is modeled by the *Reaction-Diffusion equation* ([WK91])

$$\frac{\partial C}{\partial t} = \alpha_C^2 \nabla^2 C - \beta_C C + R_C, \tag{1}$$

where $\nabla^2 C$ is the 2-dimensional *Laplacian* of C and is defined as

$$\nabla^2 C = \frac{\partial^2 C}{\partial x^2} + \frac{\partial^2 C}{\partial y^2}.$$

The constants α_C and β_C control the rate of the diffusion and dissipation respectively. The reaction term, R_C, is a function which generally depends on the other abutting morphogens. The Laplacian is a measure of the concentration of the morphogen C at one point relative to its concentration nearby. If the concentration is higher nearby then $\nabla^2 C$ will be positive, and C will diffuse toward this point. If the concentration is lower nearby then $\nabla^2 C$ will be negative and C will diffuse outward in the direction of the lowest concentration. As the primary concern of this paper is generating surface textures, the diffusion process is modeled in a 2D medium. The planar model, however, easily scales to other dimensions: Stam and Fiume ([SF93]) use a similar 3D model to simulate wind fields.

In practice, the model is discretized and represented as a 2D array of concentrations. Using a Taylor expansion about the point $C_{i,j} = C(x, y)$ and finite differences, an estimate of the second partial derivative is obtained:

$$\frac{\partial^2 C}{\partial x^2} \approx \frac{C_{i+1,j} + C_{i-1,j} - 2C_{i,j}}{h^2},$$

where h is the distance between two consecutive points. Using this, the discrete Laplacian $\nabla^2 C$ is estimated by

$$\nabla^2 C_{i,j} \approx \frac{C_{i+1,j} + C_{i-1,j} + C_{i,j+1} + C_{i,j-1} - 4C_{i,j}}{h^2}. \tag{2}$$

The discrete Laplacian can be succinctly expressed as a 3x3 convolution mask acting on the concentration array:

$$L = \frac{1}{h^2} \begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}, \tag{3}$$

where the entries in the mask are just the coefficients of (2). Multiplying L by α_C^2 and including the term $-\beta_C C_{i,j}$ yields the convolution mask

$$M_C = \frac{1}{h^2} \begin{pmatrix} 0 & \alpha_C^2 & 0 \\ \alpha_C^2 & -4\alpha_C^2 - h^2\beta_C & \alpha_C^2 \\ 0 & \alpha_C^2 & 0 \end{pmatrix}. \tag{4}$$

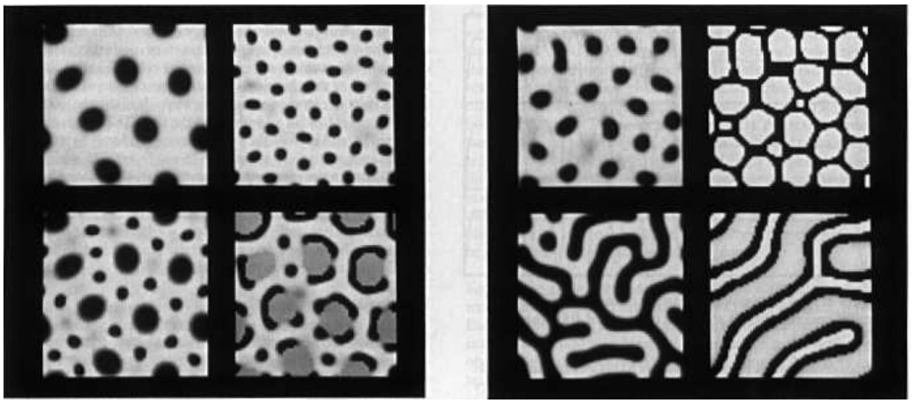

Figure 1: Examples of RD textures.

The mask M can now be incorporated into the RD model,(1), yielding the compact form

$$\dot{C} = M_C * C + R_C,$$

where '$*$' is the discrete convolution operator.

The actual computation of C requires integrating $\dot{C}$ through time. A simple method of integration known as *Euler's method* may be applied, yielding

$$C_{t+\Delta t} = \Delta t(M_C * C_t + R_{C_t}), \tag{5}$$

for the basic time-step Δt.

To obtain a stable solution, equation (5) must be iterated sufficiently for all morphogens until the entire system converges. Finite-time convergence is not guaranteed. If and when it does indeed occur, the particular solution reached depends on the reaction functions, R_C, the rate constants, α_C and β_C, and, of course, the number of morphogens. Using this method, Turk ([Tur91]) and Witkin and Kass ([WK91]) have generated an assortment of different spot, and stripe patterns similar to those seen on zebras, leopards, giraffes, and various tropical fish. Figure 1, which appears in [Tur91], shows some examples of these.

3.2 Extensions

Extensions to this model have been developed to allow for more interesting textures. Preserving the simplicity of the original model, Turk has suggested a cascading process of generating the more complex patterns. Under this process, various areas of the concentration arrays are periodically "frozen" in time, while the parameters are altered for the remaining areas. In this way he has obtained spots and stripes of various sizes and intensities on the same texture.

Witkin et al. use a slightly more complicated method by permitting anisotropic diffusion. Allowing a morphogen C to diffuse at different rates in different directions, they define the diffusion matrix

$$D = \begin{pmatrix} a_1^2 \cos^2\theta + a_2^2 \sin^2\theta & (a_2^2 - a_1^2)\cos\theta\sin\theta \\ (a_2^2 - a_1^2)\cos\theta\sin\theta & a_2^2\cos^2\theta + a_1^2\sin^2\theta \end{pmatrix}, \tag{6}$$

where a_1 is the rate of diffusion in the principle direction $(\cos\theta, \sin\theta)$, and a_2 is the rate of dissipation in the principle direction $(-\sin\theta, \cos\theta)$. With this they derive a new convolution mask M' to replace M in (5):[1]

$$M' = \frac{1}{h^2}\begin{pmatrix} d_{12} & d_{22}-d_{12} & 0 \\ d_{11}-d_{12} & 2(d_{12}-d_{11}-d_{22})-h^2 b & d_{11}-d_{12} \\ 0 & d_{22}-d_{12} & d_{12} \end{pmatrix},$$

where the $d_{i,j}$ terms are the elements of the diffusion matrix D. Using the new convolution mask, they introduce the concept of *space-varying* diffusion by allowing the diffusion matrix, D, to vary with position. Results of this method can be found in [WK91].

3.3 Mapping Reaction-Diffusion Textures

The algorithmic synthesis of reaction-diffusion textures endows them with two primary strengths over external (*e.g.*, scanned) textures. One, a large variety of RD textures can be stored in a relatively small space, as all that is required is the set of various parameters necessary to synthesize the textures. An added benefit of the generation process is that it permits the introduction of stochastic noise, and hence can easily generate a variety of very similar, *non*identical textures. This could be useful for rendering scenes with multiple animals, for it will ensure their skins are recognizably similar yet sufficiently distinct from one another, avoiding unrealistic cloned appearances.

The second strength of RD textures is that they lend themselves easily to non-distorted mapping. One such method of mapping these textures is detailed by Witkin et al. in [WK91], where in order to compensate for the apparent distortion on the object, the texture is pre-warped in the opposite direction. Parametric stretch and shear are corrected by altering the diffusion matrix D (of equation 6) based on the surface's *Jacobian* matrix.

A different, more general method is developed by Turk in [Tur91]. The basis of his method is the generation of a new canonical surface mesh on top of any polyhedral model representing the surface geometry. The advantage of this mesh, and what makes it particularly endearing, is that all the mesh cells are of approximately equal size[2]. The mesh generation process proceeds roughly as follows. Depending on the desired density of the cells, an appropriate number of points is randomly distributed upon the surface of the model. Using a relaxation procedure, these points are then shuffled across the surface until they are relatively equidistant from one another. Finally, the Vornoi regions of the points is computed, producing the final cells. Having produced this mesh, Turk grows the RD texture directly upon the surface,

[1] The actual mask cited in [WK91] is

$$M' = \frac{1}{2h^2}\begin{pmatrix} -d_{12} & 2d_{22} & d_{12} \\ 2d_{11} & -4(d_{11}+d_{22})-2h^2 b & 2d_{11} \\ d_{12} & 2d_{22} & -d_{12} \end{pmatrix},$$

but I have been unable to reproduce this result of theirs.

[2] This may be useful for overcoming a large obstacle of general texture mapping (recall the original goal). These cells could be created sufficiently small so as to be considered planar; essentially requiring no more than an identity mapping from texture space to object space. We leave this, however, for future work.

thus eliminating the need for mapping – and, hence, for overcoming distortion – altogether. As the mesh is no longer a regular, rectangular grid, the discrete Laplacian of the texture generation process (equation 2) is generalized by Turk to account for the irregularity.

4 Surface Flattening for General Texture Mapping

Although procedural textures are useful in some applications, their service is largely limited. The majority of textures are acquired digitally, precluding the use of creative synthesis techniques, hence a more general non-distorted mapping method is required. It must be pointed out that no *grab-all* map exists, and that various maps outperform others in different situations. The technique described below attempts to strike a balance between texture distortion and texture discontinuities, and is based on the work of Bennis et al. in [BVI91].

The basic idea of the method is to flatten a surface along some parametric lines, keeping the incurred distortion beneath some tolerance level. The flattening method used is aimed at the preservation of the differential-geometric notion of *geodesic curvature*, and heavily relies on a preservation operator $\mathcal{P}$ which will presently be defined.

4.1 Preliminaries

We begin with a brief review of some necessary concepts of differential geometry, details of which can be found in [Far93] and [dC76]. Let $C \subset \mathbf{R}^3$ be a space curve given by the arclength parameterization

$$C := \{\mathbf{x}(s) \in \mathbf{R}^3 \quad | \quad s \in [a, b] \subset \mathbf{R}\},$$

where the Cartesian coordinates (x, y, z) of $\mathbf{x}$ are each differentiable functions of the arclength s.

The *Frenet frame* is a tool used in differential geometry to describe the local behavior of a curve in terms of a local coordinate system at each point. The Frenet frame $(\mathbf{x}, \mathbf{t}, \mathbf{m}, \mathbf{b})$ is a frame centered at the point $\mathbf{x} \in C$, whose axes $(\mathbf{t}, \mathbf{m}, \mathbf{b})$ are given by the formulas

$$\mathbf{t} = \mathbf{x}', \qquad \mathbf{m} = \frac{\mathbf{x}''}{\|\mathbf{x}''\|}, \qquad \mathbf{b} = \mathbf{t} \wedge \mathbf{b},$$

where primes denote arclength derivatives, and the wedge '$\wedge$' denotes the cross product. The axes have received intuitive names: $\mathbf{t}(s)$ is called the *tangent vector*, $\mathbf{m}(s)$ is called the *main normal vector*, and $\mathbf{b}(s)$ is called the *binormal vector* vector of the curve at s. It is not difficult to show that the variation of the Frenet frame is given by the *Frenet-Serret* formulas:

$$\begin{cases} \mathbf{t}' &= \kappa \mathbf{m} \\ \mathbf{m}' &= -\kappa \mathbf{t} - \tau \mathbf{b} \\ \mathbf{b}' &= -\tau \mathbf{m}. \end{cases}$$

The variables $|\kappa(s)| = \|\mathbf{t}'(s)\|$ and $|\tau(s)| = \|\mathbf{b}'(s)\|$ (recall that $\mathbf{m}$ has unit length) are called the *main curvature* and *torsion* at s respectively.

Now consider the parametric surface $S \subset \mathbf{R}^3$ given by the bivariate parameterization

$$S := \{\mathbf{x}(u,v) \in \mathbf{R}^3 \quad | \quad (u,v) \in [a,b]^2 \subset \mathbf{R}^2\},$$

where the Cartesian coordinates of the surface points are differentiable functions of both parameters. A surface is called *orientable* if it makes sense to define an inside and an outside direction[3]. We will restrict ourselves to smooth orientable surfaces, and let $\mathbf{n}(u,v)$ be the unit outward-pointing *normal vector* to S at $\mathbf{x}(u,v)$. The *tangent plane*, $\mathbf{T_p}$, of S at $\mathbf{x}$ is spanned by the set of points $\mathbf{y} \in \mathbf{R}^3$ satisfying $(\mathbf{x}-\mathbf{y}).\mathbf{n} = 0$, where the dot denotes the *dot product*.

Let $C_S \subset \mathbf{R}^3$ be a curve on the surface S, given by arclength parameterization $\mathbf{x}(s)$. Its second arclength derivative $\mathbf{x}'' = \mathbf{t}'$, the acceleration of a particle moving along the curve, can be separated into two components. One, $\mathbf{t}'_n$, is normal to the surface; the other, $\mathbf{t}'_g$, is tangent to it:

$$\begin{aligned} \mathbf{t}'_n &= (\mathbf{t}'.\mathbf{n}).\mathbf{n} \\ \mathbf{t}'_g &= \mathbf{t}' - \mathbf{t}'_n. \end{aligned}$$

Looking from a viewpoint on the surface, only the tangential component can be seen. Based on this fact we define a special kind of curvature for surface curves which is analogous to the notion of (plain) curvature of planar curves.

Definition. *The* geodesic curvature κ_g *of a curve* C_S *belonging to a surface* S *at the point* $\mathbf{x}$ *is the norm of the tangential acceleration of the curve at that point:*

$$|\kappa_g| = \|\mathbf{t}'_g\|.$$

Note that the the *geodesic curvature* κ_g is not the same as the *main curvature* κ which contains the normal component as well.

Using geodesic curvature, the concept of a straight line can be extended from planes to surfaces. The curve C_S is said to be *geodesic at a point* $\mathbf{x}$ if κ_g vanishes there, and is called a *geodesic* if it is geodesic everywhere. That is, a surface curve is a geodesic if its acceleration is restricted to the "up" direction at every point. For an observer on the surface, a particle moving along a geodesic curve will appear to be moving in a straight line. Just as straight lines provide the shortest path between two points on a plane, geodesics provide the shortest path between two points on a surface. Moreover, if C_S is projected onto the surface tangent plane $\mathbf{T_p}(\mathbf{x})$ at a point $\mathbf{x}$, the projection curve C_{T_P} (on $\mathbf{T_p}(\mathbf{x})$) will be locally straight in a neighborhood of $\mathbf{x}$ if and only if C_S is geodesic at $\mathbf{x}$. With the above notation, this claim is stated without proof in [BVI91] as the following lemma:

Lemma. *The geodesic curvature* κ_g *of the curve* C_S *at the point* $\mathbf{x}$ *is equal to the curvature at* $\mathbf{x}$ *of the planar curve* C_{T_P} *obtained by projecting* C_S *to the surface tangent plane at* $\mathbf{x}$.

[3] An example of an orientable surface is a sphere. The classic example or a non-orientable surface is the Möbius strip.

4.2 Faithful Surface Curve Projection

In practice, surfaces are sampled and surface curves are given by chord-line approximations whose vertices lie at the sampled points. Arclength is approximated by chordlength, given as the Euclidian distance $d(\cdot,\cdot)$ between two points. In what follows, let $S \subset \mathbf{R}^3$ be the sampled surface which is to be flattened, and let $F \subset \mathbf{R}^2$ be the flattening plane. We will call the planar curve $C_F \subset \mathbf{R}^2$ a *faithful representation* of the surface curve $C_S \subset \mathbf{R}^3$ if the curvature at each vertex on the chord-line C_F is identical to the geodesic curvature at the corresponding point on C_S, and if the chordlengths of the corresponding segments on both curves agree.

The previous lemma offers a method for mapping a chord-line on a surface to a chord-line on a plane. The geodesic curvature of a curve can be preserved through the projection process, by using a projection map that respects the curvature of the local projection of the curve on the tangent plane at each point. The lemma provides the basis for just such an operator, which in addition to respecting curvature preserves distances between neighboring points on isoparametric curves.

The operator is parameterized by two corresponding pairs of points, one on the surface, the other on the plane. Suppose $M_0, M_1, M_2 \in S$ are three neighboring surface points and that $P_0, P_1 \in F$ are the points on the flattening plane that correspond to the first two surface points. Ideally we would like the operator $\mathcal{P}_{P_0,P_1}^{M_0,M_1}$ to take the point $M_2 \in S$ and produce the point $P_2 \in F$ that simultaneously satisfies the two conditions:

$$\begin{array}{lrcll} (i) & \angle P_0P_1P_2 & = & \angle M_0M_1M_2, & \quad \text{and} \\ (ii) & d(P_1,P_2) & = & d(M_1,M_2), & \end{array}$$

where $\angle PQR$ is the angle between the chords $\overline{PQ}$ and $\overline{QR}$.

Denoting by $\mathbf{T_{P}}_i$ and $\mathbf{n}_i$ the tangent plane and surface normal at the point M_i, we define the operator $\mathcal{P}_{P_0,P_1}^{M_0,M_1} : S \to F$ which sends $M_2 \mapsto P_2$ by the following algorithm (see figure 2):

Algorithm (preservation operator):

1. *Project M_0 and M_2 onto $\mathbf{T_{P}}_1$, providing the points $\widetilde{M}_0$ and $\widetilde{M}_2$ given by:*

$$\widetilde{M}_i = M_i + ((M_1 - M_i).\mathbf{n}_1).\mathbf{n}_1, \qquad i \in \{0,2\}.$$

2. *Dilate $\widetilde{M}_2$ in $\mathbf{T_{P}}_1$ giving $\widehat{M}_2$ such that $d(M_1,M_2) = d(M_1,\widehat{M}_2)$:*

$$\widehat{M}_2 = M_1 + \frac{\|M_2 - M_1\|}{\|\widetilde{M}_2 - M_1\|}(\widetilde{M}_2 - M_1).$$

3. *Define a right handed local, orthogonal coordinate system $(M_1, \widehat{e}_1, \widehat{e}_2)$ on $\mathbf{T_{P}}_1$ whose $\widehat{e}_1$ axis is in the direction $M_1 - \widetilde{M}_0$, and let (ξ,η) be the coordinates of $\widehat{M}_2$ in that system. Then P_2 is the point with coordinates (ξ,η) in the local coordinate system (P_1, e_1, e_2) on F whose e_1 axis is in the direction $P_1 - P_0$:*

$$P_2 = P_1 + (\widehat{x}_1 e_1 + \widehat{x}_2 e_2),$$

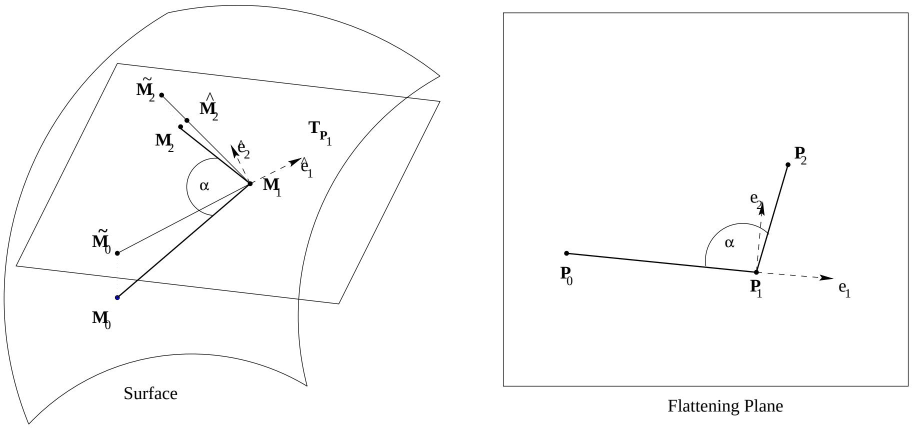

Figure 2: The preservation operator $\mathcal{P}$

where the axes $\widehat{e}_1, \widehat{e}_2, e_1, e_2$ and the coordinates (ξ, η) are given by

$$\begin{array}{rcl}
\widehat{e}_1 & = & \frac{M_1 - \widetilde{M}_0}{\|M_1 - \widetilde{M}_0\|} \\
\widehat{e}_2 & = & \mathbf{n}_1 \times \widehat{e}_1 \\
e_1 & = & \frac{P_1 - P_0}{\|P_1 - P_0\|} = a\widetilde{\imath} + b\widetilde{\jmath} \\
e_2 & = & -b\widetilde{\imath} + a\widetilde{\jmath} \\
\xi & = & (\widehat{M}_2 - M_1).\widehat{e}_1 \\
\eta & = & (\widehat{M}_2 - M_1).\widehat{e}_2
\end{array}$$

Given a chord line approximation of a surface curve, the operator $\mathcal{P}$ may be used to produce a planar projection that preserves chord-length and geodesic curvature. Suppose $M_0, \ldots, M_n \in S$ are $n+1$ surface points defining a curve which is to be projected onto the plane F, producing the planar curve $P_0, \ldots, P_n \in F$. Projecting the segment M_0M_1 requires nothing more than selecting a starting point P_0 and a direction, and placing P_1 at a distance of $d(M_0, M_1)$ from P_0 in the desired direction. The remaining remaining flattened points are given by

$$P_i = \mathcal{P}^{M_{i-2}, M_{i-1}}_{P_{i-2}, P_{i-1}}(M_i), \qquad i = 3, \ldots n.$$

Since $d(P_0, P_1) = d(M_0, M_1)$, and by step (2) of the algorithm $d(P_{i-2}, P_{i-1}) = d(M_{i-2}, M_{i-1})$ for $i = 3, \ldots, n$, all arclengths are preserved. For a sufficiently refined sampling, the tangents to the curve are approximated by the chord segments, so the angles between successive chord segments on the tangent-plane projection $\mathbf{T}_{\mathbf{P}_i}$ approximate the curvature of the curve. The geodesic curvature at points $M_1, \ldots, M_{n-1}$ is thus preserved according to the lemma, and hence the flattened curve is a faithful projection of the surface curve.

4.3 Surface Flattening

Bennis et al. propose several methods for piecewise surface flattening using the flattening operator $\mathcal{P}$. The general strategy, common to all methods, is to define a distortion metric, and then flatten a region until the total distortion incurred in it reaches some threshold level.

When the threshold is surpassed the region is "closed" and flattening is resumed on a new region. In this way distortion is minimized inside the regions at the expense of introducing texture discontinuities between adjacent ones.

Regions are flattened by successively flattening sets of surface curves that cover them. Each individual region is *grown* by selecting an internal isoparametric curve and flattening it faithfully. Transversal isoparametric curves are developed in the same manner. Since flattening non-developable surfaces necessarily introduces distortion, the curves parallel[4] to the original isoparametric one become distorted as neither their geodesic curvature nor their arclength can be controlled. The various methods presented in [BVI91] differ in the techniques they use to alleviate this warping, and the distortion metrics used in conjunction with them.

The most interesting method of development entails the flattening of the curves parallel to the original one. Contrary to the original curve, the flattening of these is not guaranteed to preserve curvature and arclength. It proceeds as follows. Suppose that $C'_S \subset S$ is a surface curve parallel to the original one, and that $M \in C'_S$ is a point on that curve. Assuming that S is sampled over a regular rectangular grid, M has four neighbors $N_0, \ldots, N_3$. Each of these is surrounded by three additional neighboring points N_i^0, N_i^1, N_i^2; the fourth one N_i^3 being M (see figure 3). For the time being let us suppose that all these neighboring points have been previously projected onto the flattening plane F, so that $Q_i \in F$ corresponds to $N_i \in S$ and that $Q_i^j \in F$ corresponds to $N_i^j \in S$. Each of the neighbors N_i provides three different methods of applying the preservation operator $\mathcal{P}$ to M, producing the three (perhaps identical) points $\bar{P}^{ij} \in F$:

$$\bar{P}^{ij} = \mathcal{P}_{Q_i^j, Q_i}^{N_i^j, N_i}(M), \qquad j = 0, 1, 2.$$

We call these *candidate points*. Applying $\mathcal{P}$ in this way to all four neighbors of M produces 12 candidate points $\bar{P}^{ij}$, ($j \in \{0,1,2\}, i \in \{0,1,2,3\}$). Because the preservation operator requires many points Q_i and Q_i^j on the projection plane F that may not have been evaluated, the desired point $P = \mathcal{P}(M)$ is given as the centroid of all those candidates $\bar{P}^{ij}$ that were indeed available at the time of preservation.

For a developable surface S all the candidate points coincide, preserving the geodesic curvature and arclengths of both isoparametric curves of S intersecting at M. In general, however, this will not occur, and distortion will be introduced. Moreover, this distortion will be compounded as the distance between the parallel curves and the original one increases.

Two metrics for evaluating this distortion are suggested. Under the first metric, the distortive effect of the flattening is measured by comparing the arclengths of the surface curve segments with that of their projected counterparts, and summing up the discrepancies. With $\{M_i\} \subset C'_S$ being the set of consecutive points on the curve C'_S, and $\{P_i\} \subset F$ being their projection, the arclength distortion metric $\mathcal{D}_{\mathcal{A}}$ is given by

$$\mathcal{D}_{\mathcal{A}}(C'_S) = \sum_i \frac{|d(M_i, M_{i+1}) - d(P_i, P_{i+1})|}{d(M_i, M_{i+1})}.$$

Clearly a low value of $\mathcal{D}_{\mathcal{A}}$ indicates a low discrepancy between the corresponding arclengths, and a less distorted projection.

[4] In parameter space.

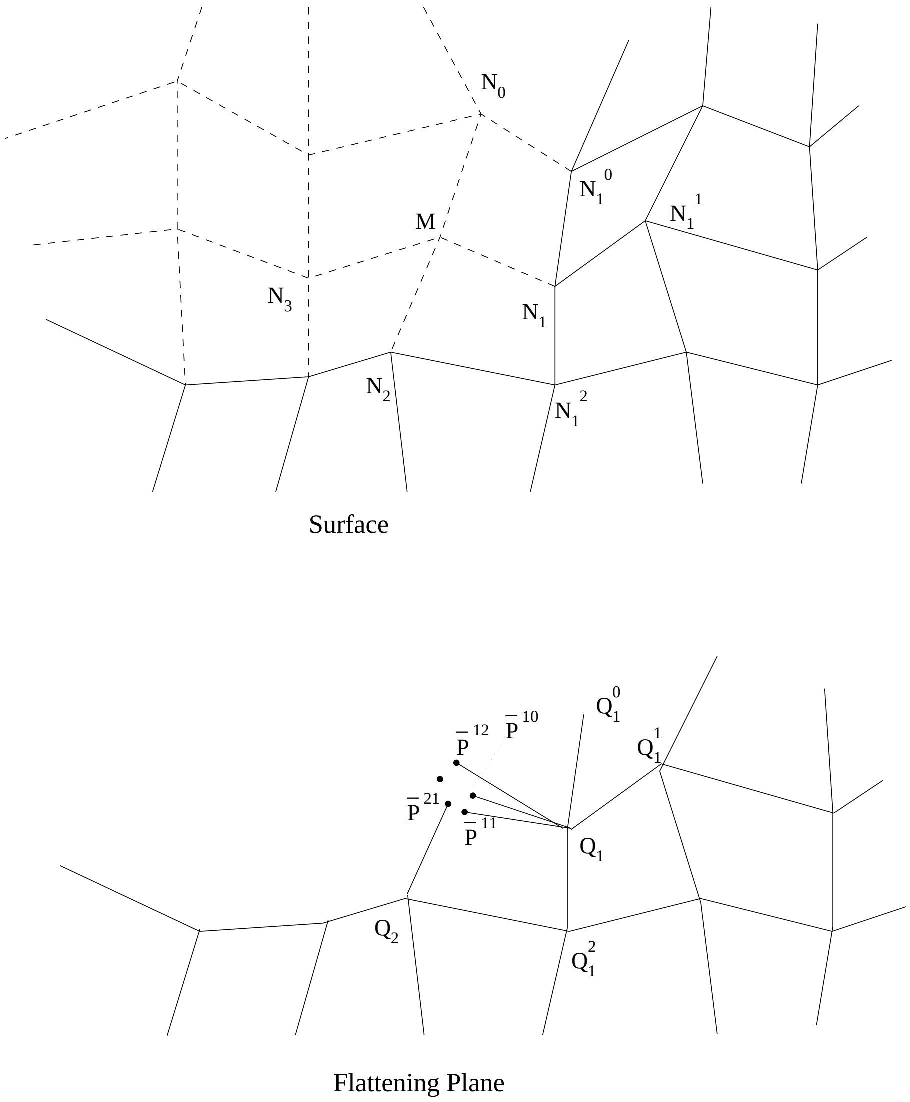

Figure 3: Flattening surfaces using the preservation operator $\mathcal{P}$

The second metric $\mathcal{D}_{\mathcal{D}}$ measures the distortive effect of the flattening by determining how widely dispersed the candidate points are. For a cluster of n coplanar points $\{P_k\}$ sharing the plane with a point P, one says that the *dispersion of the cluster around* P is given by

$$\operatorname{dsp}(P) = \frac{1}{n} \sum_{k=1}^{n} \|P - P_k\|^2.$$

The value of $\operatorname{dsp}(P)$ is minimized when P is at the centroid of the cluster, in which case $\operatorname{dsp}(P)$ is called the *dispersion of the cluster*. With $\{M_i\} \subset C'_S$ and $\{P_i\} \subset F$ as above, the metric $\mathcal{D}_{\mathcal{D}}$ is given as the cumulative dispersion of the candidate points $\{\bar{P}^{ij}\}$ defining each point in the projection:

$$\mathcal{D}_{\mathcal{D}}(C'_S) = \sum_i \operatorname{dsp}(P_i).$$

The denser the candidate clusters are, the lesser the value of their dispersion, and the better the flattening achieved of S.

4.4 Mapping Textures on Composite Surfaces

The construction of surfaces using piecewise collections of surface patches requires special care on boundary curves to ensure smooth visual interpolation. Texture cohesion must also be maintained along these curves in order to keep the boundaries from showing through the composite surface. This cohesion may be attained by using the boundary of each patch as the initial flattening curve. The boundary will thus be projected with no distortion, ensuring a visually smooth transition of textures from one patch to the next. As the flattening scheme relies heavily on surface normals, it is necessary that the tangent planes of adjacent patches coincide along the patch boundaries. In other words, C^1 continuity between adjacent patches is required.

Once the patch boundary has been faithfully flattened, the patch interior must be developed in a consistent fashion. In lieu of using the parallel curves described in section 4.3, concentric rings of isoparametric curves are used for this matter. Because the direction of development flows inwardly from the patch boundary, each point on an internal ring is given as the centroid of all the candidates formed by the points directly behind it. The number of contributing candidates depends on the parameterization of the patch and on the location of the point. For example, points on a corner[5] of a ring are defined by candidates formed by both isoparametric curves, and hence have more contributors than points defined by candidates formed from one curve only. Both distortion metrics $\mathcal{D}_{\mathcal{A}}$ and $\mathcal{D}_{\mathcal{D}}$ may be used to determine when an acceptable distortion level has been surpassed. Once this level has been exceeded, a discontinuity is introduced to the texture by halting the development process and faithfully flattening the next concentric ring. Development is then resumed as before until the center of the patch is reached and the entire patch has been flattened.

[5] A *corner* of an isoparametric ring is a point that simultaneously lies on two of the isoparametric curves comprising the ring.

5 Implementation

Both texturing techniques discussed in this paper have been implemented to various degrees of success. Because the relative simplicity of the RD-texture generation algorithm is intrinsically coupled with the high time-complexity of its iterative nature, an abortive attempt was made to develop a parallel implementation of the algorithm. Having failed to successfully complete the parallelization due to multiple hardware problems, procedural texture synthesis was dropped in favor of piecewise surface flattening. The latter has been implemented successfully on triangular surface patches. Implementation notes for both projects are discussed below.

5.1 Parallel Reaction Diffusion: A Worst Case Scenario With a Healthy Dose of Bad Luck

The RD texture generation algorithm calls for convolving an array of morphogen concentrations with some convolution mask. In practice this necessitates the evaluation of a new concentration array for each time step. Each cell in the new array is evaluated as a linear combination of neighboring cells in the previous array. The array for time t must thus be evaluated in its entirety before the array for time $t + \Delta t$ can be touched. However, because the value of each cell is completely determined by the values of neighboring cells in the *previous* array and is influenced by none of its *current* neighbors, the cells of time $t + \Delta t$ may be evaluated concurrently. The generation process was thus made parallel by subdividing the new concentration array into strips of equal size, and treating a time step of the RD system on each strip as an independent process.

The multiprocessor used for this implementation was an ISG Pulsus with 16 processors accessing a global shared memory. The processors sit on two boards – eight on each – while a single board holds the shared memory. Global memory on the Pulsus may be cached or uncached depending on the memory address used. Uncached access to memory is provided by altering one bit in the address for each access, using a supplied macro. The ISG box is driven by a Sun Sparc station: programs are run on the Sun, and may delegate tasks to be performed on the Pulsus. All video output is done through the Pulsus.

The implementation made use of a two chemical system. In each time step the main program cut up each concentration array into sixteen strips, sent each strip as a separate thread to the ISG for evaluation, and waited for all threads to rejoin. When they converged, the concentration of both chemicals had been established and the results were rendered. Rendering consisted of assigning each chemical a color, with the concentration of each chemical governing the intensity of the respective color.

The preliminary results of this implementation were favorable, although additional investigation was desirable. Unfortunately, before this work could be completed an unfortunate sequence of hardware failures ensued resulting in the loss of all the code for the implementation. The project was subsequently aborted.

A possible alternative method of accelerating the synthesis of Reaction-Diffusion textures is to use a multi-grid approach. Using this approach, the RD texture is generated initially on a coarse grid, which is then incrementally refined. Each step in the refinement generates a grid of finer detail than that of the previous level. The RD process is then iterated on the finer grid until a stable solution is obtained, whence a further refining step is taken. The entire process is terminated once a sufficient refinement has been obtained.

5.2 Piecewise Surface Flattening of Triangular Bézier Patches

The advantage of defining texture maps over piecewise surfaces is that local changes in surface geometry generally require only local changes to the texture map. When a texture mapping process is unaware of the underlying surface representation, it cannot disregard the regions unaffected by the change and must remap the texture everywhere. Context sensitive maps, however, can determine when no changes are necessary along a patch boundary, and absolve the interior from remapping with no further ado.

The method of piecewise surface flattening has been implemented on triangular Bézier patches. The code was written in C++ using OpenGL, and is designed to run on an SGI under IRIX 5.3. Surfaces flattening is carried out in accordance with the algorithm outlined in section 4.4, and is used for assigning texture coordinates to surface points. The implementation of the patches themselves has been taken from previous work which will be discussed at length in the forthcoming thesis [Rei96].

Although nominally the texture is parameterized over the unit square, the texture *space* is regarded as a torus giving the texture an infinite extent. Texture coordinates outside of the unit square are reduced to their congruence class modulo 1, giving interior coordinates. The texture is then drawn onto the surface by interpolating between the mapped points using OpenGL's 2-dimensional texture mapping facilities `glTexImage2D*()`, `glTexEnv*()`, and `glTexCoord2*()`.

Flattening of the patches is performed after the patch is completely sampled, but before it is rendered. The sampling process consists of evaluating the patch over a sampled parameter domain (the so called *domain triangle*), and maintaining the list of sampled points in an array. As the preservation operator $\mathcal{P}$ requires surface normals, an additional corresponding array is maintained to house the normal vectors at each sampled surface point. The process of evaluating a surface point produces, as a byproduct, three non-collinear points on the tangent plane (as described in [Far86, Far93, HL93]). These three points *span* the tangent plane, and hence define the surface normal vector at the sampled point. If $\mathbf{a}, \mathbf{b}, \mathbf{c} \in \mathbf{T_p}$ are the three points, the normal $\mathbf{n}$ is given by

$$\mathbf{n} = \frac{(\mathbf{a} - \mathbf{b}) \wedge (\mathbf{c} - \mathbf{b})}{\|(\mathbf{a} - \mathbf{b}) \wedge (\mathbf{c} - \mathbf{b})\|}.$$

As long as the three points are taken in a consistent order, the direction of the normals will be consistent.

For the algorithm of section 4.4, each point on an internal ring has two neighbors on each isoparametric curve behind it, giving a total of two neighbors for a standard point, four for a corner, and sometimes six for the center point. Each of the neighbors gives rise to one candidate, the creation of which requires two previously flattened points. One such point is supplied by the neighbor. The other chosen is the one on the external ring which immediately follows the neighbor (see figure 4). The accumulated distortion is calculated using the dispersion metric $\mathcal{D}_{\mathcal{D}}$.

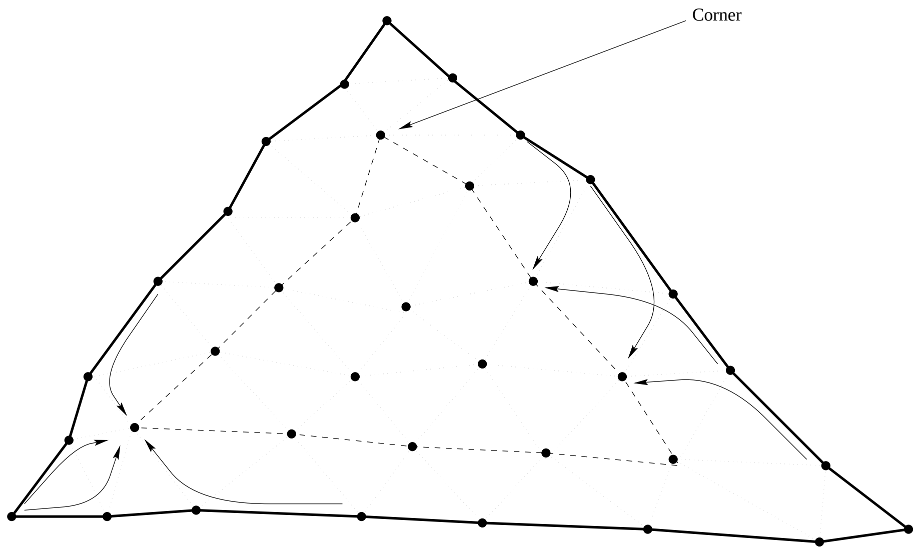

Figure 4: Flattening Bézier triangles: The thick line indicates the patch boundary that has been faithfully preserved; the dashed line indicates the concentric ring currently being flattened. The curved arrowheads show the formation of candidates for the interior points.

For the purpose of demonstration, figure 6 shows a Bézier triangle with a checkerboard texture that was mapped onto it using the preservation algorithm. Compare the results with figure 7, in which the texture coordinates assigned to each sampled point were the domain parameters. Figure 5 shows the wireframe rendering of the patch surface.

5.3 Problems and Future Work

An apparent distortion seems to appear consistently in one corner of the patch. This is probably the result of a bug in the code rather than a bug in the algorithm, as all interior points are treated identically. This bug may be due to some numeric instability in the computation of normals.

The current implementation does not cover the complete algorithm of section 4.4. The implementation does indeed track the incurred distortion, but does not halt the development process when the distortion tolerance is surpassed. In addition, the implementation does not deal with the texturing of multiple connected patches. Both these issues will be dealt with in the immediate future.

A problem which may arise from this work is that patch boundaries may become visible due to the fact that they are always faithfully mapped. Distortion will be incurred only on patch interiors, and discontinuity will occur along concentric rings. One method for alleviating this problem is by use of some relaxation procedure inside each patch which will spread-out the distortion more evenly. This technique is left for future work.

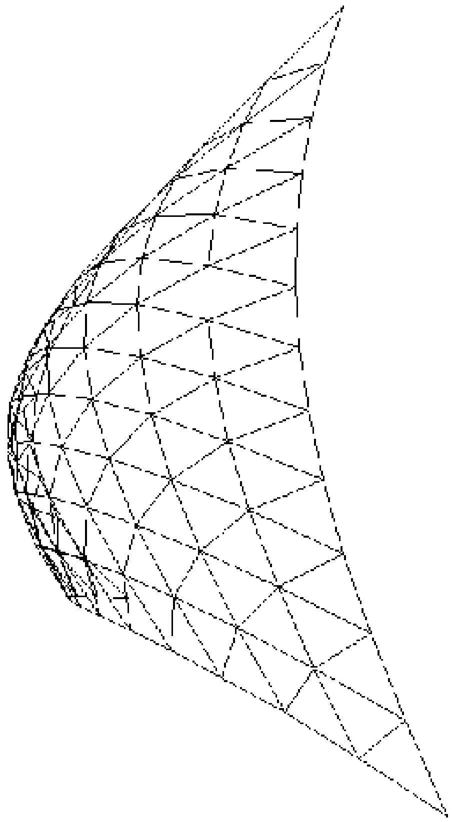

Figure 5: Wireframe rendering of the triangular Bézier patch

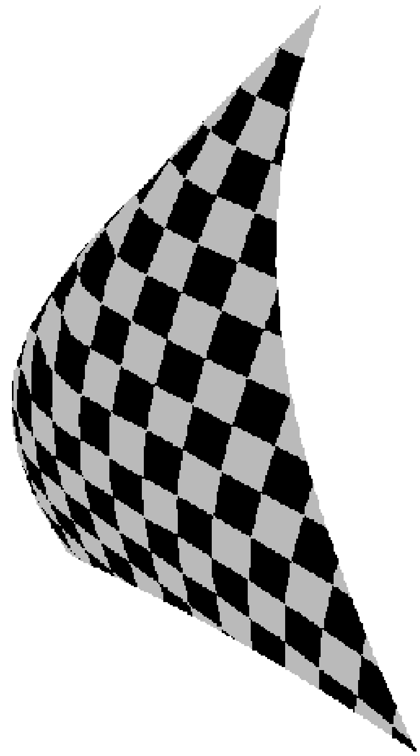

Figure 6: Non-distorted map of a checkerboard on the patch

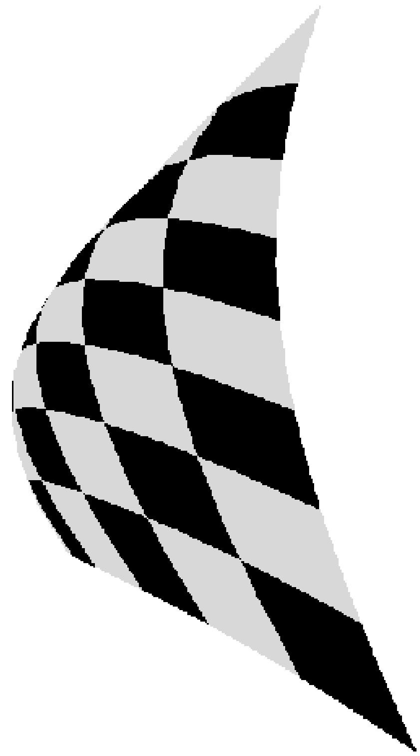

Figure 7: Distorted map of a checkerboard on the patch

References

[BVI91] Chakib Bennis, Jean-Marc Vézien, and Gérard Iglésias. Piecewise surface flattening for non-distorted texture mapping. *SIGGRAPH 91, Proceedings of Computer Graphics*, 25(4):237–246, July 1991.

[Cat74] Edwin A. Catmull. *A Subdivision Algorithm for Computer Display of Curved Surfaces*. PhD thesis, Dept. of Computer Sciences, University of Utah, December 1974.

[dC76] Manfredo P. do Carmo. *Differential Geometry of Curves and Surfaces*. Prentice-Hall Inc., Englewood Cliifs, NJ, 1976.

[Far86] Gerald Farin. Triangular Bernstein-Bézier patches. *Computer Aided Geometric Design*, 3:83–127, 1986.

[Far93] Gerald Farin. *Curves and Surfaces for CAGD*. Academic Press, London, 3rd edition, 1993.

[FFC87] Eugene Fiume, Alain Fournier, and V. Canale. Conformal texture mapping. In G. Maréchal, editor, *Eurographics 87*, pages 53–64. Elsevier Science Publishers B. V. (North-Holland), 1987.

[HL93] Josef Hoscheck and Dieter Lasser. *Fundamentals of Computer Aided Geometric Design*. A K Peters, Wellesley MA, 1993. Larry L. Shumaker trans.

[Rei96] Jesse Reiter. Surface modeling using triangular Bézier patches. Master's thesis, Dept. of Computer Science, University of Toronto, January 1996. Forthcoming.

[SF93] Jos Stam and Eugene Fiume. Turbulant wind fields for gaseous phenomena. In *SIGGRAPH 93, Computer Graphics Proceedings*, Anual Conference Series, 1993, pages 369–376, New York, August 1993. ACM SIGGRAPH.

[Tur52] Alan Turing. The chemical basis of morphogenesis. *Philosophical Transactions of the Royal Socienty (B)*, 237:37–72, 1952.

[Tur91] Greg Turk. Generating textures on arbitrary surfaces using Reaction-Diffusion. *SIGGRAPH 91, Proceedings of Computer Graphics*, 25(4):289–298, July 1991.

[WK91] Andrew Witkin and Michael Kass. Reaction-Diffusion textures. *SIGGRAPH 91, Proceedings of Computer Graphics*, 25(4):299–308, July 1991.