Generating Textures on Arbitrary Surfaces
(A Thesis Proposal)

Greg Turk, June 1991

Thesis Statement

I hope to demonstrate that noticeably improved biological textures on complex surfaces can be generated by first tessellating a surface into fairly uniform regions and then simulating a reaction-diffusion system on that mesh to create a final texture.

Introduction

There are currently two common approaches to texturing an object for a computer generated image. Both methods have drawbacks when used for texturing natural objects. The first method is to acquire a flat texture (usually digitally scanned or painted) and then to define a mapping from this flat geometry to the surface of the given object. Unfortunately, it is often difficult to find a mapping that does not noticeably distort the texture. The second approach to texturing is called *solid texturing*, where a texture is defined in 3-space by a composition of texture basis functions. A given point on the surface of an object is assigned to be the color dictated by the texture function's value at that point in space. This approach is well-suited for creating objects that appear to be carved out of a solid material such as wood or marble. Solid textures do not seem to be applicable to creating natural textures such as spots or stripes that follow the surface of an object.

My work on generating reaction-diffusion patterns on surfaces is based on models used by developmental biologists who have long been interested in pattern formation in animals as they develop. Examples of patterns in animals are the segments of a fruit fly embryo and the spots found on a cheetah's coat. A reaction-diffusion pattern is generated by two or more chemicals diffusing over a surface and reacting with one another to form a stable pattern. Applying this to synthetic image generation has not been accomplished.

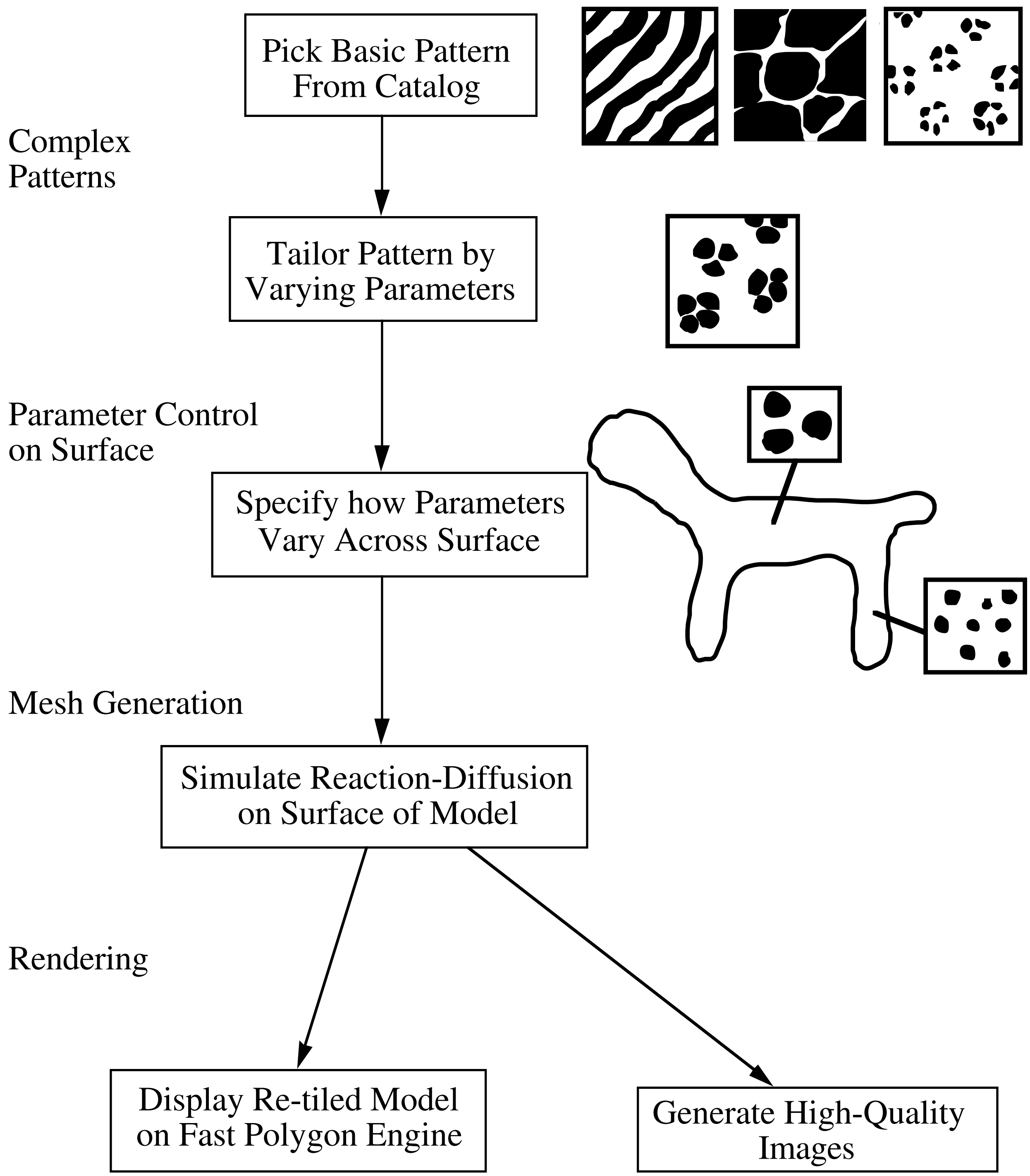
There are two distinct contributions that I hope to make in using reaction-diffusion for texture synthesis. The first contribution is to show how complex patterns can be created by using a cascade of processes. The theoretical biology literature shows simulations of simple reaction-diffusion systems that result in simple spot or stripe patterns of roughly uniform size. I plan to show how more complex patterns can be generated by having one reaction-diffusion system lay down an initial pattern and then having one or more later systems refine this pattern. This adds to the vocabulary of patterns that can be artificially generated. Two examples of more complex patterns that can be formed by cascade processes are the clustering of spots on leopards known as rosettes and the stripes-within-stripes found on the lionfish.

The first contribution I hope to make is to show how any reaction-diffusion system can be simulated on a complex surface. The biology literature shows examples of reaction-diffusion patterns formed on flat surfaces and on a simple tapered cylinder, but does not contain examples of these patterns on more complex surfaces. I plan to demonstrate how a given reaction-diffusion system can be simulated on an arbitrary polyhedral surface to produce a texture. Creating a pattern directly on the surface of a model avoids distortion of the texture and frees the user from having to assign texture coordinates to a model's surface.

An Example

What follows is a scenario describing how a user might interact with a system for texture synthesis based on reaction-diffusion. This is an idealized system, not yet built, that I believe would accomplish the goals of the thesis statement that begins this proposal. The example will serve to highlight the major issues involved in texturing a model with a reaction-diffusion pattern. Let us say that a user has a polyhedral model of a large cat, and wants to fit a leopard spot texture to this model. Traditionally, the user would create a spot pattern on a rectangular surface (probably using a digital paint program) and then painstakingly try to re-map this flat surface onto the surface of the cat. Using this older method, the user would have trouble avoiding visible seams when wrapping the texture about the legs and would also encounter problems when trying to make the pattern uninterrupted at the juncture between the legs and the main body. As an alternative to this method, let us go through the steps used to texture this surface using reaction-diffusion.

An Ideal System



The first step is to find a reaction-diffusion system or a cascade of such systems that produces the desired pattern of spot clusters. The user picks a pattern that most closely resembles the desired pattern out of a catalog of cascade patterns. Perhaps the spots look too regular, so the suer decides to increase the randomness in the underlying substrate upon which the pattern forms. A simulation of this on a small grid gives the look the user wants, and now he or she begins to make decisions about how the ptattern's scale should vary on the main portion of the body. The user specifies this by selecting key points on the head and legs of an un-textured image of the cat and indicating that at these locations the spots should be roughly three centimeters in diameter. Likewise, key positions on the cat's body are selected to have six centimeter spots. These parameters are automatically interpolated across the model's surface, and these values are bundled with the parameters of the cascade process and sent to a program that generates the final texture on the surface of the cat. The internal steps to this process are that a mesh is generated for the surface of the model and then the given reaction-diffusion process is simulated on this mesh to create the final texture.

The resulting texture can be viewed interactively on a graphics engine that displays a re-polygonalized version of the model that liinearly interpolates the texture colors. For a higher quality image, a renderer is used that is enhanced to display textures directly from a texture mesh. Sucha renderer uses a cubic weighted average of mesh values to avoid visual artifacts resulting from the discrete nature of the underlying mesh. If the cat is positioned far from the viewer, the renderer may generate the cat's image from a more coarse representation of the model's geometry, and may also use a version of the texture where the higher frequency components have been eliminated.

There are four issues that this example illustrates:

- creation of complex patterns
- generating a texture that fits the geometry of a model
- specifying how parameters vary across a surface
- efficient and artifact-free rendering of the textured model

In the sections to follow, I will describe how I plan to solve each of these problems. Some of these issues have been addressed in my paper that will be published in the proceedings of SIGGRAPH '91. In what follows I will identify which problems already have a satisfactory solution and which will require more work.

Creating Complex Patterns Using Reaction-Diffusion

The patterns that have been published in the biology lierature have been of rather simple spots or stripes of fairly uniform size. My article explores one way of making more complex patterns by cascading two reaction-diffusion systems by "freezing" portions of the surface. This work on creating two-dimensional patterns using cascade processes is new. There are a wide variety of ways one chemical system can leave a history that will affect a later system, and I plan to explore several of these other methods. These include changing diffusion rates, varying the initiating substrate of the reaction, and changing parameters of the reaction functions. I plan to assemble a catalog of cascade systems that will be a guide to generating textures by reaction-diffusion simulation. Because there are no closed-form solutions known for all but the most simple reaction-diffusion systems, studying these systems will entail a good deal of empirical exploration. Although my focus is on texture synthesis for computer graphics, I hope to write an article about this topic for publication in a theoretical biology journal.

Creating Texture Meshes

As described in my article, reaction-diffusion systems can be simulated on a mesh that has been generated on a given polyhedral surface. Such a simulation creates a texture that fits the surface, and thus requires no work on a user's part to define a mapping from texture space to the object's surface. This kind of mesh is made by first randomly distributing points across the surface and then using relaxation to evenly space the points. Regions called *cells* surrounding each point are computed by finding the Vornoi regions for each point, and from these are derived the diffusion coefficients between adjacent cells. The resulting mesh is used to simulate diffusion that advances uniformly in all directions. If a vector field is given over the object's surface, a mesh can be made that gives anisotropic diffusion over the surface. Making an anisotropic mesh differs from building an isotropic mesh only in that just before the cell for a particular mesh point is found, surrounding points are scaled in the direction of anisotropy given by the vector field. All of this basic mesh generation work was presented in my article.

A simple extension to the mesh generation algorithm will allow more efficient and higher quality rendering of texture meshes. This extension is to build several meshes with different numbers of mesh points for a single object, forming a mesh hierarchy. A given cell of a mesh at one level of the hierarchy would be identified with a group of cells in the lower level (more points) of the hierarchy. Such a multi-level collection of meshes could be used for more efficient anti-aliasing of textures and could give a family of re-tilings of an object for speed/quality tradeoffs during rendering of the object. The section below on rendering will conver this in more detail. With the goal of re-tiling in mind, a mesh on an object could be made to vary in point density based on the local curvature of the object, with more mesh points distributed at areas with high curvature. Each of these topics are extensions of the basic mesh-generation algorithm that I wish to explore.

Controlling Parameters Across a Surface

Patterns found in nature are not uniform across an entire surface, but instead vary in color, feature density and regularity. For instance, the stripes of a zebra become wider near the hind legs and the spots on a giraffe are lighter on the belly and become small on the head and leags. Users who are creating synthetic textures need a way to control this variation. I plan to write an interactive tool to allow a user to specify parameter values at key points on a surface. Thge values at these key points will be interpolated across the surface, and the result of this will be used as parameters to the texture synthesis process. This tool will be an integral part of the idealized system described earlier. For example, a user could pick a low rate of diffusion and then click on the body of a giraffe model, then choose a higher diffusion rate and click at points on the legs and head of the model. These key points would then be used to guide the diffusion rate (and thus the feature size) when reaction-diffusion is simulated on the model's surface.

An important feature of such a tool is to have the values at the key points be interpolated in a natural way to other points on the surface. Diffusion of values across a mesh should provide a mechanism to do this. Using diffusion of parameter values over the surface will give a value at a particular point that is a surface distance-weighted value from nearby key points. Notice that this will give different (and I believe more natural) results than just interpolating in 3-space with no attention to the surface geometry. I have not heard of any such tool for specifying space-varying parameters for texture synthesis, and I feel that building such a tool is a logical part of my thesis work.

There are several extensions to key point parameter specification that I may explore. The first of these is that vector values could be specified at key points. This would be a natural way to set the framework for anisotropic patttern creation. Another possibility is to use the tool to specify (u,v) texture coordinates on a model for the mapping of image textures onto the surface. This would use my mesh-based texture method to aid the more traditional method of rectangular image texture mapping. A third possibility is to have the tool automatically compute surface properties such as curvature and let these values be combined with user-specified parameters to give further control of texture generation.

Rendering

Rendering is the final step in the texturing process, and follows after texture creation and the mapping of a texture onto a surface. My article describes how a texture defined on a mesh can be rendered directly from the mesh description after a reaction-diffusion system has been simulated. A point on the surface of an object is colored based on a weighted average of chemical concentration from nearby mesh points. Bump mapping is an easy extension to this by using the gradient of the chemical concentration as a perturbation vector of the surface normal. The article describes how such a texture can be anti-aliased by using a set of increasingly blurred versions of the texture, each version defined over the entire mesh. A more space-efficient scheme that I plan to implement is to have the more blurred versions of the texture be defined on less dense meshes. This is a natural follow-on to the meshes of different levels of detail mentioned in an earlier section, and is the mesh-based analog to an image pyramid.

Using a texture mesh to re-tile a given object will help achieve the goal of fater or higher quality texture rendering. The dual of Voronoi diagram is known as the Delauney triangulation, and this set of triangles would be easy to create from a texture mesh. If the vertex of each triangle is colored based on the texture mesh, then this collection of triangles can be rendered by almost any polygon scan-conversion device. Bump map perturbations can be added to the vertex positions during re-tiling to create bumps that are a part of the geometry of the object, not just as a change in surface lighting. Re-tiling based on the meshes at different levels of detail can be used to form a set of polyhedral models for a given object. Such a set of models can be useful for interactive graphics applications where there is a fixed polygon budget per frame. Less detailed models can be used when an object is far away from the observer. Each of these extensions based on re-tiling are steps towards improving the display of texture meshes, either by enhancing the quality of the final image or by speeding up the display of textured objects.

Summary of Goals

The above sections described how these four problems can be solved: generating complex patterns, mesh generation, control of parameters across a surface and rendering. The solution of these problems will make the dream system a reality. I feel that accomplishing this would adequately demonstrate my thesis that noticeably improved textures can be generated on a complex surface using reaction-diffusion.

There are two additional topics that I believe to be interesting extensions to this work but that would not be necessary to achieve the idealized system. The topics are mosaic textures and rendering of textures using Gaussian splatting, and the details both are given in two appendices. Much of the reason these topics are interesting to me is that they are odd enough ideas that I think each of them has less than a fifty percent chance of succeeding. If either idea happens to work out then this will enhance my dissertation, but I would like the successful demonstration of my thesis not to depend on either of them.

Schedule

Here is my schedule for completing the doctorate:

Summer '91	research (cascade processes & parameter control)
September '91	oral exam
Fall '91	research (mosaic textures & rendering) and writing
Spring '92	finish writing dissertation, teach intro computer course
April '92	thesis defense

Appendix One: Mosaic Textures

The meshes described in an earlier section were created with simulation of reaction-diffusion systems in mind. Such meshes may, however, prove useful for other kinds of texture synthesis. One possibility is to use a mesh as an aid to texture element placement. Suppose we wanted to place a number of variable sized disks on the surface of a complex object so that no two disks overlapped. We could use a pre-computed mesh as a method of checking whether a disk has already been placed in a particular location on the surface. Each cell in the mesh would be marked empty to start, later to be marked as full when a texture element is placed over the cell's location. A hierarchy of meshes would allow checks at different scales to be made efficiently.

Another possibility would be to use the mesh as a framework for trying to fit a regular tessellation onto a complex surface. For example, we might want to try fitting a hexagonal pattern onto the surface of a sphere. (It is provable that there exists <u>no</u> tiling of a sphere by hexagons alone.) Using constraints from the features of a hexagonal grid, nearby cells could compete for orientation alignment by a relaxation process. The end result of such a process may not meet all the geometric constraints everywhere, but it would give a near-fit to the problem.

Appendix Two: Rendering of Texture Meshes by Gaussian Splatting

Generating images directly from texture meshes, without polygons, may be possible using a volume rendering technique known as *splatting*. This technique composites Gaussian footprints, each representing a volume density element, into a framebuffer. It may be possible to let each mesh oint represent an opaque volume density and to render the collection of mesh points as Gaussian footprints. The color of each footprint is derived from the texture mesh color. A possible stumbling block to this scheme is if the mesh points are not spaced evenly enough over the surface then portions of the background might leak through. Also, it is not immediately clear now to sort the mesh points with respect to the eye point in an efficient manner. I intend to look at texture mesh splatting because it is an odd enough idea to be interesting.

Thesis Outline

Introduction

Previous Work in Texture Synthesis and Mapping

Reaction-Diffusion Patterns

- introduction to reaction-diffusion (done)
- stripe initiation (done)
- catalog of cascade processes
- anisotropic patterns

Creating Texture Meshes

- basic meshing algorithm (done)
- anisotropic meshes (done)
- meshes at different levels of detail
- mesh density varying with degree of curvature

Controlling Parameters Across a Surface

- diffusion of parameters from key positions
- creating vector fields for anisotropy
- (u,v) coordinates for image mapping
- parameters tied to surface properties

Mosaic Textures

- placement of texture elements checked by mesh
- use relaxation to make good fit of regular tessellation to surface

Rendering

- use weighted average of mesh values for surface color (done)
- anti-aliasing by mesh analog of "image pyramid"
- re-tile surface based on mesh
    - color-interpolated polygons
    - bump map creates new geometry
    - multiple levels of geometric detail
- use "splatting" to render directly from mesh without using polygons