Institut für Computergraphik und Algorithmen
Technische Universität Wien
Karlsplatz 13/186/2
A-1040 Wien
AUSTRIA
Tel: +43 (1) 58801-18601
Fax: +43 (1) 58801-18698

Institute of Computer Graphics and Algorithms
Vienna University of Technology
email:
technical-report@cg.tuwien.ac.at
other services:
http://www.cg.tuwien.ac.at/
ftp://ftp.cg.tuwien.ac.at/

TECHNICAL REPORT

Rendering Imperfections: Dust, Scratches, Aging,..

Michael Schwärzler
Vienna University of Technology, Austria

supervised by Michael Wimmer
Vienna University of Technology, Austria

TR-186-2-07-09
September 2007

Keywords: imperfections, dust, scratches, aging, materials, details

Rendering Imperfections: Dust, Scratches, Aging,..

Michael Schwärzler*
Vienna University of Technology, Austria

supervised by Michael Wimmer†
Vienna University of Technology, Austria

Abstract

In order to incrase the realism of an image or a scene in a computer-graphics application, so-called "imperfections" are often used during rendering. These are techniques which add details like dirt, scratches, dust or aging effects to the models and textures.

Realism is improved through imperfections since computer generated models are usually too "perfect" to be accepted as "realistic" by human observers. By making them, for example, dusty and scratched, people can imagine them being part of their real world much more easily.

This article gives an overview of currently used imperfections techniques and algorithms. Topics like textures, scratches, aging, dust, weathering, lichen growth and terrain erosion are covered.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Color, shading, shadowing, and texture

Keywords: imperfections, dust, scratches, aging, materials, details

1 Introduction

Computer-generated images suffer from the fact that most of the underlying models are heavily influenced by mathematical approaches and are therefore "too perfect" to be perceived as images from the real world we live in. The introduction of so-called imperfections deals with this problem by modifying object and scene properties in order to add dirt, scratches, dust, and a lot more kinds of blemish.

One of the most straight-forward approaches is to simply generate detailed textures and objects "by hand". But this is a very time-consuming and difficult task, which can only be performed by skilled and gifted artists and designers. Therefore, some techniques were developed which allow (semi-)automatic creation of such blemish textures by using simplified physical models, statistical distributions or rule-based systems.

To be able to model and render scratches or dust, some special methods had to be developed due to their geometry: Conventional surface description approaches are not really well adapted to defects close to pixel range, as the width of scratches or dust particles is usually very small (but they can nevertheless remain strongly visible).

Weathering effects (for example erosion on terrains or rust on objects) are another kind of imperfections which must be taken care of, since nearly all objects in nature are influenced by them. Furthermore, "biological imperfections" such as lichen growth and terrain erosion are strongly related to weather.

2 Overview

Starting with general texture techniques, two approaches are presented to generate blemish textures in an efficient and easy way. The third paper in this section is about solid textures, called hypertextures, which can be used for modeling objects with special surfaces not easily describable by the usual methods.

The next section deals with scratches and aging of surfaces. The first approach uses specific BRDFs to display scratches; the second describes a semi-automatic way to simulate surface aging by impacts.

Dust simulation using surface properties and geometrical information is discussed in section 5.

Three completely different approaches to simulate weathering effects are presented then: The first uses γ-tons for simulation of weather, the second one concentrates on the impact of water & rain on a scene, and

*e-mail: michael@schwaerzler.com
†e-mail:wimmer@cg.tuwien.ac.at

the third gives detailed information about modeling and rendering of corrosion.

The growth of lichen is described in section 7, followed by some information about terrain erosion.

3 General Texture Techniques

Using textures in 3D computer graphics is definitely one of best ways to create realistic pictures in an efficient and fast way. They offer the opportunity to "fake" many details without specifying additional geometry. Therefore, they are widely used to create imperfections like dirt, dust, etc.

3.1 A Geometry Dependent Texture Generation Framework for Simulating Surface Imperfections

Wong et al. introduced a two-step texture generation framework in between manual texture synthesis and automatic physical simulation [Wong et al. 1997]. The formation of blemish may be due to human factors, physical laws, chemical reactions, etc. It is usually complex and sometimes even unknow, but usually looks irregular. Still, the underlying systematic distribution of such a pattern is observable in most cases. Therefore, the first step in the generation process is to model this so-called tendency distribution. Tendency is a scalar expressing the potential for the occurrence of surface imperfections.

In order to simplify the simulation process, abstract imperfection sources are introduced. For instance, scratches are usually often found near the handle of a leather handbag, due to frequent human contact. By placing these sources, the user can specify where to introduce blemishes.

Figure 1: A brand new Beethoven statue and the same a few years later with patina formed on less exposed regions

Since the underlying distribution of these patterns is usually geometry dependent, geometric information such as curvature and exposure can be used to automate the generation of the detailed textures.

The flux density represents the effect of an imperfection source on that surface point. It depends on the geometrical relationship between the surface point and the imperfection source, and takes values in the range [0,1]. A value of 0 means the surface is free of blemishes while a value of 1 means it is full of blemishes.

Various abstract imperfection sources are modeled similar to light sources: ambient, point, directional, spotlight, and slide projector. If more than one imperfection source is given, flux densities are summed up after being scaled separately. Negative sources are possible as well.

Apart from flux density, local geometry parameters like exposure or curvature affect the actual distribution of blemishes.

So the final tendency T at a surface point P is the overall flux density F_s perturbed by a function α of all geometric factors G_j:

$$T' = F_s * \alpha(G_1, G_2, G_3, ...), \tag{1}$$

$$T = \begin{cases} 0 \text{ if } T' < 0, \\ T' \text{ if } 0 \leq T' \leq 1, \\ 1 \text{ if } T' > 1. \end{cases} \tag{2}$$

The perturbation function $\alpha()$ is a function of all geometric factors and returns a positive real value.

Surface exposure at a surface point P is a measure of exposure to air. A nearby obstacle results in a low exposure. As the distance of the obstacle increases, its effect on the surface exposure decreases, hence it is more exposed. To calculate its value for a point P, n_0 infinite rays are fired from P and evenly distributed on the upper hemisphere. Its value ranges from 1 (completely exposed) to 0 (touched by an object):

$$G_\xi \approx \frac{1}{n_0} \sum_{i=1}^{n_0} \omega(d_i), \tag{3}$$

where d_i is the point of intersection and $\omega(d_i)$ a weight function.

By firing the rays in random directions, a rough approximation can be achieved which is much more economic in computational cost. The drawback is the introduction of noise due to randomness.

Figure 2: Ray intersecting with obstacle (left) and evenly distributed rays emitted from point P (right)

Other ways to determine surface exposure are calculating tangent-sphere and offset-distance accessibility as proposed by [Miller 1994], or using radiosity methods.

Although surface exposure can be computed on the fly, it is suggested to precompute and store it for later use, for example in a 2D texture map. As surface exposure is related to the topic of "Ambient Ocllusion", further improvements could be achieved using the techniqes presented in [Knecht 2007].

Surface curvature is a quite important geometric factor for imperfections as well: For example, paint on a protrusive area is more likely to be peeled off than that on a flat one. It can either be analytically determined in case of a biparametric surface, or it is approximated by the length of the vector resulting from the subtraction of the unit normal vectors of polygons sharing the vertex in case of a mesh.

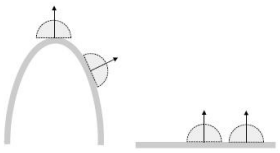

Figure 3: Paint on the convex surface has a larger tendency to be peeled off than that on a flat one, even though they have the same surface exposure.

Whenever the surface curvature has been determined, it can be stored in a 2D texture map just as in the case of surface exposure.

Figure 4: Average and Gaussian curvature of a teapot

Once the tendency value T is determined, texture can be generated according to the tendency distribution: Places with higher tendency should obtain more blemishes. Solid texturing [Peachey 1985; Perlin 1985] is used to prevent texture distortion: In order to obtain irregular blemish patterns, volumes are either scanned or modeled by using fractional Brownian motion [Mandelbrot 1983] and Perlin's noise and turbulence functions [Perlin 1985].

By defining appropriate functions for the surface properties, the authors of this technique achieved remarkable results in the simulation of dust accumulation, patina, peeling, and other imperfection scenarios. Figures 5, 6 and 7 show some examples.

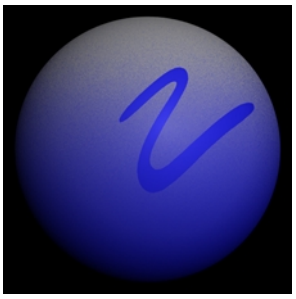

Figure 5: A dusty sphere with a scrap pattern

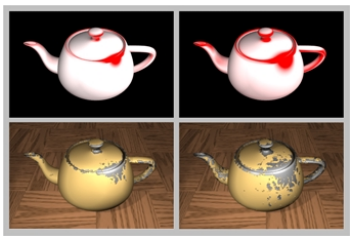

Figure 6: Growth of peeling

Figure 7: A synthesized image of a peeled teapot

3.2 Imperfection for Realistic Image Synthesis

This approach suggests using a rule-based system to create textures instead of simulating complex scenes, because most imperfections are viewed from distances where exact details of distribution or local appearance are indistinguishable [Badler and Becket 1990].

The approach involves two steps:

- Blemish instance modeling: Finding some technique to model a localized instance or concept of blemish
- Blemish placement: designing rules that place or control the distribution and local simulation of instances. This process constructs relevant statistical parameters for local simulation given simple object information such as shape and composition and specific contextual information such as the use of the object or location of adjoining objects.

In order to create the textures representing the blemish instances, Gaussian and random distribution functions are used as well as rule-guided aggregation and 2D fractal subdivision.

Rule-based aggregation can be used to construct tree-like clusters by simulating the diffusion of randomly moving particles in a "sticky" environment. Some particles are defined as sticky, so that there is a chance of clustering on every collision during the diffusion process. By replacing the stickiness probability by a growth probability based on any set of growth rules considering distances from centre or other particles, interesting blemishes like rust or complex stains can appear.

Figure 8: Rule-based aggregation (Rust)

Using fractal algorithms, 2D arrays of values are generated, which can achieve blemishes exhibiting fractal boundaries or densities. The fractal dimension of the array can be

- interpreted directly as a mapping between two surface qualities.
- cut at a thresholding value to a binary map giving filled regions of circular Brownian motion.
- clipped to a range to form fractal Brownian motion rings.

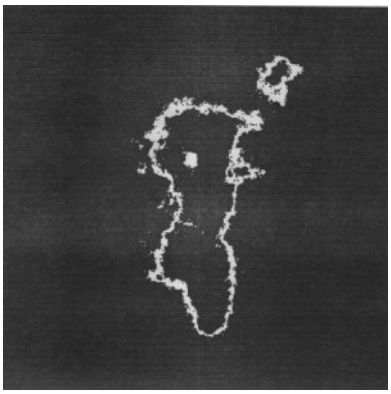

Figure 9: Fractal (Coffee stain)

By using texture mapping, the created blemishes are then positioned on the object surface, depending on the effect to be visualized. Positioning can be controlled by a rule-based system which is based on a natural language interface. A parser reads the commands given by the user. The resulting structure is then analysed by various rules considering the material, the object, the severity and location of the imperfection, etc. to create appropriate attribute map structures which are used for blemish placement.

Some Examples:

- Scratches are simple geometric primitives, just lines modifying surfaces either by increasing the facet distribution value or revealing what is underneath a composite surface. They tend be appear oriented towards some dominant axis. A quick approximation can exploit this, accomplished easily by writing lines of Gaussian-deviant orientation from the given dominant axis. The intensities should then vary as a Gaussian of a given intensity.
- Splotches of very viscous fluids or solutions such as tar or mud, after having hit a surface, appear as several boundary occurrences appearing to have fractal characteristics. Modeling dried splotches is easily accomplished by interpreting a 2D normalized fractal.

- Smudges and corrosion can be modeled as fractal-based intensity distributions using the fractal dimension as an interpolation from the normal surface to the disturbed surface. Corrosion interpolates from the original surface to the attribute of an oxide.
- Mould can be modeled as just a Gaussian distribution in dots of central points in randomly chosen clusters.
- Stains from a distance appear as fractal boundaries, and rust on a surface can be modeled using a rule-guided aggregation.

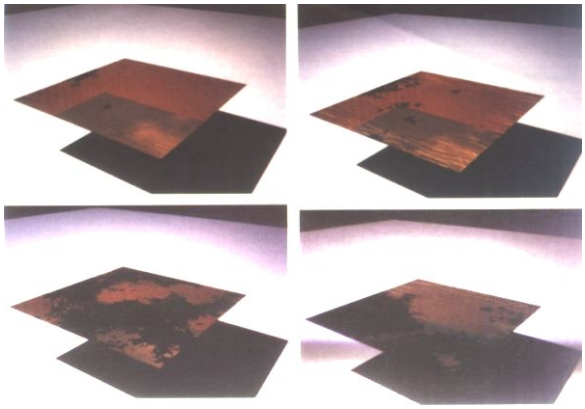

Figure 10: Planes in a room increasing from left to right in scratches and from top to bottom in tar splotches.

3.3 Hypertexture

The approach presented in [Perlin and Hoffert 1989] is quite different from other texture techniques: Instead of surfaces, three dimensional textures (distributions of density) are used to represent the objects. The main advantage of this technique is to be able to render objects which can hardly be described by surfaces, like for example woven materials, fluids, clouds, flames, smoke or (and this is why this paper is presented here) eroded materials.

The following concepts have to be introduced:

- an Object Density Function D(x) with range [0,1] which describes the density of a 3D shape for all points x throughout R^3. The soft region of an object consists of all x such that $0 < D(x) < 1$.
- a Density Modulation Function (DMF) f_i, which is used to modulate an object's density within its soft region. Each DMF is used to control some aspect of an object's spatial characteristics; a collection of DMFs comprises a volume modeling toolkit.

In order to create a hypertexture, DMFs f_i are applied successively to an object's D(x):

$$H(D(x),x) = f_n(..f_2(f_1(f_0(D(x))))) \qquad (4)$$

A so-called soft object is a density function D(x) over R^3, where D is 1.0 inside the object, 0.0 outside the object, and $0.0 < D < 1.0$ in a region of nonzero thickness in between.

The Boolean operators are extended to soft objects A and B through their density functions a(x) and b(x):

- intersection: $A \cap B \equiv a(x)b(x)$
- complement: $\overline{A} \equiv 1.0 - a(x)$
- difference: $A - B = A \cap \overline{B} \equiv a(x) - a(x)b(x)$
- union: $A \cup B \equiv a(x) + b(x) - a(x)b(x)$

The base level DMFs used for modeling are as follows:

- bias is used to either push up or pull down an object's density
- gain is used to make an object's density gradient either flatter or steeper
- noise (introduces randomness)
- turbulence
- arithmetic base functions such as abs and sine

One example shown in this paper is an eroded cube, modeled by using the generalized Booleans operator "intersection" to combine a fractal sphere with a cube. Furthermore, the turbulence function was used here as well in order to create color variations.

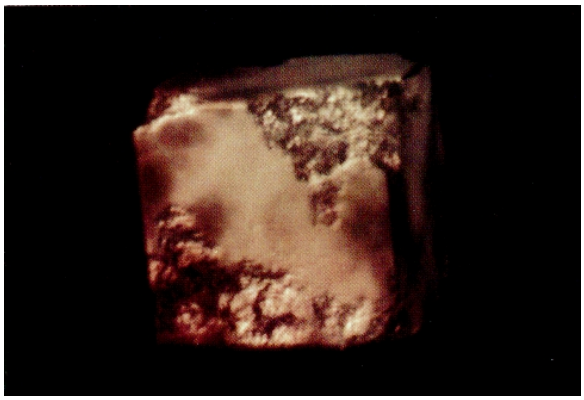

Figure 11: An eroded cube

Hypertextures can be rendered using volume rendering techniques. An algorithm called “Ray Marching” is proposed, but since the presented paper is already quite old (compared to recent developments), some more sophisticated approaches using current shader models should be used.

4 Scratches and Aging

Scratches and aging effects are some of the most important and mostly noticed techniques to apply imperfections. But because of their geometrical properties, finding a suitable way to model and render them can be quite challenging.

4.1 Surface Scratches: measuring, modeling and rendering

A very realistic and accurate way to model and render scratches is suggested in [Mérillou et al. 2001b]. It uses separate reflection models for scratched and unscratched areas.

There are several ways to create imperfection and aging effects on models: Colors can be modified as well as the texture(s); bump- and displacement-maps were introduced and are widely used. BRDFs (bi-directional reflectance distribution functions) help to achieve a high degree of realism. But in order to model and render scratches, none of the methods mentioned above are suited for rendering them efficiently (bump maps and displacement maps, for example, suffer from aliasing effects close to the pixel range). In [2001b], a new method based on existing BRDFs and classical 2D texture mapping is suggested.

In order to obtain realistic results, scratches on “real” models were measured using a surface measurement device. The values were used to define corresponding BRDFs for each material.

Since the reflection properties of the scratch differ from the rest of the object, the scratch and the rest are examined separately: Using the data from the measurements, theoretical cross-sections are defined. Each profile is divided into six zones: four central zones, corresponding to the “scratched” are and two “unscratched” zones on the left and right side. Their properties (like angle or width) are later used in the BRDF calculation.

In order to describe the location of the scratch on the object itself, a simple 2D texture mapping technique is used: One color describes the scratched area, the other the unscratched. The map is not used to change the color of the object, but to change its BRDF.

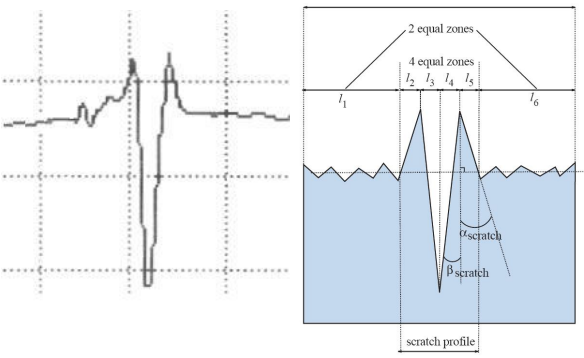

Figure 12: A measured scratch profile and a cross-setion geometry of the scratch model

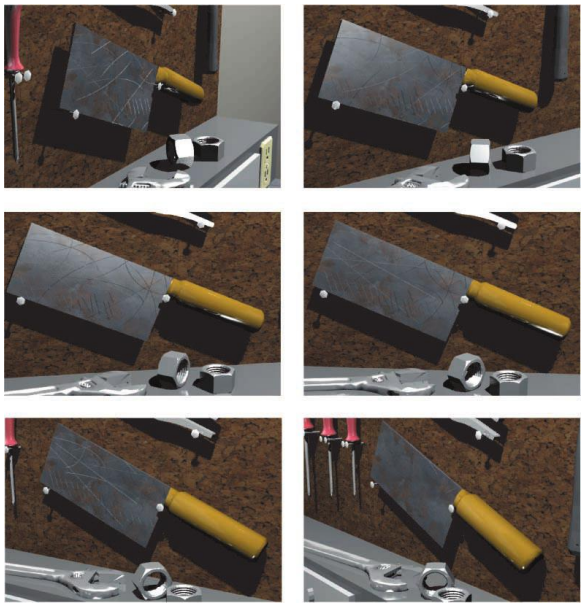

Figure 13: Fresh scratches on an old tool

This way, two BRDFs are computed for the rendering process: One for the unscratched area, and one for the scratches using the previously measured values in the 4 “scratch-zones”.

It has to be said that this technique is not sufficient for *extremely* close view points like in microscopic views, for which a computationally more expensive approach like displacement mapping should be used: If the view point is too near, the observer expects to see geometric deformations, which are not created by the previously described technique.

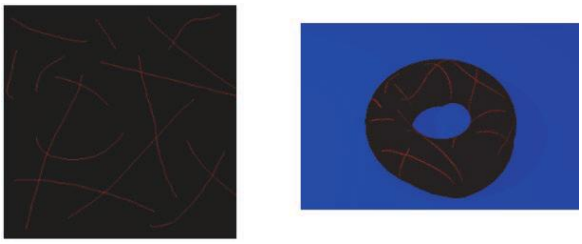

Figure 14: The scratch map appliead on a torus

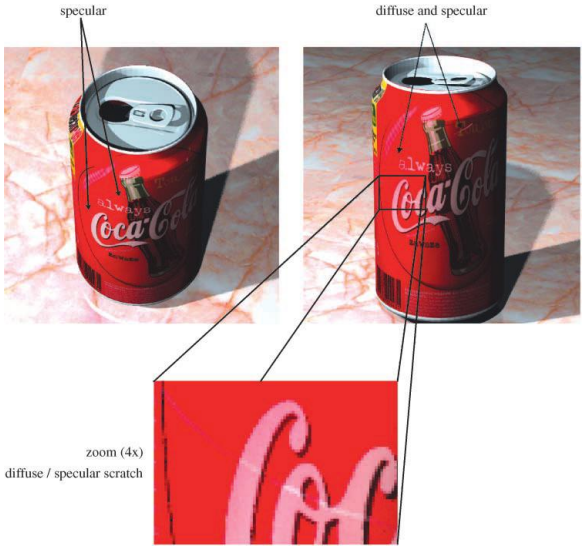

Figure 15: Results obtained on a painted surface

4.2 Surface Aging by Impacts

A semi-automatic surface-aging technique that simulates the deformation of an object caused by repetitive impacts over long periods of time is decribed in [Eric Paquette 2001]. This is done by simulating another object hitting the surface, followed by mesh refinement.

Figure 16: Examples of aged objects

As an initial step, the user selects a so-called "tool" which simulates the object which repeatedly hits the surface. The tool's parameters, such as shape, size, and compaction volume (defines the reaction of the object on the impact) are set, and are either used directly or as input to statistical distributions (e.g. uniform, Gaussian, etc.).

A linear tool path is then defined interactively in a 3D view, which describes the tool motion when hitting the object. The user specifies how many paths are created by a single mouse click. Each path represents a hit during the simulation. The computed paths can be executed immediately and/or recorded and stored in a file, which is useful for re-applying an effect after the original model has been edited.

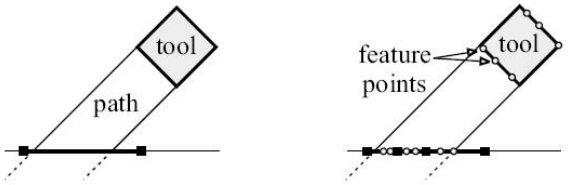

Figure 17: The first two stages of a simulation step: finding and refining the affected faces

In order to refine the mesh, the tool feature points are projected onto the mesh surface. The projected points are then used to refine the mesh by subdividing the corresponding triangles. To avoid regularity, the position of the new vertices along the edge is jittered. To avoid self-intersections and deformation, the vertices are forced to move along a deformation direction. T-Vertices are avoided by dividing adjacent faces as well.

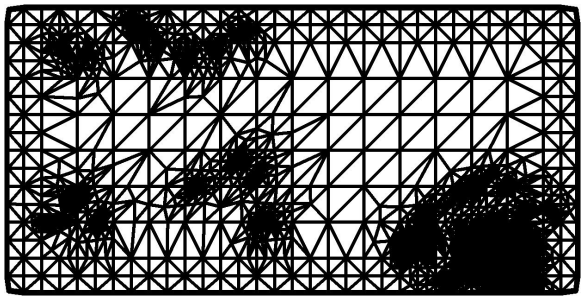

Figure 18: Adaptively refined mesh

When mesh refinement is completed, their displacement is computed. The technique described uses a copy of the faces and vertices, and moves them first along the path direction and then along their deformation direction. Most likely, this step could nowadays also be done using recent shader technologies, which would moreover be a lot more efficient.

In order to render the deformed model, new normals have to be computed for proper display.

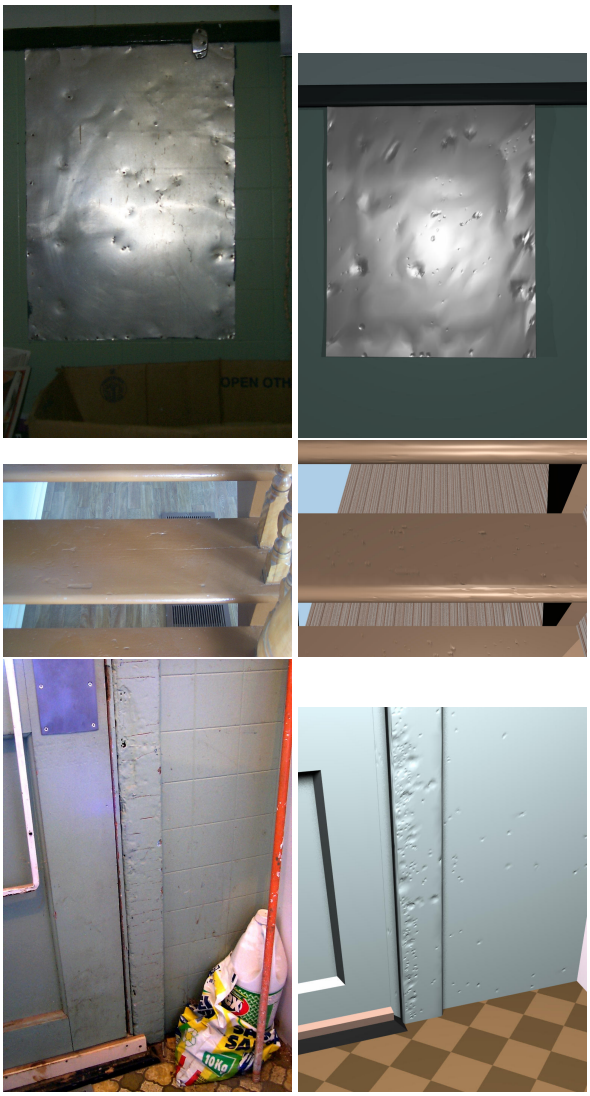

Figure 19: Real (left) and synthetic images (right)

5 Dust

Apart from places like laboratories or surgery rooms, nearly every room and every object in the world is more or less dusty. Although dust is not noticed by us most of the time, it is often necessary to include it in rendered scenes to obtain really realistic images, since the reflection and scattering of light on a dusty surface differs from that on a clean one.

5.1 Visual Simulation of Dust Accumulation

The paper by [chi Hsu and tsin Wong 1995] describes a dust modeling technique. Dust accumulation is first predicted by examining the surface properties like inclination and stickiness, and then adjusted according to some external factors like surface exposure, wind or scraping off by other objects. The dust accumulation pattern can then be stored as a texture map for subsequent use.

To predict the dust accumulation, a so-called “normal dust function” is defined. This function approximates the amount of dust using surface orientation, dust source orientation, and surface stickiness. It is somehow similar to Phong’s specular reflection model:

The influence of external factors is modeled using the factor alpha, which gives information about how much of the normal dust amount is actually realized in the environment. Surface exposure (see 3.1 and [Knecht 2007]) for a point P is measured by shooting and tracing several random rays from that point and calculating the intersections. Depending on the environment, the surface exposure can have a negative (fewer particles can reach the surface there) as well as a positive effect (dust has less chance of being removed by wind or cleared by scraping) on dust accumulation.

Scraping effects are modeled using a dust mapping technique: The dust amount on the surface is changed due to the looked-up values in the dust map (they are used to change the perturbation effect).

To improve realism, several dust sources can be used. This can be compared to the effect of multi-pass spraying.

Rendering is done by first multiplying the dust amounts with Perlin’s noise function in order to perturb it even more. The final surface properties are then linearly interpolated with the perturbed values when fully covered with dust and the original surface properties.

To gain more accurate results, BRDF tables with different amounts of dust can be precomputed. During rendering, the reflectance can be interpolated among these tables.

[Blinn 1982] studies the rendering of clouds and dust by statistically simulating their reflection properties. The presented model has been the base for a lot of research in this area.

Figure 20: Dusty sphere modified with a dust map

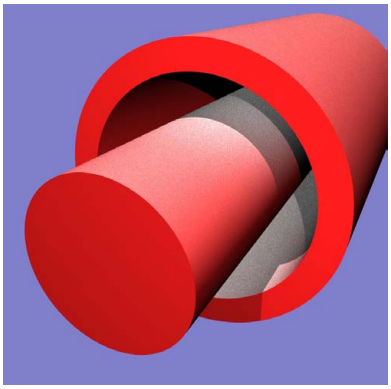

Figure 21: The effect of using a negative exposure scaling factor: the less-exposed area accumulated more dust

6 Weathering

Modeling weathering phenomena is definitely a very important aspect in generating realistic images, as nearly every outdoor object in "real world" is influenced by weathering effects. Of course weathering effects can simply be modeled by creating detailed textures and combining them, but this is usually a very time consuming task. Furthermore, this approach is not very flexible, since only a small change in geometry could make it necessary to begin anew. The following more sophisticated methods offer various possibilities to achieve the same or even better results in less time and effort.

6.1 Visual Simulation of Weathering by γ-ton Tracing

The proposed technique is based on aging-particles called γ-tons [Chen et al. 2005]. The main idea is to use γ-sources, which emit these γ-tons. These particles are traced through the scene, comparable to the way photons are traced in photon tracing [Jensen 1996]. The results are saved in a so-called γ-ton map, which is then used in a second pass to generate the actual weathering effect by modifying surface material properties and geometry through multitexturing, texture synthesis or displacement mapping. This process can be repeated iteratively in order to simulate and model cumulative weathering effects over time.

The γ-sources model the sources of aging in this algorithm. Such sources can come in the form of points (e.g. for effects like dribble of rust down a wall from a leaky pipe), areas, or even environments (like, for example, a hemisphere which simulates the surrounding polluted atmosphere). Within each iteration, thousands of γ-tons are emitted from the γ-sources and are shot into the scene. As these particles hit surfaces, they catalyze weathering effects, as they can deposit or pick up substances on the surfaces, which determines the distribution of blemishes. Once a γ-ton runs out of energy after one or several bounces, it settles down. This so-called γ-ton propagation is stochastically determined by their motion probabilities and the γ-reflectance of the encountered surfaces.

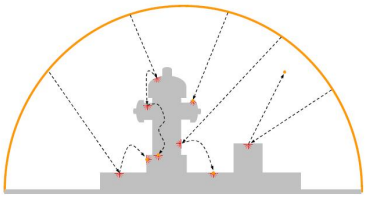

Figure 22: γ-tons shot from the hemispherical environment γ-ton surface bounce in the scene and induce weathering

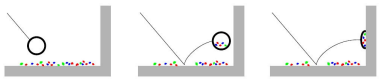

Figure 23: γ-transport by a γ-ton

To be able to record the stochastic weathering contribution and to represent the local surface attributes, a point-based model is generated from the input scene by resampling. Each of these points has its own values for γ-reflectance (affects how a γ-ton deflects from the surface) and for material properties (keeps track of the essential substances for weathering). The blemish being modeled is one of these material properties, and its saved distribution forms the so-called γ-ton map.

In the next iteration step, the surface values have changed due to γ-transport and γ-ton propagation, lead-

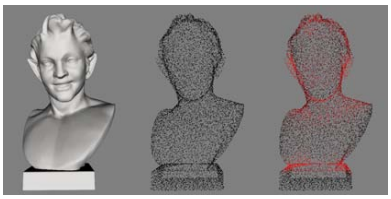

Figure 24: The original model (left) is resampled into a point-based representation (middle). After γ-ton tracing, the γ-ton map is obtained (right). Here the map indicates the presence of metallic patina.

ing to different surface behavior. For example, the introduction of rust particles increases the surface roughness, decreasing the γ-reflectance of the material.

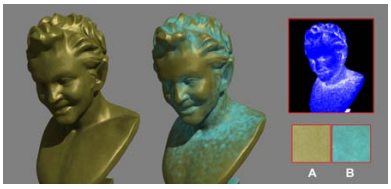

Figure 25: Patina on a bronze sculpture. The two textures used for appearance rendering are labeled as "A" and "B".

For rendering, the γ-ton map values are used in multitexturing or even to displace the surface (for example in cases such as corrosion or erosion).

6.2 Flow and Changes in Appearance

The approach described in [Dorsey et al. 1996] concentrates on modeling the influence of the flow of water, using a particle system where each particle represents a "drop". The particle motion is controlled by physical factors such as gravity, friction, wind, roughness, and constraints that force them to maintain contact with the surface. The model consists of three basic inputs: surface geometry, materials and the environment.

The environment description specifies the initial distribution of water droplets in the scene. The "drops" are created according to a distribution function for incident rain, influenced by wind. These particles are traced until they intersect a model and are deposited in an exposure map (This is usually done in a preprocessing step for efficiency reasons). The particle flow is controlled by rate equations, which take into account the physical factors described above. Comparing the surface normals at regular time steps, obstacles can be detected. The amount

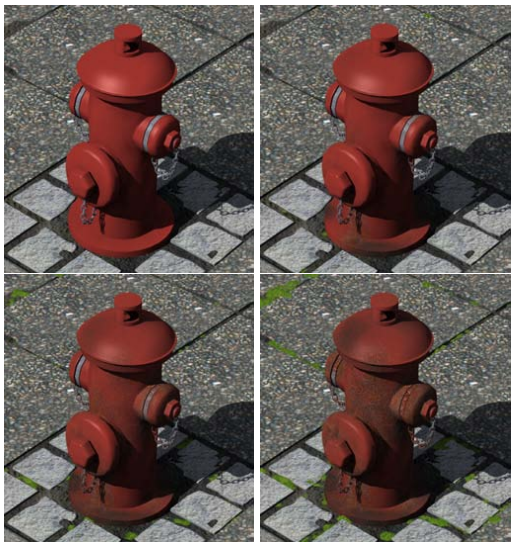

Figure 26: A weathering sequence generated by the γ-ton system.

of carried water in a simulated "drop" decreases as it encounters surfaces with absorbing materials. On the other hand, the particles can dissolve material such as dirt, and transport it to another location.

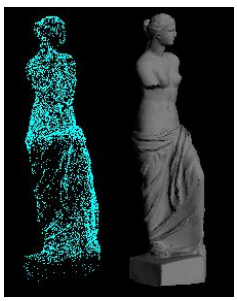

Figure 27: Particle simulation on a complex model

The surface is modeled as a collection of parametric patches, either polygons or cubic spline patches. The need for a continuous flow across a patch boundary makes it necessary to store adjacacency information. Furthermore, a set of two-dimensional set of texture coordinates is stored for each surface, since some surface information is stored in textures (e.g. a saturation map).

Two different simple rough surface models were created during development of this algorithm: The first uses a scalar to control the diffusion process of the flow (the higher the scalar, the stronger a force in the tangent plane of the surface, leading to higher dispersion), the second uses a displacement map (the particle simply follows the cracks and bumps of the map). Moreover,

droplets are allowed to fall off a surface and to hit another surface under influence of gravity.

The absorption of water by a surface is controlled by three parameters: absorption, absorptivity, and saturation. Absorption is the maximum amount of water the surface can hold, while absorptivity is the rate at which water can be absorbed. Saturation indicates the ratio of the water currently held to the maximum absorption. During a simulation, it is stored in a saturation map, which can be used by the rendering system to modify its appearance accordingly.

Furthermore, materials have to store solubility (the rate at which water picks up surface deposits) and adhesion (the rate of redeposition) in order to enable a working deposition model used for sedimentation processes.

Rendering is done using multitexturing again, using the information from corresponding maps: The color of deposit is computed by summing the color of each deposit, weighted by the value from the appropriate texture map, which is then alpha-blended with the original surface. Wet surfaces are simply made darker by modifying the diffuse reflectivity according to the saturation.

Figure 28: Rendering without (left) and with (right) flow patterns.

6.3 Corrosion: Simulating and Rendering

The techniques described in this paper focus on modeling and rendering porous corrosion layers (rust for example) by modifying geometry ("real holes" are produced), color and reflectance [Merillou et al. 2001a].

As a first step, the user sets the number of corrosion starting points for each metallic object (represented as a mesh) in a scene. The location of such a starting point is then computed automatically using several parameters.

In detail, initially each face F_i of an object has the same probability coefficient p_i to obtain a starting point equal to Area(F_i)/ΣArea(F_i). This coefficient is then modified by so-called internal and external parameters.

Internal parameters are, for example, surface imperfections (which usually increase the corrosion probability) or dirtiness (greasy layers usually hinder corrosion). External parameters are connectedness with other objects (leading to corrosion at the contacting faces at the less noble object) or differential aeration (a sudden difference of the oxygen accessibility over two parts of the same material). A high differential aeration value occurs whenever the angle between the face and the face of a nearby object is significant. For example, imagine a pipeline coming out of a wall: the faces of the pipeline close to the wall are characterized by high differential aeration values, leading to higher corrosion.

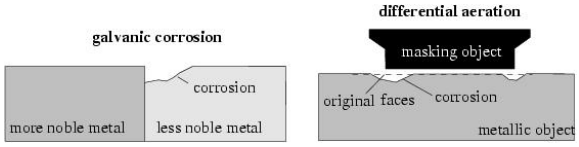

Figure 29: Galvanic corrosion (left) and differential aeration corrosion (right) principles

Once all starting probabilities are set, the corrosion starting points can be chosen randomly according to the coefficients of the faces.

The next step consists of creating a "thick plate" or corrosion-map, which is then mapped onto the object. The previously computed corrosion points are placed on the map using an inverse transformation mapping. Color, a porosity coefficient, a roughness coefficient and an elevation coefficient (decreasing according to the spread of corrosion) are saved.

Starting from the saved points, corrosion is expanded using a random walk technique. On eroded "pixels", roughness, color and porosity are changed, while the height is decreased. Since corrosion does not affect all points at the same time, a "local corrosion time" is calculated for each pixel which influences the corrosion speed accordingly.

Rendering is done using a model described in [Mérillou et al. 2000] which accounts for both porosity and roughness. On non-eroded parts, the "Cook and Torrance BRDF for iron" [Cook and Torrance 1981] is used.

Figure 30: Affected geomtry of a tool

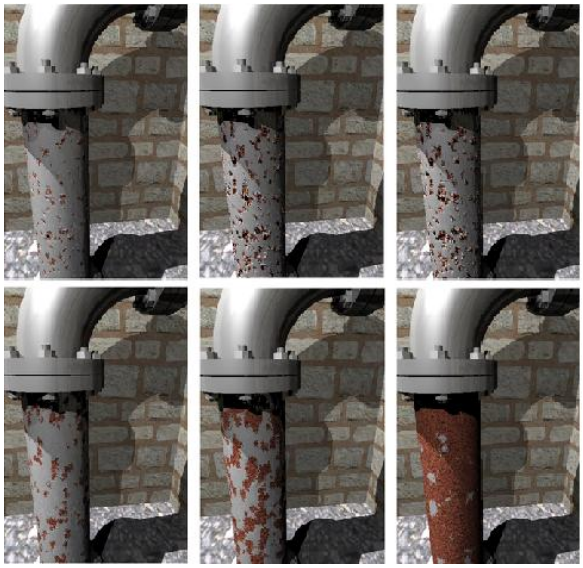

Figure 31: Pitting corrosion (top) compared to uniform corrosion (bottom)

6.4 Related work:

Several other authors published very interesting articles on weathering phenomena, but they are too specific to be presented here in detail. In [Dorsey et al. 1999], modeling and rendering of weathered stone is focused, whereas in [Dorsey and Hanrahan 1996] the same is done for metallic patinas.

7 Lichen Growth

The appearance of objects in nature is heavily influenced by weathering and aging phenomena. Moreover, fungi and lichen growth are an important factor. The following paper uses a fractal algorithm to simulate and model lichen growth.

7.1 Simulating and modeling lichen growth

[Desbenoit et al. 2004] The technique described in this paper consists of three main steps: First, lichen spores are spread over the objects that will be colonized. Second, the lichens propagate over the surface. As the last step, the complex geometry and texture of lichens is created.

For spreading the spores, wind and water flow simulations are used. They are randomly emitted from spherical regions of the scene which are placed by the user. Depending on the characteristics of the substrate, the local geometry, the type of lichen and the local accessibility of the surface, spores stick to specific regions. For example, a candidate in a highly accessible area is removed to take into account the fact that wind may blow the spore away; spores in hardly accessible regions are removed as well. Flow simulations, as proposed in [Dorsey et al. 1996] may be used as well to gain a realistic distribution.

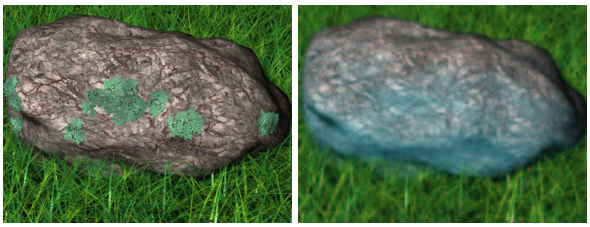

Figure 32: Seed regions and lichen growth using accessibility and water flow simulations.

Since such a simulation can be computationally demanding, a painting technique is proposed as an alternative seeding method. It consists in spreading spores directly over the surface by painting. The painted color defines the local intensity of the spores then.

Surface propagation is done in two steps. First, the moisture and lighting parameters of the colonized surfaces are evaluated. The trajectory of sun in the sky-hemisphere, dispersion of rain and water and moisture sources such as rivers and lakes are taken into account during simulation. The second step consists of invoking an Open Diffusion Limited Aggregation (Open DLA) model that distributes articles in order to form clusters.

Lichens prefer indirect light sources, while direct lighting limits their development. Therefore, two different

lighting maps are created for objects to be colonized (which can be easily identified in the scene graph as the nodes that hold the lichen spores created during the seeding process). Stochastic Monte Carlo raytracing techniques are suggested to approximate lighting. The mesh triangles are reorganized into a BSP data structure to speed up computations. Direct lighting is computed by evaluating the integral of light coming from the sun in the day, where the trajectory of the sun is discretized into a set of light sources.

Since simulating the flow of rain water in the whole scene as well as the influence of water sources such as rivers or ponds is a very complicated and computationally expensive task, the moisture map is created by the user using a three-dimensional painting interface, allowing him to specify wet and dry regions.

As mentioned before, spore propagation is done using an Open DLA model (a modified version of the DLA model proposed in [Witten and Sander 1981]). Spores are first transformed into seed particles that will progressively grow into clusters that form shapes with multiple dendrites organized in a fractal pattern. In contrast to the original DLA model, the characteristics of the environment are taken into account by using a probability function $\mathscr{E}$(p) for a particle p that defines whether the environment favors or limits lichen growth:

$$\mathscr{E}(p) = min(\mathscr{I}(p), \mathscr{L}(p), \mathscr{W}(p)) \tag{5}$$

where $\mathscr{I}$(p), $\mathscr{L}$(p) and $\mathscr{W}$(p) are the functions for indirect lighting, direct lighting and moisture conditions. Every kind of lichen is characterized by its own optimal lighting and moisture values. So the probability of aggregation of a particle p is defined as follows:

$$\mathscr{P}(p) = \mathscr{E}(p) \times \mathscr{A}(p) \tag{6}$$

where $\mathscr{A}$(p) is the probability of aggregation of an article, based on the Pareto law (rich get richer):

$$\mathscr{A}(p) = \alpha + (1-\alpha)\mathrm{e}^{-\sigma(n(p)-\tau)^2} \tag{7}$$

The number of neighbouring particles n(p) is evaluated by searching the number of particles within a distance ρ of p. α, σ and τ are control parameters which define the pattern style.

The particles are then randomly moved over the surface, building clusters on contact by evaluating the function $\mathscr{P}(p)$ described above.

The difference to the original DLA algorithm is that new particles are created nearby existing clusters to shorten simulation time, so the radius for candidates joining the cluster can be kept a lot smaller. As soon as a particle collides with another cluster or moves out of the defined radius, it is removed from the simulation.

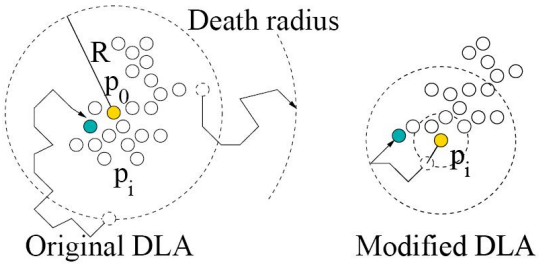

Figure 33: The original (left) and the modified DLA (right) algorithms. On the left side, particles are created at a fixed distance R of p_0. Once they move too far away, they are removed from the simulation. On the right side, the creation distance as well as the death radius are much smaller, reducing computation time.

Since a lot of collision detection has to be done in this propagation simulation, only particles in the 1-ring neighborhood of the triangle the candidate lies in are checked for collisions.

Using the information generated by the propagation method, the lichens have to be visualized. Since all of them have a very complex color and texture patterns, the suggestion is to create a lichen atlas, where a variety of previously modeled lichen shapes are stored. The particles are instantiated into mesh models taken from the lichen atlas, which saves memory and enables large areas to be covered efficiently. Parameters like type of lichen, age and relative orientation influence the selection process.

In order to model crustose lichens that form almost two dimensional bumpy crusts and shells, an approach using textured height fields was developed. Starting from a lichen image, the designer selects interesting feature regions and extracts smaller images, and interactively defines a corresponding height field for every small image. The final so-called cellular texture [Fleischer et al. 1995] is obtained by mapping the sub-image on the height field.

Lichens that produce three-dimensional leaf-like lobes and little scrubs growing upward are created using specific L-System; the corresponding texture map is again obtained from real images.

Figure 34: A complex scene with different types of lichens

8 Terrains

Terrains are usually modeled using fractal algorithms such as Brownian motion or Perlin Noise, which makes the statistical character of the surface, by design, the same everywhere. But in nature, terrain geography is heavily influenced by rain and erosion. The approaches described in this section try simulate these effects by creating appropriate models.

8.1 The Synthesis and Rendering of Eroded Fractal Terrains

Using an erosion-model on fractal terrains, [Musgrave et al. 1989] tries to simulate realistic landscapes influenced by rain and thermal weathering.

Initially, a standard fractal terrain is created by using well-known fractal algorithms (which will not be explained here). In the next step, a hydraulic erosion model is implemented by associating an altitude a_t^v, a volume of water w_t^v and an amount of sediment s_t^v suspended in the water with each vertex v at time t. At each time step, excess water and suspended sediment are passed from v to each neighboring vertex u. The amount of water δw is defined as:

$$\delta w = min(w_t^v, (w_t^v + a_t^v) - (w_t^u + a_t^u)) \qquad (8)$$

If δw is less than or equal to zero, some sediment suspended in the water is deposited at v. The constants K_c, K_d and K_s are, respectively, the sediment capacity constant, the deposition constant and the soil softness constant. K_c specifies the maximum amount of sediment which may be suspended in a unit of water. K_s specifies the softness and soil and is used to control the rate at which soil is converted to sediment. K_d specifies the rate at which suspended sediment settles out of a unit of water and is added to the altitude of a vertex.

Through the described process, water and soil are transported from higher points on the landscape to and deposited in lower areas.

Rainfall on landscapes is usually heavily influenced by so-called adiabatics, or the behavior of moisture-laden air as it rises and descends in the atmosphere. It is easy to include a rough approximation of adiabatic effects by making precipitation a linear function of altitude, which has a significant effect on the erosion patterns produced.

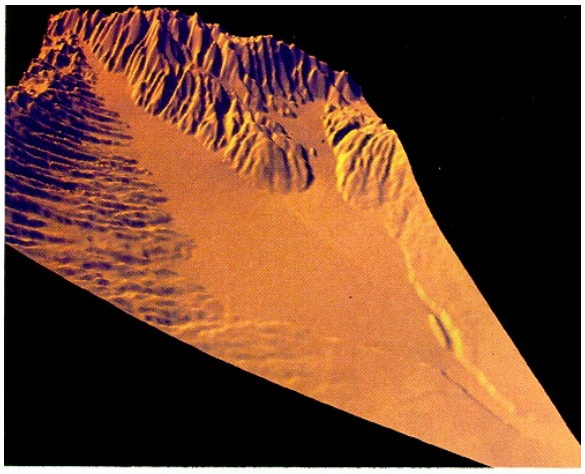

Figure 35: Terrain generated using fractal methods and erosion as proposed in the paper after 2000 time steps.

The second modeled erosion process is thermal weathering. This includes all processes knocking loose material, which then falls down to pile up at the bottom of an incline. At each time step t+1, the difference between the altitude a_t^v at the previous time step t of each vertex v and its neighbors u to the so-called global constant talus angle T is compared. If the computed slope

is greater than the talus angle, some fixed percentage of the difference is moved to the neighbor. So the slope to the neighboring vertices asymptotically approaches the talus angle.

8.2 Related Work

[Kelley et al. 1988] uses a virtual drainage system, which is basically a tree lying on the terrain surface, to simulate water flow and erosion effects on a landscape.

9 Conclusion & Future Work

A lot of different methods and algorithms from the area of imperfections have been presented here. Nearly all of them try to build up simplified physical models inspired by real-world processes in order to create realistic scratches, dirt, aging and weathering effects, etc.

The usefulness of these algorithms depends heavily on the situation: for example, some of them can't be used in real-time applications (yet) as they are computationally too expensive, and others need renderers which are only available in specific setups.

But since there are so many different approaches for solving the same problem, chances are high to find a suitable solution for applying imperfections in most scenarios.

In the future, it is to expect that some of the approaches are modified in order to be calculated on the GPU, especially since the Shader Model 4 introduces geometric shaders, allowing new geometry primitives to be created directly in the graphics accelerator card's memory (shaders offer the opportunity to modify the previously fixed grapics pipeline at will). Using this technology, fast and efficient mesh refinement could enable complex real-time mesh deformation.

Imperfections will definitely stay an active research area for the next decade, since the need for more and more realistic computer generated images, videos and games will abide.

References

BADLER, N. I., AND BECKET, W. 1990. Imperfection for realistic image synthesis. 26–32.

BLINN, J. F. 1982. Light reflection functions for simulation of clouds and dusty surfaces. In *SIGGRAPH '82: Proceedings of the 9th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 21–29.

CHEN, Y., XIA, L., WONG, T.-T., TONG, X., BAO, H., GUO, B., AND SHUM, H.-Y. 2005. Visual simulation of weathering by γ-ton tracing. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*, ACM Press, New York, NY, USA, 1127–1133.

CHI HSU, S., AND TSIN WONG, T. 1995. Simulating dust accumulation. *IEEE Comput. Graph. Appl. 15*, 1, 18–22.

COOK, R. L., AND TORRANCE, K. E. 1981. A reflectance model for computer graphics. In *SIGGRAPH '81: Proceedings of the 8th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 307–316.

DESBENOIT, B., GALIN, E., AND AKKOUCHE, S. 2004. Simulating and modeling lichen growth. Tech. Rep. RR-LIRIS-2004-007, LIRIS UMR 5205 CNRS/INSA de Lyon/Universit Claude Bernard Lyon 1/Universit Lumire Lyon 2/Ecole Centrale de Lyon, Mar.

DORSEY, J., AND HANRAHAN, P. 1996. Modeling and rendering of metallic patinas. *Computer Graphics 30*, Annual Conference Series, 387–396.

DORSEY, J., PEDERSEN, H. K., AND HANRAHAN, P. 1996. Flow and changes in appearance. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 411–420.

DORSEY, J., EDELMAN, A., LEGAKIS, J., JENSEN, H. W., AND PEDERSEN, H. K. 1999. Modeling and rendering of weathered stone. In *Siggraph 1999, Computer Graphics Proceedings*, Addison Wesley Longman, Los Angeles, A. Rockwood, Ed., 225–234.

ERIC PAQUETTE, PIERRE POULIN, G. D. 2001. Surface aging by impacts. In *Graphics Interface 2001*, 175–182.

FLEISCHER, K. W., LAIDLAW, D. H., CURRIN, B. L., AND BARR, A. H. 1995. Cellular texture generation. *Computer Graphics 29*, Annual Conference Series, 239–248.

JENSEN, H. W. 1996. Global Illumination Using Photon Maps. In *Rendering Techniques '96 (Proceedings of the Seventh Eurographics Workshop on Ren-*

dering), Springer-Verlag/Wien, New York, NY, 21–30.

KELLEY, A. D., MALIN, M. C., AND NIELSON, G. M. 1988. Terrain simulation using a model of stream erosion. In *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 263–268.

KNECHT, M. 2007. State of the art report on ambient occlusion.

MANDELBROT, B. B. 1983. *The Fractal Geometry of Nature*. W. H. Freedman and Co., New York.

MÉRILLOU, S., DISCHLER, J.-M., AND GHAZANFARPOUR, D. 2000. A brdf post-process to integrate porosity on rendered surfaces. *IEEE Transaction on Visualization and Computer Graphics 6*, 4.

MERILLOU, S., DISCHLER, J.-M., AND GHAZANFARPOUR, D. 2001. Corrosion: Simulating and rendering. In *Proceedings of Graphics Interface 2001*, B. Watson and J. W. Buchanan, Eds., 167–174.

MÉRILLOU, S., DISCHLER, J.-M., AND GHAZANFARPOUR, D. 2001. Surface scratches: measuring, modeling and rendering. *The Visual Computer 17*, 1, 30–45.

MILLER, G. 1994. Efficient algorithms for local and global accessibility shading. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 319–326.

MUSGRAVE, F. K., KOLB, C. E., AND MACE, R. S. 1989. The synthesis and rendering of eroded fractal terrains. *Computer Graphics 23*, 3, 41–50.

PEACHEY, D. R. 1985. Solid texturing of complex surfaces. In *SIGGRAPH '85: Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 279–286.

PERLIN, K., AND HOFFERT, E. M. 1989. Hypertexture. *SIGGRAPH Comput. Graph. 23*, 3, 253–262.

PERLIN, K. 1985. An image synthesizer. In *SIGGRAPH '85: Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 287–296.

WITTEN, T. A., AND SANDER, L. M. 1981. Diffusion-limited aggregation, a kinetic critical phenomenon. *Physical Review Letters 47*, 19 (November), 1400+.

WONG, T.-T., NG, W.-Y., AND HENG, P.-A. 1997. A geometry dependent texture generation framework for simulating surface imperfections. In *Proceedings of the Eurographics Workshop on Rendering Techniques '97*, Springer-Verlag, London, UK, 139–150.