Evaluation of Gammaton Tracing for Texture Synthesis of Weathered Materials

Philipp Stadler

MASTERARBEIT

eingereicht am
Fachhochschul-Masterstudiengang

Interactive Media

in Hagenberg

im September 2018

Declaration

I hereby declare and confirm that this thesis is entirely the result of my own original work. Where other sources of information have been used, they have been indicated as such and properly acknowledged. I further declare that this or similar work has not been submitted for credit elsewhere.

Hagenberg, September 25, 2018

Philipp Stadler

Contents

Preface

The following thesis has been made possible with the support of many helping hands. Specifically, I would like to give credit to the following institutions (in alphabetical order) for sharing their work under permissive licenses, so they could be utilised in the presented research and implementation:

- *Blender* was used for modelling and texturing of simulation scenes as well as rendering of simulation results,
- *Mozilla Research* originally developed the programming language *Rust*, used for implementation,
- the *Stanford Computer Graphics Laboratory* provided the Stanford Buddha Model and the Stanford Bunny Model, which have been valued for a long time by the graphics community as effective test geometry, and indeed were of great help for weathering simulation of complex surfaces.

Finally, I would like to thank all of the people who provided me with personal support during this time. First of all, I would like to thank DI Roman Divotkey for his support and kind words in his function as my thesis supervisor. I also thank my family and friends for their love and support.

Abstract

The devised weathering solution *aitios* is a variation of γ-ton tracing specifically optimised for texture-based workflows. *aitios* facilitates the procedural weathering of materials in virtual scenery. Ageing-inducing particles are traced from sources in the scene to simulate the emission and transport of weathering-inducing substances throughout the scene. Resulting substance distributions drive additional texture synthesis to create texture maps for use in physically-based rendering. The work contains an evaluation of the practical applicability of the technique, especially regarding runtime performance and physical validity of results.

Kurzfassung

Die entwickelte Lösung für Materialalterung *aitios* ist eine Variation des γ-ton Tracing-Verfahrens. *aitios* ermöglicht es, Materialien in virtueller Szenerie prozedural zu altern. Alternde Teilchen werden von Quellen ausgehend durch die Szene verfolgt, um Aussendung und Transport alterungserzeugender Substanzen zu simulieren. Die sich ergebenden Substanzverteilungen werden weiter zur Textursynthese für die Verwendung in Physically-based Rendering verwendet. Die Arbeit enthält eine Evaluierung der praktischen Anwendbarkeit des Verfahrens für Textursynthese, in der besonders das Laufzeitverhalten und die physikalische Sinnhaftigkeit der Ergebnisse analysiert werden.

Chapter 1

Introduction

The history of computer graphics is a collaborative effort of the scientific community along with artists and developers to create an illusion of reality of ever-increasing quality. It has been possible for dozens of years to faithfully reproduce the appearance of a perfectly smooth and spherical glass ball inside a box, lit by a point light source. Much thought has since then gone into the rendering of less idealised objects with more complex surfaces. As an important milestone, the advent of texture mapping, pioneered by Edwin Catmull in the mid-seventies, allowed for the adding of surface detail by encoding spatially varying properties of surfaces such as albedo, displacement and reflectivity, allowing renderings of many types of virtual objects to appear more lifelike. An important aspect of a realistic set of texture maps for an object is the presence of blemishes and surface imperfections. The following thesis presents a variation of the γ-ton tracing algorithm originally proposed by Chen, Xia, Wong, Tong, Bao, Guo and Shum in 2005, and investigates its applications to texture synthesis of sequences of weathered surfaces, tailoring blemishes to scene geometry and weathering sources.

1.1 Motivation

The advent of textured objects in computer graphics provided artists with an effective means of taking weathering effects into account when designing the surfaces of objects. Such effects include visible signs of ageing such as scratches, dust and corrosion. The process of creating convincing weathered textures has traditionally been a labour-intensive one, requiring large amounts of human intervention, even with modern tooling such as *Substance Designer*[1]. Weathering effects spanning multiple objects in the final scene, such as rust dripping from a leaky pipe onto the floor below, being hard to mimic manually, are usually not taken into account. While it is often sufficient to model a single point in time, obtaining consecutive states of ageing requires either additional human intervention or even a repetition of the process.

When the first version of the presented γ-ton tracing variation has been designed and implemented under the name of *aitios*, a consistent re-implementation of the original algorithm was the initial goal. The high level concept proposed by the original publication

[1] *Substance Designer* is a material authoring software developed by *Allgeorithmic SAS* and available at https://www.allegorithmic.com/products/substance-designer.

was clearly defined, and fascinating in its apparent simplicity. Yet, this original publication remained fuzzy in some details, making an exact re-implementation infeasible. When preparing a simulation, how are surfels distributed? How, exactly, do particles in flow move over complex surfaces? How can textures for rendering be prepared from the surfels? Over the course of time, filling in unknowns in the process, *aitios* has grown into its own variation with both classical and some novel aspects. Well-defined, concrete solutions for unclear parts of the tracing algorithm are the first main contribution of this thesis.

A local intensity of ageing is often referred to as *weathering degree* in literature. γ-ton tracing in general heavily concentrates on calculation of such blemish distributions and can be considered an adequate approximate solution to this problem. Texture synthesis based on this distribution, by contrast, is treated as a marginal aspect by classical γ-ton tracing and many later extensions. The presented variation proposes a conversion technique of the point cloud-like surfel tree to greyscale mask textures, indicating the distribution of blemishes over the surface in a way that better integrates with existing rendering pipelines and other tooling. Building on substance textures, a weathering effect pipeline based on blending and layering is proposed, that generates final weathered appearances. This texture synthesis approach is the second main contribution offered by this work.

1.2 Aims

To overcome the limitations of popular techniques in the industry, a variation of γ-ton tracing will be defined and evaluated as a potential alternative. The algorithm simulates material ageing in scenes by generating new textures. The base algorithm is an iterative, probabilistic approach that simulates the emission and propagation of weathering-inducing substances as ageing-inducing particles that are traced through the scene. Result of the substance simulation is a set of surface samples with associated local concentrations of substances. This set is denoted a γ-ton map and organized in a spatial data structure for efficient access. Such a γ-ton map, holding the local concentrations of substances, guides the iterative re-texturing of the scene using texture synthesis techniques after each iteration, generating variations of the scene in consecutive stages of ageing. The technique should at least be applicable to the two forms of chemical weathering presented: rusting corrosion and patination on surfaces. The applicability depends primarily on the convincingness of the results, secondarily on performance with sufficiently complex scenery.

1.3 Overview

After this introductory chapter, the rest of this document is organised as follows:

- Chapter 2 outlines a model of weathering and the state of the art in procedural time-varying appearance modelling,
- chapter 3 provides insight into the specific course taken in the implementation used for the evaluation,

- chapter 4 first outlines the methodology of evaluation and then puts the developed ideas to the test,
- next, chapter 5 summarises the results of the evaluation, outlines potential for future extension of the work, and provides some closing remarks.

Chapter 2

Related Work

This chapter first defines a taxonomy and a model of weathering as a whole. On that basis, the rest of the chapter provides an overview of the state of the art in weathering simulation techniques.

Automation of aspects of time-varying appearance modelling became an increasingly active field of research over the past two decades. Techniques for obtaining surfaces with imperfections typically rely on one or more of the following:

1. manual design by artists,
2. physical measurement,
3. simulation.

The thesis is primarily concerned with the latter. However, most simulation techniques overlap with the first two by requiring human intervention or measurement data. To differentiate simulation techniques for weathering, three main types are defined to organise the chapter.

Physically-based simulations, described in section 2.2, primarily model known physical or chemical phenomena based on domain-specific knowledge. This typically provides quite accurate results within a limited scope of weathering problems, mostly at the cost of intensive computation.

Solutions that employ a *transfer technique* include various models that primarily analyse input samples of weathered objects, such as photographs or BRDF measurements, and aim to convincingly apply these measurements to virtual objects, yielding an aged appearance. Section 2.3 provides an overview over select techniques of this type.

Finally, *phenomenological* approaches aim for convincing approximations based on models not directly derived from physical or chemical research, attempting to provide more general models at lower computational cost. Techniques of this class, to which γ-ton tracing has been assigned, are described in section 2.4.

2.1 Weathering Classification

Before presenting actual weathering simulation techniques, some taxonomy definitions for the underlying processes being modelled are necessary. Gradual appearance changes to objects when exposed to environment conditions are mostly referred to with the umbrella terms *weathering* [2–4, 6, 26] or simply *ageing* or *decay* [17] within the field.

The following work shall stick to this convention, using *weathering* and *ageing* as interchangeable concepts, caused by attack classes as outlined in section 2.1.1.

In contrast to uses in geology, literature in time-varying appearance modelling sometimes does not treat erosion as a completely distinct concept, instead applying weathering to many ageing processes, often including erosion on the level of individual objects, but not on entire landscapes. Classical γ-ton tracing, for instance, uses the term *erosive weathering* for deformations from erosion [4].

2.1.1 Attack Classes

Lu et al. propose to classify ageing effects by the type of process responsible for the change in appearance, namely:

- chemical,
- mechanical,
- biological [18].

They further suggest cascading different processes using their approach, such as chemical corrosion with mechanical crackling [18].

Mérillou and Ghazanfarpour re-iterate on the taxonomy in their weathering survey, referring to the types of processes as *attack classes* on a perfectly clean *reference surface* with smooth geometry. They further emphasise that the attack classes often affect real surfaces simultaneously and may often depend on each other [19]. As exemplified by metallic corrosion, a process might start as a chemical process, but cause large-scale mechanical changes in the long run. The three attack classes are better understood as a spectrum on which to place specific effects. Thus, realistic material ageing often requires taking effects of all classes into account.

They further emphasise the distinction between weathering and manufacturing-related material properties and defects, such as the initial roughness and porosity, which may not be attributed to weathering and instead are initial properties of the reference surface [19]. However, these properties are often of major importance in the weathering process.

2.1.2 Chemical Attacks

This kind of attack affects objects on a chemical level, leading to changes in the material composition of objects, in turn leading to changes in physical characteristics [18]. For example, a copper statue exposed to water and air will eventually develop patina, a film of chemically altered material on the surface [9]. These composition changes mostly affect the surface of objects but can also lead to large-scale geometric changes, e.g., in destructive corrosion [19]. Some examples of this attack class are: patina and rusting corrosion (see figure 2.1), acidic corrosion, burning and limescale deposition. Chemical attacks are the main focus of the thesis.

2.1.3 Biological Attacks

This class describes effects caused by living organisms including mould and other fungi as well as algae [18]. Changes by biological processes involve geometry deformation,

(a) (b) (c)

Figure 2.1: Metal structures with corroded surfaces exemplify chemical attacks. Note the geometry-dependent variations. Artefacts tend to develop flow patterns on vertical surfaces. In (a), rust dissolved in water gets deposited on the floor below.

the addition or removal of geometry and structural changes. Some examples include: rotting, colonization by funghi and plants.

Micro-organisms affecting interior and surface of objects are a special case and can also be seen as an instance of chemical attacks, since the distinction is not always clear in such cases. Mérillou and Ghazanfarpour thus propose "to classify each process that deals with organic materials as *biological*, and the others as *chemical*" [19]. This definition still leaves room for interpretation in the case of microbial weathering in rocks, where micro-organisms produce weathering-inducing organic chemicals, in which case an organic weathering source induces chemical weathering. Therefore, for the purposes of this thesis, only processes where organisms that are directly observable with the naked eye colonize either organic or inorganic objects shall be referred to as biological attacks. By this definition, the previously provided examples are included. However, it does not apply to dehydration and wrinkling of an organic target material, which would instead be classified mechanical.

2.1.4 Mechanical Attacks

These effects occur due to mechanical impacts [18] or material stress. Such effects can have natural causes, such as the exposure to varying temperatures, wind and weather [9]. These changes usually remove material or lead to splitting of the object. Some examples include: cracks and fractures, scratches, dimples due to impacts of objects. These effects can be the cause of large-scale changes to both the geometry and the surface of objects.

2.2 Physically-Based Simulations

This simulation type can be set apart from others in that weathering models are based on domain-specific knowledge derived from physical and chemical research. This requires that the underlying process is both known and a large part of it well-understood. In most cases, parts of the model have to be approximated. Hence, the term *physically-based* is favoured over "physical". Approximations can simply reduce computation time

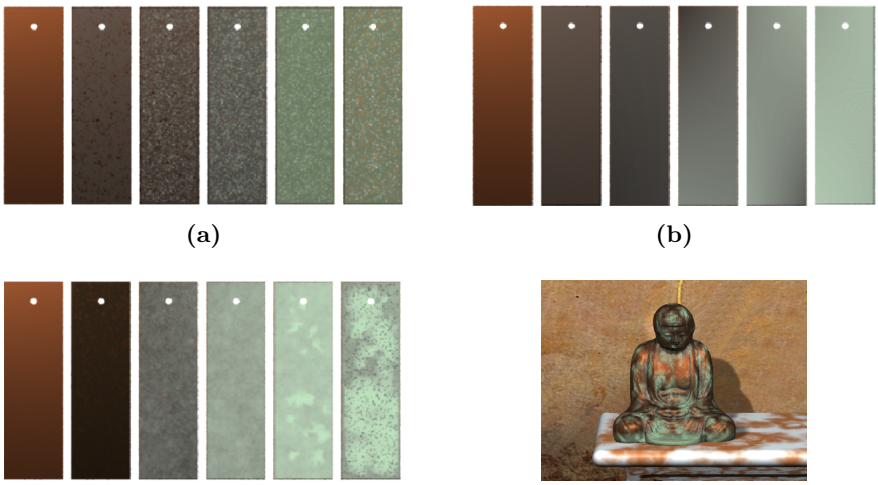


Figure 2.2: Thirty years of patination (six-year increments) on copper strips in marine (a), rural (b) and urban (c) environments simulated with the Dorsey/Hanrahan patination technique. (d) shows the same technique on a complex, scanned model [6].

or account for aspects of a phenomenon that are not well-understood.

Depending on the complexity of the underlying model, computation cost can be intense. The model may require artist-specified or measured parameters for object compositions and the environment. Finding the correct parameters to achieve a certain look can be a daunting task.

Many such specialised algorithms have been proposed to simulate well-understood mechanical, chemical or biological phenomena using a model that aims to approximate the real-world effect as closely as possible with domain-specific knowledge [9]. While *physically-based* is a common term for this class of techniques within the discourse [16, 17], some refer to it as *semi-physical* [11]. Depending on the degree of approximation involved, some authors prefer the term *phenomenological* [7], which is used to describe a distinct class of algorithms in this work (see section 2.4).

2.2.1 Metallic Patinas

Julie Dorsey and Pat Hanrahan present an approach for the modelling and rendering of metallic patinas. Since no experimental data to support a completely accurate physical simulation was available at the time, the approach is phenomenological, but based on a detailed chemical model. Material available in the environment and geometric considerations are accounted for [6]. The degree of physical insight warrants the use of the term physically-based.

Patina Model

Drawing on earlier work, the researchers attribute patination primarily to atmospheric corrosion. Patinas are highly layered, with individual layers being visible in a cross-section of patinated metal [6].

In the case of copper exposed to atmospheric conditions, being the main focus of the paper, a brown layer of tarnish quickly forms. This layer gradually changes into reddish-brown color, indicating the presence of copper oxide or mineral cuprite. The next layers take significantly longer to form and consist of more copper oxides but also copper sulphides and copper salts. Copper sulphide is a dark brown color, but surprisingly shiny since it forms metallic crystal structures. Copper salts, such as sulphates, chlorates and nitrates, come in different colours, with sulphates contributing the characteristic greenish color of aged copper. The minerals formed differ in marine, rural and urban environments, due to different materials available for chemical reactions. For the visual effects of different environments see figure 2.2. Local wetness also influences patination, with horizontal or inclined surfaces patinating more rapidly due to more available stagnant water. Other factors, such as temperature and variations in surface thickness are also deemed to have an influence on the patination process but are deliberately ignored since their influence is not well-understood [6].

Patina material is modelled as a stack of n layers. Every layer has an assigned thickness, albedo, roughness and subsurface scattering properties. Layer 0 represents the base material and is assumed to have infinite thickness. All other layers have a zero or positive thickness. The combined thickness of all layers except the base layer is assumed to be small in comparison to the covered area. The authors further suggest techniques for the rendering of the simulated layered structure (see figure 2.2 (d)) in ray tracing contexts [6].

Operators

To model patina evolution, layers are affected by *operators* in a pre-configured sequence. Multiple types of operators are available.

- The *coat* operator applies a new layer of material with a maximum thickness. The maximum thickness is modulated with the thickness of the layer below.
- By applying an *erode* operator, a desired amount of thickness is removed, starting at the top layer, until the desired amount is reached. The process possibly exposes underlying layers.
- *Fill* applies material up to a given absolute height above the base material. This simulates deposition in cracks.
- With the *polish* operator, material can be removed until a given absolute height, smoothing the surface.
- *Offset* first uniformly applies a thick coat, then removes parts of the coat that are accessible to a spherical probe. The accessibility calculation is performed using the method proposed by Miller [6, 21].

Growth Models

If the operators alone were used for simulation, the result would lack the visual richness of real patina. Specifically, variations and detail must be added. For this purpose, the researchers employ several growth models. They point out that the models can not only be selected by the user but can also be influenced by surface geometry and environment factors. The presented growth models are:

- *Steady thickening* uniformly samples a small number of points, assigns a thickness, and increases thickness by a user-specified growth rate. Interpolation is performed in between sample points. Subtle noise is additionally added for a more natural look, yielding a relatively even weathering pattern.
- *Random deposition* lets a particle move vertically from a sampled point above the surface, resulting in a quite rough surface.
- *Random deposition with surface relaxation* additionally allows for the particle to diffuse to neighbouring surface points with a lower height and up to a finite distance, yielding smoother surfaces compared to standard random deposition without surface relaxation.
- *Ballistic deposition* also starts on a random position above the surface, falling straight down until it hits a surface, where it now deposits material, either increasing the height by one, or setting it to the height of a neighbouring surface point, if one is thicker, yielding lateral growth patterns.
- *Directed percolation depinning* employs growable patches on a 2D lattice of blocked and unblocked cells, with unblocked cells indicating the presence of water. Patches spread outward to their neighbours, preferring unblocked cells.

The presented growth models are described by the researchers as a "black box" for aspects of patina formation that are not yet fully understood. As soon as additional information about patina formation is available, the growth models could be replaced by more physically accurate ones [6].

2.2.2 Flow

Dorsey et al. present a particle-based system to model the weathering effects of flowing water on virtual objects. Water particles are modelled with a mass and amounts of dissolved materials, as well as position and velocity. Differential equations describe the rules for movement as well as exchange of water and dissolved materials from and to surfaces. Potential stain-inducing dissolved agents are: exhaust, bird droppings or dirt [7]. Figure 2.3 shows a rendering of a statue with stains shaped by water flow.

Water Model

The approach simulates a range of natural effects with a rather simple model. An arrangement of water sources forms on the object. The primary way for water sources to form is at parts more exposed to rain. Also, *splashback* occurs when a wall meets the ground, having dirt splatter from the ground a short distance up the wall, flowing down again from there. *Primary flow* describes the flow from sources in direction of gravity, and influenced by obstacles. This flow typically breaks up into several streams. The

(a) (b)

Figure 2.3: Flow simulation by Dorsey et al. on a statue. Model without flow patterns for comparison (a). Flow patterns are clearly visible after simulation (b).

formed patterns are often self-reinforcing. *Secondary flow* may occur when a particle falls off a surface and continues its path in direction of gravity. Interaction of water with a surface is governed by *absorptivity* (rate of water uptake) and *absorption* (the capacity of the surface to absorb water). Once a patch of the surface gets increasingly saturated with water, absorptivity decreases in a material-dependent manner. If not all of the incident water can be absorbed any more, surface *runoff* occurs. Since exposed parts are saturated first, water runs off to dryer regions below. These dryer regions will too get saturated until their absorption rate is lower than the amount of incident water, continuing flow further down. *Differential flow* occurs when flow crosses material boundaries and transports stains. This can happen with many materials such as concrete and corroded metals, where material is dissolved and transported in flow direction. Also, when a highly absorbing material is placed under a low-absorptivity material, flow from the top material causes intense staining in the material below, where dissolved material is deposited. *Saturation staining* is a mostly darkening material-dependent effect, limited to areas with high exposure [7].

Simulation

Given UV-mapped geometry with assigned materials as well as the concentration of materials in water, simulation is performed. Surface properties, such as saturation and deposits, are stored and indexed in textures. An initial distribution of particles on the surface is generated based on exposure to rain. The rain direction is based on a random variation of wind direction. On its path, a particle loses mass due to evaporation and absorption of the underlying surface. Evaporation can be modelled with pre-computed *total solar irradiance*, which includes shadowing. Absorption is controlled by differential equations involving particle and surface. Once no more mass is left, the particle can be removed from the simulation. Also, particles may dissolve material in one location to drop it in another. This sedimentation process has its own set of coupled differential equations, controlled by *solubility* and *adhesion*. Solubility describes the rate of water to

pick up a specific substance. Conversely, adhesion controls redeposition onto surfaces. Particle motion is governed by gravity, friction, self-repulsion and diffusion. With one exception, this motion is restricted to surfaces. To this end, velocity is projected on the tangential plane of a patch of geometry, possibly crossing patch boundaries. The exception allows for secondary flow: If the angle between particle velocity and tangent direction of the patch surpasses a critical value, the particle instead follows gravity through a pre-computed landing position table. Similarly, an abrupt change in surface normals indicates an obstacle. A probability distribution determines whether the path should be continued or diverge in this case. To implement patch transitions effectively, the geometry data structure provides fast access to neighbouring patches. To account for diffusion on rough surfaces, two approaches are proposed. The first approach works with a scalar roughness. A random force offset on the tangential plane can be applied, scaled by the surface roughness. The alternative roughness model works with displaced surfaces. The displacement map perturbs the surface normals to form a displaced mesh. Now, after projecting the force vector on the original tangential plane, it is again projected onto the perturbed plane to form a new vector. This causes water to collect in cracks and valleys and flow more slowly over bumpy surfaces. The researchers further propose a rendering technique: To obtain the final texture maps for albedo, the colours of sediments are summed, each weighted by the final concentration. Similarly, an alpha value is computed for blending with the original texture. To simulate wetness, diffuse reflectivity is modulated with surface saturation [7].

The approach imposes implicit limitations on simulation meshes: Texture regions of triangles may not overlap, since surface properties are encoded in textures and re-using them would lead to unwanted bleeding of substance concentrations and other surface properties from one region to another. However, a uniform texel scale is not required, since the simulation equations are defined in terms of surface area and particle mass. Further, regarding mesh topology, the patch transitions of flowing particles effectively require proper n-manifold geometry. A problematic situation occurs if three or more faces share the same edge, in which case it is unclear which face the particle would transition to. One example of this type of problematic non-manifold would be a mesh consisting of two equally-sized cubes sharing exactly one edge (four faces meet at one edge). Given only valid geometry, individual meshes should further not overlap to prevent particles from entering the interiors of objects when following the surface of another object. For instance, if an iron bar is stuck in a rock, flowing particles will not transition to the rock, but follow the surface of the iron bar inside the object, unless precautions are taken for this case.

Each surface in the simulation requires a set of nine texture maps. It is left unspecified how these textures are indexed for surface interactions. One possible solution would be to index the nearest texel to the position of a particle in UV space. If it is assumed that a particle may move by more than one texel at a time, multiple texels could instead be indexed with a filtering scheme, taking into account the distance to the actual particle.

2.3 Transfer Techniques

Ageing models based on transfer, often referred to as *data-driven* [16] or *measurement-based*, analyse and process systematic measurements of decaying objects in the physical world to synthesise a similar appearance on virtual objects, such that the effect is consistent with measurements. The process of applying the texture of an existing object (physical or virtual) to a different, virtual object is often referred to as *texture transfer* [8]. A complete understanding of the scientific theory underlying an ageing phenomenon is usually not required. Rather, a physically-oblivious model derives virtual appearances from organised measurement data. Depending on the specific technique, some human intervention in addition to measuring may be required, e.g., by classifying weathered and non-weathered areas in the source image for consumption by the algorithm. Measurement data of a sample object must be available in at least one point in time. Typical measurement data types include, depending on the specific algorithm: casual photographs [27], 3D scans and spatially varying BRDF data obtained with a linear light source device from surface samples [26].

2.3.1 Appearance Manifolds

In 2006, Wang et al. proposed an approach based on a data structure that can be created from sample surfaces measured at one given point in time using a linear light source device for spatially varying BRDF measurements. Surface samples are organised in an n-dimensional appearance space. With some user intervention for the identification of most weathered parts, the BRDF samples are ordered by weathering degree. Techniques for weathering, de-weathering, texture transfer from one object to the next, and the creation of timed sequences of textures for a given object and material using appearance manifolds are presented (see figure 2.4) [26].

Despite being oblivious to underlying physical processes, the results are quite convincing. A notable disadvantage is that it relies on accurate measurements, which requires equipment, time and effort. However, only measurement of one point in time is required, whereas other measurement-based techniques may require capturing at later points in time as well.

Xuey et al. present an appearance manifold approach that works with casual photographs as input and output instead of spatially varying BRDF measurements. Source images are defined as a product of reflectance and illuminance. By statistical means, those are statistically decomposed. The authors provide a scheme for transferring weathering between photographs as well as for weathering and de-weathering of photographs (see figure 2.5) [27].

2.3.2 Context-Aware Textures

Lu et al. present a measurement-based approach to material ageing. Applications to the simulation of effects due to chemical attacks on metals and biological attacks on cheeses are presented in the publication. Extensive and costly physical measurements in multiple weathering states are necessary to capture a new effect. However, once measurement is completed, the results can be re-used in the form of a texture library. The researchers directly capture the time-varying appearance of objects in a precisely con-

Figure 2.4: Wang et. al. present synthesised appearances of the gargoyle in different points in time. The time-varying appearance was derived from the surface sample on the left using the appearance manifold method [26].

Figure 2.5: Xuey et al. demonstrate, based on the source image on the left: de-weathering, weathering and texture transfer on casual photographs [27].

trolled environment and in regular intervals. The result of measurement, apart from a reconstructed mesh, is an unlit texture augmented with *context parameters* obtained from local shape and environmental factors, thus capturing the geometry-dependent nature of weathering processes. The resulting *context-aware texture* can then be applied to the captured model or transferred to other geometry using a patch-based approach, considering time, geometric and environmental factors of the target model and correlating it with measurement data (see figure 2.6) [18].

Since the complete weathering progression is physically captured, long-running processes are infeasible to capture unless artificially accelerated. For instance, the researchers conducted a measurement where a copper pan would be sprayed with constant amounts of chemical agents dissolved in one litre of water every twelve hours for eleven days, successfully covering the pan in patina [18].

Precisely predicting the location where imperfections will form would require microscopic precision in the capture process and would be infeasible for end users to provide for target models. Instead, measurable parameters on a larger scale, which have an influence on the ageing process, the before-mentioned *context parameters*, are both measured and their relationship to the observed weathering process analysed. An example of this interdependence is the property of rust to form faster in areas that are more exposed to the rusting agent. Among the measured local features are:

- *Ambient Occlusion*, as a measure of local exposure to the environment,
- *Source Direction*, the local surface orientation relative to the weathering agent,

(a) **(b)**

Figure 2.6: Using the context-aware textures approach, (a) shows measurement data that has been transferred to the output model (b). Note the large geometric differences between the flat plate and the seahorse model [18].

optionally taking shadowing effects into account,

- *Signed Mean Curvature* is positive on convex and negative on concave local neighbourhoods,
- *Principal Directions* are the directions towards minimal and maximal curvature at a given vertex [18].

As pointed out by the authors, results using this approach can be quite effectively validated by applying a synthesised texture back to captured geometry. The synthesized version can then be compared to the measurement data of the real weathered object. The researchers results were similar, though not identical [18]. Judging by the quite convincing renderings provided with the paper, the approach has been astoundingly successful.

2.4 Phenomenological Models

Where physically-based simulations aim for realism, it instead suffices for a *phenomenological model* to produce results that are *convincing*. A detailed model of the physical nature of the underlying effect is intentionally not provided. This typically leads to less complex models that are applicable to a wider range of effects compared to physically-based approaches. Alternative names for this class of technique sometimes used in the field are "physically-oblivious" [2] or simply "faking". The aforementioned *transfer techniques* in section 2.3 can be seen as a special case of phenomenological models that use some form of physical measurement as configuration data. Since the physical processes that are the cause of weathering effects are often inter-dependent, not fully understood or too complex to simulate [4], the field saw the emergence of several techniques that are oblivious to the underlying cause and instead apply phenomenological models not derived from domain knowledge. These models often achieve quite convincing approximations.

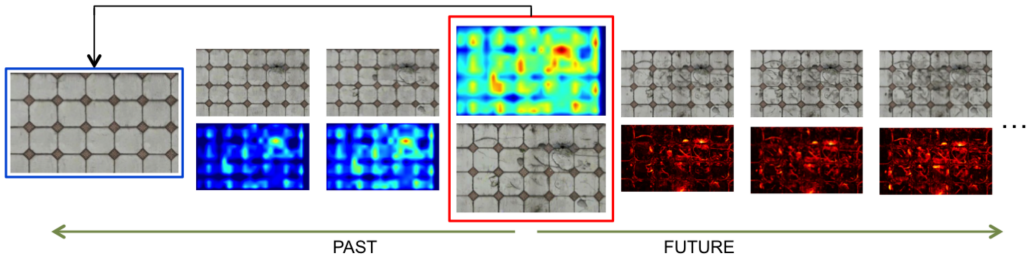


Figure 2.7: Diagram of the texture-space algorithm by Bellini et al., note how artefacts gradually appear instead of blending in [2].

2.4.1 Weathering in Texture Space

In 2016, Bellini et al. proposed a weathering simulation technique generating time-varying texture variants working with only a weathered texture as input, not requiring user input and oblivious to the underlying physical process. The algorithm estimates an *age map* containing the degree of weathering of individual texels of the input texture based on the assumption that weathered regions are less likely to be repeated than unweathered regions. A past *age map* can be interpolated with a threshold-based technique where values above the threshold are reduced, while values below are discarded. Conversely, the age map can be extrapolated for future points in time by scaling of texel weathering degrees. The researchers further propose a technique to synthesize an *intact texture* with apparent signs of weathering removed. Input and intact texture can now be blended with the interpolated age map as a guide to obtain a convincing time sequence with artefacts gradually appearing and small artefacts only starting to appear later on in the time sequence. With a technique similar to the creation of the *intact texture*, textures can be synthesized for either a painted age map or a future extrapolated age map, yielding potential future states of weathering (see figure 2.7) [2].

A particular advantage of the approach is that weathering effects can be directly learnt from provided input textures and, in contrast to similar approaches, such as Appearance Manifolds [26], no human intervention is necessary for the weathering degree estimation [2]. A notable disadvantage of the technique is that the geometry of the weathered object, as well as the geometry of other objects in the scene, is not taken into account during weathering, since the algorithm works exclusively in texture space.

2.4.2 γ-ton Tracing

Weathering-driving and sometimes weathering-resisting factors are modelled with an ageing-inducing particle denoted γ-ton. These are shot from sources in multiple iterations and get traced through the scene (see figure 2.8) while exchanging material with a point-based scene representation. The final concentration of materials in the point cloud is then used to drive the actual weathering effect [4]. The paper recommends multi-texturing and texture synthesis to perform the actual weathering effect on the scene in a second pass using the information in the point-based representation. Application examples for all major attack classes by means of using γ-ton transport information

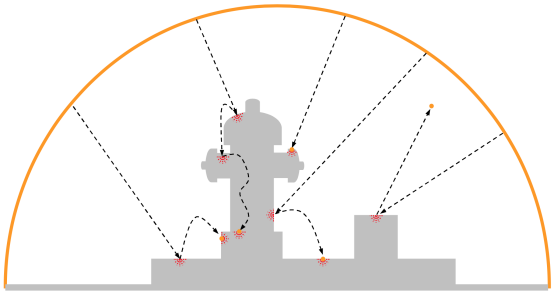


Figure 2.8: A diagram from the original γ-ton publication, outlining how gammatons are shot from a hemispherical environment source on a straight line and traced through the scene in one of three motion states, interacting with geometry [4].

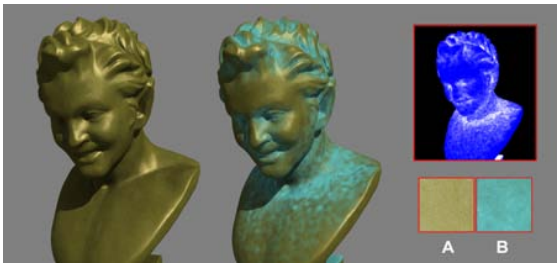


Figure 2.9: Rendering of a weathered object using linear blending of two textures guided by the gammaton map in the top right [4].

for either vertex displacement or for texture blending are provided in the paper (see figure 2.9) [4]. As pointed out in [9], only small geometry changes can be modelled in addition to shading changes, since no model for the representation of the object interior is provided.

Advantages of the approach are the wide range of weathering phenomena that can be modelled with the technique and that results take the whole scene geometry into account. This enables support for *global transport effects*, also referred to as *stain bleeding* (see figure 2.1), giving blemishes the ability to spread from one object to the next [4].

To describe the behaviours of different types of weathering-inducing materials and their sources, many parameters can be configured, including: γ-ton source positions, shapes and emission counts, initial surfel and gammaton attributes, flow distances of γ-tons, transport rules for surfels and γ-tons [4]. Being a purely phenomenological approach, these parameters have no direct equivalent in the real world, making meaningful configuration rather difficult. Finding the right parameters mostly requires some experimentation.

No detailed strategy is provided on when or how to perform appearance rendering using a γ-ton map in combination with the suggested techniques of texture synthesis and multi-texturing. In raytracing, the γ-ton map could be used in its spatially organised form during rendering to synthesize material composition for a given point in a

similar way that a photon map would be consulted for indirect illumination. Another possibility would be to synthesize standard texture maps for rendering ahead of time (albedo, metallicity, reflectivity, etc.). In scenarios where simulation is performed on-line and material concentration should thus change over the course of the program, the material concentration texture could also be used as an input for surface shader programs that then perform weathering on a per-fragment basis. All texture-based approaches, however, would rely on entities in the scene having non-repeating UV coordinates, so each texel of a particular object could be uniquely mapped to a world position. If this is not already the case, either a new set of UV coordinates could be generated for the scene, or material concentration could be saved as the property of vertices in model formats that support custom attributes.

The algorithm shares similarities with classical photon tracing as proposed by Jensen in 1995 [13] but applies them to the fundamentally different domain of weathering [4]. A similar system of dissolved materials in particles being exchanged with surfaces is also present in the flow system by Dorsey (see section 2.2.2), though that system includes more domain-specific knowledge on fluid behaviour and uses textures instead of kD-trees as the primary data structure for surface properties. The *stain bleeding* feature of γ-ton tracing has its equivalent in the *differential flow* of the Dorsey system [7].

Several extensions to the algorithm have been proposed. Jiao and others propose a technique for realistic weathering of fur in the context of raytracing using γ-ton tracing, extending the approach with a physically-based dust accumulation model [14].

João Montenegro Almeida presents a specialized version of the tracing algorithm for large scenes in his Masters Thesis and proposes a workflow for texture-based appearance rendering tailored to the presented tracing method. Instead of sampling the scene into surfels, it is rendered from the viewpoint of multiple cameras, each rendering a depth map and a normal map. The tracing process is then performed on the implicit representation of the geometry provided by the set of depth and normal maps instead of the original mesh. An output camera is used to synthesize an output dirt texture from the point of view of this camera. For best results, the output camera should be identical to the camera later used for rendering [1]. The precision is limited in comparison to the original algorithm and the algorithm only solves the problem for one viewing angle, but can provide approximations for scenes that would otherwise be too large for weathering simulation.

Jiayin and Mingquan propose an improved tracing method using height field profile tracing. Instead of retiling the mesh for displacement, as was done in the original implementation, the tracing process is adapted to a two-reference surface height field [15].

2.4.3 μ-ton Simulation

Joseph T. Kider presents the μ-ton system in his PhD thesis, comparing it to γ-ton tracing. Both systems have conceptual similarities with photon tracing [16].

μ-tons describe the scene in a point-based representation. They come in three main flavours: emitter, decay and material. Further, they may mutate from one type into the other. All three share a common base interface. New decay effects interact with the system through this base interface and can be derived from existing effects [16].

Emitter μ-tons can be located in empty space, inside objects or on the surfaces of objects and encode emission attributes, such as the emitted particle count, their type and the emission shape. During the emission process, emitters divide their energies among emitted child μ-tons, which are then propagated through the scene. This makes emitter energy roughly proportional to the emission count. However, energy may be added or removed to the emitter μ-ton over the course of the simulation. Further, this type of ton defines a time, which controls the emission frequency of different sources. Emitters can not only be placed by artists but also be created as a result of a μ-ton intersection. This powerful features enables the modelling of multi-layer and complex interactions [16].

Decay μ-tons capture the behaviour of a specific decay process such as mould growth and propagate through the scene, inducing material changes. They describe the type of decay and decay-specific attributes as well as sphere of influence size, positional and motion information. The μ-ton propagation process differs from the generalised, probabilistic γ-ton propagation in that the details of motion are specialised over the type of decay and tailored to scene and material structure. For instance, fluids use a smooth particle hydrodynamic, whereas mould growth μ-ton motion is governed by reaction-diffusion equations. The four motion states of γ-ton tracing can also be incorporated for "randomness". Collisions can lead to absorption of the decay μ-ton, diffusion into the geometry or reflection [16].

Material μ-tons represent surfaces and their attributes in a similar way as surfels in the γ-ton system and come in the form of texture and geometry μ-tons. While geometry μ-tons are placed on the surface with a 3D constrained Voronoi Cell method and quite similar to surfels, texture μ-tons define position with respect to a texture that is mapped to the object. Texture μ-tons are better suited for the efficient representation of thin layers of materials. The initial amount of material μ-tons is often low, since new μ-tons can be added in response to an intersection. Certain material μ-tons have *constraints*, which connect them in a force-based lattice system. The strength of the forces is dependent on the material and influences the likeliness of breakage. The lattice system represents the underlying mesh and allows for modification of the mesh by modification of forces by reaction equations to capture some physical and chemical decay effects. Based on the forces, parts of objects may mutate, collapse or break off. Movement of severed parts is facilitated with a rigid body system [16].

μ-tons are collected into time-varying distribution maps. Those can be in the form of point clouds of μ-tons or in the form of textures with white representing maximum point density and black indicating no μ-tons. The maps are used to guide the formation of decay effects. Some decay processes also use lighting information in addition to the distribution maps. In particular, ambient occlusion and normal maps may be of interest for many effects, since they provide means of recognising surface features such as corners, cracks or top faces. In addition to guiding effects, distribution maps also influence the propagation process itself and are used to facilitate interdependent phenomena. An example of such interdependence would be fungal colonies using up nutrients in a fruit [16].

Mutations due to μ-tons can not only affect shading properties, but also geometry. The mesh mutation approach uses the Voronoi lattice system with an accompanying Delaunay representation of its center points and ensures higher tessellation in regions

Figure 2.10: Manually shot γ-tons visibly flow over a surface and interact with it [12].

where more decay μ-tons have interacted with material μ-tons [16].

Effects that can be captured with γ-ton tracing can be modelled in the μ-ton system as well and the systems have large conceptual similarities. Conversely, the μ-ton provides unique features that enable a wider range of ageing effects to be modelled. As pointed out by Kider, a key difference is that ageing effects exhibit "local growth and are not merely transported around" [16]. For instance, μ-tons have the ability to mutate into different types, to diffuse inside geometry and allow for sources to be contained inside geometry. This readily captures the behaviour of organic growth. The geometry representation also exhibits larger differences between the two approaches. While geometry changes in γ-ton tracing rely on displacement maps over retiled high-res meshes, the μ-ton geometry system constantly performs custom triangulation during simulation to better capture splitting effects, crumbling and wrinkling.

2.4.4 Interactive γ-ton Tracing

A technique for accelerated tracing of ageing-inducing particles has been published in 2012 by Günther, Rohmer and Grosch with the main goal of providing a fast ageing simulation with which artists can directly interact. According to the authors, this is the first interactive GPU-accelerated method for material ageing to be published. In contrast to classical γ-ton tracing, which the proposed algorithm is based on, the simulation works at interactive frame rates and the tracing of particles can be directly observed and controlled by the user (see figure 2.10).

Properties of simulation, particles and surfaces can even be changed while the simulation is running and allow artists to precisely control changes in weathering sources over the course of the simulation. Instead of a point sample representation, the authors chose to maintain surface state in a set of textures they refer to as a *material atlas*, providing fast lookups for surface properties and less friction when re-using the resulting textures in external applications. Apart from surface data representation, the approach differs in other details. Notably, it avails russian roulette to determine motion state in favour of a velocity-based scheme in time steps. γ-tons do not settle but are either alive or become immediately available for emission again if they leave the scene or have near-

zero speed. γ-tons in mid-air or just reflected off a surface are considered live [12] and constantly moving, based on their current velocity. The velocity gets reassigned when the path intersects geometry and is influenced by friction. On surface contact, substance is exchanged.

The material atlas also maintains a texel-to-world-scale texture to compensate for distortion and non-uniform scales introduced with texture mapping. Artefacts at texture discontinuities depend on the splat size used for texture painting. If it is larger than one texel in the texture map, artefacts can occur. Normalization of the splat size to one texel in the atlas is shown to produce no visual seams [12]. In a short paper, Frerichs, Vidler and Gatzidis take note of the problem regarding splat size described by Günther et. al and propose a scheme for larger splat sizes. A square is projected onto the geometry with its center point aligned with the intersection point. Any intersections with texture discontinuities are handled by splitting the square into smaller polygons at the discontinuities, assigning two different positions in UV space to the two new edges. The deposition process is then carried out in a compute shader, optionally using a mask texture to support other deposition shapes [10].

Substance Painter[1] is a commercial product for texture painting that employs particle systems to paint directly onto textures or onto textures mapped to 3D objects. The implementation has similar capabilities as the GPU-particles implementation by Günther et al. but aims primarily at painting individual objects instead of weathering full scenes with inter-object substance transport.

[1] https://www.allegorithmic.com/products/substance-painter

Chapter 3

Implementation

This chapter provides a detailed description of the implementation. A variation of the γ-ton tracing approach, described in section 2.4.2, will be developed.

3.1 Overview

The algorithm calculates time-varying appearances of scenes that are subject to weathering sources, such as sunlight, humidity and air. The algorithm is iterative, yielding consecutive states of weathering over longer periods of time. As mentioned in the introduction, the main goal of the implementation is to devise a variation of the γ-ton tracing algorithm for the visual simulation of ageing effects in a texture-based workflow. Specifically, the approach should integrate well with typical 3D asset production workflows.

The term *classical γ-ton tracing* will be used to explicitly refer to the originally proposed version published in 2005, as described in section 2.4.2 of the previous chapter. By contrast, the presented variation will be referred to either as the *aitios* variation, after the name of the reference implementation, or simply as *the implementation* in unambiguous situations.

aitios is primarily based on classical γ-ton tracing. However, it is not a strict superset of it and deviations from classical γ-ton tracing will be pointed out and justified. The implementation further incorporates ideas from the GPU-based γ-ton tracing variation by Günther et al. as described in section 2.4.4 for the tracing of γ-tons in flow.

The core idea of all γ-ton tracing variations that have been mentioned is to attribute the weathering degree of entities to the distribution of simulation-defined weathering-inducing substances [4, 12]. These substances occur in surface point samples, denoted *surfels* and in moving particles, denoted *γ-tons*. The behaviour of γ-tons, surfels and their interactions with each other model changes in the substance distribution over longer periods of time. The distribution of substances over surfels in one instant in time, denoted γ-ton map, implies a state of weathering. Using this distribution information, a weathered appearance is generated with texture synthesis techniques [4]. Note that this publication uses the term *material* exclusively to refer to a named set of attributes associated with an *entity* which define its appearance[1] when rendered or serialized, while

[1] Classical γ-ton tracing uses a different taxonomy where *material properties* describe attributes of

substance describes the amount of one of n substances associated with a surfel.

γ-tons originate from γ-ton sources, such as the sky, which could emit humidity particles to model rainfall, or streets, which could emit exhaust particles to model urban pollution. Being traced through the scene, the particles model the transport of substances, such as humidity, dirt or spores. The tracing leads to exchange of substances between γ-tons and surfels in a process denoted substance transport[2]. After each iteration, the substance distribution information over these surfels is utilised for appearance rendering, where weathered materials are synthesised with frame-to-frame consistency.

3.2 Requirements

The implementation should support most features of classical γ-ton tracing in a variation that is optimized for a texture-based production workflow.

Practical Applicability: The resulting variation should be applicable in practice. This applicability requires resulting textures and scenes to produce convincing results for two reference effects: *rusting corrosion* and *patination*. Efficiency of the approach should not be a major obstacle to its use in practise. The extent to which the implementation lives up to these requirements will be evaluated in chapter 4.

Completeness: A secondary aim of the implementation is to propose working solutions for aspects of the classical γ-ton tracing algorithm that are not specified in detail by the original publication. The description of the presented variation in this thesis strives to attain completeness, up to the point that it can be re-implemented in a consistent way. Aspects that will receive additional clarification here is the precise behaviour of γ-tons in parabolic and flow-like motion, as well as the mode for selection of surfel positions.

Texture Support: Distributions of substances in γ-ton tracing are represented with a point-based data structure, just as in the original implementation [4]. Such points with associated properties, denoted surfels, have also been proposed as a rendering primitive by Pfister et al. [24]. However, modern computer graphics, specifically in gaming contexts, typically work on textured polygonal meshes as rendering primitives. Texture synthesis based on γ-ton maps is mentioned in classical γ-ton tracing, with some examples as well as suggestions for additional techniques to use, such as multi-texturing. They present renderings where the point-based γ-ton map guides the blending between two textures to capture weathering effects. Little details are provided as to how this process works [4]. Instead, the original paper concentrates on the core functionality of calculating the distribution of substances in surfels. This thesis, by contrast, explicitly describes an approach to map the substance information contained in the surfels into texture space. These texture maps can be used by subsequent texture synthesis steps to obtain a final weathered appearance. For these subsequent steps, an effect pipeline grounded on a layering-based synthesis approach is presented. A progression of partly

surfels that hold amounts of a specific substance. γ-tons hold *carrier attributes* instead [4].

[2]Classical γ-ton tracing uses the term γ-transport to describe the same concept [4].

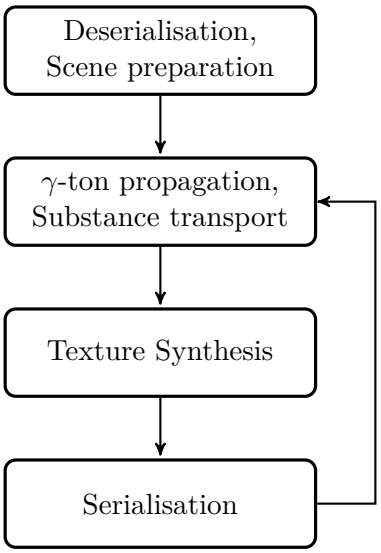


Figure 3.1: A high-level diagram of control flow in the implementation.

transparent blemish textures is applied on top of a base material, guided by the substance distribution, to generate weathered materials. As an alternative to the built-in effect pipeline, the substance textures can directly be exported, and appearance rendering be left to the application used for rendering.

3.3 Architecture

Figure 3.1 provides a high-level diagram of control flow in the implementation. In essence, scenery is first deserialised from files and organised in optimised data structures for surface and geometry. Next, a user-specified amount of simulation iterations is run, which first update the surface according to the substance transport performed by propagated γ-tons, then update the materials of entities with synthesised textures based on the updated surface, and finally serialise the results back to disk.

Result of the implementation is a library *aitios*, that serves as a reference implementation for the presented variation. It can be included into target applications where procedural weathering of scenery is desired. If adding *aitios* as a dependency in the target application is not desired, off-line generation of weathering effects is possible with an accompanying command line tool. It allows for the data-driven running of simulations described in *simulation specification fragments*. These are YAML markup files with partial simulation descriptions that can be combined to form a complete simulation specification. Results of individual iterations are serialized into textures, material files and scenes.

3.3.1 Organisation

Internally, the *aitios* project is split up into three layers:

- geometry,
- simulation,
- application.

Geometry: The geometry layer implements lower-level functionality for geometric primitives, meshes, materials, spatial data structures and sampling strategies.

Simulation: The simulation layer depends on the geometry layer to build core weathering functionality. Point-based surfaces, texture synthesis, and the core γ-ton tracing algorithm are implemented here as independent, though relatively tightly coupled components. The core simulation component traces and propagates γ-tons through scenes to perform substance transport on a surface model. Texture synthesis takes geometry with an associated surface model to synthesise substance textures. These can be further processed into weathered textures derived from the substance textures.

Application: Finally, the application layer provides user-level access to the functionality via simulation specification in YAML files. These can be passed to the command-line tool implemented in the application layer. Alternatively, the application layer exposes specification and instantiation functionality, such that custom frontends can be built that use the same specification format. This feature could be exploited for integration into 3D modelling and rendering software as well as game engines. Other potential use cases are graphical applications with procedural geometry to be weathered at runtime. The application layer is optional and a host application can directly call into the simulation layer instead. This may be useful if specification functionality is not needed, or if aspects of simulation are intended to be replaced with a custom technique.

3.4 Surface Model

To assign spatially varying material and weathering properties to surfaces of entities, the scene is discretised into a point-based representation. Individual point samples are n-tuples referred to as *surfels*. To facilitate performant queries for points near a given position, the set of surfels is organised in a kD-tree. Use of this data structure has also been suggested in [4]. Each surfel lies on the surface of a triangle in the scene and defines a number of associated properties:

- $\vec{s}$, the position of the surfel on a surface in the scene,
- Δ_s, Δ_p, Δ_f, reflectance properties with higher values indicating a rougher surface that traps more γ-tons (see section 3.6),
- $s_0, s_1, \ldots, s_n$, the amount of each weathering-inducing substance currently at the surface point,
- $a_0, a_1, \ldots, a_n$, substance-specific maximum fraction of absorbed material from γ-tons settling near the surface position.

At the end of an iteration, the amount of *substances* in surfels may be used to synthesize a new *material* using the techniques described in section 3.8.

3.4.1 Surface Sampling

Classical γ-ton tracing does not explicitly propose an algorithm for selecting surfel positions in their γ-ton tracing publication. However, when introducing the concept of a surfel, the authors cite the work of Pfister et al. on surfels as rendering primitives [4]. Pfister proposes a scheme where the scene would be scanned with rays from three orthographic views. The surfels would be placed on intersections of the rays with front- and backsides of triangles [24]. Multiple surface sampling approaches have been tested in the development of *aitios*.

Barycentric

Let T be any triangle defined with three vertices, i.e.,

$$T = \begin{pmatrix} \vec{t}_0 & \vec{t}_1 & \vec{t}_2 \end{pmatrix}^\mathsf{T}. \tag{3.1}$$

Osada et al. propose a technique for uniform sampling of such a triangle in an aspect of their work on shape signatures. It can be uniformly sampled with a weighted sum of its vertices. Osada et al. use two uniformly distributed random numbers u and v in range $[0, 1)$ to obtain

$$\mathrm{bary}(u, v) = \begin{pmatrix} 1 - \sqrt{u}, & \sqrt{u} \cdot (1 - v), & \sqrt{u} \end{pmatrix}^\mathsf{T}, \tag{3.2}$$

yielding the scalar weights [23, p. 814]. These can be applied to the triangle in the form

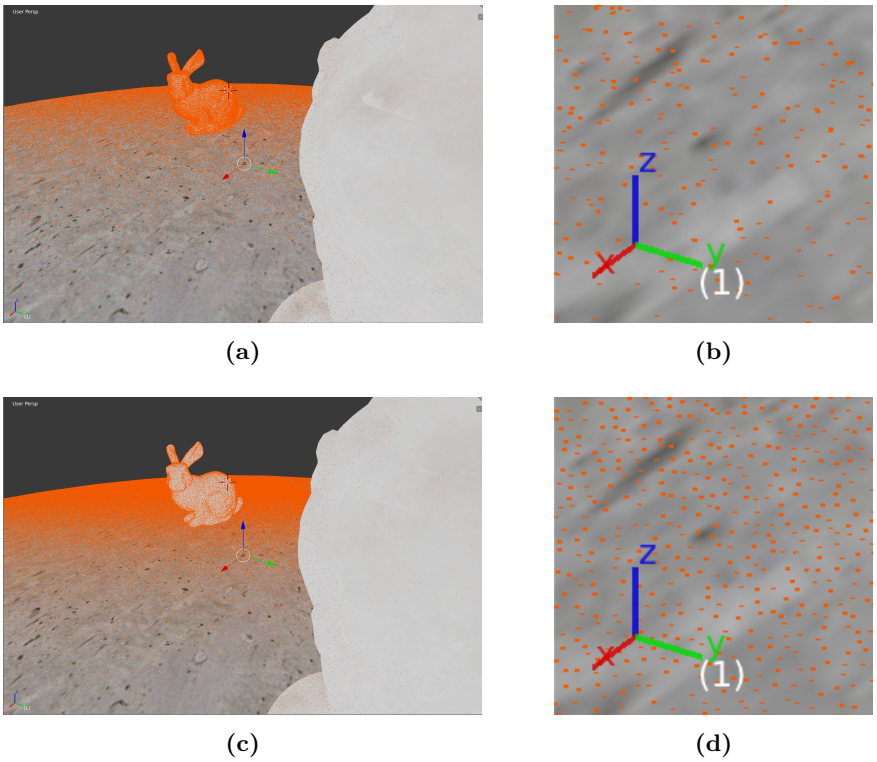
$$\vec{r}(u, v) = \sum_{i=0}^{2} \vec{t}_i \cdot \mathrm{bary}(u, v)_i, \tag{3.3}$$

with the resulting point $\vec{r}$ being uniformly distributed on T.

Now, for each triangle in the scene, n surface points are sampled, where the sample count n is directly proportional to its area A and inversely proportional to a user-defined sample density per square unit of surface area ρ, rounding up to guarantee a whole number and at least one surfel per triangle. n is evaluated as

$$n = \left\lceil \frac{A}{\rho} \right\rceil. \tag{3.4}$$

The area-weighted per-triangle sampling approach is efficient in terms of both time and space complexity. It is a pleasingly parallel problem, where each triangle can be sampled independently. However, the points show a tendency to clump together on smaller scales (see figure 3.2 (b)), leading to statistical bias. This could cause observable artefacts in substance distribution. For instance, given an interaction location and radius, the amount of interacting surfels could vary by a non-trivial amount. Since substances carried by γ-tons are split among interacting surfels in interaction, this made some areas more susceptible to weathering than others, depending on local surfel density. In some cases, the amount of interacting surfels could be zero, in which case the surface is assumed under-sampled, and the interaction process falls back to interacting with the nearest surfel only. For texture synthesis, the technique leads to similar biases when associating texels with surfels, causing artefacts. Note that the ceiling of n also adds some statistical bias. It became clear that an even spacing of points would be desirable



(a) (b)

(c) (d)

Figure 3.2: Two surfel distributions in Blender (a) and (c) with a blowup of the bottom left corner depicted in (b) and (d). The top row is obtained with area weighted triangle sampling, while the bottom row utilises Poisson Disk sampling by dart throwing on surfaces. Both seem quite uniformly distributed at first glance, especially on larger scales. Note, however, that surfels in the bottom row have no tendency to clump together or form holes due to their minimum distance requirement. These properties make results in substance transport and texture synthesis more predictable.

to minimise statistical bias, warranting the use of a new approach. While the initial approach is relatively fast, it provides no guarantees on how surfel positions are distributed on the face of a single triangle.

Dart Throwing

To add additional guarantees and accomplish the goal of low statistical bias, the Dart Throwing algorithm by Cline [5] was implemented as the final solution to the problem. It generates a maximal Poisson Disk set on surfaces made from fragments that can be split, including an optimised version for triangle meshes. Any Poisson Disk set guarantees a minimum distance $2r$ between contained points. Maximal Poisson Disk sets additionally guarantee that no further points can be added without violating the minimum distance constraint. The resulting point spacing (see figure 3.2) was satisfactory for the purposes of the implementation and lead to predictable results.

In the case of triangle meshes, the Dart Throwing algorithm first forms an active

list of triangles. To generate a new point, it selects and removes a triangle weighted by area from the active list and tries to add a uniformly sampled point on its surface to the Poisson Disk set, unless the new point would violate the minimum distance constraint with respect to the already generated points in the set. Regardless of whether a point was added or not, it is checked if the selected triangle is completely covered by the point set. If covered, no new triangles are added to the active list and the next point can be generated. If not yet covered, the triangle is split at edge midpoints and the subset of sub-triangles not covered by the point set is added back to the active list. This process is repeated until no triangles are left in the active list, yielding a maximum poisson disk set [5].

An optimisation scheme for the active list as proposed by Cline et al. is deployed. The active list is organised in logarithmic bins, in which triangles can be sampled, on average, in constant time [5]. The first bin contains the largest triangle in the input mesh with area $A_{\max}$ and all triangles with more than half the area of the largest triangle. The bounds are halved for the next bin [5]. Generally, the bin with index $n \in \mathbb{N}$ holds triangles with A in the interval

$$\left[2^{-n} \cdot A_{\max}, 2^{-n-1} \cdot A_{\max}\right). \tag{3.5}$$

First, a bin is selected with a probability proportional to the combined area of the contained triangles [5]. In the implementation, this is done in a way similar to Russian Roulette in motion probability selection. Let $A_0, \ldots, A_n$ be the combined areas of triangles in the bin with index n. To sample a bin, a uniformly distributed random number r_{-1} in the range

$$\left[0, \sum_{i=0}^{n} A_i\right) \tag{3.6}$$

is selected and seeds a sequence defined for each bin index $n \geq 0$ as

$$r_n = r_{n-1} - A_n. \tag{3.7}$$

The bin with the lowest index n, for which r_n is negative, will be selected. Then, from within the bin n, triangles are selected with rejection sampling. That is, random triangles are selected from the bin, until one is accepted, according to an acceptance probability. This probability is defined by the area of the triangle in question, divided by the upper area bound of the bin. Consequently, the acceptance probability is in range $[0.5, 1)$ and the rejection sampling will terminate after few attempts in the average case [5]. The improved statistical properties of the Poisson disk sampling approach made results during tracing and transport more predictable and seemed to also improve the visual results in texture synthesis.

The first implemented versions of this technique suffered from numerical stability issues, especially on large sets of triangles. To alleviate these problems, per-bin triangle areas and the sum of per-bin triangle areas are both approximated as integer multiples of a common floating point area quantum in *aitios*. The quantum is chosen as the area of the smallest triangle, scaled by a constant factor. This allows bin areas as well as their sum to be cached and continuously updated as triangles are added or removed, without floating point error accumulating and having the actual and cached area drift apart.

Textures as Surface Model

A potential alternative to surfels altogether is present in the GPU-based Günther et al. approach, where surfels are ditched entirely in favour of textures for storage of local surface attributes. Special handling is required to account for texture discontinuities (UV seams) and for non-uniform texel-to-world scales over the texture [12]. The flow simulation model by Dorsey also uses textures as the primary data structure to store sediments on geometry as well as for export of blemishes [7].

3.5 Particle Model

A γ-ton is a particle for transport of weathering-inducing or de-weathering substances in a simulation scene. As with classical γ-ton tracing, trajectories start at a γ-ton source with an initial motion state of straight line movement [4]. After each intersection with geometry, material gets transported between the ton and nearby surfels, motion state probabilities will be deteriorated, and a new motion state will be probabilistically selected until the γ-ton either settles on a surface or leaves the scene bounds. A γ-ton can be defined by the following properties:

- m, the current mode of travel: either straight, parabolic, flow-like, or settled,
- $\vec{s}$, the current position of the particle,
- $\vec{v}$, current direction of travel,
- p_s, p_p, p_f, the current motion probabilities for entering each mode of travel after a surface interaction (see section 3.6),
- r, a radius within which the particle can interact with surfels (see section 3.7).
- $c_0, c_1, \ldots, c_n$, the concentration of each weathering-inducing substance currently held by the particle,
- $k_0, k_1, \ldots, k_n$, the rate of absorption from surfels within r for each substance.

For the motion probabilities, it is required that

$$p_s + p_p + p_f \leq 1, \tag{3.8}$$

such that the probability of settling can be implicitly represented with

$$1 - p_s - p_p - p_f, \tag{3.9}$$

as in classical γ-ton tracing [4].

3.6 Tracing

The tracing process calculates the subsequent interaction locations of a γ-ton with scene geometry. A γ-ton will eventually be removed from the simulation when it either settles or leaves the bounds of the scene. Note that the intersection testing during tracing is carried out directly on the triangulated scene, independent of surfels. On each point of contact with scene geometry, a γ-ton interacts with nearby surfels in a process laid out in detail in section 3.7. This will transport material between ton and surfels and change the probabilities of motion of the γ-ton according to reflectance properties of

the surfels. This motion deterioration process plays a vital role in the tracing process, ensuring that the tracing of a γ-ton will eventually terminate.

3.6.1 State Transitions

Akin to classical γ-ton tracing, a new motion state m' will be selected after each surface contact, the initial state always being set to $m = \text{straight}$. The new state is selected according to the updated motion probabilities p_s, p_p, p_f associated with the γ-ton using a Russian Roulette technique [4]. Given a uniformly distributed random variable $\xi \in [0, 1)$ the transition can be described as

$$m'(\xi) = \begin{cases} \text{straight} & \text{for } \xi \in [0, p_s), \\ \text{parabolic} & \text{for } \xi \in \big[p_s, p_s + p_p\big), \\ \text{flow} & \text{for } \xi \in \big[p_s + p_p, p_s + p_p + p_f\big), \\ \text{settle} & \text{for } \xi \in \big[p_s + p_p + p_f, 1\big). \end{cases} \tag{3.10}$$

3.6.2 Intersection Tests

During tracing, non-settled γ-tons are subsequently moved from one intersection point to the next, as long as they stay within the scene bounds. All three active motion states rely on ray-triangle intersection tests performed with the Möller-Trumbore intersection algorithm [22], accelerated by spatial partitioning of scene triangles in an Octree. Outgoing directions of motion are generally selected uniformly from the surface of a hemisphere with $r = 1$ and its base aligned with the surface of the interacting triangle. This process represents a form of diffuse sampling, frequently used in ray tracing contexts.

Straight

If $m = \text{straight}$, the particle motion can be described as a ray. For newly emitted γ-tons, the ray origin is uniformly selected from a triangle of the emission shape, with diffuse sampling on the triangle surface selecting an outgoing direction for the ray. Depending on user configuration, the initial direction can also coincide with the geometric normal of the triangle at the emission point (see figure 3.3). The scheme for efficiently selecting triangles weighted by surface area from emission shapes is akin to the logarithmic binning approach used in surfel generation. In the second instance of straight tracing, when an already emitted γ-ton is deflected from an interaction point, this point is used as the ray origin. The triangle on which the intersection point is contained further defines a space for diffuse sampling of the outgoing direction.

Parabolic

For $m = \text{parabolic}$, the particle will enter a parabolic trajectory starting at the intersection point $\vec{s_0}$. In classical γ-ton tracing, the "trajectory is piecewise linearly approximated" [4]. For the proposed implementation, a rather simple linear approximation scheme has been devised: The starting direction will be sampled over the upper hemisphere of the selected triangle akin to the straight tracing scheme. This starting direction

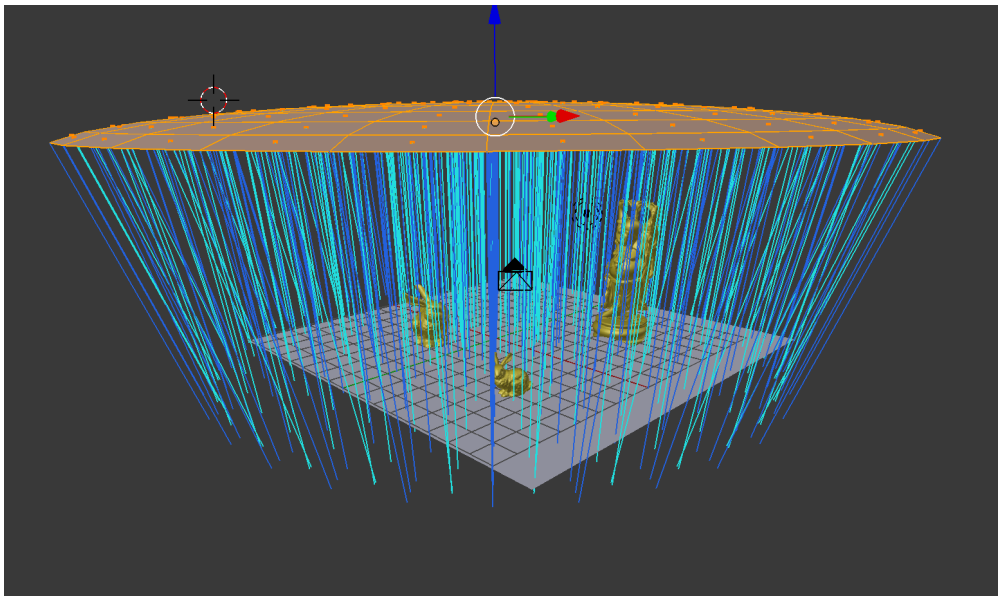


Figure 3.3: Straight, initial trajectories of particles (cyan/blue) emitted from a mesh source.

will be scaled by an indirectly user-specified starting velocity scale k_v, yielding $\vec{v_0}$. The current velocity will be constantly modified by the gravitational acceleration $\vec{g}$, which can directly be specified by the user. The time-varying velocity can hence be described as

$$\vec{v}(t) = \vec{g} \cdot t + \vec{v_0}. \tag{3.11}$$

Using an Euler integration scheme, the time-varying velocity modifies the particle position in fixed time steps, yielding consecutive positions. These positions form perfectly straight line segments, which can be efficiently intersected with the scene Octree until the first intersecting line segment is found. With a sufficiently small time step, this approximates a parabolic trajectory described with

$$\vec{s}(t) = \frac{\vec{g}}{2} \cdot t^2 + \vec{v_0} \cdot t + \vec{s_0}. \tag{3.12}$$

Letting the user directly select the magnitude of $\vec{v_0}$, named k_v, turned out to be somewhat counter-intuitive, since it is not immediately obvious how far the particle would travel, or what the resulting parabola height might be, given a k_v value. Given the idealised form of the approximated parabola $\vec{s}(t)$, a method was derived to specify the k_v indirectly by instead specifying h, the height of the resulting parabola, relative to s_0, in the special case where

$$-\frac{\vec{g}}{\|\vec{g}\|} = \frac{\vec{v_0}}{\|\vec{v_0}\|}, \tag{3.13}$$

that is, the initial direction of motion has the inverse direction as the gravitational acceleration, which is also the constellation where the parabola has maximum height.

A velocity magnitude k_v can then be calculated for this special case as

$$k_v(\vec{g}, h) = \|\vec{v_0}\| = \sqrt{2 \cdot \|\vec{g}\| \cdot h}. \tag{3.14}$$

Flow

Finally, m = flow describes flow-like motion originating from an interaction location. Classical γ-ton tracing describes the behaviour of flow as moving in tangential direction by a small delta and again interacting with near surfels [4].

A naïve interpretation of this rather vague definition would be to scale a randomly selected direction on the tangential plane of the current triangle by a fixed distance, yielding a delta in Euclidean space. This definition, however, leaves the behaviour unclear in cases where a patch boundary is crossed. Specifically, the particle might end up in mid-air or inside geometry, depending on local surface curvature. Another problematic situation occurs when flow follows the surface of an object that is partly stuck inside another object. In this case, the particle should transition to the outer surface instead of entering it.

The primary direction of flow is determined by projecting an incoming direction $\vec{v}$ onto the tangential plane of the triangle that the flow event originates from. Geometric tangent, binormal and normal of the triangle are used to form an orthonormal basis, that in turn is combined into a matrix $\boldsymbol{T}$ that tansforms world-space directions into tangent space of the triangle. Now, to obtain a normalized flow direction $\hat{v}_t$, that is aligned with the tangential plane of the triangle, $\vec{v}$ is transformed into tangent space, the component in normal direction is dropped, and the result is transformed back into world space to be normalized again, evaluating

$$\begin{aligned} \vec{v_t} &= \boldsymbol{T}^{-1} \cdot \left[\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \cdot \boldsymbol{T} \cdot \vec{v} \right], \\ \hat{v}_t &= \frac{\vec{v_t}}{\|\vec{v_t}\|}. \end{aligned} \tag{3.15}$$

Instead of directly using this idealised flow direction, the actual trajectory describes a sawtooth-like function over surfaces, described with two adjacent line segments and one ray, tested for intersection in sequence. The technique is a modified version of the flow technique employed by Günther et al., that aims to add the ability for γ-tons to better float around some overhangs and further guarantees that the point of interaction will be located on the surface of the underlying mesh used for intersection testing [12]. The technique does not require adjacency information for triangles and is reasonably efficient. It further prevents particles from accidentally entering the interior of objects when following the surface of other objects, given that the overlapping objects do not have holes in their meshes. The user configures a flow distance for a perfectly flat patch of geometry $\|\Delta\vec{s}\|$, as well as a vertical offset ϵ with higher values corresponding to a higher tendency to stay in contact with the surface, while lower values make it easier for the γ-ton to enter secondary flow. According to $\|\Delta\vec{s}\|$ and ϵ, the actually travelled distance will be shorter for concave local curvature, but longer for a convex neighbourhood.

Evaluating the target of a flow event, first, the ϵ offset in normal direction is added to $\vec{s_0}$, the origin of the flow event. This is described as a line segment with the parameter

$t \in [0, 1]$, equivalent to the GPU variation [12], i.e.,

$$f_0(t) = s_0 + \epsilon \cdot t. \tag{3.16}$$

In some cases, such as for γ-tons in small cavities, this might already result in an intersection, which is understood by the implementation to be the next interaction location. Otherwise, $f_0(1)$ marks the origin of a second line segment $f_1(t)$ that ends in the expected target point for flat geometry at $t = 1$, where $t \in (0, 1]$, defined as

$$f_1(t) = f_0(1) + t \cdot \sqrt{\epsilon^2 + \|\Delta\vec{s}\|^2} \cdot \frac{f_0(0) + \hat{v}_t \cdot \|\Delta\vec{s}\| - f_0(1)}{\|f_0(0) + \hat{v}_t \cdot \|\Delta\vec{s}\| - f_0(1)\|}. \tag{3.17}$$

If an intersection is detected, it is selected as the next point of interaction. Otherwise, with any $t > 0$, the flow event continues on a straight trajectory towards the direction of gravity as

$$f_2(t) = f_1(1) + t \cdot \vec{g}, \tag{3.18}$$

simulating the effects of secondary flow. The effects of gravity are usually only considered when contact with the surface is lost. Nonetheless, the technique quite accurately mimics the effects of flow, especially on complex models. *aitios* additionally gives the user the option to project the direction of gravity instead of $\vec{v}$. This feature leads to more interesting flow patterns on extremely flat geometry such as walls, while on complex models, the effect of the change is rather subtle.

A potential alternative would have been a variation of the flow system employed by Dorsey et al. in their work on flow [7]. Given patch adjacency information, the particle could travel a pre-configured geodesic distance on the mesh, re-projecting velocity on the new triangle when crossing an edge, until the exact geodesic distance is travelled. This potentially leads to more predictable results. Furthermore, it would naturally and more accurately than the Guenther method capture the real-world effect of bumpy surfaces slowing down flow, since bumpiness would imply a greater geodesic distance to be travelled. This alternative technique would require modification to be applicable to some non-manifold topologies. For instance, a mesh containing a single triangle without a back side has no defined neighbour for its edges. To account for secondary flow, given an abrupt change in surface normal, one could use a landing position texture akin to Dorsey et. al. or transition to a parabolic trajectory.

3.7 Interaction

The original γ-ton tracing publication does not specify with how many surfels a γ-ton will interact, though the use of the plural “interacting surfels” suggests more than one interacting surfel. Further, the usage of a kD-tree is recommended, which facilitates fast queries for both the n nearest entries or the entries within a distance around a specified point [4].

In the implementation, it was decided to select surfels within an interaction radius r, instead of a fixed number of surfels. The rationale was that the surfel density could be increased for more detail without also affecting the interaction radius of γ-tons. If the n nearest surfels were selected instead, generating more surfels would effectively decrease the interaction radius implied by n.

Surfels with normals that differ by more than a critical angle from the normal at the impact location are rejected for interaction. This avoids glitches on very thin materials where surfels from the back side could be influenced by impacts on the front side when only the distance between impact point and surfel is considered.

More formally, given an interaction location $\vec{s_i}$, nearby surfels will be selected for interaction by Euclidean distance. All surfels at a position $\vec{s_s}$ with a lower distance than the interaction radius r of the interacting γ-ton will be applicable, except if the surfel normal $\vec{n_s}$ differs in orientation by at least a critical angle Θ from the normal at the interaction location $\vec{n_i}$. For a surfel to be selected for interaction, it must hold true that

$$\|\vec{s_s} - \vec{s_i}\| < r \quad \text{and} \quad \vec{n_s} \cdot \vec{n_i} < \cos(\Theta). \tag{3.19}$$

3.7.1 Motion Deterioration

In a first attempt, the probabilities of motion of an interacting, non-settled γ-ton deteriorated compatible to the deterioration equations in [4] of the form

$$\begin{aligned} p_s' &= \max(p_s - \Delta_s, 0), \\ p_p' &= \max(p_p - \Delta_p, 0), \\ p_f' &= \max(p_f + p_p' - \Delta_f, 0). \end{aligned} \tag{3.20}$$

From the perspective of the author, the rationale behind the special treatment for p_f is unclear. One consequence is that any γ-ton that is capable of deflecting more than once, where $p_p > \Delta_p$, will possibly also enter a transition into flow, even when p_f has initially been zero. As a consequence, it is not possible to both move exclusively on parabolic trajectories, and deflect more than once. Exclusively straight trajectories, on the other hand, are possible.

The equations also seem to cause unsoundness in the approach with respect to equation 3.9, requiring the sum of probabilities to be less than or equal to one, while no matching constraint is defined for reflectance probabilities. As an example for the unsoundness, consider a gammaton with $p_s = 0$, $p_p = 0.8$, $p_f = 0.2$ interacting with a surfel that has $\Delta_s = 1$, $\Delta_p = 0.1$, $\Delta_f = 0.1$. In this case, $p_s' = 0$, $p_p' = 0.6$, $p_f' = 0.7$, which clearly conflicts with equation 3.9. The unsoundness can be resolved by limiting p_f to a maximum of $1 - p_s - p_p$, i.e.,

$$\begin{aligned} p_s' &= \max(p_s - \Delta_s, 0), \\ p_p' &= \max(p_p - \Delta_p, 0), \\ p_f' &= \min(\max(p_f + p_p' - \Delta_f, 0), 1 - p_s' - p_p'). \end{aligned} \tag{3.21}$$

A user unaware of this flow intricacy might falsely attempt to create a γ-ton that moves only on parabolic trajectories and give it a $p_p = 1$, while defining the reflectance properties of all surfels with $\Delta_p = 0.2$ and $\Delta_f = 0.0$, expecting no flow events at all. If a γ-ton happens to bounce twice in such a simulation, Δ_f will remain at a value of 1, making it impossible for the γ-ton to settle. If the γ-ton cannot escape the scene bounds in flow, the tracing will never terminate. To ensure termination, an upper bound should be established for the depth of tracing events.

Also note that a γ-ton typically interacts with more than one surfel. Nearby surfels are expected to have similar reflectance, except on material boundaries, where surfels built from distinct materials may be mixed. The actual reflectance properties used for deterioration should be averaged over all interacting surfels, such that these edge cases are properly handled.

3.7.2 Substance Transport

Absorption and sedimentation are described with linear equations. The equations are similar to the γ-transport equations proposed in [4], of the form

$$a \leftarrow a + b \cdot k, \tag{3.22}$$

but modified to support multiple interacting surfels, negative absorption rates and other features.

According to the transport equations, γ-tons will interact with surfaces on each point of contact until settlement. For each substance with index i, an absorption rate a_i as well as the amount of dissolved sediment s_i is defined for the γ-ton. Surfels, on the other hand, define a local concentration of sediment c_i and its deposition rate k_i. To disambiguate the values of all n surfels, the index of the surfel is notated as an additional index, $c_{i,j}$ and $k_{i,j}$.

Two transport modes are implemented, namely *classic* and *consistent* transport. The former resembles the transport rules used for the patina example in the original γ-ton publication. If the particle bounces, it will first absorb substance, and once it settles, it will dispose of some or all of its substance. The latter performs transports consistently accross surface contacts, including the last surface contact where the γ-ton settles, running absorption and deposition as competing processes. *classic* transport seems to be most useful when a substance prefers cavities, such as moss that colonises cracks in a floor, while *consistent* transport seems to provide better results for flow artefacts and corrosion.

Classic Transport

On each non-settling stop, the γ-ton will *absorb* sediments from the surface, for each sediment i and an amount of interacting surfels $n \geq 1$, the absorption process can be described as

$$s_i' = \max\left(0, s_i + \sum_{j=0}^{n} \frac{1}{n} \cdot a_i \cdot c_{i,j}\right). \tag{3.23}$$

To conserve the amount of material currently in the simulation, the same amount must be removed from the individual surfels in the form

$$c_{i,j}' = \max\left(0, c_{i,j} - \frac{1}{n} \cdot a_i \cdot c_{i,j}\right). \tag{3.24}$$

On these intermediate stops, the surface usually will not receive any substance unless the γ-ton absorption rate is negative. Positive absorption rates lead to increased substance concentrations in inaccessible areas, such as cracks. Negative absorption rates, on the other hand, lead to higher concentrations in areas with high accessibility. In any case,

transported substance is guaranteed to be removed from its source γ-ton or surfel, as to guarantee a constant amount of sediment in the scene during transport.

When a γ-ton finally runs out of motion probability and settles, it will *deposit* some or all of the absorbed substance onto n nearby surfels. This disposal adheres to a substance-defined deposition rate associated with each surfel k_i, such that

$$c'_{i,j} = c_{i,j} + \frac{1}{n} \cdot k_{i,j} \cdot s_i. \tag{3.25}$$

Since the γ-ton is removed from the simulation after settling, its sediments need not be updated as in the absorption process shown above. A typical value for k_i is $k_i = 1$, leading to all dissolved substances being deposited at the settlement location. This makes sense for water drops with dissolved sediments that would be left behind once the water is fully evaporated. Values lower than one can be used for γ-tons with cleaning effects, while $k_i > 1$ can be used as either an unnatural exaggeration effect or to model organic growth.

Consistent Transport

In some simulations, the direction of substance exchange should not be determined by motion state, but instead by the combination of γ-ton and surface material, using the relative absorption and deposition rates. This differential style of transport is referred to as *consistent transport* within the implementation, since the equations themselves are unchanging with respect to the motion state. The substance transfer occurring from the γ-ton to $n \geq 1$ surfels using consistent transport can be described as

$$c'_{i,j} = \begin{cases} c_{i,j} + \left(k_{i,j} - a_i\right) \cdot s_i & \text{for } \left(k_{i,j} - a_i\right) \geq 0, \\ c_{i,j} + \left(k_{i,j} - a_i\right) \cdot c_{i,j} & \text{for } \left(k_{i,j} - a_i\right) < 0. \end{cases} \tag{3.26}$$

with the substance amount in the γ-ton changing respectively in the form

$$s'_i = \begin{cases} s_i - \sum_{j=0}^{n} \left(k_{i,j} - a_i\right) \cdot s_i & \text{for } \left(k_{i,j} - a_i\right) \geq 0, \\ s_i - \sum_{j=0}^{n} \left(k_{i,j} - a_i\right) \cdot c_{i,j} & \text{for } \left(k_{i,j} - a_i\right) < 0. \end{cases} \tag{3.27}$$

This mode of transport seems more suitable to generate flow artefacts with γ-tons of high p_f, especially for the modelling of corrosion effects. The original intention behind consistent transport was to provide a means of changing the direction of transport depending on material.

3.7.3 Ageing Rules

Substance concentrations are not only influenced by the transport process, but also by static ageing rules that influence surfels independently and irrespective of accessibility, curvature or other geometric factors. They come in two scopes–*simulation-scoped* and *material-scoped*–as well as in two styles: *transfer* and *deteriorate*-rules.

An important use case of *material-scoped* rules is to model substances that only occur when another substance is present on a specific material. For instance, increased humidity induces corrosion on metals, but does not lead to the same reactions if present

on plants, concrete or plastic. Now consider an exterior scene where an iron barrel stands on top of a concrete floor and both objects are exposed to rainfall. This could be modelled with one substance for humidity and one additional substance for rust. The objects are shot with humidity-bearing γ-tons from a source tied to the sky. A material-scoped ageing rule that is specific to surfels on iron materials would then convert stagnant water on the barrel to reddish, iron-specific rust, with more water leading to faster rust accumulation rates. Since the rule is specific to iron, the floor will not rust, even if water is present. Note that this solution is more powerful than ditching humidity and instead having rust fall from the sky directly, using different absorption/deposition rates for the iron and concrete materials. Accurate stain bleeding requires the transfer rule approach: the water drops can still absorb some of the rust on the barrel and flow onto the floor, depositing rust on the floor, even though the floor itself is not made of iron and cannot rust by itself.

In the implementation, this style of inter-substance ageing is denoted a *transfer* ageing rule, encoding a simple chemical reaction. It deteriorates the concentration of one substance i, adding an equal amount of another substance j, according to a transfer rate k, i.e.,

$$\begin{aligned} c_i' &= c_i - k \cdot c_i, \\ c_j' &= c_j + k \cdot c_i. \end{aligned} \tag{3.28}$$

Apart from transfer, the gradual reduction of substance concentration over time can also be modelled with a static ageing rule. The prime use case is the evaporation of water over time, which the implementation calls a *deteriorate* ageing rule. Making the evaporation rule *simulation-scoped* establishes a global evaporation rate for all materials, while *material-scoped* rules could differentiate between concrete and sponges in evaporation rate. Deterioration can be defined as

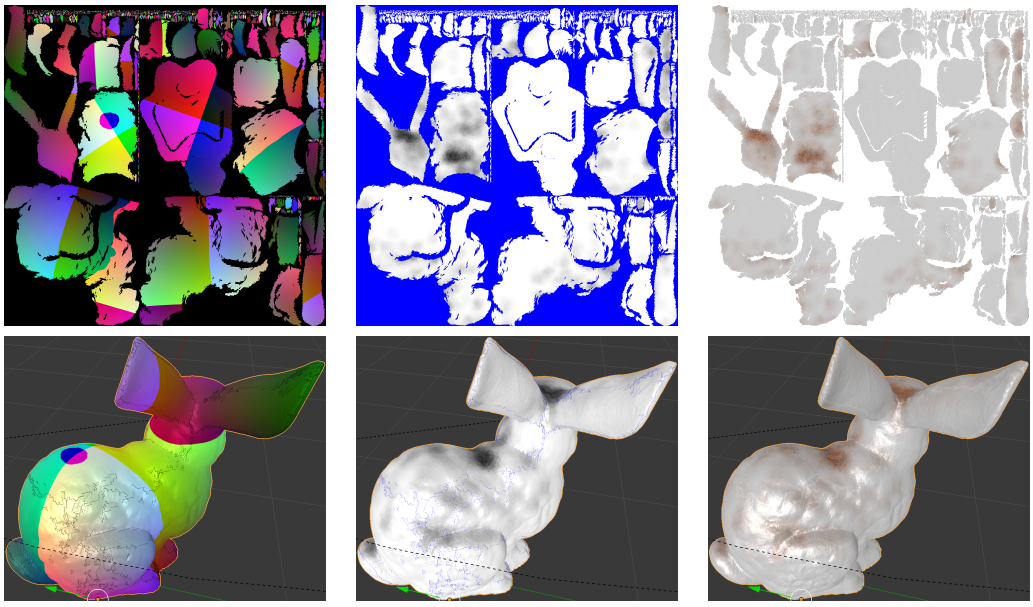
$$c_i' = (1 + k) \cdot c_i. \tag{3.29}$$

Custom ageing rules similar to the just presented types are also present in the chain example in [4]. More complex static ageing rules are thinkable, which would include geometric factors such as surfel normal or accessibility. For instance, as pointed out by Lu et al., sunlight exposure and shadowing influence evaporation rate [18]. Pressure points on fruit are more susceptible to biological attacks [17]. Such phenomena could be represented in ageing rules, given geometric context parameters.

3.8 Texture Synthesis

Once a γ-ton tracing iteration is complete, the substance concentration information stored in surfels can further be used for texture synthesis of weathered materials. Per entity, the texture synthesis process involves the following steps:

1. Calculate a unique texture for the entity where each texel encodes the mapped world-space position of the center of the texel on the surface of the entity (see figure 3.4 (a)).
2. For each texel in the world-space position texture, look up the nearest surfels to the world-space position and store them in a surfel association texture of the same dimensions.



(a) (b) (c)

Figure 3.4: Fractional part of position texture, rendered as RGB (a), substance texture with black pixels indicating maximum substance amount, built from the position texture and the current substance distribution (b), texture blended on the basis of the substance texture (c).

3. For all substances in the simulation, look up the amount of substance of each associated surfel for a texel and calculate a combined and filtered substance amount which is again stored in a *substance map* as the basis for the actual synthesis of the textures used for rendering of the entity (see figure 3.4 (b)).
4. Guide further texture synthesis with the *substance map* to obtain synthesised maps for albedo, roughness, metallicity, displacement and normal, according to a pipeline of layering effects (see figure 3.4 (c)).

Per entity, the core functionality in *aitios* calculates one *substance map* for each substance in the simulation. There are two main ways of utilising these maps for the last step, obtaining a final weathered appearance.

The first option is to load the resulting substance map into the target environment (modelling applications, games, etc.) and to perform further processing there to calculate the final weathered appearance. In this approach, substance amount calculation and appearance rendering are decoupled and carried out in distinct environments, making the approach quite flexible. Possible target environments would be node setups in *Blender Cycles*, *Unreal Engine* or *Substance Designer* or fragment shaders in *OpenGL* or *Vulkan*-based interactive applications. The target environments use the substance map texture as an input for on-line texture synthesis of the final weathered appearance.

Alternatively, *aitios* provides built-in texture synthesis methods based on an effect pipeline of blending-based operations. These augment the original textures in the input scene with weathered texture samples depending on local substance concentration to

obtain modified texture maps for albedo, metallicity, roughness and other texture maps typically used for physically-based rendering. The newly obtained texture maps are exported and a new version of the input scene with modified materials can be exported after each iteration of the simulation as an OBJ/MTL file pair.

Both the built-in functionality for synthesis of texture maps for physically-based rendering as well as recommendations for external appearance rendering in Blender are further described in section 3.8.4 on appearance rendering.

3.8.1 Position Texture

The first step of a texture synthesis pipeline is the calculation of a unique world-space position texture (see figure 3.4 (a)) for each entity in the simulation. To obtain the texture for an entity, each of its triangles is first transformed to its equivalent in the texture coordinate plane. This is done by setting the vertex positions to their associated positions in UV space, scaled by the size of the output texture, and associating the original world-space position as an attribute to each vertex. If the resulting triangle has clockwise winding order with respect to the unit vector in positive z direction, the first two vertices are flipped. The transformed triangles are then rendered into a framebuffer of texture size with the half-space rasterisation method [20]. For each rasterised pixel, the world-space position will be obtained with barycentric interpolation of the known world-space positions on the triangle vertices.

Additionally, *island bleed* of user-defined width in pixels is added. This feature extrapolates world-space positions near triangles that have no direct neighbour in UV space, that is, at texture discontinuities. This step avoids bleeding artefacts at UV island margins. Further, it avoids problems with triangles of sub-pixel area in texture space or with triangles that span multiple pixels but degenerate into line segments in UV space with no or almost no area.

Given a simulation workload without displacement, a memory-for-speed optimisation is applied. In such cases, the world-space position texture can be cached after the first rasterisation and does not need to be recalculated for the remainder of the simulation.

3.8.2 Surfel Association Texture

The surfel association texture defines a mapping of each texel of the output texture to a set of associated surfels. This mapping might change from iteration to iteration if displacement maps are involved. The referenced surfels, in turn, each define a point sample of local substance amount. Given a world-space position texture of the same size as the output texture and a kD-tree of surfels, a surfel association texture can be calculated. For each texel in the corresponding position texture, a number of surfels is selected depending on user configuration. Either, a fixed number of nearest surfels is selected, or all surfels within a given distance are. References to the associated surfels are stored in the texels of the surfel association texture. If the world-space position texture can be cached, surfel associations can also be cached.

3.8.3 Substance Texture

To estimate a substance amount for a given texel of a surfel association texture, the point samples of substance amount are averaged over the associated near surfels, optionally applying a weight factor, resulting in the final substance amount of a texel.

A substance texture can be calculated for each combination of entity and substance in the simulation. Given a surfel association texture and one type of substance, let c_i be the concentration of the same substance for surfel i of n associated with the texel. The unfiltered texel concentration can be obtained with the arithmetic mean, i.e.,

$$\frac{1}{n} \cdot \sum_{i=0}^{n} c_i. \tag{3.30}$$

To attenuate the influence of γ-tons that are farther away and make concentration changes less abrupt, the texture can optionally be filtered. For this purpose, consider the distances between the represented texel center and the actual positions of the associated surfels as d_i, with a maximum of $d_{\max}$ to obtain a weight w_i, shaped by a constant k. The weights are defined as

$$w_i = \left(\frac{d_{\max} - d_i}{d_{\max}} \right)^k. \tag{3.31}$$

The resulting weight is in the range $[0, 1]$, with a higher value indicating higher influence. The k parameter shapes the falloff, e.g., $k = 0$ gives every surfel the same weight and $k = 1$ represents a linear falloff. The weights need to be scaled in the form

$$\hat{w}_i = \frac{w_i}{\sum_{j=0}^{n} w_j}, \tag{3.32}$$

such that their sum is one. The scaled weights can then be applied to the surfel concentrations to obtain a filtered amount as

$$\sum_{i=0}^{n} \hat{w}_i \cdot c_i. \tag{3.33}$$

3.8.4 Appearance Rendering

In a final step, n substance concentration textures of an entity are combined into a set of texture maps used for rendering. This step is referred to as *apperance rendering* and can be performed with the built-in effect pipeline of *aitios*, or externally, with only the substance textures synthesized by *aitios* and further texture synthesis performed in a different environment.

Effect Pipeline

The weathering model in *aitios* is based on the understanding that weathering is a degradation process starting when an object is exposed to environment conditions after its creation. The originally mostly clean and smooth reference surface only contains manufacturing-related blemishes. This surface is subject to physical, chemical and biological attacks as outlined in *Attack Classes* in section 2.1.1 over the course of the

simulation. The attacks are attributed to the weathering-contributing substances stored in substance maps. A sequence of effects, denoted an effect pipeline, modifies the base material textures guided by the substance texture associated with the effect.

Each effect in the pipeline applies a new layer to the base material to simulate the combination of attacks the base material is subject to. For this, the effect defines an associated substance type to obtain a substance texture. Also, a progression of *blend stops* can be defined for each of albedo, roughness, metallicity, displacement and normal. Each of these blend stops defines a weathered texture sample and an associated *cenith*, which is the substance amount where the sample has maximum influence. Given each substance amount texel, one or two blend stops can be selected. If the substance amount c is lower than the first cenith or higher than the last cenith, the blend layer colour can be directly sampled from the first, respectively last, blend stop sample. In all other cases, the blend stop with the highest cenith lower than or equal to the local substance amount c_0 is selected, along with the blend stop with the next-highest cenith c_1. An alpha value for interpolation between the blend stops can be calculated as

$$\alpha = \frac{c - c_0}{c_1 - c_0}. \tag{3.34}$$

A texel at the same UV offset as the substance texture texel is sampled from the sample texture of each relevant blend stop. The texture samples are usually partly transparent such that the base material can shine through. Let the sample at the lower blend stop be a colour $\vec{s_0}$ and the sample at the higher blend stop be $\vec{s_1}$. For all of albedo, roughness, metallicity and displacement, an adequate intermediary value s between the blend stops can be obtained with linear blending as

$$s = (1 - \alpha) \cdot \vec{s_0} + \alpha \cdot \vec{s_1}, \tag{3.35}$$

and used as a colour in the blending texture. Each effect in the pipeline generates a blending texture in this way. These blending textures are blended on top of the original texture of affected objects in the order they were defined. The modified textures are exported and versions of the scene with weathered materials applied can be exported.

While linear blending provides meaningful results for texture maps that directly encode a colour or offset, linear blending is not admissible for normal maps, where instead a direction is encoded. To interpolate between normal blend stops, the colours are first transformed to tangent-space directions $\vec{n}_0$ and $\vec{n}_1$, assuming values in the range $[0, 1]$ for the individual colour channels of $\vec{s_0}$ and $\vec{s_1}$ and the usual convention of mapping the colours to the range $[-1, 1]$ with blue values always being greater or equal than 0.5, such that the texture normal is within 90 degrees of $\hat{z}$, the geometric normal in tangent space. The normal can be obtained from the colour value as

$$\begin{aligned} \vec{n}_0 &= 2 \cdot \vec{s_0} - \begin{pmatrix} 1 & 1 & 1 \end{pmatrix}^\mathsf{T}, \\ \vec{n}_1 &= 2 \cdot \vec{s_1} - \begin{pmatrix} 1 & 1 & 1 \end{pmatrix}^\mathsf{T}. \end{aligned} \tag{3.36}$$

Then, to interpolate directions n_0 towards n_1 by α, partial derivative blending [29] can be applied as

$$\vec{n}_d = \begin{pmatrix} (1 - \alpha) \cdot \frac{1}{\vec{n}_{0,z}} \cdot \vec{n}_{0,x} + \alpha \cdot \frac{1}{\vec{n}_{1,z}} \cdot \vec{n}_{1,x} \\ (1 - \alpha) \cdot \frac{1}{\vec{n}_{0,z}} \cdot \vec{n}_{0,y} + \alpha \cdot \frac{1}{\vec{n}_{1,z}} \cdot \vec{n}_{1,y} \\ 1 \end{pmatrix}, \tag{3.37}$$

yielding an unnormalised vector. A vector of unit length is obtained as

$$\hat{n}_d = \frac{\vec{n}_d}{\|\vec{n}_d\|}. \tag{3.38}$$

This form provides adequate results for blending between materials, as pointed out in [29], and also seems to be a good fit for the similar problem of blending between blend stops for normal maps. For the adding of the resulting $\hat{n}_d$ to the base material, however, the technique provides inadequate results with unintended flattening of geometric detail. The underlying problem is different, since the resulting normal $\hat{n}_d$ poses a detail normal that should be added on top of the base normal, sampled from the original material $\vec{n}_b$. As a specific constraint, details of the original normal map should be retained and augmented with $\hat{n}_d$, rather than being flattened out. The partial derivative blending just shown does not always produce satisfactory results for this step, since it masks the base texture with high alpha values instead of augmenting it with more details.

Better results are achieved with the reoriented normal map blending technique as described in [29], which is tailored to the described problem. We build a transformation $\boldsymbol{T}$ such that

$$\vec{n}_b = \boldsymbol{T} \cdot \hat{z}, \tag{3.39}$$

that is, it rotates the geometric normal, always being $\hat{z}$ in tangent space, towards the tangent normal $\vec{n}_b$ from the base material. We finally calculate $\vec{n}$ by applying the same transformation to the detail normal $\hat{n}_d$ instead, i.e.,

$$\vec{n} = \boldsymbol{T} \cdot \hat{n}_d, \tag{3.40}$$

yielding the final weathered normal to use in the synthesised appearance. This models the concept of adding blemishes on top of a clean reference surface. As a last step, $\vec{n}$ is converted back into a colour $\vec{s}$ with its components in range $[0, 1]$ by reverting the previously defined transformation from colour to direction, evaluating

$$\vec{s} = \frac{1}{2} \cdot \vec{n} + \begin{pmatrix} \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{pmatrix}^{\mathsf{T}}. \tag{3.41}$$

External Appearance Rendering

Figure 3.5 shows a setup with Blender nodes. By applying the substance concentration texture pixel-wise to gradient functions, new textures are created without the aid of a sample texture. Although *aitios* can generate the same textures with fully opaque blend stops of constant colour, the specification of such gradients in configuration files would be quite tedious. After modification of the effect pipeline, the simulation inevitably needs to be run again. Blender, in contrast, allows for rapid iteration after the substance textures have been calculated once. By modifying the gradients within Blender, new variations can be tested in rapid sequence, using the excellent preview capabilities of Blender Cycles.

3.8.5 Texture Irregularities Compensation

A polygonal mesh in three-dimensional Euclidean space consists of a set of faces, with each face defining a set of vertices in three-dimensional object space. A texture map

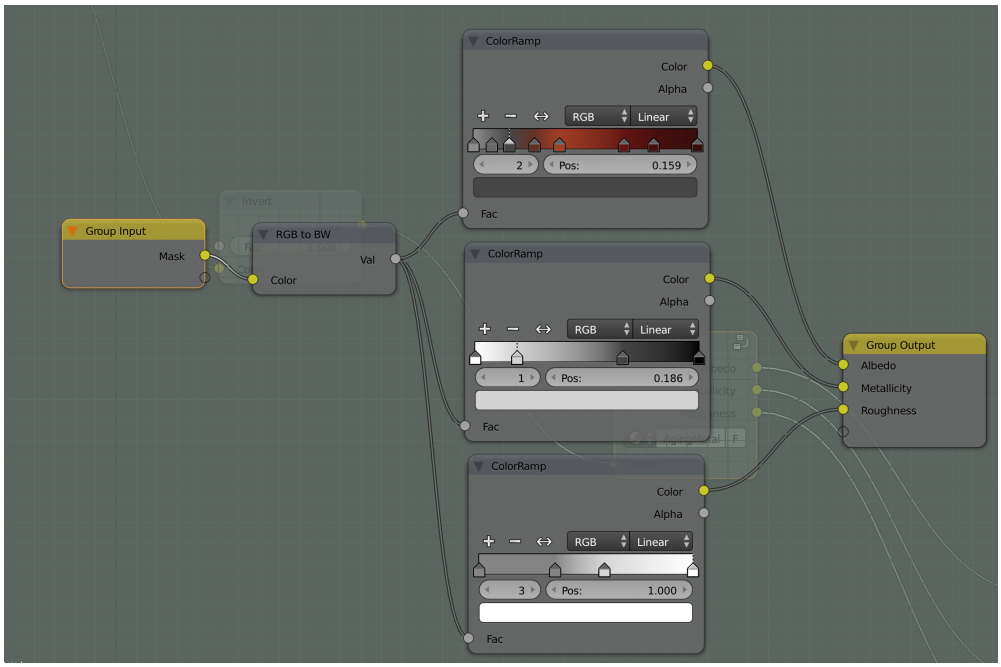


Figure 3.5: A node-based setup in Blender for the synthesis of textures for rusted iron. It takes a substance concentration map calculated with *aitios* as input. Output are maps for albedo, metallicity and roughness.

additionally assigns a two-dimensional position in the Euclidean plane to each vertex, such that each vertex has an equivalent in texture space.

Texture maps for meshes in three-dimensional Euclidean space must exhibit at least one of two kinds of irregularities, given that not all faces lie on a common plane in object space:

1. non-uniform texel-to-world scale,
2. texture discontinuities.

Non-uniform texel-to-world scale arises from distortion when projecting non-flat three-dimensional geometry onto a plane for UV mapping. This process is often referred to as UV-unwrapping. To minimise these irregularities, the primitives of a mesh can be divided into projection groups. Each group is individually projected onto the plane, such that the projected groups do not overlap. Edges between projection groups appear at two distinct places in UV space and are referred to as *texture seams* or *island margins*. Such seams pose texture discontinuities. That is, neighbouring patches in object space are not adjacent in UV space. For instance, a three-dimensional triangle mesh can be projected with uniform scale and without distortion of triangle aspect ratios if each triangle is projected individually onto the plane. However, if each patch forms its own projection group as in this example, each edge on the mesh poses a texture discontinuity. In other words, less projection groups minimise the amount of texture discontinuities or seams, while more projection groups provide a more uniform texel-to-world scale at the cost of more texture seams. Note that the non-uniformity can also be added for

artistic and technical reasons, e.g., to meet the requirement for additional detail in the facial area of a humanoid mesh when compared to the bottom of the humanoid feet, which will hardly ever be visible. In this case, the facial area texels will be given a low texel-to-world scale, such that it takes up more space in the UV map.

Günther et al. use textures as their primary data structure for sediment concentrations and keep track of texel-to-world scales in a pre-calculated texture to compensate for them. To compensate for texture discontinuities, the authors propose to limit the splat size for texture painting to a single pixel, otherwise artefacts occur [12].

Frerichs et al. propose a different approach of handling texture discontinuities in the context of weathering simulation as an alternative to the Günther et al. approach of limiting the splat size. The approach is based on the projection of a square centered around the intersection point onto the affected mesh. At texture seams, the square will be split. Each square fragment will be triangulated and individually projected into texture space for painting. The method also proposes to augment the square with an alpha map, such that different shapes can be drawn into texture space [10].

Both classical γ-ton tracing in [4], as well as the variation presented in this thesis store local substance concentrations in a point-based data structure independent of any textures. This representation does not suffer from texture discontinuities, since the points are positioned in three-dimensional space. While surfels do not have a size, similar artefacts as for non-uniform texel scales can arise if regions of the mesh are under- or oversampled. In these cases, surfels at a local surfel density maximum would change their concentration at a lower rate, since interaction would be split up among more surfels. Surfel density maxima during texture synthesis lead to some surfels having a greater influence on the final output colour than others.

Regarding the transport of substances, the presented approach avoids such problems by generating a maximal Poisson Disk set of surfels. This guarantees a minimum distance requirement between surfels along with the guarantee that no new surfel could be added without violating this constraint. Hence, the distances between neighbouring surfels are mostly uniform, in turn making the surfels influence a mostly uniform radius around them. When in the texture synthesis step, surfels are converted to the substance concentration texture. This is done by selecting the surfels within a radius r around the world-space position of the center of the texel. By modulating r with the associated texel-to-world scale, non-uniformity in scale can be compensated for. To avoid artefacts at texture discontinuities, each UV island will add a user-defined bleed by extrapolating triangles at seams.

3.9 Optimizations

A naïve γ-ton tracing implementation is computationally intensive and requires large amounts of memory. Some aspects in *aitios* have been optimised to better utilise available resources and to organize data into more efficient data structures.

3.9.1 Parallelism

Significant time is spent on the tracing of γ-tons through scenes and calculating their interactions with surfels. The core tracing process only requires read access to the γ-ton

currently being traced and the scene geometry for intersection tests. This observation hints at potential for optimisation, since each γ-ton could be traced independently. Substance transport, however, additionally requires write access to both the γ-ton and the surfels. Motion deterioration requires read access to surfels and write access for the γ-ton.

The implementation originally carried out substance transport directly after motion probability deterioration during tracing. Parallel processing of two γ-tons could hence potentially write to the same surfel at the same time, if the trajectories are similar enough that the set of interacting surfels overlaps. The implementation resorted to a sequential implementation to avoid a race condition during updates to surfels. γ-tons are emitted one after the other, traced through the scene, interacting at each stop until finally settling. At this point, the next γ-ton can be emitted. This scheme of tracing is akin to recursive ray tracing or a depth-first search in a tree. The results are predictable across multiple runs of the software, given the same seed is set for the pseudo-random number generator. The original implementation had the advantage of being relatively simple, perfectly predictable and having advantageous space complexity and data locality. However, only a fraction of the available resources were used, compared to a more aggressively parallel implementation.

A mutex lock on each surfel or the use of atomic operations on surfel substance access potentially solves the race condition on surfel updates. However, the synchronisation cost would be considerable. Furthermore, the interaction order would be undefined, potentially leading to slightly different results on multiple runs with the same seeding value.

The problem has been split into a parallel tracing and deterioration part and a sequential substance transport part. For the partly parallel implementation to reach an equal level of predictability as the sequential tracing scheme, a breadth-first tracing scheme can be deployed at the cost of additional memory: All live γ-tons are placed in an ordered data structure. After parallel tracing to the next interaction points is finished for all γ-tons currently in the simulation, substance transport can then have the γ-tons interact at their new positions sequentially (in the order of emission), in a next step removing settled γ-tons from the data structure. To further ensure predictable sampling of outgoing directions, the thread-local PRNG needs predictable seeding in the tracing worker threads. A potential solution would be to derive the seed from the index of the ton emission and the depth of tracing before sampling. The results of this variation would still differ slightly from results of sequential tracing due to its breadth-first nature yielding a different, but equally consistent, interaction order.

3.9.2 Spatial Data Structures

The implemented algorithm relies on spatial lookups for commonly occurring operations, including:

- intersections of γ-tons with geometry on straight paths, in parabolic motion, in flow,
- the selection of surfels within a radius r for interaction, relative to an intersection point of incident γ-tons,
- n-queries for nearest surfels to a given texel offset during gathering of substance

concentrations into textures.

These requirements in combination with the large search space for each operation warrant the use of spatially organised data structures.

Octree

In the first iterations of implementation, the triangle intersection part of the tracing process was the largest bottleneck of the algorithm. The naïve triangle intersection algorithm would linearly search for the nearest intersected triangle. The approach becomes infeasible for scenes with millions of triangles. Spatial organisation for the triangles was needed.

An Octree was implemented as the data structure for accelerated triangle intersection. The intersection algorithm pruned large parts of the tree if a ray for straight trajectories did not intersect the corresponding subtree. For the piecewise linear approximation of parabolic trajectories, the trajectory would be split into line segments. Each line segment was used for a line segment cast (raycast with limited range) in the Octree. Without the optimisation of intersection, the piecewise linear approximation of parabolic γ-ton trajectories would have been extremely inefficient, especially for large and detailed scenes.

kD-tree

To facilitate substance transport and motion deterioration, it is necessary to look up nearby surfels for a γ-ton that has intersected geometry at a specific point. A natural candidate for this nearest-neighbour lookup was the kD-tree, which was also used for surfel lookups in [4]. By organizing the surfels in a kD-tree, both the substance transport part of the tracing process and the texture gathering could be sped up by multiple orders of magnitude.

Chapter 4

Evaluation

Practical applicability of the presented variation of the γ-ton tracing method to texture synthesis problems in weathering simulation will be evaluated. The property of practical applicability is evaluated according to the methods defined in section 4.1 and, for the purposes of this evaluation, consists of two main aspects. *Physical validity*, the first aspect, is further described in section 4.1.1 and denotes the extent to which the visible results of a given natural effect can be mimicked by the simulation. Practical applicability cannot be asserted if the method is incapable of producing the results of an effect pattern or consistently produces less convincing results than typical textures produced for the same problem with primarily manual labour. The second aspect is the *runtime performance* of the approach in terms of time and space complexity in practise. Section 4.1.2 provides a more detailed description of the methods and rationale of the evaluation of performance. The presented approach must be scalable to complex simulation scenarios with scenes of non-trivial size. Otherwise, use in production workflows becomes infeasible and practical applicability of the method cannot be justified. Practical applicability evaluation is performed for two related chemical weathering phenomena. Both are associated with the corrosion of metals under the influence of wind and weather, which has been widely studied in literature. Section 4.2 provides a short overview of corrosion phenomena and secondary effects associated with it, such as differential flow. Based on these definitions, section 4.3 presents a simulation performed with the presented γ-ton tracing implementation that aims to reproduce the described natural phenomena. Finally, in section 4.4, simulation results are presented and evaluated.

4.1 Methodology

The main device for evaluation will be the reproduction of observed natural phenomena found in reference photographs. For this purpose, scenes in photographs will be digitized in a simplified form. Artefacts found in the original photographs or aspects thereof should be reproducible with γ-ton tracing on the digitized versions. The extent to which the artefacts can be reproduced, as well as the runtime performance of the designed simulations, will be key factors in the evaluation of practical applicability.

The evaluation aspect of *physical validity* of the simulation results, given each class

Score	Requirements
0	Not applicable. The technique either was not designed for the weathering pattern or otherwise consistently fails to reproduce it in a way that can similarly be found in a reference photograph.
1	The weathering pattern can be reproduced in a way that is intended and noticable, however clearly less convincing with respect to the reference photograph than a typical hand-painted texture.
2	The weathering pattern can be reproduced such that the result is not easily distinguishable from a hand-painted texture.
3	The weathering pattern can be reproduced with results that are highly compatible with reference photographs, with similar results being impractical to achieve with manual texture production techniques.

Table 4.1: Values of the score for physical validity of a simulation result, along with their definition, given a reference photograph and a hand-painted texture of the desired weathering pattern.

of weathering effect, will be grounded on a simple model of known chemical, physical and biological factors in play, Characteristic weathering patterns found in nature will be identified and described on the basis of these models. As the applicability of γ-ton tracing to a class of weathering effects is evaluated, a simulation will be designed that produces described weathering patterns. It should be noted that the primary goal is to reach a level of validity that suffices for a *convincing* result, as opposed to a completely naturalistic or realistic result. For each weathering pattern, a value will be assigned to the *physical validity* of the simulation result, according to the self-set definitions laid out in table 4.1. Section 4.1.1 further describes evaluation of this aspect.

The second part of applicability is based on *runtime performance*. The simulations for physical validity, with adequate quality settings, should run within a time that makes deployment of the technique feasible within typical game production workflows. The evaluation of this aspect is further described in section 4.1.2.

4.1.1 Physical Validity

Evaluation of physical validity is done individually for each characteristic weathering pattern within a class of weathering effects. To evaluate the extent to which a type of artefact can be reproduced using a given simulation technique, a comparison of:

- reference photograph for the specific pattern,
- hand-painted texture for comparison,
- rendered simulation results,

will be performed. Given the results of this comparison, a score value can be assigned that indicates the extent to which the simulated result is convincing for a given weathering pattern found in nature. The score value is a natural number in the interval $[0, 3]$, that is, without intermediary values between whole numbers. The value represents a partial result for convincingness using the definitions laid out in table 4.1. Irrespective

Symbol	Unit	Definition
t_γ	s	Average time in a single iteration used for tracing of γ-tons and calculating their interactions with surfels.
t_T	s	Average iteration time used for the calculation of weathered textures and persisting the new textures to disk.
t_P	s	Time spent on preparation tasks before the first iteration, notably including scene deserialisation, surfel kD-tree and Octree generation and surfel association table calculation.

Table 4.2: Definitions for measured performance metrics.

of the definitions for the score values, personal biases of the evaluator cannot be completely factored out of this assignment, since convincingness is inherently dependent on the perception of the observer. The relatively coarse granularity of four natural numbers as possible values, each representing a distinguishable set of requirements, is an attempt to minimise the effects of these biases.

The unweighted arithmetic mean of all applicable scores will define the overall *physical validity* of results that can be achieved for the defined set of weathering patterns using the presented technique. For scores $s \in S$, where $s \in \mathbb{N}$, the combined physical validity $\overline{s} \in \mathbb{R}$ is

$$\overline{s} = \frac{1}{|S|} \cdot \sum_{s \in S} s. \tag{4.1}$$

4.1.2 Runtime Performance

Performance plays a vital role in the practical applicability of the presented technique. It will be evaluated as the amount of resources, primarily time and memory, spent on loading, tracing, transport of substances, texture synthesis and texture persistence during a single iteration. Specifically, the simulation should keep running times low enough such that experimentation and parameter tweaking by the artist is not severely inhibited with small to medium-sized scenes. The bulk of experimentation is expected to be performed on such small or medium test scenes, to be later applied to larger scenes, once the details of the simulation are worked out. Extremely large scenes may have longer running times, where experimentation is hardly possible. Memory requirements however, should still be low enough that such non-trivial scenes can still be simulated, albeit with higher iteration times. These aims, along with adequate physical validity, must be met to attain practical applicability.

The evaluation of runtime performance will thus be based on three key metrics, laid out in table 4.2. The metrics will be measured on modified versions of the simulations for physical validity with varying counts of entities and texture sizes. Remaining running times that have not been captured, such as simulation specification parsing or logging, are sufficiently masked by the captured running times to be considered negligible for the purposes of this evaluation.

The implementation leaves it up to configuration how many simulation iterations

are calculated and how often weathered textures are exported. Per default, the texture synthesis pipeline is executed before the first iteration and after the last iteration. The initial synthesis process is considered time zero of the simulation and is useful as a reference for the weathered appearance calculated with the first simulated substance distribution. This initial output can differ from the input scene if the simulation is configured in such a way that objects initially carry substances in their surfels, even before the first tracing iteration.

Using the default configuration, n iterations of simulation take approximately

$$t_{\min}(n) = t_{\mathrm{P}} + n \cdot t_{\gamma} + 2 \cdot t_{\mathrm{T}}, \tag{4.2}$$

to complete. Providing custom settings, texture synthesis can be performed after every $m \geq 1$ iterations, yielding intermediate states of weathering. In this case, the simulation takes

$$t(n, m) = t_{\mathrm{P}} + n \cdot t_{\gamma} + \left(1 + \left\lfloor \frac{n}{m} \right\rfloor\right) \cdot t_{\mathrm{T}}, \tag{4.3}$$

to run to conclusion, though intermediate results are immediately usable. The first form $t_{\min}(n)$ with only two instances of texture synthesis is especially relevant for the practical applicability of the technique, since this mode is most likely to be used by artists when tweaking parameters of a simulation. If for a simulation, $t_{\min}(1) < 1800$ does not hold true, it is considered practically inapplicable in terms of performance. Tweaking of the simulation configuration has become impractical at this point. Even smaller waiting times on the order of fifteen minutes can be considered an inconvenience, though the technique can at least be considered usable. The complexity of scenes where the mentioned property still holds true establishes the *scalability* of the approach to larger scenes. Note that the performance of the actual rendering process, based on textures that have been generated with *aitios*, is not under evaluation. This part of the functionality is covered by *Blender* in this work and is unrelated to the core algorithm. Where possible, metrics will be put into context by comparing them to related techniques.

Measurement

The partial running times described in section 4.2 are captured with benchmarking functionality built into *aitios*. This functionality writes the measured performance metrics to CSV files in the background with little overhead. Internally, the monotonic system clock is used for measurement, rather than CPU time. When timings for multiple iterations are available, the values are averaged. As described in equations 4.2 and 4.3, the time required for a full simulation depends on the amount of iterations and the effect interval, but can trivially be calculated from t_{P}, t_{γ} and t_{T}.

All metrics have automatically been captured in a single session on an evaluation machine with hardware specifications in table 4.3. Apart from performance measurement and system background processes, the machine was kept idle during evaluation.

Relative Performance

While these absolute time values provide useful information by themselves, a comparison to the running times of related approaches is also of interest. Accurately comparing the resulting performance characteristics with other algorithms, however, poses

CPU	AMD Ryzen 5 1600, 6 × 3.20GHz, 64-bit
RAM	Crucial CT2K8G4DFD8213, 8GB × 2, 2133 MT/s
Swap	16GB on SSD (Crucial MX300 CT525MX300SSD1 525 GB, 530 MB/s sequential reading)

Table 4.3: The most relevant hardware specifications of the machine used for runtime performance evaluation. See `hardwarespecs`-directory on the accompanying CD for detailed specifications.

practical problems. In the common case where no reference implementation is available for the other algorithm, the comparison can only produce fuzzy results without re-implementation of the algorithm to compare to. In this case, the only performance metrics available, if any, might be the ones published along with the paper that originally proposed the technique. In the concrete case of comparing metrics captured on two different sets of hardware, a high degree of uncertainty is involved. The farther apart the two points in time are at which metrics were captured, the higher the uncertainty of relative performance is, due to advancements in computer hardware made in the meantime. Hence, comparing the *aitios* implementation with the performance of other techniques, only a rough estimate can be given. For this estimate, only similar effects calculated for scenes that are also similar in complexity, will be calculated to cautiously answer the question of relative performance.

Scalability

While the question of relative performance can only cautiously be answered, the evaluation will also aim to provide insights into the *scalability* of the approach, which cannot easily be boiled down to a single number. Past publications have, for the most part, provided performance numbers on some select, small to medium-sized scenes with some tens of thousands of triangles. While these numbers give a first impression of the performance of the approach, it gives little information on the limits of the approach. To learn about these limits, the evaluation will make use of stress tests. Simulation scenes will be systematically scaled up in terms of entity count, triangle count and other simulation metrics. To obtain more complex variations of geometry of simulation scenes and emission shapes, the command-line tool *litter*[1] has been developed. The used functionality in *litter* derives complex variations by duplicating all elements in OBJ files and arranging them in larger grids of configurable cell size. After calculating consistent, complex variations for scene geometry and γ-ton emission meshes, the emission count of γ-ton sources needs to be manually scaled up to match the grid size. Otherwise, the γ-ton hits per square unit of area on scene geometry would decrease with the increased emission shape surface area due to duplication, resulting in less intense weathering effects. Other simulation properties naturally grow with larger grid sizes, such as the surfel count, which is configured with the minimum surfel distance r, or the generated texture count, which is carried out on a per-entity basis. The resulting performance metrics for

[1] The tool *litter* for geometry stress testing is publicly developed by the author on *GitHub*.

the stress test variations will be arranged into tables and graphs for insights into the scalability of *aitios*.

4.2 Corrosion

The effects under evaluation primarily belong in the domain of *corrosion*. Widely found in both urban and rural environments, corrosion poses a readily observed weathering phenomenon. The visible results of this class of weathering effect come in different flavours, depending on the composition of the alloy or metal and the environment. With some exceptions such as gold and platinum, most unprotected metals that are exposed to environment conditions are subject to corrosion. Corrosion initially attacks metals on a chemical level. Physical factors, such as changes to the material integrity in response to corrosion, also play a major role in shaping the visual results of the effect. Biological factors are not considered.

4.2.1 Patterns

Corrosion occurs in many different forms. All forms share the common characteristic of converting the metallic base material to a more chemically stable form. Typical corrosion products are oxides and salts of the corroding metal, such as sulphites and nitrates [7]. In the following, some types that can be readily observed in nature will receive closer attention.

Rusting Corrosion

The formation of some iron compounds due to corrosion is often referred to as *rusting*. It is a subset of corrosion effects where iron interacts with its environment through redox reactions, forming at least one of sixteen known chemical compounds associated with rusting corrosion. These compounds are known as iron oxides and iron oxyhydroxides. Typical rust has a characteristic earthy, dark-red colour. Rusting corrosion products are flaky, porous and easily soluble in water. Figure 4.1 shows typical examples of rusting in nature. Artists impressions of rusting objects with manually made textures can be seen in figure 4.2.

Patination

Patina denotes a highly layered material film on the surface of some metallic objects, especially copper and bronze [9]. Typical brass patination can be seen in figure 4.3 and figure 4.4 shows a 3D scan of a patinated equestrian statue along with a simplified and manually textured version. The appearance of patina is highly dependent on the unique combination of base metal and environmental factors. It develops due to adding and removal of materials on the surface. This process is mostly driven by chemical processes of corrosion. These processes are enabled by exposure of the metal to water, air and environment-specific minerals.

Typical patina is quite thin and leads to a behaviour often referred to as *passivation*, where underlying layers are protected from corroding by upper layers. Aluminium is another prime example of this. It develops a thin layer of aluminium oxide when

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

Figure 4.1: Typical appearances of rusting corrosion in nature. (a) shows typical primary flow effects on a rusting gate, (b) shows moderate surface corrosion on iron [34], (c) exhibits intense corrosion with destructive effects, especially cracking and flaking [35]. Finally (d) shows a pure sample of an iron oxide [30], exhibiting characteristic colouring that can be observed in most instances of rusting corrosion as its primary reaction product.

(a) (b)

Figure 4.2: Artist renderings of rusting objects with hand-painted textures. (a) depicts a fantasy helicopter [33] and exhibits typical surface rust colours and patterns, (b) shows a rusting sink [28], posing an artist impression of differential flow.

first exposed to air, protecting the bare aluminium underneath from corrosion. Such behaviour leaves objects quite chemically stable, but still leaves it susceptible to other attacks such as contact with strong acids. Apart from patination on copper and other materials, rusting on some iron-containing alloys such as *weathering steel* exhibit similar behaviour.

Section 2.2.1 provides an overview of the patina technique by Dorsey and other researchers. According to them, the development of patina is influenced by many environmental and geometric factors, including, among others:

- wetness, increasing patination rate,
- temperature and exposure to sunlight, which causes water to evaporate especially during the day and in areas accessible to sunlight, locally reducing the rate of corrosion,
- stagnant water increases the rate of patination, decreasing patination on inclined and horizontal surfaces and increasing patination in inaccessible areas,

(a) **(b)**

Figure 4.3: Patina on a brass statue. Variation (a) shows the statue after many years of intense patination with strong flow artefacts on the stone pedestal [32]. In variation (b), the statue has been cleaned of patina [31].

(a) **(b)**

Figure 4.4: Renderings of two patinated versions of the equestrian statue of Napoleon. Variation (a) shows the geometry and albedo texture of the original 3D-scan by *Ima Solutions* [37] rendered with constant metallicity and roughness over the model. Variation (b) is a simplified and re-textured variation by Loïc Norgeot with manually designed albedo, metallicity, roughness and normal maps [36].

- polishing and pitting de- respectively increase patination rate [6].

Destructive Corrosion

Rather than building a film on top of the base material, *destructive corrosion* leads to loss of material in the form of holes and deformations [9] and may eventually lead to

complete disintegration of the base material. This is the case with many flaky corrosion products, such as in acidic corrosion and some instances of rusting corrosion. In figure 4.1 (c), a rusting chain shows noticeable crackling, leading to material flaking off. This is quite possibly due to expansion of the rusting metal and could be seen as an instance of destructive corrosion. In such cases, the porous resulting weathering products seem not to provide protection for the underlying material. Given enough time under exposure to the weathering source, the process will likely only cease once no more base material is left to fuel the reaction.

Differential Flow

Many corrosion processes involve fluids as part of a chemical reaction. Rusting corrosion reactions, for instance, often involve water molecules to form iron oxyhydrates. In addition to their chemical role, fluids also play a role in the physical distribution of corrosion products throughout their environment. Soluble corrosion products get dissolved by incident fluids to be re-deposited in different places in a process often referred to as *differential flow*. Dorsey et al. describe differential flow in their work on flow and changes in appearance [7], while classical γ-ton tracing describes the effect as *global transport effects* [4]. The effect heavily influences the appearance of the corroding metal as well as of neighbouring objects. Observing the patinated statue in figure 4.3 (a), brass corrosion products get dissolved in rain, leaving long streaks on the stone pedestal that the statue is mounted on as well as on the statue itself. An example of differential flow on a hand-painted texture can be seen in figure 4.2 (b).

4.3 Simulation Design

To evaluate the practical applicability of γ-ton tracing to corrosion effects, the described natural effects will be reproduced with weathering simulation. For this purpose, the corrosion of statues of iron and copper under the influence of air and rain will be simulated with *aitios*. The target model is an equestrian statue of Napoleon approximately 3.5 metres in height and visually similar to the statue in figure 4.3, but made of copper instead of brass. A scanned, textured model made by the Paris-based company *IMA Solutions* using an *Artec Eva* scanning device is available [37] and a rendering of it depicted in figure 4.4 (a). Loïc Norgeot provides a simplified version with adjusted texture coordinates and hand-painted textures [36], with a rendering in figure 4.4 (b) that is used as the second reference for comparison. A second evaluation for rusting corrosion is performed on the same model, with reference imagery available in figure 4.1 and renderings with hand-painted textures in 4.2 for comparison.

To create a simulated weathered appearance, the simplified model by Norgeot was used as simulation geometry. The horse statue has been assigned a base material for the metal under evaluation with mostly uniform, polished, post-manufacturing appearance. The base material maps have mostly constant colour with only subtle manufacturing-related blemishes such as mild pitting effects. Texture maps for albedo, metallicity, roughness and displacement are part of the base material. Analogously, the pedestal has been given a sandstone-like base appearance. Using the consistent transport mode, the influence of rain and air on the entities over time is simulated with two γ-ton sources

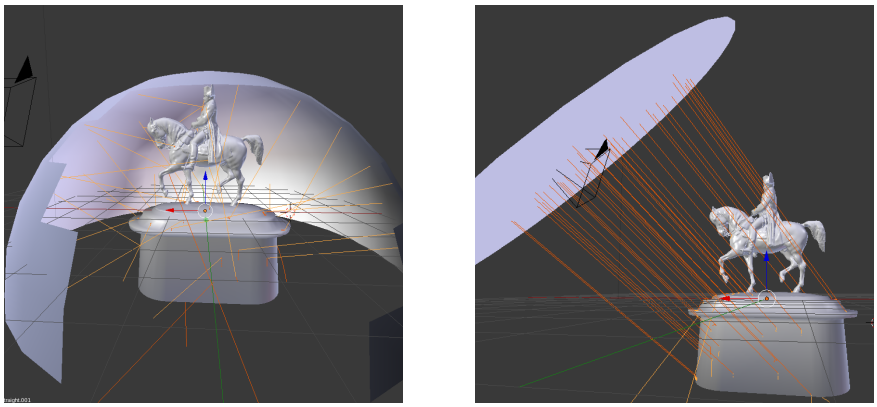


Figure 4.5: Trajectories of a small subset of the emitted γ-tons from the sky (a) and from the prevailing wind direction (b).

that simulate the distribution of humidity and existing corrosion products throughout the scene. A statue-specific transfer ageing rule models the gradual corrosion in the presence of water, while a global deterioration ageing rule models the evaporation of water. These ageing rules kick in before each texture generation. Performance metrics are accumulated over the iterations of the simulation process. Subsequent runs of the simulation with modified workloads are used to capture additional performance data. Analysis of these metrics determines *runtime performance*. Note that the final rendering process with *Blender Cycles* using the materials weathered with *aitios* is considered unrelated to γ-ton tracing itself and will not be evaluated for performance.

4.3.1 Scene

To adapt the simplified version of the 3D scan for the simulation, it has been transformed in the following ways:

1. scaled the statue to approximately match the real size of 3.5 metres, assuming a conversion factor of 1 between metres and world-space unit lengths,
2. added simplified version of the stone pedestal in figure 4.3 under statue to allow for differential flow to span multiple entities.

For simulation, two variations of this preprocessed scene will be used that differ in the material of the statue. Iron and brass will be evaluated separately by applying that material.

4.3.2 Emission

Rain exposure will be simulated by the emission of γ-tons from two primary sources depicted in figure 4.5. In total, 150000 humidity-carrying gammatons are emitted for the main variation. Two thirds are emitted from a hemispherical emission shape that represents the sky. Initial trajectories are diffusely sampled over the upper hemisphere of the uniformly selected emission point on the emission mesh, as to allow a wide range of possible trajectories, rather than having all γ-tons fly in direction of the scene origin.

The last third of γ-tons is emitted non-diffusely from a spotlight-like source that models the prevailing wind direction.

The γ-ton properties are modelled after rain drops. Since water drops carry weight and their initial velocity is relatively high, it is plausible for them to enter a parabolic trajectory after their first encounter with the surface and splatter onto nearby objects. In such a case, part of the water would remain at the initial impact location. This behaviour is modelled by having part of the γ-tons bounce after the first encounter. Further, water has a natural tendency to adhere to a surface and flow over it. The water drops have a relatively low weight and low viscosity. This is achieved by assigning a high value to the flow motion probability of the tons and assigning the γ-reflectance of surfaces for flow to a low value. Considering these desired behaviours, the initial motion probabilities for rain γ-tons are defined as

$$p_s = 0,\ p_p = 0.05,\ p_f = 0.9. \tag{4.4}$$

4.3.3 Material Properties

The metal is assumed to have a low amount of holes and pores, giving it low absorption and absorptivity. This property allows water drops to flow relatively long distances over the metal surface of the model before it has been fully absorbed. Consequently, the γ-reflectance of both iron and copper will be defined as

$$\Delta_s = 1,\ \Delta_p = 0.05,\ \Delta_f = 0.01. \tag{4.5}$$

The stone pedestal is thought to be made of porous limestone. The porosity gives it a relatively high absorption and absorptivity when compared to metals. This is modelled with a relatively high value for flow motion probability deterioration, i.e.,

$$\Delta_s = 1,\ \Delta_p = 1,\ \Delta_f = 0.1. \tag{4.6}$$

4.3.4 Interaction

The absorption and deposition constants of γ-tons have been configured to occur within a radius of 1 cm with absorption constants of

$$a_{\text{water}} = 0.07,\ a_{\text{patina}} = 0.2,\ a_{\text{rust}} = 0.2. \tag{4.7}$$

The deposition rates of surfels have similar values between 0.05 and 0.1 for the metals, with some tweaking performed to better match the natural weathered appearance. The deposition rate of the stone is set at 0.6 for both corrosion products.

4.3.5 Ageing Rules

Both simulations will have water evaporate from surfels at a constant rate of -50% per iteration from iron, brass and stone. Air exposure will be assumed to be uniform on the model and is implemented as ageing rules specific to the statue that process water into corrosion products. For iron, all surfels will process 15% of absorbed water into new rust at the end of each iteration. The rule does not apply to stone, however rust can flow from the statue onto the pedestal below for differential flow. Rust is the only substance in the rust simulation with a direct effect on texture synthesis on both the statue and the pedestal. In the simulation of patina, the transfer rule processes only 8%.

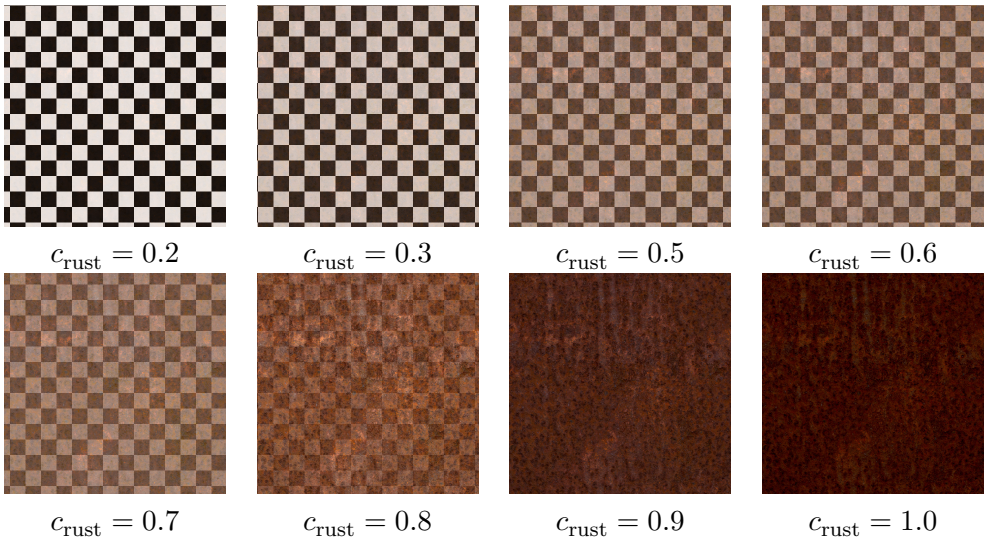


Table 4.4: Eight of nine texture samples used for the synthesis of rusted textures. The numbers below the textures indicate the concentration cenith, at which the texture sample has maximum influence over the synthesised texture. The ninth texture has its cenith at 0 and is fully transparent.

4.3.6 Effect Setup

The general scheme for texture synthesis in *aitios* is described in detail in section 3.8.4. Both the statue and the pedestal have a base material defined. The base material is mostly devoid of blemishes and contains only deformations that can be directly attributed to the manufacturing process.

The corrosion products stored in concentration maps will guide layering effects on top of the base material. All metals receive a synthesised albedo, normal and roughness texture depending on the respective corrosion products. The iron variation will additionally use displacement mapping to account for deformations seen in additive and destructive corrosion. A similar layering effect will be used for the sandstone pedestal to account for differential flow. The texture sequence used for the finishing layer of the albedo texture in the iron statue in the rusting corrosion simulation can be seen in table 4.4, along with the concentrations at which each texture sample has maximum influence. This value is denoted the *cenith* of the sample. While the rust samples encode high-frequency surface details, most other samples are of almost constant colour. The samples for patina, having a more uniform appearance in nature as rust, are tiny regions of the horse statue photograph with modified opacity.

4.4 Results

This section presents renderings for the evaluation of physical validity as well as performance metrics for the evaluation of runtime performance. Resulting renderings of the rust evaluation are depicted in increments of ten iterations in table 4.5. Renderings of

the patina evaluation are depicted in table 4.6. An interpretation of the resulting renderings in terms of physical validity, using terms defined in section 4.2, is organized in table 4.7. Both sequences seem to provide acceptable visual quality for the problem at hand and will be evaluated in detail in section 4.4.1 for physical validity. Performance data of the technique for more complex variations of these two simulations is available in tabular form in table 4.8 for patina and table 4.9 for rust, with more detailed evaluation in section 4.4.2.

4.4.1 Physical Validity

Table 4.7 shows the numeric results for physical validity in the context of corrosion effects, as selected by the author. Corrosion effects can quite naturally and intuitively be captured with γ-ton tracing. An effective approach involves at least two substances with one representing humidity and the others representing corrosion products.

The resulting substance textures themselves, with sufficient island bleed, are resilient in the presence of texture discontinuities. The underlying surfel representation is three-dimensional, providing smooth interpolation across seams. Only in the special cases of constant colour textures and textures adapted to the UV maps of affected objects, however, does this property extend to the final generated textures. Uninformed texture samples will generate similar mapping artefacts as when the texture would be directly applied to the object.

Nonetheless, high-frequency texture samples are recommended for weathering effects that require fine-grained detail, such as the rough and porous appearance of rusted iron. The substance distribution can determine the low-frequency weathering state across the model without individual holes, scratches, or comparable high-frequency weathering patterns. This detail will then be added in texture synthesis, when low-frequency substance controls the added high-frequency detail. The results are visually pleasing without the additional processing power required to use significantly more surfels and γ-tons to increase the level of detail.

A similar level of detail in the final textures is not easily reachable using only constant colour textures for texture synthesis or a gradient-based approach as outlined in the implementation chapter in section 3.8.4. Firstly, an excessive amount of surfels might be needed to match the level of detail required, e.g., to simulate individual holes due to corrosion, leading to increased runtime complexity. Secondly, to use the increased amount of surfels effectively for fine-grained patterns, enormous amounts of γ-tons with low interaction radii need to be emitted. While the results of such an approach might be satisfactory visually, performance quickly declines with increasing surface area of the simulation scene.

The physical validity with a combined value of $\overline{s} = 1.75$ suggests that the simulated results are not significantly less convincing than hand-painted textures. Neither a significant advantage nor a significant disadvantage in terms of convincingness of the simulated versions can be asserted on the basis of the resulting scores. Consistent weathering among objects with similar exposure, however, is less of a challenge than with hand-painting. Simulating also provides opportunities to re-use parts of a simulation configuration in different contexts and to re-arrange objects later on in the production process, re-calculating weathering signs accordingly.

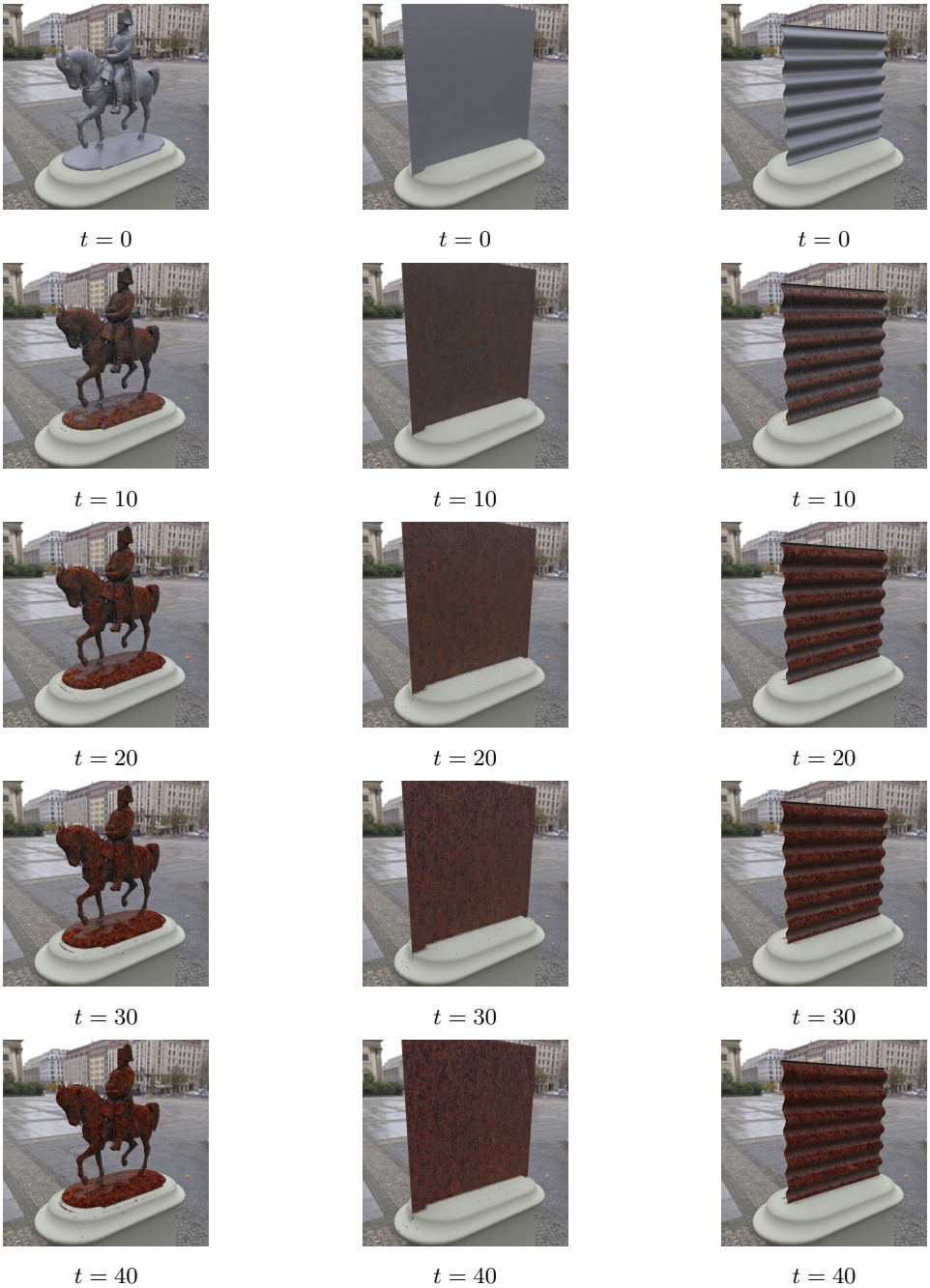


Table 4.5: Renderings for the evaluation of rusting corrosion. (a) is the evaluation sequence, the equestrian statue of Napoleon in different stages of weathering. $t = 0$ is the base material without weathering simulation. $t > 0$ is the state after t iterations of weathering simulation. (b) and (c) additionally show the same simulation with one extremely flat geometry and one curved geometry.

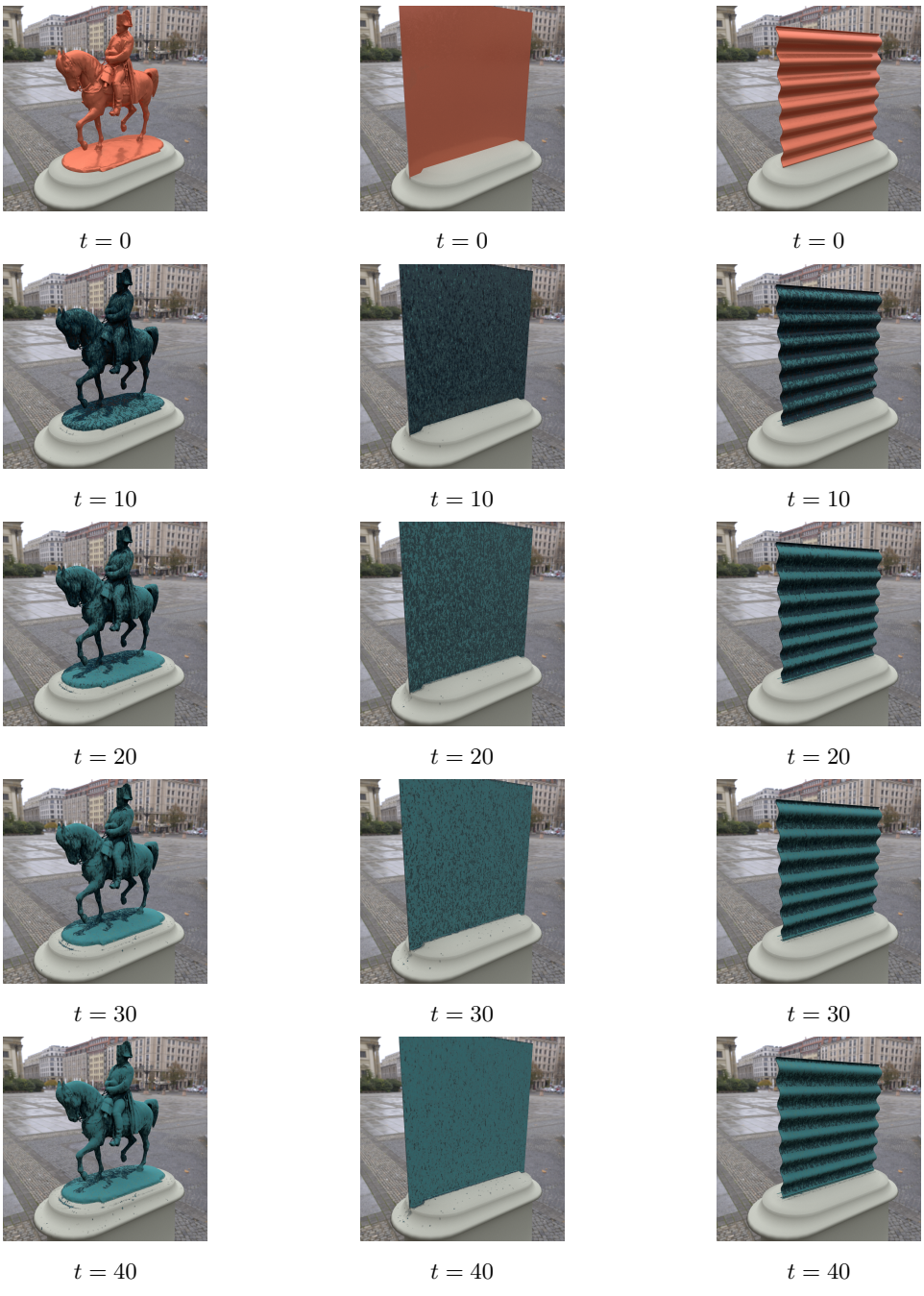


Table 4.6: Renderings for the evaluation of patina (a). Equestrian statue of Napoleon in different stages of weathering. $t = 0$ is the base material without weathering simulation. $t > 0$ is the state after t iterations of weathering simulation. (b) and (c) show the same simulation operated on different geometries.

	Score	Rationale
Destructive Corrosion	2	Reasonably convincing implementation with displacement mapping for smaller-scale deformations. However, objects will not disintegrate or break apart. Gradual development of cracks could be modelled with multiple blend stops, where new cracks abruptly appear instead of blending in. The μ-ton system by Kider can simulate parts falling off. Weathering in texture space by Bellini and other researchers seems to more convincingly simulate gradual crack development. Hand-painting does not seem to have a clear edge over simulation.
Rust	2	Implementable as humidity-based ageing rule. Looks convincing if texture samples encode high-frequency rust details. Not easily distinguishable from an artist rendering of a rusted object.
Patination	2	Easily implementable, can simulate effects of environment and material composition as well as geometric factors such as curvature and exposure. While somewhat convincing, it is challenging to model the colour progressions of patina in a way that each consecutive frame looks convincing. The result compares favourably to the brass statue reference photograph, exhibiting similar effects on appearance for both accessible and inaccessible areas, albeit with different colouring. Hand-painted patina does not seem to have a clear edge over the simulated variant, but seems also convincing.
Differential Flow	1	Material can be globally transported among objects to simulate differential flow. Prime use case for γ-ton tracing. However, cannot capture some advanced properties of water flow such as self-repulsion or saturation that can be present in a hand-painted texture (See figure 4.2), but also in simulated textures with the flow approach by Dorsey and other researchers. In effect, the flow artefacts shown in the simulation are recognizable as synthetic and inexact, making the before-mentioned techniques clearly superior.
Physical Validity	1.75	Arithmetic mean.

Table 4.7: Results of the evaluation of corrosion.

Rust: Produces convincing results for rusting corrosion of some kinds of iron. The shading of types of iron that exhibit some sort of passivation (cast iron, weathering steel, etc.) can be simulated by applying layers to base materials that account for albedo, metallicity and roughness of the emerging rust. Depending on the intensity of additive or destructive effects of the rusting process, deformations can be accounted for using normal maps (small-scale deformations), or using displacement maps (medium-scale

deformations) on most target models. However, since γ-ton tracing includes no dynamics engine, objects can not automatically crack or fall apart in a physically accurate way in response to rusting processes over hundreds of years. Using the example, no matter how intensely corroded the legs of the horse of Napoleon are, the statue will never fall over.

Patination: When comparing the simulation result with the hand-painted version of the patinated Napoleon statue, none of the 56 simulated frames bear a striking resemblance. The hand-painted version, while certainly convincing, also differs quite a lot from the 3D-scanned version. Apart from the relatively shiny appearance, the artist chose to add dark, flow-like streaks in some places, differing from the streakier parts of the scanned variation. Closely observing the 3D scanned version, geometric factors are more obvious than in the hand-painted and in the simulated version. Inclined surfaces with low global accessibility, such as the horse legs, seem to develop noticeable dark streaks that could be attributed to water dissolving some of the patina and redepositing it. While such streaks are also present in the simulated version, the relationship to geometry is much more obvious in the 3D scanned version. Note that the scanned model, consisting of millions of triangles, has vastly more geometric detail.

Validity of Related Techniques

Apart from γ-ton tracing variations, there are quite specialised techniques tailored to the exact problem at hand. One such approach for Patina by Dorsey et al. has been described in section 2.2.1 and stands out due to the degree of physical and chemical insight applied to the problem. The difference in available substances in urban, marine and rural environments described by Dorsey et al. in their work on metallic patinas [6] can be captured with additional substances in γ-ton tracing. However, there is no obvious equivalent for their surface operators, such as the polishing operator. While global accessibility or water flow do not play a role in the patination approach by Dorsey and Hanrahan, the results still look quite convincing. Being specialized to copper patina, the approach takes multiple types of corrosion products into account that are unique to the corrosion of copper. For instance, the composition of copper salts is attributed to available minerals in the environment and urban patina will look distinct from rural patina.

The flow approach by Dorsey et al. [7] (see section 2.2.2) may provide more accurate flow artefacts, due to their force-based approach more accurately simulating fluid dynamics than the probabilistic motion approach in γ-ton tracing.

Appearance Manifolds, described in detail in section 2.3.1, might be better suited to generate fine-grained details in the presence of texture discontinuities. The approach, however, explicitly does not specify an algorithm to calculate the weathering degree across the model surface, instead proposing to delegate this step to other simulation techniques or user configuration [26]. Substance distributions calculated with γ-ton tracing might be a good fit for this sub-problem in Appearance Manifold weathering.

As outlined in section 2.4.1, Bellini and other researchers present a technique to calculate time-varying sequences of textures directly in the image plane and based on a single input texture as the only relevant user interaction. Artefacts gradually appear

	256×256	512×512	1024×1024	2048×2048	4096×4096
8	$t_\mathrm{T} = 6.29$ $t_\gamma = 32.29$ $t_\mathrm{P} = 12.26$	$t_\mathrm{T} = 7.81$ $t_\gamma = 32.14$ $t_\mathrm{P} = 14.01$	$t_\mathrm{T} = 11.50$ $t_\gamma = 32.20$ $t_\mathrm{P} = 19.98$	$t_\mathrm{T} = 17.35$ $t_\gamma = 36.95$ $t_\mathrm{P} = 39.80$	—
18	$t_\mathrm{T} = 13.96$ $t_\gamma = 211.71$ $t_\mathrm{P} = 28.56$	$t_\mathrm{T} = 17.47$ $t_\gamma = 213.33$ $t_\mathrm{P} = 32.86$	$t_\mathrm{T} = 26.06$ $t_\gamma = 211.47$ $t_\mathrm{P} = 47.95$	—	—
32	$t_\mathrm{T} = 24.95$ $t_\gamma = 218.31$ $t_\mathrm{P} = 52.89$	$t_\mathrm{T} = 31.01$ $t_\gamma = 220.86$ $t_\mathrm{P} = 61.09$	$t_\mathrm{T} = 46.06$ $t_\gamma = 220.44$ $t_\mathrm{P} = 90.24$	—	—
50	$t_\mathrm{T} = 38.70$ $t_\gamma = 960.26$ $t_\mathrm{P} = 85.05$	$t_\mathrm{T} = 48.73$ $t_\gamma = 959.64$ $t_\mathrm{P} = 99.90$	$t_\mathrm{T} = 551.19$ $t_\gamma = 987.04$ $t_\mathrm{P} = 184.61$	—	—
72	$t_\mathrm{T} = 54.78$ $t_\gamma = 993.44$ $t_\mathrm{P} = 127.50$	$t_\mathrm{T} = 69.07$ $t_\gamma = 991.53$ $t_\mathrm{P} = 150.86$	—	—	—

Table 4.8: Performance metrics of the patina corrosion simulation. t_T denotes the time spent on texture synthesis, per iteration. t_γ is the time spent on the actual simulation, that is, γ-ton tracing and substance transport. The columns represent constant texture sizes (column headers) with growing entity counts (row headers) towards the bottom of the table. Where no data is specified, the technique is deemed infeasible, either because $t_\mathrm{min}(1) > 3600$ or because the simulation crashed due to insufficient available memory. If enough memory were supplied, $t_\mathrm{min}(1)$ is still expected to exceed one hour.

one after the other, instead of blending in [2], as would be the case in the presented approach. This difference is especially obvious in the case of cracking patterns. The presented blending-based approach cannot handle such highly stochastic patterns in texture samples in an equally convincing way. One possibility with the presented approach would be to prepare consecutive states of crackling in advance, where additional cracks are added on top of the last cracking state. The blend stops could then be set up such that each cracking state appears twice, the second instance having the same cenith as the next cracking state, such that cracks only abruptly appear instead of blending in. The Bellini et. al. approach, however, could achieve similar effects with a single input texture and at high quality. Results in the *Time-varying Weathering in Texture Space*-approach are quite impressive in the plane, yet the application of these textures to complex, three-dimensional, UV-mapped objects, gracefully handling texture discontinuities and with frame-to-frame consistency, can be considered a non-trivial problem, as it is with the presented γ-ton tracing variation.

	256×256	512×512	1024×1024	2048×2048	4096×4096
8	$t_\mathrm{T} = 12.81$ $t_\gamma = 32.09$ $t_\mathrm{P} = 13.57$	$t_\mathrm{T} = 15.01$ $t_\gamma = 36.04$ $t_\mathrm{P} = 15.40$	$t_\mathrm{T} = 18.54$ $t_\gamma = 32.05$ $t_\mathrm{P} = 21.01$	$t_\mathrm{T} = 32.37$ $t_\gamma = 32.39$ $t_\mathrm{P} = 39.78$	$t_\mathrm{T} = 499.39$ $t_\gamma = 34.72$ $t_\mathrm{P} = 123.70$
18	$t_\mathrm{T} = 28.25$ $t_\gamma = 212.30$ $t_\mathrm{P} = 30.97$	$t_\mathrm{T} = 31.99$ $t_\gamma = 213.00$ $t_\mathrm{P} = 35.22$	$t_\mathrm{T} = 41.89$ $t_\gamma = 214.17$ $t_\mathrm{P} = 49.98$	$t_\mathrm{T} = 72.79$ $t_\gamma = 214.44$ $t_\mathrm{P} = 98.57$	—
32	$t_\mathrm{T} = 50.33$ $t_\gamma = 223.98$ $t_\mathrm{P} = 56.83$	$t_\mathrm{T} = 57.05$ $t_\gamma = 220.42$ $t_\mathrm{P} = 64.86$	$t_\mathrm{T} = 73.91$ $t_\gamma = 220.12$ $t_\mathrm{P} = 94.18$	$t_\mathrm{T} = 554.56$ $t_\gamma = 228.89$ $t_\mathrm{P} = 215.22$	—
50	$t_\mathrm{T} = 78.65$ $t_\gamma = 957.94$ $t_\mathrm{P} = 91.59$	$t_\mathrm{T} = 89.41$ $t_\gamma = 957.56$ $t_\mathrm{P} = 106.07$	$t_\mathrm{T} = 117.98$ $t_\gamma = 960.40$ $t_\mathrm{P} = 156.06$	—	—
72	$t_\mathrm{T} = 113.35$ $t_\gamma = 993.93$ $t_\mathrm{P} = 137.35$	$t_\mathrm{T} = 128.48$ $t_\gamma = 990.82$ $t_\mathrm{P} = 158.18$	$t_\mathrm{T} = 311.05$ $t_\gamma = 1003.58$ $t_\mathrm{P} = 244.37$	—	—

Table 4.9: Timings for the rusting corrosion simulation.

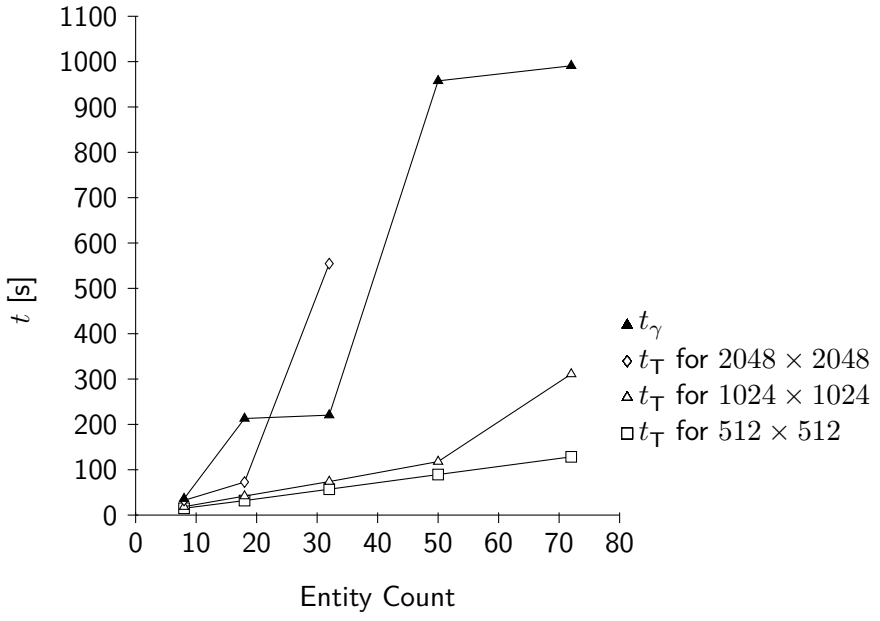


Figure 4.6: Performance metrics of the rusting corrosion simulation. t_T denotes the part of the iteration spent on texture synthesis. t_γ is the part of the iteration time used for γ-ton tracing and substance transport. The sharp spike in t_γ at 50 entities is assumed to occur due to usage of swap space on disk becoming necessary for the application to run.

4.4.2 Runtime Performance

All performance metrics have been captured on a machine with hardware specifications described in table 4.3. To learn more about the performance-defined practical limits of the algorithm, two techniques to derive variations of the simulations with increased workload have been defined. Variations of textures size and entity count are used in combination to create a matrix of running times for each of patina and rust, listing the performance of combinations of entity count and texture size (see tables 4.8 and 4.9). Also see the graph in figure 4.6, depicting the growth of running times for different texture resolutions with respect to entities in the scene for rust simulation.

Texture Size Limitations: One set of variations differs from the base simulation in the resolution of the generated textures. There are variations for 512×512, 1024×1024 (base size), 2048×2048 and 4096×4096. The resulting iteration times are depicted in 4.6. The system scales fairly well until 1024×1024 as the resolution for each of the generated textures, with t_T growing almost linearly with the entity count, until reaching approximately 72 objects with large surface area. 2048×2048 remains usable until approximately a dozen objects, depending on the simulation, and 4096×4096 seems to only be practical for less than eight objects. Even larger textures from 8192×8192 upwards currently seem impractical. For one, the image files get fairly large at 8192×8192, using up about 15MB on disk per file. Each of the files needs to be encoded and written to disk in a blocking manner, which takes quite a while for dozens of textures. For accumulating effects, the textures need to be loaded from disk again. The implementation currently only keeps the surfel association tables permanently in memory. The surfel association table optimisation, however, seems counterproductive for such large texture sizes, especially when combined with high entity counts. The high memory consumption of keeping all these tables in memory leads to excessive swap space usage or even crashes, when swap space is depleted as well. All other textures, e.g., blend stops, need to be loaded from disk again whenever needed. This is desired behaviour in principle, limiting the amount of memory used at one instant in time. A more sophisticated caching strategy for textures can potentially provide a significant increase in performance for simple scenes, where most textures fit into memory, as well as prevent some crashes on extremely complex scenes, which need most available memory for surfels and triangles, such that most textures have to be cached on disk.

Entity Count Limitations: The previous performance evaluation variation varied a single parameter, namely the output texture sidelength. The following set of variations only varies the emission count and transforms the simulation scene instead. Not a single statue, but a grid of statues on pedestals is the new simulation geometry (see figure 4.7). Each instance of statue and pedestal also gets its own copies of the γ-ton sources in the scene. Hence, a grid of 10×10 statues has a scaling factor of 100 for the following simulation properties:

- surfel count,
- emission count,
- triangle count,
- output texture count.

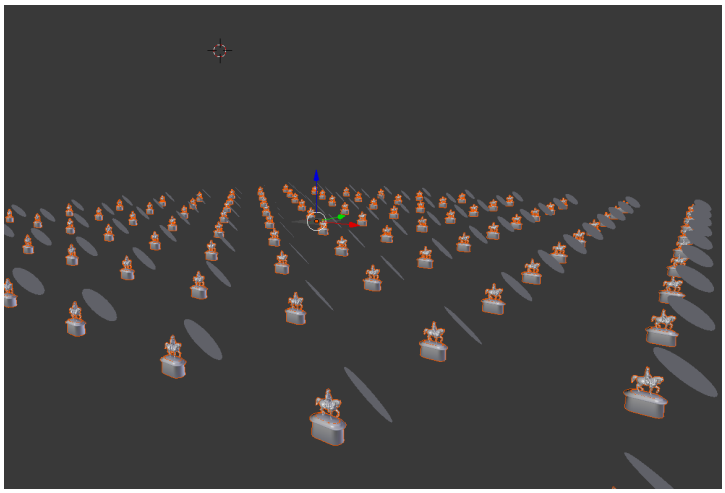


Figure 4.7: The largest variation of entity count stress tests, showing 10×10 Napoleon statues on pedestals.

By measuring the iteration times, the scalability of the approach to large scenes can be evaluated. The resulting timings are depicted in a graph in figure 4.6. The variations up to 50 entities (a 5×5 grid of two entities) with textures at resolution 1024×1024 or less handle the workload reasonably well. Most space requirements can be met with physical memory, and the running time grows relatively slowly. Performance becomes abysmal for more entities or larger textures, with $t_{\min}(1)$ exceeding one hour. The 8×8-variation with 128 entities already used up all physical memory and swap space, and had to be killed by the operating system while still preparing the simulation.

The numbers indicate adequate performance for small to medium-sized scenes. Extremely large scenes, are, as of now, infeasible for use with the implementation. Handling these extremely large workloads seems to require at least a different approach of memory management to gracefully handle the enormous amounts of surfels and triangles. A scheme for subdivision of the simulation in smaller units might be necessary to handle the memory cost. Extremely large scenes further require a high number of generated textures. This might make ahead-of-time generation of textures too expensive in terms of disk space. When distributing an application with objects weathered by *aitios*, a potential alternative to pre-calculating the textures would be to perform weathering simulation just-in-time in the target application at runtime.

Limiting Factors: Propagation performance in classical γ-ton tracing is said to depend primarily on the number of γ-tons emitted and on the number of surfels [4], which, when the triangle count is also considered (relevant for intersection check performance), also seems to be a sound analysis for the tracing time t_γ in the *aitios* variation. Regarding the complete iteration time $t(n, m)$, the performance of scenes is primarily limited by triangle count (Octree size and performance), surface area (surfel count, kD-tree), and entity count (texture count). The most performance-sensitive parameters of the simulation itself are the surfel distance, γ-ton emission count, and the amount and resolution

of textures utilised or generated in the effect pipeline. It appears that the performance is primarily limited by high memory requirements and by time-intensive intersection checks in complex scenes. Much time is also spent on blocking I/O tasks. Significant memory is required for triangles, surfels, live γ-tons, textures currently required in memory and various acceleration data structures.

Related Techniques in Comparison

γ-ton tracing, in the presented variation, is capable of off-line simulation of corrosion, taking into account exposure to weathering sources and differential flow. The performance of generation will, where data is available, be compared with related techniques in the following.

The authors of the original publication on γ-ton tracing from 2005 published the running times of the part of iterations spent on γ-ton propagation (comparable to t_γ in *aitios*). The following performance metrics have been measured on a PC with a 3 GHz Pentium IV processor (32-bit, single-core) and were published by the researchers:

- An example with 100k surfels and 15k emitted γ-tons is said to take approximately one minute to trace per iteration.
- Another example of 100k surfels and 63k γ-tons is said to take between 1.5 and 2 minutes to trace.
- For a scene of 200k surfels and 50k vertices, propagation of 10k γ-tons, mesh displacement and mesh retilement with the Turk method, takes a combined 3 minutes [4].

They make no claims about the performance of their appearance rendering technique based on multi-texturing and γ-ton map-guided blending [4], which would be an equivalent to the texture synthesis running time t_T. The classic technique is nonetheless assumed to be more performant, since only one blending operation is required. The presented blending approach in *aitios* requires two blending operations. One operation blends between two neighbouring blend stops, and one applies the blending result on top of the base material. Also, multiple blending layers may be stacked on top of each other.

It may appear at first sight that the presented implementation has superior tracing time t_γ, since one minute suffices for the tracing of vastly more γ-tons in scenes with more entities, each of greater complexity. Such a direct comparison of the metrics is clearly not admissible. The evaluation hardware for the presented implementation is 13 years younger and provides 6 cores at a similar core frequency (3.6 GHz), instead of a single one. The newer hardware can thus profit from parallelisation of the problem. Even if the worker thread count was limited to one and the core frequency limited to 3GHz, a faithful comparison is still impossible. Superior caching, speculative execution and other modern processor optimisations would still set the two models apart. Apart from the processor, all other hardware characteristics of the evaluation machine for the classical performance metrics are unknown, particularly the RAM and HDD characteristics. It is assumed that the single-threaded tracing performance would be similar when running on the same hardware, since the techniques are closely related. However, no serious claim can be made about relative performance for the lack of a reference implementation for classical γ-ton tracing.

While the original implementation appears to be CPU-based just like the presented one, Günther, Rohmer and Grosch presented a GPU-based variation in 2012, which is capable of running at interactive frame rates with the workloads presented in the publication [12]. The general process differs in many aspects. A time step in the Günther et al. variation cannot directly be compared to an iteration in aitios or an aspect of it. In *aitios*, an iteration traces a γ-ton to conclusion, with multiple substance transports for each particle, while the approach by Günther is designed to be interactive and does not terminate particles. Particles are usually in mid-air and will only sometimes collide with geometry to exchange substances. As a result, only a fraction of the particles exchange substances from one time step to the next. See section 2.4.4 for additional details on the approach.

The authors of the approach captured performance data on a computer with an Intel Core 2 Quad Q9650 CPU and an Nvidia GeForce GTX 560 Ti GPU with 3.5 GB VRAM and 4GB RAM, with promising results. On this hardware, interactive framerates are possible for all example scenes listed in the paper, using a single material atlas of size $1k \times 1k$, with a complete time step taking between $16.93ms$ and $18.04ms$. The bulk of this time is spent on texture composition ($5.91ms$ to $6.04ms$) and preview rendering ($7.95ms$ to $8.81ms$) and not on tracing ($2.72ms$ to $2.93ms$) and substance transport ($0.06ms$ for both γ-ton and surface updates) or ageing rules ($0.25ms$). At a texture atlas size of $4k \times 4k$, a complete time step drops to approximately $112ms$ for all examples. The authors see the main bottleneck in a high transfer rate during composition. The approach is limited to a small number of objects, but opens new possibilities to its use, such as interactive usage in gaming contexts [12].

The measured metrics do not have equivalents in the running times presented for *aitios*. The composition step, which takes about one third of a time step in the GPU approach, comes close to the texture synthesis running time t_{T}, though the latter includes image export time to disk, masking the core task of texture synthesis. It can be said though, that *aitios* is orders of magnitude away from running at interactive framerates and providing instant feedback to the artist, while the GPU approach can handle the task gracefully with moderate workloads.

Its CPU-based nature, on the other hand, gives *aitios* access to a large portion of the main RAM of the computer and makes simulation with medium-sized scenes of at least some dozens of objects feasible. The space complexity of the CPU variation, however, is not actually lower. With large scenes of 50 or more objects, physical memory will likely be insufficient for all surfels, triangles and the textures currently being operated on. This leads to excessive swap space usage, rendering the application effectively unusable due to extremely long iteration times. Particularly the employed optimisation of precalculating surfel associations or position textures for all entities becomes a limiting factor and would need to be disabled for large scenes.

Performance limitations may render the technique infeasible for scenes of vast dimensions. A variation of γ-ton tracing by João explicitly targets arbitrarily complex scenes, but imposes other limitations [1].

To ease future performance comparison with this concrete implementation, *aitios* has built-in benchmarking functionality and is publicly developed on GitHub[2]. The

[2] https://github.com/krachzack/aitios-cli

presented simulations are available as simulation specification files with associated resources on the accompanying CD.

4.4.3 Practical Applicability

Many corrosion effects can be effectively modelled using γ-ton tracing in the shown variation. The resulting level of physical validity and runtime performance can be considered acceptable for many use cases.

High-frequency details will often require either high-resolution and high-quality textures for blending, excessive processing power to calculate substance textures with very fine-grained details, or even the use of a different appearance rendering technique instead of the built-in blending, that is more suitable to fine-grained details. The appearance manifold technique produces high-quality details [26] and might be a good fit for an alternative appearance rendering technique.

Still, the technique lags behind hand-painting in some aspects. Especially in extremely small scenes with little potential for automation, the increased complexity of configuring and running a simulation instead of hand-painting might not be justified.

Apart from runtime performance, it can be said that γ-ton tracing-based weathering simulation has a non-trivial cost during production due to time spent on simulation preparation, specification, refinement, computation time itself (defined by runtime performance of the algorithm and available hardware), as well as preparation of the resulting output textures for shipping in the final product. The cost of the simulation itself can grow extremely high for complex simulations, inhibiting experimentation and parameter tweaking by the artist in some cases, or even failing to run to conclusion in the most complex of simulations. Since the approach is purely phenomenological, parameters of weathering effects can not be measured.

Summarizing the practical applicability of the *aitios* variation for corrosion effects, adequate running times and visual results can be reached with small to medium-sized scenes with up to 50 objects of moderate complexity (approximately 35k triangles each) and simulation parameters set to typical values.

Chapter 5

Conclusion

With this work, a well-defined variation of γ-ton tracing with associated texture synthesis functionality has been presented and implemented in the form of the weathering tool *aitios*. The resulting work has been evaluated and shown to be feasible for weathering effects in scenery consisting of some dozens of entities.

The variation primarily built on ideas present in classical γ-ton tracing, extending it with ideas drawn from the existing body of work around γ-ton tracing and some novel aspects. The resulting concrete variation lends itself to future comparison, improvement, extension and also re-implementation.

Algorithms for surfel-based texture synthesis including a conversion technique of surfel maps to substance distribution textures, have been proposed, allowing for integration of the technique with existing rendering pipelines and other tooling. As an alternative to external appearance rendering based on the plain substance textures, the simplistic blending-based technique that has been present in classical γ-ton tracing was extended to a layering-based approach. Instead of a direct blending of two textures, weathered appearances can be defined as progressions of optionally transparent weathered texture samples that are layered on top of the base material of the affected object. Where linear blending is not admissible, as is the case for normal maps, alternative blending techniques have been proposed.

Some advantages of the surfel representation have been pointed out, such as their resilience against texture discontinuities. However, when compared to a texture-based representation of surface data, the computational and memory overhead associated with surfels cannot be neglected, especially for scenes of large dimension. A purely texture-based workflow might not only be more memory efficient, but also provide faster lookup for near surfels. Such an approach is present in [12], and [7] also seem to store local surface information in textures.

The presented approach provides an effective means of simulating a range of natural surface ageing effects without domain-specific knowledge. In the evaluation chapter, applications in the context of corrosion effects have been presented and evaluated for physical validity and runtime performance. Limitations of the technique have been pointed out and contrasted with other techniques.

5.1 Limitations

The texture-based approach imposes some limitations on simulation geometry. These limitations, along with possible solutions, shall be summarized at this point.

Entity Count: Captured performance metrics suggest that a patina simulation with one iteration, 72 large entities and medium quality settings for 180 textures can be simulated in thirty minutes. Iteration times below one minute, where tweaking and experimentation by the artist is conveniently possible, are typically only possible for less than ten objects in a scene and texture resolutions below 2048×2048.

Surface Area: The space and time complexity of generation and lookup of surfels depend primarily on the combined area of all entities in the scene. Furthermore, the additional memory required to store a unique texture for every weathered object in the scene might exceed the available memory budget, depending on the generated texture sizes and whether texture synthesis is performed just-in-time or ahead-of-time. For highly complex and large scenes, running time of the algorithm or the combined space requirements of generated textures might render use of the algorithm infeasible without further optimisation.

Setup Cost: The γ-ton tracing technique allows for adequate estimation of substance distributions. One or more substances involved in different effects can be accounted for. Designing an effect requires careful observation of natural phenomena as well as some amount of experimentation and tweaking. Firstly, the main contributing substances have to be identified. Their source and transport properties need to be determined. Once the distribution of involved substances has been solved, designing the effect pipeline for texture synthesis again requires manual labour. Not only γ-ton emission needs to be configured, but also material composition. For each relevant material in the scene (fallback materials are supported), the properties of generated surfels need to be configured. Such properties include the γ-reflectance of the surface, representing the roughness of the surface, or the likeliness of stopping γ-tons and having them settle. If only few objects are affected by the weathering effects, manual design of the textures might actually be more efficient than deploying a simulation scheme. However, once substance behaviour and effect pipeline of a γ-ton tracing simulation have been designed, aspects of an effect may turn out to be highly reusable and can frequently be used in other contexts and for different objects without modification. A library of pre-defined simulation fragments might alleviate some of the configuration complexity.

UV Coordinates: The algorithm requires high-quality UV coordinates. In particular, UV islands should have padding, such that island bleed will only bleed into unmapped texture space. Furthermore, UV space cannot be shared by more than one triangle, since that would make it impossible to uniquely assign a material concentration to a texel. The point-based surfel representation of substance distribution data, in contrast, defines three-dimensional positions and hence does not require UV mapping. At least for offline rendering, a potential workaround would be to defer the substance concentration lookup to render time and perform a direct lookup into the surfel tree, performing the

weathering effect at the latest possible point. For GPU-based rendering at interactive framerates, a texture-based approach currently seems most efficient. It should be noted, however, that spatial data structures for use in shader programs are an active field of research. The means for efficient lookup of nearest surfels in GPU contexts may already be available, such that composition based on surfels might already be feasible at render time.

5.2 Further Prospects

This is not the first publication extending upon the γ-ton tracing approach, yet there are still some specifics that would warrant additional research. These shall be outlined in the following.

5.2.1 Scalability

The presented approach cannot handle arbitrarily complex scenes. Practical limits become obvious when many objects play a part in γ-ton tracing and texture synthesis. Future work could tackle this problem. One candidate for large amounts of geometry already exists in [1]. A new scheme for the handling of vast scenery would be especially useful if the specific limitations of GPU-based variations, such as [12], which currently exhibits similar scalability issues, were also considered.

5.2.2 Mechanical Phenomena

While phenomena that are directly attributable to the transport of substances can reasonably well be simulated with the presented approach, these often cause secondary effects that are not covered by the algorithm. For instance, painted metal protects metal from corrosion and the same paint peeling off can kick off corrosion beneath the painted surface in a separate layer, in turn causing more paint to peel off. For the problem of local growth, classical γ-ton tracing updates the reflectance properties via a user-defined "simple function", based on transport information after each iteration [4]. This functionality, unimplemented in *aitios* as of yet, can be exploited to increase weathering rates of already weathered areas. Kider shows promising work towards local growth in his μ-ton simulation framework. The particle-based approach explicitly handles diffusion into materials and local growth of blemishes, providing support for mechanical crackling and peeling [16].

5.2.3 Texture Synthesis

The synthesis functionality in *aitios* is based on the layering and blending of weathered texture maps over original textures. While this ensures frame-to-frame consistency, it can be limiting in some situations.

Projection Artefacts: The texture maps used for blending over originals are often not made for the specific geometry of the affected model, as this would pose substantial overhead for a scene with many objects. Rather, a specific substance is usually tied to one weathered texture progression shared by multiple objects to reduce production overhead.

The weathered texture thus cannot take seams in UV maps of affected objects into account, leading to artefacts at texture discontinuities. A pattern might seem awkwardly cut off or rotated when crossing a texture seam, making the texture appear synthetic. While it would be entirely possible to provide a tailored weathered texture for each affected object, doing so would require substantial human labour. Algorithms exist for the problem of adapting textures to different geometry without human intervention. These include the *Image Welding* approach [25], which would allow for the application of tiles of a single input texture directly onto surfaces of objects. The process would allow for the synthesis of entity-specific weathering blend stops, tailored to the objects UV map and geometry. This texture could again be used for simple blending operations to retain the property of frame-to-frame consistency.

Another potential problem arises with non-uniform texel-to-world-scales. The mapping of triangles on the surface of an object to UV space might exhibit non-uniform scales for technical reasons. Triangles are usually grouped with neighbouring triangles and projected into flat UV space, mostly resulting in some distortion. The non-uniformity can also occur on purpose, when some regions should have additional detail in texture painting, for instance the face on a humanoid model. Further, an instanced object might appear multiple times but in different sizes in the scene, making the texel-to-world-scale between objects different. Distinct objects with different geometries and texture maps might also exhibit inconsistent texel to world scales with respect to each other. This inconsistency arising from the stated reasons results in differently scaled weathering patterns both among objects and throughout the surface of a single object, if the same texture is applied during blending and the texture is not adapted to projection and scaling. If, as suggested before, custom made weathering textures are applied, a possible non-uniformity in texel-to-world-scale should be taken into account.

Visual Quality: Implementing *Time-varying Weathering In Texture Space* as proposed by Bellini and others [2] would allow for blemishes to gradually appear instead of blending in, without domain-specific knowledge about the weathering effect. This would presumably make blemish development more convincing with highly stochastic textures, such as cracking patterns. The work would allow for the generation of a weathered texture progression based on a single texture sample, instead of manually preparing the blend stops to *aitios*.

Appendix A

DVD Contents

The accompanying DVD contains supplementary data to the thesis, including:

- Source code of the implementation used for evaluation,
- simulation specification for the implemented simulation software,
- renderings using simulation results,
- 3D software data files used for rendering,
- a script for re-evaluation of performance.

A.1 Root Directory

Path: /

README.md	Contents of this appendix in Markdown format
thesis.pdf	Digital copy of this document
online/	Copies of online resources as referenced in the bibliography, accessed 2018-09-13

A.2 Binaries

Path: /bin

linux	ELF binaries for *aitios* and *litter*, pre-compiled for 64-bit Linux with glibc
mac	Mach-O binaries for *aitios* and *litter*, pre-compiled for Mac OS X
windows	Binaries for *aitios* and *litter*, pre-compiled for Windows systems
RUNNING.md	Instructions on using the binaries and running the shown simulations

A.3 Source Code

Path: /src

aitios.zip	Compressed snapshot of the source code of *aitios*, split among various sub-projects
litter.zip	Compressed snapshot of the source code of *litter*, used for performance evaluation

A.4 Evaluation Hardware Specifications

Path: /hardwarespecs

_SUMMARY.md	Human-readable hardware specifications, as summarized by the author
hwinfo_long.txt	Full output of the `hwinfo` tool on the evaluation machine running Arch Linux
hwinfo.txt	Shortened output of the `hwinfo` tool on the evaluation machine running Arch Linux
pci.txt	`lspci` output on the evaluation machine running Arch Linux

A.5 Simulations

Path: /simulations

data	Measured performance metrics as `*.data` files, suitable for use with `pgfplots`
effects	Specifications for the effect pipelines of the rust and patina evaluations
generated	Automatically generated complex versions of the top level simulations, with results, as well as results of the unmodified simulations, used for rendering
latex	Automatically generated tables for renderings and performance metrics as included into the thesis
render	Blender files used for the renderings in the thesis
run	Supporting files for the automatic simulating, rendering, stress testing and templating for the thesis
scenes	Simulation scenes in OBJ/MTL format, having a clean post-manufacturing state
simulations	Metadata and basic configuration of simulations
sources	Emission shapes in OBJ format and γ-ton source specifications
surfels	Surfel specifications for iron, copper and stone
templates	Templates for rendering and stress test tables

latest-patina-stresstest-timings.yaml Latest results of stress tests on the patina simulation in YAML format

latest-rust-stresstest-timings.yaml Latest results of stress tests on the patina simulation in YAML format

run.rb . . . . . Script for simulating, rendering, performance evaluation and snippet building

References

Literature

[1] João Montenegro Almeida. “Particle Driven Weathering System”. MA thesis. Bournemouth, UK: Bournemouth University, National Centre for Computer Animation, Sept. 2007 (cit. on pp. 17, 68, 72).

[2] Rachele Bellini, Yanir Kleiman, and Daniel Cohen-Or. “Time-varying Weathering in Texture Space”. *ACM Transactions on Graphics* 35.4 (2016), 141:1–141:11 (cit. on pp. 4, 14, 15, 63, 73).

[3] Carles Bosch et al. “Image-guided Weathering: A New Approach Applied to Flow Phenomena”. *ACM Transactions on Graphics* 30.3 (2011), 20:1–20:13 (cit. on p. 4).

[4] Yanyun Chen et al. “Visual Simulation of Weathering By Gamma-ton Tracing”. *ACM Transactions on Graphics* 24.3 (2005), pp. 1127–1133 (cit. on pp. 4, 5, 14–17, 21, 22, 24, 25, 28, 29, 31–34, 36, 43, 45, 54, 66, 67, 72).

[5] David Cline et al. “Dart Throwing on Surfaces”. *Computer Graphics Forum* 28.4 (2009), pp. 1217–1226 (cit. on pp. 26, 27).

[6] Julie Dorsey and Pat Hanrahan. “Modeling and Rendering of Metallic Patinas”. In: *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques* (New Orleans, Louisiana, USA). Ed. by John Fuji. New York, New York, USA: ACM, Aug. 1996, pp. 387–396 (cit. on pp. 4, 7–9, 53, 62).

[7] Julie Dorsey, Hans Køhling Pedersen, and Pat Hanrahan. “Flow and Changes in Appearance”. In: *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques* (New Orleans, Louisiana, USA). Ed. by John Fuji. New York, New York, USA: ACM, Aug. 1996, pp. 411–420 (cit. on pp. 7, 9–11, 17, 28, 32, 51, 54, 62, 70).

[8] Alexei A. Efros and William T. Freeman. “Image Quilting for Texture Synthesis and Transfer”. In: *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques* (Los Angeles, California, USA). Ed. by Eugene Fiume. New York, New York, USA: ACM, Aug. 2001, pp. 341–346 (cit. on p. 12).

[9] Dhana Frerichs, Andrew Vidler, and Christos Gatzidis. “A Survey on Object Deformation and Decomposition in Computer Graphics”. *Computers & Graphics* 52.C (2015), pp. 18–32 (cit. on pp. 5–7, 16, 51, 53).

[10] Dhana Frerichs, Andrew Vidler, and Christos Gatzidis. “Object Weathering Simulation Avoiding Texture Space Stretching and Discontinuities”. In: *Proceedings of the International Conference on Computer Graphics and Interactive Techniques* (Shenzhen, China). New York, New York, USA: ACM, Dec. 2014, 37:1–37:1 (cit. on pp. 20, 43).

[11] Stéphane Gobron and Norishige Chiba. “Crack Pattern Simulation Based on 3D Surface Cellular Automaton”. In: *Proceedings of the International Conference on Computer Graphics* (Geneva, Switzerland). Washington, DC, USA: IEEE Computer Society, June 2000, pp. 153–162 (cit. on p. 7).

[12] Tobias Günther, Kai Rohmer, and Thorsten Grosch. “GPU-accelerated Interactive Material Aging”. In: *Proceedings of the 17th International Workshop on Vision, Modeling and Visualization* (Magdeburg, Germany). Ed. by Michael Gösele et al. Geneva, Switzerland: Eurographics Association, Nov. 2012, pp. 63–70 (cit. on pp. 19–21, 28, 31, 32, 43, 68, 70, 72).

[13] Henrik Wann Jensen and Niels Jørgen Christensen. “Photon maps in bidirectional Monte Carlo ray tracing of complex objects”. *Computers & Graphics* 19.2 (1995), pp. 215–224 (cit. on p. 17).

[14] Shaohui Jiao, Gang Yang, and Enhua Wu. “Weathering fur simulation”. In: *Proceedings of the 16th ACM Symposium on Virtual Reality Software and Technology* (Kyoto, Japan). Ed. by Steven N. Spencer. New York, New York, USA: ACM, Nov. 2009, pp. 271–272 (cit. on p. 17).

[15] Z. Jiayin and Z. Mingquan. “Improved Gammaton Tracing Technique Using Height Field Profile Tracing”. In: *Proceedings of the 38th Annual Conference on Computer Applications and Quantitative Methods in Archaeology* (Granada, Spain). Ed. by Mercedes Farjas Contreras Francisco and Francisco Javier Melero. Oxford, UK: Archaeopress, Apr. 2010, pp. 63–70 (cit. on p. 17).

[16] Joseph T. Kider. “Simulation of 3D Model, Shape, and Appearance Aging by Physical, Chemical, Biological, Environmental, and Weathering Effects”. PhD thesis. Philadelphia, USA: University of Pennsylvania, 2012 (cit. on pp. 7, 12, 17–19, 72).

[17] Joseph T. Kider, Samantha Raja, and Norman I. Badler. “Fruit Senescence and Decay Simulation”. *Computer Graphics Forum* 30.2 (2011), pp. 257–266 (cit. on pp. 4, 7, 36).

[18] Jianye Lu et al. “Context-aware Textures”. *ACM Transactions on Graphics* 26.1 (2007), 3:1–3:22 (cit. on pp. 5, 6, 13, 14, 36).

[19] Stéphane Mérillou and Djamchid Ghazanfarpour. “A Survey of Aging and Weathering Phenomena in Computer Graphics”. *Computers & Graphics* 32.2 (2008), pp. 159–174 (cit. on pp. 5, 6).

[20] Peter Mileff, Károly Nehéz, and Judit Dudra. “Accelerated Half-Space Triangle Rasterization”. *Acta Polytechnica Hungarica* 12.7 (Dec. 2015), pp. 217–236 (cit. on p. 38).

[21] Gavin Miller. “Efficient Algorithms for Local and Global Accessibility Shading”. In: *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*. New York, New York, USA: ACM, 1994, pp. 319–326 (cit. on p. 8).

[22] Tomas Möller and Ben Trumbore. “Fast, Minimum Storage Ray-triangle Intersection”. *Journal of Graphics Tools* 2.1 (1997), pp. 21–28 (cit. on p. 29).

[23] Robert Osada et al. “Shape Distributions”. *ACM Transactions on Graphics* 21.4 (2002), pp. 807–832 (cit. on p. 25).

[24] Hanspeter Pfister et al. “Surfels: Surface Elements As Rendering Primitives”. In: *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques* (New Orleans, Louisiana, USA). New York, New York, USA: ACM, July 2000, pp. 335–342 (cit. on pp. 22, 25).

[25] Muath Sabha and Philip Dutré. “Image Welding for Texture Synthesis”. In: *Proceedings on Vision, Modeling, and Visualization* (Aachen, Germany). Ed. by L. Kobbelt et al. Berlin, Germany: Akademische Verlagsgesellschaft Aka GmbH, Nov. 2006, pp. 97–104 (cit. on p. 73).

[26] Jiaping Wang et al. “Appearance Manifolds for Modeling Time-Variant Appearance of Materials”. *ACM Transactions on Graphics* 25.3 (2006), pp. 754–761 (cit. on pp. 4, 12, 13, 15, 62, 69).

[27] Su Xuey et al. “Image-based Material Weathering”. *Computer Graphics Forum* 27.2 (2008), pp. 617–626 (cit. on pp. 12, 13).

Online sources

[28] Joki 3D. *Old rusty horror sink*. Model of a rusty sink with handpainted rust textures, licenced Turbosquid Royalty Free License, All Extended Uses. Dec. 2016. URL: https://www.turbosquid.com/FullPreview/Index.cfm/ID/1105321 (visited on 08/19/2018) (cit. on p. 52).

[29] Colin Barré-Brisebois and Stephen Hill. *Blending in Detail*. Blog post, comparison of normal blending techniques. July 2012. URL: https://blog.selfshadow.com/publications/blending-in-detail/ (visited on 07/09/2018) (cit. on pp. 40, 41).

[30] Benjah-bmm27. *Sample of iron(III) oxide*. May 2007. URL: https://en.wikipedia.org/wiki/Iron(III)_oxide#/media/File:Iron(III)-oxide-sample.jpg (visited on 09/19/2018) (cit. on p. 52).

[31] Joaquim Alves Gaspar. *Equestrian statue of King D. José. Praça do Comércio, Lisboa, Portugal*. Image of a copper statue recently cleaned of patina, Wikimedia Commons, Licensed CC BY-SA 3.0. 2015. URL: https://en.wikipedia.org/wiki/File:Lisboa_January_2015-22.jpg (visited on 06/26/2018) (cit. on p. 53).

[32] Joaquim Alves Gaspar. *The statue of D. José I, Lisboa, Portugal*. Image of patina on a copper statue, Wikimedia Commons, Licensed CC BY-SA 3.0. 2011. URL: https://commons.wikimedia.org/wiki/File:Lisboa_November_2011-13.jpg (visited on 06/26/2018) (cit. on p. 53).

[33] haridon. *3D Rusty helicopter*. Model of a fantasy helicopter, with handpainted rusting textures, licenced Turbosquid Royalty Free License, All Extended Uses. Apr. 2017. URL: https://www.turbosquid.com/FullPreview/Index.cfm/ID/1143883 (visited on 08/19/2018) (cit. on p. 52).

[34] mali maeder. *Untitled*. Photograph of corroded iron bars. June 2016. URL: https://www.pexels.com/photo/abandoned-daylight-decay-destroyed-241523/ (visited on 09/19/2018) (cit. on p. 52).

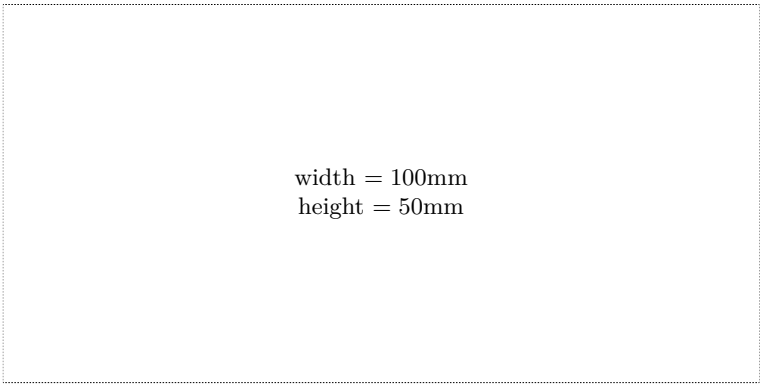
[35] Marlith. *File:RustyChainEdit1.jpg*. Surface breakdown, cracking and flaking in a photograph of a rusting chain, licensed Creative Commons Attribution-Share Alike 3.0 Unported. Nov. 2008. URL: https://en.wikipedia.org/wiki/File:RustyChainEdit1.jpg (visited on 07/09/2018) (cit. on p. 52).

[36] Loïc Norgeot. *Equestrian statue of Napoleon*. Simplified, UV-unwrapped and retextured version of a 3D Scan of a copper statue, licensed CC BY 4.0. 2017. URL: https://sketchfab.com/models/ad3fe0536f4f40c5a2de5b5649b2e639 (visited on 06/28/2018) (cit. on pp. 53, 54).

[37] IMA Solutions. *Napoleon*. 3D Scan of a copper statue on Champs-Élysées, Paris, Scanned by IMA Solutions for Musée de la Révolution française. 2017. URL: https://www.artec3d.com/de/3d-models/napoleon (visited on 06/27/2018) (cit. on pp. 53, 54).

Check Final Print Size

— Check final print size! —



— Remove this page after printing! —