HOCHSCHULE
RAVENSBURG-WEINGARTEN
UNIVERSITY
OF APPLIED SCIENCES

Studiengang Angewandte Informatik
Fakultät Elektrotechnik und Informatik

Bachelor Thesis

to the subject of

Analysis of procedural materials within fragment shaders

by
Eric Dolch

Adviser:
Prof. Dr. Daniel Scherzer - Hochschule Ravensburg-Weingarten
M.Sc. Dennis Reimer - Hochschule Ravensburg-Weingarten

Statutory Declaration

I hereby declare, that the present thesis was written and developed self-reliant and under exclusive use of the stated literature and resources. The thesis was not submitted in the same or similar form or in extracts to any testing authority yet.

_____ (Signature)

_____ (City, Date)

Table of content

Abstract

Render applications that offer shading interfaces for objects can be used to create entire procedural materials without dependencies to textures or other external references. It is assumed that these interfaces work like fragment shaders without access to buffers and same restrictions to evaluate surface information at arbitrary points. The process of creating such materials can be complex, time consuming and requires an understanding of the underlying algorithms. The thesis aims to create an understanding for surfaces in general, abstract technical knowledge to make it more accessible and structurize the workflow of the creation.

1 Introduction

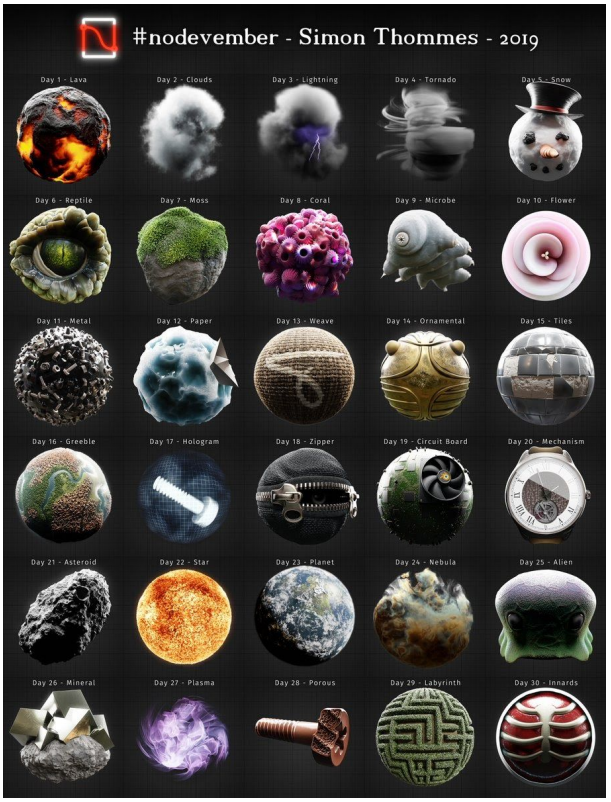
Procedural texturing always has been a subject in computer graphics. Researchers sought for algorithms and improvements to synthesize textures to represent natural looking surfaces. Early algorithms like Perlin-Noise [KP] or Worley-Noise [SW] are still present today and essential for procedural generations, due to their natural looking appearance, which suits replicating natural surface properties.

[Figure01] *Images from early papers; Left: Perlin noise [KP]; Right: Worley noise [SW].*

Today, "With the ever increasing levels of performance for programmable shading in GPU architectures, hardware accelerated procedural texturing in GLSL is now becoming quite useful[...]" [SG], because "[...] modern shader-capable GPUs are mature enough to render procedural patterns at fully interactive speeds [...]" [SG]. That enables essential types of algorithms for procedural pattern generation like noise or distance fields to be used as implicit implementation in shaders, because shaders will query for information about arbitrary points, set by the current pixel distribution.

Today, a variety of modern 3D applications like "Blender" [BL], "Unity 3D" [UN], "Unreal Engine" [UR] or "Cinema 4D" [CN] offer interfaces to attached renderers to handle the shading of objects in a modular manner, as proposed in the paper "shade trees" [RC]. Besides enabling modifications to the shading of the mesh and creative abstract lighting models like toon shading, the interfaces enable at the same time the generation of procedural surface property information. While the interfaces of real time renderer already make use of the GPU with fragment shaders, offline render applications ,which are able to use ray tracing instead of rasterizing, also require implicit algorithms to evaluate a surface color at arbitrary points by ray intersections. Therefore the functionality and behavior of these interfaces are oriented towards fragment shaders. The interfaces are also shipped with abstract implementations of various useful algorithms to hide the underlying complexity [2.4].

Artists like Simon Thommes [Figure02] already pushed the limits of these interfaces really far and they were able to create stunning materials, using procedural methods without dependencies to external resources like textures.



[Figure02] *Procedural Materials in Blender, created for "Nodevember"; Made by Simon Thommes 2019 [TS].*

1.1 Motivation

Due to the possibility to use and layer multiple algorithms in shaders and interfaces of various render applications, creating procedural materials can be still a tedious and complex task. Manipulating results of algorithms in the right manner often relies on repetitive tasks and practical knowledge to get convincing results. Because the endless possibilities and creative freedom for manipulations and choice of algorithms in shaders and interfaces of render applications will not enforce or guide creators to a specific workflow to create procedural materials. And additionally, only implicit algorithms for pattern generation can be used, because of the shader architecture. This excludes the use of post processing algorithms like blur, normal map generation from height or ambient occlusion. Post processing algorithms rely on neighbor information, which cannot be accessed without buffers in fragment shaders. Buffers are not available in every render application interface and if it does, the usage of them can vary for each render application interface.

1.2 Objectives

This thesis will serve multiple objectives. First, an understanding for real world surfaces and their composition should be created. Therefore it has to be analyzed, how they can be decomposed in distinct information, layers, forms and patterns. Secondly, to know, which algorithms are suited for procedural generation and which common use cases are occurring, a categorization based on their task and type needs to be created. Thirdly, guidelines have to be defined, in order to reduce Trial-And-Error phases and guiding creators to a structural process. Finally the analysis, categorization and the capabilities of the workflow are tested by creating a procedural texture and documenting each step.

The order of the named objectives will also represent the structure of the thesis. Details about implementation or specific algorithms are not part of this work. As well as a performance analysis of algorithms or entire procedural materials.

2 Prerequisites

While dealing with render applications, procedural texture generation and shaders, some terms can have similar meaning and sound. Therefore their meaning in this work has to be defined to prevent misunderstandings.

2.1 Procedural

The paper "A Survey of Procedural Noise Functions" defined “procedural” as:
"The adjective procedural is used in computer science to distinguish entities that are described by program code rather than by data structures. Procedural techniques are code segments or algorithms that specify some characteristic of a computer-generated model or effect." [PF]

2.2 Textures

Textures are images, where information about surface properties is stored. Combined with the definition for "procedural", a “procedural texture” is defined by Dr Sebastien Deguy as: "[...] a computer-generated image created using an algorithm [...], instead of a digital painting or image processing application[...]" [DS]. Render applications are then using these textures to feed the properties of an assigned lighting model.

2.3 Materials

Most render applications using the term "material" for the combination of the used lighting model and the collection of information for the lighting model, like albedo or specularity. This thesis uses the term "material" for the collection of information only, excluding the lighting model. Because the lighting model has only a minor influence on the process of replicating a surface in a procedural manner.

[Figure03] *Different lighting models, same material information; Right: Physical based (PBR); Left: Non-photorealistic (NPR).*

As seen in [Figure03], with different lighting models the visual appearance of surfaces can change drastically, even if the offered information about the surface properties are the same. The lighting model can influence the process of procedural materials by requiring special information. Nonetheless this will not change the general approach of how to analyze and replicate surfaces, as well as the underlying techniques and algorithms.

2.4 Implicit and explicit algorithms

Implicit algorithms are more suited for procedural materials than explicit ones. The difference between those two is, that an implicit algorithm will answer a query about an arbitrary point and returns information exclusively for it, while an explicit algorithm returns the whole result, evaluated for a resolution defined by the renderer and not by the shader itself. [PA] Due to the architecture of shaders, regardless of whether used by ray tracing or rasterization, the task of shaders is to evaluate arbitrary points of a surface [HP]. These points are defined either with rasterization by the current resolution and view and in case of ray tracing, the points are determined by the point of impact of emitted rays. Implicit algorithms which can return results to these points without dependencies to neighbor point information and resolution are therefore preferred or even necessary.

2.5 Benefits & drawbacks of procedural patterns within fragment shaders

Related work already has explored and analyzed several advantages and disadvantages of pattern creation in fragment shaders, besides the named problems for the motivation. While they are not part of this thesis they still have to be pointed out. The book “Texturing and Modeling - A Procedural Approach” [PA] already made a good listening of these pros & cons, which can be represented shortened as:

- The size of a procedural representation is way smaller than saving their result as texture.
- The evaluation can be executed at any resolution.
- Procedural materials can be parameterized to change the appearance and features.

Disadvantages are:

- The development process can be difficult, because of the complexity of algorithms and the lack of debug possibilities.
- Results of chaining algorithms are hard to plan without practical knowledge.
- Evaluating can be slower than accessing textures.
- Aliasing can be a problem, especially for far zoomed out textures.

3 Analysis of surfaces

The overall objective of the analysis should not be confounded with building a physical and chemical understanding of real world materials, which nonetheless may be helpful or necessary. The main objective is to extract patterns and geometric information about the visual appearance. To retrieve these information, the analysis of surfaces is carried out in three steps:

- Extracting surface layers
- Visual properties of materials
- Environmental influences

The steps were derived on one hand by looking through guides of the specialized software for procedural texture creation “Substance Designer” [SD01], and on the other hand by transferring the knowledge to a pure shader based environment. While creating several textures, these steps have been evolved over the time by analyzing several types of surfaces with the goal in mind to get the information as structured as possible. To support each step in their explanation, their concept is applied to an example reference photo of a wooden floor in a Pub.

The proposed extraction steps rely on pattern, noise and shape recognition. The retrieved information about the surface composition and visual features is reused later by replicating them with matching algorithms, order and techniques, that are available to fragment shaders. To utilize the extracted information, they can either be implemented directly, while analyzing the surface in parallel or persisted in any preferred way. The implementation does rely on the content of the information, not on its persistence.

3.1 Extracting surface layers

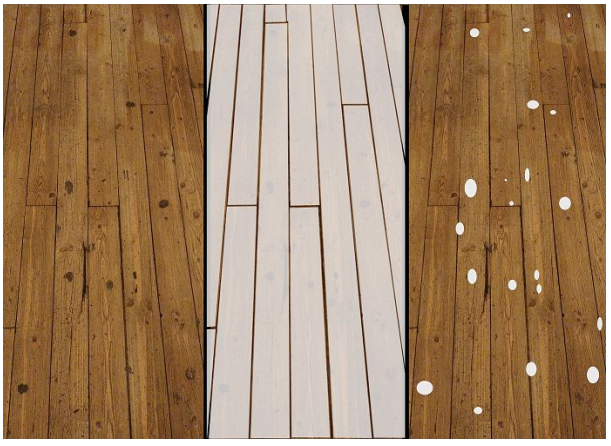
Separating surfaces into different layers of sub-materials helps, later to reassemble the surface in the material in the same manner as image editing software does this through blending them into an final image. In addition the separation serves as simplification to the reference-surface to overcome possible complex compositions, which then can be understood more easily detached from other influences.

A hierarchical approach for looking out for layers is recommended, because the depth of the analysis will differ by the required level of detail for the material. Further the hierarchical approach can also mimic the creation process, physical and chemical processes over the lifetime. These influences create natural layers of materials on a surface, like leaves on the ground, oxidations or screws in furniture, which almost matches the chronological appearance and history of the surface. Another advantage of a hierarchical approach is the option that replicated materials can be reused in other materials, because the layering should only control their distribution in the final material.

Factors which are decisive for separations are:

- Physical factors; Many surfaces are already separated naturally through manufacturing processes, like stone walls and floors, where the final surface is man made. It also includes occurrences, where different materials are placed on top or worn away without initial intent, like posters on a wall, leaves on grass or peeled plaster
- Chemical Factors; surfaces will change their appearance over time. Oxidation is a process typically occurring everywhere for metals, where the metal slowly converts to rust.
- Abstract patterns; Surfaces may also have noticeable features which are expressed in height, color or other properties without dependency to a distinctive material. Patterns on wallpapers or height structures on surfaces to provide more grip are an example of that.

The proposed separation into layers should not be confounded with layered material rendering where specialised lighting models simulate light interactions of layered surfaces [WJ]. The idea separating a surface into layers is to prepare information to control their visibility for the attached lighting model. Besides that, if a complex lighting model is used that has the ability to simulate layers of materials on a surface, the separation might be useful to create required information for the model.



[Figure04] *Left: Floor of a Pub; Mid: Separation by planks; Right: Separation by trampled gums.*

The first separation, see [Figure04], is made by the wooden planks, because:

- The planks are physically separated and are independent to each other.
- they are the most perceptible feature of the floor.
- they control the base height of the surface.
- the arrangement and shape results in an almost regular pattern.

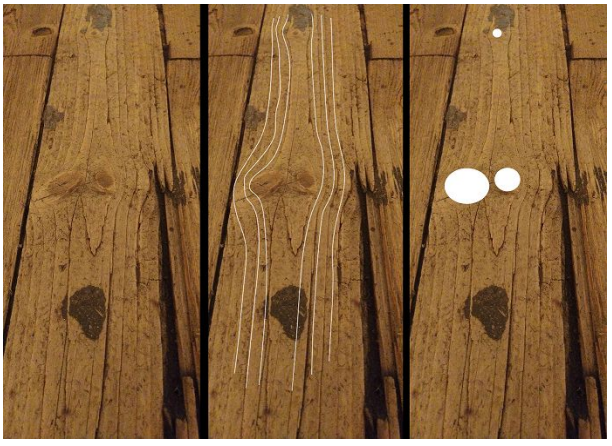
While the first separation is oriented to the wood structure, the black dots on the floor appear to be independent to the plank structure. This is because these dots are old trampled chewing gums. Therefore a second separation is made since:

- they do not show a dependency to the plank or wood structure and overlap some planks.
- they are a different material than wood.
- they are physically placed on top of the floor.

3.2 Visual properties of materials

After separating a surface in multiple layers, visual properties about the isolated materials have to be extracted for recreation. Again a hierarchical approach is recommended, this time driven by the obtrusiveness of visual features, which ensures that the material is later immediately recognizable. The hierarchical approach matches the later proposed workflow for implementation by iterating from rough to small details. This only takes features into account, which are necessary and required by level of detail for the material.

While the factors for the separation often are quite obvious, the extraction of features from materials is still oriented to the named factors earlier.



[Figure05] *Left: Single plank of the floor; Mid: Annual rings, Right: Branches.*

The floor planks are made of wood. The two most recognizable features of wood are annual rings and branches, which are also represented in the reference photo. There are more features in the reference photo, which can be extracted, but the two initial features, annual rings and branches are sufficient for a first recognizable implementation of a wood material.

3.3 Environmental influences

While a surface can be separated into layers, and the materials which made up the layers are treated separately, a surface is still part as a single instance of the environment. This means the environment, where the surface exists, has a strong influence on the appearance. Trampled chewing gums on the floor are a result of an environmental influence, because the floor is located in a public place, a wooden floor in a living room may not have the feature of trampled gums.

Visual features which may appear at first glance inexplicable often can be explained by looking though the history of a surface. Environmental influences are the factors which make materials substantially more authentic. Thinking about the environment and gathering background information about the history, conditions or story can leave visual impressions on surfaces. This knowledge can be applied factionary to surfaces to integrate them in the resulting material to fit more convincingly in the final environment.

[Figure06] *Left: Floor in a Pub; Mid: burned spots from trampled cigarettes; Right: color variation due to spilled liquids.*

The wooden floor is no exception to that, because it is located in a smoking area and in the reference photo are all over the place small dark points like "freckles" on the floor. With a close inspection these freckles are identified burned spots from trampled cigarettes. Another information to consider in a pub is the possibility of spilled liquids. These liquids may not be wiped away immediately, so the liquid will be soaked up from the floor, which can cause discolorations to the wood.

4 Toolbox of algorithms

Various algorithms are required to replicate the extracted information from the analysis in the form of shapes, noises and patterns. Through exploring different purposes of algorithms that are suited to be implemented in fragment shaders, a library of algorithms dedicated for procedural material generation has been formed. This library serves as the origin for the derivation of the proposed categorization and as a toolbox for the workflow to avoid reimplementation.

Adding algorithms to the library resulted in an evolving categorization by their purpose. This categorization has been settled and shown to cover all tasks of procedural material creation with fragment shaders. The categorization was made to solve different purposes:

- To keep the library organized and make collected algorithms easily locateable by the required task, because their name does not always conclude to their purpose. The naming of basic mathematical methods like “absolute” or “sinus” might be clear, complex algorithms, including self-made ones, may have arbitrary names like “valleys” or “voronoi”.
- By implementing algorithms in an abstract manner and without definite specialization for use cases, they enable a modular use of them, which can be compared to Lego bricks. Despite the abstraction, they still lead to distinctive and individual results through chaining and layering. Furthermore the categorization and abstract implementation should support the exploration of algorithm combinations for new results, because of their exchangeability and compatibility.
- Each interface of render applications will ship with different subsets of already implemented algorithms. And in case of pure GLSL, only basic mathematical methods are given. A categorization allows changes for available collections to complete them by adding missing required algorithms. It also enables the classification of altered or specific algorithms of own choice, which may be explored while creating new materials.

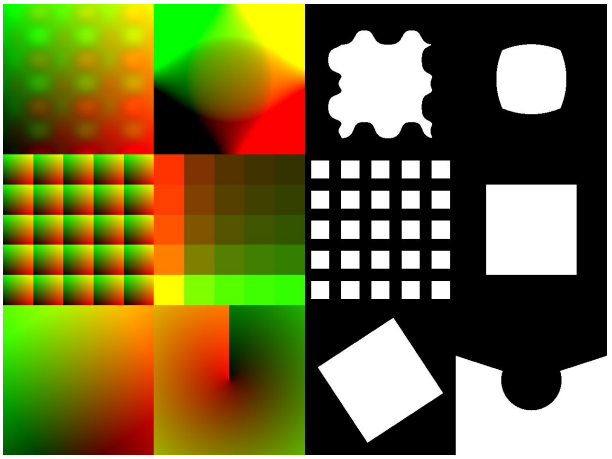
4.1 Basic math

Basic mathematical methods like "floor", "absolute" or "sinus" are essential for procedural generation and forming the base of every algorithm. This includes mathematical operations as vector math, trigonometry and matrices. Another use case of basic operations is the processing in results of previous algorithms, where results are either used to be combined or as origin to create new information.

4.2 UV

As mentioned in [2.4] (Implicit and explicit algorithms), through ray tracing or rasterization are positions on surfaces evaluated, where the underlying material information is required. The evaluated position, which usually is two or three dimensional, results through a conversion in texture coordinates that are commonly named “UV”. The UV represents the required location of the material information as two dimensional coordinates. These coordinates are usually used to access a texture with persisted material information for the lighting model. By generating procedural information, the UV serves as a pointer of the location for the implicit generative algorithms and therefore the resulting information [HP], which makes them just as important as basic math functions.

Because the UV defines the location of generated information, manipulating them concludes to a change in the generated information distribution, which take effect in the visual appearance of subsequent algorithms. While implementing transformations directly into the generative algorithms like scaling, translation or rotation can be quite complicated but applying changes to the UV results in two major advantages. First, many algorithms require the UV coordinates anyway, because of that manipulations done in beforehand and outside of the algorithm can be shared and reused by multiple algorithms. Second, abstracting UV manipulations enables transferring them to any algorithm that utilizes UV coordinates. This leads to endless possible combinations which otherwise could not be achieved.



[Figure07] *Left: UV manipulations; Right: rectangle with same size and position drawn with UV's from left.*

Besides basic transformations such as scaling, translating and rotating are more complex manipulations possible. These may be often required or enable unique results. Tilling is one common manipulation, where a given UV is fractured in regular sub-UV's. This is a common approach to mimic repetitive patterns. [Figure07] shows the UV and their result in the first column and middle row.

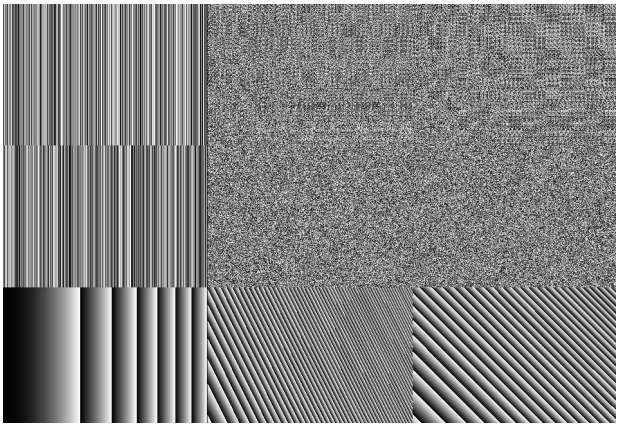
Another manipulation is the conversion into another coordinate system. The UV usually represents the cartesian coordinate system. There are other systems and projections where cartesian coordinates can be transformed to. An useful transformation could be to the polar system, where the position is described as angle and length. [Figure07] shows this in the second column and bottom row.

4.3 Noise

Noise algorithms are the element, which mimic unpredictable patterns or random distributions on surfaces. There are two kinds of algorithms for this category. The first are algorithms that will work as pseudo random number generators (RNG), the second are noise algorithms, which rely on the RNG's to create unpredictable, yet still repetitive gradients.

4.3.1 Hashing as random number generator

The base of all noise algorithms and random distributions is the access to a random number generator (RNG). While true randomness is hard to achieve with computers, it is even undesirable for creating procedural noise. A RNG for procedural materials has to be unpredictable and reproducible at the same time. Often algorithms need to restore random values e.g. accessing the value of neighbor points in a lattice [PF].



[Figure08] *White noise; Left to right: 1D, 2D, 3D; Bottom to top: Scaled by x0.0001, x1.0, x1000.0.*

Hashing is the perfect solution to be used as RNG, because the results can be unpredictable but still controllable through input [WN]. A good result of such a hash function is named "white noise", which contains all frequencies at once. But not any function is suited to be used as a RNG. The hash algorithms should be consistent over the used UV scale in the procedural material. As shown in [Figure08], the randomness of hash algorithms can break in extreme scales. There is a good listing in the book "Texture & Modeling A procedural approach" of other properties that a hash algorithms have to fulfill in order to be used as RNG:

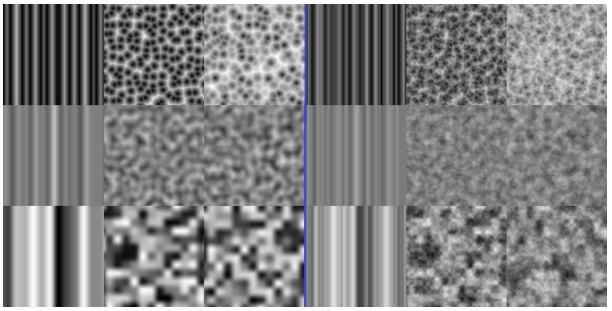
"The properties of an ideal noise function are as follows:

- noise is a repeatable pseudorandom function of its inputs.
- noise has a known range, namely, from −1 to 1.
- noise is band-limited, with a maximum frequency of about 1.
- noise doesn't exhibit obvious periodicities or regular patterns. [...]
- noise is stationary - that is, its statistical character should be translationally invariant.
- noise is isotropic - that is, its statistical character should be rotationally invariant."

(the term "noise" from the quote is referred as white noise) [PA]

4.3.2 Noise base functions

Base functions of noise are considered as functions which create repetitive unpredictable gradients by interpolating random numbers. The paper "A Survey of Procedural Noise Functions" [PF] gives a good insight about the noises themselves and a categorization of their types. The most known algorithms are perlin noise [KP] and voronoi noise (also known as cellular noise) [SW].

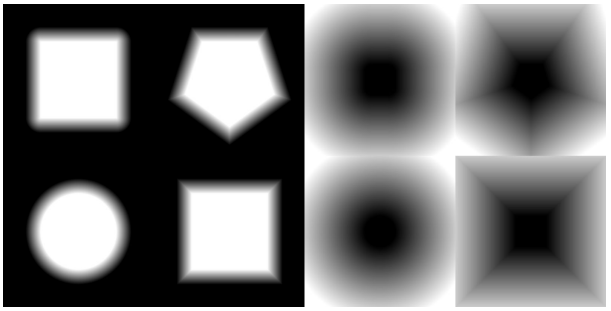


[Figure09] *Several noises; Left to right: 1D, 2D, 3D; Bottom to top: value, perlin, voronoi; Left: pure noise; Right: fractal Brownian motion based on left sided algorithms.*

As shown in [Figure09] the visual appearance of noise can be very different. This is useful in terms of mimicking different natural patterns, where some algorithms are better suited than others. Another consideration of collecting algorithms, besides the visual appearance, is their application in different dimensions than 2D. One dimensional noises are quite useful when it comes to created color gradients. Their result could either be used to mix defined colors, or their results could control a single color channel. A good example of controlling color channels was made by Inigo Quilez, where he used a single offsetted and scaled cosines function for each channel [IQ04]. On the other hand, three dimensional noises are useful when it comes to accurately mimicking a cross section, the third dimension can be replaced for that with a consistent number.

4.4 Shapes

While noise algorithms are utilized to mimic unpredictable surface structures, shape algorithms are the complementary solution for geometric structures. These structures will appear likely on man-made surfaces like rooftops or fabrics. But also natural surfaces can show derivatives of geometric forms like leaves or pebbles in mud. To mimic geometric features, algorithms for basic shapes are needed. These algorithms should produce, like noise algorithms, a single value within the range of 0 and 1. Not every complex geometric shape has to be implemented as an algorithm. They can be often reassembled by combining basic shapes with boolean operations as subtraction, intersection and union.

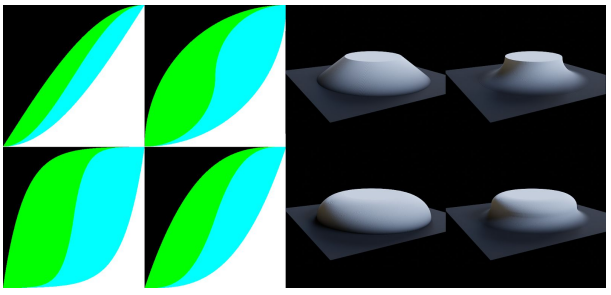


[Figure10] *Left: various shapes with blur; Right: Distance fields.*

An important feature which every shape algorithm has to provide is blur, as shown in [Figure10]. Unfortunately this has to be done within the shape generation, because after the generation there is no way to blur a shape or information without access to neighbor information, which is not given in shader. Blurred shapes are essential for creating procedural materials, because geometric shapes, like noise, have transitions that are represented in various features of a surface, where a transition into another height or color is needed. Without blur there would be an instantaneous change in information, which does not reflect most surfaces. To provide blurred basic shapes, using a distance field as base is recommended because the blur can be controlled through a start and end distance. Inigo Quilez made a convenient listing of several distance functions in 2D and 3D space on his website [IQ02][IQ03]. Another detail for blurring shapes is that the blur should be linear, because the linearity allows easier modifications to the transition distribution easier and exchangeable.

4.5 Easing

Easing is a well known technique for animations to improve linear interpolation between keyframes to emphasize them [RP]. For procedural materials, easing functions enable the recreation of characterizing information, like height or color, from gradients which are given through shapes or noises (shown in [Figure11]). Easing is an elegant way to change the distribution of gradients to create new deviated results.



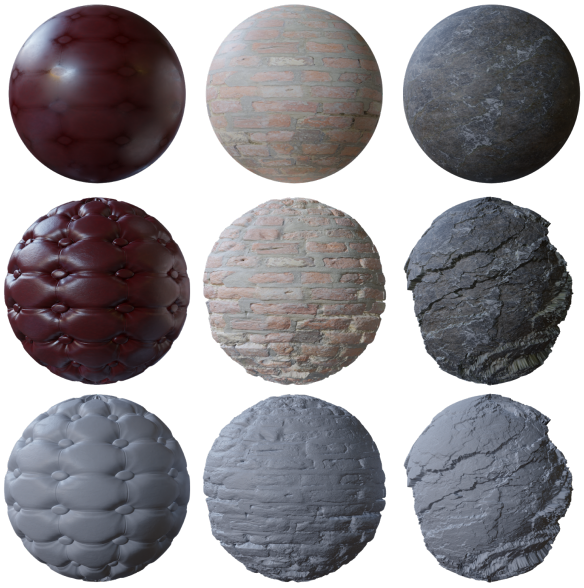
[Figure11] *Left: Left to right, bottom to top: Exponential, Power, Sinus, Circular; Right: blurred circle as displacement and color with circular easing.*

5 Workflow

Transferring the gathered information from the analysis into an actual implementation, that represents the reference-surface, can be overwhelming in the first place. This section will propose several guidelines to support a structured approach of recreation.

5.1 Height first

Besides color, height information has a strong influence on the surface characteristics. Differences in height influences the interaction of light by distorting reflection angles, the amount of received light, ambient occlusion, and self shadowing [JK].



[Figure12] *Top to bottom: Excluded height information, both together, excluded color information.*

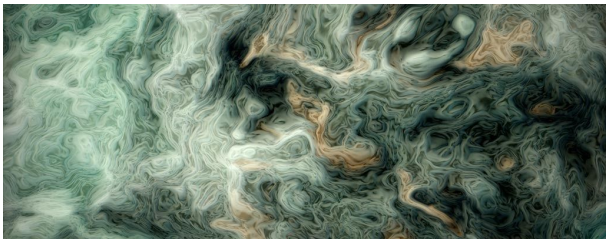
Another observation is that height information resembles structural features that are gathered from the analysis. The correlation is expressed through transitions between different materials or structural features. These are almost always represented in height differences. In addition through the physical meaning of height, other features like dust deposits and wearing can be derived to implement and distribute environmental influences in a convincing manner. Derived features can not only result in color, but also in roughness or ambient occlusion information, basically any information that may be required by the attached lighting model.

Because of this important meaning of height for materials, it is recommended to start the implementation with it. This will support the recognizability of materials and simplifies the implementation process. Height should be expressed as a single value within a range between zero and one, because many algorithms already result and expect parameters in this range. The representation is also visually easy to understand.

To recreate the height, the hierarchical order of the analysis should be considered. The proposed steps of analysis already return information ordered by their obtrusiveness. This leads to an approach where rough details will be implemented first, followed by more and more detailed features. Through this workflow is it possible to create derivations at any point of the height to control other information of the material, where a more detailed height information might be undesirable.

5.2 Distorting parameters

Shape and noise algorithms are the basis to reassemble surface structures, but these algorithms may be too uniform or too poor in details to replicate wanted surface structures. Structures in natural surfaces often have imperfections or turbulences because of factors like human failure or aging. While it is possible to implement deformations directly into shape and noise algorithms themselves, his approach is not recommended, because this will limit the generating algorithm so specific use cases. To break the perfection or lack of detail of noise and shape algorithms, manipulating their parameters is an efficient and reusable approach. The most interesting parameter to manipulate may be the UV. Manipulating the UV with a noise can mimic turbulence or irregularities. [Figure13] shows this technique, where the UV is manipulated multiple times.



[Figure13] *manipulated UV with noise for noise: "f(p) = fbm(p + fbm(p + fbm(p)))" [IQ01].*

The general concept of distorting parameters is to have a base value, which then is manipulated slightly to create deviations. This can be applied to any parameter that an algorithm offers, which could result in differences like color, position or orientation of similar elements. Using the RNG to create deviations besides noise is also a valid technique to create incoherent differences.

5.3 Seed

As mentioned earlier, the random number generator for procedural materials is based on hashing. Hashing needs an input to create pseudo random numbers. Noise utilizes hashing for creating random values which are interpolated with neighbor values [PF]. While this is a valid approach for the noise itself, procedural materials often have to utilize noise of different kinds, scale, rotation, offset and values. The risk of using the same noise over and over can lead to coincidences by layering or making decisions based on them. Coincidences like repetitive distribution can be immediately recognized by humans and will break the immersion. Another requirement within procedural materials is the ability to break continuous structures of noises, which is used to mimic separate surface elements that are made of the material, like floor planks where the planks are from the same type of wood but the planks are assembled in an arbitrary order.

To enable the different results for generative algorithms, which depends on the RNG, it is recommended to propagate a seed parameter. This parameter should be then used as a base, where every required random value is built upon. This can be easily achieved by either incrementing the initial seed value with every use and also by combining the seed parameter with internal seed values. Through this recommendation, the algorithm then depends on the seed parameter that concludes to endless distinctive, but still similar results. Further, this approach should not only be used in the algorithms themselves, creating a seed parameter for a whole procedural material and sub-materials allows to have endless variations with minimal effort.

5.4 Make more noise

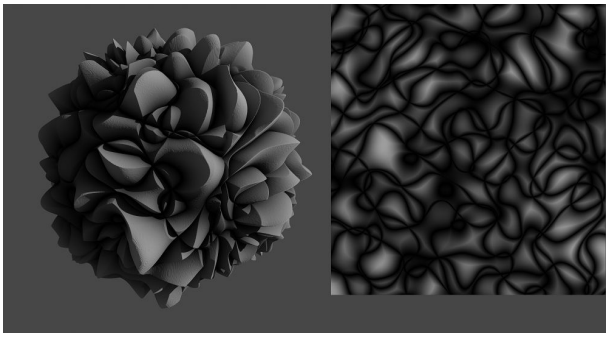
While the base noise functions are useful to create unpredictable patterns, often their appearance is not detailed enough for many purposes. The lack of details is usually caused by limited frequency in the noise. Lattice based noises will have a single frequency projected in their result [PF]. Layering base functions can overcome the lack of detail of noises or create other visually appealing results.

5.4.1 Fractal Brownian motion

Fractal Brownian motion (fBm) is an approach to combine multiple frequencies to create noise rich in details. It describes an additive layering of the same noise algorithm with different weights, scale and iterations. This can be done basically with any noise as shown in [Figure09] on the left side [BS] [SM]. This technique is commonly used to generate terrains, however the results of fBm are also interesting for procedural generation of surface information due to the generation of details in noise.

5.4.2 Noise by Noise

The base noise algorithms, even extended by fBm, may not cover all cases of required unpredictable patterns. To extend the toolbox of noises even more, noise can be combined in any way to create new complex noises for reassembling grunge or other surface features.



[Figure14] *Complex noise; Left: complex noise used for displacement and color; Right: generated complex noise from base noise functions.*

```
complex = 1.0
for i=0; i < 3; i++
    noise = noise_perlin(uv, seed++)
    noise = abs(noise * 2.0 - 1.0)
    complex = min(complex, noise)
complex = pow(complex, 3.0)
```

Pseudocode for the complex noise shown in [Figure14].

[Figure14] shows a complex noise which was generated by combining the results of base noise functions. This noise used a single algorithm as base, however there are no restrictions in any way for creating complex noise. Many render applications make use of this technique to provide their users a custom selection of patterns. These complex noises can amongst other things mimic surface features like scratches, cracks and grunge.

5.5 Imperfections

By looking at real world surfaces, they have one thing in common: They have all flaws in some way. These flaws may not be perceptible at first glance but they still exist. Imperfections in surfaces are irregularities which occur due to the interaction with its environment or during creation. Examples of imperfections are scratches, dents, discolorations, dust, fingerprints or inaccuracy. Nonetheless imperfections are very subtle features of surfaces [MS].

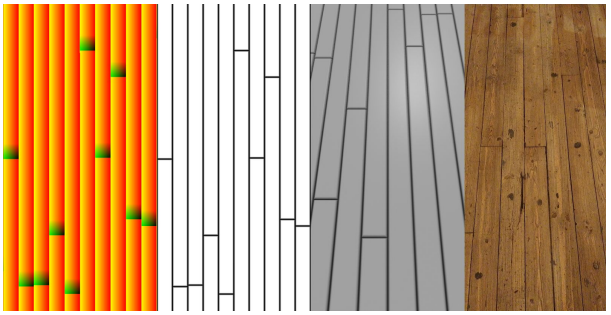
Imperfections are important when it comes to creating materials that should have photorealistic appearances. Materials that are too perfect in their composition will break the illusion of the surface, because they will break the continuity of material properties. Good examples can be found in bathroom tiles. Common imperfection could be the position, orientation and tilt of the individual tiles. By laying the tiles and filling the gaps with mortar they often show minimal deviations through the mortar. This can be caused either through inaccurate laying of the tiles and or also varying distributions of the mortar. Nonetheless differences should be very subtle and therefore considered as imperfection. Another illustrative example are fingerprints on smartphones. Looking at the switched off screen, these fingerprints may be not perceptible in various angles. But in certain angles, where the screen reflects light sources or bright surfaces in the environment to the viewer, they are going to be visible.

6 Applied surface recreation

To continue the example from the analysis and illustrate the proposed algorithm categorization and concepts of the workflow, the implementation of the wooden floor is described in detail. The objective of the recreation is to show the concepts and effects of algorithms, not to create a photorealistic representation. Nonetheless will the implementation provide a base for continuation to achieve such result by adding more and more details to the material in a hierarchical and iterative manner.

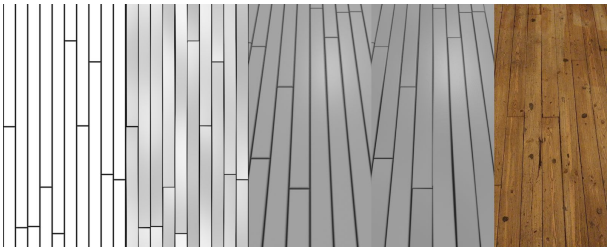
6.1 Planks

In the first iteration the most perceptible features of the floor are recreated. This includes the planks and the wood structure of them. To begin the recreation, the order of layers from the analysis is used.



[Figure15] *Left to right: Tilling & scaling, rectangle as height, render, reference photo.*

As noticed from the analysis, the planks are a repetitive geometric pattern of rectangles. Placing each plank individually is therefore impractical, because through this the material would be either limited in size or also in variation. Therefore the global UV was tiled to create endless equal sized UV's that match the shape of the planks in the reference photo. The used tiling algorithm takes a continuous UV as input and returns a fractured UV, and in addition a two dimensional ID for each tile, which is an important information, that is utilized later by other requirements. In addition to the tilling, every column of planks is randomly shifted to mimic the irregular shift in the photo reference. The tiled UV is then used to draw rectangles to resemble the base heightmap. The rectangle which was drawn is seen as the second item in [Figure15] and makes use of a small blur with easing, see [Figure11], to mimic the fall off of the planks at the border.



[Figure16] *Left to right: variation in distortion and tilt, render comparison, reference photo.*

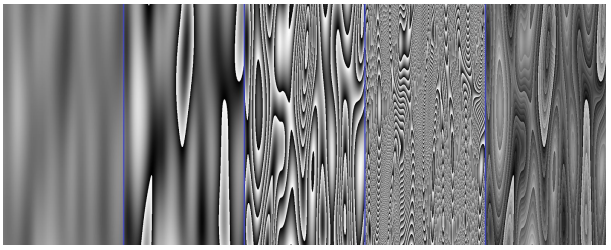
The first result of the material shows similarities to the reference photo. Besides the lack of detail in many ways, the current layout of the planks is too perfect. The heightmap describes them as perfectly perpendicular to each other, the edges are perfect straight and every plank is in level and has the same height. This is where environmental influences and imperfections take place. As wood planks age, in this case drying, they often bend themselves. This can be seen in the reference photo where the distance between the planks varies. To mimic the variations, the UV for the planks is modified which concludes to a change of the heightmap, which resembles the bend in the XY axes. The values of the heightmap are then modified to add the bend information in height. Both modifications use a large scaled perlin noise. [Figure16] shows these changes which are subtle, nonetheless improving the plausibility of the heightmap.

6.2 Wood material

The creation of the wood material is split in two parts, recording to the analysis and showed in [Figure05].

6.2.1 Rings

There are several ways to recreate a wood ring pattern. One way to mimic the pattern accurately would be to recreate the whole tree trunk with his branches as a three dimensional distance field. A taken cross-section simulates then the saw cut of the log. While recreating material structures as accurate distance fields can work out, the computation might be expensive and complex to implement. Often it is easier to fake structures to create lookalikes, which will be convincing enough to trick the viewer. The applied process to fake the structure is one of many approaches that can work.

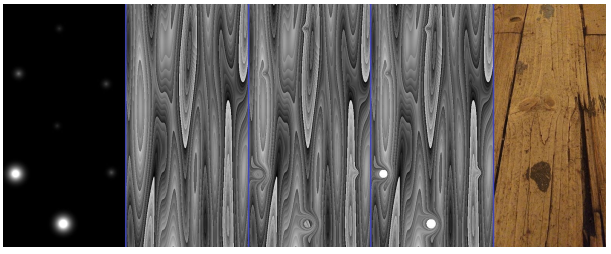


[Figure17] *Left to right: Scaled noise, first fracture, second fracture, third fracture, combined as fractal Brownian motion.*

The fake of the structure, see [Figure17], starts by stretching a perlin noise in one direction to mimic the run direction of the wood trunk. The resulting noise values are then fractured in three different gradations and finally combined as fBm, which reassembles the changing colors of wood rings due to different climatic conditions and cycles of summer and winter. An alternative technique to achieve similar patterns is to bring turbulence into the UV before passing it to the noise algorithm. Turbulence can be added through relative rotations in the UV, controlled by another noise, shown in [Figure12]. The applied technique however utilizes only a single noise.

6.2.2 Branches

The other important recognizable feature of wood are branches. The shape of branches are often circular, because of that, random placed circles are a good start to mimic them. While the applied approach will utilize perfect circles, these circles can be manipulated in further iterations to gain more realism by achieving elliptical and irregular shapes.

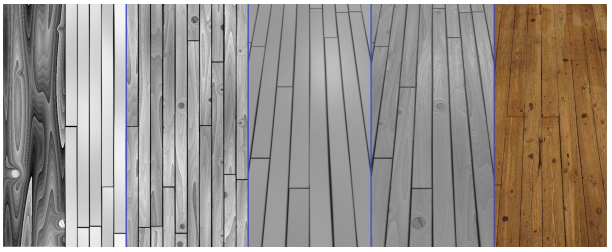


[Figure18] *Left to right: random circles in size and position, wood ring structure, modified structure, modified structure with branches.*

To create random placed circles in an endless manner, tilling the underlying UV is necessary. In each created sub-UV is a circle placed with random position and size. Nonetheless this technique can still appear retentive, which will reveal the invisible grid. To conceal that, only a percentage of the circles are drawn, which is determined by a random value generated through the tile id.

The other part of recreating branches in wood materials is the integration. As mentioned in the analysis, the branches are influencing the layout of the wood rings, where the branches push the rings away from them. To trick the viewer again, the UV nearby branches are rotated. Because of that the circles, which represent the branches, are blurred. The blur is then used as rotation information. Finally the center of the circles and the wood structure are merged together by adding the values. The created heightmap will be the base for further information like albedo or roughness.

6.3 Composition



[Figure19] *Left to right: prepared wood material and planks, merged planks with wood, previous render, render with merged wood material, reference photo*

The composition of the planks and the wood material is applied in two steps. First the material has to utilize the plank UV's, then the heightmap is merged with the height of the planks, see [Figure19].

The global UV has been tiled beforehand to recreate the planks of the floor. Either the global UV or the plank UV can be used as input for the wood material. Which UV is used will depend on the related effects. While the global UV is in the example untouched, the plank UV is modified to mimic the bending of the planks. Therefore the plank UV is used to transfer the modifications into the wood material. The problem of using the plank UV's is that the wood material has for every plank the same appearance, because the UV are all uniform. But through the concept of seed propagation this can be solved by utilizing the id of the planks as seed for the wood material. This will cause for each plank different random values for the underlying algorithms.

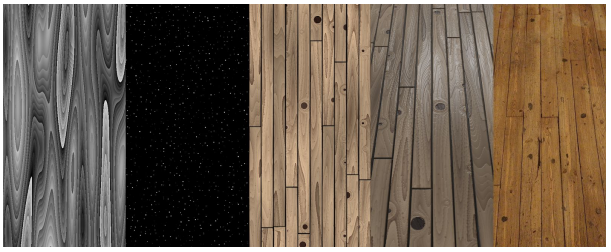
To merge the information of plank and wood heights, the applied approach is achieved by scaling the wood information and subtracting it from the plank height. This provides full control over the intensity of the wood structure through the scaling that is represented in the planks. Finally the values of the merged heightmap have to be clamped, because the subtracting will cause values below zero between the planks. The approach ensures in addition the linearity of the plank and wood height information, which would not be given by a multiplication. Nonetheless, the merge algorithm depends on the required result, where multiplication might be preferred on other materials or heights.

6.4 Height derivations

The amount of properties and the way of derivation from the heightmap depends on the lighting model. The chosen lighting model for the example is based on Disney's paper for physically based shading [BB]. The procedural material will feed the color and roughness parameter besides the height displacement.

6.4.1 Color

The color of the floor is reassembled in three steps: First, by the wood itself and then by the different planks and finally through the environment.



[Figure20] *Left to right: wood structure, burned cigarette spots, generated surface color, render, reference photo*

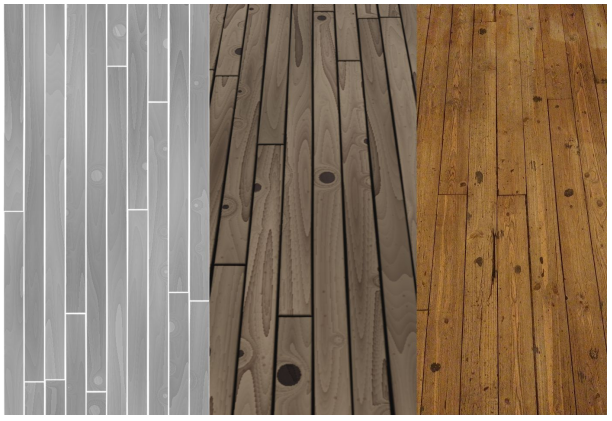
Colorizing the wood material is quite straight forward. The generated height is used as an interpolation between two brownish colors, which represent the darkest and brightest color of the wood. To refine the distribution between bright and dark colors, the heightmap was eased to show more bright than dark values.

The color information is then utilized by the floor layout, where the existing heightmap of the floor is simply multiplied to the color. This is possible due to specific coincidences. First, the gaps between the planks are filled with deep black dust, which will match the color of the heightmap on this position where the lowest points of the material are represented by zero. Then the second coincidence is the height variation of the planks which mimic the bend. Every plank will have a small color variation to other ones, because not every plank is manufactured from the same trunk and position. By multiplying the height information with albedo, the color variation for each plank and the dust is achieved.

Finally the cigarette burns are added to the surface color. The spots are reassembled, like the branches in the wood structure, as circles with random position, size and strength on a tiled UV. This mask is then used to blend between a reddish dark brown, to mimic nearly burned wood, and the existing albedo.

6.4.2 Roughness

The roughness parameter of the lighting model controls the diffusion of light. In general the floor from the reference photo is a very rough surface mainly because of wearing through dirty shoes. The shoes also cause deposits of dirt and brushing the surface of the floor.



[Figure21] *Left to right: Roughness map, render, reference photo*

To simulate the different levels of roughness in the example surface, the heightmap as it is will be utilized, because dirt deposits are located in lower places of the surface. Therefore the height is inverted and remapped to cover a small range in the higher level of roughness.

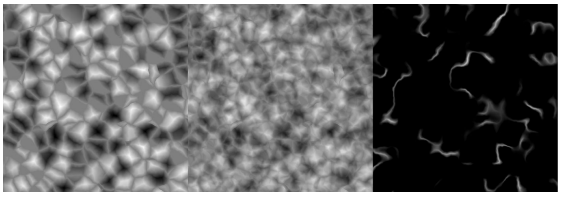
7 More materials

There have been other procedural materials created through this thesis, which use different algorithms and are located in other environments. Nonetheless, these materials were created with the proposed techniques of analysis and workflow to show their versatility and capability.

7.1 Cliff

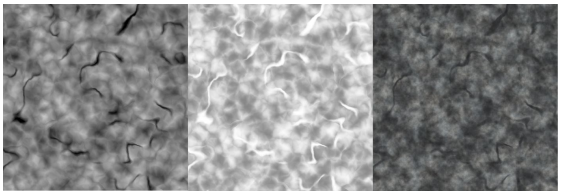
[Figure22] *Closeup photo of a rock wall, used as reference*

[Figure22] shows the idea for the procedural material, which has been created to mimic rock walls.



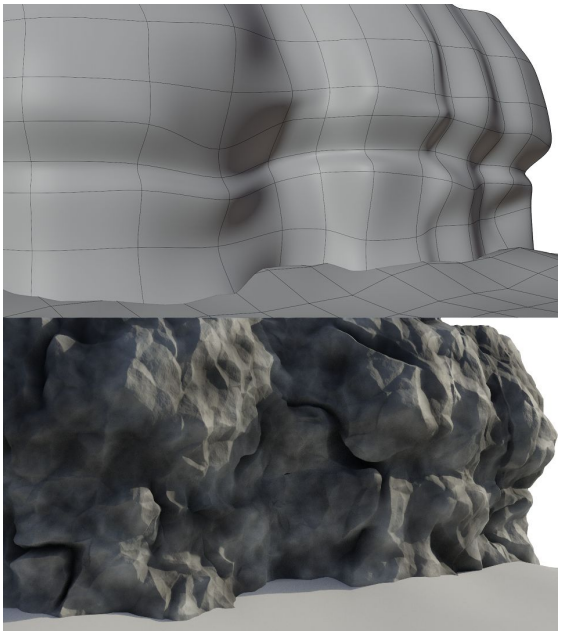
[Figure23] *Left to right: Specialized voronoi, fBm of specialized voronoi, crack noise*

The most characterizing feature of the rock wall reference is the chaotic and still geometric height information. This has been simulated by creating a derivation of an already specialized voronoi noise, where the distance is measured to the edge instead to the centre [IQ05]. The custom derivation will flip the distance into negative by random to create dents.



[Figure24] *Left to right: Height, roughness, color*

[Figure24] shows besides the height information for the surface the other needed information for the used lighting model (Disney's BRDF [BB]), which were derived from the height information. The material was applied to a rough modeled cliff to show the resulting changes in the geometry based on the generated height information.

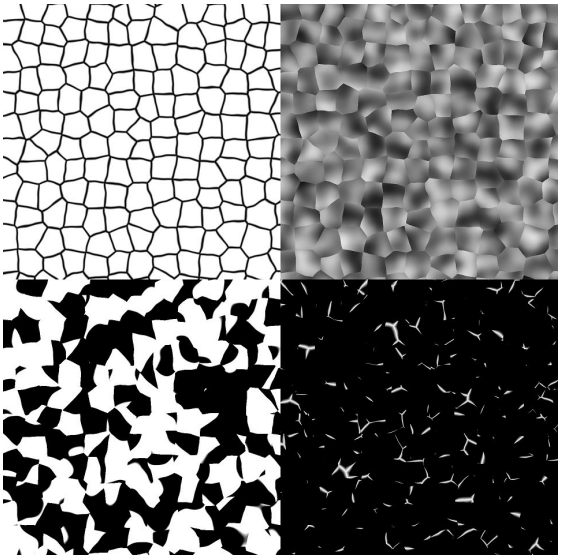


[Figure25] *Top: Base mesh of the cliff without material; Bottom: Base mesh with the procedural material;*

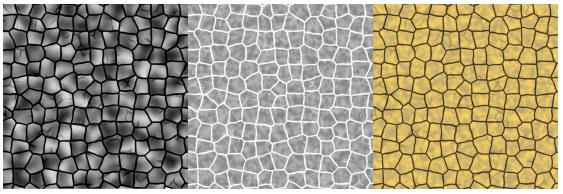
7.2 Stylized ground

[Figure26] *Closeup photo of a dry ground.*

Dried ground can create interesting patterns, which are very similar to voronoi noise. The voronoi noise from Ingo [IQ05] the base of this material, which defines the individual tiles of the ground. Tilt and elevation changes are simulated perlin noise. The cracks are generated with a combination of another value noise and voronoi noise.

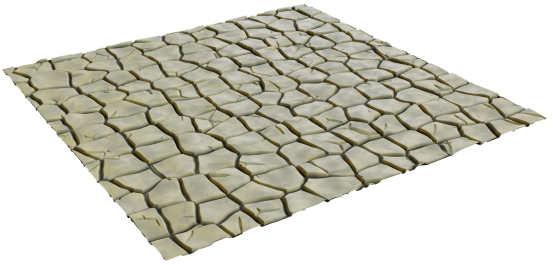


[Figure27] *Top to bottom, left to right: Base height, tilt information, elevation variations, cracks.*



[Figure28] *Left to right: Height, roughness, color.*

[Figure28] shows the derivations from the information that are given through [Figure27]. [Figure29] shows the finished material rendered on a quad with tesselation for the height information.



[Figure29] *Render of the procedural material that mimics dry ground.*

8 Conclusion

Using complex materials entirely created in fragment shaders might not be useful in real time applications, where every frame per second matters. As shown in the applied recreation, procedural materials may be built upon many algorithms which have a negative impact to the performance. The approach of procedural material within fragment shaders might work for offline render applications, it would be interesting to test how much impact a procedural material will have to the render time besides ordinary textures.

8.1 Analyzing surfaces

The analysis of surfaces pointed out to be an essential part of creating procedural materials. The applied recreation of surfaces showed that the natural layering and composition of a surface can be transferred into the implementation and extracted layers from the analysis are corresponding to created UV layouts and general height information, as shown in the example of the pub floor. The splitted analysis of layers and surface materials can be also used to implement surface materials separately to make them reusable for other procedural materials. The wood material for the pub floor could be used after the implementation in further materials, like a furniture material. This will save time and effort by creating such materials. By thinking about environment influences, interesting and characterizing details are added to the material. With these influences, materials are becoming convincing for the viewer. In addition, many environmental influences can be transferred easily to other materials, without the need of reference photos or existing materials. Therefore, some of them can be also prepared as separate material, like cracks or grease, for reuse.

The proposed approach of the analysis showed also to have no dependencies to programming languages or applications. And the restriction for shader programming did not influence the analysis either. Therefore it can be assumed that the approach of the analysis, with its three steps and hierarchical method, can be applied not only to implicit procedural surface creation. The approach may also work for explicit environments and applications like Substance Designer [SD02].

8.2 Algorithm categorization

The defined categorization showed to be useful and helpful while creating procedural materials. On the one hand helped the categorization looking for new algorithms to achieve new and distinct results besides the results from existing algorithms, because through the name and appointed task of the categories. On the other hand showed the categorization to be useful by exploring the results of new combinations of algorithms. Due to the sorting of the algorithms is the structure of parameters similar. This made it easy to exchange specific algorithms with others of the same kind to explore new results.

While the analysis showed to be independent from the implementation, the categorization is influenced by the restrictions of shaders. Due to the limitation that only implicit algorithms are suited for shaders, all generative algorithms like noise and shapes depending on UV coordinates. These UV coordinates may be not given in other environments or applications, again like Substance Designer. Therefore the defined categorization might only work for fragment shader implementations.

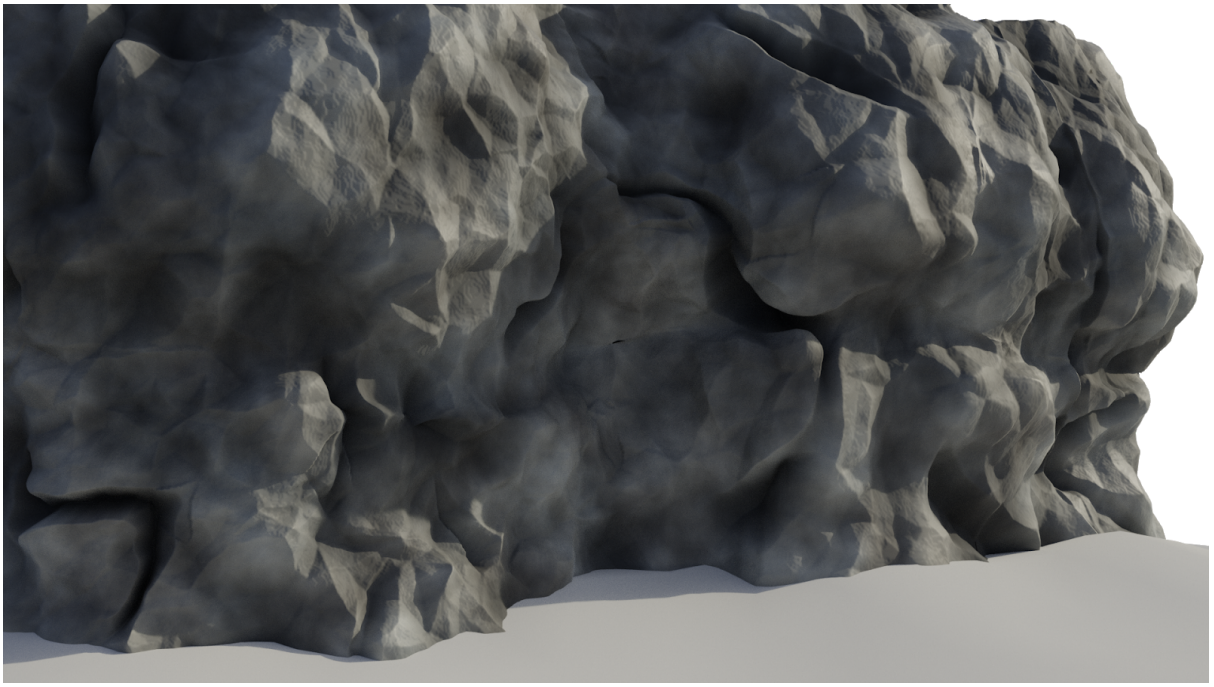
Nonetheless showed the categorization that through sorting algorithms and implementing them abstract, without specializations, they can be used like Lego bricks. It also abstracts the complexity of the algorithms which makes them usable for a wider range of users and separates creativity from technology.

8.3 Workflow

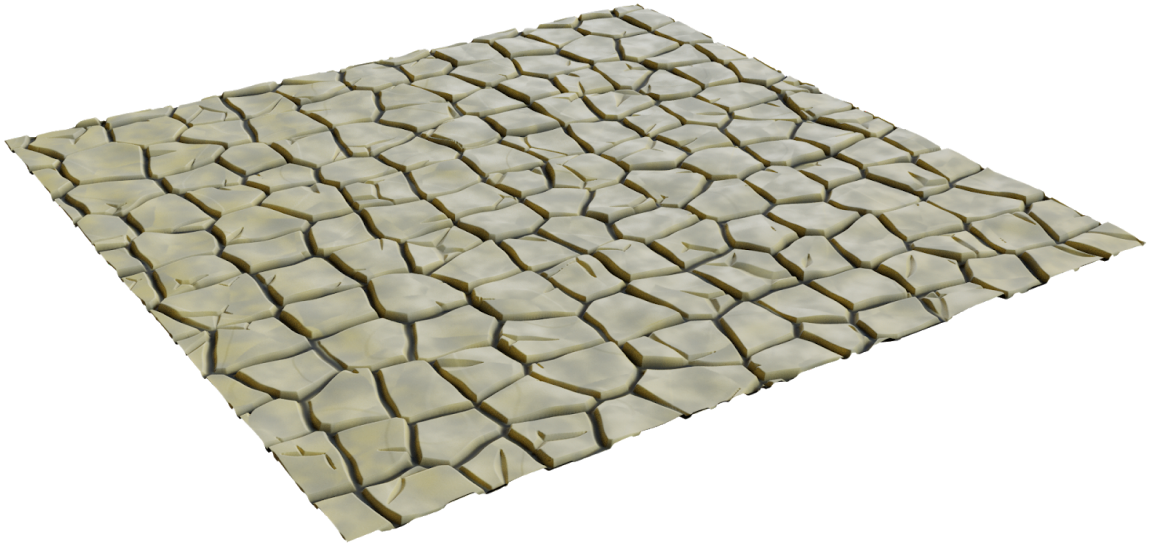
The proposed workflow guidelines showed to be useful and ensured a structured procedure. The workflow matched also with the hierarchical approach of the analysis, therefore iteration to reach the desired level of detail is built in and is easy to integrate. Similar to the analysis showed the guidelines of the workflow no dependencies to an environment or application, which they may apply to them.

While creating materials, the workflow guidelines showed yet to be vague. And Try-And-Error phases taking still a big portion of the creation process. Foreknowledge of combinations of algorithms cannot be eliminated by the named guidelines. Nonetheless they will help to guide experiments to reach desired results faster.

9 Appendix

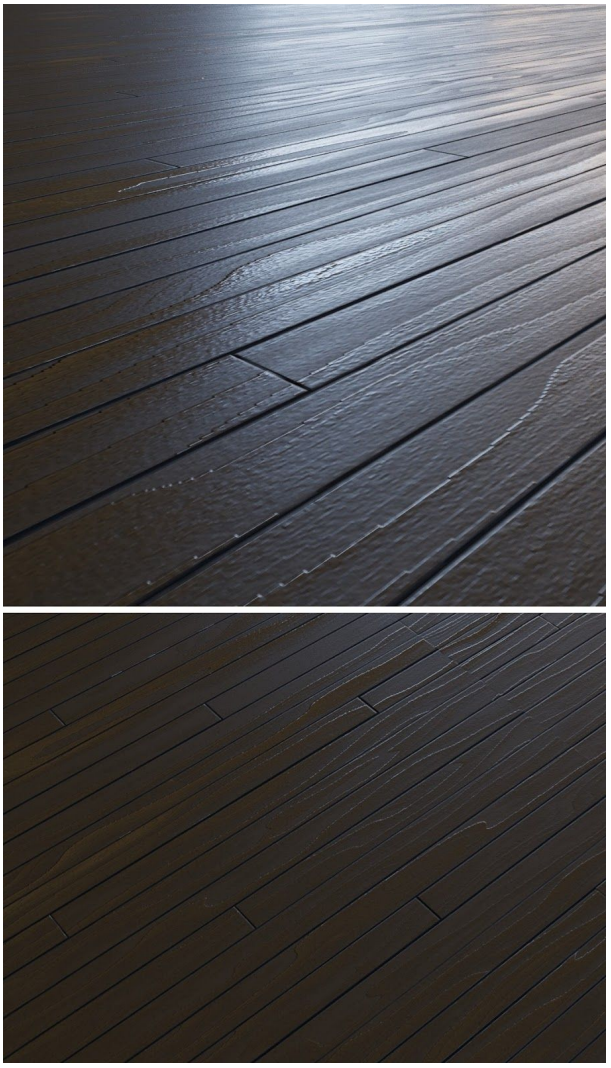


[Figure30] *Render of [Figure25].*



[Figure31] *Render of [Figure29].*

[Figure32] *Render of [Figure20].*



[Figure33] *Alternative wood floor material.*

[Figure34] *Render of a procedural granit material. The statues utilising only the color information to represent a polished surface. The material also offers color parameters to create variations.*

[Figure35] *Render of a procedural paving stone material.*

10 References

[PF]: Lagae, Ares, et al. *"A survey of procedural noise functions."* Computer Graphics Forum. Vol. 29. No. 8. Oxford, UK: Blackwell Publishing Ltd, 2010.

[KP]: Perlin, Ken. *"An image synthesizer."* ACM Siggraph Computer Graphics 19.3 (1985): 287-296.

[SW]: Worley, Steven. *"A cellular texture basis function."* Proceedings of the 23rd annual conference on Computer graphics and interactive techniques. 1996.

[SG]: Gustavson, Stefan. *"Procedural textures in GLSL."* (2012): 105-119.

[BB]: Burley, Brent, and Walt Disney Animation Studios. *"Physically-based shading at disney."* ACM SIGGRAPH. Vol. 2012. 2012.

[PA]: Ebert, David S., et al. *Texturing & modeling: a procedural approach*. Morgan Kaufmann, 2003.

[DS]: Deguy, Sébastien. *"The New Age of Procedural Texturing."* (2019).

[WN]: Tzeng, Stanley, and Li-Yi Wei. *"Parallel white noise generation on a GPU via cryptographic hash."* Proceedings of the 2008 symposium on Interactive 3D graphics and games. 2008.

[WJ]: Jakob, Wenzel, et al. *"A comprehensive framework for rendering layered materials."* ACM Transactions on Graphics (ToG) 33.4 (2014): 1-14.

[SM]: Fournier, Alain, Don Fussell, and Loren Carpenter. *"Computer rendering of stochastic models."* Communications of the ACM 25.6 (1982): 371-384.

[MS]: Schwärzler, Michael. *"Rendering imperfections: Dust, scratches, aging,.."* Institute of Computer Graphics and Algorithms, Vienna University of Technology, Tech. Rep. TR-186-2-07 9 (2007).

[HP]: Heckbert, Paul S. *"Survey of texture mapping."* IEEE computer graphics and applications 6.11 (1986): 56-67.

[JK]: Kajiya, James T. *"The rendering equation."* Proceedings of the 13th annual conference on Computer graphics and interactive techniques. 1986.

[RC]: Cook, Robert L. *"Shade trees."* Proceedings of the 11th annual conference on Computer graphics and interactive techniques. 1984.

[RP]: Penner, Robert. *"Motion, tweening, and easing."* Programming Macromedia Flash MX (2002): 191-240.

[BL]: Blender Foundation. *Blender Shader Nodes.* 24 July 2020: https://docs.blender.org/manual/en/latest/render/shader_nodes/index.html

[UN]: Unity Technologies. *Unity ShaderGraph*. 24 July 2020: https://unity.com/de/shader-graph

[UR]: Epic Games. *Unreal Material Editor.* 24 July 2020: https://docs.unrealengine.com/en-US/Engine/Rendering/Materials/Editor/index.html

[CN]: Maxon. *Node-Based Materials.* 24 July 2020: https://www.maxon.net/de/produkte/cinema-4d/features/texturing/node-based-materials/

[IQ01]: Inigo Quilez. *domain warping.* 24 July 2020: https://www.iquilezles.org/www/articles/warp/warp.htm

[IQ02]: Inigo Quilez. *2D distance functions.* 24 July 2020: https://www.iquilezles.org/www/articles/distfunctions2d/distfunctions2d.htm

[IQ03]: Inigo Quilez. *distance functions.* 24 July 2020: https://iquilezles.org/www/articles/distfunctions/distfunctions.htm

[IQ04]: Inigo Quilez. *Palettes.* 24 July 2020: https://iquilezles.org/www/articles/palettes/palettes.htm

[SD01]: Allegorithmic, Adobe Inc. *Substance Designer.* 24 July 2020: https://academy.substance3d.com/courses/Creating-your-first-Substance-material/

[SD02]: Allegorithmic, Adobe Inc. *Substance Designer.* 24 July 2020: https://www.substance3d.com/products/substance-designer/

[BS]: The Book of Shaders by Patricio Gonzalez Vivo & Jen Lowe. F*ractal Brownian Motion.* 24 July 2020: https://thebookofshaders.com/13/

[TS]: Simon Thommes. *My Nodevember 2019.* 24 July 2020: https://pbs.twimg.com/media/EL857feW4AAiqYr.jpg