Studiengang Informatik

Fakultät Elektrotechnik und Informatik

Bachelor Thesis

to the subject

Analysis of procedural Materials

within Fragment-Shader

Eric Dolch

Malerstraße 21

88250 Weingarten

Adviser:

Prof. Dr. Daniel Scherzer - Hochschule Ravensburg-Weingarten

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) (Ciy, Date)

[**1 Introduction**](#_7rpj8tt8dtge) **3**

[1.1 Motivation](#_qjc0q3di0onm) 4

[1.2 Objectives](#_png0adqqbp1z) 4

[**2 Prerequisites**](#_wwigdn3qc5wd) **4**

[2.1 Procedural](#_kyfjtte2q00z) 4

[2.2 Texture](#_fxbn9cia2i0b) 4

[2.3 Material](#_ff7gpcgpd2th) 4

[Implicit and explicit algorithms](#_6w0yiy2j1usw) 5

[Pro & contra of procedural patterns within fragment shaders](#_v1sf2ph8d0qd) 5

[**3 References**](#_gdwagcvfltqx) **5**

[**4 Figures**](#_ddw7y5oef95i) **5**

# 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 [(P01)] or Worley-Noise [(W01)] are still present today and essential for procedural generations, due to their natural looking appearance which suits replicating natural surface properties.



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[...]" [(G01)], because "[...] modern shader-capable GPUs are mature enough to render procedural patterns at fully interactive speeds [...]" [(G01)]. Which enables essential algorithms types 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" [(BLE01)], "Unity 3D" [(UNI01)], "Unreal Engine" [(UNR01)] or "Cinema 4D" [(CIN01)] offering interfaces to attached renderers to handle the shading of objects in a modular manner, as proposed in the paper "shade trees" [(C01)]. Besides enabling modifications to the shading of the mesh and creative abstract lightning models like toon shading, it enables at the same time generating 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 ray tracing the scene instead of rasterizing, require also 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.

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



## 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 they have 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

By 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." [(LLC01)]

## 2.2 Texture

Textures are images where a single or more 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[...]" [(D01)]. Render applications are then using these textures to feed the properties of an assigned lightning model.

## 2.3 Material

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



As seen in the figure, 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 lightning 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.

## 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. [(EMP01)] Due to the architecture of shaders, regardless of weather used by ray tracing or rasterization, the task of shaders is to evaluate arbitrary points of a surface. These points are defined either in rasterization by the current resolution and view, in case of ray tracing are the points determined by the hit location 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.

## Pro & contra 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” [(EMP01)] 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 References

# 4 Figures