

Computergraphik II

3. Texturen II

Carsten Rudolph

Graphische Datenverarbeitung und Visualisierung
Fakultät für Informatik



TECHNISCHE UNIVERSITÄT
CHEMNITZ

Sommersemester 2019

3.1 Texturerzeugung

Arten der Texturerzeugung

Texturierung dient der Simulation von Oberflächendetails. Bei der Erzeugung von Texturen sollen diese Oberflächendetails **beschrieben** und ggf. aus der echten Welt **eingefangen** werden.

Unterscheidung zwischen zwei Hauptansätzen:

- Algorithmisch/Prozedural: durch formalen Algorithmus errechnet
- Artistisch: Kreatives Werk eines Menschen

In der Praxis meist Kombination beider Ansätze

3.2 Prozedurale Texturen

Formeller Texturbegriff

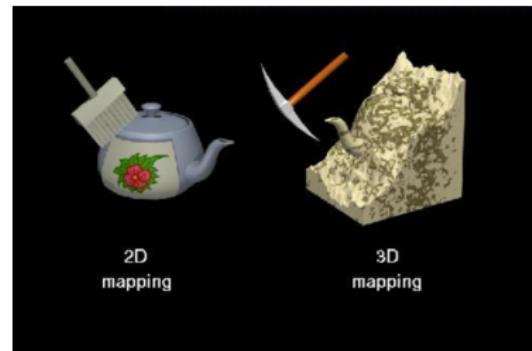
Prozedurale Texturen sind solche, die durch Lösen einer **Texturfunktion** erzeugt werden.

- Vorteil: sehr effizient, benötigt wenig Speicher, schnelle Berechnung
- Nachteil: Erzeugen der Texturfunktion sehr aufwändig und meist nicht intuitiv.

Dimensionalität

Texturfunktionen können in unterschiedlichen Dimensionalitäten erzeugt werden.

- 2D: Textur für Texture Mapping, analog zu "Tapete"
- 3D: Volumentextur, Lösung für Punkt in Raum



Beispiel: Volumentexturen

Volumentexturen können als räumliche Funktion interpretiert werden, die jeden Punkt $p \in \mathbb{R}^3$ eine Farbe $c \in \mathbb{R}^3$ zuweist.

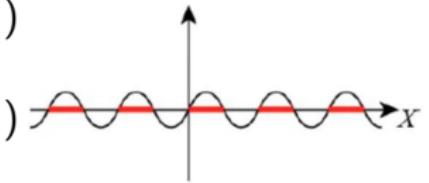
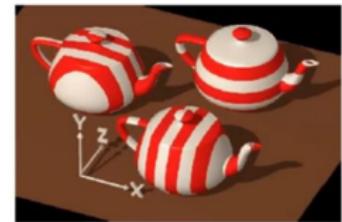
Anwendung z.B. im Pixel/Fragment-Shader:

- Koordinate auf Oberfläche (im Objektraum): $p = (x_p, y_p, z_p)$
- Lösen der Texturfunktion $c_p = T(p)$

Beispiel: Volumentexturen

Einfache Texturfunktion: Streifen entlang einer Achse (z.B. x-Achse).

```
in vec4 pos;  
out vec4 color;  
  
void main() {  
    if (sin(PI * pos.x) > 0)  
        color = vec4(1, 1, 1, 1)  
    else  
        color = vec4(1, 0, 0, 1)  
}
```



Beispiel: Volumentexturen

Erweiterung noch relativ simpel, z.B.:

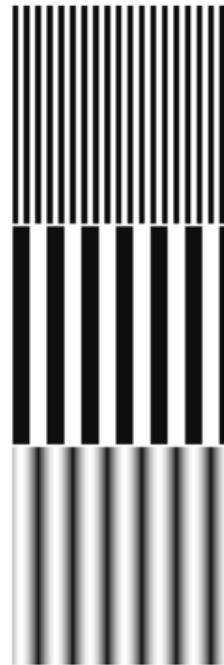
- Variation der Streifenbreite:

$$\sin(\pi \cdot x)$$

- Interpolieren zwischen verschiedenen Farben:

$$t = \frac{1 + \sin(\pi \cdot x)}{2}, c = (1 - t) \cdot c_1 + t \cdot c_2$$

Aber: mit jeder Erweiterung wird Funktion immer komplexer.



Generieren Prozeduraler Texturen

Wie kann eine prozedurale Textur verwendet werden, um Oberflächen aus der echten Welt zu simulieren?

Analyse und Modellierung des zu simulierenden Materials:

- Bestimmung der Grundeigenschaften (z.B. Farbe, Musterung, ...)
- Erkennen von Regelmäßigkeiten (Regularität, Symmetrie, ...)
- Statistische Analyse von Unregelmäßigkeiten

Generieren Prozeduraler Texturen

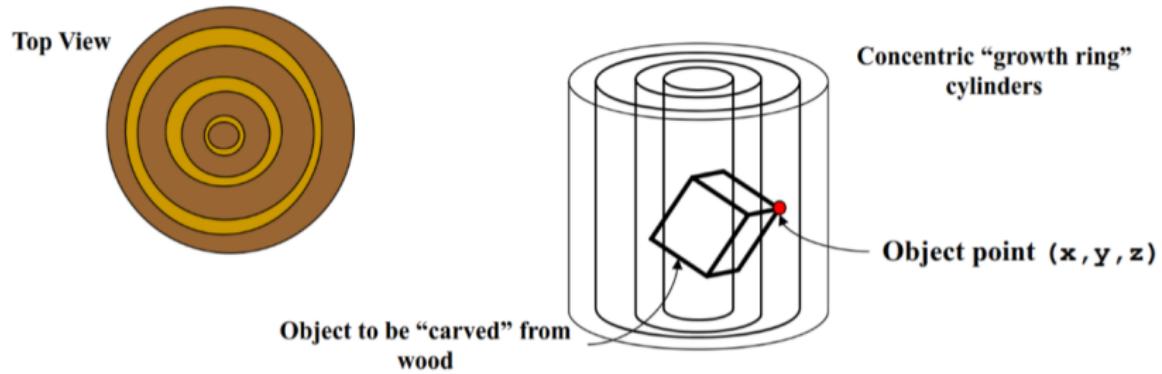
Modellierung der Funktion dabei häufig aus Kombination mehrerer einfacher Basisfunktionen:

- Polynome, trigonometrische Funktionen
- Modulo-, Treppen-, Sprungfunktionen, ...

Zur Vermeidung von Periodizitäten (Wiederholungen von einzelnen Pattern) können (pseudo-)zufällige Faktoren und Rauschverfahren genutzt werden.

Beispiel: Holztextur

Die **Maserung** von Holz wird hauptsächlich durch die Jahresringe bestimmt. Beim “schnitzen” eines Objektes, bleibt Maserung entsprechend erhalten.

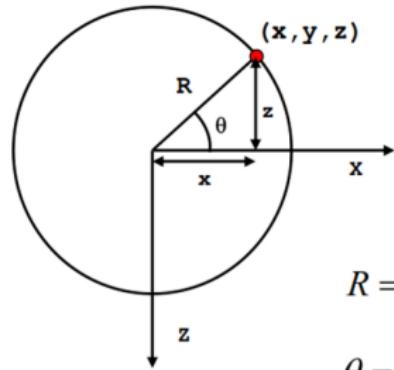


Gesucht: Funktion $f(x, y, z)$, die für einen Punkt $p = (x_p, y_p, z_p)$ die entsprechende Farbe errechnet.

Modellieren von Jahresringen

Die Jahresringe können als Menge konzentrischer Zylinder modelliert werden. Liegt ein Punkt nahe einer Zylinderoberfläche, wird er dunkler gefärbt.

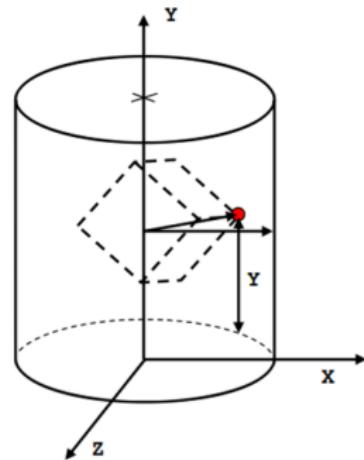
Top View



$$R = \sqrt{x^2 + z^2}$$

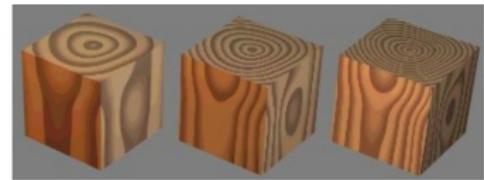
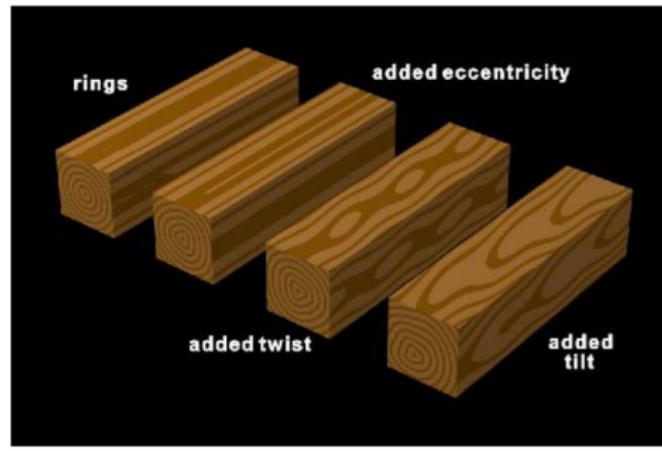
$$\theta = \tan^{-1}\left(\frac{z}{x}\right)$$

$$H = y$$



Modellieren von Unregelmäßigkeiten

Bäume wachsen nicht gleichmäßig von innen nach außen, deshalb Einführen weiterer Parameter, z.B.: Exzentrizität, Verdrehung, Neigung, usw.



Variations in ring density



Variations in Ring Width

Beispiel: Marmortextur

Viele Gesteine besitzen eine **Marmorierung**. Marmor selbst ist ein metamorphes Gestein, welches durch Umwandlung von Kalkstein und Karbonatgestein entsteht. Mineralische Verunreinigungen werden bei der Metamorphose chaotisch verteilt.



Erzeugen von Marmorierungen

Marmorierung folgen einer zufälligen Verteilung, welche für verschiedene Marmorarten unterschiedlich aussieht.

Zur Modellierung wird Algorithmus benötigt, der

- Zufälliges Rauschen erzeugt und
- Dennoch Kontrolle über Ausgabe erlaubt

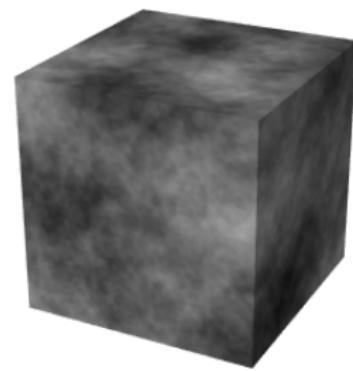
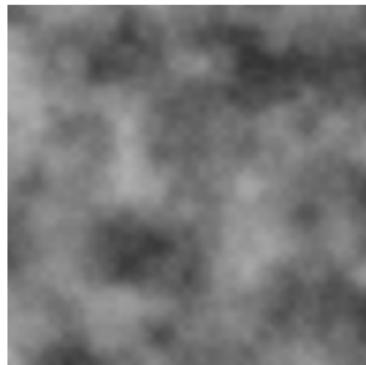
Ken Perlin 1985 [2]: "Perlin Noise"¹

¹Perlin wurde für die Entwicklung des Algorithmus 1997 mit Oscar ausgezeichnet

Perlin Noise

Perlin Noise basiert auf folgenden Ideen:

- 1 Initialisiere ein 1-dimensionales Array mit zufälligen Einheitsvektoren (Perlin verwendet 256 Vektoren)
- 2 Eine Hash-Funktion erzeugt einen Zufallswert für jeden Punkt p , der als Index für das Array dient
- 3 Skalarprodukt aus Zufallsvektor und \vec{p} erzeugt einen skalaren Wert am Punkt p
- 4 Interpolieren der Werte vermeidet visuelle Artefakte



Perlin Noise

Zufälliges Generieren von Einheitsvektoren \vec{v} zur Initialisierung:

$$\begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} = \begin{bmatrix} 2 \cdot \xi - 1 \\ 2 \cdot \xi' - 1 \\ 2 \cdot \xi'' - 1 \end{bmatrix}; \xi, \xi', \xi'' \in [0, 1]$$

Der Vektor \vec{v} hat eine zufällige Länge, deshalb gilt:

- $|\vec{v}| \leq 0$: Normalisiere Vektor und speichern in Array
- $|\vec{v}| > 0$: Verwerfen und neu generieren

Perlin Noise

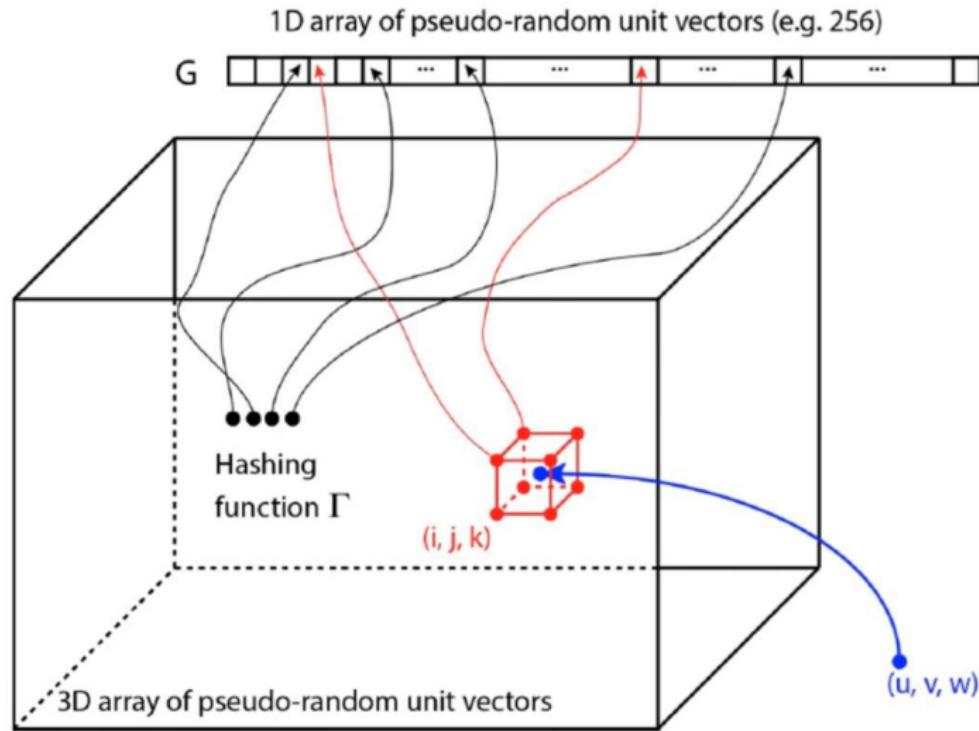
Ziel ist das Auswählen eines Zufallsvektors für einen Punkt $p \in \mathbb{Z}^3$.

$$\begin{aligned}\Gamma_{ijk} &= \Phi(i + \Phi(j + \Phi(k))) \\ \Phi(i) &= i \bmod 256\end{aligned}$$

Die Rauschfunktion $n(x, y, z)$ berechnet sich damit aus folgender Vorschrift:

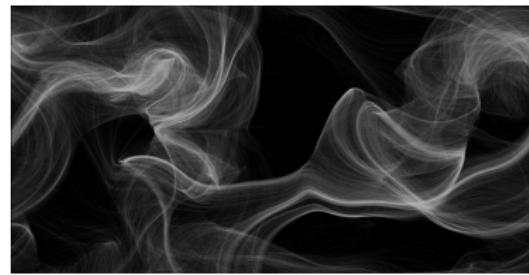
$$\begin{aligned}n(x, y, z) &= \sum_{i=|x|}^{|x|+1} \sum_{j=|y|}^{|y|+1} \sum_{k=|z|}^{|z|+1} \Omega_{ijk}(x - i, y - j, z - k) \\ \Omega_{ijk}(u, v, w) &= \omega(u)\omega(v)\omega(w)(\Gamma_{ijk} \cdot (u, v, w)) \\ \omega(t) &= \begin{cases} |t| \leq 1 & 2|t|^3 - 3|t|^2 + 1 \\ |t| > 1 & 0 \end{cases}\end{aligned}$$

Perlin Noise



Perlin Noise Beispiele

Perlin Noise eignet sich überraschend gut zur Erzeugung einer Vielzahl natürlicher Texturen, z.B.: Wolken, Rauch, Feuer, Holz, Marmor, usw.



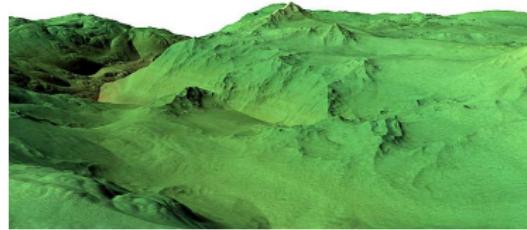
<https://davenewt.github.io/perlin-noise-flow-field/>



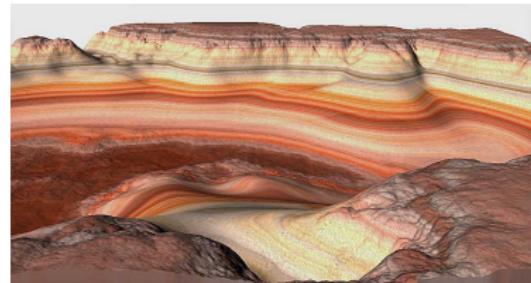
<https://peterkutz.com/>

Perlin Noise Beispiele

Je nach Interpretation des Rauschsignals ergibt sich noch eine Vielzahl weiterer Anwendungen. Als Höhenfeld genutzt, können so z.B. Terrains erzeugt werden.

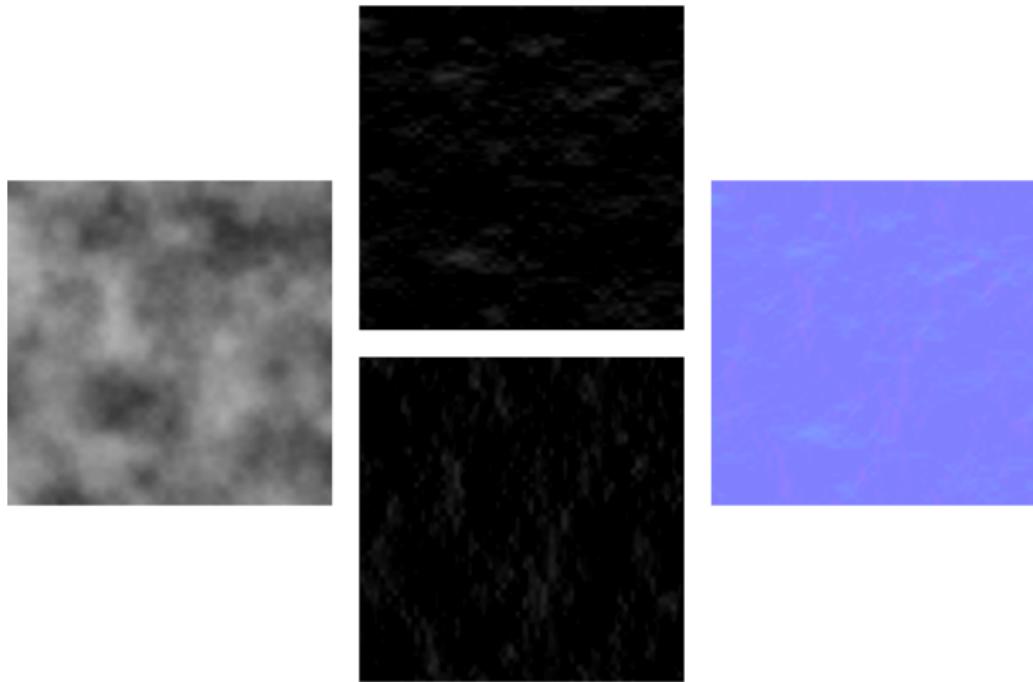


<https://www.world-machine.com>



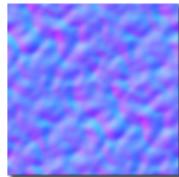
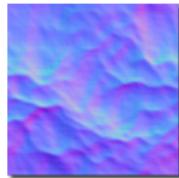
Perlin Noise Beispiele

Aus einer Höhentextur lässt sich eine Normal-Map generieren, indem sie mit einem Sobel-Operator gefiltert wird.



Perlin Noise Beispiele

Kombiniert man mehrere dieser Normal Maps verschiedener Skalierung, lassen sich z.B. Wasserbewegungen simulieren. Durch Gegeneinanderverschieben können diese einfach animiert werden.



3.3 Texture Scans

Texture Scans

Problem prozeduraler Texturen: um Komplexität der Realität abzubilden, wird ebenfalls komplexe Funktion benötigt.

Alternativer Ansatz: Erzeugen einer Textur auf Grundlage eines Materials der realen Welt, z.B. aus einer oder mehreren Fotografien.



Author: Dave Riganelli (<https://daverig.artstation.com/>)

Source: <https://www.allegorithmic.com/blog/scan-anything-dave-riganelli-and-his-homemade-scanbox>

Texture Scans

Da Fotografien nur Abbildungen der Oberfläche ermöglichen, werden hier 2D Texturen für Texture Mapping erzeugt.

- Vorteil: Kann sehr gute Qualität erzeugen, Einfangen kleinster natürlicher Details
- Nachteil: Material muss geeignet sein, ggf. spezielle Aufnahmegeräte, nicht-triviale Nachbearbeitung



<https://www.artstation.com/pixelgoat>

Texturen aus Fotografien

Problem von Fotografien: Wir sehen das Resultat der Interaktion mit Licht an der Oberfläche, und *nicht* die Oberflächendetails selbst!

- Licht als Summe verschiedener physikalischer Prozesse (z.B. Diffuse und Spekulare Reflexion, Refraktion, Subsurface-Scattering, ...)

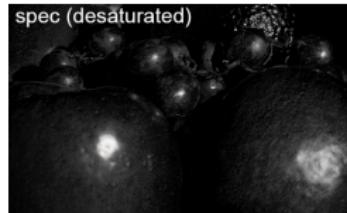
Weitere Nachteile von Fotografien:

- Fotografien in seltensten Fällen “kachelbar”
- Ggf. perspektivische Verzerrung

Trennen von Lichtanteilen

Trennen von Spekularem und Diffusem Licht durch **Kreuz-Polarisation**.

- Polarisieren des Lichts an der Lichtquelle
- Polarisationsfilter vor Kameralinse (90° rotiert)
- Ergebnis: Albedo (allerdings mit Ambient Occlusion)
- Spekularanteil durch Subtraktion von Albedo von unpolarisiertem Foto



<https://docs.sharktacos.com/photography/xpol.html>

Umgebungsverdeckung

Umgebungsverdeckung (Ambient Occlusion) entspricht restlichem Schatten in Albedo. Einfacher Ansatz zum Heraustrennen aus Fotografie:

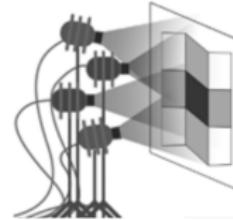
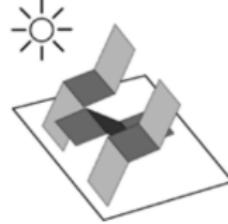
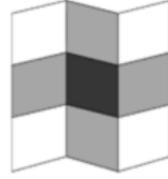
- Entzärtigen des Bildes (z.B. in HSV-Farbraum): dunkle Flächen entsprechen Schatten aus Umgebungsverdeckung
- Aufhellen der entsprechenden Stellen in Albedo Map
- Nachteil: Physikalisch nicht korrekt

Alternative: Oberflächenrekonstruktion und Raytracing ("Baking").

Mikro- und Mesogeometrie

Problem: Erscheinung eines Objekts in einem Foto kann unterschiedliche Ursachen haben. [1]

- Normalen und Höhe aus einzelner Aufnahme nicht unbedingt eindeutig
- Selbst für Menschen mitunter schwierig (vgl. optische Täuschung)
- Nur kontrollierte Beleuchtung lässt Rückschlüsse auf Oberfläche zu



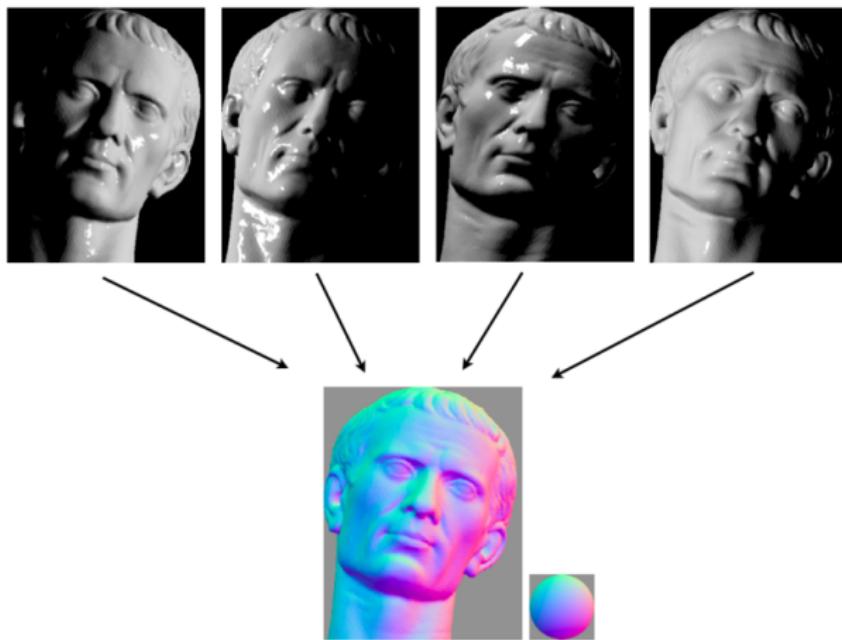
Mikro- und Mesogeometrie

Mehrere Ansätze zum Erfassen der Oberfläche:

- Einfacher Ansatz: RGB-D Kamera nimmt Höhenfeld auf, Normalen aus horizontalen und vertikalen Gradienten (Sobel-Operator)
- Alternativ: Beleuchten aus mehreren Richtungen, Rekonstruktion der Normalen, Höhenfeld durch Integration
 - Beleuchtete Bereiche: Oberfläche reflektiert zur Kamera (positiver Gradient)
 - Schatten: negativer Gradient

Photometric Stereo

Photometric Stereo: Mehrere Aufnahmen mit Licht aus verschiedenen Richtungen erlaubt Abschätzen der Oberflächennormalen [3].

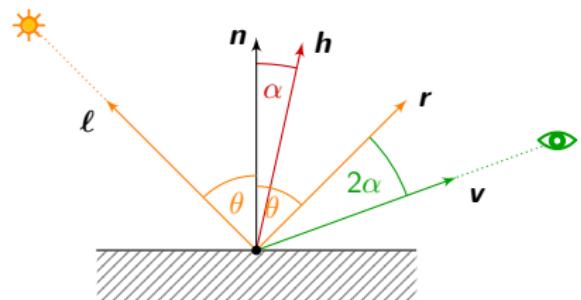


Normal Map Estimation

Diffuse Reflexion an einem Punkt
wird berechnet als:

$$I_d = k_d \cdot \langle \mathbf{n}, \ell \rangle$$

Umgekehrte Vorgehensweise als
beim Shading: I_d bekannt, \mathbf{n}
gesucht.



Normal Map Estimation

Lambert'sches Reflexionsgesetz: Intensität I_d entspricht Helligkeit an Punkt. Je heller, desto näher ist r an v .

Errechnen der Normalen:

- Aufstellen und Lösen eines Gleichungssystems
- Normalen sind Koordinaten in 3-dimensionalem Raum
- benötigt (mindestens) 3 Aufnahmen mit unterschiedlichen Lichtquellen

$$\underbrace{\begin{bmatrix} I_1 \\ I_2 \\ I_3 \end{bmatrix}}_{I=[3 \times 1]} = k_d \cdot \underbrace{\begin{bmatrix} \ell_1^T \\ \ell_2^T \\ \ell_3^T \end{bmatrix}}_{L=[3 \times 3]} \cdot \underbrace{\mathbf{n}}_{[3 \times 1]}$$

$$I = L \cdot k_d \cdot n$$

Normal Map Estimation

Ergebnisse lassen sich durch Einsetzen von mehr Lichtquellen verbessern.

$$\begin{bmatrix} I_1 \\ I_2 \\ \vdots \\ I_n \end{bmatrix} = k_d \cdot \begin{bmatrix} \ell_1^T \\ \ell_2^T \\ \vdots \\ \ell_n^T \end{bmatrix} \cdot \mathbf{n}$$

Lösen durch Methode der kleinste Quadrate:

$$\begin{aligned} I &= \mathbf{L} \cdot k_d \cdot \mathbf{n} \\ \mathbf{L}^T \cdot I &= \mathbf{L}^T \mathbf{L} \cdot k_d \cdot \mathbf{n} \\ k_d \cdot \mathbf{n} &= (\mathbf{L}^T \mathbf{L})^{-1} (\mathbf{L}^T \cdot I) \end{aligned}$$

Height Map Estimation

Berechnen der Height Map aus Normalenfeld:

- Normalen beschreiben Gradienten (Änderung der Höhe) auf Oberfläche
- Integrieren durch “Entlangwandern” auf Oberfläche in horizontale (\mathbf{v}_h) und vertikale Richtung (\mathbf{v}_v):

$$\begin{aligned}\mathbf{v}_h &= \begin{bmatrix} c_{x+1} \\ c_y \\ z_{x+1,y} \\ c_x \end{bmatrix} - \begin{bmatrix} c_x \\ c_y \\ z_{xy} \\ c_x \end{bmatrix} = \begin{bmatrix} c_{x+1} - c_x \\ 0 \\ z_{x+1,y} - z_{xy} \\ 0 \end{bmatrix} \\ \mathbf{v}_v &= \begin{bmatrix} c_{y+1} \\ z_{x,y+1} \end{bmatrix} - \begin{bmatrix} c_y \\ z_{xy} \end{bmatrix} = \begin{bmatrix} c_{y+1} - c_y \\ z_{x,y+1} - z_{xy} \end{bmatrix}\end{aligned}$$

Height Map Estimation

Weitere Annahme: "geglättete Oberfläche", d.h. benachbarte Punkte haben etwa gleiche Höhe (\mathbf{v}_h und \mathbf{v}_v fast senkrecht zu \mathbf{n})

$$\langle \mathbf{n}, \mathbf{v}_h \rangle \approx \langle \mathbf{n}, \mathbf{v}_v \rangle \approx 0$$

Durch Einsetzen ergibt sich:

$$\begin{aligned} 0 &= \langle \mathbf{n}, \mathbf{v}_h \rangle = \mathbf{n}_x(c_{x+1} - c_x) + \mathbf{n}_z(z_{x+1,y} - z_{xy}) \\ 0 &= \langle \mathbf{n}, \mathbf{v}_v \rangle = \mathbf{n}_y(c_{y+1} - c_y) + \mathbf{n}_z(z_{x,y+1} - z_{xy}) \end{aligned}$$

Lösen des Gleichungssystems ergibt Näherung für z_{xy} .

Diskussion

Vorgestellter Ansatz funktioniert mit Einschränkungen des aufgenommenen Materials:

- Nur auf Lambert'schen Oberflächen (Diffusoren); Keine Reflektoren/Refraktoren!
- Keine Löcher erlaubt
- Es kann Stellen geben, die im Dauerschatten liegen können

Außerdem: Kamera und Lichtquellen müssen genau kalibriert werden, d.h. ℓ muss genau genug bestimmt werden können.

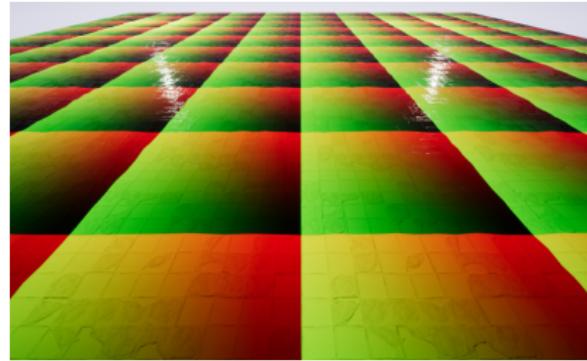
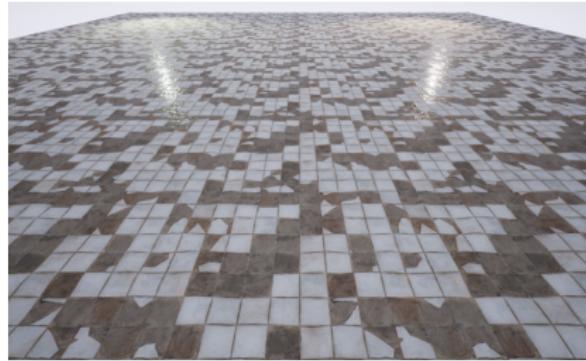
Für gute Albedo-Maps, sollten die Lichtquellen den sichtbaren Lichtteil optimal abdecken (vgl. "Normlicht D65").

3.4 Exkurs: Kleine Auswahl weiterer Themen (Nicht Prüfungsrelevant)

Tiling

Das räumliche Wiederholen einer Textur entlang einer Oberfläche nennt man **Tiling**.

- Texturen haben begrenzte Auflösung (begrenzt durch verfügbaren Speicher)
- Texturierung größerer Flächen oft durch Nebeneinanderlegen einer Textur



Erzeugen einer Tile-Textur

Textur muss für Tiling angepasst werden, damit Übergänge nicht sichtbar sind.

Tiling kann unterschiedlich erzeugt werden:

- Prozedurale Texturen: Definieren einer periodischen Funktion
- Fotografien: manuelle Nachbearbeitung, Glätten im Fourier-Raum, ...

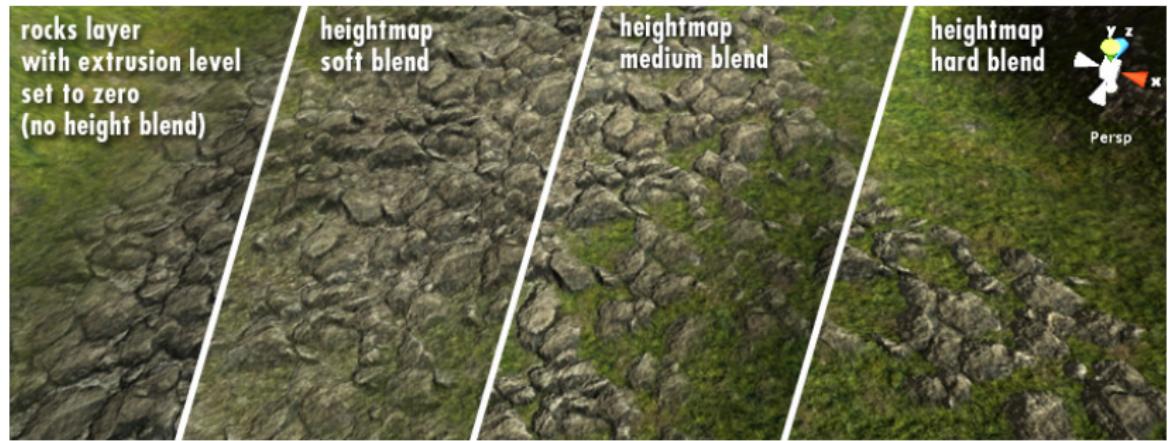
Tiling-Artefakte

Durch periodisches Wiederholen der Textur entstehen (oft unnatürliche) Regelmäßigkeiten, **Tiling-Artefakte**.



Texture Blending

Blending: Vermeiden von Tiling-Artefakten und harten Übergängen durch Vermischen verschiedener Texturen.



Source: <https://www.unrealengine.com/>

Eigenschaften einer Textur

Bisher: Texturen immer nur als 2-dimensionales Feld von Pixeln betrachtet. Beschreibung für viele Anwendungen ungenügend:

- Pixelfarbe (-normale, -höhe, etc.) in der Natur oft abhängig von Umgebung
- Deshalb: Beschreibung als *Markov-Ketten*
- Mathematisch: Pixel ist abhängig von jedem anderem Pixel, mit Entfernung als Gewicht.
- Annäherung: Betrachten kleines “Fenster” um Pixel (z.B. 3×3 , 5×5)
- Erlaubt Klassifikation von Texturen nach bestimmten Eigenschaften

Lokalität

Schaut man durch die “Fenster” um zwei *benachbarte* Pixel, so sehen Nachbarschaften ähnlich aus.

Gilt diese Beobachtung für die gesamte Textur, heißt sie **lokal**.



Lokalität ist eine Eigenschaft der meisten natürlichen Texturen.

Inhomogenität

Eine Textur heißt **stationär**, die gleiche Beobachtung für zwei *zufällig gewählte* Nachbarschaften gilt.

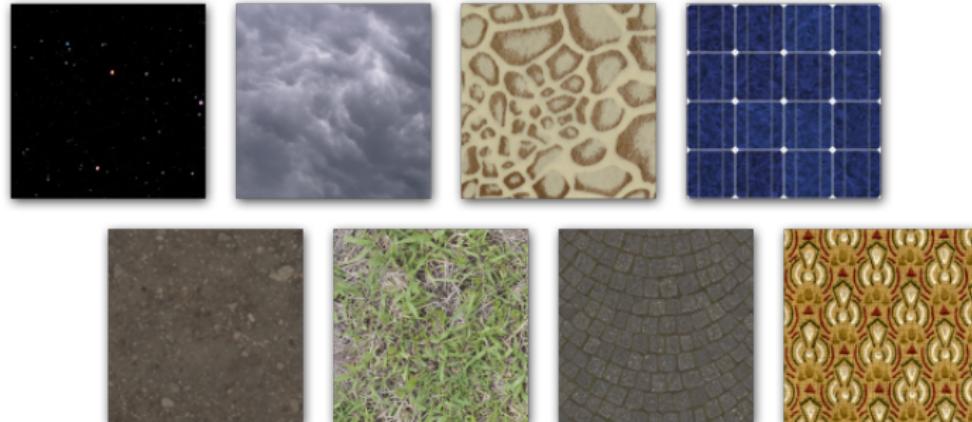


Nicht-Stationarität (oder *Inhomogenität*) ist ebenfalls eine Eigenschaft vieler natürlicher Texturen.

Regularität

Texturen lassen sich auch durch die Größe und Verteilung der dargestellten *Features* beschreiben. Besteht die Textur aus sich periodisch wiederholenden Features, heißt sie **regulär**. Lassen sich kaum individuelle Features ausmachen, heißt die Textur **stochastisch**.

Reguläre Texturen sind oftmals künstliche (menschgemachte) Oberflächen, während natürliche Texturen häufig *semi-regulär* sind, also irgendwo zwischen stochastisch und regulär liegen.

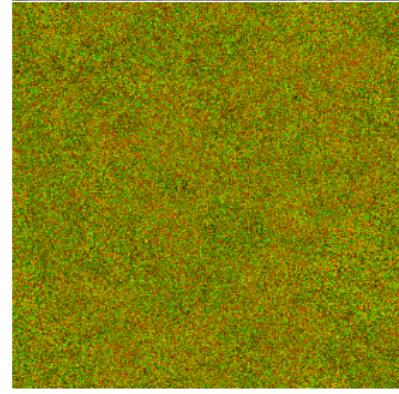
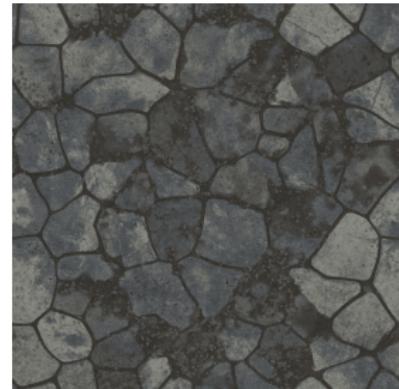
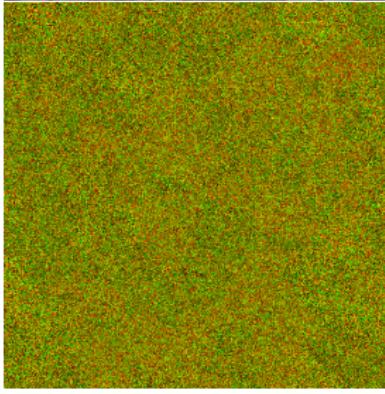
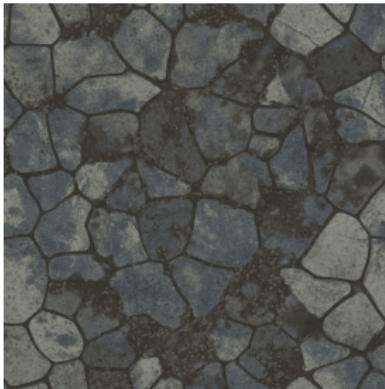
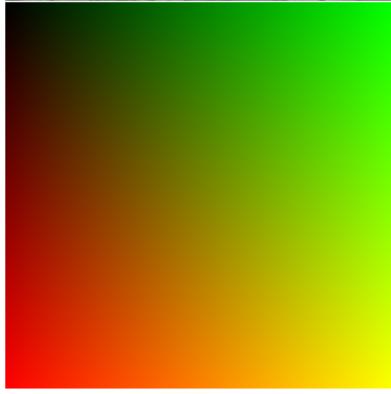
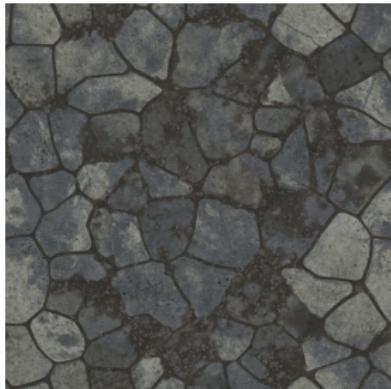


UV Remapping

Interessante Beobachtung: durch Umordnen der UV-Koordinaten lässt sich Textur dennoch rekonstruieren.

- Code zum selbst ausprobieren: <https://git.io/fjJj2>

UV Remapping



Textursynthese

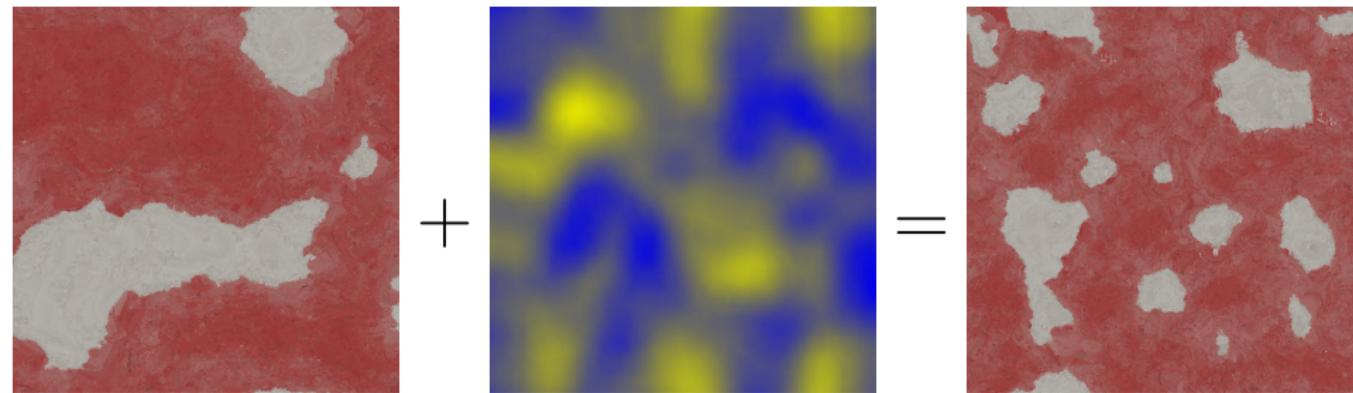
Nutzen der Beobachtung: Können wir die Struktur einer Textur (Lokalität, Inhomogenität, Regularität) als Funktion beschreiben, können wir neue Texturen mit visueller Ähnlichkeit generieren.

Exemplar-basierte **Textursynthese**: Beschreiben einer neuen Textur mit visueller Ähnlichkeit zu einem vorgegebenen Exemplar.

- Erlaubt Erzeugung einer (theoretisch) beliebig großen Textur aus kleinem *Exemplar*
- Kann genutzt werden um Tiling-Artefakte zu umgehen
- Gleichzeitig: Reduzieren des Speicherbedarfs
- Periodische Domäne: generierte Textur automatisch “tileable”
- Viele weitere Anwendungen in Bildverarbeitung und -manipulation
- Forschungsthema an der TUC :-)

Kontrollierte Synthese

Vorgeben der Parametrierung (hier z.B. Inhomogenität) erlaubt Kontrolle über Ergebnis.



Quellen und Literaturverweise

- [1] E. H. Adelson and A. P. Pentland. The perception of shading and reflectance. *Perception as Bayesian inference*, pages 409–423, 1996.
- [2] K. Perlin. An image synthesizer. *ACM Siggraph Computer Graphics*, 19(3):287–296, 1985.
- [3] R. J. Woodham. Photometric stereo: A reflectance map technique for determining surface orientation from image intensity. In *Image Understanding Systems and Industrial Applications I*, volume 155, pages 136–144. International Society for Optics and Photonics, 1979.