

# Computergraphik II

## 2. Texturen I

Carsten Rudolph

Graphische Datenverarbeitung und Visualisierung  
Fakultät für Informatik



TECHNISCHE UNIVERSITÄT  
CHEMNITZ

Sommersemester 2019

## 2.1 Recap: Texturen, Texture Mapping

# Motivation

- Polygonale Modelle approximieren Oberfläche nur
- Hoher Detailgrad: viele, kleine Polygone
- Speicher- und Renderzeit nicht akzeptabel
- Deshalb: Andere Darstellung von Details
- **Texture Mapping** zur Projektion "höher-frequenter" Merkmarke auf geometrische Oberfläche 多边形模型仅接近表面  
高水的细节：许多小的多边形  
内存和渲染时间不可接受  
因此：细节的不同呈现  
用于在几何表面上投影“更高频率”特征的纹理映射

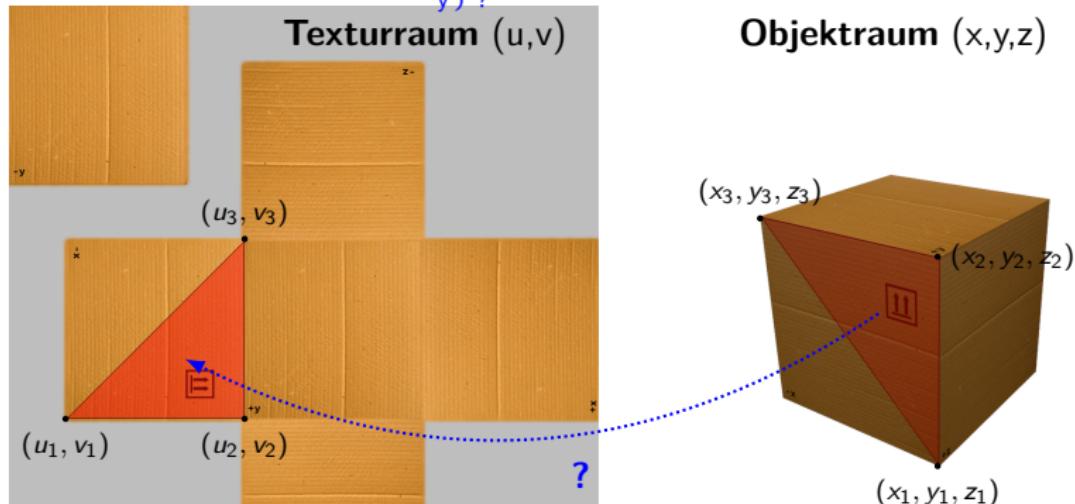


# Recap: Texture Mapping

## Texture Mapping

Vorgang der Abbildung einer Textur auf Flächenelemente der Szene  
在场景的表面元素上映射纹理的过程

Koordinaten  $u, v \in [0.0, 1.0]$  beschreiben Punkt in Texturraum. Element an Punkt  $(u, v)$  heißt **Texel** (vgl. "Pixel"). Texture Mapping: Welches Texel  $(u, v)$  gehört zum Pixel  $(x, y)$ ? 坐标  $u, v \in [0.0, 1.0]$  描述纹理空间中的点。点到  $(u, v)$  的元素称为 texel (参见 "像素" )。纹理映射：哪个纹素  $(u, v)$  属于像素  $(x, y)$  ?



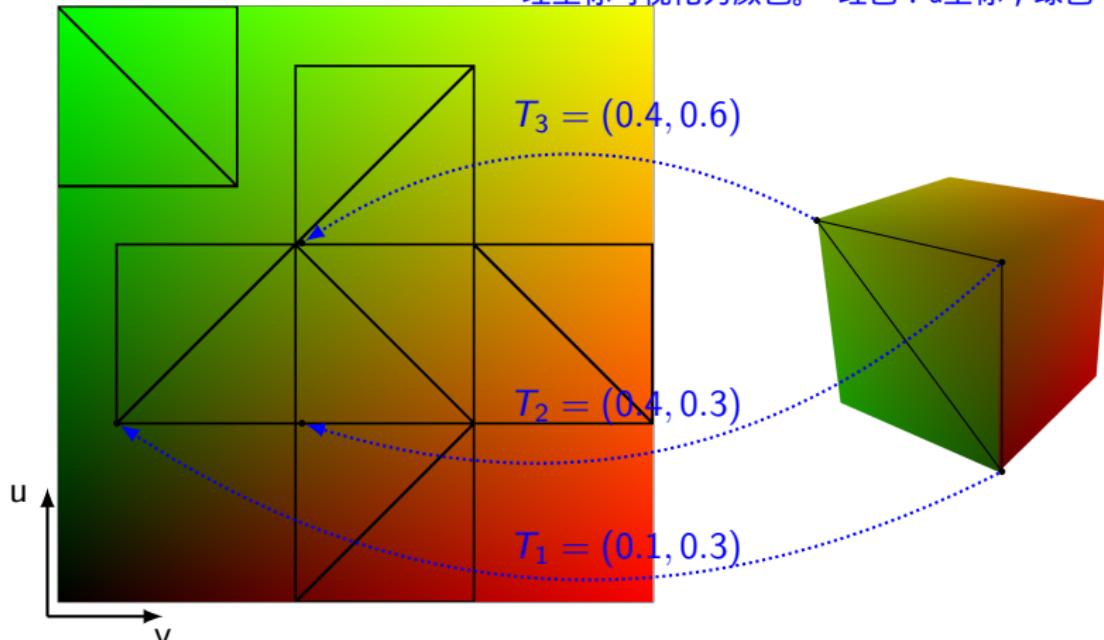
# Recap: Texture Mapping

Assoziieren jedes Vertex mit Texturkoordinaten  $T = (u, v)$

Zur Veranschaulichung: Visualisierung der Texturkoordinaten als Farben. Rot:

$u$ -Koordinate, Grün:  $v$ -Koordinate

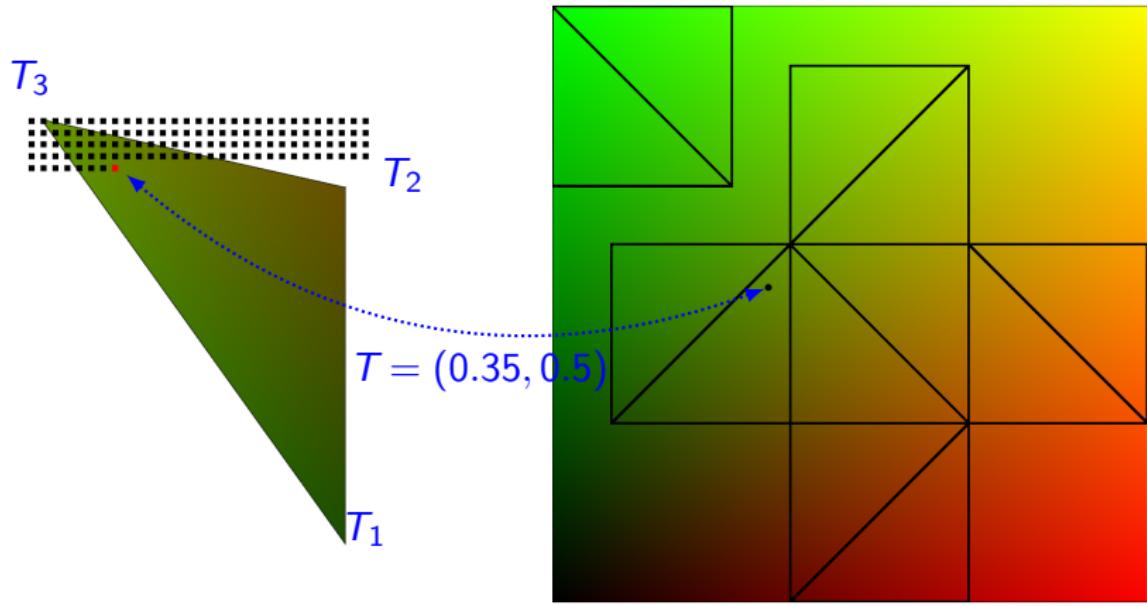
将每个顶点与纹理坐标  $T = (u, v)$  相关联来说明：将纹理坐标可视化为颜色。 红色： $u$  坐标，绿色： $v$  坐标



# Recap: Texture Mapping

Zum Rendern der Textur:

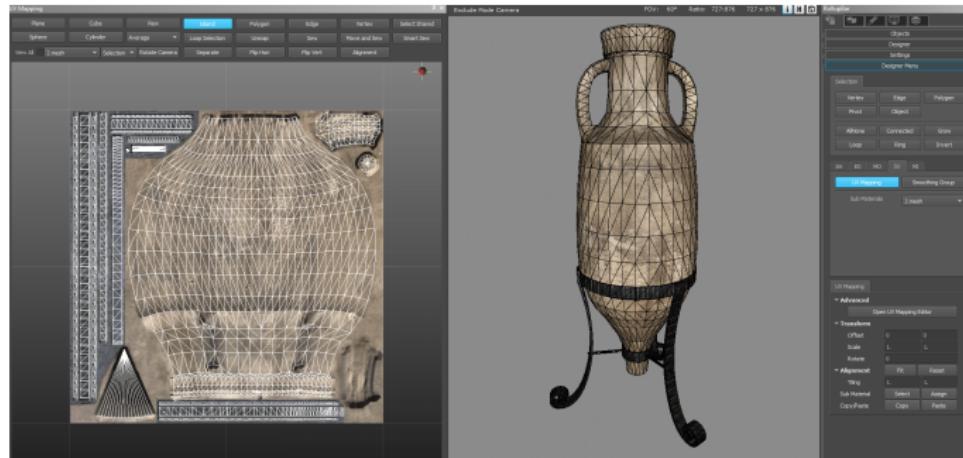
- 1 Rasterisierer interpoliert Texturkoordinaten 光栅化插值纹理坐标
- 2 Texel-Lookup durch Fragment-Prozessor 通过片段处理器查找Texel



# Recap: Texture Mapping

用纹理纹理不同的对象

- Texturierung verschiedener Objekte mit einer Textur
- UV-Atlas
- ...



Quelle: <https://docs.cryengine.com/>

## 2.2 Erweiterte Anwendungen

# Erweiterte Anwendungen von Texturen

到目前为止：纹理存储纹素的颜色值 (RGB / A)  
将纹理用于材质参数允许与不同着色器进行交互  
与照明的互动：反照率，镜面反射，法线  
与几何的相互作用：高度，位移

- Bisher: Texturen speichern Farbwerte (RGB/A) für Texel
- Verwenden von Texturen für Materialparameter ermöglicht Interaktion mit verschiedenen Shadern
- Interaktion mit Beleuchtung: Albedo, Spekularität, Normalen
- Interaktion mit Geometrie: Height, Displacement

Texturen können nicht nur Farbe, sondern beliebige Information speichern.  
Ermöglicht variieren von Materialeigenschaften entlang von Primitiven.

纹理不仅可以存储颜色，还可以存储任意信息。  
允许沿基元改变材料属性。

## 2.3 Texturen für Beleuchtungsparameter

# Was ist Farbe?

- Bisher: Texturen speichern Farbwerte (RGB/A) für Texel  
到目前为止：纹理存储纹素的颜色值 (RGB / A)  
颜色从哪里来？
- Woher kommt die Farbe in einem Punkt?  
答：低光，不吸收。
- Antwort: Restlicht, welches nicht absorbiert wird.  
不同材料以不同方式反射/吸收不同波长的光。
- Verschiedene Materialen reflektieren/absorbieren Licht verschiedener Wellenlängen unterschiedlich gut.
- Farbe also abhängig von Reflexionsverhalten an Punkt einer Oberfläche  
因此颜色取决于表面点处的反射行为

*Blätter sind Grün, weil nur der grüne Anteil des weißen Sonnenlichts nicht absorbiert wird.* 叶子是绿色的，因为只有白色阳光的绿色部分不被吸收。

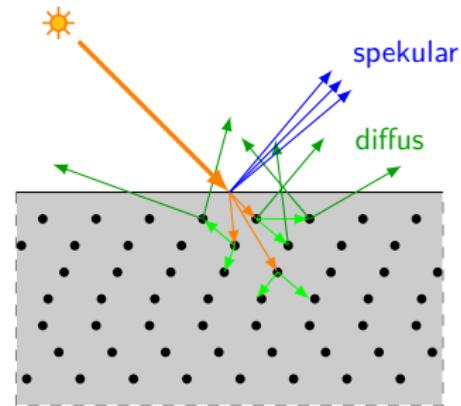
# Reflexionsarten

光线照射到表面时的不同现象：

Unterschiedliche Phänomene beim Auftreffen von Licht auf eine Oberfläche:

镜面部分：光线在介质之间的过渡处反射，表面的不均匀性提供“扇形”

- 1 spekularer Anteil: Licht wird am Übergang zwischen Medien reflektiert, Unebenheiten der Oberfläche sorgen für „Auffächern“
- 2 diffuser Anteil: Licht dringt in das Material ein, wird dort weiterreflektiert und absorbiert (*subsurface scattering*), ein Teil gelangt wieder nach außen



漫射部分：光线穿透到材料中，进一步被反射和吸收（次表面散射），一部分返回到外部

## Blinn-Phong Beleuchtungsmodell

Das Blinn-Phong Beleuchtungsmodell [4][1] beschreibt Farbe an Oberflächenpunkt als Summe dreier Komponenten:

$$\hat{I} = k_a \cdot I_a + k_d \cdot I_d \cdot \max(\langle n, \ell \rangle, 0) + k_s \cdot I_s \cdot [\max(\langle n, h \rangle, 0)]^s$$

ambienter Anteil  
diffuser Anteil  
spekularer Anteil

Annahme: Wechselwirkung zwischen Licht und farbiger Oberfläche lässt sich komponentenweise im Farbraum (z.B. RGB) beschreiben.

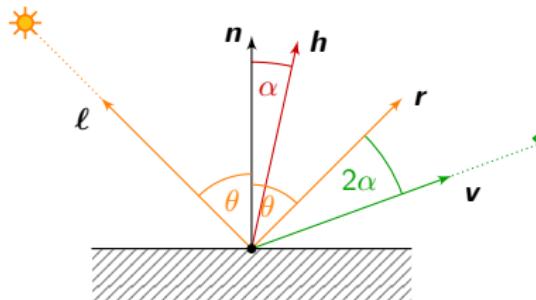
Ambientes Licht entspricht Restlicht der Umgebung, ist also nicht abhangig von Oberflachenbeschaffenheit.

假设：光和彩色表面之间的相互作用可以在颜色空间中逐个组件地描述（例如RGB）。环境光对应于环境的残余光，因此它不依赖于表面条件。

# Blinn-Phong Beleuchtungsmodell

Vektoren im Blinn-Phong Modell:

- $\ell$  Licht
- $r$  Reflexionsvektor
- $n$  Normale
- $v$  Kamera/Eye
- $h$  Halfway (zwischen  $v$  und  $\ell$ )



$$\hat{I} = k_a \cdot I_a + k_d \cdot I_d \cdot \max(\langle n, \ell \rangle, 0) + k_s \cdot I_s \cdot [\max(\langle n, h \rangle, 0)]^s$$

ambienter Anteil  
diffuser Anteil  
spekularer Anteil

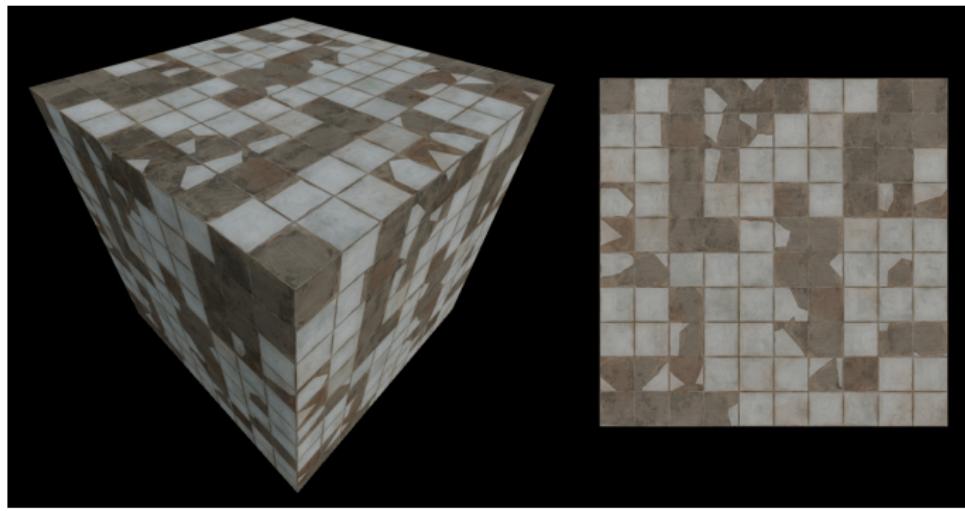
# Albedo

**Albedo:** Diffuser Anteil entspricht nicht-absorbiertem Restlicht

漫射分数对应于未被吸收的低光

- Ähnlich zu klassischer RGB Textur 与经典的RGB纹理相似
- Farbwerte entsprechen diffusem Licht an Texel ( $u, v$ ) 颜色值对应Texel上的漫反射光 ( $u, v$ )

$$\hat{I}_d(u, v) = k_d \cdot I_d(u, v) \cdot \max(\langle \mathbf{n}, \ell \rangle, 0)$$

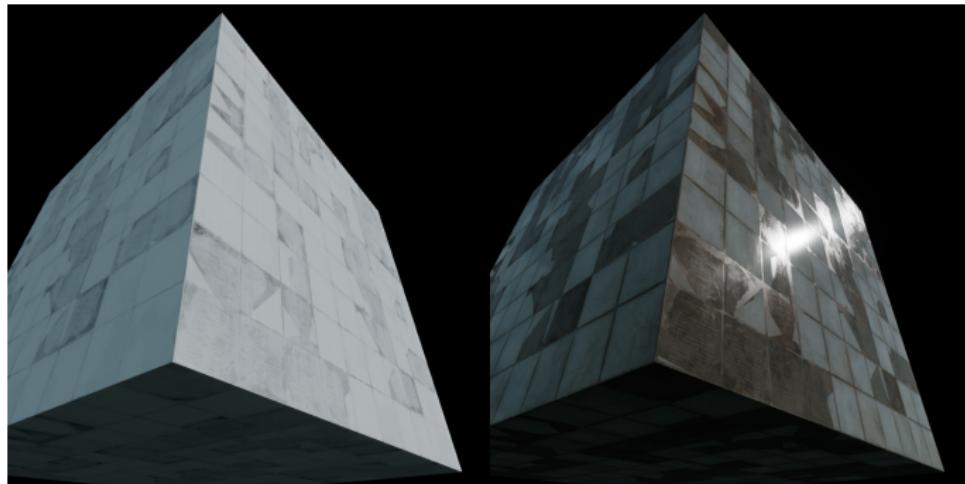


# Spekularität

**Specularity Map:** Weist einem Texel  $(u, v)$  einen bestimmten Spekularexponenten zu. Ein hoher Wert beschreibt einen guten Reflektor (glatte Oberfläche), ein niedriger Wert beschreibt einen Diffusor (raue Oberfläche).

Specularity Map : 为 texel  $(u, v)$  指定一个特定的镜面反射指数。高值表示良好的反射器（光滑表面），低值表示漫射器（粗糙表面）。

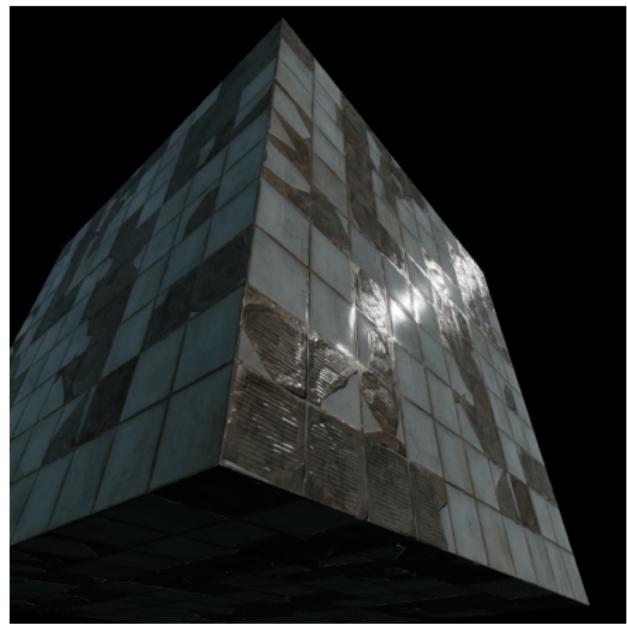
$$\hat{l}_s(u, v) = k_s \cdot l_s \cdot [\max(\langle \mathbf{n}, \mathbf{h} \rangle, 0)]^{s(u, v)}$$



# Normalen

Bisher: Alle Austrittswinkel gleich, also Oberfläche perfekt glatt. Das ist in der Realität allerdings fast nie der Fall, deshalb weist die **Normal Map** einem Texel  $(u, v)$  eine Normale zu.

到目前为止：所有出口角度相等，因此表面非常光滑。  
然而，实际上几乎不是这种情况，  
因此法线贴图将纹理分配给纹素  $(u, v)$ 。



$$\hat{l}(u, v) = k_d \cdot l_d(u, v) \cdot \max(\langle \mathbf{n}(u, v), \ell \rangle, 0) + \\ k_s \cdot l_s \cdot [\max(\langle \mathbf{n}(u, v), \mathbf{h} \rangle, 0)]^{s(u, v)}$$

# Information einer Normale

漫反射贴图包含Id的颜色值，即3D RGB矢量。Specularity Map存储标量系数s。

Diffuse Map enthält Farbwerte für  $I_d$ , also einen 3D RGB Vektor. Die Specularity Map speichert einen skalaren Koeffizienten s.

Wie werden Normalen dargestellt? 法线如何显示？

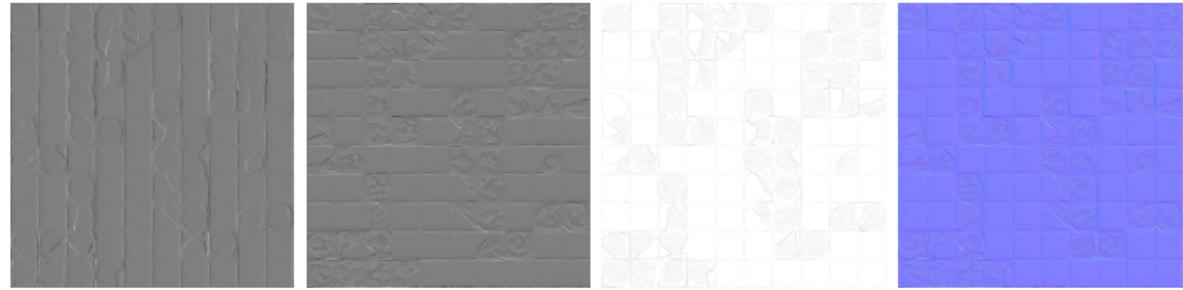
- Normalen sind ebenfalls 3D Vektoren 法线也是3D矢量  
法线不是颜色，而是坐标
- Normalen sind keine Farben, sondern Koordinaten
- z.B. 8-Bit RGB:  $\mathbf{c} = \{r, g, b \in \mathbb{Z} | 0 \leq r, g, b \leq 255\}$ .
- Normalen:  $\mathbf{n} = \{x, y, z \in \mathbb{R} | -1.0 \leq x, y, z \leq 1.0\}$ .

# Umrechnen von Normalen in Normal Maps

要创建法线贴图函数需要  $f : ()$

Zum Erstellen einer Normal Map wird Funktion benötigt  $f: N \mapsto C$ .

$$\begin{aligned}f(\mathbf{n}) &= 255 \cdot \left( \frac{1}{2}\mathbf{n} + 0.5 \right) \\f^{-1}(\mathbf{c}) &= \frac{1}{255} \cdot (2\mathbf{c} - 1.0)\end{aligned}$$



Anm.: Faktor 255 entsprechen Normalisierung auf 8-Bit Maps!

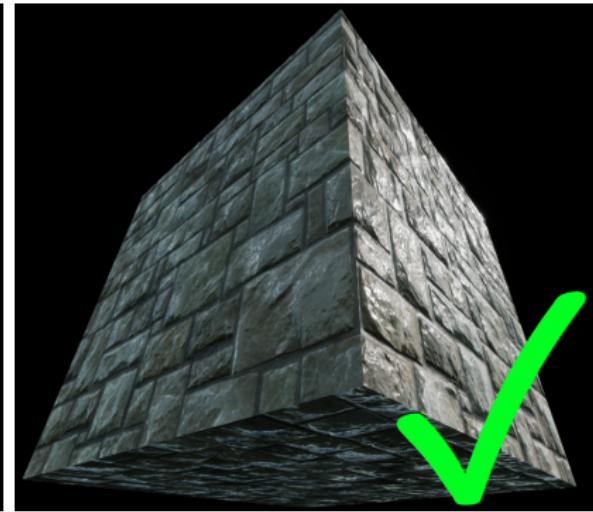
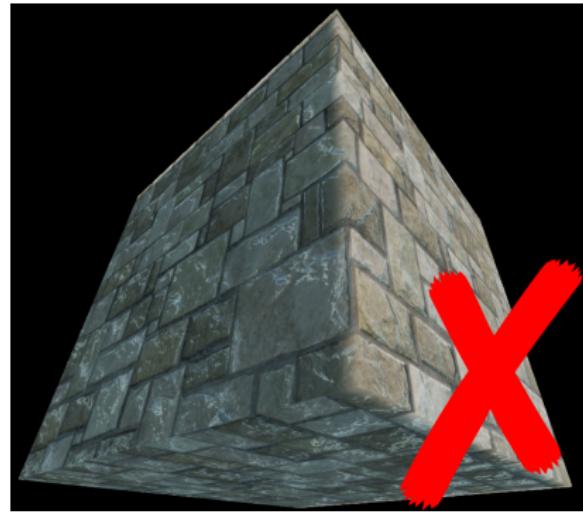
# Tangentialraum

Normalen sind Koordinaten, aber in welchem Raum?

法线是坐标，但在哪个空间？

Relativ zur Oberfläche, also weder *Object*, *World* (linkes Bild), noch  
*Camera Space*.

Transformation in Raum auf Oberfläche: **Tangent Space** (rechtes Bild).  
表面空间变换：切线空间（右图）。



# Tangentialraum

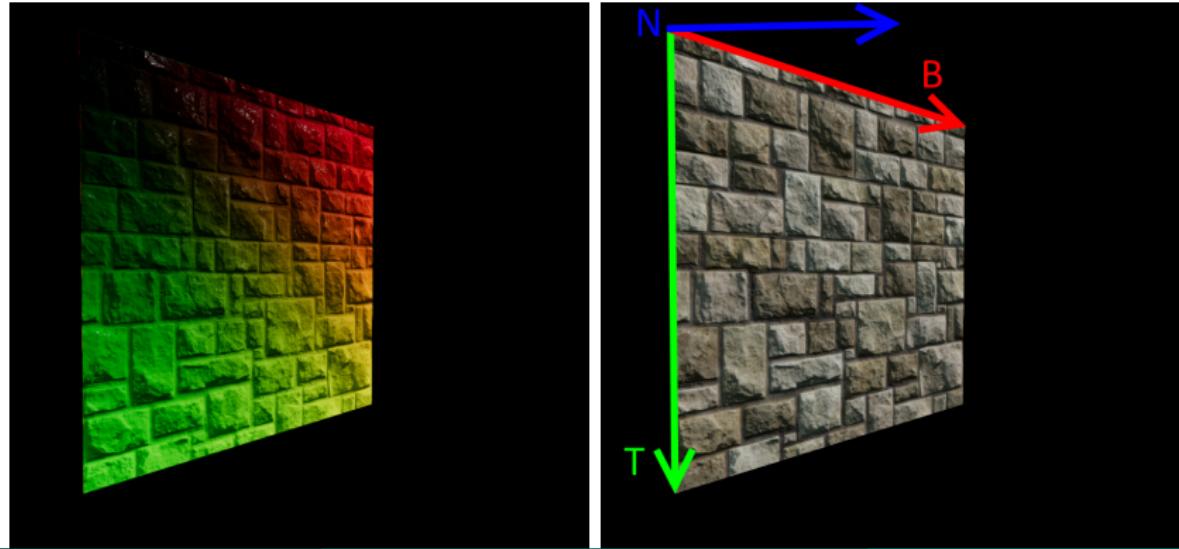
**Tangent Space:** Raum, welcher durch 3 linear unabhängige Vektoren aufgespannt wird: 切线空间：由3个线性独立向量跨越的空间：

- 1 Die Oberflächennormale des Primitivs  $\vec{N}_0$
- 2 Die Tangente  $\vec{T}$  in Richtung der aufsteigenden Texturkoordinate  $u$
- 3 Die Bitangente  $\vec{B}$  in Richtung der aufsteigenden Texturkoordinaten  $v$

片元 $N_0$ 的表面法线

在上升纹理坐标 $u$ 的方向上的切线 $T$ .

在上升纹理坐标 $v$ 的方向上的位切线 $B$ .

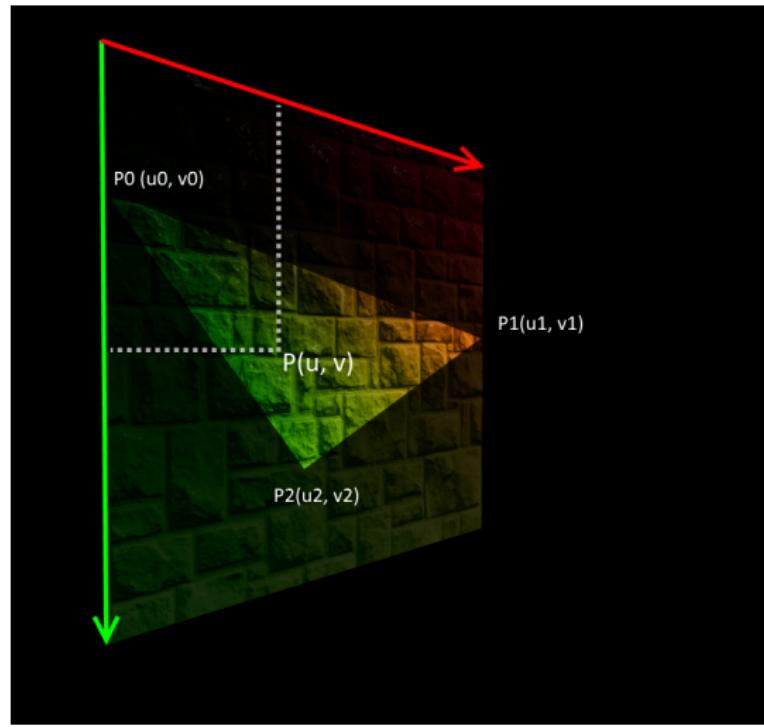


# Rechnen im Tangentialraum

在Tangent Space中，基元的点可以描述为：

Im Tangent Space lässt sich  
Punkt eines Primitivs lässt  
sich beschreiben als:

$$\begin{aligned}\vec{p}(u, v) &= \vec{p}_0 \\ &+ (u - u_0) \vec{T} \\ &+ (v - v_0) \vec{B}\end{aligned}$$



# Berechnung des Tangentialraumes

Durch Aufstellen eines Gleichungssystems können die Vektoren  $\vec{T}$  und  $\vec{B}$  berechnet werden:

通过建立方程组，可以计算向量T和B：

$$\begin{aligned}\vec{p}_1 - \vec{p}_0 &= (u_1 - u_0) \cdot \vec{T} + (v_1 - v_0) \cdot \vec{B} \\ \vec{p}_2 - \vec{p}_0 &= (u_2 - u_0) \cdot \vec{T} + (v_2 - v_0) \cdot \vec{B}\end{aligned}$$

Umstellen nach  $\vec{T}$  und  $\vec{B}$  ergibt:

$$\begin{aligned}\vec{T} &= \frac{(v_1 - v_0)(\vec{p}_2 - \vec{p}_0) - (v_2 - v_0)(\vec{p}_1 - \vec{p}_0)}{(u_2 - u_0)(v_1 - v_0) - (u_1 - u_0)(v_2 - v_0)} \\ \vec{B} &= \frac{(u_1 - u_0)(\vec{p}_2 - \vec{p}_0) - (u_2 - u_0)(\vec{p}_1 - \vec{p}_0)}{(u_1 - u_0)(v_2 - v_0) - (u_2 - u_0)(v_1 - v_0)}\end{aligned}$$

Der Normalenvektor lässt sich über das Kreuzprodukt ermitteln:

法向量可以通过叉积确定：

$$\vec{N} = \vec{T} \times \vec{B}$$

# Koordinatentransformation

- Gegeben sei ein Punkt im Tangentialraum (Tangent Space) der Form  $(u, v, h)^T$  给定形式的切线空间中的一个点  $(u, v, h)$
- Gesucht sei der zugehörige Punkt  $\vec{p}(u, v, h)$  im Objektraum (Object Space). 我们正在寻找对象空间(对象空间)中的对应点  $p(u, v, h)$

Aufstellen der TBN-Matrix  $\mathbf{M} = [\vec{T}, \vec{B}, \vec{N}]$

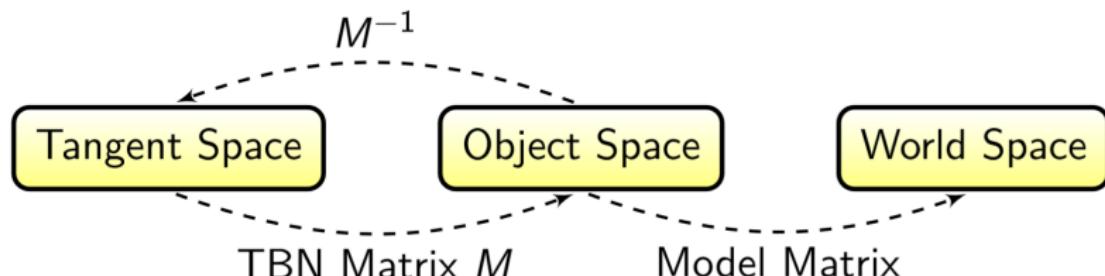
Anschließend lösen der Gleichung:

设置TBN矩阵  $M =$   
然后解决方程式：

$$\vec{p}(u, v) = \vec{p}_0 + \mathbf{M} \cdot (u, v, h)^T$$

Für die entgegengesetzte Richtung, inverse Matrix  $\mathbf{M}^{-1}$  bilden.

对于相反方向，逆矩阵  $M$  形成。



# Orthonormaler Tangentialraum

矢量  $\vec{T}$ ,  $\vec{B}$  和  $\vec{N}$  在切线空间中是正交的，但不一定在对象空间中。然而，这将是期望的，因为它允许通过转置  $MT$  非常简单且有效地计算逆 TBN 矩阵  $M^{-1}$ 。

Die Vektoren  $\vec{T}$ ,  $\vec{B}$  und  $\vec{N}$  sind orthonormal im Tangentialraum, allerdings nicht notwendigerweise auch im Objektraum. Das wäre jedoch wünschenswert, denn damit kann die inverse TBN-Matrix  $\mathbf{M}^{-1}$  sehr einfach und performant durch Transponieren  $\mathbf{M}^T$  berechnet werden.

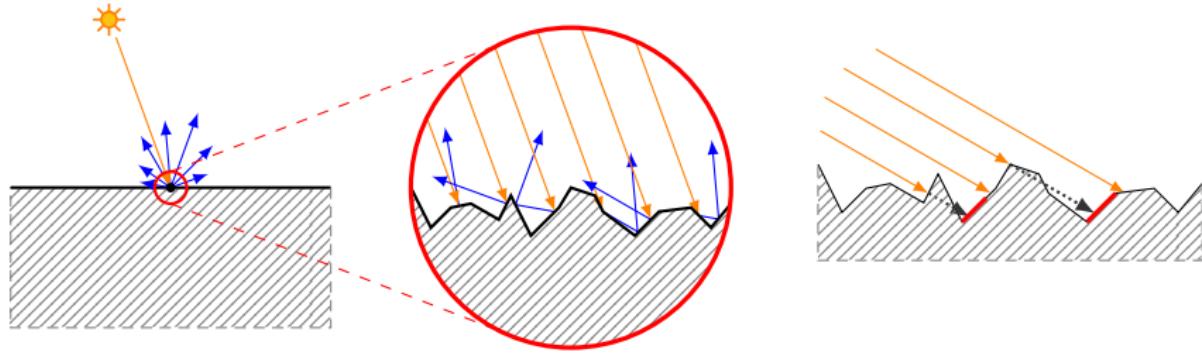
Daher in der Praxis häufig folgender Algorithmus:

- 1 Berechnung von  $\vec{T}$ ,  $\vec{B}$ , wie beschreiben
- 2 Berechnung von  $\vec{N}$  als  $\vec{T} \times \vec{B}$
- 3 Neuberechnen der Bitangente  $\vec{B}$  als  $\vec{N} \times \vec{T}$
- 4 Normalisieren der Vektoren  $\vec{T}$ ,  $\vec{B}$  und  $\vec{N}$

# Selbstbeschattung

Normalen beeinflussen lediglich, in welche Richtung das Licht gebrochen wird. In der Realität sind Oberflächen aber immer uneben, weshalb weitere Effekte auftreten, so z.B. die **Umgebungsverdeckung** (Ambient Occlusion). Umgebungsverdeckung beschreibt Selbstbeschattung durch unebene Oberflächen.

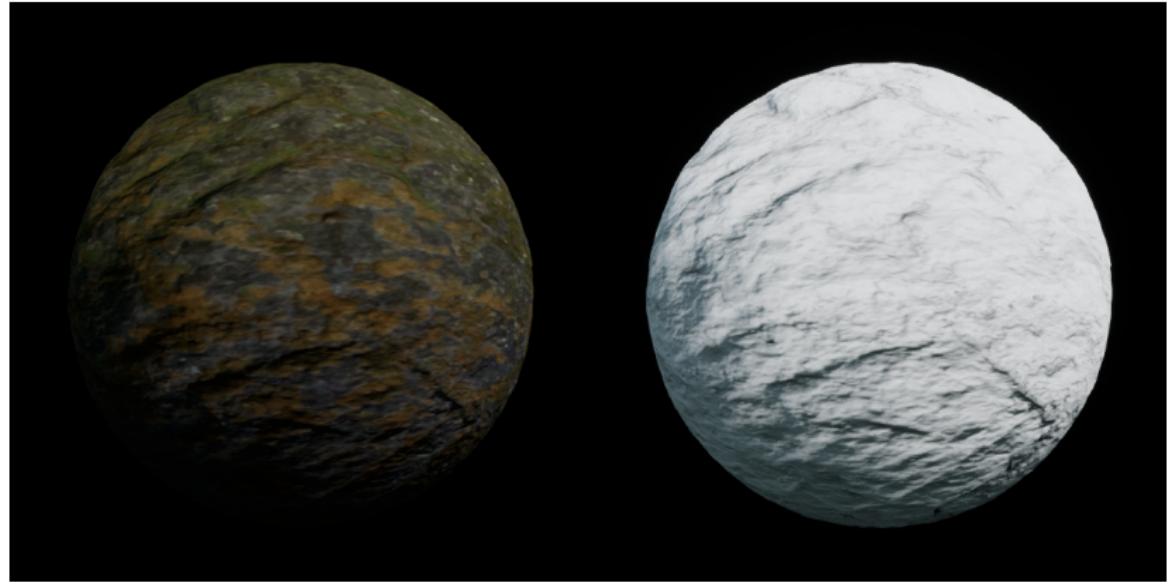
法线仅影响光折射的方向。然而，实际上，表面总是不均匀的，因此会产生进一步的影响，例如，环境遮挡。环境遮挡描述了不平坦表面的自阴影。



# Ambient Occlusion Map

环境遮挡图将点( $u, v$ )指定为描述入射光束被环境遮挡反射或吸收的可能性的因子。

Die **Ambient Occlusion Map** weist einem Punkt ( $u, v$ ) einen Faktor zu, der beschreibt, wie wahrscheinlich ein eintreffender Lichtstrahl reflektiert oder durch Umgebungsverdeckung absorbiert wird.



## 2.4 Texturen für Geometrieparameter

# Höhenfelder

Alle bisherigen Verfahren dienen der Simulation von **Mikro- und Mesodetails**. Sie beschreiben allerdings nur, wie sich Licht an den detaillierten Stellen verhält. Sie beeinflussen nicht die tatsächliche Geometrie.

以前的所有方法都用于模拟微观和中间尾部。但是，它们仅描述了光在详细位置的行为方式。它们不会影响实际几何体。

Problem: Objektsilhouette bleibt "flach", Details besitzen keine plastische Tiefe.

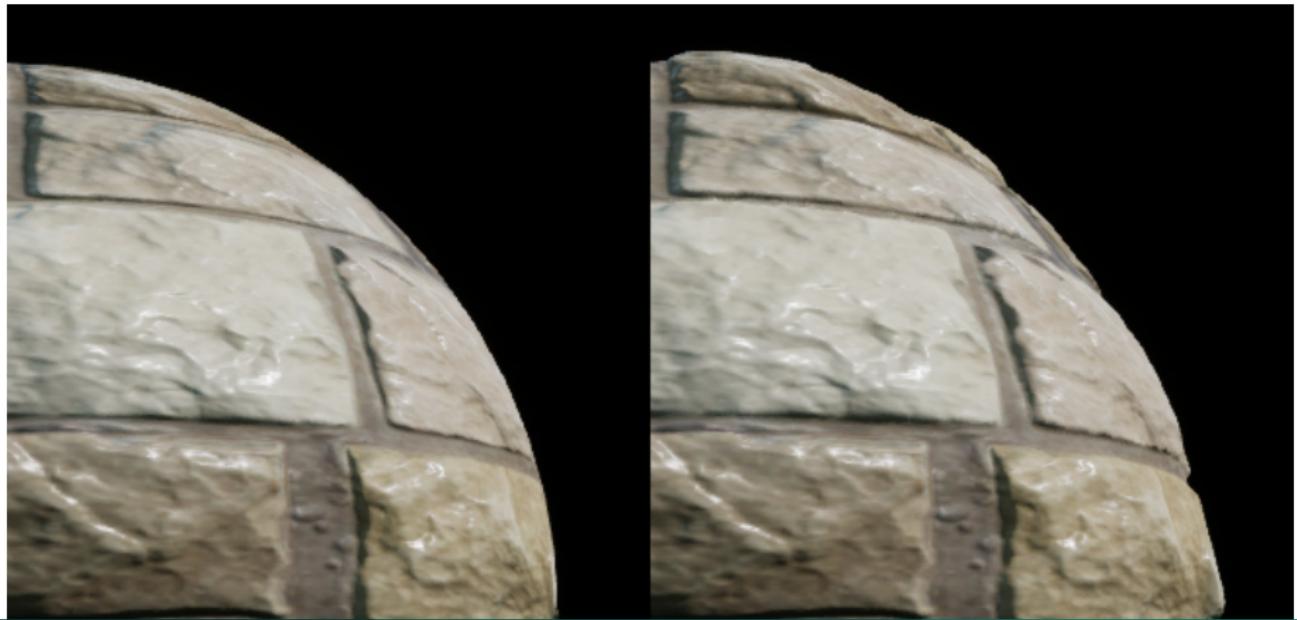
问题：物体轮廓保持“平坦”，细节没有塑料深度。



# Height Map

高度图将高度值分配给点 ( $u, v$ )。由此描述的高度场，例如，可用于沿切线空间中的法线移动顶点。这也影响了轮廓。

Eine **Height Map** weist einem Punkt  $(u, v)$  einen Höhenwert zu. Das dadurch beschriebene *Höhenfeld* z.B. genutzt werden, um Vertices zu entlang der Normale im Tangentialraum zu verschieben. Dadurch wird auch die Silhouette beeinflusst.



# Displacement Mapping

作为位移贴图指沿正常顶点的移动（位移）。

Als **Displacement Mapping**<sup>1</sup> bezeichnet man das Verschieben (*displacement*) der Vertices entlang der Normale.

每顶点：高度贴图作为顶点着色器中的输入

- **Per-Vertex:** Height Map als Eingabe in Vertex Shader
  - 对于顶点 $p$ 的纹理坐标 $(u, v)$ ，让我们定义法向量 $n(u, v)$ 和高度值 $h(u, v)$
- Für Texturkoordinate  $(u, v)$  eines Vertex  $p$  sei ein Normalenvektor  $n(u, v)$ , sowie ein Höhenwert  $h(u, v)$  definiert
  - 计算切线空间中 $p + h(u, v) \cdot n(u, v)$ 的坐标
- Errechnen der Koordinaten aus  $\vec{p} + h(u, v) \cdot n(u, v)$  im Tangentialraum

Eigenschaften:

- Pro: Ändert Geometrie, dadurch korrekte Silhouette
- Contra: hohe Genauigkeit erfordert hohe Vertex-Dichte

Pro：改变几何图形，从而修正轮廓

缺点：高精度要求高vertex密度

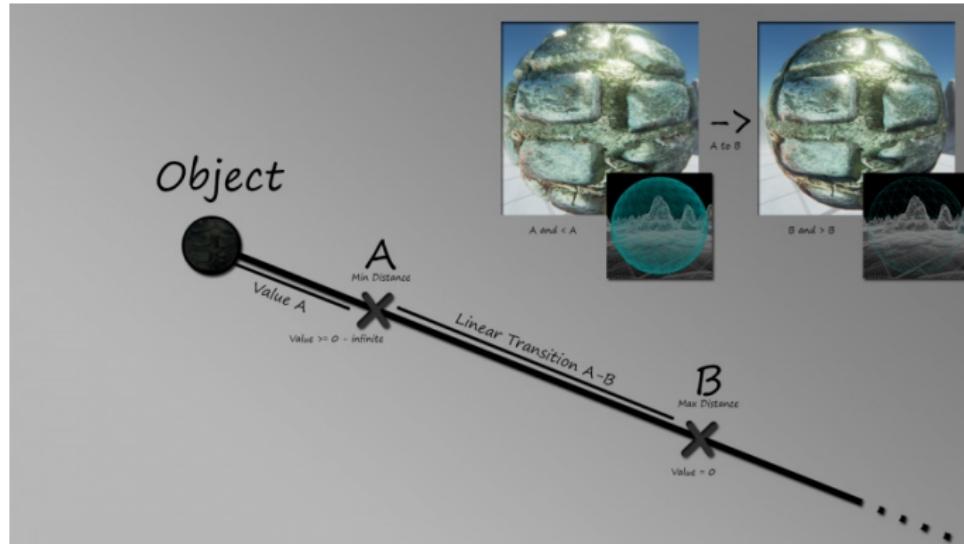
术语“位移”和“高度图”通常可互换使用！

<sup>1</sup> Die Begriffe Displacement und Height Map werden oft austauschbar verwendet!

# Tessellation

Unter **Tessellation** versteht man das (adaptive) Unterteilen und Verfeinern der Geometrie. In Kombination mit Höhenfeldern, kann so stufenweise der Detailgrad erhöht werden.

曲面细分是几何体的（自适应）细分和细化。结合高度场，可以逐渐增加细节程度。



Quelle: <https://wiki.unrealengine.com/>

# Parallax Mapping

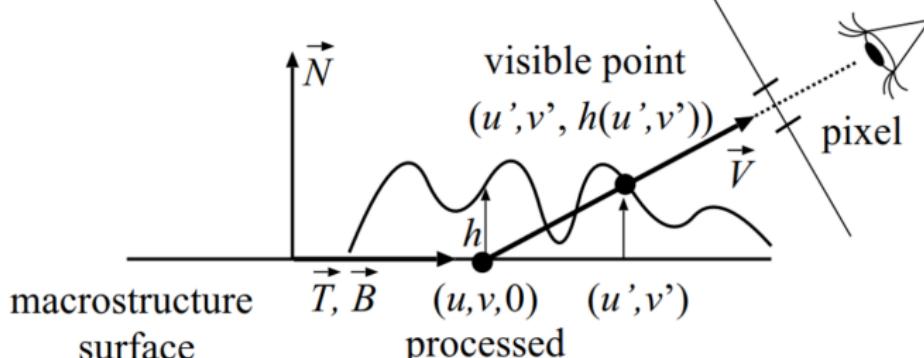
Beim **Parallax Mapping** wird der *sichtbare Punkt* ausgehend von der Kamera berechnet.

视差贴图计算来自摄像机的可见点。

每像素：高度贴图作为像素/片段着色器中的输入；几何形状没有变化

- **Per-Pixel:** Height Map als Eingabe in Pixel/Fragment Shader; Keine Veränderung der Geometrie mehr möglich 切线空间  $(u, v, 0)$  中像素的坐标可能与可见点  $(u', v', h(u', v'))$  的坐标不同
- Koordinaten des Pixels im Tangentialraum  $(u, v, 0)$  möglicherweise ungleich Koordinaten des sichtbaren Punktes  $(u', v', h(u', v'))$
- Bestimmen des ersten Schnittpunktes zwischen Sichtstrahl  $\vec{V}$  und Höhenfunktion:  $(u', v', h(u', v')) = (u, v, 0) + t \cdot \vec{V}$

确定视觉光束 $V$ 和高度函数之间的第一个交点： $(u', v', h(u', v')) = (u, v, 0) + t \cdot V$



# Eigenschaften Parallax Mapping

Im Vergleich zu Displacement Mapping: 与位移映射相比：

- Benötigt keine hohe Geometriekomplexität 不需要高几何复杂度
- Ermöglicht Selbstbeschattung 允许自我着色

Parallax Mapping hat ebenfalls Nachteile: 视差映射也有缺点：

- Suche ähnlich Per-Pixel Raytracing, deshalb sehr langsam
- Lösung nicht eindeutig
- Texturkoordinaten  $(u', v')$  können außerhalb der Textur liegen
- Z-Test verwirft Pixel/Fragment ggf. vorzeitig
- Das Ersetzen von  $(u, v, 0)$  durch  $(u', v', h(u', v'))$  hat keinen Einfluss auf Tiefenpuffer

搜索类似于每像素光线跟踪，所以非常慢

解决方法不明确

纹理坐标  $(u', v')$  可以在纹理之外

如有必要，Z-Test会过早丢弃像素/片段

用  $(u', v', h(u', v'))$  替换  $(u, v, 0)$  对深度缓冲区没有影响

## 2.5 Environment Mapping

2.5环境映射

# Umgebungstexturen

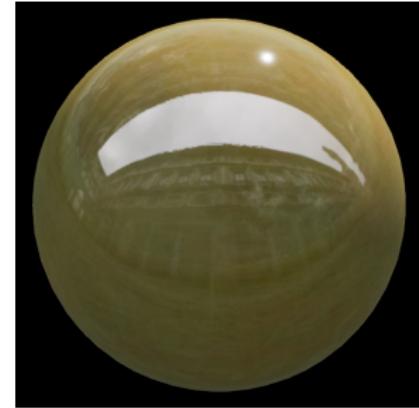
Bisher: Diffuse Oberflächen; Hauptanteil des Lichts wird diffus reflektiert oder absorbiert.

到目前为止：漫射表面；光的主要部分被漫反射或吸收。

Reflektierende Oberflächen: Großteil des eintreffenden Lichts wird zurückgeworfen.

反射表面：大部分入射光被反射回来。

- Laufzeit-Verfahren: Raytracing, Photon-Tracing, ...; sehr aufwändig!
- Environment Mapping: Projektion einer Textur der Umgebung auf Geometrie  
运行时方法：光线跟踪，光子跟踪，.....；很贵！  
环境映射：将环境纹理投影到几何体上



# Umgebungstexturen

环境纹理或环境贴图描绘了环境。

假设：

Umgebungstexturen oder **Environment Maps** bilden die Umgebung ab.  
Annahmen:

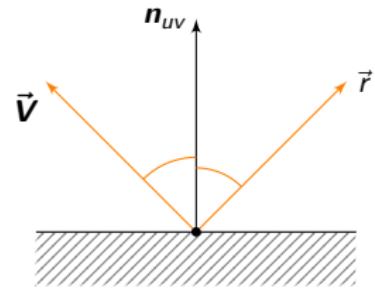
- Umgebung besteht aus Objekten und Lichtquellen, die weit vom zu rendernden Objekt entfernt sind
- Objekt ist konvex; kann also nur Umgebung und nicht sich selbst reflektieren.

环境由远离要渲染的对象的对称的对称和光源组成  
物体是凸的；所以它只能反映环境而不是视图本身。

# Reflexionsberechnung

设表面的点p被赋予纹理坐标 (u, v), 从而赋予法线向量n<sub>uv</sub>。反射向量根据规则计算：

Einem Punkt p einer Oberfläche seien die Texturkoordinaten (u, v) und dadurch der Normalenvektor  $\mathbf{n}_{uv}$  zugeordnet. Der Reflexionsvektor berechnet sich nach der Vorschrift:



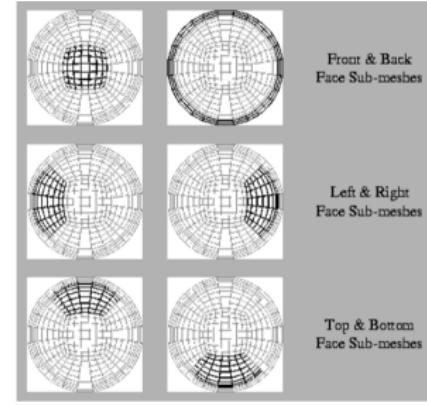
$$\vec{r} = \vec{v} + 2((\mathbf{n}_{uv} \cdot \vec{v})\mathbf{n}_{uv} - \vec{v}) = 2(\mathbf{n}_{uv} \cdot \vec{v})\mathbf{n}_{uv} - \vec{v}$$

# Sphärisches Environment Mapping

Erster Algorithmus eingeführt von Blinn und Newell [2]. Mehrere Wege der Erzeugung: Blinn和Newell [2]引入的第一种算法。 多种生产方式：

- Fotografie einer spiegelnden Kugel
- Kamera mit speziellem Objektiv
- Raytracing
- Perspektivisches Verzerren von Bildern aus 6 Richtungen

镜面地球的照片  
相机配特殊镜头  
光线追踪  
从6个方向的图像的透视畸变



# Sphärisches Environment Mapping

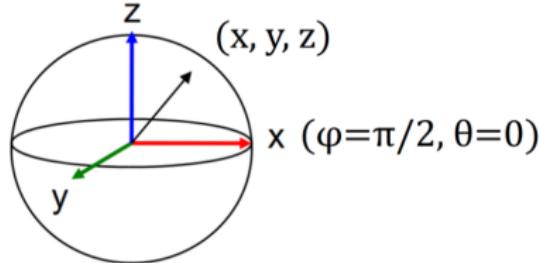
Da sphärisches Mapping die Umgebung auf eine Kugeloberfläche projiziert, rechnen in sphärischen Koordinaten: 由于球形映射将环境投影到球面上，因此球面坐标计算：

- Transformiere den Reflexionsvektor  $\vec{r}$  in Polarkoordinaten  $(\phi, \theta)$ :

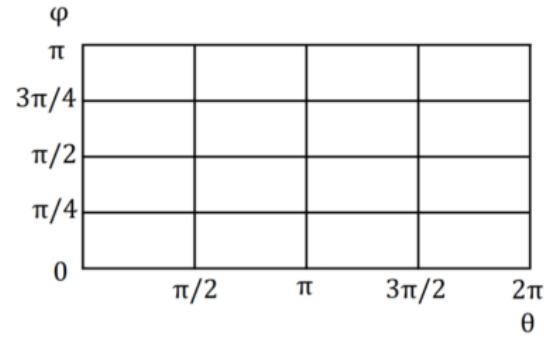
$$\begin{aligned}\phi &= \sin^{-1}(z) \in [0, \pi] && \text{将反射向量r转换为极坐标} \\ \theta &= \tan^{-1}(\frac{y}{x}) \in [0, 2\pi]\end{aligned} \quad ( , ) :$$

- Normalisiere  $(\phi, \theta)$ , so dass  $\phi, \theta \in [0, 1]$ .
- Verwende  $(\phi, \theta)$  als Eingabe für  $(u, v)$

$$(\varphi=0, \theta=0)$$



$$(\varphi=\pi/2, \theta=0)$$



# Kubisches Environment Mapping

由Ned Greene [3]介绍，投影在立方体而不是球上，然后展开。

Eingeführt von Ned Greene [3], Projektion auf Würfel anstatt Kugel, anschließendes Abrollen.

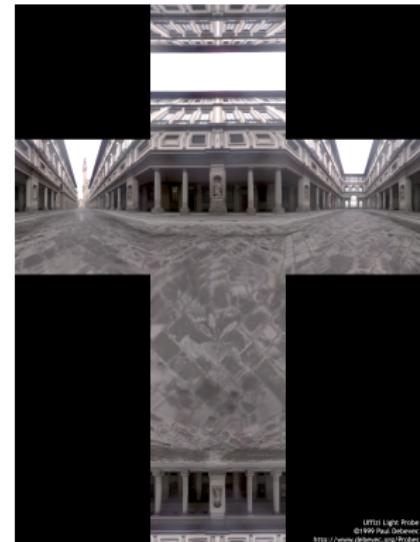
Erzeugung:

- Abfotografieren in 6 Richtungen
- Algorithmisch durch Rendern der Szene in 6 Richtungen

生产：

拍摄6个方向

通过在6个方向上渲染场景来实现算法



# Kubisches Environment Mapping

Annahmen für kubisches Environment Mapping: 立方环境映射的假设：

- Reflexionsvektor  $\vec{r}$  ist normiert 反射向量  $r$  被归一化
- Seiten des Würfels sind achsenparallel 立方体的两侧是轴平行的
- Zentrum des Würfels entspricht Nullvektor 立方体的中心对应于零矢量

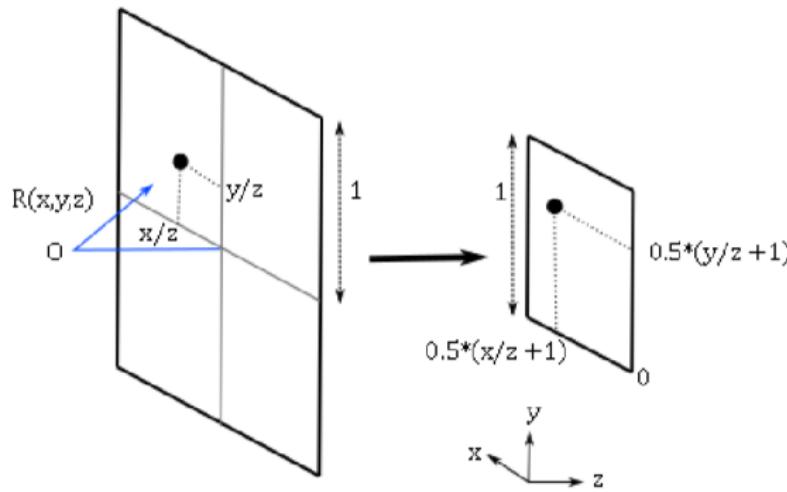
Bestimmen der korrekten “Seite” des Würfels:

- Durch Normierungsannahme: Betragsmäßig größte Komponente gibt Richtung an 确定多维数据集的正确“边”：
- Bsp.:  $\vec{r} = (0.6, 0.3, 0.8)^T$  通过归一化假设：最大分量表示方向
- Z-Richtung, weil  $|\vec{r}_z| > |\vec{r}_x| > |\vec{r}_y|$   $ry |$  示例： $r = (0.6, 0.3, 0.8)^T$
- positive Z-Richtung, weil  $\vec{r}_z > 0$  正Z方向，因为  $rz > 0$

# Kubisches Environment Mapping

Berechnen der Texturkoordinate  $(u, v)$  aus den verbleibenden beiden Komponenten: 从剩余的两个组件计算纹理坐标  $(u, v)$  :

- In unserem Beispiel:  $u = \frac{\vec{r}_x}{\vec{r}_z}$ ,  $v = \frac{\vec{r}_y}{\vec{r}_z}$ ;  $u, v \in [-1, 1]$
- Normalisiere, so dass  $u, v \in [0, 1]$



# Ende der Vorlesung

Ende

# Quellen und Literaturverweise

- [1] J. F. Blinn. Models of light reflection for computer synthesized pictures. In *ACM SIGGRAPH Computer Graphics*, volume 11, pages 192–198. ACM, 1977.
- [2] J. F. Blinn and M. E. Newell. Texture and reflection in computer generated images. *Communications of the ACM*, 19(10):542–547, 1976.
- [3] N. Greene. Applications of world projections. In *Proceedings of Graphics Interface'86*, pages 108–114, 1986.
- [4] B. T. Phong. Illumination for computer generated pictures. *Communications of the ACM*, 18(6):311–317, 1975.  
[http://www.cs.northwestern.edu/~ago820/cs395/Papers/Phong\\_1975.pdf](http://www.cs.northwestern.edu/~ago820/cs395/Papers/Phong_1975.pdf).