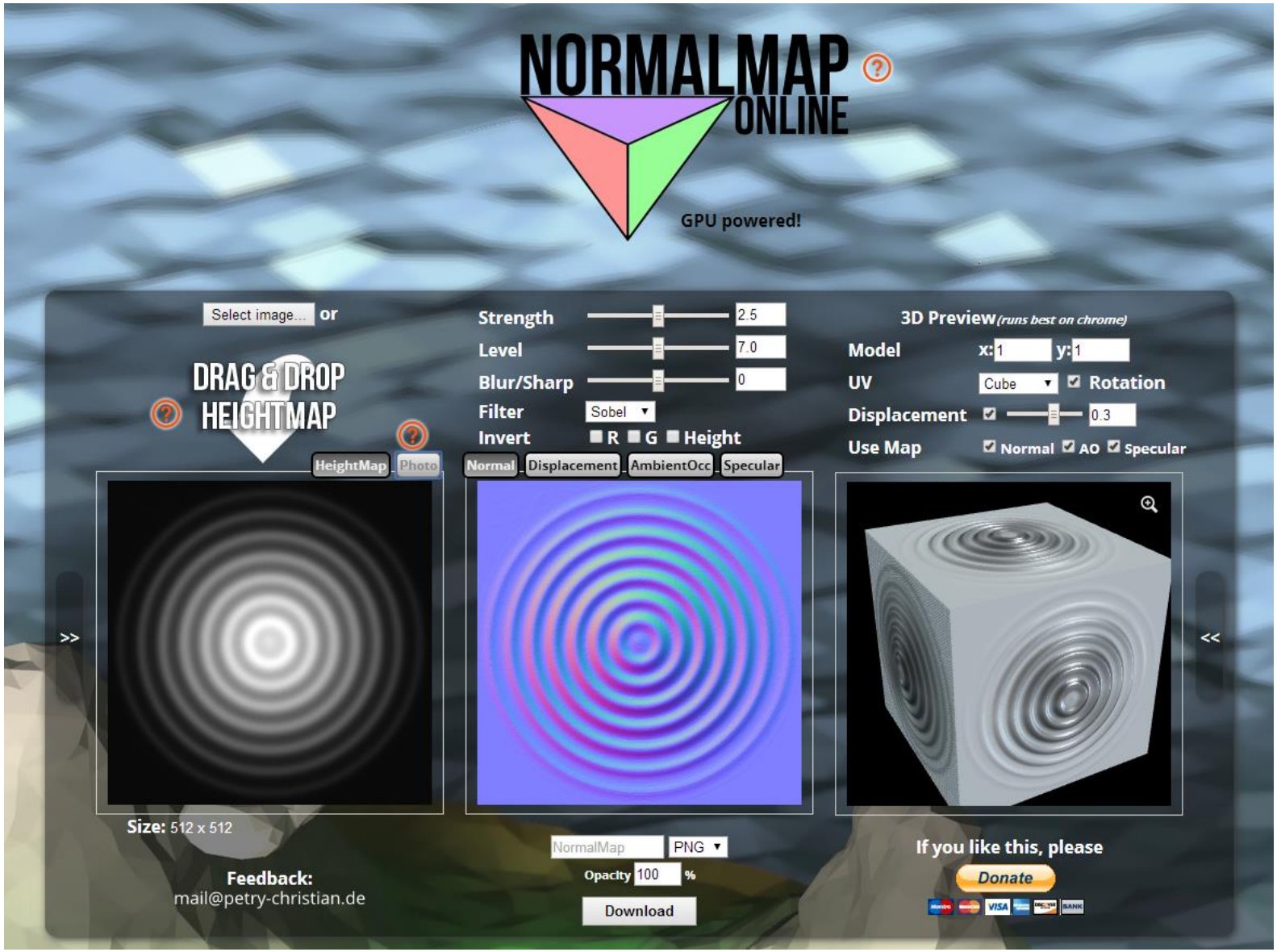
Übung: Interaktive Computergrafik

Blatt 1

- Gouraud Shading (Per-Vertex Lighting)
- Phong Shading (Per-Pixel Lighting)
- Normal Mapping (mit “Fake Tangent Space”!)
- Gamma Correction

Quellen für Normal Maps

- Höhenfelder / “Bump Maps”

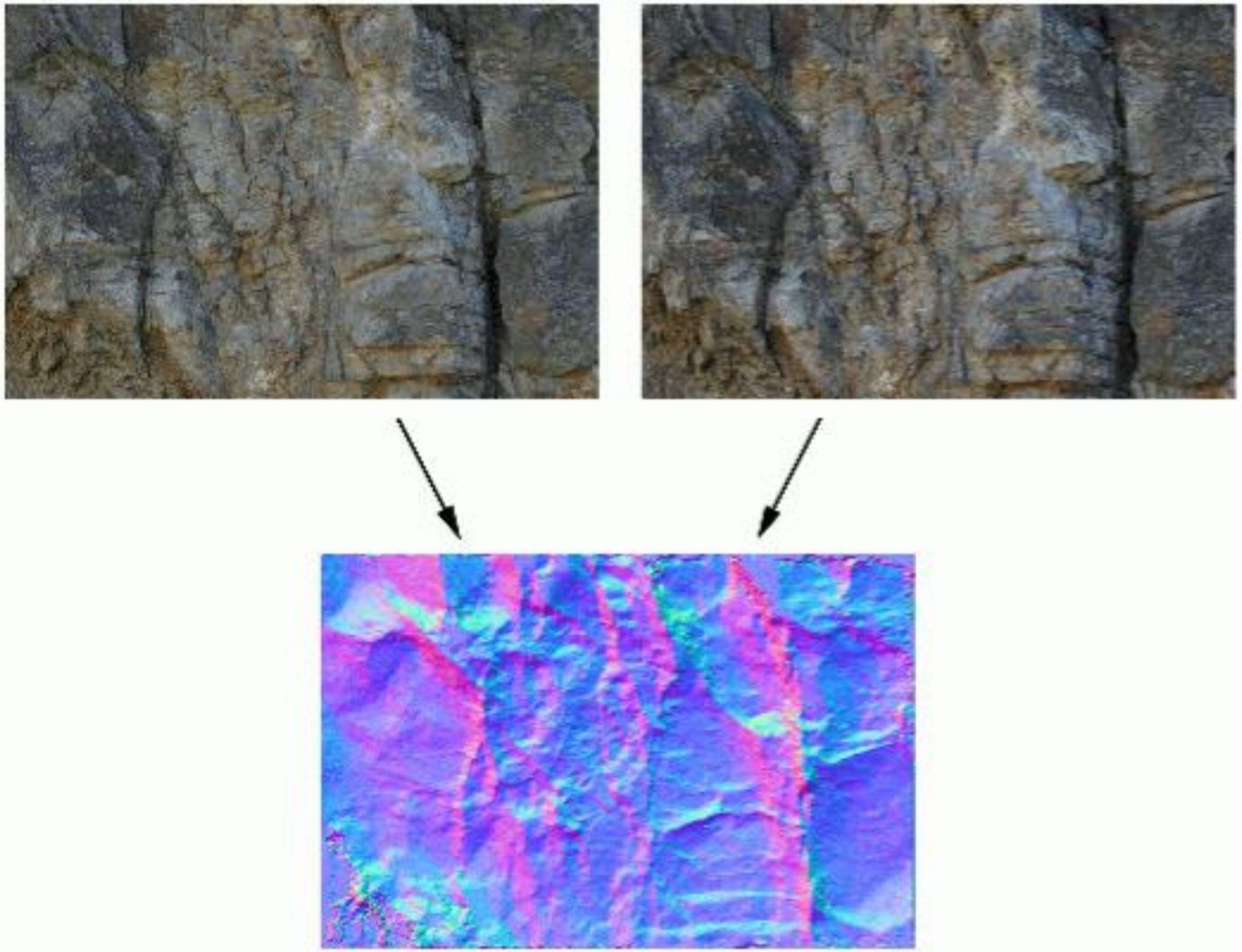


Quellen für Normal Maps

- High-Poly Modelle (Sculpting Tools)

Quellen für Normal Maps

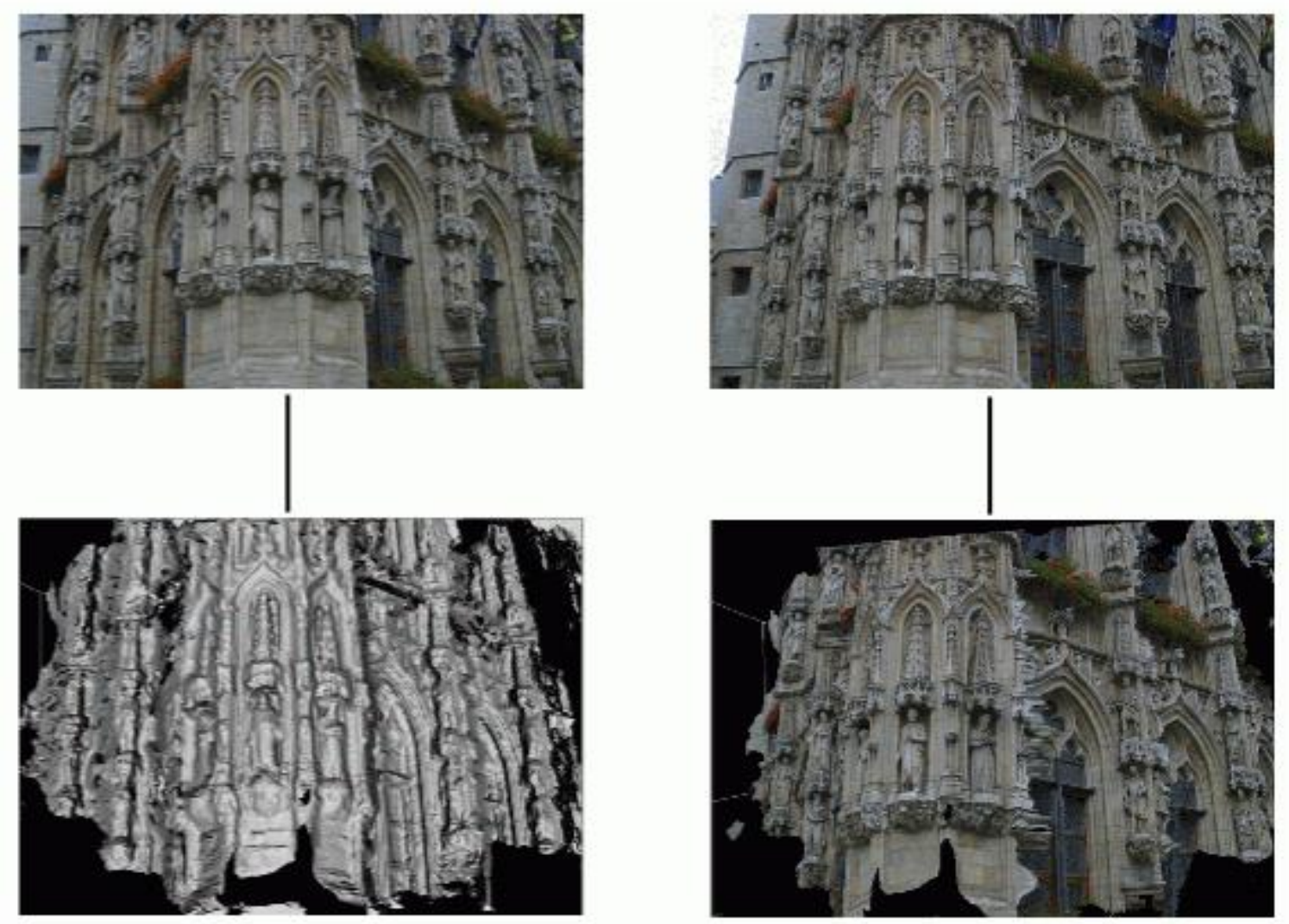
- Stereofotos



http://docs.cryengine.com/display/SDKDOC2/Photobump

Quellen für Normal Maps

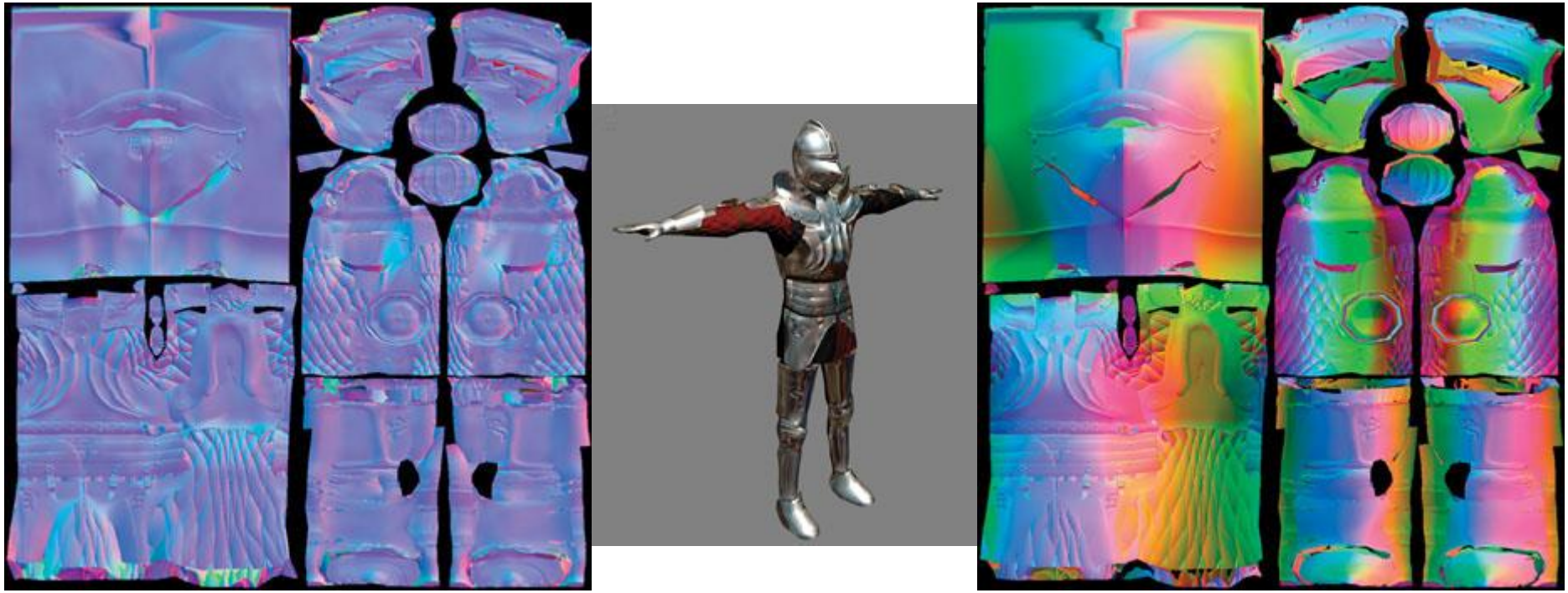
- Stereofotos



http://docs.cryengine.com/display/SDKDOC2/Photobump

Tangent Space vs. Object Space

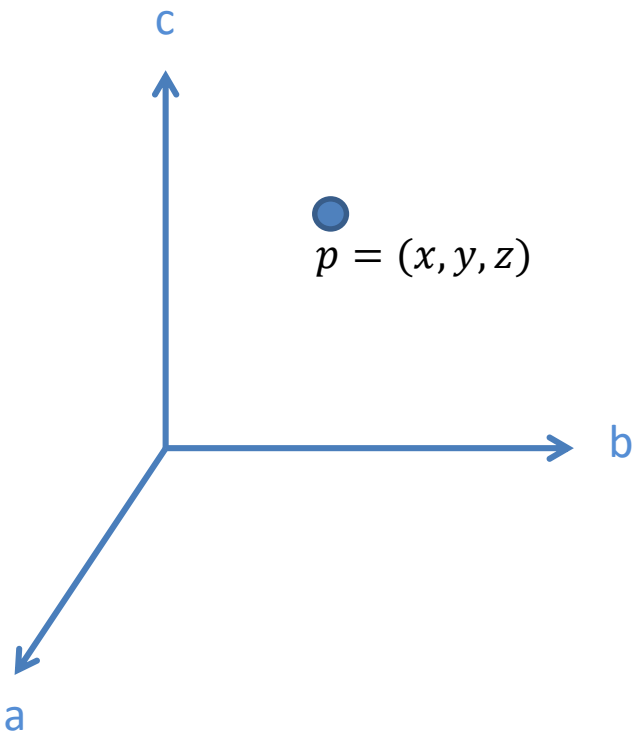
- Gleiche Information, anderes Koordinatensystem



Anschauung Co-Frame

- $M = (a \mid b \mid c)$ transformiert von Objektraum in Weltraum
  - $p' = M\,p = x\,a + y\,b + z\,c$

- $M^{-1} = (\mathrm{co{-}a} \mid \mathrm{co{-}b} \mid \mathrm{co{-}c})^T$ transformiert von Welt- in Objektraum
  - $x = < p', \mathrm{co{-}a} >$
  - $y = < p', \mathrm{co}-b >$
  - $z = < p', \mathrm{co}-c >$

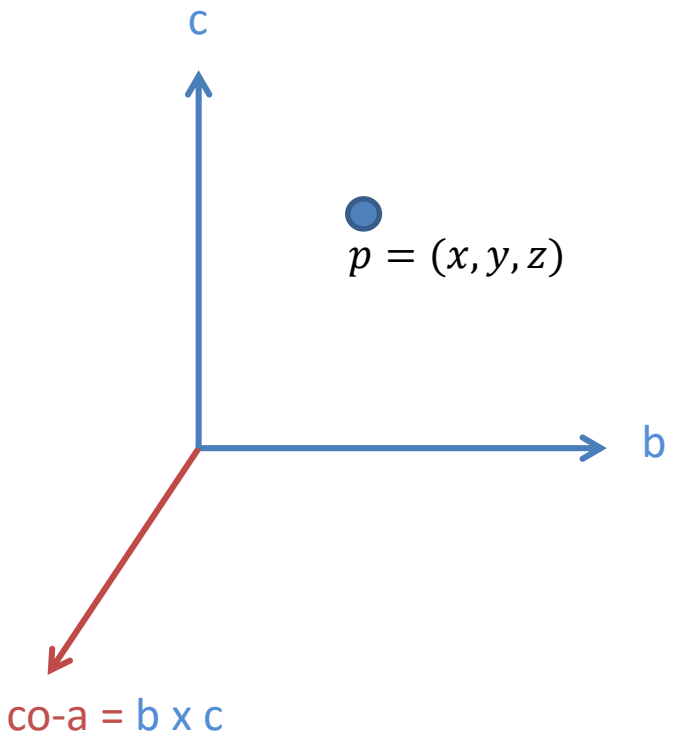


Orthonormale Basis

- $M = (a \mid b \mid c)$ transformiert von Objektraum in Weltraum
  - $p' = M\,p = x\,a + y\,b + z\,c$

- $M^{-1} = (\text{co}-\text{a} \mid \text{co}-\text{b} \mid \text{co}-\text{c})^T$ transformiert von Welt- in Objektraum
  - $x = < p', \text{co}-\text{a} >$
  - $y = < p', \text{co}-b >$
  - $z = < p', \text{co}-c >$

- Spezialfall: **Orthonormale Basis** a, b, c:
  - $< a, \text{co}-\text{a} > = < a, a > = 1$
  - $< b, \text{co}-\text{a} > = < b, a > = 0$
  - $< c, \text{co}-\text{a} > = < c, a > = 0$

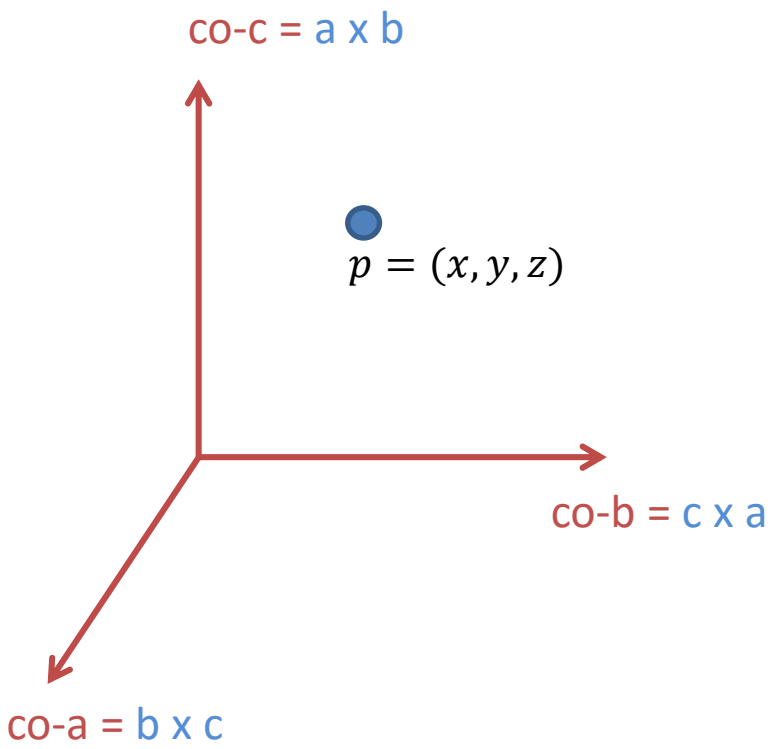


Orthonormale Basis

- $M = (a \mid b \mid c)$ transformiert von Objektraum in Weltraum
  - $p' = M\,p = x\,a + y\,b + z\,c$

- $M^{-1} = (a \mid b \mid c)^{\mathrm{T}}$ transformiert von Weltraum in Objektraum
  - $x = < p', a >$
  - $y = < p', b >$
  - $z = < p', c >$

- Spezialfall: **Orthonormale Basis** a, b, c:
  - $< a, \mathrm{co{-}a} > = < a, a > = 1$
  - $< b, \mathrm{co{-}a} > = < b, a > = 0$
  - $< c, \mathrm{co{-}a} > = < c, a > = 0$

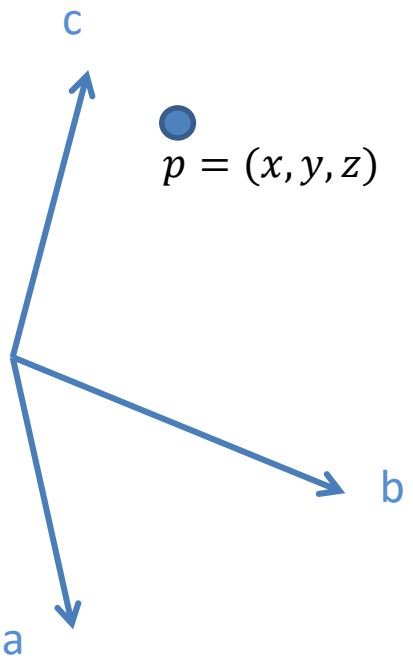


Beliebige Basis

- $M = (a \mid b \mid c)$ transformiert von Objektraum in Weltraum
  - $p' = M\,p = x\,a + y\,b + z\,c$

- $M^{-1} = (\mathrm{co{-}a} \mid \mathrm{co{-}b} \mid \mathrm{co{-}c})^T$ transformiert von Welt- in Objektraum
  - $x = < p', \mathrm{co{-}a} >$
  - $y = < p', \mathrm{co}-b >$
  - $z = < p', \mathrm{co}-c >$

- Allgemein: **Beliebige Basis** a, b, c:

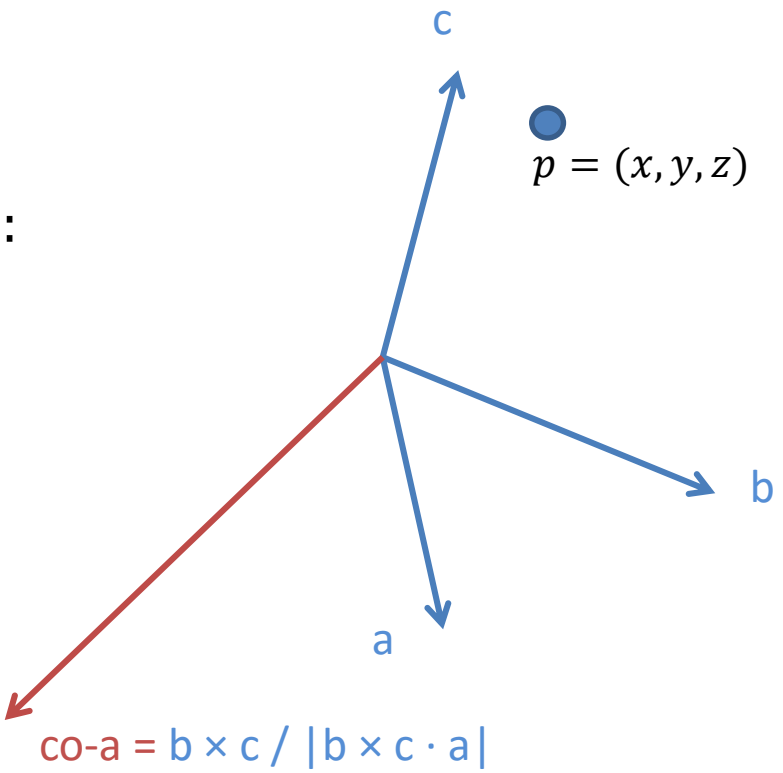


Allgemeines Co-Frame

- $M = (a \mid b \mid c)$ transformiert von Objektraum in Weltraum
  - $p' = M\,p = x\,a + y\,b + z\,c$

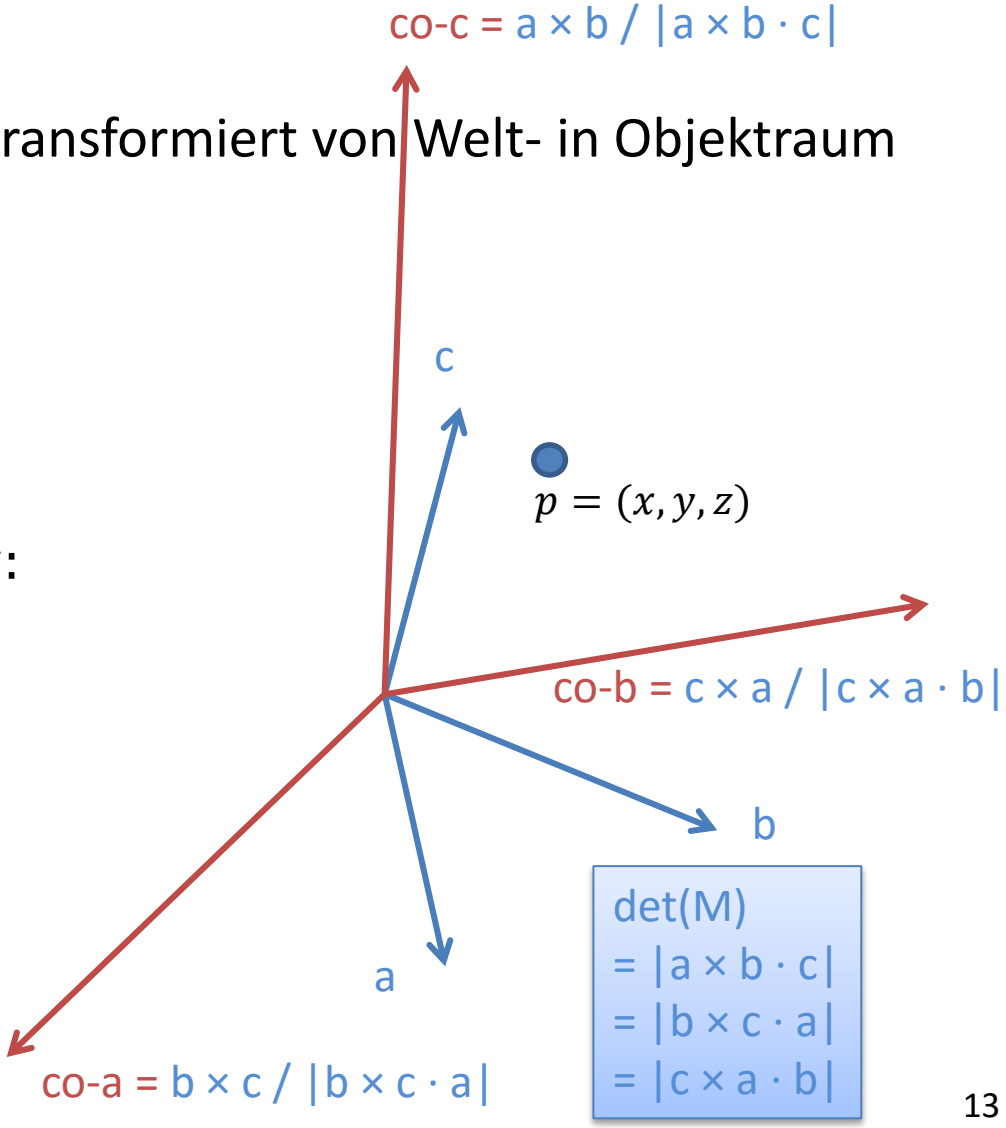
- $M^{-1} = (\mathrm{co{-}a} \mid \mathrm{co{-}b} \mid \mathrm{co{-}c})^T$ transformiert von Welt- in Objektraum
  - $x = < p', \mathrm{co{-}a} >$
  - $y = < p', \mathrm{co{-}}b >$
  - $z = < p', \mathrm{co{-}}c >$

- Allgemein: **Beliebige Basis** a, b, c:
  - $< a, \mathrm{co{-}a} > = 1$
  - $< b, \mathrm{co{-}a} > = 0$
  - $< c, \mathrm{co{-}a} > = 0$



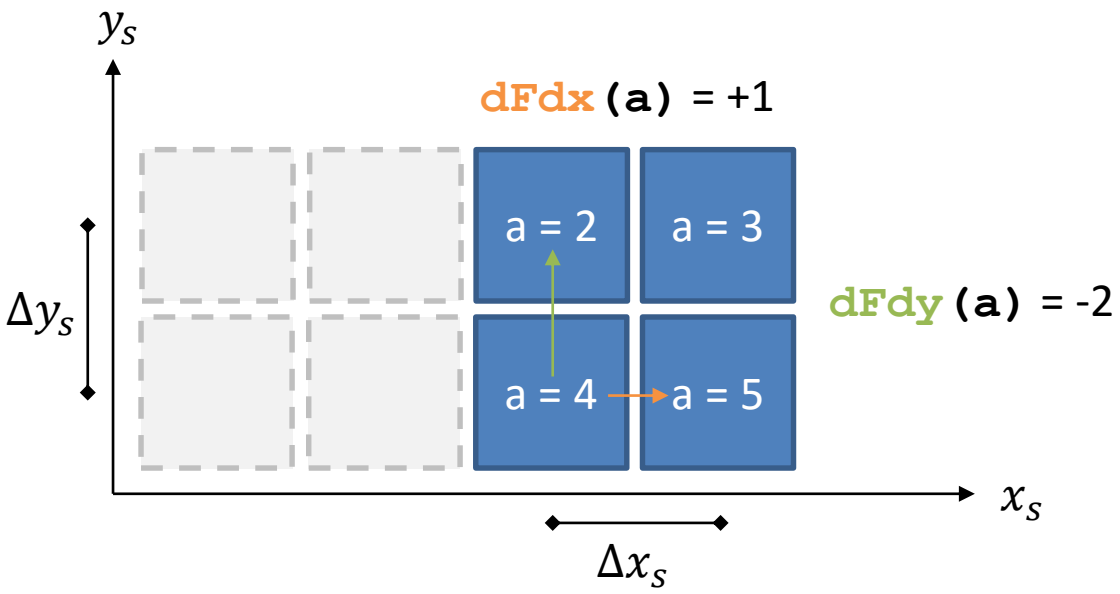
Allgemeines Co-Frame

- $M = (a \mid b \mid c)$ transformiert von Objektraum in Weltraum
  - $p' = M\,p = x\,a + y\,b + z\,c$
- $M^{-1} = (\mathrm{co{-}a} \mid \mathrm{co{-}b} \mid \mathrm{co{-}c})^T$ transformiert von Welt- in Objektraum
  - $x = < p', \mathrm{co{-}a} >$
  - $y = < p', \mathrm{co{-}}b >$
  - $z = < p', \mathrm{co{-}}c >$
- Allgemein: **Beliebige Basis** a, b, c:
  - $< a, \mathrm{co{-}a} > = 1$
  - $< b, \mathrm{co{-}a} > = 0$
  - $< c, \mathrm{co{-}a} > = 0$



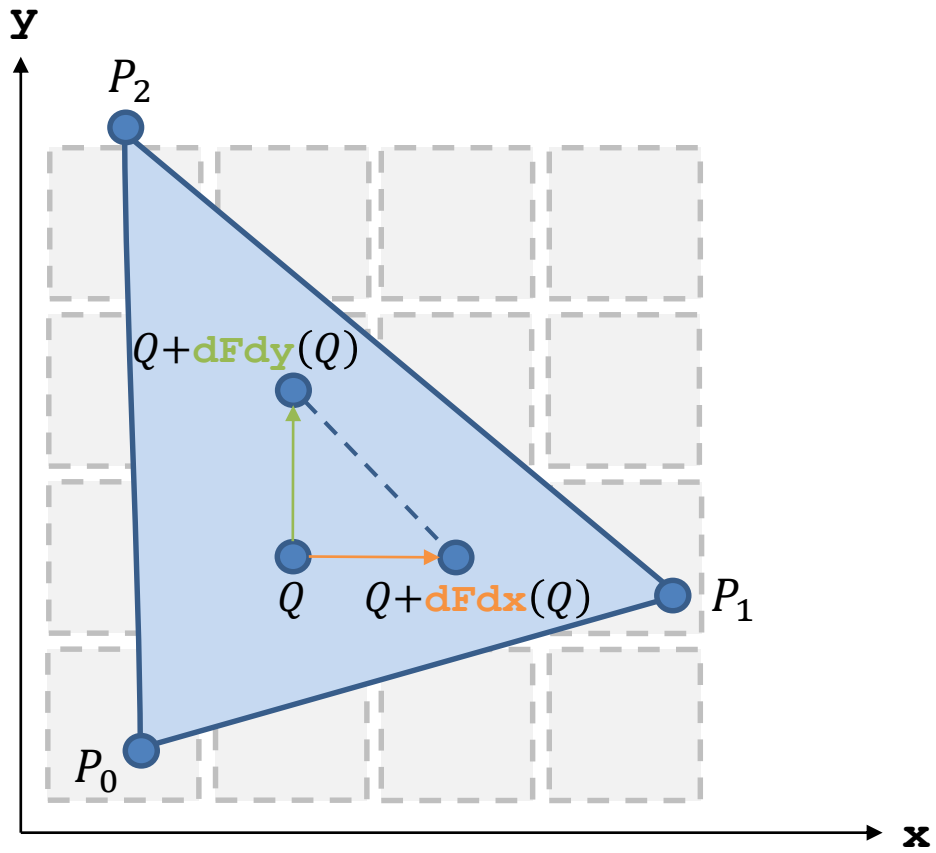
co-Tangentenraum ad hoc im Fragment Shader

- **“Pixel Derivatives”**
  - Ableitung von *Variable* nach Bildschirmkoordinate im Fragment Shader
  - Berechnung Anhand der Differenzen in 2x2 Pixelquadrant
  - **GLSL:** $\texttt{dFdx(a)} = \frac{\partial \mathbf{a}}{\partial x_s}\Delta x_s$, $\texttt{dFdy(a)} = \frac{\partial \mathbf{a}}{\partial y_s}\Delta y_s$
    an Bildposition (x_s, y_s) mit Pixelabstand $(\Delta x_s, \Delta y_s)$



co-Tangentenraum ad hoc im Fragment Shader

- Spanne (Teil-)Dreieck mit Ableitungen entlang der Bildschirmachsen auf
- Löse Gleichungssystem zur Berechnung des co-Tangentenraums
  - Analog zu Gleichungssystem für volles Dreieck!



- $\Delta s = \Delta p \cdot \mathrm{co{-}T}$
  - $\mathtt{dFdx}(s) = \mathtt{dFdx}(Q) \cdot \mathrm{co{-}T}$
  - $\mathtt{dFdy}(s) = \mathtt{dFdy}(Q) \cdot \mathrm{co{-}T}$
- $\Delta t = \Delta p \cdot \mathrm{co{-}B}$
  - $\mathtt{dFdx}(t) = \mathtt{dFdx}(Q) \cdot \mathrm{co{-}B}$
  - $\mathtt{dFdy}(t) = \mathtt{dFdy}(Q) \cdot \mathrm{co{-}B}$
- $N \cdot \mathrm{co{-}T} = 0, \quad N \cdot \mathrm{co{-}B} = 0$

co-Tangentenraum ad hoc im Fragment Shader

- Löse Gleichungssystem zur Berechnung des co-Tangentenraums
- Inverse 3x3-Materix lässt sich mit Kreuzprodukten schreiben
  - Ergebnis erfüllt dann “offensichtlich” alle Gleichungen (Kreuzprodukt → Orthogonalität, Spatprodukt → Normierung)

$$\begin{bmatrix} \mathtt{dFdx}(Q) \\ \mathtt{dFdy}(Q) \\ N \end{bmatrix} \cdot \mathrm{co{-}T} = \begin{pmatrix} \mathtt{dFdx}(s) \\ \mathtt{dFdy}(s) \\ 0 \end{pmatrix}$$

$$\mathrm{co{-}T} = \begin{bmatrix} \mathtt{dFdx}(Q) \\ \mathtt{dFdy}(Q) \\ N \end{bmatrix}^{-1} \begin{pmatrix} \mathtt{dFdx}(s) \\ \mathtt{dFdy}(s) \\ 0 \end{pmatrix}$$

$$\begin{bmatrix} \mathtt{dFdx}(Q) \\ \mathtt{dFdy}(Q) \\ N \end{bmatrix}^{-1} = \frac{1}{(\mathtt{dFdx}(Q) \times \mathtt{dFdy}(Q)) \cdot N} \begin{bmatrix} \mathtt{dFdy}(Q) \times N \\ N \times \mathtt{dFdx}(Q) \\ \mathtt{dFdx}(Q) \times \mathtt{dFdy}(Q) \end{bmatrix}$$

- $\Delta s = \Delta p \cdot \mathrm{co{-}T}$
  - $\mathtt{dFdx}(s) = \mathtt{dFdx}(Q) \cdot \mathrm{co{-}T}$
  - $\mathtt{dFdy}(s) = \mathtt{dFdy}(Q) \cdot \mathrm{co{-}T}$
- $\Delta t = \Delta p \cdot \mathrm{co{-}B}$
  - $\mathtt{dFdx}(t) = \mathtt{dFdx}(Q) \cdot \mathrm{co{-}B}$
  - $\mathtt{dFdy}(t) = \mathtt{dFdy}(Q) \cdot \mathrm{co{-}B}$
- $N \cdot \mathrm{co{-}T} = 0, \quad N \cdot \mathrm{co{-}B} = 0$

co-Tangentenraum ad hoc im Fragment Shader

- Löse Gleichungssystem durch Kreuzprodukte
- Ergebnis erfüllt alle Gleichungen

```
mat3 cotangent_frame( vec3 N, vec3 p, vec2 uv )
{
    // get edge vectors of the pixel triangle
    vec3 dp1 = dFdx( p );
    vec3 dp2 = dFdy( p );
    vec2 duv1 = dFdx( uv );
    vec2 duv2 = dFdy( uv );

    // solve the linear system
    vec3 dp2perp = cross( dp2, N );
    vec3 dp1perp = cross( N, dp1 );
    vec3 T = dp2perp * duv1.x + dp1perp * duv2.x;
    vec3 B = dp2perp * duv1.y + dp1perp * duv2.y;

    // divide by determinant to complete
    float invdet = 1 / dot( dp1, dp2perp );
    return mat3( T * invdet, B * invdet, N );
}
```

adaptiert von http://www.thetenthplanet.de/archives/1180

- $\Delta s = \Delta p \cdot \text{co}-\text{T}$
  - $\texttt{dFdx}(s) = \texttt{dFdx}(Q) \cdot \text{co}-\text{T}$
  - $\texttt{dFdy}(s) = \texttt{dFdy}(Q) \cdot \text{co}-\text{T}$
- $\Delta t = \Delta p \cdot \text{co}-\text{B}$
  - $\texttt{dFdx}(t) = \texttt{dFdx}(Q) \cdot \text{co}-\text{B}$
  - $\texttt{dFdy}(t) = \texttt{dFdy}(Q) \cdot \text{co}-\text{B}$
- $N \cdot \text{co}-\text{T} = 0, \quad N \cdot \text{co}-\text{B} = 0$

co-Tangentenraum ad hoc im Fragment Shader

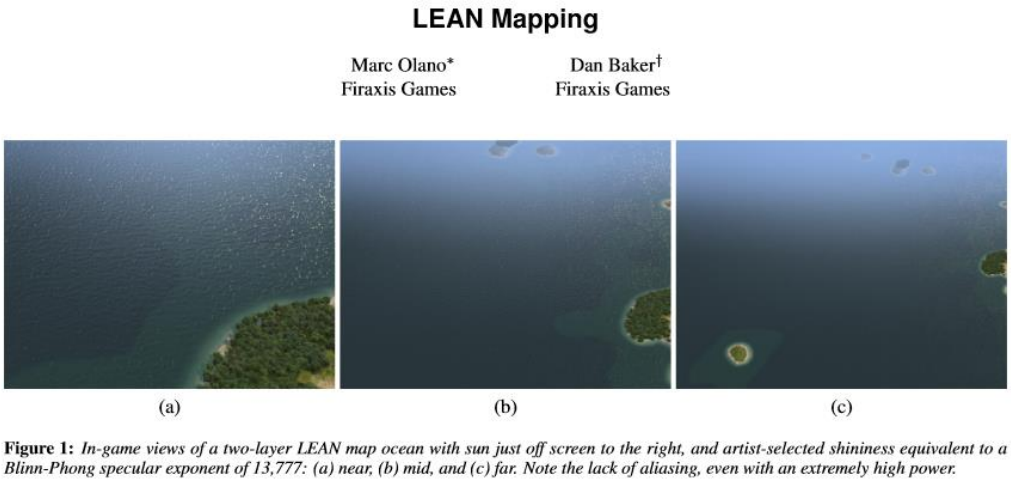
- Manchmal skalierungsinvarianter co-Tangentenraum gefragt
  - **Normal Mapping:** Übertrage Normalen aus Normal Map ohne orthogonale geom. Streckung/Stauchung (erhalte Kontrast/Profil)

```
mat3 cotangent_frame( vec3 N, vec3 p, vec2 uv )
{
    // get edge vectors of the pixel triangle
    vec3 dp1 = dFdx( p );
    vec3 dp2 = dFdy( p );
    vec2 duv1 = dFdx( uv );
    vec2 duv2 = dFdy( uv );

    // solve the linear system
    vec3 dp2perp = cross( dp2, N );
    vec3 dp1perp = cross( N, dp1 );
    vec3 T = dp2perp * duv1.x + dp1perp * duv2.x;
    vec3 B = dp2perp * duv1.y + dp1perp * duv2.y;

    // construct a scale-invariant frame
    float invmax = inversesqrt( max( dot(T,T), dot(B,B) ) );
    return mat3( T * invmax, B * invmax, N );
}
```

siehe http://www.thetenthplanet.de/archives/1180

- $\Delta s = \Delta p \cdot \text{co}-\text{T}$
  - $\texttt{dFdx}(s) = \texttt{dFdx}(Q) \cdot \text{co}-\text{T}$
  - $\texttt{dFdy}(s) = \texttt{dFdy}(Q) \cdot \text{co}-\text{T}$
- $\Delta t = \Delta p \cdot \text{co}-\text{B}$
  - $\texttt{dFdx}(t) = \texttt{dFdx}(Q) \cdot \text{co}-\text{B}$
  - $\texttt{dFdy}(t) = \texttt{dFdy}(Q) \cdot \text{co}-\text{B}$
- $N \cdot \text{co}-\text{T} = 0, \quad N \cdot \text{co}-\text{B} = 0$

Nächste Woche: Besprechung LEAN Mapping



LEAN Mapping

Marc Olano*
Firaxis Games

Dan Baker†
Firaxis Games

(a) (b) (c)

Figure 1: *In-game views of a two-layer LEAN map ocean with sun just off screen to the right, and artist-selected shininess equivalent to a Blinn-Phong specular exponent of 13,777: (a) near, (b) mid, and (c) far. Note the lack of aliasing, even with an extremely high power.*

Marc Olano and Dan Baker: LEAN Mapping
http://www.csee.umbc.edu/~olano/papers/lean/