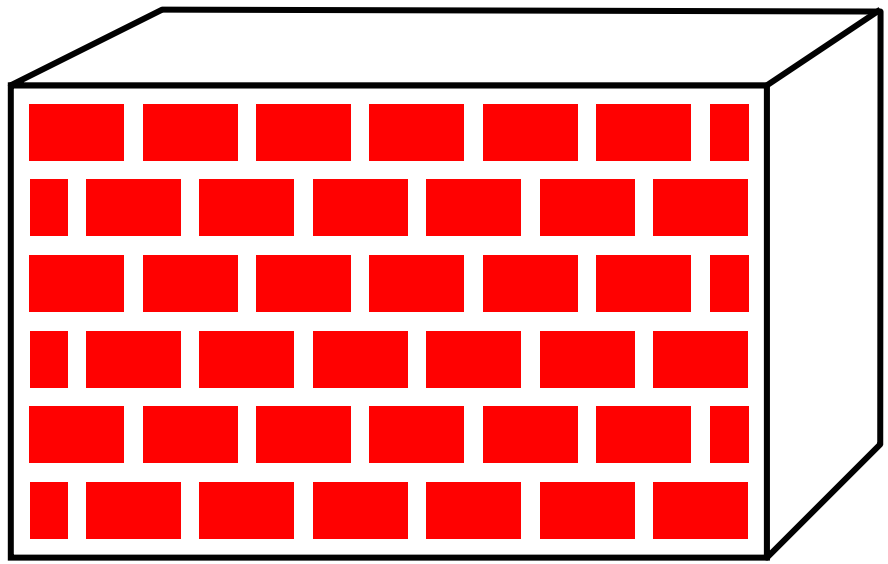
Kapitel 19:
Texturing

strukturierte Fläche

Beispiel: Steinmauer

lege viele rote Rechtecke
auf ein großes weißes Rechteck:



Nachteil: aufwändige Geometrie

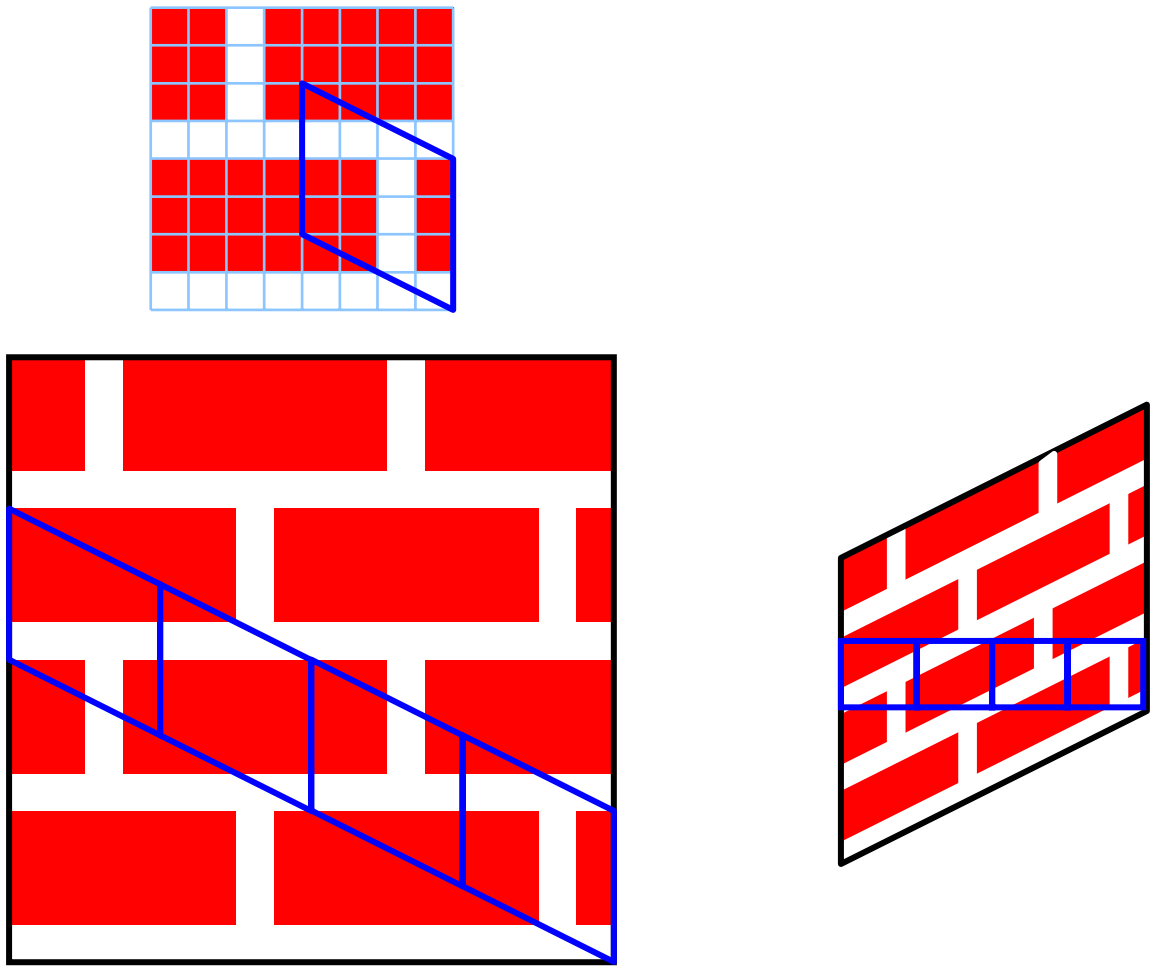
Pixel statt Geometrie

Lösung: Bild auf Objekt legen

genauer: beim Rastern einer Scanline wird 2-dimensionale Pixelmatrix eingearbeitet.

Materialfarbe wird ersetzt und/oder mit Textur kombiniert

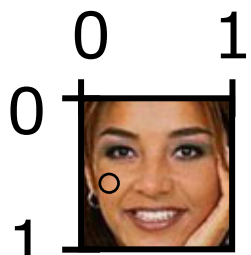
Texture Mapping

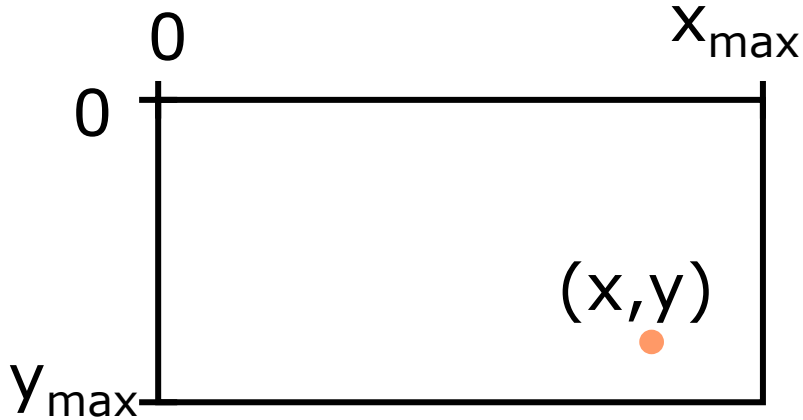


Zugriff auf Texture Map

Inverse Projektion ergibt (x,y) := $g^{-1}(x',y',z')$

Faktoren f_x, f_y





$$u = \frac{x}{x_{max}} \cdot f_x \quad \frac{320}{400} \cdot 4 = 3.2$$

$$v = \frac{y}{y_{max}} \cdot f_y \quad \frac{160}{200} \cdot 2 = 1.6$$

bei 16×16 Textur T:

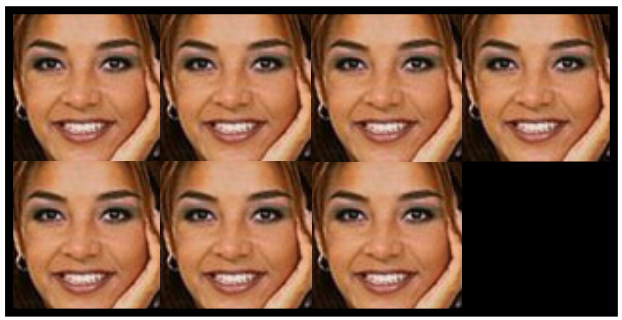
T[16*0.2][16*0.6]

Phasen des Texture Mapping

- Raumkoordinaten des Flächenpunktes berechnen
- Abbildung in den Parameterraum durchführen
- Texturkoordinaten berechnen
- Texturwerte anpassen
- Erscheinungsbild mit dem Texturwert modifizieren

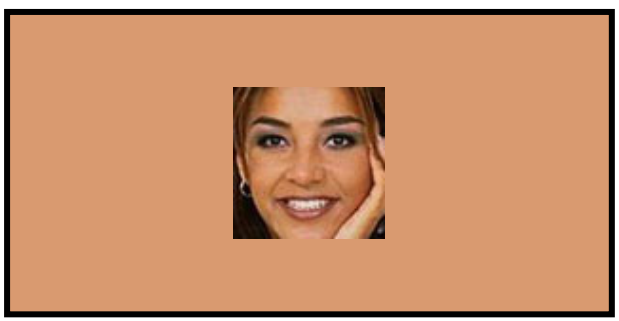
Korrespondenzfunktion

repeat

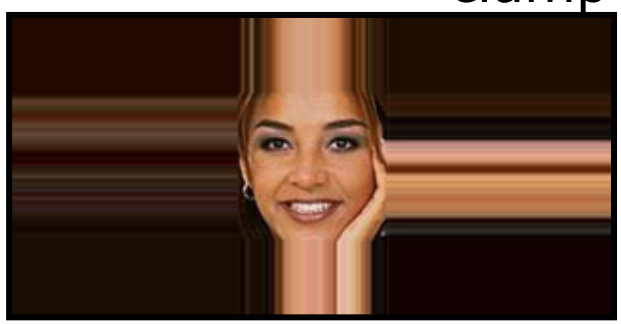


mirror

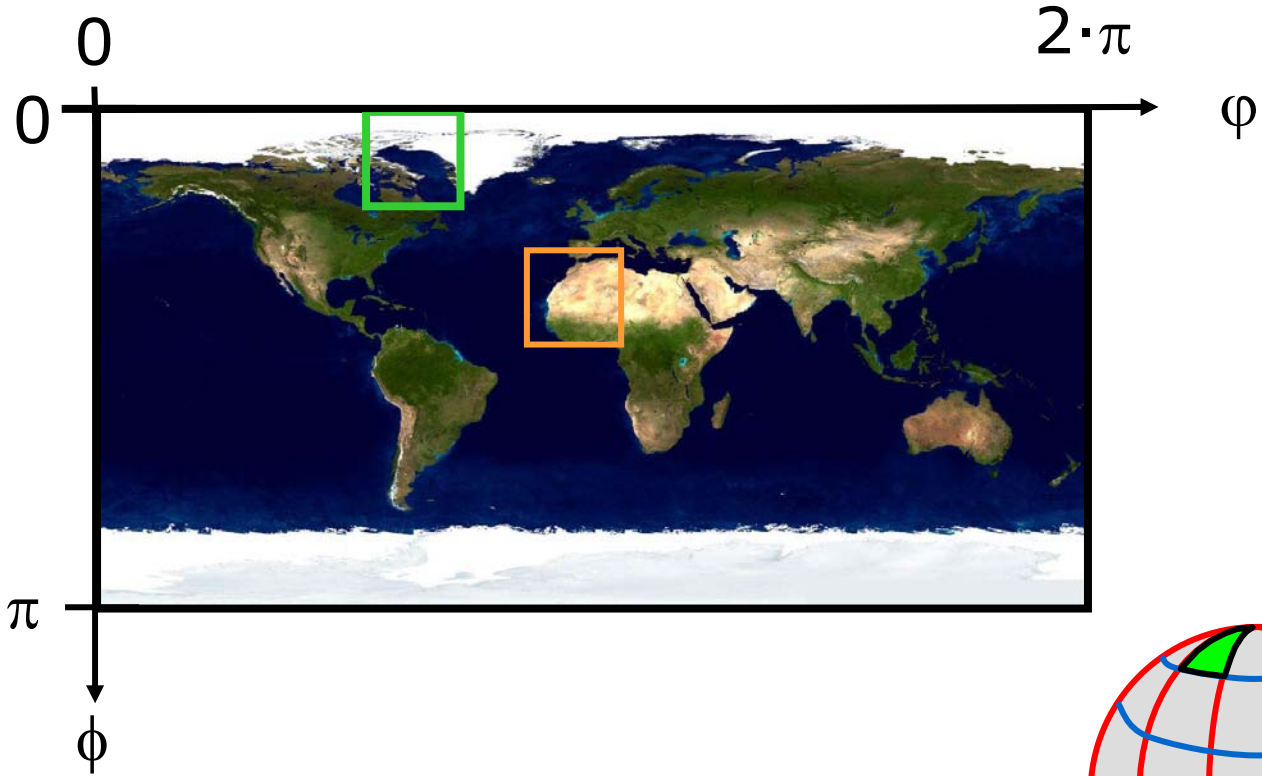
border



clamp



Sphärische Projektion

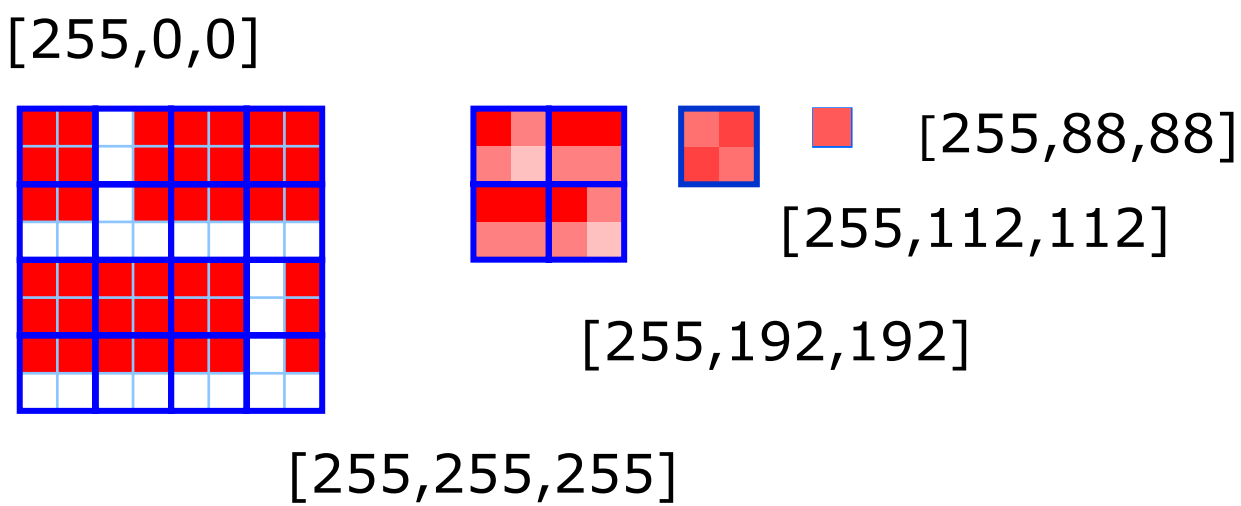


$[\sin(\phi)\cdot \cos(\varphi), \sin(\phi)\cdot \sin(\varphi), \cos(\phi), 1]$

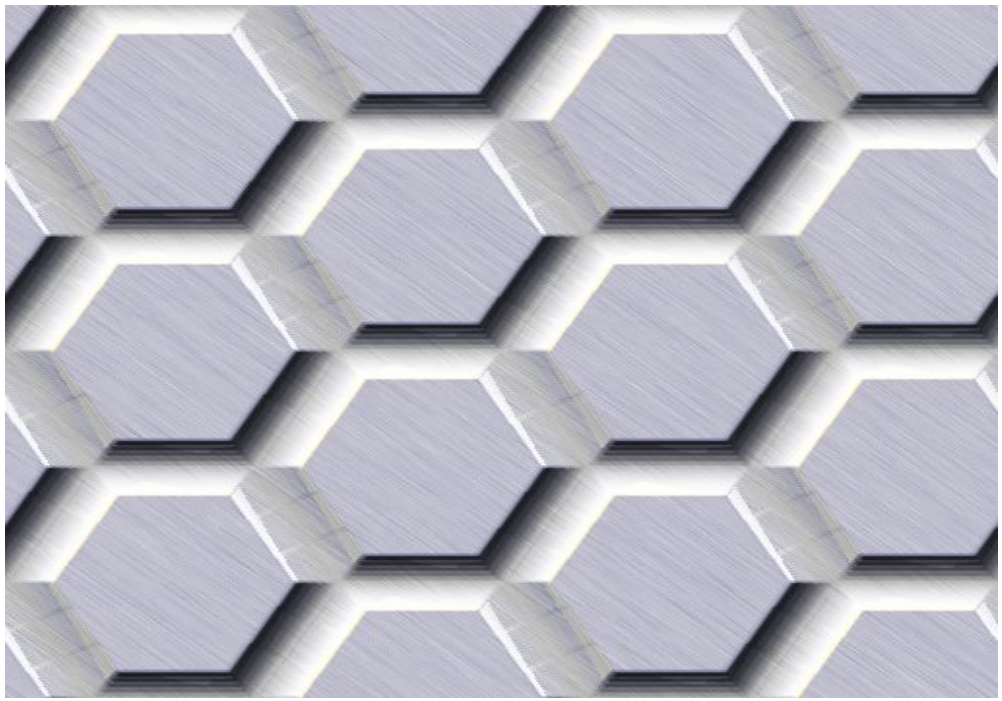
Mip mapping

(multum in parvo mapping)

halte verschiedene Textur-Auflösungen vor für verschiedene level of detail (LOD)



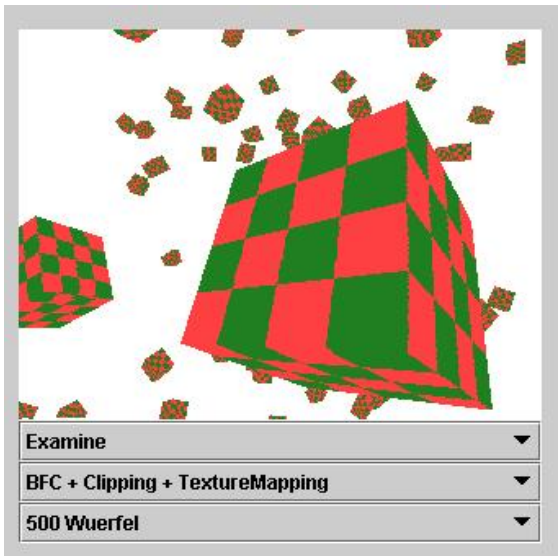
Gallery

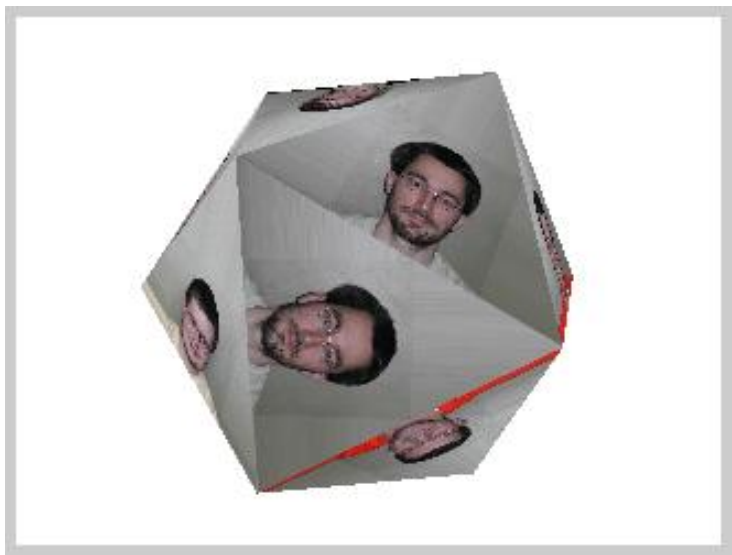


http://www.texturemaker.com/gallery.htm

Java-Applet zu Texturen

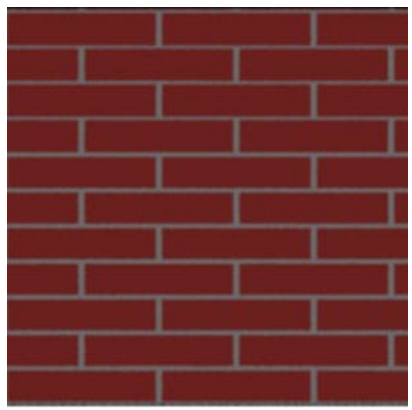
~cg/2004/skript/node170.html



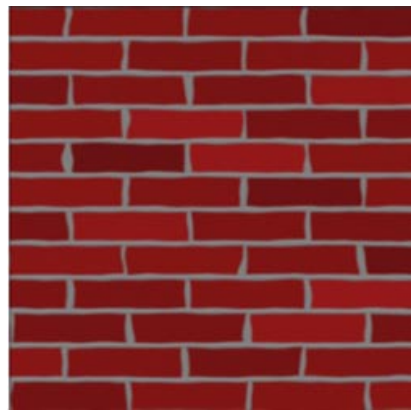


~cg/2004/skript/node171.html

Prozedurale Textur



statisch



mit Störungen

Shadow Map

Schatten vorberechnen
und in Textur ablegen

$$C_{gesamt,diffus}[x,y] = C_{lighting,diffus}[x,y]$$

$$* \; ShadowMap[u(x,y), v(x,y)]$$

Alpha Mapping

Textur enthält Alphawerte

0	völlig durchsichtig
0 < x <255	teilweise durchsichtig
255	undurchsichtig

Baum als Kreisfläche mit Löchern

Alpha Mapping Implementation

$$C_{gesamt,alpha}[x,y] =$$

$$C_{Baum,lighting}[x,y] * AlphaMap[u(x,y),v(x,y)]$$

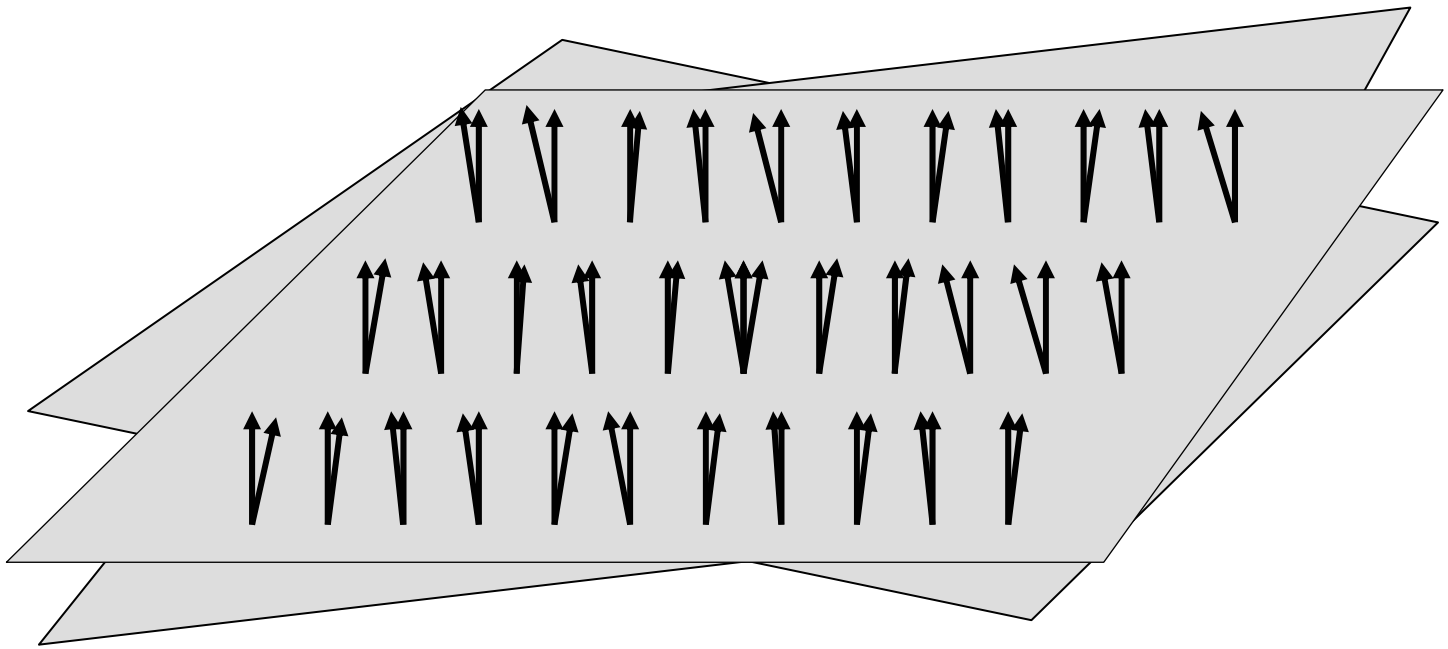
$$+ \; C_{Hintergrund,lighting} * (1 - AlphaMap[u(x,y),v(x,y)])$$

Environment Mapping

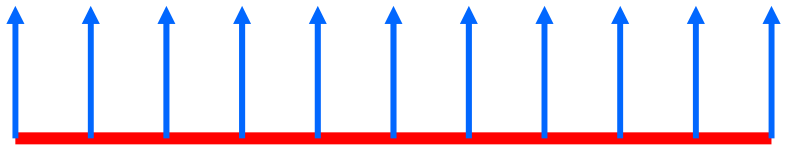
Textur enthält Projektion der Umgebung

Zugriff abhängig vom Augenpunkt

Modifikation der Normalen

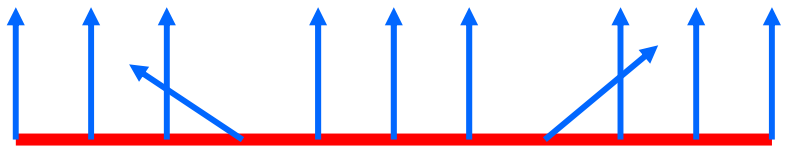


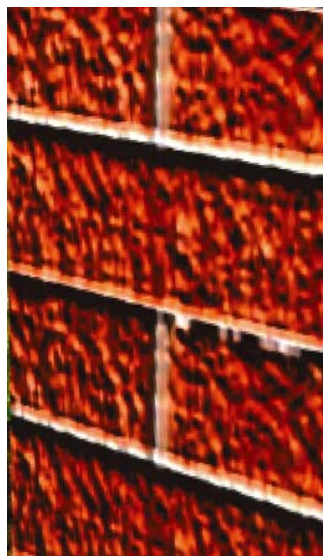
Bump Mapping



kombinierbar
mit Textur:

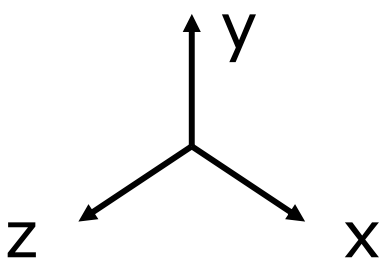
modifiziere Normalenvektor





Simulation von Unebenheit:

Bump Mapping Matrix




```
[10,8] [-4,6]  ...
[-4,3] [ 7,9]  ...
 ...     ...   ...
```

Die Einträge werden zu den x und z-Werten der Normale addiert

Obacht:
die suggerierten Höhendifferenzen sind von der Seite nicht sichtbar !

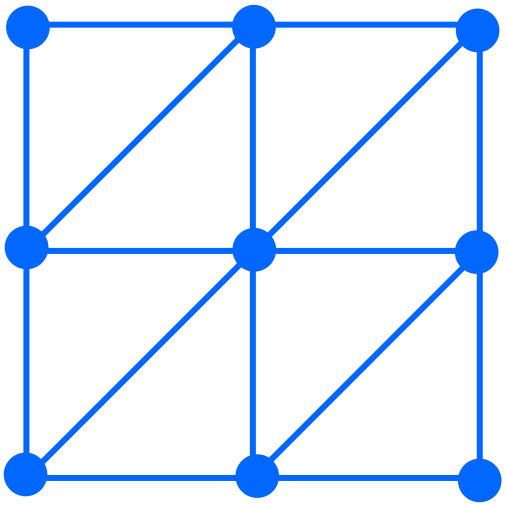
Displacement Mapping

Textur enthält Angaben zur Veränderung der Geometrie

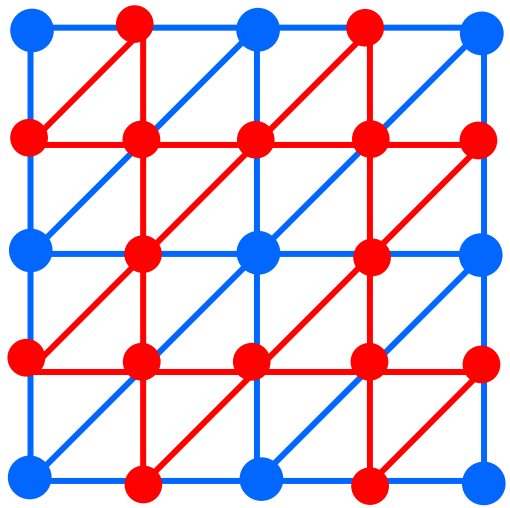
Vorteile:

- Displacement Map + grobe Geometrie braucht weniger Platz als feine Geometrie
- eine Geometrie mit mehreren Displacements (Skins) nutzbar

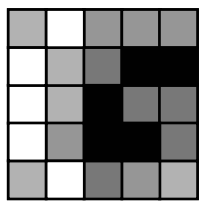
Verfeinerung der Geometrie



Ausgangsnetz

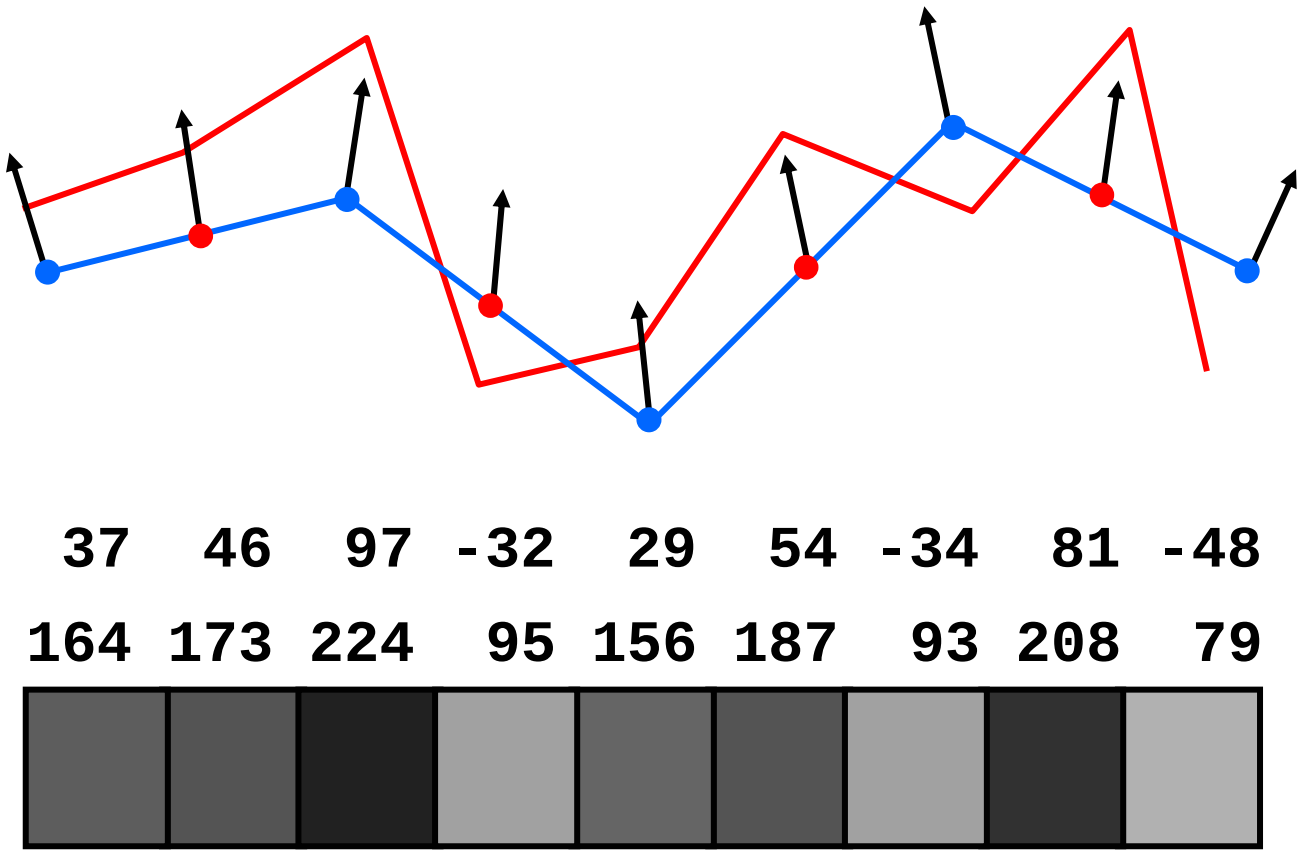


verfeinertes Netz

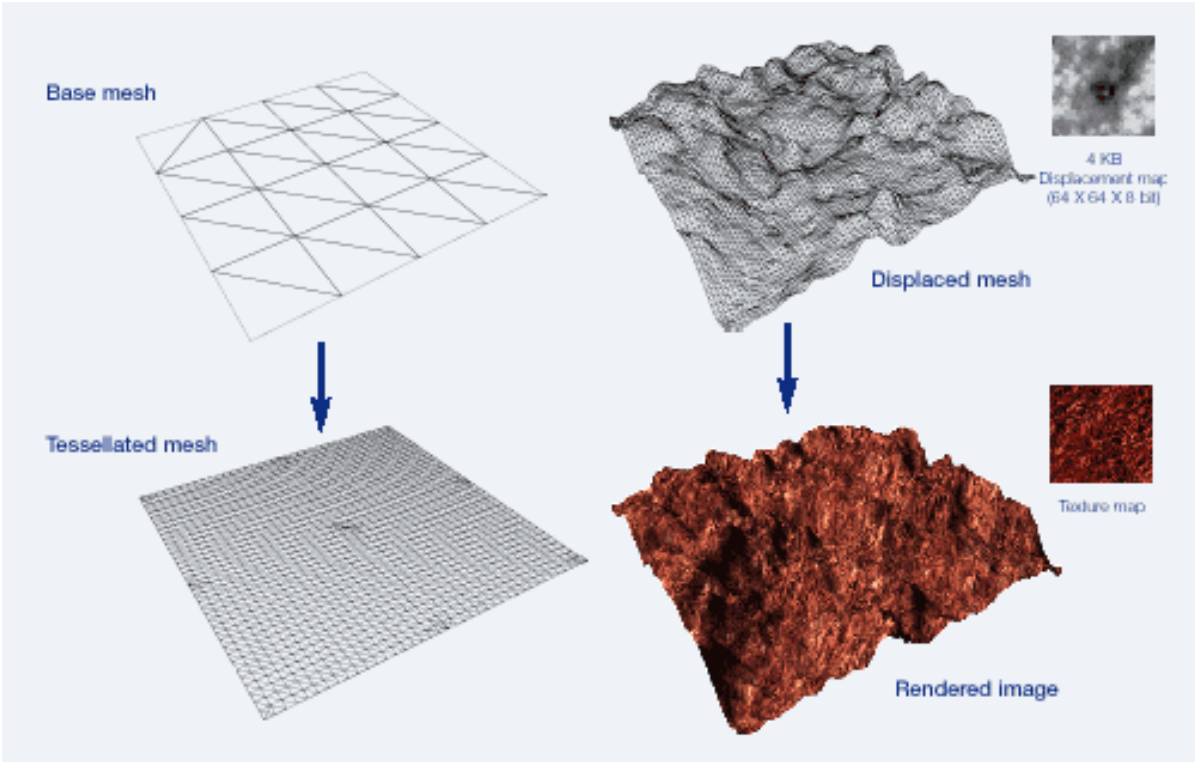


Displacement Map

Verformung der Geometrie



Matrox

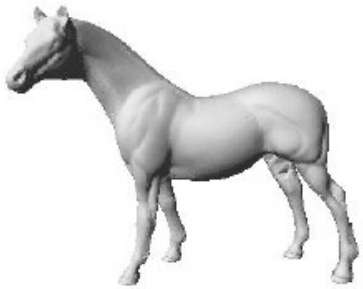

Base mesh
Tessellated mesh
Displaced mesh
4 KB
Displacement map
(64 X 64 X 8 bit)
Texture map
Rendered image

Netzvereinfachung

Ziel:

Polygone dezimieren

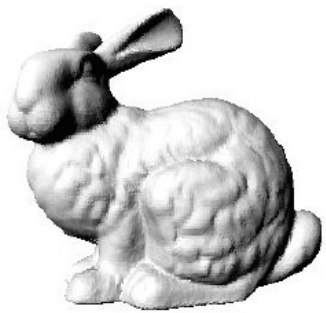
Beispiel von
Collins & Hilton,
University of Surrey



96.000

502

Fehler < 0.1 %



69.000

388

Platzersparnis

n Polygone à 3 Knoten vom Grad 6 $\Rightarrow$ n/2 Knoten

Feinstruktur:

pro Polygon:	1 Farbwert:	3 Bytes
	3 Verweise auf Knoten:	12 Bytes
pro Knoten:	homogene Koordinate:	8 Bytes
	homogene Normale:	8 Bytes
		15n + 8n = 23n Bytes

Grobstruktur:

n/100 Polygone:	23/100n Bytes
n/2 Displacementwerte:	n/2 Bytes

$\Rightarrow$ Reduktionfaktor = 23/(23/100+1/2) $\approx$ 30

Skins