

# Computergrafik 2014

## Oliver Vornberger

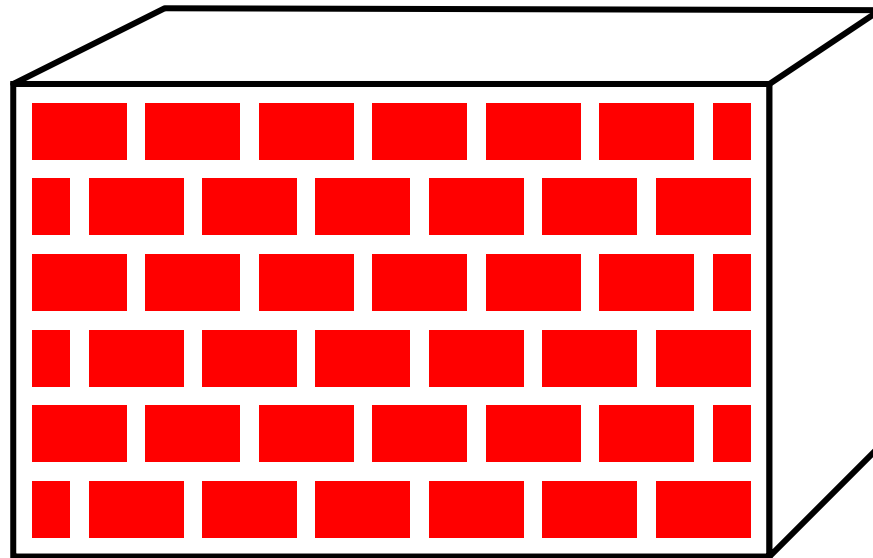
Vorlesung vom 23.06.2014

Kapitel 19:  
Texturing

# Strukturierte Fläche

Beispiel: Steinmauer

lege viele kleine rote Rechtecke  
auf ein großes weißes Rechteck:



Nachteil: aufwändige Geometrie

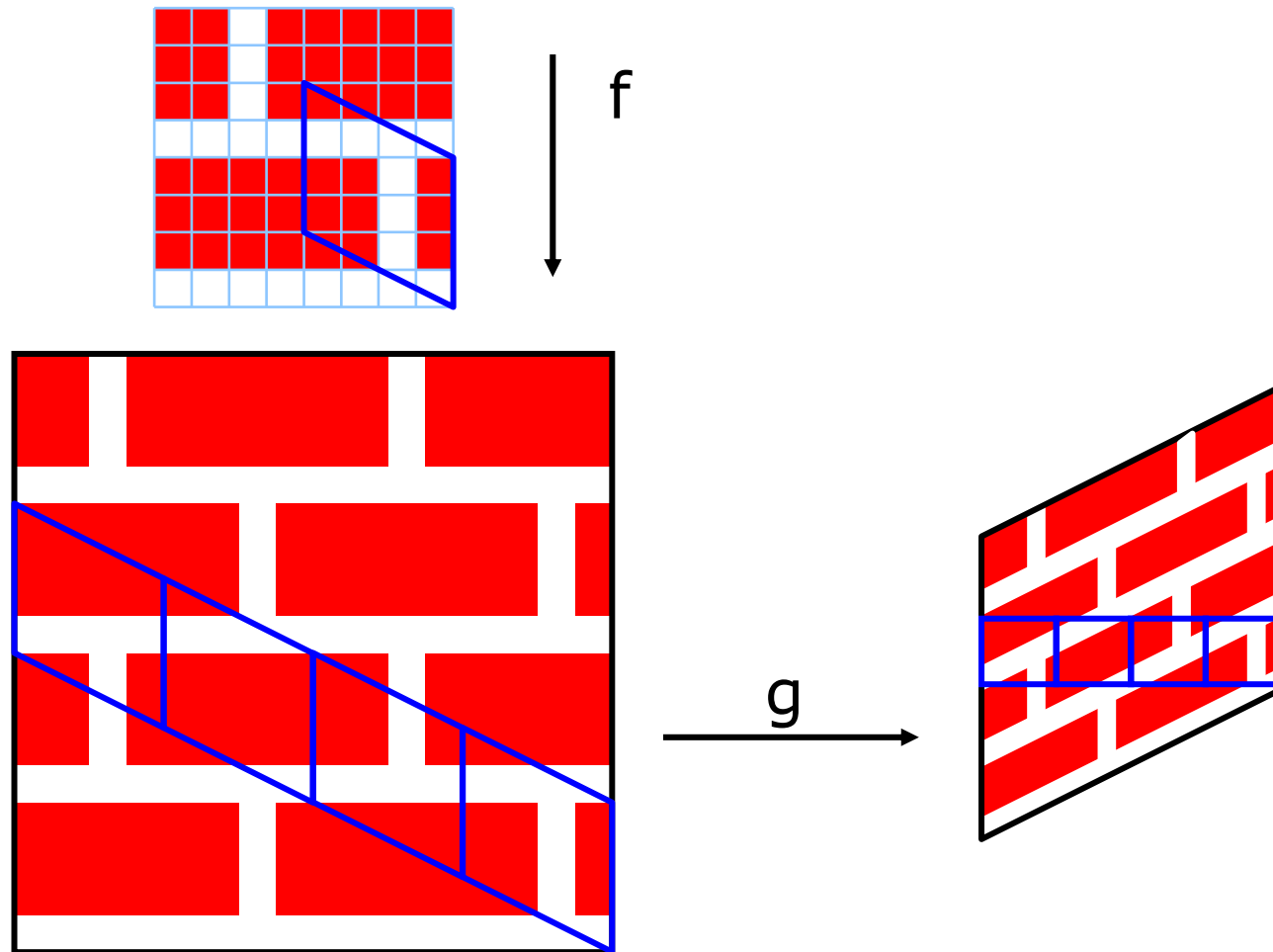
# Texel statt Geometrie

Lösung: Bild auf Objekt legen

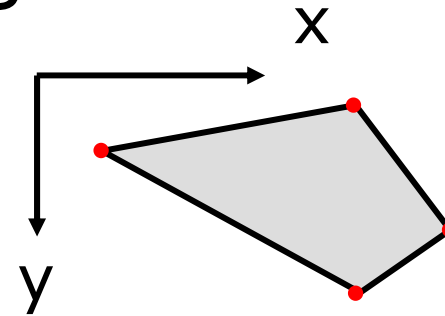
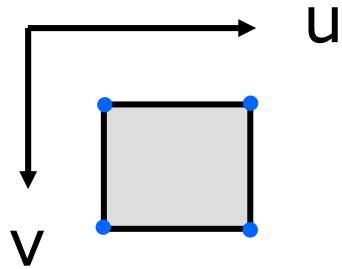
genauer: beim Rastern einer Scanline  
wird 2-dimensionale Pixelmatrix  
eingearbeitet.

Materialfarbe wird ersetzt  
und/oder kombiniert  
mit Texturpixel = Texel

# Texture Mapping



# Bildverzerrung



$$\begin{pmatrix} u \\ v \\ w \end{pmatrix} := \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

$$u = \frac{ax+by+c}{gx+hy+1} \qquad v = \frac{dx+ey+f}{gx+hy+1}$$

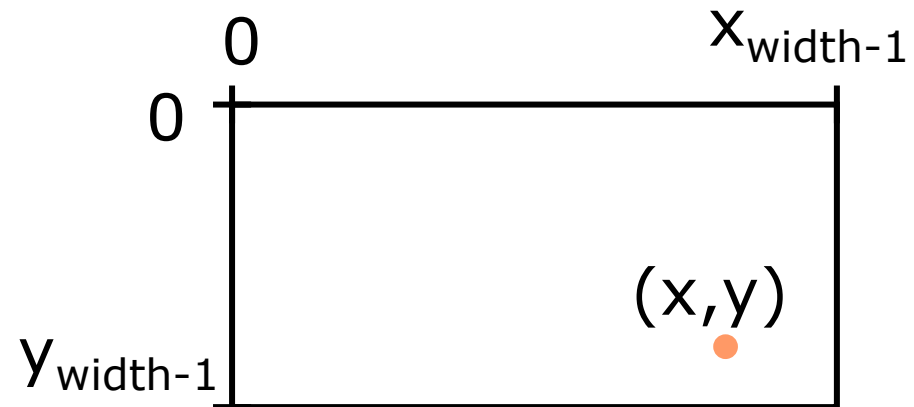
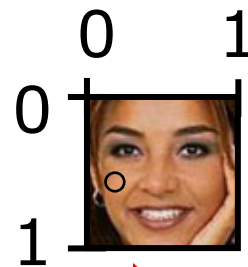
$\Rightarrow$  8 Gleichungen, 8 Unbekannte

Ergebnis liefert Transformationsmatrix  $M$

# Zugriff auf Texture Map

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

Faktoren  $f_x, f_y$



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

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

bei 16×16 Textur T:  
T[16\*0.2][16\*0.6]

# Phasen des Texture Mapping

- Raumkoordinaten des Flächenpunktes berechnen  $\Rightarrow (x', y', z')$
- zugehörige Flächenkoordinaten berechnen  $\Rightarrow (x, y)$
- Abbildung in den Parameterraum durchführen  $\Rightarrow (u, v)$
- Texturkoordinaten berechnen (Korrespondenzfunktion berücksichtigen)
- Texturwerte ermitteln
- Erscheinungsbild mit dem Texturwert modifizieren

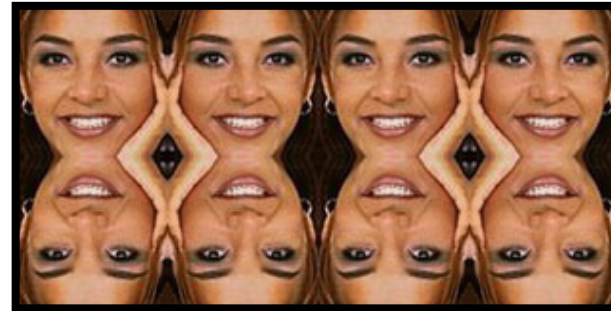
# Korrespondenzfunktion



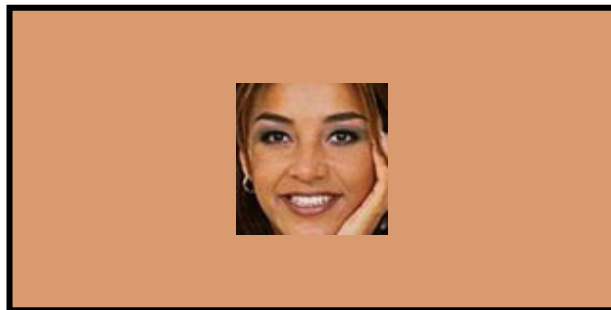
repeat



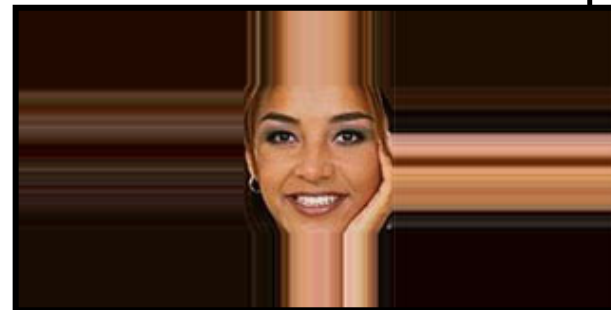
mirror



border

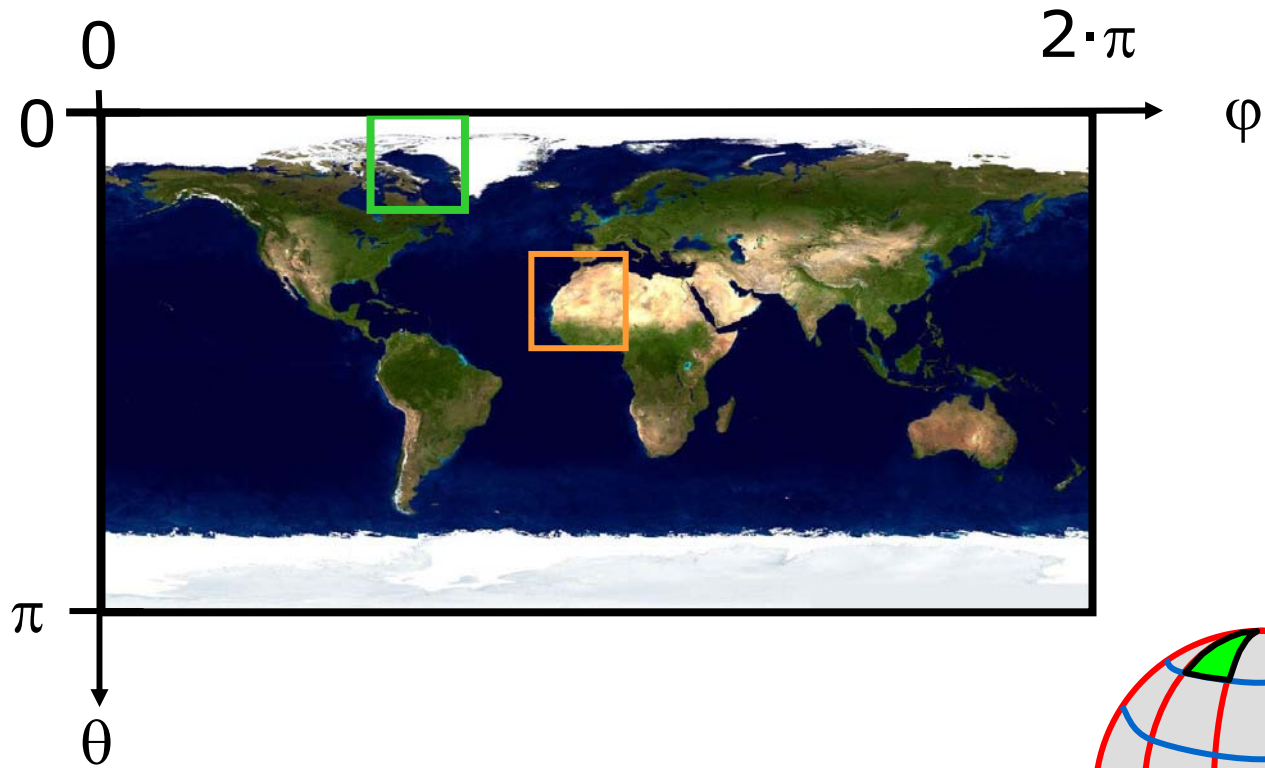


clamp

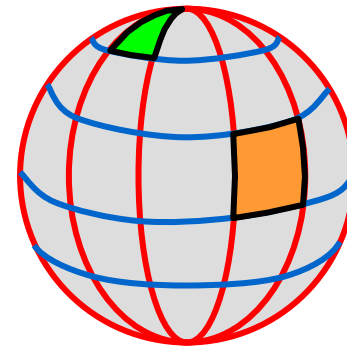




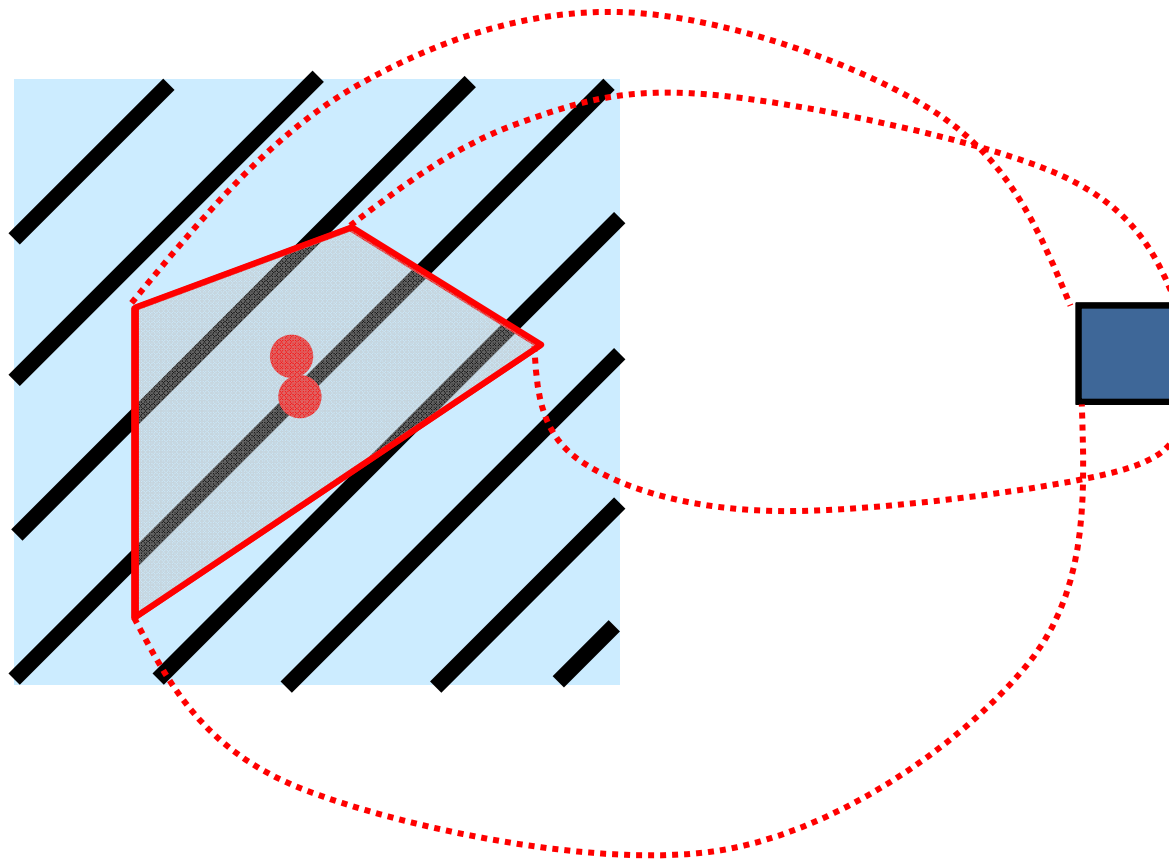
# Sphärische Projektion



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



# Textur-Artefakte

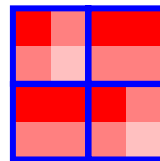
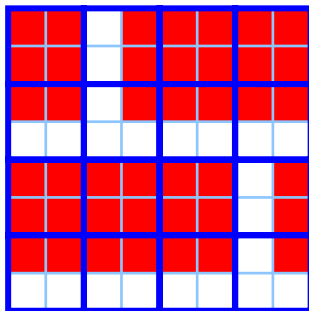


# Mip mapping

(multum in parvo mapping)

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

[255,0,0]



[255,88,88]

[255,112,112]

[255,192,192]

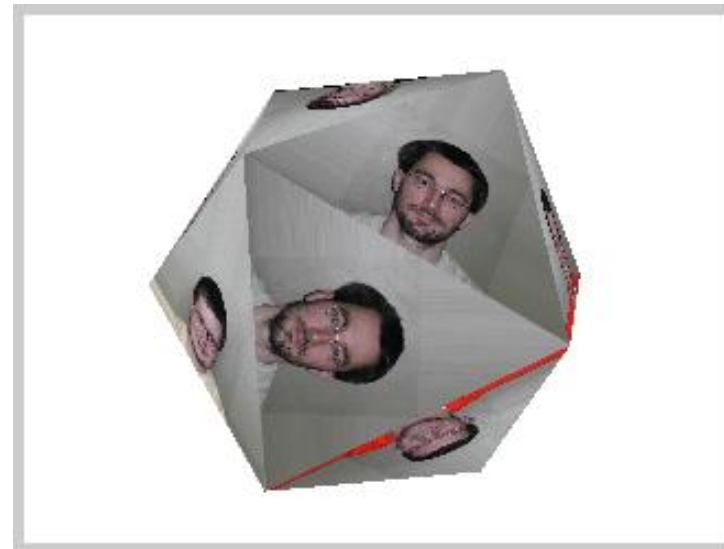
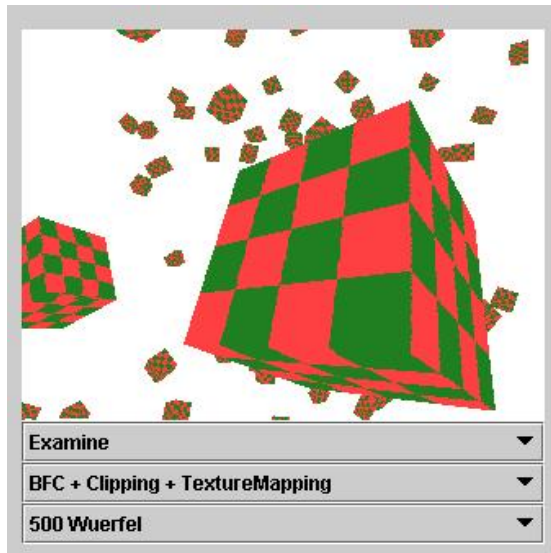
[255,255,255]

<http://www.texturemaker.com>



# Java-Applet zu Texturen

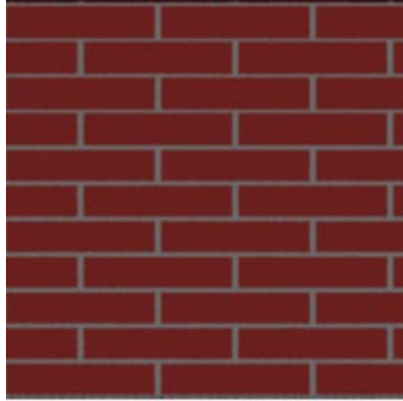
[~cg/2014/skript/Applets/Texturemap/app-1.html](http://~cg/2014/skript/Applets/Texturemap/app-1.html)



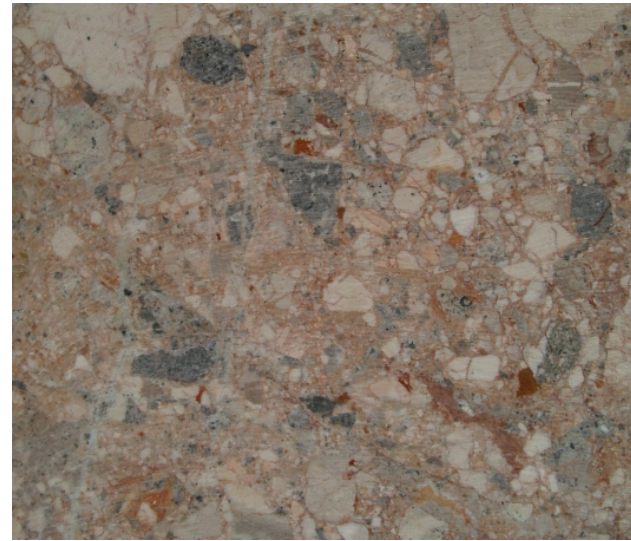
[~cg/2014/skript/Applets/Texturemap/app-2.html](http://~cg/2014/skript/Applets/Texturemap/app-2.html)



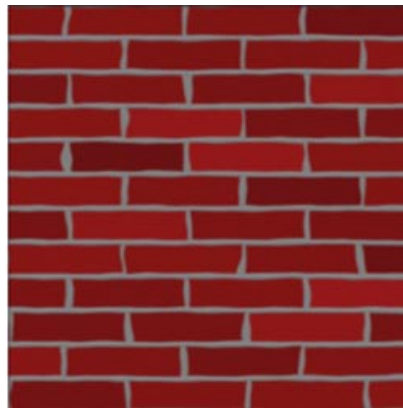
# Algorithmen für Texturen



statisch



prozedural



mit Störungen

# Light Map



Pro Face die Beleuchtung vorberechnen  
und in Light Map ablegen

$$C_{gesamt,diffus}[x,y] = C_{lighting,diffus}[x,y] \\ * LightMap[u(x,y),v(x,y)]$$

# LightMapDemo



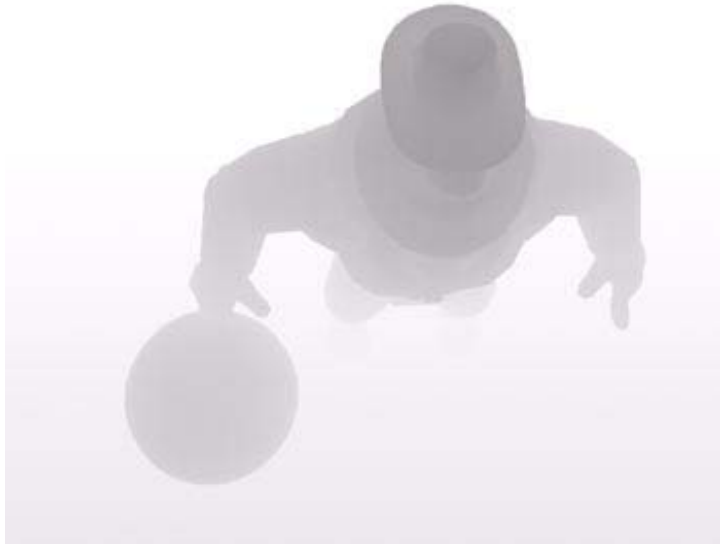
kombiniert

Szene

LightMap



# Shadow Map



Berechne z-Buffer aus  
Sicht der Lichtquelle.  
Lege in Shadow Map ab



Moduliere Pixelfarbe mit  
Hilfe der Shadow Map

Artefakte:



# Alpha Mapping

Textur enthält Alphawerte

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

Baum als Kreisfläche  
mit Löchern



Obacht: Reihenfolge beachten !

# Alpha Mapping Implementation

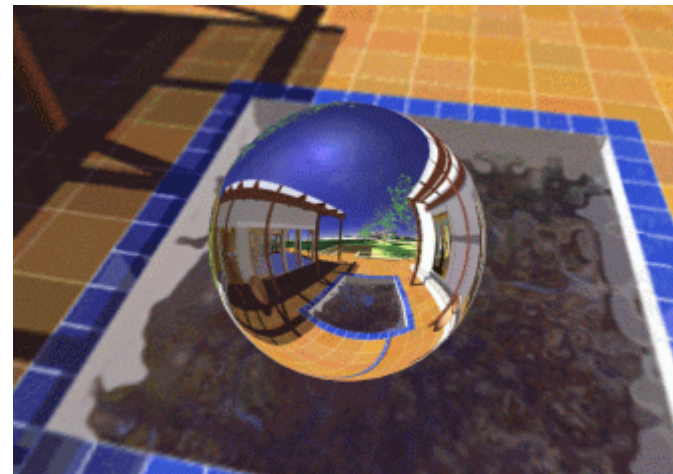
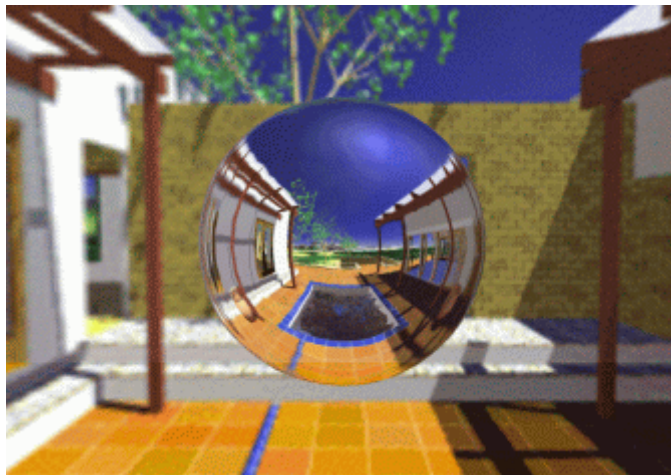
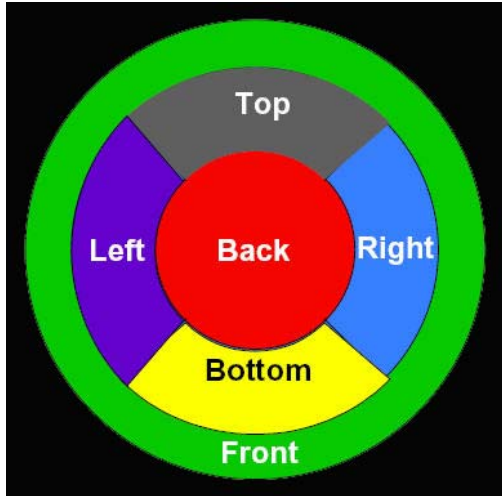
$$\begin{aligned} C_{gesamt,alpha}[x,y] = & \\ & C_{Baum,lighting}[x,y] \\ & \quad * AlphaMap[u(x,y), v(x,y)] \\ + & C_{Hintergrund,lighting} \\ & \quad * (1 - AlphaMap[u(x,y), v(x,y)]) \end{aligned}$$

# Environment Mapping

Textur enthält Projektion der Umgebung

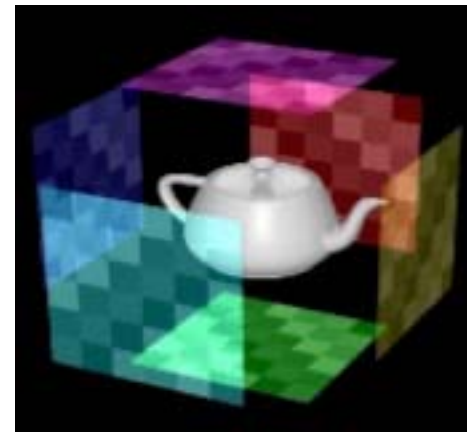
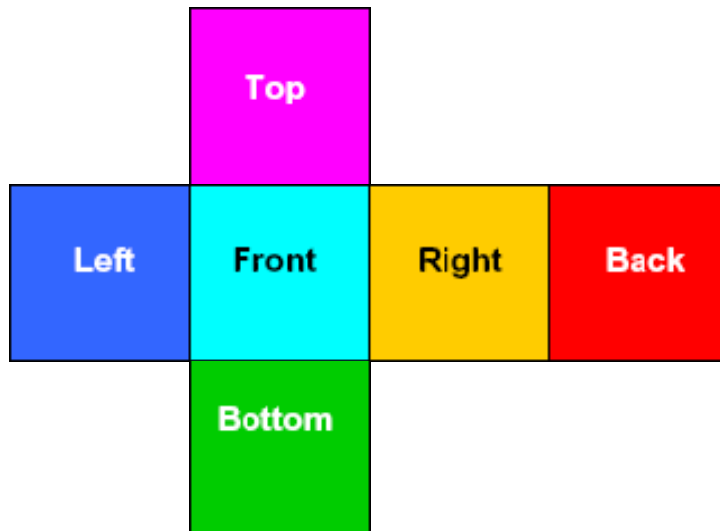


# Sphere Environment Mapping

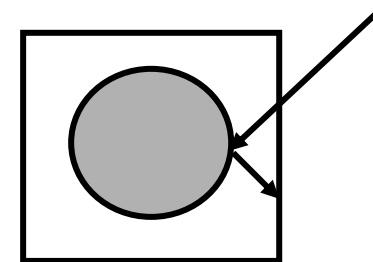


# Cube Environment Mapping

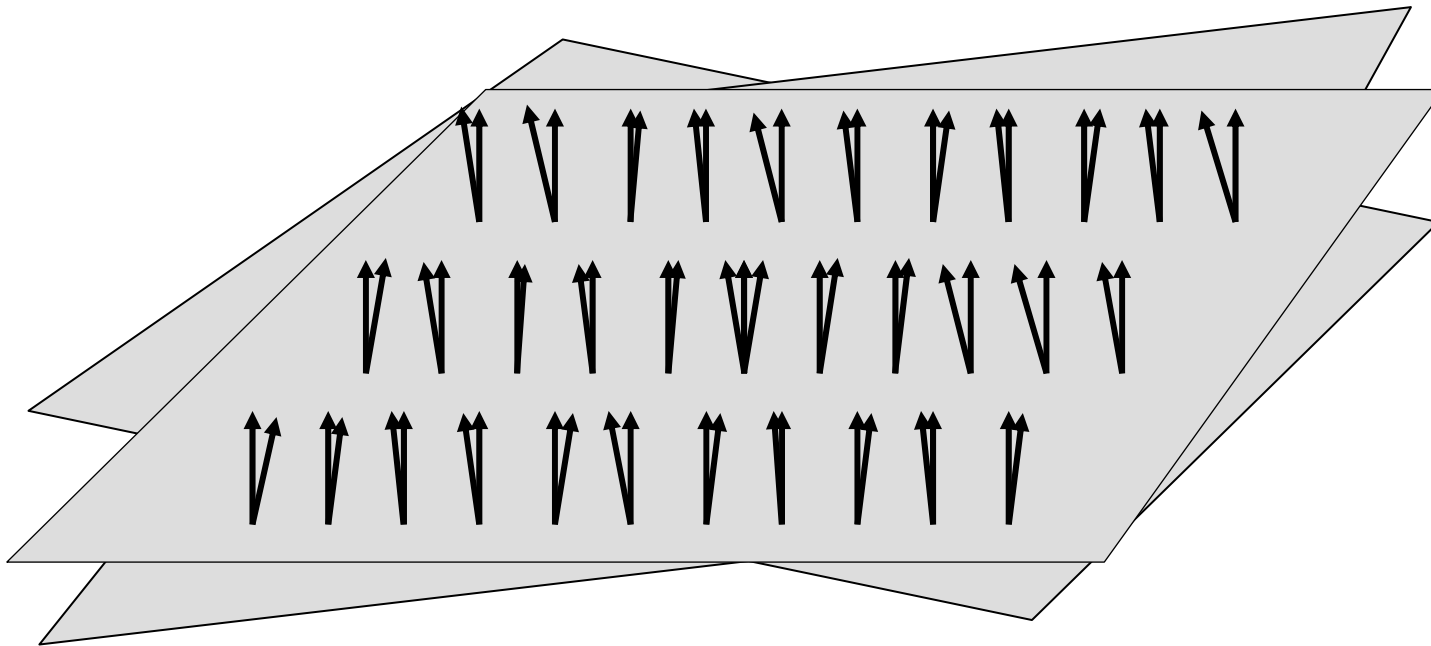
Speichere pro Objekt sechs Projektionen:



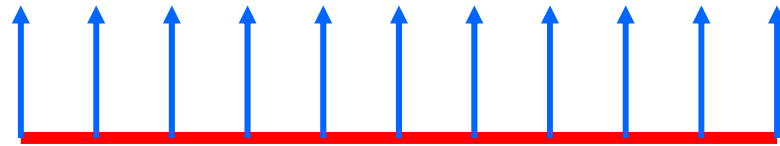
Zugriff abhängig vom Augenpunkt



# Modifikation der Normalen

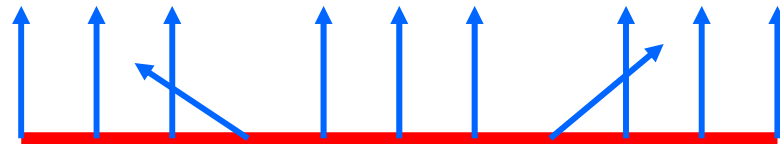


# Bump Mapping

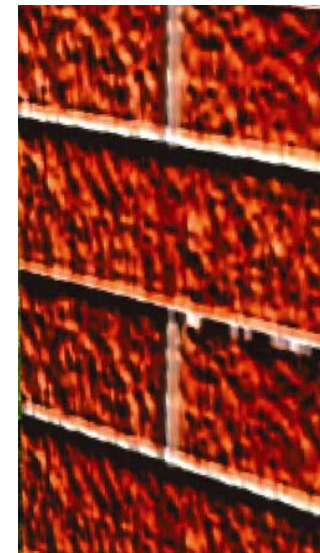


kombinierbar  
mit Textur:

modifiziere Normalenvektor

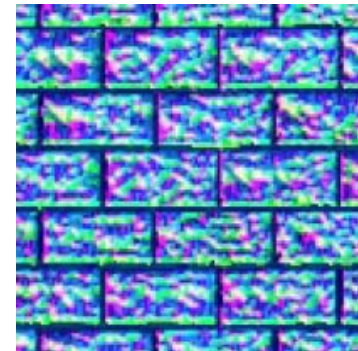
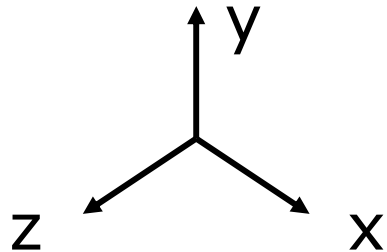


Simulation von Unebenheit:





# Bump Mapping Implementation



Height Mapping (1 Wert):

Grauwertmatrix enthält Höhenänderungen

Normal Mapping (3 Werte):

Farbmatrix enthält Normalenrichtungsänderung

Obacht:

die suggerierten Höhendifferenzen  
sind von der Seite nicht sichtbar !

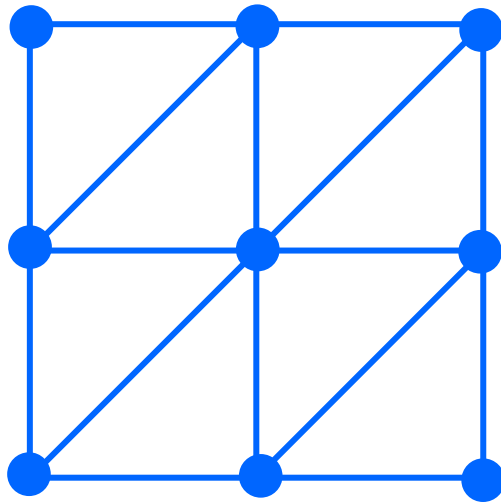
# Displacement Mapping

Textur enthält Angaben zur Veränderung der Geometrie

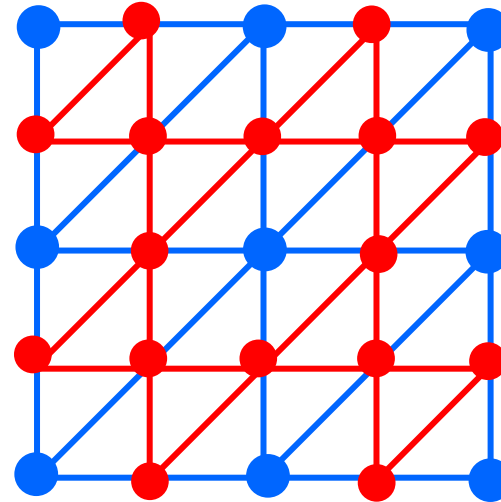
Vorteile:

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

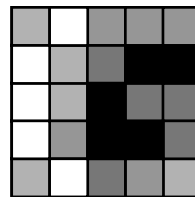
# Verfeinerung der Geometrie



Ausgangsnetz

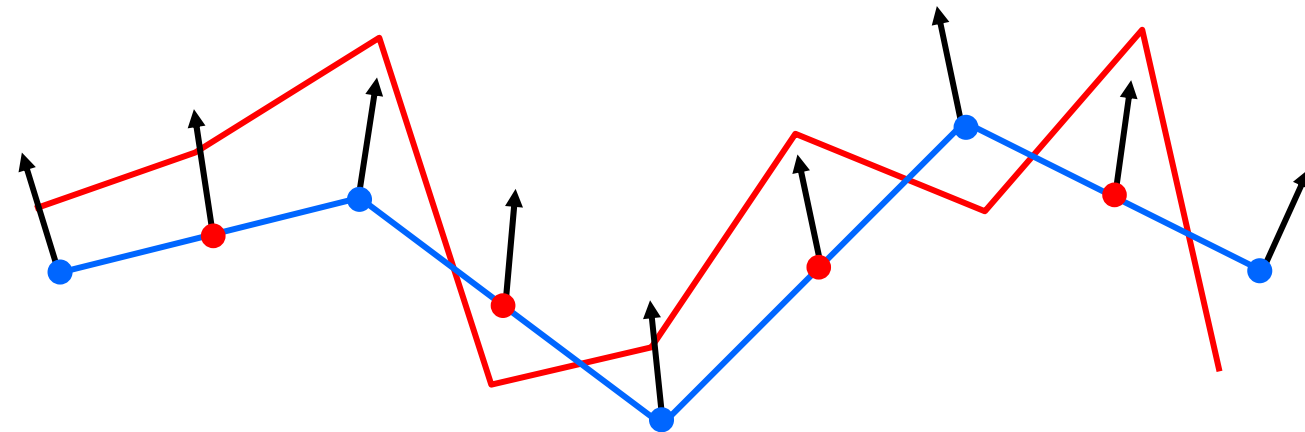


verfeinertes Netz



Displacement Map

# Verformung der Geometrie



Höhen-  
differenzen

37 46 97 -32 29 54 -34 81 -48

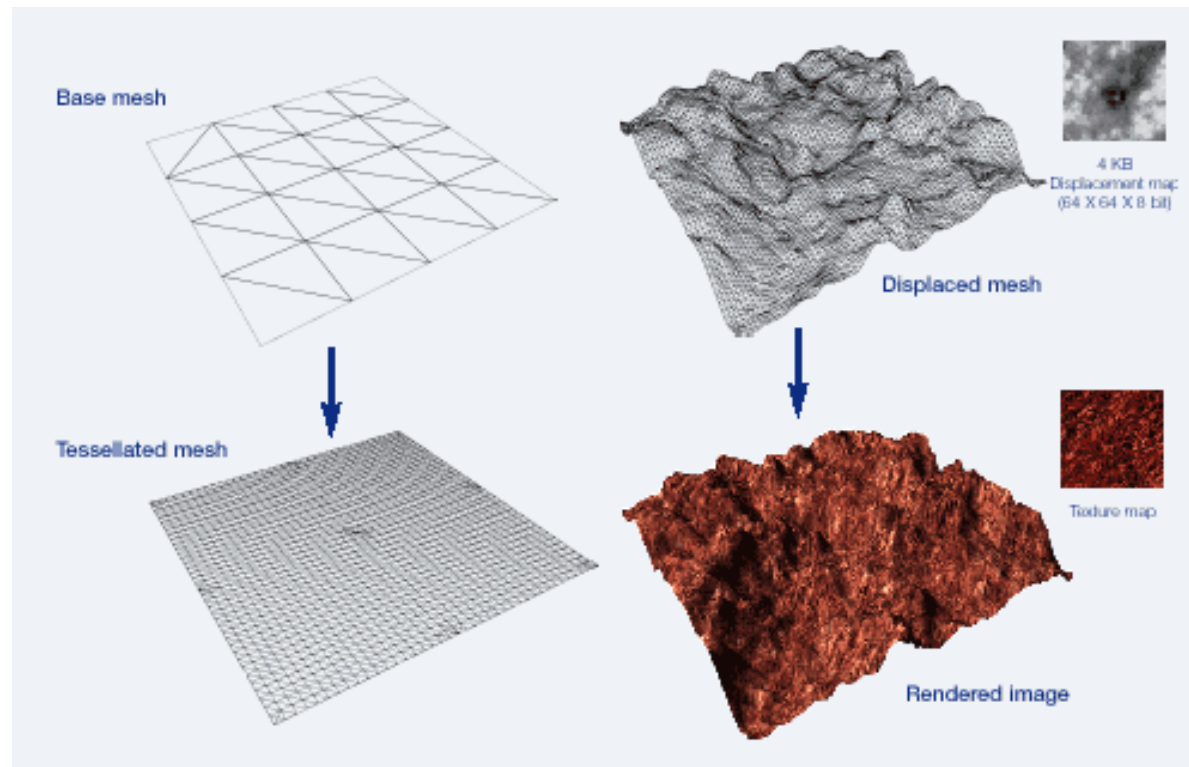
+127

164 173 224 95 156 187 93 208 79

Grauwerte



# Landschaft & Displacement

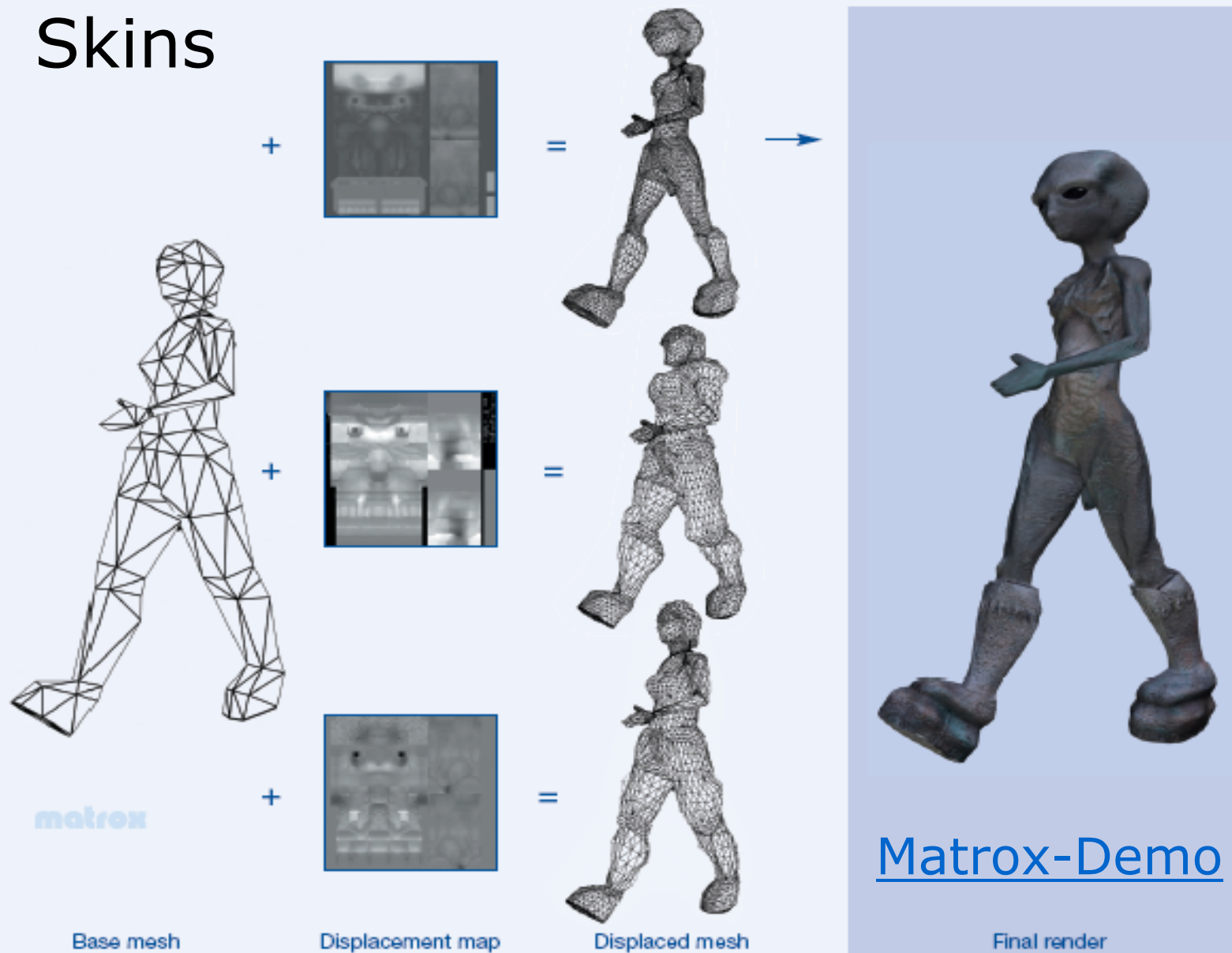


# Kopf & Displacement



modelliert von Sami Sorjonen, gerendert von Mathias Wein

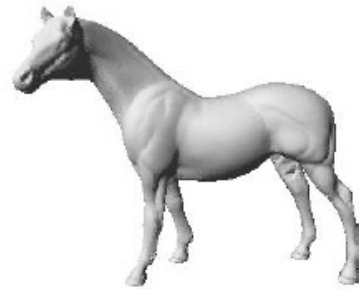
# Skins



# Netzvereinfachung

Ziel: Zahl der  
Polygone dezimieren

Beispiel von  
Collins & Hilton,  
University of Surrey



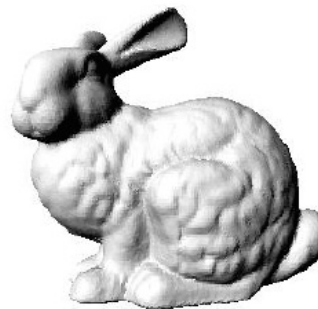
96.000



502

Reduktion  $< 1\%$

Fehler  $< 0.1 \%$



69.000



388



# Platzersparnis

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

## Feinstruktur:

pro Polygon: 3 Farbwerte + Transparenz: 4 Bytes

3 Verweise auf Knoten: 12 Bytes

pro Knoten: homogene Koordinate: 16 Bytes

homogene Normale: 16 Bytes

$16n + 16n = 32n$  Bytes

## Grobstruktur:

$n/100$  Polygone:  $32/100n$  Bytes

$n/2$  Displacementwerte:  $n/2$  Bytes

$\Rightarrow$  Reduktionsfaktor =  $32/(32/100 + 1/2) \approx 40$