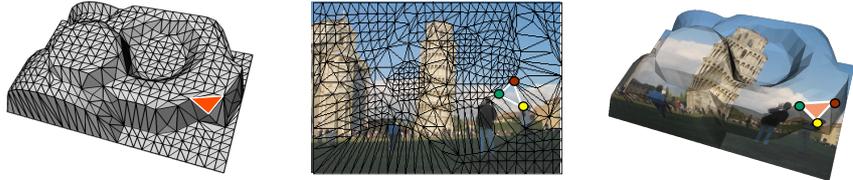


Parametrisierung

- Wie kommt man zu den Texturkoordinaten an jedem Vertex?
- Triviale Texturierung eines Terrains:
 - 3D-Koordinaten nach unten projizieren

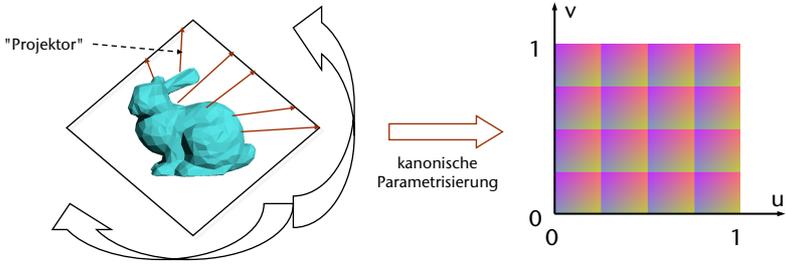


- Achtung: dies ist nicht notwendig eine "gute" Texturierung!

G. Zachmann Computer-Graphik 2 - SS 08 Texturen 66

- Idee: 2-stufiger Prozess [Bier & Sloan, 1986]
 - Lege (konzeptionell) einen "kanonisch" parametrisierbaren Hüllkörper um das ganze Objekt

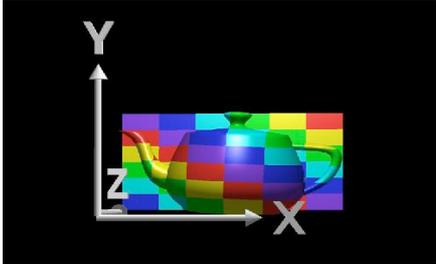
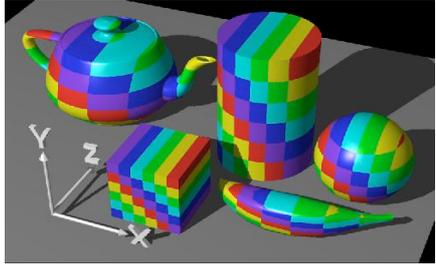
- Projiziere Vertices (nicht notw. Vertex-Koord.!) auf diesen Hüllkörper
- Verwende die Texturkoordinaten des projizierten Punktes auf dem Hüllkörper



G. Zachmann Computer-Graphik 2 - SS 08 Texturen 67

Einige Hüllkörper und deren Parametrisierung

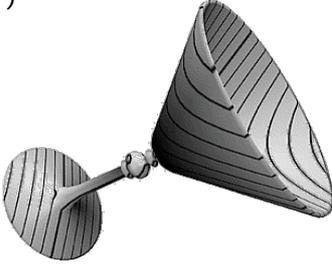
- Ebene:
 - Projiziere Punkt (x,y,z) auf Ebene
 $\rightarrow (x,y)$
 - $(u,v) = (s_x x + t_x, s_y y + t_y)$
- Verallgemeinerung:
 - Definiere 2 beliebige Ebenen E_1 und E_2
 - $u := \text{dist}(P, E_1)$
 $v := \text{dist}(P, E_2)$
 - Dieses Feature bietet OpenGL

G. Zachmann Computer-Graphik 2 - SS 08
Texturen 68

Beispiel

- Erzeuge Höhenlinien mittels dieser Technik:
 - 1D-Textur
 - $u := \text{dist}(P, E_1)$

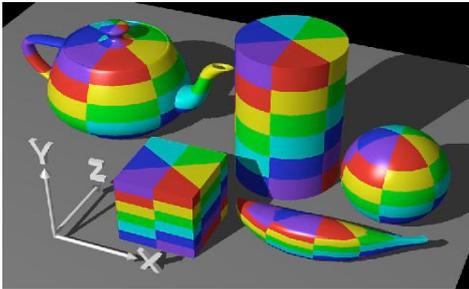
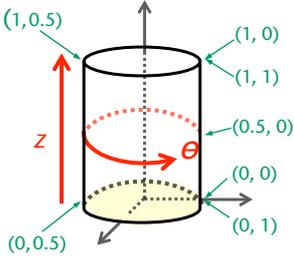


- Viele weitere ungewöhnliche Anwendungen von Texture-Mapping auf
<http://www.graficaobscura.com/texmap/index.html>

G. Zachmann Computer-Graphik 2 - SS 08
Texturen 69

■ Zylinder-Parametrisierung:

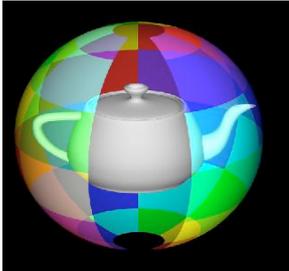
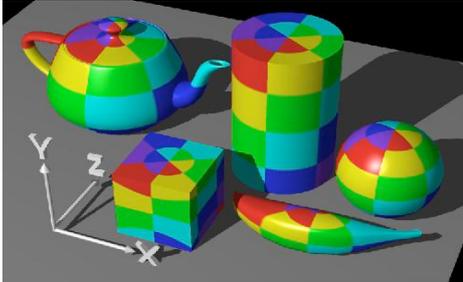
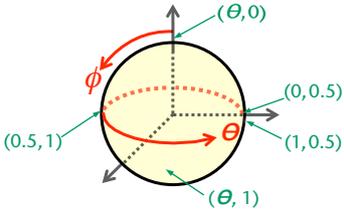
- Konvertiere kartesische Koord. (x,y,z) in zylindrische Koord.
 $\rightarrow (r \sin \theta, r \cos \theta, z)$
- $(u, v) = (\theta/2\pi, z)$
- Beachte "Naht" bei $(\theta=0 \text{ \& } 2\pi)$

G. Zachmann Computer-Graphik 2 - SS 08 Texturen 70

■ Kugel-Parametrisierung:

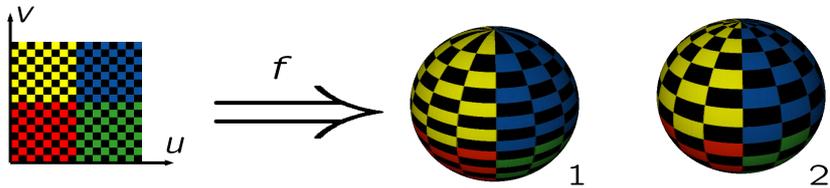
- Stelle Punkt in sphärischen Koord. dar
 $\rightarrow r \cdot (\sin \theta \sin \phi, \cos \theta \sin \phi, \cos \phi)$
- $(u, v) = (\theta/2\pi, \phi/\pi + 1)$
- Beachte: Singularität bei Nord- und Südpol!

G. Zachmann Computer-Graphik 2 - SS 08 Texturen 72

▪ Hier gibt es 2 Varianten:

1. $f(u, v) = \begin{pmatrix} \sin(\pi v) \cos(2\pi u) \\ \sin(\pi v) \sin(2\pi u) \\ \cos(\pi v) \end{pmatrix}$
2. $f(u, v) = \begin{pmatrix} \sqrt{2v-1} \cos(2\pi u) \\ \sqrt{2v-1} \sin(2\pi u) \\ 2v-1 \end{pmatrix}$



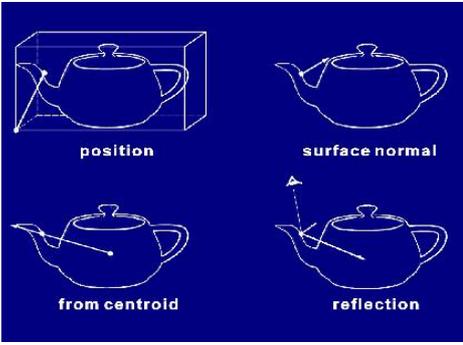
G. Zachmann Computer-Graphik 2 - SS 08 Texturen 73

Was soll man projizieren?

▪ Bisher: einfach die Koordinaten (x, y, z) des Vertex auf den (gedachten) Hüllkörper projiziert

▪ Verallgemeinerung: statt dessen kann man genauso gut (oder schlecht) andere Attribute des Vertex projizieren, z.B.

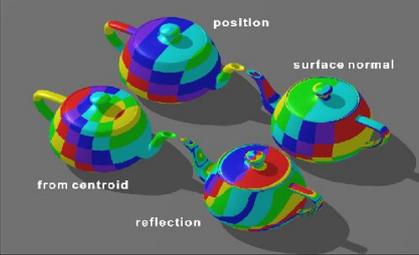
- Normale
- Vektor vom Zentrum des Objektes durch den Vertex
- Reflektierter Viewing-Vektor
- ...



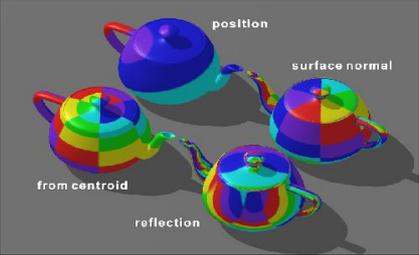
G. Zachmann Computer-Graphik 2 - SS 08 Texturen 74

Beispiele:

planar



cylindrical



G. Zachmann Computer-Graphik 2 - SS 08 Texturen 75

Automatische Erzeugung von Textur-Koordinaten in OpenGL

- `glEnable(GL_TEXTURE_GEN_S); // S, T, R, Q`
- `glTexGeni(GL_S, GL_TEXTURE_GEN_MODE, mode);`
- mode =
 - `GL_OBJECT_LINEAR` : Texturkoord. = Distanz des Vertex von einer Ebene; die Ebene wird spezifiziert mit `glGenTexfv(GL_S, GL_OBJECT_PLANE, v)`
 - `GL_EYE_LINEAR` : benutze Vertex-Koord. **nach** `MODEL_VIEW`
 - `GL_SPHERE_MAP` : für Environment-Mapping (später)
 - `GL_NORMAL_MAP`
 - `GL_REFLECTION_MAP`

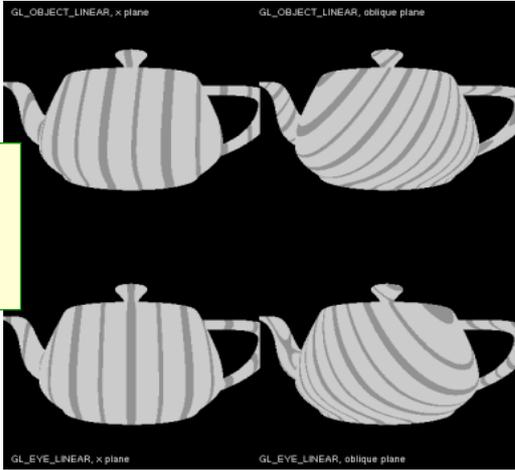
G. Zachmann Computer-Graphik 2 - SS 08 Texturen 76

Beispiel

```

glEnable( GL_TEXTURE_GEN_S );
glTexGeni( GL_S,
           GL_TEXTURE_GEN_MODE,
           GL_OBJECT_LINEAR );
glTexGenfv( GL_S,
            GL_OBJECT_PLANE,
            xPlane );

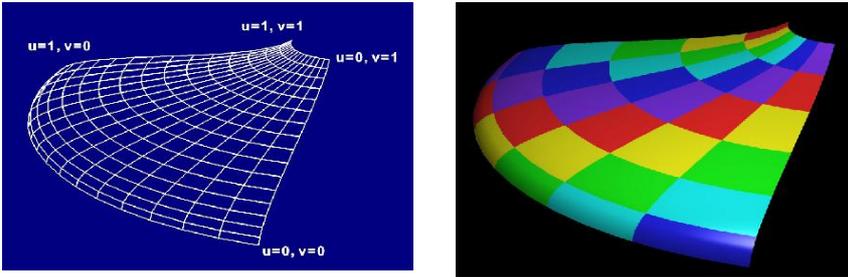
```



The image displays four teapots arranged in a 2x2 grid, illustrating different texture generation modes. The top-left teapot is labeled 'GL_OBJECT_LINEAR, x plane'. The top-right teapot is labeled 'GL_OBJECT_LINEAR, oblique plane'. The bottom-left teapot is labeled 'GL_EYE_LINEAR, x plane'. The bottom-right teapot is labeled 'GL_EYE_LINEAR, oblique plane'. Each teapot shows a different pattern of vertical stripes, demonstrating how the texture is mapped to the surface of the teapot based on the specified generation mode and plane.

G. Zachmann Computer-Graphik 2 - SS 08 Texturen 77

Parametrisierung parametrischer Flächen ist oft "kanonisch":

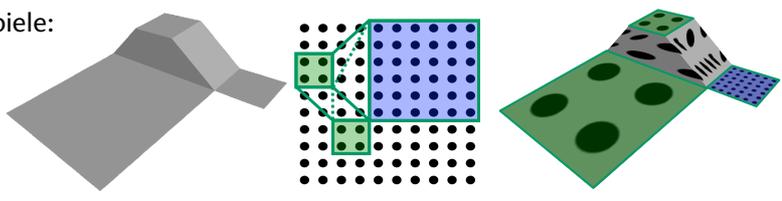
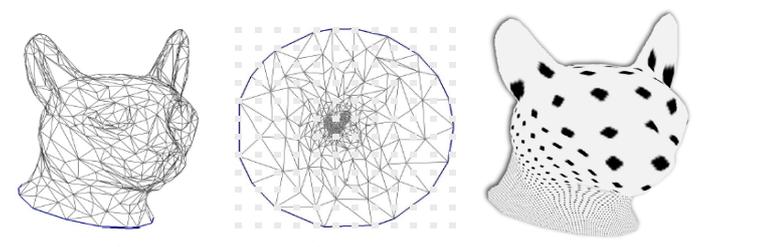


The image shows two side-by-side views of a curved surface. The left view is a wireframe grid representing the surface's parametrization. The grid lines are labeled with their respective u and v coordinates: 'u=1, v=0' at the top-left corner, 'u=1, v=1' at the top-right corner, 'u=0, v=1' at the bottom-right corner, and 'u=0, v=0' at the bottom-left corner. The right view shows the same surface with a colorful checkerboard pattern, where each square in the grid is a different color, illustrating the canonical parametrization of the surface.

G. Zachmann Computer-Graphik 2 - SS 08 Texturen 78

Probleme bei der Parametrisierung

- Verzerrungen: Größe & Form
- Folge: **Relatives over-** bzw. **under-sampling**
- Beispiele:

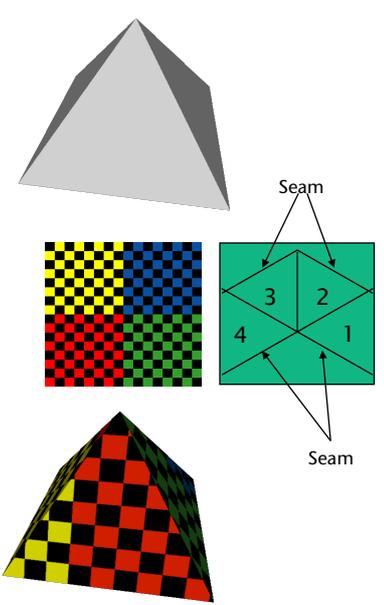



Mesh Einbettung Verzerrung

G. Zachmann Computer-Graphik 2 - SS 08 Texturen 79

Seams (Textursprünge)

- Ziel: Verringern der Verzerrung
- Idee: Aufschneiden des Netzes entlang gewisser Kanten
- Ergibt „doppelte“ Kanten, auch „*seams*“ genannt
- Unvermeidlich bei nicht-planarer Topologie



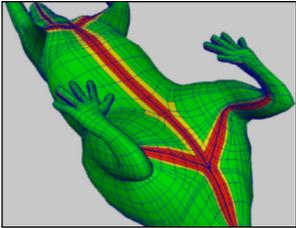
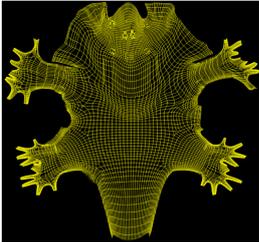
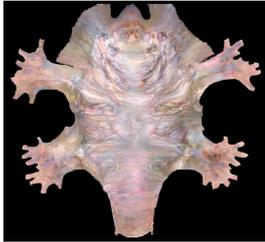
Seam

Seam

G. Zachmann Computer-Graphik 2 - SS 08 Texturen 80

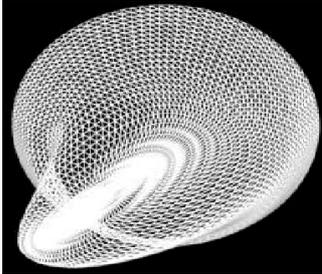
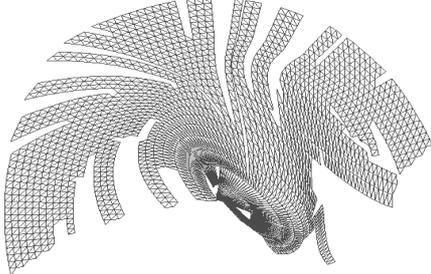
- Idee 1 [Piponi 2000]:
 - Das Objekt entlang nur **einer** zusammenhängenden Kante so aufschneiden, daß eine topologische Scheibe entsteht
 - Dieses aufgeschnittene Netz dann in die Ebene einbetten



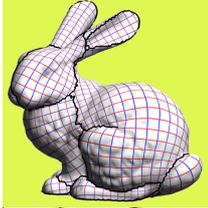
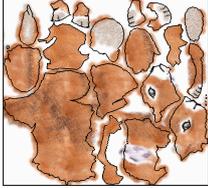
G. Zachmann Computer-Graphik 2 - SS 08
Texturen 81

- Probleme:
 - Es gibt immer noch Verzerrungen
 - Mehrfaches "Einschneiden" ergibt stark "zerfranstes" eingebettetes Gitter

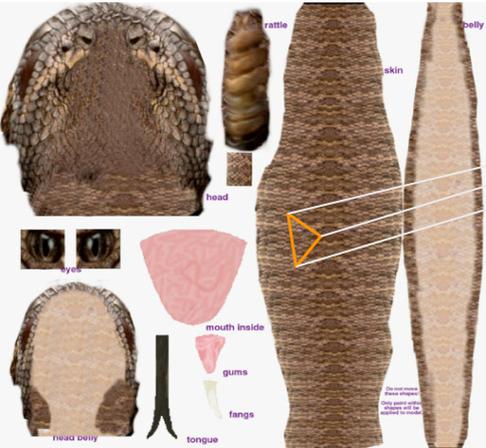
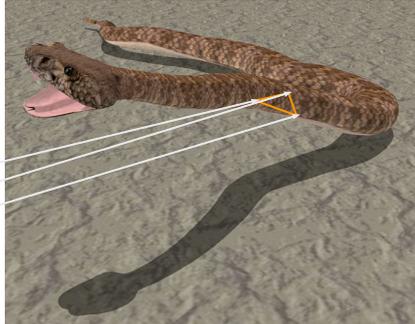
G. Zachmann Computer-Graphik 2 - SS 08
Texturen 82

- Idee 2:
 - Zerschneide Fläche in einzelne **Patches**
 - **Karte (map)** = einzelnes Parametergebiet im Texture-Space für ein Patch
 - **Textur-Atlas** = Vereinigung dieser Patches mit ihren jeweiligen Parametrisierungen
- Problemstellung:
 - Wähle Kompromiß zwischen Seams und Verzerrung
 - „Verstecke“ Schnitte in wenig sichtbaren Regionen
 - Möglichst kompakte Anordnung der Texturpatches (ein sog. Packing-Problem)

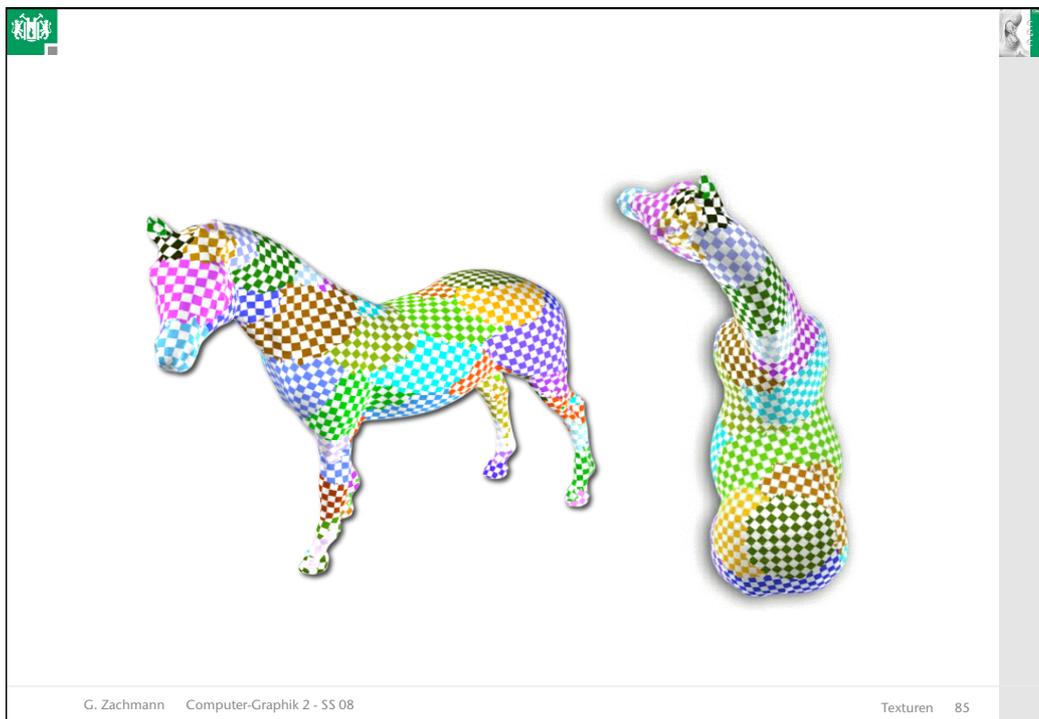




G. Zachmann Computer-Graphik 2 - SS 08
Texturen 83

- Beispiel

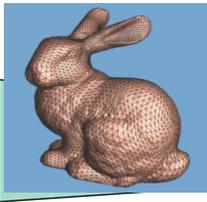



G. Zachmann Computer-Graphik 2 - SS 08
Texturen 84

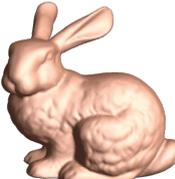


Verzerrung oder Seams?

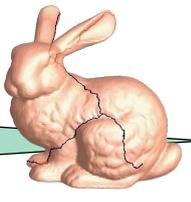
in Dreiecke zerschneiden

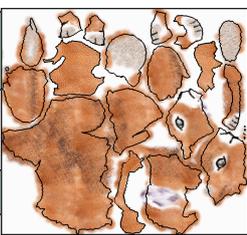


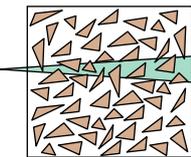
Seams

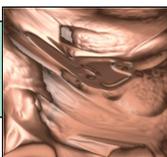


in ein Patch aufschneiden









Verzerrung

Texture Atlas:

- kleine Anzahl Patches
- kurze & versteckte Seams

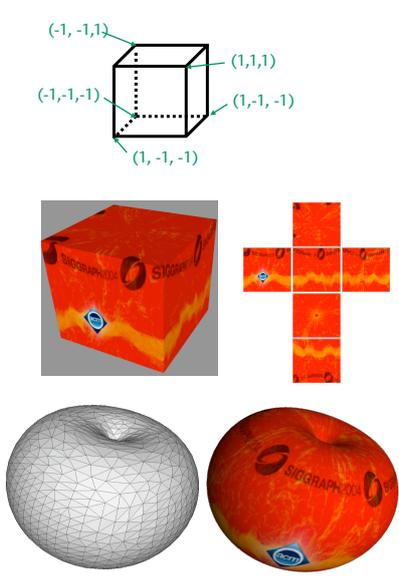
G. Zachmann Computer-Graphik 2 - SS 08 Texturen 87

Cube Maps

[Greene '86, Voorhies '94]

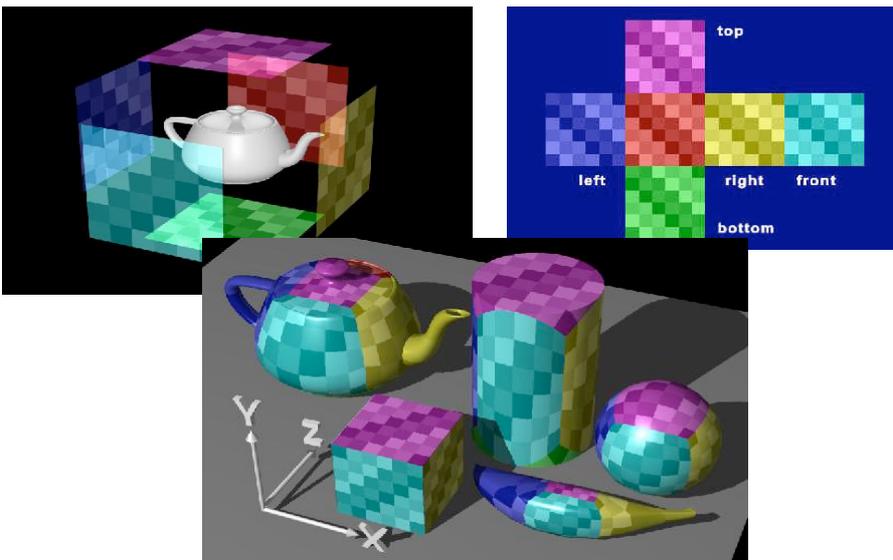
- Ω = Einheits-Würfel:
 - Sechs quadratische Textur-Bitmaps
 - 3D Texturkoordinaten:


```
glTexCoord3f( s, t, r );
glVertex3f( x, y, z );
```
 - Größte Komponente von (s, t, r) wählt die Karte aus, Schnittpunkt liefert (u, v) innerhalb der Karte
- Rasterisierung
 - Interpolation von (s, t, r) in 3D
 - Projektion auf Würfel
 - Texture look-up in 2D
- Vorteile: rel. uniform, OpenGL
- Nachteil: man benötigt 6 Bilder



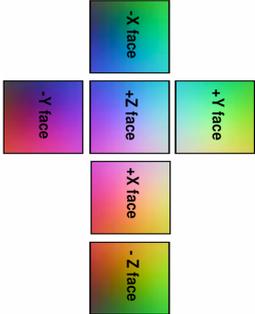
G. Zachmann Computer-Graphik 2 - SS 08 Texturen 88

Beispiele



G. Zachmann Computer-Graphik 2 - SS 08 Texturen 89

- Weitere Anwendung: man kann eine Cube-Map auch sehr gut verwenden, um **irgendeine** Funktion der **Richtung** zu speichern! (vorberechnet als LUT)
- Beispiel: Normierung eines Vektors
 - Jedes Cube-Map-Textel (s, t, r) speichert in RGB den Vektor
$$\frac{(s, t, r)}{\|(s, t, r)\|}$$
 - Jetzt kann man beliebige Texturkoordinaten mittels `glTexCoord3f()` angeben, und bekommt den normierten Vektor
 - Achtung: bei dieser Technik sollte man (meistens) jegliche Filterung ausschalten!



-X face
 -Y face +Z face +Y face
 +X face
 -Z face

G. Zachmann Computer-Graphik 2 - SS 08 Texturen 90

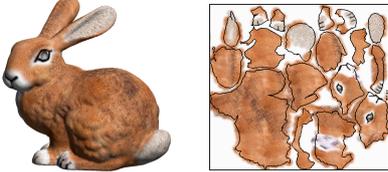
Cube-Maps in OpenGL

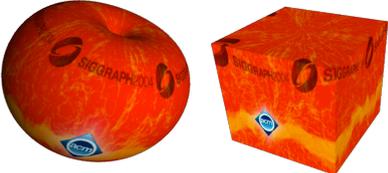
```

glBindTexture( GL_TEXTURE_CUBE_MAP );
glTexImage( GL_TEXTURE_CUBE_MAP_POSITIVE_X, 0, GL_RGB, w, h,
           0, GL_RGB, GL_UNSIGNED_BYTE, image );
glTexParameteri( GL_TEXTURE_CUBE_MAP, ... );
...
glEnable( GL_TEXTURE_CUBE_MAP );
glBindTexture( GL_TEXTURE_CUBE_MAP );
glBegin( GL_... );
glTexCoord3f( s, t, r );
glVertex3f( ... );
...
  
```

G. Zachmann Computer-Graphik 2 - SS 08 Texturen 91

Textur-Atlas vs. Cube-Map



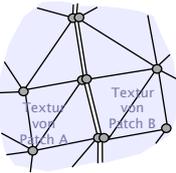


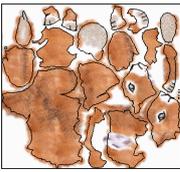
- Seams
- Dreiecke innerhalb der Patches
- nur für ein Dreiecksnetz
- Mip-Mapping schwierig
- verschwendete Texel
- Sampling-Artefakte an den Rändern der Patches

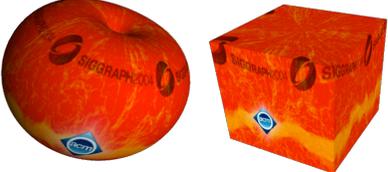
- Keine seams
- Dreiecke in mehreren Patches
- für viele Dreiecksnetze
- Mip-Mapping okay
- alle Texel benutzt
- Keine Ränder, keine Sampling-Artefakte

G. Zachmann Computer-Graphik 2 - SS 08 Texturen 92

Textur-Atlas vs. Cube-Map



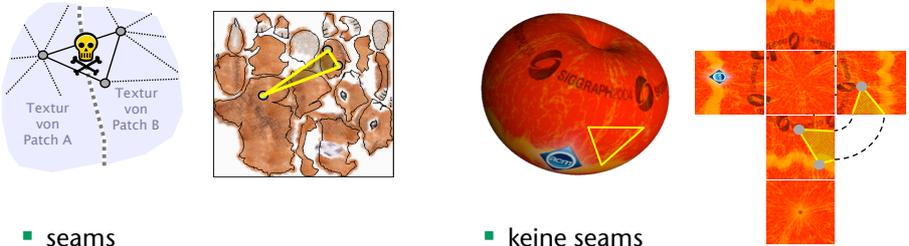




- seams
- Dreiecke innerhalb der Patches
- nur für ein Dreiecksnetz
- Mip-Mapping schwierig
- verschwendete Texel
- Sampling-Artefakte an den Rändern der Patches

- keine seams
- Dreiecke in mehreren Patches
- für viele Dreiecksnetze
- Mip-Mapping okay
- alle Texel benutzt
- keine Ränder, keine Sampling-Artefakte

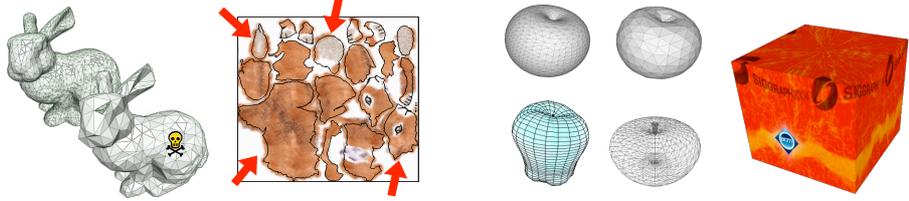
G. Zachmann Computer-Graphik 2 - SS 08 Texturen 93



- seams
- Dreiecke innerhalb der Patches
- nur für ein Dreiecksnetz
- Mip-Mapping schwierig
- verschwendete Texel
- Sampling-Artefakte an den Rändern der Patches

- keine seams
- Dreiecke in mehreren Patches
- für viele Dreiecksnetze
- Mip-Mapping okay
- alle Texel benutzt
- keine Ränder, keine Sampling-Artefakte

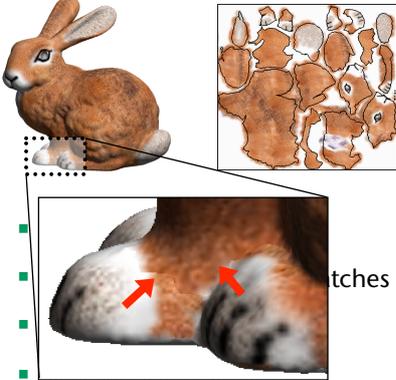
G. Zachmann Computer-Graphik 2 - SS 08 Texturen 94



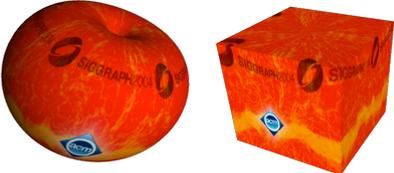
- seams
- Dreiecke innerhalb der Patches
- nur für ein Dreiecksnetz
- Mip-Mapping schwierig
- verschwendete Texel
- Sampling-Artefakte an den Rändern der Patches

- keine seams
- Dreiecke in mehreren Patches
- für viele Dreiecksnetze
- Mip-Mapping okay
- alle Texel benutzt
- keine Ränder, keine Sampling-Artefakte

G. Zachmann Computer-Graphik 2 - SS 08 Texturen 95

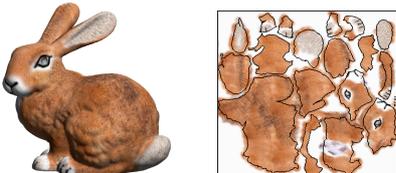


- verschwendete Texel
- Sampling-Artefakte an den Rändern der Patches



- keine seams
- Dreiecke in mehreren Patches
- für viele Dreiecksnetze
- Mip-Mapping okay
- alle Texel benutzt
- keine Ränder, keine Sampling-Artefakte

G. Zachmann Computer-Graphik 2 - SS 08 Texturen 96



- seams
- Dreiecke innerhalb des Patches
- nur für ein Dreiecksnetz
- Mip-Mapping schwierig
- verschwendete Texel
- Sampling-Artefakte an den Rändern der Patches

für beliebige Meshes



- keine seams
- Dreiecke in mehreren Patches
- für viele Dreiecksnetze
- Mip-Mapping okay
- alle Texel benutzt
- keine Ränder, keine Sampling-Artefakte

nur für „Kugeln“

G. Zachmann Computer-Graphik 2 - SS 08 Texturen 97

Polycube-Maps

- Polycube statt eines einzelnen Würfels
- An Geometrie und Topologie angepaßt

G. Zachmann Computer-Graphik 2 - SS 08 Texturen 98

Beispiele

G. Zachmann Computer-Graphik 2 - SS 08 Texturen 99