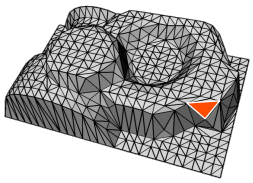
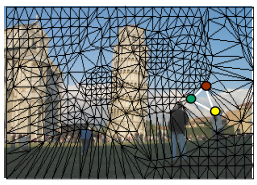
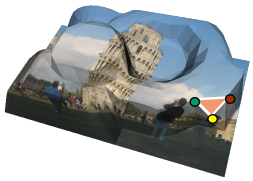
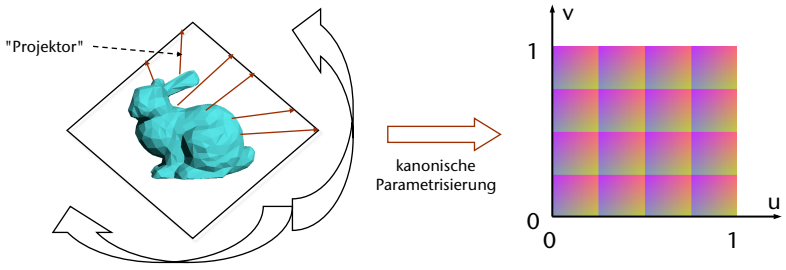
Parametrisierung

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

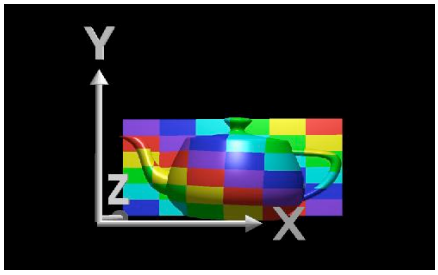
  - Achtung: dies ist nicht notwendig eine "gute" Texturierung!

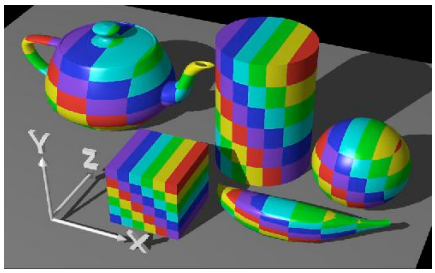
- Idee: 2-stufiger Prozess [Bier & Sloan, 1986]
  - Lege (konzeptionell) einen "kanonisch" parametrisierbaren Hüllkörper um das ganze Objekt
  1. Projiziere Vertices (nicht notw. Vertex-Koord.!) auf diesen Hüllkörper
  2. Verwende die Texturkoordinaten des projizierten Punktes auf dem Hüllkörper



Einige Hüllkörper und deren Parametrisierung

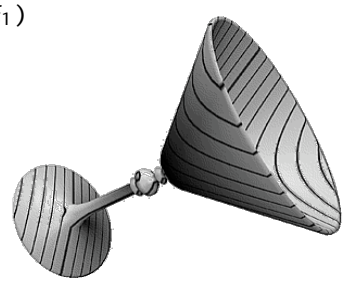
- Ebene:
  - Projiziere Punkt (x,y,z) auf Ebene $\rightarrow (x,y)$
  - $(u,v) = (s_x x + t_x, s_y y + t_y)$
- Verallgemeinerung:
  - Definiere 2 beliebige Ebenen E_1 und E_2
  - u := dist(P, E_1)
    v := dist(P, E_2)
  - Dieses Feature bietet OpenGL





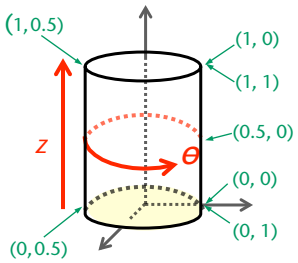
Beispiel

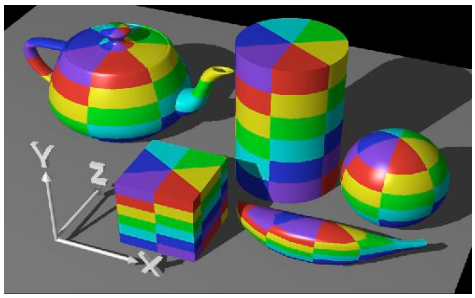
- Erzeuge Höhenlinien mittels dieser Technik:
  - 1D-Textur
  - u := dist(P, E_1)



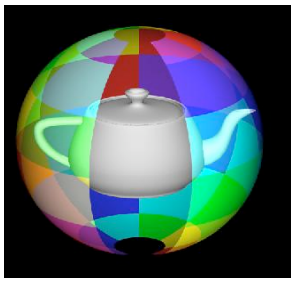
- Viele weitere ungewöhnliche Anwendungen von Texture-Mapping auf http://www.graficaobscura.com/texmap/index.html

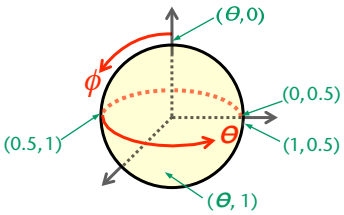
- Zylinder-Parametrisierung:
  - Konvertiere kartesische Koord. (x,y,z) in zylindrische Koord.
    $\rightarrow (r \sin \Theta, r \cos \Theta, z)$
  - $(u, v) = (\Theta/2\pi, z)$
- Beachte "Naht" bei (θ=0 & 2π)

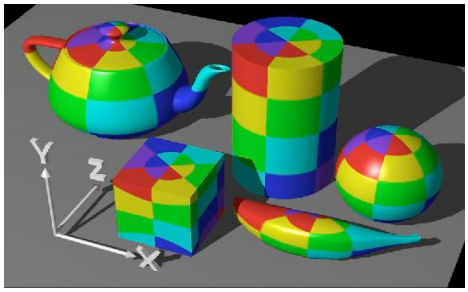




- Kugel-Parametrisierung:
  - Stelle Punkt in sphärischen Koord. dar
    $\rightarrow\ r \cdot (\sin\theta \sin\phi, \cos\theta \sin\phi, \cos\phi)$
  - $(u, v) = (\theta/2\pi, \phi/\pi + 1)$
- Beachte: Singularität bei Nord- und Südpol!



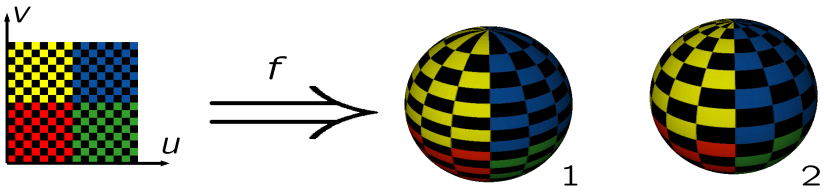




- Hier gibt es 2 Varianten:

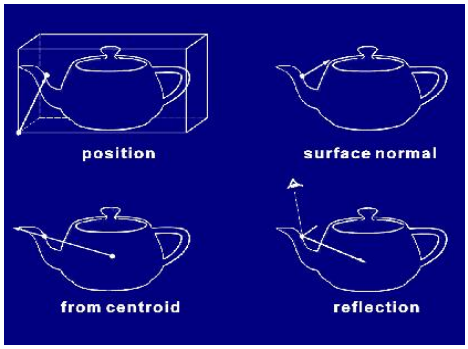
1. $f(u,v) = \begin{pmatrix} \sin(\pi v)\cos(2\pi u) \\ \sin(\pi v)\sin(2\pi u) \\ \cos(\pi v) \end{pmatrix}$

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



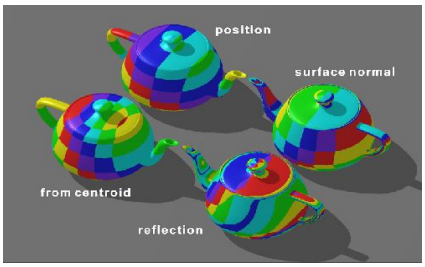
Was soll man projizieren?

- 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
  - ...

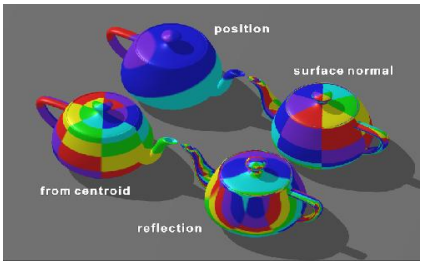


- Beispiele:

planar



cylindrical



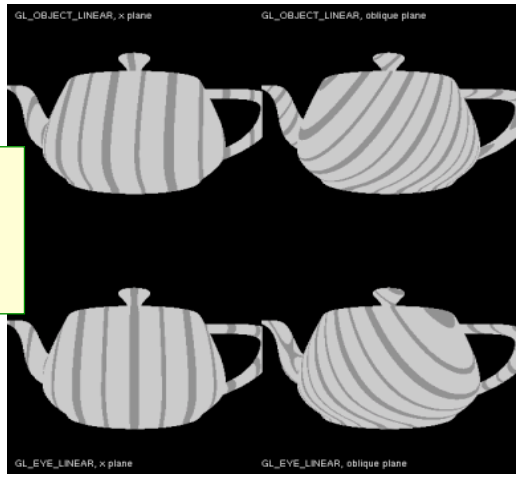
Automatische Erzeugung von Textur-Koordinaten in OpenGL

- `glEnable( GL_TEXTURE_GEN_S );` // S, T, R, Q
- `glTexGeni( GL_S, GL_TEXTURE_GEN_MODE`, *mode*);
- mode =

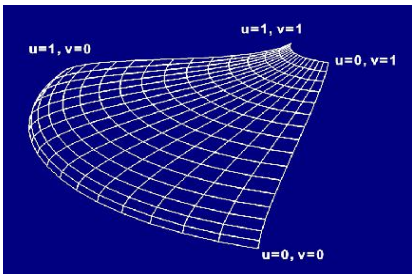
`GL_OBJECT_LINEAR`	: Texturkoord. = Distanz des Vertex von einer Ebene; die Ebene wird spezifiziert mit `glGenTexfv( GL_S, GL_OBJECT_PLANE, v )`
`GL_EYE_LINEAR`	: benutze Vertex-Koord. **nach** `MODEL_VIEW`
`GL_SPHERE_MAP`	: für Environment-Mapping (später)
`GL_NORMAL_MAP`	
`GL_REFLECTION_MAP`	

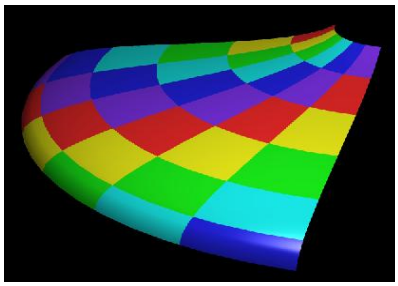
Beispiel

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



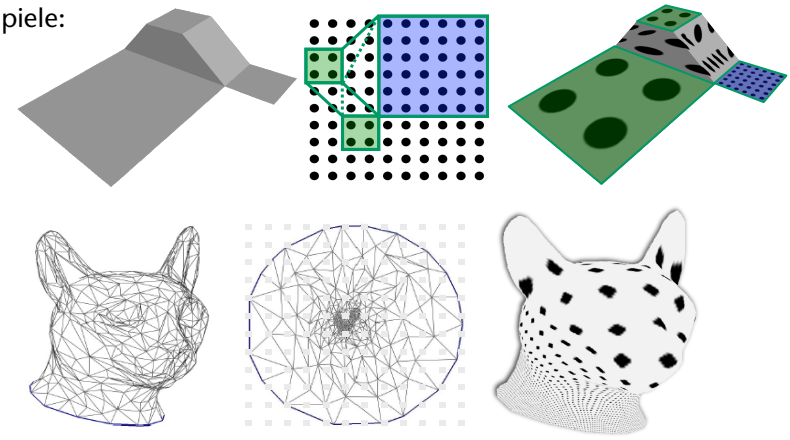
- Parametrisierung parametrischer Flächen ist oft "kanonisch":





Probleme bei der Parametrisierung

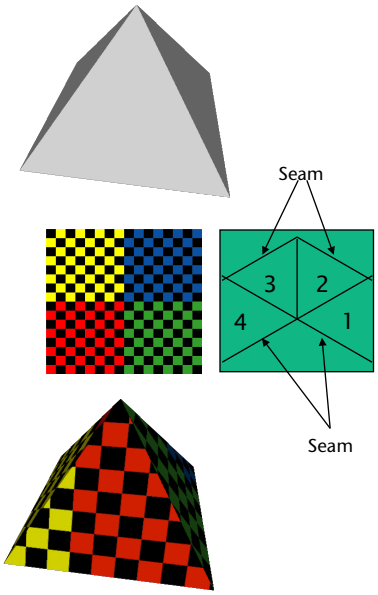
- Verzerrungen: Größe & Form
- Folge: Relatives over- bzw. under-sampling
- Beispiele:



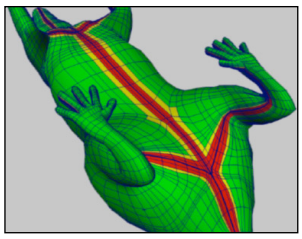
Mesh Einbettung Verzerrung

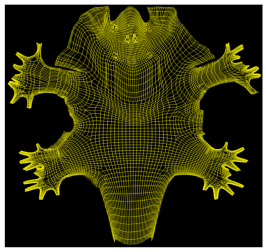
Seams (Textursprünge)

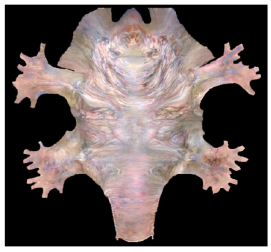
- Ziel: Verringern der Verzerrung
- Idee: Aufschneiden des Netzes entlang gewisser Kanten
- Ergibt „doppelte“ Kanten, auch „*seams*“ genannt
- Unvermeidlich bei nicht-planarer Topologie



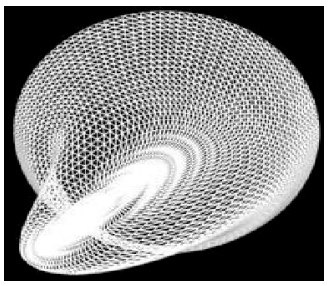
- Idee 1 [Piponi 2000]:
  - Das Objekt entlang nur einer zusammenhängenden Kante so aufschneiden, daß eine topologische Scheibe entsteht
  - Dieses aufgeschnittene Netz dann in die Ebene einbetten

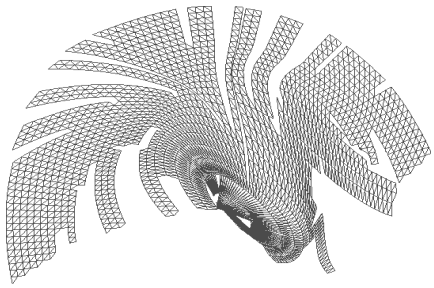




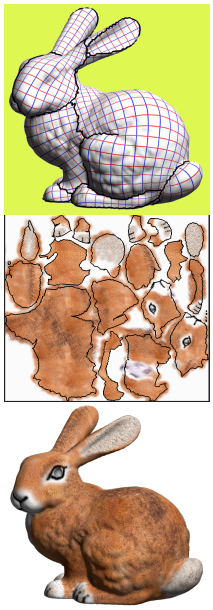


- Probleme:
  - Es gibt immer noch Verzerrungen
  - Mehrfaches "Einschneiden" ergibt stark "zerfranstes" eingebettetes Gitter

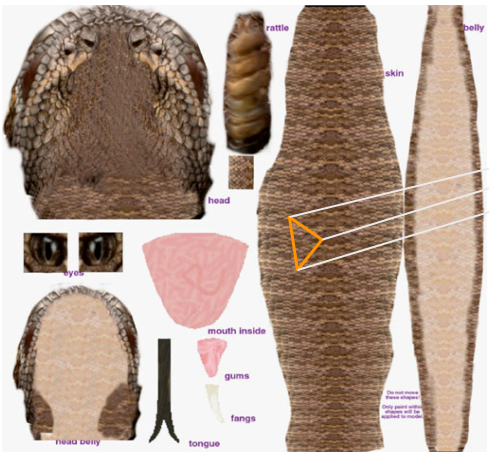


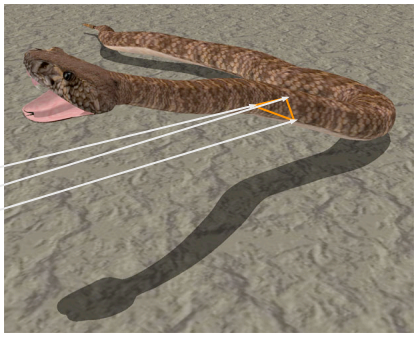


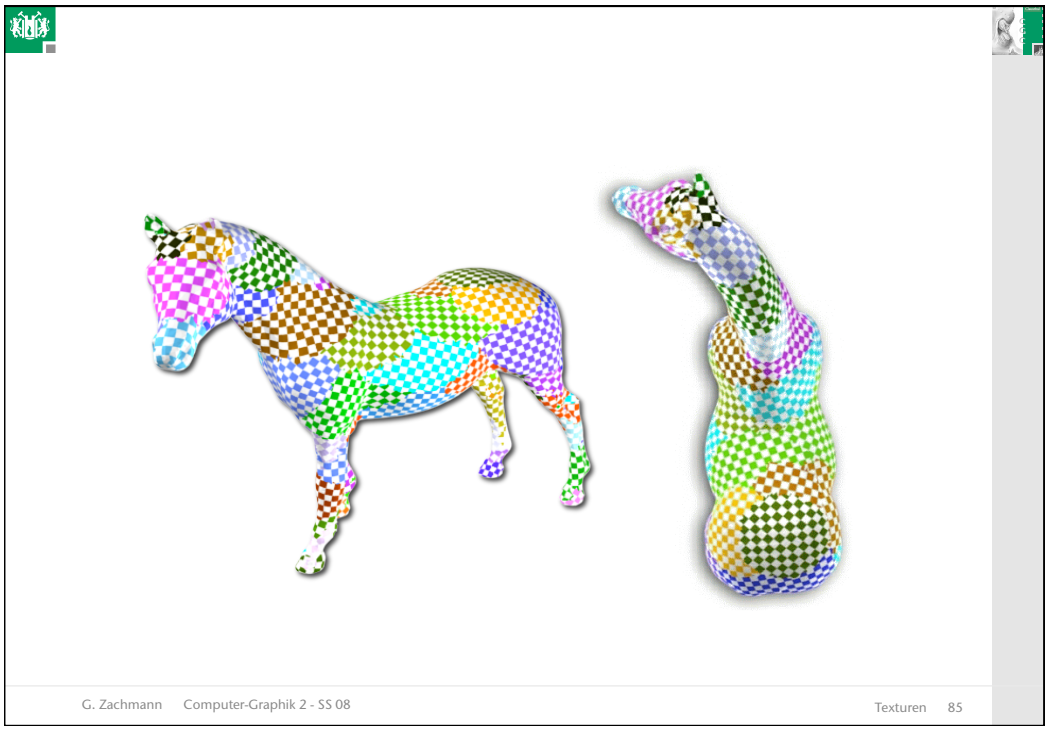
- Idee 2:
  - Zerschneide Fläche in einzelne Patches
  - Karte (map) = einzelnes Parametergebiet im Texture-Space für ein Patch
  - Textur-Atlas = Vereinigung dieser Patches mit ihren jeweiligen Parametrisierungen
- Problemstellung:
  - Wähle Kompromiß zwischen Seams und Verzerrung
  - „Verstecke“ Schnitte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



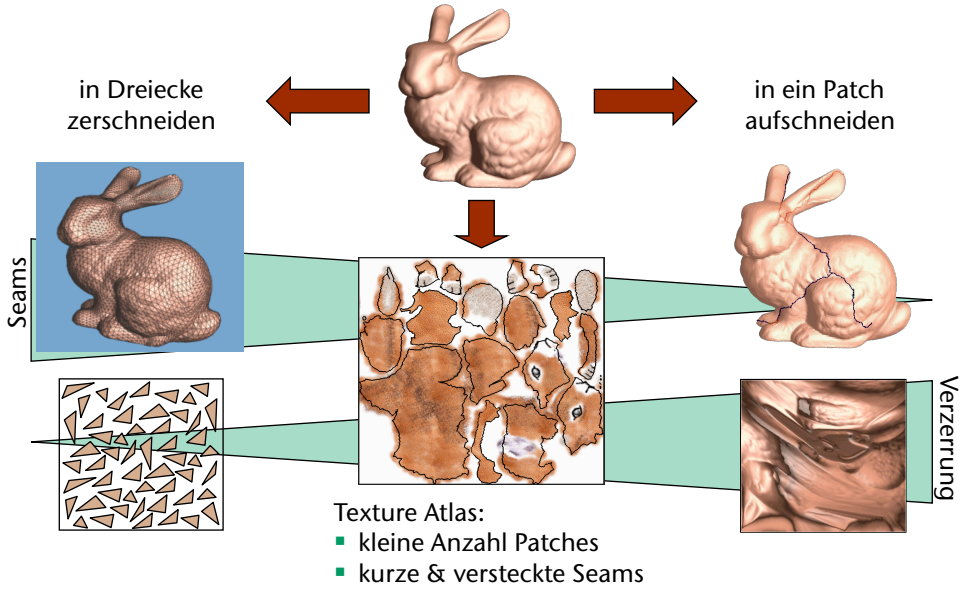
- Beispiel







Verzerrung oder Seams?



Texture Atlas:
- kleine Anzahl Patches
- kurze & versteckte Seams

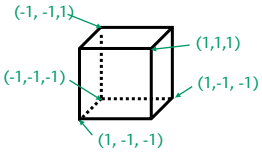
Cube Maps

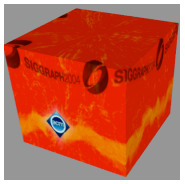
[Greene '86, Voorhies '94]

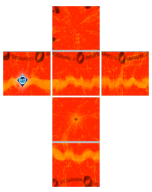
- Ω = Einheits-Würfel:
  - Sechs quadratische Textur-Bitmaps
  - 3D Texturkoordinaten:

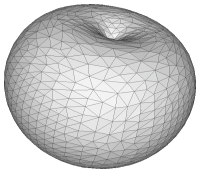
```
glTexCoord3f( s, t, r );
glVertex3f( x, y, z );
```

  - 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

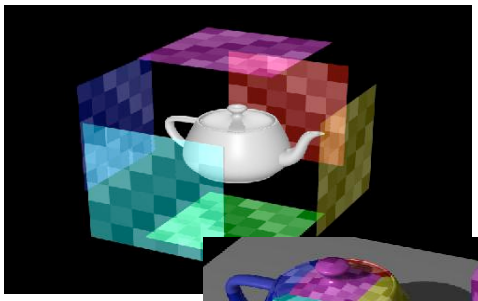


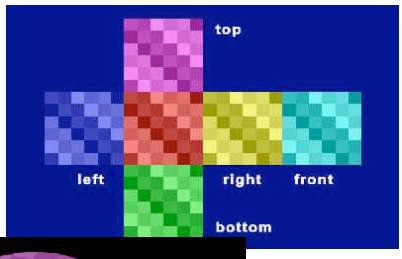


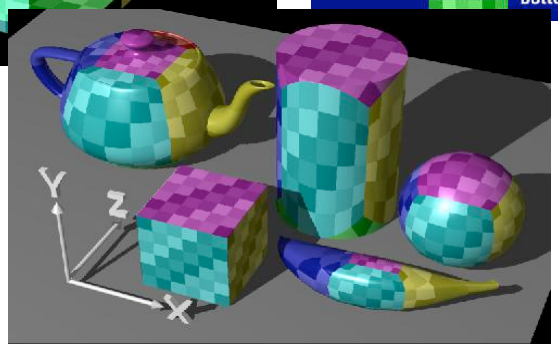




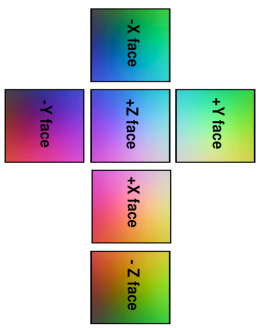
Beispiele







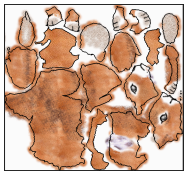
- Weitere Anwendung: man kann eine Cube-Map auch sehr gut verwenden, um irgendeine Funktion der Richtung zu speichern! (vorberechnet als LUT)
- Beispiel: Normierung eines Vektors
  - Jedes Cube-Map-Texel (s,t,r) speichert in RGB den Vektor $\frac{(s,t,r)}{\|(s,t,r)\|}$
  - Jetzt kann man beliebige Texturkoordinaten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

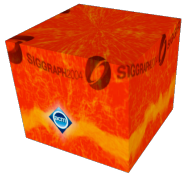


Cube-Maps in OpenGL

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

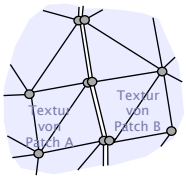
Textur-Atlas vs. Cube-Map

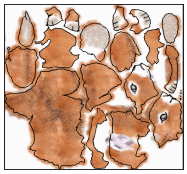



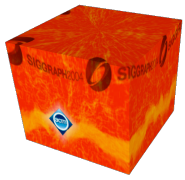


- Seams
- Dreiecke innerhalb der Patches
- nur für ein Dreiecksnetz
- Mip-Mapping schwierig
- verschwendete Texel
- Sampling-Artefakte an den Rändern der Patches

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

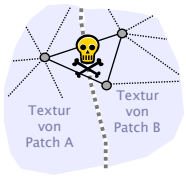


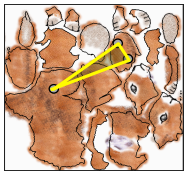


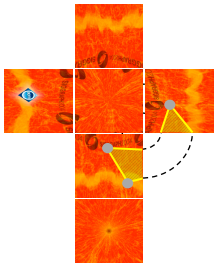


- seams
- Dreiecke innerhalb der Patches
- nur für ein Dreiecksnetz
- Mip-Mapping schwierig
- verschwendete Texel
- Sampling-Artefakte an den Rändern der Patches

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

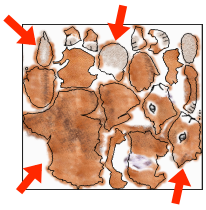


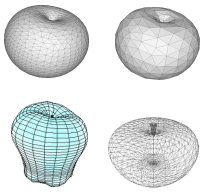


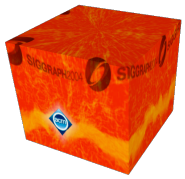


- seams
- Dreiecke innerhalb der Patches
- nur für ein Dreiecksnetz
- Mip-Mapping schwierig
- verschwendete Texel
- Sampling-Artefakte an den Rändern der Patches

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

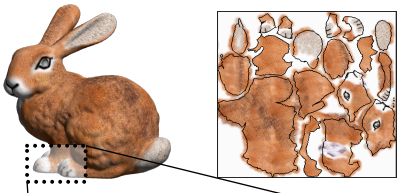


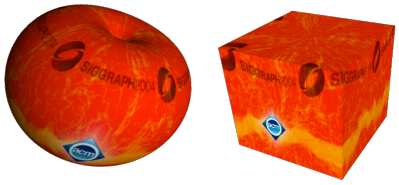




- seams
- Dreiecke innerhalb der Patches
- nur für ein Dreiecksnetz
- Mip-Mapping schwierig
- verschwendete Texel
- Sampling-Artefakte an den Rändern der Patches

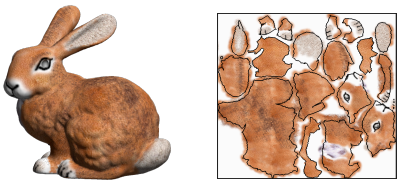
- keine seams
- Dreiecke in mehreren Patches
- für viele Dreiecksnetze
- Mip-Mapping okay
- alle Texel benutzt
- keine Ränder, keine Sampling-Artefakte

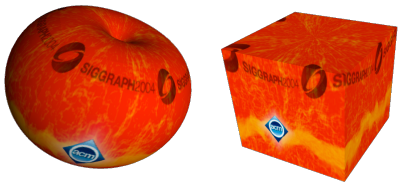




- tches
- verschwendete Texel
- Sampling-Artefakte an den Rändern der Patches

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





- seams
- Dreiecke innerhalb
- nur für ein Dre
- Mip-Mapp
- versch
- S
- Rändern der Patches

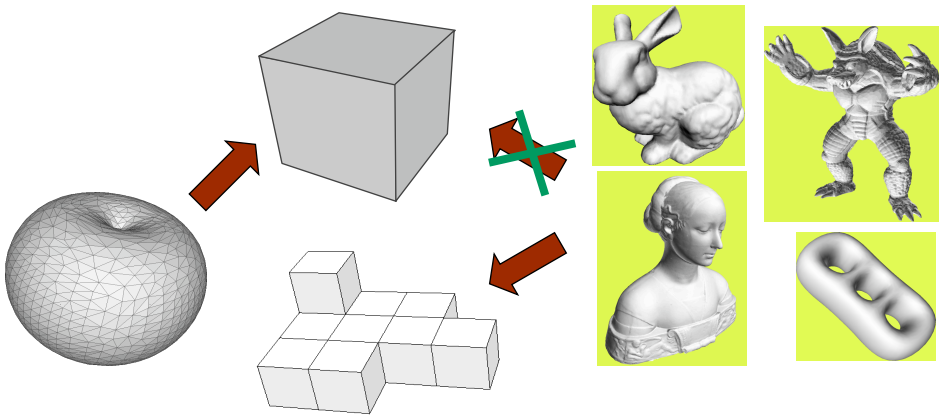
für beliebige Meshes

- keine seams
- Dreiecke in mehreren Pat
- für viele Dreiecksnetze
- Mip-Mapping ok
- alle Texel be
- keine R
- Artefakt

nur für „Kugeln“

Polycube-Maps

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



Beispiele