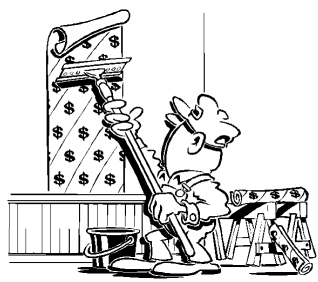


Computer-Graphik II

Texturen



G. Zachmann
Clausthal University, Germany
cg.in.tu-clausthal.de



Motivation

- Was fehlt? ...



G. Zachmann Computer-Graphik 2 – SS 10 Texturen 2

- ... Oberflächendetails

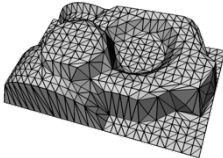




"Shutter bug", Pixar

- Großes Spektrum geometrischer Formen und physikalischer Materialien:
 - Strukturen unebener Oberflächen, z.B. Putzwände, Leder, Schale/Rinde von Orangen, Baumstämme, Maserungen in Holz und Marmor, Tapeten mit Muster, etc.
 - Wolken
 - Objekte im Hintergrund (Häuser, Maschinen, Pflanzen und Personen)
- Solche Objekte durch Flächen nachzubilden ist in der Regel viel zu aufwendig
- Mit Texturen kann man Objekte **visuell komplexer** gestalten:
 - Die Wand kann durch ein Rechteck modelliert werden und die Tapete wird **als Bild** aufgebracht
- Dies nennt man **Texturierung**

Grundidee der Texturierung

- Objekt mit Textur „tapezieren“
- Visuelles Detail trotz grober Geometrie

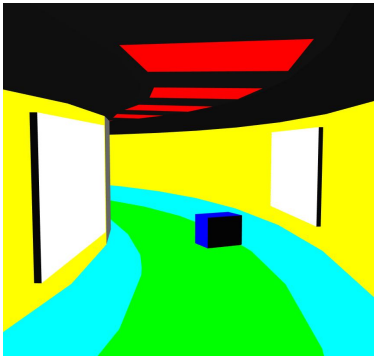
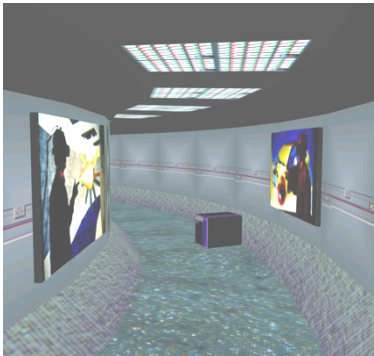

+

=


Objekt (Geometrie)
Textur (Farbe)

- Ursprung: Catmull (1974), Blinn and Newell (1976), u.a.

G. Zachmann Computer-Graphik 2 – SS 10
Texturen 5

Weitere Beispiele

G. Zachmann Computer-Graphik 2 – SS 10
Texturen 6

- Kaustik durch Texturen verstärkt den Unterwassereindruck



G. Zachmann Computer-Graphik 2 – SS 10 Texturen 7


Übersicht

- Arten von Texturen: **diskret** oder **prozedural**
- **Dimension** der Texturen: 1D, 2D, 3D, 4D(?)
- Wichtige Punkte bei den diskreten 2D-Texturen:
 - **Interpolation** der Texturkoordinaten
 - **Anwendung** der Textur auf die **Beleuchtung** o. a. Oberflächeneigensch.
 - **Parametrisierung** der Fläche
 - **Filterung**
- Wie funktioniert es in OpenGL
- **Environment-Mapping**

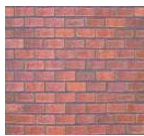
G. Zachmann Computer-Graphik 2 – SS 10 Texturen 8

Texturarten

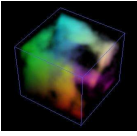
1D Texturen




2D Texturen



3D Texturen



Cubemap
Texturen



- Textur kann als Funktion einer, zweier oder dreier Koordinaten, oder als Funktion einer Richtung gesehen werden

G. Zachmann Computer-Graphik 2 – SS 10
Texturen 9


Einfacher Fall: 3D-Texturen


- 3D-Texturen nennt man auch Festkörper-Texturen (z.B. Holz und Marmor) ("*solid texture*")
- Die **lokalen Koordinaten** der Obj.oberfläche (x,y,z) indizieren direkt die Textur:

$$(r, g, b) = C_{\text{tex}}(x, y, z)$$

- Die Textur ist also an **jedem** Punkt im Raum definiert
- Das Objekt wird quasi aus dem Texturvolumen "**herausgeschnitzt**"

2D-
Texturierung





3D-
Texturierung

G. Zachmann Computer-Graphik 2 – SS 10
Texturen 10

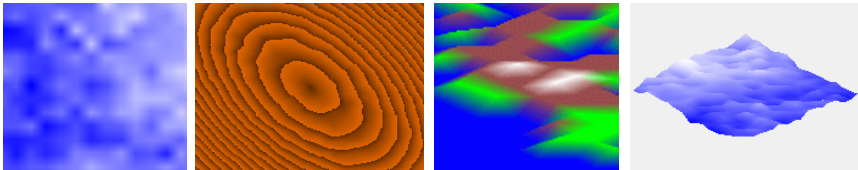
■ Beispiele:



G. Zachmann Computer-Graphik 2 – SS 10 Texturen 11

■ Diskrete und prozedurale Texturen

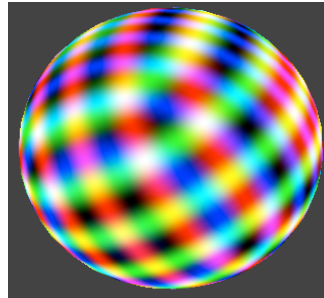
- Man unterscheidet diskrete und prozedurale Texturen
- Eine **diskrete 3D-Textur** = 3-dimensionales Array $C[u,v,w]$
 - $C[u,v,w]$ = Vektor mit 3 Farbkomponenten, ein "Texel" (*texture element*)
 - Pro Pixel benötigt man 3 **Texturkoordinaten** (u,v,w) zum Indizieren in das Array
- **Prozedurale Texturen** werden bei jedem Auslesen aus math. Funktion oder fraktalem Algorithmus berechnet

$$C_{\text{tex}}(x, y, z) := f(x, y, z)$$


G. Zachmann Computer-Graphik 2 – SS 10 Texturen 12

- Einfaches Beispiel für eine prozedurale 3D-Textur:

$$C = \begin{pmatrix} \frac{1}{2}(1 + \sin(\frac{\pi}{w_x} P_x)) \\ \frac{1}{2}(1 + \sin(\frac{\pi}{w_y} P_y)) \\ \frac{1}{2}(1 + \sin(\frac{\pi}{w_z} P_z)) \end{pmatrix}$$



- **Vorteile** der prozeduralen Texturen:
 - Speicheraufwand ist minimal
 - Texturwerte können an **jeder** Stelle (u,v) , bzw. (u,v,w) berechnet werden
 - Optimale Genauigkeit (kein Runden von Koord., keine Interpolation)
 - Texturen sind im gesamten Raum definiert (kein Wrap-Around / Clamping)
- **Nachteile:**
 - Schwer zu erzeugen (selbst für Experten)
 - Mindestens Grundkenntnisse der Fourier-Synthese, bzw. fraktaler Geometrie erforderlich
 - Komplexere Texturen nahezu unmöglich
 - Kosten rel. viel Zeit (Echtzeit?)

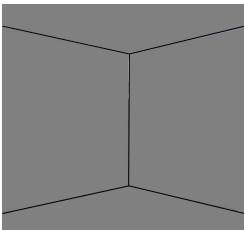
Diskrete 2D-Texturen

- **Vorteile:**
 - Vorrat an Bildern nahezu unerschöpflich
 - Erzeugung ist einfach (z.B. Photographie)
 - Anwendung auf eine Oberfläche ist sehr schnell
- **Nachteile:**
 - Kontext (Sonnenstand, Schattenwurf, etc.) stimmt meist nicht
 - Bilder hoher Auflösung haben großen Speicherbedarf
 - Fortsetzung meist sehr kompliziert
 - Beim Vergrößern und Verkleinern treten Artefakte auf
 - Verzerrung beim Mapping auf beliebige Fläche

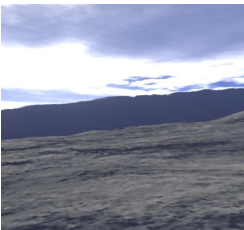
G. Zachmann Computer-Graphik 2 – SS 10 Texturen 15

Beispiel 1: Skybox

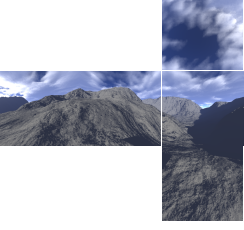
- Die Umgebung einer virtuellen Szenen modelliert man oft durch eine Kugel oder einen Würfel mit entsprechenden Texturen



Ohne Skybox



Die Skybox

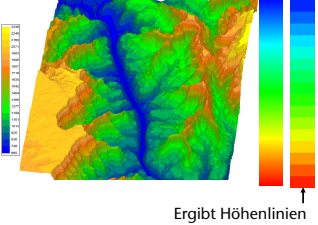
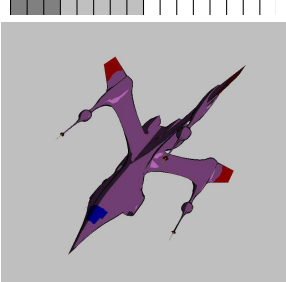


Vom Boden aus

G. Zachmann Computer-Graphik 2 – SS 10 Texturen 16

1D Texturen

- In der Visualisierung möchte man oft einen Parameter durch **Falschfarben-darstellung** intuitiv erfassbar machen
 - z.B. Höhe auf einem Terrain, Temperatur ...
 - Verwende dazu eine 1D-Textur mit einer Farbskala
 - Parameter (z.B. Höhe = y-Koord.) → 1D-Textur-Koord.
- Toon Shading:
 - Berechne Punktprodukt des Licht- und des Normalenvektor oder das Punktprodukt der View- und des Normalenvektors
 - Verwende das als Index in die Farbtabelle (1D-Textur)

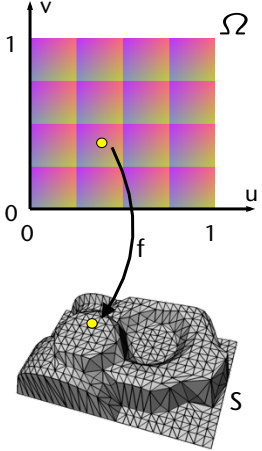
G. Zachmann Computer-Graphik 2 – SS 10 Texturen 17

Formalisierung

- Zu texturierendes Objekt $S = \text{Dreiecks-Mesh}$
- Textur** :=
 - Parameterraum Ω
 - Pixelbild oder Funktion (diskret / prozedural)
 - Parametrisierung / Mapping** = Abbildung f zwischen Textur und Objekt:
$$f : \Omega \leftrightarrow S$$
- Texturierung** ist ein 2-stufiger Prozeß
 - Inverses Mapping:

$$(u, v) = f^{-1}(x, y, z)$$
 - Farbe:

$$(r, g, b) = C_{\text{tex}}(u, v)$$



G. Zachmann Computer-Graphik 2 – SS 10 Texturen 18

Texturkoordinaten

- Texturierung eines kompletten Dreiecksnetzes:



- Für jeden Eckpunkt müssen zusätzlich **Texturkoordinaten** definiert / berechnet werden, die angeben, welcher Ausschnitt aus der Textur auf das Polygon gemappt wird

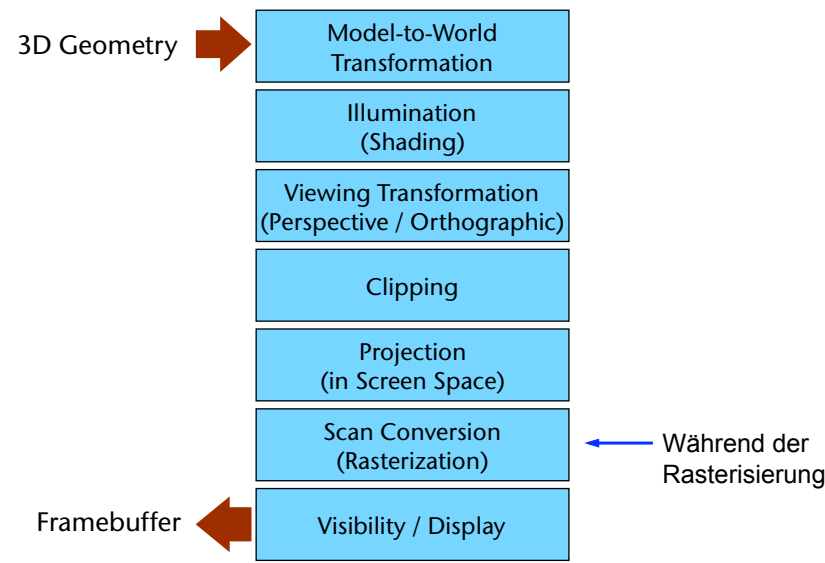


```

glBegin( GL_... )
  glTexCoord2f (...);
  glNormal3f (...);
  glVertex3f (...);
  ...
glEnd();
    
```

G. Zachmann Computer-Graphik 2 – SS 10 Texturen 19

Wo in der Pipeline wird texturiert?



3D Geometry →

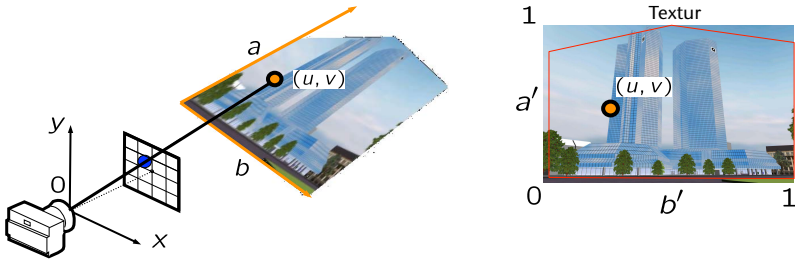
- Model-to-World Transformation
- Illumination (Shading)
- Viewing Transformation (Perspective / Orthographic)
- Clipping
- Projection (in Screen Space)
- Scan Conversion (Rasterization) ← Während der Rasterisierung
- Visibility / Display

Framebuffer ←

G. Zachmann Computer-Graphik 2 – SS 10 Texturen 20

Interpolation der Texturkoordinaten

- Bei der Rasterisierung wird für jedes Pixel eine Texturcoordinate (u,v) interpoliert
- Diese bestimmt im Koordinatensystem der Textur das Texel, das auf das Pixel gemapt wird



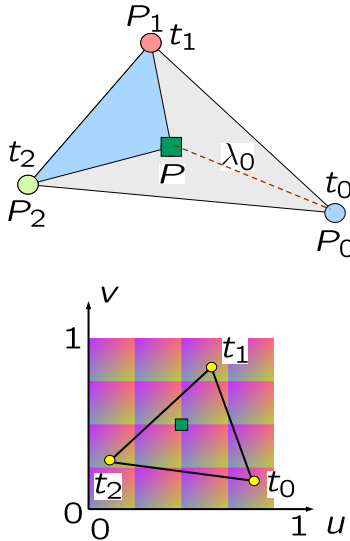
G. Zachmann Computer-Graphik 2 – SS 10 Texturen 21

Generierung der Textur-Koordinaten pro Fragment im Rasterizer

- Baryzentrische Koordinaten

$$\lambda_i(P) = \frac{A(P, P_{i-1}, P_{i+1})}{A(P_0, P_1, P_2)}$$
- Gewichtete lineare Interpolation

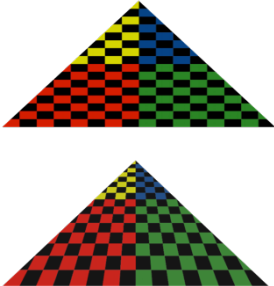
$$t(P) = \sum_{i=0}^2 \lambda_i(P) t_i$$

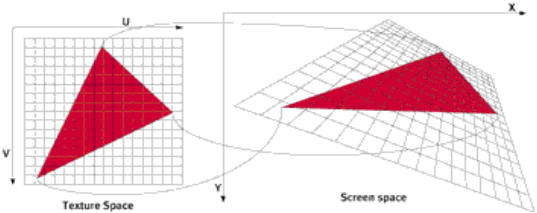



G. Zachmann Computer-Graphik 2 – SS 10 Texturen 22

Perspektivisch korrekte Texturkoordinateninterpolation

- Problem: bei dieser einfachen, linearen Interpolation im Screen Space entstehen perspektivisch inkorrekt Bilder!
- Ziel: perspektivisch korrekte Interpolation
- Problem: der Rasterizer hat die Koordinaten nur **nach** der perspektivischen Division!







Demo (mehr auf der VL-Homepage)

G. Zachmann Computer-Graphik 2 – SS 10
Texturen 23