Grundlagen der Spieleprogrammierung

Teil I: 3D-Graphik

Kapitel 6: Texturem

Peter Sturm
Universität Trier

Outline

1. Übersicht und Motivation
2. Mathematische Grundlagen
3. Das Ideal: Photorealistisch (Raytracing, Radiosity)
4. Die Realität: DirectX und OpenGL (Übersicht)
5. Schritt 1: Modellierung und Drahtgitter
6. Schritt 2: Texturen
7. Schritt 4: Licht, Filter, etc.
8. Schritt 5: Fortgeschrittene Techniken (Vertex-, Pixel-Shader, …)
9. 3D-Hardware
10. 3D-Engines im Überblick, Cg von nvidia
11. Spielekonsolen
12. Zusammenfassung und Ausblick

Motivation

- Realistisch modellierbare Eigenschaften der Oberflächenbeschaffenheit
  - Reflektionsgrad
  - Refraktionsgrad und Brechungsindex (IOR)
  - Rauhheit (Variation des Normalenvektors)
  - Filtereigenschaften bzgl. Farbe (Reflektion und Refraktion)
- Was macht Holz zu Holz, Metall zu Metall?
- Physikalische Modellierung praktisch unmöglich
- Alternativansatz
  - „Bekleben" der Objektoberfläche mit einer Tapete (Textur)

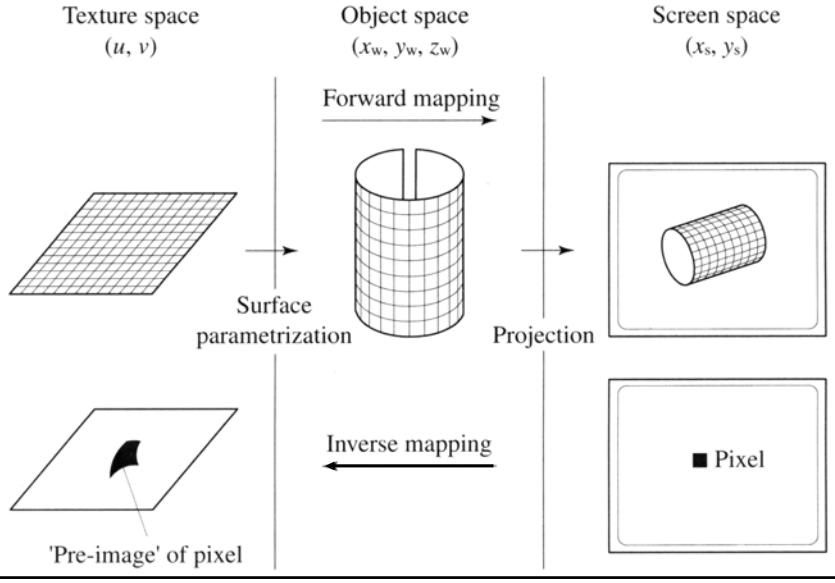
Einsatzgebiete

- Texture
  - Visuell sichtbare Oberflächenstruktur
  - Color Map
- Light Map
  - Vereinfachte Darstellung von Licht und Schatten
- Bump Map
  - Vereinfachte Darstellung der Oberflächenbeschaffenheit
- Environment Map
  - Umgebung wird von Objekt reflektiert

Komplizierter als man denkt …

- Formatunterschiede
  - Abbildung meist rechteckiger Texturen auf das Drahtgitter (Dreiecke, etc.)
- Dimensionsunterschiede
  - Abbildung der 2D-Textur auf 3D-Objekte
- Alias-Effekte
  - Größenunterschiede
  - Wiederholungen
  - Räumliche Tiefe und Texturen

Abbildung zweidimensionaler Texturen



Darstellung von Alan Watt

Inverse Mapping

- Direkte 2D-Abbildung: Rechteck auf Rechteck

$$(x_s, y_s) \rightarrow (u, v)$$

- Projektive Transformation:

$$x = \frac{au + bv + c}{gu + hv + i}, y = \frac{du + ev + f}{gu + hv + i}$$

  - i auf 1 setzen
  - 8 Gleichungen, 8 Unbekannte

- Alternative Sichtweise

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

Inverse binlineare Interpolation

- Gegeben:

$(x_0, y_0, u_0, v_0) \ldots (x_3, y_3, u_3, v_3)$

  - Ecken des Rechtecks
  - Zugeordnete Koordinaten in der Textur

(x_s, y_s, u_s)

  - Bildschirmkoordinate inkl. Tiefe

  – Projektionsmatrix T (Rechteck auf View)

- Berechne

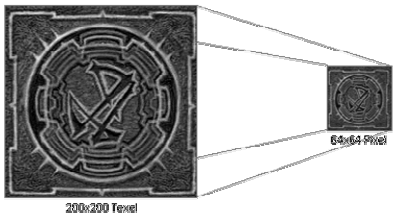
$$(x_t, y_t, u_t) = T^{-1} \cdot (x_s, y_s, u_s)$$

Color Mapping

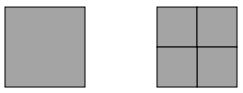
- Verwendung einer "Texture Map"
- Liefert Farbwert an gegebenem Punkt
- Zugriff im Verlauf der Projektion auf View (Z-Buffer)
- Probleme
  - Untypischer Zugriff (Caching)
  - Speicherplatz
- Zugang über inverse Projektion
- Beispiel: Tutorial

Qualitätsaspekte

- Alias-Probleme beim Abbilden
  - Pixel entspricht großem Flächenausschnitt (weit weg)
  - Pixel entspricht kleinem Flächenausschnitt (nah dran)
  - Rundunsfehler führen zu "Shimmering"-Effekten



- Alternativen
  - Supersampling
    - Höhere "interne" Pixelauflösung
    - Z.B. 4x4 Supersampling
    - Sehr aufwendig
  - Mip Mapping

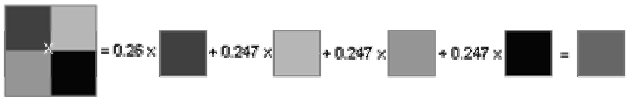


Mip Mapping

- Vorheriges Downsampling der Texturen
- Speicherung aller Versionen auf der Graphikkarte
  - Zwischen 5 und 10 Versionen
- Geeignete LOD (Level of Detail) bestimmen

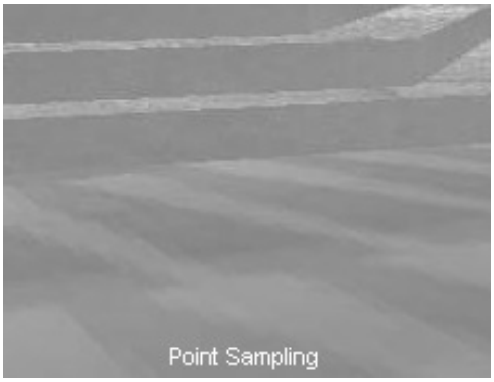
Weitere Verbesserungen

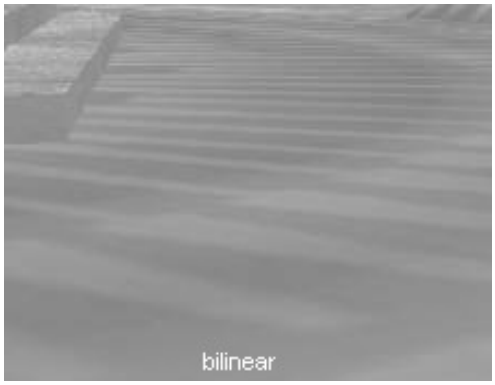
- Vermeiden des Pixelflakerns (Shimering)
  - Farbwechsel je nach Standort des Betrachters
- Filtertechniken
  - Point Sampling – Runden von (u,v) und Zugriff (kein Filter)
  - Bilinear
    - Gewichtetes Mittel benachbarter Texel



  - Mip Mapping
    - LOD pro Pixel
    - LOD pro Dreieck (schlecht bei räumlicher Tiefe)

Beispiel: Point Sampling vs. Bilinear

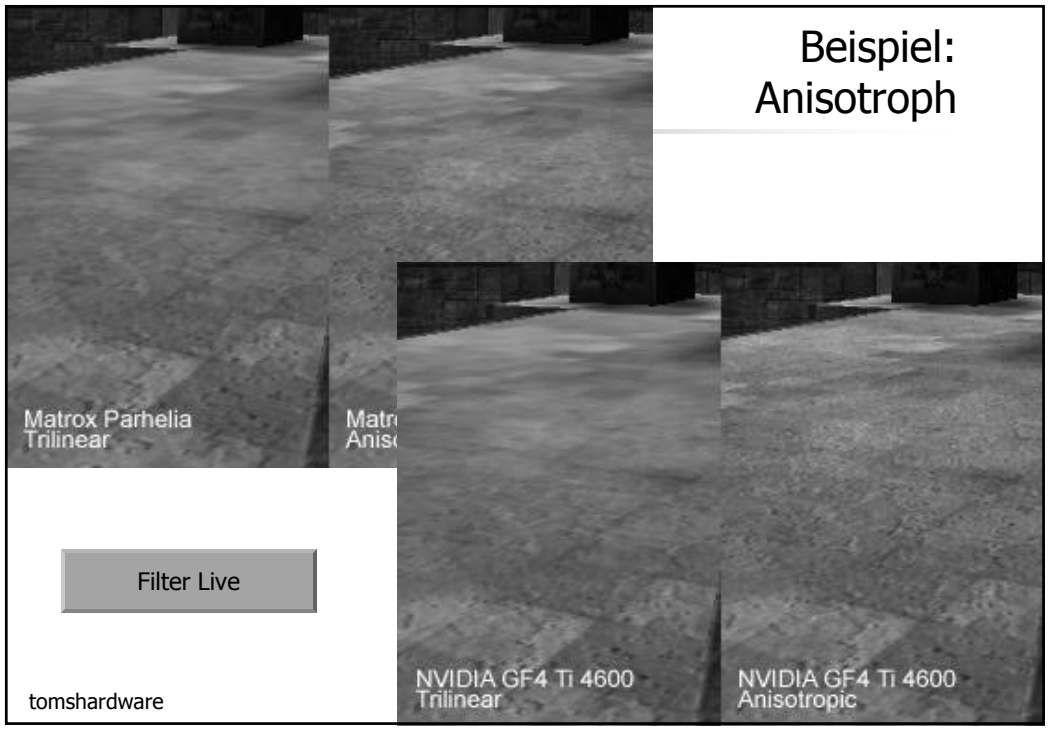




Weitere Filtertechniken

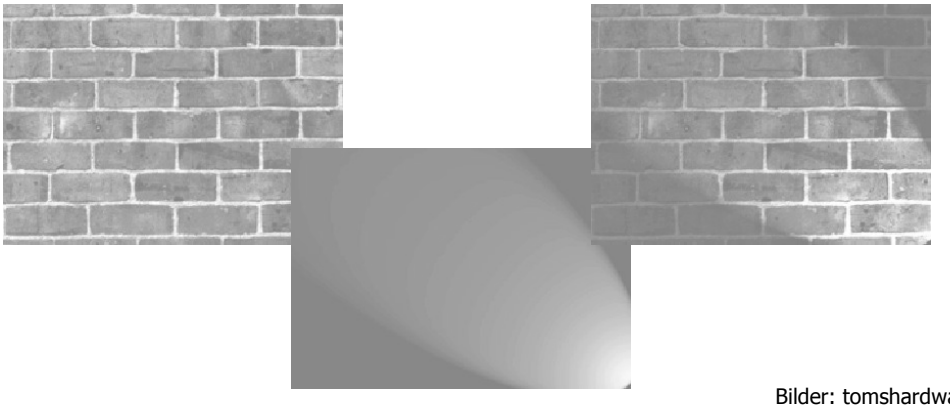
- Trilineares Filtern
  - Verbessern des MIP Bandings (Welche LOD Textur ist optimal?)
  - Abbrüche beim Übergang der Texturen
  - Bilineares Mitteln der Texel in beiden MIP Texturen
  - Lineares Mitteln über die Tiefe
  - Insgesamt 8 Texel wirken auf Pixel
- Anisotrophes Filtern
  - Uneinheitlicher, die räumliche Tiefe, berücksichtigender Filter
  - Noch kein standardisiertes Verfahren

from www.3dconcept.ch
JPEG compressed
2:52
3:50
r players
Beispiel: Trilinear
Übergang sichtbar
bilinear gefiltert (5X)
trilinear gefiltert (5X)
0
0
0
0


Beispiel:
Anisotroph
Matrox Parhelia
Trilinear
Filter Live
tomshardware
NVIDIA GF4 Ti 4600
Trilinear
NVIDIA GF4 Ti 4600
Anisotropic

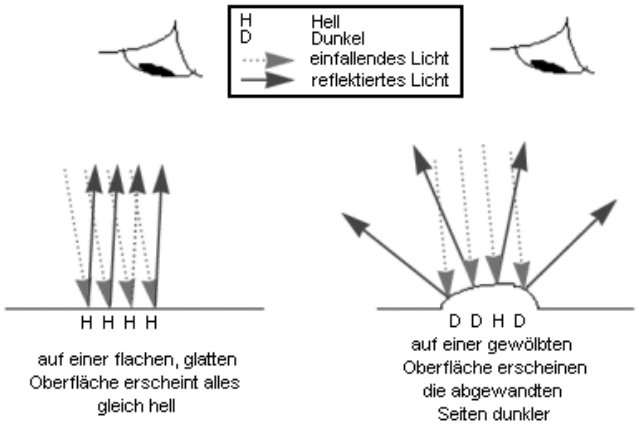
Light Mapping

- Vorberechneter Licht- und Schattenwurf
- Überlagern eines Punktes mit zwei Texturen



Bilder: tomshardware

Bump Mapping



- Flachheit der Texturen wirkt unnatürlich
- Viele "echte Texturen" entstehen durch kleine Variationen in der Oberflächengeometrie
- Modellierung durch höhere Polygonanzahl unbezahlbar

Dot3 Bump Mapping

- Bestimmen der Normalen an einem Punkt O(u,v)
- Neue Oberflächenposition ergibt sich aus

$$O'(u,v) = B(u,v) \cdot N$$

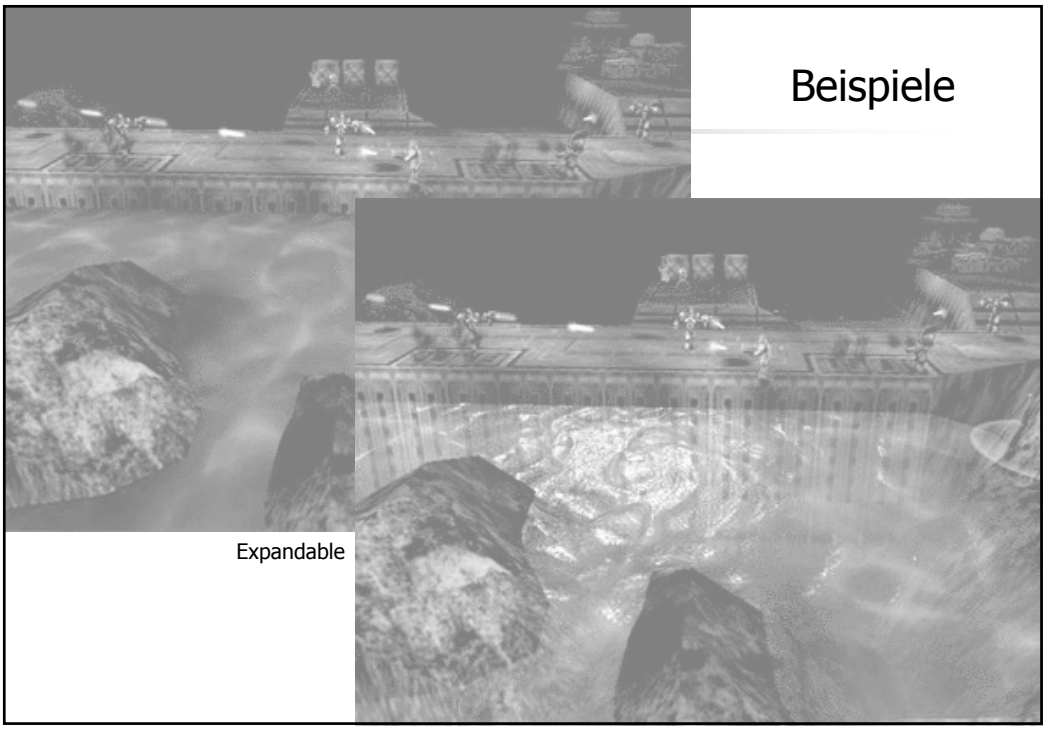
$$N = \frac{O_u \times O_v}{\left| O_u \times O_v \right|}$$

- Differenzieren liefert für kleine B

$$O'_u = O_u + B_u N$$

$$O'_v = O_v + B_v N$$

Beispiele

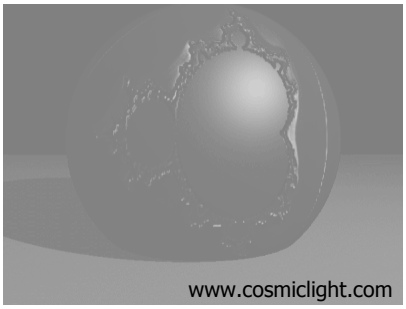


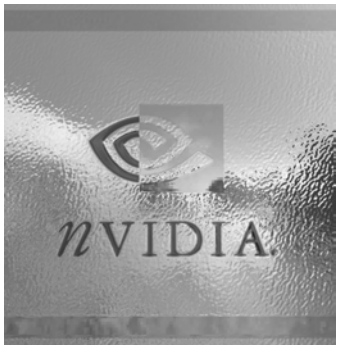
Expandable

Variationen

- Bump Map
- Prozedurales Bump Mapping
  - Displacement Mapping
  - Vertex Shader
- In Kombination mit Textur
  - Texture Map bildet Grundlage für Bump Map

Displacement Mapping



Environmental Mapping

- Spiegelung der Umgebung in reflektiven Oberflächen
- Ansatz
  - Kubus nähert den umgebenden Himmel an
  - 6 Kubusflächen berechnen
  - Bei Schnittberechnung
    - Reflektionsvektor ist Index in erreichte Kubusfläche
- Vorzugsweise Standortunabhängig vorberechnet
  - Nur Annäherung