Inhaltsverzeichnis - Themen

7. Abbildungsverfahren

Motivation

Reale Umgebung verfügt über ein großes Spektrum geometrischer Formen und physikalischer Materialien

- Maserungen und Muster von Oberflächen
- Strukturen unebener Flächen
- Hintergrund und Spiegelungen mit hohem Detailgrad

Exakte Nachbildung dieser Objekte ist meist zu aufwändig.

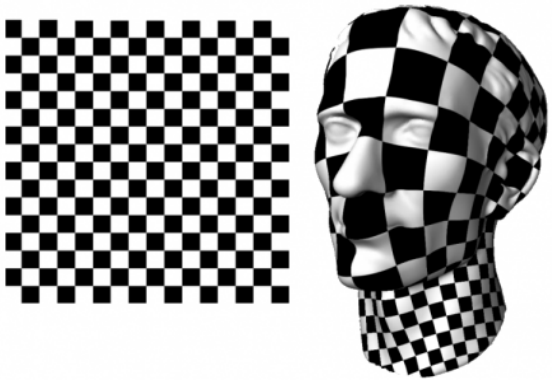
7. Abbildungsverfahren

Überblick

1. Texture-Mapping:
   Muster auf eine glatte Fläche aufbringen
2. Bump-Mapping:
   Glatte Oberflächen rauh erscheinen lassen
3. Environment-Mapping:
   Umgebung auf einer Fläche abbilden
4. Weitere Verfahren:
   1. Chrome-Mapping
   2. Displacement-Mapping
   3. Opacity-Mapping

Was ist Texture-Mapping?

- Aufbringen von 2D-Texturen auf eine 3D-Oberfläche
- Ermöglicht komplexe Gestaltung einfacher Objekte



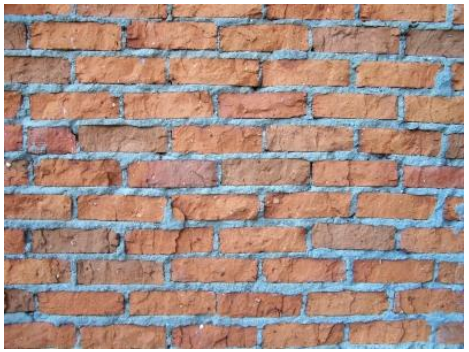
Beispiel

- Blick aus dem Fenster
- Spiegelbild
- Parkettboden

7.1 Texture-Mapping

Texturen

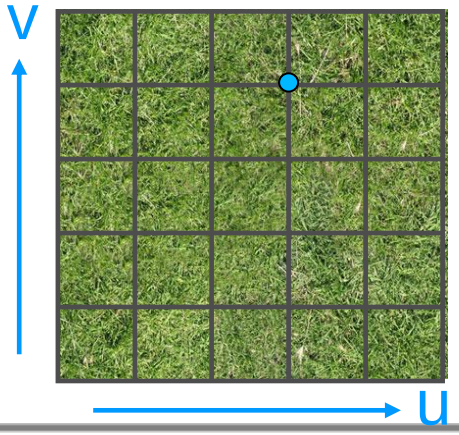





- Funktionen, die Punkte des (u,v)-Texturraums auf (r,g,b)-Werte abbilden:

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

$$mit \quad u,v \in [0;1]$$



Diskrete Texturen

- Diskret: Werden als Vektorfelder C abgespeichert.
  Ein Vektor $C[i, j]$ enthält Farbkomponenten und wird als Texel bezeichnet.
- Vorteile
  - Vorrat unerschöpflich (Smartphone, Fotoapparat, Scanner, Downloads…)
  - Photorealismus möglich
- Nachteile
  - Hoher Speicherbedarf
  - Unstimmiger Kontext von Szene und Textur (Schattenwurf…)
  - Anfällig für Artefakte und Aliasing
  - Rekonstruktion der Texturwerte notwendig

Prozedurale Texturen

- Prozedural: Bei jedem Aufruf von $C_{tex}(u,v)$ wird eine mathematische Formel bzw. ein Algorithmus ausgewertet.
- Vorteile
  - Minimaler Speicheraufwand
  - Texturen im gesamten Raum definiert
  - Auflösungsunabhängig
- Nachteile
  - Mathematische Beschreibung komplexer Texturen ist schwierig.
  - Nicht photorealistisch

diskret prozedural

7.1 Texture-Mapping

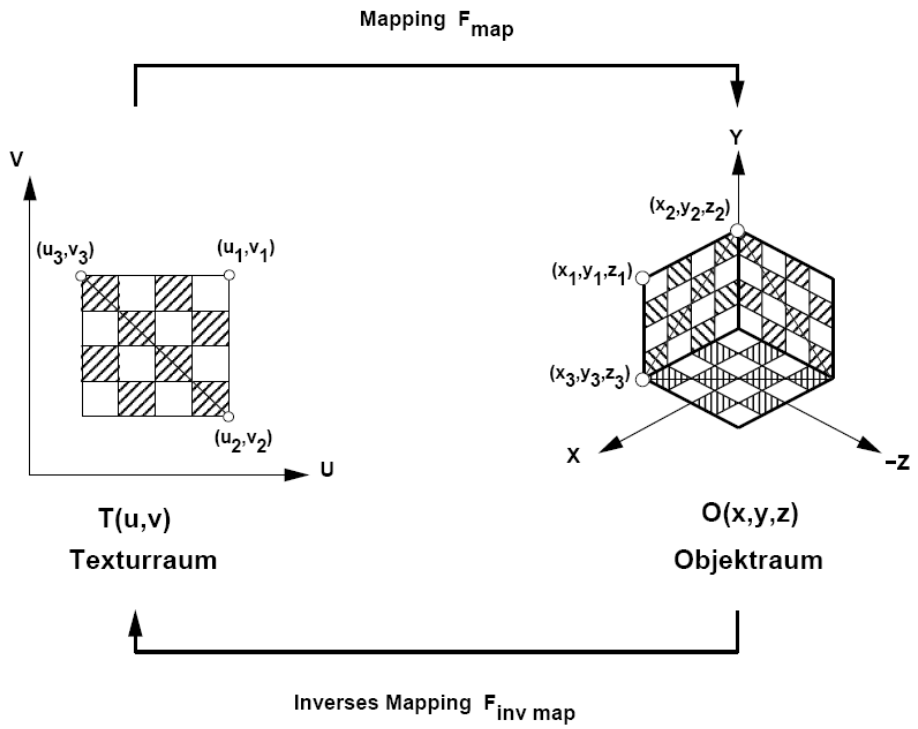
Prinzip

- Forward-Mapping:

$$(x, y, z) = F_{map}(u, v)$$

- Zur Visualisierung muss das inverse Mapping-Problem gelöst werden:

$$(u, v) = F_{invmap}(x, y, z)$$

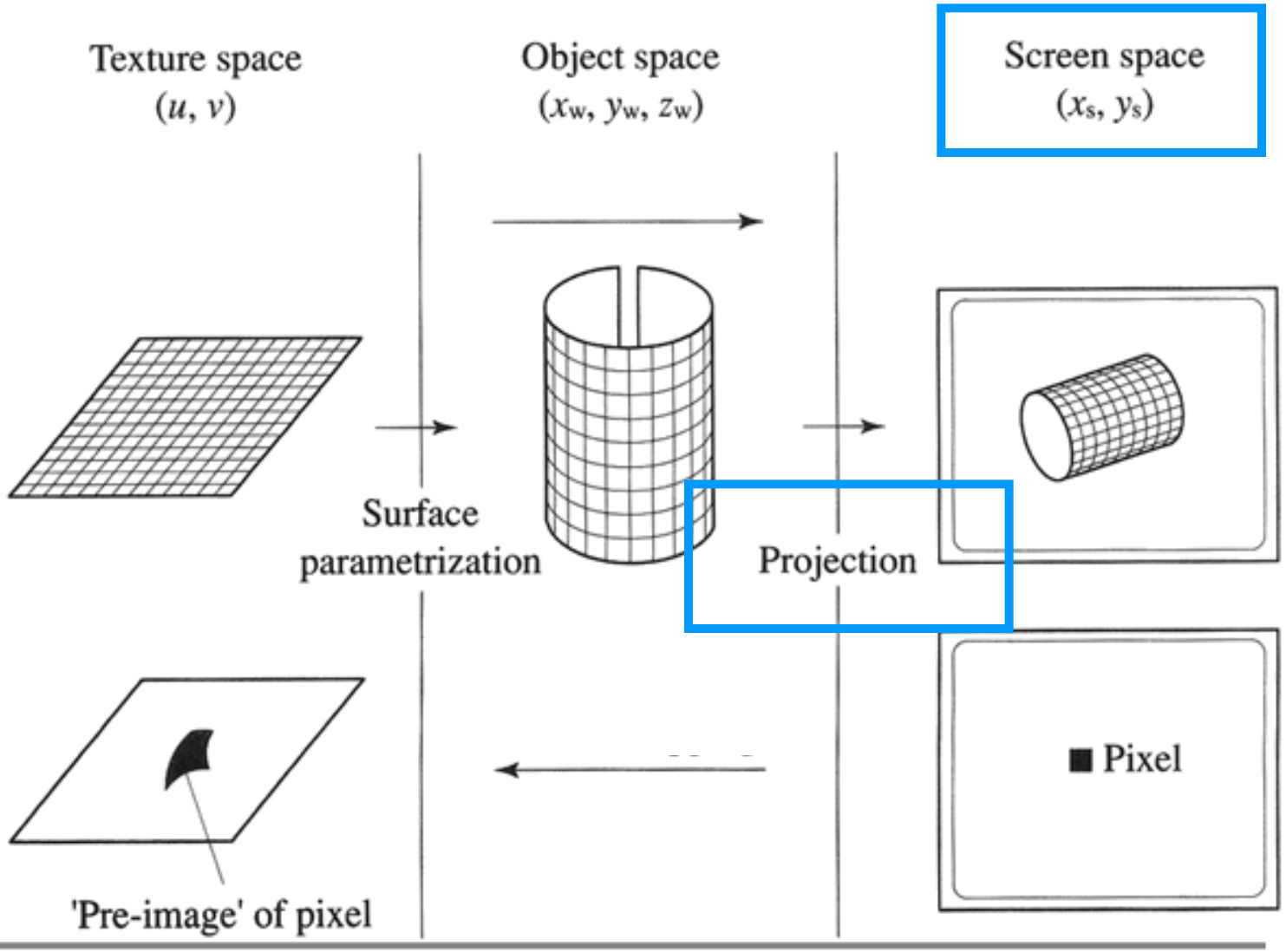


- Texturierung entspricht mathematisch der Hintereinanderausführung von inversem Mapping und Textur-Funktion:

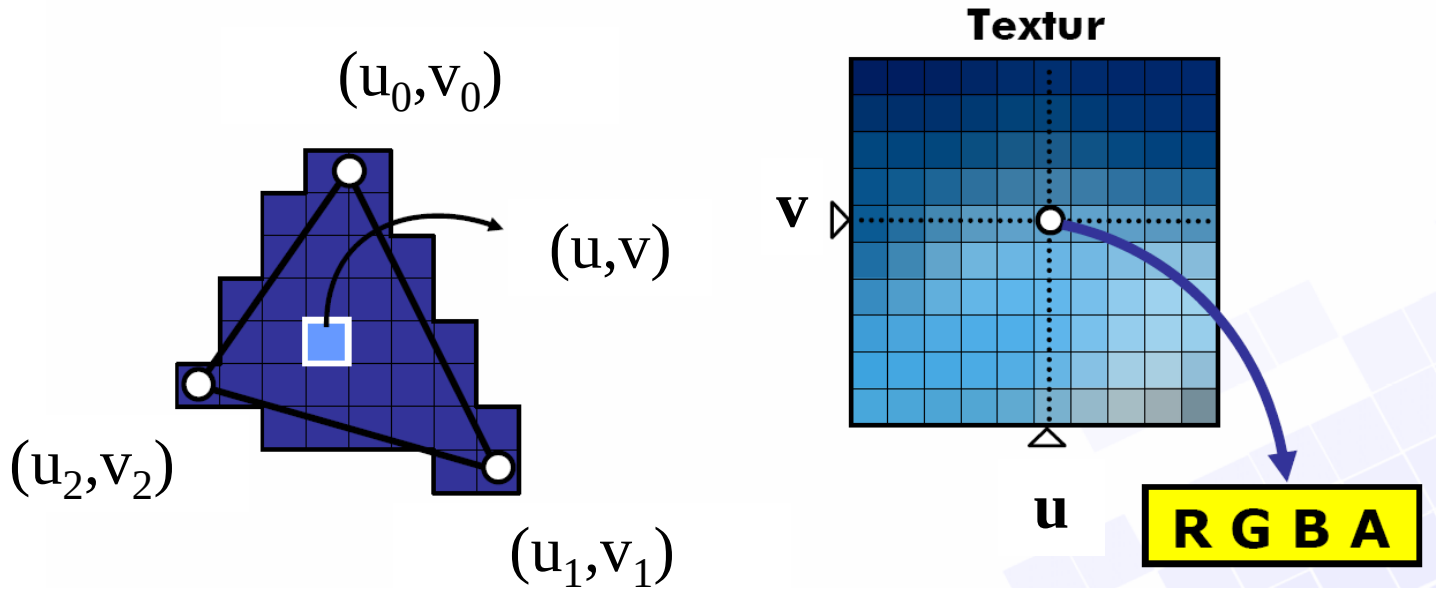
$$(r, g, b) = C_{tex}\left(F_{invmap}(x, y, z)\right)$$

7.1 Texture-Mapping

Prinzip



Interpolation der Texturkoordinaten



Für jedes Pixel:

Interpolation der Texturkoordinaten

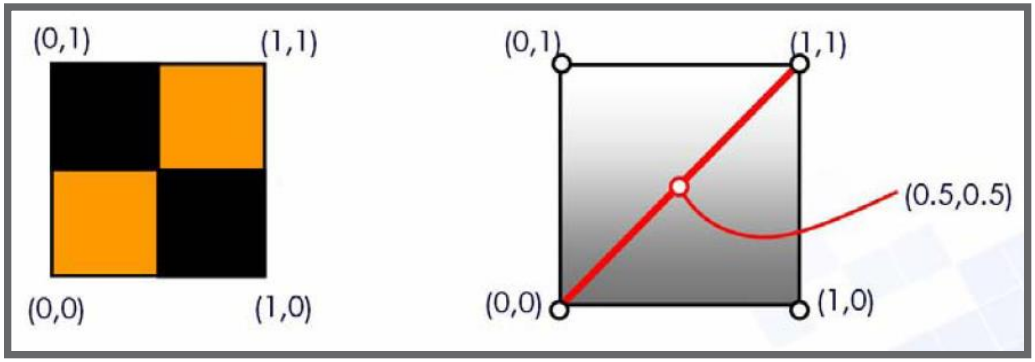
Texture-Lookup:

Interpolation der Texturwerte

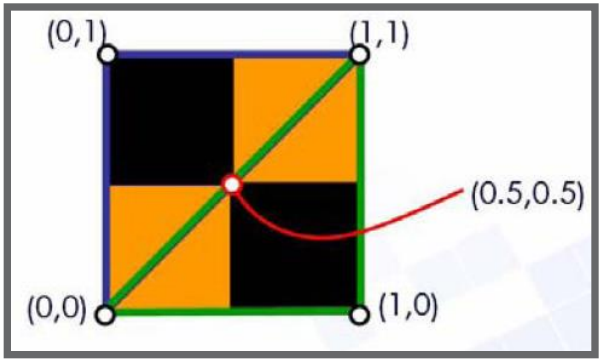
1. Nearest Neighbour
2. Bilineare Interpolation

7.1 Texture-Mapping

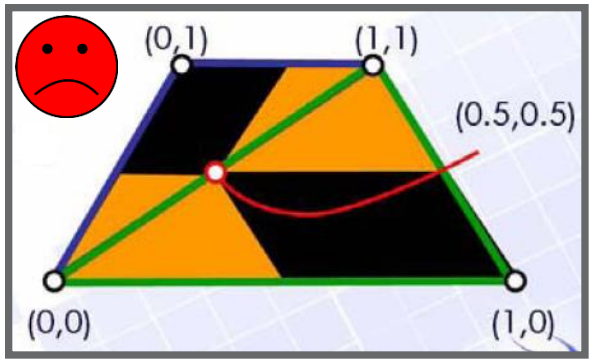
Perspektivische Texturierung

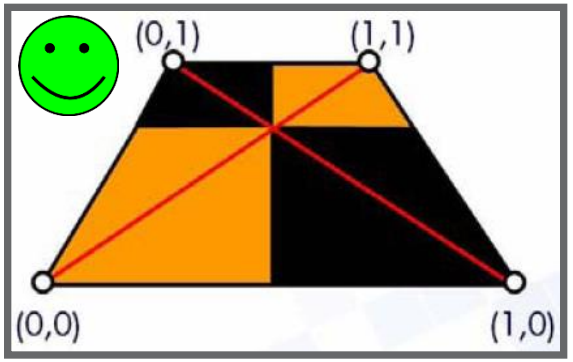


Textur Zu texturierende Fläche



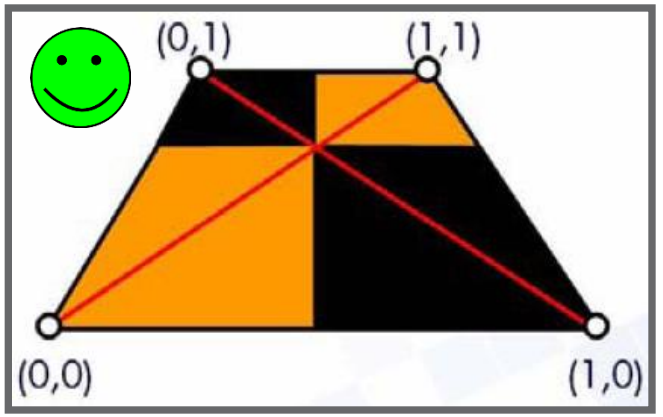
Texturierte Fläche





Lineare Interpolation der Texturkoordinaten liefert falsches Ergebnis.
Texturkoordinaten müssen ebenfalls perspektiv transformiert werden.

7.1 Texture-Mapping



Perspektivische Texturierung

- Perspektivische Transformation von Vertex $v = (x, y, z, 1.0)$ ergibt:

$$v_{trans} = (x, y, z, w) \quad \text{mit } w \neq 1.0$$

- Bildschirmkoordinaten: $\left(\frac{x}{w}, \frac{y}{w}, \frac{z}{w}\right)$
- Lineare Interpolation der Texturkoordinaten liefert falsches Ergebnis

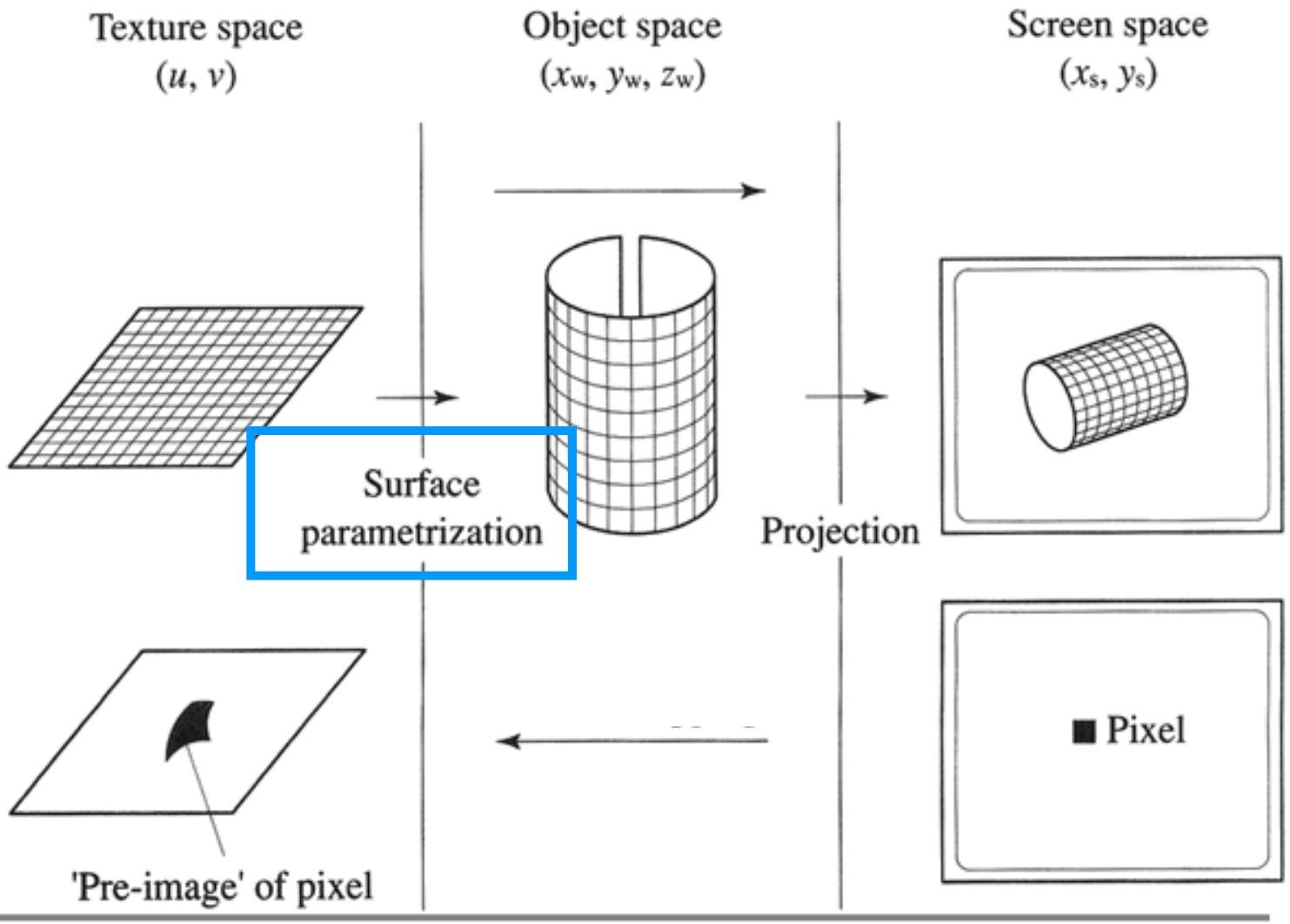
Lösung

1. Transformation der Texturkoordinaten $u_{trans} = \frac{u}{w}, \quad v_{trans} = \frac{v}{w}, \quad p = \frac{1}{w}$
2. Lineare Interpolation der Texturkoordinaten innerhalb des Polygons
3. Textur-Abfrage für jedes Pixel mit rücktransformierten Texturkoord.

$$(u, v)_{\text{interp}} = \left(\frac{u_{trans}}{p}, \frac{v_{trans}}{p}\right)$$

7.1 Texture-Mapping

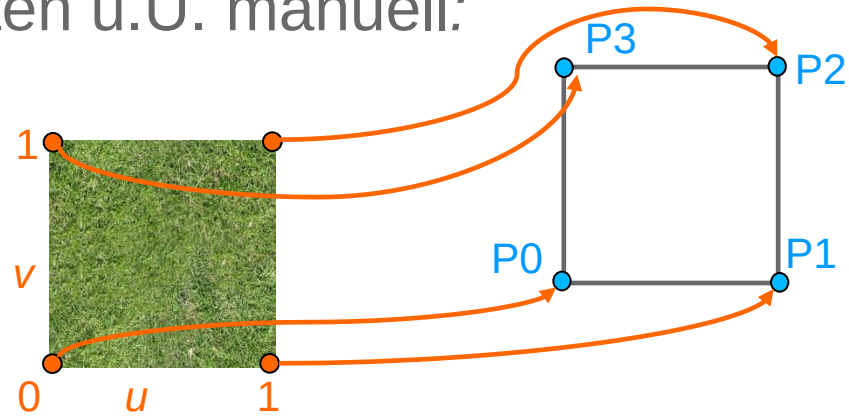
Prinzip



7.1 Texture-Mapping

Zuordnung von Polygonecken und Texturkoordinaten

- Bei einfachen Objekten u.U. manuell*:*

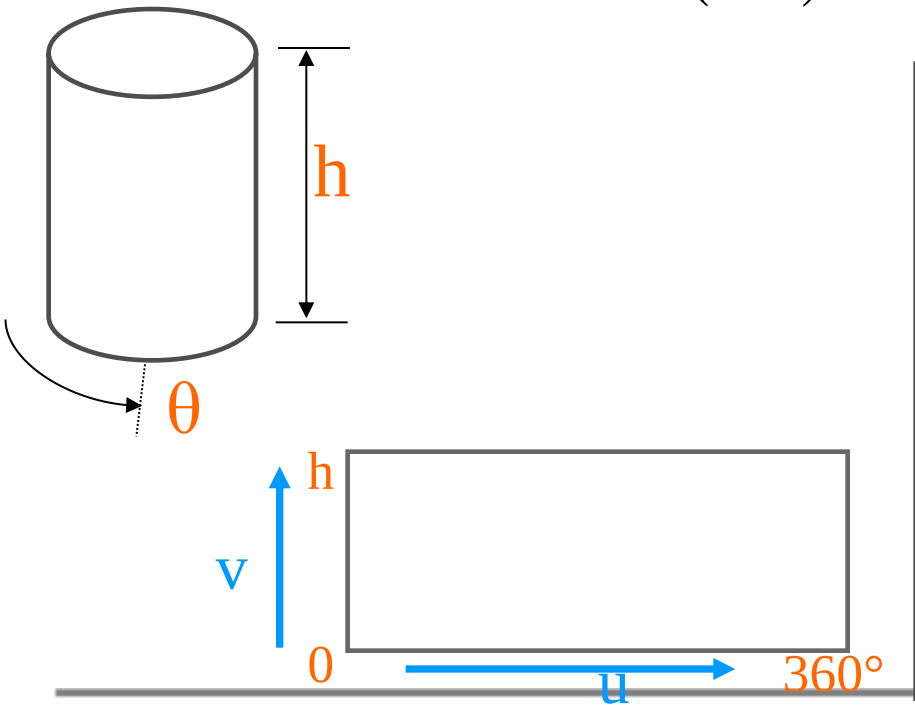


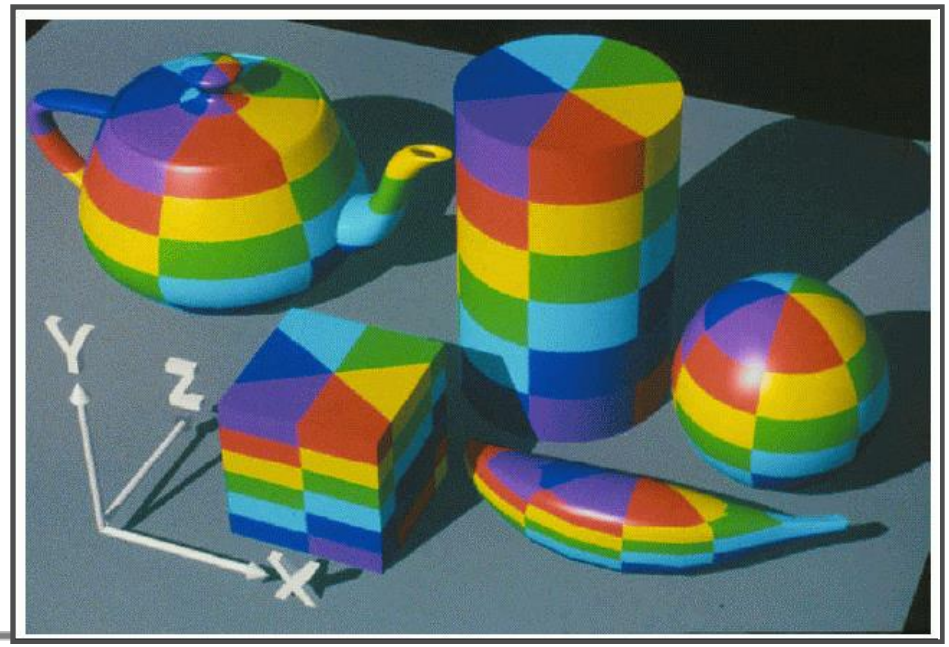
- Bei komplexeren Objekten: Two-Part Mapping
  1. S-Mapping
     Abbildung der Textur auf eine einfache, virtuelle Zwischenfläche (z.B. Quader, Zylinder, Kugeln)
  2. O-Mapping
     Übertragung von der umhüllenden Zwischenfläche auf das zu texturierende Objekt

7.1 Texture-Mapping

Zylinder-Mapping

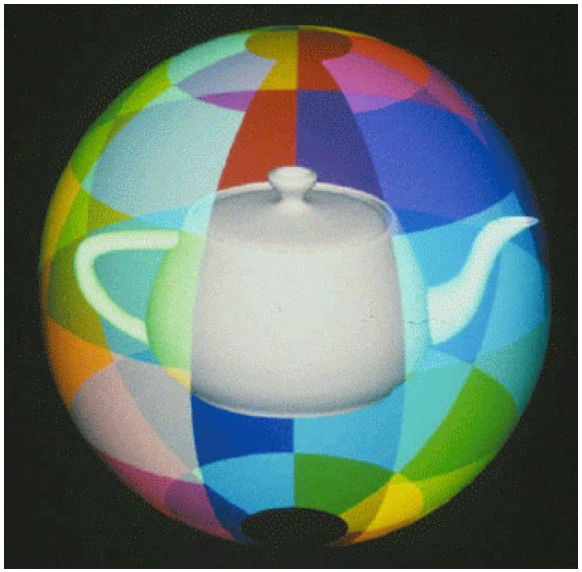
- Geeignet für rotationssymmetrische Objekte
- Diskontinuität an der Naht (parallel zur Achse)
- Parametrisierung $(u,v) \rightarrow (\theta, z)$

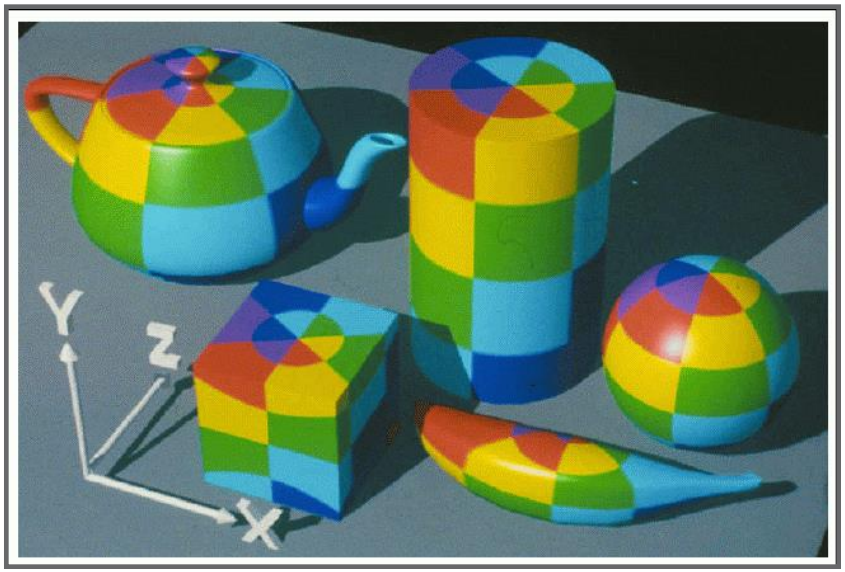




Kugel-Mapping

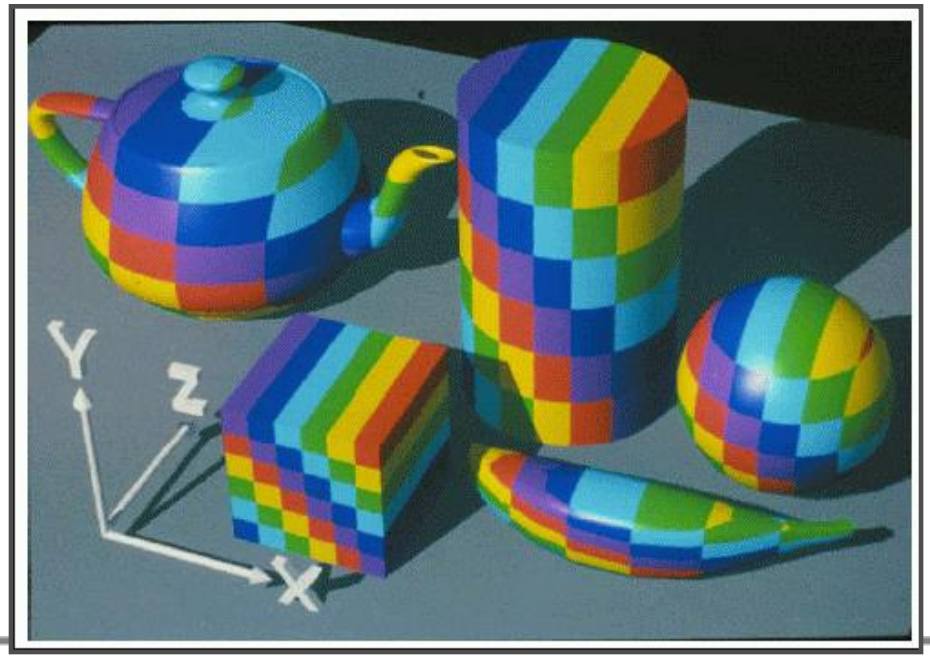
- Parametrisierung durch Kugelkoordinaten $(u,v) \rightarrow (\phi,\theta)$
- Verzerrung, v.a. an den Polen





Box-Mapping

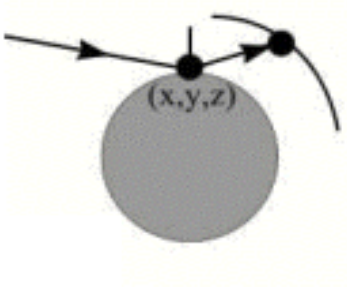
- Quader als umhüllende Fläche, meist achsenparallele Bounding-Box des Objekts
- Mögliche Parametrisierung
  - u-Achse: Längste Kante des Quaders
  - v-Achse: Zweitlängste Kante des Quaders



7.1 Texture-Mapping

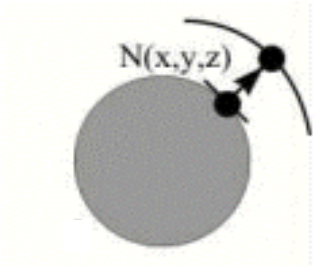
Techniken des O-Mapping

1. Reflexionsstrahl

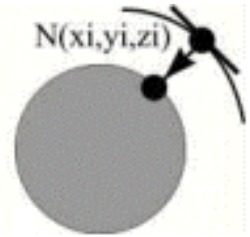


2. Objektzentrum

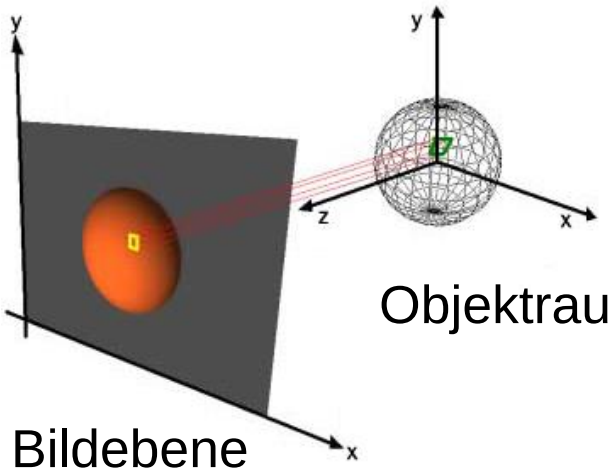
3. Normalenvektor



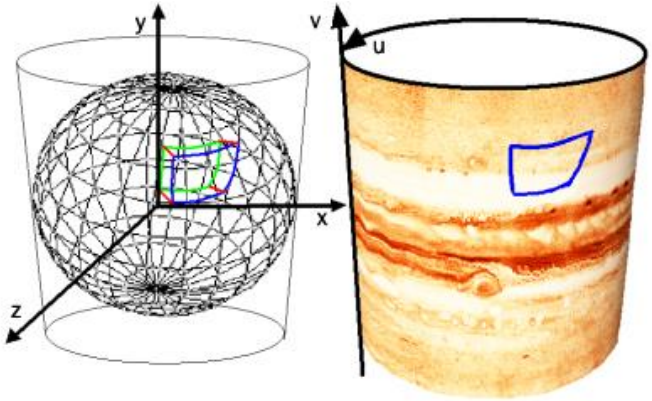
4. Hilfsobjektnormale



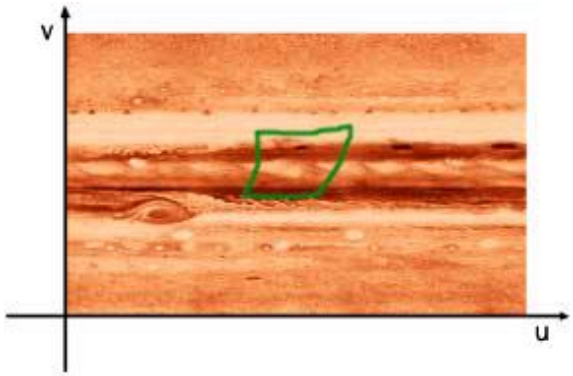
Inverse Abbildung mit Zwischenobjekt



Objektraum

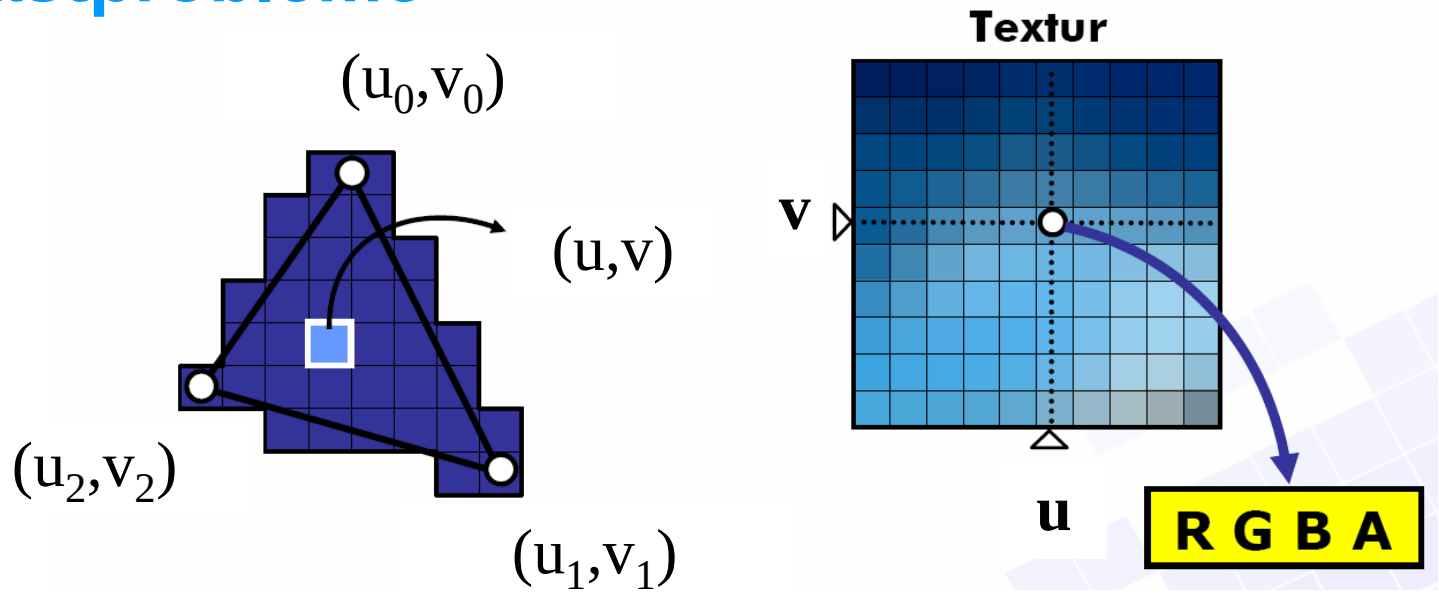


Zwischenobjekt



Texturebene

Abtastprobleme



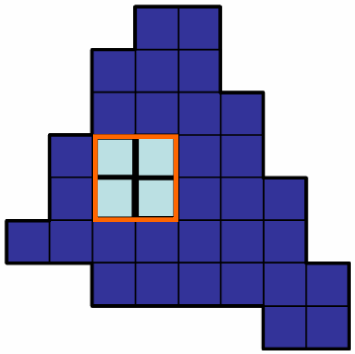
Pixel-Raster Textur-Raster

Welche Probleme können bei der Abbildung von Pixel- auf Textur-Raster auftreten?

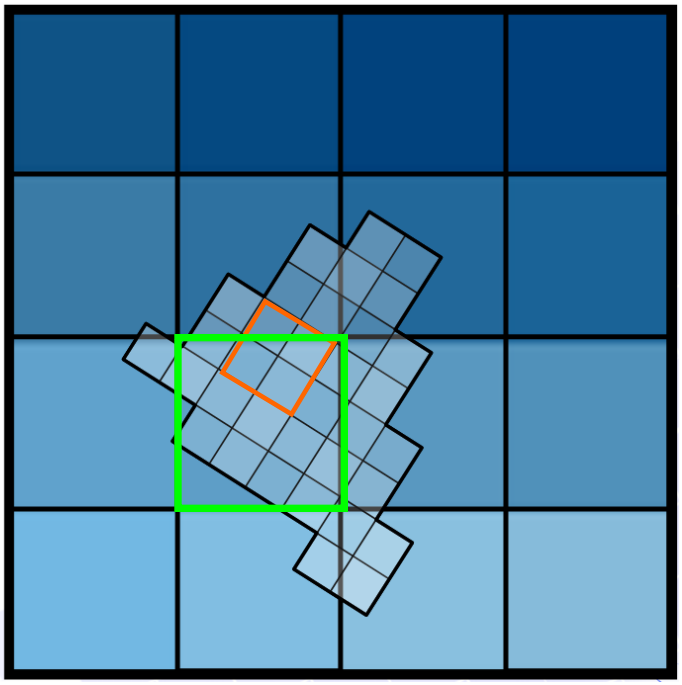
7.1 Texture-Mapping

Magnification / Oversampling

- Pixel-Raster ist feiner als Textur-Raster



Pixel-Raster



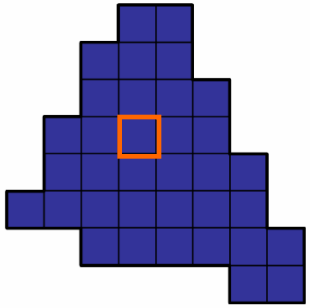
Textur-Raster

- Ein Texel wird auf mehrere Pixel abgebildet.
- Texturen erscheinen daher verschwommen.

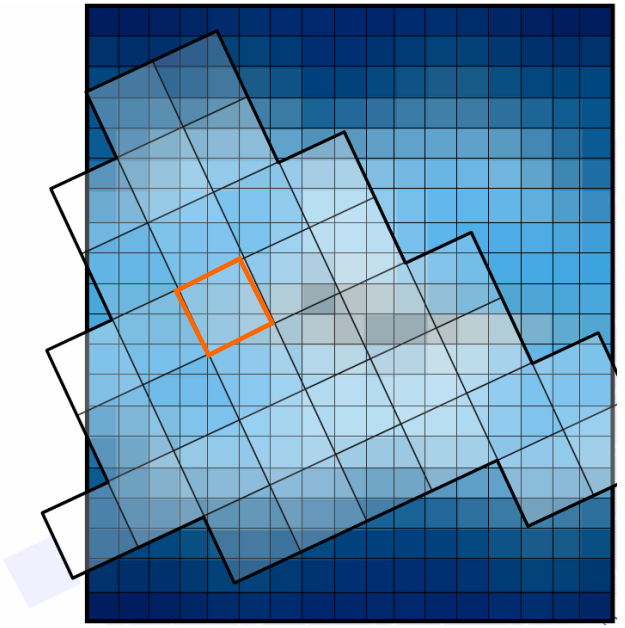
Abhilfe: Auflösung der Textur erhöhen.

Minification / Undersampling

- Pixel-Raster ist gröber als Textur-Raster



Pixel-Raster



Textur-Raster

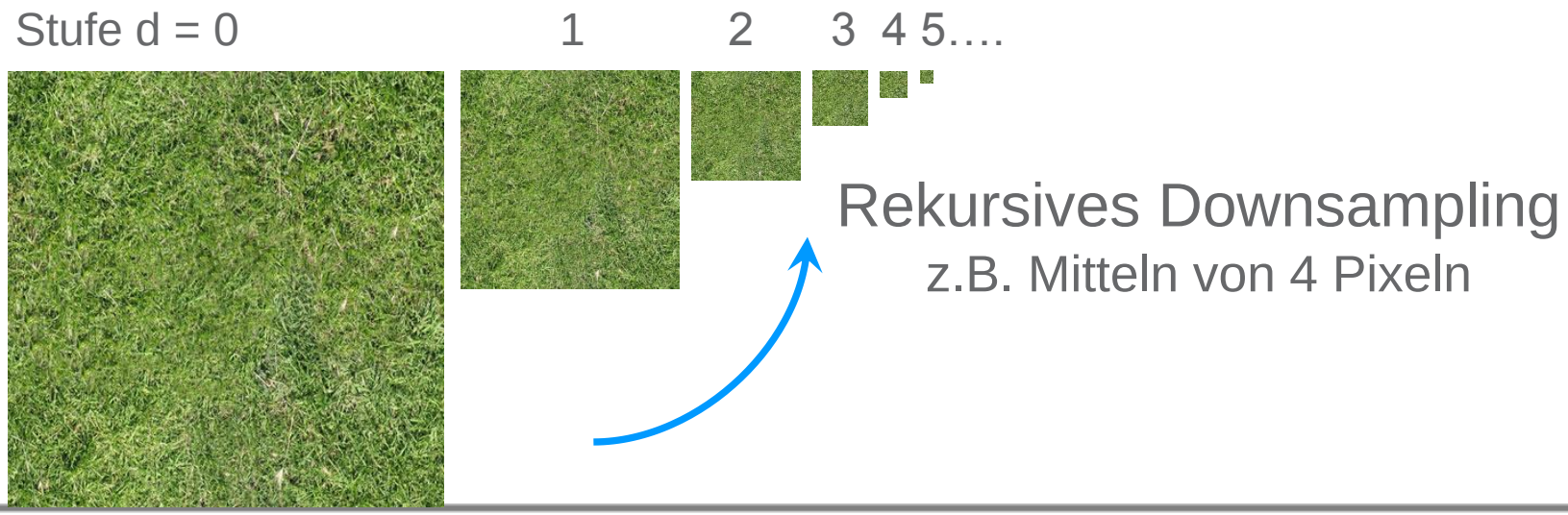
- Ein Pixel wird auf mehrere Texel abgebildet.
- Aliasing, da Abtastfrequenz zu gering

Abhilfe: Nach Fläche gewichteten Mittelwert der Texel bestimmen.
Exakte, aber zu aufwändige Lösung

7.1 Texture-Mapping

Mip-Mapping

- Auflösung der Textur sollte der im Bildraum entsprechen
- Problematisch, wenn Auflösung im Bildraum variiert
- Textur in verschiedenen Auflösungen bereithalten
- Quadratische Texturen mit Kantenlänge $n = 2^k$ (Zweierpotenz)

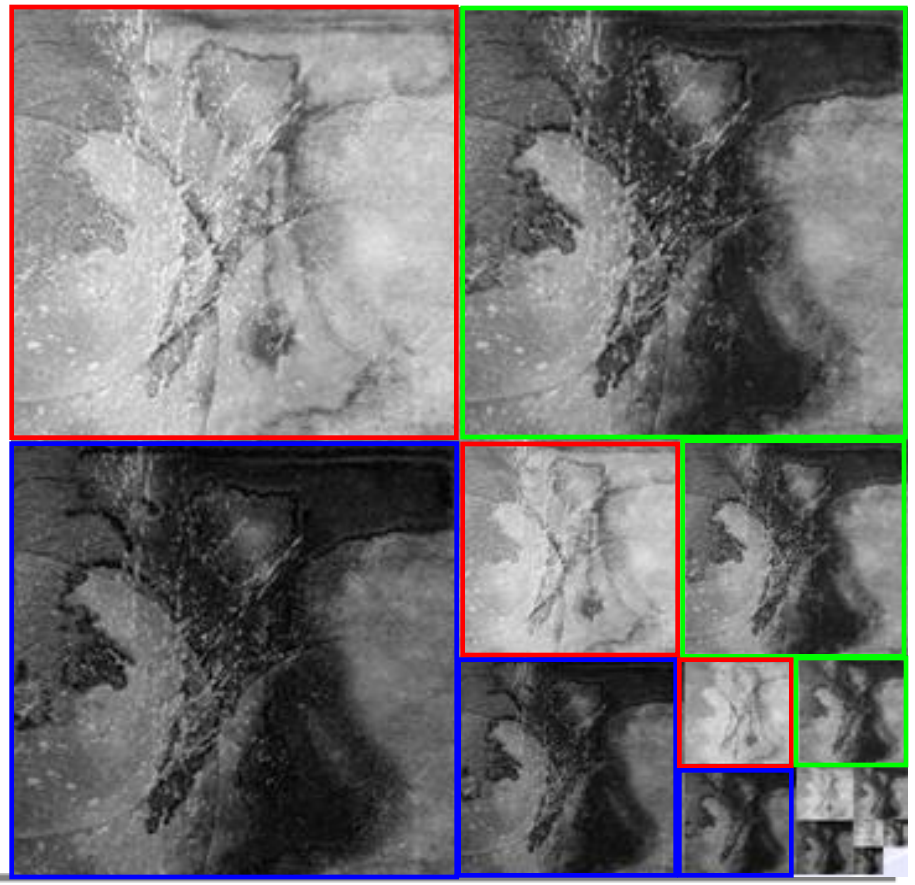


7.1 Texture-Mapping

Mip-Mapping

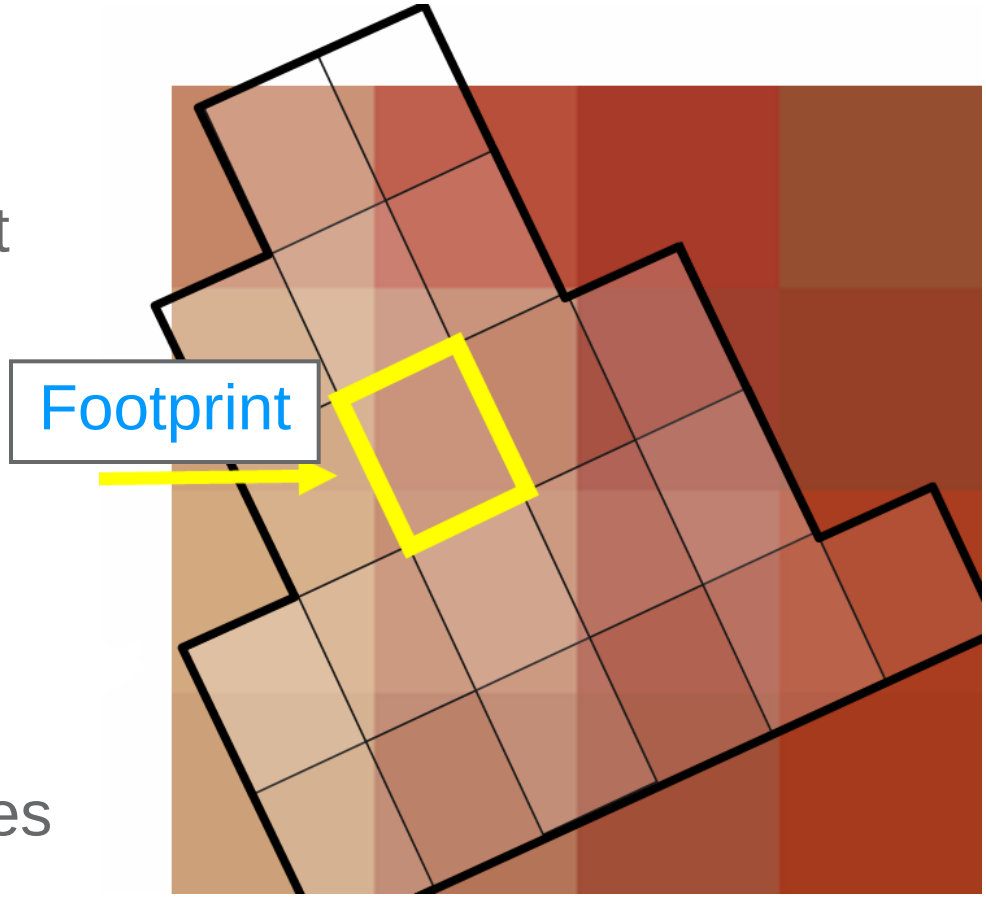
Benötigt nur etwa 1/3 mehr Speicher als einzelne Textur.

Beispiel: RGB-Textur



Mip-Mapping – Idee

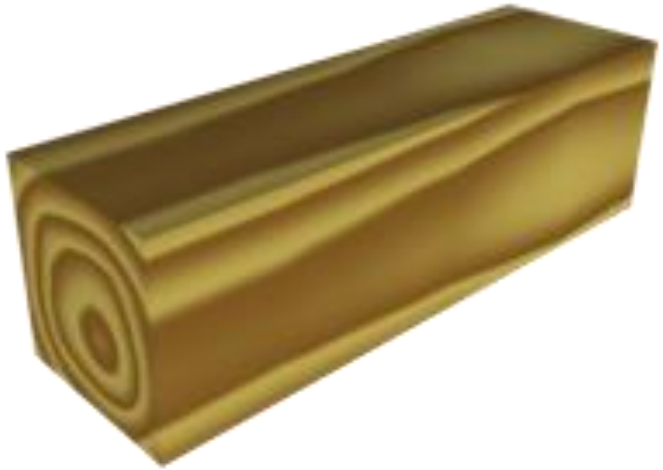
1. Bestimme Footprint des Pixels auf der Textur
2. Längste Kante bestimmt Mipmap-Stufe d
3. Interpoliere Texel aus Textur der Stufe d



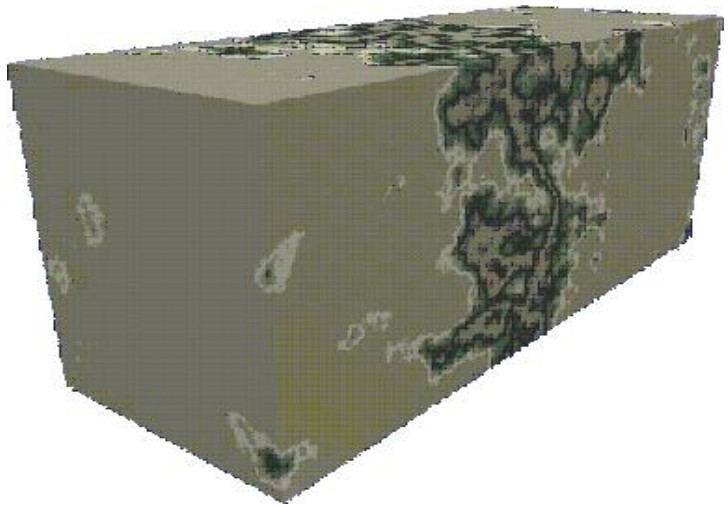
Ergebnis: Antialiasing
…da der verwendete Texel immer in etwa der Größe des Footprints entspricht

3D-Texturen / Festkörpertexturen

- Handhabung analog zu 2D-Texturen
- (Anschaulicher) Unterschied zu 2D-Texturen:
  Objekt wird aus Textur ausgeschnitten statt damit beklebt
- Prozedurale Ansätze erlauben wirklichkeitsgetreue 3D-Muster:



Holzmaserung



Perlin Marmor

7. Abbildungsverfahren

Motivation

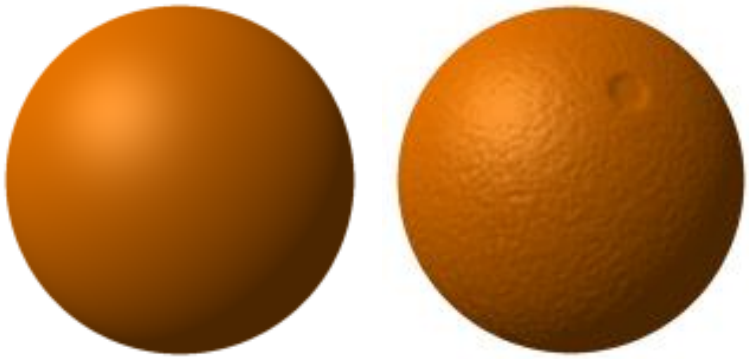
Reale Umgebung verfügt über ein großes Spektrum geometrischer Formen und physikalischer Materialien

- Maserungen und Muster von Oberflächen
- Strukturen unebener Flächen
- Hintergründe und Spiegelungen mit hohem Detailgrad

Exakte Nachbildung dieser Objekte ist meist zu aufwändig.

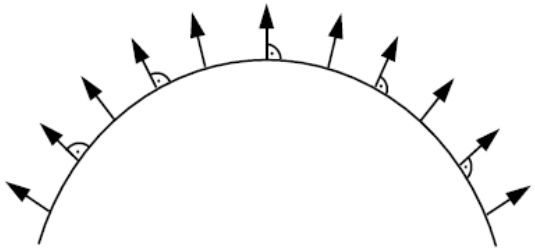
Bump-Mapping

- Ziel
  - Rauhere und plastischere Erscheinung eines Objekts
  - Ohne Veränderung der Geometrie
- Simulation von Oberflächenunebenheiten durch
  - Manipulation der Normalenvektoren
  - Struktureffekt nur durch Beleuchtung

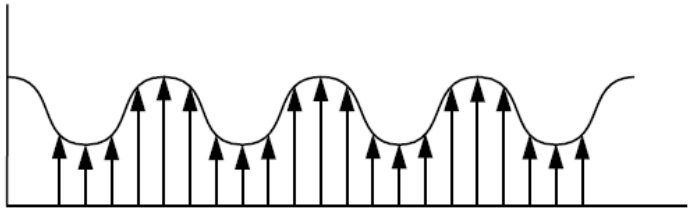


7.2 Bump-Mapping

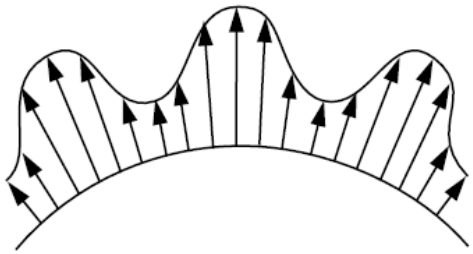
Exakte Modellierung sähe so aus….



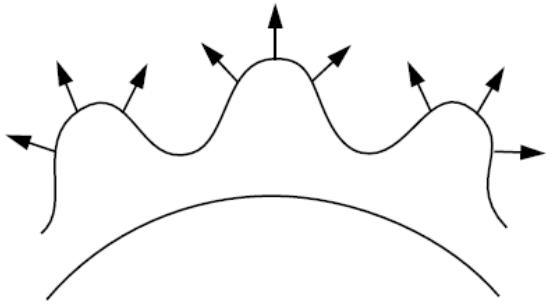
(a) Originalfläche **P**(u)
mit Normalen **N**(u)



(b) Bump Map h(u)



(c) Offsetfläche **P'**(u)



(d) Perturbierte Normalen **N'**(u)

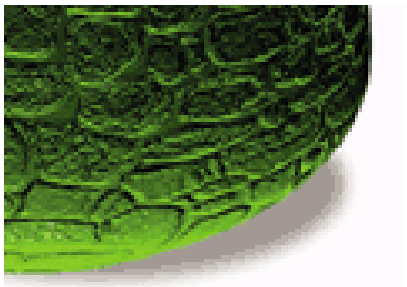
7.2 Bump-Mapping

Prinzip

- Für Beleuchtung nur Normale relevant, nicht Lage des Punktes auf der Fläche. Daher:
- Originalgeometrie mit veränderten Normalen kombinieren

Anmerkungen

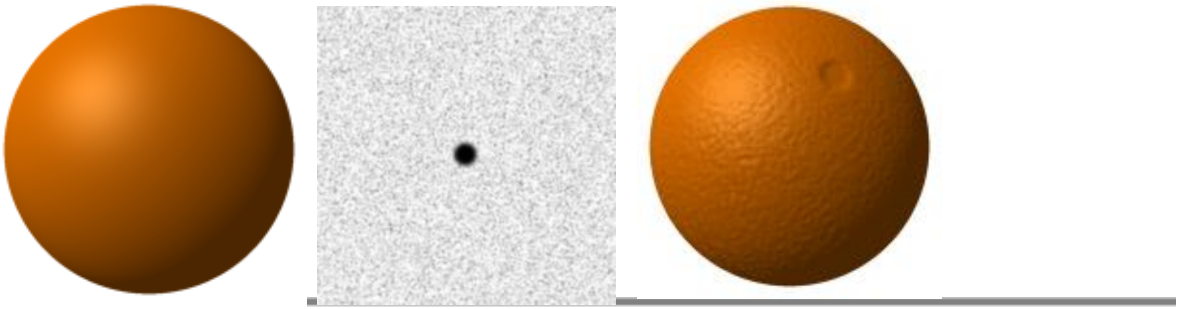
- Bump-Mapping nur mit Verfahren möglich, die Beleuchtung explizit in jedem Flächenpunkt auswerten (Phong-Shading, Raytracing, aber kein Gouraud)
- Schatten und Silhouetten der Objekte bleiben glatt.

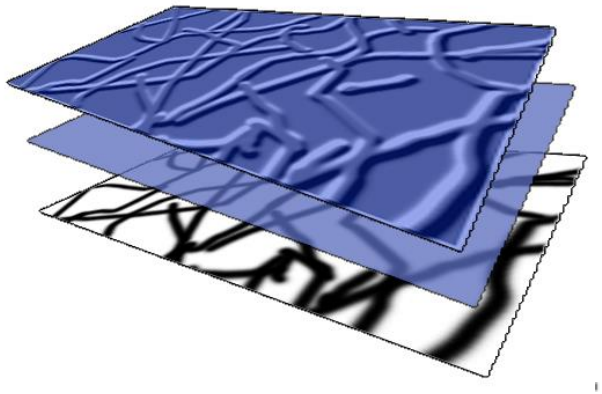


7.2 Bump-Mapping

Umsetzung

- Prozedurale Veränderung der Normalen
- Normalen als RGB-Textur gegeben (Normal Map)
- Höhenfeld als 2D-Skalarfeld (Grauwerttextur) gegeben
  - Addition des Höhenfeldes auf die Geometrie
  - Berechnung der Normalen über Richtungsableitungen dieser Offsetfläche $N' = P'_u \times P'_v$





Motivation

Reale Umgebung verfügt über ein großes Spektrum geometrischer Formen und physikalischer Materialien

- Maserungen und Muster von Oberflächen
- Strukturen unebener Flächen
- Hintergrund und Spiegelungen mit hohem Detailgrad

Exakte Nachbildung dieser Objekte ist meist zu aufwändig.

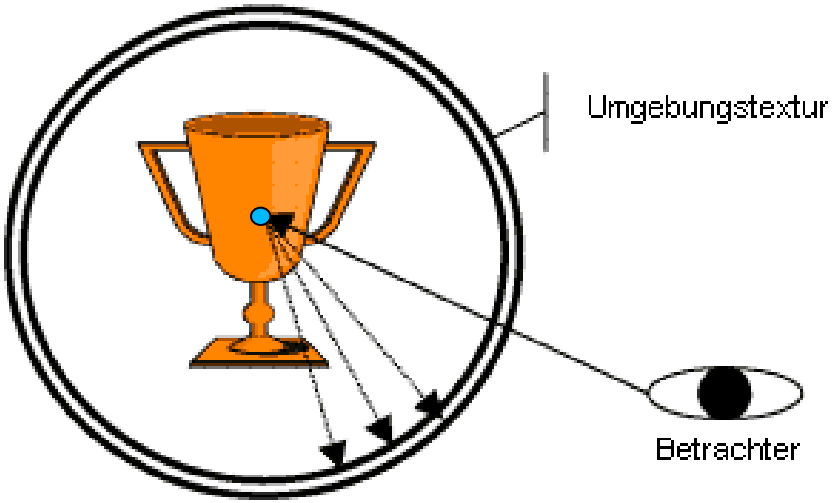
Environment-Mapping

- Modellierung von Interobjekt-Reflexionen
- Abbildung einer komplexen Umgebung auf ein spiegelndes Objekt mit Hilfe von Texturen
- Günstige Alternative zum Raytracing
- Keine Verdeckungsrechnung

7.3 Environment-Mapping

Vorgehen

- Zwischenobjekt (Würfel, Kugel) umhüllt Objekt
- Umgebungstextur auf Innenseite des Zwischenobjekts aufgetragen
- Texturkoordinaten abhängig vom Reflexionsvektor

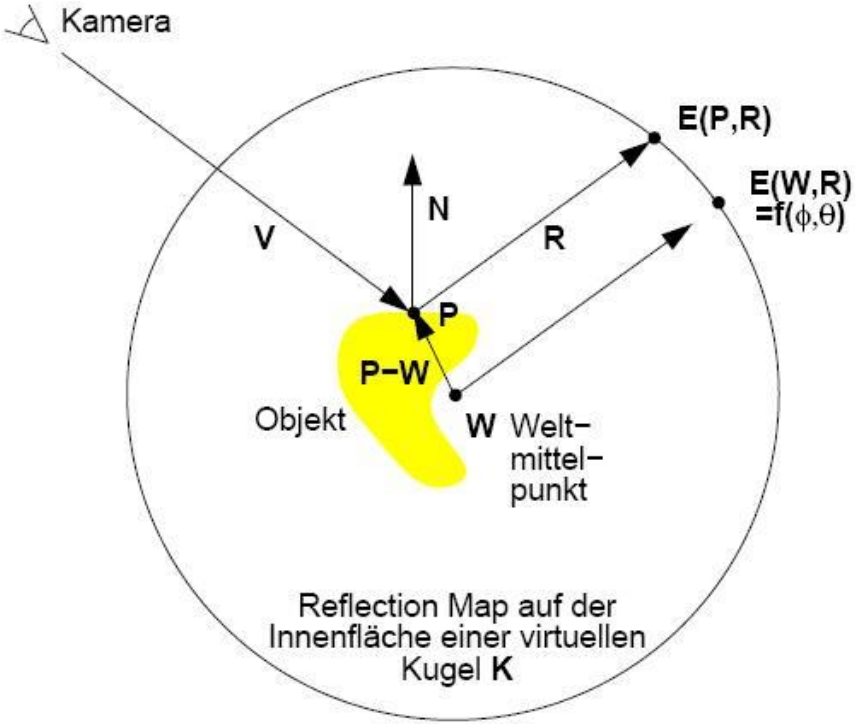


7.3 Environment-Mapping

Verwendung einer Kugel

Probleme

- Reflexionsberechnung nur exakt, wenn sich Objekt P im Mittelpunkt befindet
- Schwierige Erstellung der Umgebungstextur
- Schlechte Parametrisierung
- Aliasing-Probleme



Sphere Mapping

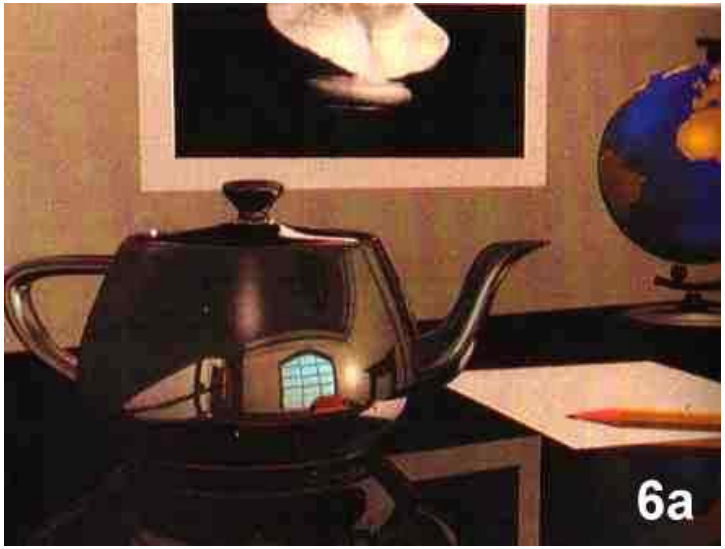
- Unterstützt von OpenGL
- Abbildung einer umhüllenden Kugel auf eine 2D-Textur

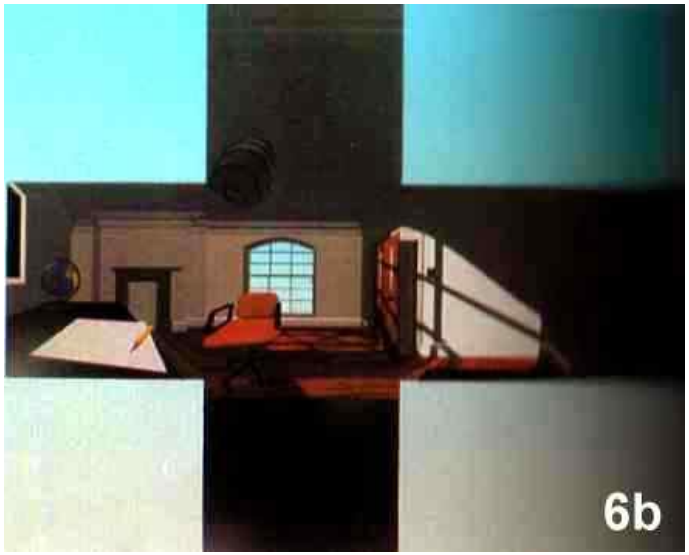
Sphere Mapping - Probleme

- Ebenfalls schwierige Erstellung
  - Rendering
  - Foto einer spiegelnden Kugel
  - Umrechnung aus planaren Texturen (Würfel)
- Abtastrate über Textur nicht regelmäßig
- Interpolation der Texturkoordinaten führt zu Fehlern, v.a. am Rande der Sphere

Verwendung eines Würfels

Beispiel

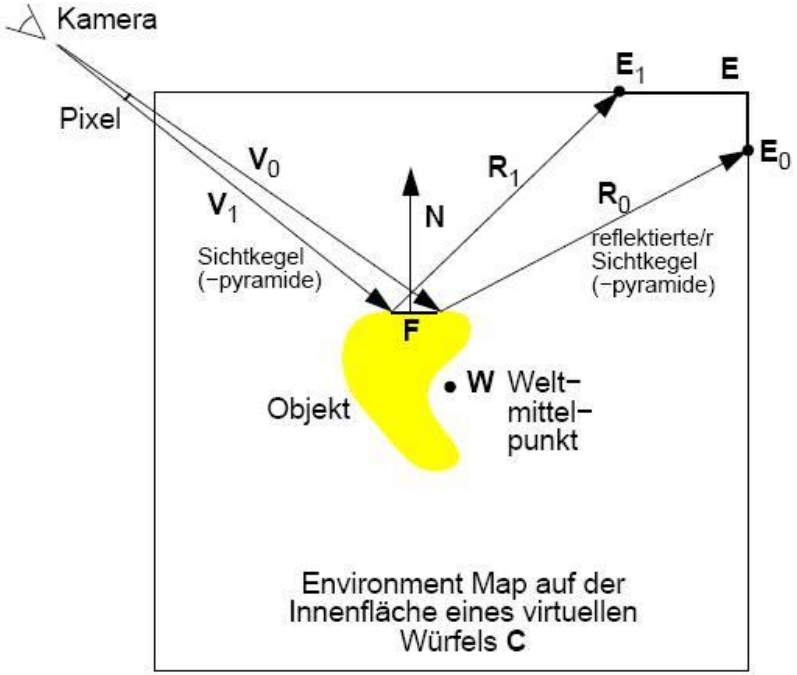




7.3 Environment-Mapping

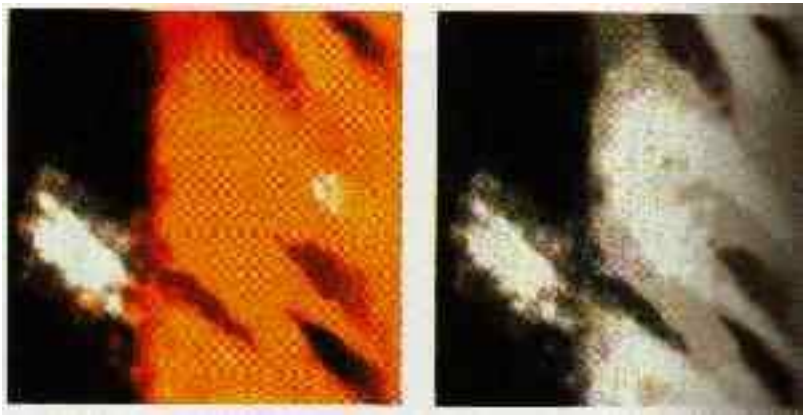
Verwendung eines Würfels

- Einfachere Erstellung der Textur
- Anti-Aliasing durch Sichtpyramide
- Schlechte Parametrisierung, doch besser als bei Kugel



Chrome-Mapping

- Abbildung eines willkürlichen Musters (chrome map) auf eine reflektierende Oberfläche
- Künstliche Unschärfe
- Textur fest im Raum (daher geeignet für Animationen)

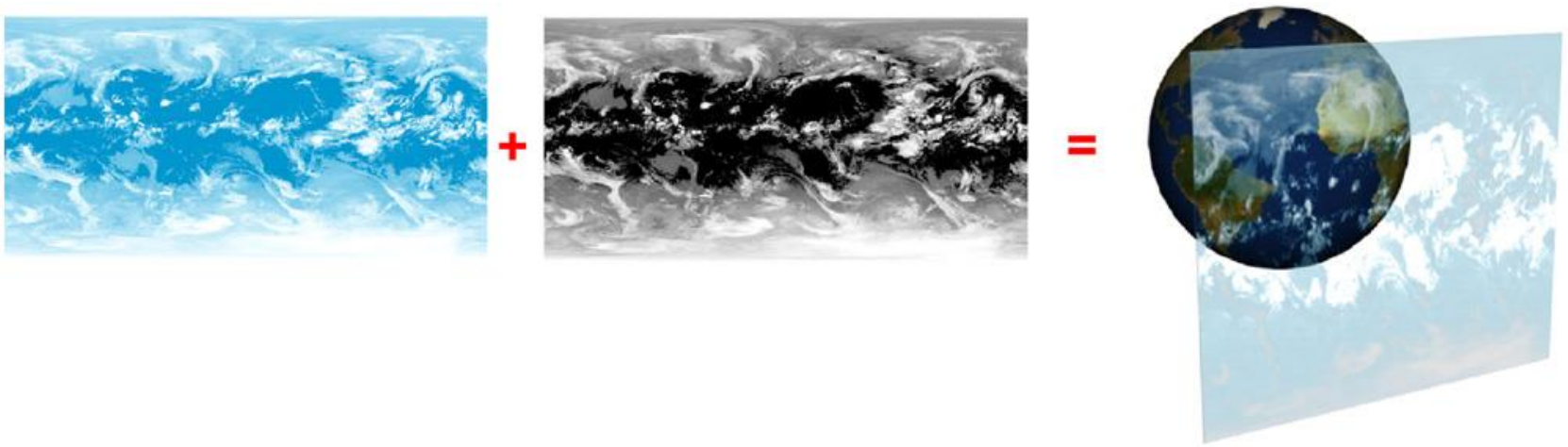


Displacement Mapping

- Ziel: Plastischere Erscheinung der Oberfläche
- Unterschied zum Bump-Mapping:
  - Verschieben der Oberflächenpunkte gemäß Höhenfeld entlang der Normalen
  - Meist Kombination mit Textur
- Vorteil: Schatten & Silhouette erscheinen nicht mehr glatt.

Opacity-Mapping / Transparency-Mapping

- Ziel: Transparenz von Objekten lokal kontrollieren
- Opacity-Map
  - Grauwerte entsprechen Alpha-Wert
  - Von schwarz (opaque) bis weiß (transparent)



Abbildungsverfahren

- Erzielte Effekte
  - Veränderte Maserungen und Muster einer glatten Fläche (Texture-Mapping, Chrome-Mapping)
  - Modellierung rauher Oberflächen (Bump- oder Displacement-Mapping)
  - Abbildung komplexer Umgebung auf Objekte (Environment-Mapping)

- Kombination der Verfahren möglich
- Echtzeit durch Hardwareunterstützung kein Problem

Quellen

- Computergraphik, Universität Leipzig
  (Prof. D. Bartz)
- Graphische Datenverarbeitung I, Universität Tübingen
  (Prof. W. Straßer)
- Graphische Datenverarbeitung I, TU Darmstadt
  (Prof. M. Alexa)
- Computergraphik, Uni Siegen (Prof. Klomfaß)