Computergraphik Grundlagen

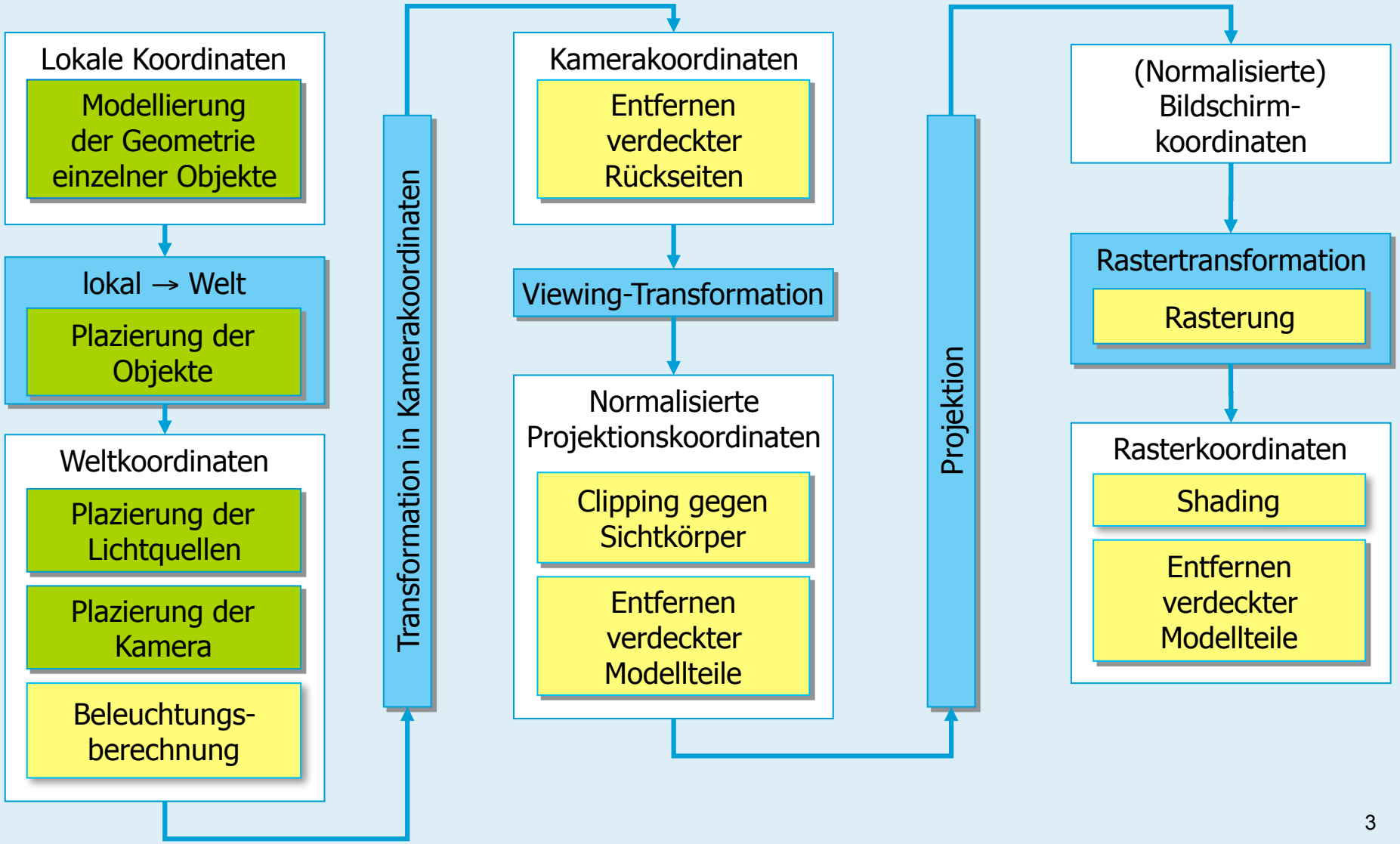
IX. Texturen und Schatten

Prof. Stefan Schlechtweg
Hochschule Anhalt
Fachbereich Informatik

Inhalt – Lernziele

1. Texture Mapping
    1. Texture Pipeline
    2. Environment Mapping
    3. Bump Mapping
    4. Displacement Mapping
    5. Zusammenfassung
2. Schattenberechnung
    1. Kern- und Halbschatten
    2. Position der Lichtquelle
    3. Generelle Vorgehensweise
    4. Algorithmen
3. Zusammenfassung

- Texture-Mapping als Möglichkeit des Einbringens von Details in Computergraphiken verstehen
- Verschiedene Möglichkeiten des Texture Mapping kennen
- Generellen Ablauf der Texturing-Pipeline erklären können
- Schattenalgorithmen kennen
- Vor- und Nachteile der Mapping-Verfahren gegenüber detaillierter Modellierung verstehen


Lokale Koordinaten
Modellierung der Geometrie einzelner Objekte
lokal → Welt
Plazierung der Objekte
Weltkoordinaten
Plazierung der Lichtquellen
Plazierung der Kamera
Beleuchtungs-berechnung
Transformation in Kamerakoordinaten
Kamerakoordinaten
Entfernen verdeckter Rückseiten
Viewing-Transformation
Normalisierte Projektionskoordinaten
Clipping gegen Sichtkörper
Entfernen verdeckter Modellteile
Projektion
(Normalisierte) Bildschirm-koordinaten
Rastertransformation
Rasterung
Rasterkoordinaten
Shading
Entfernen verdeckter Modellteile

1. Texture Mapping

- Jim Blinn: „All it takes is for the rendered image to look right“
- Polygonale Modelle approximieren eine Oberfläche nur.
- Sehr detaillierte Oberflächen erfordern sehr viele, kleine Polygone.
- nicht akzeptabel in Bezug auf Rendering-Zeit und Speicherplatz
- daher andere Möglichkeiten der Darstellung von Details notwendig
- → Texture Mapping
  - Hinzufügen von Detailinformationen auf Polygone
  - Detailinformationen sind typischerweise 2D-Bilder
  - Erhöhung des Photorealismus
  - aber: rechen- und speicheraufwendig
- eingeführt von Ed Catmull 1974, dann verfeinert von Jim Blinn 1976

1. Texture Mapping

- Das Modell (links) nach dem Shading (Mitte)
- Details der Hautstruktur, Farbnuancen und weitere Details werden durch Texturen hinzugefügt (rechts)

1. Texture Mapping

- Folgende Eigenschaften eines geometrischen Modells können mit Hilfe von Texturen verändert werden:
  - Farbe:
    Modulation des diffusen Reflektionskoeffizienten mit der Farbe aus der Textur
  - Spekulare „Farbe“:
    Erzeugen des Anscheins, daß das Objekt die Umgebung spiegelt (environment mapping)
  - Normalenvektor:
    Erzeugen eines 3D-Relief-Eindrucks auf der Objektoberfläche (bump mapping)
  - Objektpunkte:
    Verschieben der Oberflächenpunkte in Normalenrichtung auf Basis der Textur (displacement mapping)
  - Transparenz:
    Steuerung der „Durchsichtigkeit“ einer Oberfläche auf Basis der Textur (opacity mapping)

1. Texture Mapping

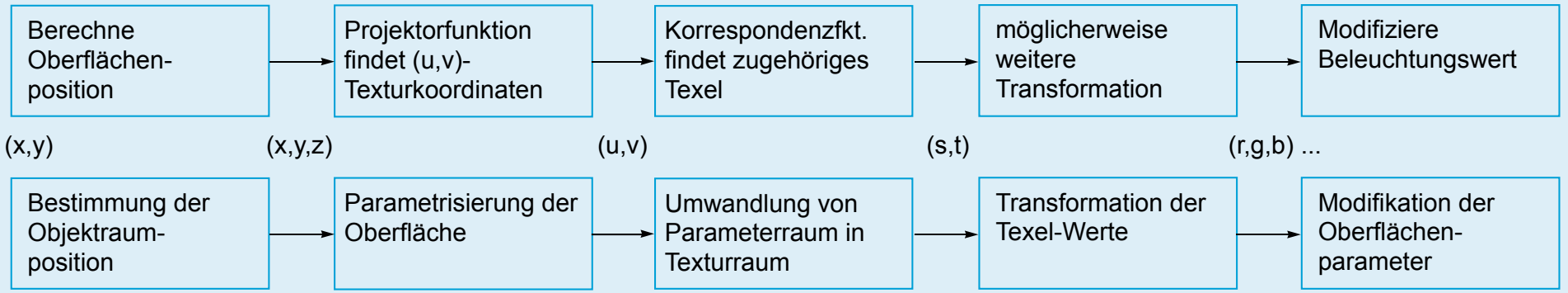
- Textur ist typischerweise ein 2D-Bild
  - Bildelemente (Pixel) hier Texel genannt
  - Wert des Texels beeinflußt Aussehen der Oberfläche
- Mapping der Textur auf die Oberfläche bestimmt, wie die Textur auf der Oberfläche liegt
- Mapping einer Textur auf ein Dreieck ist einfach.
- Mapping einer Textur auf eine beliebige 3D-Oberfläche ist kompliziert.
- Rendering unter Benutzung von Texture-Mapping:
  - Finde die zum Pixel gehörende sichtbare Oberfläche
  - Finde den Punkt auf dieser Oberfläche, der auf den Pixel abgebildet wird
  - Finde den Punkt in der Textur (Texel), der zu dem Oberflächenpunkt gehört
  - Benutze die Farbe des Texels, um das Shading des Pixels zu ändern

1. Texture Mapping

- allgemeine Definition:
  - *Zweidimensionale Texturen* oder kurz 2D-Texturen sind Funktionen, die Punkte der *(u, v)*-Ebene auf *(r, g, b)*-Werte abbilden: $(r, g, b) = c_{tex}(u, v)$.
  - Das *Mapping* (engl.: Abbildung) beschreibt, wie eine 2D-Textur bzw. ein Ausschnitt aus einer 2D-Textur auf eine Fläche aufgebracht wird.
  - Beim Rendering muß jedoch das inverse Mapping-Problem gelöst werden, d. h. den bekannten *(x, y, z)*-Koordinaten des Flächenpunktes *P* müssen *(u, v)*-Koordinaten zugeordnet werden: $(u, v) = F_{invmap}(x, y, z)$.

1. Texture Mapping
1.1. Texture Pipeline

- Was passiert an einem Punkt, z. B. dem Eckpunkt eines Polygons?
  1. Berechnung der Farbe aus Licht, Material, Betrachterstandpunkt
  2. Textur modifiziert die berechneten Farbwerte abhängig von der Position auf der Oberfläche
- → Allgemeine Texturing-Pipeline:



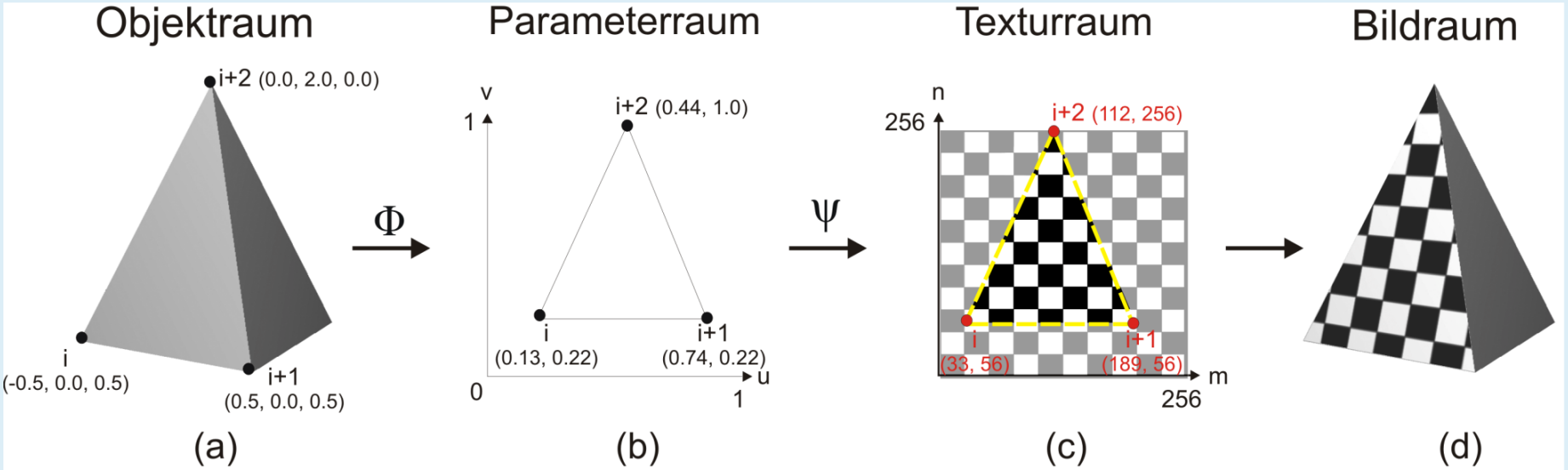
1. Texture Mapping
1.1. Texture Pipeline

- Bestimmung der Objektraumposition
  - Für jeden Eckpunkt i eines Polygons werden die 3D-Koordinaten (x_i, y_i, z_i) im Objektraum, in dem die Oberfläche definiert ist, bestimmt
- Parametrisierung der Oberfläche
  - Raumkoordinaten des Eckpunktes i werden in 2D-Koordinaten im Parameterraum (u_i, v_i) überführt: Projektorfunktion Φ
  - Mapping: $\Phi(x_i, y_i, z_i) \rightarrow (u_i, v_i)$
  - Parameterkoordinaten (u_i, v_i) liegen in [0,1]

1. Texture Mapping
1.1. Texture Pipeline

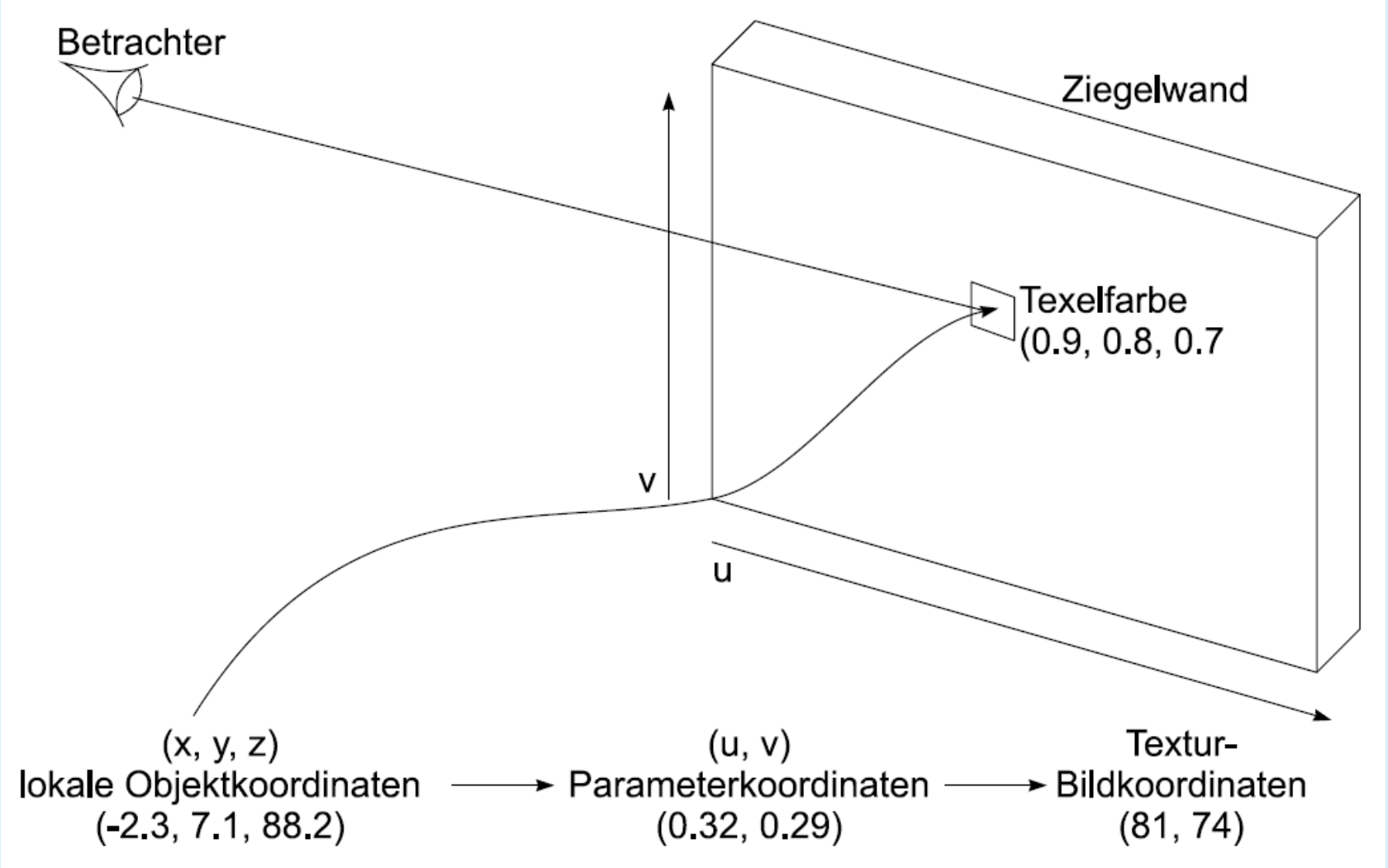
- Umwandlung vom Parameterraum in den Texturraum
  - Parameterkoordinaten (u_i, v_i) geben relative Position eines Eckpunktes *i* in der Textur an
  - Korrespondenzfunktion ψ wandelt diese in Texturkoordinaten um und bestimmt somit die ganzzahligen Index-Koordinaten zum Zugriff auf den Texel
- Transformation der Texel-Werte
  - Filterung und ähnliche Aänderungen der Texel-Werte sind möglich
- Modifikation der Oberflächenparameter
  - Texel-Wert modifiziert oder ersetzt bestimmte Parameter des Eckpunktes *i* (Farbe, Normale, …)

1. Texture Mapping
1.1. Texture Pipeline



1. Texture Mapping

1.1. Texture Pipeline

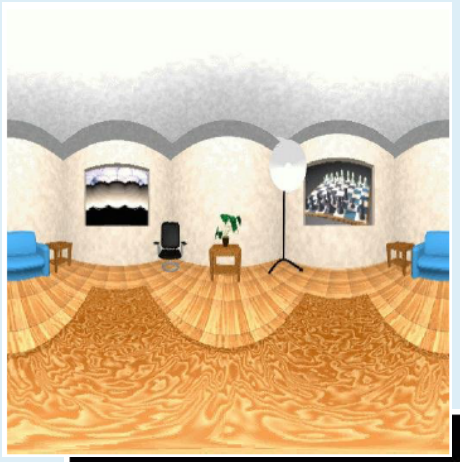


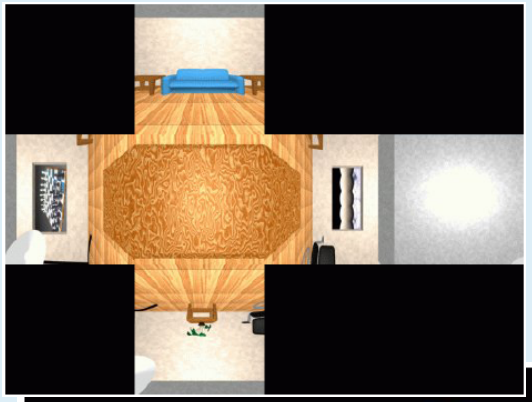
1. Texture Mapping
1.2. Environment Mapping

- Spiegelungen sind ein globales Phänomen (Objekt-Objekt-Interaktion)
- eine Möglichkeit, Reflektionen (approximativ) mit Hilfe von Textur-Hardware (und ohne Raytracing, d.h. schnell) zu berechnen
- Annahme: Umgebung besteht aus Objekten und Lichtquellen, die weit vom zu rendernden Objekt entfernt sind, d. h. Objekt ist klein im Vergleich zum Abstand zu umgebenden Objekten
- einfallende Beleuchtung kann für Objekt vorberechnet und in Texturgespeichert werden
- Vorgehen: Mapping eines Bildes der Umgebung auf eine (große) Kugel, deren Zentrum der Objektmittelpunkt ist
- andere Möglichkeit: Textur auf einem Würfel und Mapping der Würfeloberfläche auf das Objekt

1. Texture Mapping

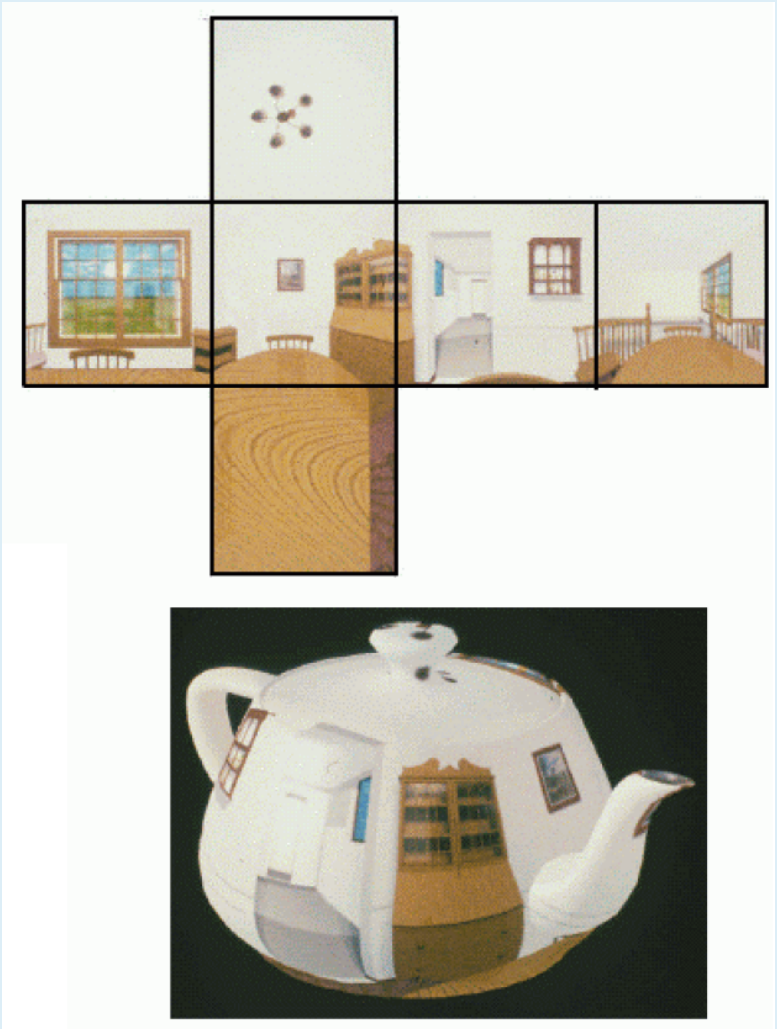
1.2. Environment Mapping

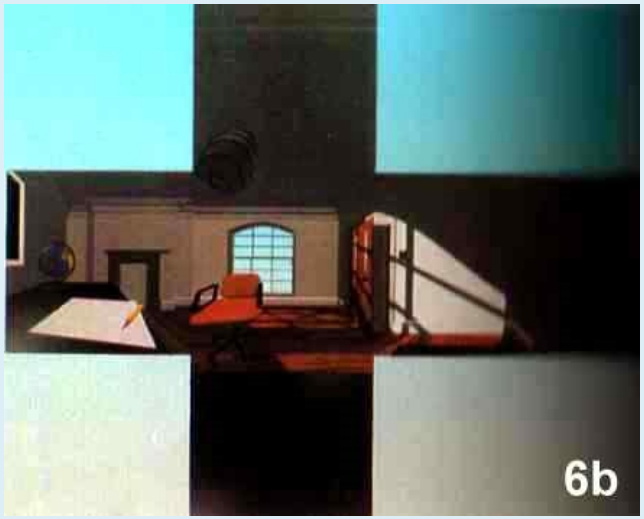




1. Texture Mapping

1.2. Environment Mapping





6b

6a

1. Texture Mapping
1.3. Bump Mapping

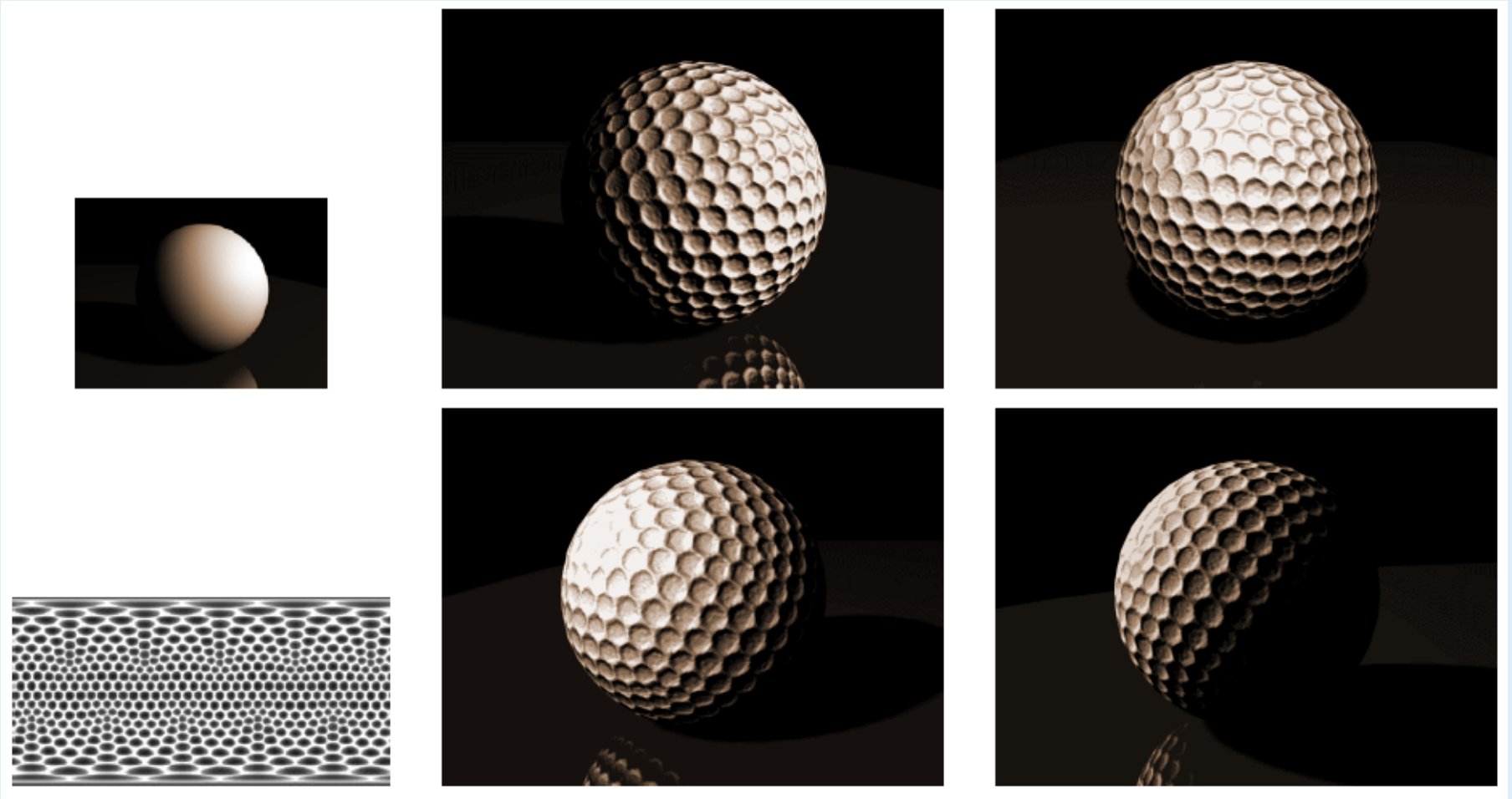
- Texture Mapping verändert das Shading (Farbberechnung)
- Oberfläche bleibt geometrisch glatt („smooth“)
- Texturierung rauher Oberflächen problematisch: Lichtverhältnisse in der Textur (im Bild) sind anders als die am Modell beim Rendering
- Bump Mapping verändert die Normalen der Oberfläche und damit die Geometrie, bevor die Beleuchtungsberechnung ausgeführt wird
- entwickelt von Jim Blinn, 1978

1. Texture Mapping
1.3. Bump Mapping

1. offset vector bump map oder offset map
   - Speicherung zweier Werte pro Texel b_u und b_v
   - Variation der Normale entlang der u- bzw. v-Achse um diesen Betrag
   - also: b_u und b_v als Skalierungsfaktor für zwei Vektoren, die senkrecht zur Normale in der Tangentialebene an dem entsprechenden Punkt senkrecht zueinander stehen und dann Addition dieser skalierten Vektoren zur ursprünglichen Normale
2. hightfield bump map
   - Bump map wird als Höhenfeld aufgefaßt
   - Skalierung der Vektoren in u- bzw. v-Richtung wird aus der Differenz benachbarter Texel abgeleitet

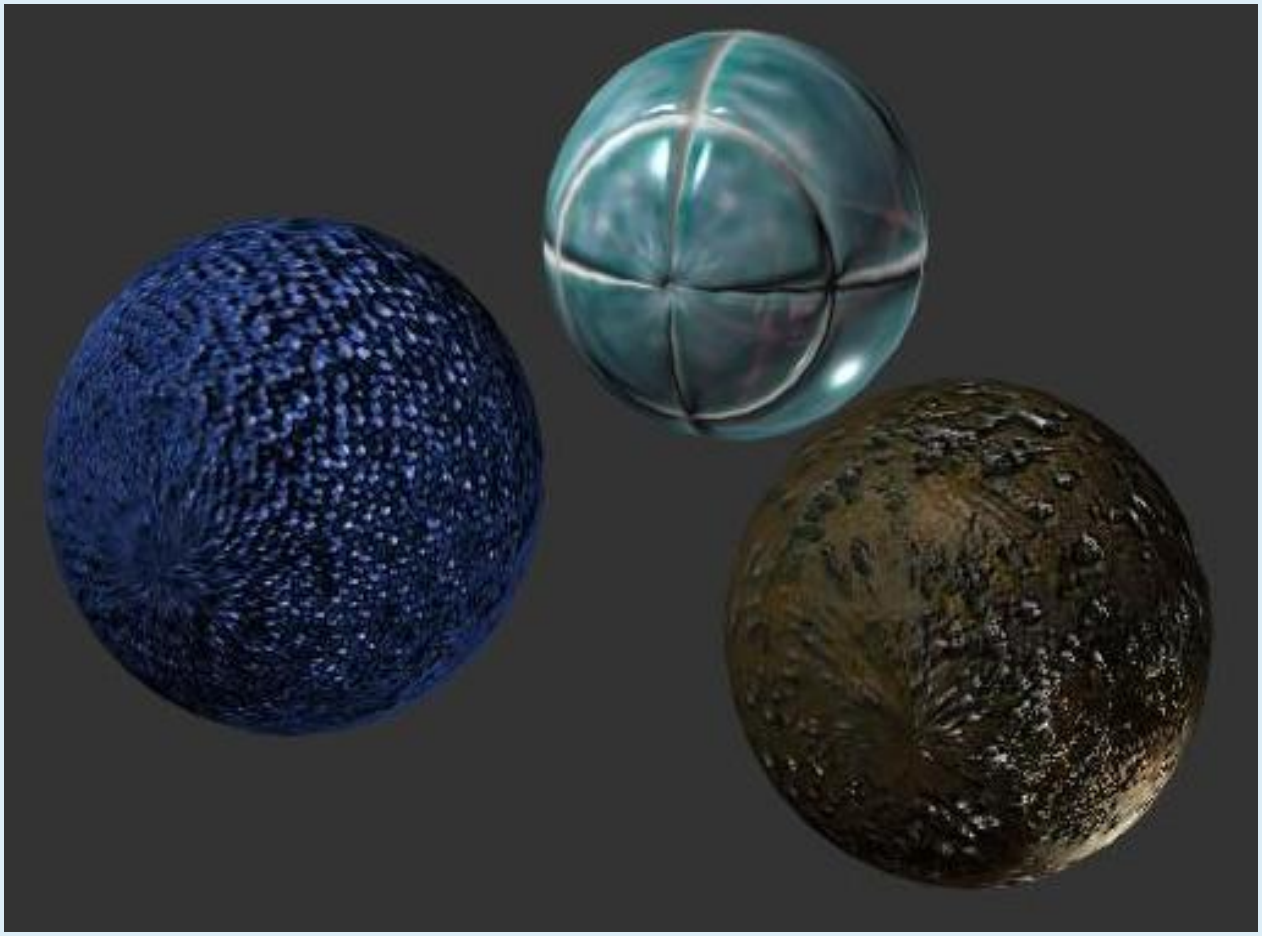
1. Texture Mapping

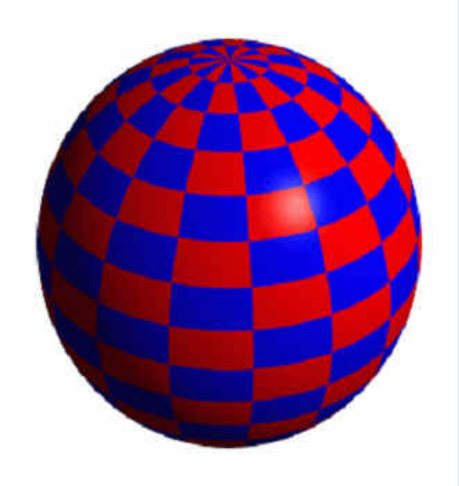
1.3. Bump Mapping

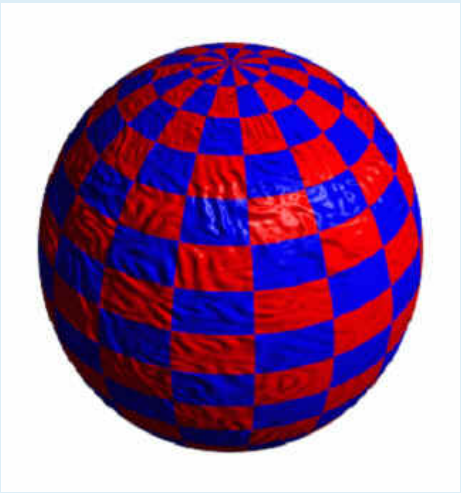


1. Texture Mapping

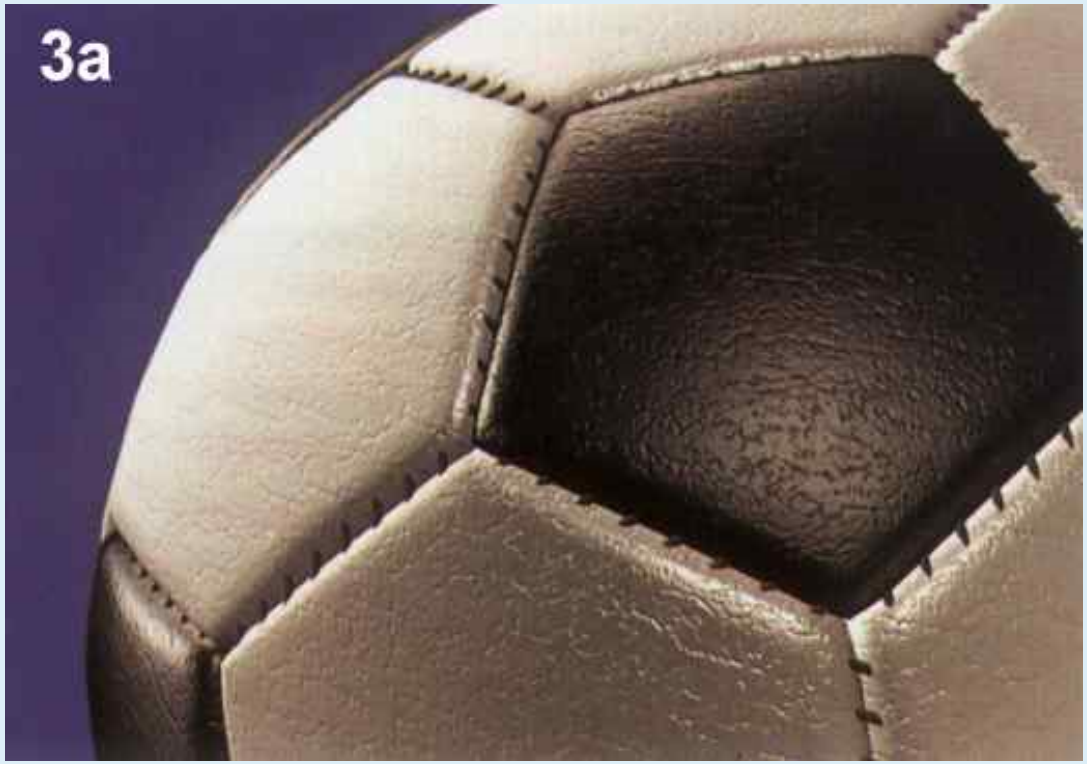
1.3. Bump Mapping

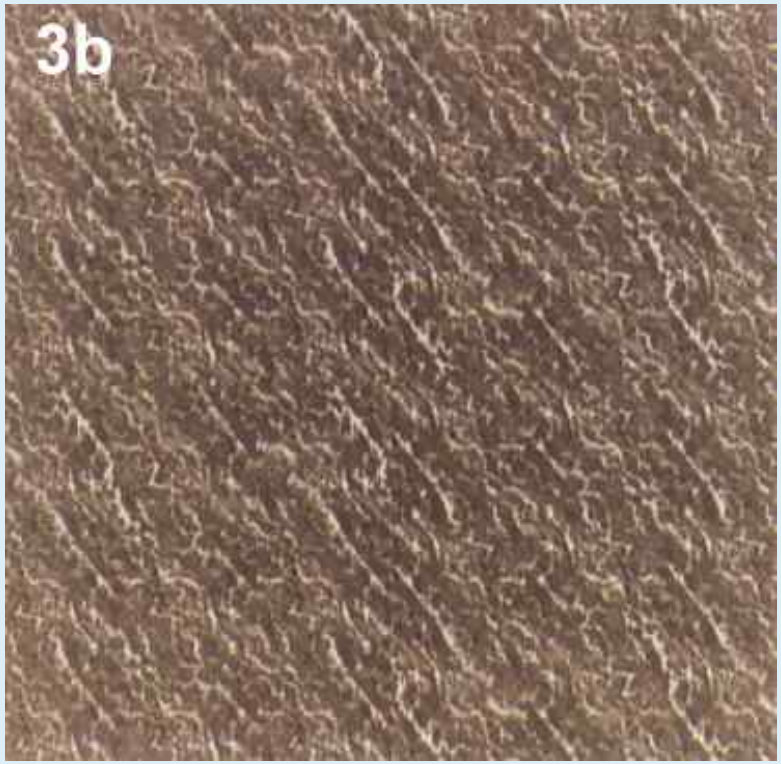






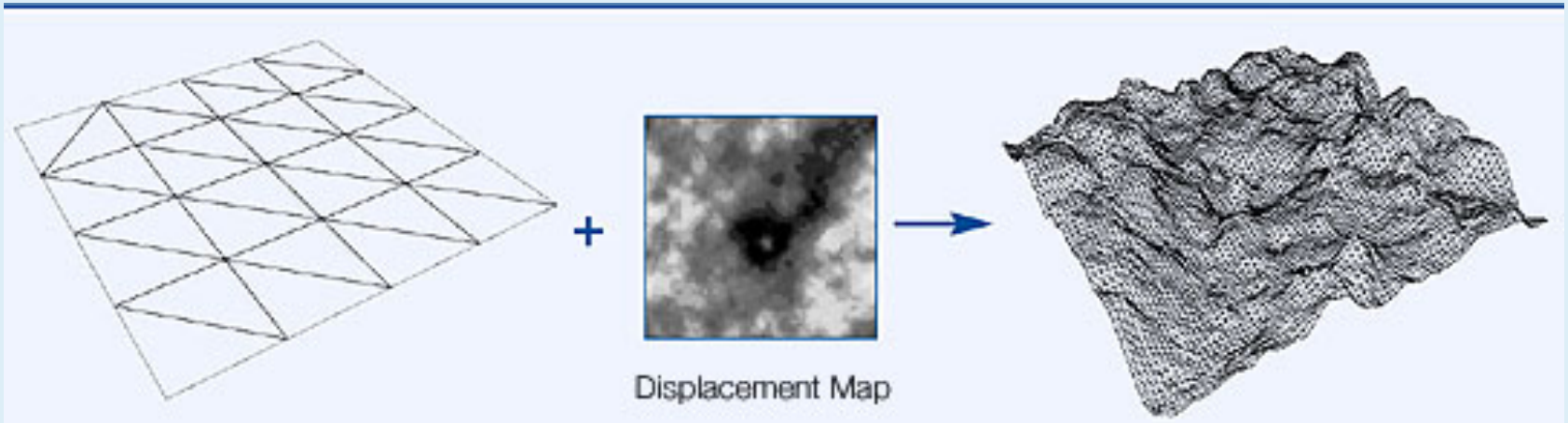
1. Texture Mapping
1.3. Bump Mapping





1. Texture Mapping
1.4. Displacement Mapping

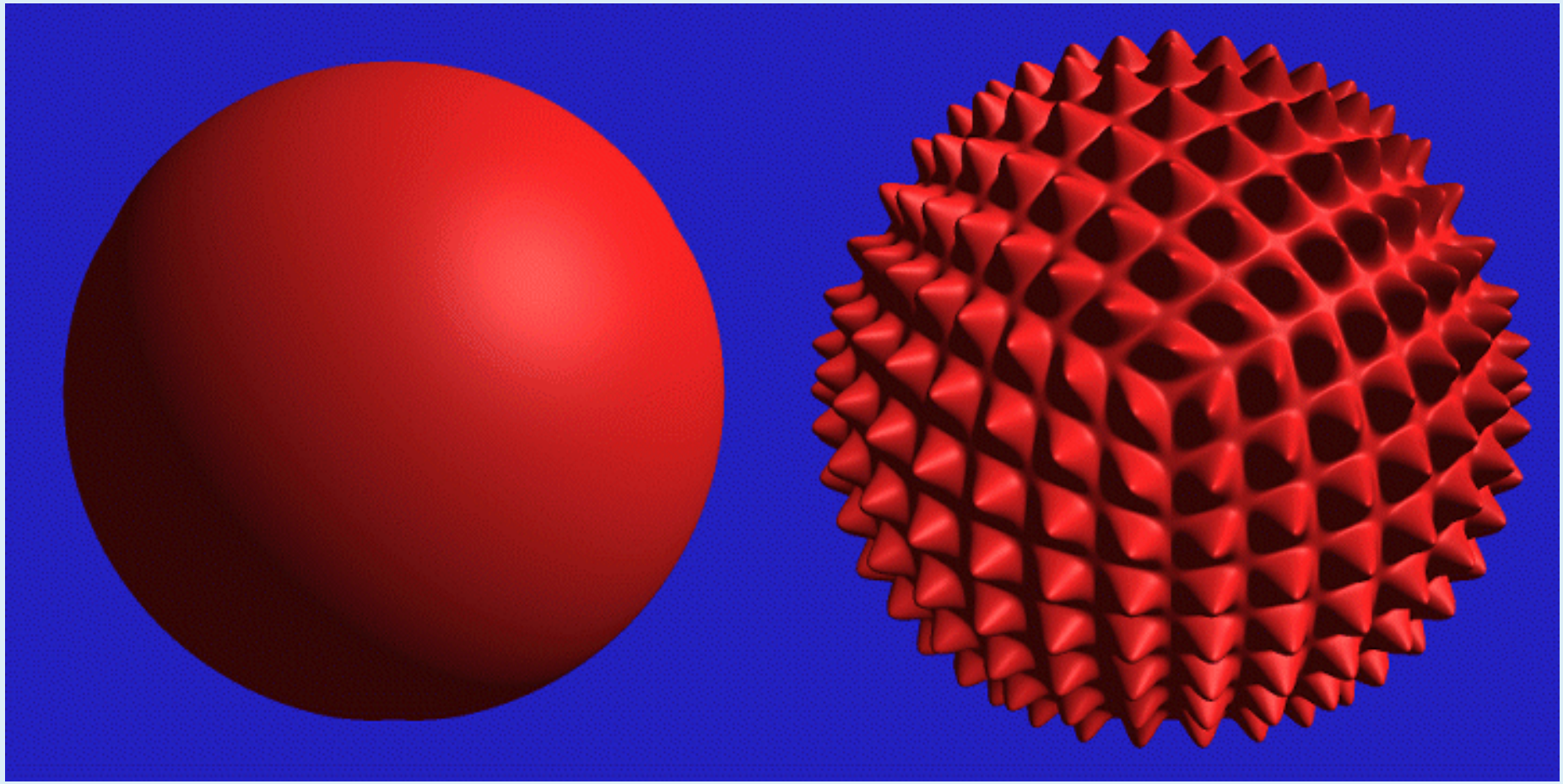
- Bump-Mapping erzeugt den Eindruck einer rauhen Oberfläche
- Konturen / Schatten bleiben die der Geometrie
- Displacement Mapping verändert die Geometrie des Modells durch Verändern de Koordinaten von Oberflächenpunkten
- Grobgeometrie + Displacement Map => Feingeometrie



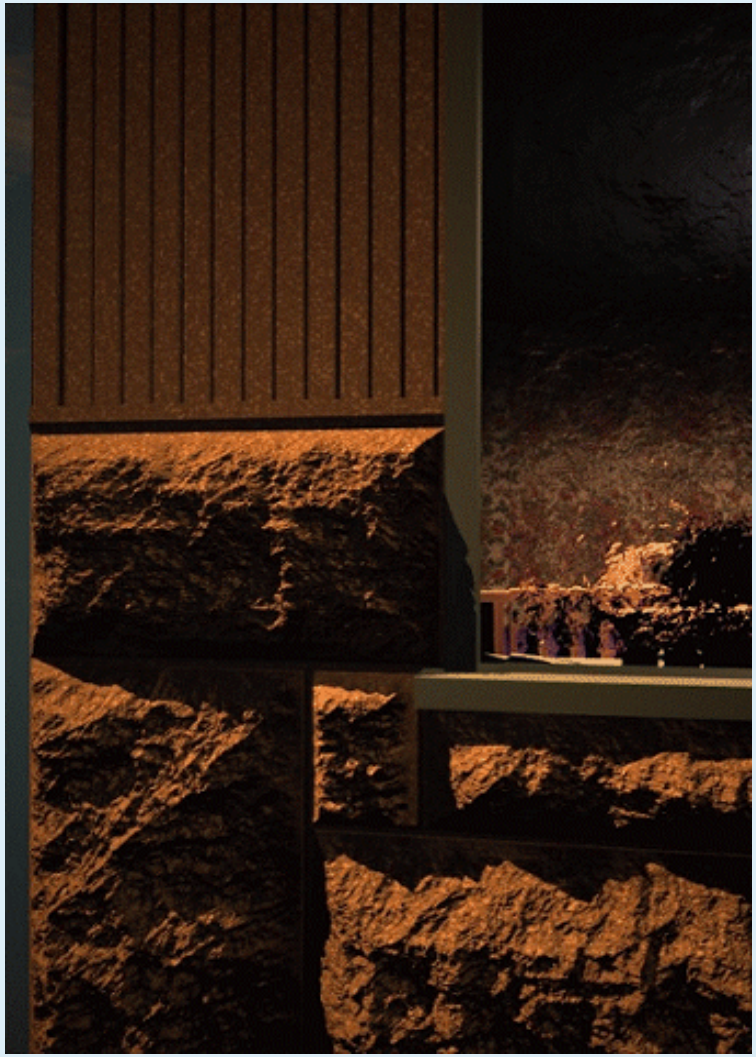
- hellere Texel bedeuten stärkere Verschiebung

1. Texture Mapping

1.4. Displacement Mapping



1. Texture Mapping
1.4. Displacement Mapping



Quelle: http://art.sacada.net/forum/album_picm.php?pic_id=211

Quelle: Matt Pharr, Pat Hanrahan: *Geometry Caching for Ray-Tracing Displacement Maps*

1. Texture Mapping
1.5. Zusammenfassung

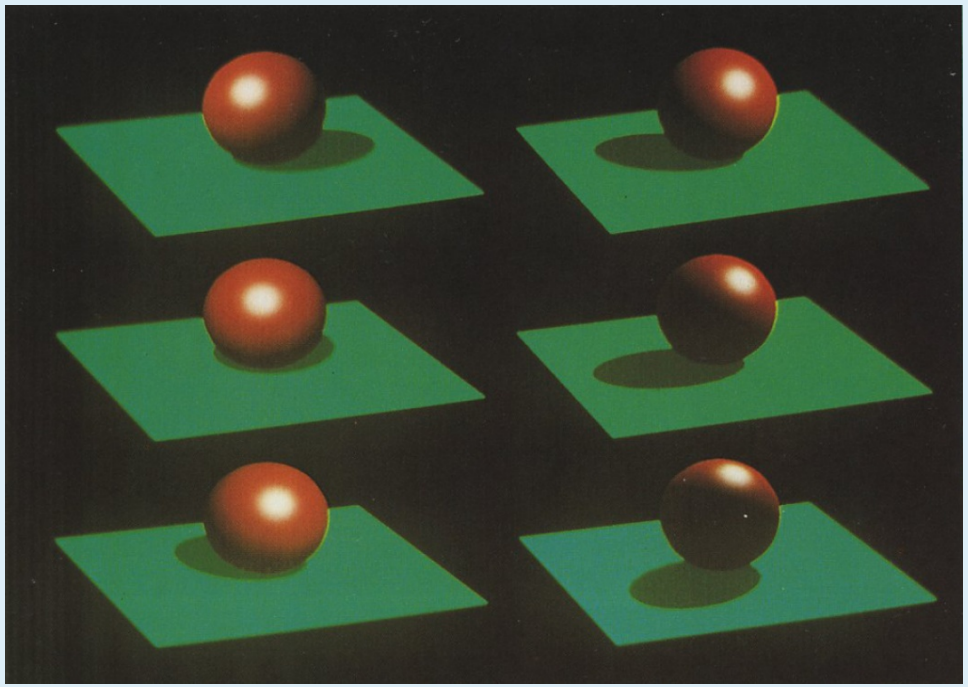
- Vorteile des Texture Mappings
  - Szene interessanter gestalten
  - Realismus erhöhen durch verhältnismäßig wenig Rechenaufwand
  - Einfache Möglichkeit Preprocessing zu integrieren (global illumination)

1. Texture Mapping
1.5. Zusammenfassung

- Nachteile/Probleme des Texture Mappings
  - Eventuell stark sichtbare Aliasing-Artefakte
  - genaue Berechnung der u,v-Paramter schwierig (Interpolieren zwischen Gitterpunkten)
  - durch verstärkte Implementierung von Texture Mapping Funktionalitäten in Grafikhardware Konzentration auf typisches Szenario 2D-Texture auf Oberfläche

2. Schattenberechnung

- „verankert“ Objekte in der Szene (keine „fliegenden“ Objekte)
- hebt die Richtung der Beleuchtung hervor
- Steigerung des Realismus des Bildes
- Schatten geben Hinweise zum Aufbau der Szene
- Verbesserung der Tiefenwahrnehmung
- verwendbar für Simulationen (Energie-, Wärmeverteilung)

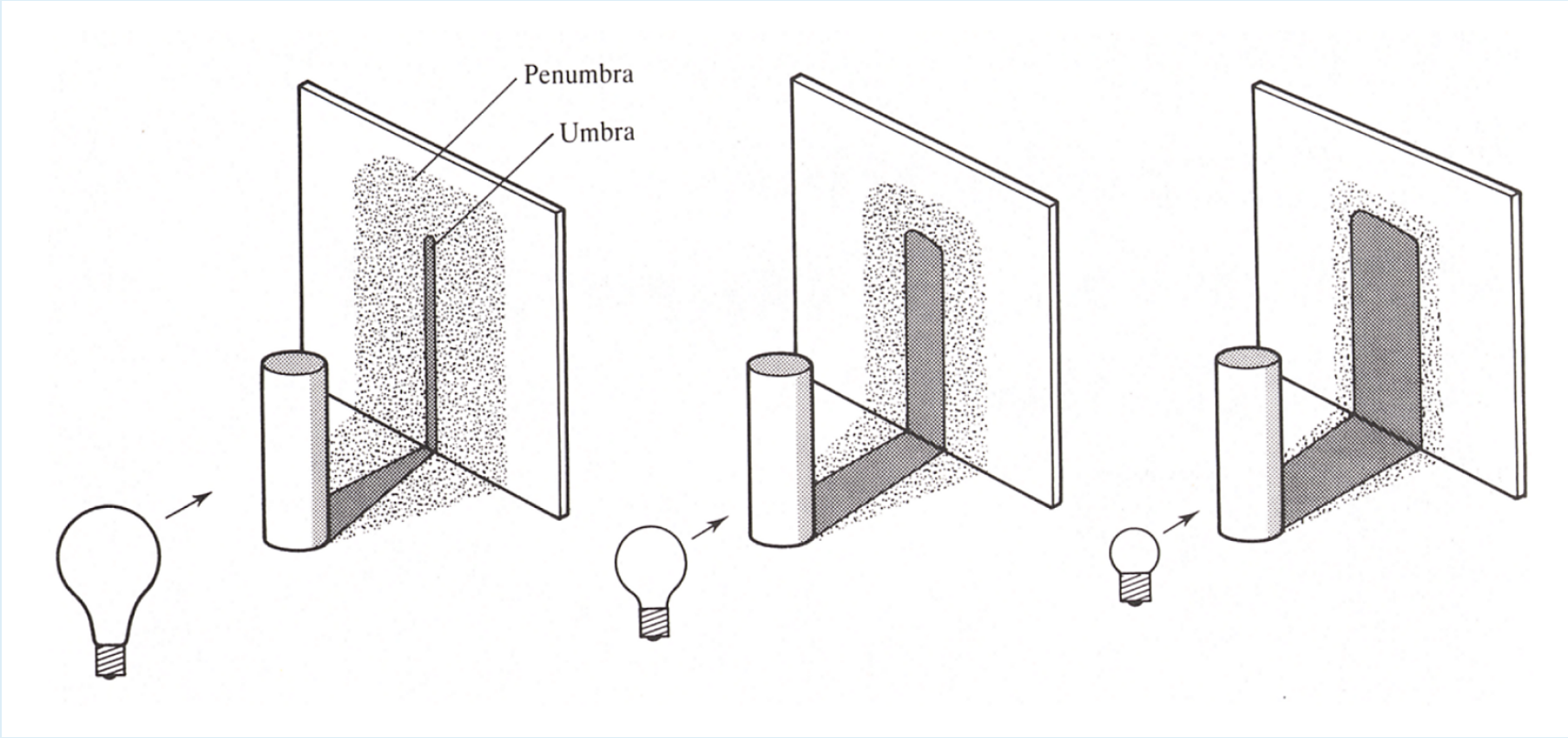


2. Schattenberechnung
2.1. Kern- und Halbschatten

- Schatten bestehen aus zwei Teilen: Kernschatten (umbra) und Halbschatten (penumbra)
- Kernschatten: zentraler, scharf abgegrenzter dunkler Teil im Zentrum
- Halbschatten: helleres Gebiet, das den Kernschatten umgibt
- Punktlichtquellen generieren nur Kernschatten
- flächige Lichtquellen erzeugen beides, Kernschatten erhalten gar kein Licht, Halbschatten nur wenig (wegen der Ausdehnung der Lichtquellen)
- Halbschatten sehr aufwendig zu berechnen, da in der Computergraphik Punktlichtquellen eingesetzt werden, häufig nur Kernschattenberechnung

2. Schattenberechnung

2.1. Kern- und Halbschatten



2. Schattenberechnung
2.2. Position der Lichtquelle

- Komplexität der Schattenberechnung abhängig von Position der Lichtquelle
- keine Schatten, wenn (einzige) Lichtquelle im Betrachterstandpunkt
- Lichtquelle im Unendlichen: orthographische Projektion zur Bestimmung des Schattens → einfachster Fall
- Lichtquelle an endlicher Position außerhalb des Gesichtsfeldes: perspektivische Projektion zur Bestimmung des Schattens → etwas komplizierter
- Lichtquelle innerhalb des Gesichtsfeldes: Unterteilung der Szene in Sektoren, Schattenberechnung separat für die Sektoren → kompliziertester Fall
- Schatten ändern sich bei Animation

2. Schattenberechnung
2.3. Grundlegende Vorgehensweise

- HSR/VSD-Algorithmen bestimmen, welche Flächen vom Betrachterstandpunkt aus sichtbar sind.
- Schattenberechnung bedeutet, die Flächen zu bestimmen, die von der Position der Lichtquelle aus nicht sichtbar sind.
- → VSD/HSR-Algorithmen können genutzt werden.
- Sichtbarkeit aus Sicht der Lichtquelle: entweder sichtbar oder nicht
- wenn Oberflächenpunkt von Lichtquelle aus nicht sichtbar
  - Punkt liegt im Schatten
  - Beleuchtungsberechnung muß entsprechend angepaßt werden

2. Schattenberechnung

2.3. Grundlegende Vorgehensweise

- Faktor *S* bestimmt, ob Licht von Lichtquelle ein Objekt an einem gegebenen Punkt erreicht:

$$S = \begin{cases} 0 & \text{wenn Lichtquelle blockiert ist} \\ 1 & \text{wenn Lichtquelle nicht blockiert ist} \end{cases}$$

- Hinzufügen zur Beleuchtungsberechnung:

$$I_{Phong} = L_a k_a + S\left(\frac{1}{a + bd + cd^2}\left(L_i k_d (LN) + L_i k_s (RV)^e\right)\right)$$

- Gebiete im Schatten immer noch durch das ambiente Licht beleuchtet

2. Schattenberechnung
2.4. Algorithmen

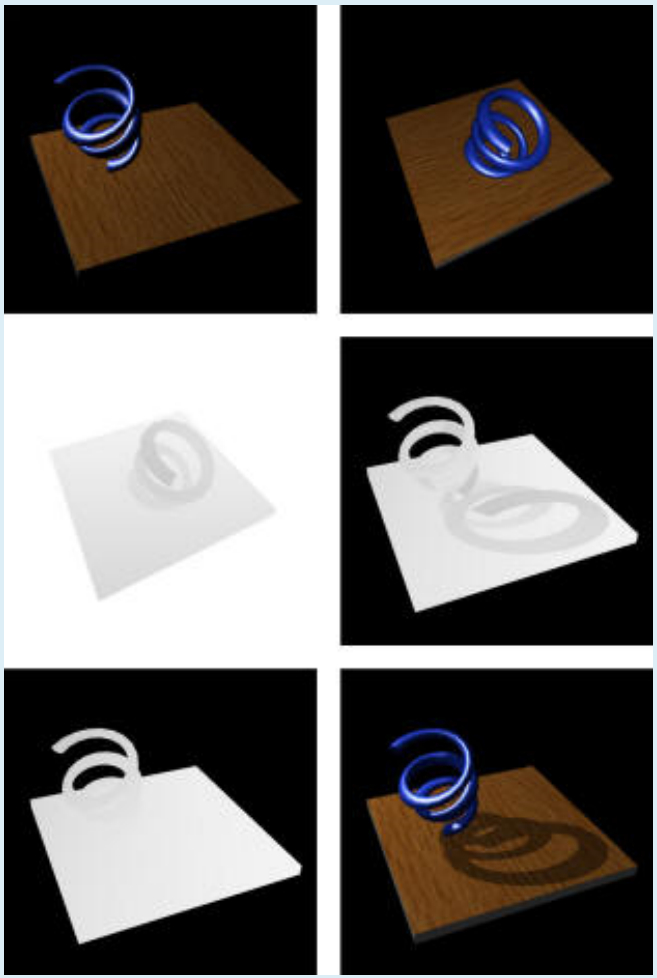
- Projektive Schatten
  - Schatten wird aus Sicht der Lichtquelle auf die Objekte projiziert
  - nur sinnvoll bei Ebenen (Schatten auf einer Grundebene)
- Schattenvolumen
  - Raum zwischen der Fläche die Schatten wirft und einer (entweder im unendlichen, oder wirklichen) Fläche, auf die der Schatten fällt
  - Berechnen der Begrenzungspolygone dieses raumes
  - Man nehme einen Strahl vom Betrachterstandpunkt zu einem Punkt P in der Szene
  - P liegt im Schatten, wenn dieser Strahl öfter von außen nach innen in das Schattenvolumen geht als umgekehrt
  - Spezialfall, wenn Betrachter im Schattenvolumen liegt

2. Schattenberechnung
2.4. Algorithmen

- Hidden Surface Removal aus Sicht der Lichtquelle
  - Verwenden eines Objektraum-HSR-Algorithmus, um die Schattenpolygone zu bestimmen
  - Transformieren dieser in die Kameraansicht und Verknüpfen mit den Objektpolygonen bei der Beleuchtungsberechnung
- Shadow z-Buffer
  - seperater Z-Buffer für die Schattenberechnung einer Lichtquelle
  - Schatten-Z-Buffer wird gefüllt, indem die Szene aus der Sicht der Lichtquelle betrachtet wird
  - Szene mit dem normalen Z-Buffer rendern.
  - Für sichtbare Punkte Transformation in die Koordinaten für den Schatten-Z-Buffer und Vergleichen der z-Werte mit denen aus dem Schatten-Z-Buffer verglichen
  - Darstellung als im Schatten oder nicht abhängig vom Ergebnis des Vergleichs

2. Schattenberechnung
2.4. Algorithmen

- Shadow Maps (Schatten-Z-Buffer)
  a) Kamerasicht der Szene ohne Schatten
  b) Ansicht der Szene von der Lichtquelle
  c) Shadow map (depth map) konstruiert von der Lichtquelle aus
  d) Shadow map projiziert in die Kamerasicht
  e) Entfernung zur Lichtquelle projiziert in die Kamerasicht
  f) Szene mit Schatten



3. Zusammenfassung

- Texturen und Schatten tragen zur Erhöhung des Realismus bei
- Heute Standard-Techniken bei Real Time Graphiken
- Programmierbare Shader in Graphikkarten für Mapping-Algorithmen gut geeignet
- Weitere Mapping-Algorithmen für weitere Effekte auf dem Vormarsch (wegen der Hardware-Entwicklung)