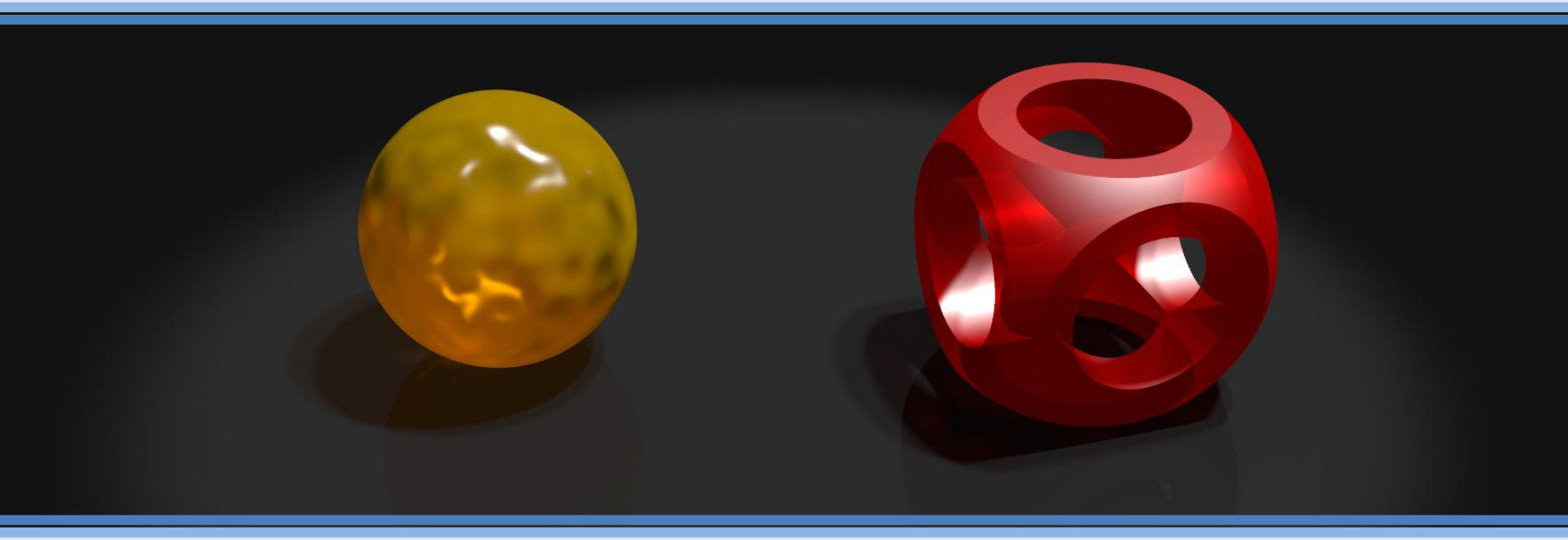
Computergrafik

T. Hopp

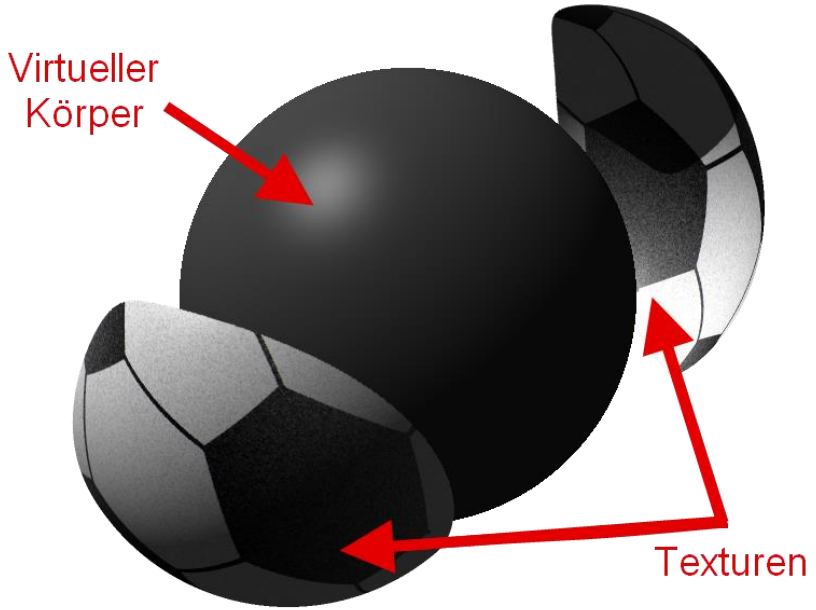


Themenübersicht

1. Einführung
2. Programmierbibliotheken / OpenGL
3. Geometrische Repräsentation von Objekten
4. Koordinatensysteme und Transformationen
5. Zeichenalgorithmen
6. Buffer-Konzepte
7. Farbe, Beleuchtung und Schattierung
8. **Texturen**
9. Animationen
10. Raytracing
11. Volumenvisualisierung

Texturen

- Bisher: Farbzuordnung eines Pixels basierend auf Shadingverfahren, Farbdefinition etc.
  - i.d.R. künstliches Aussehen
  - Extrem hoher Aufwand um Fotorealismus zu erreichen
- Textur: „Überzug“ eines Objektes mit einer Struktur
  - Einfachste Variante: Texture Mapping



https://de.wikipedia.org/wiki/Textur_%28Computergrafik%29

Anwendungen von Texturen

- **Texture Mapping**: „Fototapete“
  - Ändert diffuse Reflexion eines Objektes
- Gloss Mapping: Glanztexturen
  - Ändert spekulare Reflexion eines Objektes
- **Bump Mapping**: Reliefartige Darstellung
  - Ändert Normalenfeld eines Objektes
- **Environment**-/Shadow-**Mapping**: Approximation globaler Beleuchtung
- Light-Mapping: Imitation von Radiosity-Verfahren

8.1. TEXTURE MAPPING

Texture Mapping

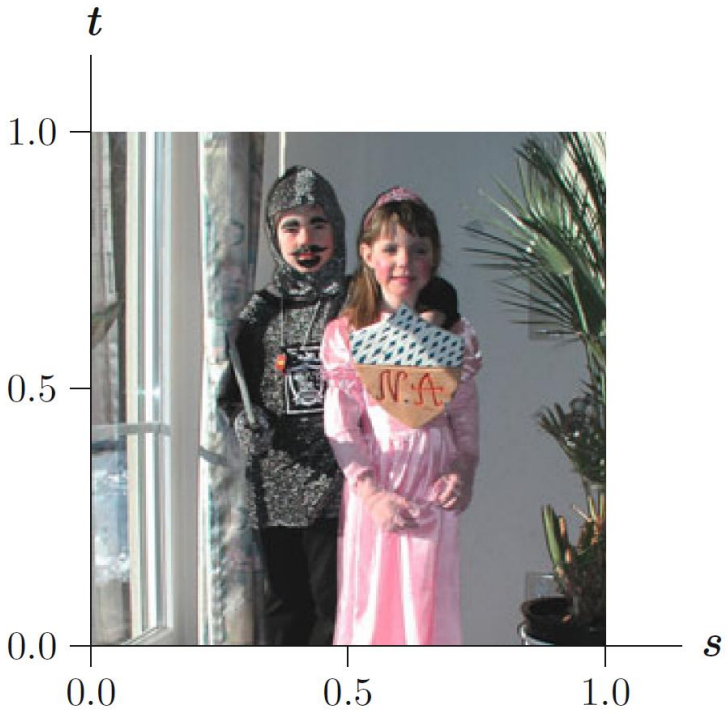
- Verwendung von Bildern als Texturen
  - 1D, 2D, 3D
  - Grauwertbilder, Farbbilder, Farbbilder + Alpha-Kanal
- „Überziehen“ eines Objektes mit einem Bild
  - Zuordnung: Texturkoordinate auf Vertices der Oberfläche
  - Lineare Interpolation in der Rasterisierungsstufe der Renderingpipeline
- Ablauf beim Texture Mapping:
  - Spezifikation der Textur
  - Festlegung wie die Textur auf jedes Pixel aufgetragen wird
    - Texturfilter
    - Mischung von Textur und Beleuchtungsfarbe
  - Zuordnung von Texturkoordinaten zu Vertices
  - Einschalten des Texture Mappings: **`glEnable(GL_TEXTURE_2D)`**

Texture Mapping

- Typische Transformationen

Spezifikation der Textur

- Bildmatrix: $\mathbf{T}$
- Texturkoordinatensystem: (s, t), normiert auf Parameterbereich $[0, 1]$
- Bildpunkt = „Texel“:
  - 1 Wert (Graustufen)
  - 2 Werte (Grausstufen + Transparenz)
  - 3 Werte (Farbe)
  - 4 Werte (Farbe + Transparenz)
- Quantisierung: 1 Bit, 4 Bit, … 32 Bit
- OpenGL: **`glTexImage2D(…)`**
  - Daten aus dem Hauptspeicher



A. Nischwitz et al. „Computergrafik und Bildverarbeitung“, Band 1: Computergrafik“

Spezifikation der Textur

- Beispiel in OpenGL: 512 x 256 Texel, RGBA

```
GLint width = 512;
GLint height = 256;
static GLubyte image[width][height][4];
:
:
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA,
          width, height, 0,
          GL_RGBA, GL_UNSIGNED_BYTE, image);
```

Level (siehe Mipmapping)

Border (Breite des Randes der Textur)

Datentyp

- Laden/Einlesen des Bildes (hier: *image*) obliegt dem Programmierer
- Oft Skalierung/Zuschneiden des Bildes erforderlich, z.B. auf Zweierpotenz.
  - Extern, oder per GLU-Bibliothek: **`gluScaleImage(…)`**
- Textur ist Teil des OpenGL Zustandsautomaten
  - Zustandswechsel möglichst minimieren!

Spezifikation der Textur

- Texturspeicher: extrem schneller Zugriff innerhalb der Grafikkarte
- Überprüfung ob Speicher für Textur ausreicht:

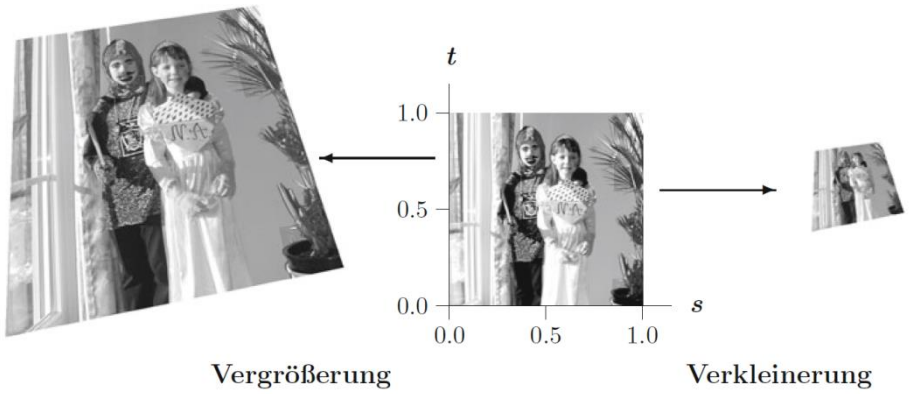
```
glTexImage2D(GL_PROXY_TEXTURE_2D, …);
glGetTexLevelParameteriv(GL_PROXY_TEXTURE_2D,
                         level, GL_TEXTURE_WIDTH,
                         &width);
```

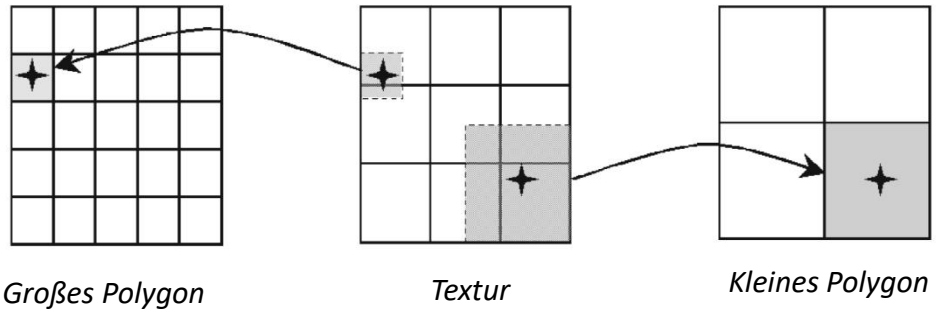
  - Width = 0 → Texturspeicher reicht nicht aus

- Statt Neudefinition auch Modifikation einer Textur möglich
- Sub-Texturen zur Vermeidung von Neudefinitionen
- Multipass-Rendering: rekursive Verwendung von Texturen aus dem Bildspeicher
  - Erheblicher Geschwindigkeitsverlust bei steigender Rekursionstiefe

Textur-Filter

- Beim Abbilden der Texturkoordinaten in die 3D-Szene können erhebliche Verzerrungen der Textur entstehen





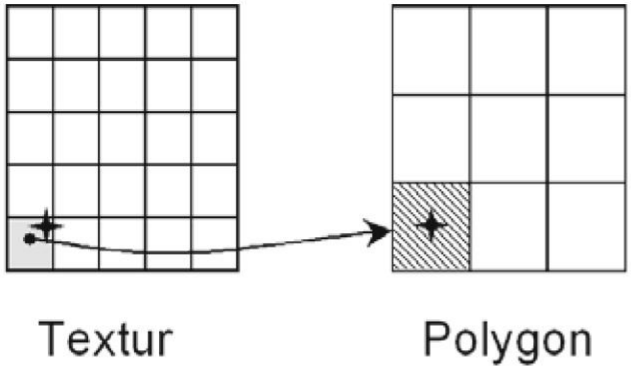
- Keine eindeutige Zuordnung zwischen Texel und Pixel
- Interpolation erforderlich

A. Nischwitz et al. „Computergrafik und Bildverarbeitung“, Band 1: Computergrafik“

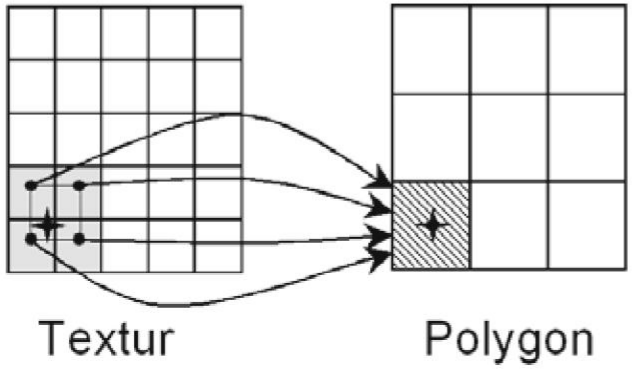
Textur-Filter

Standard-Filter-Methoden

Nearest neighbour



Linear



A. Nischwitz et al. „Computergrafik und Bildverarbeitung“, Band 1: Computergrafik“

Mipmapping

- Vergrößerungsfaktor sollte nicht > 2 oder $< \frac{1}{2}$ werden.
- Mipmapping: Vorfilterung der Texturen und optimale Skalierung
  - Pro Level: Halbierung der Kantenlänge bis zur Größe 1×1

A. Nischwitz et al. „Computergrafik und Bildverarbeitung“, Band 1: Computergrafik“

Mipmapping

- Berechnung „offline“ (z.B. in der Initialisierung)
- Generierung der Gauß-Pyramide:
  - In externem Programm: Aufruf von **`glTexImage2D(…)`** zum Einlesen jedes Levels
  - Per GLU-Bibliothek: **`gluBuild2DMipmaps()`**
- Speicherbedarf: $1 + {}^{1}/_{4} + {}^{1}/_{16} + \ \ldots = 1 + \sum_{n\to\infty} \frac{1}{n^2} = 1\frac{1}{3}$

- Auswahl des Mipmap-Levels:
  - Verkleinerungsfaktoren in x-/y-Richtung: ρ_x, ρ_y
  - Berechnung des Skalierungsfaktors $\lambda = -\log_2\left(\max\left(\frac{1}{\rho_x}, \frac{1}{\rho_y}\right)\right)$
  - Skalierungsfaktor entspricht Mipmap-Level

  Beispiel: maximaler Verkleinerungsfaktor $\rho = \frac{1}{4}$ ➔ $\lambda = 2$

- Verkleinerungsfilter: zwei Interpolationen (Skalierungsfaktor, Farbwerte)

Mipmapping

http://www.tomshardware.com/reviews/ati,819-2.html

Mipmapping

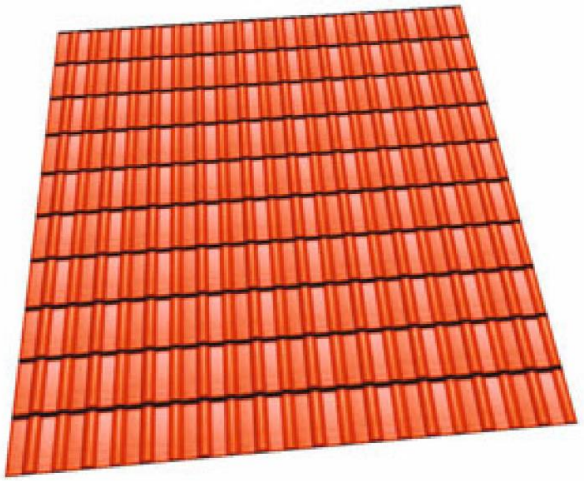
- Probleme bei sehr flachem Blickwinkel

- Mögliche Abhilfen:
  - Vorberechnung zusätzlicher Level:
    - z.B. 8×8-Textur: 8×4, 8×2, 8×1, 4×8, 2×8, 1×8, 4×4,...
    - Hoher Speicherbedarf, heute meist nicht mehr verwendet

  - Abhilfe durch anisotrope Filterung:
    - Auswahl der Mipmap für geringeren Verkleinerungsfaktor
    - Vermeidung von Aliasing-Artefakten: in der anderen Richtung durch Mittelwertbildung über größere Anzahl an Texeln (stärkere Filterung → „anisotrop“)

Texture Wraps

- Texturen in vielen Fällen regelmäßige Struktur
- Textur-Fortsetzungsmodus (Texture Wrap):
  - Definition kleiner Ausschnitte
  - Fortsetzung der Textur über gesamtes Polygon

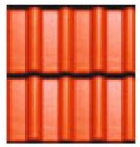


- OpenGL: Zulassen von Texturkoordinaten außerhalb des Bereichs $[0,1]$
  - Z.B. $[0,5]$: fünfmalige Wiederholung in jeweils x- und y-Richtung = 25 Kacheln
  - In OpenGL: hier Texturfortsetzungsmodus **GL_REPEAT**

A. Nischwitz et al. „Computergrafik und Bildverarbeitung“, Band 1: Computergrafik“

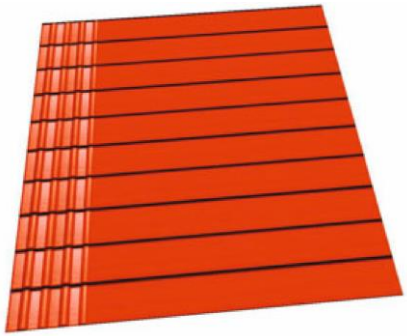
Texture Wraps

- Textur muss geeignet sein:
  - Unterer Rand ähnlich oberem Rand
  - Linker Rand ähnlich rechtem Rand



- Möglichkeit zur Spiegelung
  - In OpenGL: Texturfortsetzungsmodus auf **`GL_MIRRORED_REPEAT`**
- Kappen von Werten außerhalb der Textur: Wiederholung der ersten/letzten Zeile/Spalte
  - In OpenGL: **`GL_CLAMP`**

- Getrennte Einstellung für x-/y-Richtung



A. Nischwitz et al. „Computergrafik und Bildverarbeitung“, Band 1: Computergrafik“

Mischung von Textur und Beleuchtung

- Textur-Fragmentfarbe g_t muss mit Beleuchtungsfarbe g_f aus dem Shading kombiniert werden.

```
glTexEnvf(target, GL_TEXTURE_ENV_MODE, param)
```

- Verschiedene Kombinationsmöglichkeiten:
  - Ersetzen (param = **`GL_REPLACE`**): resultierende Fragmentfarbe $g_r = g_t$
  - Modulieren (param = **`GL_MODULATE`**): komponentenweise Multiplikation:

$$g_r = \begin{pmatrix} R_t \cdot R_f \\ G_t \cdot G_f \\ B_t \cdot B_f \\ A_t \cdot A_f \end{pmatrix}$$

  - Gewichtung mit der Alpha-Komponente der Textur (param = **`GL_DECAL`**)

$$g_r = \begin{pmatrix} (1 - A_t)R_f + A_t R_t \\ (1 - A_t)G_f + A_t G_t \\ (1 - A_t)B_f + A_t B_t \\ A_f \end{pmatrix}$$

Mischung von Textur und Beleuchtung

- Gewichtung mit dem Farbwert der Textur (param = **`GL_BLEND`**)

$$g_r = \begin{pmatrix} (1-R_t)R_f + R_t R_h \\ (1-G_t)G_f + G_t G_h \\ (1-B_t)B_f + B_t B_h \\ A_t \cdot A_f \end{pmatrix}$$

- Addition (param = **`GL_ADD`**): komponentenweise Addition:

$$g_r = \begin{pmatrix} R_t + R_f \\ G_t + G_f \\ B_t + B_f \\ A_t \cdot A_f \end{pmatrix}$$

- Kombination von Beleuchtung und Textur meist per Modulation
  - Definition weißer Materialeigenschaften für Shading → Textur bestimmt Farbe, Beleuchtung bestimmt Helligkeit (3D-Eindruck!)

Mischung von Textur und Beleuchtung

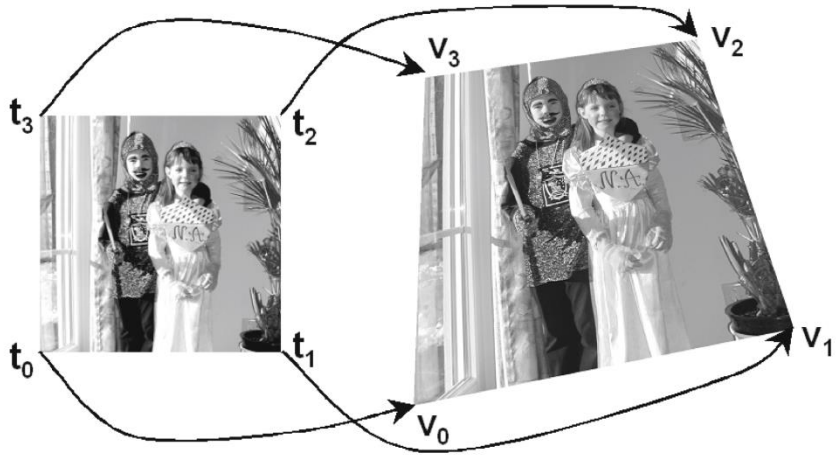
- Glanzlichter (spekulare Beleuchtung) gehen meist verloren
- Abhilfe: Berechnung von zwei Farben pro Vertex/Fragment:
  - Primäre Farbe: diffuser, ambienter und submissiver Beleuchtungsanteil
  - Sekundäre Farbe: spekularer Anteil
- Nachträgliche Addition der Sekundärfarbe
- Beschränkung auf den Wertebereich [0, 1]

http://www.3dgep.com/texturing-and-lighting-with-opengl-and-glsl/

Zuordnung von Texturkoordinaten zu Vertices

- Texturabbildung: $(s, t) \rightarrow (x_w, y_w, z_w)$
  - Jedem Vertex wird eine Texel-Koordinate in der Textur zugewiesen
- Explizite Zuordnung:
  - Definition einer Texel-Koordinate in OpenGL: **`glTexCoord2f`**
  - Zuordnung bei der Definition eines Polygons
  - Beispiel 2D-Textur auf 3D-Polygon aufbringen

```
glBegin(GL_QUADS);
   glTexCoord2fv(t0); glVertex3fv(v0);
   glTexCoord2fv(t1); glVertex3fv(v1);
   glTexCoord2fv(t2); glVertex3fv(v2);
   glTexCoord2fv(t3); glVertex3fv(v3);
glEnd();
```



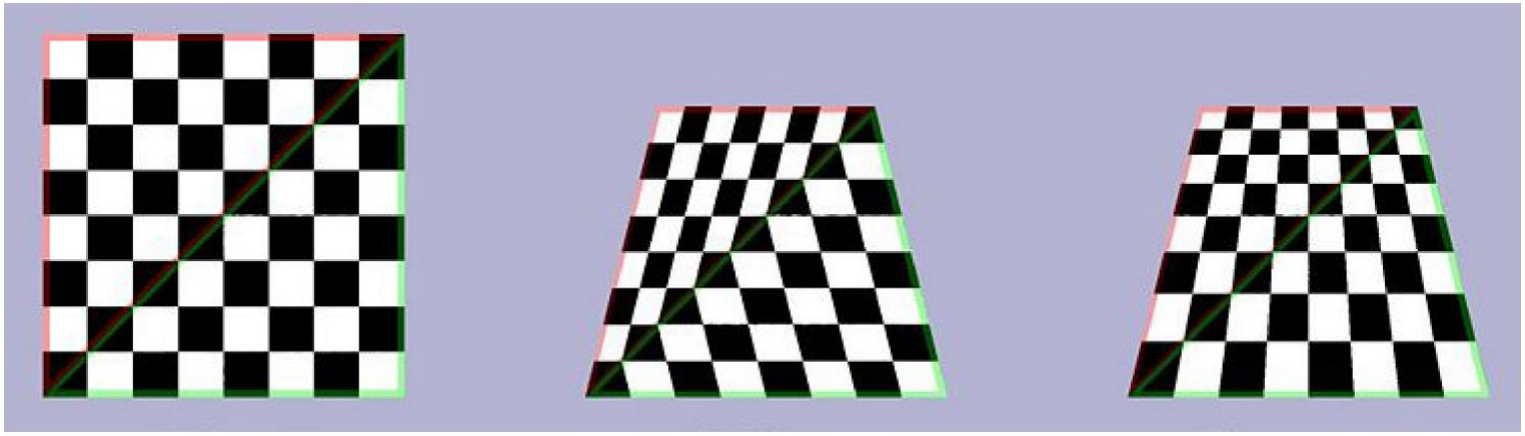
- Rasterisierungsstufe führt Interpolation der Texturkoordinaten durch.

A. Nischwitz et al. „Computergrafik und Bildverarbeitung“, Band 1: Computergrafik“

Zuordnung von Texturkoordinaten zu Vertices

- Perspektivische Projektion: unbefriedigenden Ergebnisse, da Texturkoordinaten nach der Projektion interpoliert werden.
- Lösung: Perspektivkorrektur
  - Statt s und t: $\frac{s}{z}, \frac{t}{z}$ sowie $\frac{1}{z}$ linear interpolieren (z = Koordinate in Sichtrichtung)
- Um für ein Pixel die Texturkoordinate zu berechnen gilt nun:

$$s' = {}^{\frac{s}{z}}\Big/_{\frac{1}{z}} \quad \text{bzw.} \quad t' = {}^{\frac{t}{z}}\Big/_{\frac{1}{z}}$$



https://en.wikipedia.org/wiki/Texture_mapping

Zuordnung von Texturkoordinaten zu Vertices

- Ungleiche Seitenverhältnisse von Textur und Polygon:
  - Verzerrung der Textur (b)
  - Ausschnitt aus der Textur ohne Verzerrung (c)
  - Wiederholung der Textur ohne Verzerrung (d)

(a)

(b)

(c)

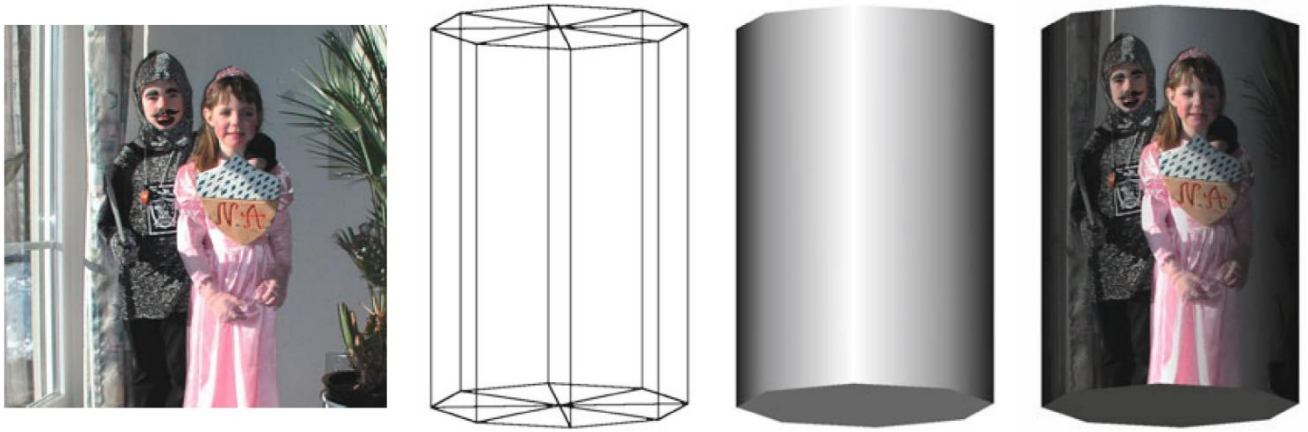
(d)

A. Nischwitz et al. „Computergrafik und Bildverarbeitung“, Band 1: Computergrafik“

Zuordnung von Texturkoordinaten zu Vertices

- Beispiel: Auftragen einer Textur auf Manteloberfläche eines Zylinders
  - Grundidee: Abwicklung der Mantelfläche ergibt Rechteck

```
glBegin(GL_QUAD_STRIP);
   for(phi=0; phi<2*3.1415; phi+=3.1415/4) {
      x = cos(phi);
      y = sin(phi);
      s = phi/(2*3.1415);
      glNormal3f(x,y,0); glTexCoord2f(s,0); glVertex3f(r*x,r*y,0);
      glNormal3f(x,y,0); glTexCoord2f(s,1); glVertex3f(r*x,r*y,h);
   }
glEnd();
```



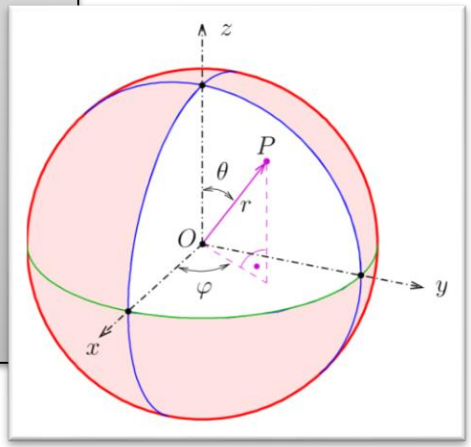
A. Nischwitz et al. „Computergrafik und Bildverarbeitung“, Band 1: Computergrafik“

Zuordnung von Texturkoordinaten zu Vertices

- Beispiel: Auftragen einer Textur auf Kugeloberfläche

```
for(theta=3.1416/2; theta>-3.1416/2; theta-=3.1416/32) {
   thetaN = theta - 3.1416/32;
   glBegin(GL_QUAD_STRIP);
      for(phi=0; phi<2*3.1416; phi+=3.1416/32) {
         x = cos(phi) * cos(theta);
         y = sin(phi) * cos(theta);
         z = sin(theta);
         s = phi/(2*3.1416);
         t = theta/3.1416 + 0.5;
         glNormal3f(x,y,z); glTexCoord2f(s,t); glVertex3f(r*x,r*y,r*z);

         x = cos(phi) * cos(thetaN);
         y = sin(phi) * cos(thetaN);
         z = sin(thetaN);
         t = thetaN/3.1416 + 0.5;
         glNormal3f(x,y,z); glTexCoord2f(s,t); glVertex3f(r*x,r*y,r*z);
      }
   glEnd();
}
```



https://de.wikipedia.org/wiki/Kugel

Zuordnung von Texturkoordinaten zu Vertices

- Starke Verzerrung, insb. am Pol
- Verzerrungen abhängig von der Krümmung der Oberfläche

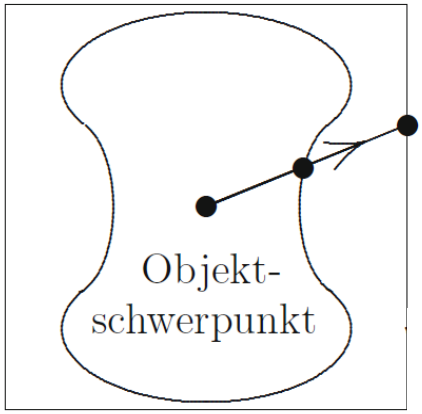
A. Nischwitz et al. „Computergrafik und Bildverarbeitung“, Band 1: Computergrafik“

Zuordnung von Texturkoordinaten zu Vertices

- Allgemeiner Fall: Aufbringen von Texturen auf Polygonnetze
- Zweiteiliges Abbilden:
  - S-Mapping: Textur auf analytisch beschreibbare Oberfläche, z.B. Ebene, aufbringen
  - O-Mapping: Texturkoordinaten der Zwischeneben auf eigentliches Objekt abbilden

- O-Mapping-Varianten:

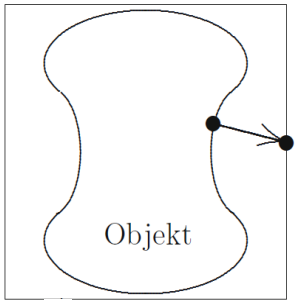
  (1) Strahl von Objektschwerpunkt durch Vertex der Oberfläche auf Zwischenebene



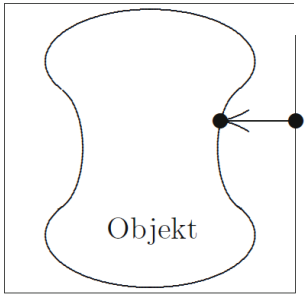
A. Nischwitz et al. „Computergrafik und Bildverarbeitung“, Band 1: Computergrafik“

Zuordnung von Texturkoordinaten zu Vertices

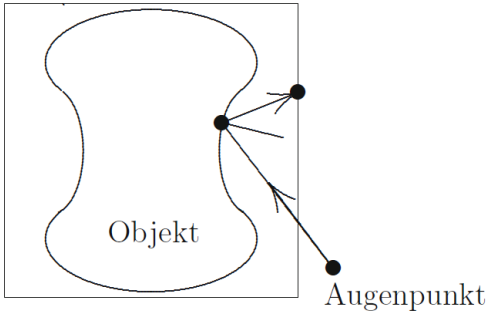
(2) Schnittpunkt von Vertex-Normalenvektor mit Zwischenebene



(3) Schnittpunkt des Zwischenebenen-Normalenvektors am Oberflächen-Vertex mit Zwischenebene



(4) Schnittpunkt der Zwischenebene mit der idealen Reflexion eines Strahls an der Oberfläche, der vom Augpunkt ausgeht.



A. Nischwitz et al. „Computergrafik und Bildverarbeitung“, Band 1: Computergrafik“

Zuordnung von Texturkoordinaten zu Vertices

- OpenGL erlaubt eine automatische Texturkoordinatengenerierung
  - Beschreibung einer Zwischenebene in Abhängigkeit der Vertices in Objekt- oder Weltkoordinaten (S-Mapping)

    `glTexGen*(coord, name, param)`

    mit `name` = GL_TEXTURE_GEN_MODE
- `param` = **`GL_OBJECT_LINEAR`**: feste Verbindung von Textur und Objekt

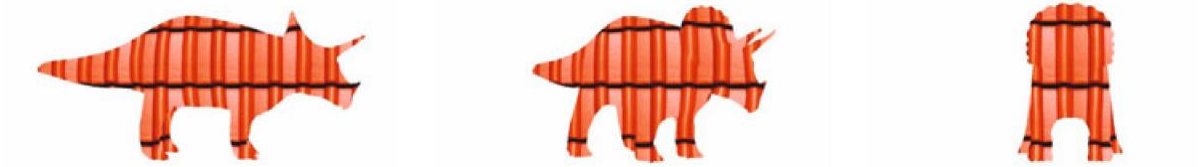
$$s(x_o, y_o, z_o, w_o) = ax_o + by_o + cz_o + dw_o$$

Objektkoordinaten

- `param` = **`GL_EYE_LINEAR`**: feste Verbindung von Textur und Augpunkt

$$s(x_e, y_e, z_e, w_e) = ax_e + by_e + cz_e + dw_e$$

Weltkoordinaten

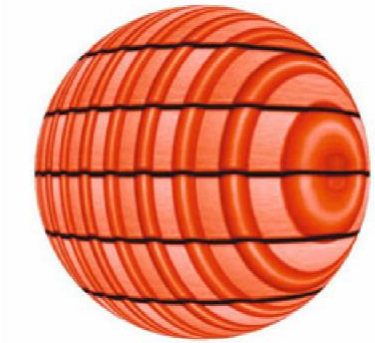


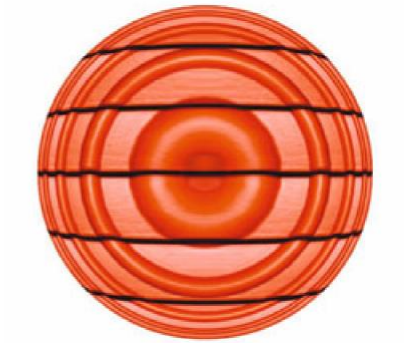
- Parameter a,b,c,d definieren Skalierung und Lage der Ebene (Koordinatenform)

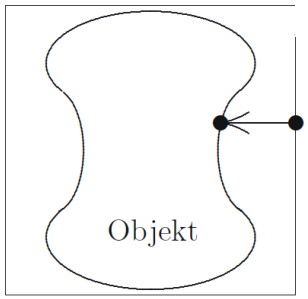
A. Nischwitz et al. „Computergrafik und Bildverarbeitung“, Band 1: Computergrafik“

Zuordnung von Texturkoordinaten zu Vertices

- Ebenengleichung definiert Mapping der Texturkoordinaten durch orthographische Projektion auf Vertices eines Objektes (O-Mapping)





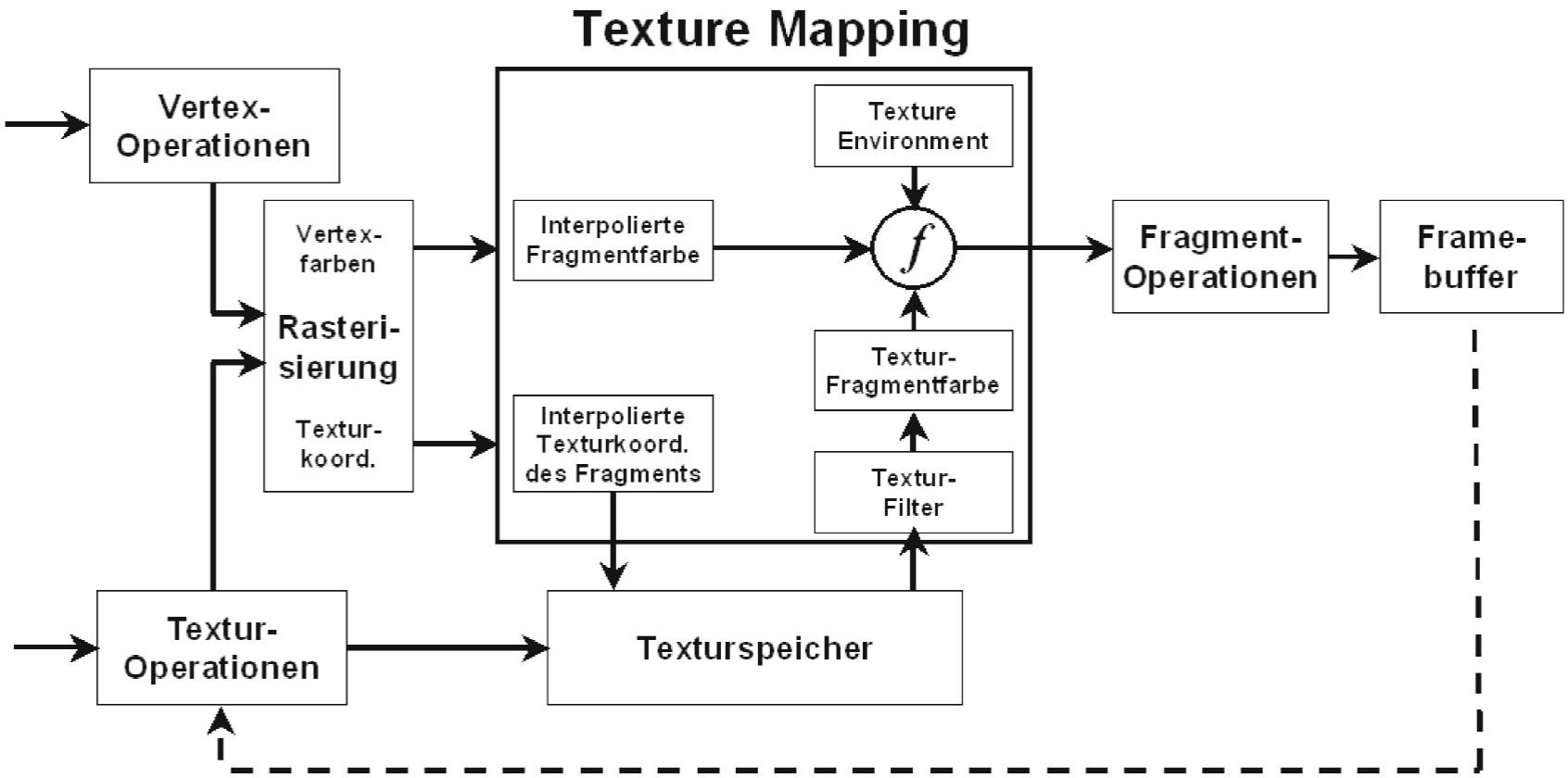


- Beispiel: Angabe der Parameter der Ebenengleichung

```
GLfloat s_plane[] = (a,b,c,d);
glTexGeni(GL_S, GL_TEXTURE_GEN_MODE, GL_OBJECT_LINEAR);
glTexGenfv(GL_S, GL_TEXTURE_OBJECT_PLANE, s_plane);
glEnable(GL_TEXTURE_GEN_S);
```

A. Nischwitz et al. „Computergrafik und Bildverarbeitung“, Band 1: Computergrafik“

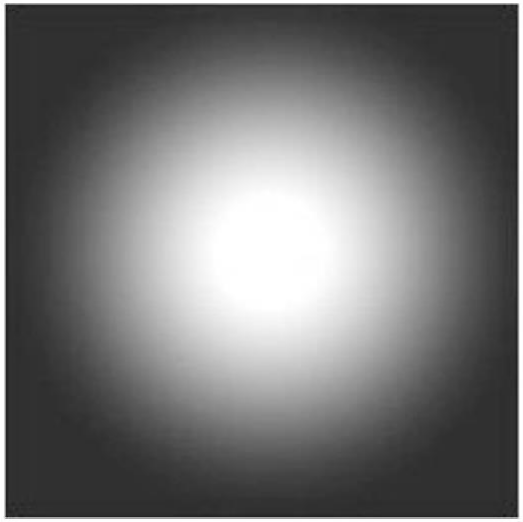
Texture Mapping: Ablauf



A. Nischwitz et al. „Computergrafik und Bildverarbeitung“, Band 1: Computergrafik“

Multitexturing

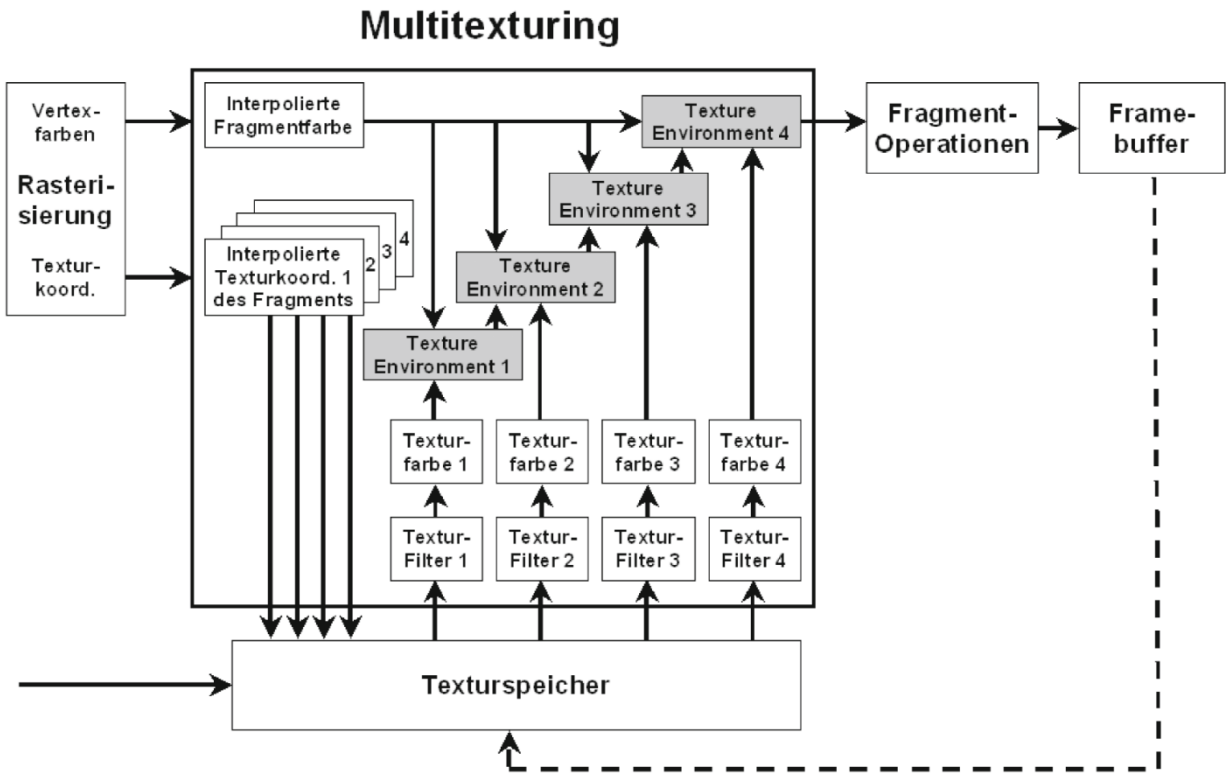
- Bisher: Aufbringen einer Textur auf ein Polygon
- Noch mehr Möglichkeiten durch Aufbringen von mehreren Texturen in einem Rendering-Durchlauf
  - Z.B. Lightmaps



A. Nischwitz et al. „Computergrafik und Bildverarbeitung“, Band 1: Computergrafik“

Multitexturing

- Mathematisch: Kombination von zwei oder mehr Texturen durch Interpolation + Multiplikation/Addition
- In OpenGL: üblicherweise 8 oder 16 Texturen möglich

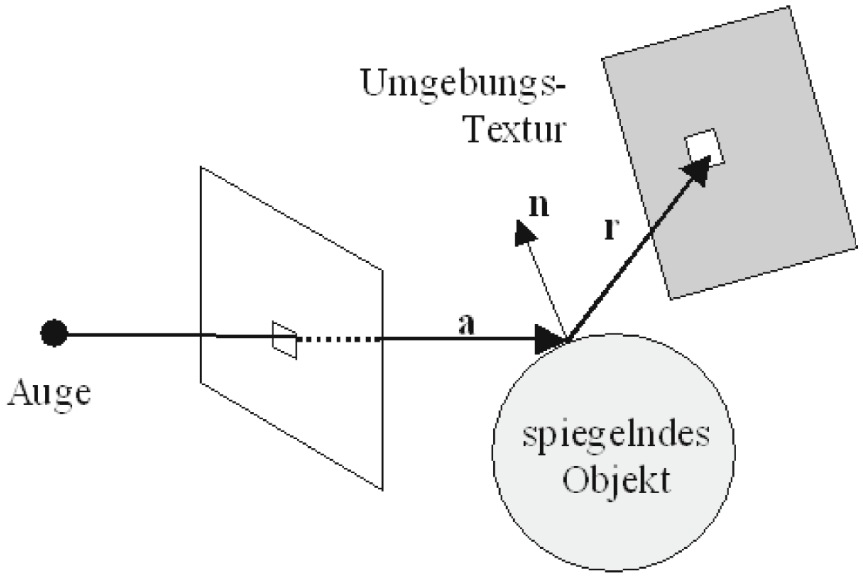


A. Nischwitz et al. „Computergrafik und Bildverarbeitung“, Band 1: Computergrafik“

8.2. ENVIRONMENT MAPPING

Environment Mapping

- Approximation von Spiegelungen einer umgebenden Textur auf Objekt



- Strahlverfolgung vom Augpunkt zu spiegelndem Objekt
- Bestimmung des Reflexionsvektors
- Texturkoordinaten in Umgebungstextur bestimmen

A. Nischwitz et al. „Computergrafik und Bildverarbeitung“, Band 1: Computergrafik“

Environment Mapping

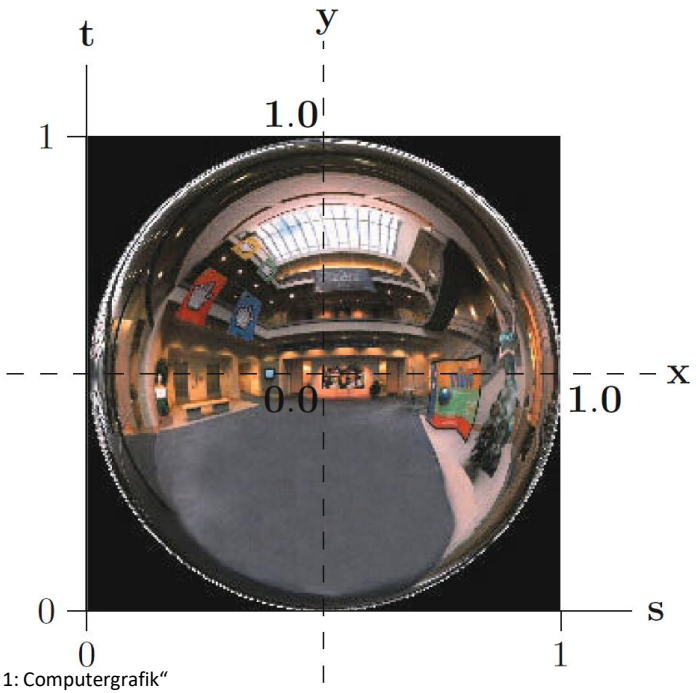
- Berechnung der Texturkoordinate:
  - Pro Vertex:
    - Berechnung des Reflexionsvektors und Texturkoordinaten nur an Vertices
    - Pixelbezogene Interpolation der Texturkoordinaten
  - Pro Pixel:
    - Berechnung des Reflexionsvektors und Texturkoordinaten für jedes Pixel

- In der interaktiven CG zwei übliche Verfahren:
  - Sphärische Texturierung (Sphere Mapping)
  - Kubische Texturierung (Cube Mapping)

Environment Mapping: Sphere Mapping

- Grundidee: Textur entspricht…
  - … orthografischer Projektion einer ideal verspiegelten Kugel
  - … Spiegelung der Kugel-Umgebung in der Kugel

- Kugel deckt kreisförmigen Bereich mit Mittelpunkt $(0{,}5 \,/\, 0{,}5)$ und Radius $0{,}5$ einer quadratischen Textur ab



A. Nischwitz et al. „Computergrafik und Bildverarbeitung“, Band 1: Computergrafik“

Environment Mapping: Sphere Mapping

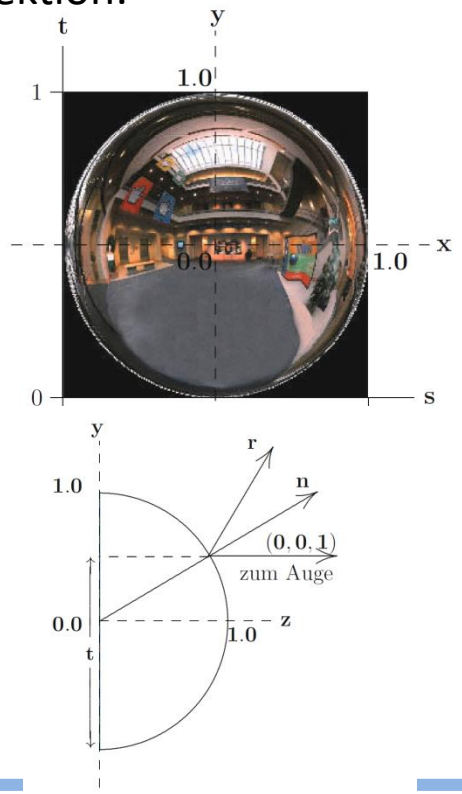
- Gesucht: Abbildung der Richtung des Reflexionsvektors auf einen Punkt in der sphärischen Textur

- Zusammenhang zwischen (planaren) Texturkoordinaten (s, t) und Punkt (x, y, z) auf der (sphärischen) Einheitskugel durch orthographische Projektion:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 2s - 1 \\ 2t - 1 \\ \sqrt{1 - x^2 - y^2} \end{pmatrix}$$

- Einheitsnormalenvektor: für Einheitskugel durch die Koordinaten (x, y, z) gegeben:

$$n = \begin{pmatrix} n_x \\ n_y \\ n_z \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 2s - 1 \\ 2t - 1 \\ \sqrt{1 - x^2 - y^2} \end{pmatrix}$$



A. Nischwitz et al. „Computergrafik und Bildverarbeitung“, Band 1: Computergrafik“

Environment Mapping: Sphere Mapping

Zusammenhang zwischen Reflexionsvektor und Normale der Objekt-Oberfläche:

- Reflexionsvektor r berechnet sich mit Hilfe des Reflexionsgesetzes:

$$r = a - 2(a \cdot n) \cdot n$$

  mit a = Vektor von Augpunkt zu Oberflächenpunkt

- a verläuft bei orthographischer Projektion entlang der negativen z-Achse: $a = (0,0,-1)$. Eingesetzt ergibt sich:

$$\begin{pmatrix} r_x \\ r_y \\ r_z \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix} - 2\left(\begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix} \cdot \begin{pmatrix} n_x \\ n_y \\ n_z \end{pmatrix}\right) \cdot \begin{pmatrix} n_x \\ n_y \\ n_z \end{pmatrix}$$

$$= \begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix} + 2n_z \begin{pmatrix} n_x \\ n_y \\ n_z \end{pmatrix}$$

- Aufgelöst nach dem Normalvektor und normiert ergibt sich:

$$n = \begin{pmatrix} n_x \\ n_y \\ n_z \end{pmatrix} = \frac{1}{\sqrt{{r_x}^2 + {r_y}^2 + (r_z+1)^2}} \begin{pmatrix} r_x \\ r_y \\ r_z + 1 \end{pmatrix}$$

Environment Mapping: Sphere Mapping

- Setzt man nun beide Normalen gleich und löst nach s bzw. t auf, ergeben sich die gesuchten Texturkoordinaten in Abhängigkeit vom Reflexionsvektor r (Koordinatentransformation):

$$\begin{pmatrix} 2s-1 \\ 2t-1 \\ \sqrt{1-x^2-y^2} \end{pmatrix} = \frac{1}{\sqrt{{r_x}^2+{r_y}^2+(r_z+1)^2}} \begin{pmatrix} r_x \\ r_y \\ r_z+1 \end{pmatrix}$$

$$s = \frac{r_x}{2\sqrt{{r_x}^2+{r_y}^2+(r_z+1)^2}} + \frac{1}{2}$$

$$t = \frac{r_y}{2\sqrt{{r_x}^2+{r_y}^2+(r_z+1)^2}} + \frac{1}{2}$$

- OpenGL übernimmt diese Berechnung bei Verwendung der automatischen Texturkoordinatengenerierung:

```
glTexGeni(GL_S, GL_TEXTURE_GEN_MODE, GL_SPHERE_MAP);
```

Environment Mapping: Sphere Mapping

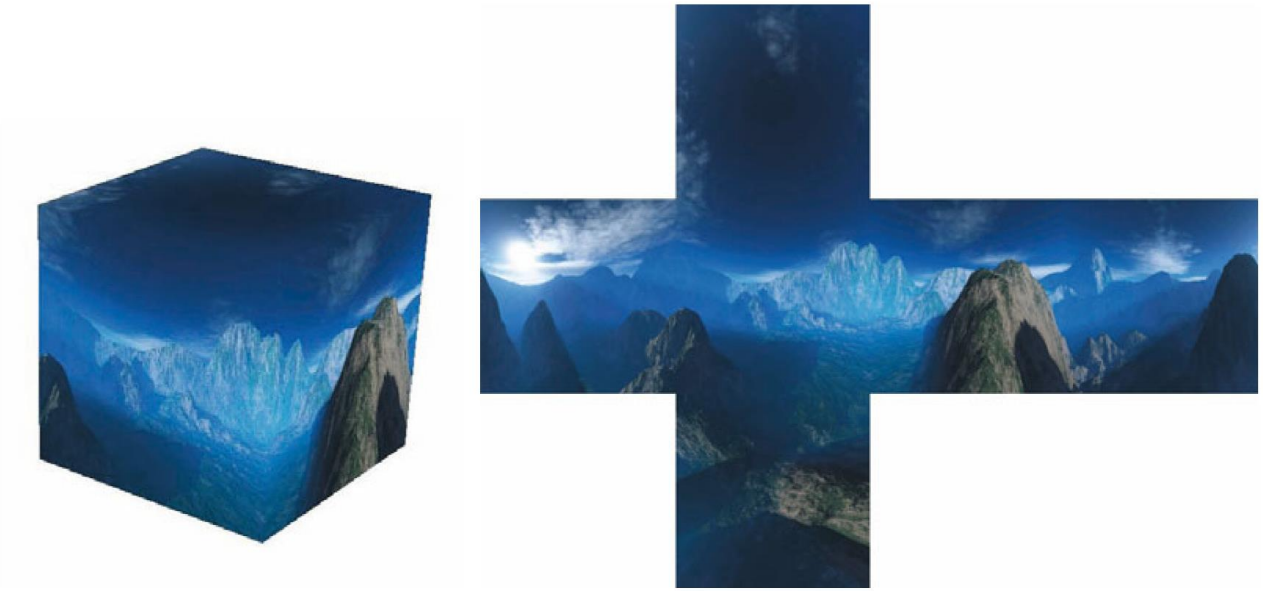
- Beeindruckende Ergebnisse bei vergleichsweise geringem Rechenaufwand

- Einschränkungen:
  - Probleme bei konkaven Objekten: keine Eigenspiegelung möglich!
  - Gilt nur für einen Blickwinkel

A. Nischwitz et al. „Computergrafik und Bildverarbeitung“, Band 1: Computergrafik“

Environment Mapping: Cube Mapping

- Umgebungstextur auf 6 Flächen eines Kubus
  - Gewinnung durch Fotografie/Rendern mit Öffnungswinkel 90° in alle Richtungen
  - Nahtloser Übergang an den Kanten des Kubus
- Zu texturierendes Objekt befindet sich in der Mitte des Kubus

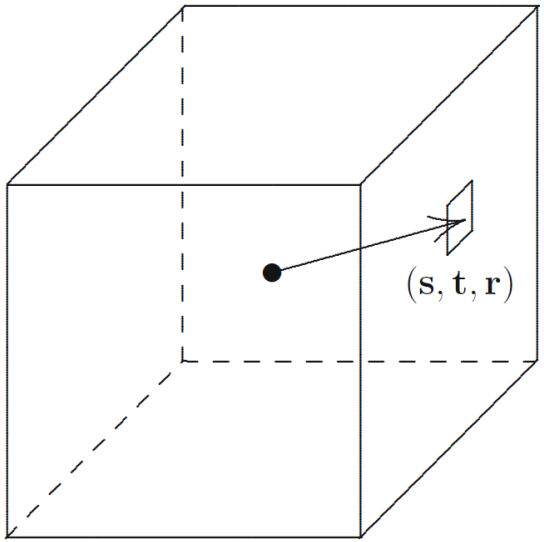


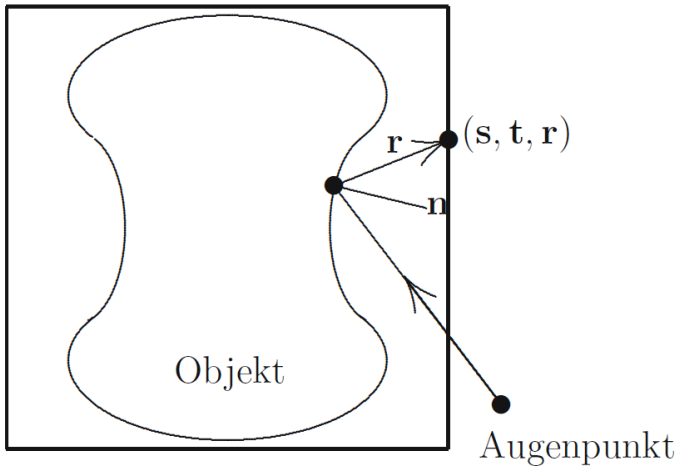
A. Nischwitz et al. „Computergrafik und Bildverarbeitung“, Band 1: Computergrafik“

Environment Mapping: Cube Mapping

- Bestimmung der Texturkoordinaten:
  - Reflexionsvektor dient als Richtungsvektor im Mittelpunkt des Kubus
  - Koordinate des Reflexionsvektors mit größtem Absolutwert wählt Fläche des Kubus
  - Verbleibende Koordinaten werden bestimmt durch (Beispiel für Fläche y):

$$x = \frac{1}{2}\left(\frac{r_x}{r_y}\right) + 0{,}5 = s \qquad z = \frac{1}{2}\left(\frac{r_z}{r_y}\right) + 0{,}5 = t$$





A. Nischwitz et al. „Computergrafik und Bildverarbeitung“, Band 1: Computergrafik“

Environment Mapping: Cube Mapping

- Bei Drehung des Augpunktes um das Objekt: inversen rotatorischen Anteil der Augpunkttransformation auf automatisch generierte Texturkoordinaten anwenden (=Drehung des Texturkubus)
- Skybox: Texturierter Kubus der gesamte Szene umschließt

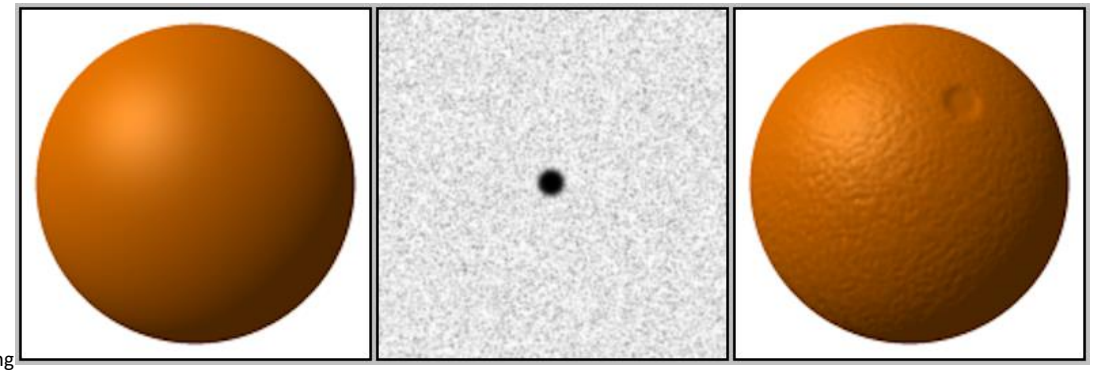
- Vorteile Cube Mapping:
  - Blickwinkelunabhängig
  - Texturen einfacher zu generieren
  - Geringere Verzerrungen

A. Nischwitz et al. „Computergrafik und Bildverarbeitung“, Band 1: Computergrafik“

8.3. BUMP MAPPING

Bump Mapping

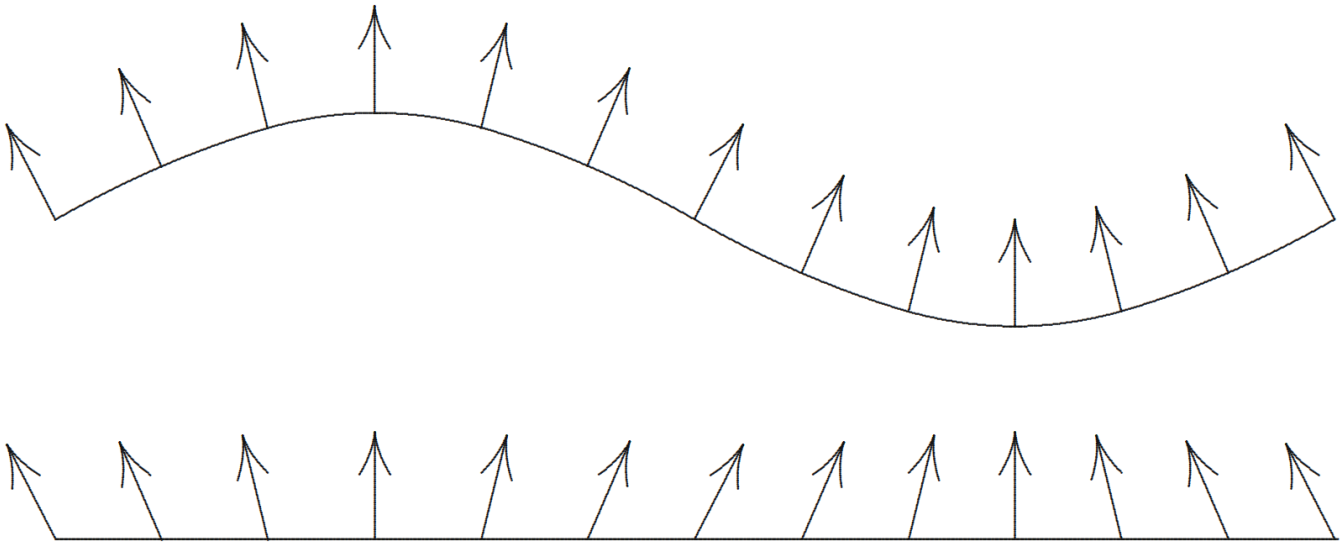
- Nachbildung einer rauen/reliefartigen Oberfläche ohne Änderung der Objektgeometrie
- Mit bisherigen Verfahren: Auftrag eine Fototextur
  - Nur für statische Szenen verwendbar, da Fototextur nur aus einem Blickwinkel aufgenommen wurde
- Abhilfe schafft das Bump Mapping (J.F. Blinn, 1978)
- Lokale Veränderung der Oberflächennormalen zur Nachahmung eines Reliefs + Pixelbezogene Beleuchtungsrechnung (Phong-Shading)
  - Bump Map: Werte geben Modifikation der Normalen an
  - Normal Map: Gibt modifizierte Normalen für jedes Pixel direkt an



https://de.wikipedia.org/wiki/Bumpmapping

Bump Mapping

- Grundprinzip ähnlich dem (Phong-)Shading:
  - Shading: Interpolation der Normalen an Vertices/Pixeln zur virtuellen Krümmung einer Oberfläche (Ziel: glatte Übergänge)



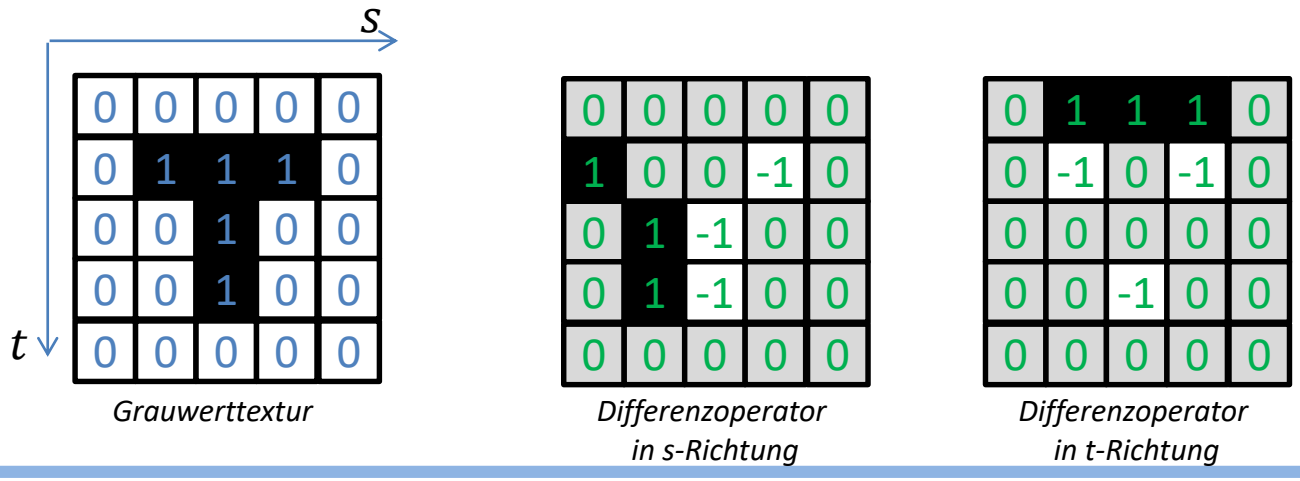
  - Bump Mapping: Modifikation der Normalen pro Pixel zur virtuellen Oberflächenkrümmung mit hoher Frequenz

A. Nischwitz et al. „Computergrafik und Bildverarbeitung“, Band 1: Computergrafik“

Bump Mapping für ebene Rechtecke

- Voraussetzung:
  - Vorliegen einer Grauwert-Textur (oder Generierung einer Grauwerttextur aus RGB-Textur durch geeignete Operation, z.B. Mittelung)
- Vorgehen zur Erstellung einer Normal Map
  1. Berechnung der Gradienten (Ableitung) der Grauwert-Textur in Richtung der s- und der t-Texturkoordinaten für jedes Texel → z-Komponente eines Gradientenvektors

z.B. durch Differenzoperator: $g_s(s,t) = \begin{pmatrix} 1 \\ 0 \\ g(s+1,t) - g(s,t) \end{pmatrix}$, $g_t(s,t) = \begin{pmatrix} 0 \\ 1 \\ g(s,t+1) - g(s,t) \end{pmatrix}$



Bump Mapping für ebene Rechtecke

2. Berechnen des Kreuzproduktes der beiden Gradientenvektoren g_s und g_t um einen Normalenvektor zu erhalten

$$n = \begin{pmatrix} 1 \\ 0 \\ g(s+1,t) - g(s,t) \end{pmatrix} \times \begin{pmatrix} 0 \\ 1 \\ g(s,t+1) - g(s,t) \end{pmatrix} = \begin{pmatrix} g(s,t) - g(s+1,t) \\ g(s,t) - g(s,t+1) \\ 1 \end{pmatrix}$$

3. Normieren des Normalenvektors → Einheitsnormalenvektor (Wertebereich [-1 1]), Länge 1

$$n = \frac{1}{\sqrt{\big(g(s,t) - g(s+1,t)\big)^2 + \big(g(s,t) - g(s,t+1)\big)^2 + 1}} \begin{pmatrix} g(s,t) - g(s+1,t) \\ g(s,t) - g(s,t+1) \\ 1 \end{pmatrix}$$

4. Transformieren des Einheitsnormalenvektors in den Wertebereich $[0,1]$ zur Kodierung der x-,y-,z-Koordinate in einer RGB-Textur

$$\boldsymbol{n} = \frac{1}{2}(n+1)$$

Bump Mapping für ebene Rechtecke

Originaltextur

Rot-Kanal
$\boldsymbol{n}_x$-Komponente

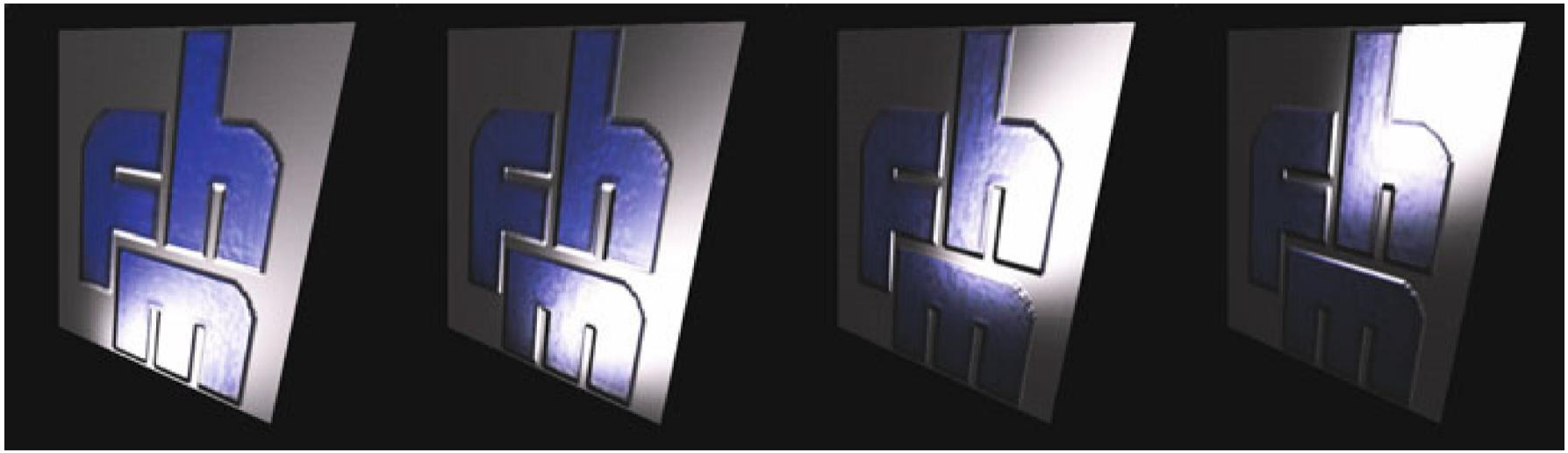
Grün-Kanal
$\boldsymbol{n}_y$-Komponente

Blau-Kanal
$\boldsymbol{n}_z$-Komponente

A. Nischwitz et al. „Computergrafik und Bildverarbeitung“, Band 1: Computergrafik“

Bump Mapping für ebene Rechtecke

- Vorgehen zum Mapping:
  - Erstellte RGB-Textur + Phong-Shading nutzen
  - Statt linearer Interpolation der Normalen an jedem Pixel (=Phong Shading) können die Normalen aus der RGB-Textur direkt für die Beleuchtungsberechnung verwendet werden.



A. Nischwitz et al. „Computergrafik und Bildverarbeitung“, Band 1: Computergrafik“

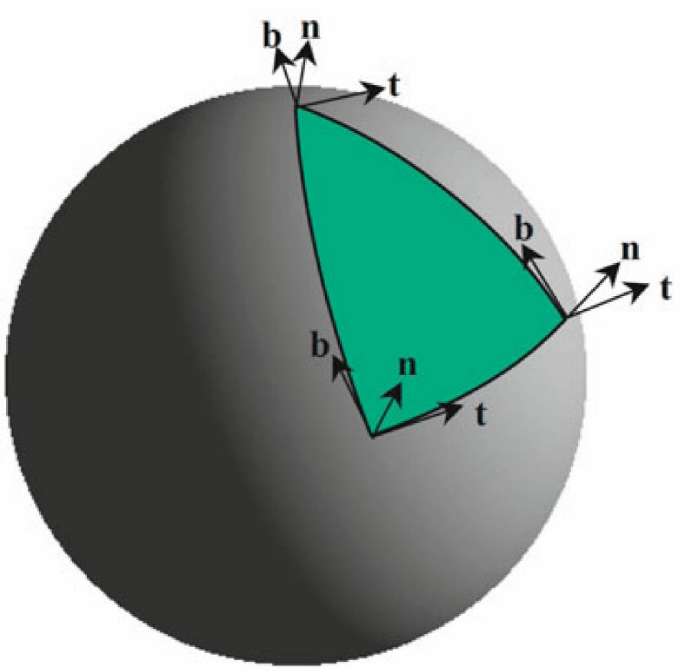
Bump Mapping

- Normalenvektoren des Rechtecks, auf die die Textur aufgebracht wird, sind einheitlich, z.B. $(0,0,1)$ für alle Pixel auf dem Rechteck.
- Bei Drehung, Verschiebung werden die Bump Mapping Normalen entsprechend mit der Transformationsmatrix gedreht bzw. verschoben

- Gekrümmte Objekte:
  - Bump Mapping Normalen können nicht verwendet werden, da sie für eine ebene Fläche (= ursprüngliche Textur) bestimmt wurden.
  - Gekrümmte Struktur aufwendig, nicht allgemein wiederverwendbar

Bump Mapping für gekrümmte Oberflächen

- Abhilfe schafft eine Koordinatensystemtransformation:
  - Für jede (interpolierte) Oberflächennormale n ist eine Tangentialebene definiert: aufgespannt durch Tangentialvektor t und Binormalenvektor b.
  - t, b und n definieren ein oberflächenlokales Koordinatensystem (=orthonormale Basis)



A. Nischwitz et al. „Computergrafik und Bildverarbeitung“, Band 1: Computergrafik“

Bump Mapping für gekrümmte Oberflächen

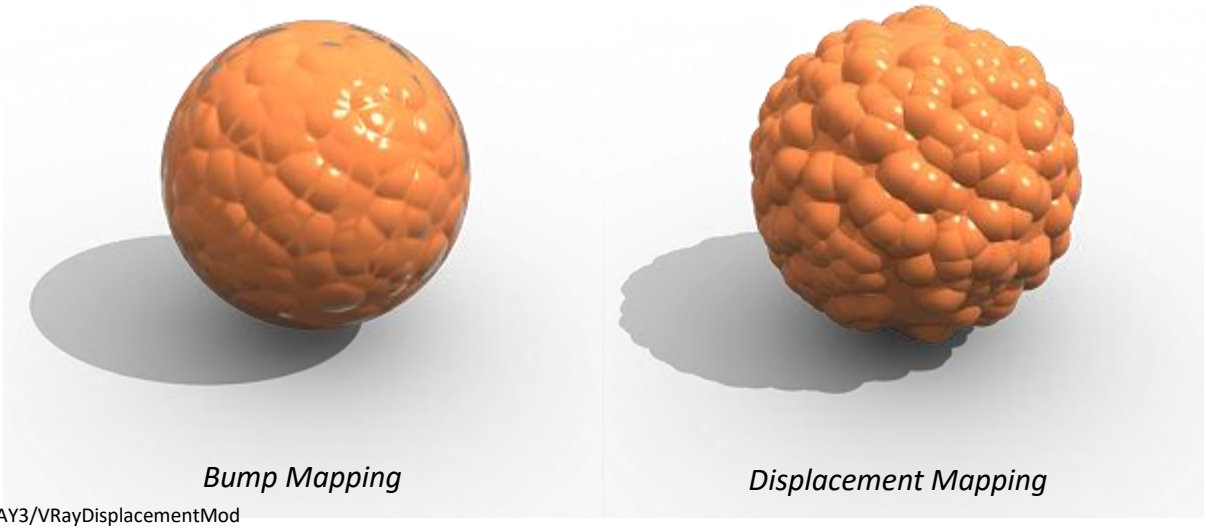
- Multiplikation mit der TBN-Matrix basierend auf der orthonormalen Basis transformiert den Normalenvektor aus der Normal-Map in das Weltkoordinatensystem:

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} t_x & b_x & n_x \\ t_y & b_y & n_y \\ t_z & b_z & n_z \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

- Tangentenvektor t sollte konsistent gewählt werden um Ecken in der Textur zu vermeiden.
- t wird üblicherweise so gewählt dass er der s-Achse der Textur folgt.
- Binormalenvektor b steht senkrecht auf t und n → Kreuzprodukt

Displacement Mapping

- Verschiebung der Oberflächenpunkte eines Objektes auf Basis der Höhenkarte entlang der Oberflächennormalen
  - Senkrecht zur Oberfläche
  - Verschieben der Punkte verändert die Normalen der neuen Oberfläche
  - In Abhängigkeit von der Feinheit des Polygonnetzes kann eine Verfeinerung notwendig werden (= Tesselation)
- Vorteil gegenüber Bump Mapping:
  - Tatsächliche Änderung der Geometrie



Bump Mapping *Displacement Mapping*

http://docs.chaosgroup.com/display/VRAY3/VRayDisplacementMod

ZUSAMMENFASSUNG

Zusammenfassung

- Texturen sind ein einfaches Mittel um Fotorealismus zu erlangen
- Texture Mapping: „Fototapete“
  - Spezifikation der Textur
  - Festlegung wie die Textur auf jedes Pixel aufgetragen wird
    - Texturfilter, Mipmapping, Texture Wraps
    - Mischung von Textur und Beleuchtungsfarbe
  - Zuordnung von Texturkoordinaten zu Vertices
  - Multitexturing
- Environment-/Shadow-Mapping: Approximation globaler Beleuchtung
  - Approximation von Spiegelungen einer umgebenden Textur auf Objekt
  - Sphere Mapping oder Cube Mapping
- Bump Mapping: Reliefartige Darstellung
  - Lokale Veränderung der Oberflächennormalen zur Nachahmung eines Reliefs + Pixelbezogene Beleuchtungsrechnung (Phong)

Übungsfragen Kapitel 8

- Was ist Mipmapping?
- Was sind Texture Wraps?
- Warum ist bei der perspektivischen Projektion eine Perspektivkorrektur für das Texture Mapping notwendig?
- Erläutern Sie wie aus einer Grauwert-Textur eine Normal Map erstellt werden kann.
- Nennen und beschreiben Sie kurz drei O-Mapping-Varianten.
- Erläutern Sie kurz den Unterschied zwischen Bump Mapping und Displacement Mapping.