

# Computergrafik WS2016/2017

---

Manuel Lang

7. März 2017

## INHALTSVERZEICHNIS

<b>1 Farbe, Darstellung und Perzeption</b>	<b>4</b>
1.1 Transferfunktion . . . . .	4
1.2 Dynamikumfang . . . . .	4
1.3 Gamma-Korrektur . . . . .	4
1.4 Alpha-Kanal . . . . .	4
1.5 Graßmannsche Gesetze . . . . .	4
1.6 Farbräume . . . . .	5
1.7 Weber-Fechner-Gesetz . . . . .	8
1.8 Nyquist-Shannon-Abtasttheorem . . . . .	8
1.9 Aliasing-Effekt . . . . .	8
<b>2 Raytracing</b>	<b>8</b>
2.1 Grundidee . . . . .	8
2.2 Baryzentrische Koordinaten . . . . .	9
2.3 Erzeugen von Sichtstrahlen . . . . .	9
2.4 Schnittpunktberechnung . . . . .	9
2.5 Schattierung/Beleuchtungsberechnung . . . . .	10
2.6 Sekundärstrahlen für Spiegelung und Transmission . . . . .	11
2.7 Recap Raytracing . . . . .	11
2.8 Aliasing . . . . .	11
2.9 Distributed Raytracing . . . . .	12
<b>3 Transformationen und homogene Koordinaten</b>	<b>13</b>
3.1 2D Transformationen . . . . .	13
3.2 3D Transformationen . . . . .	13

<b>4 Texture Mapping</b>	<b>14</b>
4.1 Grundidee . . . . .	14
4.2 Parametrisierung von 2D-Texturen . . . . .	14
4.3 Zylindrische Parametrisierung . . . . .	14
4.4 Texture Wrapping . . . . .	14
4.5 Textur-Filterung . . . . .	15
4.6 Texturierungstechniken . . . . .	16
4.7 Environment Map . . . . .	17
4.8 Shadow Mapping . . . . .	18
4.9 Fazit . . . . .	18
<b>5 Räumliche Datenstrukturen</b>	<b>19</b>
5.1 Bounding Volume-Hierarchie . . . . .	19
5.2 Reguläre und adaptive Gitter . . . . .	20
5.3 Oktalbaum/Octree . . . . .	20
5.4 Vergleich Gitter vs Octree . . . . .	20
5.5 BSP-Baum und kD-Baum . . . . .	21
5.6 Surface Area Heuristics . . . . .	22
5.7 Fazit . . . . .	23
<b>6 Rasterisierung, Clipping und Projektionstransformationen</b>	<b>23</b>
6.1 Tiefenpuffer . . . . .	23
6.2 Clipping . . . . .	24
6.3 Sutherland-Hodgeman Algorithmus . . . . .	24
6.4 Frustum . . . . .	24
6.5 Rasterisierung vs. Raytracing . . . . .	24
<b>7 OpenGL und Grafik-Hardware</b>	<b>25</b>
7.1 Gouraud-Shading . . . . .	25
7.2 Phong-Shading . . . . .	25
7.3 Backface Culling . . . . .	25
7.4 Stencil-Puffer . . . . .	25
7.5 Geometry Shader . . . . .	25
7.6 Tiefentest . . . . .	26
7.7 Blending . . . . .	26
<b>8 Prozedurale Modelle, Content Creation, Texturen &amp; Geometrie</b>	<b>26</b>
8.1 Textursynthese . . . . .	26
8.2 Noise Funktion nach Ken Perlin (1985) . . . . .	27
8.3 Turbulenz . . . . .	27
8.4 Prozedurale Texturen . . . . .	27
8.5 Hypertextures (Perlin 1989) . . . . .	28
8.6 Distanzfelder . . . . .	28
8.7 Lindenmayer-Systeme (L-Systeme) . . . . .	28
8.8 Fazit . . . . .	29
<b>9 Kurven und Flächen</b>	<b>30</b>
9.1 Bézierkurven . . . . .	30
9.2 Bernstein Polynome . . . . .	30
9.3 de Casteljau-Algorithmus . . . . .	30

9.4 Bézierssplines . . . . .	31
------------------------------	----

# 1 FARBE, DARSTELLUNG UND PERZEPTION

## 1.1 TRANSFERFUNKTION

Beschreibung der Abbildung Wert -> Helligkeit durch eine Transferfunktion mit den Werten  $[0, N]$ :  $f : [0, N] \rightarrow [I_{min}, I_{max}]$

## 1.2 DYNAMIKUMFANG

Der Dynamikumfang beschreibt den erreichbaren Kontrast eines Wiedergabegrätes (Bildschirm, Beamer):  $R_d = \frac{I_{max} + k}{I_{min} + k}$ , wobei  $k$  das Umgebungslicht ist und  $I_{max}$  die maximale bzw.  $I_{min}$  die minimale Helligkeit des Displays.

## 1.3 GAMMA-KORREKTUR

- Ein Monitor bildet einen Pixelwert  $n$  mit  $N$  Schritten auf die Intensität  $I(n)$  ab:  $I(n) \propto (n/N)^\gamma$
- Der Gamma-Wert  $\gamma$  charakterisiert das Display.
- Das Verhalten des Displays muss vor der Darstellung mit einer Gamma-Korrektur kompensiert werden.

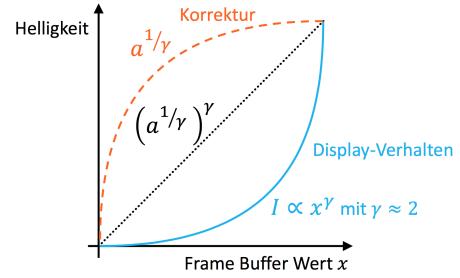


Abbildung 1.1: Gamma-Korrektur

## 1.4 ALPHA-KANAL

Oftmals werden Bilder mit 32 Bit/Pixel kodiert, beispielsweise im RGBA-Format, das neben den Farbwerten zusätzlich 8 Bit für die Opazität  $\alpha$  bereitstellt. Die Opazität ist das Gegen teil von Transparenz, wird im Frame Buffer der Grafikkarte, in PNGs und Texturen verwendet und ist essentiell für Bildbearbeitung/Manipulation, Matting/Blue-Screen-Techniken, Texturen in der Computergrafik, ...

## 1.5 GRASSMANNSCHE GESETZE

- jeder Farbeindruck kann mit 3 Grundgrößen beschrieben werden
- Superpositionsprinzip: Intensität einer additiv gemischten Farbe entspricht Summe der Intensitäten der Ausgangsfarben

## 1.6 FARBRÄUME

- Farbraum: Menge der Farben, die mit einem bestimmten Modell beschrieben werden können
- Farbmodell: mathematisches Modell mit dem Farben durch Wertetupel beschrieben werden können
- Tristimuluswerte beschreiben eine Farbe in einem bestimmten Farbraum eines Farbmodells

### 1.6.1 RGB

Biologisch und technisch motivierte Farbdarstellung mit den 3 Primärfarben rot, grün und blau. Additiver Farbraum ( $rR+gG+bB$ ) mit r, g und b als Tristimuluswerte.

### 1.6.2 CMY(K)

Subtraktiver Farbraum, der dual zu RGB funktioniert, d. h.  $(C,M,K) = (1,1,1) - (R,G,B)$ . Jede Primärfarbe absorbiert einen Teil des Spektrums. Verwendung für Tinte, Farbstoffe und Pigmente. Beim Drucken wird zur Tinteneinsparung meist noch eine 4. Key-Farbe (schwarz) verwendet -> CMYK.

### 1.6.3 HSV

- Farbton (Hue)
- Sättigung (Saturation)
- Helligkeit (Value)
- weder additiv noch subtraktiv

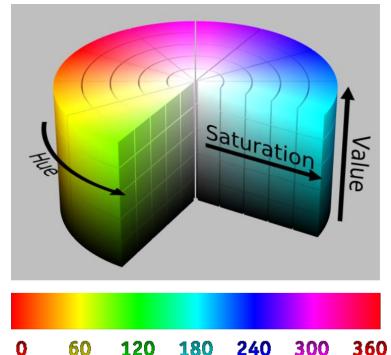


Abbildung 1.2: HSV Farbraum

### 1.6.4 CIE COLOR MATCHING

- zwei Farben wurden in einem Experiment auf einen Bildschirm projiziert
  - Referenzfarbe wurde vorgegeben
  - Vergleichsfarbe durch Mischung dreier Primärfarbe dargestellt
  - Proband sollte die Intensitäten einstellen bis beide gleich erschienen

- es konnten nicht alle Farben reproduziert werden!
  - in diesem Fall wurde eine der Primärfarben zur Testfarbe addiert
  - Farbeinstellung mit den verbleibenden Primärfarben
  - die Primärfarbe wurde dann als negativer Vergleichswert angepasst
- durch die Reproduktion der Spektralfarben mit den vorgegebenen RGB-Primärfarben erhält man die sog. **Color Matching Funktionen**
- „Wie muss ich die Lampe einstellen, um eine Spektralfarbe zu reproduzieren?“
- Wie berechnet man eine metamere Farbe (in demselben RGB-Farbraum mit Primärfarben 435.8nm, 546.1nm, 700nm) zu geg. Spektrum  $P(\lambda)$ ?
  - Tristimuluswerte: Antwort auf die 3 Color Matching Funktionen  $r = \int \bar{r}(\lambda)P(\lambda)d\lambda$   
 $g = \int \bar{g}(\lambda)P(\lambda)d\lambda$   $b = \bar{b}(\lambda)P(\lambda)d\lambda$
  - metamere Farben erhalten wir durch Wahl der Intensität der Primärfarben gemäß der 3 Antworten
- Problem
  - negative Vergleichswerte
  - einige Spektralfarben/Spektren sind nicht realisierbar durch eine Kombination aus 3 Primärfarben
  - RGB ist kein perfekter Farbraum, aber eben technisch realisierbar

#### 1.6.5 XYZ COLOR SPACE (CIE 1931)

- Ziel: Farbraum zur standardisierten Konversion zwischen Farträumen
  - Beschreibung aller wahrnehmbaren Farben
  - Farbraum mit rein positiven Color Matching Funktionen
  - lineare Abbildung XYZ <=> RGB
  - Y-Komponente so gewählt, dass sie der Luminanz entspricht
- Festlegung der Color Matching Funktionen bedeutet imaginäre übersaturierte Primärfarben, die nicht physikalisch realisierbar sind
  - $\bar{y}(\lambda)$  = Luminanz
  - $\bar{z}(\lambda) \approx$  Empfindlichkeit S-Rezeptor
  - $\bar{x}(\lambda) \approx$  Linearkombination der Empflichkeitskurven so, dass  $\bar{x}(\lambda) > 0$
  - die Primärfarben vorstellbar als rot, grün und blau
- XYZ sind die Tristimuluswerte für den „CIE Standard Observer“
  - $X = \int \bar{x}(\lambda)P(\lambda)d\lambda$
  - $Y = \int \bar{y}(\lambda)P(\lambda)d\lambda$
  - $Z = \int \bar{z}(\lambda)P(\lambda)d\lambda$

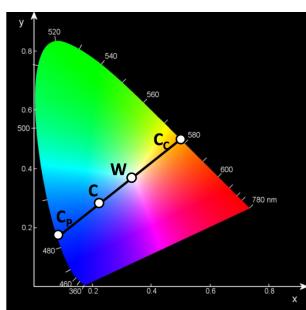
### 1.6.6 CHROMATIZITÄT (VOM CIE XYZ ZUM CIE XY Y FARBRAUM)

Darstellung aller sichtbaren Farben

- Beobachtung: alle  $k(X, Y, Z)(k > 0)$  repräsentieren dieselbe Farbe, nur mit unterschiedlicher Intensität
- Normalisierung auf die  $X + Y + Z = 1$  Ebene
- anschließend: Projektion auf die XY-Ebene (z weglassen)
- $xy$ -Diagramm enthält nach wie vor alle Farbtöne und -sättigungen
- teile Information in Helligkeit  $Y$  und Farbe (Chromatizität)  $xy$  auf
  - mit  $x = \frac{X}{X+Y+Z}$ ,  $y = \frac{Y}{X+Y+Z}$ , nicht gespeichert:  $z = \frac{Z}{X+Y+Z} = 1 - x - y$
- wird Farbe durch  $x, y$  vorgegeben, muss noch die Helligkeit  $Y$  angegeben werden -> dann können Tristimuluswerte aus CIE xyY bestimmt werden.
  - $X = \frac{Y}{y}x$  und  $Z = \frac{Y}{y}(1 - x - y)$

### 1.6.7 CHROMATIZITÄTSIDIAGRAMM

- enthält alle sichtbaren Farben (**Gamut** der menschlichen Wahrnehmung)
- Weißpunkt  $W(x = y = z = \frac{1}{3})$  (entspricht etwa Sonnenlicht)
- Spektralfarben befinden sich entlang der Randkurve und entsprechen monochromatischem Licht
- die seltsame Form ergibt sich aus den Empfinglichkeitskurven
- Purpurlinie: Menge von gesättigten Farbvalenzen, die ein Mensch wahrnehmen kann
- Farben auf der Strecke zwischen 2 Punkten können durch additives Mischen der Farben an den Endpunkten erreicht werden
- die reine Farbe  $C_P$  zu einer Farbe  $C$  findet man durch Verlängern der Linie vom Weißpunkt durch  $C$
- Komplementärfarbe  $C_C$ : Linie durch den Weißpunkt zum gegenüberliegenden Rand



## 1.7 WEBER-FECHNER-GESETZ

Die subjektiv empfundene Stärke von Sinneseindrücken ist proportional zum Logarithmus der Intensität des physikalischen Reizes. Es gilt:  $\frac{\Delta L_{IND}}{L} = const \approx 1\% \text{ bis } 2\%$

## 1.8 NYQUIST-SHANNON-ABTASTTHEOREM

Ein kontinuierliches, bandbegrenztes Signal mit einer max. Frequenz  $f_{max}$  muss mit einer Frequenz größer als  $2f_{max}$  abgetastet werden, damit aus dem diskreten Signal das Ursprungssignal rekonstruiert werden kann.

## 1.9 ALIASING-EFFEKT

Fehler beim Abtasten von Signalen. Im rekonstruierten Signal treten Frequenzen auf, die im Original nicht enthalten sind. Als Lösungsansätze kann das Signal gefiltert werden, was im allgemeinen Fall allerdings nicht möglich ist oder eine Überabtastung verwendet und anschließend gefiltert.

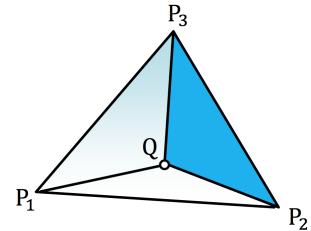
# 2 RAYTRACING

## 2.1 GRUNDIDEE

- Verfahren zur Bildsynthese basierend auf Gesetzen der geometrischen Optik
- Ausbreitung des Lichts wird von der Kamera aus zurückverfolgt
- betrachte jeden Pixel des Bildes: verfolge einen Strahl durch diesen Pixel und finde das nächste Objekt entlang des Strahls
- berechne Schattierung des Objektes
- setze Strahlverfolgung fort, wenn Flächeln spiegeln oder transmittieren
- Aufbau eines Raytracing-Programms:
  - Erzeugung der Sichtstrahlen (ray generation)
  - Schnittberechnung (ray intersection)
  - Schattierung bzw. Beleuchtungsberechnung (shading)

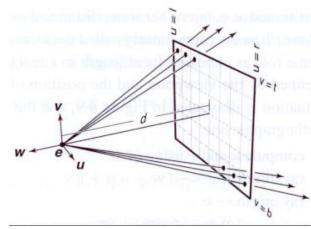
## 2.2 BARYZENTRISCHE KOORDINATEN

Wenn ein Punkt  $Q$  in der Form  $Q = \lambda_1 P_1 + \lambda_2 P_2 + \dots + \lambda_k P_k$  mit  $\lambda_1 + \lambda_2 + \dots + \lambda_k = 1$  (Affinkombination) dargestellt werden kann, so bezeichnet man  $(\lambda_1, \lambda_2, \dots, \lambda_k)$  als die baryzentrischen Koordinaten von  $Q$  bzgl. der Basispunkte  $P_1, \dots, P_k$



## 2.3 ERZEUGEN VON SICHTSTRAHLEN

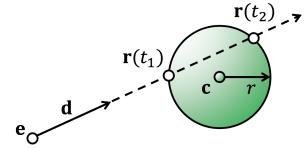
- virtuelle Kamera definiert durch Position  $e$ , Zielpunkt  $z$ , „Up-Vektor“  $u$  und negative Blickrichtung  $w$ .
- Bildebene definiert durch Abstand zur Kamera  $d$ , linker und rechter Rand  $l$  und  $r$ , unterer Rand  $b$  und oberer Rand  $t$ .
- Erzeugen von Sichtstrahlen: Vektor  $s$  zu Punkten auf der Bildebene:  $s = u \cdot u + v \cdot v - d \cdot w$  mit  $u \in [l, r]$  und  $v \in [b, t]$
- Strahlgleichung  $r(t) = e + ts$ , i.d.R  $r(t) = e + td$ , mit  $d = s/|s|$



## 2.4 SCHNITTPUNKTBERECHNUNG

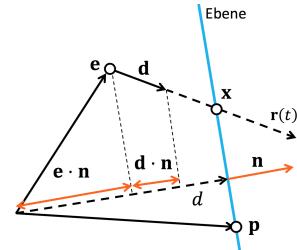
### 2.4.1 STRAHL-KUGEL-SCHNITT

- Strahlgleichung  $r(t) = e + td$
- Kugel (implizite Darstellung)  $|x - c|^2 - r^2 = 0$
- $(e - c) \cdot (e - c) - r^2 + 2(td \cdot (e - c)) + (td) \cdot (td) = 0$
- Lösung durch Mitternachtsformel



## 2.4.2 STRAHL-EBENE-SCHNITT

- Strahlgleichung  $r(t) = e + td$
- Ebene (implizite Repr., HNF)  $x \cdot n - d = 0$  mit Normalenvektor  $n$  und Abstand vom Ursprung  $d$
- Lösung durch Einsetzen:  $(e + td) \cdot n - d = 0, t = \frac{d - e \cdot n}{d \cdot n}$



## 2.4.3 STRAHL-DREIECK-SCHNITT

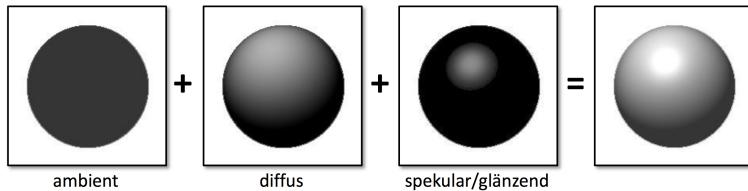
Direkte Berechnung über baryzentrische Koordinaten.

## 2.5 SCHATTIERUNG/BELEUCHTUNGSBERECHNUNG

### 2.5.1 PHONG-BELEUCHTUNGSMODELL

Das Phong-Beleuchtungsmodell ist phänomenologisches Modell, das die Reflexion mit drei Komponenten realisiert:

- ambient: indirekte Beleuchtung, Licht von anderen Oberflächen
- diffus: nach dem Lambertschen Gesetz
- spekular: imperfekte Spiegelung



**Lamertsche (ideal-diffuse) Reflexion:** gleiche Helligkeit unabhängig von der Betrachterrichtung:  $I_d = k_d \cdot I_L \cdot \cos\theta = k_d \cdot I_L \cdot (N \cdot L)$

**perfekte Spiegelung** findet nur in Richtung  $R_L$  statt:  $R_L = 2N \cdot (N \cdot L) - L$

**spekulare Reflexion:**  $I_s = k_s \cdot I_L \cdot \cos^n \alpha = k_s \cdot I_L \cdot R_L \cdot V)^n$

$$I = I_a + I_d + I_s = k_a \cdot I_L + k_d \cdot I_L \cdot (N \cdot L) + k_s \cdot I_L \cdot (R_L \cdot V)^n$$

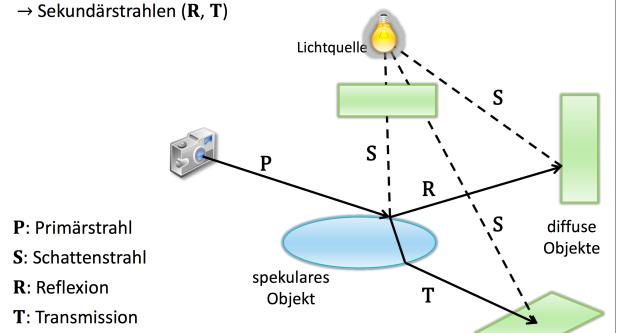
Beachte:

- $k_a, k_d, k_s$  sind wellenabhängig, ebenso die Lichtintensität  $I_L$
- diffuse Reflexionen haben meist die Farbe der Oberfläche
- spekulare Reflexionen haben die Farbe der Oberfläche, wenn es sich um Metalle handelt, sonst typischerweise die Farbe der Lichtquelle
- Nur positive Skalarprodukte sind interessant, daher wird  $\max(0, N \cdot L)$  verwendet.

## 2.6 SEKUNDÄRSTRÄHLEN FÜR SPIEGELUNG UND TRANSMISSION

- Spiegelung der Szene auf reflektierenden Oberflächen
- Behandle Reflexionsstrahl (fast so) wie einen Sichtstrahl
- addiere Farbe des Reflexionssstrahls zur Farbe am Ausgangspunkt

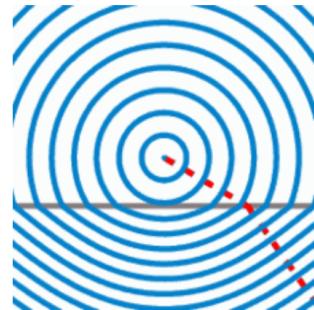
Rekursion um Spiegelung und Lichtbrechung darzustellen  
→ Sekundärstrahlen (R, T)



77

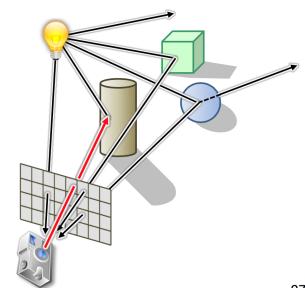
### 2.6.1 SNELLSCHES BRECHUNGSGESETZ

- beschreibt die Richtungsänderung einer Welle (Licht) beim Übergang von einem Medium in eines mit anderer Brechzahl
- Brechung bei Übergang ins optisch dichtere Medium ( $\eta_t > \eta_i$ ) zum Lot hin
- Fresnel-Effekt: Verteilung der Strahldichte



## 2.7 RECAP RAYTRACING

- verfolge den Weg des Lichts, das den Betrachter erreicht, zurück
- Lichttransport basiert auf geometrischer Optik
- bildbasiertes Verfahren (Image-Order Rendering)

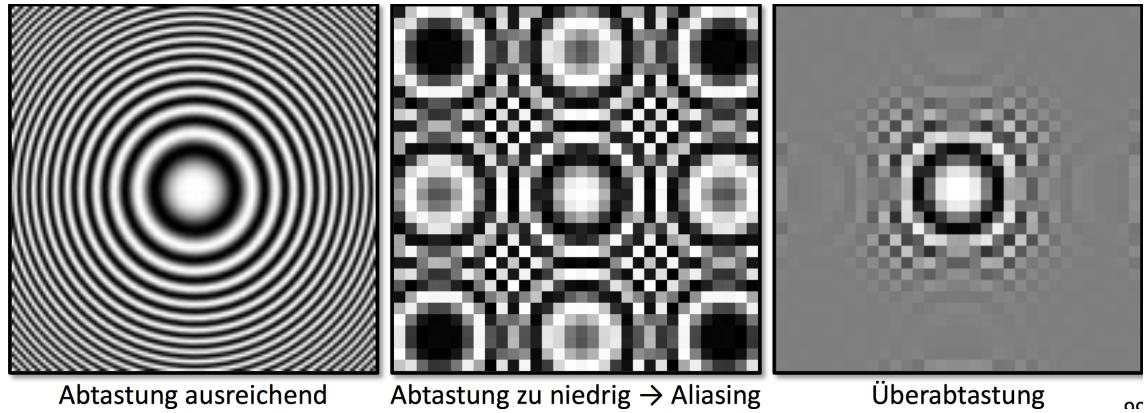


77

## 2.8 ALIASING

Aliasing-Effekte (ungewollte Geometrien) entstehen durch falsche Abtastung von Signalen (Bildern). Als Lösung existieren verschiedene Anti-Aliasing-Strategien, meist **Überabtastung**:

- Uniformes Supersampling: statt Abtastung eines Punktes innerhalb eines Pixels (ein Sample) tasten wir  $k^2$ -mal in äquidistanten Intervallen ab
- Adaptives Supersampling: Pixel wird nur dann unterteilt, wenn Differenz benachbarter Samples zu groß ist.
- Stochastisches Sampling: zufällige Samples mit Stratifikation (Stratified Sampling) unterteilt Pixel in ein Gitter und wählt Zufallspunkt pro Zelle, reduziert Aliasing, aber führt zu mehr rauschen.



## 2.9 DISTRIBUTED RAYTRACING

Beim Whitted-Style Raytracing Verfahren sehen die Bilder zu makellos aus: perfekte Spiegelung und Transmission, harte Schattenkanten, unendliche Schärfentiefe, etc.

### 3 TRANSFORMATIONEN UND HOMOGENE KOORDINATEN

**Lineare Abbildungen** haben zwei Eigenschaften:

- additiv:  $T(p + q) = T(p) + T(q)$
- homogen:  $T(ap) = aT(p)$

Eine Translation ist nicht linear. Sei  $T(p) = p + t$ .  $T(ap) = ap + t \neq aT(p) = a(p + t)$

#### 3.1 2D TRANSFORMATIONEN

##### 3.1.1 SKALIERUNG

$$scale(s_x, s_y) = \begin{pmatrix} s_x & 0 \\ 0 & s_y \end{pmatrix}$$

##### 3.1.2 SCHERUNG

$$shear_x(s) = \begin{pmatrix} 1 & s \\ 0 & 1 \end{pmatrix}, shear_y(s) = \begin{pmatrix} 1 & 0 \\ s & 1 \end{pmatrix}$$

##### 3.1.3 SPIEGELUNG

$$\text{Spiegelung an der y-Achse: } \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}, \text{ Spiegelung an der x-Achse: } \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

##### 3.1.4 ROTATIONEN

$$rotate(\phi) = R(\phi) = \begin{pmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{pmatrix}$$

#### 3.2 3D TRANSFORMATIONEN

$$scale(s_x, s_y, s_z) = \begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, shear_z(d_x, d_y) = \begin{pmatrix} 1 & 0 & d_x & 0 \\ 0 & 1 & d_y & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$R_x(\phi) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\phi) & -\sin(\phi) & 0 \\ 0 & \sin(\phi) & \cos(\phi) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, R_y(\phi) = \begin{pmatrix} \cos(\phi) & 0 & \sin(\phi) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\phi) & 0 & \cos(\phi) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$R_z(\phi) = \begin{pmatrix} \cos(\phi) & -\sin(\phi) & 0 & 0 \\ \sin(\phi) & \cos(\phi) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

## 4 TEXTURE MAPPING

### 4.1 GRUNDIDEE

- realistischere Aussehen einer Oberfläche kann man durch „Feinstrukturierung pro Pixel“ erreichen
- kombiniere Geometrie mit Bildern
- dadurch erreicht man einfach hohe Detailgrade, auch mit größeren Netzen

### 4.2 PARAMETRISIERUNG VON 2D-TEXTUREN

#### 4.2.1 PLANARE PROJEKTION

- definiere Ebene, z. B. durch einen Punkt  $p$  und zwei aufspannende Vektoren  $s, t$
- Texturkoordinate eines Oberflächenpunkts  $x$ :  $s = (x - p) \cdot s$ ,  $t = (x - p) \cdot t$
- der Bereich  $[0, 1]^2$  repräsentiert die ganze Textur -> Unabhängigkeit von der tatsächlichen Auflösung

#### 4.2.2 SPHÄRISCHE PARAMETRISIERUNG

- Darstellung der Objektkoordinaten in Polarkoordinaten  $(r, \phi, \theta)$
- Texturkoordinaten  $\begin{pmatrix} s0 \\ t0 \end{pmatrix} := \begin{pmatrix} \phi/2\pi \\ \theta/2\pi \end{pmatrix}$

### 4.3 ZYLINDRISCHE PARAMETRISIERUNG

- Darstellung der Objektkoordinaten in Zylinderkoordinaten  $(r, \phi, y)$
- Texturkoordinaten  $\begin{pmatrix} s0 \\ t0 \end{pmatrix} := \begin{pmatrix} \phi/2\pi \\ y/2\pi \end{pmatrix}$

#### 4.3.1 WÜRFEL PARAMETRISIERUNG

- mehrere mögliche Abbildungen:
  - Projektion von 6 Ebenen (Bild)
  - lese die Würfeltextur dort aus, wo ein Strahl vom Objektmittelpunkt durch einen Oberflächenpunkt einen umgebenden Würfel schneidet
- andere Abbildungen Objekt <-> Hilfskörper natürlich ebenfalls möglich

### 4.4 TEXTURE WRAPPING

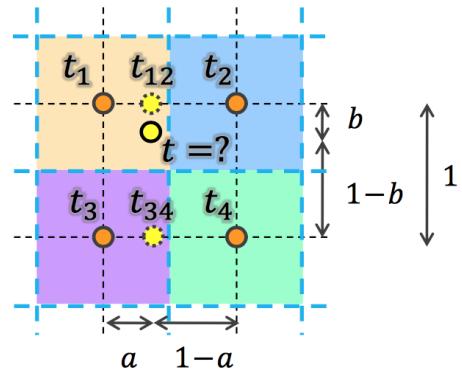
- Repeat/Wrapping: Fortsetzen/Kacheln einer Textur über  $[0, 1]^2$  hinaus
- Adressierung wird für jede Dimension separat gewählt

## 4.5 TEXTUR-FILTERUNG

- Abbildung weniger Texel auf viele Pixel
- Nearest Neighbor
  - verwendet Farbe des nächstliegenden Texels
- Bilineare Interpolation
  - Interpolation der 4 nächsten Texel
  - Nutzung der „Nachkommastellen“
  - dadurch wird das Textursignal geglättet

### 4.5.1 BILINEARE INTERPOLATION

- Interpolation für Punkt aus seinen Nachbarpixeln
- $a, (1-a), b$  und  $(1-b)$  sind die relativen Abstände zu den Textelmittelpunkten entlang der  $s$ - und  $t$ -Achsen
- bilineare Interpolation ist nicht linear, sondern quadratisch



### 4.5.2 MIP-MAPPING

- einfache Vorfilterung von Texturen
- speichere rekursive Texturen mit  $\frac{1}{4}$  Größe (Halbierung entlang jeder Achse) -> AuflösungsPyramide mit nur 33% mehr Speicherbedarf
- meist Mittelung über je 2x2 Texel (kein optimaler Tiefpass)
- beachte: gleichzeitige Filterung und Auflösungsreduktion
- wähle Texturauflösung (Mip-Map Stufe  $n$ ) so, dass
  - Texelgröße( $n$ )  $\leq$  Größe Pixelfootprint auf Textur < Texelgröße( $n+1$ )
  - $n = 0$  entspricht der höchsten Auflösungsstufe
- der endgültige Farbwert wird durch **trilineare Interpolation** zwischen zwei Mip-Map-Stufen (8 nächstliegende Texel) bestimmt
  - bilinear auf Stufe  $n$ , bilinear auf Stufe  $n+1$
  - anschließend linear zwischen diesen Farben

#### 4.5.3 ANISOTROPE TEXTURFILTERUNG

- Mip-Mapping resultiert oft in sehr verwaschenen Details
- der Abdruck (Footprint) eines Pixels im Texturraum ist oft eher länglich
  - die Vorfilterung bei Mip-Mapping ist isotrop (gleichförmig in  $s$  und  $t$ )
- wie kann man bei der Vorfilterung darauf Rücksicht nehmen?
  - RIPmaps (enthält Vorfilterung unabhängig in jeder Achse, 4x Speicherbedarf, löst das Problem nur teilweise und behandelt nur Anisotropie entlang einer der Achsen)
  - anisotrope Filterung in der Praxis: Abtasten des Bereichs durch eine geschickte Kombination von Abtastung verschiedener Mip-Map-Stufen mit typischerweise mehreren Abtaststellen (z.B. 8 oder 16)
  - Summed Area Tables (nicht wichtig?)

#### 4.5.4 BESTIMMUNG DES FOOTPRINTS

- einfachste Möglichkeit: schicke einen Primärstrahl durch die Ecken eines Pixels oder betrachte einen 2x2 Pixelblock und bestimme die 4 Texturkoordinaten
- Differenz der Texturkoordinaten liefert Größe/Form des Footprints
  - Mip-Mapping: bestimme max(Breite, Höhe) und daraus  $n$
  - Rip-Mapping, SAT: betrachte Breite und Höhe
- Textur-Lookup an berechneter Stelle mit entsprechender Filterung

### 4.6 TEXTURIERUNGSTECHNIKEN

#### 4.6.1 DIFFUSE TEXTUR

- Kontrolle der Eigenfarbe eines Materials
- Bsp. Phong-Beleuchtungsmodell:  $k_d$  aus Textur
- $I = k_a \cdot I_L + k_d \cdot I_L \cdot (N \cdot L) + k_s \cdot I_L \cdot (R \cdot V)^n$

#### 4.6.2 BUMP ODER NORMAL MAPPING

- Variation der Normale einer Oberfläche berechnet aus der Veränderung einer Basisfläche durch eine Bump Map oder Displacement Map
- verändertes  $N$  aus Textur (und dementsprechend auch anderes  $R$ )
- $I = k_a \cdot I_L + k_d \cdot I_L \cdot (N \cdot L) + k_s \cdot I_L \cdot (\mathbf{R} \cdot V)^n$
- Fläche bleibt aber geometrisch flach und nur die Normale variieren
- Berechnung meist im Tangentenraum

#### 4.6.3 GLOSS-MAP/GLOSS-TEXTUR

- Kontrolle der Stärke und Streuung der spekularen Reflexion
- Bsp. Phong-Beleuchtungsmodell:  $k_s$  und  $n$  aus Textur
- $I = k_a \cdot I_L + k_d \cdot I_L \cdot (N \cdot L) + k_s \cdot I_L \cdot (R \cdot V)^n$

#### 4.6.4 DISPLACEMENT MAPPING

- Verschiebung der Oberfläche und Änderung der Normale
- mehr als nur Änderung der Beleuchtungsberechnung
- z.B. durch Geometrie-Tesellierung und Verschiebung (GPU-unterstützt)

#### 4.6.5 INVERSE DISPLACEMENT MAPPING

- besonders schnelle Approximation geeignet um „Bodendetail“ darzustellen
- Geometrie wird nicht wirklich erzeugt: führt Schnittpunktberechnung stattdessen im Texturraum durch

#### 4.6.6 AMBIENT OCCLUSION

- Kontrolle der ambienten Beleuchtung (Umgebungslicht)
- $k_a$  aus Textur, meistens wird auch der diffuse Term modifiziert
- $k_a \cdot I_L + k_d \cdot I_L \cdot (N \cdot L) + k_s \cdot I_L \cdot (R \cdot V)^n$

#### 4.6.7 TEXTUR-ATLAS

- spezielle (bijektive) Parametrisierung
  - jedem Oberflächenpunkt entspricht eine Stelle in der Textur
  - keine Stelle der Textur taucht an mehr als einem Oberflächenpunkt auf
- Erstellung aufwändig per Hand oder automatisch

### 4.7 ENVIRONMENT MAP

Eine Environment-Map ist eine Textur zur Darstellung der Umgebung. Bei Environment-Maps nimmt man an, dass der Betrachter weit genug von der Umgebung entfernt ist, sodass die Position keine Rolle spielt und ausschließlich die Blickrichtung wichtig ist. Übliche Parametrisierungen von Environment-Maps sind:

- Cube Maps
  - + ist bei korrekter Filterung nahtlos

- + keine Singularität am Rand
- Sphere Maps
  - - Singularität am Rand
  - + mit Kamera, Chromkugel und Photoshop kann sie recht einfach aufgenommen / erstellt werden
- LatLong Maps
  - - die Pole werden ungleichmäßig abgetastet

#### 4.8 SHADOW MAPPING

- Eine Textur speichert für viele Richtungen wie weit die Oberflächen von der Lichtquelle entfernt sind (mehr in Kapitel 7: OpenGL).

#### 4.9 FAZIT

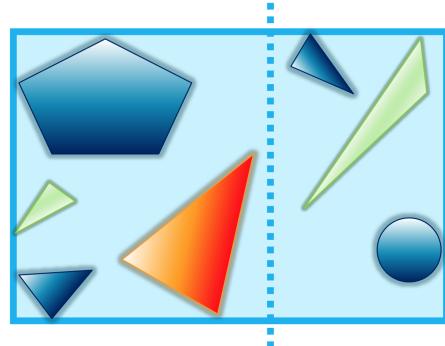
- Texture Mapping ist eine fundamentale Technik in der Computergrafik
- erlaubt Detailreichtum, der mit einer Geometrie nicht möglich wäre
- vielseitiger Einsatz
  - Farbtexturen
  - Normalen
  - Displacement
  - Environment Mapping
  - Image-Based Lighting
  - Shadow Mapping
  - Visualisierung
  - Lookup-Tabellen für aufwändige Berechnungen

## 5 RÄUMLICHE DATENSTRUKTUREN

### 5.1 BOUNDING VOLUME-HIERARCHIE

#### 5.1.1 AUFBAU DER HIERARCHIE

- bestimme die Bounding Box der Objekte
- teile die Objekte jeweils wieder in zwei Gruppen auf
- verfahre so rekursiv weiter (z.B. bis nur noch  $m$  Objekte in jeder Gruppe enthalten sind)



#### 5.1.2 BESCHLEUNIGUNG DER SCHNITTBERECHNUNG MIT BVHS

- überprüfe auf Schnitt mit der Wurzel (= AABB der gesamten Szene)
- dan steige rekursiv ab (beginnend mit dem näheren Kindknoten)
- teste auf Schnitt mit Objekten im Kindknoten (steige evtl. weiter ab)
- prüfe anschließend noch die weiter entfernten Knoten (sofern ein näherer Schnittpunkt existieren kann)

#### 5.1.3 FAZIT BVHS

Vorteile:

- Konstruktion und Traversierung ist einfach
- resultiert in einem Binärbaum mit fixer, geringer Verzweigung
- Komplexität für den Aufbau  $O(n \log n)$  wenn in der Mitte der AABBs unterteilt wird,  $O(\log^2 n)$  wenn die Objekte in einer AABB in zwei (etwa) gleich große Gruppen geteilt werden und eine  $O(n \log n)$  Sortierung verwendet wird.

Herausforderungen

- Finden einer guten Unterteilung ist schwierig
- ungeschickte Verteilung kann zu schlechter Aufteilung führen

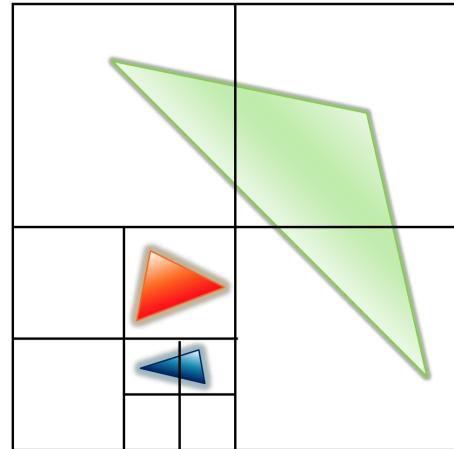
## 5.2 REGULÄRE UND ADAPTIVE GITTER

### 5.2.1 GRUNDIDEE, PRINZIP

- Unterteilung des Raumes in Zellen gleicher Größe und Forum
- Eintrag der Objekte in Zellen, die von ihnen geschnitten werden
- Traversierung der vom Strahl getroffenen Zellen und Schnittberechnung mit den enthaltenen Objekten

## 5.3 OKTALBAUM/OCTREE

- beginne wieder mit der Bounding Box für die ganze Szene
- rekursive 1-zu-8 Unterteilung jeweils in der Mitte in jeder Richtung, falls noch zu viele Primitive in einer Zelle sind
- adaptive Unterteilung erlaubt große Schritte im leeren Raum
- feine Unterteilung nur dort, wo Geometrie ist



## 5.4 VERGLEICH GITTER VS OCTREE

### Gitter

- einfacher Algorithmus
- schlecht bei ungleich verteilter Geometrie: kein effizientes Überspringen von leerem Raum
- adaptiv nur durch verschachtelte Gitter
- ähnlicher Aufwand für ähnliche Strahlen

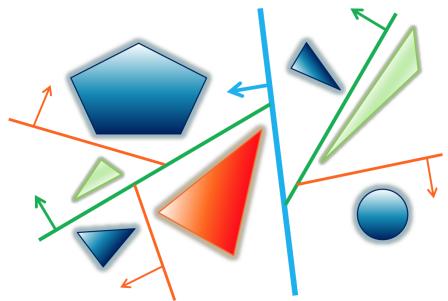
### Octree

- Adaptivität mit feiner Granularität
- häufige Vertikalbewegung
- Traversierungskosten für zwei ähnliche Strahlen können stark variieren

## 5.5 BSP-BAUM UND KD-BAUM

### 5.5.1 GRUNDIDEE

- Erweiterung von Binärbäumen auf  $k$  Dimensionen
- verwende Ebenen um den Baum rekursiv zu unterteilen
- Unterschied: BSP-Bäume verwenden beliebig orientierte Ebenen, kD-Bäume nur Ebenen die senkrecht zur  $x$ -,  $y$ - und  $z$ -Achse sind



### 5.5.2 KONSTRUKTION

- Konstruktion findet rekursiv, ähnlich wie BVH, statt
  - initialisiere Wurzelknoten: enthält alle Objekte/Primitive der Szene
  - unterteile den Wurzelknoten rekursiv
    - \* bis ein Knoten nur noch eine vorgegebene maximale Anzahl an Primitiven enthält
    - \* oder eine maximale Rekursionstiefe erreicht ist
  - unterteile den Raum und ordne die Primitive einem linken oder rechten Kindknoten zu

### 5.5.3 EIGENSCHAFTEN

- echte Raumunterteilung: Knoten überlappen sich nicht (vgl. BVH)
- gute Anpassung an Geometrie möglich (v. a. bei BSP-Bäumen)
- die Blattknoten speichern die Primitive bzw. Verweise/Indizes darauf
- innere Knoten speichern die Split-Ebenen
  - BSP-Baum: Normale der Ebene und Abstand zum Ursprung
  - kD-Baum: die zur Ebene senkrechte Achse und die Position entlang der Achse
  - Zeiger auf Kindknoten
- kd-Bäume sind einfacher zu konstruieren und werden daher häufiger eingesetzt

#### 5.5.4 TRAVERSIERUNG

- Strahl  $r(t) = e + td$ , mit  $t_{min} \leq t \leq t_{max}$
- Ziel: Traversierung der Knoten von vorne nach hinten
- verwende Stack, um die Knoten für die Bearbeitung zu halten
- zu Beginn enthält der Stack den Wurzelknoten (0)
- schneide Strahl mit der Split-Ebene (entnehme den ersten Knoten vom Stack)
  - der Schnitt liegt bei  $t'$  mit  $t_{min} \leq t' \leq t_{max}$
  - fahre mit beiden Kindern rekursiv fort
- wenn Blattknoten erreicht, dann teste Schnittpunkte mit Primitiven
- gebe Schnittpunkte zurück, wenn innerhalb  $[t_{min}, t'']$
- Vorteil echter Raumunterteilung: kein möglicher Schnitt weiter hinten

#### 5.6 SURFACE AREA HEURISTICS

- Kostenfunktion für das Unterteilen eines kD-Baums/BVH Knotens p  

$$C = C_T + \frac{AS(B_l)}{SA(B_p)} + |P_l|C_i + \frac{SA(B_r)}{SA(B_p)}|P_r|C_i$$
- $B_l, B_r$  und  $B_p$  sind die Bounding Boxes der Primitive im linken und rechten Kindknoten, bzw. des betrachteten Knotens p
- $C_i$  sind die Kosten eines Strahl-Primitiv-Schnitttests
- $C_T$  sind die Kosten der Traversierung eines kD-Baum- bzw. BVH-Knoten
- Ziel: finde die Unterteilung, die die Kostenfunktion minimiert

##### 5.6.1 OPTIMIERUNG/BESTIMMUNG DER UNTERTEILUNG

- aufwändige Suche nach der besten Unterteilung (niedrigste SAH-Kosten), da der Suchraum groß sein kann
- Variante 1: approximative Konstruktion (nicht elegant)
  - erzeuge einige Unterteilungskandidaten innerhalb der Bounding Box  $B_p$
  - berechne Kostenfunktion für jeden Kandidaten
- Variante 2: Konstruktion mit inkrementeller Berechnung
  - sortiere die  $n$  Primitive nach x-, y- und z-Achse
  - teste jede der möglichen  $n - 1$  Aufteilungen

- inkrementelle Berechnung der AABBs und damit vergleichsweise effiziente Berechnung der SAH möglich
- Konstruktion mit SAH verursacht nicht vernachlässigbare Kosten
- gut konstruierte kD-/BSP-Bäume bzw. BVHs können um ein Vielfaches schneller sein, als schlecht konstruierte

## 5.7 FAZIT

- Kombination verschiedener Beschleunigungstechniken
  - oft SIMD-Optimierung zusammen mit BVH oder kD-Baum
    - \* BVHs mit AABBs: schnelle Erzeugung, gute Hierarchie
    - \* kD-Baum: etwas aufwändiger zu konstruieren, aber typischerweise etwas besser als BVH, kombiniert mit AABBs pro Knoten
    - \* BSP-Baum: aufwändig zu konstruieren, oft nur ein bisschen besser als ein gut konstruierter kD-Baum
    - \* Gitter: eher selten, aber Aufbau auf (Grafik-)Hardware einfacher
    - \* hierarchische Gitter und Octrees: eher eingesetzt für Simulationen
- weitere Verwendung (nicht nur) in der Computergrafik
  - Bounding Volumes und BVH: Culling in der Echtzeit-Grafik
  - kD-/BSP-Bäume: Sortierung, Suche, ...
  - Octrees: Speicherung von räumlichen Daten, z.B. 3D-Texturen, Kollisionserkennung (Physiksimulation, Robotik)

# 6 RASTERISIERUNG, CLIPPING UND PROJEKTIONSTRANSFORMATIONEN

## 6.1 TIEFENPUFFER

- bildbasierter Ansatz: speichere für jeden Pixel Distanz zur nahsten Fläche
- Entfernung-Tiefen-Wert wird pro Vertex berechnet und interpoliert
- zusätzlich zum Farbpuffer gibt es dazu den Z-Buffer (16 bis 32 Bit/Pixel)

Nachteile:

- zusätzlicher Speicherbedarf und -bandbreite (heutzutage nicht mehr)
- begrenzte Genauigkeit und z-Aliasing
- transparente Flächen können nicht behandelt werden
- viel unnötiger Aufwand in Szenen mit hoher Tiefenkomplexität

- Tiefenkomplexität = Anzahl der Schnitte entlang eines Primärstrahls
- unnötiger Aufwand, weil verdeckte Flächen rasterisiert werden

Vorteile:

- Dreiecke können in beliebiger Reihenfolge verarbeitet werden
- Z-Buffering ist Standard in allen Rasterisierern (inkl. Grafik-Hardware)
- für die meisten der obigen Probleme existieren heute spezielle Lösungen oder Rendering-Techniken

## 6.2 CLIPPING

- Abschneiden von Linien/Polygon-Teilen die außerhalb des Bildschirms liegen
- kann wichtig für Effizienz sein
- ist unbedingt notwendig zur Vermeidung problematischer Fälle bei Projektionen

## 6.3 SUTHERLAND - HODGEMAN ALGORITHMUS

Clipping gegen eine Kante nach der andren.

## 6.4 FRUSTRUM

Ein Frustum ist ein Kegelstumpf, wobei in der Computergrafik eher ein Pyramidenstumpf gemeint ist. Das View Frustum ist der Bereich der Szene, der sichtbar ist.

## 6.5 RASTERISIERUNG VS. RAYTRACING

Rasterisierung:

- sehr effizient, Hardware-Umsetzung ist einfach
- nur für Primärstrahlen: globale Effekte nur über spezielle Techniken
- Handhabung komplexer Szenen durch räumliche Datenstrukturen
- spezielle Techniken, z. B. zur Einschränkung der Schattierungsberechnung auf sichtbare Flächen
- Anwendung: Echtzeit-Rendering

Raytracing:

- konzeptionell einfaches Verfahren
- Sekundärstrahlen, komplexe Beleuchtungseffekte sind einfach
- Handhabung komplexer Szenen durch räumliche Datenstrukturen
- Offline-Rendering, bedingt interaktives Rendering

## 7 OPENGL UND GRAFIK-HARDWARE

### 7.1 GOURAUD-SHADING

Berechne Parameter wie z.B. Farbe an den Eckpunkten; interpoliere innerhalb des Polygons.

### 7.2 PHONG-SHADING

Beleuchtungsberechnung mit interpolierter Normalen. Phong-Shading hat mit dem Phong-Beleuchtungsmodell inhaltlich nichts zu tun.

### 7.3 BACKFACE CULLING

Dreiecke, auf deren Rückseite man blickt werden üblicherweise nicht gezeichnet.

```
glEnable(GL_CULL_FACE); glCullFace(GL_BACK);
```

### 7.4 STENCIL-PUFFER

Ein Stencil-Puffer ist eine Stanzmaske, welche für jeden Pixel im Framebuffer einen 8-bit Wert speichert. Im einfachsten Fall begrenzt der Stencil-Puffer das Renderinggebiet.

### 7.5 GEOMETRY SHADER

Geometry Shader bearbeiten Primitive und können Primitive vervielfachen, entfernen oder umwandeln. Sie werden nach dem Vertex und Tessellation Shader ausgeführt. Sie können nicht beliebig viel Geometrie ausgeben.

### 7.6 SCISSOR-TEST

Der Scissor-Test dient lediglich dazu Fragmente außerhalb eines Rechtecks zu entfernen.

### 7.7 DITHERING

Fehlerdiffusion

### 7.8 BLENDING

Kombination der Farbwerte im Framebuffer (Source) mit Farben von Fragmenten (Destination) anhand Gewichten, die aus den RGBA-Werten erzeugt werden oder als konstant festgelegt werden.

Zeichen eines semitransparenten Objekts:

```
 glEnable( GL_BLEND )
 glEnable(GL_BLEND)
 glBlendFunc( GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA )
```

Die Blend Equation legt die Verknüpfungsoperation Op fest.

```
glBlendEquation( GL_FUNC_ADD ) (default)
```

Semitransparente Objekte:

```
glEnable( GL_DEPTH_TEST );
glDisable( GL_BLEND );
glUseProgram( DrawOpaqueProgram );
<drawscene> // nur Fragmente von opaken Objekte ausgeben
 glEnable( GL_BLEND );
glBlendFunc( GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA );
glDepthMask( GL_FALSE );
glUseProgram( DrawTransparentProgram );
<drawscene> // blende sortierte transparente Objekte
glDepthMask( GL_TRUE );
```

Multiplikation der Source- mit der Destination-Farbe:

```
glBlendFunc( GL_DST_COLOR, GL_ZERO )
```

Addition der beider Farbvektoren

```
glBlendFunc( GL_ONE, GL_ONE )
```

## 8 PROZEDURALE MODELLE, CONTENT CREATION, TEXTUREN & GEOMETRIE

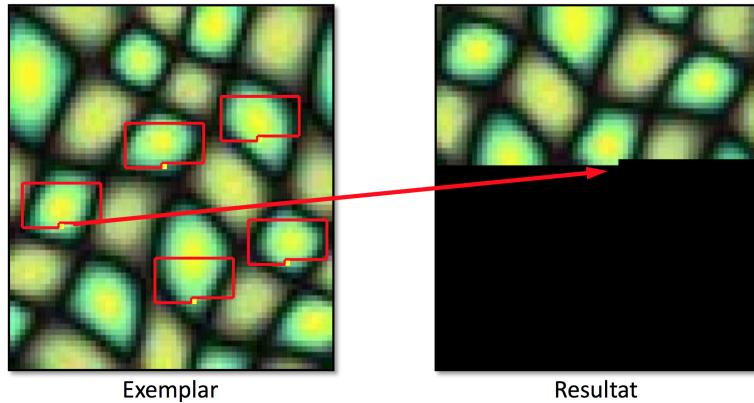
### 8.1 TEXTURSYNTHESE

- oft kleine Textur (Exemplar) vorgegeben (z. B. extrahiert aus Fotografie)
  - Tiling (Kachelung) führt zu sichtbaren Wiederholungen
- Ziel: Berechnen (Synthese) einer großen Textur die „genauso“ aussieht
  - Verfahren in dieser Vorlesung langsam: keine Auswertung zur Laufzeit
  - „gleich aussehen“ bedeutet „gleiche statistische Eigenschaften“
  - achte auch darauf genügend Variationen zu erzeugen

#### 8.1.1 PIXELBASIERTE TEXTURSYNTHESE

- Ausgabetextur wird Pixel für Pixel von links nach rechts und oben nach unten erzeugt
- betrachte Nachbarschaften des nächsten zu erzeugenden Pixels
- suche ähnliche Nachbarschaften in der Eingabetextur (Summe über alle Pixeldifferenzen klein)
- kopiere den Pixel einer der ähnlichsten Nachbarschaften

- sehr einfaches Prinzip/Algorithmus, jedoch sehr langsam



### 8.1.2 PATCHBASIERTE TEXTURSYNTHESSE

**TODO**

## 8.2 NOISE FUNKTION NACH KEN PERLIN (1985)

- Grundlage für stochastische Modellierung/prozedurale Texturen
- Basis bilden Pseudo-Zufallszahlen aus denen wir Bilder mit den gewünschten Eigenschaften erzeugen
- zweig häufig verwendete Arten von Noise-Funktionen
  - Lattice Value Noise (lattice value = Werte auf einem Gitter)
  - Lattice Gradient Noise (Gradient statt Absolutwerte, nicht behandelt)

## 8.3 TURBULENZ

$$turbulence(x) = \sum_k (\frac{1}{2})^k n(2^k \cdot x)$$

- Aufsummieren von k Oktaven, skaliert mit  $(\frac{1}{2})^k$
- höhere Oktaven erzeugen höherfrequente Anteile, deren Amplitude aber geringer wird
- Frequenzspektrum  $\frac{1}{2}$  (hier f=2), „rosa Rauschen“
- Wie kann man hohe Frequenzen entfernen? -> einfach Oktaven weglassen!

## 8.4 PROZEDURALE TEXTUREN

- algorithmische Beschreibung von Texturen
  - kompakte Repräsentation von Texturen
  - auflösungsunabhängig

- für beliebig große Flächen ohne sichtbare Wiederholungen
- parametrisierbar
- implizite Methoden
  - Aufruf des Shaders für jeden Pixel/Schnittpunkt
  - benötigt evtl. Zugriff auf benachbarte Pixel (Ableitung)
  - Filterung ist i.A. nicht trivial
- explizite Methoden
  - generiere 2D/3D-Textur vor der Verwendung
  - verwende Texture-Mapping wie bisher
  - Vorteil: Texture-Filterung ist einfach

### 8.5 HYPERTEXTURES (PERLIN 1989)

- Ziel: Erzeugung von komplexen 3D-Volumenmodellen (z.B. Feuer, Wolken)
  - Analogie zu Displacement-Mapping von Flächen: verändern von volumetrischen Gebilden
- Beschreibung eines Hypertexture-Objekts durch eine Dichtefunktion (DF) im  $\mathbb{R}^3$ 
  - innen  $D(x) = 1$ , außen  $D(x) = 0$ , Übergangsbereich  $0 < D(x) < 1$
  - komplexe Strukturen durch Noise-Funktionen und Modulation

### 8.6 DISTANZFELDER

- Distanzfunktion  $f(x), x \in \mathbb{R}^3$ , die für jeden Punkt  $x$  die kürzeste Entfernung zur Oberfläche  $f(x) = 0$  liefert
- einfache Objekte durch Distanzfunktion beschreibbar
- idealerweise ist Abstand vorzeichenbehaftet, z.B.  $f(x) < 0$  im Inneren
- wird  $f(x)$  diskret gespeichert, dann spricht man von einem **Distanzfeld**
- **Sphere Tracing:** Finden des Schnittpunktes durch Ray Marching und Ausnutzen der Distanzfunktion/-felds für die Schrittweite

### 8.7 LINDENMAYER-SYSTEME (L-SYSTEME)

- theorethisches Modell für biologische Entwicklung und Morphogenese
  - basiert auf formalen Grammatiken
  - ursprünglich entworfen für die Beschreibung der Entwicklung von mehrzelligen Organismen

- später: Erweiterung auf Pflanzen und Strukturen mit Verzweigung
- Grundidee: definiere ein komplexes Objekt durch sukzessives Ersetzen von Teilen eines einfacheren Objekts (vgl. Wachstumsprozesse)
- L-Systeme arbeiten wie Chomskys formale Grammatiken
- ein L-System ist definiert durch ein Quadrupel  $G = (V, \Sigma, S, P)$ 
  - Alphabet (Zeichen)  $V$
  - Terminalsymbole (Zeichen)  $\Sigma \subset V$
  - Startwort/-symbol  $S \in V^*$
  - Produktionsregeln  $P \subset (V^* \setminus \Sigma^*)xV^*$
  - wichtiger Unterschied: Produktionsregeln werden parallel angewendet um die biologischen Prozesse nachzubilden
- L-System alleine macht noch kein geometrisches Objekte, da eine Interpretationsvor- schift benötigt wird

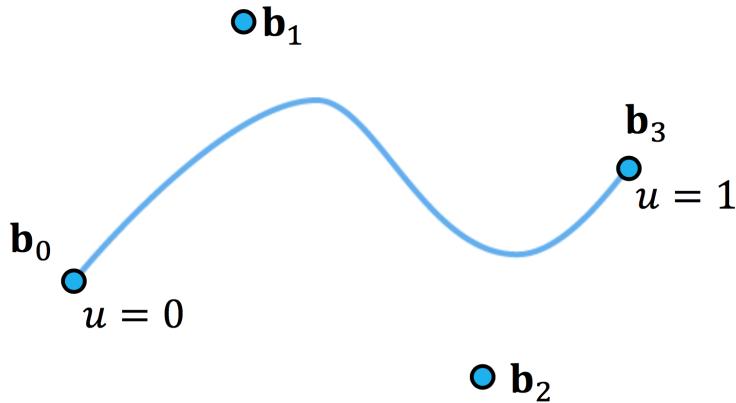
## 8.8 FAZIT

- Prozedurale Modellierung ist ein mächtiges Werkzeug
- manchmal aber schwer zu kontrollieren
- zunehmend wichtiger: die Menge des grafischen Inhalts von Spielen, Filmen etc. steigt schnell
- kompakte Beschreibung, ideale Form der „Datenkompression“

## 9 KURVEN UND FLÄCHEN

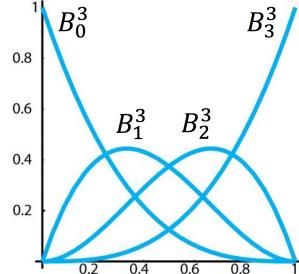
### 9.1 BÉZIERKURVEN

$$P(u) = (1-u)^3 b_0 + 3u(1-u)^2 b_1 + 3u^2(1-u) b_2 + u^3 b_3 \text{ mit } b_0 = \begin{pmatrix} x_0 \\ y_0 \end{pmatrix}, b_1 = \begin{pmatrix} x_1 \\ y_1 \end{pmatrix}, \dots$$



### 9.2 BERNSTEIN POLYNOME

- $B_0^3(u) = (1-u)^3$
- $B_1^3(u) = 3u(1-u)^2$
- $B_2^3(u) = 3u^2(1-u)$
- $B_3^3(u) = u^3$

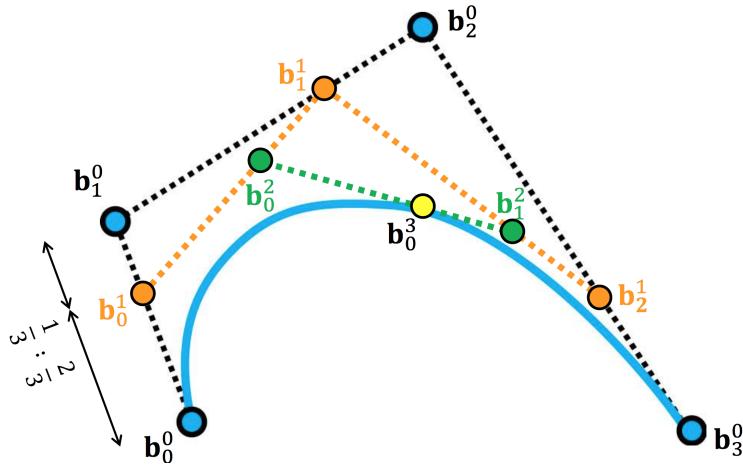


### 9.3 DE CAUSTELJAU-ALGORITHMUS

Effiziente Auswertung und Unterteilung von Bézierkurven

- möglich durch die rekursive Darstellung der Bernstein-Polynome
  - $B_i^n(u) = u \cdot B_{i-1}^{n-1}(u) + (1-u) \cdot B_i^{n-1}(u)$
  - wir haben die rekursive Darstellung symbolisch betrachtet, setzt man die tatsächlichen Kontrollpunkte ein, wertet man die Kurve direkt aus
- Algorithmus
  - $b_i^0 := b_i$
  - $b_i^1 := (1-u) \cdot b_i^0 + u \cdot b_{i+1}^0$ , usw.

- allgemein:  $b_i^j := (1-u) \cdot b_i^{j-1} + u \cdot b_{i+1}^{j-1}, i = 0, \dots, n-j$
- Berechnung von  $F(u) = \sum_{i=0}^n B_i^n(u) b_i$  durch fortgesetzte lineare Interpolation, ohne die Bernstein-Polynome direkt zu berechnen



#### 9.4 BÉZIERSPLINES

- Modellieren von komplexeren Formen mit Bézierkurven
  - verwende eine Bézierkurve von hohem Grad
    - \* u. U. numerische Probleme, aber noch wichtiger: Jeder Kontrollpunkt beeinflusst die ganze Kurve -> schwierig bei der Modellierung
  - füge mehrere Bézierkurven niedrigen Grades stückweise aneinander
- Bézier-Spline: stückweise polynomische Kurve, deren einzelne Abschnitte durch Bézierkurven beschrieben sind
  - (parametrische) Stetigkeit des Übergangs? Glattheit der Kurve?

