

Farbe, Darstellung, Perzeption

Frame Buffer = Speicher in dem Bild für Darstellung abgelegt ist
 ↳ Darstellung dann unabh. von dessen Speicherlogik/zuweisung

Rasterbild = diskrete Repr. eines Bildes → Pixel

Fehlerdiffusion (Dithering) = fehlende Farbe durch bestimmte Anordnung verfügbarer Farben nachbilden

Transferfunktion

höhere RGB-Werte = hellere Farben

→ Beschreibung von "wert → Helligkeit" durch Transferfunktion $f: [0, N] \rightarrow [l_{\min}, l_{\max}]$

- l_{\max} = maximale Displayhelligkeit
- l_{\min} = min. " "
- k_c = am Display reflektiertes Umgebungslicht → Einfluss auf Kontrast

Dynamikumfang = erreichbarer Kontrast $R_d = \frac{l_{\max} + k_c}{l_{\min} + k_c}$

• 2%. Helligkeitsunterschied wahrnehmbar → minim. Helligk.-unt. aufeinanderfolgender just noticeable difference $\frac{\Delta L_{JND}}{L} \approx 1\% - 2\%$ ↳ in dunklen Bereichen kleinere Schritte
 Untergrundhelligkeit

Quantisierung der Transferfunktion:

(1) linear: $I(n) = \frac{n}{N} l_{\max}$ → verwendet bei Berechnung von Bildern

(2) potenzfunktionsbasiert: $I(n) = (\frac{n}{N})^\gamma l_{\max}$ → Speicherung von Bildern

Gamma-Korrektur

$I(n) \propto (\frac{n}{N})^\gamma$ γ-Wert charakterisiert Display
 ↳ Abbildung des Pixel-Wertes n mit N Schritten

→ wir möchten aber lineares Verhalten (doppelter Wert, doppelt so hell)

→ Gamma-Korrektur vor Darstellung

⇒ $I(n) \propto n^\alpha$ (α = berechneter Pixelwert)
 $\sim n \propto q^{1/\gamma}$

ohne Gamma-Korrektur: Bilder zu dunkel / übersättigt

Alpha-Kanal

Bilder als 32 Bit / Pixel → RGBA : 24 Bit Farben, 8 Bit Alpha-Kanal

• α = Opazität

Licht = elektromagn. Strahlung

- Frequenz v
- Wellenlänge $\lambda = \frac{c}{v}$ (c = Lichtgeschw.) jedes λ eine Spektralfarbe
- Energie Photon $E = h v$
- Planck'sches Wirkungsquantum h

Radiometrie = Strahlungslehre, Messung der elektromagn. Strahlung
Photometrie = Einbeziehen der Empfindlichkeit des Betrachters

Spektralfarben: monochromatisches Licht (nur eine bestimmte Wellenlänge) erscheint als helle und reine Farbe

Auge - Rezeptoren

- > Zapfen: Tagsehen + trichromatisches Farbssehen
- 3 Arten: untersch. Empfindlichkeit gegenüber Lichtspektren
 - > S (7%) \approx blau
 - > M (37%) \approx grün
 - > L (56%) \approx rot

> Stäbchen: Nachsehen

- lichtempfindlicher ($1000\times$ als Zapfen)

Trichromatisches Farbssehen perzeptuelle Antwort auf Licht untersch. Wellenlänge

$$S = \int s(\lambda) P(\lambda) d\lambda \quad m = \int m(\lambda) P(\lambda) d\lambda \quad l = \int l(\lambda) P(\lambda) d\lambda$$

($P(\lambda)$ = Strahlungsleistung der Wellenlänge λ)
 \rightarrow Verhältnisse von s, m, l ergeben Farbtöne

Metamensismus: untersch. Spektren können gleich aussehen
 \hookrightarrow untersch. $P(\lambda)$ rufen selbe Antwort (s, m, l) hervor

Farbraume

Additive Farbmischung: Farbkombi durch Addieren der Spektren

RGB-Farbraum: 3 Primärfarben: rot, grün, blau
 $\rightarrow C = r \cdot R + g \cdot G + b \cdot B \quad (r, g, b) \in [0, 1]^3$
 r, g, b = Intensitätswert

Subtraktive Farbmischung: Multiplikation der Spektren

CMY (K)-Farbraum: cyan, magenta, yellow (, key: schwarz \rightarrow drucken)

C	$1 - R$
M	$1 - G$
Y	$1 - B$

 jede Primärfarbe absorbiert Teil des Spektrums

Grämannsche Gesetze

- > Farbe = 3 dim. Größe
- > Intensität einer additiv gemischten Farbe = Summe der Intensitäten der Ausgangsfarben
- > Farbtöne einer additiven Mischfarbe nur abh. vom Farbindruck der Ausgangsfarben (nicht deren getrennten Spektren) \rightarrow keine Rückschlüsse auf spektrale Zusammensetzung möglich

HSV-Farbraum

- > Farbtone (Hue), Sättigung (Saturation), Helligkeit (Value)
- > weder additiv noch subtraktiv

RGB \rightarrow HSV

$$V = \max(r, g, b)$$

$$S = \frac{\max - \min}{\max}$$

$$H \in [0^\circ, 360^\circ]$$

 $0^\circ = \text{rot}$ $120^\circ = \text{grün}$ $240^\circ = \text{blau}$

Farbmodell - mathem. Modell mit dem Farben durch Wertetypen beschreibbar
 Farbraum = Menge der Farben, die mit bestimmtem Modell beschreibbar
 Tristimuluswerte = beschreiben Farbe in bestimmtem Farbraum

Color Matching

Reproduktion der Spektralfarben mit vorgegebenen RGB-Primärfarben

=> color Matching Funktionen

- . wie berechnet man metrische Farbe mit Primärfarben zu geg. Spektrum $P(\lambda)$?
- $r = \int f_r(\lambda) P(\lambda) d\lambda$ $g = \int f_g(\lambda) P(\lambda) d\lambda$ $b = \int f_b(\lambda) P(\lambda) d\lambda$
- Wahl der Intensitäten gemäß der 3 Antworten

→ nicht alle Spektren realisierbar: RGB kein perfekter Farbraum

XYZ Colorspace

Farbraum zur standardisierten Konversion zw. Farträumen

- Beschreibung aller wahrnehmbaren Farben

- lineare Abb. $x \ y \ z \leftrightarrow RGB$

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = H \begin{bmatrix} R \\ G \\ B \end{bmatrix} \quad H = \begin{bmatrix} 0,49 & 0,31 & 0,20 \\ 0,18 & 0,81 & 0,01 \\ 0,00 & 0,01 & 0,99 \end{bmatrix}$$

XYZ-Tripel kann auf negative RGB-Werte abgebildet werden → nicht darstellbar
 $XYZ \not\propto RGB$ Primärfarben

$$\bar{y}(\lambda) = \text{Luminanz} \quad \bar{z}(\lambda) \approx \text{Empfindlichkeit S-Rezeptor}$$

$$x = \int \bar{x}(\lambda) P(\lambda) d\lambda \quad y = \int \bar{y}(\lambda) P(\lambda) d\lambda \quad z = \int \bar{z}(\lambda) P(\lambda) d\lambda$$

 $\bar{x}(\lambda) \approx \text{Lichtkomb. der Empfindlichkeitskurven sodass } \bar{x}(\lambda) > 0$ Chromatizität mit $k > 0$: kx, ky, kz beschreiben gleiche Farbe→ normalisiere auf $x+y+z=1$ Ebene + z-weglassen

$$x = \frac{x}{x+y+z} \quad y = \frac{y}{x+y+z} \quad (z = 1-x-y)$$

→ Information geteilt in Helligkeit Y und Farbe xy → Chromatizitätsdiagramm

blau: x,y klein grün: y groß rot: x groß

Chromatizitätsdiagramm:

> enthält alle sichtbaren Farben (Gamut der menschl. Wahrnehmung)

> Weißpunkt W ($x=y=z=1/3$)

> Spektralfarben entlang Randkurve, entsprechen monochrom. Licht

Farbgamut: darstellbare Farben eines Ausgabegeräts

Weber-Fechner-Gesetz

subjektiv empfundene Stärke von Sinnesindrücken ist proportional zum Logarithmus der Intensität des physikalischen Reizes

$$\frac{\Delta I}{I} = \text{const.}$$

Kontrastsensitivität

- > bei Chrominanz: bei niedrigen Frequenzen ausgeprägt, schwach bei hohen Frequenzen
- > bei Luminanz: fällt erst sehr viel später ab

Weber-Fechner-Gesetz

subjektiv empfundene Stärke von Sinnesindrücken ist proportional zum Logarithmus der Intensität des physikalischen Reizes

$$\frac{\Delta I}{I} = \text{const.}$$

Kontrastsensitivität

- > bei Chrominanz: bei niedrigen Frequenzen ausgeprägt, schwach bei hohen Frequenzen
- > bei Luminanz: fällt erst sehr viel später ab

Raytracing

Abtastung

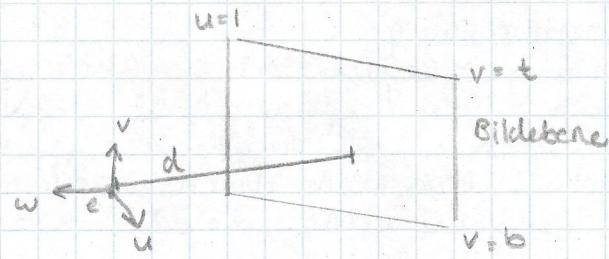
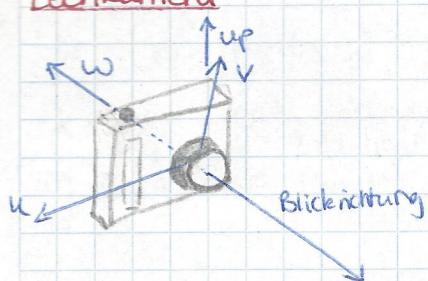
Abtasttheorem: ein kontinuierliches, bandbegrenztes Signal mit einer max. Frequenz f_{\max} muss mit einer Frequenz größer $2f_{\max}$ abgetastet werden, damit aus dem diskreten Signal das Ursprungssignal exakt rekonstruiert werden kann

Aliasing Fehler beim Abtasten von Signalen

↳ im rekonstruierten Signal treten Frequenzen auf, die im Original nicht enthalten sind

Lösung: Filtern vor Abtastung
höhere Abtastrate + anschließende Filterung

Lochkamera



RAYTRACING

Verfahren zur Bildsynthese → Licht entlang von Strahlen von Kamera aus zurückverfolgen

- betrachte jeden Pixel des Bildes; verfolge Strahl durch Pixel, finde nächste Fläche
 - ~> berechne Schattierung
 - ~> siehe Strahlverfolgung fertig wenn Flächen spiegeln/transmittieren

Schritte des Algo.

- (1) Erzeugung der Sichtstrahlen durch jeden Pixel (ray generation)
- (2) Schritt berechnung (ray casting, ray intersection)
- (3) Schattierung, Beleuchtungsberechnung (shading)
- (4) Sekundärstrahlen für Spiegelung und Transmission

Koordinatensysteme

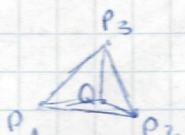
Baryzentrische Koordinaten geg. k Punkte $P_1, \dots, P_k \in \mathbb{R}^n$, $k \leq n+1$

Punkt $Q = \lambda_1 P_1 + \dots + \lambda_k P_k$ mit $\lambda_1 + \dots + \lambda_k = 1$

~> $(\lambda_1, \dots, \lambda_k)$ = baryzentrische Koordinaten von Q bzgl. P_1, \dots, P_k

Beispiele: → alle $Q = \lambda_1 P_1 + \lambda_2 P_2$ liegen auf Geraden von P_1, P_2 falls $\lambda_1 + \lambda_2 = 1$

→ $Q = \lambda_1 P_1 + \lambda_2 P_2 + \lambda_3 P_3$ in Dreiecke $(P_1, P_2, P_3) \Leftrightarrow \lambda_1 + \lambda_2 + \lambda_3 = 1, \lambda_1, \lambda_2, \lambda_3 > 0$



$$\lambda_1 = \frac{A_\Delta(Q, P_2, P_3)}{A_\Delta(P_1, P_2, P_3)}$$

$$\lambda_2 = \frac{A_\Delta(P_1, Q, P_3)}{A_\Delta(P_1, P_2, P_3)}$$

$$\lambda_3 = \frac{A_\Delta(P_1, P_2, Q)}{A_\Delta(P_1, P_2, P_3)}$$

Anwendung: Farbwert interpolation: gesucht: Farbe c_Q von Punkt Q in $\Delta(P_1, P_2, P_3)$ mit Farben $C_1, C_2, C_3 \Rightarrow$ berechne $\lambda_1, \lambda_2, \lambda_3$

$$\Rightarrow c_Q = \lambda_1 C_1 + \lambda_2 C_2 + \lambda_3 C_3$$

(1) Ray generation

$$\vec{u} = \vec{u}_p \times \vec{w}$$

$$\vec{v} = \vec{w} \times \vec{u}$$

→ Normalisieren

$$\vec{\omega} = \frac{(\vec{e} - \vec{z})}{\|\vec{e} - \vec{z}\|}$$

$$\text{Richtungsfeld. } \vec{s} = u \cdot \vec{u} + v \cdot \vec{v} - d \cdot \vec{\omega}$$

$$u \in [l, r]$$

l, r linker/rechter Rand

$$v \in [b, t]$$

b, t unterer/obere Rand

$$\text{Strahlgleichung } \vec{r}(t) = \vec{e} + t \vec{s}$$

$t=0$: kamerapos.

$t=1$: Pixelmitte

$$\text{id.R. } \vec{r}(t) = \vec{e} + t \vec{d} \quad \vec{d} = \frac{\vec{s}}{\|\vec{s}\|}$$

Stereo rendering für jedes Auge ein Bild $\Rightarrow 2$ Kameras

(2) Ray Casting Finde Objekt, das den Sichtstrahl am nächsten zur Kamera schneidet
 $\rightarrow t' > 0$ (vor Kamera) und $t' < t$ (nächster?)

Schnittpunktberechnung

Strahlgleichung in Kugel/Ebenenform - Gleichung einsetzen

mit Dreieck: $\lambda_1, \lambda_2, \lambda_3$ baryz. Koord. eines mögl. Schnittpunktes

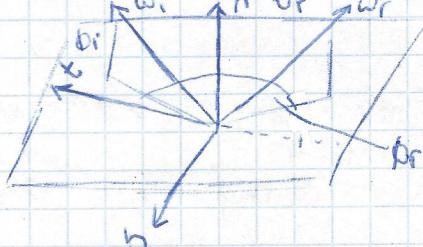
Schnittpunkt: Lege Ebene durch Dreiecke, berechne Schnitt mit Ebene
 \rightarrow berechne $\lambda_1, \lambda_2, \lambda_3$, teste auf Positivität

(3) shading

Reflexion Konzept zur Reflexionsbeschreibung an einem Oberflächenpunkt:
Bidirektionale Reflektanzverteilungsfunktion

- > Proportionalitätskonstante $f_r(w_i, x, w_r)$
Verhältnis von ausgehendem (w_r) zu einfallendem (w_i) Licht
- > Verwendung Referenzkoord.-sys. um BRDFs unabh. von bestimmtem Oberflächenpunkt angeben zu können

$$f_r(w_i, x, w_r) = f_r(\theta_i, \phi_i, x, \theta_r, \phi_r)$$



t = Tangente
 b = Bitangente
Polarwinkel $\theta \in [0, \pi]$ zur Normale
Azimutalwinkel $\phi \in [0, 2\pi]$

- > Isotropie = Rotationsinvarianz um die Normale
- > Anisotropie = Reflexion abh. von Rotation um Normale

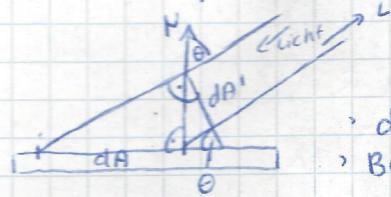
Phong-Berechnungsmodell modelliert Reflexion mit 3 Komponenten

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

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

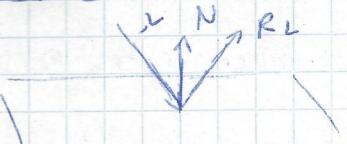
Schattierungsberechnung

Lambertsche Reflexion ideal-diffus



- Lichtstrahl aus Richtung L , Intensität I_L ,
- Materialkoeffizient k_d
- $dA = dA'/\cos \theta$ effektiv beleuchtete Oberfläche
- Bestrahlungsstärke der Fläche $I_L dA \sim \cos \theta$

Sekundärstrahlen



$$R_L = 2(L \cdot N)N - L$$

Spiegelnde Reflexion und Glanzlichter durch gerichtete Reflexion entstehen

Glanzlichter \rightarrow perfekte Spiegelung nur in Richtung R_L

$$\text{Spiegelnde Reflexion } I_S = k_s \cdot I_L \cdot \cos^n \alpha = k_s \cdot I_L \cdot (R_L \cdot V)^n$$

strahl zur Kamera

Schattierung von Dreiecksnetzen

Flat Shading: Verwendung der Dreiecksnormale für Beleuchtung

Normalen für Dreiecksnetze:

Vertex-Normalen bestimmen über

gewichtete Summe der Normalen angrenzender Flächen

Normalen für ein Dreieck über Kreuzprodukt

\rightsquigarrow (1) berechne Normale für jedes Dreieck

(2) für jedes Vertex: summiere Normalen aller angrenzenden Dreiecke

(3) normalisiere Vertex-Normalen

Interpolation von Normalen:

- Berechnung durch lineare komponentenweise Interpolation der Vertex-Normalen anhand der baryzentrischen Koordinaten

Bsp.: (1) n_1, n_2, n_3 Normalen des $\Delta(P_1, P_2, P_3)$ (2) interpolierte Normale n_Q an einem Punkt Q auf Δ

$$= \Rightarrow n_Q = \lambda_1 n_1 + \lambda_2 n_2 + \lambda_3 n_3$$

\rightsquigarrow Beleuchtungsberechnung mit Normale $n_Q / \|n_Q\|$

\rightarrow lineare komponentenweise Interpolation nicht längenerhaltend

Beleuchtungsberechnung mit interpolierten Normalen = Phong Shading

Licht und Schatten andere Objekte der Szene können Schatten werfen

\rightarrow sende Schattenstrahl zur Lichtquelle

\hookrightarrow teste ob Strahl auf dem Weg von der Oberfläche zur Lichtquelle ein anderes Objekt schneidet

Lichtquellen

> Punktquelle: Position p , Intensität I_L



> paralleles Licht / direktionale Lichtquelle: Richtung d, Flussdichte e



(4) Sekundärstrahlen für Spiegelung und Transmission

Berechnung von Spiegelung und Transmission: Rekursives Raytracing

Spiegelungen: Reflexionsstrahlen

sichtstrahl P wird an Oberfl. refl. \rightarrow Reflexionsstrahl R

\rightarrow berechne Schnittpunkt, gebe Farbe zurück

rekursive Verfolgung weiterer Schatten- (Sekundärstrahlen)

\rightarrow addiere Farbe am Ausgangspunkt

Rekursionstiefe

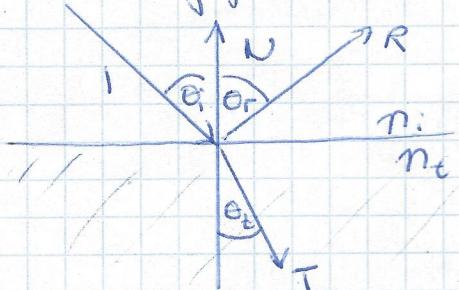
vorgegebene max. Anzahl von Reflexionen oder Rekursion bis Beitrag der Farbe vernachlässigbar

Snellsches Brechungsgesetz Richtungsänderung einer Welle bei Medienübergang

$>$ Brechzahl $n = c_0/c_n$

$>$ Licht bewegt sich untersch. schnell in untersch. Medien

$>$ Ausbreitungsgeschw. senkrecht zur Wellenfront



$$n_i \sin \theta_i = n_t \sin \theta_t$$

$>$ Brechung bei Übergang in optisch dichteres Medium ($n_t > n_i$) zum Lot nn'

$>$ Fresnel-Effekt: Verteilung der Strahldichte

$>$ Grenzwinkel, Totalreflexion: $\sin \theta_c = n_i/n_t$

$>$ Transmissionsvektor $T = -\frac{\sin \theta_t}{\sin \theta_i} (1 - N \cos \theta_i) - N \cos \theta_t$

$$= -\frac{n_i}{n_t} 1 + \left(\frac{n_i}{n_t} \cos \theta_i - \sqrt{1 - \left(\frac{n_i}{n_t} \right)^2 (1 - \cos^2 \theta_i)} \right) N$$

Transmission

$>$ transparentes Material mit keiner, konst. Abschwächung

$>$ exponentieller Abfall durch Absorption

$>$ Dispersion (Aufteilung in Wellenlängen \rightarrow Prismen)

(5) Bildzeugung = Abtastung

Aliasing-Effekte durch ungenügende Abtastung von Signalen

Lösungen: Filtern \rightarrow wir können Signal nicht bandbegrenzen
Überabtasten

Überabtastung = Supersampling, versuche herauszufinden welchen Beitrag die Objekte zu Pixelfarben liefern

Uniformes Supersampling: statt Abtastung eines Punktes innerhalb eines Pixels tastet k^2 -mal in äquidistanten Intervallen ab

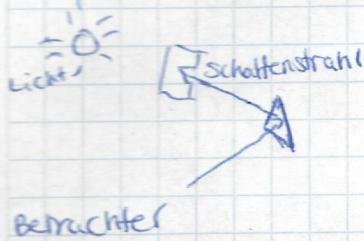
\rightarrow nehme Mittelwert als Pixelfarbe

! nicht mehrere Samples an derselben Stelle

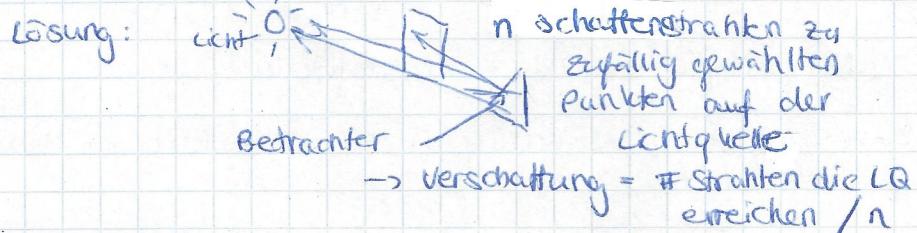
stochastisches Sampling: verwenden zufällige Samples in einem Pixel
 (mit Stratifizierung: teile Pixel in Gitter; Zufallspunkt pro Zelle)
 ~ mehr Rauschen

Distributed Raytracing

Probleme des whitted-style Raytracing: Bilder zu makellos



- perfekte Spiegelung / Transmission
- harte Schattenkanten
- unendliche Schärftiefe



→ Verschattung = # Strahlen die LQ erreichen / n

Bewegungsunschärfe

Verteilung der Strahlen in der Zeit

erzeuge für jeden Pixel n Strahlen für Zeitpunkt $t' \in [t; t + \Delta t]$

→ Raytracing zum Zeitpunkt t'

→ mittlere Farbwerte

Tiefenunschärfe

Modell der „dünnen Linse“:

weitere Samples für das Abtasten der Linsenfläche

• wähle Punkt auf Bildebene P_b und der Linse P_l

• berechne Punkt auf Fokusebene P_f

• erzeuge Primärstrahl von P_l durch P_f

Imperfekte Spiegelung und Transmission

Distributed Raytracing: um alle Lichtwege zu berücksichtigen sind mehrere Strahlen für Reflexion, Transmission notwendig (pro Schnittpunkt!)
 → Explosion des Strahlenbaums
 ~ wähle zufällige Schatten- und Sekundärstrahlen

Lichttransport, Beleuchtungssimulation

Lichtverteilung in einer Szene:

$$L(x, \omega) = \underbrace{L_e(x, \omega)}_{\text{Strahlstärke}} + \int_{\Omega} f_r(w_i, x, \omega) L(x, \omega_i) \cos \theta_i \, d\omega_i$$

Emissionsstrahl

Integrationsbereich: Ω^+ positive Hemisphäre (nur Reflexion)

Ω alle Richtungen

• $\cos \theta_i$: Einfallrichtung

• $f_r(\omega_i, x, \omega)$: Reflektanzverteilungsfkt.: wie viel Licht aus Richtung ω_i wird in Richtung ω reflektiert

• $L_i(x, \omega_i)$: einfallendes Licht

~ $L_i(x, \omega_i) = L(y, -\omega_i)$: ausgehendes Licht

$y = \text{raycast}(x, \omega_i)$

Monte-Carlo-Integration

berechne Integral näherungsweise durch Auswertung an N zufällig gleichverteilten Stellen $x_i \in [a, b]$:

$$\int_a^b f(x) dx \approx \frac{b-a}{N} \sum_{i=1}^N f(x_i)$$

→ Distrib. Raytr.

$$L(x, w) = L_e(x, w) + \frac{2\pi}{N} \sum_{i=1}^N f_r(w_i, x, w) L_i(x, w_i) \cos\theta_i$$

(2π entspricht Oberfl. einer Halbkugel)

→ rekursive Auswertung findet Ausbreitungswege von der Lichtquelle zum Betrachter

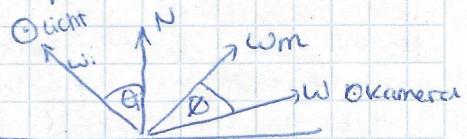
Umsetzung

Prinzip-Beleuchtungsmodell als BRDF:

$$f_r(w_i, x, w) = k_d + k_s ((2N(N \cdot w_i) - w_i) \cdot w)^+ / (N \cdot w)^+$$

$$w_m = 2N(N \cdot w_i) - w_i$$

w = Richtung für die BRDF ausgewertet



Flächenlichtquellen: Emissionsterm zur Materialdef. hinzufügen
 $L_e(z, w) > 0$

Algo: erzeuge zufällige Richtungen (in Kugelkoord.):

- z gleichverteilte Zufallszahlen $\mu_1, \mu_2 \in (0, 1)$

- $\theta = \arccos(1 - 2\mu_1)$, $\phi = 2\pi\mu_2$

- $w_i = (\sin\theta \cos\phi, \sin\theta \sin\phi, \cos\theta)$

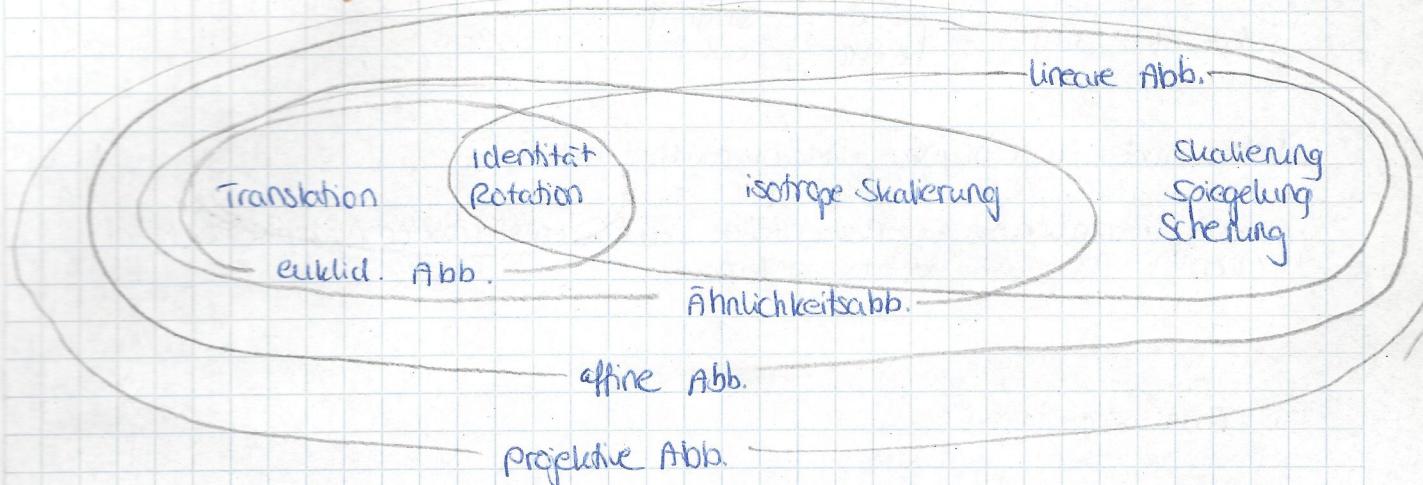
- verwirfe Richtung, erzeuge neue falls $N \cdot w_i < 0$

rekursiv berechnen bis Rekursionstiefe N

Path Tracing

nur jeweils ein Strahl, dafür mehrere Pfade pro Pixel
→ mehr abgetastete Richtungen nahe Kamera

Transformationen



euklidische / rigide Transf.

- erhalten Abstände, Inhaltsgrößen, Winkel

Ähnlichkeitsabb.

- erhalten Winkel

Lineare Abb.

- additiv $T(p+q) = T(p) + T(q)$
- homogen $T(a \cdot p) = a \cdot T(p)$

affine Abb.

- parallele Linien bleiben erhalten
- Teilverhältnisse

projektive Abb.

- Geraden werden auf Geraden abgebildet

2D Transformationen

Skalierung

ändert Längen, Winkel falls $s_x \neq s_y$

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

Spiegelung - negative Skalierung

Spiegelung an y-Achse $\rightarrow \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}$

Spiegelung an x-Achse $\Rightarrow \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$

Scherung

verschiebung parallel zu einer Achse \rightarrow Flächeninhalt bleibt erhalten
Länge Verschiebungsvektor prop. zum Abstand von der Achse

horizontal (y bleibt gleich) $\text{shear}_x(s) = \begin{pmatrix} 1 & s \\ 0 & 1 \end{pmatrix}$

vertikal (x bleibt gleich) $\text{shear}_y(s) = \begin{pmatrix} 1 & 0 \\ s & 1 \end{pmatrix}$

Rotation

Rotation um ϕ gg. Uhrzeigersinn

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

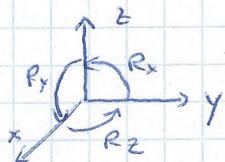
Hintereinanderausführung = Matrixmultiplikation

assoziativ $(RS)T = R(ST)$

nicht kommutativ \rightarrow Reihenfolge der Transformationen entscheidend
(Konvention: von rechts anfangen)

3D Transformationen

Rotation



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

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

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

\rightarrow alle orthogonal $\rightarrow M^T \cdot M = M \cdot M^T = I_{n \times n} \quad |\det(M)| = 1$
 $M^{-1} = M^T$

Rotation Koord. sys. in u,v,w auf kart. Koord.

$$R_{uvw} = \begin{pmatrix} u_x & u_y & u_z \\ v_x & v_y & v_z \\ w_x & w_y & w_z \end{pmatrix}$$

R_{uvw}^T bildet kart. Koord. auf u,v,w ab

$\rightarrow p$ in kart. Koord. $\rightarrow R_{uvw} \cdot p \hat{=} p$ in Objektkoord. u,v,w

\rightarrow andere Richtung mit R_{uvw}^T

Rotation um Achse d um Winkel ϕ

\rightarrow bestimme orthonormales KoSys mit d als Achse $d = (d_x, d_y, d_z), |d| = 1$

\rightarrow wähle e zB $e = \frac{1}{\sqrt{d_x^2 + d_z^2}} (0, -d_z, d_y)$

\rightarrow wähle f = d x e

$$\Rightarrow M = \begin{pmatrix} d_x^T \\ e^T \\ f^T \end{pmatrix} = \begin{pmatrix} d_x & d_y & d_z \\ e_x & e_y & e_z \\ f_x & f_y & f_z \end{pmatrix} \Rightarrow R_{d,\phi} = M^{-1} R_x(\phi) M$$

Euler Rotation jede Rotation $\hat{=}$ 3 Rotationen um Hauptachsen

$$\Rightarrow R = R_z(\phi) R_y(\theta) R_x(\psi)$$

ψ, θ, ϕ = Eulerwinkel \rightsquigarrow beschreiben Orientierung eines Objekts zusammen mit Achsenfestlegung, Reihenfolge

> gegeben Rot. Matrix R, Achsenkonvention: Eulerwinkel ablesbar

! Lösung nicht eindeutig

: Gimbal Lock

Inverse Transformationen

→ Matrixinversion

Eigenschaften:

- $S^{-1}(x, y, z) = S\left(\frac{x}{s}, \frac{y}{s}, \frac{z}{s}\right)$
- $R^{-1}(\phi) = R(-\phi) \hat{=} R^{-1} = RT$ Rotation
- $T^{-1}(x, y, z) = T(-x, -y, -z)$ Translation
- $(AB)^{-1} = B^{-1} A^{-1}$

Affine Abbildungen

- Linien werden auf Linien abgebildet
- Parallelität bleibt erhalten
- Teilverhältnisse
- nicht winkelerhaltend

Rot., Transl., Skal., scher.

Homogene Koordinaten

- Menge aller Geraden durch Ursprung im \mathbb{R}^3 = reeller projektiver Raum $P(\mathbb{R}^3)$
- $v \in \mathbb{R}^3$ definiert Gerade, λv ($\lambda \in \mathbb{R}, \lambda \neq 0$) def. selbe Gerade
- $\dim(P(\mathbb{R}^3)) = 2$

Repräsentation von ...

... Punkten: $(x, y)_2 \rightarrow (x, y, 1)_n \equiv \{(x', y', w) \mid (\frac{x'}{w}, \frac{y'}{w}) = (x, y)\}$

... Richtungen: $(x, y)_2 \rightarrow (x, y, 0)_n$

$\rightsquigarrow P(\mathbb{R}^3)$ enthält affine Punkte, Richtungen des \mathbb{R}^2 ohne "Falluntersch."

- $(ax, ay, 0)_n$ $a \neq 0$ repr. gleichen Punkt im \mathbb{R}^2

λM ($x \neq 0$) beschreiben bei homog. Koord. dieselbe Abb.

Grundlegende 3D Transformationen

$$\text{scale}(sx, sy, sz) = \begin{pmatrix} sx & 0 & 0 & 0 \\ 0 & sy & 0 & 0 \\ 0 & 0 & sz & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\text{Shear}_z(dx, dy) = \begin{pmatrix} 1 & 0 & dx & 0 \\ 0 & 1 & dy & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Rotationen:

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

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

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

Koordinatensysteme in CG

Objektkoordinaten



Weltkoordinaten



Kamerakoordinaten



Kamera transformation

virtuelle Kamera def. durch Position e, negative Blickrichtung w, 'up'

$$\rightsquigarrow u = upxw \quad v = wxu$$

u, v, w - Basis des kamerakS

Hierarchisches Modellieren

Modelltransformationen = zusammengesetzte Transformationen

- > Vereinfachung: mehrfache Kopien von Objekten ermöglichen
 - ~> Objekte zu Gruppen zusammenfassen
 - ~> Gruppen anordnen
- > Szenengraph = gerichteter acyklicher Graph
 - ~> erlauben effiziente Operationen in komplexen Szenen
 - u.a. Wiederverwendung von Matrizen durch Matrix Stacks

Transformation von Normalen

Normalenvektor = Einheitsvektor lokal senkrecht zur Oberfläche

- > Normale = Biulektoren: stehen senkrecht auf Tangentialfläche, nicht durch Differenz zweier Ortsvektoren definiert

~> verwende homogene Koord., transformiere Tangentenebene zur Normale statt Normalenvektor selbst

$$\text{Spiral} \quad \text{Tangentialebene } E : n_x x + n_y y + n_z z + d = 0$$

$$n = (n_x, n_y, n_z, d) \quad ; \quad p \in E \quad p = (x, y, z, 1) \quad \rightarrow n^T p = 0$$

Trick: transformiere $p \in E$, schaue wie transformierte Normale sein muss

$$\rightarrow n^T p = 0, \text{ einsetzen: } p = (M^{-1} H)p$$

$$\rightarrow n^T (M^{-1} H) p = 0$$

$$\rightarrow \underbrace{(n^T M^{-1})}_{=n'^T} \cdot \underbrace{(H p)}_{p'} = 0 \quad \text{falls } n'^T = (n^T M^{-1}) \\ \Rightarrow \boxed{n' = (M^{-1})^T n}$$

Schnitttests in Modellkoord. einfacher

weltkoord.: jedes Primitiv behandelt Transf. selbst, kompliziert:

$$p_{ws} = M p_{os} \rightarrow p_{os} = M^{-1} p_{ws}$$

$$\rightarrow \text{neuer Strahlungspunkt } e_{os} = M^{-1} e_{ws}$$

$$\rightarrow \text{neue Strahlrichtung } d_{os} = M^{-1} d_{ws}$$

$$\rightarrow e_{ws} + t_{ws} d_{ws} \rightsquigarrow e_{os} + t_{os} d_{os}$$

: d_{os} i.A. nicht normalisiert

Fall 1: verwende $\frac{d_{os}}{\|d_{os}\|}$ (normal.)

$$\rightsquigarrow t_{ws} + t_{os}$$

Fall 2: verwende d_{os}

$$\rightsquigarrow t_{ws} = t_{os}$$

=> Strahlzeugung in Welt-/Kamera K.S. aber Schnittpunkt übl. in Modell K.S.

Texture Mapping

Textur-Quellen

- Rasterbilder bestehend aus Texels
- 1D/2D Texturen einfache Einstellung / Inkquisition
- 3D aus Simulation

2D Textur

Planare Projektion:

- definiere Ebene z.B. durch Punkt p_1 , 2 aufspannende Vektoren s, t
- Texturkoord. eines Oberflächenpunkts x :
$$s = (x - p_1) \cdot s \quad \leftarrow \text{bei 1D nur das}$$

$$t = (x - p_1) \cdot t$$
- Bereich $[0, 1]^2$ definiert Textur \rightarrow unabh. von tats. Auflösung

Mapping von 2D Texturen

Standardkörper mit natürlicher Parametrisierung

z.B. Kugel: Punkte auf Oberfl. mit Kugelkoord. (r, ϕ, θ)

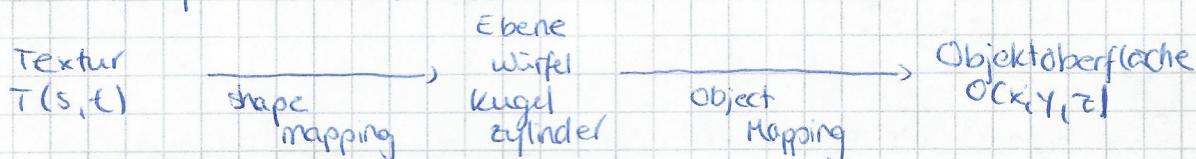
$$\begin{pmatrix} s \\ t \end{pmatrix} = \begin{pmatrix} \phi / 2\pi \\ \theta / \pi \end{pmatrix}$$

Würfel-Parametrisierung

- Projektion von 6 Ebenen je nach Oberflächennormale
- Oder: lese Würfeltextur dort aus, wo ein Strahl vom Objektmitelpunkt durch einen Oberflächenpunkt einen umgebenden Würfel schneidet

Texturierung beliebiger Objekte

Standardkörper als Zwischenschritt



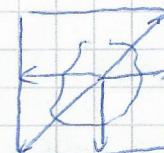
Object Mapping:



Normale der Hilfsfläche
 \rightarrow Oberfläche



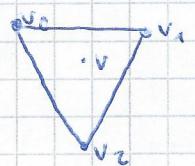
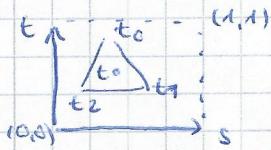
Normale der Oberfl. \rightarrow Hilfsfl.



Linie durch Mittelpunkt

Texturkoord. für Dreiecksnetze

- Parametrisierung in Texurkoord. speichern:
→ jedem Eckpunkt $v_i = (x_i, y_i, z_i)$ eines Dreiecks wird Texurkoord. (t_x, t_y) zugewiesen
- affine Abb. zwischen 2D Texurräum, 3D Objektraum
→ Interpolation mit baryzentrischen Koord.



$$v = v_0 + \lambda_1 \cdot (v_1 - v_0) + \lambda_2 \cdot (v_2 - v_0)$$

$$t = t_0 + \lambda_1 \cdot (t_1 - t_0) + \lambda_2 \cdot (t_2 - t_0)$$

→ Stückweise lineare Approximation einer Parametrisierung

Texture Wrapping

Texurkoord. > 1,0 oder < 0,0 ?

- Repeat/Wrapping: Fortsetzen/Kacheln einer Textur über $[0,1]$ hinaus
- Adressierung pro Dimension wählbar
- clamping: Abschneiden, nicht fortsetzen

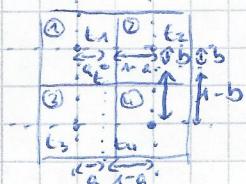
Textur Filterung

Vergrößerung = Magnification

Abbildung weniger Texel auf viele Pixel

→ Nearest Neighbor
verwendet Farbe des nächstliegenden Texels

→ Bilinear interpolation



$$\text{Farbe} = ①(1-a)(1-b) + ②f_1a(1-b) + ③(1-a)b + ④ab$$

→ höhere Ordnung

menschl. Wahrnehmung nimmt Unstetigkeiten in der Krümmung wahr

→ teuer, nicht nativ von Grafik-HW unterstützt

Verkleinerung - Minification

Abbildung mehrerer Texel auf einen Pixel

→ Aliasing-Artefakte falls nur 1 Texel ausgewählt obwohl Pixel mehrere Texel bedeckt
→ Unterabtastung

→ Vorfiltern (hohe Freq. von Abtastung entfernen)
Überabtastung (teuer!)

Mip-Mapping einfache Verfilterung von Texturen

CG

Speichere Rekursiv Texturen mit $\frac{1}{4}$ Größe (Halbierung entlang jeder Achse)
→ $\frac{1}{3}$ mehr Speicherbedarf

- Mittelung über je 2×2 Texel (kein optimaler Tiefpass!)
- gleichzeitige Filterung und Auflösungsreduktion

• wähle Texturauflösung (Mip-Map Stufe n) so, dass:

$$\text{Texelgröße}(n) \leq \text{Größe Pixelfootprint auf Textur} < \text{Texelgröße}(n+1)$$

• $n = 0 \hat{=}$ höchste Auflösungsstufe

• entgültiger Farbwert: m lineare Interpolation zw. 2 Mip-Map Stufen:
→ bilinear auf Stufe n, bilinear auf Stufe n+1
→ linear zw. diesen Farben

Anisotrope Texturfilterung

• Footprint eines Pixels im Texturraum oft eher länglich
→ Mip-Mapping isotrop (gleichförmig in s, t)

→ RIP Maps Rectangular MipMaps (selten in der Praxis)

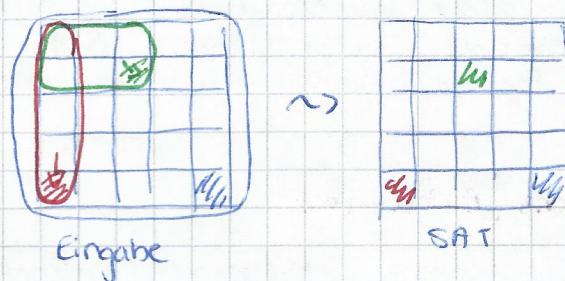
- Vorfilterungen unabh. in jeder Achse
- länglicher Abdruck im Texturraum → wähle entspr. vorgefilterte Textur
- 4facher Speicherbedarf
- rüst Anisotrope nur entlang der Achsen

Praxis

• Abtasten des Bereichs durch geschickte Kombi von Abtastungen versch. Mip-Map Stufen

Summed Area Tables (SATs)

$$sum = \sum_{i=1}^m \sum_{j=1}^n t_{i,j}$$



Anwendung: Filterung der Eingabetextur mit einem beliebigen rechteckigen, nach den Achsen ausgerichteten Box-FILTER

Footprint Bestimmung

• schicke Primärstrahl durch die Ecken eines Pixels oder betrachte einen 2×2 Pixelblock und bestimme die 4 Texturkoord.

• Differenz der Texturkoord. liefert Größe/Form des Footprints
→ Textur-Lookup an berechneter Stelle (mit Filterung)

Ray Differentials Schnitt eines Primärstrahls mit Objekt

→ pos. x, Normale n → def. Tangentialialebene

→ betrachte Veränderung der Texturkoord.

→ bestimme daraus Größe/Form des Footprints

Texturierungstechniken

Diffuse Textur Kontrolle der Eigenfarbe eines Materials

Bsp. Phong-Modell: kd aus Textur

$$I = k_a \cdot I_L + k_d \cdot I_L \cdot (\mathbf{N} \cdot \mathbf{L}) + k_s \cdot I_L \cdot (\mathbf{R} \cdot \mathbf{V})^n$$

Bump / Normal Mapping Variation der Normale einer Oberfläche

→ verändertes N aus Textur (damit auch anderes R)

→ berechnet aus Veränderung einer Basisfläche durch Bump Map / Normal Map
- Fläche bleibt geometrisch flach, nur Normale variieren

Gloss-Map / Gloss-Textur Kontrolle der Stärke, Streuung der spekularen Reflexion
→ verändertes ks, n aus Textur

Displacement Mapping Verschiebung der Oberfläche, Änderung der Normale
- mehr als nur Änderung der Beleuchtungsberechnung

Inverse Displacement Mapping Schnittpunktberechnung im Texturraum
Darstellung von Oberflächendetails

Ambient Occlusion Kontrolle der ambienten Beleuchtung (Umgebungslicht)
→ ka aus Textur, meist auch kd

Textur-Atlas spezielle (bijektive) Parametrisierung

- jedem Oberfl.-punkt entspricht eine Stelle in der Textur
- keine Stelle der Textur taucht an mehr als einem Oberfl.-punkt auf
- Erstellung aufwendig per Hand od. automatisch

Anwendung:

- Speichern einer Funktion / Daten auf Oberfl.
- Feststellen einer Textur direkt auf Objekt
- Berechnung von Normal-Maps aus fein aufgelösten Dreiecksnetzen

Transparenz, Alpha-Test

Semi Transparenz: verwende Alpha-Kanal um die Transparenz der Oberfl. an einem Punkt zu bestimmen Alpha MAP

Alpha-Test: verwirfe Objekt-/Schnittpunkt falls $a < threshold$ Alpha MASK

3D Texturen

Solid-Textures: "Herausschneiden einer Skulptur" durch Zuweisung von 3D-Texturekoord. an Vertices

Volumenvisualisierung von Simulations- / Messdaten

Environment Mapping

Darstellung reflektierender Objekte mit Spiegelung der Umgebung ohne geom. Repräsentation

→ Approximation der Refl. ohne Raytracing

- Raytracing: mitt ein Strahl kein Objekt, dann wird das ankommende Licht aus der Env. Map ausgelassen
- Env. Map ≈ Kugel um Objekt (Szene)

vereinfachende Annahme:

verwende nur Richtung r des refl. Strahls, ignoriere Ausgangspunkt \times

Latitude/Longitude-Maps mögl. Parametrisierung von Env. Maps

- Parametrisierung über Polarwinkel $\theta \in [0, \pi]$ zur z-Achse, Azimuthalwinkel $\phi \in [0, 2\pi]$

Berechnung der Winkel aus $r = (r_x, r_y, r_z)$, $\|r\|=1$
 $\theta = \arccos(r_z)$, $\phi = \arctan_2(r_y, r_x)$, $t = \theta/\pi$, $s = \phi/2\pi$

⇒ teuer, ungleichmäßige Abtastung an Polen

Sphere Mapping

Bild der Umgebung ebenfalls als 2D Textur
aber: Aufnahme der Env. Map mit Kamera möglich

- Fotografiere Spiegelkugel mit Teleskopobjektiv
- Bild auf Kugel = Sphere Map

Berechnung Texturkoord.:

(!) aktuelle Betrachterrichtung r

Oberfl. normale n_x

Refl. Richtung r

Richtung v_0 aus der Sphere Map aufgenommen

(?) welcher Punkt auf Spiegelkugel entspricht Refl. Richtung r ?

Beob.: gesucht ist Punkt, dessen Normale n mittig zw. r und v_0 liegt

$$\Rightarrow h = (r + v_0) / \|r + v_0\| = n$$

Half-Way vector

$$\Rightarrow \begin{pmatrix} s \\ t \end{pmatrix} = \frac{1}{2} \begin{pmatrix} n_x + 1 \\ n_y + 1 \end{pmatrix}$$

⇒ ungleichm. Abtastung an den Ecken

Cube Environment Maps Abbilden der Umgebung auf Würfel um Betrachter

geg.: $r = (r_x, r_y, r_z)$

→ betragsmäßig größte Komponente von r wählt Würflfläche

$$|r_x| > |r_y| \wedge |r_x| > |r_z| \Rightarrow r_x > 0 \quad ? \text{ right : left}$$

$$|r_y| > |r_x| \wedge |r_y| > |r_z| \Rightarrow r_y > 0 \quad ? \text{ back : front}$$

→ bestimme Schnittpunkt

Vorfilterung von Env Maps

statt mehrfachem Abtasten

Beob.: bei imperfekter Spiegelung strögt Licht aus einem Bereich um r zum refl. Licht bei

Räumliche Datenstrukturen

Optimiere Raytracing: weniger Schnittberechnungen

- vermeide Berechnungen mit weit vom Strahl entfernten Objekten
- Raum unterteilen um potentiell geschnittene Geometrie schneller zu finden

Konservative Hüllkörper Bounding volumes

möglichst eng anliegend

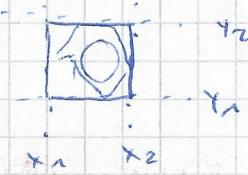
- überprüfe ob ein Schnitt mit einem Hüllkörper existiert

Varianten:

- Kugel: schneller Schnittalgo, schlechte Effizienz da zu groß
- achsenparallele Box: einfache Berechnung, wichtigster Hüllkörper
- orientierte Bounding Box: aufwändige Berechnung
- Slabs: Schnitt von Paaren paralleler Halbebenen

Axis Aligned Bounding Box (AABB)

Schnittberechnung Strahl - AABB



- für jede Dimension (hier für x)

- $d_x = 0$ (Strahl parallel zur yz-Ebene) $\wedge (x_1 < x_{\text{ex}}, x_{\text{ex}} > x_2)$
→ kein Schnitt

- berechne t_1, t_2

$$t_1 = \frac{x_1 - ex}{dx}$$

$$t_2 = \frac{x_2 - ex}{dx}$$

$t_1 &> t_2 \rightarrow$ tauschen von t_1, t_2

$t_{\text{near}} > t_{\text{far}} \Rightarrow t_{\text{near}} = t_1$

$t_2 < t_{\text{far}} \Rightarrow t_{\text{far}} = t_2$

{ sodass nach allen Dimensionen max/min }

- $t_{\text{near}} > t_{\text{far}}$ → Box nicht getroffen
- $t_{\text{far}} < 0$ → Box hinter Strahl
- $t_{\text{near}} > 0$ → nächster Schnitt bei t_{near}
- sonst → nächster Schnitt bei t_{far}

Bounding volume hierarchien

zusammenfassen von Primitiven / Objekten / Gruppen in Gruppen

Aufbau: top-down

- Box aller Objekte → teilen in 2 Gruppen → teilen in 2 Gruppen
→ ... → bis nur noch m Objekte pro Gruppe

- jeder Knoten speichert eigene AABB, verweist auf Kindknoten
(z.B. Primitive, Liste von Prim. oder Gruppen)

Unterteilungskriterien

- in der Mitte senkrecht zur Achse der größten Ausdehnung
- entlang $x \rightarrow$ entlang $y \rightarrow$ entlang $z \rightarrow$ entlang $x \rightarrow \dots$
- so, dass in jeder Teilmenge \approx gleiche # Objekte
- Szene graph, Modellhierarchie
- Kostenfunktion minimieren

Schnittberechnung

Schnitt mit Wurzel, rekursiv absteigen

- teste Schnitt mit Objekten im Kindknoten (erte weiter absteigen)
- prüfe ob weitere Schnittpunkte in entfernten Knoten existieren kann

! Hüllkörper können überlappen \rightarrow Schnitt nicht sofort zurückgeben

- ④ Konstruktion, Traversierung einfach
- Binärbaum mit fixer, geringer Verzweigung im Mittel O(log n) $n = \#$ Primitive pro Szene ohne BVH O(n)
- ⑤ Unterteilung finden schwierig
→ Schlechte Unterteilung kann zu schlechter Performance führen

reguläre Gitter

Unterteilt Raum in Zellen gleicher Größe, Form ausgehend von Bounding Box der Szene

- Eintrag der Objekte in Zellen, die von ihm geschnitten werden
- Traversierung aller vom Strahl getroffenen Zellen, Schnittberechnung mit den enthaltenen Objekten

Schnittpunktberechnung

- teste für jede Zelle entlang Strahl: Objekt in der Zelle?
- nein: verwerten, weiter
- ja: gebe nächsten Schnittpunkt zurück

Option 1: verwirfe Schnittpunkte außerhalb aktueller Zelle

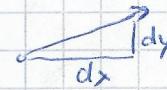
Option 2: Mailboxing: Schnitttest nur einmal pro Objekt, Schnittpunkte speichern

Gitter-Traversierung finde Zellen die Strahl durchläuft

- Start-Zelle: schneide Strahl mit Bounding Box der Szene
- Achtung: kann auch in der Bounding Box liegen
- ↓ liefert

(i, j) Zellenindex

$$i = \lfloor (ex - x_{min}) / g_x \rfloor \quad j = \lfloor (ey - y_{min}) / g_y \rfloor$$



$$\rightarrow ex + t_{next,x} \cdot dx = (i+1) \cdot g_x + x_{min} \quad (dx > 0) \quad ! \quad dx, dy \text{ können } 0 \text{ sein}$$

$$\rightarrow t_{next,x} = \dots$$

- ⑥ Konstruktion, Traversierung einfach

- ⑦ falls wenig Zellen belegt, einzelne Zellen dicht belegt ("Teapot in a Stadium")

adaptive Gitter teile Zellen rekursiv bis max. Anzahl Primitive/ Schritte CG

Octree

Bounding Box für Szene

- rekursive 1-zu-8 Unterteilung jeweils in der Mitte falls noch zu viele Primitive in Zelle
- feine Unterteilung nur dort wo Geometrie ist

Traversierung

- Test gegen AABBB der Wurzel
- Schnitt z. → Test mit Objekten gespeichert in Wurzel, überprüfe Kindknoten ob näher als Schnittpunkt
 - testen nicht-leerer Kindknoten
 - rekursiv bis unten
- Effizienz szenenabh.
 - Octree gut bei ungleich verteilter Geometrie

BSP-Baum, kd-Baum

Binary Space Partitioning Tree

- Erweiterung von Binäräbäumen auf k Dimensionen

- Verwendete Ebenen um Raum rekursiv zu unterteilen
 - ↳ Ebenen beliebig orientiert

Kd-Bäume

- Ebenen senkrecht zur x/y/z-Achse
- Primitive die Split Ebene schneiden in beide Kindknoten einfügen
- Unterteilung bis max. Anz. Primitive od. max. Anz. Schnitte

Eigenschaften

- echte Raumunterteilung: Knoten überschappen nicht
- Blattknoten = Primitive, innere Knoten = Split-Ebenen

Traversierung Strahl $r(t) = e + t d$, $t_{\min} \leq t \leq t_{\max}$

- Stack für Knoten zur Bearbeitung
 - init: Wurzelknoten

- schneide Strahl mit Split-Ebene (beginnt bei Wurzel)
 - $t_{\min} \leq t' \leq t_{\max}$: Schnitt auf Strahlsegment
 - traversiere beide Kinder rekursiv (zuerst das das e enthält)
 - t' nicht auf Strahlsegment:
 - traversiere nur geschnittenes Kind
 - Strahlsegment weiter unterteilen mit t'', t''', \dots
- Blatt erreicht:
 - feste Schnitt mit Primitiven
 - gebe Schnittpunkt zurück falls innerhalb des aktuellen Strahlsegm.
- Reihenfolge: von vorne nach hinten

Surface Area Heuristics

$SA(x) = \text{surface area}(x) = \text{Oberfläche}(x)$

Kosten für Unterteilung eines Vd / BVH Knoten p

$$C = C_T + \frac{SA(B_L)}{SA(B_p)} |P_L| c_i + \frac{SA(B_R)}{SA(B_p)} |P_R| c_i$$

- B_L, B_R, B_p BB der Primitive im rechten, linken Kindknoten, Knoten p

- $|P_L|, |P_R| = \# \text{ primitive rechter, linker Kindknoten}$

- $c_i = \text{kosten Strahl-Primitive Schnitttest}$

- $C_T = \text{kosten Traversierung vdl / BVH}$

→ Unterteile falls $C < (|P_L| + |P_R|) c_i$

Rasterisierung, Clipping, Projektionstransformationen

Bildsynthese

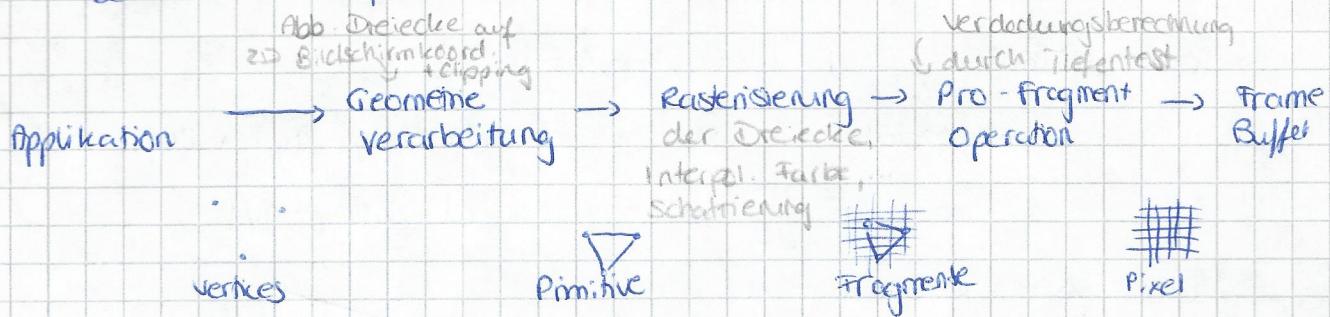
erzeugt Rasterbild aus Szenenbeschreibung (objekte)
 ↳ welche Objekte beeinflussen Pixelfarbe

Bildbasiert → Raytracing

- betrachte Pixel einzeln, bestimme welches Primitiv sichtbar → Pixelfarbe

Objektbasiert → Rasterisierung

betrachte Objekt / Primitiv / Fläche einzeln, bestimme welche Pixel Primitiv bedecken → Pixelfarbe



Rasterisierung

Rasterisierung von Linien

(1) Endpunkte des Liniensegments $(p_x, p_y), (q_x, q_y) \in \mathbb{Z}^2$

(2) Menge der Pixel die gesetzt werden sollen

$$\text{Steigung } m = \frac{q_y - p_y}{q_x - p_x}, \quad c = p_y - m p_x$$

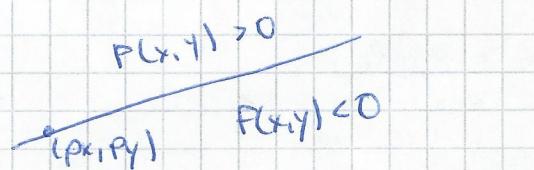
$$y = mx + c = \frac{q_y - p_y}{q_x - p_x} x + \frac{p_x q_x - q_y p_x}{q_x - p_x}$$

Variante 1: Brute force: jedes x einzeln einsetzen

Variante 2: inkrementelle Berechnung $y_n = mx_n + c$
 $y_{n+1} = m(x_{n+1}) + c = y_n + m$

Implizite Linendarstellung

$$F(x, y) = (y - p_y) - m(x - p_x)$$



werte $F(x, y)$ für $M = (x + \frac{1}{2}, y + \frac{1}{2})$ zwischen E, NE aus
 → $F(M) < 0$: M unter Linie → Linie verläuft haupts. durch NE

Bresenham-Algorithmus

Init: $d = F(p_x + \frac{1}{2}, p_y)$
 $(x_0, y_0) = (p_x, p_y)$

$d < 0 \Rightarrow NE \rightarrow (x_{i+1}, y_{i+1}) = (x_i + 1, y_i + 1)$

$$F(x_{i+1} + \frac{1}{2}, y_{i+1}) - F(x_i + 1, y_i + \frac{1}{2}) = 1 - m$$

$$d = d + 1 - m$$

$d \geq 0 \Rightarrow E \rightarrow (x_{i+1}, y_{i+1}) = (x_i + 1, y_i)$

$$F(x_{i+1} + \frac{1}{2}, y_{i+1}) - F(x_i + 1, y_i + \frac{1}{2}) = -m$$

$$d = d - m$$

→ Integer-Variante: $d = 2(q_x - p_x) F(x, y)$

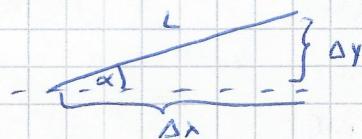
$$\text{→ Updates: } NE \rightarrow d = d + 2(p_y - q_y) + 2(q_x - p_x)$$

$$E \rightarrow d = d + 2(p_y - q_y)$$

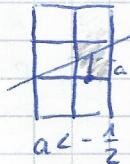
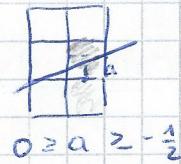
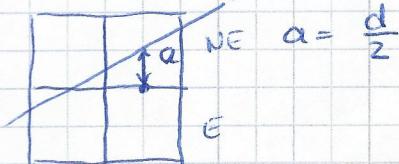
⇒ setzt $\Delta x = L \cos \alpha$ Pixel

Optimal: L

⇒ erhöhe Intensität um $\frac{1}{\cos \alpha}$



Modifizierung: zwei Pixel setzen, Intensitäten summieren zur Gesamtintens.



Rasterisierung von Polygone
n Eckpunkte P_1, \dots, P_n

Konvex → alle Innenwinkel < 180°

Brute Force (konvexe Polygone)
· Geradengleichungen der Kanten

Scanline

eine Pixel Zeile nach der anderen, Schnitte mit Polygon finden, Pixel in diesem Teil setzen

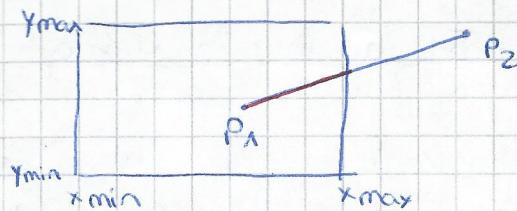
- P_1, \dots, P_n umlaufend sortiert, Rasterisiere Zeilen zw. oberstem, unterstem Punkt
- nutze begrenzte Geraden um Start-, Endpunkt zu bestimmen
- ersetze Geraden wenn Zwischenpunkt erreicht

Interpolierte Werte für Farbe, Textur, ... entlang der Kanten \Rightarrow

Clipping

Clipping von Linien

bestimme Teil der Linie innerhalb des Rechtecks



Cohen-Sutherland

jedes Bit
= Kante
Bit gesetzt
=> Punkt
außerhalb
bzw. Kante

1001	0001	0101
1000	0000	0100
1010	0010	0110

→ nicht trivial:

- Es gibt Kante(n) für die das Bit in Outcode(P_1) gesetzt, in Outcode(P_2) nicht gesetzt
- P_1P_2 schneidet Kante \rightarrow Schnittpunkt S
- P_1 außerhalb dieser Kante? weiter mit $SP_2 : SP_1$

$$\alpha\text{-Clipping} \quad WEC_E(P) < 0 \Leftrightarrow P \text{ außerhalb bzgl. Kante } E$$

$$\cdot WEC_{left}(P) = p_x - x_{min}$$

$$\cdot WEC_{bottom}(P) = p_y - y_{min}$$

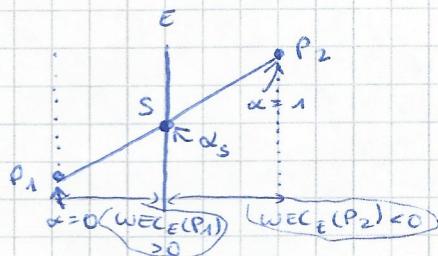
$$\cdot WEC_{right}(P) = x_{max} - p_x$$

$$\cdot WEC_{top}(P) = y_{max} - p_y$$

Outcode-Bit gesetzt falls $WEC < 0$

Schnittberechnung: parametrisierte Linie P_1P_2 mit α :

$$P = P_1 + \alpha(P_2 - P_1) \quad \alpha \in [0, 1]$$



α -Wert für Schnittpunkt mit Kante:

$$\frac{P_1S}{P_1P_2} = \alpha_S = \frac{WEC_E(P_1)}{WEC_E(P_1) - WEC_E(P_2)}$$

→ Algo:

- WECs von P_1, P_2
- Outcodes bestimmen
- testen auf trivial accept/reject
- ansonsten:
 - $\alpha_{min} = 0, \alpha_{max} = 1$
 - \forall Kanten E , für die Outcode-Bit gesetzt für P_1 oder P_2
 - α_S berechnen
 - Outcode(P_1) \in
 - $\alpha_{min} = \max(\alpha_{min}, \alpha_S)$
 - $\alpha_{max} = \min(\alpha_{max}, \alpha_S)$
 - $\alpha_{min} > \alpha_{max}$: Linie außerhalb
 - Segment $(P_1 + \alpha_{min}(P_2 - P_1), P_1 + \alpha_{max}(P_2 - P_1))$

Nichtkonvexe Clipping Regionen

unterteilen in konvexe Teilregionen \rightarrow Clipping gegen jede Teilregion

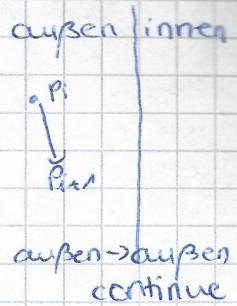
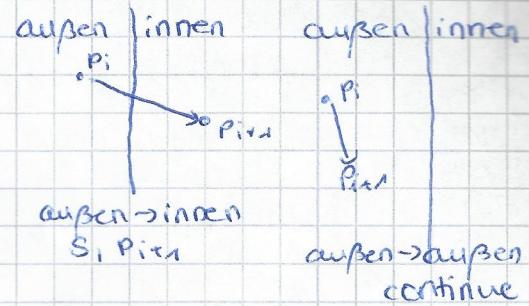
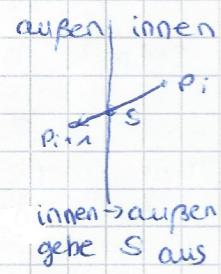
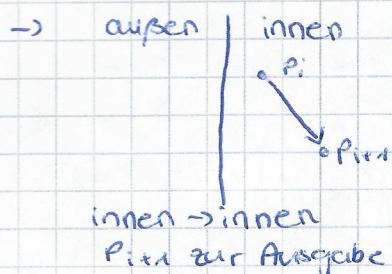
\rightarrow zusammensetzen

Clipping von Polygonen

Sutherland-Hodgeman Polygon Clipping

Clipping gegen jede Kante des Clipping-Polygons nacheinander

- Clipping gegen Kante: P_1, \dots, P_n , betrachte $P_i P_{i+1}$ einzeln
klassifizierte P_i, P_{i+1} als innen/aussen



Sichtbarkeit

Hacker-Algorithmus

sortiere Polygone nach Abstand zur Kamera

Problem: Zyklen

Tiefenpuffer, Z-Puffer

speichere für jeden Pixel Distanz zur nächsten Fläche