Mathematik Grundlagen

Vektor $\vec{x} = (x_1, x_2, ..., x_n)$

Ortsvektor $(x, y, z, 1)^T$
Richtungsvektor $(x, y, z, 0)^T$

Multiplikation $\alpha * \vec{x} = (\alpha * x_1, \alpha * x_2, ...)$
Addition $\vec{x} + \vec{r} = (x_1 + r_1, x_2 + r_2, ...)$
Linearkombination $\vec{o} = (\alpha * \vec{p}) + (\beta * \vec{q}) + (\gamma * \vec{r})$
Länge $\vec{p} = (x, y, z) : |\vec{p}| = \sqrt{x^2 + y^2 + z^2}$
Skalarprodukt $\vec{x} * \vec{r} = \sum_{i=0}^{n-1} x_i * r_i$
Winkel $\vec{a} * \vec{b} = |\vec{a}| * |\vec{b}| * cos(\phi)$ mit $cos(\phi) = \frac{\vec{a}*\vec{b}}{|\vec{a}|*|\vec{b}|}$
Vektorprodukt $\vec{a} \times \vec{b} = \begin{pmatrix} a_y b_z - a_z b_y \\ a_z b_x - a_x b_z \\ a_x b_y - a_y b_x \end{pmatrix}$
Ebenen $p = \vec{q} + \alpha * \vec{r} + \beta * \vec{s}$
Dreieck $\vec{A} + \alpha * (B - A) + \beta * (C - A)$
Strahlensatz $y'_P = \frac{y_P * z_e}{z_P}$
Kugeloberfläche $A_k = 4\pi r^2$

2D Transformation

Translation um den Vektor $\vec{t}$
Skalierung Stauchung oder Streckung
Spiegelung
- an x-Achse $S = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$
- an y-Achse $S = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}$
- am Ursprung $S = \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix}$

Scherung $S = \begin{pmatrix} 1 & S_x \\ S_y & 1 \end{pmatrix}$
Rotation mit Polarkoordinaten $P' = (r, \phi + \theta)$;
$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{pmatrix} * \begin{pmatrix} x \\ y \end{pmatrix}$
Koordinatentransformation $P \to P'$
$P' = T * P = \begin{pmatrix} x_x & x_y \\ y_x & y_y \end{pmatrix} * \begin{pmatrix} P_x \\ P_y \end{pmatrix}$

Kartesische Koordinaten bezeichnen die Koordinaten direkt
Homogene Koordinaten besitzen ein zusätzliches Gewicht w

Homogene Vektorräume

kartesischer Vektor $(\frac{x}{w}, \frac{y}{w})$
oft $w = 1$ gewählt (1=Punkt, 0=Richtung)

Skalierung, Projektion, Spiegelung
$\begin{pmatrix} F_x & 0 & 0 \\ 0 & F_y & 0 \\ 0 & 0 & 1 \end{pmatrix} * \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} F_x * x \\ F_y * y \\ 1 \end{pmatrix}$
$F_x, F_y > 0$, uniform bei $F_X = F_y$
$F_x = 0 / F_y = 0$:Projektion auf y/x-Achse
$F_x = -1 / F_y = -1$ Spiegelung an y/x-Achse
$F_x = F_y = -1$Spiegelung am Ursprung
Scherung $\begin{pmatrix} 1 & a & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} * \begin{pmatrix} x \\ y \\ w \end{pmatrix} = \begin{pmatrix} x + a * y \\ y \\ w \end{pmatrix}$
Rotation $R_\theta * P = \begin{pmatrix} cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix} * (x \quad y \quad 1) =$
$\begin{pmatrix} xcos(\theta) - ysind(\theta) \\ xsin(\theta) + ycos(\theta) \\ 1 \end{pmatrix}$

Invertierung

Transformation $T^{-1}_{\Delta x, \Delta y} = T_{-\Delta x, -\Delta y}$
Skalierung $S^{-1}_{F_x, F_y} = S_{\frac{1}{F_x}, \frac{1}{F_y}} = \begin{pmatrix} \frac{1}{F_x} & 0 & 0 \\ 0 & \frac{1}{F_y} & 0 \\ 0 & 0 & 1 \end{pmatrix}$
Rotation $R_{-\theta} = \begin{pmatrix} cos(\theta) & sin(\theta) & 0 \\ -sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix} = R^T_\theta$
Verknüpfungen $(A * B * C)^{-1} = C^{-1} * B^{-1} * A^{-1}$

Affine Abbildung

$$\begin{pmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ 0 & 0 & 1 \end{pmatrix} * \begin{pmatrix} x_1 \\ y_1 \\ 1 \end{pmatrix} = \begin{pmatrix} x'_1 \\ y'_1 \\ 1 \end{pmatrix}$$

- die letzte Zeile der affinen Matrix bleibt immer 0,0,1
- paralleles bleibt bei affinen Abbildungen stets parallel

Homogene Transformation in 3D

(a, b, c, d) wobei $(a, b, c) = (nx, ny, nz)$ und d der Abstand der Ebene zum Ursprung

- Ebene definiert durch 3 Punkte

$$\begin{pmatrix} x_1 & x_2 & x_3 & 0 \\ y_1 & y_2 & y_3 & 0 \\ z_1 & z_2 & z_3 & 0 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

- Translation um Vektor $(\Delta x, \Delta y, \Delta z)$

$$\begin{pmatrix} 1 & 0 & 0 & \Delta x \\ 0 & 1 & 0 & \Delta y \\ 0 & 0 & 1 & \Delta z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- Skalierung um Faktor F_x, F_y, F_z

$$\begin{pmatrix} F_y & 0 & 0 & 0 \\ 0 & F_y & 0 & 0 \\ 0 & 0 & F_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- Rotation um die x-Achse

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

- Rotation um die y-Achse

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

- Rotation um z-Achse

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

- Berechnung

$$\begin{pmatrix} a & b & c & 0 \\ d & e & f & 0 \\ g & h & i & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} * \begin{pmatrix} x \\ y \\ z \\ 0 \end{pmatrix} = \begin{pmatrix} ax + bx + cz \\ dx + ey + fz \\ gx + hy + iz \end{pmatrix}$$

Kameratransformation

Kamera ist definiert durch

- Lage des Augpunktes E (in Weltkoordinaten)
- Blickrichtung D
- Oben-Vektor U ('view up vector', senkrecht zu D)

Transformation durch

- Ursprung in Augpunkt
- negative z-Achse in Blickrichtung
- y-Achse nach oben

Projektion

Orthogonale Projektion

- Projektionsebene ist parallel zur XY Ebene
- Projektionsrichtung stets parallel zur z-Achse (rechtwinklig zur Projektionsebene)
- z Koordinaten werden auf gleichen Wert gesetzten

Schiefwinklige Parallelprojektion

- typische Parallelprojektion mit 2 Parametern
- Projektionsebene ist parallel zur XY Ebene
- Projektionsrichtung hat zwei Freiheitsgrade und ist typischerweise nicht orthogonal zur Projektionsebene
- Projektionsrichtung (Schiefe) ist über 2 Winkel parametrisierbar
- Herleitung $P = \begin{pmatrix} 1 & 0 & -cos(\alpha) * f & 0 \\ 0 & 1 & -sin(\alpha) * f & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$
- es gilt: $x' = x - cos(\alpha) * f * z$ und $y' = y - sin(\alpha) * f * z$

Zentralperspektive

- entspricht einer Lochkamera bzw etwa dem 'einäugigen' Sehen
- Augpunkt im Ursprung des Kamerakoordinatensystems
- Projektionsfläche ist eine Ebene parallel zu XY Ebene
- Eigenschaften
  - perspektivische Verkürzung
  - parallele Linien des Objekts fluchten oft in einen Fluchtpunkt
- Strahlensatz: $\frac{y_p}{d} = \frac{y}{z} \Rightarrow y_p = \frac{d*y}{z}$

$$\begin{pmatrix} d & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} * \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} d * x \\ d * y \\ 1 \\ z \end{pmatrix} \to \begin{pmatrix} \frac{d*x}{z} \\ \frac{d*y}{z} \\ \frac{1}{z} \\ \frac{z}{z} \end{pmatrix}$$

Fluchtpunkte

- Wird aus einer Richtung r und dem Augpunkt eine Gerade definiert, dann schneidet diese Gerade die Projektionsfläche im Fluchtpunkt für die Richtung r
- hat ein Modell parallele Kanten oder parallele Striche in Texturen, dann ergibt sich für jede solche Richtung r in der Abbildung ein Fluchtpunkt, auf den diese parallelen Kanten/Striche hinzu zu laufen scheinen
- es gibt jedoch Ausnahmen, bei denen Paralleles in der Abbildung Parallel bleibt (z.B. horizontale Kanten bei Schwellen)
- Da es beliebig viele Richtungen geben kann, sind auch beliebig viele Fluchtpunkte in einem Bild möglich
- Rotationen können Fluchtpunkte ändern, Translationen jedoch nicht
- Ermittlung: aus Richtung r und Augpunkt eine Gerade, dann schneidet diese Gerade die Projektionsfläche im Fliuchtpunkt für die Richtung r.

Modellierung

Boundary Representation (B-Rep)

- Beschreibung durch die begrenzende Oberflächen
- Darstellungsform eines Flächen- oder Volumenmodells
- Definition des Ojekts über vef-Graph (vertex, edge, face)
  - Knotenliste: beinhaltet Koordinatenpunkt
  - Kantenliste: pro Kante zwei Punkte referenziert
  - Flächenliste: pro Fläche die Reihenfolge der Kanten
- Szene: dreidimensionale Beschreibung von Objekten, Lichtquellen und Materialeigenschaften mit Betrachter
- Szenegraph: Gruppierung der Objekte in einer Szene

Vertex Shader

- verarbeitet alle Eckpunkte (Vertices) mit Shader
- ermöglicht eine Beeinflussung der Objektform
- Transformation der 3D Position auf 2D Koordinaten
- Input: Vertices relevanter Objekte und gewünschte Transformation
- Output: projizierte 2D Koordinaten mit Tiefeninformationen

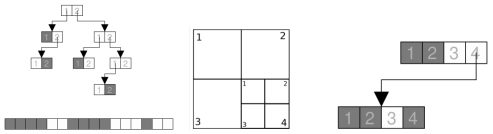
Model View Projection

- Gegeben
  - Modell als Vertices mit kartesischen 3D Koordinaten
  - betrachtende Kamera (3D Position, Ausrichtung)
- Umsetzung
  1. $M = T * R * S$ Transformation von Modellraum in Weltkoordinaten (Model)
  2. $V = T_V^{-1} * R_V^{-1}$ Transformation in Kameraraum (View)
  3. Projektion auf Kamerabildebene und Umrechnung in Bildraum (Projektion)
- Ergebnis
  - MVP-Matrix $P * V * M = MVP_{Matrix}$
  - Bildraumprojektion des Modells $p'_m = P * V * M * p_m$

Effiziente geometrische Datenstrukturen

Bintree

- logarithmische Komplexität pro Zugriff möglich
- Gefahr: lineare Komplexität, wenn nicht balanciert
- typisch Teilung in Mitte (bisektion)
- Bereiche mit homogenem Inhalt werden nicht unterteilt
- Komprimierungseffekt



Quadtree

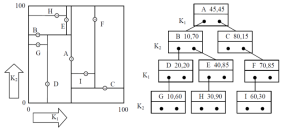
- eine Fläche wird in vier gleichgroße Quadranten unterteilt
- Fläche wird unterteilt bis homogenität
- Komprimierung, da nur strukturierte Bereiche unterteilt

Octree

- Objekte in hierarchische Strukturen einsortiert
- jeder Knoten hat 0 oder 8 Kindknoten (8 Unterbereiche)
- beschleunigte räumliche Suche
- Zeitaufwand Tiefe des Baumes $O(\log n)$

KD Tree

- mehrdimensionaler binärer Baum (k-dimensional)
- Teilung nicht zwangsläufig mittig → an Daten angepasst
- pro Hierarchiestufe stets wechsel der Teilungsrichtung
- Median-Cut Strategie: Teilung in zwei gleichgroße Hälften
  - Baum garantiert balanciert und Tiefe minimal
  - $O(\log n)$ Verhalten garantiert
  - Probleme bei lokalen Häufungen (Cluster)
  - unnötige Unterteilung weit weg (Artefakt)
- Middlecut-Strategie:
  - nicht balanciert
  - keine Unterteilung weit weg vom Cluster
- ein Octree lässt sich auf kd-Baum abbilden, beide Baumarten haben daher vergleichbare Eigenschaften



BSP Tree

- Verallgemeinerung des kd-Baums
- Trennebenen nicht nur achsenparallel
- Unterteilung adaptiv an Modellflächen angepasst
- Trennebenen können weiter weg liegende Objekte schneiden
- führt bei konvexen Polyedern zu entarteten Bäumen

Hüllkörper Hierarchie

AABB (Axis-Aligned-Bounding-Box) sehr einfache Abfrage (nur ein Vergleich < pro Koordinatenrichtung) einfach zu erstellen (min, max), dafür nicht optimale Packungsdichte bei schräger Lage der Objekte

OBB (Oriented Bounding Boxes) passen sich besser der räumlichen Ausrichtungen an, lassen sich auch leicht transformieren. Jedoch schwieriger zu erstellen (Wahl der Richtung), komplexere Überlappungsberechnung. Typischerweise weniger tief, weniger räumliche Abfragen dafür wesentlich mehr Berechnungsaufwand pro Rekursionsstufe.

KDOP (k-dim. Discretly Oriented Polytopes) Polyeder mit festen vorgegebenen Richtungen. Eigenschaften zwischen AABB und OBB. Bessere Raumausnützung als AABB, weniger Transformationene als OBB.

BS (Bounding Spheres) Schnelle 3D Überlappungstest (Abstand der Mittelpunkte < Summe der Radien). Langgezogene Objekte können mit mehreren Hüllkugeln begrenzt werden um besseren Füllgrad zu erreichen. BS sind, bis auf die Lage der Kugelmittelpunkte, invariant gegenüber Rotation (geeignet für Kollisionserkennung bewegter Objekte).

weitere Anwendungsfälle Kollisionserkennung in Computeranmiation. Reduktion der potenziellen Kollisionspaare durch räumliche Trennung. Beschleunigung des Echtzeitrenderings großer Datenmengen. Reduktion des Aufwands durch Culling (Weglassen)

Ray Picking mit KD Baum

- Vorverarbeitung von Objekten im kd-Baum $O(n \log n)$
- Strahl/Objektschnitt: als rekursive Suche im kd-Baum
- *treeIntersect*: Findet Schnittpunkt des Strahls mit den im Baum gespeicherten Dreiecken
- *triangleIntersect*: Findet Schnittpunkt des Strahles mit Menge von Dreiecken in node
- *subdivide*: Findet rekursiv den nächstgelegenen Schnittpunkt (kleinstes t) des Strahls im Parameterbereich

Aufwandsabschätzung bzgl Dreiecksanzahl

- konvexes Objekt: Komplexität einer räumlichen Punktsuche, also zur Untersuchung einer Baumzelle $O(\log n)$
- Polygonnebel: viele kleine Dreiecke im Such-Volumen
- alle Zellen enthalten konstante kleine Anzahl von Dreiecken → Aufwand proportional zur Anzahl durchlaufener Baumzellen
- Anzahl dieser Zellen ist proportional zur Länge des Strahls durchs Volumen, da der 1. Schnitt sehr wahrscheinlich mitten im Volumen oder gar nicht stattfindet → Anzahl ist proportional zur Seitenlänge des Suchvolumens
- bei n Dreiecken im Suchvolumen ist die Anzahl t der zu untersuchenden Zellen also ca $t = O(\sqrt{n})$ → Suchaufwand pro Strahl folglich $O(\sqrt{n}\log(n))$

Aufwandsabschätzung in fps

- absoluter Gesamtaufwand zum Raytracing einer Szene ist proportional zur Anzahl der Strahlen
- rekursives RT (Reflexion, Brechung, Schattenstrahlen etc) entsprechend mehr Strahlen, d.h. weniger Performance
- Parallelisierung einfach möglich → früher CPU, heute GPU

Heurisitk zur Unterteilung

- Surface Area Heuristic (SAH):
  - Strahl i trifft Zelle j mit Wahrscheinlichkeit $P(i,j)$, zudem sei n_j die Anzahl Dreiecke in Zelle j
  - Aufwand für Raytracing pro Zelle proportional zur Baumtiefe und Anzahl der dortigen Dreiecke n_j;→ Gesamtaufwand für Strahl i sei $\sum(P(i,j) * n_j)$
- große Zellen mit wenigen Dreiecken senken Gesamtaufwand
- $P(i,j)$ ist proportional zur Oberfläche einer Zelle
- SAH optimiert das Produkt der Zellgröße mal Anzahl Dreiecke im Teilbaum. Für den kD-Baum in Richtung k: $D_k = D_{k_{links}} + D_{k_{rechts}}$
- bei ungleicher Verteilung der Dreiecke enthalten große Zellen wenige oder keine Dreiecke und Baum ist nicht balanciert → implizite Abtrennung des Clusters vom Rest des Baums (vgl. Middle-Cut-Strategie)

Behandlung ausgedehnter Objekte

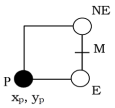
- Punkte haben keine Ausdehnung und können an einem eindeutigen Ort im kD-Baum abgelegt sein
- Ausgedehnte Objekte können räumlich mehrere Blatt-Zellen überlappen. Diese Objekte müssen dann in mehreren Blattzellen einsortiert werden

1. Auftrennung von Objekten, d.h. Objekte müssen an der Zellgrenze aufgeteilt werden. Einsortierung der Teilobjekte in passende Zellen. Geht gut für Dreiecke
2. Keine Unterscheidung zwischen Blattknoten und inneren Knoten. In diesem Ansatz werden Objekte soweit oben im Baum einsortiert, dass sie keine Zellgrenze schneiden. Nachteil: auch relativ kleine Objekte müssen in große Zellen einsortiert werden, wenn sie deren Unterteilungsgrenze schneiden
3. Loose Octree: die Zellen des Octrees werden so vergrößert, dass sie mit ihren direkten Nachbarn in jeder Richtung um 50% überlappen. Objekte, die im einfachen Octree aufgrund ihrer Größe Grenzen schneiden würden, können im Loose Octree in den Zwischenknoten gespeichert werden. Ein Objekt mit Durchmesser bis zu $\frac{D}{2^L}$ kann auf der Ebene L abgelegt werden. Eine Suche im Loose Octree muss daher außer der direkt betroffenen Zelle auch die überlappenden direkten Nachbarn berücksichtigen. Dadurch vergrößert sich der Aufwand einer Suche um einen konstantne Faktor. Beachte: Die asymptotosche Komplexität (O-Notation) ist dadurch nicht beeinflusst.

Rastergrafik

Midpoint Algorithmus

- Effizient durch Ganzzahlen, Vermeiden von $*, /$,
- Nutzung inkrementeller Arbeitsweise
- Bresenham-Algorithmus: Mittelpunkt M; jeweils aktuellen Punkt P, der rechts von im liegende E (east) und der rechts oben liegende NE (north-east) benannt.
- die Linie wird als Funktion repräsentiert: $y = \frac{\delta y}{\delta x} * x + B$
- implizierte Form: $d : F(x, y) = \delta y * x - \delta x * y + B * \delta x = 0$
- für Punkte auf der Linie wird $F(x, y) = 0$
- für Punkte unterhalb der Linie wird $F(x, y) > 0$
- für Punkte oberhalb der Linie wird $F(x, y) < 0$
- Herleitung: Steigung der Linie m $(-1 < m < 1)$, Punkt vertikal zwischen zwei Pixeln E und NE. Falls $F(x_p + 1, y_p + \frac{1}{2}) > 0$ wird das nächste Pixel NE, andernfalls E
- Insgesamt acht verschiedene Fälle nach Oktanten



Anti Aliasing

- Treppenstufeneffekt bei gerasterten Linien
- Auflösungsvermögen des Auges für Punkte sei e. Strukturen wie Linien werden durch Mittelwertbildung (Fitting) vom Auge viel genauer als e lokalisiert. Eine Stufe wird umso eher erkannt, je länger die angrenzenden Segmente sind.
  - Statt Linie wird Rechteck mit Breite eines Pixels betrachtet
  - Graustufen darunter liegender Pixelflächen entsprechen jew. Überdeckungsgrad
- Praktische vereinfachte/effiziente Umsetzung
  - Rasterkonvertierung der Linie bei doppelter örtlicher Auflösung (Supersampling)
  - Replizieren der Linie (vertikal und/oder horizontal) um Linienbreite näherungsweise zu erhalten
  - Bestimmmung des Überdeckungsgrades pro Pixel in der ursprünglichen Auflösung (Downsampling)
  - Bestimmung des Farbwertes entsprechend des Überdeckungsgrades
- Ideales Antialiasing: wegen beliebig komplexen Geometrie allgemein sehr hoher Aufwand
- Ansatz für eine 'reale Lösung'
  - ideale Berechnung von Farbwerten irrelevant
  - Ansätze mit gut abschätzbarem/konstanten Aufwand
  - Verwendung mehrerer Samples pro Pixel
- A.A. erhöht empfundene räumlich Auflösung

Supersampling + Downsampling

- Grafik in höherer Auflösung gerendert (z.B. 4-fach) und aus Samples ein Farbwert gemittelt
- Ohne A.A. pro Pixel eine Sampleposition⇒ gefärbt o. nicht
- Es gibt immer eine Abstufung mehr als Subpixel pro Pixel
- Bei vier Subpixeln können 0-4 Subpixel im Pixel gesetzt sein, d.h. 5 Intensitäten von 0%, 25%, 50%, 75% oder 100%
- bei Formabhängigkeit gibt es nur eine Zwischenstufe nach Phasenlage → Kante 'pumpt' bei Objektbewegung.
- $Pixelfarbe = g * Linienfarbe + (1 - g) * Hintergrundfarbe$

Supersampling + Rotated Grids

- minderung der Formabhängigkeit
- kleine Winkel führen zu langen Stufen der Polygonkante
- bessere Verhältnisse der Grauabstufung für flache Winkel, wenn ordered-grid statt rotated-grid verwendet wird
- Rotated grids bei anderen Winkeln etwas schlechter als ordered grid
- gute Grauabstufung bei sehr flachen Kanten
- optimaler Winkel bei ca. 20-30° z.B. $arctan(0.5) \approx 26,6°$
- sehr dünne Linien bleiben auch bei Bewegung zusammenhängend (Vermeidung von 'Line Popping')

Supersampling + Multisampling

- ein Superbackpuffer (großem Buffer)
  - Nachteil (bei rotated grid): Anpassung der Rasterkonvertierung an verschobene Positionen
  - Vorteil: Verwendung von mehr Texturinformation (Textur wird subpixelgerecht eingetragen)
- mehrere Multisamplebuffer (mehrere kleinere Buffer)
  - Mehrfachrendering in normaler Größe mit versetzter Geometrie (Vertexverschiebung pro Sub-Bild)
  - Vorteil: keine Veränderung im Rendering
  - Nachteil: nur ein Texturwert pro Makro-/Sub-Pixel
- Gezielter Ressourceneinsatz durch Kantenglättung
  - Effizienzsteigerung durch Beschränkung auf reine Kantenglättung möglich
  - Aliasing bei Mustern in Texturen schon beim Auslesen der Werte aus Pixeltextur unterdrückbar
  - Kantenpixel bekannt als separate Linien oder Berandung von Polygonen/Dreiecken
- adaptives Samplen: statt feste Anzahl nach dem Bedarf

Quincunx Verfahren

- 2x Multisampling mit rotated grid; Informationszuwachs durch doppelte Anzahl von Samples
- Information für Kantenglättung beruht auf 2 Subpixeln
- Entspricht zusätzlicher Tiefpass-Überfilterung. Durch Unschärfe sehen Polygonkanten glatter aus.
- Harte Kanten nicht mehr möglich; Rand 'Zappeln' reduziert
- Aber: Texturinformation, von 2>Subpixeln, verschmiert

Pseudozufälliges Supersampling

- Kombination: Supersampling, Multisampling und Quincunx
- bei Überwindung der Grenzen für Füllrate und Bandbreite überwiegen Vorteile des Supersamplings
- Ordered/rotated grid weisen nach Strukturklassen Vor-/Nachteile auf. Verbleibende Artefakte wiederholen sich bei großen Flächen - oft als störend empfunden
- pszufällige Auswahl von Abtastmustern für Supersampling
- nachträgliche Abminderung regelmäßiger Strukturen durch vorsichtiges Verrauschen (Rauschfilter)
- entfernungsabhängiges Antialiasing
- pseudozufällig
  - Samples können nur an n vordefinierten Positionen stattfinden (Sample-Positionsmuster)
  - Je nach Methode werden daraus m Positionen für das Samplen zufällig ausgewählt (beachte: $m < n$)
  - Anzahl der Muster als kombinatorisches Problem: m aus n (ohne Wiederholungen)

Downsampling Mittelwertbildung: lineare Filterung (2x - AA), bilineare Filterung (4x - AA). Gleichgültig ob ordered/rotated grid. Beim pseudozufälligen Supersampling ist entsprechend der 'frei gewählten' Positionen der 'Subpixel' zu modifizieren (z.B. Gewichte nach Abstand der Abfragepositionen zur Makropixelposition)

Polygonfüllalgorithmus

- Ansatz
  - finde die Pixel innerhalb des Polygons
  - weise ihnen Farbe zu
  - dabei zeilenweises Vorgehen pro Rasterlinie
  - schneide die Polygonkante mit der aktuellen Bildzeile
  - füge Schnittpunkt x_s in eine Liste ein
  - sortiere Schnittpunkte der Bildzeile in x-Richtung
  - Paritätsregel: fülle die Pixel jeweils nur zwischen ungeraden und nächstem geraden Schnittpunkt
- Allgemeine Sicht auf die Strategie: Ein Pixel wird mit der Farbe des Polygons gefüllt, das sich rechts von ihm befindet. Sollte dort eine Kante sein, so wird die Farbe des oberen Polygons verwendet.
- Effiziente Ermittlung der Schnittpunkte
  - Polygonkanten von unten nach oben bearbeitet
  - horizontale Polygonkanten nicht bearbeiten → $m \neq 0$
  - $d_y = y_1 - y_0$ ist stets positiv (auch nie 0)
  - $d_x = x_1 - x_0$ kann positiv und negativ sein
  - damit können 4 Bereiche unterschieden werden
  - Berechnung von x bzw y:
    - $y = y_0 + m(x - x_0) = y_0 + \frac{y_1 - y_0}{x_1 - x_0}(x - x_0)$,
    - $x = x_0 + \frac{1}{m}(y - y_0) = x_0 + \frac{x_1 - x_0}{y_1 - y_0}(y - y_0)$
  - x-/y-Werte noch nicht ganzzahlig
  - Die Rundung kann inkrementell ermittelt werden
  - Die Rundungsregel für Bruchwerte hängt davon ab, ob es eine linke oder rechte Kante ist. Links wird z.B. aufgerundet
- Edge-Tabelle
  - Verkettete Liste/Array für nicht-horizontalen Kanten
  - Sortierung nach Scan-Line, wo Kanten beginnen
  - In Scan-Line wieder Liste mit z.B. x_0, y_1, Zähler
- Active-Edge-Tabelle
  - speichert Kanten die gegenwärtige Scan-Linie schneiden
  - Liste hat die gleiche Struktur wie eine Zeile der ET
  - Kanten gelöscht wenn oberes Ende der Kante erreicht
- Scan Convert Polygon: Es existiert immer eine gerade Anzahl Kanten. Viele Grafikbibliotheken beschränkt auf konvexe Polygone. Nichtkonvexe Polygone müssen vorher in konvexe Komponenten zerlegt werden.
- Bemerkungen zur Effizienz: Polygon belegt meist mehr Pixel als es Eckpunkte/Kanten besitzt. Deshalb sind effiziente per-Pixel-Operationen wichtig. Der Rechenaufwand sollte vermieden werden (fallende Priorität) für: pro Pixel (sehr häufig auszuführen), pro Rasterzeile, pro Kante (möglichst viel vorberechnen)

Füllmuster

- Füllen eines Polygons mit Pattern statt Farbwert
- benutze dazu BITMAPs
- 2-dimensionales Array
- besteht aus M Spalten und N Zeilen
- $BITMAP = ARRAY[0...(M-1), 0...(N-1)]$

Dithering - Floyd-Steinberg-Algorithmus

- Ersetzen 'genauer' Farbwerte durch grobe Quantisierung
- Durchlaufen aller Pixel beginnend links oben
- pro Pixel P die beste Ersetzung in Tabelle finden & setzen
- verursachte Fehler δ nach Schema auf unbearbeitete Nachbarpixel verteilen
- bei kleinen Bildern mit hoher Auflösung kaum erkennbar
- erhöht Farbauflösung → Verringert räumlichen Auflösung
- komplementär zu A.A.

Farbräume

Farbwahrnehmung - Phänonmenologie

- Hell- und Farbempfinden als Sinneseindruck beschrieben
- Tageslicht als weiß/grau mit unterschiedlichen Helligkeiten aber farblos empfunden
- Abwesenheit von Licht wird als schwarz empfunden
- Regenbogen bunt mit verschiedenen Farbtönen empfunden

Farbton (Hue)

- Farbpalette aus Abstufung grober Farbtöne
- direkt nebeneinander liegende Farben im Farbspektrum werden als ähnlich empfunden
- Farbwerte lassen sich ordnen
- als bunt empfunden (voll gesättigte Farben im Gegensatz zu Grautönen)

Farbsättigung (Saturation)

- Stufen zwischen Bunt und Grau
- Pastelltöne sind weniger bunt aber nicht farblos
- Grauton (keine Farbwerte unterscheidbar)
- jedem Farbton können Abstufungen bis Grau zugeordnet werden

Helligkeitsstufen (Lightness/Brightness/Value/Intensity)

- unterschiedliche Helligkeitsabstufungen bis Schwarz
- im Schwarzen sind keine Farbtöne mehr unterscheidbar

Modell der Farben

DIN 5033 Farbe ist die Empfindung eines dem Auge strukturlos erscheinenden Teils eines Gesichtsfeldes, durch die sich dieser Teil bei einäugiger Beobachtung mit unbewegtem Auge von einem gleichzeitig gesehenem, ebenfalls strukturlos angrenzendem Bezirk allein unterscheidet.

HSL Farbraum (bzw HSB, HSV, HSI)

- Dimension des Farbtons wiederholt sich periodisch
- Darstellung als Winkelkoordinate eines Polarkoordinaten-Systems in der HS-Ebene oder dreidimensional als Zylinderkoordinaten HSL darstellt.
- Darstellungsformen nicht fest vorgeschrieben. Eine Darstellung als (Doppel)-Kegel oder sechseitige (Doppel-) Pyramide ist ebenso möglich
- Der HSL Farbraum entspricht grob unserer Farbwahrnehmung. Daher geeignet zur intuitiven und qualitativen Einstellung von Farben in Illustrationsgrafiken
- Relative Skala 0-255
- Quantisierbarkeit der Farben und Helligkeit
- Bezug zur Physik des Lichtes (Energie, Spektrum)

RGB Farbraum

- Hypothese, dass Farbsehen auf drei Arten von Sinneszellen beruht (rot, grün, blau) (Young)
- Farbwahrnehmungen durch drei beliebige, linear unabhängige Größen darstellbar (Graßmann)
- Mit Grundfarben Rot, Grün und Blau können weitere Farben additiv gemischt werden
- Bestimmen der Anteile der Mischfarben
  - Empfindlichkeitskurven R,G,B und zugehörige Lichtquellen r,g,b
  - alle 3 Lichtquellen zusammen ergeben weis wahrgenommenes Licht: $r = g = b = 1$
  - damit 3d-Farbraum (RGB-Farbraum) aufspannen
  - Lage einer monochromatischen Lichtwelle: $x(\lambda_0) = p * r + \gamma * g + \beta * b$
  - Achtung: hängt von Wellenlängen der verwendeten Grundfarben r,g,b (Primärvalenzen) ab.
- Beispiel für Reizung durch monochromatisches Licht (Laser):
  - $r = 0,2R(\lambda)$
  - $y = 0,5R(\lambda) + 0,3G(\lambda)$
  - $g = 0,2R(\lambda) + 0,5G(\lambda)$
  - $b = 0,02B(\lambda)$
- Intensität: $I = \frac{R+G+B}{3}$
- Innere Farbmischung: mischen direkt aus Grundfarben
- Äußere Farbmischung: hinzufügen von Grundfarben zu bestehender Mischung

Farberzeugung durch Mischung:

$$y = 1,9r + 0,6g = 0,5R(\lambda) + 0,3G(\lambda)$$

Idee:

- drei linear-unabhängige Größen benötigt, zur Beschreibung und (technischen) Reproduktion der Farbempfindung
- zunächst werden folgende Werte gewertet
  - die additive Mischung als Reproduktionsmethode
  - drei Primärfarben Rot, Grün, Blau
  - drei linear unabhängige Größen spannen stets einen 3D Raum auf
- die RGB Werte werden den drei ortogonalen Achsen dieses Raumes zugeordnet

Darstellung des RGB Farbraums:

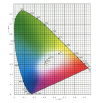
- alle technisch/additiv erzeugbaren Farben liegen innerhalb eines Würfels
- Im Koordinatenursprung befindet sich Schwarz
- auf der Raumdiagonalen liegen dazwischen die Graustufen

RGB Farbtafel:
Alle Farben gleicher Buntheit führen zum gleichen Farbort, der durch die Farbwertanteile r,g,b beschrieben wird:

$$r = \frac{R}{R+G+B}, g = \frac{G}{R+G+B}, b = \frac{B}{R+G+B} \leftrightarrow r + g + b = 1$$

Aus dem rechten Teil der Gleichung folgt mit $b = 1 - r - g$, dass sich die Buntheit allein durch r und g darstellen lässt (entspricht R^2).

CIE System



Um eine Relation zwischen der menschlichen Farbwahrnehmung und den physikalischen Ursachen des Farbreizes herzustellen, wurde das CIE-Normvalenzsystem definiert. Es stellt die Gesammtheit der wahrnehmbaren Farben dar.

Farbkörperunterschiede

Es finden sich Unterschiede welche Farbbereiche nach dem CIE Normalvalenzsystem von den jeweiligen Systemen dargestellt werden können:

- menschliche Farbwahrnehmung ca. 2-6 Mio Farben
- Monitor ca. 1/3 davon. Bei Monitoren wird die additive Farbmischung verwendet, da die einzelnen Lichtquellen aufsummiert werden.
- Druckprozess deutlich weniger Farben. Es werden einzelne Farbschichten auf Papier gedruckt und das resultierende Bild wird über subtraktive Farbmischung bestimmt

Subtraktive Farbmischung

Je nachdem welche Farbe ein Material hat, werden entsprechende Farbanteile absorbiert oder reflektiert. Eine gelbe Oberfläche sieht gelb aus, da sie das Blau aus weißem Licht absorbiert, aber Rot und Grün reflektiert.
Achtung: Dies gilt nur für die Bestrahlung mit weißem Licht. Wird beispielsweise ein gelbes Blatt mit blauem Licht bestrahlt, dann wirkt es schwarz, da das blaue Licht vom gelben Blatt absorbiert wird.

Umwandlung

Von einem Farbort im XYZ-System $F_P = (XYZ)^T$ in das equivalente CIE Diagramm $F'_P = (X'Y')^T$ durch

$$F'_P = \begin{pmatrix} X' \\ Y' \end{pmatrix} = \begin{pmatrix} \frac{X}{X+Y+Z} \\ \frac{Y}{X+Y+Z} \end{pmatrix}$$

Farbraum	gerätespezifisch	gleichabständig
RGB	ja	nein
CMY	ja	nein
HSI	ja	nein
HLS	ja	nein
XYZ (CIE)	nein	nein
$L * a * b* (CIE_{Lab})$	nein	ja

Licht & Reflexion

Licht Teil der elektromagnetischen Strahlung
Photon Elementarteilchen der elektromagnetischen Wechselwirkung
Radiometrie Messung elektromagnetischer Strahlung
Photometrie Messverfahren im Wellenlängenbereich
Strahlungsäquivalent $K = \frac{\phi_v}{\phi_e}$
Lumen 1 Lumen ist der Lichtstrom einer 1,464 mW starken 555-nm-Lichtquelle mit 100% Lichtausbeute
Raumwinkel $\Omega = \frac{Flaeche}{Radius} = \frac{A}{r^2}[sr]$

Radiometrie (energetisch "$_e$")

Q Strahlungsenergie $Q = \#Photonen * Photonenenergie[J]$
Φ Strahlungsleistung $\Phi = \frac{Q}{t}[W]$
I Strahlstärke $I = \frac{\Phi}{\Omega}[\frac{w}{sr}]$
E Bestrahlungsstärke $E = \frac{\Phi}{A_i}[\frac{W}{m^2}]$
L Strahldichte $L = \frac{I}{A'_r} = \frac{\Phi}{cos(\phi * A_r * \Omega)}$

Photometrie (visuell "$_v$")

Q Lichtmenge $lm * s$
Φ Lichtstrom $[Lumen]$
I Lichtstärke $[Candela]$ cd
E Beleuchtungsstärke $\frac{lm}{m^2} = I_{in}\cos(\Phi)[Lux]$
L Leuchtdichte $\frac{cd}{m^2}$

Lichtquellen

Ambient Licht strahlt gleichmäßig aus jeder beliebigen Richtung

Parallel Licht strahlt in gleichmäßigen, parallelen Strahlen aus einer festgelegtenRichtung

Punkt Licht wird gleichmäßig in alle Richtungen abgestrahlt

Spot Licht wird in den Halbraum abgestrahlt, wobei das meiste Licht in eineVorzugsrichtung abgegeben wird

Räumliche Ausbreitung

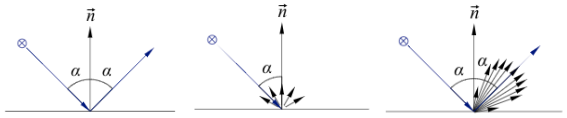
Flächen Energieübertragung

- der Abstand zwischen den beiden Flächen beträgt r
- die Flächen stehen nicht notwendigerweise senkrecht zur Ausbreitungsrichtung des Lichts
- abstrahlende und empfangende Fläche jeweils in Ausbreitungsrichtung mit projizierten Flächen A'_r und A'_i.
- Punktlichtquellen von der abstrahlenden Fläche A_r, welche ihre Strahlungsleistung in den Raumwinkel Ω abgeben
- Ω ist somit die in Abstrahlrichtung reduzierte Fläche A'_i , projiziert auf die Einheitskugel: $\Omega = A'_i \backslash r^2$
- Die übertragene Energie nimmt quadratisch zu r ab

Reflexion

Nach Auftreffen auf einer opaken Oberfläche wird Strahlung spektral unterschiedlich stark und geometrisch auf unterschiedliche Weise reflektiert. Fälle der Reflexion:

- spekulär (spiegelnde) Reflexion (Einfallsw.=Ausfallswinkel)
- Diffuse Reflexion
- Diffus-gerichtete Reflexion



Diffuse Reflexion

Eingestrahlte Strahlstärke verteilt sich durch Projektion auf größere Fläche. Die Bestrahlungsstärke ist dadurch proportional zum Vergrößerungsfaktor der Fläche abgeschwächt. In Richtung Betrachter reflektierte Strahlstärke I_{out} Aufgrund von Interferenz phasengleicher Lichtstrahlen → Projektion auf Normalenrichtung $I_{out} = E_{refl} * \cos(\phi)$

- Senkrecht zur Oberfläche: Maximale Kohärenz (Addition)
- Parallel zur Oberfläche: Keine Kohärenz (Auslöschung)

$$\frac{A_r}{A'_r} = \frac{1}{\cos(\phi)} \rightarrow L = \frac{I_{out}}{\cos(\phi)} = I_{refl}$$

Ein Betrachter mit flachem Blickwinkel sieht Licht aus größerer Fläche A_r durch Kombination dieser Effekte, kürzt sich der Einfluss des Betrachterwinkels $\cos(\phi)$ weg und es bleibt nur der Einfluss des Lichteinfallswinkels übrig: Strahldichte des reflektierten Lichtes: $L = I_{in} * k_d(\lambda) * \cos(\phi)$

Spekuläre Reflexion

(gestreut spiegelnd)

- Speckles bzw. Facetten sind einzeln jeweils 'ideal'
- spiegelnd: Einfallswinkel $\phi = \neg$Ausfallswinkel $= -\phi$
- Microfacettenausrichtung weichen von Gesamtflächennormalen ab
- dadurch Streuung des Lichts (Keule) um den Winkel θ der idealen Spiegelung herum
- Je größer der Winkel θ zwischen idealer Spiegelrichtung und Richtung zum Betrachter, desto schwächer ist die Reflexion
- Modellierung meist per $\cos^k(\theta)$ (Phong-Modell)

Gestreute Spiegelung im Phong Modell mit $L = I * k_s * \cos^k(\theta)$

- glänzende Fläche: großer Exponent k; kleine Streuung ϵ
- matte Fläche: kleiner Exponent k; große Streuung ϵ

Für Energieerhaltung zusätzlicher Normierungsfaktor benötigt:

- physikalisch nicht korrekt: $L = I * k_s * \cos^k(\theta)$
- gebräuchliche Normierung $L = I * k_s * \frac{k+2}{2\pi} * cos^k(\theta)$

Remittierende Flächen

ideal diffus remittierende weiße Flächen ($\beta(\lambda) = 1$):

- von Quellen in Fläche dA eingetragene Leistung führt zu Bestrahlungsstärke E_λ
- bei vollständiger Reflexion $\beta(\lambda) = 1$ ist $E_\lambda = R_\lambda$
- zugehörige Strahlungsfluss $d\phi = R_\lambda * dA = E_\lambda * dA$ wird bei ideal diffusen streuenden Oberflächen gleichmäßig über den Halbraum verteilt, wobei die Strahldichte (Lambertsches Gesetz) konstant ist.

BRDF: Bidirektionale Reflexionsverteilung

- Funktion für das Reflexionsverhalten von Oberflächen eines Materials unter beliebigen Einfallswinkeln
- nach gewählter Genauigkeit sehr komplex
- $f_r(\omega_i, \omega_r) = \frac{dL_r(\omega_r)}{dE_i(\omega_i)} = \frac{dL_r(\omega_r)}{L_i(\omega_i)\cos(\theta_i)d\omega_i}$
- BRDF beschreibt wie gegebene Oberfläche Licht reflektiert.
- $p(\lambda) = \frac{L_r}{E_i} = [\frac{1}{sr}]$
- BRDF ist 5-dim skalare Funktion: $p(\lambda, \phi_e, \theta_e, \phi_i, \theta_i)$
- Reziprozität: $\rho(\lambda)$ ändert sich nicht, wenn Einfalls- und Ausfallsrichtung vertauscht werden
- $\rho(\lambda)$ kann anisotrop sein, d.h. der Anteil des reflektierten Lichtes ändert sich, wenn bei gleicher Einfalls- und Ausfallsrichtung die Fläche um die Normale gedreht wird
- Superposition gilt ⇒ mehrere Quellen überlagern sich linear

Für Menge Q von Lichtquellen gesamte reflektierte Strahlstärke: $L_r = p_a * E_a + \sum_{1 \le j \le Q} E_j * (k_d * p_d + k_s * p_s)$ mit $k_d + k_s = 1$

Rendering-Equation

Für ambiente und gerichtete Lichtquellen aus der Hemisphäre:
$L_r = p_a + \int_{Omega} L * (k_d * p_d + k_s * p_s)\omega_i * nd\Omega$

Strahlungsquellenarten

Ambiente Strahlung
- stark vereinfachtes Modell für Streuung der Atmosphäre
- Strahlung kommt von allen Seiten
- keine Abhängigkeit von Winkeln und Entfernungen
- Beschreibung indirekt durch konst. Bestrahlstärke
- $E = \frac{\Phi}{A} = E_a$

Parallele Strahlung
- Strahlung ist gerichtet und parallel
- Richtung und Strahlungsleistung auf senkrecht zur Ausbreitungsrichtung stehende Fläche $R = E_q = \frac{\Phi}{A_q}$
- für Schattierungsrechnung lässt sich Bestrahlungsstärke der Oberfläche berechnen: $E = \frac{\Phi}{A} = \frac{E_q * A_q}{A} = E_q * \cos(\phi) = E_q * V_I^T * n$

Ideale Punktlichtquelle
- Ort bekannt und Strahlstärke in alle Richtungen konstant $I = \frac{\Phi}{\Omega} = konstant$
- Bestrahlungsstärke eines physikalischen vorliegenden, beliebig orientierten Flächenelementes A: $E = \frac{\Phi}{A} = \frac{I * \Omega}{A}, \Omega = \frac{A}{r^2} * \cos(\phi) * \omega_r \rightarrow E = \frac{I}{r^2} * \cos(\phi) * \omega_r$
- für Adaptionsfähigkeit des Auges oft $E = \frac{I}{c_1 + c_2 * |r| + c_3 * r^2} * \cos(\phi) * \omega_r$

Remittierende Flächen von reflektierenden Fläche weitergegebenen Strahldichte L sind Bestrahlungsstärken E für unterschiedlichen Quellen mit Faktor $\frac{\beta(\lambda)}{\pi\omega_r}$ bewerten

Quelle	Ref.	Spektale Strahldichte $L(\lambda)$
ambient	diffus	$L(\lambda) = \frac{E(\lambda)}{\pi\omega_r} * \beta(\lambda)$
gerichtet	diffus	$L(\lambda) = \frac{E(\lambda)}{\pi\omega_r} * \cos(\phi) * \beta(\lambda)$
punktförmig	diffus	$L(\lambda) = \frac{I(\lambda)}{\pi r^2} * \cos(\phi) * \beta(\lambda)$
gerichtet diffus	diffus	$L(\lambda) = \frac{I(\lambda)}{\pi r^2} * \cos^m(\theta) * \cos(\phi) * \beta(\lambda)$

Beleuchtungsmodelle

Lokale simulieren Verhalten von Licht auf einzelnen Materialoberflächen; nur Beleuchtungseffekte die direkt durch Lichtquellen auf einzelnen Objekt entstehen
Global simulieren Ausbreitung von Licht innerhalb der Szene; dabei wird Wechselwirkung in der Szene beachtet (Schatttenwurf, Spiegelung, indirekte Beleuchtung)

Phong-Modell

- lokales Beleuchtungsmodell
- zur Darstellung von glatten, plastikähnlichen Oberflächen
- widerspricht dem Energieerhaltungssatz
- Allgemein: $L = I_{out} = I_{ambient} + I_{diffus} + I_{specular}$
- Ambiente: $I_{ambient} = I_a * k_a$
- Diffus: $I_{diffus} = I_{in} * k_d * \cos(\phi)$
- Spiegelnd: $I_{specular} = I_{in} * k_s * \frac{n+2}{2\pi} * \cos^n(\theta)$
  - I Lichtstärke/Intensität der Lichtquelle
  - k_a Materialkonstante
  - $k_{d/s}$ empirischem Reflexionsfaktor
  - ϕ Winkel: Oberflächennormale - Richtung Lichtstrahl
  - θ Winkel: ideale Reflexionsrichtung - Blickrichtung
  - n konstante Exponent zur Beschreibung der Oberflächenbeschaffenheit
- $\frac{n+2}{2\pi}$ Normalisierungsfaktor zur Helligkeitsregulierung
- $I_{out} = I_a * k_a + I_{in} * k_d * \cos(\phi) + I_{in} * k_s * \frac{n+2}{2\pi} * \cos^n(\theta)$
- $\cos(\phi) = V_I^T * n_i$, $cos^n(\theta) = (V_r^T * V_e)^n$

Cook-Torrance

- Lichtstreuung um Winkel der idealen Spiegelung
- Berücksichtigt auch die gegenseitigen Abschattung
- Vollständig physikbasiertes Modell, spekulare Reflexion
- Aufwendige Berechnung
- Beckmann-Verteilung: $l_{spec} = \frac{exp(-\frac{tan^2(\alpha)}{m^2})}{\pi m^2 cos^4(\alpha)}$ mit $\alpha = arccos(N * H)$

Schattierungsverfahren

Direkte Schattierung

- Zerlegung gekrümmter Flächen in Polygone
- Positionen der (Eck-)Punkte und Normalen im 3D sowie der Punkte im 2D-Bild sind bekannt
- Pixelpositionen für Polygone im Bild per Scanline-Alg.
- lokale Beleuchtungsmodelle für 3D-Punkte

Flat-Shading Arbeitsweise

- eine Berechnung, gleiche Farbe für alle Pixel des Polygons
- stets nur 1 Farbwert pro (ebener) Fläche,
- Stelle der Berechnung frei wählbar (mögl. repräsentativ),
- z.B. Punkt (Ort mit Normale) in der Mitte der Fläche

Auswirkungen

- 'flaches' Aussehen und Helligkeitssprünge an Kanten
- gut für abstraktere technische Darstellungen
- wichtig für realistische Darstellung kantiger Körper
- schneller als die anderen Verfahren,
- d.h. nur ca. 1 Pixel pro Polygon gerendert wird ($n == 1$)

Gouraud-Shading [H. Gouraud 1971]

- pro Eckpunkt eine Farbe berechnen, dann lineare Interpolation (pro Dreieck) für jedes Pixel
- schattiert Dreiecke kontinuierlich,
- beseitigt die Diskontinuitäten des Flat-Shadings,
- meist gleiche Normalen pro Vertex (pro Dreieck wirken 3 verschiedene Richtungsvektoren)
- lineare Interpolation der Schattierung (Intensitäten) im Inneren des Dreiecks aus den 3 Farbwerten der Eckpunkte
- Normalenvektoren n_i für jeden Eckpunkt P_i des Polygons
- Herleitung der 'Normalenvektoren' n_i aus der Originaloberfläche oder Nachbarpolygonen
- jeder Eckpunkt: Berechnung der Beleuchtungsintensität I_i
- Normalen n_i der Eckpunkte direkt aus Flächen oder aus Flächennormalen benachbarter Polygone durch flächengewichtete Mittelung
- Die Schattierungsrechnung erfolgt für Eckpunkte und liefert reflektierte Leuchtdichte I_i
- Bei der Rasterkonvertierung wird zwischen Eckwerte I_i linear interpoliert und damit die Intensität jedes Pixels der Rasterlinie berechnet
- Interpolation erfolgt nach gleichen arithmetischen Muster wie Interpolation der x-Werte beim Polygonfüllalgorithmus
- Für farbige Oberflächen werden die Leuchtdichten an Polygonecken durch RGB-Werte beschrieben und zwischen den Ecken linear interpoliert
- Resultat: Kontinuierlich schattierte 3-dim Oberflächen

Artefakte des Gouraud-Shading, durch lineare Interpolation:

Fehlende Glanzlichter Auf Grund der linearen Interpolation von Intensitäten können Glanzlichter, die auf spekulare Reflexion zurückzuführen sind, verloren gehen oder abgeschwächt/verschmiert werden. Das wird umso kritischer, je spitzer die spekulare Reflexion ist (großes n im $\cos^n$- Term). Feinere Unterteilung der Oberfläche verbessert Resultat

Mach-Band-Effekt (Ernst Mach, 1865)
Die lineare Interpolation der Leuchtdichte zwischen den Polygonkanten entlang der Rasterlinie führt zu einem Verlauf, der durch plötzliche Änderungen im Anstieg der Intensität gekennzeichnet ist.

- Kontrastverstärkung durch Auge an den Übergängen zwischen Polygonen (helle Bänder)
- Bei Sprüngen in der Helligkeitsänderung stört dieser Effekt
- Gleiche Information benachbarter Rezeptoren wirkt bei weiterer visueller Verarbeitung lateral hemmend auf lokale Lichtempfindung.
- Modellhaft entstehen neben eigentlichen Helleindruck auch Signale, die Helligkeitsgradienten und Laplacefilter-Output entsprechen
- Empfindung wird insgesamt nicht nur durch Lichtintensität, sondern auch durch Überlagerung mit ihrer ersten und zweiten räumlichen Ableitung bestimmt
- führt zu einer Verstärkung von Konturen an 'Sprungkanten'
- Liegen eine helle und dunkle Fläche nebeneinander, beobachtet man einen dunklen Streifen auf der dunkleren Seite und einen hellen Streifen auf der helleren Seite (Kontrastverstärkung)
- Bei Abfolge von Flächen unterschiedlicher Graufärbung sind entlang der Grenzen machsche Streifen

Phong-Shading (Phong 1975)

- Lineare Interpolation der Normalenvektoren zwischen Polygonecken anstelle von Interpolation der Intensität
- Exakte Berechnung der $\cos^n$-Funktion im Modell für jedes Pixel : Glanzlichter werden erhalten!
- Keine Diskontinuität der ersten Ableitung: Mach-Band-Effekt wird vermieden!
- eine Berechnung pro Pixel, davor aber jeweils lineare Interpolation der Normalen pro Pixel

3D-Rendering

mehrere konvexe Objekte oder konkave Objekte sollen gerendert werden. Verdeckungen sind möglich!

- Korrekte Behandlung von Verdeckungen bedarf spezieller Ansätze/Datenstrukturen (Reihenfolgeproblems)
- Rein opake Szenen sind typischerweise wesentlich leichter zu implementieren als transparente (Berechnungsproblem)
- Zeichenreihenfolge ist teilweise wichtig
- Ansätze unterscheiden sich auch in der Granularität und Genauigkeit was auf einmal gezeichnet/sortiert wird
- ganze Objekte nach z-Position sortieren, dann jew. zeichnen
- allg. (ggfs. überlappende) Polygone: Painters-Algorithmus,
- überlappungsfreie Polygone: Depth-Sort-Algorithmus,
- Pixel: Z-Buffer-Verfahren (in Verbindung mit Obj.-Sort.)
- Beliebte Testszene sind sich zyklisch überlappende Dreicke

Painter's-Algorithmus

- Gegeben sei 3D-Szene aus grauen Polygonen mit diffus reflektierender Oberfläche und gerichtete Lichtquelle
- Für jedes Polygon wird reflektierte Strahldichte L auf Basis des eingestrahlten Lichts und Flächennormale berechnet:
- $I_{out} = L = I_{in} * k_d * \cos(\phi)$
- Polygone durch perspektivischer Kameratransformation in Bildraum transformiert und nach absteigendem z-Wert (Distanz Polygonschwerpunkts) sortiert
- sortierte Polygone (entfernte zuerst) mit Polygonfüllalgo. in Pixelraster der x/y Bildebene konvertiert
- Pixel für jedes Polygon per Overwrite-Modus mit Farbwert L im Bildspeicher gespeichert
- Verdeckungsprobleme lösen sich durch Reihenfolge selbst

Depth-Sort-Algorithmus

- Unterteilung in sich nicht überlappende und vollständig überdeckende Teilpolygone
- Ist in der Projektionsebene durch gegenseitigen Schnitt aller Polygone möglich (allerdings blickabhängig - muss in jedem Bild neu berechnet werden!).
- Die sichtbaren Teilpolygone können nun ausgegeben werden:
- Zeichnen der nicht überlappenden Teilpolygone
- Von den sich vollständig überlappenden Teilpolygonen wird nur das vordere gezeichnet.
- Teilpolygone sollten möglichst nicht größer sein als Tiefenunterschied, damit sie in jeder Situation eindeutig sortiert werden

Anwendungsbereiche des Painter's/Depth-Sort Algorithmus:

- Einfache kleine Szenen/Objekte, unterschiedlicher z-Werte
- keine Hardware-Unterstützung für 3D-Rendering angeboten

Z-Buffer-Verfahren (CATMULL 1974)

- Einer der einfachsten 'visible surface'-Algorithmen
- Probleme des Painters-Algorithmus werden überwunden
- Berechnung des z-Wertes für jeden Punkt jedes Polygons und Speicherung des zur Projektionsebene nächstliegenden Farb- und Z-Wertes
- zusätzlicher Speicher (z-Buffer) für jedes Pixel notwendig
- weder Vorsortieren noch Polygonzerlegung erforderlich
- Initialisierung: Für alle Pixel
  - Setze Farbe auf Hintergrundfarbe
  - Setze alle Z -Werte auf ∞ (max. Wert)
  - Setze Z min auf Wert der Near-Plane
- Für alle Polygone (im 3D-Kamerakoordinatensystem)
  - Rasterumwandlung in der Projektionsebene durch modifizierten 2D-Polygonfüllalgorithmus
  - zusätzliche Berechnung des z-Wertes für jedes Pixel
  - Write Pixel Prozedur: z-Wert des aktuellen Pixels kleiner als bereits abgespeicherte z-Wert $\Rightarrow$ z-Buffer Farbe sowie z_p überschrieben
  - näher an Kamera liegende Pixel überschreiben weiter weg liegende
  - Pixelgenaue Sichtbarkeitsbestimmung und -behandlung der Polygone
- Berechnen der z-Werte durch lineare Interpolation
  - Tiefenwerte auch nach Ansichten-Transformation nur für die Eckpunkte gegeben
  - Zunächst lineare Interpolation der z-Werte entlang der Polygonkanten P_iP_j für y-Position der Scanline
  - Danach Füllen der Bildzeile der Interpolation der z-Werte entsprechend
- Berechnung der z-Werte eines Pixels x/y
  - Die y-Koordinate reicht zur Interpolation von z_A, z_B
  - Pixel-z-Wert z_p wird äquivalent ermittelt
  - z_A, z_B, x_A, x_B, in z_p werden gleichzeitig mit x_A-Werten von einer Rasterlinie zur nächsten inkrement.
  - Brüche bleiben in allen Ausdrücken rational
  - Ausdrücke für die z-Werte haben identische Form wie die der x-Werte beim Polygonfüllalgorithmus
- Z-Buffer-Ergebnis ist vergleichbar mit Painters-Algorithmus
- bei opaken Objekten keine vorgängige Sortierung der Polygone nötig
- pixelgenau: überlappende Polygone korrekt dargestellt
- Kaum Mehraufwand gegenüber Polygonfüllalgorithmus
- Problem: Korrekte Berücksichtigung von Transparenzen

Transparenz Alpha-Blending-Verfahren:

- Annahme: Verwendung eines Z-Buffers
- Mit dem Alpha-Blending-Verfahren kann die transparente Überlagerung zweier Objekte im Bildspeicher gelöst werden
- C_f Farbe des Objekts im Vordergrund (kleinster z-Wert)
- α Opazität der Vordergrundfarbe, Wert $(0, 1)$
- C_b Hintergrundfarbe (letzte Farbe im Bildspeicher)
- resultierende Farbe C: $C = \alpha * C_f + (1 - \alpha) * C_b$
- Bildspeicher um Opazitätswert α erweitert
- Speicherbedarf pro Pixel typ. min. 48 Bit: RGBZ + α
- Reines Z-Buffering ignoriert alle Objektepixel, die weiter entfernt sind als vorn liegende Objektpixel
- Bei Berücksichtigung von α-Werten ist die Renderreihenfolge für korrekte Ergebnisse aber sehr wichtig
- Transparenz-Problem: Objekt nachträglich gezeichnet, kann die Farbe nicht korrekt bestimmt werden
- Zuerst: Darstellung aller opaken Objekte nach dem Z-Buffering (reihenfolgeunabhängig)
- Dann Sortieren aller semitransparenten Polygone nach der Tiefe und Zeichnen
- Restfehler: sich zyklisch überlappende oder sich durchdringende semi-transparente Flächen $\rightarrow$ exakte Behandlung durch die vorn beschriebenen Maßnahmen

Globale Beleuchtung

Ray-Tracing Strahlenverfolgung, nicht rekursiv

- Strahlen vom Augpunkt (Kamera Ursprung) durch jedes Pixel des Rasters senden
- Schnittpunktberechnung → Schnittpunkt mit dem größtem z-Wert stammt vom sichtbaren Objekt
- Strahlverfolgung und Aufsummierung der Anteile aufgrund von material- und geometrieabhängigen Parametern → Helligkeits-/Farbwert pro Pixel
- Bestimmung der diffusen und spekularen Lichtreflexion nach dem Phong-Beleuchtungsmodell
- nur einfache, lokale Beleuchtung

Rekursiver Ansatz Reflexion, Schatten

- Berechnung von Sekundärstrahlen am Auftreffpunkt
- Annäherung der Interreflektionen durch ideale Spiegelung
- beim Auftreffen des Strahls auf weiteres Objekt Berechnung der diffusen und spekularen Reflexion der jeweiligen Lichtquelle sowie Erzeugung eines weiteren Strahls durch ideale Spiegelung
- Addition der Sekundärstrahlen an Objekt B zum Farbwert des Pixel am Objekt A → Rekursion kann abgebrochen werden, wenn Beitrag vernachlässigbar!
- Anzahl der Operationen wächst zusätzlich, d.h. Multiplikation des Aufwandes mit der Anzahl der Reflexionen und Refraktionen und Lichtquellen

Brechungseffekte Richtung des gebrochenen Strahls

berechnet sich aus Einfallswinkel zum Normalenvektor sowie den material- und wellenlängenabhängen Brechungsindices.
$\eta_{e\lambda} * sin(\theta_e) = \eta_{t\lambda} * sin(\theta_t)$
Berechnung des Einheitsvektors $\vec{V}_t(\vec{V}_e, n, \theta_t)$ in Richtung Brechung

- An Grenzflächen mit unterschiedlichen Brechungsindizes tritt neben Transparenz ($\vec{V}_t$) auch Reflexion ($\vec{V}_r$) auf
- $\vec{M}$ ist ein Einheitsvektor mit der Richtung von $\vec{n} * \cos(\theta_e) - \vec{V}_e$
- es gilt: $\vec{M} * sin(\theta_e) = \vec{n} * \cos(\theta_e) - \vec{V}_e \rightarrow \vec{M} = \frac{\vec{n} * \cos(\theta_e) - \vec{V}_e}{\sin(\theta_e)}$
- Simulation brechungsbedingter Verzerrungen wird so möglich
- Transparentes/reflektierendes Material erzeugt 2 weiter zu verfolgende Sekundärstrahlen

Erweiterungen Unzulänglichkeiten des einfachen Ansatzes

- Reale Objekte sind eher diffus spekular, d.h. ein ganzes Set von Sekundärstrahlen wäre zu verfolgen.
- Die ideale Spiegelung zur Erzeugung von Sekundärstrahlen ist eine sehr starke Vereinfachung
- Aus Umkehrbarkeit von Licht- und Beleuchtungsrichtung ließe sich Menge von Sekundarstrahlen ermitteln
- muss aus Aufwandsgründen vereinfacht werden

Monte Carlo Ray-Tracing

- Reflexion ist selten ideal spekular, meist entsteht ein Bündel von Strahlen
- Verfolgung mehrerer zufälliger Sekundärstrahlen, deren Beitrag zum Farbwert des Pixel statistisch gewichtet
- Je gestreuter Reflexion, um so mehr Sekundärstrahlen nötig
- Breite Remissionskeulen oder diffuse Interreflexionen sind wegen des Aufwandes nicht behandelbar
- Weiteres Anwachsen der Anzahl an erforderlichen Operationen durch zusätzliche Verfolgung sehr vieler Sekundärstrahlen

Beleuchtungsphänomen Kaustik

- Das Licht der Lichtquelle werde zuerst spekular, dann diffus reflektiert
- Vom Auge ausgehendes Ray Tracing versagt wegen des vorzeitigen Abbruchs der Rekursion am diffus remittierenden Objekt.
- Inverses Ray Tracing [Watt/Watt 1992]: Man erzeugt einen von der Lichtquelle ausgehenden Strahl und reflektiert diesen an glänzenden Oberflächen
- Die reflektierten Lichtstrahlen wirken als zusätzliche Lichtquellen, die dann zu diffusen Reflexionen führen
- Exponentielle Komplexität wenn alle Objekte gleichzeitig transparent und reflektierend sind.

Optimierungsmöglichkeiten

- Berechnung von achsenparallelen Bounding Boxes oder Bounding Spheres um Objekte
- Zunächst Test, ob der Strahl die Hülle schneidet und falls ja
- → Schnittpunktberechnung von Strahl mit allen Polygonen
- → Berechnung des Schnittpunktes mit jew. Polygonebene
- → danach effizienter Punkt-im-Polygon-Test
- Effiziente Zugriffsstruktur auf die Hüllquader: Bäume für rekursive Zerlegungen
- Verwendung von direktem, hardware-unterstützten Rendering
- Verwendung von Hardware mit RTX-Unterstützung

Zusammenfassung

- Erzeugung realistischerer Bilder, da indirekte Beleuchtungsphänomene physikalisch viel genauer
- Ray-Tracing ist aufgrund der hohen Komplexität für interaktive Anwendungen oft wenig geeignet
- Interaktive Programme (CAD, Spiele) verwenden noch eher direktes Rendering mit Texturen
- effiziente räumliche Suchstrukturen können die Anzahl auszuführender Schnittoperationen reduzieren
- Implementierung ist konzeptionell einfach + einfach parallelisierbar
- RT ist sehr gut geeignet, wenn die spiegelnde Reflexion zwischen Objekten frei von Streuung ist
- Cone-Tracing - statt eines Strahles wird ein Kegel verwendet, der die Lichtverteilung annähert

Photon Mapping [Henrik Jensen 1995]

1. Phase: Erzeugung der Photon Map

- Von Lichtquelle ausgestrahlte Photonen werden zufällig in Szene gestreut. Photon Energie kann absorbiert, reflektiert oder gebrochen werden.
- Speichern der Photonen in der Photon Map, also Position, Richtung beim Auftreffen und Energie für Farbkanäle RGB
- Photon wird in Suchstruktur gespeichert (Irradiance cache)
- Reflektionskoeffizienten als Maß für Reflektionswahrscheinlichkeit (analog Transmissionswahrsch.)
- Energie bleibt nach Reflexion unverändert. Neue Richtung wird statistisch auf Basis der BRDF gewählt

2. Phase: Aufsammeln der Photonen (gathering)

- Ray-Tracing um für Primärstrahl von der Kamera durch einen Pixel den Schnittpunkt x mit der Szene zu bestimmen
- Sammle die nächsten N Photonen um x herum auf durch Nächste-Nachbar-Suche in der Photon Map (N = konst.)
- S sei die (kleinste) Kugel, welche die N Photonen enthält
- Für alle Photonen: dividiere die Summe der Energie der gesammelten Photonen durch die Fläche von S (→ Irradiance) und multipliziere mit der BRDF angewendet auf das Photon.
- Dies ergibt die reflektierte Strahldichte, welche von der Oberfläche (an der Stelle x) in Richtung des Beobachters abgestrahlt wird.

Radiosity

- Ans: Erhaltung der Lichtenergie in geschlossener Umgebung
- Radiosity: Energierate, die eine Oberfläche verlässt
- gesamte Energie, die von Oberfläche emittiert oder reflektiert wird, ergibt sich aus Reflexionen oder Absorptionen anderer Oberflächen
- Es erfolgt keine getrennte Behandlung von Lichtquellen und beleuchteten Flächen
- nur diffuse Strahler $rightarrow$ Unabhängigkeit der Strahldichte vom Blickwinkel
- Lichtinteraktionen im 3D-Objektraum berechnet
- Danach viele Ansichten schnell ermittelt

Strahldichte an dA_r (s=Sender, r=Reveiver):
$L_r = \beta_r(\lambda) * \int_{A_s} \frac{L_s}{\pi * r^2} * \cos(\theta_s) * \cos(\theta_r) * dA_s$
mittlere Strahldichte:
$L_r = \beta_r(\lambda) * \frac{1}{A_r} * \int_{A_r} \int_{A_s} \frac{L_s}{\pi * r^2} * \cos(\theta_s) * \cos(\theta_r) * dA_s * dA_r$
Sichtbarkeitsfaktor H_{sr}

$$F_{sr} = \frac{1}{A_R} \int_{A_r} \int_{A_s} \frac{\cos(\theta_s) * \cos(\theta_r)}{\pi * r^2} * H_{sr} * dA_s * dA_r, H_{sr} = \begin{cases} 1 \rightarrow A_s \text{ sichtba} \\ 0 \rightarrow A_s \text{ unsicht} \end{cases}$$

Für im Verhältnis zum Abstand kleinen Flächen können θ_s, θ_r und Radius r konstant angenommen werden.

$$F_{sr} = A_S \frac{\cos(\theta_s) * cos(\theta_r)}{\pi * r^2} * H_{sr}$$

Jedes Patch wird nun als opaker Lambertscher Emitter und Reflektor betrachtet.
emittierende Strahldichte: $L_r = L_{emr} + R_r * \sum_S F_{sr} * L_s$
Es ergibt sich schließlich als Gleichungssystem:

$$\begin{pmatrix} 1 - R_1F_{11} & -R_1F_{12} & \dots & -R_1F_{1n} \\ 1 - R_2F_{21} & -R_2F_{22} & \dots & -R_2F_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 1 - R_nF_{n1} & -R_nF_{n2} & \dots & -R_nF_{nn} \end{pmatrix} * \begin{pmatrix} L_1 \\ L_2 \\ \vdots \\ L_n \end{pmatrix} = \begin{pmatrix} L_{em1} \\ L_{em2} \\ \vdots \\ L_{emn} \end{pmatrix}$$

($R_r, R_{rR}, R_{rG}, R_{rB}, L_{emr}$ sind allg. wellenlängenabhängig)

Adaptives Refinement Adaptives Radiosity-Verfahren:

- vereinfachte Formfaktor-Berechnung ist ungenau bei eng beieinander liegenden Flächenstücken
- adaptive Unterteilung solcher Flächen in feinere Polygone
- Notwendigkeit durch erste Berechnung geschätzt

Progressive Refinement

- das Radiosity-Verfahren ist sehr aufwendig
- viel weniger Samples als Monte-Carlo-Raytracin
- inkrementelle Approximation des Ergebnisses des exakten Algorithmus
- akkumuliert mehr Energie in jedem Schritt, verletzt Gleichgewicht der Strahlung → Korrektur notwendig: $L_r^{k+1} = L_{emr} + R_r * \sum_s F_{sr} * L_s^k$

Radiosity Eigenschaften

- ausschließlich Berücksichtigung der diffusen Reflexion
- blickwinkelunabhängig, direkt im 3D-Raum arbeitend
- realistische Schattenbilder, insbesondere Halbschatten
- sehr rechenintensiv, deshalb meist Vorausberechnung
- → Beleuchtungsphänomene wie z.B. indirektes Licht sehr gut/realistisch darstellbar
- → Kombination von Radiosity und Ray Tracing ermöglicht Szenen mit sehr hohem Grad an Realismus

Texture Mapping

Bildbasiertes Rendering

- typisch: Applizieren von 2D-Rasterbildern auf 3D-Modellen
- 3D-Objekte mit relativ einfachen Polygonen modelliert
- Details als Texturen - 'Impostor' genannt.
- Texture-Mapping als Erweiterung des Pattern-Filling
- als Verallgemeinerung auch Image-based Rendering genannt

Erzeugung von Texturen:

- 'reale' Texturen aus realen rasterisierten/digitalen Fotografien (Pixel werden Texel)
- 'berechnete' Texturen → synthetische Computergrafik
- vorberechnete reguläre Texturen (basieren auf Texeln)
- nach Bedarf erzeugte statistische bzw. prozedurale Texturen

Anwendung von Texturen - Grundprinzipien:

- Texturraum in den Bildraum der Darstellung
- Verwendung untersch. geometrischer Transformationen
- Resampling: transformiertes Texturraster wird auf Bildraster gerundet
- Filtern: Verhindern/Abmildern von resampling-basierten Aliasing-Effekten
- Beleuchtung: RGB-Werte der Textur dienen als Materialattribute bei der Beleuchtungsrechnung

Unterschiedliche Arten des Texturmappings

Parametrisch auf 3D-Polygon, indem den Eckpunkten des Polygons 2D-Texturkoordinaten explizit zuordnet werden
Affin direkte affine Abbildung der Textur auf projizierte Polygone im Bildraum
Perspektivisch Zwischenabbildung der Textur in 3D-Objektraum und perspektivische Projektion in Bildraum
Projektives Verwendung unterschiedlicher Projektionsarten (parallel, perspektivisch, eben, zylindrisch, sphärisch)
Environment Spiegelung der Textur an der Oberfläche (bzw. Refraktion) mit entsprechender Verzerrung

Affines Texturemapping

Durch Zuordnung von 3 Punkten im Bildraster zu den entsprechenden 3 Punkten im Texturraster erhält man ein Gleichungssystem mit 6 Gleichungen und 6 Unbekannten ($a_u, b_u, c_u, a_v, b_v, c_v$):

- $P_1 : u_1 = a_u * x_1 + b_u * y_1 + c_u; v_1 = a_v * x_1 + b_v * y_1 + c_v$
- $P_2 : u_2 = a_u * x_2 + b_u * y_2 + c_u; v_2 = a_v * x_2 + b_v * y_2 + c_v$
- $P_3 : u_3 = a_u * x_3 + b_u * y_3 + c_u; v_3 = a_v * x_3 + b_v * y_3 + c_v$

Für jedes Pixel(x,y) im Polygon: Resampling der Textur(u,v) bei der Rasterkonvertierung (Polygonfüllalgorithmus)
Für jedes Pixel(x,y) finde die Texturkoordinaten(u,v), d.h.:

- Rückwärtstransformation vom Ziel zum Original → keine Löcher im Bild!
- Texturkoordinate kann übersprungen oder wiederholt sein
- Störsignale (Aliasing) → Filterung notwendig!

Affines Mapping der Vertices x,y auf u,v → lineare Interpolation der u/v-Texturkoordinaten zwischen den Vertices für jedes Pixel (ähnlich wie RGB- bzw. Z-Werte im Polygonfüllalgorithmus, durch Ganzzahlarithmetik)
Problem: Durch affine 2D-Abbildungen können nur Transformationen wie Rotation, Skalierung, Translation, Scherung in der Bild-Ebene abgebildet werden, aber keine Perspektive! → Abbildungsfehler zwischen den Eckpunkten! (kleine Dreiecke → kleiner Fehler!)

Perspektivisches Texture-Mapping

entspricht affinem Textur-Mapping mit Zwischenschritt

- Matrix M_{to}: Koordinatentransformation vom Texturraum in den 3D-Objektraum (Translation, Rotation, Skalierung)
- Matrix M_{oi} : Koordinatentransformation vom Objektraum in den Bildraum (Kameratransformation)
- Matrix M_{ti}: gesamte Koordinatentransformation vom Texturraum direkt in den Bildraum: $M_{ti} = M_{to} * M_{oi}$
- Matrix M_{ti}^{-1}: Inverse Koordinatentransformation vom Bildraum zurück in den Texturraum

→ 4x4-Matrix für homogene Koordinaten.
Vergleich: Perspektivisches / Affines Texture Mapping

- perspektivische Ansicht geometrisch korrekter Bilder
- pers. höherer Berechnungsaufwand pro Polygon/Pixel
- affines Mapping kann u/v zwischen Polygonecken linear interpolieren

Textur-Mapping mit Polygon-Schattierung

- Bestimmung der zum Polygon gehörenden sichtbaren Pixel im Bildraum (Polygonfüllalgorithmus)
- Ermittlung der zur jeder Pixelkoordinate gehörenden Texturkoordinate mit Hilfe der inversen Transformationsmatrix M_{ti}^{-1}
- Ermittlung der Farbe des zu setzenden Pixels aus dem Texturraster (und ggf Beleuchtung)
- Beleuchtungsrechnung: Multiplikation der Helligkeit einer beleuchteten diffusen weißen Oberfläche mit den rgb-Werten der Textur (Lambert Modell)

Weitere Texturarten

Texturen mit Transparenz RGBA-Wert zu jedem Pixel gespeichert, d.h. Rendern mit Alpha Blending
Video Texture zeitlich veränderliche Textur, d.h. dynamische Veränderungen
Solid Textures Textur als 3D-Array → RGB(A)-Werte pro Voxel

- Abbildung über affine 3D-Transf. xyz auf uvw
- beim Rendern entweder auf Vertices und für Pixel linear interpoliert oder für jedes Pixel einzeln angewendet (Pixelshader)

Projektives Textur-Mapping

Berechnung der Texturkoordinaten aus der aktuellen Position der einzelnen Polygone (Analogie: Projektion eines Diapositivs auf ein räumliches Objekt)

- Parallelprojektion der Textur auf einen Zylinder mit abgeschrägten Endflächen
- Projektion ist relativ zum Objekt definiert, Textur bewegt sich mit dem Körper
- markierte Bereiche haben auf Zylinder stets identische Positionen
- keine explizite Zuordnung von uv-Koordinaten zu Polygoneckpunkten notwendig, weniger Modellieraufwand!

projektives Textur-Mapping (Parallel-/Zentralprojektion) Darstellung geometrischer Eigenschaften
- einfache Darstellung von Parameterlinien
- Simulation eines Lichtkegels (Leuchtdichteverteilung)

Zylindrisches Textur-Mapping
- radiale Projektion der Textur-Koordinaten auf Zylinderoberfläche
- visueller Effekt für zylinderähnliche Objekte ähnlich zu parametrischem Textur-Mapping

Sphärisches Textur-Mapping
- Zentralprojektion der Textur-Koordinaten auf eine Kugeloberfläche
- Vorteil des projektiven Texturmappings: Eine explizite Zuordnung der 3D-Punkte zu Texturkoordinaten mit stetiger Fortsetzung der Parametrisierung an den Polygongrenzen entfällt → weniger Modellieraufwand!

Environment Mapping

Spezialfall des projektiven Textur-Mapping

- Simulation der Reflexion der Umgebung an reflektierenden Flächen
- Darstellung abhängig von der Position des Betrachters sowie von den Normalen der reflektierenden Fläche
- Textur entspricht der Lichtquelle für die Beleuchtung durch die Umgebung: Sphere Map bzw. Cube Map
- relativ große Entfernung der reflektierten Objekte von der spiegelnden Fläche

Mapping der Textur auf die spiegelnde Oberfläche

- Aussenden eines Strahls vom Auge auf einen Punkt der spiegelnden Oberfläche
- Ermittlung der Reflexionsrichtung entsprechend dem Einfallswinkel des Strahl zur Flächennormale
- damit Bestimmung des zu reflektierenden Punktes in der Umgebung (Textur-Pixels aus der Environment Map)

Erzeugung einer Cube Map-Textur

- Aufteilung der Environment Map in sechs Bereiche, die sechs Flächen eines Würfels um spiegelnde Fläche entsprechen
- Rendern der Umgebung sechs mal mit einem Kamera-Sichtfeld von jeweils 90 Grad aus dem Mittelpunkt des Würfels
- Alternativ: Digitale Aufnahme und Einpassen der sechs Flächen mittels Image Warping in die jeweiligen Zonen
- Strahlverfolgung: Sehstrahl wird an den Eckpunkten des Objekts gespiegelt und dreidimensional mit den 6 Wänden der Cube Map geschnitten
- Zuordnung von Objektkoordinaten (x/y/z) zu Texturkoordinaten (u/v)
- Transformation kann wie beim perspektivischen Texturmapping berechnet werden und beim Rasterisieren für die dazwischen liegenden Pixel angewendet werden
- Effekt ähnlich wie bei Raytracing, jedoch geometrisch angenähert
- keine aufwändigen Strahl-Objektschnitte notwendig
- Näherung ungenau, wenn spiegelndes Objekt weit weg von Kamera
- nur Einfachreflexion
- kann dynamisch generiert werden. Bewegte gespiegelte Objekte in Echtzeit darstellbar
- gespiegeltes Dreieck kann auf mehrere Wände der Cube Map fallen. Kann durch mehrfaches Mapping und Clipping gelöst werden

Environment Mapping [Haeberli/Segal 1993] für Kugel und Torus:

- Unterschiedliche Ausrichtung der Objektoberfläche sorgt für korrekte Verzerrung der spiegelnden Objekte. Die Darstellung der spiegelnden Objekte steht beim Environment-Mapping im Vordergrund und nicht die korrekte geom. Darstellung gespiegelter Objekte
- Alle Raumrichtungen werden auf der Kugeloberfläche abgebildet. Je nach Aufnahmegeometrie mehr oder weniger großer blinder Fleck hinter der Kugel
-

Erstellung einer Spherical-Environment-Map-Textur

- spiegelnde Kugel in der Mitte einer Szene
- Fotografie der Kugel mit einer Kamera sehr großer Brennweite aus großem Abstand
- Entstehung einer kreisförmigen Region in der Textur-Map mit den Tangenten jeweils an den Außenkanten
- Texturwerte außerhalb des Kreises werden nicht benötigt
- Wahl der Blickrichtungen wichtig für Anwendung

Anwendung einer Spherical Environment Map

- Zur Bestimmung der Texturkoordinate eines dargestellten Punktes wird zuerst die Normale n an diesem Punkt bestimmt

- Normale n wird auf die x/y- Ebene projiziert. Projizierter Normalenvektor entspricht Texturkoordinaten in Sphere Map, welche die an dieser Stelle reflektierte Umgebung zeigt
- Reflexion nicht von der Lage des reflektierenden Punktes abhängig (nur von Normalenrichtung)

Environment Map in latitude-/longitude-Koordinaten

- Spiegelung wird aus Richtung des gespiegelten Strahls in Winkelkoordinaten berechnet
- entweder pro Pixel oder pro Vertex mit anschließender Interpolation pro Pixel
- keine Berücksichtigung der Position des spiegelnden Objekts
- korrekt nur für unendlich entfernte gespiegelte Objekte → geeignet zur Spiegelung weit entfernter Objekte

High-Dynamic Range Imaging (HDRI) Environment-Maps

- enthalten 'gesamte Dynamik' des Lichts (Floating Point)
- Wesentlich realistischere Bilder
- Tone Mapping: berechnete HDRI-Bilder werden anschließend auf die Dynamik des Monitors reduziert
- Refraktion/Brechung: wie Spiegelung, jedoch Sekundärstrahl aus Sehstrahl über Brechungsindex und Oberflächennormale, statt gespiegelt

Mip-Mapping

Aus Originaltextur Bildung einer Menge jeweils kleinerer Texturen (halbe Kantenlänge). Vermeidung/Abmilderung von Aliasing-Effekten durch Vorfilterung und Anwendung der passend aufgelösten Textur(-en) per bilinearer/trilinearer Filterung

Sampling-Artefakte Aliasing-Effekte

- Pixel der Textur und Bildes weisen unterschiedliche Rastergrößen auf
- Berechnung der transformierten Texturkoordinaten als Floating-Point-Werte und Rundung auf ganze Zahlen
- bei inverser Transformation vom Zielbild zurück zur Textur keine Lücken im Bild, aber die Pixel der Textur können ausgelassen oder mehrfach verwendet werden
- durch das Resampling der Textur auf das resultierende Bildraster entstehen oft Aliasing-Artefakte

Zwei unterschiedliche Situationen

- Abbildung mehrerer Texturpixel auf Bildpixel (Unterabtastung - Minification)
- Abbildung Texturpixels auf mehrere Bildpixel (Überabtastung - Magnification)
- Filteroperationen zur Interpolation der Bildpixel-Färbung notwendig
- Ansonsten Verletzung des Abtasttheorems/Nyquistfrequenz

Aufbau

- In 3D-Szenen oft Körper mit selber Textur vom Betrachter unterschiedlich weit weg
- im Bild oft Unter- oder Überabtastung und Aliasing
- Vorberechnung derselben Textur für versch. Entfernungen
  - Stufe 1: volle Auflösung
  - Stufe 2: halbe Auflösung in jeder Richtung (1/2)
  - Stufe k: Auflösung $(1/2)^k$
  - Stufe n: niedrigste Auflösung (je 1 Pixel für RGB)
- Speicherbedarf:
- (hypothetische) Annahme: Anordnung im Array (getrennt f. RGB) → Alle niedrigen Auflösungen verbrauchen zusammen nur ein Viertel des Speicherplatzes
- Mip:*multum in parvo* = viel (Info) auf wenig (Speicher)
- niedrige Auflösungsstufen werden durch Filterung aus den höheren berechnet
  - einfach: z.B. Mittelwert aus 4 Pixeln (Box-Filter)
  - aufwendiger: z.B.: Gaußfilter
- Filter-Operationen können bei Initialisierung der Textur vorausberechnet werden
- nur ein Drittel zusätzlicher Speicherplatzbedarf

Anwendung

- OpenGL-Filteroperationen im Bildraum (zur Laufzeit)
- GL_NEAREST: Annahme nächstliegender Textur-Pixels
- GL_LINEAR: bilineare Interpolation aus nahem 2x2-Feld
- Genauere Interpolationsverfahren (z.B. bikubisch) gelten als zu aufwendig für Echtzeitanwendung

Darstellung mit Mip-Map Texturen (zur Laufzeit)

- passende Auflösungsstufe k Skalierung berechnet aus der Entfernung zum Betrachter und der perspektivischen Verkürzung: $d/z = (1/2)^k \rightarrow k = log_2(z) - log_2(d)$
- Transformation der Pixel zwischen Textur- Eckkoordinaten der gewählten Auflösung auf Polygon im Bildraum
- Vermeidung von Aliasing-Effekten durch Trilineare Filterung: Mip-Map-Stufe wird linear gewichtet zwischen zwei Mip-Map-Stufen interpoliert: z. B. wenn $k = 2.3 \rightarrow 30\% Anteil_{k=3}$ und $70\% Anteil_{k=2}$

Anti-Aliasing durch trilineare Filterung

- Durch perspektivische Verkürzung wird weiter hinten liegende Textur verkleinert und im Vordergrund vergrößert
- Bei einer Skalierung kleiner als 1 überspringt die gerundete inverse Texturtransformation Pixel in der Textur (minification). Die im Bildraum gesampelten Texturpixel werden somit 'willkürlich' ausgewählt. Dadurch können Treppenstufen und Moiré-Muster entstehen. Durch Mip-Mapping werden an diesen Stellen geringer aufgelöste (gefilterte) Texturen verwendet
- Vergrößerte Darstellung: Trilinearen Filterung = lineare Filterung zwischen den zwei aufeinander-folgenden Mip-Map-Stufen + bilineare Filterung in jeder der beiden Stufen → Kantenglättung, Tiefpassfilter

Rip-Maps Anisotrope Filterung

- z.B. bei flacher Aufsicht ist Verkleinerung in y-Richtung viel stärker als in x-Richtung
- Ohne spezielle Maßnahmen müsste jeweils Mip-Map-Stufe mit kleinsten Auflösung verwendet werden
- → führt zur unscharfen Texturabbildung.
- Abhilfe: Anisotrope Mip-Maps (=Rectangular Mip-Map)
- Verschiedene Auflösungsstufen in x- und y-Richtung erzeugt
- Vierfacher Speicherbedarf gegenüber höchster Auflösung

Bump-Map

- Reliefartige Texturen: Herkömmliche Texturen von Nahem erscheinen flach
- Idee: Verwendung zusätzlicher Texturen, welche Tiefeinformationen beinhalten
- Offset zur Polygonebene in Richtung der Normale als Grauwert der Textur kodiert
- Anwendung des Offsets auf Polygonfläche: Normale wird als Gradient der Bumpmap berechnet. Beleuchtung wird daraus wie bei der Normalmap pro Pixel berechnet
- Offset nicht berücksichtigt → Als Konturen nicht erkennbar

Normal-Map

- Normalen Vektor x/y/z als RGB-Wert kodiert
- Polygon: als Schnitt mit Normalenrichtung
- Anwendung der Normal-Map auf Polygonfläche: Normale der N-Map modifiziert die Flächennormale. Bei der Beleuchtungsberechnung wird für jedes Pixel die modifizierte Normale verwendet
- Offset nicht berücksichtigt → Als Konturen nicht erkennbar

Parallax-Map Tomomichi Kaneko 2001

- Ausgangsdaten: Bump Map
- Die u/v-Koordinaten der angezeigten Textur werden Entsprechend der Blickrichtung beim Look-up um $\delta u = h * \tan(\phi)$ verschoben
- Anwendung des Offsets auf Polygonfläche: Anwendung der Bump Map des Offests auf Polygonfläche. Normale wird als Gradient der Bumpmap berechnet. Beleuchtung wird daraus wie bei Normalmap pro Pixel berechnet.

Displacement-Map

- Ausgang: Bump Map, korrekter Schnitt eines Sehstrahls mit Bump Map durch iterative Suche des Schnittpunktes
- Finde u_0 , sodass $u - u' = h(u') * \tan(\phi)$ mittels Bisektion entlang dem Sehstrahl
- Bei Mehrdeutigkeit: Finde u_0 am weitesten weg von u
- Silhouetten: Auch u/v-Koordinaten außerhalb der Polygongrenzen müssen berücksichtigt werden
- aufwendige Shader Programme nötig

Shadow Mapping

- Erzeugen der Shadow Map
- Darstellung (mit z-Werten) aus Sicht der Lichtquelle
- Kamera Koordinaten in Lichtquelle zentriert (Matrix L)
- z-Puffer als Textur speichern
- Kamera Ansicht: View Matrix V (mit z-Puffer)
- Um Schatten zu erzeugen benötigen wir Shader mit Lookup
- 4x4-Matrix: $M = V^{-1} * L$

Shadow map look-up:

- Transformiere jedes Pixel aus dem Kameraraum in den Lichtraum
- $p' = L * V^{-1} * p$
- Vergleiche transformierte z-Werte (p'_z) mit den z-Werten der Shadow Map (z_s)
- ($p'_z > z_s$): im Schatten - keine Beleuchtung von der Lichtquelle
- sonst: Punkt ist von der Lichtquelle her sichtbar, wende Beleuchtung in der Schattierung des Pixels an

Probleme Z-fighting beim Schattentest:

- Schattentest ($p'_z <= z_s$) sollte für beleuchtete Pixel korrekt ($p'_z = z_s$) ergeben.
- Aufgrund der Rechenungenauigkeit der Fließkomma-Arithmetik wird Gleichheit selten erreicht!
- Beleuchtete Polygone schatten sich teilweise selbst ab.
- Lösung: kleiner Offset im Schattentest: $IF(p'_z <= z_s + Offset...)$
- durch das Offset wird sichergestellt, dass keine falschen Schatten entstehen

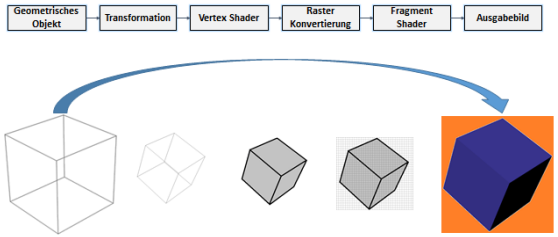
Uniform Shadow-Map Probleme: zu niedrige Auflösung der Shadow Map im Nahbereich, Großteil der Shadow Map ist irrelevant für Kameraansicht

Perspektive Shadow-Map adaptive schiefsymtetrische Projektion; nicht uniforme perspektive Shadow Map

Zusammenfassung

- Anwendung immer als inverse Transformation
- Resampling + Rekonstruktion: Das transformierte Texturraster wird nach der Transformation durch das Bildraster neu abgetastet.
- Filter: Verhindern bzw. Abmildern von Aliasing-Effekten, verursacht durch Resampling.
- Lösung: Tiefpass-Filter vor der Transformation: Mipmapping, Anisotrope Filter.
- Beim Abtasten (Rekonstruktion):Trilineare Filterung in x, y, und k (Mip-Map-Stufe)
- Texturinhalt als Material, Beleuchtung, Geometrie interpretiert

Grafik Pipeline



- algorithmisches Konzept, sowie Realisierung der Grafikkartenhardware ist vergleichbar mit Fließband
- spezialisierte Arbeitsstationen (Spezialprozessoren)
- jedes geometrische Objekt durchläuft Arbeitsstationen sequenziell
- Arbeitsschritte können dadurch gleichzeitig auf verschiedenen Daten ausgeführt werden

Bestandteile

1. Programm API
2. Treiber
3. Vertex-Verarbeitung
4. Primitivenbehandlung
5. Rasterisierung & Interpolation
6. Fragment Verarbeitung
7. Rasteroperation
8. Bildspeicher

Allgemein

- Anwendungsprogramm:
  - läuft auf der CPU,
  - definiert Daten und Befehlsabfolge,
  - greift dazu über das Grafik-API (z.B. OpenGL, Direct3D) auf die Grafikkarte zu
- Treiber: übersetzt die Grafikbefehle des Programms in die Maschinensprache der speziellen Grafikhardware (Graphics Processing Unit)
- Befehle und Daten werden über den Bus (z.B. PCI-Express) von der CPU auf die GPU übertragen
- OpenGL-Pipeline: Abarbeitung der Grafikbefehle auf der GPU
- Ausgabe des Bildspeichers auf dem Monitor
- Treiber schickt Daten/Befehle an die GPU (z. B. via PCIe -Bus)
- Funktionsausführung auf der GPU ist dann abhängig vom aktuellen Zustand

Abarbeitungsreihenfolge auf der GPU

- Empfangen der Vertices in einer geordneten Sequenz
- Vertexverarbeitung via Vertex Shader. Jeder Input-Vertex im Datenstrom wird in einen Output-Vertex transformiert und beleuchtet
- Primitive culling (Verwerfen wenn nicht sichtbar) und clipping (Abschneiden der Polygone am Rand)
- Rasterkonvertierung (Polygon Filling) und Interpolation der Attributwerte (x-Koordinate, 1/z, R, G, B, Texturkoordinaten u/v, ...)
- Die Daten jedes Fragmentes (Pixel/Subpixel) wird mit einem Fragment Shader verarbeitet. Zu jedem Fragment gehört eine Anzahl Attribute
- Per-Sample Operationen: Blending (Alpha-Blending bei Transparenz), Tiefen- und Stencil- Operationen ...

Vertex-Verarbeitung

Transformationen Modell-Transformation, Kamera-Transformation (Model View Matrix) → Matrixmultiplikationen → Skalarprodukt

Beleuchtung Lichtquellen, diffuses & spekuläres Material: Lambert, Phong-Modell → Skalarprodukt

Skalarprodukte (Gleitkomma-Multiplikationen und Additionen) werden durch viele parallele Prozessoren auf der GPU effizient verarbeitet

Rasterkonvertierung

- Edge Tables bereits erzeugt (Polygonsetup)
- Rasterkonvertierung/Interpolation entspricht der Scan-Line-Konvertierung, generiert Pixel (Fragments)
- Interpolation der x-Werte der Kanten (left/right edge scan)
- pro Scan Line: inkrementiere x-Wert (left/right edge)
- lineare Interpolation weiterer Attribute
  - $z(1/z)$-Werte,
  - RGB-Werte (Gouraud Shading),
  - Texturkoordinaten u/v (affines Texturmapping),
  - Normalen (Phong Shading)
- sehr wenige Ganzzahloperationen pro Pixel/Bildzeile
- Ausnahme z.B. perspektivische TM (FP-Division)

Fragment-Verarbeitung

- Weiterverarbeitung auf Basis der interpolierten Attribute im Fragment Shader
- Beispiel Phong-Shading: Berechnung des Phong-Beleuchtungsmodells auf Basis der vorher linear interpolierten Fragmentnormalen, -position und Materialdaten sowie der Daten der Lichtquellen und Kameraposition

Rasteroperationen

- Abschließende Auswahl/Zusammenfassung der berechneten Fragmentdaten (pro Pixel)
- Beispiel: nicht transparente Objekte, Übernahme der Farbwerte mit z-Position, welche am dichtesten an der Kamera ist
- Beispiel: transparente Objekte, lineares Blending zwischen schon existierenden Farbwerten und neuesten entsprechend der Transparenz

Performance

Einfaches Modell zur Bestimmung der Rechenzeit T:

$$T = a * \text{Anzahl Vertices} + b * \text{Anzahl Bildpixel}$$

(a = Aufwand pro Vertex, b = Aufwand pro Pixel)

- Grafikkarten geben ihre Performance an in:
  - Anzahl Polygone / Sekunde (Polygone mit kleiner Pixelanzahl)
  - Anzahl verarbeitete Pixel / Sekunde
- Problem der Grafik-Pipeline: Langsamste Operation hält Pipeline auf (bestimmt Durchsatz)
- → Zwei extreme Situationen:

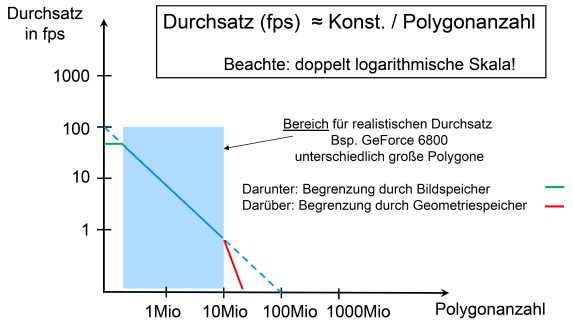
  Vertex-limited viele Vertices, kleine Polygone (wenige Pixel), einfache lineare Interpolation der Vertex-Attribute pro Fragment (kleines b)

  Fill rate limited anspruchsvolle Fragment-Shader (großes b), weniger dafür große Polygone (viele Pixel)

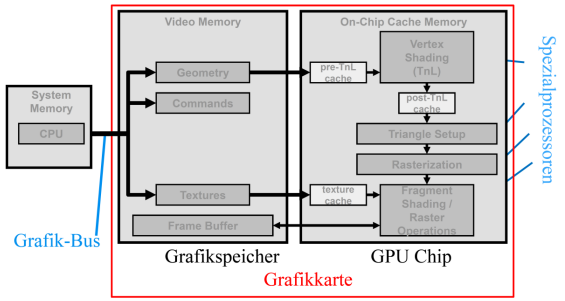
- Außerdem: Grafikspeicher-Zugriffe sind teuer (Latenz und Bandbreite beachten) z.B. Auslesen gerenderter Bilder aus dem Grafikspeicher
- Eine für die Grafikkarte angegebene Performance ist nur unter unrealistisch günstigen Bedingungen zu erreichen.
  - wegen Speicherbandbreite für Vertexzugriffe
  - da Framebuffer-Reset teuer
  - Herstellerangaben nur für optimalen Bedingungen
  - realistisch →≈ 10% der Peak Performance!

Durchsatz (fps) ≈ Konst./Polygonanzahl

- unter realistischem Durchsatz: Begrenzung durch Bildspeicher
- über realisitschem Durchsatz: Begrenzung durch Geometriespeicher



Hardware-Architektur



Bildverarbeitung

Operationen auf dem Bildraster

Problem der Vorwärtstransformation:

- Farbwerte sitzen im Zielbild an nicht ganzzahligen Koordinaten, das Ausgabegerät benötigt aber Farbwerte in einem Raster (ganzzahlige Koordinaten)
- durch Runden können Löcher im Bild entstehen, einige Pixel werden u. U. mehrfach belegt

Inverse Transformation

- da jedes Pixel im Zielbild B an Position (k,l) Ausgangspunkt der Rechnung ist, bleibt keines unbelegt
- keine Löcher mehr
- Problem: Auch das Quellbild A ist nur an ganzzahligen Rasterpunkten i,j gegeben. Wie ermittelt man A(x,y) aus den Pixeln A(i,j) im Umfeld von x,y? Wie groß muss das Umfeld sein? → Resamplingproblem (Wiederabtastung)

Rückwärtstransformation der Pixelkoordinaten

- Inverse Transformation der Koordination vom Zielbild + Sampling im Originalbild
- Es entstehen keine Lücken im Zielbild aber durch Rundung nichtganzzahliger Pixelkoordinaten auf den nächstenganzzahligen Wert können Aliasing-Artefakte entstehen $B(k.l) = A(h^{-1}(k,l))$
  - einzelne Pixel im Originalbild werden öfter gesampelt
  - einige Pixel im Originalbild werden ausgelassen

Rückwärtstransformation der Pixelkoordinaten mit Interpolation benachbarter Pixel:

- Inverse Transformation der Koordination vom Zielbild + Sampling im Originalbild
- bei nicht ganzzahligen Pixelkoordinaten kann man zwischen den benachbarten Pixelwerten im Originalbild interpolieren
- dadurch werden die scharfen Flächengrenzen unscharf
- aber die wahrgenommenen Grenzen zwischen schwarzen und weißen Flächen können so zwischen den ganzzahligen Pixelwerten positioniert werden
- die empfundene Genauigkeit wird durch Antialiasing sogar erhöht!

Verkleinern eines Bildes durch Skalierung

- Inverse Transformation der Koordination vom Zielbild + Sampling im Originalbild
  - auch hier entstehen zwar keine Lücken im Zielbild beim Transformieren zusammenhängender Bilder
  - aber beim Sampeln im Originalbild entstehen Aliasing-Artefakte:
  - z.B. Auslassen jedes zweiten Pixels im Originalbild → Zielbild wird uniform weiß (oder schwarz)
- exakte Lösung wäre nur bei doppelter Auflösung möglich
- jedoch Auflösung im Zielbild begrenzt
- Näherung durch Sampeln und Interpolation mehrerer Nachbarpixel führt zu möglicher Lösung

Vorwärtstransformation
- gesuchte Farbwerte im Zielbild liegen nicht immer an ganzzahligen Koordinaten
- durch Rundung können Lücken im Zielbild entstehen

Rückwärtstransformation
- es entstehen keine Lücken im Zielbild, dennoch können (Sampling) Artefakte (Aliasing) auftreten
- diese Artefakte können durch Interpolation mehrerer Pixel teilweise abmildert werden

Frequenzraum

Ein Bild ist im zweidimensionalen Ortsraum definiert

- Farb-/Helligkeitswert als Funktion des Ortes W(x,y)
- Bei ortsabhängigen Signalen ist mit Frequenz gemeint, nach welcher Distanz (in Pixeln) sich das Muster im Bild wiederholt
- Umwandlung von Ortsraum in Frequenzraum durch Fourier-Transformation
  - Bild im Frequenzraum f definiert
  - $S(f)$ wird Spektrum genannt

Beispiel: 2D-Spektrum eines Schachbrettmusters

- Kleinstes auf einem Pixelraster darstellbares Muster hat Durchmesser von 2 Pixel
- Höchste darstellbare Frequenz $f_x = f_y = 0,5\ \text{Pixel}^{-1}$
- Diese Frequenz nennt man Nyquist-Frequenz

Bildraster als P-dimensionaler Vektorraum:

- diskrete Grauwertbild $F(i * \delta x, j * \delta y)$ habe M Spalten und N Zeilen
- $P = M * N$ ist damit die Dimension des Primärdatenraumes des Bildes
- $M * N$ Basisbildern der Dimension $M * N$, jeweils nur ein Pixel den Wert 1 (weiß)
- Basisbilder sind damit alle orthonormal
- ergeben in grauwertgewichteten Summe das diskrete Bild F

Vektoroperationen im Bildraum

- Skalarprodukte zwischen zwei Bildern
- Basistransformation: Ganze Bilder dienen als Basis eines transformierten Bildraumes
- Die neuen Basisvektoren müssen linear unanhängig sein und idealerweise orthonormal zueinander stehen
- Eine Basistransformation entspricht einer Drehung des Vektorraumes um den Nullpunkt

Transformation

- Jedes Bild ist Linearkombination der Basisvektoren, entspricht gewichteter Addition von Bildern im Ursprungsraum
- 4 neue Basisvektoren B_i (2 x 2 Pixel) für unterschiedliche Frequenzen
- Die 4 Basisvektoren stehen orthonormal zueinander. Test mittels paarweisen Skalar-produkten:
  $B_i B_k = \begin{cases} 1 \text{ falls } i = k \\ 0 \text{ sonst} \end{cases}$
- Jedes einzelne Pixel aber auch jedes Bild kann als Linearkombination aus 4 Basisvektoren B_1 bis B_4 konstruiert werden: $P = a * B_1 + b * B_2 + c * B_3 + d * B_4$
  - Gleichungssystem: 4 Gleichungen für a,b,c,d
  - Berechnung der Frequenzanteile a bis d über Gleichungssystems $a = P * B_1$

DCT - Discrete Cosinus Transformation

- Spezielle Basistransformation: Cosinus Transformation
- Jedes Pixel im Cosinus Raum entspricht einem Bild mit Cosinus-Funktionen verschiedener Frequenzen oder Phasen
  - Links oben: Frequenz = Null (Durchschnittswert)
  - Rechts unten: Anteil der höchsten Frequenz
- Cosinus Raum bildet ein Orthonormalsystem
- ein Pixelbild im Ursprungsraum lässt sich zusammensetzen als gewichtete Addition von Bildern mit unterschiedlichen Frequenzen → Spektralzerlegung
- Ähnlich funktioniert die Fouriertransformation (Sinus und Cosinustransformation)

Fouriertransformation

- jede periodische Funktion lässt sich darstellen als Summe von sin und cos Funktionen
- Transformation: stellt Funktion nur anders dar; umkehrbar
- Hochpassfilter: ausblenden der tiefen Frequenzen
- Tiefpassfilter: ausblenden der hohen Frequenzen

Signalrekonstruktion

- Abtastfrequenz viel höher als Signalfrequenz im Original
- Signalfrequenz liegt unter halber Abtastfrequenz, der Nyquistfrequenz
- Samplingtheorem von Nyquist
  - Signale unterhalb der Nyquistfrequenz können rekonstruiert werden
  - Signale oberhalb der Nyquistfrequenz können nicht rekonstruiert werden
  - Aliasing-Effekt

Anitaliasing

- Aliasing bei Bildern entsteht u.a. wenn das Bild in Bezug auf das Abtastraster zu hohe Frequenzen enthält
- die höchste zulässige Frequenz wird als Nyquistfrequenz bezeichnet
  - gleich der halben Abtastfrequenz $K_{x,Nyqu} = \frac{1}{2 * \delta x}$, $x = Pixelabstand$
  - Nachträgliches Filtern kann Aliasing nicht beseitigen
- Rekonstruktion eines Signals mittels Interpolation durch Erzeugen weiterer Samples zwischen den gemessenen Samples (Supersampling)
- Dennoch entsteht ein etwas gestörtes Signal → abgemilderte Aliasing
- bei Rückwärtstransformation ist Samplingrate durch Zielbild bestimmt
- Bei Verkleinerung um ein Faktor 2 ist die Samplingrate im Originalbild nur halb so groß wie die Nyquistfrequenz des Originalbildes!
- Der Begriff der Nyquistfrequenz erklärt das bekannte Phänomen und verweist auf mögliche Lösungen
- Aliasing kann bei der Ausgabe von Graphiken oder Bildinhalten auf Ausgabegeräten entstehen, denn dieser Prozess ist in der Regel mit einer Rasterung (Rasterisation) verbunden
- Aliasing entsteht insbesondere dann, wenn die Originale in viel höherer Auflösung vorliegen als das Raster des Ausgabegerätes
- Aliasing kann auch bei digitalen Fotografien von kleinteiligen (analogen) Mustern entstehen
- Da sich Aliasing-Artefakte nicht nachträglich beseitigen lassen, kann Anti-Aliasing nicht erst dann ansetzen, wenn das Bild bereits fertig gerendert ist
- Tiefpassfilterung muss deshalb immer vor dem Resampling angewendet werden

Tiefpassfilter

Um Bild korrekt rekonstruieren zu können, muss zuerst im Originalbild die hohen Frequenzen, die oberhalb der Nyquistfrequenz des Zielbildes liegen, eliminiert werden.

- Zuerst im Originalbild hohe Frequenzen, die oberhalb der Samplingrate des Zielbildes liegen, eliminieren
- Danach Sampling des gefilterten Originalbildes durch inverse Transformation jedes Pixels im Zielbild
- Die höchsten Frequenzen des Originalbildes können aufgrund der zu geringen Auflösung im Zielbild nicht dargestellt werden.
- Reihenfolge wichtig: Wenn man zuerst sampelt und dann das Zielbild filtert, lässt sich u.U. die Originalinformation nicht mehr rekonstruieren

Rekonstruktionsfilter

- Nach Eliminierung der hohen Frequenzen ist die Rücktransformation erforderlich.
- selbe Ergebnis einfacher, indem die Filterfunktion vom Frequenzraum in den Ortsraum transformiert und direkt im Ortsraum angewendet wird
- Box-Filter = ideales Tiefpass-Filter in Frequenzraum; eliminiert alle Frequenzen oberhalb
- Box-Filter im FR = sinc-Funktion im Ortsraum $sinc(x) = \frac{sin(x)}{x}$

Rekonstruktion von Zwischenpixelwerten durch Interpolation benachbarter Pixel

- Inverse Transformation der Koordination vom Zielbild + Sampling im Originalbild
- Bei einer Vergrößerung findet hierbei eine Überabtastung statt
- Zur genauen Rekonstruktion von Zwischenwerten kann man interpolieren
- Dadurch werden die scharfen unscharf aber die wahrgenommene Genauigkeit nimmt zu → Antialiasing
- Zur Gewichtung können ebenfalls Filterfunktionen im Ortsraum verwendet werden
- Cut-off-Frequenz = Nyquistfrequenz im Originalbild

Exakte Interpolation

Rekonstruktionsfilter (supersampling) per Sinc-Funktion (Exakte Interpolation):

- Inverse Transformation: $B(k,l) = A(h^{-1}\{k,l\}) = A(x,y)$
- Interpolationsansatz 1 für das Resampling:
  - gewichtete Überlagerung der Abtastwerte aus der lokalen Umgebung des Quellbildes
  - Gewichte werden durch Interpolationsfunktionen festgelegt
  - Interpolationsfunktionen bestimmen Qualität des Zielbildes
  - $A(x,y) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} A(i,j) * f_{int}(x-i, y-j)$
  - Filterkern f_{int} ist zentriert um das Pixel $A(x,y)$
  - Filter ist selbst auch als Rasterbild definiert

Anwendung der Filter auf Bildinformationen:

$$G(x,y) = \sum_{m=-size}^{size} \sum_{n=-size}^{size} F(x+m, y+n) * H(m,n)$$

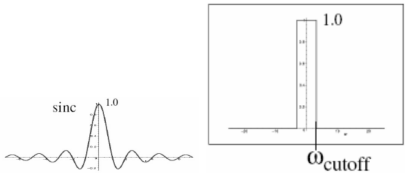
mit Ausgabebild $G(x,y)$, Eingabebild $F(x+m, y+n)$ und Filterkern $H(m,n)$
Beispiel Filterkern: 5x5 Binominalfilter

$$H_{5x5} = \frac{1}{256} * \begin{pmatrix} 1 \\ 4 \\ 6 \\ 4 \\ 1 \end{pmatrix} (1 \quad 4 \quad 6 \quad 4 \quad 1) = \frac{1}{256} \begin{pmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{pmatrix}$$

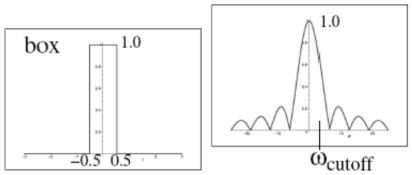
Interpolatiponsansatz 1 / Methode 1: Exakte Interpolation

- Hinreichend bandbegrenzte Bilder lassen sich theoretisch exakt interpolieren: Multiplikation des Quellbild-spektrum mit Rechtecktiefpass im Frequenzbereich
- Im Ortsbereich führt das auf eine diskrete Faltung mit der sinc-Funktion.
- In der Theorie gibt es keinerlei Störungen / keine Artefakte.
- Praxisproblem: die sinc-Funktion ist unendlich ausgedehnt!

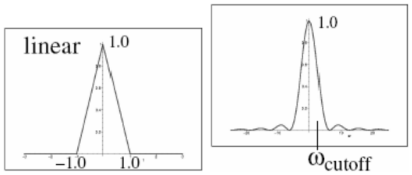
Sinc Filter
- theoretisch ideal (scharfe Grenze) bei gleichmäßiger Abtastung über der Nyquist-Frequenz
- besonders bei Kanten starke Ringing-Artefakte
- zur Berechnung des Farbwerts eines Pixels alle Abtastwerte des Bildes herangezogen
- einfaches Abschneiden der Sinc-Funktion (Lanczos-Filter) führt zu schlechten Ergebnissen
- Ortsraum & Frequenzraum:



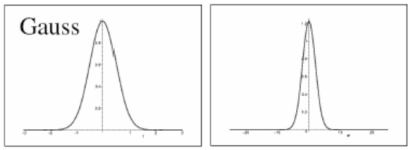
Box-Filter
- Alle Abtastwerte innerhalb eines um das Pixel gelegte Quadrat haben die gleiche Gewichtung → Mittelwertbildung
- allgemein schlechte Ergebnisse, da Fourier-Transformierte eine Sinc-Funktion, die gewünschten Frequenzbereich schlecht isoliert



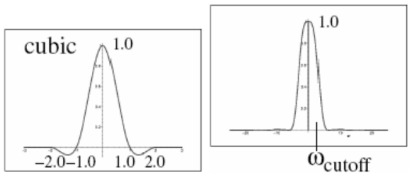
Kegel-Filter
- Gewichtung fällt mit zunehmender Distanz zum Pixel linear ab
- etwas bessere Ergebnisse als das Box-Filter
- Artefakte sind vorhanden, aber sehr abgeschwächt



Gauß-Filter
- Fouriertransformierte einer Gauß-Funktion ist Gauß-Funktion
- Filter führt zu Unschärfe
- Aliasing-Effekte sehr gut unterdrückt



Mitchell-Netravali-Filter
- sind stückweise kubische Filter mit vier Pixel breiten Trägern
- durch zwei freie Parameter änderbar
- untersuchen aus Rekonstruktionsfiltern resultierenden Artefakte
- Bei geeigneter Parameterwahl liefern die Filter einen guten Kompromiss zwischen Unschärfe, Anisotropie und Ringing.
- werden auch als bikubische Filter bezeichnet



Nearest Neighbour
- einfache Übernahme des geometrisch nächsten Nachbarn aus Quellbild
- Entspricht Faltung mit einem Rechteck der Breite δx im Ortsbereich, d.h. Multiplikation des periodifizierten Bildspektrums mit einer Sinc-Funktion im Frequenzbereich
- Massive Störungen (Artefakte) durch skalierte, z.T. phasengespiegelte Reste von periodifizierten Frequenzanteilen
- Verunschärfungen halten sich in Grenzen, da die entsprechende sinc-Funktion bei der Nyquistfrequenz noch nicht wesentlich abgefallen ist
- unsymmetrische Operation führt zu frequenzabhängigen, örtlichen Verschiebungen

Bilinearer Interpolation
- Entspricht der Faltung mit einem Dreieck der Breite $2 * \delta x$ im Ortsbereich, d.h. Multiplikation des periodifizierten Bildspektrums mit einer $sinc^2$-Funktion im Frequenzbereich
- Reduziertes Aliasing / Artefakte durch schnelleren Abfall der $sinc^2$-Funktion gegenüber sinc
- merkliche Verunschärfung durch stärkeren Abfall bei der Nyquistfrequenz
- außermittige Interpolation führt zu frequenzabhängigen örtlichen Verschiebungen

Sobelgradient

$$H_{xS} = \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix}, H_{yS} = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}$$

- z.B. Kantenextraktion
- Differenzbildung (Ableitung) → hohe Frequenzen werden verstärkt!
- im Gegensatz dazu sind Tiefpassfilter Integralfilter = gewichtete Summe der Nachbarpixel