

Computergrafik-Pipeline:

Die Abbildung eines Modells eines Objektes / Szene auf ein Bild auf dem Bildschirm nennt man **Rendering**.

Eine konkrete Implementierung beschreibt die Rendering-Pipeline.

Rasterung / Rendern:

- Rendern bezeichnet die Erzeugung eines digitalen Bildes aus einer Bildbeschreibung. Bei 3D-Szenen sind räumliche Objekt-Daten Teil der Bild- oder Szenenbeschreibung. Die Software dazu wird als Renderer bezeichnet (Wikipedia)
- Rendern ist die Zerlegung aller darzustellenden geometrischen Objekte in Bildschirmpunkte. (Bender)
- Transfer von Daten aus den Objektraum in den Bildraum.

Problem: Darstellung von Linien

Lösung: **Bresenham-Algorithmus** für Geraden

Abweichung zwischen exaktem Punkt und Mitte zwischen beiden möglichen Rasterpunkten. Der Punkt der näher zur Geraden liegt wird gezeichnet mittels Entscheidungsgröße E .

Aliasing:

Ist die fehlerhafte Rekonstruktion eines kontinuierlichen Ausgangssignals durch eine Abtastung mit zu geringer Frequenz. (Nyquist Theorem)

Anti-Aliasing:

alle Methoden, wie **Oversampling**, **Supersampling** und **Filterung** um Aliasing-Effekten entgegenzuwirken.

Vorteile:

- Eckige Kanten und harte Übergänge verschwinden
- Informationsgehalt nimmt zu

Nachteile:

- verringerte Schärfe
- kleine Schriften werden beschädigt
- Größere Dateien
- Längere Rechenzeiten
- ein Beseitigen ist nicht möglich

Oversampling (Überabtastung) / Supersampling

- ein Signal mit einer höheren Abtastrate wird bearbeitet als für die Darstellung der Signalbandbreite benötigt wird.
- ist eine Technik, bei der man das Bild in einer höheren Auflösung berechnet und dann herunterrechnet, womit Farbmittelwerte entstehen,

Monte-Carlo-Methode:

Stochastisches Sampling.

Intensitäten an zufälligen Punkten im Pixel ermitteln

- Problem:**
- Effizienz wird erhöht, neigt aber zum Flimmern von Objekten in Animationen
 - Verwedung bei Beleuchtungsmodellen nicht reproduzierbar

Vorgehensweise:

1. Form wird quadratisch umschlossen
2. zufällige Verteilung beliebiger Punkte
3. Prozentsatz der Punkte innerhalb der Form ergeben den Flächeninhalt (Graustufenwert)

Koordinaten Systeme:

Verschiedene Koordinatensysteme im 2D und 3D Raum

lokales, objekt, modelling:

- für jedes Objekt
- Lichtquellen

world, global, scene:

- Transformationen für Objekte
- zum Rendern - Umrechnung lokale in globale Koordinaten

eye, camera:

- Konzept der virtuellen Kamera
- zum Rendern - Umrechnung lokale in globale Koordinaten

view, view plane:

- Zuordnung von 3D zu 2D Koordinaten (Rasterung)
- Umrechnung von reellen zu integer Zahlen

Unterschied zwischen Window und Viewport:

Window: (view window)

- definiert ein **Sichtfenster** in der Bildebene
- definiert Teilbereich einer Szene
(Bereich aus der Szene; Kamerasicht)

Viewport:

- definiert **Bildschirmbereich**, wo der Inhalt eines Windows dargestellt werden soll
(Position der Darstellung auf dem Display)

Transformationen:

Sichtweise A: Transformation der Punkte

- Koordinatensystem bleibt fest
- Punkt ändert Ort im gleichen Koordinatensystem

Sichtweise B: Transformation des Koordinatensystems

- Koordinaten wird der inversen Transformation der Punkttransformation unterworfen
- Koordinatensystem bewegt sich
- Punkte bleiben fest

Sichtweise C: Transformation mittels Koordinatensystem

- ein dem Punkt zugeordnetes lokales Koordinatensystem wird mittels Punkttransformation im globalen Koordinatensystem in umgekehrter Reihenfolge bewegt

mögliche Transformationen:

- Translation (Verschiebung)
- Rotation (Drehung)
- Skalierung (Änderung der Größe)
- Projektion (Abbildung aus einem Raum der Dimension n in einen Raum mit einer kleineren Dimension als n)

affine Transformation:

- verzerrt nicht
- benutzt Einheitsvektoren

Clipping:

ein Verfahren mit dem alle außerhalb des Fensters liegenden Objektteile abgeschnitten werden. Dieses ist auch im 3D Raum möglich, das den Vorteil hat, dass nur tatsächlich sichtbare Objekte transformiert werden müssen.

Cohen-Sutherland Line-Clipping Algorithmus:

Bestimmt Ort einer Line -> innerhalb, außerhalb, schneidend des Fensters
Jeder Region wird ein eindeutiger 4-Bit-Code zugeordnet der Auskunft über die Lage in Bezug auf das Fenster gibt.

Culling:

Culling charakterisiert ein Verfahren, das mittels einfachen Auswählens ganze Grafikobjekte oder -primitive aus der weiteren Verarbeitung ausschließen kann. Sie werden nicht weiter berücksichtigt.

Back-Face-Culling:

In Abhängigkeit von der Position des Betrachters werden die Rückseiten undurchsichtiger Körper entfernt. Entschieden wird aufgrund der Normalen.

Ist das **Skalarprodukt** des Viewing Vectors und der entsprechenden Normalen größer null, wird das Face dargestellt, ansonsten nicht.

Boundary Representation:

- Darstellung eines 3D Objektes durch das Objekt begrenzten Flächen

Topologie:

- Wie sind die Punkte miteinander verbunden?
- Struktur des Modells (Anzahl und Richtung der Punkte)

Geometrie:

- Spezifikation der räumlichen Lage
- genaue Angaben der Koordinaten eines Objektes

Grundbausteine:

- Ecken / Eckpunkte / Vertices (V)
- Kanten / Edges (E)
 - Unterscheidung zwischen "realen" Kanten und Kanten, die nur der polygonalen Verfeinerung dienen.
- Flächen / Faces (F)

Polyeder:

Ein Körper, der von Ebenen begrenzt wird

Eulersche Polyedersatz:

$$V - E + F = 2$$

Mannigfaltigkeit:

- die Umgebung einer Ecke ist eindeutig (homeomorph) auf einen Kreis oder Halbkreis abbildbar
- eine Kante gehört zu nicht mehr als 2 Faces

Polygonale Netze:

- Topologie und Geometrie muss explizit gespeichert werden
- Möglichkeiten:** explizite Speicherung, Eckenliste, Kantenliste

Erzeugung polygonaler Objekte:

- mathematische Verfahren
- prozedurale Verfahren
- Boolean

Metaballs (Blobs):

Beschreibung von Objekten als Isoflächen in Skalarfeldern

Skalarfeld:

Ein Raum indem jeder Punkt spez. Daten hat, wie z.B. die Temeperatur. Alle Punkte die bsp. 35° haben bilden ein Skalarfeld.

LOD - Level of Detail:

- Grad der Komplexität einer Szene
- findet großen Einsatz in der Spieleindustrie - flüssiges switching in Szenen (i.d.R. Hardware abhängig) zwischen verschiedenen "polygonalen Auflösungen"
- **Geomorphs** sind Übergänge von LODs.

Problem: Vermeidung von Sprüngen zwischen einzelnen Auflösungen

Szenengraph:

hierarchische Baumstruktur zur Speicherung einer Sezne mit ihren Modellen

Teselation, Triangulation:

Prozess, der aus der mathematischen Beschreibung eine Menge von Dreiecken produziert.

Objektraum:

- 3D Raum der Szene
- geräteunabhängig
- Genauigkeit ist Abhängig von der Hardware (Maschinengenauigkeit)

Bildraum:

- 2D Raum des Ausgabegerätes
- geräteabhängig
- Genauigkeit ist Auflösung des Ausgabegerätes

z-Buffer:

- bestimmt Sichtbarkeit von Bildpunkten im **Bildraum**
- führt eine Suche nach demjenigen Polygon durch, dessen z-Wert am größten ist, d.h. am weitesten vorn liegt.
- Zur Realisierung wird zusätzlicher Speicher, der sog. z-Buffer, benötigt

z-Buffer-Algorithmus:

1. init. Bildspeicher (**Frame-Buffer**) mit HG Farbe
2. init. z-Buffer mit min. z-Wert
3. Scanne Polygone
 - a. berechne z-Wert für jedes Pixel im Polygon
 - b. ist der Wert größer als der bisherige Wert zeichne Farbe im Bildspeicher

Vorteile:

- einfache Implementierung
- unabhängig vom Objekt
- keine Beschränkung der Komplexität
- keine vorgeschriebene Reihenfolge

Nachteile:

- Auflösung des z-Buffers bestimmt Diskretisierung der Bildtiefe
- hoher Speicherbedarf
- Transparenzen nur durch aufwendige Modifikationen berücksichtigbar

Ray Casting:

- Löst Sichtbarkeitsproblem
- Verfolgt Strahlen vom Augpunkt durch alle Pixel der Bildebene
- Schnittpunktberechnung mit allen Objekten der Szene
- der naheliegende Schnittpunkt ist in diesem Pixel sichtbar

Problem: für jeden Strahl muss jedes Objekt der Szene auf Schnittpunkte getestet werden

- Lösung:**
- **Bounding Volumes** - komplexe Objekte mit einfach zu testenden Objekten umschließen. Haben die BV keinen Schnittpunkt mit einem Strahl, so sind auch die darin enthaltenden Objekte nicht auf ein Schnittpunkt zu testen.
 - **Hierarchien** - baumartige Struktur von Bounding Volumes
 - **Raumteilung** - gleichmäßige Aufteilung von Bounding Boxes in gleichgroße Abschnitte. Jede Unterteilung enthält eine Liste mit komplett oder teilweise enthaltenden Elementen, d.h. nur wenn ein Strahl eine Partition schneidet, müssen Schnittpunktberechnungen mit dem assoziierten Objekten durchgeführt werden.

Beleuchtung / Lichtquellen:

- **Punktlicht** - Licht strahlt in alle Richtungen gleichmäßig
- **Richtungslicht** - Licht strahlt aus der Unendlichkeit in eine Richtung - Strahlen sind parallel
- **Spotlicht** - Licht strahlt in einem Kegel
- **Flächenlichtquellen** - Weiche Ausleuchtung

WICHTIG: Punkte (3D) werden beleuchtet, Pixel (2D) werden schattiert!

direktes Beleuchtungssystem (lokal):

- kein indirektes Licht
- keine Schatten
- Beispiel: Phongbeleuchtung

indirekte Beleuchtungssystem (global):

bezeichnet die Simulation aller Möglichkeiten der Ausbreitung von Lichtstrahlen in einer Szene. Dadurch werden die Gesetze der geometrischen Optik sowie der Energieerhaltung vollständig erfüllt und ein relativ realistisches Bild erzeugt.

- Berechnung der Intensität (Farbe) in Abhängigkeit vom direkten und indirekten Licht
- **Ray-Tracing** und **Radiosity**

shading model:

- bestimmt wann ein **Beleuchtungsmodell** angewendet wird

Phongbeleuchtungssystem:

- lokales Beleuchtungssystem
- empirisches Modell (mit guten Ergebnissen)
- simuliert folgende Reflexionsphänomene:
 1. idealer Spiegel
 - perfekte spiegelnde Reflexion (perfekter Spiegel)
 - ohne Aufstreuung in Reflexionsgesetz reflektiert
 2. unvollkommener Spiegel
 - Lichtstrahl wird aufgespalten, es entsteht ein Reflexionskegel
 - Obeflächenzusammensetzung aus vielen kleinen verschieden ausgerichteten Spiegeln
 3. perfekte diffuse Reflexion
 - Lichtstrahl wird perfekt in alle Richtungen gleichmäßig gestreut

reflektierendes Licht besteht aus den Anteilen:

- **diffus** (Streulicht)
- **specular** (Glanzlicht)
 - Farbe wird durch die Farbe der Lichtquelle bestimmt
- **ambient** (Umgebungslicht)
 - ambiente ist oft konstant und simuliert so die globale Beleuchtung (einfache Addition von Konstanten; ersetzt Raytracing)

(diffuser und spiegelnder Anteil werden lokal berechnet)

Geometrie-Informationen:

außer der Normaleninformatmion werden keine weiteren Geometrie-Informationen benötigt

Oberflächenphysik:

Die Physik der Oberfläche wird über die Verhältnisse der einzelnen Komponenten modelliert.

Es gilt: $k_d + k_s + k_a = 1$

$$I = k_d * I_d + k_s * I_s + k_a * I_a$$

(k = konstante | I = Intensität)

Spiegelnde Reflexion: (im Term $K_s * I_s$)

es ist ein Abbild der Lichtquelle auf einer Oberfläche, das sog. **Highlight**. Um ein Highlight zu sehen muss die Betrachtungsrichtung (V) nahe der Reflexionsrichtung (R) sein.

Problem:

- Für verschiedene Lichtvektoren (L) entsteht immer der gleiche Reflexions-Intensitätskegel, d.h. die Intensität der speigelden Reflexion hängt nicht von der Ausrichtung des Lichtvektors ab!
- Objektoberflächen wirken plastikhaft

Bemerkung:

Aus Geschwindigkeitsgründen wählt man zur Berechnung des Reflexionsvektors (R) statt dem Skalarprodukt von $R*V$ das Skalarprodukt von $N*H$, da dieser Winkel mit dem vorherigen Identisch ist. H wird berechnet durch $H=(L+V) / 2$

BRDF (bi-directional reflection distribution function):

die BRDF beschreibt, wie sich die Reflexion verteilt. Die Bezeichnung betont insbesondere die Abhängigkeit des in einer beliebigen Richtung reflektierten Lichts (R) von der Richtung des einfallenden Lichts (L).

**Nachteile lokaler Beleuchtungsmodelle:
i.d.R. nur direkte Beleuchtung, d.h. kein Schattenwurf, keine
Interaktion mit anderen Objekten**

Interpolative Schattierungstechnik: Flatshading

- pro Face wird das verwendete Beleuchtungsmodell genau einmal ausgewertet.
- Grundlage bildet die **Polygonnormale** oder **Flächennormale** im **Objektraum**. Die Auswertung erfolgt meistens im Polygonschwerpunkt oder in einem Eckpunkt
- keine Interpolation
- Kanten bleiben sichtbar -> unstetiger Verlauf
- runde Objekte nur durch sehr viele Polygone darstellbar
- Anwendung: **Entwurfsansicht**

Gouraud und Phong Shading: Gemeinsamkeiten

Versuch durch Interpolation zwischen einzelnen Faces Kanten zu glätten, bzw. verschwinden zu lassen. Die Interpolation findet im **Bildraum** statt.

Dabei wird zuerst in einer ersten Interpolation der Wert (P_a) zwischen zwei Eckpunkten (P_1 , P_2) zwischen eines Faces ermittelt. Danach werden alle Werte (Q_1 - Q_n) ausgehend von P_a waagrecht bis P_b mittels Interpolation ermittelt.

Diese Algorithmus arbeitet i.d.R. scanlineweise und inkrementell. (Rechnung s. §4-80)

Grundlage stellt die Eckpunktnormalen der angrenzenden Faces dar. Diese entsteht aus Mittelung aller Polygonnormalen.

Interpolative Schattierungstechnik: Gouraud Shading

- Interpoliert Farben
- Auswertung erfolgt ausschließlich über Polygoneckpunkte und den Eckpunktnormalen im **Scanline Verfahren**.

Verfahren:

zunächst werden die Farben des darzustellenden Polygons an seinen **Eckpunkten** berechnet und die Vertizes auf die **Bildebene** projiziert. Das so entstandene zweidimensionale Abbild des Polygons wird anschließend **zeilenweise** abgearbeitet. Dabei werden die Farben an den Schnittpunkten der Kanten mit der Abtastzeile aus den Farben der Eckpunkte **interpoliert**. Die Farbwerte der Bildpunkte der Abtastzeile werden wiederum aus den Farben der Kanten interpoliert.

Ergebnis:

- Kanten werden geglättet, stetiger Verlauf, aber nicht glatt
- **Machband-Effekte** entstehen bei zu geringer polygonaler Unterteilung
- Highlights werden "schwammig" dargestellt.

Gouraud Shading stellt das Standard Schattierungsverfahren heutiger Graka dar!

Interpolative Schattierungstechnik: Phong Shading

ist auch als Normalenvektor-Interplations-Shading bekannt.

es werden an den **Eckpunkten** eines Polygons die Normalen berechnet und dann erweitert durch eine Berechnung von **interpolierten Normalen** entlang der Kanten für jede Projektion des Polygons auf einen Pixel. Farbstärken berechnen sich aus den interpolierten Normalen. Jeder projizierter Punkt der Oberfläche wird über das Beleuchtungsmodell ausgewertet

Ergebnis:

- Intensitätsverlauf ist stetig und glatt
- rechenaufwändig
- Highlights werden gut dargestellt

Nur von High-End Grakas unterstützt

Berechnung:

- Addition von diffuser Reflexion , ambienster und spekulärer Anteil
- diffuse und spekulare Reflexion wird nur lokal modelliert!

Lechners Gesetz:

Besagt, dass die Beziehung der real ins Auge fallenden **Lichtintensität** und der vom Auge wahrgenommenen nicht linear, sondern **logarithmisch** ist!

Es werden Helligkeitsunterschiede in dunklen Regionen besser wahr genommen als in hellen!

Mach Band Effekt:

Das Auge betont scharfe **Intensitätsänderungen**. Dieses führt zu einer automatischen **Konturenschärfe**.

Dies hat Auswirkungen auf das Rendering:

- Flatshading: un stetige Intensitätenwechsel: starker Mach Band Effekt
- Gouraud Shading: Mach Band Effekt ist abhängig von Polygonalisierung
- Phong Shading: glatte Intensitätenwechsel reduzieren Mach Band Effekte

Ray-Tracing: (rekursive Strahlenverfolgung | global)

- Ray Castings + Strahlenverfolgung für reflektierte und gebrochene Strahlen, Schattenberechnung
- abhängig vom Augpunkt
- führt zu **harten** Bildern
- Lichtstrahlen werden rückwärts vom Augpunkt über betreffende Oberflächen zu den Lichtquellen berechnet.
- Lichtstrahlen werden von jedem Pixel in die Szene verfolgt und bei jedem Schnittpunkt mit einem Objekt werden direkte, reflektierende und transmittierten Lichtanteile bestimmt.
- impliziert **Baumstruktur**, d.h. Komplexität steigt stark exponentiell

Abbruch der Strahlenverfolgung:

- Strahlenverfolgung beendet (nur bei einfachen Szenen)
- Reflexionsintensität unterschreitet ein bestimmtes Limit
- bestimmte Rekursionstiefe erreicht (ist die Regel)
- kein Speicher

Schattenberechnung:

- Verfolgung eines Strahls von einem gefundenen Schnittpunkt zu allen Lichtquellen
- Schneidet dieser Strahl ein Objekt, liegt der Schnittpunkt im Schatten dieses Objekts
- Strahl heißt Schattenfühler (nur Software)

Distributed Ray-Tracing:

- keine Spiegelung in der Realität spiegelt zu 100% ohne Schleier
- Distributed RT ermöglicht **unscharfe Effekte**, indem nicht nur ein Strahl reflektiert wird sondern mehrere (birnenförmig)
- weitere Erhöhung der Realistik ist die Verwendung von **flächigen Lichtquellen**. Hierbei werden viele Lichtstrahlen zu einer Lichtquelle gelegt.
- durch Einsatz von **Blendentechnik** erhält man sogar fotorealistische Bilder

Supersampling:

- es wird nicht nur ein Strahl pro Pixel verwendet sondern 5. Bei den Eckpunkten und im Schwerpunkt des Pixels.
- Weisen die Ergebnisse eine große Abweichung auf, wird das Pixel erneut unterteilt und neu gestartet bis ein gleichmäßiges Ergebnis erzielt wurde.

Alternativ: Stochastic Ray Tracing:

- Strahlen werden zufällig durch das Pixel geschossen.

Vorteil: führt schneller zum besseren Bild

Nachteil: nicht reproduzierbar

Vorteile:

- realitätsnahe Physik
- hervorragende Spiegelungen
- sehr real

Nachteile:

- nicht gut bei diffusen Reflexionen
- harte Bilder
- rechenintensiv (Schnittpunktberechnung)
- anfällig für numerische Probleme

Radiosity:

Im Radiosity-Verfahren wird für jede Fläche eine Gleichung aufgestellt, die das emittierte Licht aus dem von den anderen Flächen empfangenen Licht und evtl. ihrer eigenen Leuchtkraft bestimmt. Insgesamt ergibt sich damit ein Gleichungssystem, dessen Lösung die Helligkeit jeder einzelnen Fläche angibt.

- Trennung von **Sichtbarkeitstests** und **Schattierung**
- Alle Interaktionen des Lichts mit Objekten werden **vorberechnet**
- unabhängig vom Augpunkt
- **diffuse** Reflexionen
- führt zu **weichen** Bildern (Architekturvisualisierung)
- Ausbreitung des Lichts unter Beachtung des **Energiegleichgewichts** in einem geschlossenem System (s.u.)
- Für jede Fläche wird das ausgesandte / reflektierte Licht bei allen anderen Flächen berücksichtigt
- Berechnung benötigt: alle geometrischen Informationen aller strahlenden, reflektierenden und transparenten Objekte und die lichttechnischen Kenngrößen aller Körper

Energieerhaltungssatz:

Alles Licht, das eine Fläche empfängt und nicht absorbiert, muss sie wieder emittieren. Außerdem kann eine Fläche auch selbstleuchtend sein.

Darstellung einer Szene:

1. Berechnung der Strahlenwerte R_i für alle Flächen A_i
2. Abbildung der Szene und Bestimmung der sichtbaren Teile
3. Berechnung der Farbe für jedes Pixel

Bemerkung:

- Schritt 1 wird einmal ausgeführt
- für neue Ansichten müssen nur die Schritte 2 und 3 wiederholt werden

- für Schritt 1 müssen vor der Lösung des Gleichungssystems die Formfaktoren F_{ij} berechnet werden.

Formfaktor:

Der Formfaktor F_{ij} gibt an, welcher Anteil der über dem i-ten Patch liegenden Hemisphäre vom j-ten Patch bedeckt wird

Ein Formfaktor gilt immer zwischen zwei Patches und beschreibt die Menge der ausgetauschten Strahlung, liegt also zwischen null (keine Strahlung wird ausgetauscht) und eins (alle Strahlung wird ausgetauscht).

Der Formfaktor ist rein geometrischer Natur und wird durch die Stellung der Patches zueinander bestimmt. Des Weiteren spielt die Sichtbarkeit der Patches eine Rolle. Die Sichtbarkeitsberechnung braucht bei weitem die meiste Zeit in der Berechnung.

Vorteile:

- Berechnung von Standort und Blickwinkel ist unabhängig
- Radiosity-Berechnung nur 1 mal
- Diffuse Reflexionen sind impliziert

Nachteile:

- kann nur ideal diffuse Reflexionen
d.h. keine glänzenden oder ideal spiegelnde Reflexionen
- Simulation von globaler Beleuchtung nur teilweise effizient.
Realisiert mittels Photon-Mapping oder Monte-Carlo-Raytracing

Normalen:

Wann müssen Normalen normalisiert sein?

Zur Berechnung des Winkels (\cos) zwischen zwei Vektoren (z.B. Lichtvektor (L) und Normalenvektor (N)) ist eine Wurzelberechnung erforderlich. Diese entfällt aber bei normalisierten Vektoren. Einsatz finden normalisierte Vektoren bsp. bei der Berechnung der diffusen und specular Phongbeleuchtung!

$$\cos = (L \cdot N) / (|L| * |N|) \quad \text{wird zu:} \quad \cos = L \cdot N$$

Was passiert, wenn nicht normalisiert wird?

Da die Hardware bei der Formel (s.o.) nicht durch den Betrag der 2 Vektoren dividiert, wird das Ergebnis bei Vektoren, die vor einer Normalisierung größer sind heller, bei Vektoren, die kleiner sind dunkler.

Bei Flächen, die in der Szene nicht gesehen werden, ist das vernachlässigbar.

In **OpenGL** werden Normalen werden durch Skalierung von Objekten, ebenfalls skaliert. So können Flächen bei nicht affinen Transformationen unterschiedlich beleuchtet werden.

Wann müssen Normalen nicht normalisiert sein?

Eckpunktnormale berechnen sich aus der Addition aller beteiligten Flächennormalen.

Die Flächennormalen müssen hierzu nicht normalisiert sein, sie müssen aber die **gleiche Länge** haben! Eine Normalisierung der Eckpunktnormale findet im Anschluss statt.

Backface-Culling:

Beim Backface-Culling spielt die Länge der Flächennormalen keine Rolle, lediglich die Richtung ist wichtig! Eine Normalisierung ist für diesen Rechenschritt unnötig.

Kohärenz:

ist die Ausnutzung lokaler **Ähnlichkeiten**.

Beispiele:

Flächenkohärenz: Eigenschaften benachbarter Punkte auf einer Fläche ändern sich oft nur unwesentlich, z.B. Farbe.

Tiefenkohärenz: Die Tiefe (x,y) auf einer Fläche kann oft inkrementell berechnet werden.
z.B. z-Buffer

zeitliche Kohärenz: z.B. Cache, Zwischenspeicherung

örtliche Kohärenz: z.B. Übertragung von Blöcken

Konvex:

Als konvex (lat.: convexus gewölbt, gerundet) bezeichnet man Formen (Flächenteile, Linien), die nach **außen** gewölbt sind.

Konkav:

Als konkav (lat.: concavus ausgehöhlt, einwärts gewölbt) bezeichnet man Formen (Flächenteile, Linien), die nach **innen** gewölbt sind.

Rendering Pipelines:

beschreibt beim Echtzeitrendering den Weg von der vektoriellen, mathematischen Beschreibung einer Szene zum gerasterten Bild.

Die Renderpipeline übernimmt dabei Dinge wie die Umrechnung von Bildschirmkoordinaten in Gerätekoordinaten, Clipping oder in einigen Fällen das Antialiasing.

genereller Aufbau:

- Display traversal - heute nicht mehr benötigt
- Modelling transformation - Objekte werden gezeichnet
- Viewing operation - Kamera
- visible-surface determination | scan-conversion | shading - Sichtbarkeitstests (Culling, Clipping), z-Buffer, Shader, Beleuchtung

lokales Beleuchtungsmodell: (Gouraud Shading mit z-Buffer)

- db traversal
- modelling transformation
- trivial accept / reject - 3D Culling
- lightning - nach Phongmodell
(Daten: x, y, z Koord. inkl. Normale | Farbe)

- Viewing transformation - Projection in 2D
- Clipping - mittels Sutherland Algorithmus
(Daten: x, y, z Koord. | Farbe)
- Rasterization - Gouraud Shading + z-Buffer
- Display-Darstellung

lokales Beleuchtungsmodell: (Phong Shading mit z-Buffer)

- db traversal
- modelling transformation
- trivial accept / reject - 3D Culling
- Viewing transformation
- Clipping
(Daten: x, y, z Koord. **inkl. Normale** | Farbe)
- Rasterization inkl. Lightning
- Display

globales Beleuchtungsmodell: (Radiosity, Gouraud Shading mit z-Buffer)

- db traversal
- modelling transformation
- Vertex Intensity Calculation - Using Radiosity - Farben aller Flächen
- new db - Infos in neue db schreiben (Geometrie und Farbe)

-
- new db traversal
 - trivial accept / reject - 3D Culling
 - Viewing transformation
 - Clipping
 - Rasterization
 - Display

(1. Schritt wird nur einmal ausgeführt - s.o.)

globales Beleuchtungsmodell: (Raytracing)

- db traversal
- modelling transformation
- Raytracing
- Display

DirectX

- linkshändiges Koordinatensystem
- nur Windows
- direkter Hardwarezugriff
- besitzt Immediate (hardwarenah) und Retained Mode (abstrakt, modellorientiert)
- zuerst für Spiele entwickelt
- objektorientiert

OpenGL

- rechtshändiges Koordinatensystem
- Portabel (Linux)
- komplexer Hardwarezugriff (Windows)
- entspricht Immediatemode von Direct 3D
- professioneller Einsatz
- prozedural

Texture Mapping:

ist die Projektion zweidimensionaler Muster auf Oberflächen von Körpern
Simulation von Oberflächendetails mittels *Bitmaps*, die Einfluss haben können auf:

- Farbe
- Reflexion
- Normalenvektor
- Transparenz

Begriffe:

- Texture Map - das zu mappende Bild
- Texel - Einzelelement (Pixel) der Texture Map

Bemerkung:

Prinzipielle Unterscheidung in *forward* und *inverse mapping*.

Durchführung:

Mapping besteht aus 2 Arbeitsabschnitten (Beispiel Forward Mapping)

1. Textur wird auf eine Zwischenfläche projiziert, wie z.B. Box, Zylinder, Kugel.
Dieses stellt das sog. *s-mapping* (surface mapping) dar.
2. Von dort wird die Textur auf das wirkliche Objekt übertragen.
Dieses stellt das sog. *o-mapping* (original mapping) dar.

Texture Mapping und Aliasing:

- Problem:**
- Ein Pixel in Bildschirmkoordinaten kann nach der Rückprojektion auf die Textur den Bereich von mehreren Texels überdecken!
 - Ein Texel auf der Textur kann in Bildschirmkoordinaten mehrere Pixel überdecken!
 - Projektion eines quadratischen Pixels auf ein i.d.R. nicht quadratischen Texels

Bump Mapping:

ist eine Technik zur Darstellung von extrem detailreichen Objekten ohne die Geometriekomplexität des Objektes zu erhöhen. Nötige Informationen werden in eine Textur gepackt, mit deren Hilfe Schattierungen auf eine Oberfläche gezeichnet werden.

Bump Mapping verändert dazu nicht die Geometrie, sondern manipuliert die Normalen bei der Auswertung des Beleuchtsmodells.

Verfahren:

Die Veränderung der Normalenvektoren erfolgt prozedural oder mittels Grauwerten von Texture Maps.

Problem: Silhouette eines mit Bump Mappings dargestellten Körpers ist eben, reines Texture Mapping erzeugt nur glatte Oberflächen!

Displacement Mapping:

Über die eigentliche Oberfläche wird ein **Höhenfeld** gelegt, dessen einzelne Punkte in Richtung der Oberflächennormalen anhand einer Texture Map verschoben werden. Es werden also tatsächlich Oberflächenpunkte bewegt

Vorteil: es entsteht eine reale Silhouette, geometrie wird tatsächlich verändert

Nachteil: schwer kontrollierbare Polygonenanzahl

Opacity Mapping / Transparency Mapping:

Ähnlich dem Alpha-Kanal bei Bildern

Objekt kann entsprechend der verwendeten Bildvorlage transparent sein. (bsp. Wolken)

Procedural Mapping:

algorithmische Beschreibung von Texture Maps. (bsp. Simulation von Unregelmäßigkeiten)

Environment Mapping:

Realistische **Spiegelungseffekte** ohne vollständiges Raytracing.

Erlaubt die Integration einer komplexen Umgebung in ein fotorealistisches Bild mittels Zwischenobjekt, ohne dass die Umgebung explizit modelliert werden muss.

Chrome / Reflection Mapping:

Abbildung eines willkürlichen Musters aus dem zweidimensionalen Texturaum auf eine reflektierende Oberfläche.

Fazit Mapping:

Mapping Techniken ...

- sind anfällig für Aliasing Effekte
- können miteinander kombiniert werden
- bilden die wesentliche Grundlage für alle heute kommerziell eingesetzten Computergrafik-Techniken

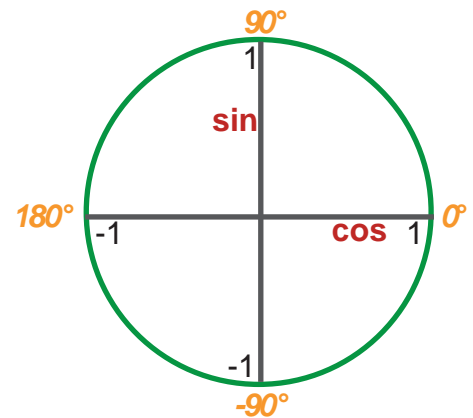
Zusammenfassung Formeln:

Darstellung der Transformationsmatrix (2D):

$$\begin{pmatrix} \boxed{1} & \boxed{0} & \boxed{a} \\ \boxed{0} & \boxed{1} & \boxed{b} \\ \boxed{0} & \boxed{0} & \boxed{1} \end{pmatrix}$$

Skalierung / Rotation
 Translation
 Projektion
 homogene Erweiterung

Einheitskreis:



Transformationen im 2D:

Rotation:

$$\begin{pmatrix} \cos_y & -\sin_y & 0 \\ \sin_y & \cos_y & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Skalierung:

$$\begin{pmatrix} s1 & 0 & 0 \\ 0 & s2 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Translation:

$$\begin{pmatrix} 1 & 0 & a \\ 0 & 1 & b \\ 0 & 0 & 1 \end{pmatrix}$$

Reflexion an der X-Achse:

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Reflexion an der Y-Achse:

$$\begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Reflexion an X-und Y-Achse

$$\begin{pmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Transformationen im 3D:

Translation:

$$\begin{pmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Skalierung:

$$\begin{pmatrix} s1 & 0 & 0 & 0 \\ 0 & s2 & 0 & 0 \\ 0 & 0 & s3 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Rotation X-Achse:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos_y & -\sin_y & 0 \\ 0 & \sin_y & \cos_y & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Rotation Y-Achse

$$\begin{pmatrix} \cos_y & 0 & \sin_y & 0 \\ 0 & 1 & 0 & 0 \\ -\sin_y & 0 & \cos_y & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Rotation Z-Achse

$$\begin{pmatrix} \cos_y & -\sin_y & 0 & 0 \\ \sin_y & \cos_y & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Betrag eines Vektors:

Der Betrag bestimmt die Länge eines Vektors

Normalisieren eines Vektors: (Einheitsvektor)

der Vektor e mit dem Betrag $|e| = 1$

Skalarprodukt:

Wenn Ergebnis den Wert 0 hat sind die Vektoren orthogonal zueinander

Kreuzprodukt / Vektorprodukt:

bestimmt orthogonalen Vektor auf die Fläche, die von den Vektoren a, b aufgespannt wird.

Winkel 2er Vektoren:

Winkel 2er Einheitsvektoren:

Flächeninhalt von Dreiecken:

Backface-Culling:

$\mathbf{v} \cdot \mathbf{N} > 0$ = sichtbar | $\mathbf{v} \cdot \mathbf{N} = 1$ = vollständig sichtbar | $\mathbf{v} \cdot \mathbf{N} \leq 0$ = nicht sichtbar

Anmerkung: Division durch den Betrag entfällt wegen Normalisierung

\mathbf{v} =Viewing Vector | \mathbf{N} = Flächennormale

Halfway Vektor (H):

Definition eines neuen Vektor $\mathbf{H} = (\mathbf{L} + \mathbf{V}) / 2$

anstatt $\mathbf{R} \cdot \mathbf{V}$ wird jetzt $\mathbf{N} \cdot \mathbf{H}$ betrachtet, da gleiches Verhalten

lokales Beleuchtungsmodell:

ohne Farbe:

$$\begin{aligned} I &= k_d * I_d + k_s * I_s + k_a * I_a \\ &= I_i * (k_d * (\mathbf{L} \cdot \mathbf{N}) + k_s * (\mathbf{R} \cdot \mathbf{V})^n) + k_a * I_a \end{aligned}$$

diffuse Reflexion:

$$I_d = I_i * \cos y \quad \text{bzw.} \quad I_d = I_i * (\mathbf{L} \cdot \mathbf{N})$$

specular Reflexion:

$$I_s = I_i * (\mathbf{R} \cdot \mathbf{V})^n \quad (\text{je größer } n \text{ desto perfekterer Spiegel})$$

mit Farbe: Für farbige Objekte (Lichtquellen) = wird das Modell getrennt auf die Farbkomponenten I_r , I_g , I_b angewendet.

$$\begin{aligned} I_r &= I_i * (k_{dr} * (\mathbf{L} \cdot \mathbf{N}) + k_{sr} * (\mathbf{N} \cdot \mathbf{H})^n) + k_{ar} * I_a \\ I_g &= I_i * (k_{dg} * (\mathbf{L} \cdot \mathbf{N}) + k_{sg} * (\mathbf{N} \cdot \mathbf{H})^n) + k_{ag} * I_a \\ I_b &= I_i * (k_{db} * (\mathbf{L} \cdot \mathbf{N}) + k_{sb} * (\mathbf{N} \cdot \mathbf{H})^n) + k_{ab} * I_a \end{aligned}$$

Anmerkung: ambient Light ist konstant!

Legende:

I = Intensität | k = Konstante | d = diffus | s = specular | a = ambient |

I_i = Intensität des einfallenden Lichts | y = Winkel zwischen \mathbf{N} und \mathbf{L}

Strahlensatz:

$$c / a = d / b$$

Anzahl der Dreiecke von Triangle- Strip und Fan:

$n + 2$ Dreiecke definieren n Dreiecke