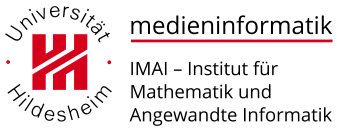
3D-Graphik

Jörg Cassens

Medieninformatik
WS 2018/2019



Einleitung

- Dreidimensionale Computergraphik ist ein Medium, dessen Bedeutung besonders in den letzten zwei Jahrzehnten zugenommen hat
  - Animationsfilme orientiert sowohl an klassischen Animationsfilmen wie auch am traditionellen Film
  - Völlig neue Bildsprache, da der 3D-Animationsfilm einerseits nicht an die Gesetze der Physik gebunden ist und anderseits die Grenzen des traditionellen Animationsfilms in Hinblick auf den Realismus überschreiten kann
  - Weiterhin treibende Kraft in der Spieleindustrie, die inzwischen das Umsatzvolumen der Filmindustrie übertrifft
  - Heutige Konsolen und PCs bieten eine Graphikleistung, die früher nur teuren Spezialrechnern vorbehalten war
  - Spiele werden immer realistischer und immer immersiver
- 3D-Graphik erfährt eine enorme technische Entwicklung und speist ein eigenes Forschungsgebiet
- Schnelle Innovationszyklen

Lernziele

- Kleiner und überschaubarer mathematischer Kern etablierter Verfahren
- Grundlegende Konzepte der 3D-Graphik
- Darstellung einer Verarbeitungskette
- Schritte zur Darstellung am Bildschirm
- Grundlegende 3D-Animationsverfahren
- Offene und verbreitete Codierungen für 3D-Graphiken

Lebenswelt

- Unsere Lebenswelt, wie wir sie wahrnehmen, hat 3 räumliche und eine zeitliche Dimension
- Die zeitliche Komponente haben wir uns im Bereich Video angesehen
- 3D-Darstellung in der Ausgabe heute noch nur beschränkt machbar
  - CNC-Maschinen
  - 3D-Drucker
  - (Pseudo-) Holographische Displays

- Ansonsten behelfen wir uns mit Tricks
  - Stereoskopische oder Auto-stereoskopische Wiedergabe
  - Das sind aber im wesentlichen zwei 2D-Bilder, die in einem speziellen Zusammenhang stehen

1 Grundlagen

Grundlegende Elemente und Funktionen

- Zur Darstellung von 2D-Ausgaben müssen ursprünglich in 3D berechnete Darstellungen auf 2D projiziert werden
- Dazu benutzen wir eine virtuelle Kamera
- Reale Kamera hat perspektivische Projektion, virtuell sind andere Projektionen möglich
- Zur Darstellung der Objekte sind noch Oberflächeneigenschaften und Lichtquellen wichtig
- Wir haben also: 3D-Modelle, Ihre Oberflächen, Lichtquellen, Kamera
- Für eine animierte Darstellung können diese Elemente beweglich oder veränderlich sein
- Ein standardisierter Verarbeitungsprozeß, die 3D Rendering Pipeline, hat sich etabliert und wird weitgehend in Hardware unterstützt

Koordinatensysteme

- Analog zu 2D: kartesisches Koordinatensystem mit linear gleich aufgeteilten Achsen
- Zwei Konventionen; rechtshändiges Koordinatensystem (häufiger) oder linkshändiges

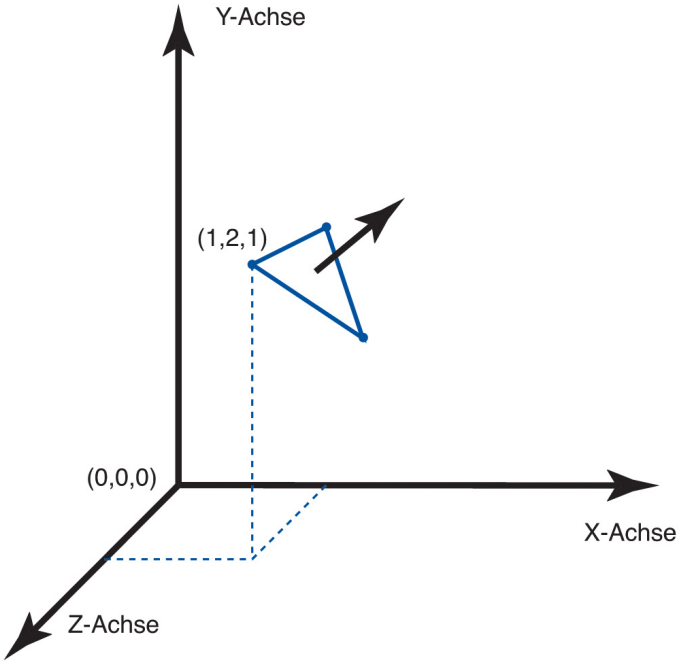


Abbildung 8.1: Rechtshändiges dreidimensionales Koordinatensystem, darin eingezeichnet ein Polygon aus drei Punkten mit seiner Flächennormalen

Lineare Transformationen

Translation als Addition

$$\begin{pmatrix} x_{neu} \\ y_{neu} \\ z_{neu} \end{pmatrix} = \begin{pmatrix} x_{alt} \\ y_{alt} \\ z_{alt} \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix} = \begin{pmatrix} x_{alt} + t_x \\ y_{alt} + t_y \\ z_{alt} + t_z \end{pmatrix}$$

Skalierung

$$\begin{pmatrix} x_{neu} \\ y_{neu} \\ z_{neu} \end{pmatrix} = \begin{pmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & s_z \end{pmatrix} \begin{pmatrix} x_{alt} \\ y_{alt} \\ z_{alt} \end{pmatrix} = \begin{pmatrix} s_x x_{alt} \\ s_y y_{alt} \\ s_z z_{alt} \end{pmatrix}$$

Lineare Transformationen

Rotation als Kombination von 3 elementaren Rotationen

$$\begin{pmatrix} x_{neu} \\ y_{neu} \\ z_{neu} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{pmatrix} \begin{pmatrix} x_{alt} \\ y_{alt} \\ z_{alt} \end{pmatrix} = \begin{pmatrix} x_{alt} \\ \cos\alpha y_{alt} - \sin\alpha z_{alt} \\ \sin\alpha y_{alt} + \cos\alpha z_{alt} \end{pmatrix}$$

$$\begin{pmatrix} x_{neu} \\ y_{neu} \\ z_{neu} \end{pmatrix} = \begin{pmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{pmatrix} \begin{pmatrix} x_{alt} \\ y_{alt} \\ z_{alt} \end{pmatrix} = \begin{pmatrix} \cos\beta x_{alt} + \sin\beta z_{alt} \\ y_{alt} \\ \cos\beta z_{alt} - \sin\beta x_{alt} \end{pmatrix}$$

$$\begin{pmatrix} x_{neu} \\ y_{neu} \\ z_{neu} \end{pmatrix} = \begin{pmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_{alt} \\ y_{alt} \\ z_{alt} \end{pmatrix} = \begin{pmatrix} \cos\gamma x_{alt} - \sin\gamma y_{alt} \\ \sin\gamma x_{alt} + \cos\gamma y_{alt} \\ z_{alt} \end{pmatrix}$$

Lineare Transformationen

- Spiegelung ist eine Skalierung um den Faktor -1 entlang einer Achse
- Überführt gleichzeitig ein rechtshändiges in ein linkshändiges Koordinatensystem
- Dabei werden auch alle Normalenvektoren umgekehrt
- Rotation oder Skalierung um andere Punkte als den Mittelpunkt wieder über vorhergehende und nachfolgende Translationen
- Ausdruck als Matrixmultiplikationen führt dazu, daß eine Kette von Manipulationen in einer einzigen Transformationsmatrix kombinierbar ist
- Analog zum 2D-Fall wechseln wir für die Translation in das homogene Koordinatensystem mit hinzugefügter 4. Dimension

Lineare Transformationen

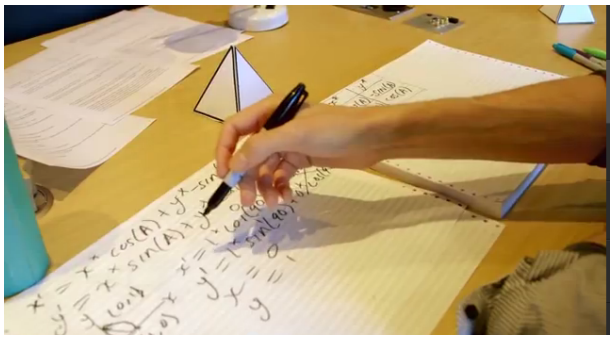
Homogenes Koordinatensystem

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} \Rightarrow \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}, \begin{pmatrix} m_{1,1} & m_{1,2} & m_{1,3} \\ m_{2,1} & m_{2,2} & m_{2,3} \\ m_{3,1} & m_{3,2} & m_{3,3} \end{pmatrix} \Rightarrow \begin{pmatrix} m_{1,1} & m_{1,2} & m_{1,3} & 0 \\ m_{2,1} & m_{2,2} & m_{2,3} & 0 \\ m_{3,1} & m_{3,2} & m_{3,3} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Translation als Multiplikation

$$\begin{pmatrix} x_{neu} \\ y_{neu} \\ z_{neu} \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_{alt} \\ y_{alt} \\ z_{alt} \\ 1 \end{pmatrix} = \begin{pmatrix} x_{alt} + t_x \\ y_{alt} + t_y \\ z_{alt} + t_z \\ 1 \end{pmatrix}$$

Video 10.1: The True Power of the Matrix



☞ The True Power of the Matrix (Transformations in Graphics) – Computerphile

14:46

Geschwindigkeitsgewinn

- Mittels homogener Koordinaten können viele Transformationen zu einer zusammengesetzt werden, die dann auf viele Punkte und Polygone angewandt wird
- Wie hoch ist dadurch der Geschwindigkeitsgewinn?
- Beispiel: Kombination von nur 5 Transformationen $A * (B * (C * (D * (E * x))))$ bzw. $(A * B * C * D * E) * x$
- Die einzelnen Multiplikationen mit 5 dreidimensionalen Matritzen benötigt $5 * 3 * 3 = 45$ Multiplikationen
- Die Multiplikation mit einer homogenen Matrix benötigt $4 * 4 = 16$ Multiplikationen und $4 * 3 = 12$ Additionen
- Einfache Optimierung (Nullen ignorieren): $3 * 4 = 12$ Multiplikationen und $3 * 3 = 9$ Additionen
- Berücksichtigt man, daß sehr komplexe Verabeitungsketten auftauchen können und diese auf viele Polygone angewandt werden, wird der Vorteil klar

Punkte, Geraden, Polygone, Polygonnetze

- Analog zu 2D werden aus Punkten und Geraden Polygone zusammengesetzt
- Für effiziente Berechnung von Beleuchtungsmodellen ist es wichtig, daß diese planar sind, d.h. daß alle Punkte in der gleichen Ebene liegen und die Flächennormale an jeder Stelle gleich ist
- Dreiecke erfüllen dies trivialerweise, für Polygone mit mehr Seiten Überprüfung aufwendig
- Jedes beliebige planare Polygon kann aber in Dreiecke aufgeteilt werden (Triangulation)
- Daher haben sich dreieckige Polynome als elementares Flächenstück durchgesetzt
- Im Beispiel des Koordinatensystems haben wir ein solches Dreieck mit der **Flächennormalen** bereits gesehen

Koordinatensysteme

- Dreieck mit Flächennormalen

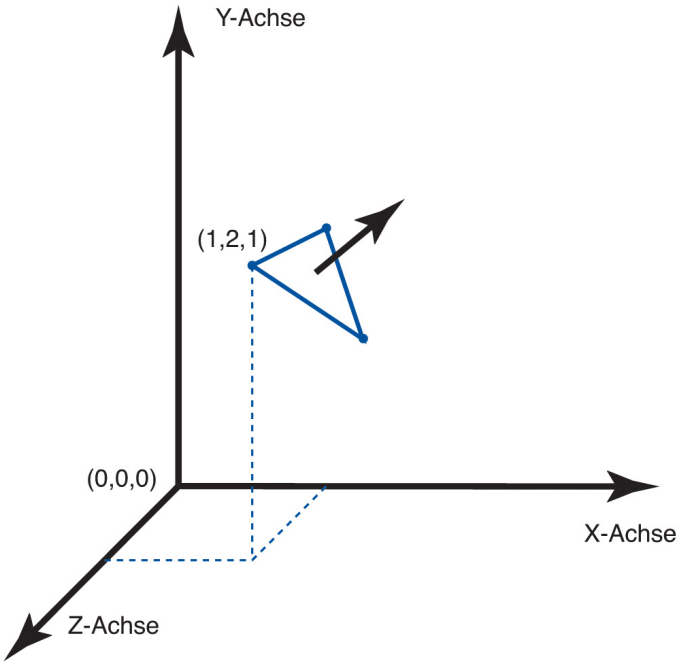
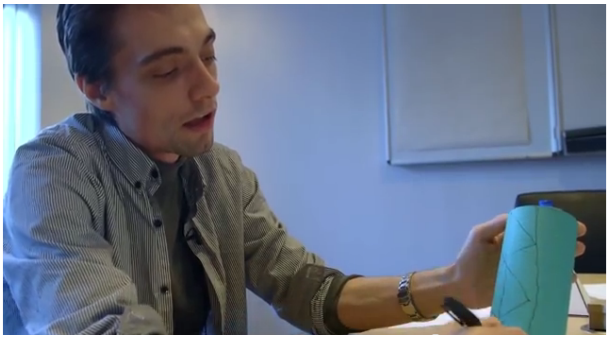


Abbildung 8.1: Rechtshändiges dreidimensionales Koordinatensystem, darin eingezeichnet ein Polygon aus drei Punkten mit seiner Flächennormalen

Punkte, Geraden, Polygone, Polygonnetze (contd.)

- Diese Polygone können zu einem **Polygonnetz** (Polygon Mesh) zusammengesetzt
- Da sich benachbarte Dreiecke immer zwei Eckpunkte teilen läßt sich eine effektive Repräsentation finden
- Durchnummerierte Liste von Eckpunkte wird gespeichert
- Dann Liste von Dreiecken speichern, die auf diese Indizes Bezug nimmt
- Diese Datenstruktur heißt **Indexed Face Set**
- Darüber hinaus gibt es noch weitere, stark optimierte Darstellungen, auf die wir hier nicht eingehen

Video 10.2: A Universe of Triangles



☞ A Universe of Triangles – Computerphile
11:24

Geometrische Primitive, CSG

- Neben Polygonmodellen bieten 3D-Graphikprograme (bzw. Bibliotheken) häufig die Möglichkeit, geometrische Primitive zu erzeugen
  - z.B. Quader, Zylinder, Kugel, Ring
- Aus diesen können durch boolesche Verknüpfung der Raumvolumen neue Formen erzeugt werden
- Diese Art der Konstruktion von Primitiven heißt **Constructive Solid Geometry**
- Die erzeugten Körper werden oft als **boolesche Körper** bezeichnet

Beispiel CSG

Abbildung 8.2: Boolesche Verknüpfung eines Quaders mit einem Zylinder

Extrusions- und Rotationskörper

- Weitere Möglichkeit: **Extrusions-** und **Rotationskörper**
- Extrusion: Eine gegebene Grundfläche wird entlang eines beliebigen Pfades (bspw. einer Geraden) immer wiederholt
  - Aus einem Kreis entsteht ein Zylinder
- Rotation: Bei einem Rotationskörper ist dieser Pfad ein geschlossener Kreis
  - Aus einem Kreis entsteht ein Ring
- Konstruktion von Gegenständen wie Vasen oder Flasche sehr effizient

Beispiel Extrusions- und Rotationskörper

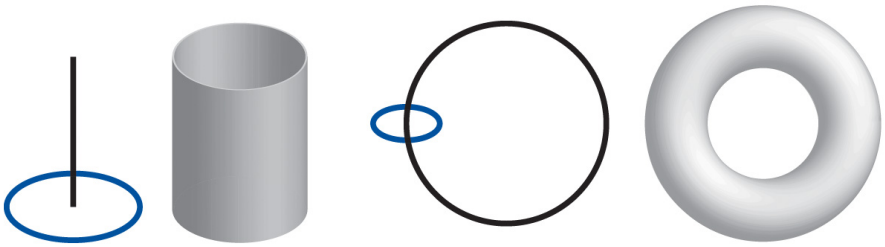
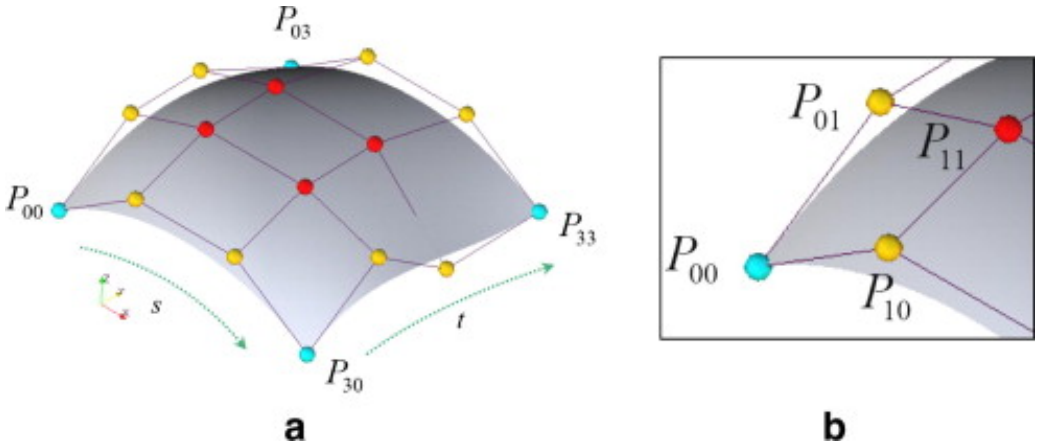


Abbildung 8.3: Extrusionskörper und Rotationskörper, jeweils mit zugehöriger Grundform (blau) und Pfad (schwarz), aus denen sie erstellt wurden

Freiformflächen

- Im 2D-Bereich haben wird für die Beschreibung beliebiger Umrisse die Interpolationskurven, insbesonders die Bézier-Kurven kennengelernt
- Aus solchen Kurven lassen sich auch Netze, sogenannte Freiformflächen, zusammensetzen
- Häufig verwendet: Zusammensetzung größerer Flächen aus Bézier-Patches
- Ein bikubischer Bézier-Patch besteht aus 4+4 kubischen Bézier-Kurven, die damit eine Fläche von 3x3 Feldern aufspannen
- Kurven teilen sich jeweils gemeinsame Kontrollpunkte, so daß die Form mit 16 Kontrollpunkten festgelegt wird

Beispiel Bézier Patch



Quelle: Czarny & Huysmans, 2008

Eigenschaften Bézier Patch

- Vorteile
  - Die beschriebene Oberfläche liegt innerhalb der konvexen Hülle aller Stützpunkte
  - Sie ist in jedem Punkt stetig
  - Die Eckpunkte des Netzes entsprechen den 4 Stützpunkten an den Ecken
  - Sehr kompakte Beschreibung durch 16 Stützpunkte
  - Transformationen im Raum entsprechen Transformationen der Stützpunkte
- Nachteile
  - Berechnung des Schnittpunktes mit einer Graden schwierig (Problem für Raytracing)
  - Flächennormale ändert sich über die Fläche
- Lösung: Annäherung von Freiformflächen durch planare Polygone (Tesselation)

Approximation Bézier Patch

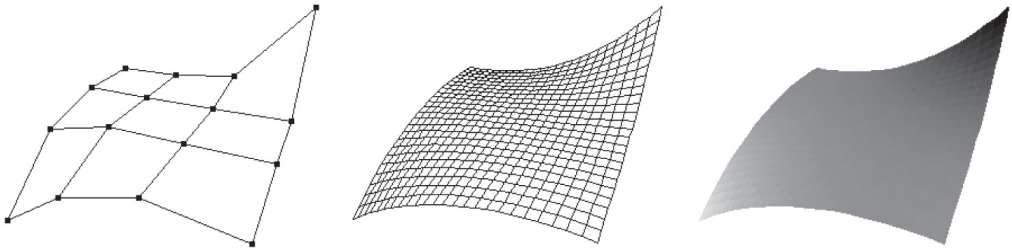


Abbildung 8.4: Bikubischer Bézier-Patch: die 16 Kontrollpunkten (links), Annäherung durch Polygone (Mitte) und schattierte Darstellung (rechts)

Andere Arten

- Bisher beschriebene Verfahren gingen davon aus, daß die dargestellten Objekte letztlich durch eine aus Polygonen zusammengesetzte Oberfläche repräsentiert werden
- Daneben gibt es weitere, hier nicht vertiefte Ansätze
- Alter Ansatz: Aufteilung des 3D-Raums in Voxel analog zu Pixeln, für die jeweils die optischen Eigenschaften beschrieben werden
- Verwendete Rendering-Funktionen damit grundlegend anders
- Beispiel für Voxel-Verfahren sind medizinische bildgebende Verfahren wie CT
- Neuere Entwicklung: punktbasierte 3D-Graphik
- Diese werden z.B. von Stereokameras oder Laserscanner erzeugt

2 Rendering Pipeline

Rendering Pipeline

- Die 3D Rendering Pipeline beschreibt eine Abfolge von Schritten, die aus
  - einem 3D Modell,
  - seinen Oberflächenbeschreibungen,
  - der Beleuchtung und
  - einer Kamera

  ein zweidimensionales Bild zur Darstellung z.B. auf einem Bildschirm machen
- Nicht alle Schritte in der Reihenfolge unbedingt festgelegt, Unterschiede in Implementierung
- Rendering-Pipeline ist heute typischerweise (zumindest teilweise) in Hardware implementiert
- Hier: generische Variante

Rendering Pipeline (contd.)

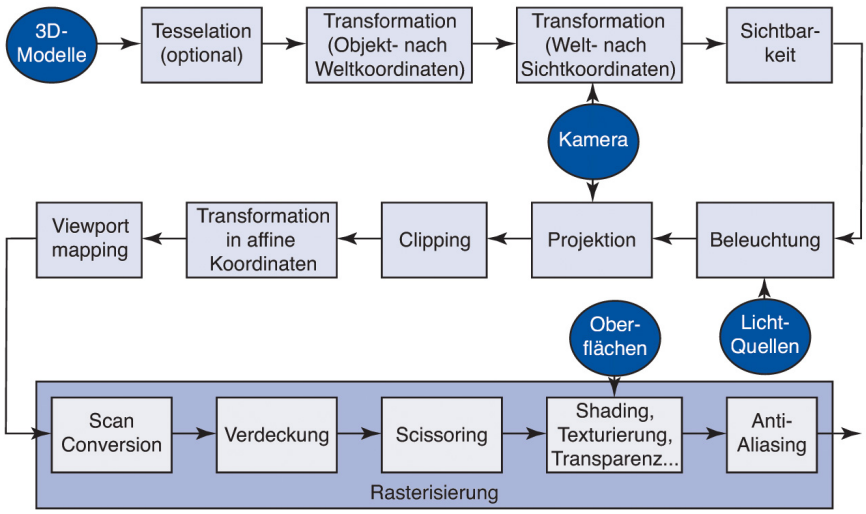


Abbildung 8.5: Die 3D Rendering Pipeline: der Weg vom 3D-Modell zum Bild unter Verwendung der Kamera, Lichtquellen und Oberflächenbeschreibungen

2.1 Tesselation

Tesselation

- Tesella (lat.) kleines, meist rechteckiges Mosaiksteinchen
- Aus diesen Steinchen mit ebener Oberfläche lassen sich auch gekrümmte Flächen annähern
- In der CG werden gekrümmte Flächen bei der Tesselation in so kleine Polygone aufgeteilt daß sie dem Betrachter noch hinreichend gekrümmt erscheinen
- Sind die Polygone Dreiecke spricht man auch von Triangulation

Utah Teapot

- Newells Teekanne (Utah Teapot) mit verschiedenen Tesselationen gerendert (sunflow.sf.net)
- Neun teilweise gespiegelte Beziér-Patches

Abbildung 8.6

Utah Teapot

- Das Original von Friesland Porzellan

Friesland-Porzellan

2.2 Szenegraph

Szenegraph

- Bisher beschriebene Objekte werden in Objektkoordinaten modelliert
- Würfel/Kugel Mittelpunkt immer im Ursprung des lokalen Koordinatensystems
- Nachdem alle Objekte beschrieben und ggf. tesseliert sind kommt die Transformation an die jeweiligen Positionen im Raum
- Dazu werden die Objekte in einem DAG (dem Szenegraph) organisiert
- Blätter geometrische Modelle samt Erscheinungen (Oberflächenbeschreibungen)
- Innere Knoten Transformationen oder Gruppierungen
- Im einfachsten Fall wieder ein Baum

Beispiel Szenegraph

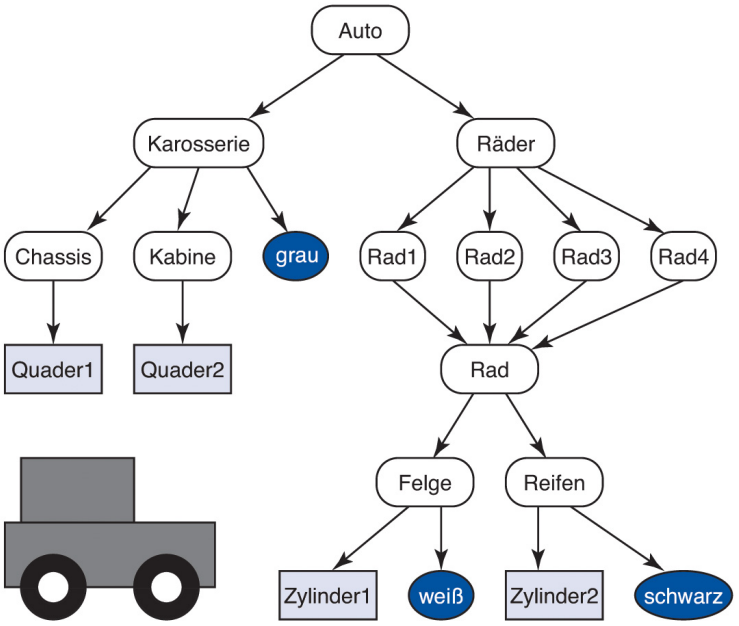
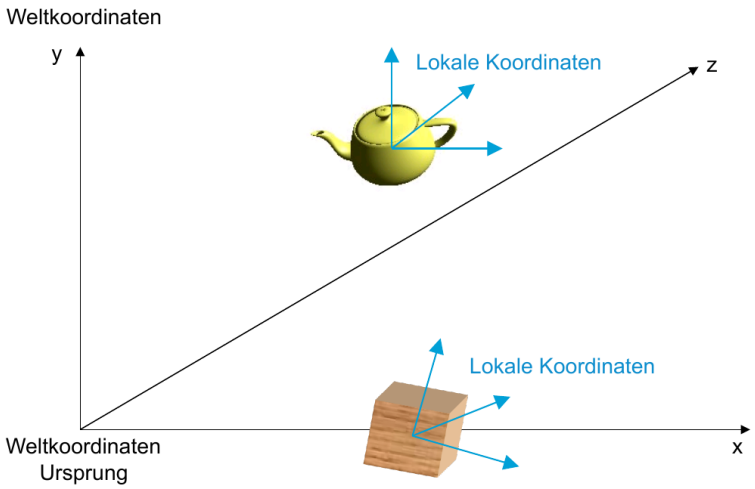


Abbildung 8.7: Szenegraph für ein Auto mit vier Rädern, in dem Geometrien, Oberflächen und ganze Baugruppen mehrfach verwendet werden

Szenegraph (contd.)

- Organisation im Graphen ermöglicht auch, komplexere Animationen logisch einfach zu beschreiben
- Transformation eines einzelnen Objektes bestimmt, indem alle Transformation von der Wurzel bis zum Blatt aufmultipliziert werden
- Angenommen, das modellierte Auto bewegt sich mit drehenden Rädern entlang eines geraden Weges
- Alle Bestandteile bewegen sich dann linear, die Räder drehen sich zusätzlich (Rotation)
- Es reicht dann, für "Rad" eine wiederholte Rotation um 360 Grad in einer festen Zeiteinheit zu beschreiben und diese dort in den Szenegraphen als Transformation einzubauen
- Durch die Verwendung der gleichen Baugruppe drehen sich dann alle Räder

Objekt- und Weltkoordinaten



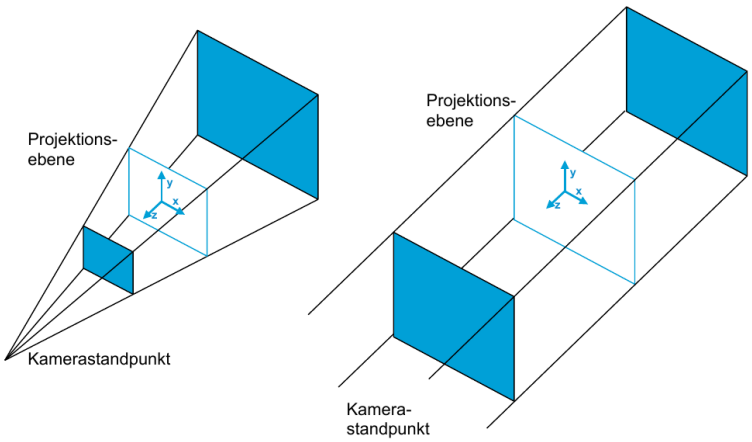
Quelle: Stefan Schlechtweg

2.3 Kameramodell

Kameramodell

- Bisher komplett in Weltkoordinaten (globales Koordinatensystem der 3D-Szene)
- Zur Abbildung auf 2D-Bildebene benötigen wir eine Projektion
- Dazu dient die Kamera
- Weitere Koordinatensysteme
  - Sichtkoordinaten (auch Kamerakoordinaten): Interne Koordinatensysteme der Kamera
  - Bildkoordinaten (auch Projektionskoordinaten): Koordinatensystem zur Darstellung der perspektivischen Projektion
    - Zumeist (aber nicht immer) perspektivische Projektion
- Weitere Annahmen im folgenden:
  - Zentrum der Projektion im Ursprung, Bildebene bei $Z = 1$
  - Kamera blickt entlang der negativen Z-Achse

Kamerakoordinaten



Quelle: Stefan Schlechtweg

Kameramodell (contd.)

Weltkoordinaten in Sichtkoordinaten (gut für Berechnung von Verdeckung und Sichtbarkeit, da Position der Punkte entlag der Z-Achse erhalten bleibt)

$$\begin{pmatrix} x_{sicht} \\ y_{sicht} \\ z_{sicht} \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \\ -z \end{pmatrix}$$

Sichtkoordinaten in Bildkoordinaten (perspektivische Division, in der weiter entfernte Objekte kleiner werden)

$$\begin{pmatrix} x_{bild} \\ y_{bild} \\ z_{bild} \\ w_{bild} \end{pmatrix} = \frac{1}{w_{sicht}} \begin{pmatrix} x_{sicht} \\ y_{sicht} \\ z_{sicht} \\ w_{sicht} \end{pmatrix} = \begin{pmatrix} x_{sicht}/w_{sicht} \\ y_{sicht}/w_{sicht} \\ z_{sicht}/w_{sicht} \\ w_{sicht}/w_{sicht} \end{pmatrix} = \begin{pmatrix} x/-z \\ y/-z \\ -1 \\ 1 \end{pmatrix}$$

Beispiel Kameramodell

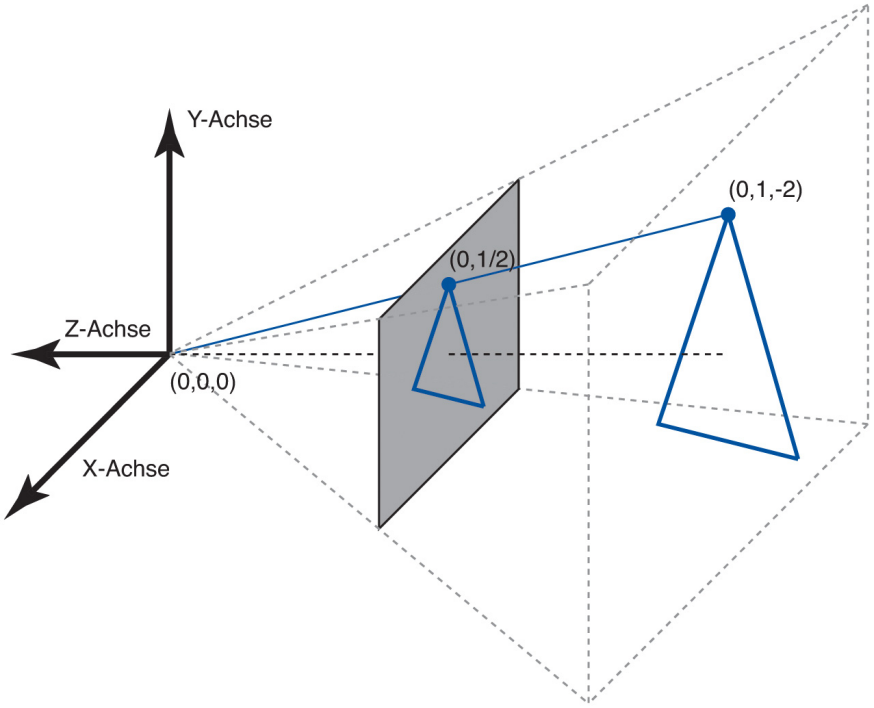


Abbildung 8.8: Perspektivische Projektion in der Kamera. Im Beispiel sind die Weltkoordinaten (0,1,-2) und damit ergeben sich die Bildkoordinaten (0,½).

2.4 Sichtbarkeit

Sichtbarkeitsbestimmung (Culling)

- Wenn alle Koordinaten in Sichtkoordinaten transformiert sind lassen sich (erhebliche) Optimierungen durchführen (Reduktion der Menge der später zu beachtenden Polygone)
- Grundsatz: Polygone von der Betrachtung ausschließen, die nicht im Bild sichtbar werden
- Zunächst alle, die außerhalb des Pyramidenstumpfes sind, der von durch die 4 Bildränder und einer festgelegten vorderen und hinteren Schnittebene (far and near clipping plane) gebildet wird
- Dieser Stumpf wird als Sichtvolumen (view volume, view frustum) bezeichnet

Beispiel Sichtvolumen

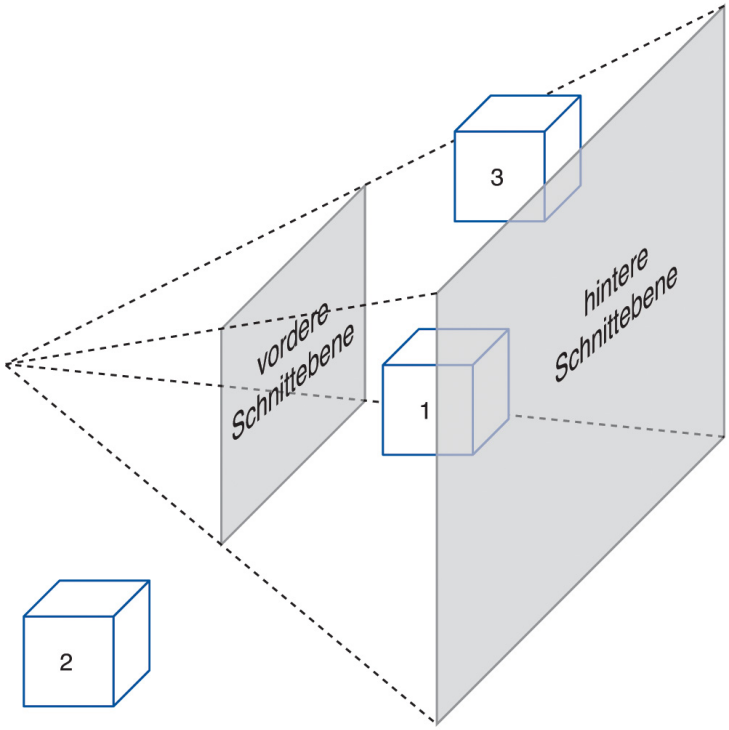


Abbildung 8.9: Sichtvolumen der Kamera mit vorderer und hinterer Schnittebene. Objekt 1 liegt vollständig innerhalb des Sichtvolumens, Objekt 2 vollständig außerhalb und Objekt 3 schneidet das Sichtvolumen.

Sichtbarkeitsbestimmung

- View Frustum Culling
  - Jedes Objekt durch umschreibenden Quader (Bounding Box in 3D) annähern und mit 4 Vergleichen ermitteln, ob der Quader komplett innerhalb/außerhalb des Sichtvolumens liegt
  - Vollständig enthaltene werden später berücksichtigt
  - Vollständig außerhalb liegende können ignoriert werden
  - Für schneidende Objekte ggf. für Unterobjekte im Szenegraphen Test wiederholen, im Zweifelsfall berücksichtigen
- Back Face Culling
  - (Einseitige) Polygone, deren Normalenvektoren von der Kamera weg zeigen, aus der Pipeline entfernen
  - Grundidee: Bei geschlossenem Polygonnetz (ohne Löcher) können solche Polygone nur auf der Rückseite von Objekten vorkommen, sie sind somit durch anderen Polygone verdeckt

Video 10.3: Triangles to Pixels

☞ Triangles to Pixels – Computerphile

8:17

2.5 Licht

Lichtquellen: Ambient

- Wir arbeiten mit verschiedenen Vereinfachungen im Vergleich mit natürlichem Licht
- Am einfachsten zu berechnende Vereinfachung ist das ambiente Licht
- In der Natur werden stets Lichtanteile von Objekten, dem Boden oder Wolken diffus reflektiert, daher ist überall etwas Licht
- Vereinfachende Modellierung, indem im Modell überall Licht von gegebener Helligkeit ohne Richtung vorhanden, die den Polygonen eine Grundhelligkeit verleiht

Lichtquellen: Gerichtet

- Auch das gerichtete Licht wird vereinfacht modelliert
  - einerseits nimmt die Intensität quadratisch mit dem Abstand ab
  - andererseits ist die natürliche Hauptquelle Sonne so weit entfernt, daß die quadratische Abnahme praktisch nicht mehr wahrnehmbar ist
- Sonnenlicht wird daher allein durch Richtung und Helligkeit angenähert und erzeugt überall in der Szene gerichtetes Licht gleicher Intensität, das keine oder harte Schatten wirft
- Mit diesem "Sonnenlicht" und dem ambienten Licht können bei akzeptablem Rechenaufwand bereits akzeptable Beleuchtungssituationen geschaffen werden
- "Sonnenlicht" etwas wärmer, ambientes Licht etwas blauer (sonnige Tage) oder weißer (bedeckte Tage)

Lichtquellen: Punkt, Spot

- Lokale Lichtquellen wie Kerzen, Glühbirnen als Punktlichter
  - Beschrieben durch Position und Lichtfarbe/Intensität
  - Gleichmäßig in alle Richtungen strahlend, Intensität mit dem Quadrat der Entfernung abbnehmend
  - Statt rein quadratischer Abnahme oft linearer Anteil
  - Zur Darstellung benutzte Monitore haben geringen Kontrastumfang, das "falsche", etwas kontrastarme Licht wird als natürlich angesehen
- Beschränkung des Abstrahlwinkels mit einem Kegel führt zu Spotlicht
- Die Nähe zum Rand des Lichtkegels beeinflußt die Intensität (Falloff)
  - Falloff kann hart oder weich sein, und näher oder entfernter von der Mitte des Kegels beginnen
- Sowohl Punkt wie Spotlight werfen harte Schatten

Beispiel Lichtquellen
Gerichtet – Punkt – Spot

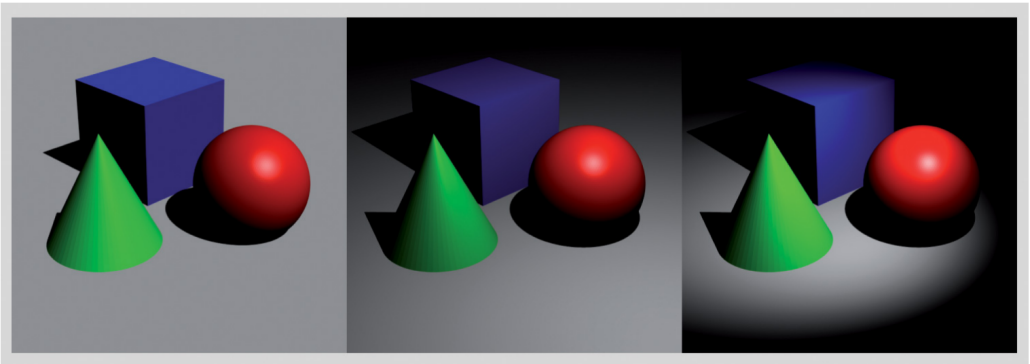


Abbildung 8.10

Lichtquellen Fläche

- Am aufwendigsten: Weich umrandete Schatten mit einer Flächenlichtquelle
- Wie 3D-Objekte durch Geometrie modelliert, die von jedem Punkt Licht aussenden
- Annäherung über verteilte Punktlichtquellen
- Aufwendig, aber beste Ergebnisse mit weich umrandeten Schatten

Lichtquellen (contd)

Abbildung 8.11

Oberflächen

- Letzter Einflußfaktor ist die Oberflächenbeschreibung der Objekte
- Erzeugung guter Texturen, prozeduraler Shader und deren Parametrisierung sehr aufwendig
- Grundsätzlich: Beschreibung des ambienten, diffusen und spekularen (spiegelnden) Anteils des reflektierten Lichts

Lichtreflexion an Objekten

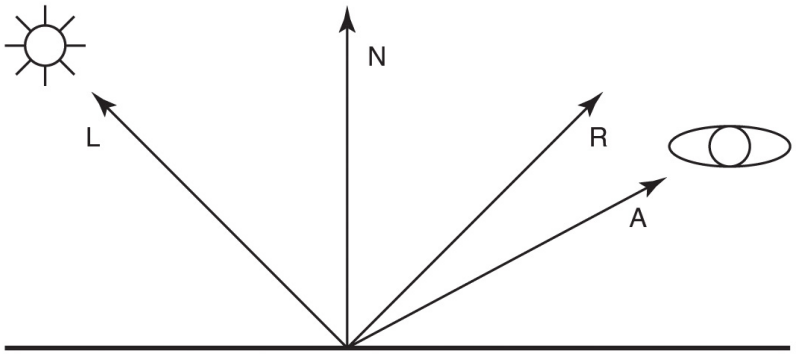


Abbildung 8.12: Vektoren zur Berechnung des Beleuchtungsmodells nach Phong: L zeigt zur Lichtquelle, N ist die Flächennormale, R der reflektierte Lichtstrahl und A zeigt zum Auge.

Ambiente Reflexion

- Für ambiente Reflexion wird ein Koeffizient angegeben, der beschreibt, welcher Anteil des ambienten Lichts mit welcher Intensität gleichmäßig reflektiert wird und in den Gesamteindruck eingeht

$$I_a = k_a I_{al}$$

- mit I_a als reflektiertem Licht, k_a Reflexionskoeffizient für ambientes Licht, I_{al} Intensität des ambienten Lichts
- Kann eingesetzt werden, um völlig schwarze Schatten zu vermeiden

Diffuse Reflexion

- Vollständig matter Körper weist bei einer bestimmten Beleuchtung die gleiche Farbe und Helligkeit auf, egal, von wo man ihn betrachtet
- Diese Art heißt diffuse oder Lambertsche Reflexion

$$I_d = k_d I_L < L, N >$$

- mit I_d als reflektiertem Licht, k_d Reflexionskoeffizient für diffuse Beleuchtung, I_L Intensität der betrachteten Lichtquelle und $< L, N >$ Skalarprodukt zwischen Flächennormale und Winkel zur Lichtquelle
- Die diffuse Reflexion nimmt mit dem Cosinus des Winkels des Auftreffens des Licht auf das Objekt ab

Glanzreflexion

- Die spekulare oder Glanzreflexion folgt einem ähnlichen Gesetz
- Ein perfekter Spiegel reflektiert mit Einfallswinkel = Ausfallswinkel
- Die meisten Flächen sind keine perfekten Spiegel
- Reflexion am stärksten bei Einfallswinkel = Ausfallswinkel, aber die Intensität läuft danach nicht gleich auf 0 zurück, sondern sinkt langsam

$$I_s = k_s I_L < R, N >^n$$

- mit I_s als reflektiertem Licht, k_s Reflexionskoeffizient für spekulare Beleuchtung, I_L Intensität der betrachteten Lichtquelle und $< R, N >$ Skalarprodukt zwischen Flächennormale und Reflexionswinkel
- Hierbei ergibt ein kleiner Wert für n einen weichen Lichtabfall, ein großer Wert scharfe Glanzpunkte

Beleuchtungsmodel von Phong

- Das Beleuchtungsmodel von Phong setzt sich damit aus diesen drei Komponenten zusammen

$$I = I_a + I_d + I_s$$

$$I = k_a I_{al} + k_d I_L < L, N > + k_s I_L < R, N >^n$$

- Vereinfachende Darstellung mit Grauwerten statt Farbwerten
- Einfache Erweiterung: Arbeit in drei Kanälen (R, G, B)

Phong: Winkel

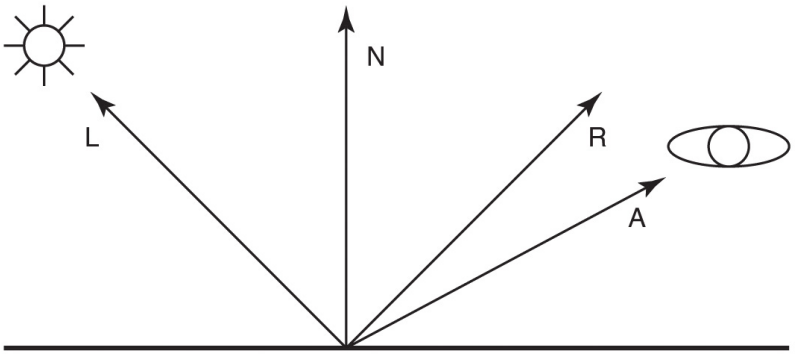


Abbildung 8.12: Vektoren zur Berechnung des Beleuchtungsmodells nach Phong: L zeigt zur Lichtquelle, N ist die Flächennormale, R der reflektierte Lichtstrahl und A zeigt zum Auge.

2.6 Texturen und Verdeckung

Texturen

- Weiterer mit den Beleuchtungsmodellen gemeinsam nutzbarer Teil
- Texturen sind Bitmaps, die bestimmte Eigenschaften der Objektoberfläche verändern
- Im einfachsten Fall beschreibt die Map die diffuse Reflexionsfarbe an der jeweiligen Stelle (diffuse map)
- Texturen können auch andere Aspekte beschreiben;
  - die Transparenz eine Objektes (transparency map)
  - Abweichung der Geometrie der Objekte (height map)
  - Veränderung der Flächennormalen (normal map)

Verdeckungsberechnung

- Berechnung der gegenseitigen Verdeckung
- Einfach wieder Painter's algorithm?
- Schlägt fehl bei sich zirkular überlappenden Elementen

Abbildung 8.13: Drei Polygone, die sich zirkulär gegenseitig überlappen und daher mit dem Painter's Algorithm nicht korrekt dargestellt werden können

Z-Buffer

- Für jeden einzelnen Bildwert nicht nur Farbwert RGB speichern, sondern auch Tiefeninformation Z
- Z-Buffer initialisiert mit größtmöglicher Entfernung zur Kamera
- Beim Zeichnen eines Polygons wird die aus den Objekteigenschaften und dem Licht berechnete Farbe eingetragen, plus Z-Wert
- Steht hier schon eine kürzere Entfernung wird das Pixel nicht gezeichnet
- Etwas komplexer bei Transparenz

- Kritisch ist die numerische Genauigkeit
- Wegen effizienter Berechnung Rechnen z.B. mit Festkommadarstellung, 16 Bit
- Sind hintere und vordere Schnittebene zu weit voneinander entfernt und liegen Polygone auf gleicher Z-Ebene kann es wg. Rundungsfehlern zu Artefakten kommen

Video 10.4: The Visibility Problem

☞ The Visibility Problem – Computerphile

7:58

3 Bilderzeugung

Lokale und Globale Beleuchtungsverfahren

- Letzter Schritt ist die Rasterisierung
- Beleuchtung unterschieden in Global und Lokal (global bzw. local illumnination)
- Globale Verfahren betrachten die Auswirkungen des Lichts in der ganzen Szene
- Einfacher zu berechnende lokale Verfahren betrachten jedes Polygon und innerhalb des Polygons jedes Pixel
- Abhängig von Flächennormalen, Lichteinfallswinkel, Kameraposition, Umwelteigenschaften

3.1 Lokal

Flat Shading

- Eine einzige Flächennormale für jedes Polygon
  - Beispielsweise das Kreuzprodukt zweier Kanten
- Mit dieser Flächennormalen sowie den Vektoren zur Kamera und zum Licht sowie der Oberflächenfarbe wird eine einfache Beleuchtungsfunktion berechnet
- Dies kann z.B. das eingeführte Model von Phong sein
- Die ermittelte Farbe gilt für alle Pixel des Polygons
- Pro Polygon muß die Beleuchtungsfunktion nur einmal ausgewertet werden
- Der entstehende Bildeindruck ist entsprechend kantig

Gouraud Shading

- an allen Eckpunkten des Polygons wird eine Beleuchtungsfunktion ausgewertet, und zwar mit den jeweiligen Flächennormalen
- So ermittelte Farben werden entlang der Kanten linear interpoliert
- Über die Polygonfläche ergibt sich ein Farbverlauf
- Da benachbarte Polygone an den Eckpunkten die gleichen Normalenvektoren haben sind die Übergänge geglättet
- Für jedes Polygon muß die Berechnung aber mehrfach durchgeführt werden
- Dazu kommt die Interpolation der Farbe für jedes Pixel

Phong Shading

- Hier wird der Normalenvektor über die gesamte Polygonfläche hinweg interpoliert
- Erheblich höherer Rechenaufwand durch die Interpolation von Vektoren
- Wesentlicher Vorteil bei der Darstellung von glänzenden Oberflächen
- Je genauer die Approximation an den Ausgaben eines Glanzlichts ist, desto eher wird ein relativ kleines Glanzlicht auch aufgenommen

Glanzpunkte

- Glanzpunkte entstehen dort, wo der Einfallswinkel des Lichts gleich dem Ausfallswinkel ist
- Die Kamera sieht genau in die Spiegelung einer Lichtquelle
- Nehmen wir an, dies sei irgendwo in einem Polygon der Fall
- Ferner nehmen wir Phong-Shading an
  - Beim Flat-Shading ist es purer Zufall, wenn ein Glanzpunkt getroffen wird, dann aber strahlt das ganze Polygon
  - Bei Gouraud-Shading erfüllt keine Flächennormale die Bedingung für den Glanzpunkt wenn er innerhalb liegt, er bleibt damit unsichtbar
  - Phong-Shading gibt die größte Chance, Glanzpunkte wiederzugeben, wenn einer der interpolierten Vektoren dem Normalenvektor des tatsächlichen Glanzpunktes nahekommt

Übersicht lokale Shader

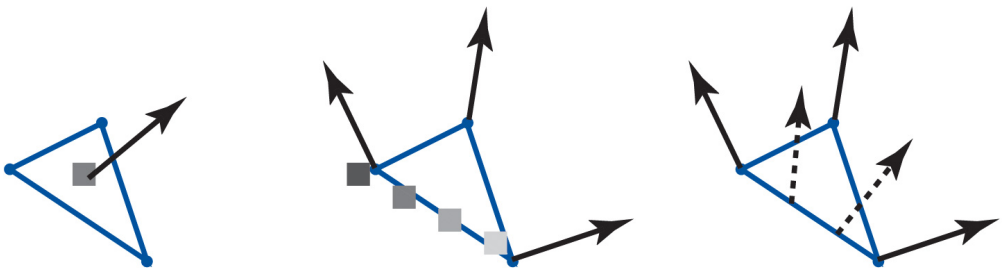


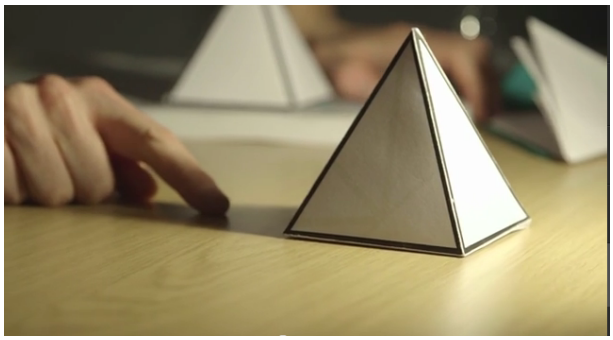
Abbildung 8.14: Funktionsweise des Flat Shading, Gouraud Shading und Phong Shading

Abbildung 8.15

Schatten bei lokalen Shadern

- Für die bisher beschriebene Modellierung der Lichts brauchen wir für die diffuse Reflexion und die Glanzreflexion den Winkel zur Lichtquelle
- Problem: Auch die Lichtquelle kann analog zu Objekten von anderen Objekten verdeckt werden
- In diesem Fall sollte das Licht nicht modelliert werden
- Lösung: Einsatz des Z-Buffers, wobei dieses mal von der Lichtquelle aus gerechnet wird
  - Alle Objekte werden von der Lichtquelle aus gesehen in einen Z-Buffer geschrieben
  - Nur bei dem Objekt, welches der Lichtquelle am nächsten ist, wird die lokale Beleuchtung (z.B. mit Phong) berechnet

Video 10.5: Lights and Shadows



☞ Lights and Shadows in Graphics – Computerphile
8:49

3.2 Global

Globale Verfahren

- Bis vor wenigen Jahren waren Globale Verfahren nur offline verwendbar (Hoher Rechenaufwand)
- Realzeitrendering rein lokal
- Durch Optimierung der Algorithmen und durch verbesserte Hardware jetzt auch online machbar
- Wichtig für Computerspiele und Visualisierungen
- Stärkere Immersion durch realistischere Darstellung möglich
- Wie betrachten nur zwei klassische Verfahren
  - Raytracing
  - Radiosity

Raytracing

- Verfolgung je eines Blickstrahls vom menschlichen Auge durch jedes Bildschirmpixel in die 3D-Szene
- Trifft auf vordersten davon geschnittene Objektoberfläche
- Dort entweder reflektiert, gebrochen oder absorbiert
- Bis zu einer gewissen Reflexionstiefe werden gebrochene/reflektierte Streifen ebenfalls verfolgt
- Beim Rücklauf durch die Rekursion ergibt sich die Pixelfarbe als Summe aller gebrochenen und reflektierten Strahlen sowie ggf. der Oberflächenfarbe am Schnittpunkt
- Die Farbe wird dabei durch Beleuchtungsfunktionen analog zu den lokalen Verfahren berechnet
- Nebenbei leistet Raytracing einen Beitrag zur Elimination von Objekten, die im Bild nicht auftauchen können
- Statt Z-Buffer tauchen nur Objekte auf, die von einem Lichtstrahl getroffen werden, der in der Kamera endet

Beispiel Raytracing

Abbildung 8.16

Radiosity

- In der Natur wird jede Oberfläche nicht nur direkt von einer Lichtquelle beleuchtet, sondern auch durch Lichtanteile, die von anderen Objekten diffus reflektiert wurden und jetzt indirekt auf die Oberfläche einwirken
- Oberflächen werden unterteilt
- Dann wird die direkt einfallende Lichtenergie berechnet
- Daraus und dem diffusen Reflexionskoordinator ergibt sich die abgestrahlte Menge
- Dann wird iterativ berechnet, welcher Anteil der Lichtenergie bei wieder anderen Oberflächen ankommt
- Nahe beieinander liegende Flächen beeinflussen sich stärker als weit entfernt liegende
- Nach vielen Durchläufen stabilisiert sich das Verfahren
- Berechnung aufwendig, quadratisch zur Anzahl der Flächen und iterativ
- Radiosity ist unabhängig von der Kameraposition!

Beispiel Radiosity

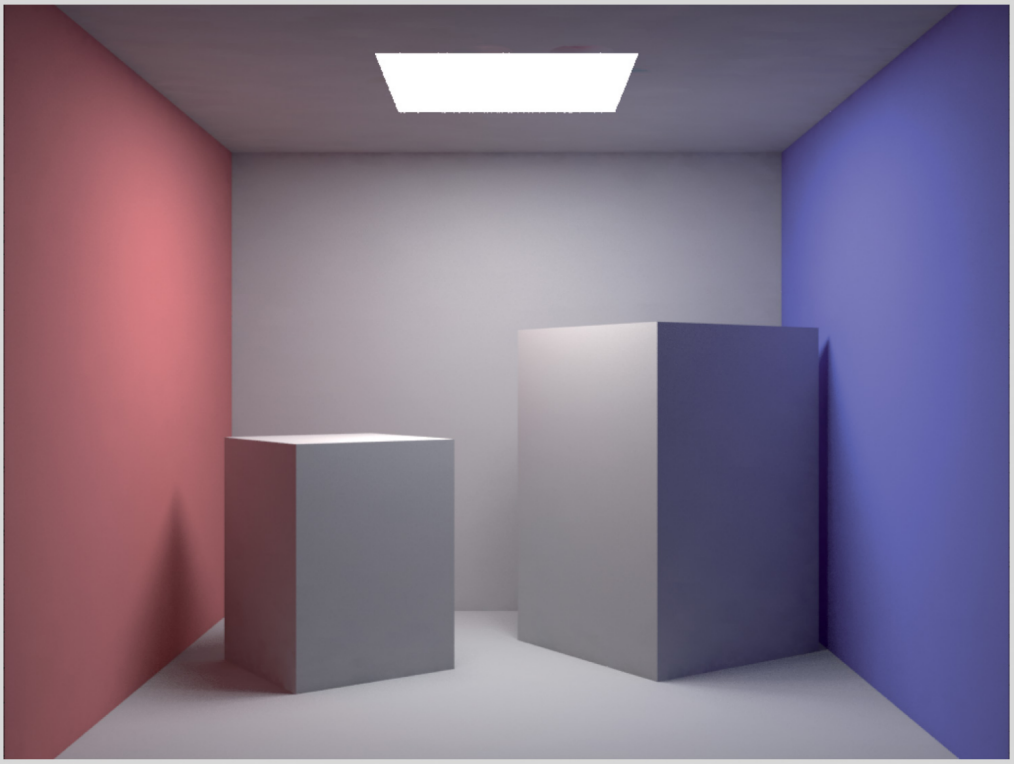


Abbildung 8.17

Kombination

- Radiosity wird häufig in Kombination mit anderen Verfahren eingesetzt, um den diffusen Charakter des Lichts aufzugreifen
- Verdeckung muß durch andere Verfahren sichergestellt sein (Raytracing, Z-Buffer)
- Da Raytracing Spiegelungen und Brechungen beherrscht, was Radiosity nicht kann, werden beide inzwischen häufiger zusammen benutzt.

Radiosity + Raytracing

Abbildung 8.19

4 Animation

Animation

- Um aus 3D-Szene, Lichtern, Oberflächen und Kamera eine bewegte Darstellung (3D Animation) zu erzeugen muß jegliche Veränderung in der Szene (Bewegung von Objekten, Lichtern, Kameras; Veränderung der Transparenz) als Veränderung in der Zeit beschrieben sein
- Bisher Keyframing eingeführt
- Der zu steuernde Parameter wird zu mindestens zwei Zeitpunkten fixiert, die Bewegung dazwischen wird durch lineare Interpolation oder Kurven bestimmt
- Neben Keyframing andere Methoden, wie physikalische Simulation (mit Aufgabe von detaillierter Kontrolle)
- Partikelsysteme, z.B. für Wolken, Vogelschwärme
- Mit der inversen Kinematik wird die Bewegung komplexer Apparate aus der Bewegung weniger Kontrollpunkte abgeleitet (Skelettanimation)
- Bei Animationsfilmen häufig auch: Motion-Capture

Andere Formen der Animation und Interaktion

- **Partikelsysteme** ermöglichen mittels einer physischen Simulation die Animation vieler einzelner Objekte
  - Einzelne Partikel bekommen bestimmte Eigenschaften, dazu gerne eine Zufallskomponente
- **Particle-in-cell-Methoden** (PIC)
  - Grundidee: Neben einer Modellierung der einzelnen Partikel wird auch die Interaktion benachbarter Partikel modelliert
  - Letztlich basierend auf den Ideen der Finite Element Method
- Programmcode kann das Verhalten von Objekten beeinflussen (**Scripting**)
- Grafiken können **interaktiv** auf Anreize reagieren

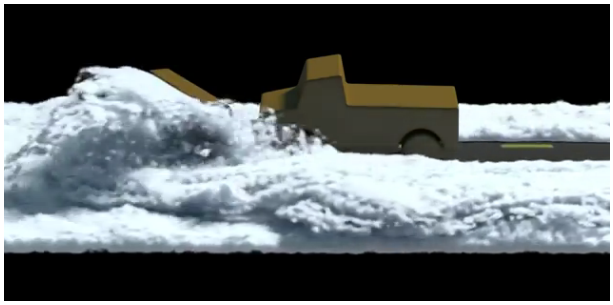
Video 10.6: Partikelsysteme

☞ Particle Dreams (1988)
1:30

Video 10.7: Artificial Life

☞ Panspermia (1990)
2:08

Video 10.8: Material Point Method



☞ Disney's Frozen - An MPM For Snow Simulation
3:55

Hintergrund

- Material Point Methods (MPM) sind eine Erweiterung von Particle-in-cell-Methoden (PIC)
- Grundidee: Überlagerungen einer ortsfixen Modellierung grundlegender Eigenschaften des zu modellierenden Objektes und einer mit den Partikeln fließenden Modellierung
- Letztlich basierend auf den Ideen der Finite Element Method
- Hier wird ein zu modellierendes Gesamtsystem in kleinere Systeme zerlegt, die für sich modelliert werden
- Statt eines großen Differentialgleichungssystem Approximation durch kleine, lokale Gleichungssysteme

5 Codierung

Dateiformate

- Viele, oft herstellerspezifische Dateiformate für 3D-Daten
- Offene Standards z.B.:
  - Virtual Reality Modeling Language (VRML)
  - X3D, offizielles Nachfolgeformat
    - XML-Syntax
    - VRML-Syntax
- Nicht betrachtet: komplexe Beschreibungen in Form einer Programmierschnittstelle
  - DirectX
  - OpenGL

        * OpenGL embedded
        * WebGL
    - Enthalten alle grundlegenden Funktionen der Rendering-Pipeline

5.1 X3D

X3D

- Verschiedene Formate definiert
  - XML, VRML, Binär
- Anwendungsmöglichkeiten
  - Bereitstellung von 3D-Szenen im Web
  - Austauschformat
- Rendering abhängig von der verwendeten Software
- Enthält
  - Geometrische Beschreibung einer 3D-Szene
  - Oberflächenbeschreibungen
  - Kamerapositionen (Viewpoints)
  - Lichtquellen

Grundlagen

- Definierte Profile, die jeweils die unterstützten Features beschreiben
- Verbreitung eingeschränkt, aber Konzepte gut verständlich
- Rendering im Browser mit WebGL und JavaScript (ohne Plugin) prinzipiell möglich

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE X3D>
<X3D profile='Immersive'>
 <head/>
 <Scene>
   <!-- Beschreibung der Szene -->
 </Scene>
</X3D>
```

Geometrische Primitive

- Quader, Zylinder, Kugel, Polygonnetze
- Jedes Element besitzt Oberflächenbeschreibung
- Gruppiert in Shape-Elementen

```
<shape>
  <appearance>
    <material diffuseColor='0,0,1'></material>
  </appearance>
  <box></box>
</shape>
```

Viewpoints

- Standardeinstellung: von (0,0,1) entlang negativer z-Achse

```
<Viewpoint position="-2,2.5,4"
  orientation="3,2,0,-0.7">
</Viewpoint>
```

- Position klar
- Orientation: Rotationsachse und Winkel
  - Achse (3,2,0)
  - Winkel -0.7 Radian

Beispiel

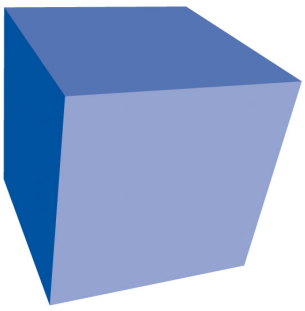


Abbildung 8.20: Blauer Würfel mit veränderter Kameraposition

☞ Demonstrationen: X3D

Gruppen, Szenegraph

- Objekte können in Gruppen zusammengefaßt und gemeinsam transformiert werden
- Bezeichner mittels DEF festgelegt
  - Wiederverwendung
  - Animation

```
<Transform DEF="t1" translation="0.75 0 0">
  <!-- Hier alle Objekte der Gruppe -->
</Transform>
```

Lichter

- Eigentlich dürften wir ohne Lichtquelle nichts sehen
- X3D erzeugt, wenn nichts anderes spezifiziert ist, ein Headlight
  - Gerichtetes Licht parallel zur Kameraachse
  - Schlagschattenfrei
- Lichtquellen enthalten Informationen wie
  - Position, Farbe, Richtung, Helligkeit, Winkel
- Lichtquellen können in Gruppen auftauchen und mit diesen transformiert/animiert werden

```
<NavigationInfo headlight='false'>
<directionalLight
  direction="3,-2,-1" intensity='1.0'>
</directionalLight>
<Spotlight location="3,-2,-1"
  direction="0 0 1" beamWidth='0.8'>
</Spotlight>
```

Animation: Ablauf

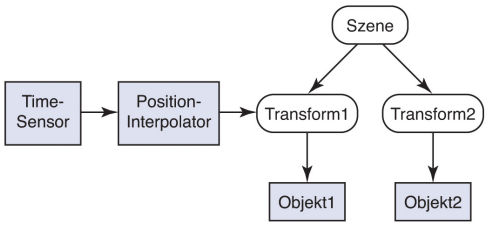


Abbildung 8.21: Animation in X3D mittels TimeSensor, PositionInterpolator und Transform Knoten

Animation: Grundlagen

- Treibendes Element ist eine Uhr, der sogenannte TimeSensor
- Beschreibt Keyframes auf einem Intervall von 0.0 bis 1.0
- Liefert ein Signal, das im Verlauf des angegebenen Intervalls von 0.0 auf 1.0 steigt und dann wieder bei 0.0 anfängt
- Dieses Signal steuert den Interpolator
- Es gibt verschiedene, z.B. für Positionen
- Angegeben sind jeweils timestamp und Position

```
<TimeSensor DEF="ts1" cycleinterval="2" loop="true">
</TimeSensor>
<PositionInterpolator DEF="pi1"
  key="0,0.5,1" keyValue="0.5 0 0, -1 0 0, 0.5 0 0">
</PositionInterpolator>
```

Animation: Verarbeitungskette

- Die angegebenen Positionswerte können eine Transformation verändern
- Eine Animation ist eine Kette aus TimeSensor, Interpolator und Transformation
- Alle Objekte werden über ihren Identifikator (DEF) angesprochen
- Interaktivität durch Sensoren, die auf Mausinteraktion reagieren
- Möglichkeit, Skripte zu definieren

```
<ROUTE fromField="fraction_changed" fromNode="ts1"
  toField="set_fraction" toNode="pi1">
</ROUTE>
<ROUTE fromField="value_changed" fromNode="pi1"
  toField="translation" toNode="t1">
</ROUTE>
```

5.2 Demo

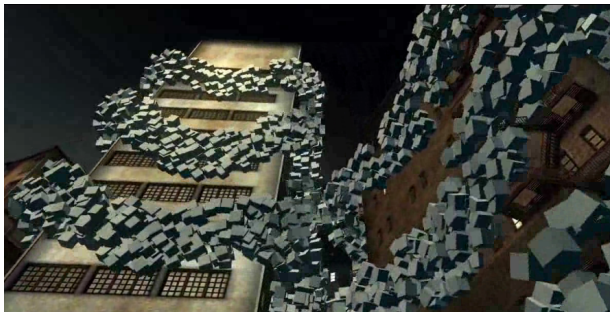
Demoszene

- Versuch, eindrucksvolle grafische und musikalische Effekte mit minimalen Mitteln zu erzeugen
- Ursprünglich erzwungen durch die Beschränkungen der Hardware (C64, Amiga)
- Beispiele für Methoden
  - Executable: komprimiert mit eingebauter Dekompression (wie UPX), Dekompresser selber klein
  - Ton: Synthesizer + MIDI-Dateien
  - Video: Ausnutzen von Systemfähigkeiten plus algorithmischer Codierung der Inhalte
    - OpenGL, Direct X API inklusive z.B. Pixelshadern
    - Algorithmische oder vektorisierte “Wireframes”

    - Prozedurale Texturen (z.B. (fraktales) Rauschen auf Flächen, zelluläre Automaten, genetische Algorithmen)

- Nur Anwendbar auf algorithmisch erstelle Inhalte
  - Aber Grundprinzipien sind Gegenstand von Forschungen für die Kompression natürlicher Videos (z.B. genetische Algorithmen für die Rekonstruktion von Frames)

Video 10.9: fr-041: debris



☞ Vimeo
☞ farbrausch
177kB, 2007

6 Erstellung

Synthese und Abtastung

- Die bisher besprochene Rendering-Pipeline wird auf bereits definierte Objekte angewandt
- Die Frage ist dann, wie diese Objekte und die Szene, in denen sie auftauchen, in eine digitale Repräsentation umgesetzt werden
- Wie auch bei anderen Medientypen besprochen gibt es zwei grundlegende Vorgehensweisen:
  - Synthese
  - Abtastung
- Wir werden kurz die Möglichkeiten zur Synthese von 3D-Szenen erörtern und dann eine Beispiel für die Abtastung sehen

6.1 Synthese

Grundlagen

- In der Regel mehrere Schritte
  - Storyboard/Skizzen
  - Erstellung von Modellen bzw. Auswahl aus Datenbanken
    - Materialien und Oberfläche der Objekte
  - Ausleuchtung der Szene
    - Prinzipien aus Film und Photo
    - Ausnutzen der Möglichkeiten digitaler Modellierung
  - Position der Kamera
  - Animation
  - Rendering

Werkzeuge: Grundlagen

- Breites Spektrum
  - Texteditor
  - Blender
  - 3D-Studio Max, Maya, Cinema4D
- Gerade bei kommerziellen Werkzeugen häufig Trennung von Modellierung/Inszenierung und Rendering
  - Komplette Szene exportiert
  - Eigene Rendering-Software

Werkzeuge: Animation, Texturen

- Animation
  - Interaktive Definition von Keyframes mit Maus & Tastatur
  - Tracking von Menschen
- Texturen
  - Mit Bildbearbeitungsprogrammen
    - PhotoShop, Gimp
  - Prozedurale Generierung

Reflexionen, Brechungen, Texturen

Abbildung 8.19

6.2 Abtastung

Abtastung

- Bisher haben wir vor allem Fälle betrachtet, bei denen das gesamte Signal entweder abgetastet oder synthetisiert wird
- Mischformen sind aber möglich
  - Mischung von Computergenerierten Inhalten mit Live-Aufnahmen, wie im Bereich Video gesehen
- Im Bereich der 3D-Graphik spielt eine weitere Unterscheidung eine Rolle:
  - Vollständige Abtastung von sowohl Tiefeninformation als auch Bildinhalten (über Texturen)
  - Abtastung von Tiefeninformation und Überlagerung mit synthetisierten Texturen
    - Beispiel Augmented Reality
- Im folgenden ein Beispiel für Abtastung von sowohl Tiefeninformation als auch Oberflächeneigenschaften

Video 10.10: Thorskan Demo

☞ Demo: Thorskan
2:19

Video 10.10: Thorskan Setup

7 Bildnachweis

Alle Abbildungen, wenn nicht anders angegeben, aus:
Malaka, Rainer; Butz, Andreas; Hussmann, Heinrich: *Medieninformatik – Eine Einführung*. ISBN 978-3-8273-7353-3, München: Pearson Studium, 2009.