Computergraphik Grundlagen

V. Die Rendering-Pipeline

Prof. Stefan Schlechtweg
Hochschule Anhalt
Fachbereich Informatik

Inhalt – Lernziele

1. Der Begriff Rendering
2. Die Rendering-Pipeline
   - Geometrische Modellierung
   - Modellierung von Kamera und Lichtquellen
   - Entfernen verdeckter Modellteile
   - Transformation der 3D-Daten in 2D
   - Überführung in ein Pixelbind
   - Bestimmen der Farbe der Pixel
   - Modellaufbau und algorithmische Schritte
   - Koordinatentransformationen
   - Zusammensetzen der Pipeline
3. Koordinatentransformationen
   - LC → WC
   - WC → EC
   - EC → NPC
   - NPC → SC
   - SC → PC
   - Zusammenfassung Transformationen

- Prozesse bei der Bilderzeugung kennenlernen
- Strukturierung der Rendering-Pipeline erkennen und Datenfluß zwischen den Teilprozessen identifizieren
- Koordinatensysteme kennen, die im Verlauf der Rendering-Pipeline verwendet werden
- Koordinatentransformationen zwischen diesen Systemen kennen

1. Der Begriff Rendering

- Bilderzeugung = Rendering

  Gegeben sei ein Modell in dreidimensionalen Koordinaten sowie die Beschreibung einer Sicht auf die modellierte Szene (Kameramodell). Gesucht ist ein Pixelbild, das das gegebene Modell möglichst realitätsnah darstellt.

1. Der Begriff Rendering

- Was wird zur Bilderzeugung benötigt?
  - Szenenbeschreibung in mathematisch-analytischer Form
    → Geometriemodell
  - Beschreibung der Sicht auf die Szene
    → Kameramodell
  - Transformation von 3D in 2D
    → Projektion
  - Möglichkeiten zu bestimmen, was sichtbar ist
    → Hidden Surface Removal
  - Beschreibung der Beleuchtung der Szene und der Interaktion des Lichts mit den Objekten
    → Lighting und Shading
  - Transformation von Vektor- in Pixeldaten → Rasterung

Gegeben sei ein Modell in dreidimensionalen Koordinaten sowie die Beschreibung einer Sicht auf die modellierte Szene (Kameramodell). Gesucht ist ein Pixelbild, das das gegebene Modell möglichst realitätsnah darstellt.

1. Der Begriff Rendering

- Der Prozeß der Bilderzeugung setzt sich aus verschiedenen Teilprozessen zusammen, die voneinander unabhängig sind und aufeinander aufbauen.
  - Modellierung von 3D-Objekten
  - Aufbau eines Gesamtmodells (Geometrie + Licht + Kamera)
  - Entfernen nicht benötigter Modellteile (Rückseiten und alle Objekte, die außerhalb des Sichtkörpers liegen)
  - Projektion
  - Entfernen von unsichtbaren Flächen (Verdeckungen)
  - Beleuchtung (Shading) + Rasterung
- Zusammenfassung dieser Prozesse in der Rendering-Pipeline

2. Die Rendering-Pipeline

- Rendering-Pipeline setzt sich aus geometrischen und algorithmischen Schritten zusammen
- Geometrisch:
  - im allgemeinen Koordinatentransformationen
  - Wechsel zwischen verschiedenen Koordinatensystemen
  - Projektion eingeschlossen
- Algorithmisch, z.B.
  - Entfernen nicht sichtbarer Flächen(-teile)
  - Beleuchtungsberechnungen
  - Rasterung (Darstellung in der Bildebene)

2. Die Rendering-Pipeline
2.1. Geometrische Modellierung

Modellierung der Geometrie

- Darstellung von 3D-Objekten durch mathematisch-analytische Methoden am Computer
- Grundlage für
  - Berechnung geometrischer Eigenschaften (Volumen, ...)
  - graphische Darstellung
  - weitergehende graphische Anwendungen
  - Berechnung des physikalisch-geometrischen Verhaltens der Körper nach einer weiteren Attributierung mit physikalischen Eigenschaften bzw. Materialparametern

2. Die Rendering-Pipeline

2.2. Modellierung von Kamera und Lichtquellen

- Kamera beschreibt Sicht auf die Szene
  - durch Position und Orientierung
  - durch interne Parameter
  - angelehnt an reale Kamera
- Lichtquellen ermöglichen die Wahrnehmung von Objekten
  - Position der Lichtquelle und geometrische Eigenschaften der Lichtausbreitung
  - Strahlungseigenschaften des Lichts, wie Farbe
  - außerdem: Reflexionseigenschaften der Objektoberflächen

Modellierung der Geometrie

Modellieren von Kamera und Lichtquellen

2. Die Rendering-Pipeline

2.3. Entfernen verdeckter Modellteile

- nicht alles in einer Szene ist sichtbar
  - Vom Betrachter wegzeigende Flächen (Rückseiten)
  - Objekte (Flächen), die außerhalb des Sichtfeldes liegen
  - (Teil-)Flächen, die von anderen Objekten in Sichtrichtung verdeckt werden
- Entfernen dieser nicht sichtbaren Anteile spart Rechenzeit bei den anderen Prozessen
- aber: oftmal selber sehr aufwändige Älgorithmen

Modellierung der Geometrie

Modellieren von Kamera und Lichtquellen

Entfernen verdeckter Modellteile

2. Die Rendering-Pipeline

2.4. Transformation der 3D-Daten in 2D

- modellierte Kamera beschreibt eine Sicht auf die Szene und somit, wie die Projektion von 3D in 2D auszuführen ist
  - hauptsächlich bestimmt durch die internen Parameter der Kamera
  - nur der Bereich im Sichtkegel der Kamera muß projiziert werden

Modellierung der Geometrie

Modellieren von Kamera und Lichtquellen

Entfernen verdeckter Modellteile

Projektion

2. Die Rendering-Pipeline

2.5. Überführung in ein Pixelbild

- Koordinaten liegen in kontinuierlicher Form vor (Vektordaten, 3D)
- Ergebnis der Bilderzeugung ist ein Pixelbild mit diskreten Bildpunkten
- Bestimmung, welcher Punkt in der Szene auf welches Pixel abgebildet wird

Modellierung der Geometrie

Modellieren von Kamera und Lichtquellen

Entfernen verdeckter Modellteile

Projektion

Rasterung

2. Die Rendering-Pipeline

2.6. Bestimmen der Farbe der Pixel

- Mit der Rasterung ist klar, welcher Ort im Pixel dargestellt wird
- Farbe des Pixels ergibt sich aus den Beleuchtungsverhältnissen in der Szene, genauer an dem im Pixel dargestellten Ort
- Beleuchtung muß aus Lichtquellen und Oberflächeneigenschaften berechnet werden
  - Art der Berechnung bestimmt Realismusgrad des erzeugten Bildes
- Überführung der berechneten Beleuchtungswerte in die Pixelfarbe
  - Verfahren so wählen, daß für möglichst wenig Punkte in 3D die Beleuchtung direkt berechnet werden muß

Modellierung der Geometrie

Modellieren von Kamera und Lichtquellen

Entfernen verdeckter Modellteile

Projektion

Rasterung

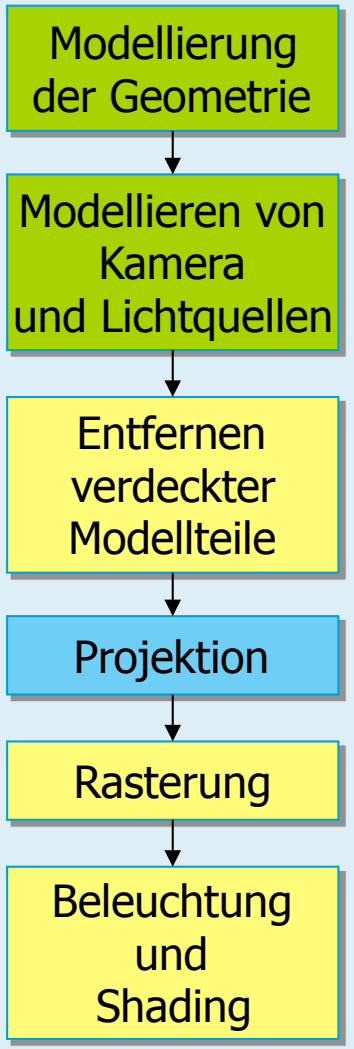
Beleuchtung und Shading

2. Die Rendering-Pipeline

2.7. Modellaufbau und algorithmische Schritte

- Geometrieobjekte, Kamera und Lichtquellen müssen erstellt werden: Modellaufbau
- Algorithmen arbeite auf dem Modell und berechnen neue Informationen (Sichtbarkeit, Pixelpositionen und –farben) Algorithmische Schritte der Rendering-Pipeline
- Projektion als Koordinatentransformation gehört zu den geometrischen Schritten der Rendering-Pipeline

- Behandlung der einzelnen Prozesse später in de Lehrveranstaltung



2. Die Rendering-Pipeline

2.8. Koordinatentransformationen

- Während der Abarbeitung der Rendering-Pipeline sind Koordinatentransformationen notwendig
  - Projektion (3D → 2D)
  - Kontinuierliche 2D-Daten → Pixelpositionen im Anzeigefenster
- weitere Koordinatensysteme erleichtern die Algorithmen durch geometrische Konventionen
- geometrische Schritte der Rendering-Pipeline: Kette von Koordinatentransformationen
  - Verschiedene Koordinatensysteme
  - Transformationen zwischen diesen

2. Die Rendering-Pipeline
2.8. Koordinatentransformationen

- Lokale Koordinaten:
  - Modellierung der Objekte erfolgt in lokalen Koordinaten
  - jedes Objekt hat sein „eigenes“ Koordinatensystem und ist in diesem definiert
  - Intrinsisches Koordinatensystem eines Objektes
  - effektiv zur Modellierung
- Weltkoordinaten
  - Bezugskoordinatensystem der Szene
  - Einheitliche Koordinaten, in dem alle Objekte der Szene sowie Kamera und Lichter definiert sind
  - Bezüge zwischen Objekten können hergestellt werden
  - effektiv für Beleuchtungsberechnungen

lokale Koordinaten

Welt-Koordinaten

2. Die Rendering-Pipeline

2.8. Koordinatentransformationen

- Kamerakoordinaten (Viewing-Koordinaten)
  - ebenfalls globales Koordinatensystem für die Szene
  - Kamera als Bezugspunkt (Ursprung)
  - Im Prinzip: Betrachten der Welt aus Sicht der Kamera
  - effektiv zum Entfernen von Rückseiten
- Normalisierte Projektionskoordinaten
  - Umrechnung basierend auf internen Kameraparametern in ein System mit normierter Größe $[-1\ldots1]^3$ oder $[-1\ldots1]^2\times[0\ldots1]$
  - Effektiv zum Entfernen der Objekte außerhalb des Sichtkegels

lokale Koordinaten

Welt-Koordinaten

Kamera-Koordinaten

Normalisierte Projektions-Koordinaten

2. Die Rendering-Pipeline
2.8. Koordinatentransformationen

- (Normalisierte) Bildschirmkoordinaten
  - zweidimensional: $[0\ldots1]^2$
- Rasterkoordinaten
  - Pixelkoordinaten auf dem Ausgabegerät
  - $[0\ldots x_{max}]\times[0\ldots y_{max}]$

lokale Koordinaten

Welt-Koordinaten

Kamera-Koordinaten

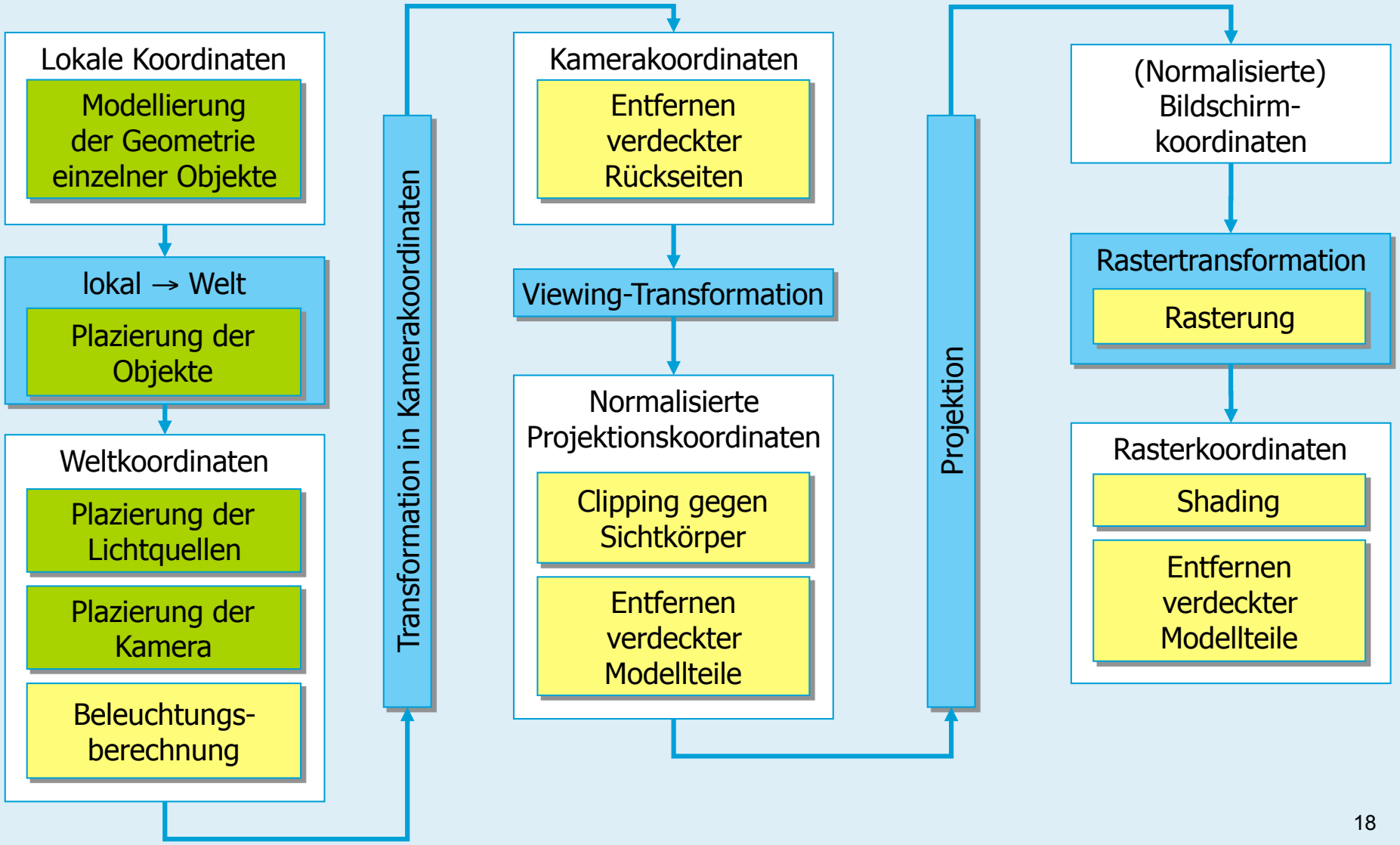
Normalisierte Projektions-Koordinaten

(Normalisierte) Bildschirm-Koordinaten

Raster-Koordinaten

2. Die Rendering-Pipeline

2.9. Zusammensetzen der Pipeline



2. Die Rendering-Pipeline

2.9. Zusammensetzen der Pipeline

- ein Beispiel für die Abfolge in einer Rendering-Pipeline
- abhängig von der Art der verwendeten Algorithmen
- insbesondere Hidden-Surface-Removal (Entfernen verdeckter Objektteile) in verschiedenen Ausführungen an unterschiedlichen Stellen der Pipeline
- auch Erweiterungen möglich
  - Texture-Mapping
  - Animation
  - Real-Time
  - weitere Algorithmen
- Pipeline kann auch komplett anders aussehen bei globaler Beleuchtung
  - Raytracing
  - Radiosity

3. Koordinatentransformationen

lokale Koordinaten

Welt-Koordinaten

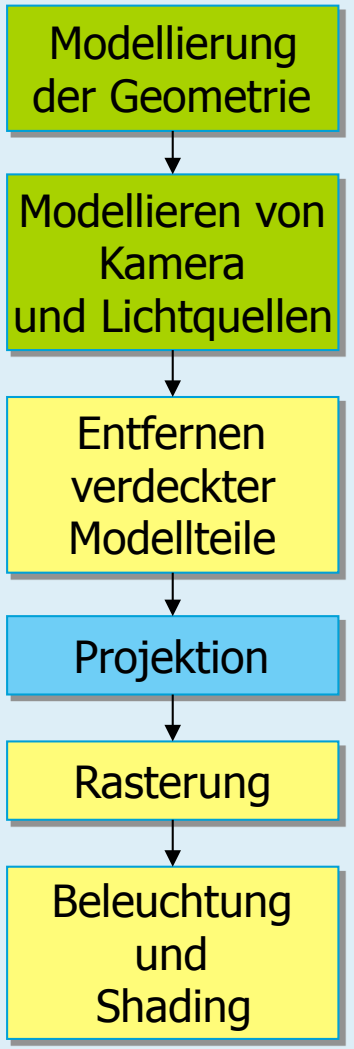
Kamera-Koordinaten

Normalisierte Projektions-Koordinaten

(Normalisierte) Bildschirm-Koordinaten

Raster-Koordinaten

- Rendering als Abfolge mehrerer Koordinatentransformationen
- Rendering als Abfolge verschiedener Prozesse
- Zusammenfassung beider Sichtweisen ergibt (eine Variante der) Rendering-Pipeline
- Frage: Welche Transformationen sind notwendig und welche Algorithmen werden in welchem Koordinatensystem ausgeführt?



3. Koordinatentransformationen

3.1. Lokale Koordinaten in Weltkoordinaten

lokale Koordinaten

Welt-Koordinaten

Kamera-Koordinaten

Normalisierte Projektions-Koordinaten

(Normalisierte) Bildschirm-Koordinaten

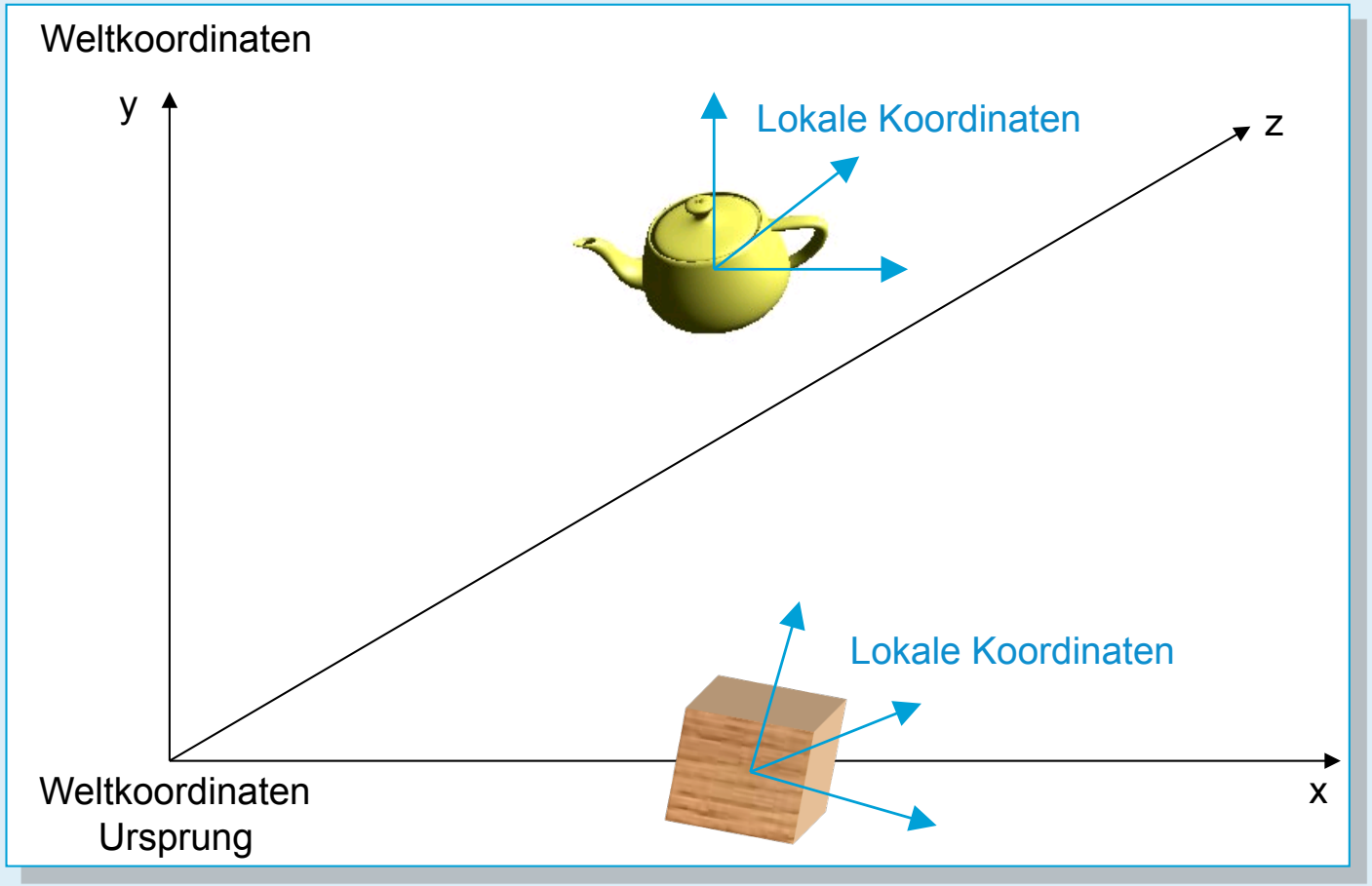
Raster-Koordinaten

- Voraussetzung:
  - Objekte liegen in lokalen Koordinaten vor, in denen sie modelliert wurden
  - Ursprung des Weltkoordinatensystems ist bekannt
- Ziel:
  - Alle Objekte sind im einheitlichen Weltkoordinatensystem beschrieben
- Erforderlich:
  - Objekte erhalten ihre Position, Orientierung und Größe im Weltkoordinatensystem durch Translation, Rotation und Skalierung

3. Koordinatentransformationen

3.1. Lokale Koordinaten in Weltkoordinaten

- lokale Koordinaten
- Welt-Koordinaten
- Kamera-Koordinaten
- Normalisierte Projektions-Koordinaten
- (Normalisierte) Bildschirm-Koordinaten
- Raster-Koordinaten



3. Koordinatentransformationen

3.2. Weltkoordinaten in Kamerakoordinaten

lokale Koordinaten

Welt-Koordinaten

Kamera-Koordinaten

Normalisierte Projektions-Koordinaten

(Normalisierte) Bildschirm-Koordinaten

Raster-Koordinaten

- Voraussetzung
  - Objekte der Szene liegen in einem einheitlichen Weltkoordinatensystem vor
- Ziel
  - Objekte im Kamerakoordinatensysten, d.h. Kameraposition ist Koordinatenursprung, Blickrichtung der Kamera entspricht z-Achse
- Erforderlich:
  - Erstellen des Kamerakoordinatensystems
  - Translation, um Ursprünge aufeinander abzubilden
  - Basiswechsel, um Koordinatensysteme ineinander zu überführen

3. Koordinatentransformationen

3.2. Weltkoordinaten in Kamerakoordinaten

3.2.1. Erstellen des Kamerakoordinatensystems

lokale Koordinaten

Welt-Koordinaten

Kamera-Koordinaten

Normalisierte Projektions-Koordinaten

(Normalisierte) Bildschirm-Koordinaten

Raster-Koordinaten

- Lage und Orientierung der Projektionsebene (view plane) im Raum eindeutig festlegen
  - Referenzpunkt in der Projektionsebene festlegen (view reference point, VRP)
  - Normalenvektor der Projektionsebene festlegen (view plane normal, VPN)
    - definiert die n-Achse des Kamerakoordinatensystems
  - Rotation der Projektionsebene um die VPN wird durch den View-Up-Vektor (VUP) festgelegt
    - darf nicht senkrecht auf Projektionsebene stehen
    - senkrechte Projektion des VUP auf Projektionsebene definiert v-Achse des Kamerakoordinatensystems
  - Koordinatensystem in der Projektionsebene braucht zweite Achse:
    - Rotation der v-Achse um -90° → u-Achse
  - Angabe von (u_{min},v_{min}) und (u_{max},v_{max}) definiert sichtbaren Bereich auf der Projektionsebene
- Vereinfachtes Kameramodell

3. Koordinatentransformationen

3.2. Weltkoordinaten in Kamerakoordinaten

3.2.1. Erstellen des Kamerakoordinatensystems

lokale Koordinaten

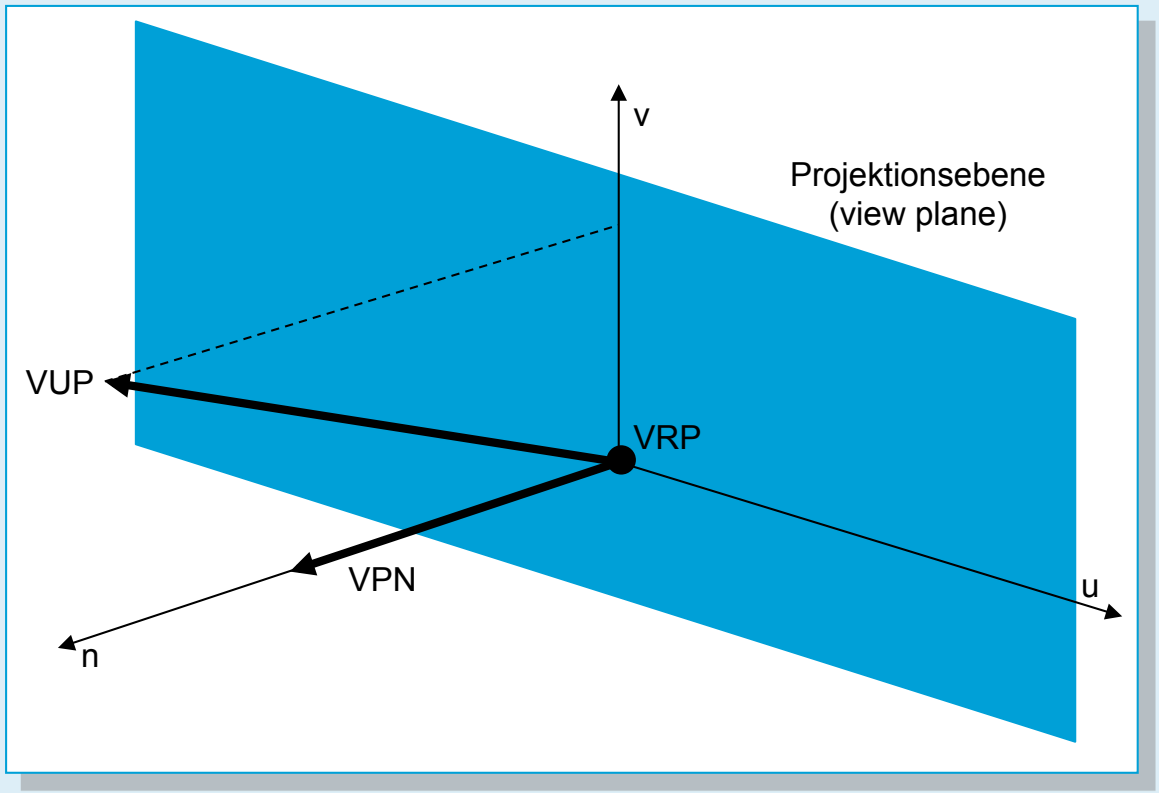
Welt-Koordinaten

Kamera-Koordinaten

Normalisierte Projektions-Koordinaten

(Normalisierte) Bildschirm-Koordinaten

Raster-Koordinaten



3. Koordinatentransformationen

3.2. Weltkoordinaten in Kamerakoordinaten

3.2.2. Überführen Kamerakoordinaten in Weltkoordinaten

lokale Koordinaten

Welt-Koordinaten

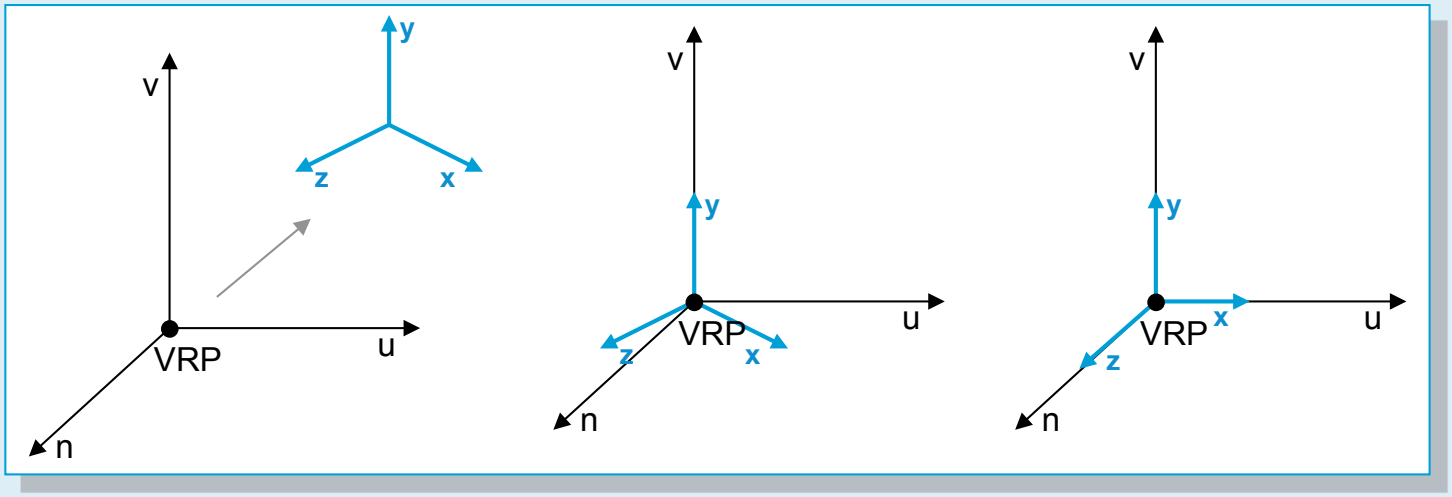
Kamera-Koordinaten

Normalisierte Projektions-Koordinaten

(Normalisierte) Bildschirm-Koordinaten

Raster-Koordinaten

- Zu erreichen:
  - Projektionsebene ist mit einer der Koordinatenebenen ausgerichtet
  - VRP liegt im Ursprung
- Schritte:
  - Verschieben des VRP in den Ursprung
  - Rotation des verschobenen Kamerakoordinatensystems so, dass *uvn* mit *xyz* ausgerichtet ist



3. Koordinatentransformationen

3.2. Weltkoordinaten in Kamerakoordinaten

3.2.2. Überführen Kamerakoordinaten in Weltkoordinaten

lokale Koordinaten

Welt-Koordinaten

Kamera-Koordinaten

Normalisierte Projektions-Koordinaten

(Normalisierte) Bildschirm-Koordinaten

Raster-Koordinaten

- Verschieben des VRP in den Ursprung ist einfache Translation

$$T = \begin{pmatrix} 1 & 0 & 0 & -VRP_x \\ 0 & 1 & 0 & -VRP_y \\ 0 & 0 & 1 & -VRP_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- Rotation des verschobenen Kamerakoordinatensystems so, dass uvn mit xyz ausgerichtet ist
  - Der Vektor VPN soll auf die z-Ebene rotiert werden
  - Die u-Achse soll senkrecht auf VUP und VPN stehen
  - Die v-Achse soll senkrecht auf n und u stehen

$$n = \frac{\mathbf{VPN}}{\|\mathbf{VPN}\|}$$

$$u = \frac{\mathbf{VUP} \times \mathbf{VPN}}{\|\mathbf{VUP} \times \mathbf{VPN}\|}$$

$$v = n \times u$$

3. Koordinatentransformationen

3.2. Weltkoordinaten in Kamerakoordinaten

3.2.2. Überführen Kamerakoordinaten in Weltkoordinaten

lokale Koordinaten

Welt-Koordinaten

Kamera-Koordinaten

Normalisierte Projektions-Koordinaten

(Normalisierte) Bildschirm-Koordinaten

Raster-Koordinaten

- Um Punkte, die in einem Koordinatensystem (hier Weltkoordinatensystem) gegeben sind, in einem anderen (hier Kamerakoordinatensystem) auszudrücken, dass gegenüber dem ersten rotiert ist, muss man sie mit einer Matrix multiplizieren, die die Basisvektoren enthält

$$R = \begin{pmatrix} u_x & u_y & u_z & 0 \\ v_x & v_y & v_z & 0 \\ n_x & n_y & n_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- Die Gesamttransformationsmatrix lautet dann:

$$M = R \cdot T = \begin{pmatrix} u_x & u_y & u_z & -(u_x \cdot VRP_x + u_y \cdot VRP_y + u_z \cdot VRP_z) \\ v_x & v_y & v_z & -(v_x \cdot VRP_x + v_y \cdot VRP_y + v_z \cdot VRP_z) \\ n_x & n_y & n_z & -(n_x \cdot VRP_x + n_y \cdot VRP_y + n_z \cdot VRP_z) \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

3. Koordinatentransformationen

3.2. Weltkoordinaten in Kamerakoordinaten

3.2.3. Zwischenzustand

lokale Koordinaten

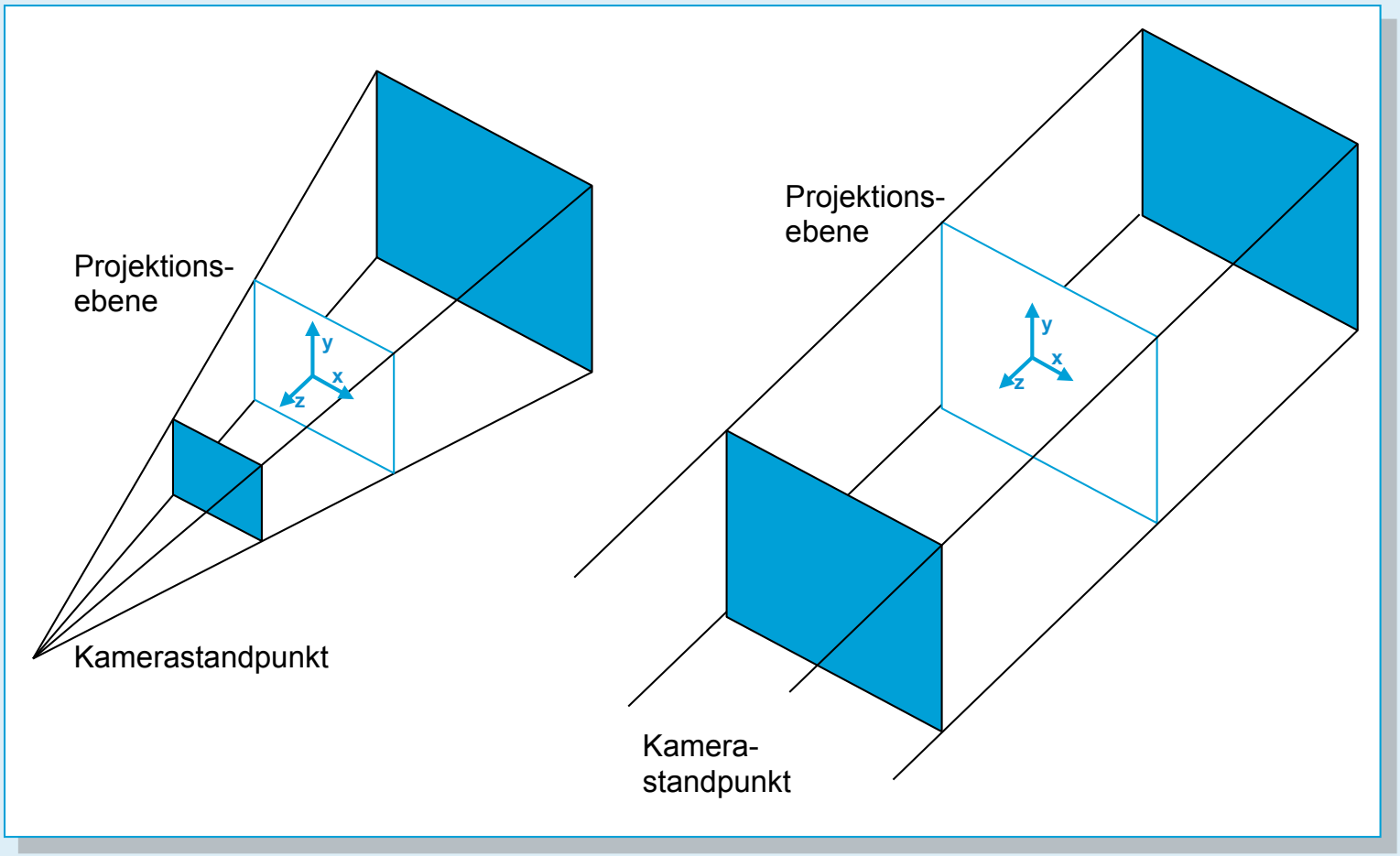
Welt-Koordinaten

Kamera-Koordinaten

Normalisierte Projektions-Koordinaten

(Normalisierte) Bildschirm-Koordinaten

Raster-Koordinaten



3. Koordinatentransformationen

3.2. Weltkoordinaten in Kamerakoordinaten

3.2.4. Perspektivische Transformation

lokale Koordinaten

Welt-Koordinaten

Kamera-Koordinaten

Normalisierte Projektions-Koordinaten

(Normalisierte) Bildschirm-Koordinaten

Raster-Koordinaten

- Bei perspektivischer Projektion muß das Ganze noch in den Kamerastandpunkt (projection reference point, PRP) verschoben werden, also zusätzlich noch die Translation

$$T_{zen} = \begin{pmatrix} 1 & 0 & 0 & -PRP_x \\ 0 & 1 & 0 & -PRP_y \\ 0 & 0 & 1 & -PRP_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- Insgesamt: $M_{zen} = T_{zen} \cdot M$

3. Koordinatentransformationen

3.3. Kamerakoordinaten in Normalisierte Projektionskoordinaten

lokale Koordinaten

Welt-Koordinaten

Kamera-Koordinaten

Normalisierte Projektions-Koordinaten

(Normalisierte) Bildschirm-Koordinaten

Raster-Koordinaten

- Voraussetzung:
  - bisher betrachtete Transformationen sind durchgeführt
  - Ursprung der Koordinatensystems ist der Betrachterstandpunkt (PRP)
  - z-Achse zeigt jetzt in die Tiefe, y-Achse nach oben und x-Achse nach rechts
  - Annahme: z-Achse geht durch Zentrum des mit (u_{min},v_{min}) und (u_{max},v_{max}) festgelegten Sichtfensters (sonst noch Scherung notwendig)
- Ziel:
  - Transformation des Sichtkörpers in den sogenannten kanonischen Sichtkörper
    - Quader mit den beiden maximalen Eckpunkten bei (-1, -1, 0) und (1, 1, -1)
- Erforderlich:
  - Skalierung des Sichtkörpers
  - bei perspektivischer Projektion zusätzlich nichtlineare Transformation, die den Pyramidenstumpf in einen Quader überführt

3. Koordinatentransformationen

3.3. Kamerakoordinaten in Normalisierte Projektionskoordinaten

lokale Koordinaten

Welt-Koordinaten

Kamera-Koordinaten

Normalisierte Projektions-Koordinaten

(Normalisierte) Bildschirm-Koordinaten

Raster-Koordinaten

- Wichtig für die weitere Betrachtung:
  - near clipping plane und far clipping plane beschränken das Sichtvolumen in der Tiefe
  - alle Objekte vor der near clipping plane und hinter der far clipping plane werden später verworfen und nicht dargestellt
  - Gründe:
    - Rechenzeitersparnis: Objekte, die weit von der Kamera entfernt sind und im endgültigen Bild kaum sichtbar sein werden, können von den Berechnungen ausgeschlossen werden
    - Einfachere Mathematik: man umgeht Singularitäten bei der Berechnung der Projektion (Objekt im Projektionszentrum) und vermeidet, daß Objekte hinter der Kamera berechnet werden müssen

3. Koordinatentransformationen

3.3. Kamerakoordinaten in Normalisierte Projektionskoordinaten

lokale Koordinaten

Welt-Koordinaten

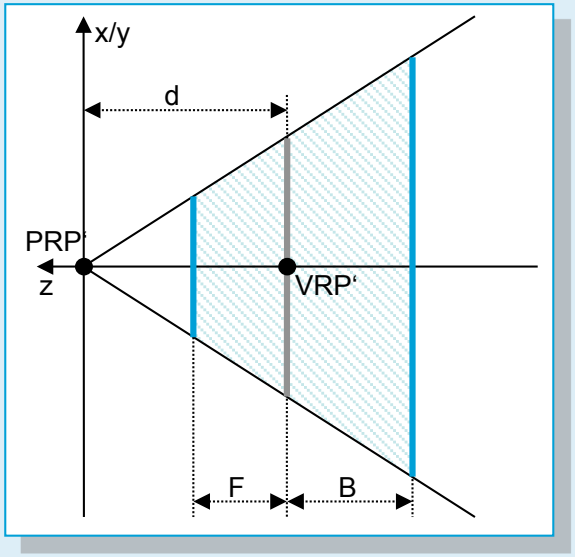
Kamera-Koordinaten

Normalisierte Projektions-Koordinaten

(Normalisierte) Bildschirm-Koordinaten

Raster-Koordinaten

- Transformation des Sichtvolumens in das kanonische Sichtvolumen erfordert zwei Skalierungen
  1. nicht-uniforme Skalierung paßt die x- und y-Achse auf die erwünschte Größe an



$$S_1 = \begin{pmatrix} \frac{2d}{u_{max} - u_{min}} & 0 & 0 & 0 \\ 0 & \frac{2d}{v_{max} - v_{min}} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

3. Koordinatentransformationen

3.3. Kamerakoordinaten in Normalisierte Projektionskoordinaten

lokale Koordinaten

Welt-Koordinaten

Kamera-Koordinaten

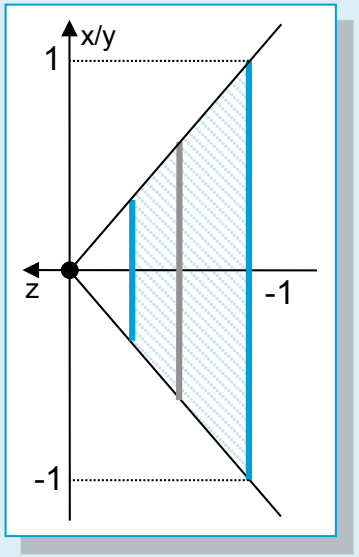
Normalisierte Projektions-Koordinaten

(Normalisierte) Bildschirm-Koordinaten

Raster-Koordinaten

- Transformation des Sichtvolumens in das kanonische Sichtvolumen erfordert zwei Skalierungen
  2. uniforme Skalierung bringt far clipping plane auf den Abstand -1

$$S_2 = \begin{pmatrix} \frac{1}{d+B} & 0 & 0 & 0 \\ 0 & \frac{1}{d+B} & 0 & 0 \\ 0 & 0 & \frac{1}{d+B} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



3. Koordinatentransformationen

3.3. Kamerakoordinaten in Normalisierte Projektionskoordinaten

lokale Koordinaten

Welt-Koordinaten

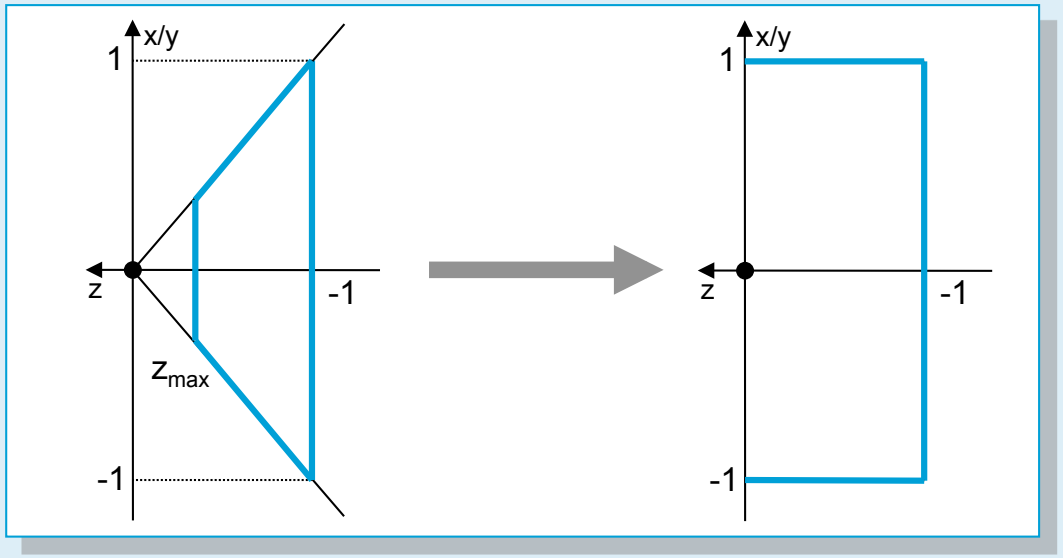
Kamera-Koordinaten

Normalisierte Projektions-Koordinaten

(Normalisierte) Bildschirm-Koordinaten

Raster-Koordinaten

- Abbildung des Sichtvolumens der perspektivi-schen Projektion in ein rechteckiges Sicht-volumen, das durch -1≤x≤1; -1≤y≤1und -1≤z≤0 begrenzt ist



3. Koordinatentransformationen

3.3. Kamerakoordinaten in Normalisierte Projektionskoordinaten

lokale Koordinaten

Welt-Koordinaten

Kamera-Koordinaten

Normalisierte Projektions-Koordinaten

(Normalisierte) Bildschirm-Koordinaten

Raster-Koordinaten

- Nicht-lineare Transformation (Division durch w ergibt die gewünschten Resultate)

$$A = \begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & \frac{-1}{1+z_{max}} & \frac{z_{max}}{1+z_{max}} \\ 0 & 0 & 1 & 0 \end{pmatrix} \quad \text{mit } z_{max} = -\frac{d-F}{d+B} \text{ und } z_{max} \neq -1$$

- Skalierung mit ½ in x- und y-Richtung ergibt dann den Sichtkörper in den gewünschten Ausmaßen

3. Koordinatentransformationen

3.3. Kamerakoordinaten in Normalisierte Projektionskoordinaten

lokale Koordinaten

Welt-Koordinaten

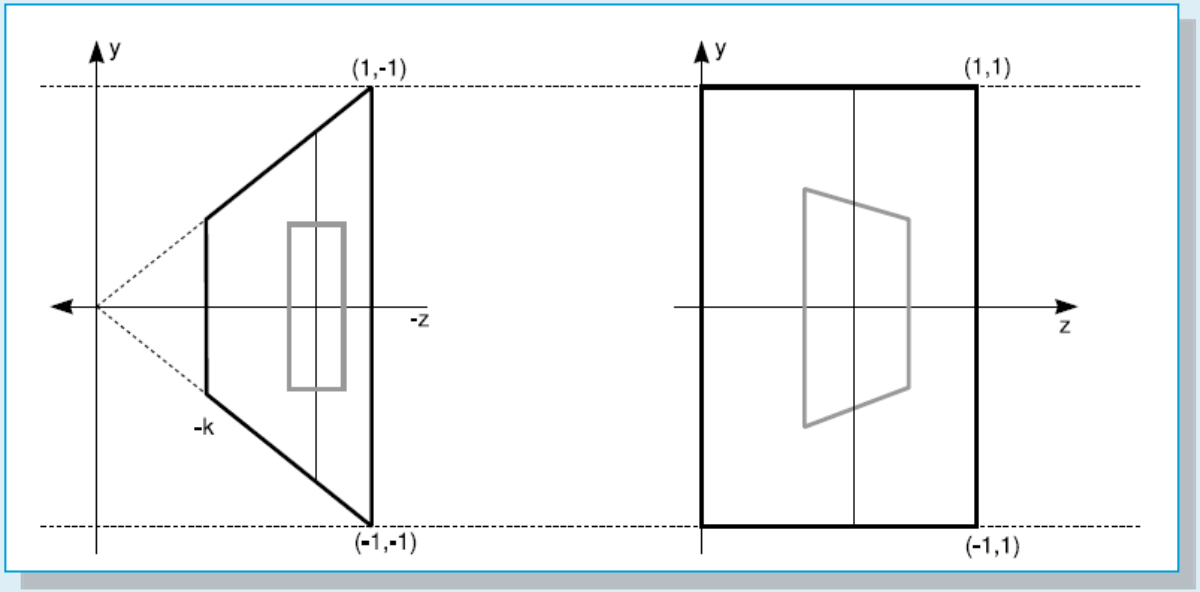
Kamera-Koordinaten

Normalisierte Projektions-Koordinaten

(Normalisierte) Bildschirm-Koordinaten

Raster-Koordinaten

- Diese Transformation erzeugt die perspektivi-sche Verzerrung vor der eigentlichen (Parallel-) Projektion. Der Unterschied zwischen der perspektivischen Transformation und einer perspektivischen Projektion besteht darin, daß erstere auch die z-Koordinate transformiert und keine Projektion durchführt.



3. Koordinatentransformationen

3.4. Normalisierte Projektionskoordinaten in Bildschirmkoordinaten

lokale Koordinaten

Welt-Koordinaten

Kamera-Koordinaten

Normalisierte Projektions-Koordinaten

(Normalisierte) Bildschirm-Koordinaten

Raster-Koordinaten

- Voraussetzung:
  - Objekte liegen jetzt im kanonischen Sichtkörper vor, der quaderförmig ist und dessen Ausdehnungen durch $-1 \leq x \leq 1$; $-1 \leq y \leq 1$und $-1 \leq z \leq 0$ gegeben sind
- Ziel:
  - Transformation in ein für die Darstellung auf dem Bildschirm ggeignetes Koordinatensystem mit Koordinaten $0 \leq x \leq x_{max}$ und $0 \leq y \leq y_{max}$, dabei Übergang zu zweidimensionalen Koordinaten
- Erforderlich:
  - Translation und Skalierung, dabei beachten, daß die y-Achse entweder nach oben oder nach unten zeigen kann

3. Koordinatentransformationen

3.4. Normalisierte Projektionskoordinaten in Bildschirmkoordinaten

lokale Koordinaten

Welt-Koordinaten

Kamera-Koordinaten

Normalisierte Projektions-Koordinaten

(Normalisierte) Bildschirm-Koordinaten

Raster-Koordinaten

- Transformationsmatrix (für nach unten zeigende y-Achse)

$$T = \begin{pmatrix} x_{max} & 0 & 0 & 0 \\ 0 & -y_{max} & 0 & y_{max} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- dritte Zeile der Matrix bewirkt einen Wegfall der z-Koordinate, also den Übergang von 3D nach 2D
- „Projektion“ als solche entspricht dem Zusammenspiel der Transformationen in Normalisierte Projektionskoordinaten und dieser letzten Transformation

3. Koordinatentransformationen

3.5. Bildschirmkoordinaten in Rasterkoordinaten

lokale Koordinaten

Welt-Koordinaten

Kamera-Koordinaten

Normalisierte Projektions-Koordinaten

(Normalisierte) Bildschirm-Koordinaten

Raster-Koordinaten

- Voraussetzung:
  - Objekte in Projektionskoordinaten
- Ziel:
  - Pixelkoordinaten für eine Darstellung auf dem Bildschirm
- Erforderlich:
  - Eigentlich keine Koordinatentransformation, sondern Rasterung – Diskretisierung der gegebenen Koordinaten auf Pixelpositionen
  - Möglicherweise Anpassung der Koordinaten (Translation) auf Fensterposition

3. Koordinatentransformationen

3.6. Zusammenfassung Transformationen

lokale Koordinaten

Welt-Koordinaten

Kamera-Koordinaten

Normalisierte Projektions-Koordinaten

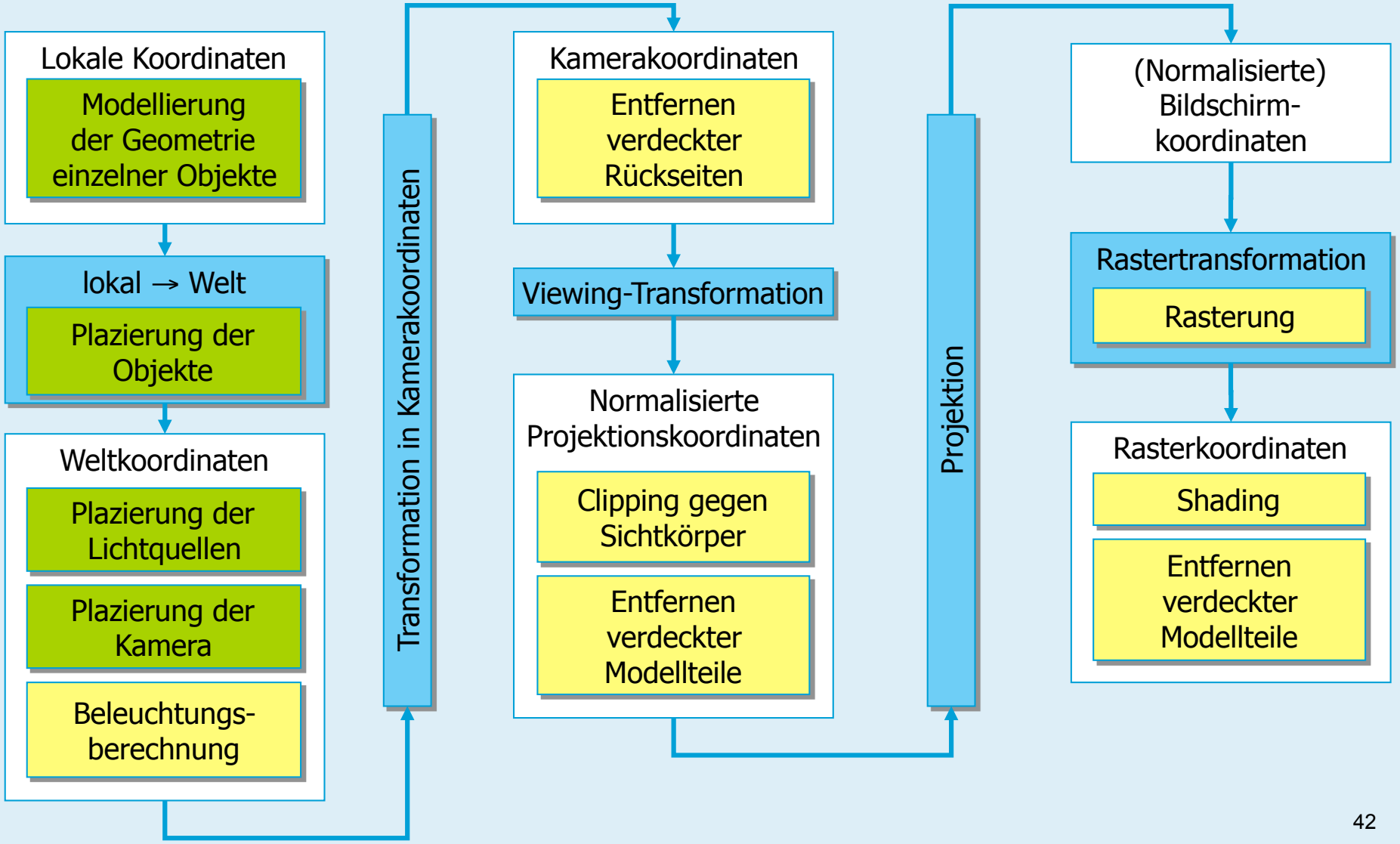
(Normalisierte) Bildschirm-Koordinaten

Raster-Koordinaten

- Rendering-Pipeline ist Kette von Koordinatentransformationen
- endgültige Pixelkoordinaten nach Anwenden der zusammengesetzten Transformation (Multiplikation aller Matrizen)
- Kette oftmals in Teile zerlegt, z.B. bei OpenGL
  - ModelView-Transformation bis ins Kamerakoordinatensystem (nur Position und Ausrichtung der Kamera notwendig
  - Projection-Transformation der Rest (interne Kameraparameter notwendig
- zwischen den Transformationen (in den einzelnen Koordinatensystemen) werden Algorithmen auf die Objektkoordinaten angewendet
- Transformationen und Algorithmen aufwändig, daher möglichst viele Objekte „abschneiden“

3. Koordinatentransformationen

3.6. Zusammenfassung Transformationen



Video: For the Birds

- Federn sind einzeln modelliert
- Academy Award for Best Animated Short Film 2002