Persönlicher Mitschrieb aus der Blockvorlesung

Einführung in die 3D-Computer-Grafik und Virtual Reality-Visualisierung

Dr. rer. Nat. Torsten Kuhlen
RWTH Aachen

vom 17. – 20. Januar 2000

Inhaltsverzeichnis

Bemerkungen

Für die Richtigkeit der Inhalte kann ich keinerlei Verantwortung übernehmen. Im Anhang befindet sich ein Auszug der Kopien, die in der Vorlesung ausgehändigt wurden. Auf diese wird teilweise im Text bezug genommen.

Mai 2000, Andreas Fritz

Definition:
Unter VR (Virtuelle Realität) versteht man eine *computergenerierte* Welt (Umgebung), die ein Betrachter mit seinen *natürlichen Sinnen* als real erlebt und mit der er *interagieren* kann.

I. Virtuelle Umgebungen

- Abbild der Realität
- Umgebungen der Vergangenheit und Zukunft
- In der Realität nicht wahrnehmbare Umgebungen
- Künstliche Welten (Artificial Reality)

I.1. Charakteristische Merkmale

- Interaktion
  - Freie Beweglichkeit in der virtuellen Umgebung (Navigationsaspekt)
  - Manipulation virtueller Objekte

- Natürliche Schnittstelle zwischen Benutzer und Computer
  - Dreidimensionaler Input/Output
  - Einbeziehung mehrerer menschlicher Sinne
    (visuell, akustisch, haptisch [taktile Rückkopplung])

- Immersion
  - Gefühl, in der Umgebung integriert zu sein

I.2. Synonyme für VR

- Cyberspace (→ Internet)
- Augmented Reality (Mischung von VR und Realität [Head-up – Display])
- Teleoperation (z. B. Fernsteuerung von weit entfernten Robotern in der Raumfahrt)
- Telemanipulation (s. Teleoperation)
- Telepräsenz (exakte VR: Gefühl, an einem bestimmten Ort zu sein)
- (Multimedia)

I.3. Historie

1949	Erste Computergrafik auf Whirlwind-Computern (MIT): Bouncing Ball	
1952	CG für Radar (SAGE-Computer)	
1956	Sensorama (M. Heilig)	
1963	Sketchpad – erstes interaktives Computergrafik-System	
1965	Erstes kommerzielles Vektor-Display	
1965	The „Ultimate“ Display, Head Mounted Display (HMD) mit Kopftracking	
1967	Erstes haptisches Display (Brooks, UNC)	
1971	Raster-Scan Prinzip (späte Einführung wegen den sehr hohen Speicherpreisen)	
1971	Gouraud-Shading	
1974	Texturierung (Catmull)	lokale Schattierungsverfahren
1975	Phong-Shading	lokale Schattierungsverfahren
1979	Raytracing	globales Schattierungsverfahren
1984	Radiosity	

1985	Erste VR-Produkte (VPL Research)
1985	VIEW: Virtual Interactive Environment Workstation, NASA
1987	British Aerospace: Virtual Cockpit
–––	***– Entwicklungssprung –***
1992	CAVE: Cave Automated Virtual Environment (Cruz Neira, Univerity of Chicago)
1993	Silicon Graphics Reality Engine: Hardware-gestütztes Gouraud-Shading, Texture-Mapping, Z-Buffering, Anti-Aliasing, 200.000 Polygone/Sekunde
1993	(Open)GL-Standard (Graphics-Kernel-System GKS 1985, GKS 3D 1988, PHIGS 1990 [hierarchisch])
1994-heute	Anwachsen der Grafikgeschwindigkeit (SGI Infinite Reality 2: 13 Mio. Polygone/Sekunde) Aufholjagd der PCs, Entwicklung von VR-Anwendungen

I.4. Repräsentation geometrischer Objekte

I.4.1. Polygonale Repräsentation

Approximation der Objekte durch Netz planarer, polygonaler Facetten

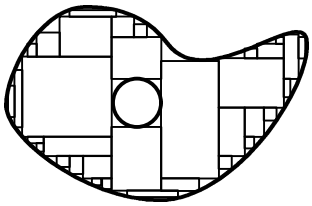
- Effiziente Schattierungsalgorithmen
- Hardwaregestütztes Rendering
- Darstellung beliebiger Geometrien
- Trade off: Genauigkeit (Anzahl Polygone) ↔ Darstellungsgeschwindigkeit

Datenstruktur: (Liste von Polygonen)

- Liste von 3D-Koordinaten (x,y,z) (Ecken der Polygone)
- Kanten implizit
- Ggf.: Normalen der Polygone und / oder Ecken (→ Schattierung)
- Hierarchische Anordnung einzelner Geometrien (Scene Graph)
- Polygone werden unabhängig voneinander dargestellt

I.4.2. Raum unterteilende Techniken

a) Octrees (2D: Quadtrees): Unterteilung einer Fläche in verschieden große Quadrate



b) Binary Space Partitioning

I.4.3. Parametrisierte Darstellung über kubische Polynome (Spline-Flächen)

I.4.4. Constructive Solid Geometry (CSG)

I.4.5. Funktionale Repräsentation

II. Rendering polygonal repräsentierter Objekte

Definition: Rendering:
Prozess der Projektion einer in einer Datenbasis abgelegten Menge von (ggf. hierarchisch) angeordneten 3D-Objekten auf eine 2-dimensionale Abbildungsoberfläche

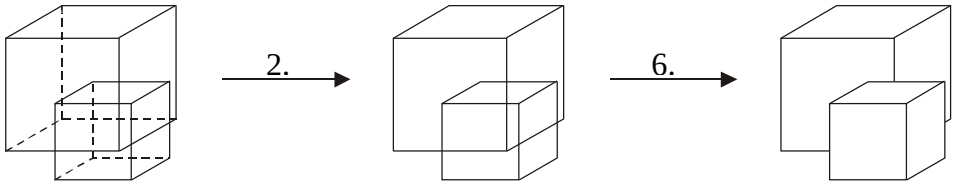
II.1. Rendering – Pipeline

II.1.1. Transformationen:

- Position und Orientierung von Objekten
- Überführung von lokalen Koordinaten einzelner Objekte in ein einziges globales Koordinatensystem
- Überführung in ein Sichtkoordinatensystem (SKS)

II.1.2. Culling:

Entfernen nicht sichtbarer Polygone



II.1.3. Perspektivische Projektion:

Überführung in Screen-Koordinatensystem

II.1.4. Clipping:

Abschneiden von Flächen außerhalb des Sichtvolumens

II.1.5. Scan Conversion:

(Rasterisierung) → Pixel

II.1.6. Hidden Surface Removal (HSR)

II.1.7. Shading:

Ermitteln der Farbwerte einzelner Pixel

Bemerkung:

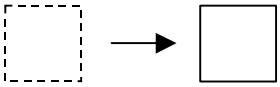
- 5. – 7.: ein einziger Algorithmus
- Reihenfolge kann variieren
- 2. ist ein Spezialfall von 6. → Performance !
- Speedup über Pipelining – Prinzip

II.2. Transformationen

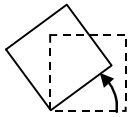
Polygon-Modell: nur Eckpunkte transformieren!
Konvention:

- Punkt als Reihenvektor (x,y,z)
- Transformation als Matrix

- Translation: $P' = P + D$

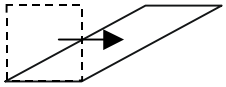


- Rotation: $P' = P \cdot R$



- Skalierung: $P' = P \cdot S$

- Scherung: $P' = P \cdot SH$



Homogene Koordinaten:
Erhöhe Dimension um 1 (2D → 3D, 3D → 4D)

→ Vereinheitlichung
→ Translation → lineare Transformation:

$P(x,y,z) \rightarrow (X,Y,Z,w)$ mit $x = X/w$, $y=Y/w$, $z=Z/w$
Konvention : $w = 1$

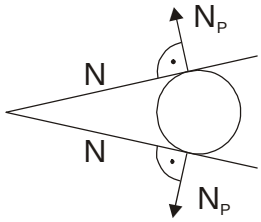
$$\rightarrow \text{Translation}: \; P' = P \cdot T \text{ mit } T = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ T_x & T_y & T_z & 1 \end{pmatrix}$$

II.3. Culling (Backface Elimination)

- Entfernen der nicht-sichtbaren Polygone
- Vergleich der Orientierung von Polygonen mit dem Viewpoint

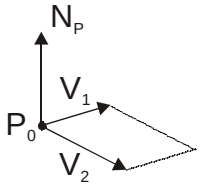
Einfacher, schneller **„Algorithmus“:**
Polygon ist unsichtbar, falls der Winkel zwischen Normalenvektor des Polygons N_P und dem „Line of sight“ – Vektor $N > 90°$ ist.

„Beweis“:



$$\Theta(N_P, N) < 90° \quad \Leftrightarrow \quad N_P \cdot N > 0 \quad \left(\cos\Theta = \frac{N_P \cdot N}{|N_P| \cdot |N|} \right)$$

- N ist Ortsvektor des Punktes im SKS
- NP wird berechnet aus 3 (nicht kollinearen) Ecken des Polygons



$V_1 = P_1 - P_0$
$V_2 = P_2 - P_0$

Bemerkungen:
- Einmalige Berechnung von N_P zu Beginn der Simulation
- Verwendung der Normalen auch beim Shading
- Im Durchschnitt ist die Hälfte aller Polygone unsichtbar

II.4. Projektion

Allgemein: n-dim. KS → n-1-dim. KS
Computergrafik: - 3D → 2D
- planare Projektion
⇒ Geraden werden auf Geraden abgebildet

Projektion

perspektivisch (Zentralprojektion)	Parallelprojektion
- Projektoren treffen sich in einem Projektionszentrum	- Projektoren laufen parallel, Zentrum im Unendlichen
-Verkleinerung der Objekte mit zunehmendem Abstand	- entspricht nicht der natürlichen Seherfahrung
- Sichtvolumen: Pyramide	- Sichtvolumen: Quader

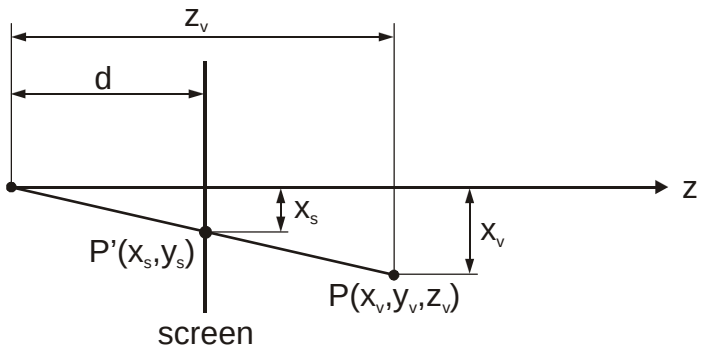
II.5. Eine *sehr* einfache Projektionsvorschrift

Annahmen:
- Zentralprojektion
- Keine „schräge“ Projektion
- Basis: SKS
- Normale der Abbildungsoberfläche ist parallel zur z-Achse
- Fester Abstand d des Projektionszentrums von der Abbildungsoberfläche

Gesucht: Für jeden Punkt P (x_v, y_v, z_v) der Objektgeometrie Schnittpunkt P’ (x_s, y_s) des zugehörigen Projektors mit der Abbildungsoberfläche.
„screen“

Ähnliche Dreiecke:

$$\frac{x_s}{d} = \frac{x_v}{z_v} \Rightarrow x_s = \frac{x_v}{\frac{z_v}{d}}$$



Homogene Koordinaten:

$$X = x_v, \quad ..., \quad w = \frac{z_v}{d}$$

$$(X, Y, Z, w) = (x_v, y_v, z_v, 1) \cdot T_{Proj} \quad \text{mit } T_{Proj} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1/d \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

II.6. VR: Betrachterzentrierte, stereoskopische Projektion

Ziel: Exakte Übereinstimmung der visuellen Wahrnehmung realer und virtueller Objekte

---

Einschub:

II.6.1. Räumliches Sehen

Physiologische Anhaltspunkte

- Stereopsis: Interokularer Abstand → Disparität der Bilder auf Netzhaut
- Okulomotorische Faktoren
  - Akkomodation: Verformung der Linse („Scharfstellen“)
  - Konvergenz: Rotation der Augen – Sehachsen schneiden sich im fixierten Punkt
- Bewegunsparallaxe: relative Positionen verschieden weit entfernter Objekte ändern sich bei Kopfbewegungen unterschiedlich stark

Psychologische Anhaltspunkte

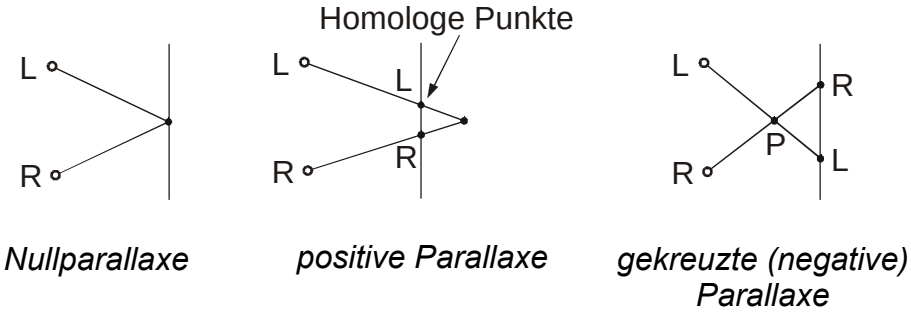
- Perspektivische Verkürzung
- Überdeckung von Objekten
- Licht und Schatten
- Texturgradienten
- Atmosphärische Perspektive
- Größe bekannter Gegenstände

II.6.2. VR-Sichtsysteme

- Kopffeste Displays: bedingt geeignet im industriellen Einsatz
- Im folgenden: Kopfferne, raumkonstante Displays

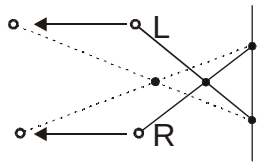
Stereopsis: → Stereobrillen
- Rot-grün (nur schwarz-weiße Darstellung möglich)
- Shutter (Nachteil: Nachleuchten [„ghosting“])
- Polarisation (2 Projektoren notwendig, Kopfdrehungen)

Je 1 Projektionszentrum pro Auge
⇒ Stereogramme



Problem: Starre Stereogramme nur korrekt für eine bestimmte Augenposition

Objekt bewegt sich mit Augenposition mit!
Für VR nicht akzeptabel!

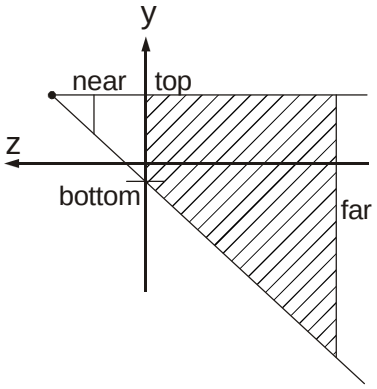


Lösung: Anpassung des Stereogramms an Augenposition des Betrachters in Echtzeit, Messung der Augenposition z.B. elektromagnetisch (Transformation: Sensor → Auge links/rechts)

Þ Betrachterzentrierte Projektion

II.6.3. Annahmen:

- Zentralprojektion
- Ausgangsbasis: Koordinatensystem mit
  - Ursprung in der Nähe des Abbildungsfensters
  - z-Achse parallel zur Normalen der Abbildungsebene
- Lage des Abbildungsfensters konstant im 3D-Raum
  top, bottom, left, right, near, far
- Variable Augenposition $E(E_x, E_y, E_z)$
  ⇒ i.d. Regel schräge Projektion



Aufgabe:

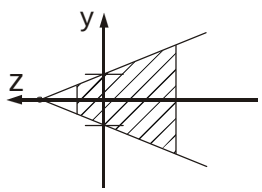
Überführung in eine Parallelprojektion im Einheitswürfel (Kanonisches View-Volume)

II.6.4. Entwicklung der Projektionsmatrix in 6 Schritten

1. Schritt: Scherung des View-Volume

⇒ E auf der z-Achse, Koordinaten des Fensters (und der abzubildenden Punkte) unverändert

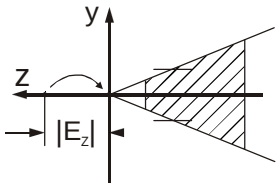
$$P_1 = SH_{xy}\left(\frac{-E_x}{E_z}, \frac{-E_y}{E_z}\right) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \frac{-E_x}{E_z} & \frac{-E_y}{E_z} & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



$$\Rightarrow (x', y', z', w') = (x, y, z, 1) \cdot SH_{xy} = \left(x - \frac{E_x}{E_z} \cdot z, y - \frac{E_y}{E_z} \cdot z, z, 1\right)$$

y' = y für z = 0
y' = y – E_y für z = E_z

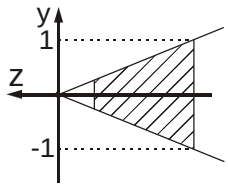
2. Schritt : Translation von E auf den Ursprung
⇒ im 4. Schritt kann durch z geteilt werden.



$$P_2 = T(0,0,-E_z) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -E_z & 1 \end{pmatrix}$$

3. Schritt:

$$P_3 = S = \begin{pmatrix} \frac{2}{right - left} & 0 & 0 & 0 \\ 0 & \frac{2}{top - bottom} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



Skalierung des Fensters auf „Einheits-Rechte-Kante“
Obere, rechte Ecke des Abbildungsfensters:

$$(right', top', z', 1) = (right, top, z, 1) \cdot S = \left(right \cdot \frac{2}{right - left}, top \cdot \frac{2}{top - bottom}, z, 1\right)$$

(left = –right, top = –bottom)

4. Schritt: Skalierung als Funktion von z

Pyramide → Quader
Zentralprojektion → Parallelprojektion

$$P_4 = SZ = \begin{pmatrix} E_z & 0 & 0 & 0 \\ 0 & E_z & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \quad \Rightarrow \quad (x', y', z', 1) = (x, y, z, 1) \cdot SZ = (x \cdot E_z, y \cdot E_z, 1, -z)$$

Normalisierung: $\left(\frac{-x \cdot E_z}{z}, \frac{-y \cdot E_z}{z}, \frac{-1}{z}, -1 \right)$ (*)

Geradengleichung für die y-Komponente der oberen Clipping-Ebene:

(1) $y = \frac{-1}{E_z} \cdot z$

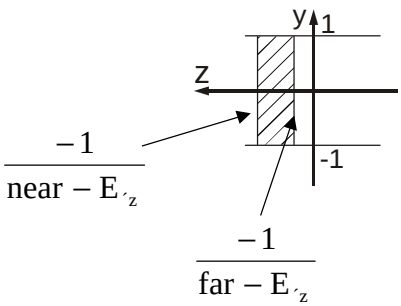
x-Komponente der beiden oberen Schnittgeraden zwischen oberer und linker/rechter Clipping-Ebene:

(2) $x = \frac{\pm 1}{-E_z} \cdot z$

Einsetzen von (1) und (2) in (*):

$$\left(\frac{\pm 1}{-E_z} \cdot z \cdot \frac{E_z}{z}, \frac{-1}{-E_z} \cdot z \cdot \frac{E_z}{z}, \frac{-1}{z}, 1 \right) = \left(\pm 1, 1, \frac{-1}{z}, 1 \right)$$

$\Rightarrow$ Kanten des View-Volumens haben (unabhängig von z) den Wert ±1.

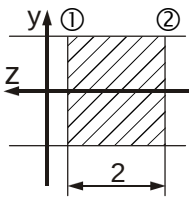


Verzerrung entlang der z-Achse

5. Schritt: Skalierung des View-Volumens in z-Richtung auf Länge 2

$$P_5 = S\left(1, 1, \frac{2(far - E_z)(near - E_z)}{far - near}\right)$$

$$\left| \frac{2(far - E_z)(near - E_z)}{far - near} \cdot \frac{-1}{far - E_z} - \frac{2(far - E_z)(near - E_z)}{far - near} \cdot \frac{-1}{near - E_z} \right| = 2 \quad \checkmark$$

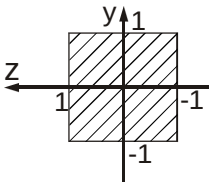


① $-\frac{2(near - E_z)}{far - near}$

② $-\frac{2(far - E_z)}{far - near}$

6. Schritt: Translation in z-Richtung → View-Volume symmetrisch zum Ursprung

$$P_6 = T\left(0,0,-\frac{2\cdot E_z - far - near}{far - near}\right)$$



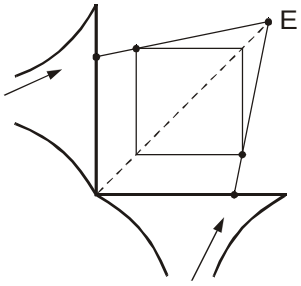
Vordere Clipping-Ebene

$$\left(x, y, -\frac{2(near - E_z)}{far - near}, 1\right)\cdot P_6 = (x, y, 1, 1)$$

Projektionsmatrix für die betrachterzentrierte Projektion:

$$\begin{pmatrix} \frac{2E_z}{right - left} & 0 & 0 & 0 \\ 0 & \frac{2E_z}{top - bottom} & 0 & 0 \\ \frac{-2E_x + right + left}{right - left} & \frac{-2E_y + top + bottom}{top - bottom} & \frac{2E_z - far - near}{far - near} & -1 \\ -\frac{right + left}{right - left}\cdot E_z & -\frac{top + bottom}{top - bottom}\cdot E_z & \frac{-E_z(2E_z - far - near) + 2(far - E_z)(near - E_z)}{far - near} & E_z \end{pmatrix}$$

II.6.5. Kombination mehrerer Abbildungsebenen:



II.7. Clipping

- punktweise: Einfach im Kanonischen Einheitswürfel
- Polygon Clipper: Sutherland-Hodgeman-Algorithmus
  Skalarprodukt-Test

II.8. Hidden Surface Removal (HSR)

z-Buffer: Speicher, der für jeden Pixel den größten z-Wert beinhaltet

Genauigkeit abhängig von

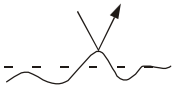
- Länge der Worte im z-Buffer (üblich: 24 Bit / Pixel)
- Wahl der vorderen und hinteren Clipping-Ebene

Achtung: Es müssen die Punkte der 3D-Objekte verglichen werden, die auf demselben Projektor liegen. ⇒ Parallelprojektion (Bei á – Blending Szene von hinten nach vorne sortieren)

III. Licht und Reflexion

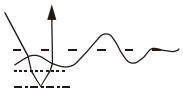
Lokale Reflexionsmodelle: Interaktion zwischen den Oberflächenpunkten der grafischen Objekte und den Lichtquellen, **keine** Reflexion von Oberfläche zu Oberfläche.

Reale Oberfläche



a) Reflexion 1. Ordnung

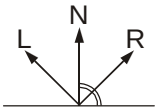
b) Mehrflächen-Reflexion



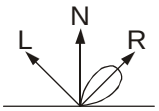
c) Subflächen-Reflexion

a)

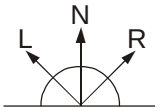
- Spiegelreflexion (Specular reflection) bei ideal glatter Oberfläche



- Gerichtete diffuse Reflexion



b) und c) diffuse Reflexion



Anteile abhängig von:

- Einfallswinkel des Lichts
- Betrachterstandpunkt
- Wellenlänge
- Oberflächenbeschaffenheit

III.1. Das Reflexionsmodell von Phong

Linearkombination aus 3 Komponenten:

– Diffuse – Specular – Ambient

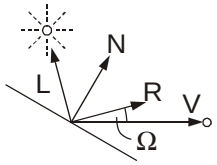
– Diffuse Komponente:

I_i : Intensität einer punktförmigen Lichtquelle
k_d: Reflexionskoeffizient
è: Winkel zwischen Polygonnormale N und „Lichtvektor“ L

$$I_d = I_i \cdot k_d \cdot \cos(q)$$

Vektorschreibweise: $I_d = k_d \cdot \sum I_{i,n} (L_n \cdot N)$

– Specular Komponente:



Ù: Winkel zwischen Blickrichtung und Reflexionsvektor
n: Index für Oberflächenrauhigkeit (perfekter Spiegel: $n \to \infty$)

$I_s = I_i \cdot k_s \cdot \cos^n \Omega$

– Ambiente Komponente:

$I_g = I_a \cdot k_a$
Beleuchtung von Flächen, die von der Lichtquelle aus nicht sichtbar sind.

Gesamtintensität: $I = I_a \cdot k_a + I_i \left(k_d \cdot (L \cdot N) + k_s \cdot (R \cdot V)^n \right)$

Problem: Berechnung von R

1. Annahme: Lichtquelle und Viewpoint im Unendlichen $\Rightarrow$ L und V konstant
2. Ersetzen von R· V durch N· H mit $H := \frac{L+V}{2}$

$\Rightarrow$ I hängt nur noch von N ab

Zusammenfassung:

- Lichtquellen punktförmig, keine Intensitätsverteilung
- Lichtquelle und Betrachter im Unendlichen
- Diffuse- und Specular-Komponente lokal
- Empirisches Modell für Specular-Komponente
- Farbe der Specular-Komponente gleich Farbe der Lichtquelle

III.2. Shading

Definition: (Inkrementelles, interpolatives) Shading
Anwendung eines Reflexionsmodells auf Polygone durch Berechnung der Intensitäten an den Polygonecken und Interpolation dieser Werte für die inneren Punkte

Computergrafik: Phong Reflexionsmodell

→ Flat Shading	–
	Qualität Komplexität
→ Gouraud Shading	
→ Phong Shading	+

III.2.1. Flat Shading

Keine Interpolation: Konstante Intensität der inneren Knoten

III.2.2. Gouraud Shading

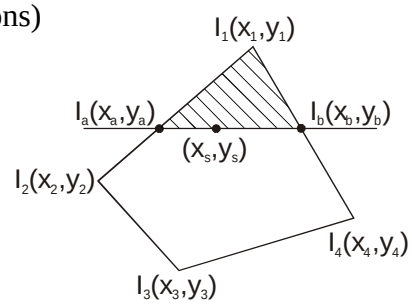
→ Diffuse Komponente

1. Berechnung der Eckennormalen als Durchschnitt aus den angrenzenden Polygonnormalen
2. Berechnung der Intensitäten an jeder Ecke gemäß Phong-Modell
   $I_v = I_i \cdot k_d \cdot (L \cdot N)$
3. Interpolationsprozess, zeilenweise (Scanline order) für die inneren Punkte:

a) Intensität an Anfangs und Endpunkt (Kanten des Polygons)

$$I_a = \frac{1}{y_2 - y_1}\left(I_1(y_s - y_2) + I_2(y_1 - y_s)\right)$$

$$I_b = \frac{1}{y_4 - y_1}\left(I_1 \ldots \qquad\qquad\qquad\qquad \right)$$



b) Intensität im Inneren der Scanline:

$$I_s = \frac{1}{x_b - x_a}\left(I_a(x_b - x_s) + I_b(x_s - x_a)\right)$$

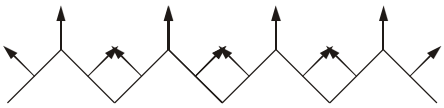
Diese Gleichung muss für jedes Pixel berechnet werden!

→ Implementierung als inkrementelle Berechnung

Probleme des Gouraud-Shading

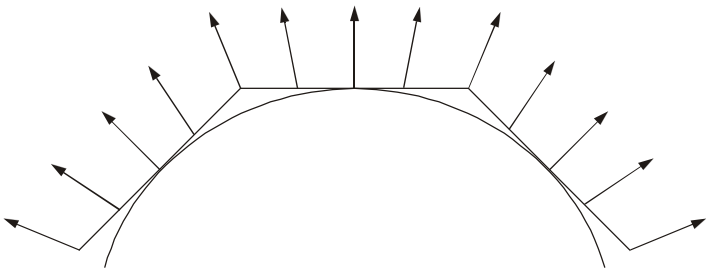
- Keine Highlights in der Mitte von Polygonen

- „Wellenauslöschung“



III.2.3. Phong Shading

Rezept: Ersetze Interpolation der Intensität durch Interpolation der Eckennormalen



- Phong Shading: 3 mal komplexer als Gouraud (→ Vektor!)
- Außerdem: Berechnung der Intensität an jedem Pixel

Beschleunigung: Kombination von Gouraud und Phong
Interpoliere N nur für jedes 2. Pixel, dazwischen nur Intensität

IV. Scan Conversion (Rasterisierung)

Definition:
Scan Conversion bezeichnet den Vorgang, jedem Pixel des Abbildungsfensters einen Wert zuzuweisen.

Rasterisierung von Linien (bei Wireframe-Darstellung)
→ lineare Sequenz von Pixeln „ohne Lücken“
⇒ Bresenham-Algorithmus, 1965

IV.1. Im folgenden: „Solid Objects“

Polygonfill über horizontale Liniensegmente (Scanlines)

Gesucht:
rechte/linke Grenze eines Elements bzw. Schnittpunkte der Scanlinie mit den Polygonkanten

Algorithmus: „Digital Differential Analyser“ DDA

Seien (x_s, y_s), (x_e, y_e) Start- und Endpunkt einer Polygonkante, o. B. d. A. $y_e > y_s$

```
x := xs
m := (xe-xs)/(ye-ys)
for y := ys to ye do
   output(round(x),y)
   x := x + m
```

m, x: float

Integer-Variante : Swanson und Tayor (1986)

Bislang : Pixel entlang der Polygonkanten, jetzt :

IV.2. Füllen der Polygone

Datenstruktur: Kantenliste für jedes Polygon als Array A
lineare Listen
Alle A[y] auf NIL
(*)

```
Für jede Kante (xs, ys), (xe,ye) des Polygons:
  For y := ys to ye:
    Wende DDA-Algorithmus an
       output(round(x),y) → Hänge round(x) an die Liste A[y]
    Sortiere A[y] nach aufsteigendem x
```

Bemerkungen:

- Geschlossene Polygone → gerade Anzahl von Listeneinträgen
- Algorithmus geeignet für konkave Polygone mit Löchern
- Insertsort: Sortieren direkt beim Einfügen in die Liste
- Verwendung ausschließlich konvexer Polygone: nur zwei Einträge pro Array-Element

Scan Conversion Algorithmus:

```
Für jedes Polygon:
  Konstruiere Kantenliste A (*)
  For y := y_min to y_max do
    Für jedes Paar (x_i, x_i+1) in A[y]
      Schattiere (x_i, y) bis (x_i+1, y)
```

Bemerkungen:
Hier: Rendering Polygon für Polygon
→ Reihenfolge der Polygone unwichtig
→ z - Buffer
→ gemeinsame Kanten werden doppelt gerendert

Alternative: Scan line order
→ mehrere Polygone gleichzeitig im Speicher

IV.3. Grafik-Hardware

Beschleunigung und Verbesserung

- Parallelismus
  - Pipelining → „makroskopisch“ (Rendering Pipe)
    → on chip (Float-Multiplikation)
  - „echt“ parallel (SIMD, ...)
- Chipentwicklung, Taktraten
- schnellere, breitere Busse
- Speicher (Auflösung, Farbauflösung, Texturen, z - Buffer)

2 Strategien

- Geometrie auf CPU, Scan Conversion auf Grafikkarte
- Beides auf Grafikkarte

2 grundsätzliche Parallelarchitekturen

- Partition
- Komposition

V. 3D - Akustik

V.1. Räumliches Hören

Lokalisation eines Schallsignals im Raum

Zusammensetzung aus:

- Lautstärke
- Tonhöhe
- Zeitlicher und spektraler Verlauf

Unterschiedlich an beiden Ohren

„binaurale Signale“

Information über Richtung und Entfernung

Kopfbezogenes Koordinatensystem

- Ursprung = Mittelpunkt des Kopfes
- Horizontalebene: Azimut-Winkel ϕ
- Medianebene: Elevationswinkel θ
- (Frontalebene)

V.1.1. Richtungshören in der Horizontalebene

Unterschiede des Schalldrucks und der Laufzeiten an beiden Ohren

Abgewandtes Ohr:
→ längere Laufzeit (0,5 – 1 ms)
→ frequenzabhängige Pegeldämpfung (10 – 25 dB) ab 150 Hz

V.1.2. Richtungshören in Medianebene

Informationsquelle: monoaural
- Spektrale Merkmale durch Reflexion an Ohrmuschel, Kopf und Schulter

V.1.3. Entfernungshören

Hauptmerkmal: Schalldruckpegel
+ Wissen über Art der Schallquelle

In Innenräumen:
Energiedifferenz zwischen Direktschall und Reflexion

Zusätzliche Hilfsmittel:
- Auge
- Peilbewegungen

V.2. Richtungsgetreue Wiedergabe akustischer Signale

2 Ansätze:

V.2.1. (Makroskopischer Ansatz)

Erzeugung des Originalschallfeldes im gesamten Raum
Großer Aufwand:
→ Felder von Mikrophonen
→ Nachbilden der Schallfeldverläufe über Felder von Lautsprechern

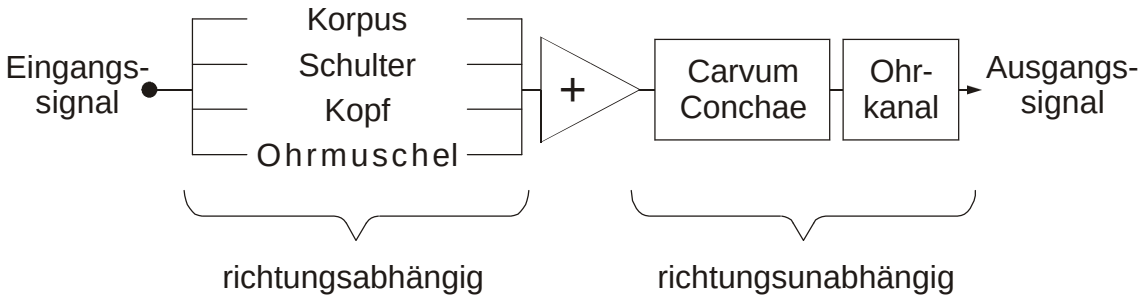
Praxis: Beschränkung auf Reproduktion der Intensitätsverläufe
Vorteil: Richtungsgetreue Wiedergabe für alle Personen im Raum

V.2.2. 2. Mikroskopischer Ansatz!

Reproduktion der Intensitäts- und Phasenverläufe des Originalsignals nur am Ohr des Hörers.

Außenohrübertragungsfunktion (HRTF)

Außenohr:
- Ohrmuschel } (5)
- Ohrmuschelhöhle
- Ohrkanal (6)
- Trommelfell (7)



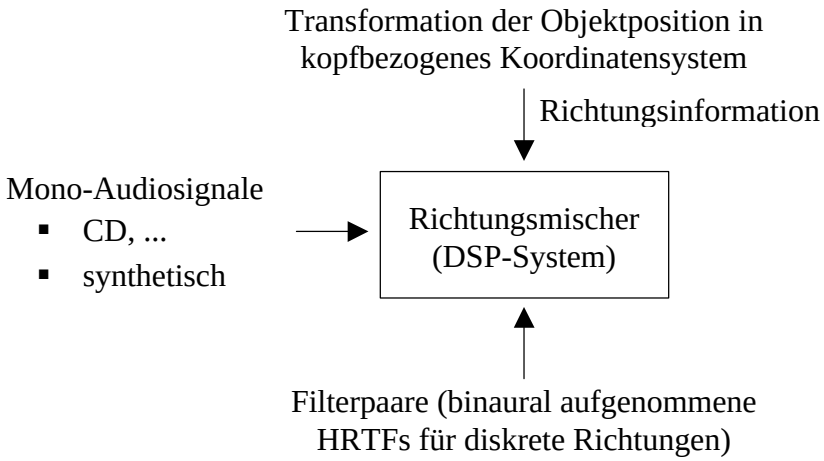
V.2.3. Messung der HRTF

Maximalfolge: periodisches, pseudostatistisches Rauschsignal
wichtig: Reflexionsfreier Raum

Elektret-Mikrofone
- in beide Ohrkanäle der Testperson → personenabhängig!
- Kunstkopfsystem

V.2.4. Wiedergabe der binauralen Aufnahmen

1. Kopfhörer
- einfach: Kanaltrennung schon gegeben
- Vorgehensweise: gegeben Position des virtuellen, geräuschverursachenden Objekts und der Kopfposition und Orientierung im Weltkoordinatensystem



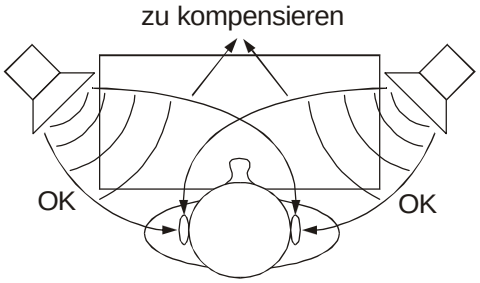
Probleme mit Kopfhörern:

- Tragekomfort
- Beeinträchtigung des Räumlichkeitsempfindens
  → Im-Kopf-Lokalisation

2. Über 2 Lautsprecher

Neue Probleme:

- Reflexionen
- Übersprechen der beiden Stereokanäle
  Lösung: Übersprechkompensation



V.2.5. Übersprechkompensation

2 Ansätze:

I. geschlossene Lösung
II. Iterative Lösung

→ Kopie

zu I:

Gesucht: Y_L und Y_R unter der Bedingung $Z_L = X_L$ und $Z_R = X_R$

Es gilt:

$$Z_L = Y_L \cdot L \cdot O_{LL} + Y_R \cdot R \cdot O_{LR} \overset{!}{=} X_L$$

$$Z_R = Y_R \cdot R \cdot O_{RR} + Y_L \cdot L \cdot O_{RL} \overset{!}{=} X_R$$

$$Y_L = L^{-1}\left[\frac{O_{RR}}{O_{LL}O_{RR} - O_{LR}O_{RL}} \cdot X_L - \ldots\right]$$

Theoretisch ideale Lösung, aber:
tiefe Frequenzen: $O_{LL} \approx O_{LR}$ und $O_{RR} \approx O_{RL}$

zu II.: Iteratives Kompensationsmodell

Abhörende Person: O_{XX}
Kunstkopf: AO_{XX}

1. Rausfiltern von Lautsprecher und HRTF der abhörenden Person

$$Z_R \overset{!}{=} X_R = X_R \cdot \underbrace{R^{-1} \cdot AO_{RR}^{-1}}_{\text{Filterblock RR}} \cdot R \cdot O_{RR}$$

2. 1. Iterationsschritt

Signal vom rechten Lautsprecher $Y_R = X_R \cdot R^{-1} \cdot AO_{RR}^{-1}$ trifft auf linkes Ohr

$$X_R \cdot \left[R^{-1} \cdot AO_{RR}^{-1}\right] \cdot R \cdot O_{LR} \underbrace{- X_R \cdot \overbrace{\left[K_{1R}\right]}^{\text{gesuchtes Filter}} \cdot L \cdot O_{LL}}_{\text{Kompensation}} \overset{!}{=} 0$$

$$\Rightarrow \quad K_{1R} = \underbrace{L^{-1} \cdot AO_{RR}^{-1} \cdot \left(\frac{O_{LR}}{O_{LL}}\right)}_{\text{Block RL}}$$

3. 2. Iterationsschritt

$$-X_R \cdot \left[L^{-1} \cdot AO_{RR}^{-1}\left(\frac{O_{LR}}{O_{LL}}\right)\right] \cdot L \cdot O_{RL} + X_R \cdot \left[K_{2R}\right] \cdot R \cdot O_{RR} \overset{!}{=} 0$$

$$\Rightarrow \quad K_{2R} = \underbrace{R^{-1} \cdot AO_{RR}^{-1}}_{\text{Wieder Block RR}} \cdot \underbrace{\frac{O_{LR} \cdot O_{RL}}{O_{LL} \cdot O_{RR}}}_{:= K}$$

4. 3. Iterationsschritt

$$X_R \cdot \left[R^{-1} \cdot AO_{RR}^{-1} \cdot K\right] \cdot R \cdot O_{LR} - X_R \cdot \left[K_{3R}\right] \cdot L \cdot O_{LL} \overset{!}{=} 0$$

$$\Rightarrow \quad K_{3R} \cdot \underbrace{L^{-1} \cdot AO_{RR}^{-1} \cdot \left(\frac{O_{LR}}{O_{LL}}\right)}_{\text{Block RL}} \cdot K$$

5. 4. Iterationsschritt

$$\Rightarrow \quad K_{4R} = R^{-1} \cdot AO_{RR}^{-1} \cdot K^2$$

$$\sum_{i=0}^{\infty} K^i = \frac{1}{1-K} \quad \text{mit} \quad K = \frac{O_{LR} \cdot O_{RL}}{O_{LL} \cdot O_{RR}}$$

$$\Rightarrow \quad ... = \frac{O_{LL} \cdot O_{RR}}{O_{LL} \cdot O_{RR} - O_{LR} \cdot O_{RL}}$$

Schlussbemerkungen:

- Kompensation nur gültig für feste Kopfposition
  → positionsabhängige Übersprechkompensation: aktuelle Forschung
- Hier gar nicht behandelt: Raumakustik
  Problem: Echtzeit

VI. Erfassung menschlicher Bewegungen

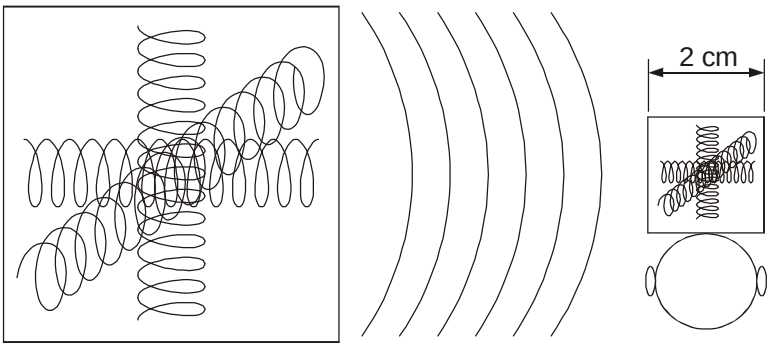
Messprinzipien: z. B. mechanisch, elektromagnetisch, optoelektronisch, Ultraschall, ...

Kriterien

- Messgröße: Position, Orientierung, Winkel, Beschleunigung, ...
- Dimension der Erfassung (VR → 3D) und Anzahl der erfassten Gelenke
- Messgenauigkeit
- Größe des Messraums
- Echtzeitfähigkeit, Automatismus
- Grad der „Rückwirkung“

VR:

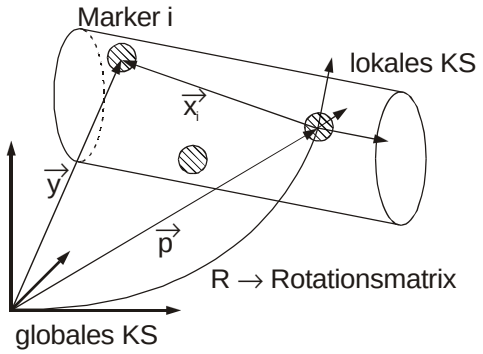
- Elektromagnetische Trackingsysteme
  + messen Position **und** Orientierung
  + keine Verdeckungsprobleme
  – Störanfälligkeit gegenüber ferromagnetischen Umgebungen
  – abnehmende Genauigkeit mit zunehmendem Abstand vom Sender
  – relativ große Sensoren



- Datenhandschuhe
  → Messung der Fingergelenkwinkel (Glasfaser, Dehnungsmessstreifen)
  → Sinnvoll nur in Kombination mit anderen Messtechniken
  + Anzahl erfasster Gelenke
  – Fehlerfortpflanzung
  – Rückwirkung
  – Problem: Daumenrotation (Greifbewegungen)

- Optoelektronische Systeme
  → Erfassung der 2D-Position von Markern über ≥ 2 Infrarotkameras
  - passiv: Licht reflektierende Marker
  - aktiv: Licht emittierende Marker
  sequentielle Ansteuerung → Identifikation trivial
  → Echtzeit
  → Ermitteln der 3D-Position aus den Kamerabildern, z.B. über direkte lineare Transformation
  + Genauigkeit und Abtastraten hoch
  + Geringe Rückwirkung
  – Verdeckung
  – Kalibrierungsaufwand
  – Keine Orientierung

Gesucht: Position und Orientierung der einzelnen Körpersegmente im 3D - Raum
Strategie: Positioniere ≥ 3 (nicht-kollineare) Marker „irgendwo“ auf dem Körpersegment
(Alternative: „Anatomic Landmarks“)
=> Referenzmessung notwendig!



Grundgleichung:
$\vec{y}_i = R\vec{x}_i + \vec{p}$
Ermitteln von x_i über Referenzmessung:
Positionierung des Segments so, dass Orientierung R_{ref} relativ zum globalen Koordinatensystem bekannt
$\vec{x}_i = R_{Ref}^{-1}\left(\vec{y}_{Ref} - \vec{p}\right)$

Gesucht: R und $\vec{p}$
Lösung: Methode der kleinsten Fehlerquadrate

Least Squares Method
Problem: $\vec{x}_i$, $\vec{y}_i$ sind ungenau
→ Ungenauigkeit des Messsystems
→ Hautverschiebungen
→ Referenzmessung

- Bestimmung unbekannter Parameter durch eine Reihe von Messungen
- Parameter nicht exakt bestimmbar, Ziel: Minimale Abweichung
- Restvektor $\vec{r} = R\vec{x}_i + \vec{p} - \vec{y}_i$
- Euklidische Vektornorm von $\vec{r}$ ist Maß für Fehler
  $\left\|\vec{r}\right\| = \sqrt{\sum_{i=1}^{3} r_i^2} = \sqrt{\vec{r}^T\vec{r}}$

- Minimiere die Summe der Quadrate der Euklidischen Vektornorm der Restvektoren aller Marker eines Segments
  $n := \text{Anzahl Marker}, n \geq 3$
  $$\text{Min}\left\{\frac{1}{n}\sum_{i=1}^{n}\vec{r}^{T}\vec{r}\right\} = \text{Min}\left\{\frac{1}{n}\sum_{i=1}^{n}\left(R\vec{x}_i + \vec{p} - \vec{y}_i\right)^T\left(R\vec{x}_i + \vec{p} - \vec{y}_i\right)\right\}$$

Algorithmus für Bestimmung von R

1. $\vec{x}_m = \frac{1}{n}\sum_{i=1}^{n}\vec{x}_i\ ,\quad \vec{y}_m = \frac{1}{n}\sum_{i=1}^{n}\vec{y}_i$
2. $x = \left(\left(\vec{x}_1 - \vec{x}_m\right), \left(\vec{x}_2 - \vec{x}_m\right), \ldots, \left(\vec{x}_n - \vec{x}_m\right)\right)$
   $y = \left(\left(\vec{y}_1 - \vec{y}_m\right), \left(\vec{y}_2 - \vec{y}_m\right), \ldots, \left(\vec{y}_n - \vec{y}_m\right)\right)$
3. $C = Y \cdot X^T$ (Korrelationsmatrix)
4. $C = U \cdot W \cdot V^T$ (Singulärwertzerlegung)
   $R = U \cdot V^T$

   $\vec{p} = \vec{y}_m - R\vec{x}_m$

VII. Kollisionen (Collision Detection, CD)

Motivation: Verbesserung der 3D-Interaktion in virtuellen Umgebungen

- **Objekt–Objekt–Interaktion**
  → Realistisches Verhalten virtueller Gegenstände
  → Objekte dürfen sich nicht durchdringen
  → Physikbasierte Modellierung
- **Benutzer–Objekt–Interaktion**
  → Direkte Interaktion im 3D-Raum
  → Insbesondere: Haptisches Feedback

VII.1. Anwendungen:

- Montagesimulation, Einbauuntersuchungen
- Architectural walkthrough, Fahrsimulation
- Roboterprogrammierung
- ...

VII.2. Fundamentaler Algorithmus

Applikation
- Gegeben:
- N Objekte $O_0, \ldots, O_{N-1}$
- Simulationsintervall $[t_0, t_1]$
- Δt_r Rendering-Zeitschritt

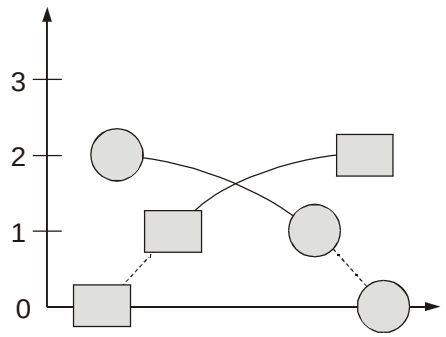
CD-Algorithmus: Δt_d Detection-Zeitschritt

mit $\Delta t_r = m \cdot \Delta t_d;\ m \in \aleph$ (u. U. $m = 1 \Rightarrow \Delta t_r = \Delta t_d$)

VII.2.1. Schwächen des Algorithmus:

- **Fixed timestep** (Δt_d, Zeile 17)

  Kollisionen zwischen Zeitschritten werden nicht erkannt



Δt_d klein: exakt
Δt_d groß: effizienter
} => Δt_d adaptiv!

Anpassen an die Wahrscheinlichkeit einer Kollision

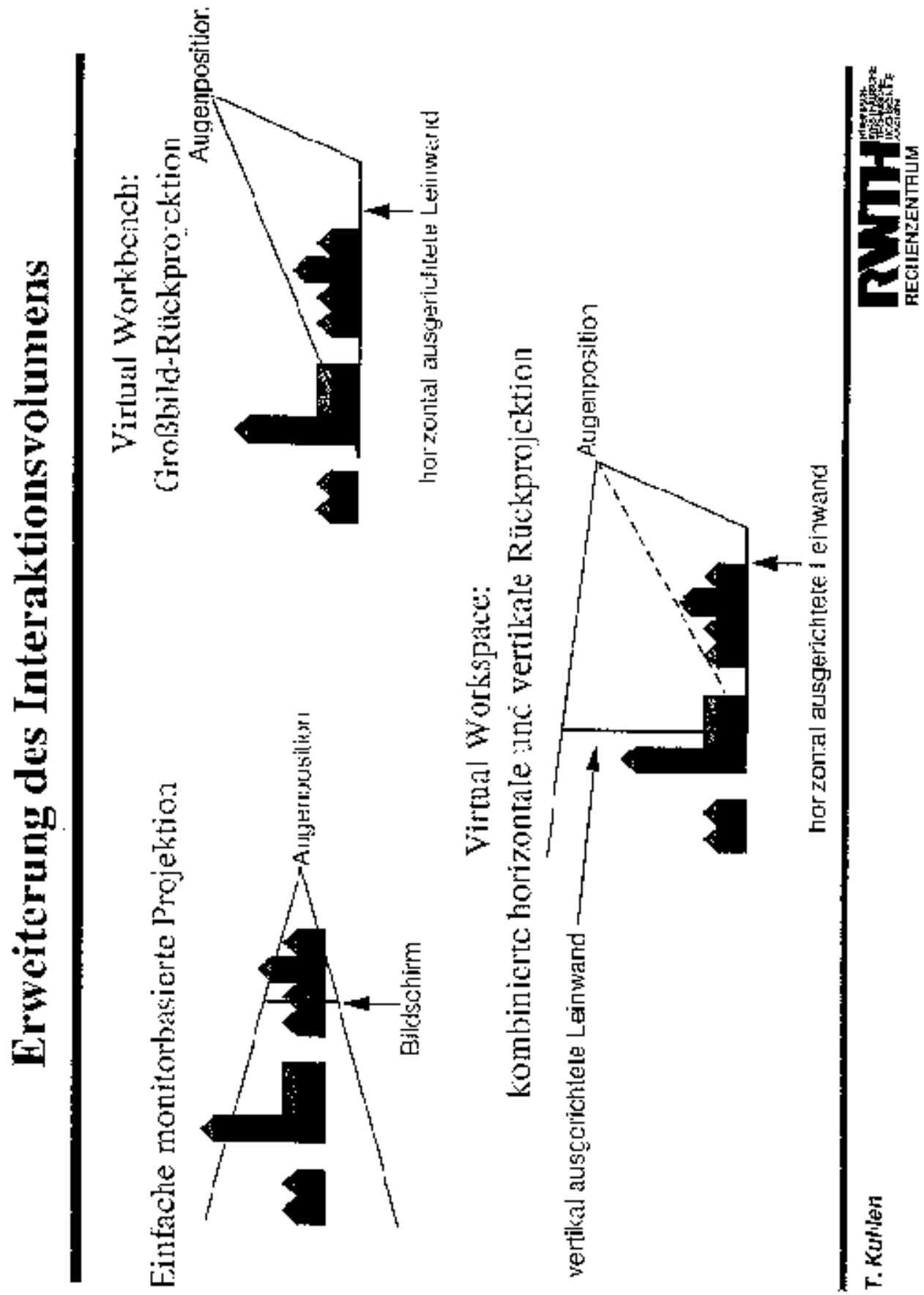
- **all pairs** (Zeile 19 ff)
  Aufwand: $O(N^2)$ (Verarbeitungszeit steigt quadratisch an!)
  => Problem für N >> 1
  besser: Weit entfernte Objekte gar nicht auf Kollision testen

- **pair processing** (Zeile 21 ff)
  Intersection Test aufwendig für komplexe Geometrien (VR: große Anzahl Polygone)

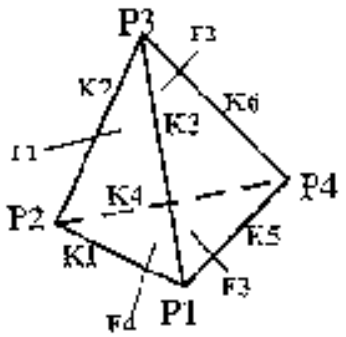
VII.2.2. Beispiele für Verbesserungen des fundamentalen Algorithmus

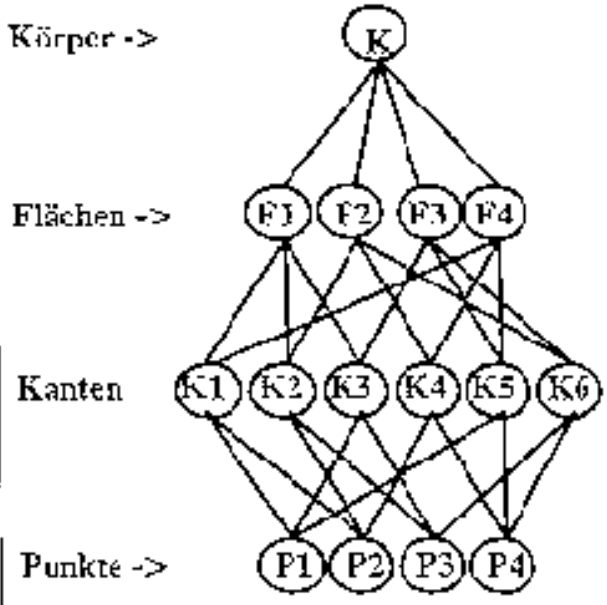
- **3D Subdivision Methods**
  Idee: - Unterteilung des 3D-Raums in Regionen
  - 2 Objekte können nur in derselben Region kollidieren
  Varianten: - Uniform Grid
  - Octrees
  - Binary Space Partitioning
- **Approximative Collision Detection**
  Idee: Nicht die Geometrien, sondern Approximationen der Geometrien werden auf Kollisionen getestet. Approximation muss konservativ gewählt werden:
  Kollision der Geometrien $\Rightarrow$ Kollision der Approximation
  $\not\Leftarrow$
  Intersection Test: nicht ja/nein
  $\rightarrow$ vielleicht / nein
- **Unterbrechbare Collision Detection**
  Idee: Basis ist approximative Collision Detection, aber: Unterstützung mehrerer Exaktheitsstufen „solange noch Zeit“
  Verfeinerte Approximation für „Vielleicht-Objekte“
  Erneuter Intersection Test
  Interpretiere noch vorhandene „Vielleicht“-Antworten als „ja“
- **Zeitkritische Collision Detection**
  Idee: Basis ist unterbrechbare Collision Detection, aber Vorhersage möglicher Kollisionen.
  2 Phasen:
  1. Broad Phase: - Obere Schranken für die Beschleunigung aller Objekte
     - Approximation (z.B. Bounding Spheres)
  2. Narrow Phase: - Beginnt, falls Broad-Phase eine „vielleicht-Antwort“ liefert
     - identisch mit unterbrechbarer Collission Detection

VIII. Anhang: Kopien



Polygonale Darstellung:
Explizites Speichern von Kanten





Beispiel: Tetraeder

Raumunterteilende Techniken

Aufbau eines Octree

- Die Quader mit der Markierung "partiell" werden in Oktanten unterteilt und es werden Bögen vom unterteilten Quader zu jedem Oktanten gezogen.

- Die Oktanten werden markiert und gegebenenfalls weiter unterteilt.

- Der Prozeß wird so lange fortgesetzt, bis alle Oktanten entweder mit "weiß" oder "schwarz" markiert sind oder eine gegebene Rekursionstiefe erreicht ist.

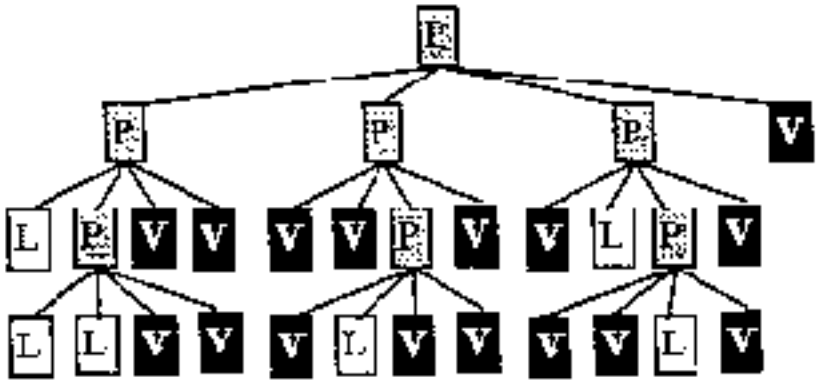
Numerierung der Quadranten

2	3
0	1

L leer

P partiell

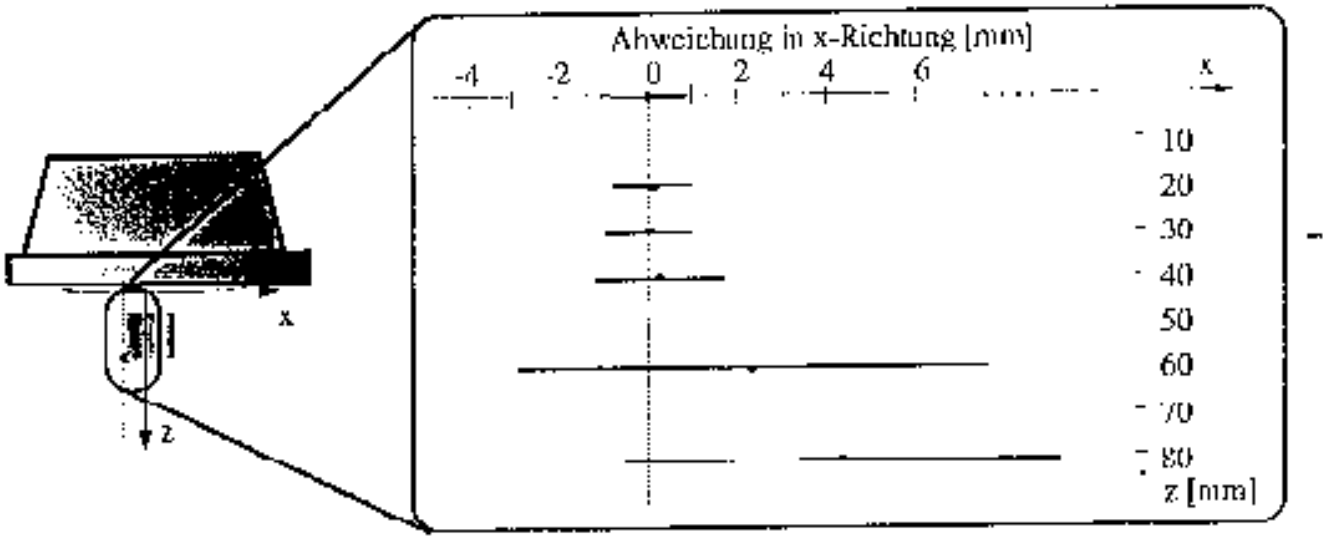
V voll



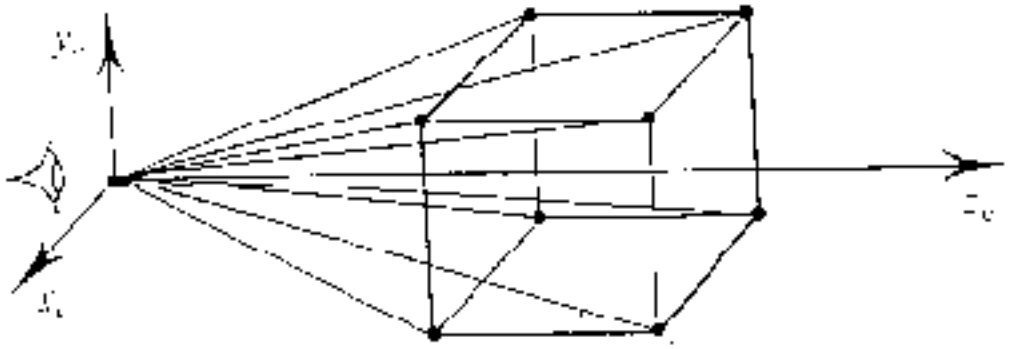
Konstruktion eines Quadtrees am Beispiel

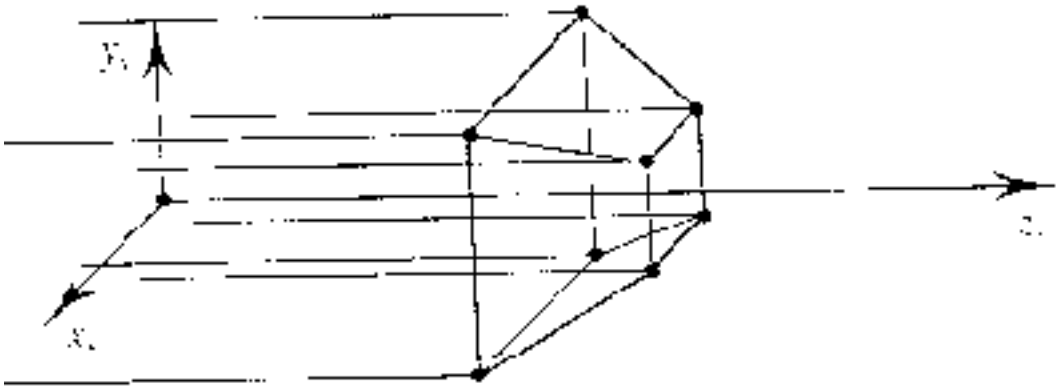
Wahrnehmung betrachterzentriert visualisierter Objekte

Abweichung der wahrgenommenen x-Position von der Sollposition
für verschiedene Abstände z zwischen Objekt und Abbildungsebene

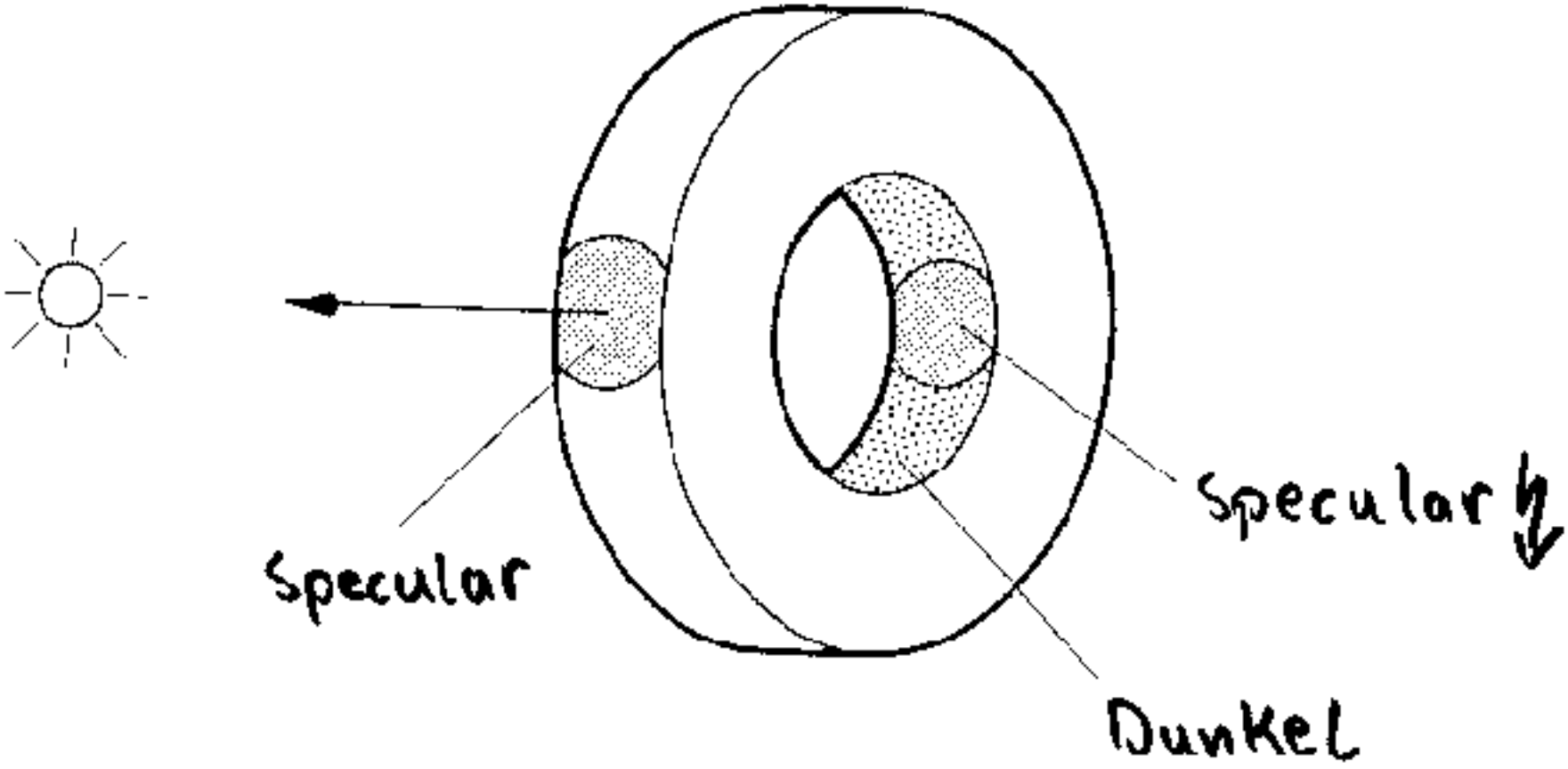


z-Buffering bei Parallelprojektion

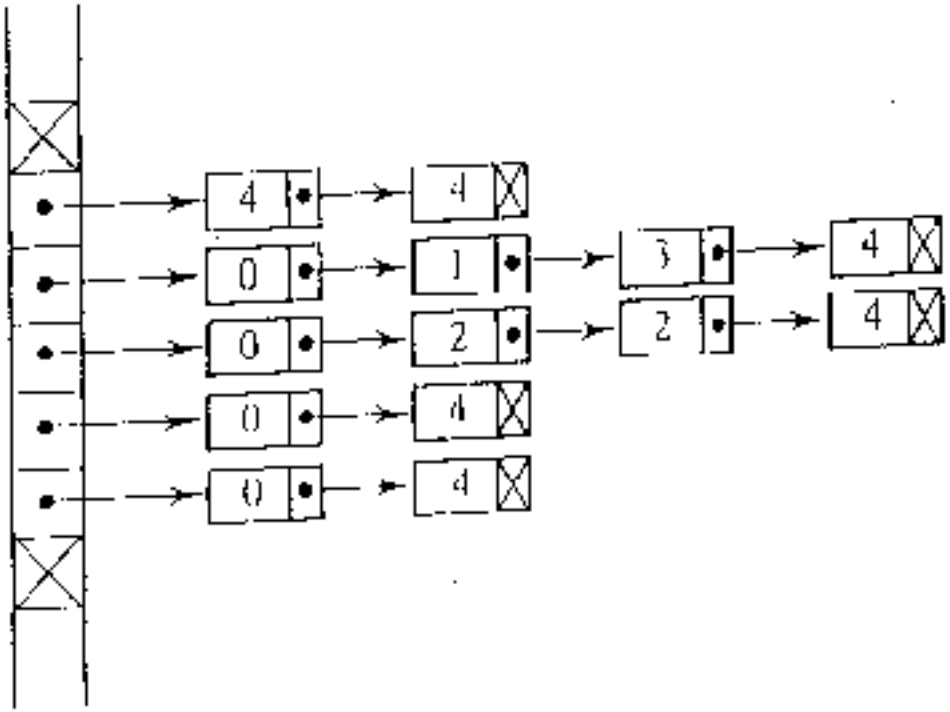


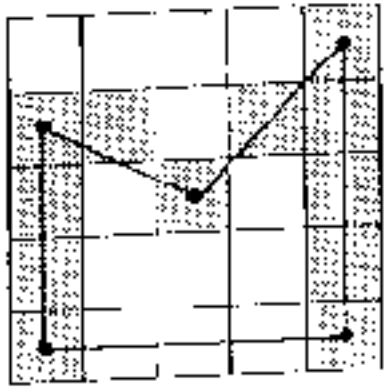


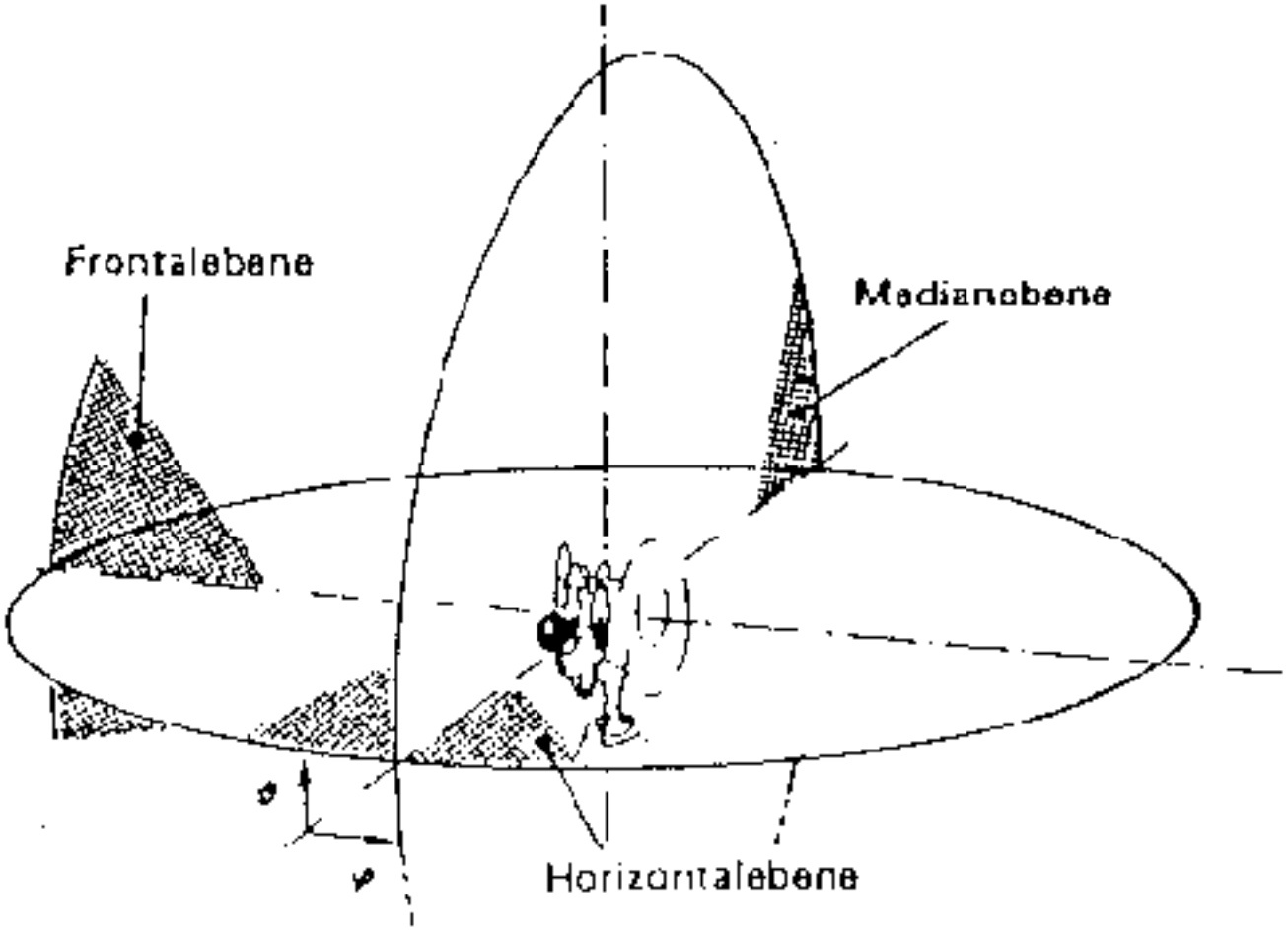
Fehler des Phong-Modells



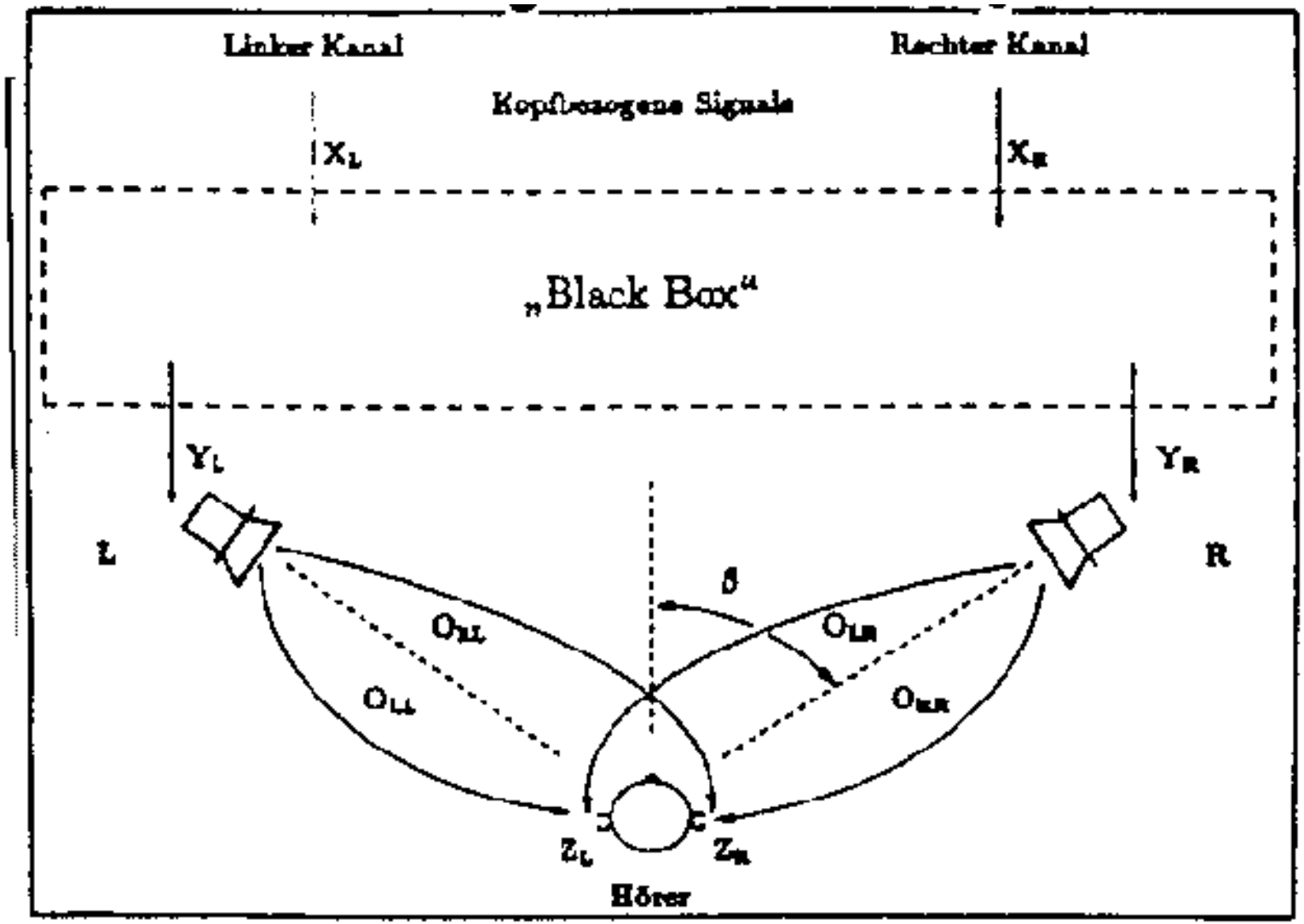
Kantenliste zur Rasterisierung von Polygonen



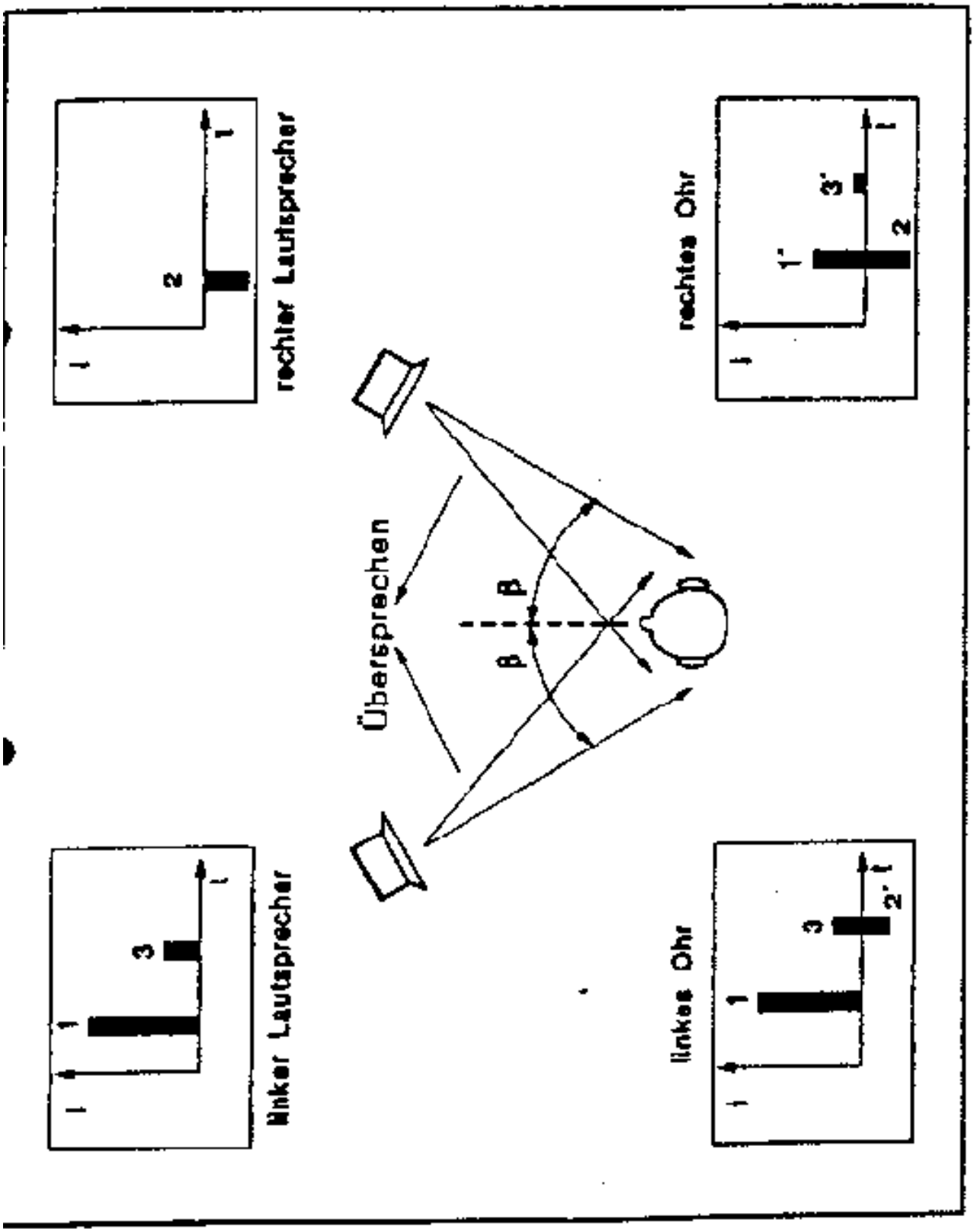


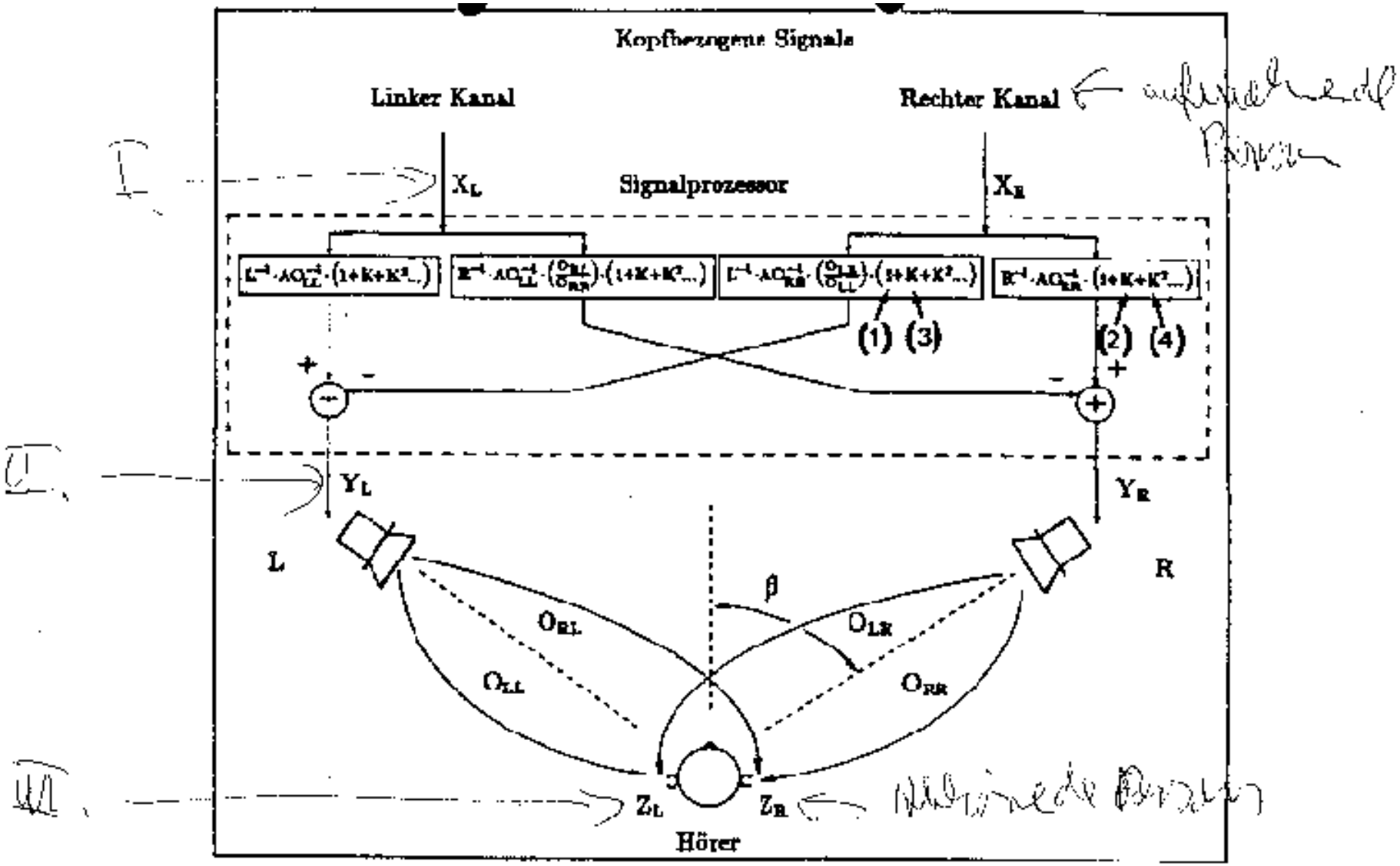


Kopfbezogenes Koordinatensystem



Schema einer Wiedergabe binauraler Signale mit Lautsprechern


rechter Lautsprecher
linker Lautsprecher
Übersprechen
β
β
rechtes Ohr
linkes Ohr
1
2
3
1'
2'
3'
t



Übersprechkompensation mit unendlich fortgesetzter Iteration

```
1   application()
2   {
3       for t ← t_0 to t_1 in steps of Δt_v {
4           get user input
5           update behavior of each object in {O_0, ...O_{N-1}} as of t
6           update any other general bookkeeping
7           do {
8               result ← detect(t, {O_0, ..., O_{N-1}})
9               if (result contains collisions)
10                  collision response
11          } while (result contains collisions)
12          render each object in {O_0, ...O_{N-1}} as of t
13      }
14  }

    /* The variable t_prev persists between calls. */
    /* Before first call, t_prev is initialized to t_0 */
15  detect(t_curr, {O_0, ..., O_{N-1}})
16  {
17      for t ← t_prev to t_curr in steps of Δt_d {
18          move each object in {O_0, ..., O_{N-1}} to its position as of t
19          for each object O_i ∈ {O_0, ..., O_{N-1}} {
20              for each object O_j ∈ {O_i, ..., O_{N-1}}
21                  if (pair-processing algorithm says O_i, O_j intersect)
22                      add O_i, O_j to collisions
23          }
24          if (collisions exist) {
25              t_prev ← t
26              return (collisions, t)
27          }
28      }
29      t_prev ← t_curr
30      return (no collisions, t_curr)
31  }
```

Figure 2.1: A basic detection algorithm for a simple application.

```
1   application()
2   {
3       for t ← t_0 to t_1 in steps of Δt_r {
4           get user input
5           update behavior of each object in {O_0, ... O_{N-1}} as of t
6           update any other general bookkeeping
7           do {
8               result ← detect(t, {O_0, ..., O_{N-1}})
9               while ((result is "maybe") and
10                      (more processing time is available))
11                  /* detect(t, result) knows to refine result. */
12                  result ← detect(t, result)
13              if (result is "maybe")
14                  collision response
15          } while (result is "maybe")
16          render each object in {O_0, ... O_{N-1}} as of t
17      }
18  }
```

Figure 3.2 A simple application that calls a time-critical detection algorithm, detect()