UNIVERSITÄT
LEIPZIG

Einleitung

COMPUTERGRAFIK

Inhaltsverzeichnis

1. Einleitung
2. Hardwaregrundlagen
3. Rasterung
4. Polygonale Repräsentation
5. Transformationen
6. Projektionen
7. Viewport und Clipping
8. Sichtbarkeit
9. Beleuchtungsmodelle
10. Texturen

Literatur

- Tom McReynolds, David Blythe
  Advanced Graphics Programming Using OpenGL
  Morgan Kaufmann, 2005

- Jose Encarnacao, Wolfgang Straßer, Reinhard Klein
  Graphische Datenverarbeitung I + II
  Oldenbourg, 2Bd.,1996

- Michael Bender, Manfred Brill
  Computergrafik
  Hanser 2003.

- James D. Foley, Andries van Dam, Steven K. Feiner, John F. Hughes
  Computer Graphics: Principles and Practice
  Addison-Wesley, 1990

- Alan Watt, Mark Watt
  Advanced Animation and Rendering Techniques
  Addison Wesley, 1992

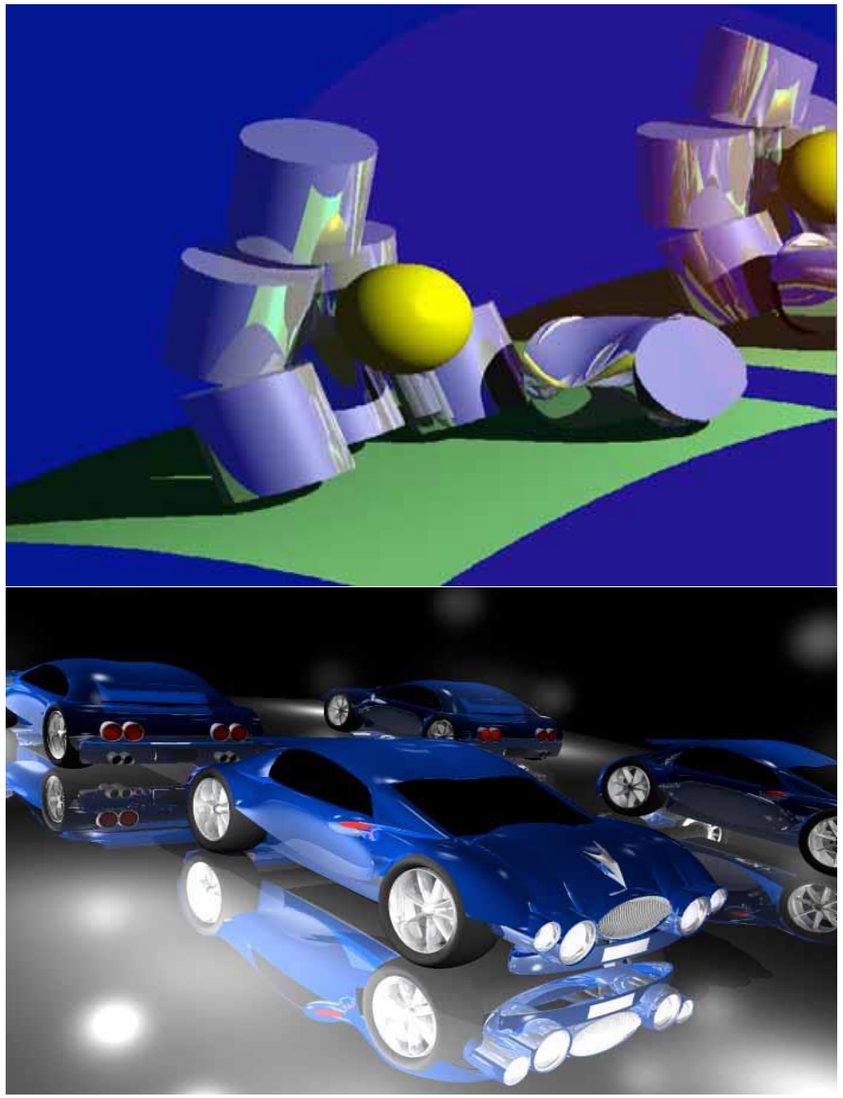
1.1 Was ist Computergrafik?

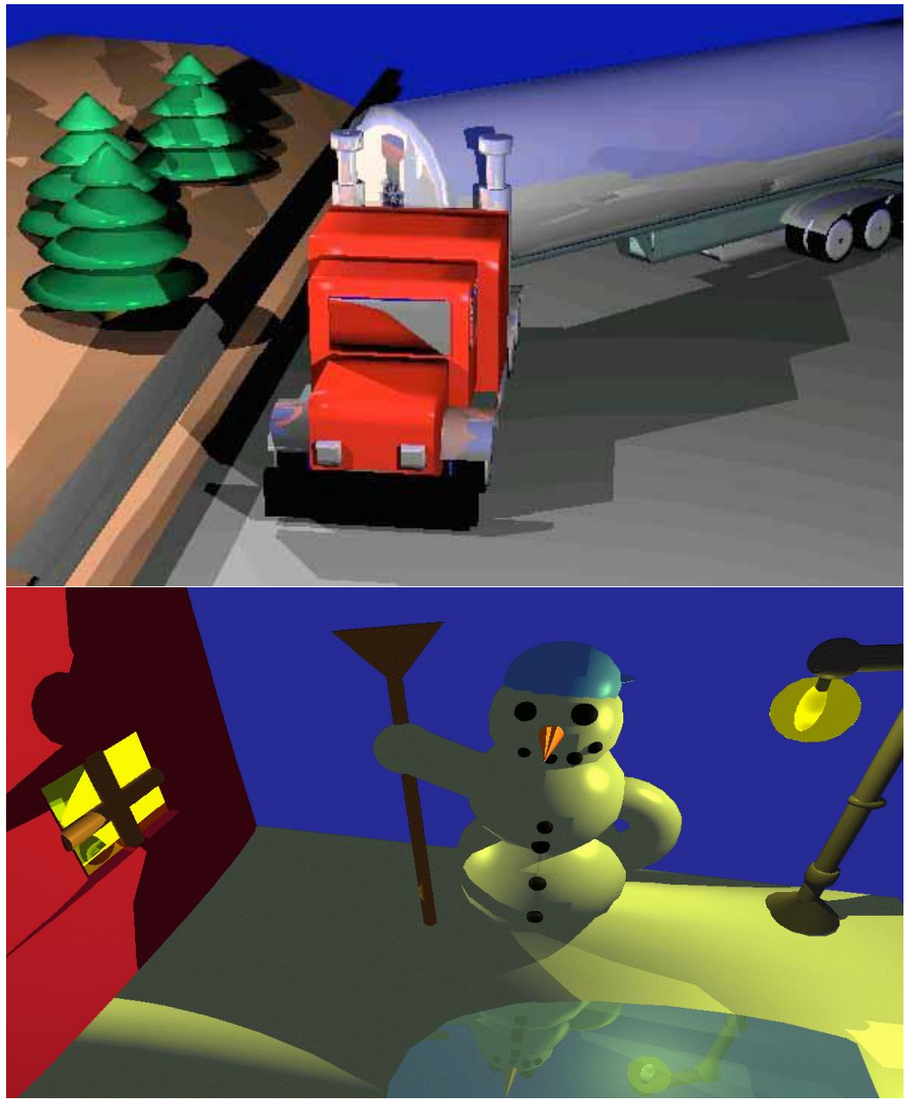
– Die Computergrafik ist ein Teilgebiet der Informatik, das sich mit der computergestützten Erzeugung und Darstellung von Bildern befasst.

1.1 Was ist Computergrafik?

– Real oder Virtuell?

1.1 Was ist Computergrafik?



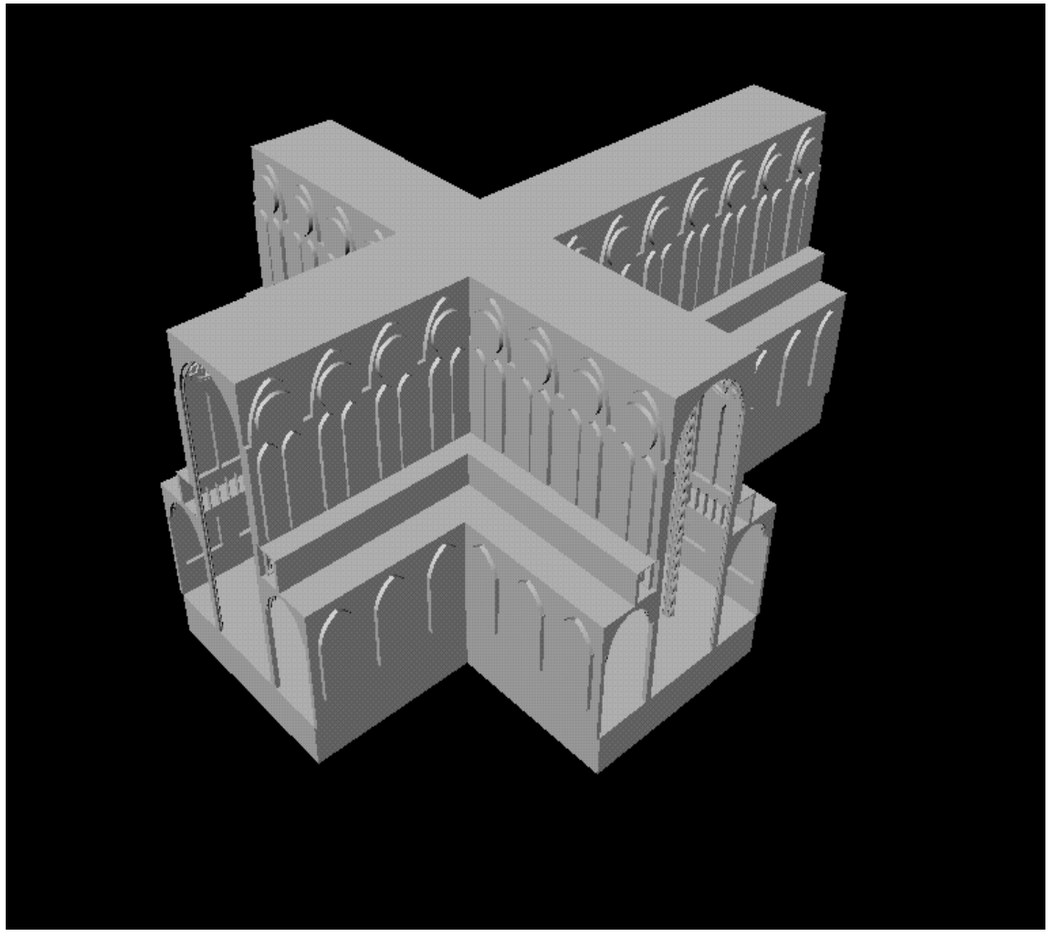


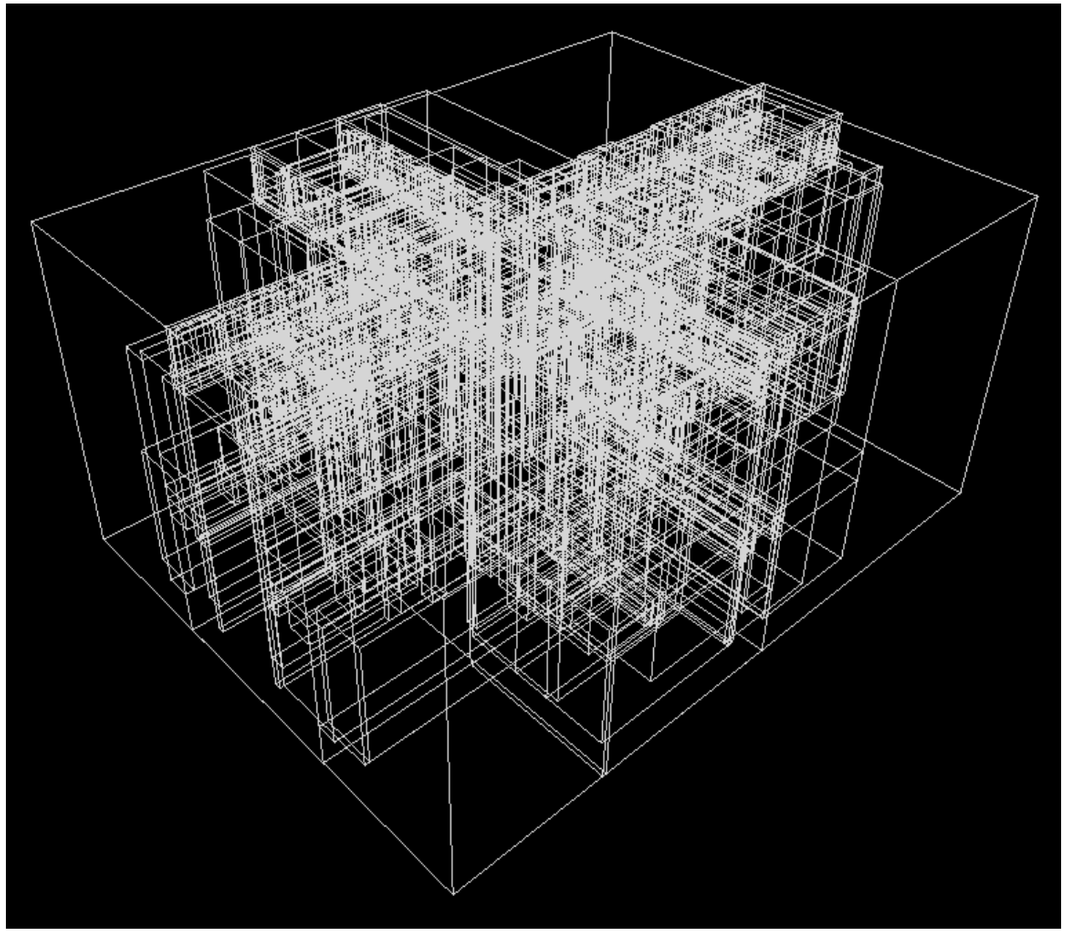
1.1 Was ist Computergrafik?

– Landschaftsgeneratoren

1.1 Was ist Computergrafik?

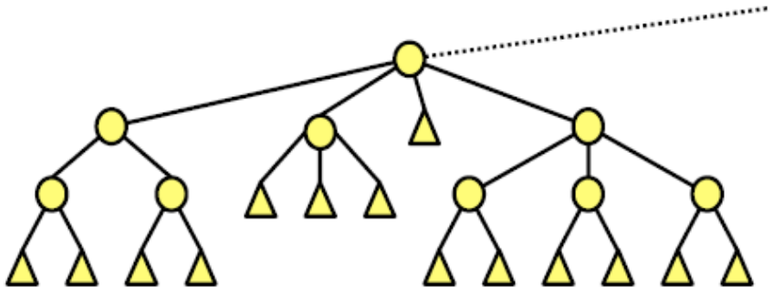
– Szenenrepräsentation

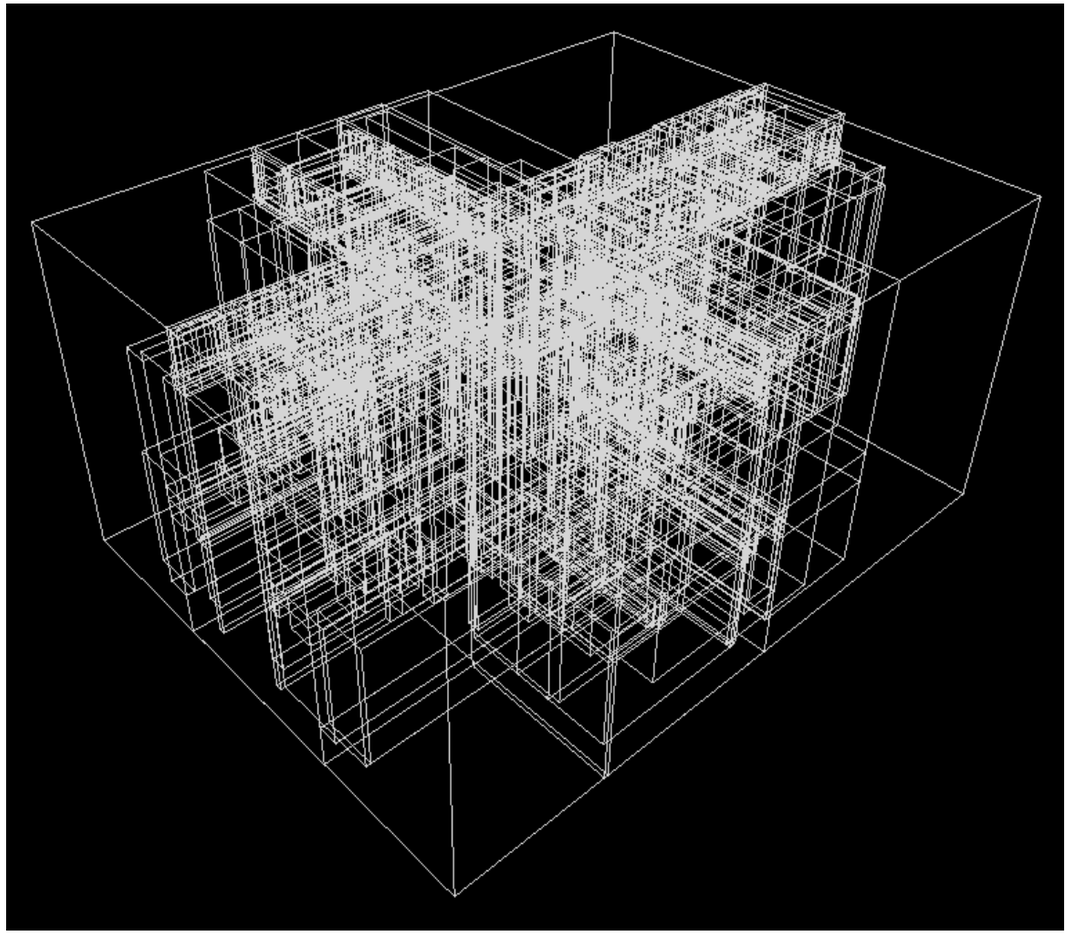




1.1 Was ist Computergrafik?

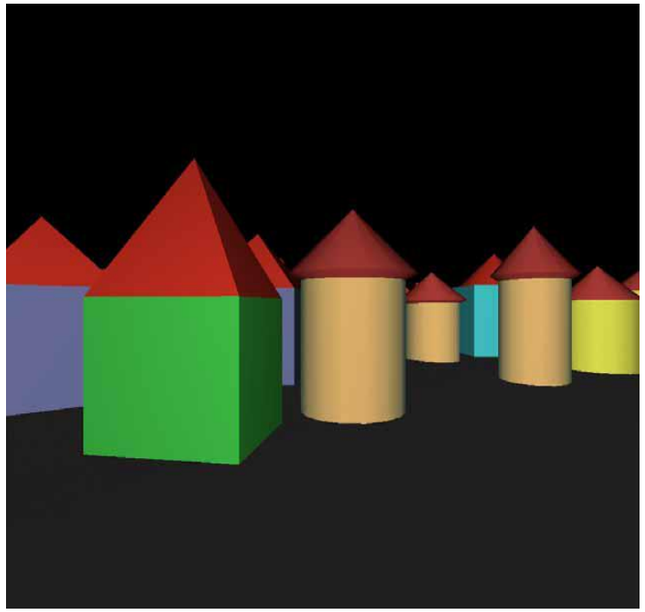
– Szenenrepräsentation

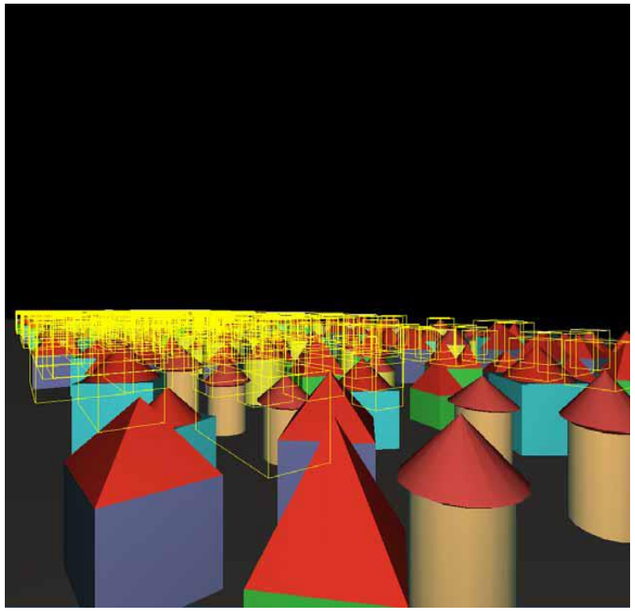


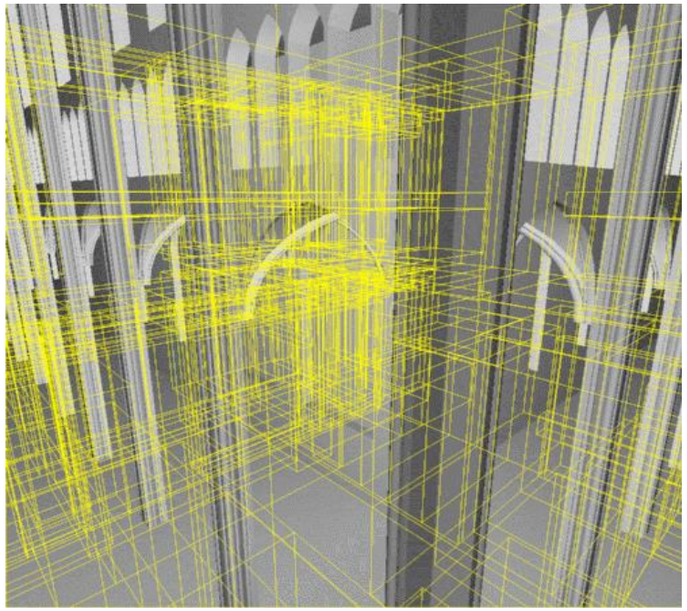


1.1 Was ist Computergrafik?

– Verdeckungsrechnung

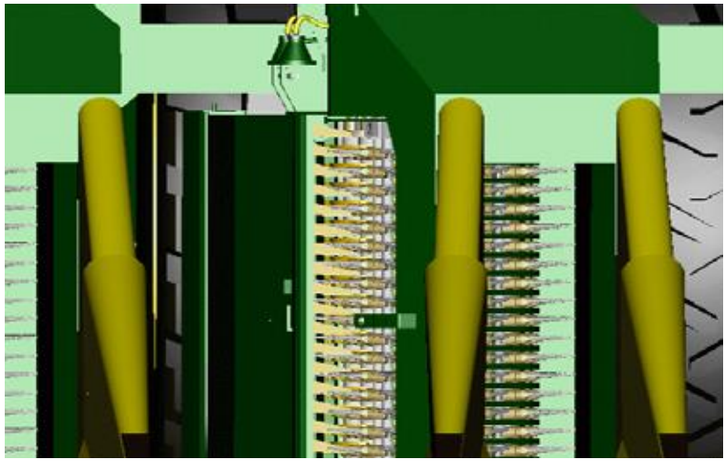






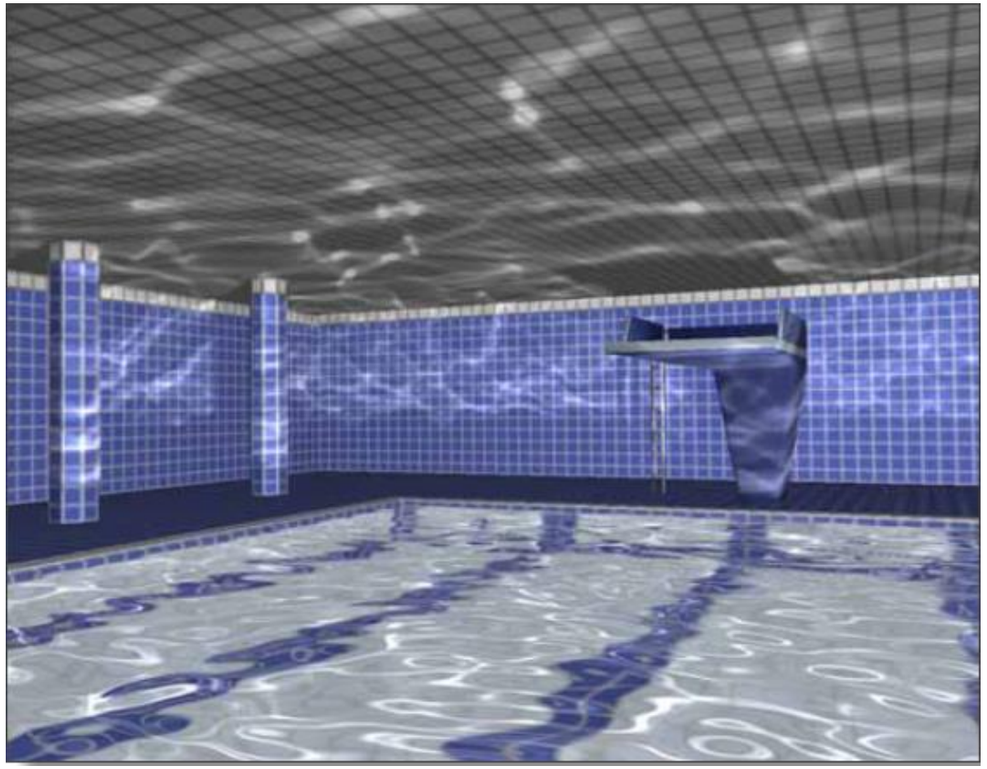
1.1 Was ist Computergrafik?

– Level-of-Detail:
  Von ca. 10,5 Millionen Dreiecken werden nur 550.000 dargestellt.



1.1 Was ist Computergrafik?

– Visuelle Simulation



1.1 Was ist Computergrafik?

– Bildbasiertes Rendern

1.1 Was ist Computergrafik?

– Globale Beleuchtung - Raytracing

1.1 Was ist Computergrafik?

– Rekonstruktion

UNIVERSITÄT LEIPZIG

Hardwaregrundlagen

COMPUTERGRAPHIK

Inhaltsverzeichnis

2 Hardwaregrundlagen

2.1 Bildschirmtechnologien

2.2 Projektionstechnologien

2.3 3D-Displaytechnologien

2.4 Bildrechner (GPU)

2.5 Weitere Ausgabegeräte

2.6 Graphische Eingabegeräte

2.1 Bildschirmtechnologien

Flüssigkristallbildschirm (LCD = Liquid Crystal Display)

- Flüssigkeitskristalle seit 1888 (F. Reinitzer)
- LCD-Bildschirme seit 1970

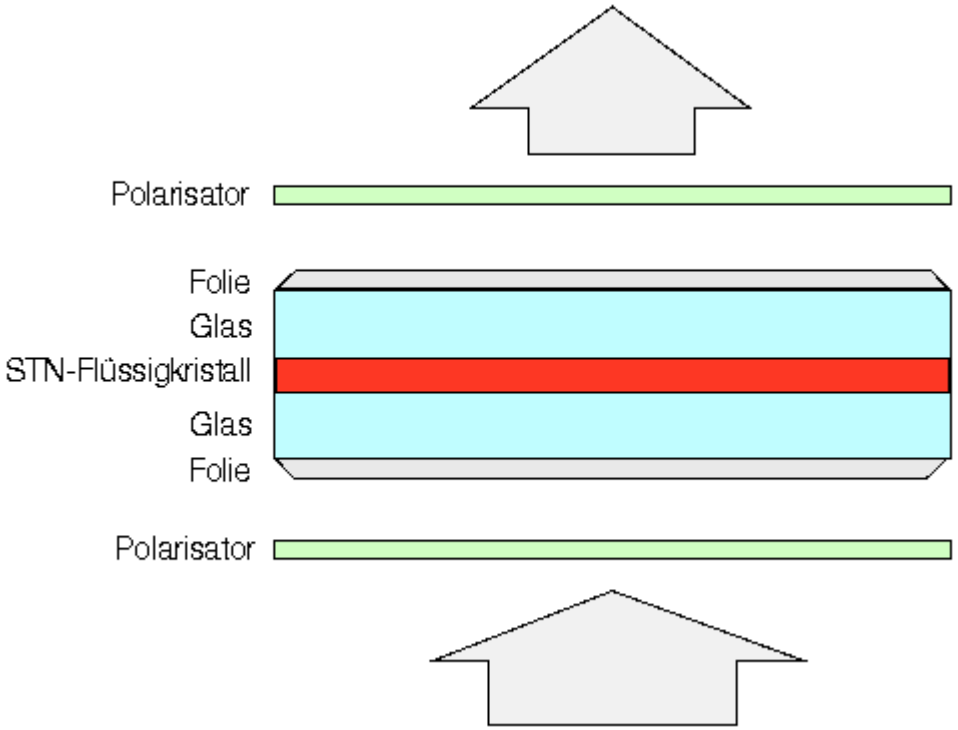
- Vorteile
  - Geringe Leistungsaufnahme (~25W)
  - Flimmerfrei
  - Guter Kontrast
  - Digital
  - Leicht/klein/mobil

- Nachteile
  - Passiv: Licht durchlassen oder reflektieren - erfordert Lichtquelle
  - Einschränkung Betrachtungswinkel
  - Kein echtes Schwarz

2.1 Bildschirmtechnologien

Flüssigkristallbildschirm (LCD = Liquid Crystal Display)

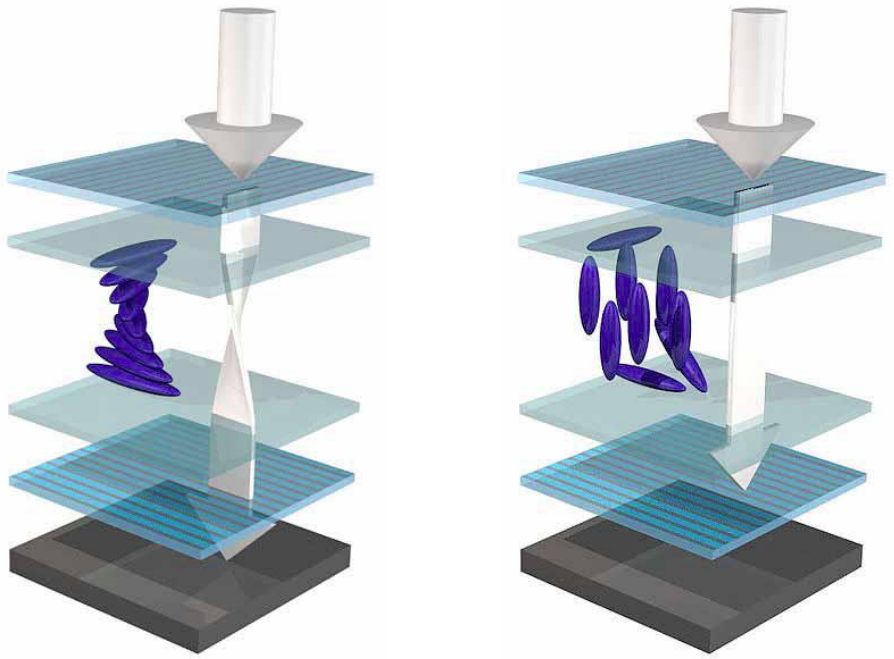
- Zwei parallele Glasplatten im Abstand 5-10µm
- Dazwischen: hermetisch abgeschlossener Flüssigkeitskristall
- An beiden Glasplatten liegen transparente Elektroden an:
  - Erzeugung unterschiedlicher elektrischer Felder in verschiedenen Bildschirmelementen
- Hintergrundbeleuchtung



2.1 Bildschirmtechnologien

Flüssigkristallbildschirm (LCD = Liquid Crystal Display)

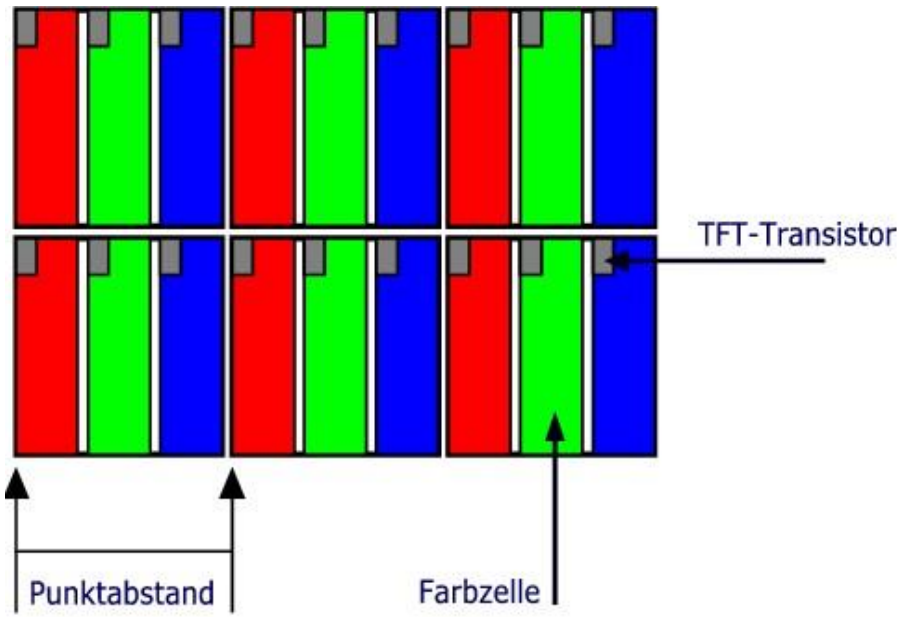
- Licht wird durch den 1. Polarisationsfilter polarisiert
- Licht wird durch die organischen Moleküle des Flüssigkeitskristalls gedreht
- Zweiter Polarisationsfilter
  - Licht passiert, wenn keine Spannung anliegt
  - Licht wird geblockt, wenn Spannung anliegt (Bild erscheint an dieser Stelle entsprechend dunkler, aber nicht Schwarz)



2.1 Bildschirmtechnologien

Flüssigkristallbildschirm (LCD = Liquid Crystal Display)

- Jedem Schalttransistor ist eine Flüssigkeitskristallzelle zugeordnet
- Für Farbdarstellungen werden drei Flüssigkeitskristallzellen zu einem RGB-Farbtripel zusammengefasst
- Die einzelnen Zellen erhalten durch Mikrofilter die Farben Rot, Grün und Blau

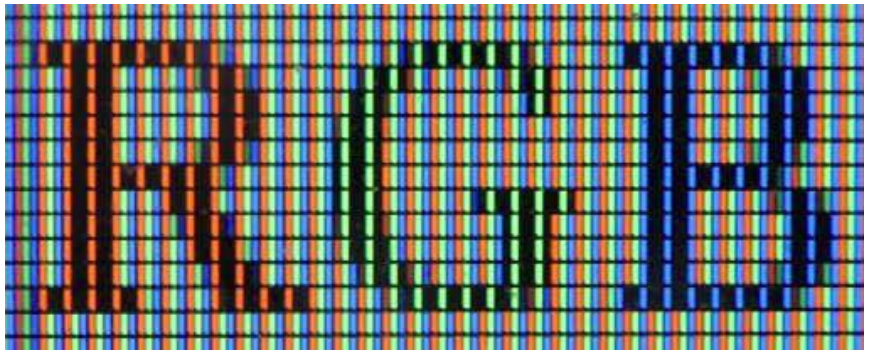


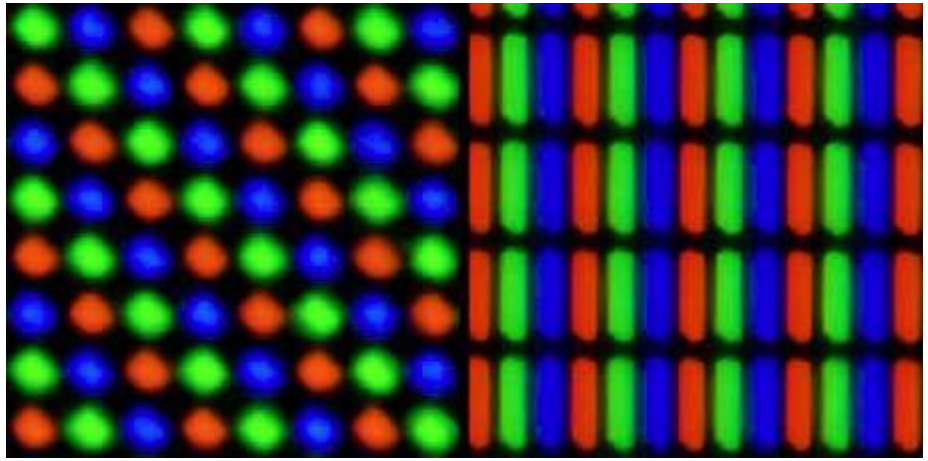
Aufbau der Bildpunkte eines TFT‘s

2.1 Bildschirmtechnologien

Flüssigkristallbildschirm (LCD = Liquid Crystal Display)

- Unterschiedliche Zellgeometrie für Anordnung der drei Farben
- Kann ggf. für Antialiasing (Subpixel-Rendering) ausgenutzt werden





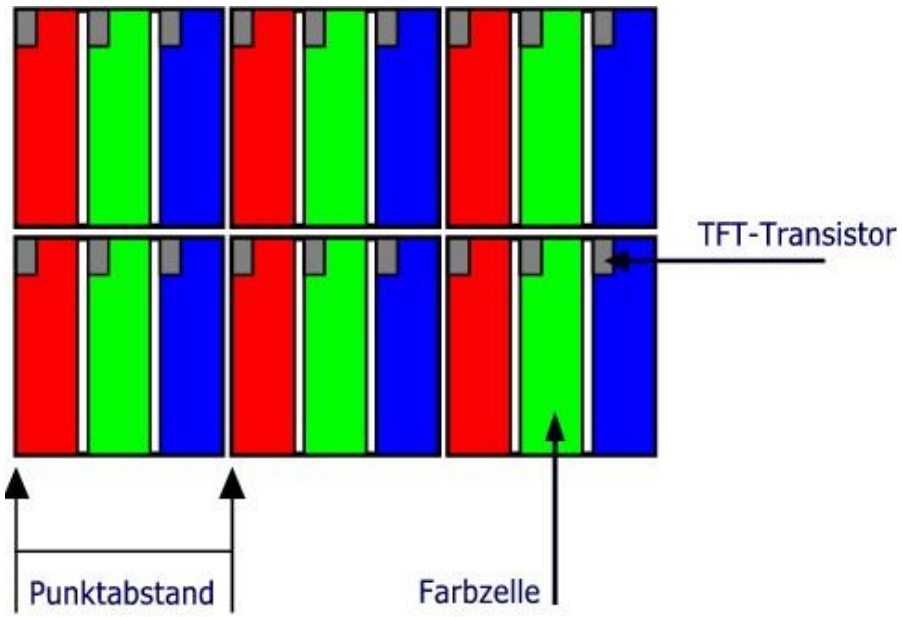
Delta-TFT, Vertical-Stripe-TFT

2.1 Bildschirmtechnologien

Flüssigkristallbildschirm (LCD = Liquid Crystal Display)

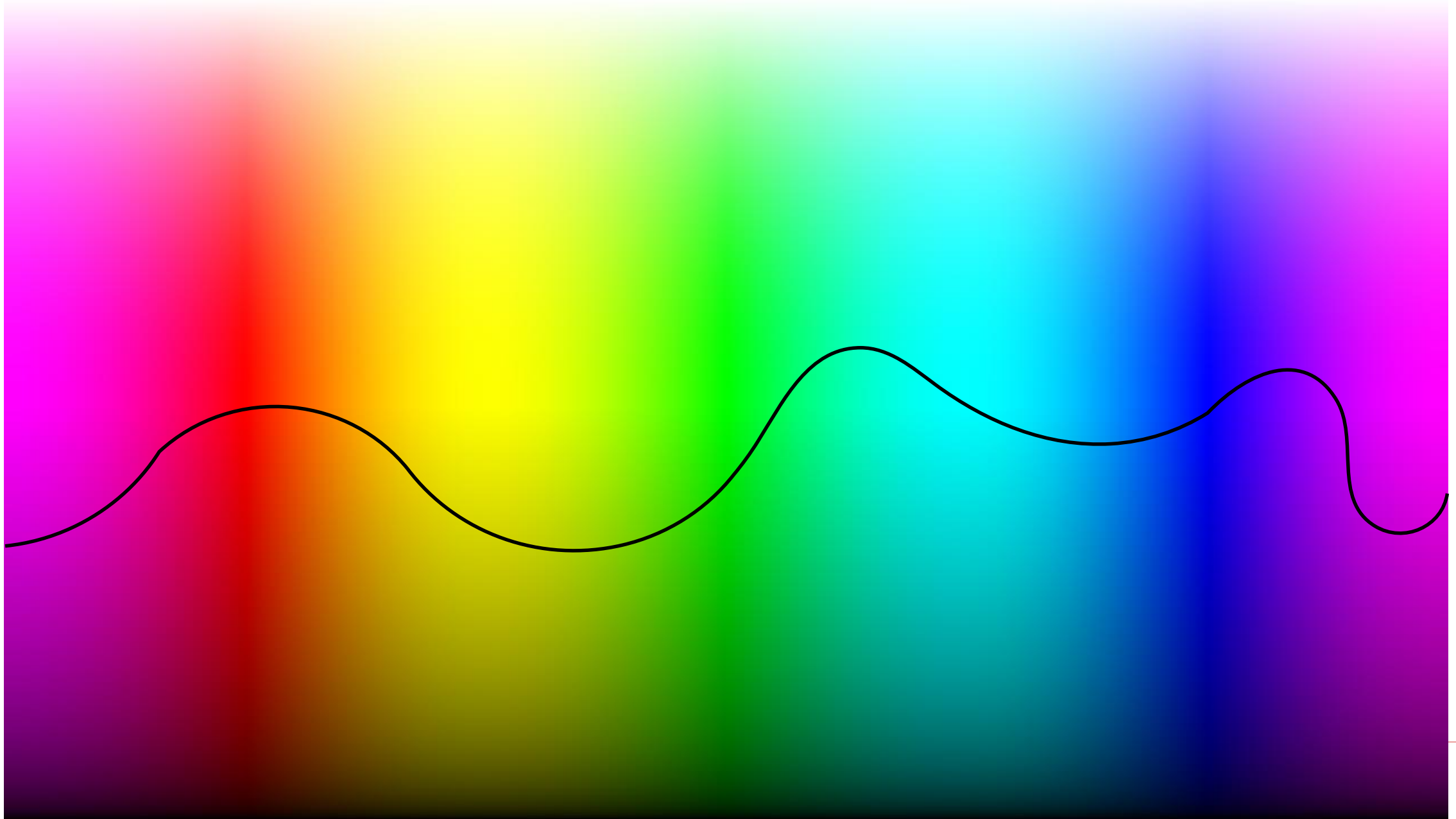
- Dünnfilmtransistoren (TFT - Thin-Film-Transistor) zur Ansteuerung der Elektroden

- Active-Matrix-Display
  - TFTs sind in einer matrixartigen Anordnung auf den Glasplatten angebracht

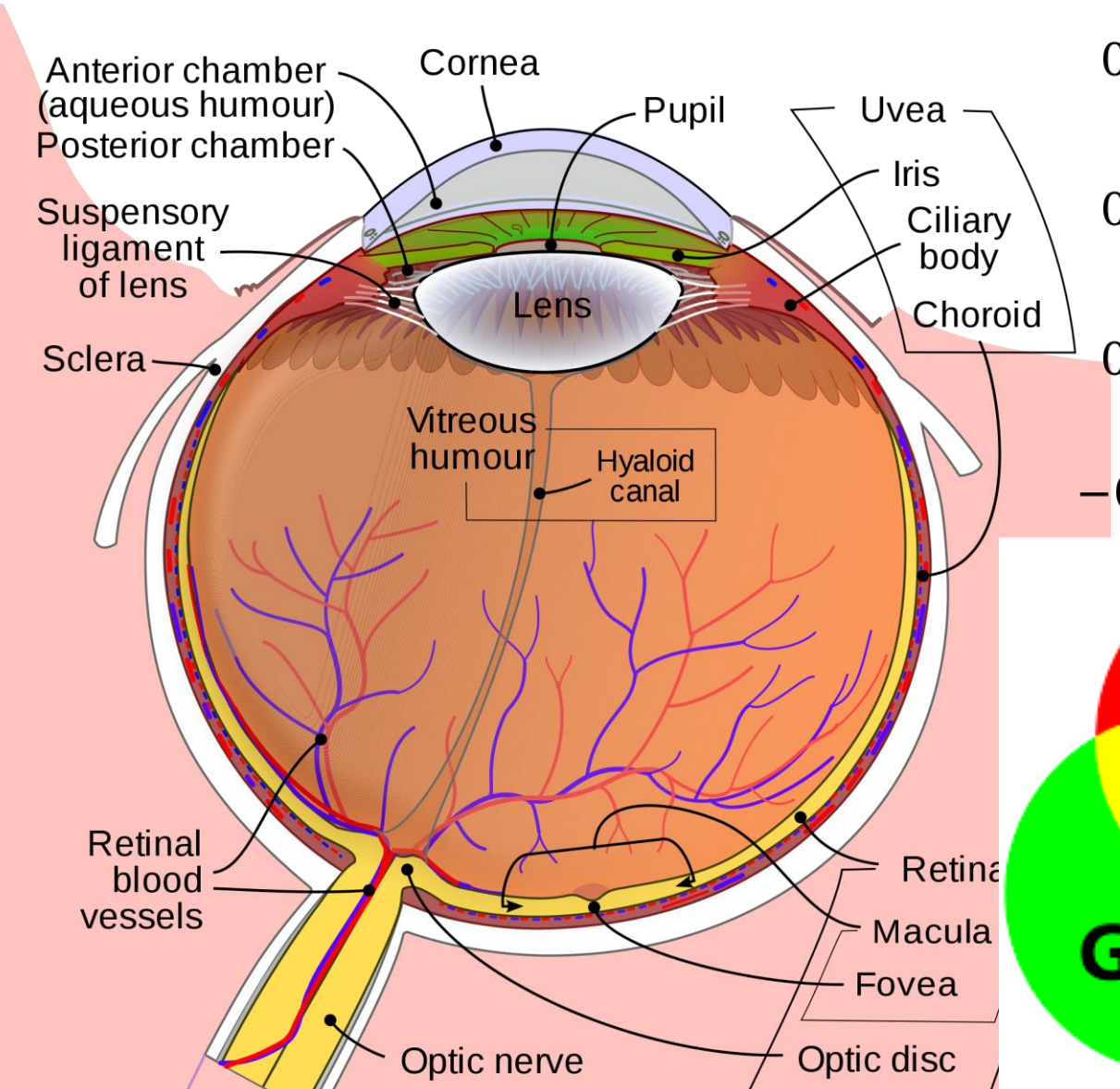


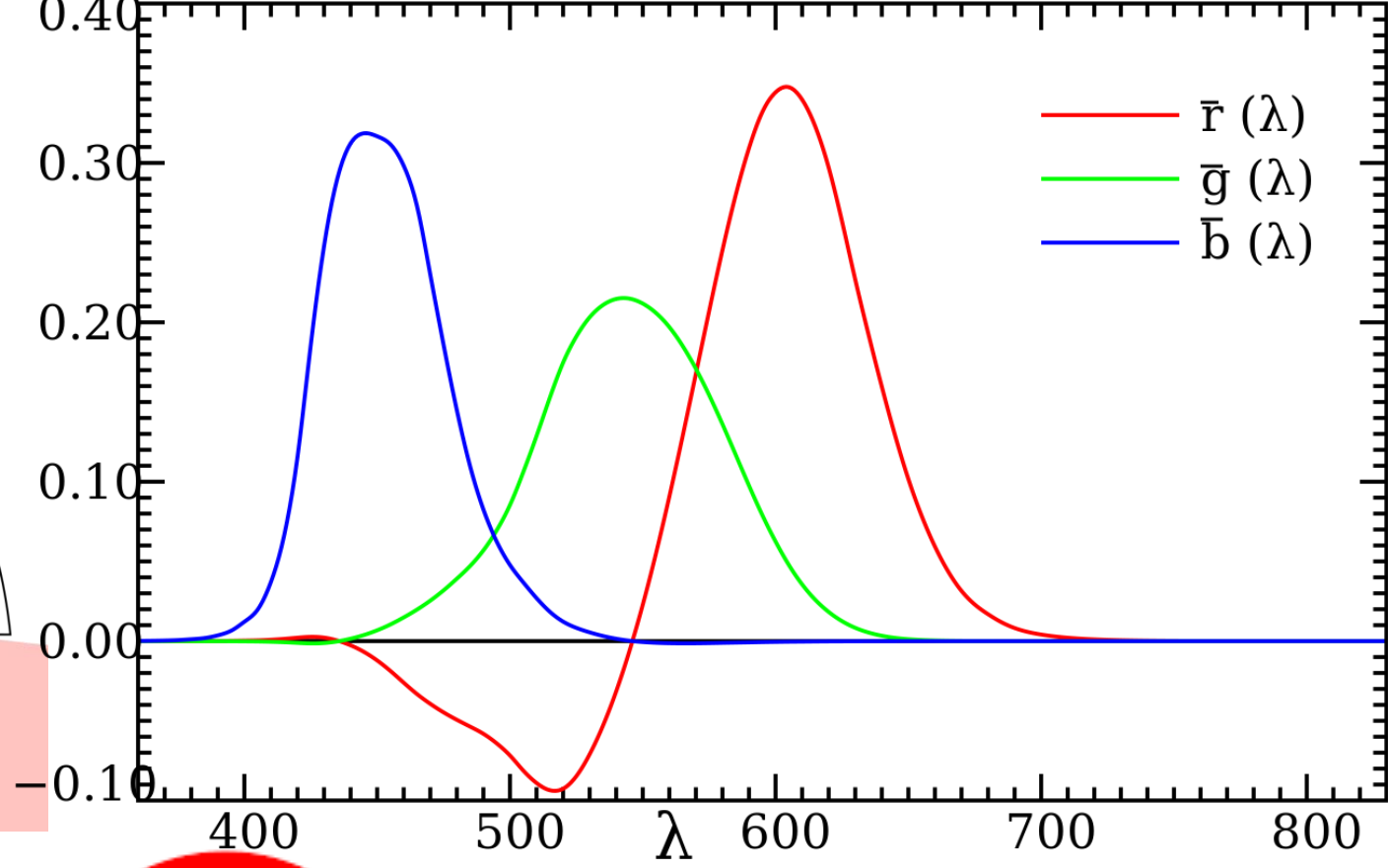
Aufbau der Bildpunkte eines TFT‘s

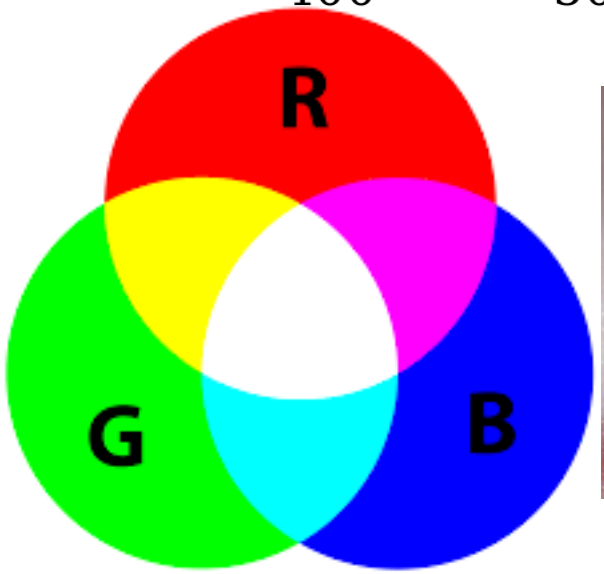
2.1 Bildschirmtechnologien



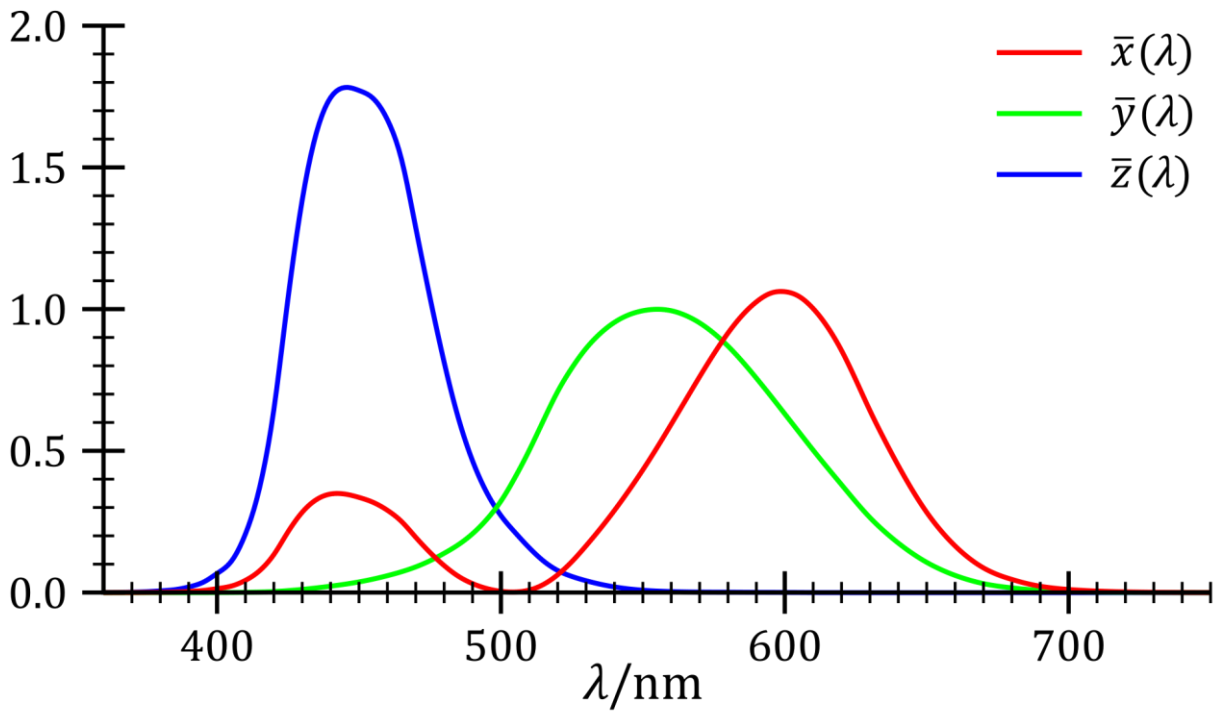
2.1 Bildschirmtechnologien

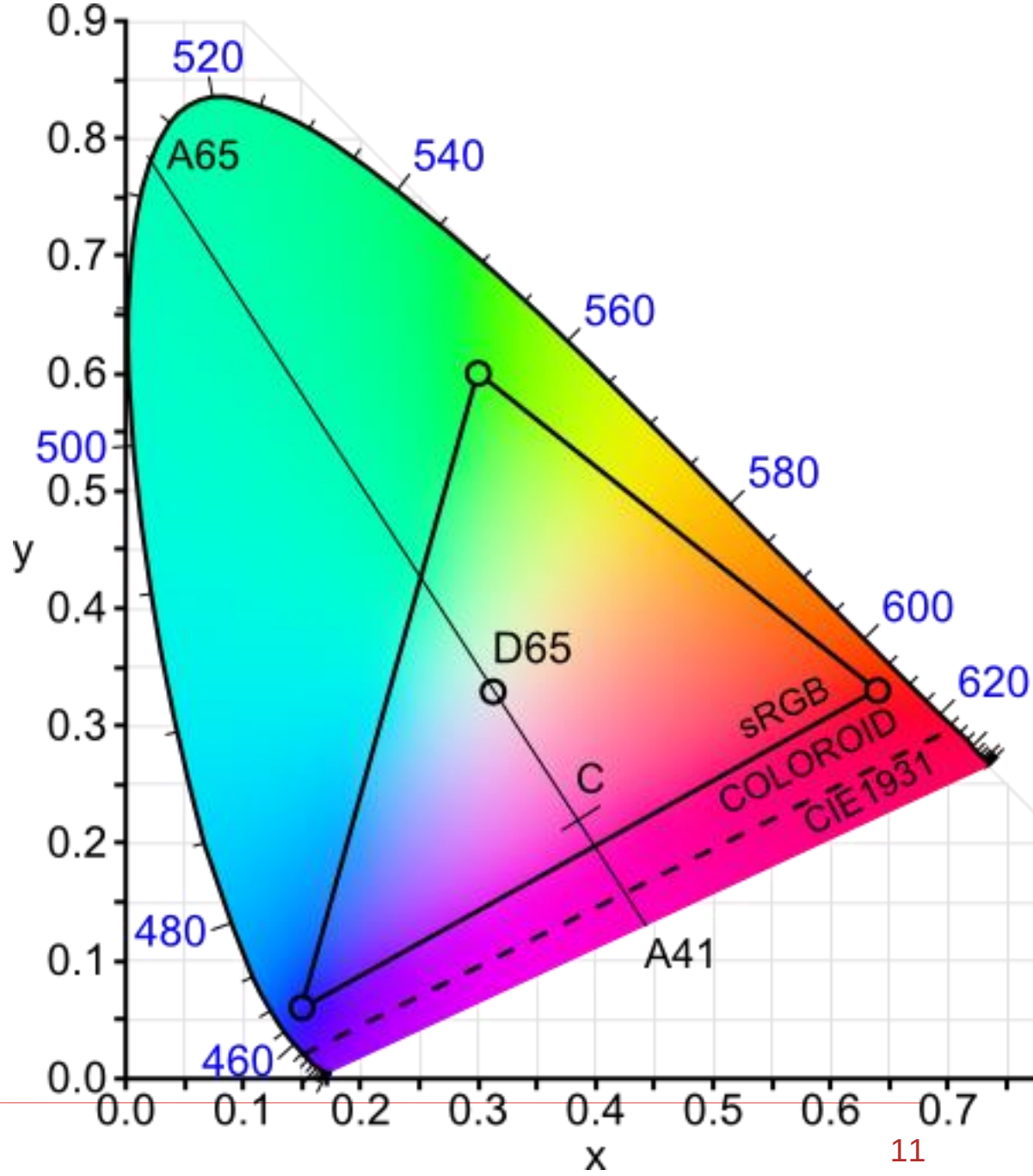






2.1 Bildschirmtechnologien





2.1 Bildschirmtechnologien

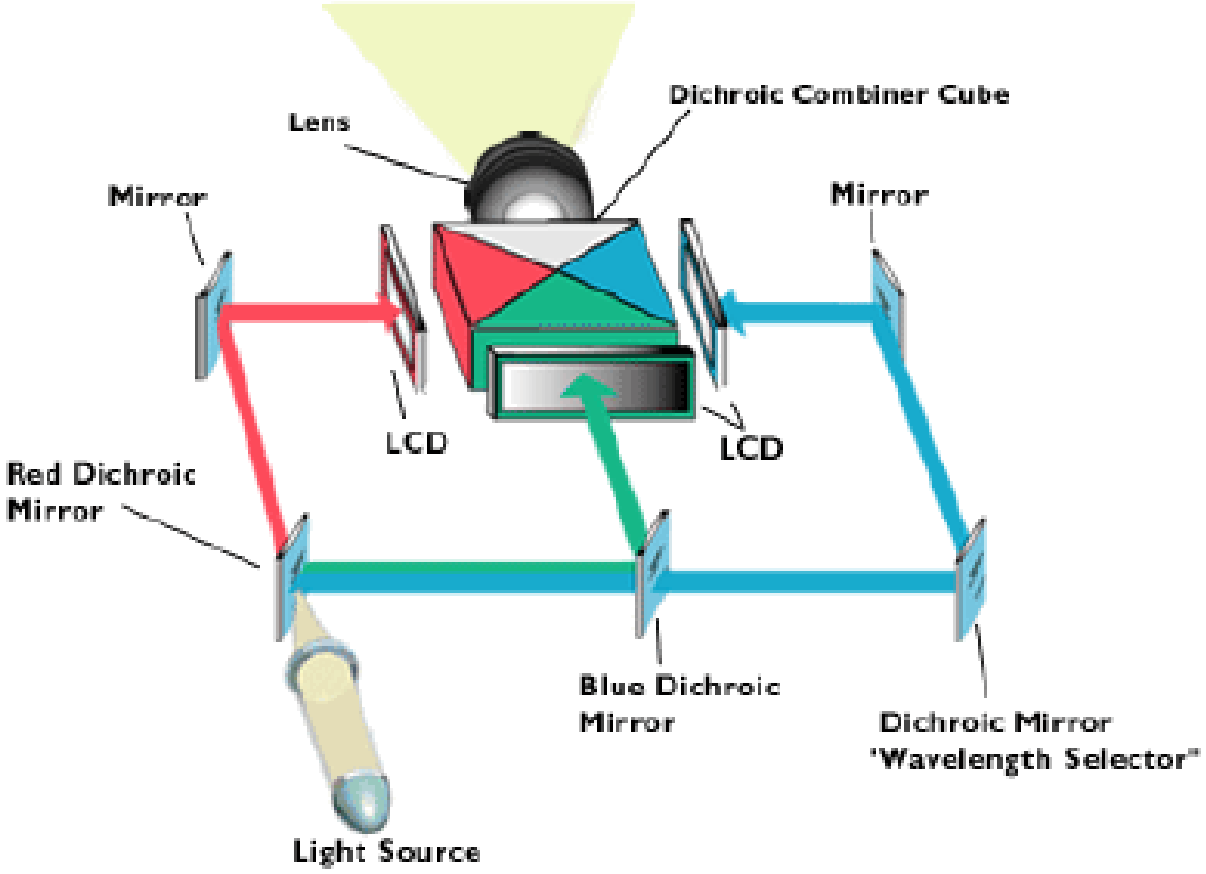
Organischer Leuchtdiodenbildschirm (OLED = organic light-emitting diode)

- Jedes Pixel hat seine eigenen Leuchtdioden aus organischen Materialien, die durch Spannungsänderung Licht emittieren.
- Keine Hintergrundbeleuchtung
- Vorteile
  - Echtes Schwarz, Guter Kontrast
  - Leicht/klein/mobil
  - Flexibel
  - Schnell
- Nachteile
  - Kurze Lebensdauer
  - Empfindlich
  - Einbrennen
  - Schlechte Lichtausbeute

2.2 Projektionstechnologien

LCD Projektoren (Liquid Crystal Display)

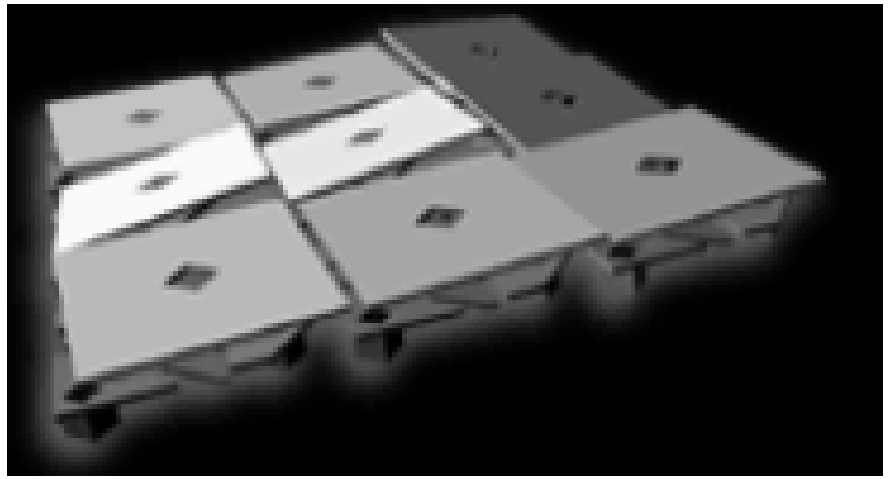
- Starke Lichtquelle
- Elektrische Ladung der Kristalle ändert Lichtdurchlässigkeit
- Kein Schwarz
- Günstig

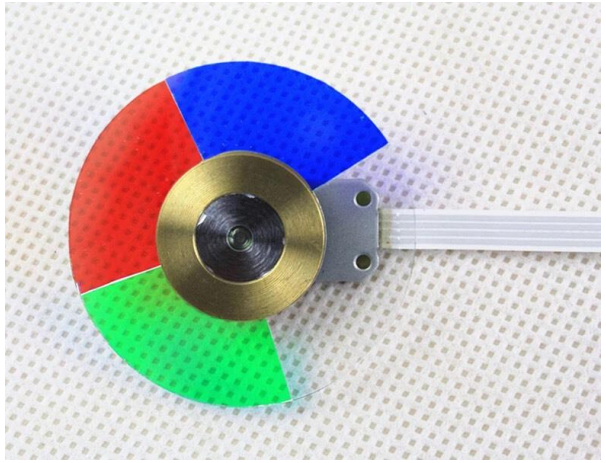


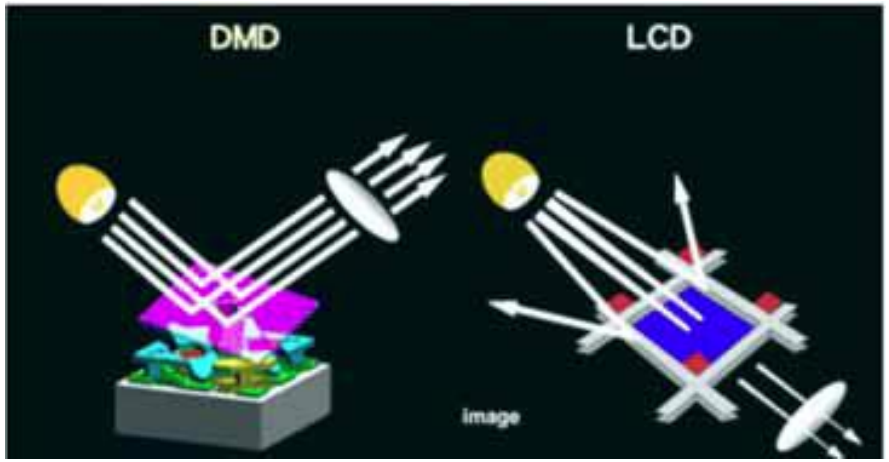
2.2 Projektionstechnologien

DLP Projektoren (Digital Light Processing)

- Auch Digital Micromirror Device (DMD TI)
- Feld von Spiegeln (ein Spiegel pro Pixel)
- Farben werden durch Farbrad zwischen Lampe und Spiegeln erzeugt

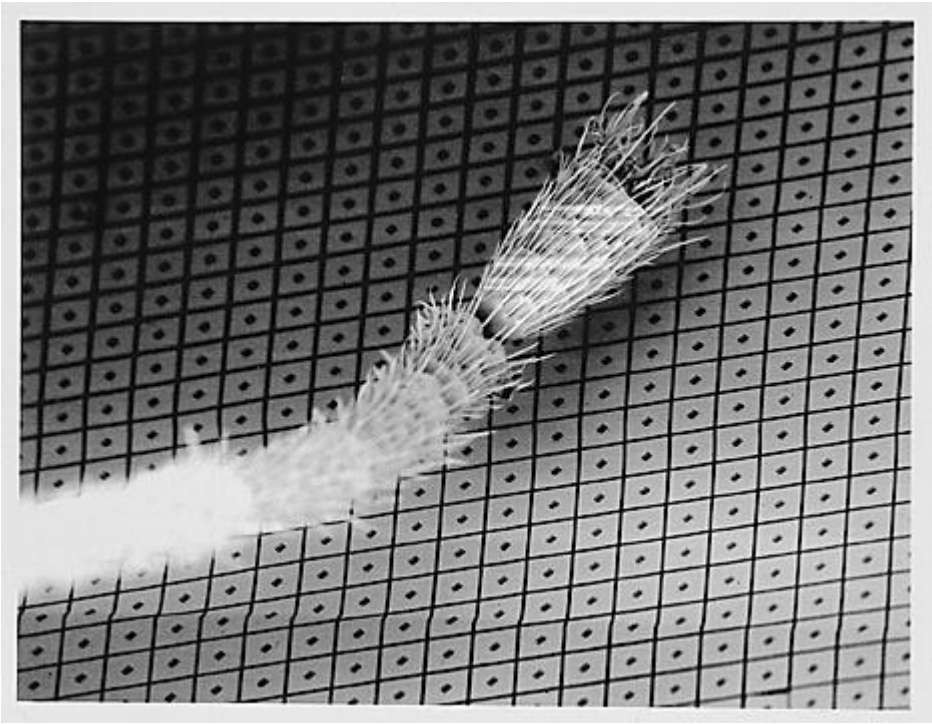


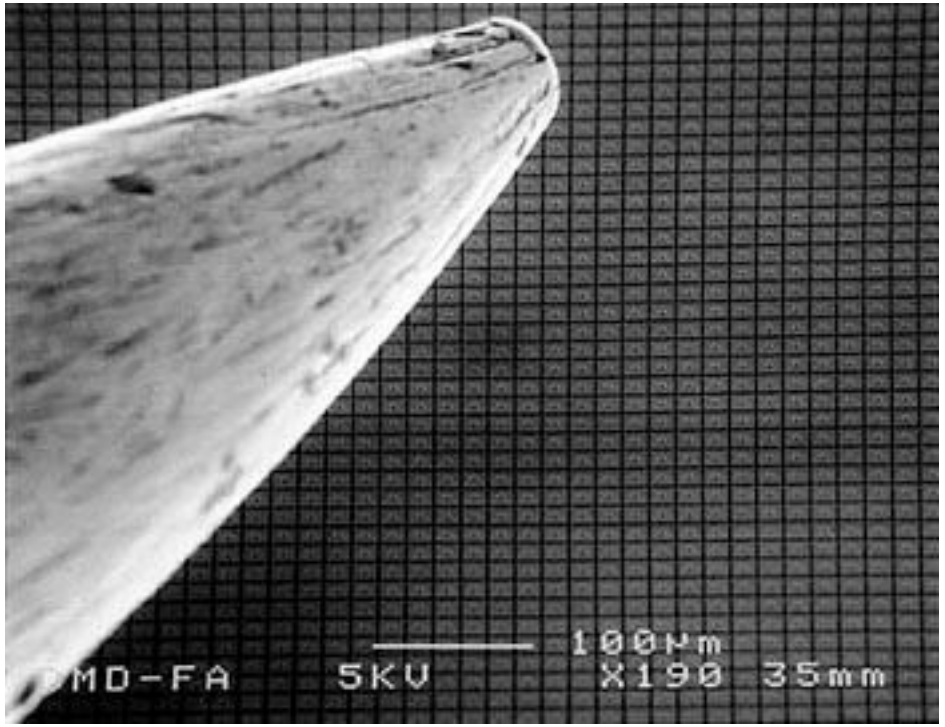




2.2 Projektionstechnologien

DLP Projektoren (Digital Light Processing)

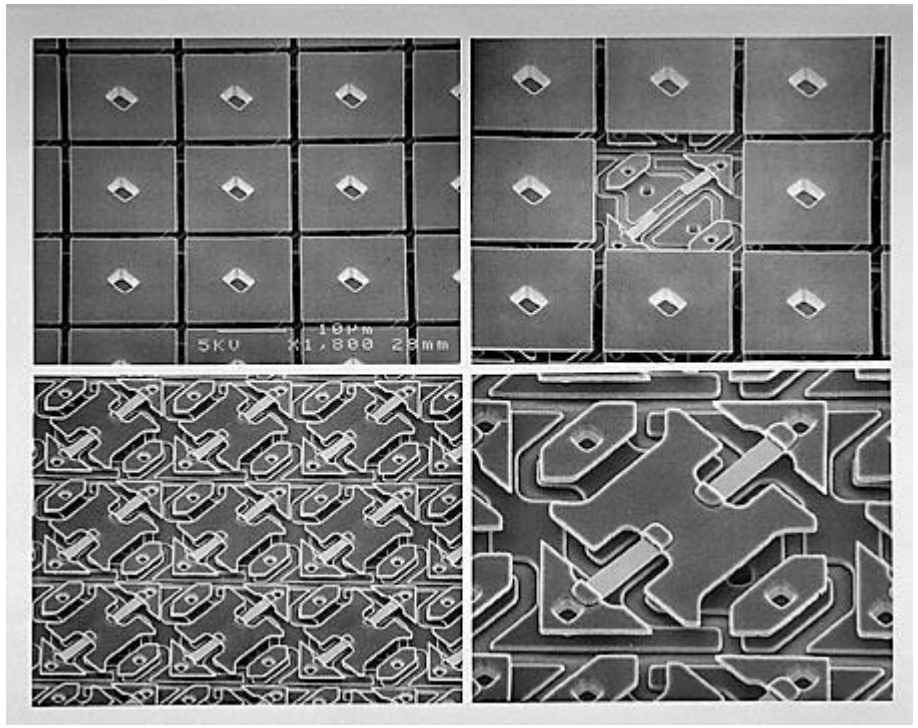


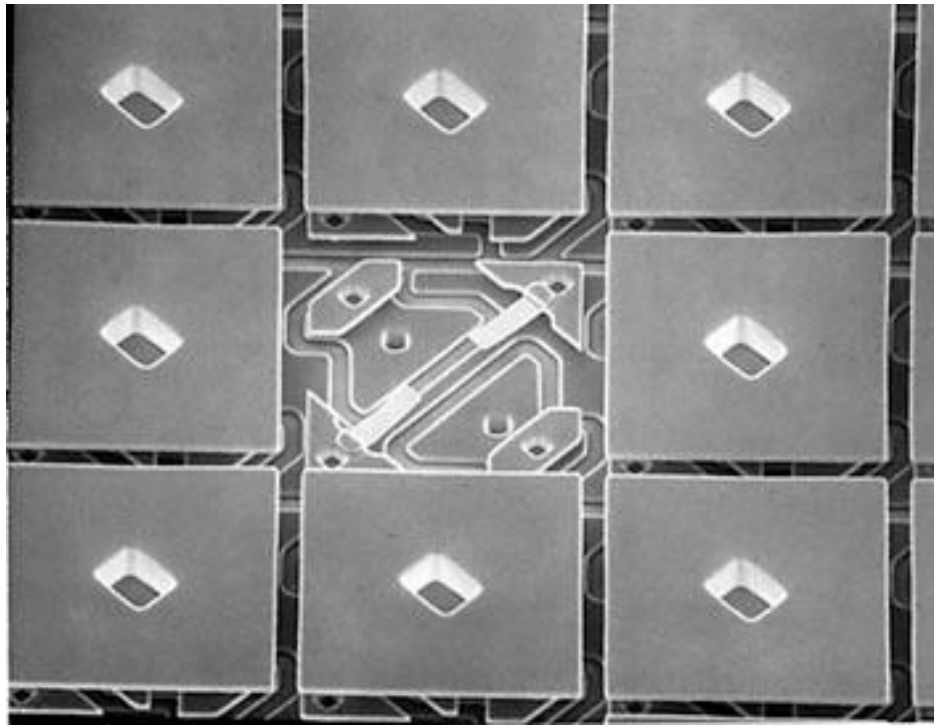


DMD: Digital Micro-Mirror Device
Mikroskopaufnahmen mit Vergleich zu Insektenhaar und Nadelspitze

2.2 Projektionstechnologien

DLP Projektoren (Digital Light Processing)

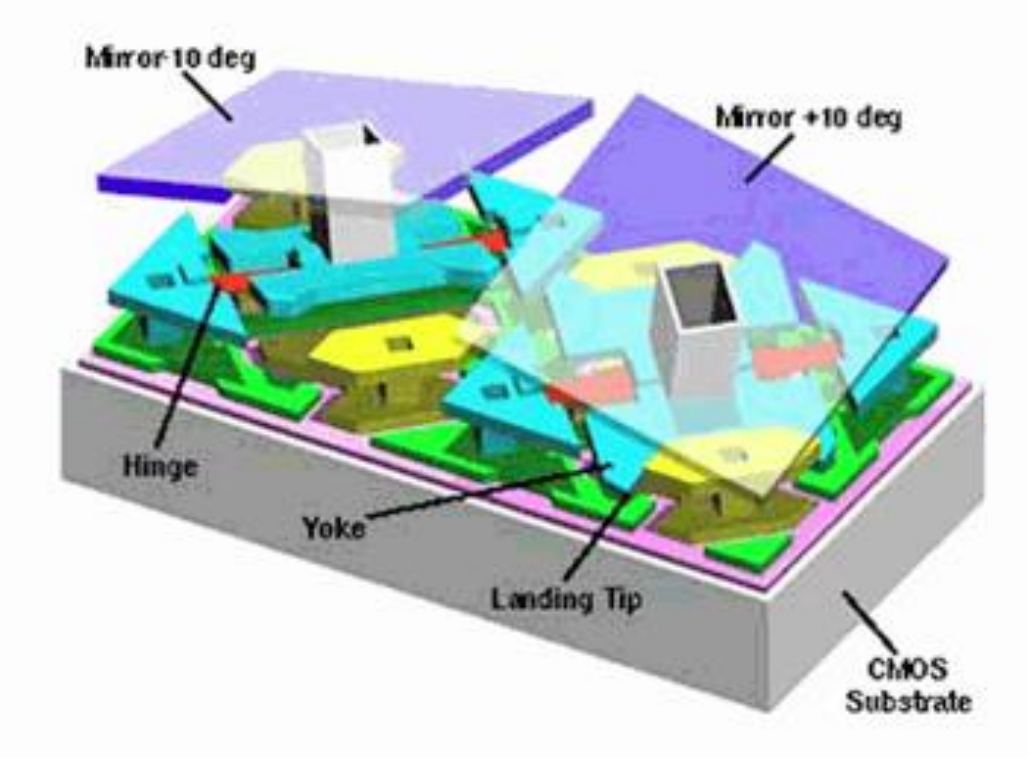


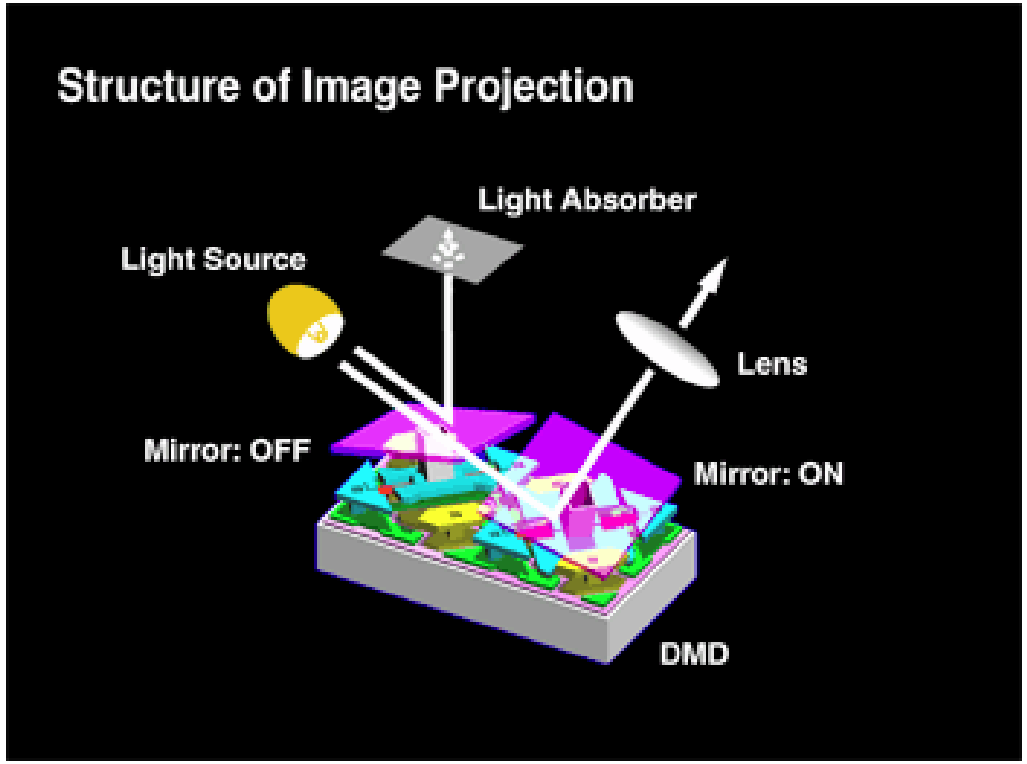


Mikrospiegel

2.2 Projektionstechnologien

DLP Projektoren (Digital Light Processing)



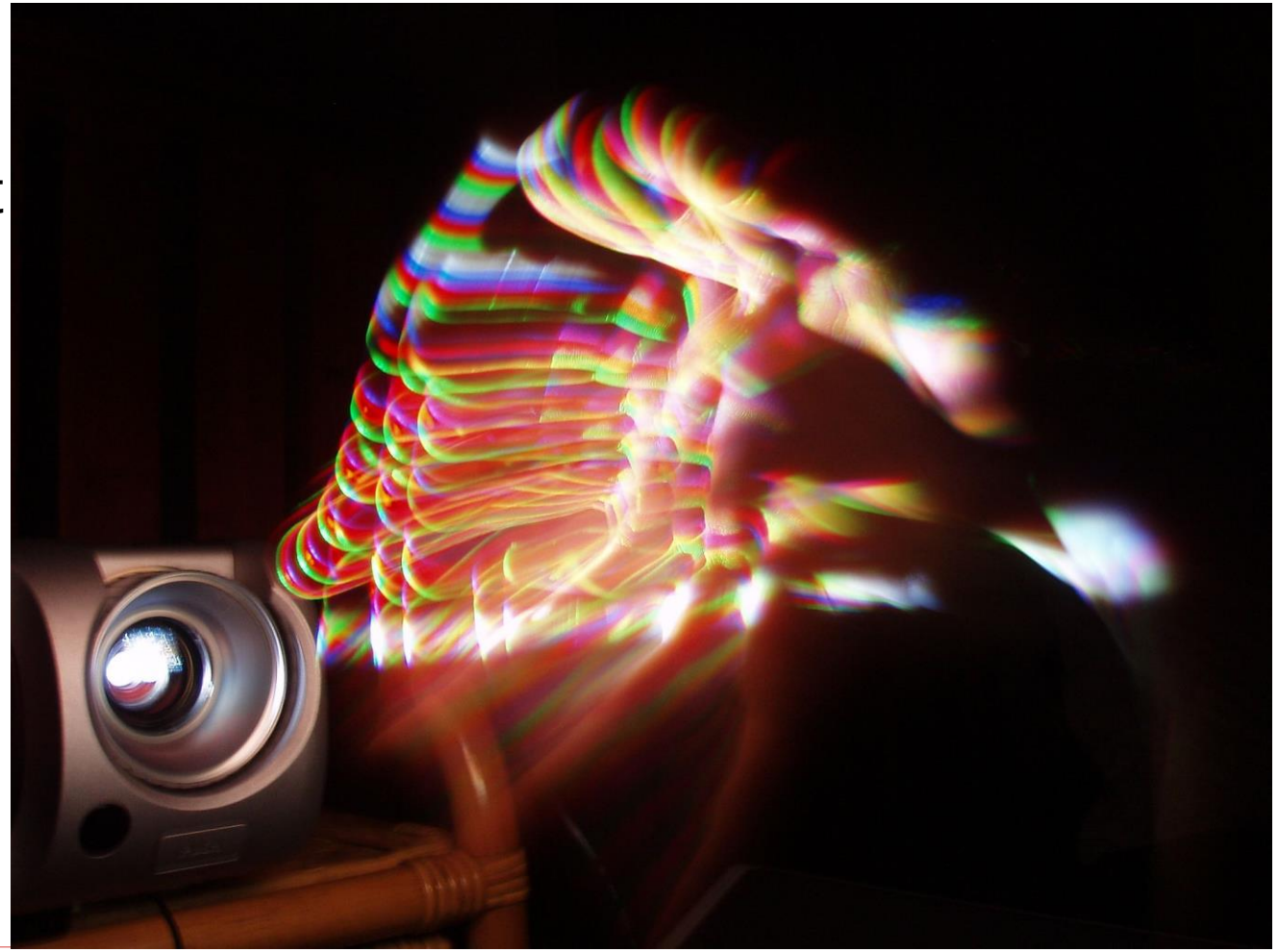


Mikrospiegel

2.2 Projektionstechnologien

DLP Projektoren (Digital Light Processing)

- Ein Chip (Single Chip) DLPs im Zeitmultiplex: langsamer, weniger Farben, Regenbogeneffekt
- Drei Chips (Three Chip DLPs): gleichzeitig, mehr Farben

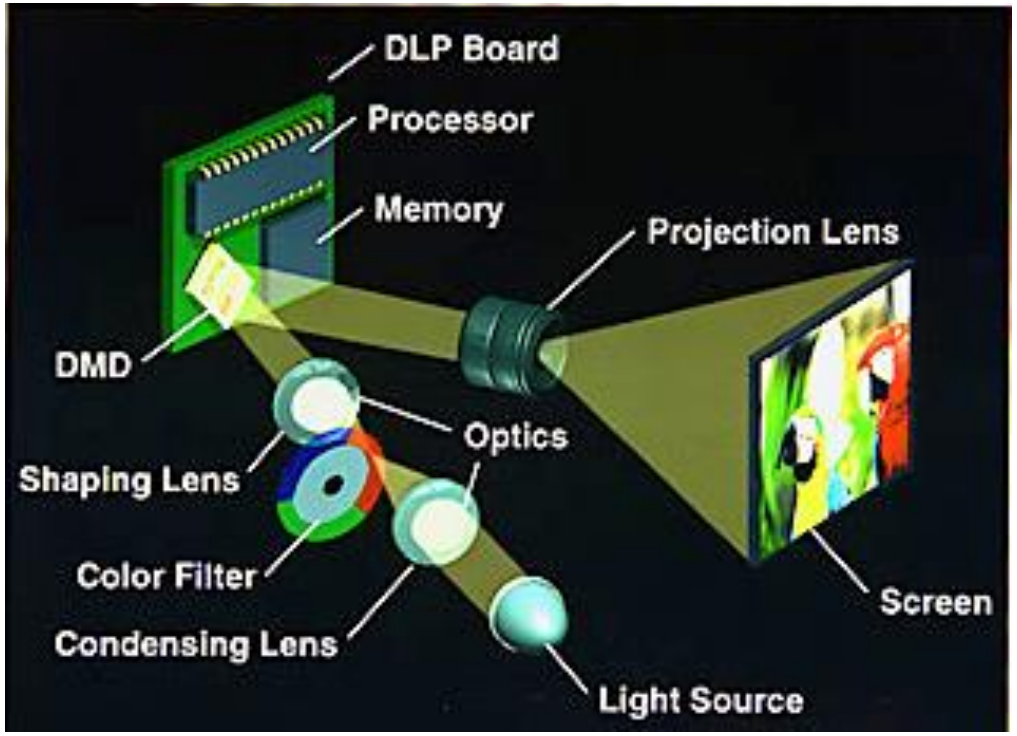


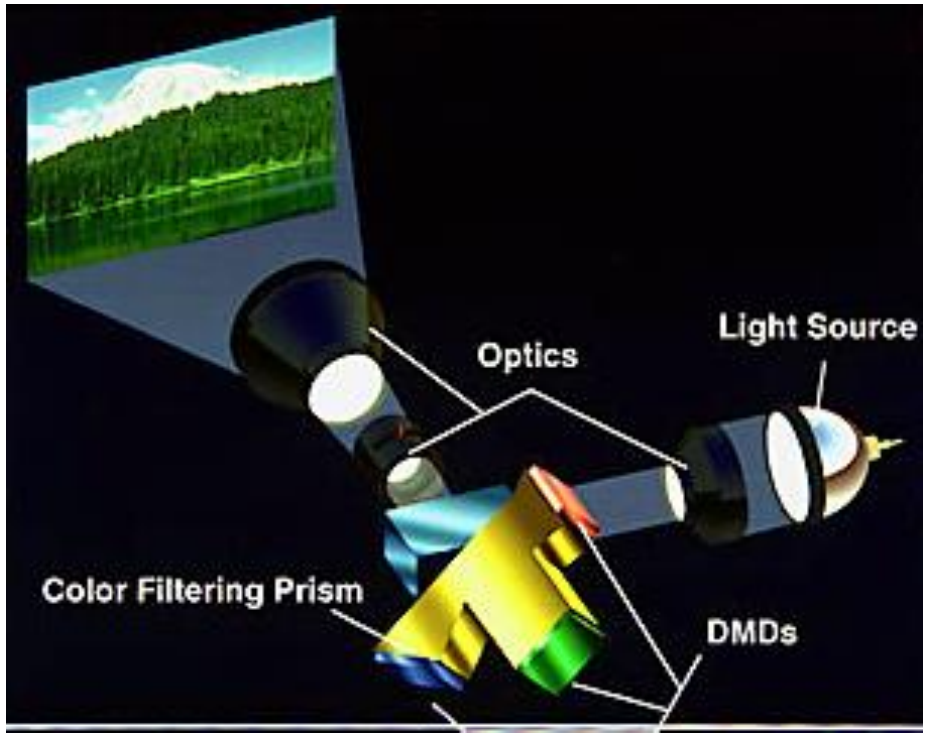
Vorteile:

- Bessere Lichtausnutzung
- Schnell
- Echtes Schwarz

2.2 Projektionstechnologien

DLP Projektoren (Digital Light Processing)

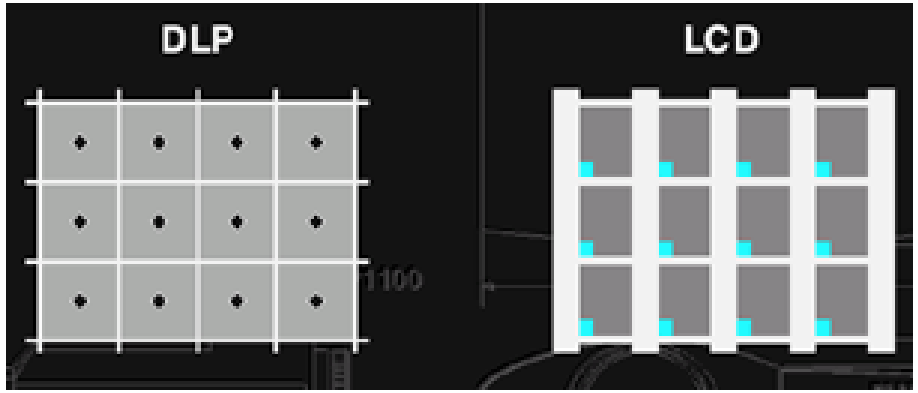


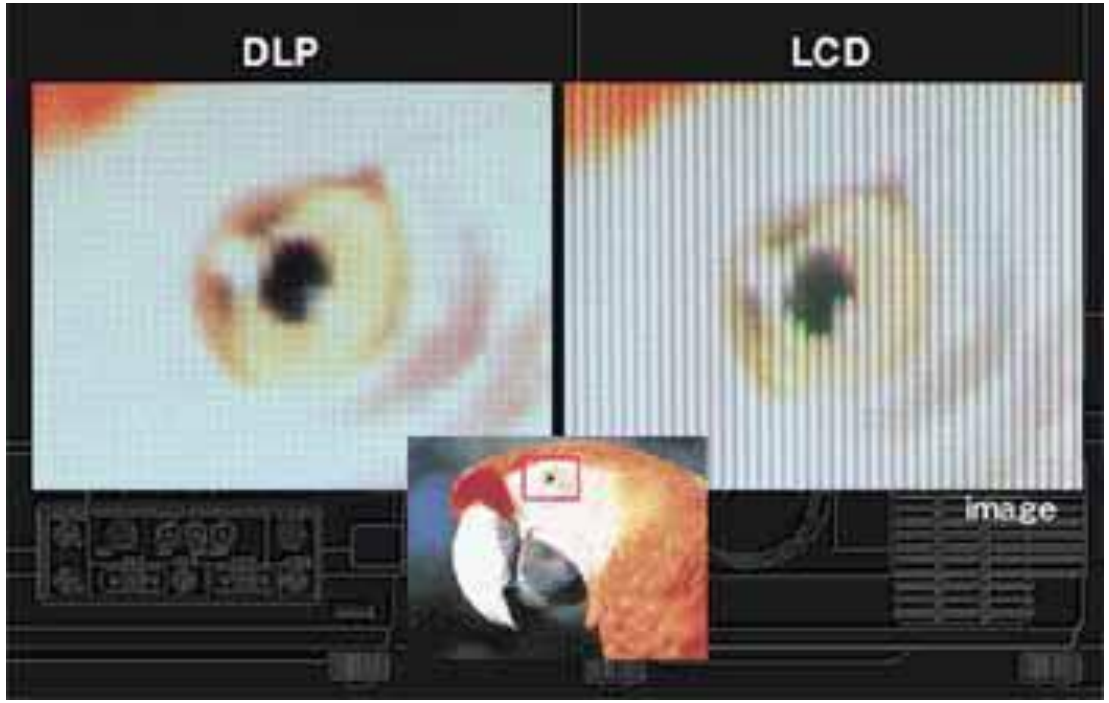


Mikrospiegel
links: 1 Chip DLP, rechts: 3 Chip DLP

2.2 Projektionstechnologien

DLP Projektoren (Digital Light Processing)





Mikrospiegelanordnung (Gitter)
Gitterartefakte stark reduziert

2.2 Projektionstechnologien

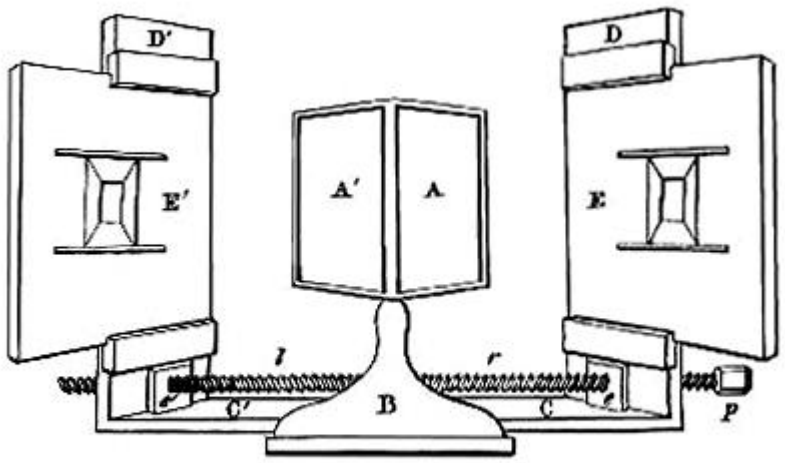
Weitere

- Liquid Crystal on Chip
  - ähnlich zu DLP
  - viel kleiner
  - anfällig für Hitze

- Laserprojektion

2.3 3D-Displaytechnologien

- Es werden getrennte Bilder für linkes und rechtes Auge errechnet
- Disparität der Bilder entscheidet über 3D-Eindruck
  - Augenabstand
  - Vergence: Augenrotation nach innen (Fokus)
- Zur Berechnung und Darstellung werden Multiplex-Verfahren eingesetzt



The Wheatstone stereoscope used angled mirrors (A) to reflect the stereoscopic drawings (E) toward the viewer's eyes.

2.3 3D-Displaytechnologien

Farbmultiplex: Anaglyphs, Infitec

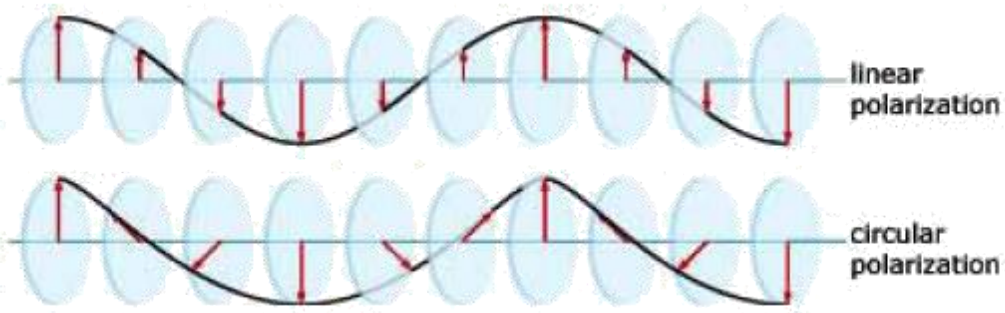
2.3 3D-Displaytechnologien

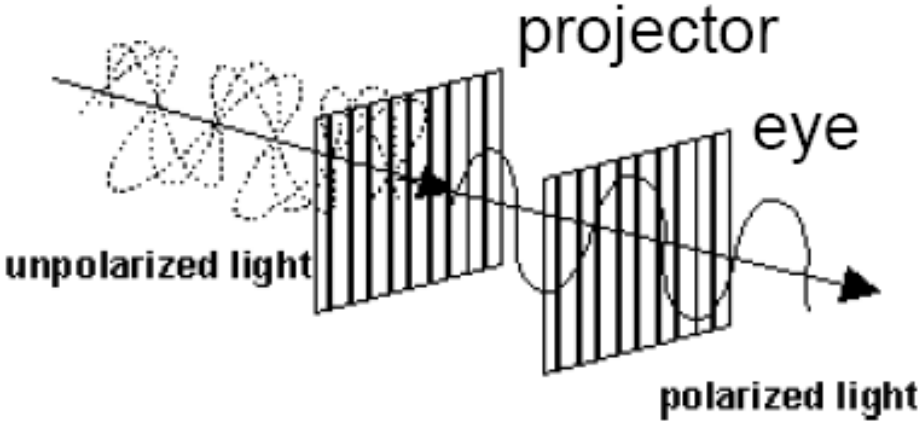
Farbmultiplex: Anaglyphs, Infitec

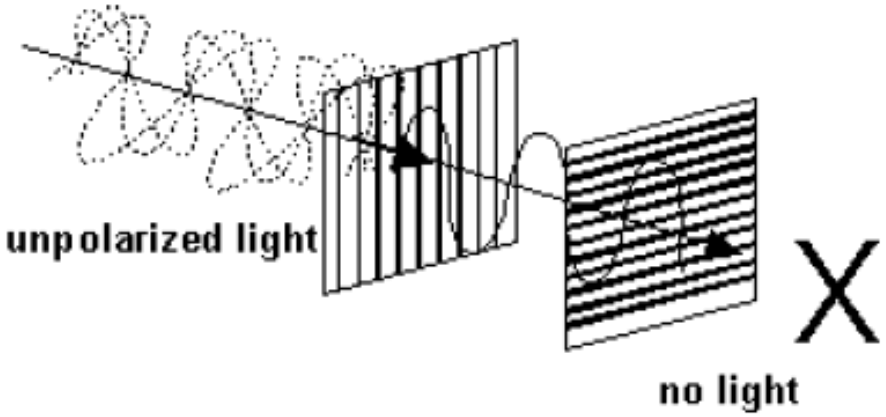
2.3 3D-Displaytechnologien

Polarisationsmultiplex: Senkrecht/Waagrecht, zirkulär

– Links-/Rechtssynchronisation durch Polarisationsfilter (Projektor und Brillen)



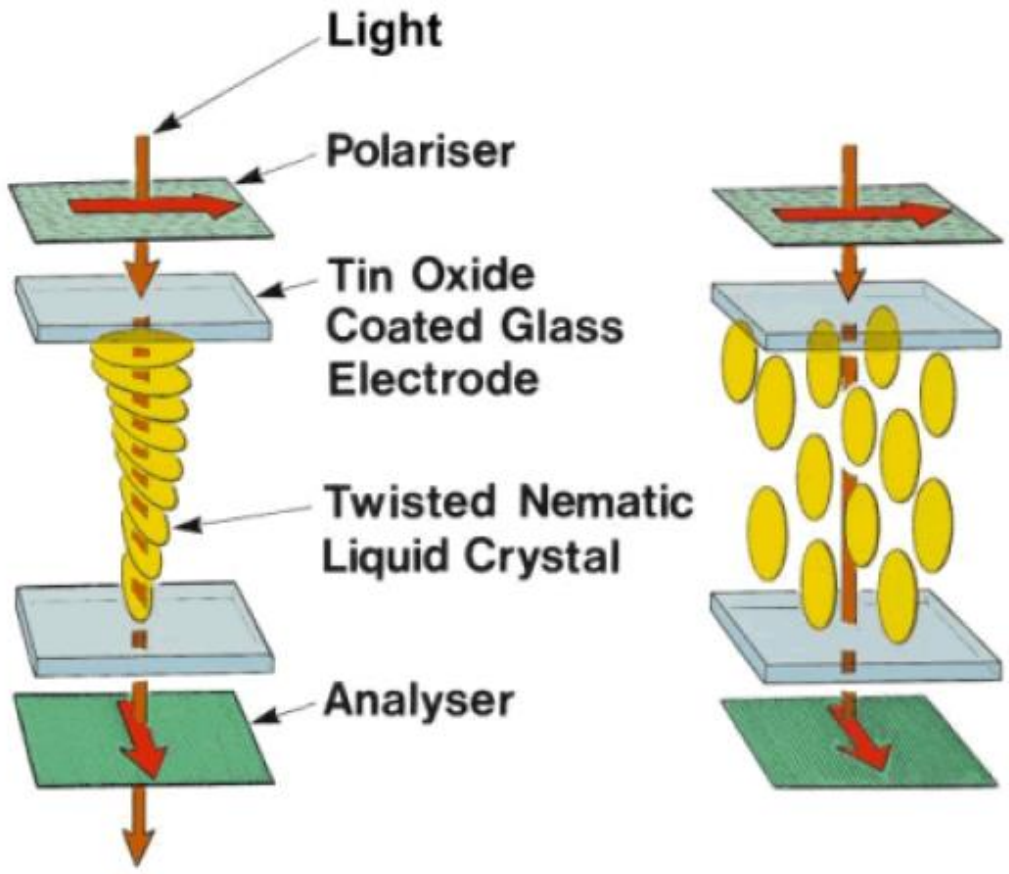




2.3 3D-Displaytechnologien

Zeitmultiplex: Shutter-Glasses

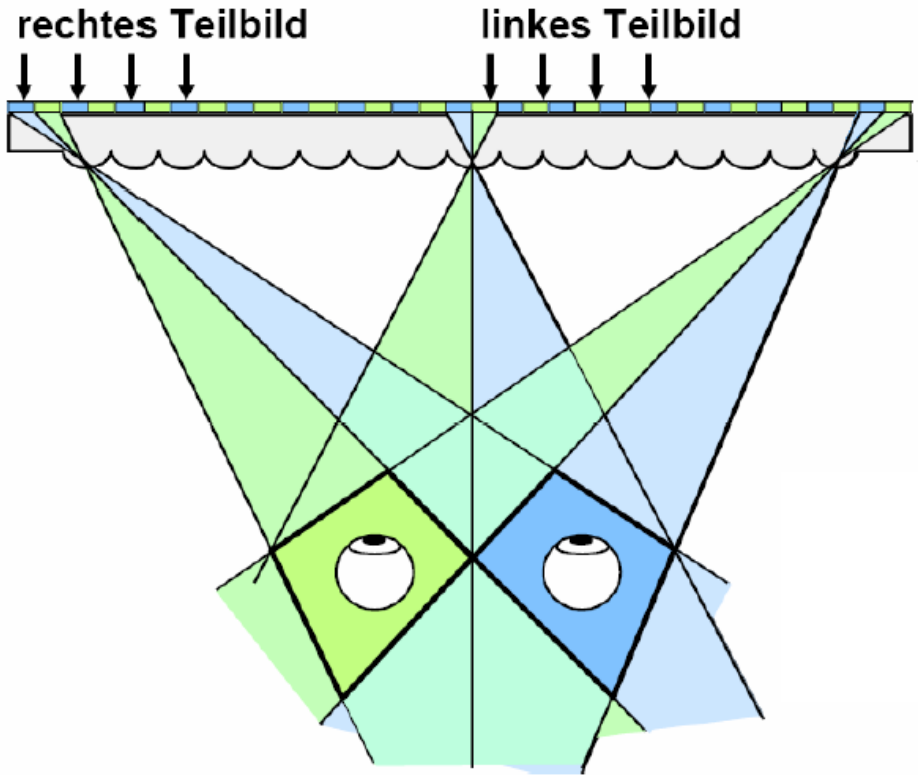
– Synchronisiertes An-/Ausschalten der Brillengläser



2.3 3D-Displaytechnologien

Richtungsmultiplex: Linsensysteme (Autostereoskopische Systeme)

- Pixellinsen für Links-/Rechtsaufteilung
- Halbe Auflösung
- Links-/Rechtsteilbilder verschränkt



2.3 3D-Displaytechnologien

Head Mounted Display (HMD)

- Besteht aus zwei Bildschirmen, die am Kopf des Benutzers angebracht sind.
- Liefert in Abhängigkeit von Position und Orientierung Stereobilder
- Es entsteht der Eindruck einer dreidimensionalen Umgebung

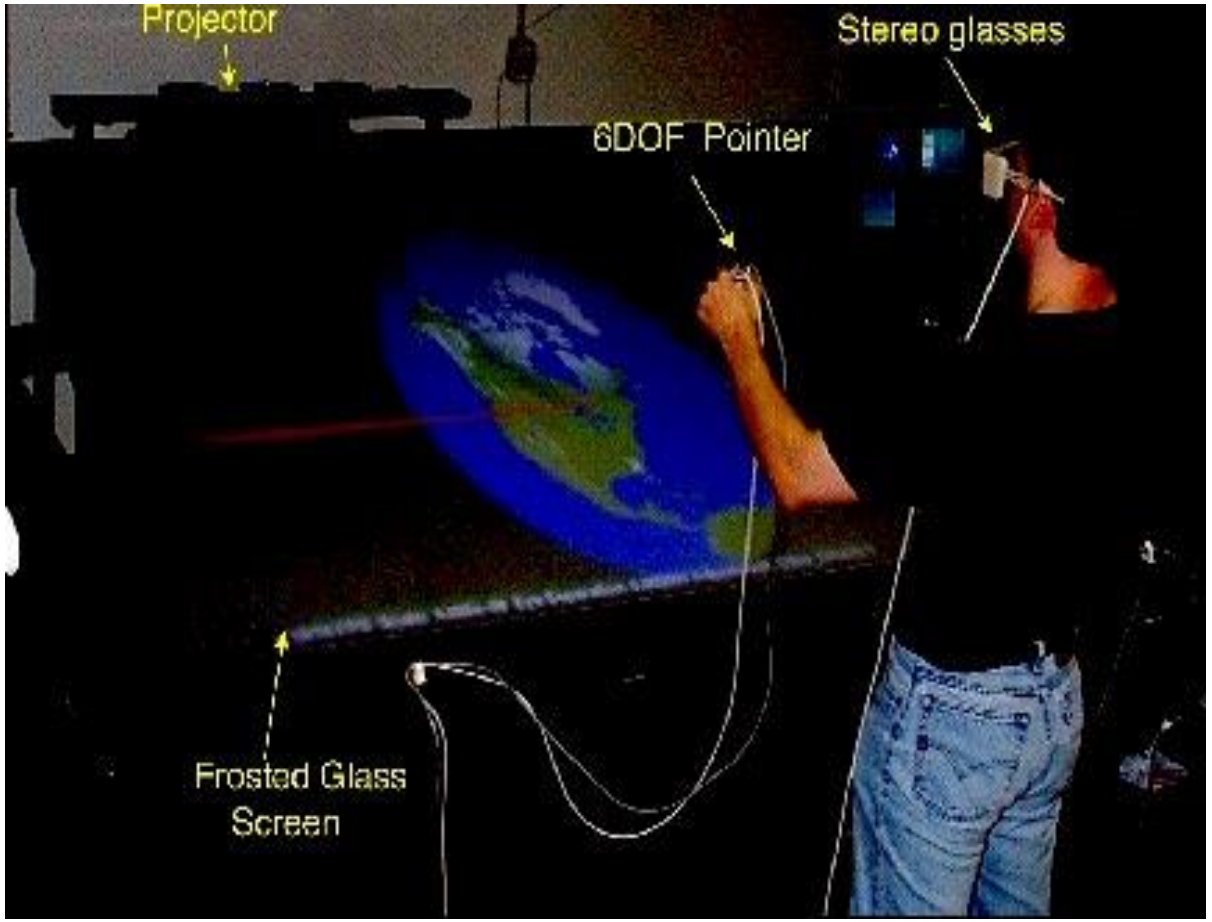
2.3 3D-Displaytechnologien

Responsive Workbench / VR Bench / Virtual Table

- Ein fest installierter Großbildschirm, der mit einer Frequenz von 100 Hz alternierende Stereobilder anzeigt
- Der Benutzer trägt dabei i.A. Shutter-Glasses, welche mit Hilfe von Polarisationsfiltern nur die Bilder für das linke bzw. rechte Auge durchlassen
- Die Position und die Orientierung der Shutter-Glasses wird durch ein Trackingssystem erfasst und an die bildgenerierende Software weitergeleitet
- Mit Hilfe von Tracking können auch virtuelle Werkzeuge durch den Benutzer positioniert werden
- An einer Workbench können bis zu zwei Benutzer arbeiten

2.3 3D-Displaytechnologien

Responsive Workbench / VR Bench / Virtual Table



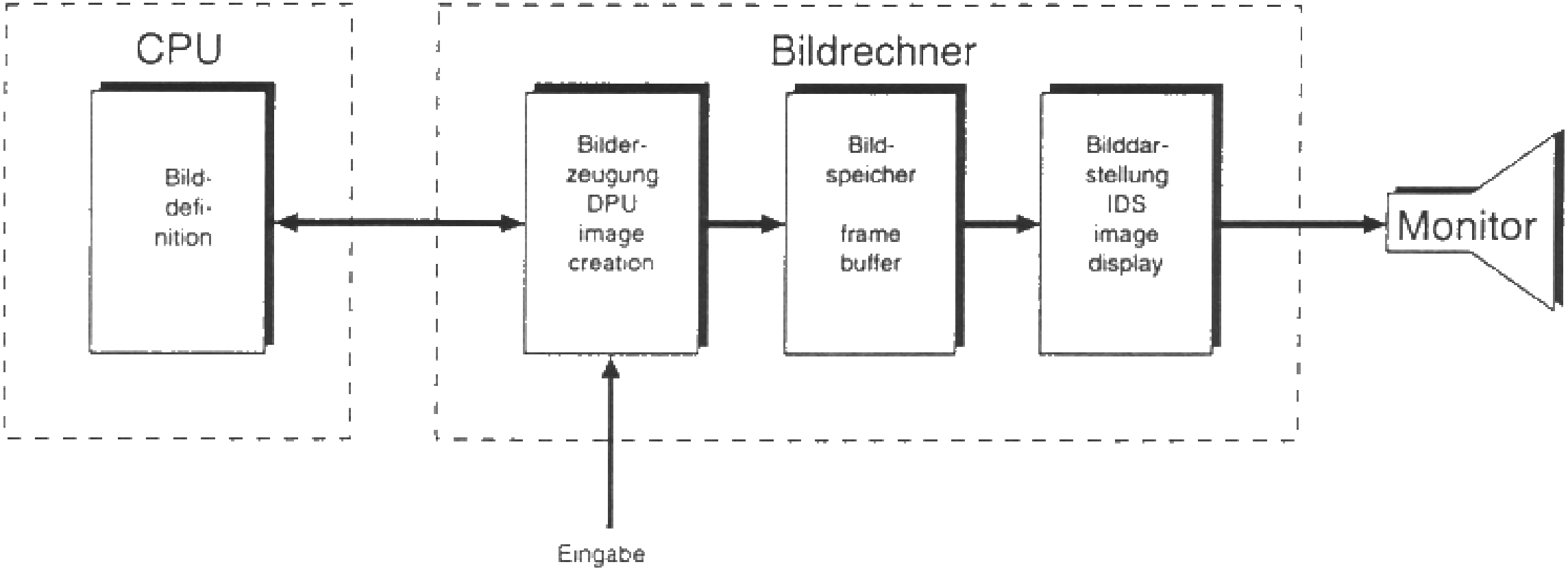
2.3 3D-Displaytechnologien

CAVE

- Die CAVE (Cave Automatic Virtual Environment) besteht aus 4-6 Großbildschirmen, die den Benutzer umgeben.
- Dadurch ist der Bildbereich nahezu unbegrenzt.

2.4 Bildrechner/GPU

Hauptkomponenten eines klassischen Rasterdisplays



2.4 Bildrechner/GPU

- Der wichtigste Teil des Bildrechners ist der Displayprozessor (Display Processing Unit), welcher die Bilddefinition des Anwendungsprogramms so aufbereitet, dass auf dem Ausgabegerät (Display) das gewünschte Bild erscheint
- Bei den heute gebräuchlichen Rastergeräten müssen die graphischen Objekte (Linien, Kreise, Dreiecke,....) gerastert werden (siehe §5) und in einem Bildspeicher (Framebuffer) abgelegt werden.

2.4 Bildrechner/GPU

- Die Aufrechterhaltung des Bildes erfolgt mittels periodischen Auslesens des Bildspeichers durch den Videocontroller (Image Display System) - RAMDAC (RAM D/A Converter), sofern die Darstellung noch analoge Signale erfordert.

- Bei digitalem Ausgang wird auf die Umwandlung in analoge Signale verzichtet.

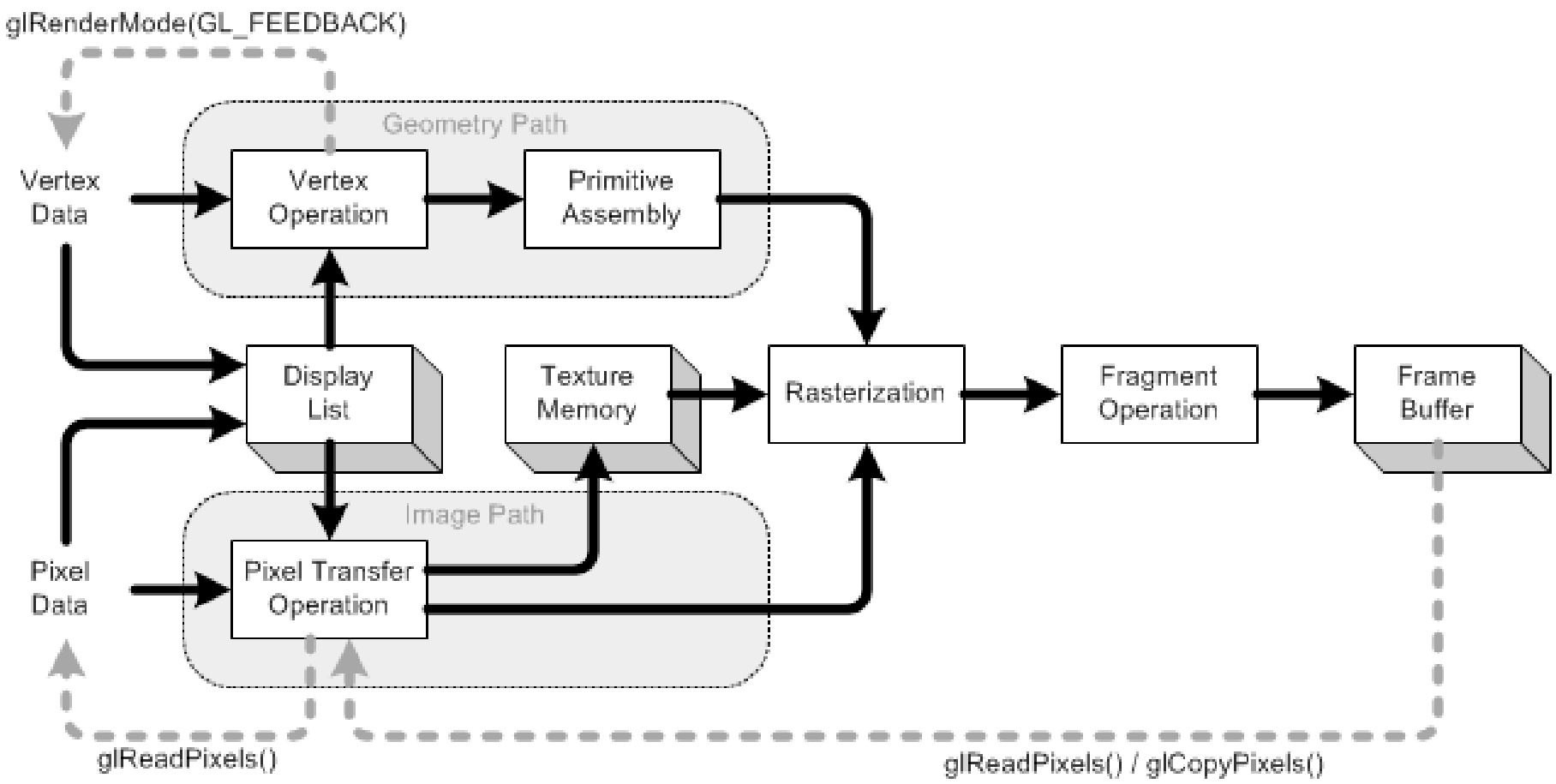
2.4 Bildrechner/GPU

- Heute werden alle Komponenten in einer Graphics Processing Unit (GPU) zusammengefasst
- Graphik-Pipeline wird auf Komponenten verteilt
- Bestimmte Komponenten sind hoch parallelisiert (SIMD/Vektorisiert)
- Pipelining

- Früher waren Komponenten statisch
- Heute sind sie programmierbar: Shader-Programme
- Hohe kanonische Leistung (GPU) gegenüber Programm-Flexibilität (CPU)

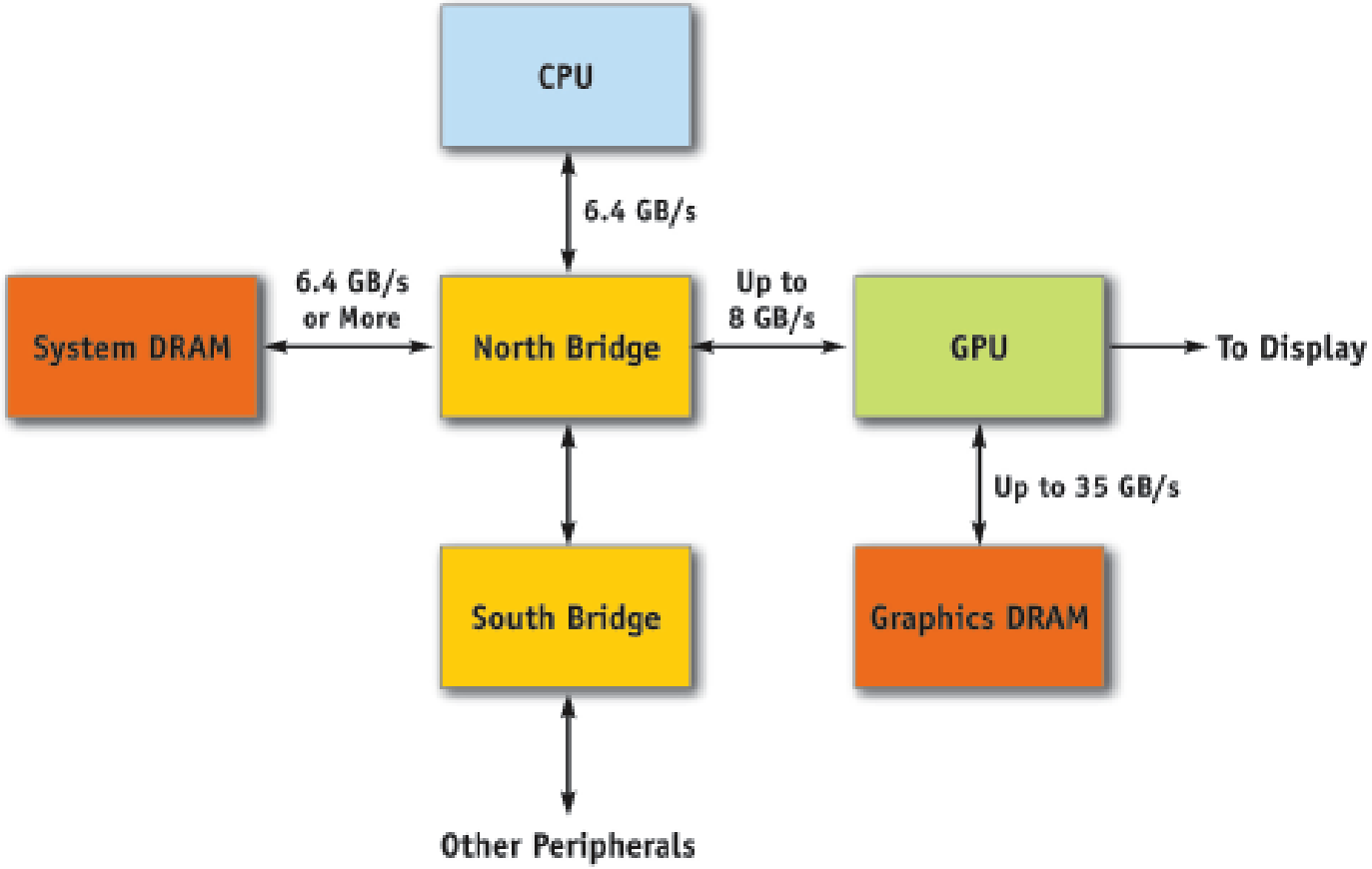
2.4 Bildrechner/GPU

OpenGL Rendering-Pipeline [www.sungho.ca]



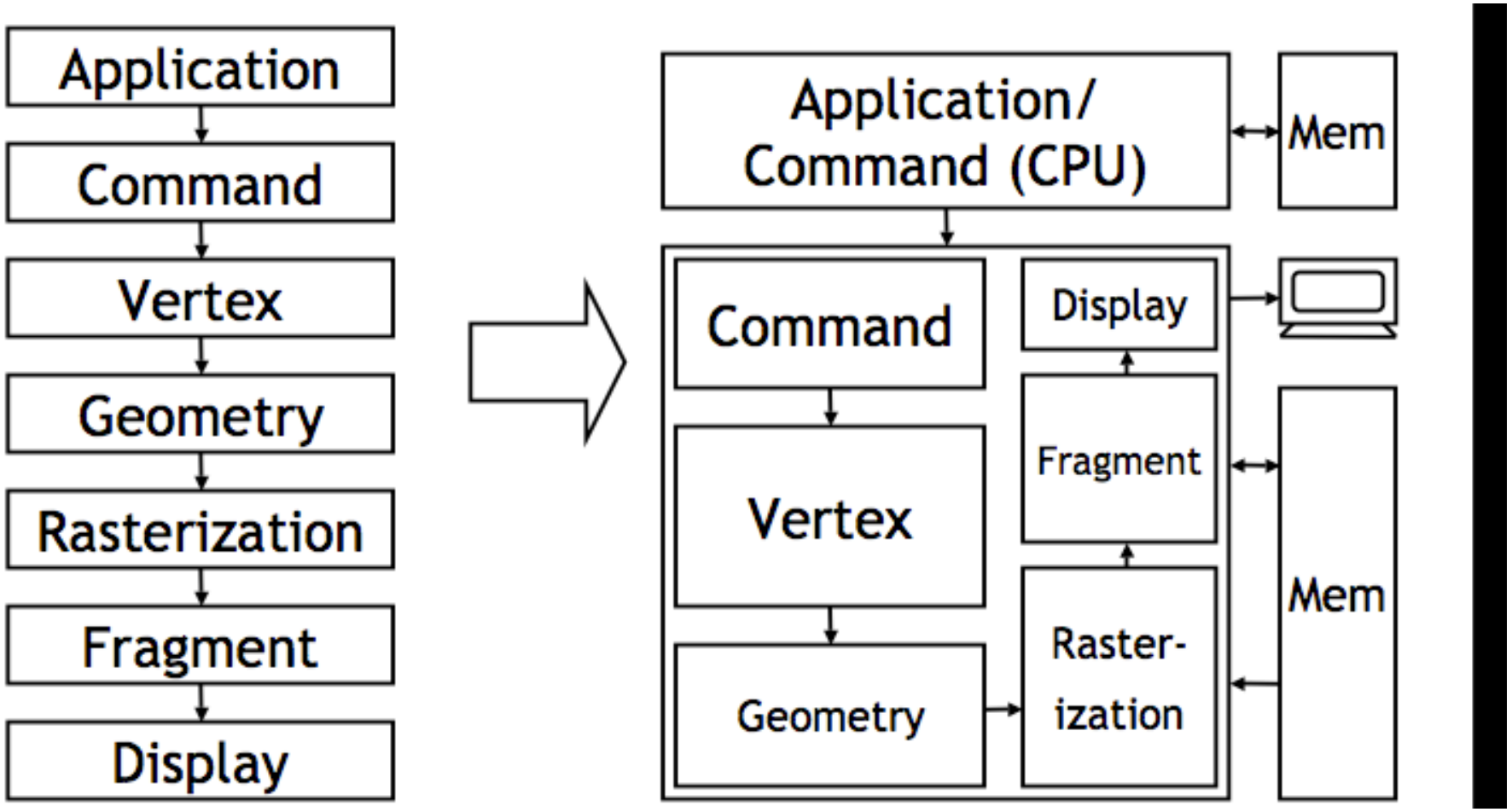
2.4 Bildrechner/GPU

Systemarchitektur eines PCs



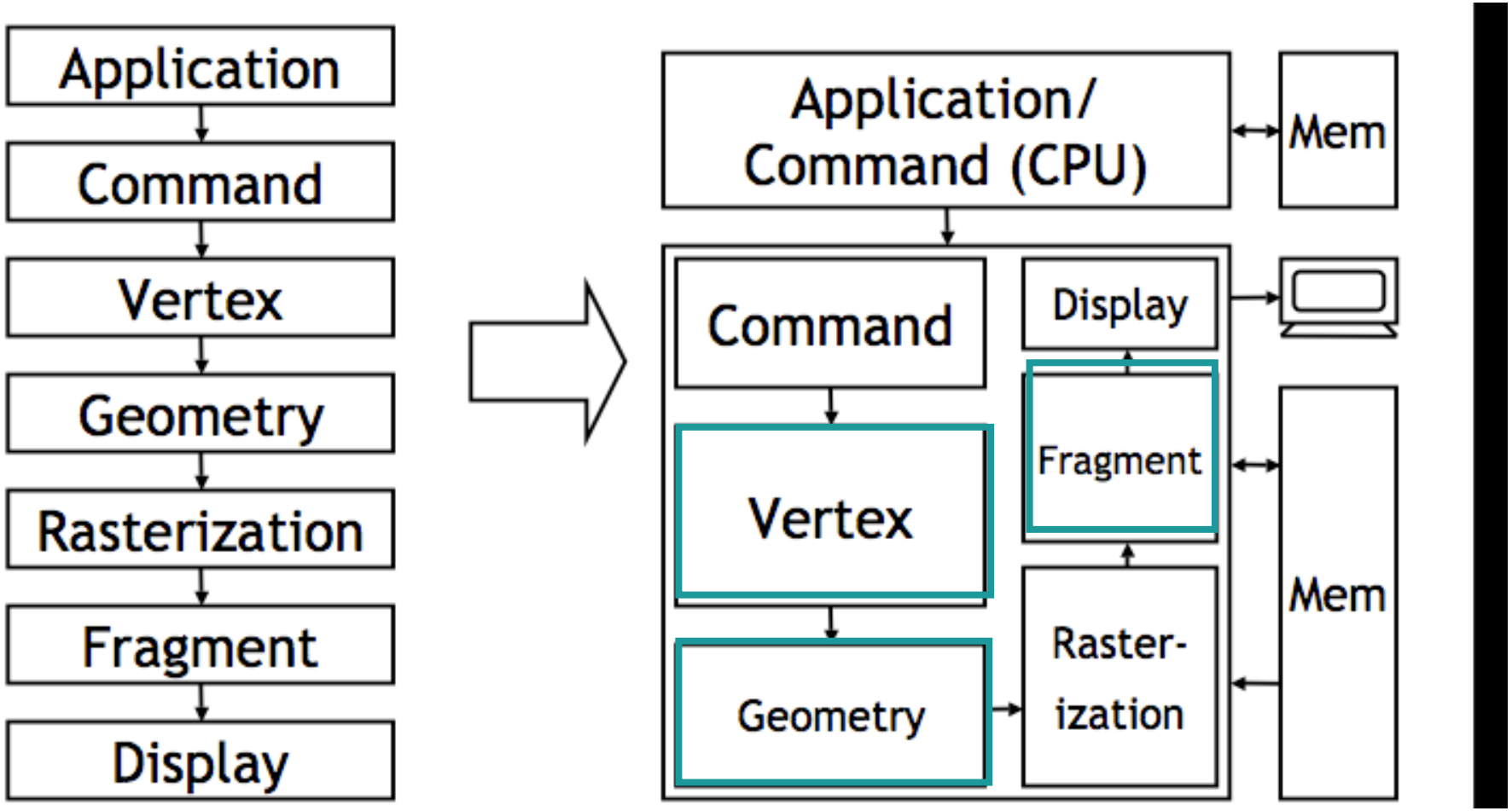
2.4 Bildrechner/GPU

Evolution der Graphikhardware (ca. 2001) [J. Owen, UC Davis]



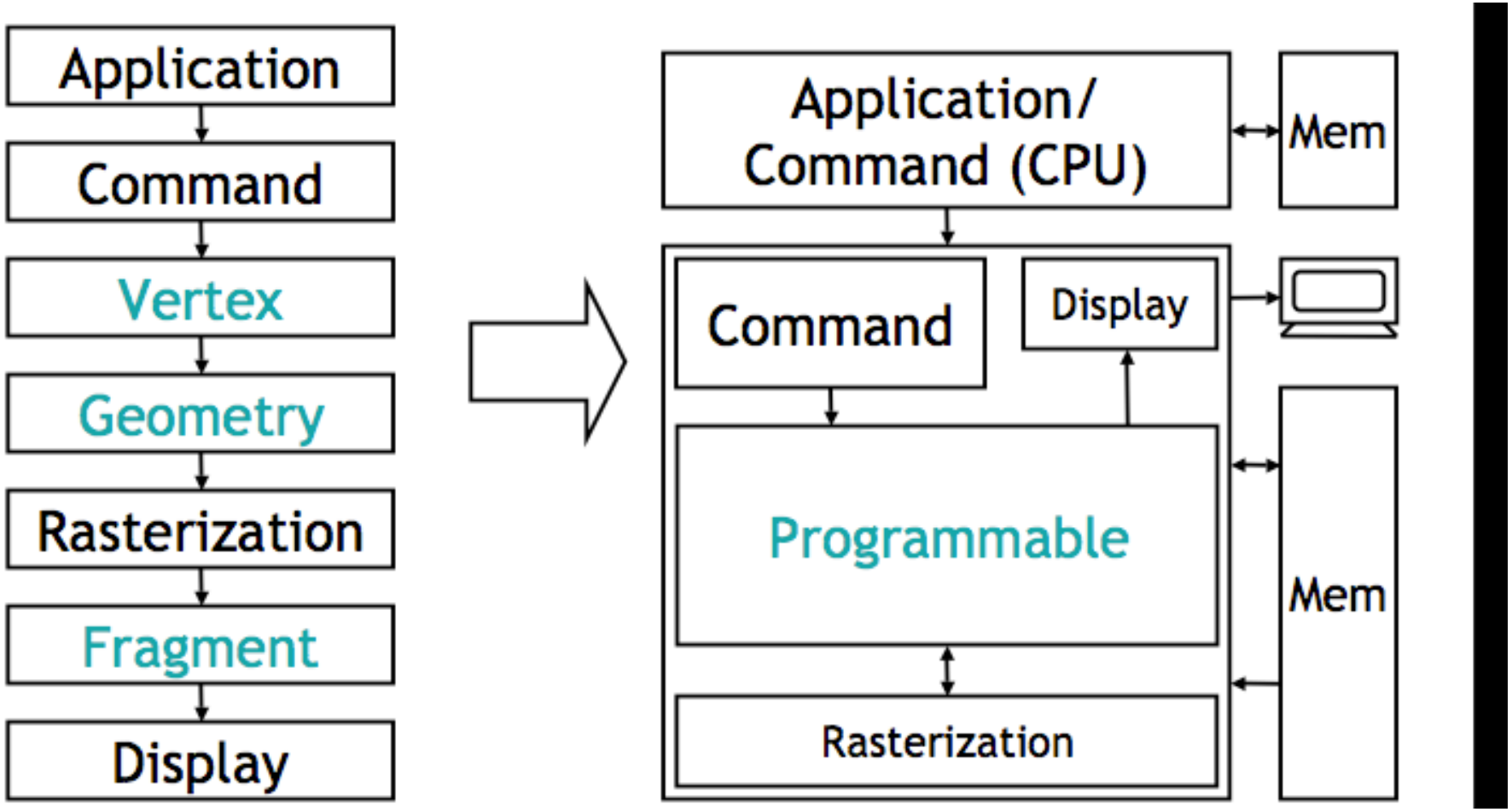
2.4 Bildrechner/GPU

Evolution der Graphikhardware (ab 2001) [J. Owen, UC Davis]



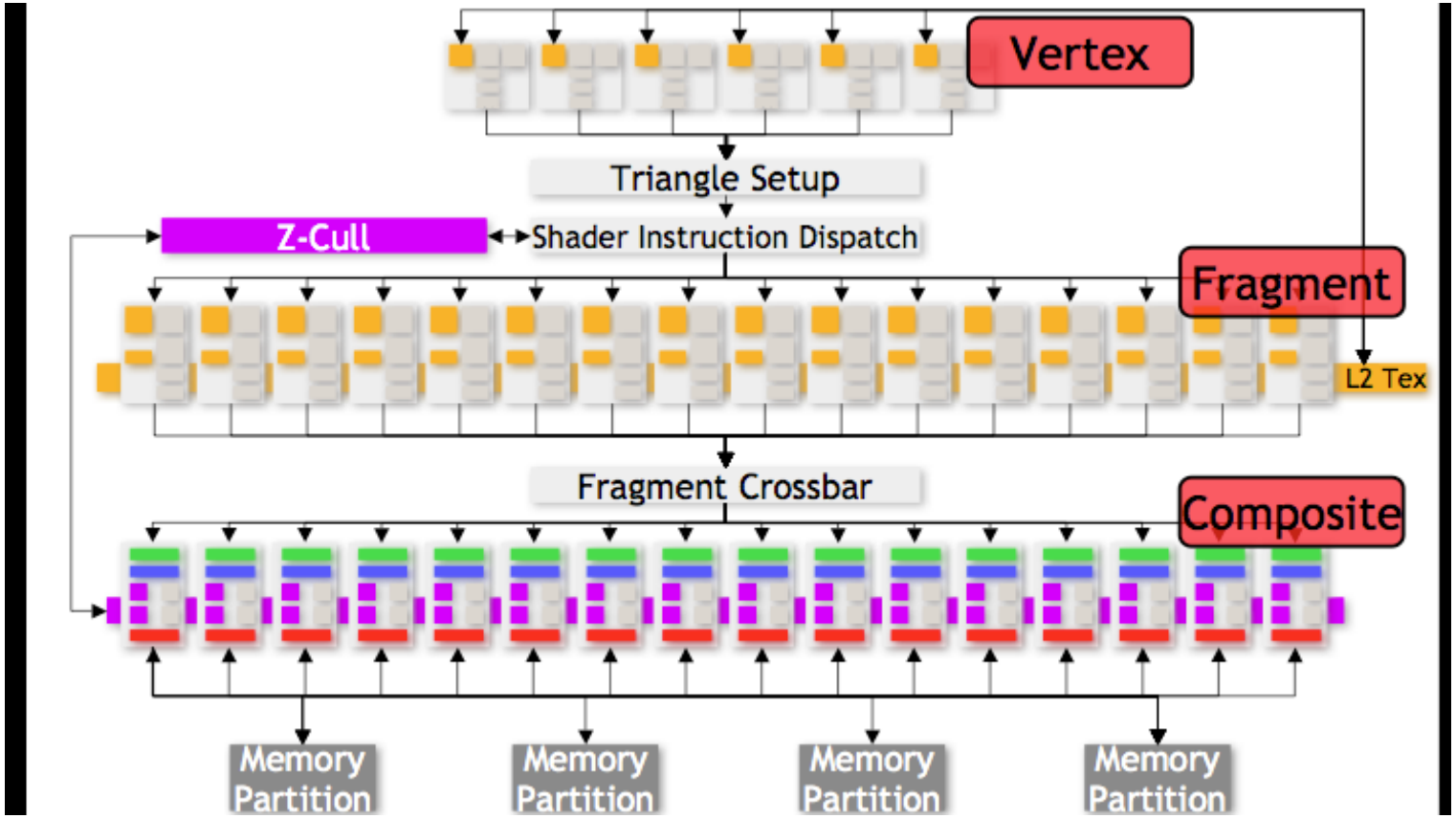
2.4 Bildrechner/GPU

Evolution der Graphikhardware (2008) [J. Owen, UC Davis]



2.4 Bildrechner/GPU

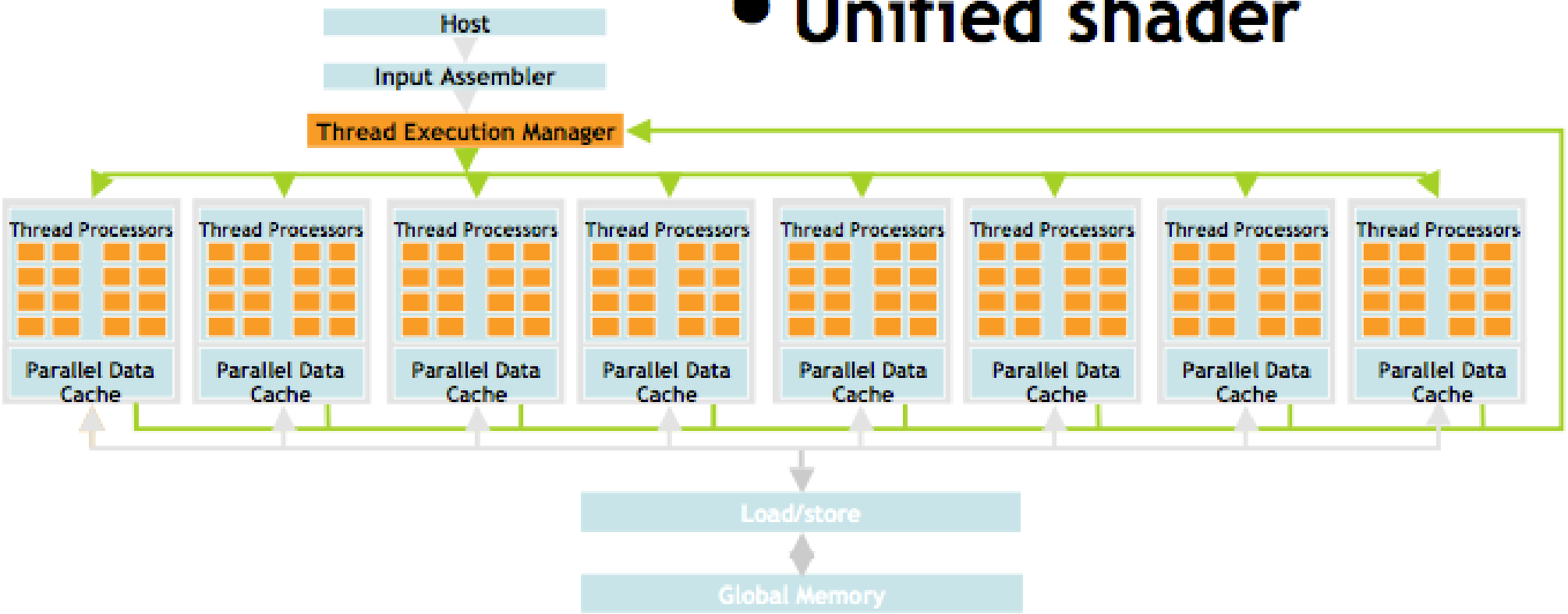
NVIDIA GeForce 6800 - on-chip (2004/5) [N. Triantos, NVIDIA]



2.4 Bildrechner/GPU

NVIDIA GeForce 8800 - on-chip [I. Buck, NVIDIA]

- **Unified shader**



2.4 Bildrechner/GPU

NVIDIA GeForce 2080RTX

2.4 Bildrechner/GPU

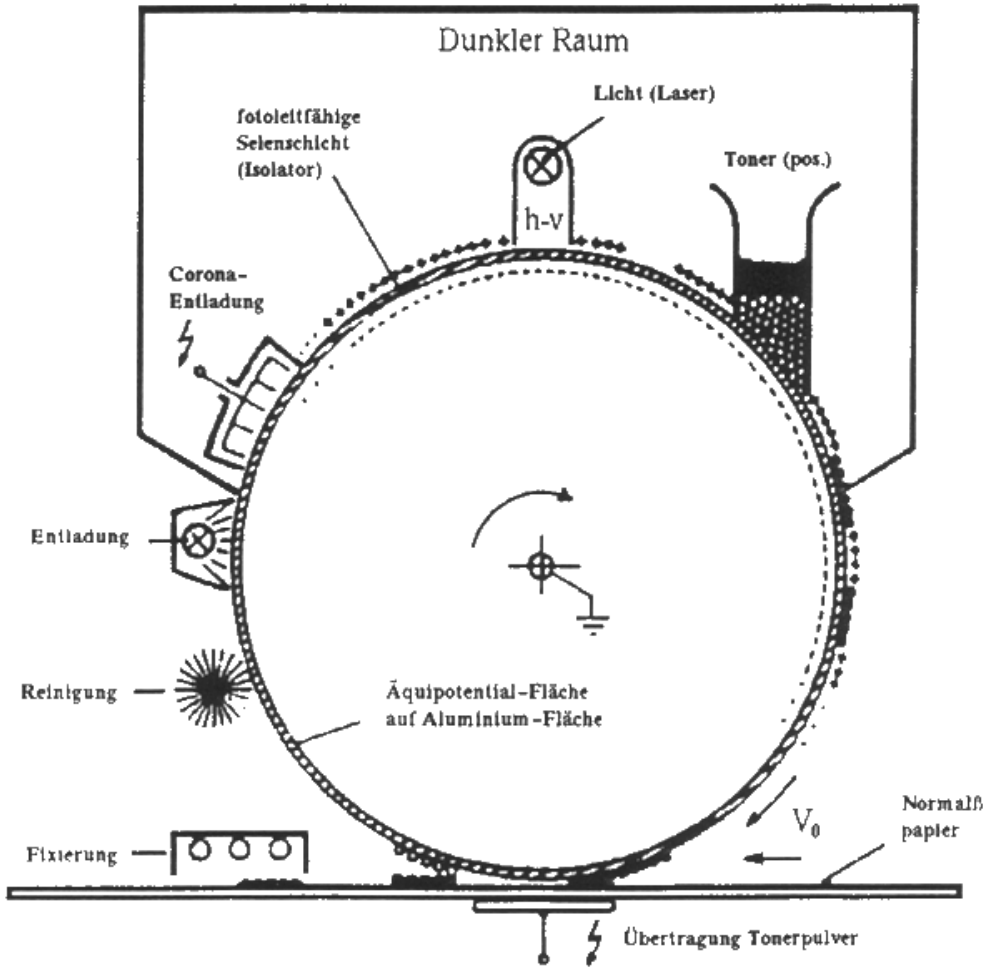
Taxonomie nach Akeley (GTXRD)

- G: Generation
- T: Traversal - Szenengraph
- X: Xformation - Geometrieverarbeitung
- R: Rasterung
- D: Display - Darstellung

- Aufteilung der Abschnitte in Software-Hardware
- GTXR-D - Nur Framebuffer (Bildspeicher, zB. VGA)
- GTX-RD – Rasterungs-unterstützung (zB. SGI O2)
- GT-XRD - T&L in Hardware (zB. SGI RE, GeForce GPUs)
- G-TXRD – Szenengraphtraver-sierung in Hardware (zB. E&S)

2.5 Weitere graphische Ausgabegeräte

Laserdrucker: Prinzip des xerographischen Druckverfahrens



2.6 Graphische Eingabegeräte

Touchscreens: Resistive Touchscreens

- Zwei leitfähige Schichten, die durch Abstandshalter getrennt sind
- Druck verbindet die beiden Schichten
- Gleichspannung entlang einer Schicht und Messen der Spannungsdifferenz zu den Rändern bestimmt die Position der Druckberührung
- Die beiden leitenden Schichten bestehen aus Indium-Zinn-Oxid, einem lichtdurchlässigem Halbleiter

2.6 Graphische Eingabegeräte

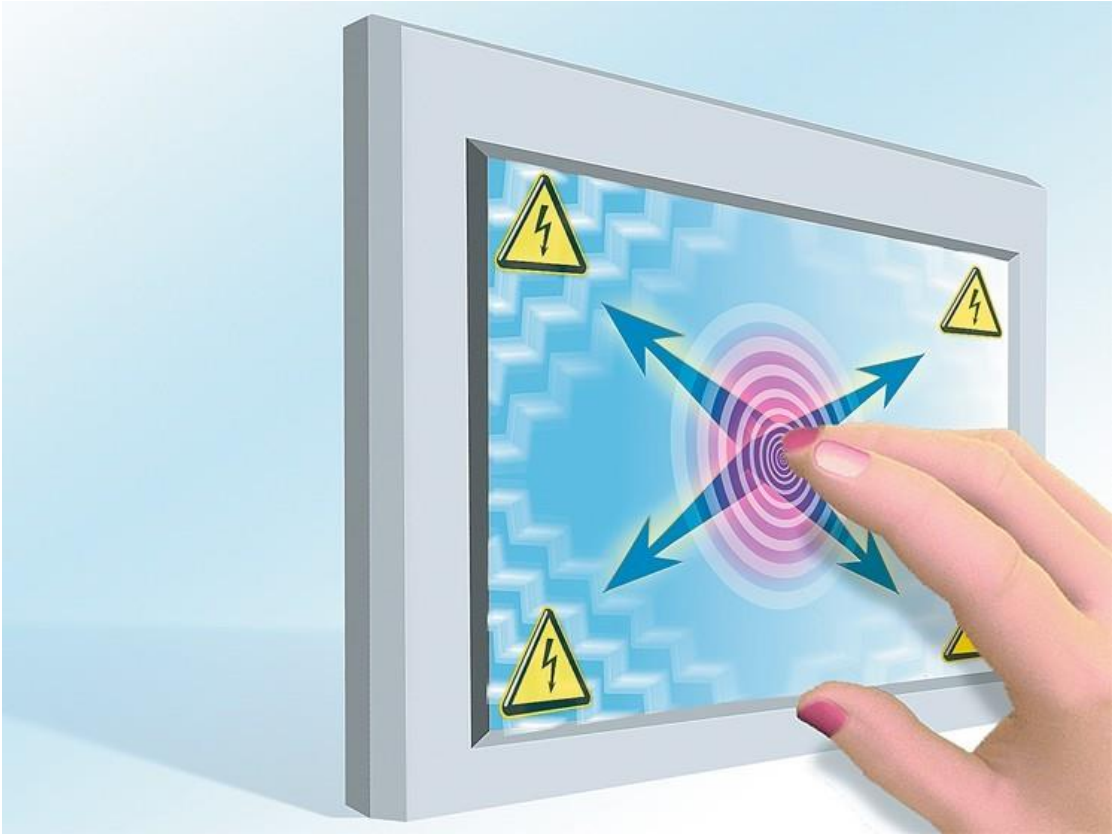
Touchscreens: Resistive Touchscreens

- Die obere Schicht ist an einer Folie angebracht, die sich verformen lässt
- Die untere Schicht ist an einer Glasplatte angebracht
- Vorteile:
  - Stiftbedienung
  - Handschuhe
- Nachteile
  - kein Multitouch
  - Verschleiß der Folie

2.6 Graphische Eingabegeräte

Touchscreens: Kapazitive Touchscreens

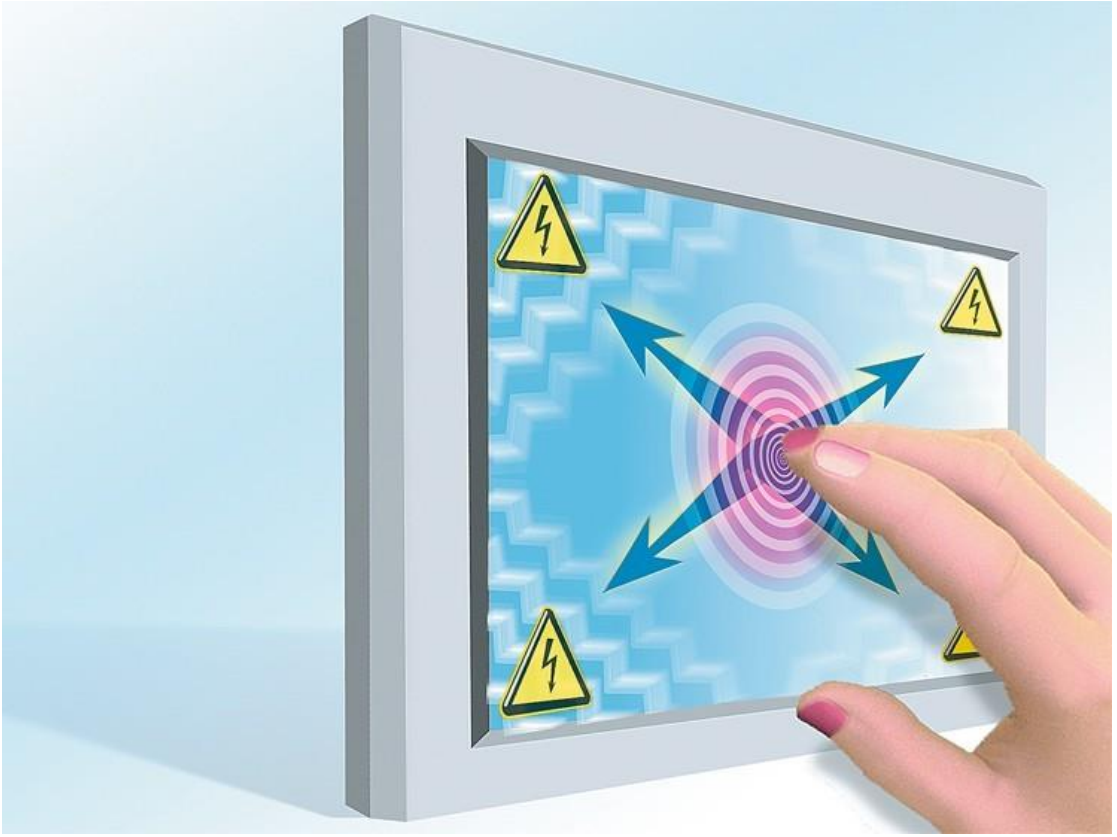
- Zwei Ebenen, die voneinander getrennt sind
- Auf diesen Ebenen sind leitfähige Muster (Streifen oder Rauten) befestigt, die einen Kondensator erzeugen
- Durch einen Finger entsteht ein geringer Ladungstransport, der die Kapazität des Kondensators reduziert
- Dies erzeugt in den Ecken schwache Ströme, aus denen die Position bestimmt wird



2.6 Graphische Eingabegeräte

Touchscreens: Kapazitive Touchscreens

- Vorteile:
  - Anbringen auf der Rückseite der Glasscheibe – kaum Verschleiß
  - Multitouch ist möglich
- Nachteile:
  - nur Hand oder spezielle Stifte
  - nur mit speziellen Handschuhen



2.6 Graphische Eingabegeräte

Indirekt graphische Eingabe

- 2D: Maus und Rollball
  - verwenden einen Cursor zur Positionierung
  - können zur graphischen Eingabe benutzt werden
- Insbesondere die Maus ist so zum geläufigsten graphischen Eingabegerät geworden
- Werden zunehmend von Touchdisplays verdrängt
- 3D: 3D-Maus
  - 6 Freiheitsgrade zum Verschieben
    - links, rechts
    - oben, unten
    - vor, zurück
  - 6 Freiheitsgrade zum Drehen
    - x-Achse: links, rechts
    - y-Achse: links, rechts
    - z-Achse: links, rechts
  - zusätzliche Funktionstasten

2.6 Graphische Eingabegeräte

Haptische Eingabegeräte

- Mit Hilfe haptischer Geräte kann der Benutzer die virtuelle Geometrie „fühlen“
- Sobald er mit seiner Hand oder mit einer Sonde die Geometrie berührt, übt das haptische Gerät eine entsprechende Kraft aus, die eine Durchdringung verhindert

2.6 Graphische Eingabegeräte

Bildgebende Eingabegeräte

- Können Bildinformationen direkt in den Rechner übertragen
  - Scanner
  - Digitale Kameras
  - Videorekorder mit A/D-Wandlern / Framegrabber

Quellen

- Computergraphik, Universität Leipzig (Prof. D. Bartz)
- Computergraphik, TU Kaiserslautern (Prof. H. Hagen)
- Graphische Datenverarbeitung I, Universität Tübingen (Prof. W. Straßer, Prof. A. Schilling)
- Graphische Datenverarbeitung I, TU Berlin (Prof. M. Alexa)
- GPU-Programmierung, Universität Stuttgart (Prof. D. Weiskopf)
- www.hardware-info.com
- www.gpgpu.org
- www.opengl.org
- www.wikipedia.org

UNIVERSITÄT LEIPZIG

Rasterung

COMPUTERGRAPHIK

Inhaltsverzeichnis

3 Rasterung

3.1 Rasterung von Geraden DDA

3.2 Rasterung von Geraden Bresenham

3.3 Rasterung von Kreisen

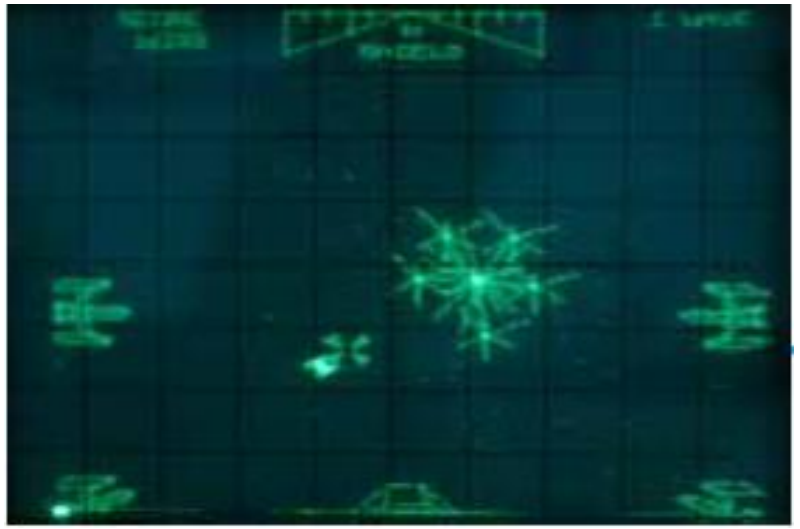
3.4 Füllalgorithmen

3.5 Aliasing

3. Rasterung

Darstellungsmöglichkeiten

- Vektordarstellung
  - Plotter, Elektronenstrahl (Oszilloskop)
  - Einzelne Linien (Vektoren) werden gezeichnet.
  - Bild setzt sich aus Linien zusammen.
  - Eingeschränkte Darstellungsmöglichkeiten



Darstellungsmöglichkeiten

- Rasterdisplays
  - Bild wird in Bildpunkte diskretisiert.
  - Erfordert Framebuffer (Bildspeicher)
    - Speichert die Bitmap des angezeigten Bildes
    - Digitale Kopie des Monitorbildes
    - Heutzutage doppelt oder dreifach gepuffert

Speicherbedarf:

1024 × 768 × TrueColor ≈ 2,25 MB

1920 × 1200 × TrueColor ≈ 6,6 MB

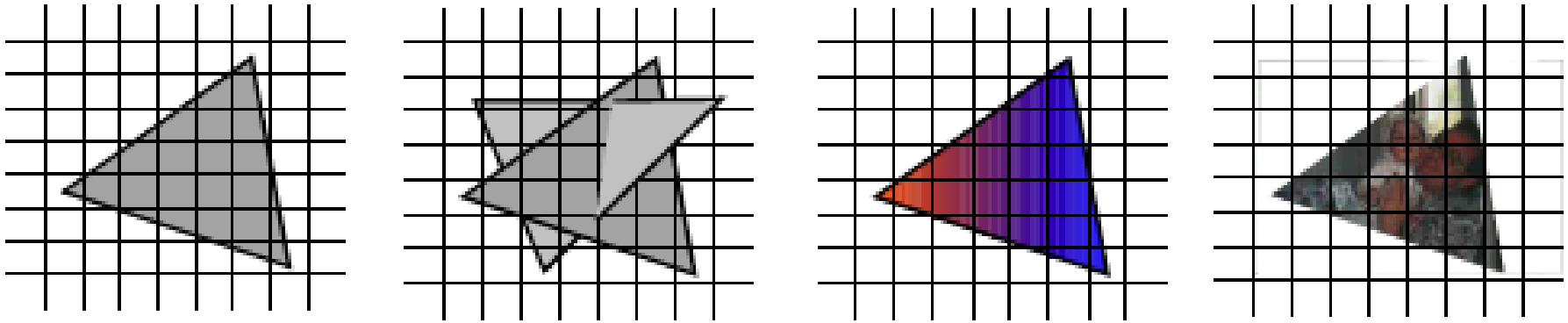
2560 × 1600 × TrueColor ≈ 12 MB

Truecolor:

- 24 Bit
- Je ein Byte (8 Bit) für R, G und B
- 2^24 = 16.777.216

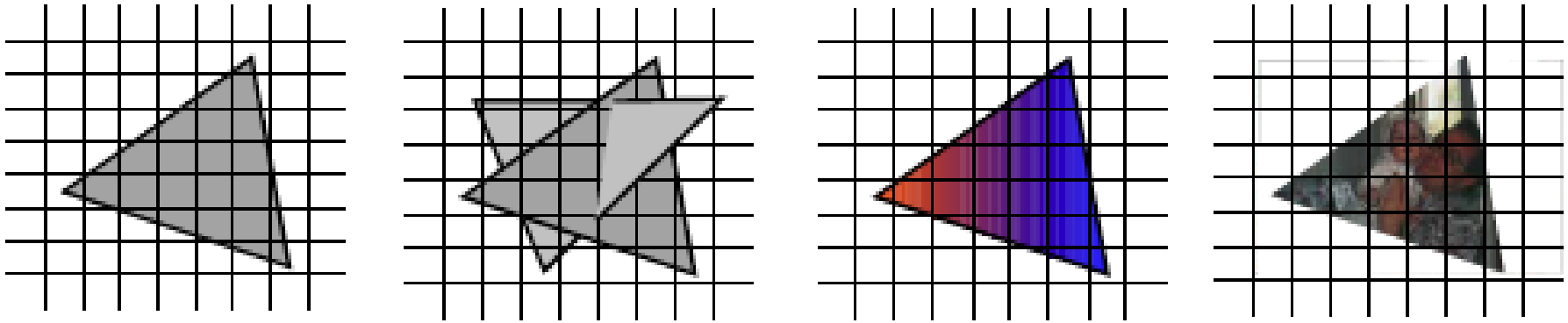
3. Rasterung

- Die dominierenden Rasterbildschirme erfordern die „Zerlegung“ aller darzustellenden geometrischen Objekte in Bildschirmpunkte.
- Dieser Prozess wird auch als Rasterung bezeichnet.
- Dies ist die Aufgabe der Bilderzeugungseinheit DPU (Display Processing Unit) des Bildrechners: Rastereinheit/Rasterprozessor



3. Rasterung

- Wir beschäftigen uns näher mit:
  - Rasterung von Geraden, Kreisen, Ellipsen, Polygonen
  - Antialiasing von Linien und Polygonen



3.1 Rasterung von Geraden

Problemstellung

– Darstellung einer Linie auf einem Rasterbildschirm erfordert die Bestimmung der „am besten passenden“ Punkte im Raster bzw. Gitter (geeignete ganzzahlige Rundung).

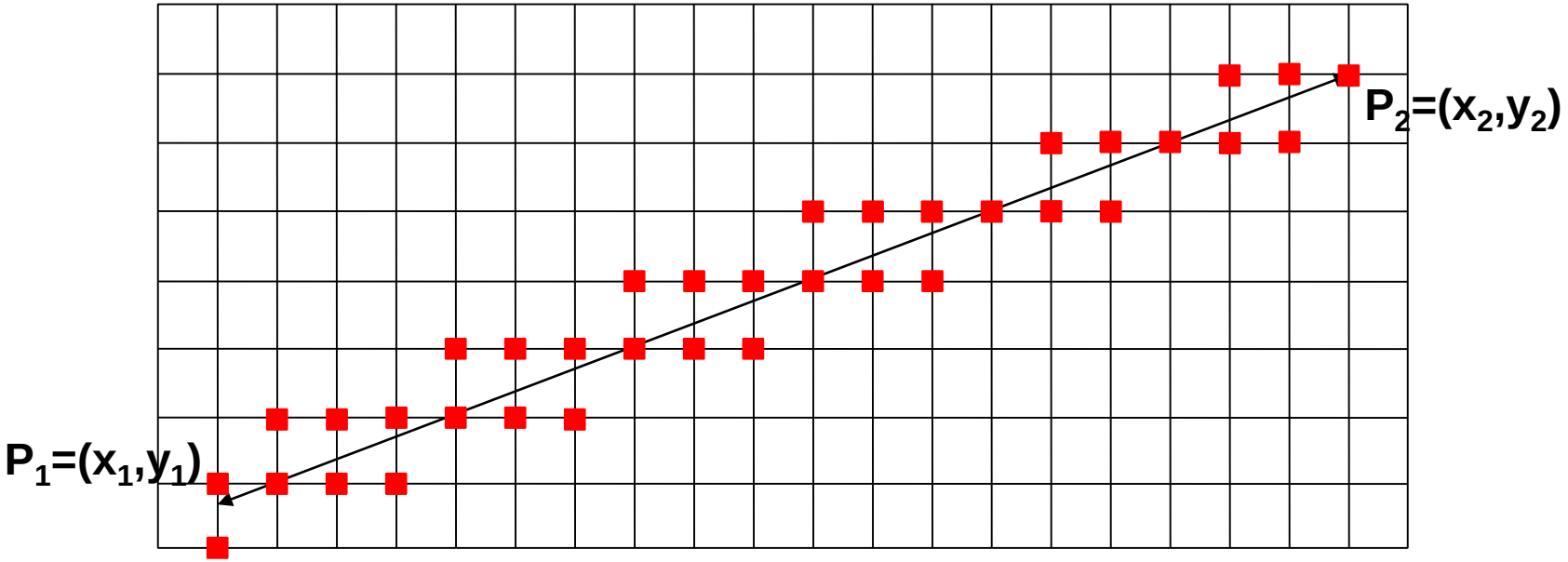


Abb. 2.1: Mögliche Rasterkandidaten

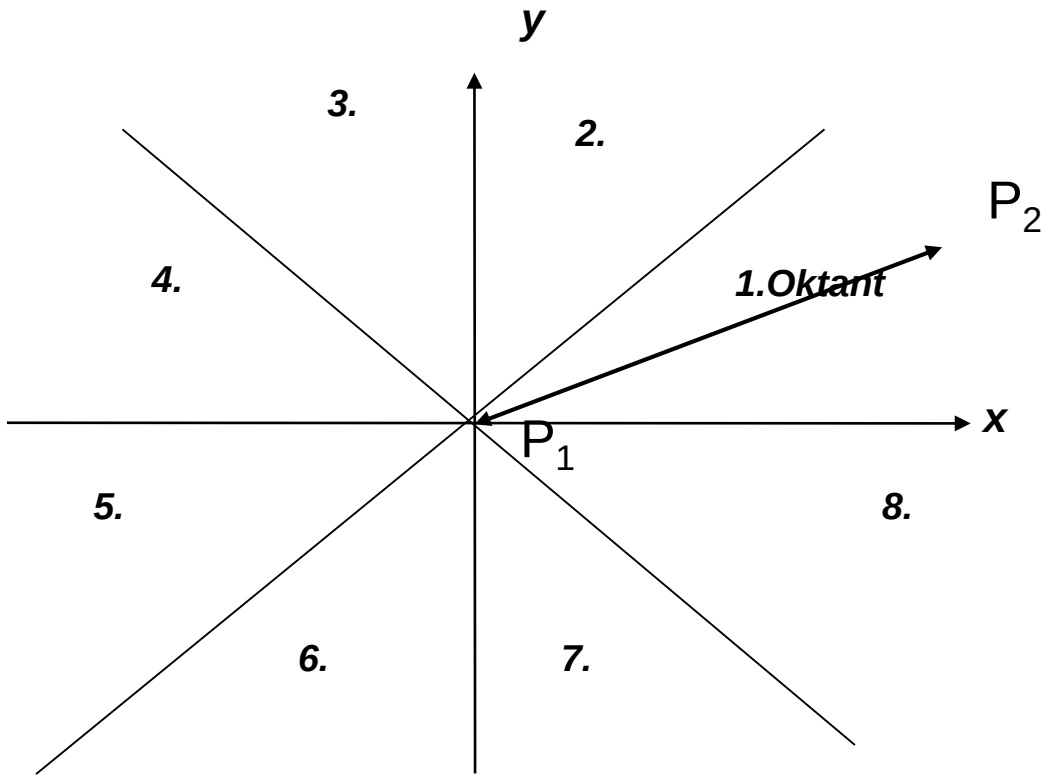
3.1 Rasterung von Geraden

Anforderungen

- Geraden
  - sollen gerade erscheinen
  - sollen gleichmäßig hell erscheinen
  - sollen schnell gezeichnet werden
- Algorithmus
  - soll leicht in Hardware implementierbar sein

3.1 Rasterung von Geraden

- Fallunterscheidung in Oktanten (nach Steigung)
- Hier:
  - Ursprung des zugeordneten Koordinatensystems in P_1
  - 1. Oktant
  - P_1 und P_2 auf Rastergitter

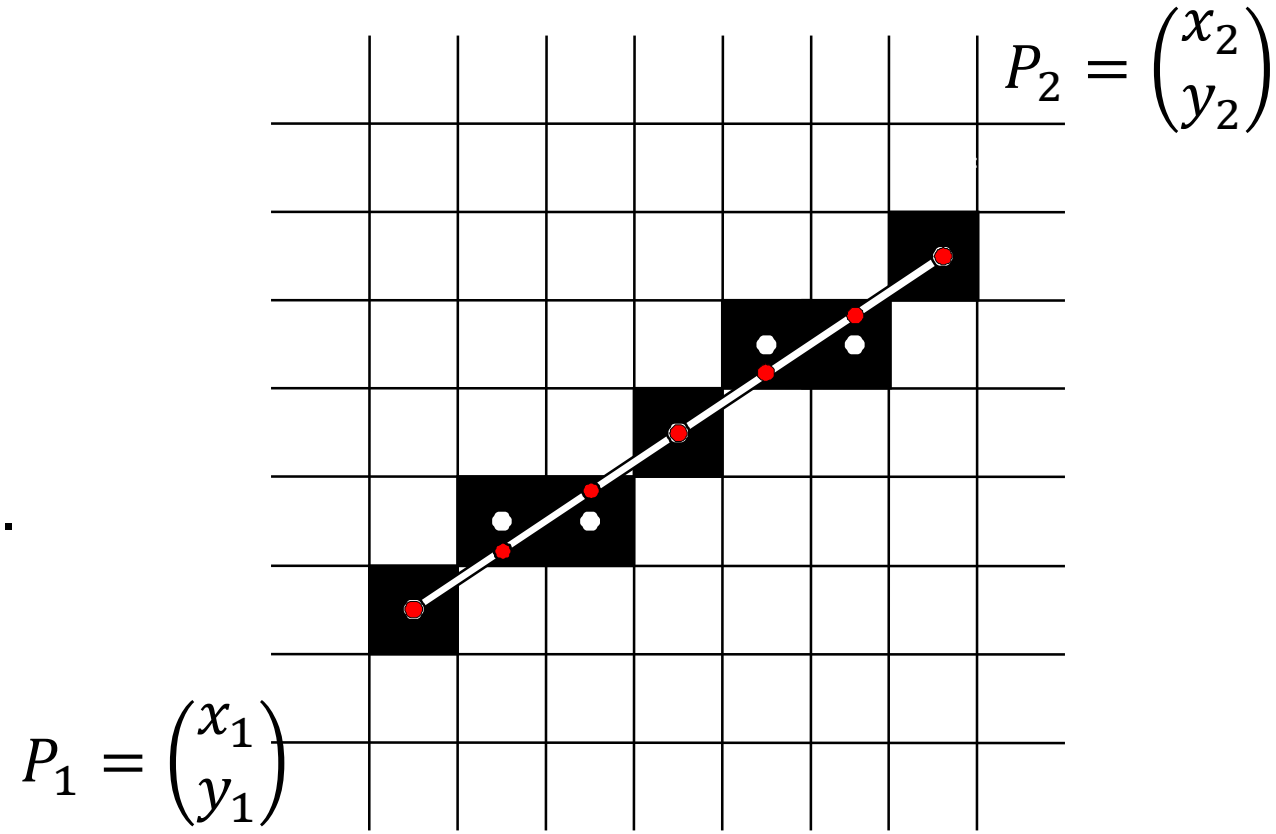


3.1 Rasterung von Geraden: DDA

DDA-Algorithmus für Geraden [1930er]

- DDA
  - Digital Differential Analyzer
  - Digitaler Integrierer

- Ein DDA-Algorithmus generiert eine Kurve (nicht nur Geraden) aus einer beschreibenden Differentialgleichung.



3.1 Rasterung von Geraden: DDA

Geradengleichung: $y = ax + b$

$\Delta x =$

$\Delta y =$

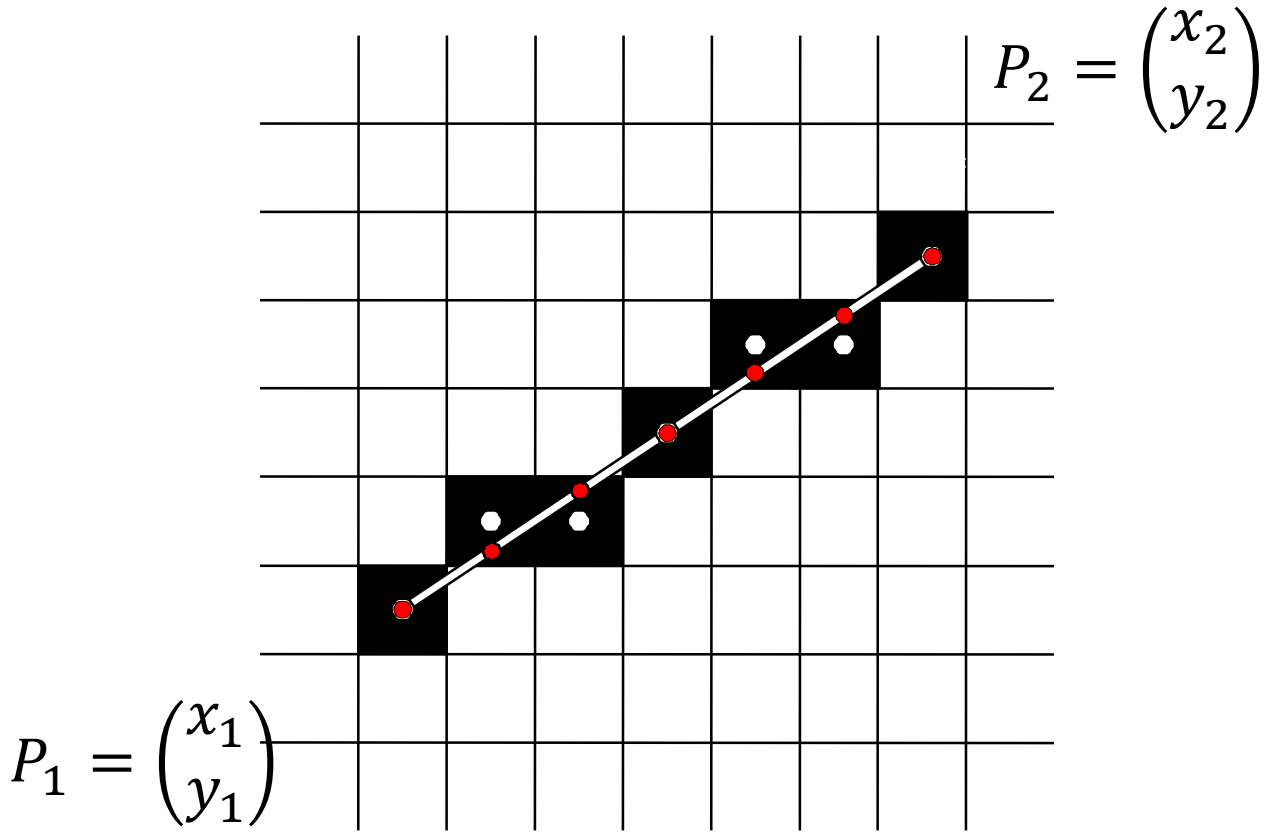
$a \quad =$

$b \quad =$

Direkte Berechnung: $\forall i = 1, \ldots, \Delta x$:

$x_{i+1} =$

$y_{i+1} =$



3.1 Rasterung von Geraden: DDA

Geradengleichung: $y = ax + b$

$\Delta x = x_2 - x_1$

$\Delta y = y_2 - y_1$

$a \quad = \frac{\Delta y}{\Delta x}, 0 \leq a \leq 1$

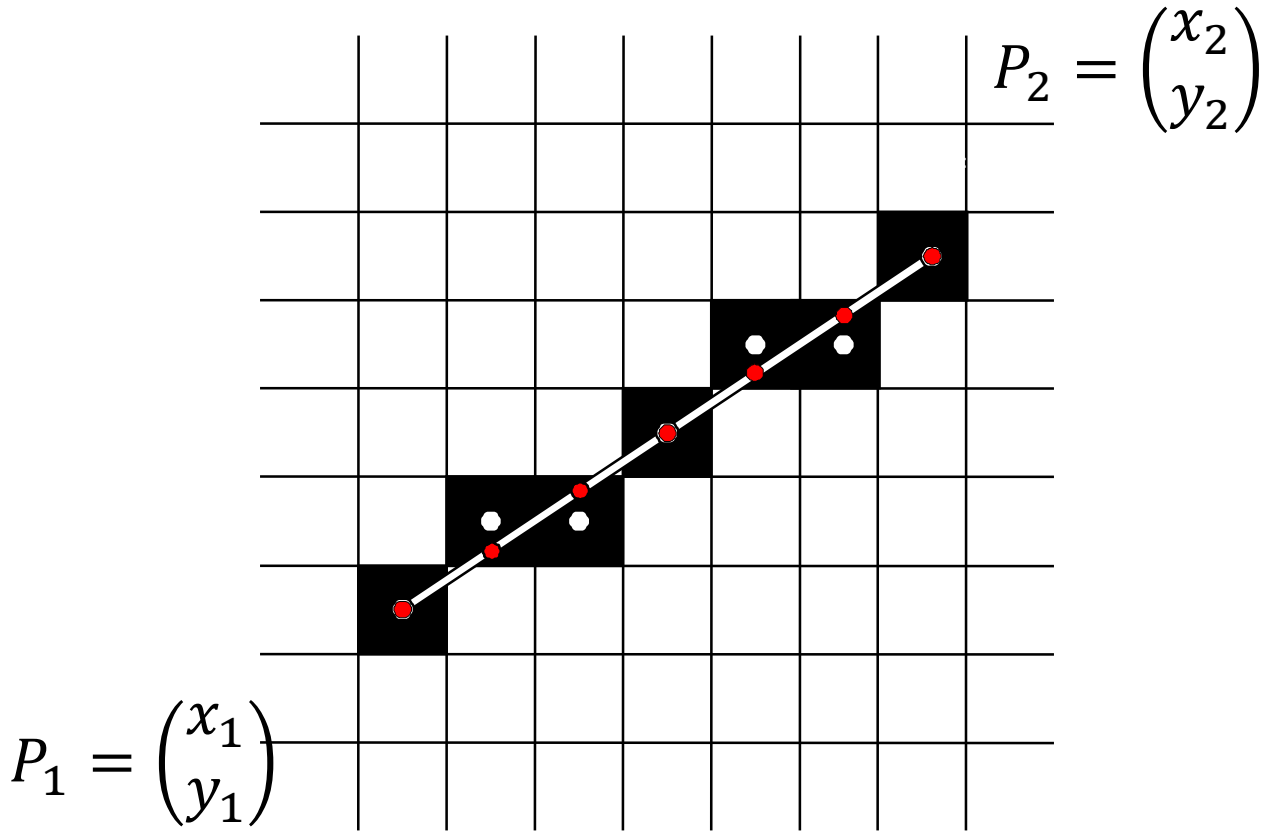
$b \quad = ax_1 - y_1$

Direkte Berechnung: $\forall i = 1, \ldots, \Delta x$:

$x_{i+1} = x_i + 1$

$y_{i+1} = ax_{i+1} + b$

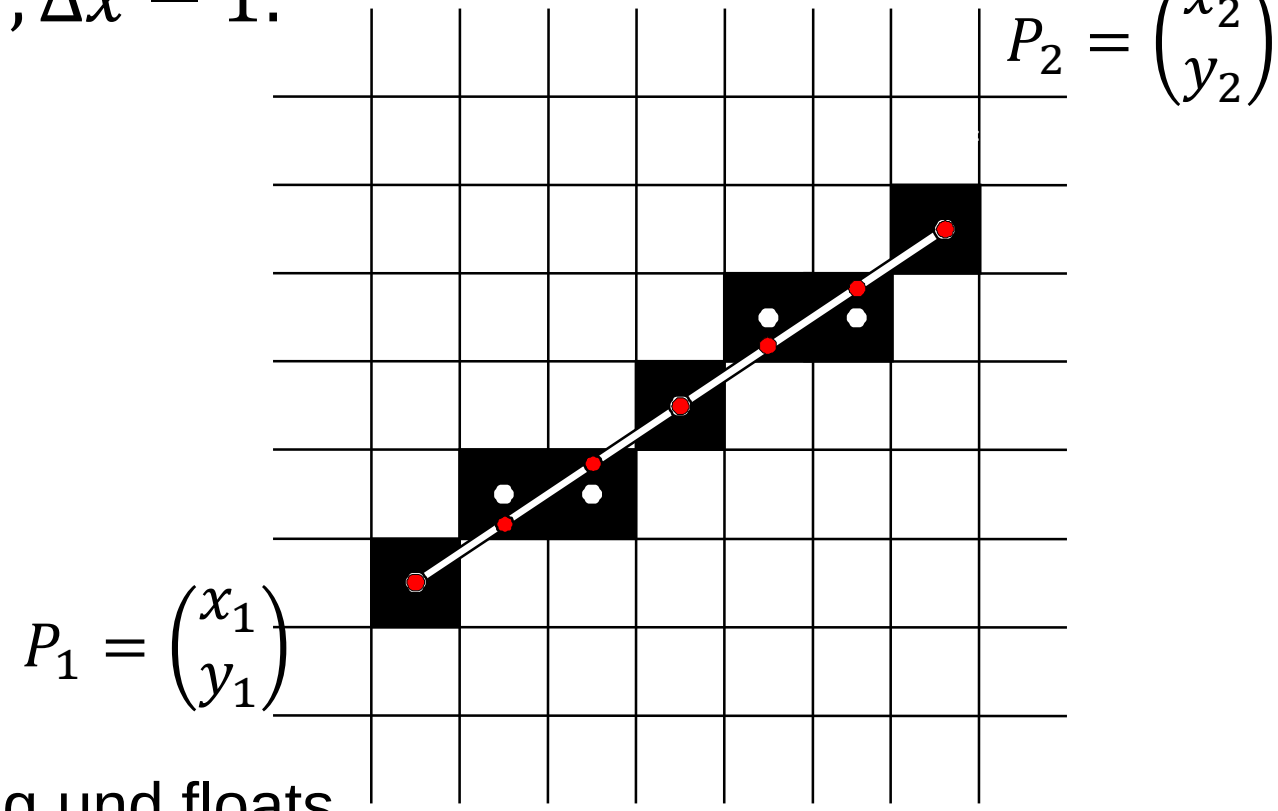
Plus Rundung



3.1 Rasterung von Geraden: DDA

Inkrementelle Berechnung: $\forall i = 0, \ldots, \Delta x - 1$:

y_{i+1}



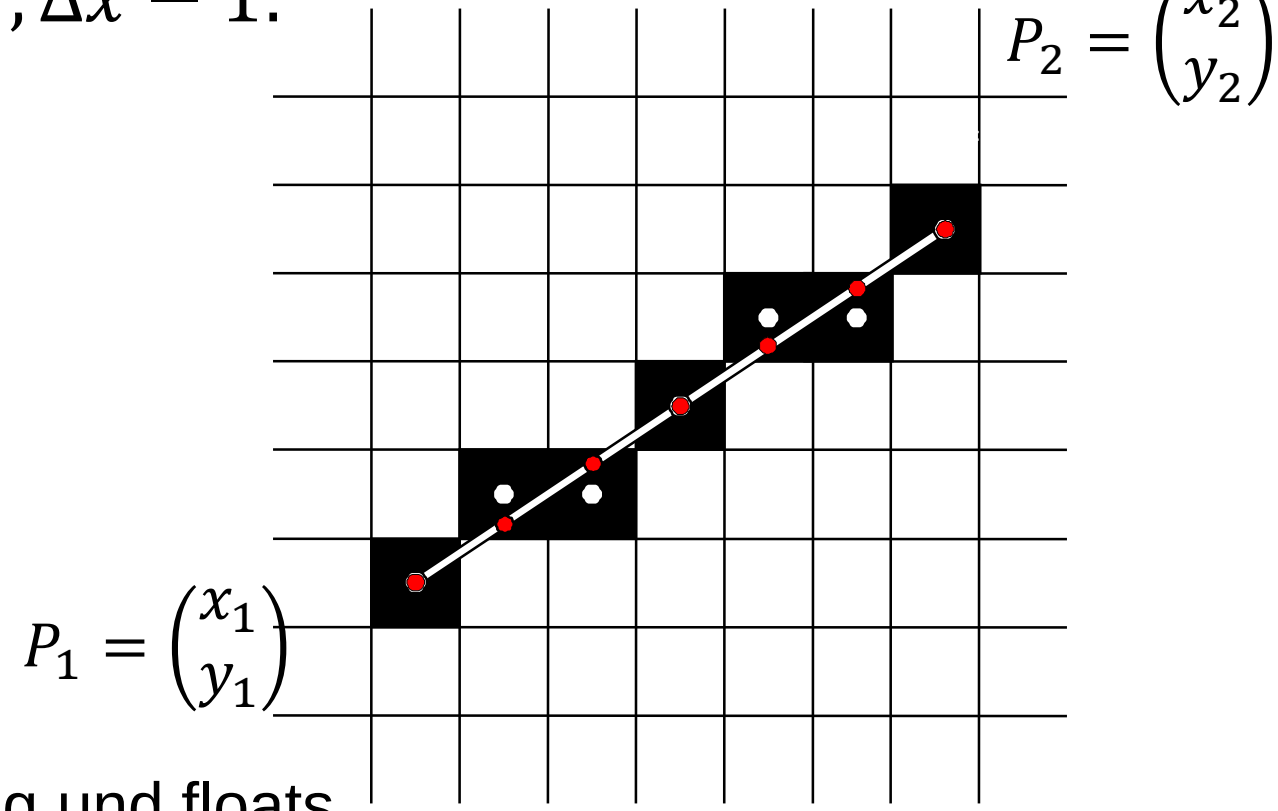
Vorteil: schneller

Nachteil beide: ungenau durch Rundung und floats

3.1 Rasterung von Geraden: DDA

Inkrementelle Berechnung: $\forall i = 0, \ldots, \Delta x - 1$:

$$
\begin{aligned}
y_{i+1} &= a x_{i+1} + b \\
&= a(x_i + 1) + b \\
&= a x_i + b + a \\
&= y_i + a \\
&= y_i + \frac{\Delta y}{\Delta x}
\end{aligned}
$$



Vorteil: schneller

Nachteil beide: ungenau durch Rundung und floats

3.2 Rasterung von Geraden: Bresenham

Bresenham-Algorithmus für Geraden [1962]

- Idee:
  - Abhängig von der Steigung der Geraden wird die x- oder y-Koordinate immer um genau eine Einheit geändert
  - Die andere Koordinate wird entweder nicht oder ebenfalls um eine Einheit geändert
  - Fallunterscheidung nach der kleineren Abweichung der Geraden zum nächsten Gitterpunkt in Koordinatenrichtung

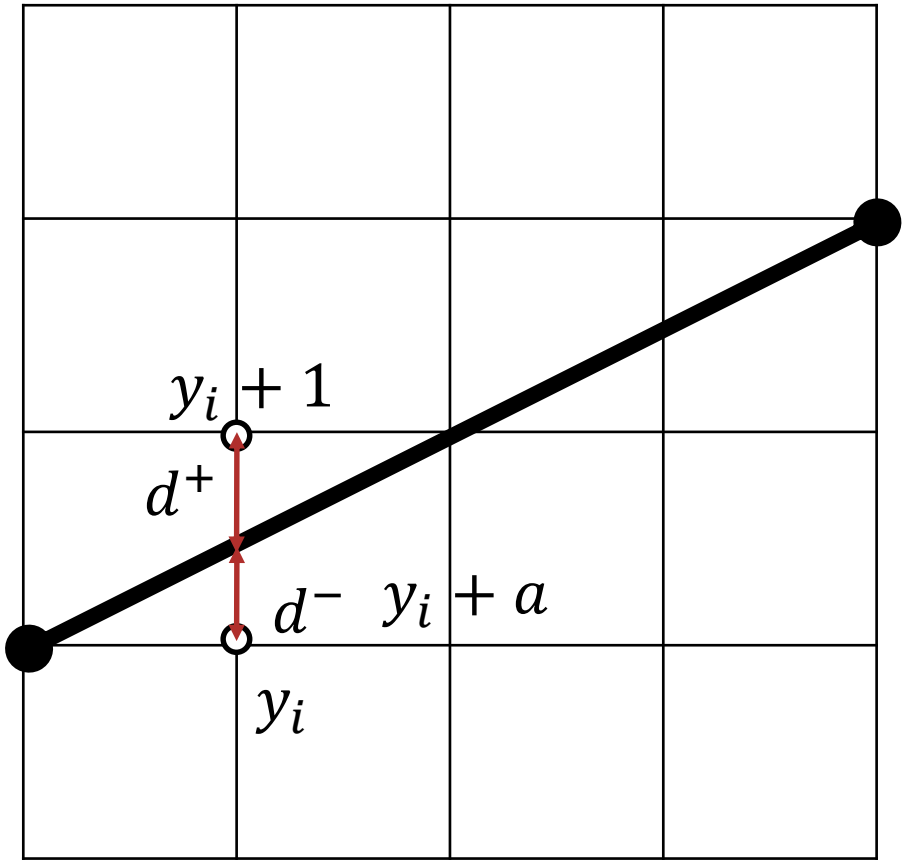
3.2 Rasterung von Geraden: Bresenham

– Entweder Punkt 1 oder Punkt 2 wird gezeichnet, je nachdem, welcher näher zur Geraden liegt

d^-

d^+

– Gehe nach oben, wenn $a \cdot \; < d^-$



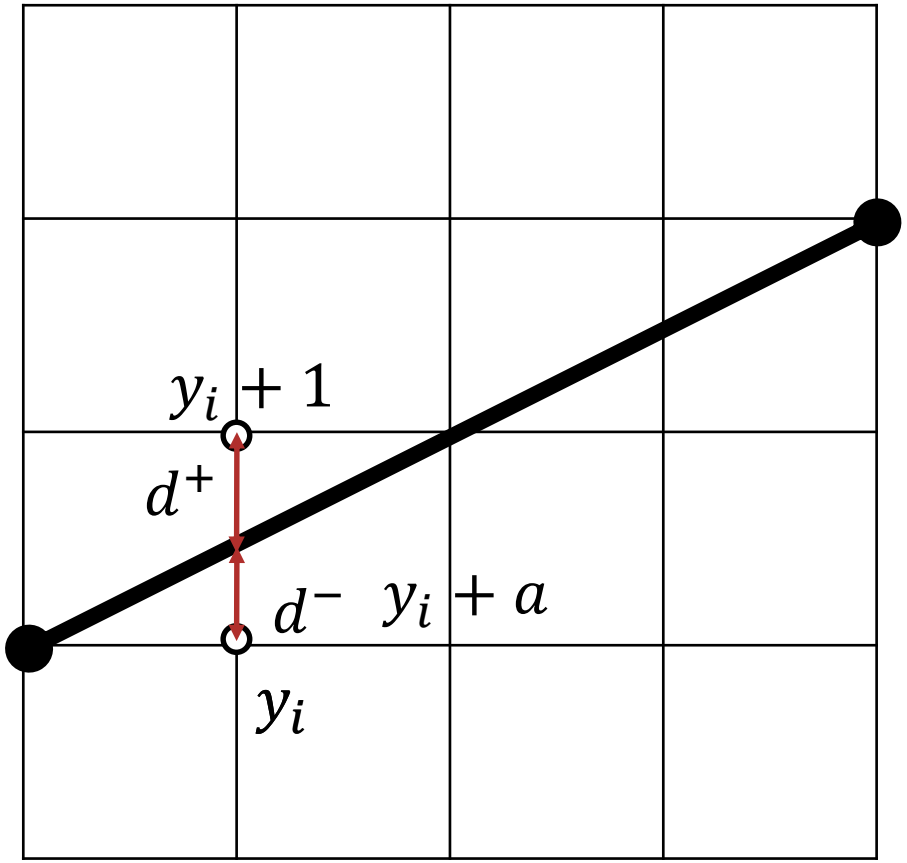
3.2 Rasterung von Geraden: Bresenham

- Entweder Punkt 1 oder Punkt 2 wird gezeichnet, je nachdem, welcher näher zur Geraden liegt

$$d^- = y_i + a - y_i$$
$$= a$$

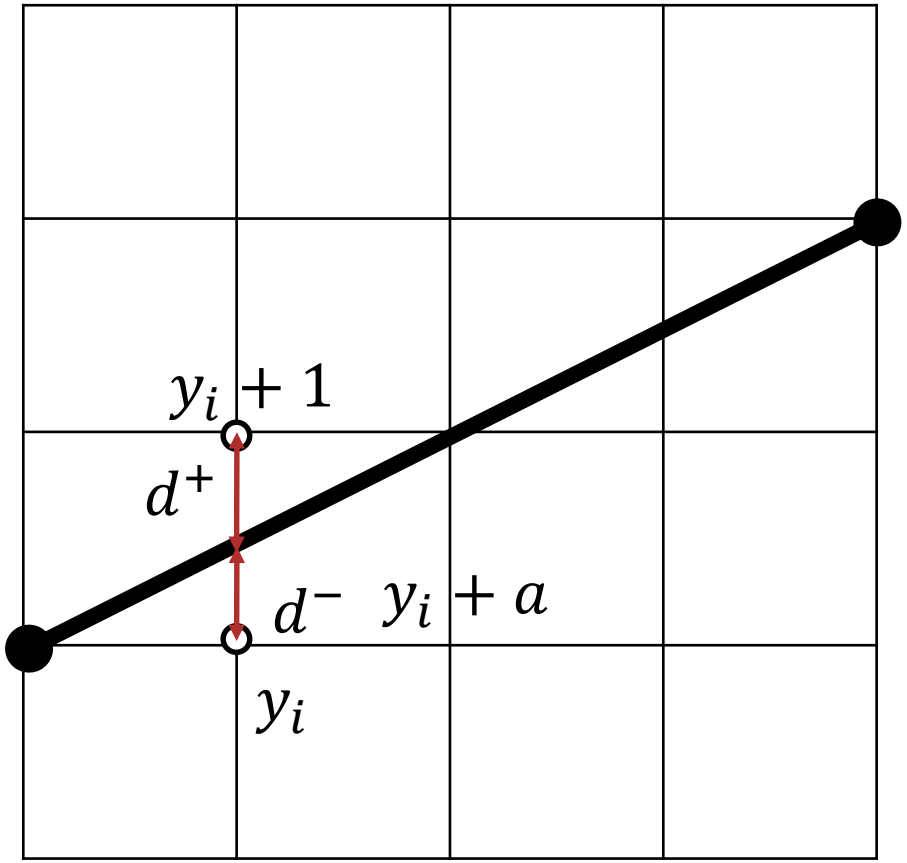
$$d^+ = y_i + 1 - (y_i + a)$$
$$= 1 - a$$

- Gehe nach oben, wenn $d^+ < d^-$



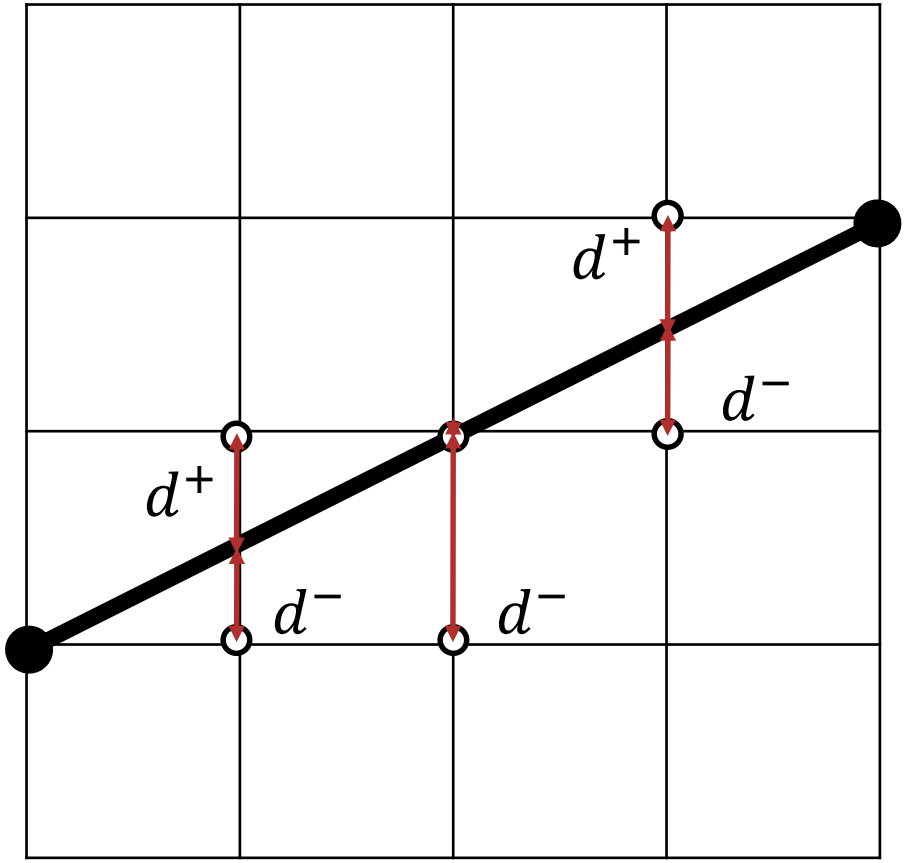
3.2 Rasterung von Geraden: Bresenham

- Zur Realisierung wird eine **Entscheidungsgröße E** eingeführt:
$$E := d^- - d^+$$
- Gehe nach oben, wenn
$$d^+ < d^- \Leftrightarrow E > 0$$
- Das **Vorzeichen von E** dient dann als Kriterium für die Rundung auf den nächsten Rasterpunkt:
  - $E \leq 0: x++$
  - $E > 0: x++, y++$



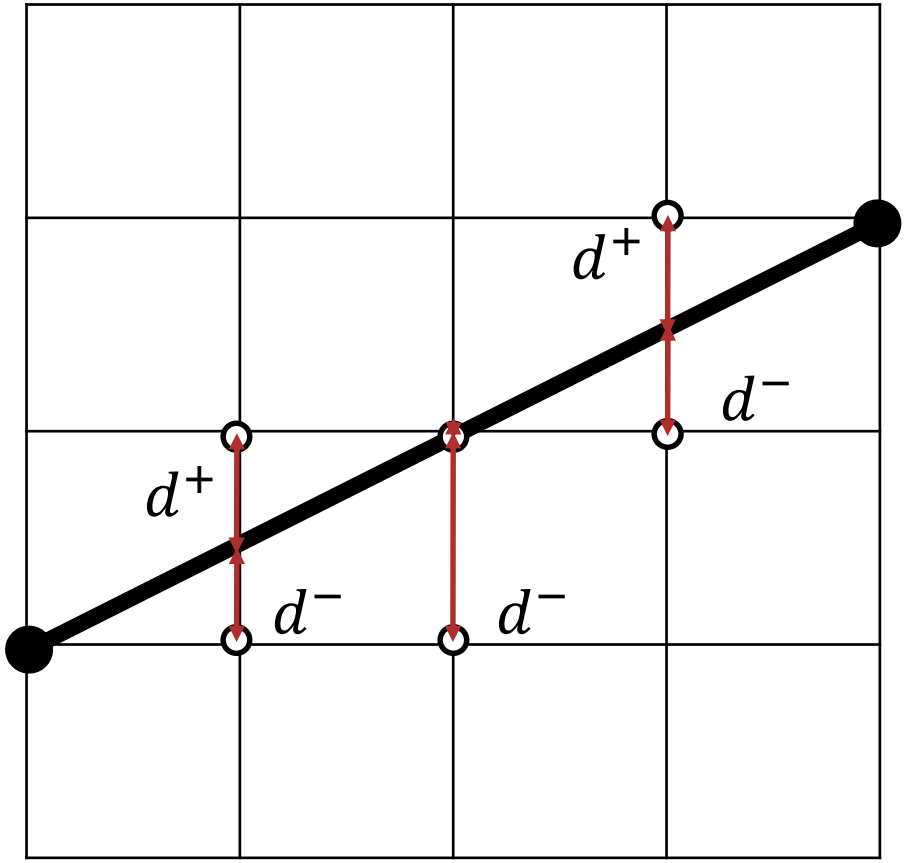
3.2 Rasterung von Geraden: Bresenham

- Zur schnelleren Berechnung, kann E **inkrementell** berechnet werden
- Das **Vorzeichen von E** dient dann als Kriterium für die Rundung auf den nächsten Rasterpunkt und die Aktualisierung von E:
  - $E \leq 0: x++,$
    $d^+ \mathrel{+}=$
    $d^- \mathrel{-}=$
    $E \mathrel{+}=$
  - $E > 0: x++, y++,$
    $d^+ \mathrel{+}=$
    $d^- \mathrel{-}=$
    $E \mathrel{+}=$



3.2 Rasterung von Geraden: Bresenham

- Zur schnelleren Berechnung, kann E **inkrementell** berechnet werden
- Das **Vorzeichen von E** dient dann als Kriterium für die Rundung auf den nächsten Rasterpunkt und die Aktualisierung von E:
  - $E \leq 0: x++,$
    $d^+ -= a$
    $d^- += a$
    $E \ += 2a$
  - $E > 0: x++, y++,$
    $d^+ += 1 - a$
    $d^- -= 1 - a$
    $E \ += 2a - 2$



3.2 Rasterung von Geraden: Bresenham

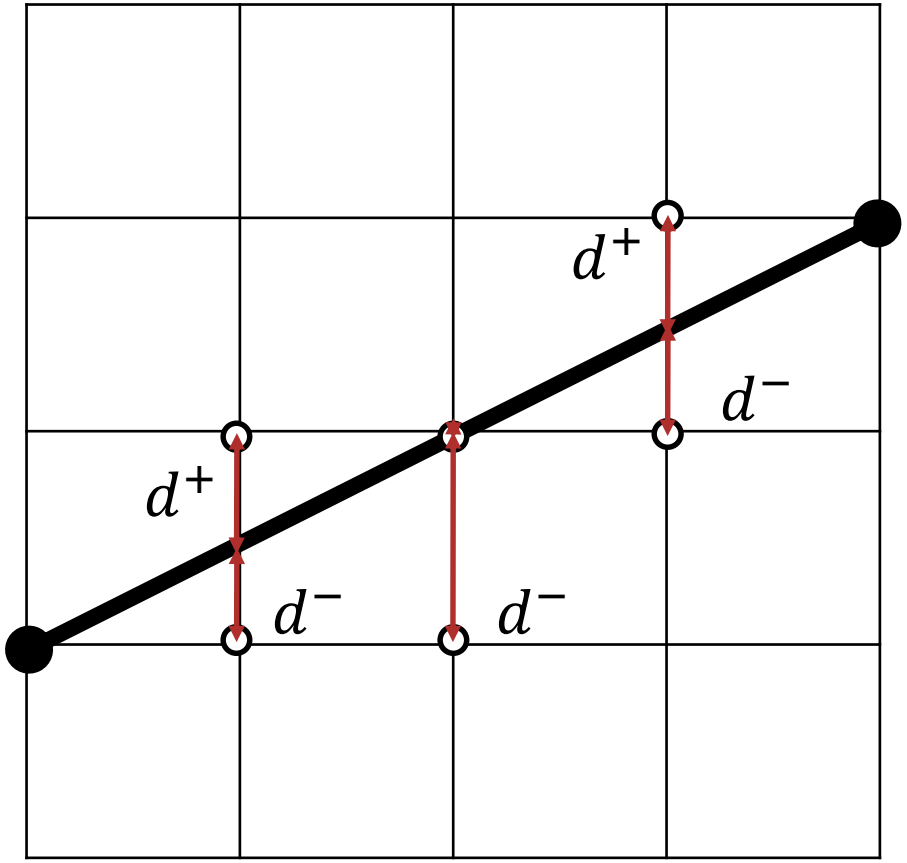
- Zur Vermeidung der Division wird **Entscheidungsgröße e** eingeführt:

$$e := \Delta x E$$

- Gehe nach oben, wenn

$$E > 0 \Leftrightarrow e > 0$$

- Das **Vorzeichen von e** dient dann als Kriterium für die Rundung auf den nächsten Rasterpunkt:
  - $e \leq 0: x++,$
  - $e > 0: x++, y++$



3.2 Rasterung von Geraden: Bresenham

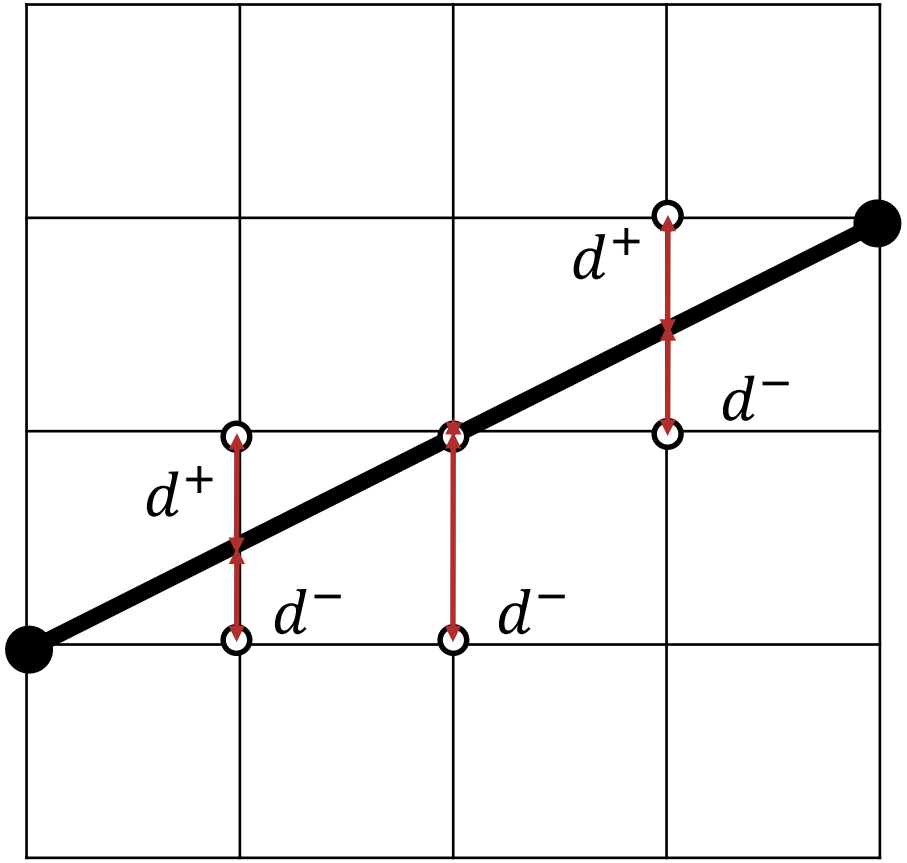
- Zur Vermeidung der Division wird **Entscheidungsgröße e** eingeführt:

$$e := \Delta x E$$

- Gehe nach oben, wenn

$$E > 0 \Leftrightarrow e > 0$$

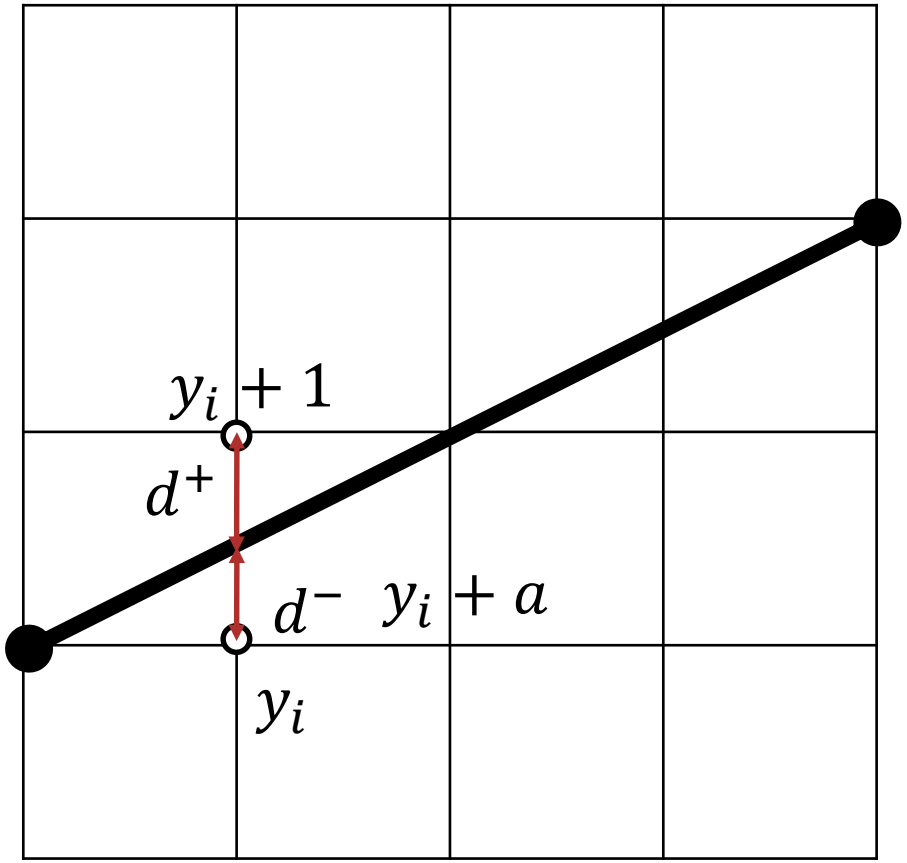
- Das **Vorzeichen von e** dient dann als Kriterium für die Rundung auf den nächsten Rasterpunkt:
  - $e \leq 0: x++, \qquad e \mathrel{+}= 2\Delta y$
  - $e > 0: x++, y++, e \mathrel{+}= 2\Delta y - 2\Delta x$



3.2 Rasterung von Geraden: Bresenham

– Startwerte:

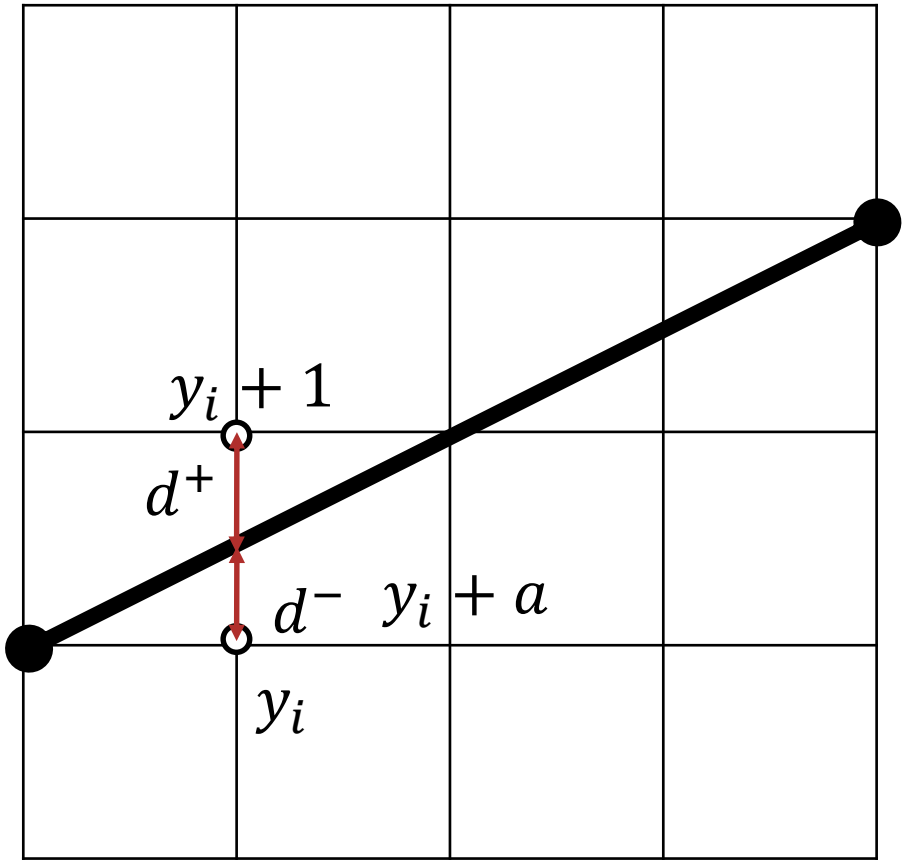
$$e_1$$



3.2 Rasterung von Geraden: Bresenham

– Startwerte:

$$
\begin{aligned}
e_1 &= \Delta x E_1 \\
&= \Delta x (d_1^- - d_1^+) \\
&= \Delta x (a - (1 - a)) \\
&= \Delta x (2a - 1) \\
&= \Delta x \left(2 \frac{\Delta y}{\Delta x} - 1\right) \\
&= 2\Delta y - \Delta x
\end{aligned}
$$



3.2 Rasterung von Geraden: Bresenham

```
// (x1, y1), (x2, y2) Ganzzahlig
// x1 < x2, y1 < y2
x = x1; y = y1;
dx = x2 - x1; dy = y2 - y1;
e = 2 * dy - dx; // Initialisierung
for(i = 1; i <= dx; i++){
  // Schleife fuer x
  plot(x, y);
  if (e >= 0) {
        // oberen Punkt Zeichnen (y erhoehen)
    ++y;  -- für 5,6,7,8
    e -= 2 * dx;
  }
  ++x;  -- für 3,4,5,6
  e += 2 * dy;
}
plot(x, y);
```

Schleife vertauschen: 2,3,6,7(alle dx zu dy und umgekehrt)

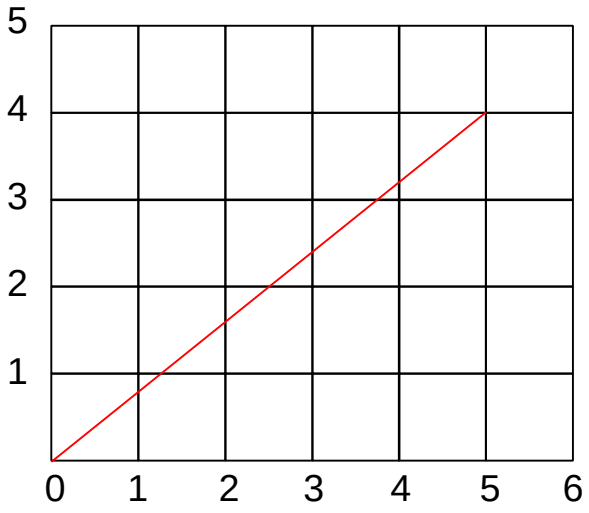
- Erster Oktant
- Ausschließlich ganzzahlige Operanden

3.2 Rasterung von Geraden: Bresenham

```
// (x1, y1), (x2, y2) Ganzzahlig
// x1 < x2, y1 < y2
x = x1; y = y1;
dx = x2 - x1; dy = y2 - y1;
e = 2 * dy - dx; // Initialisierung
for(i = 1; i <= dx; i++){
  // Schleife fuer x
  plot(x, y);
  if (e >= 0) {
        // oberen Punkt Zeichnen (y erhoehen)
    ++y;
    e -= 2 * dx;
  }
  ++x;
  e += 2 * dy;
}
plot(x, y);
```

Beispiel:

- P1=(0, 0)
- P2=(5, 4)



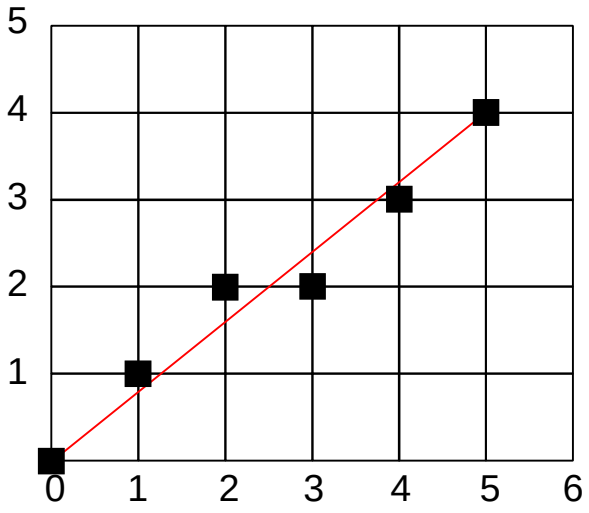
x	y	e	i	plot

3.2 Rasterung von Geraden: Bresenham

```
// (x1, y1), (x2, y2) Ganzzahlig
// x1 < x2, y1 < y2
x = x1; y = y1;
dx = x2 - x1; dy = y2 - y1;
e = 2 * dy - dx; // Initialisierung
for(i = 1; i <= dx; i++){
  // Schleife fuer x
  plot(x, y);
  if (e >= 0) {
        // oberen Punkt Zeichnen (y erhoehen)
    ++y;
    e -= 2 * dx;
  }
  ++x;
  e += 2 * dy;
}
plot(x, y);
```

Beispiel:

- P1=(0, 0)
- P2=(5, 4)

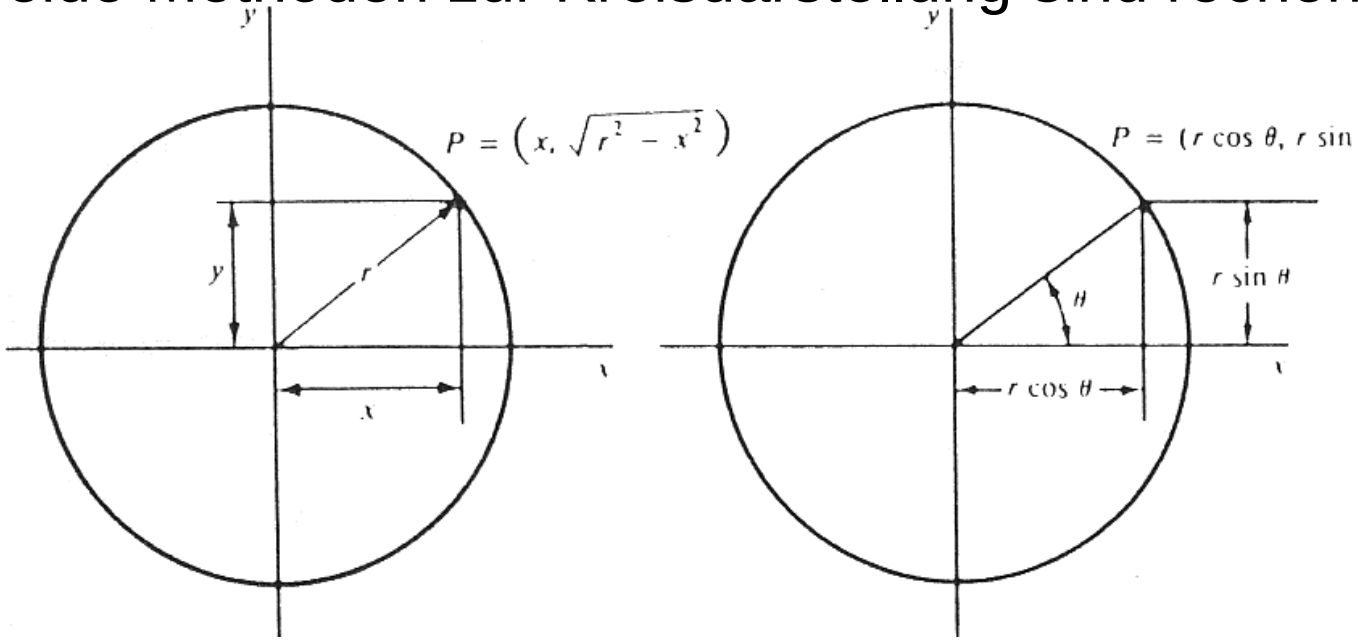


x	y	e	i	plot
0	0	3	1	(0, 0)
	1	-7		
1		1	2	(1, 1)
	2	-9		
2		-1	3	(2, 2)
3		7	4	(3, 2)
	3	-3		
4		5	5	(4, 3)
	4	-5		
5		3	6	
				(5, 4)

3.3 Rasterung von Kreisen

Darstellung eines Kreises mit Mittelpunkt (x_M, y_M) und Radius r

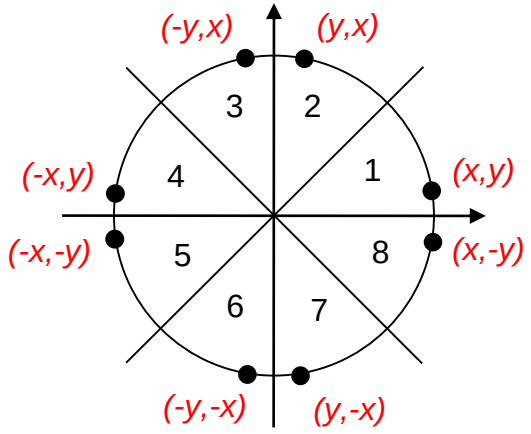
- Implizit: $f(x,y) = (x - x_M)^2 + (y - y_M)^2 - r^2 = 0$
- Parameter: $x(\theta) = x_M + r \cdot \cos\theta$, $y(\theta) = y_M + r \cdot \sin\theta$, $\theta \in [0,2\pi[$
- Nachteil: Beide Methoden zur Kreisdarstellung sind rechenaufwändig

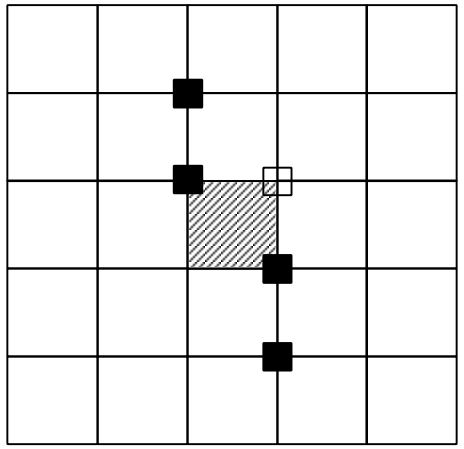


3.3 Rasterung von Kreisen

Darstellung eines Kreises: Bemerkungen

- Mit der Berechnung eines Kreispunktes sind durch **Symmetrie** 7 weitere Kreispunkte gegeben
- Für eine gleichmäßige Rasterung müssen die Pixel entlang des Kreises **möglichst gleichmäßig** verteilt sein
- Die Bewertung der Kreisapproximation im Raster ist subjektiv
- von jedem Rasterquadrant dürfen nur 2 Eckpunkte gesetzt werden



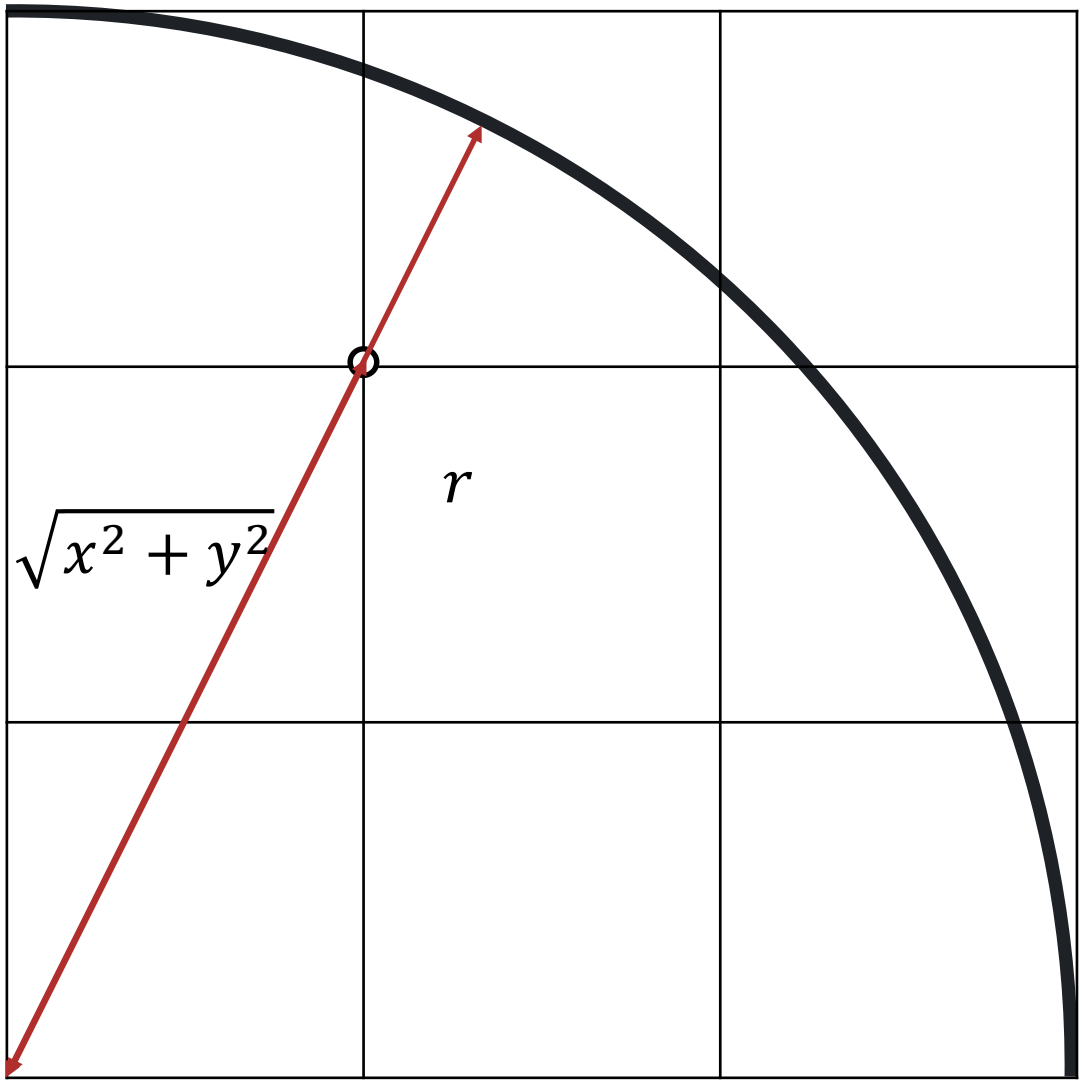


3.3 Rasterung von Kreisen

Darstellung eines Kreises: Bemerkungen

- Ein häufig verwendetes Kriterium ist die **Minimierung des Residuums**

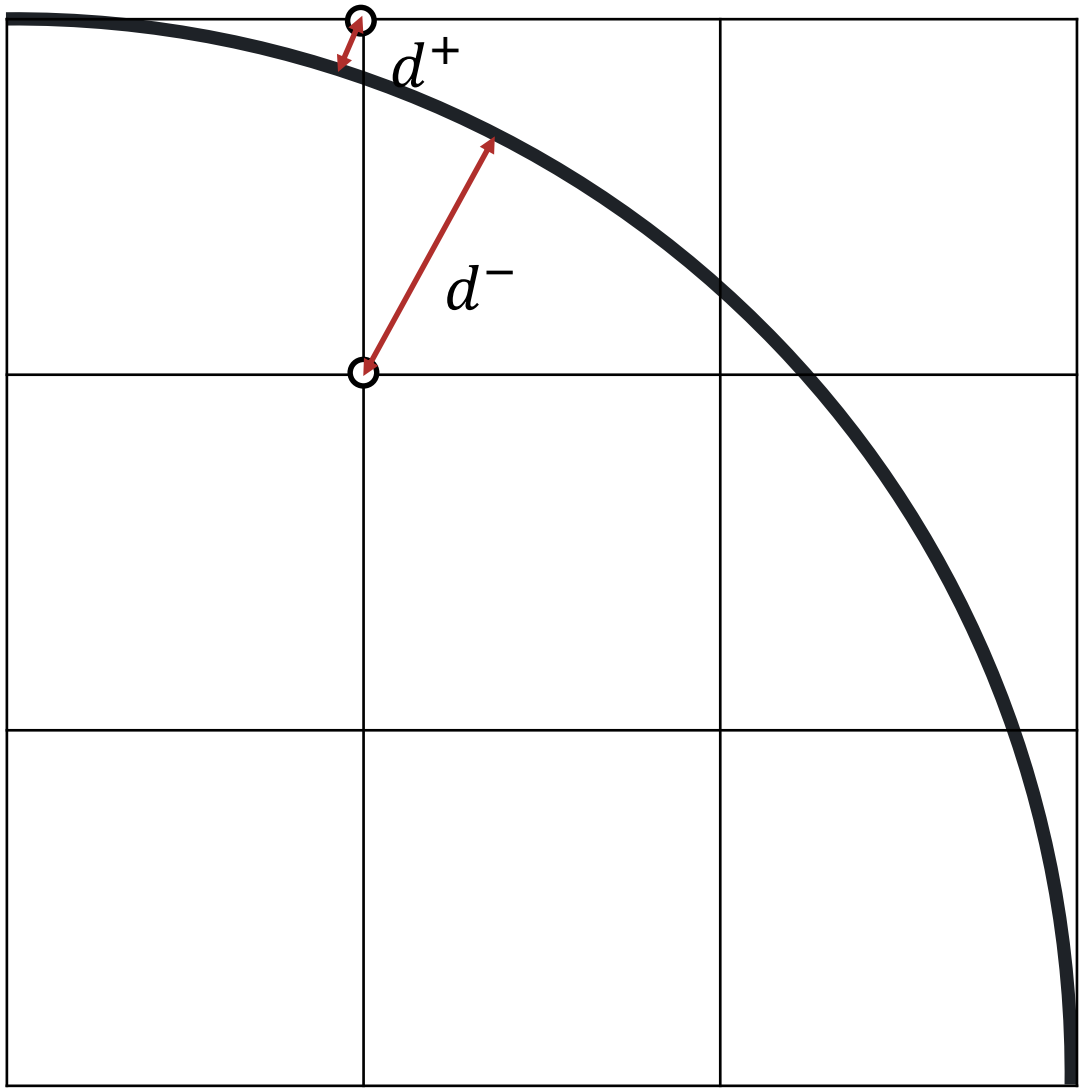
$$d = |x^2 + y^2 - r^2|$$



3.3 Rasterung von Kreisen

Bresenham-Algorithmus für Kreise

- Innerhalb des 2. Oktanten
  - Kreisfunktion monoton fallend
  - Steigung zwischen 0 und -1
  - P^+ liegt immer außerhalb der Kreislinie
  - P^- liegt immer innerhalb der Kreislinie



3.3 Rasterung von Kreisen

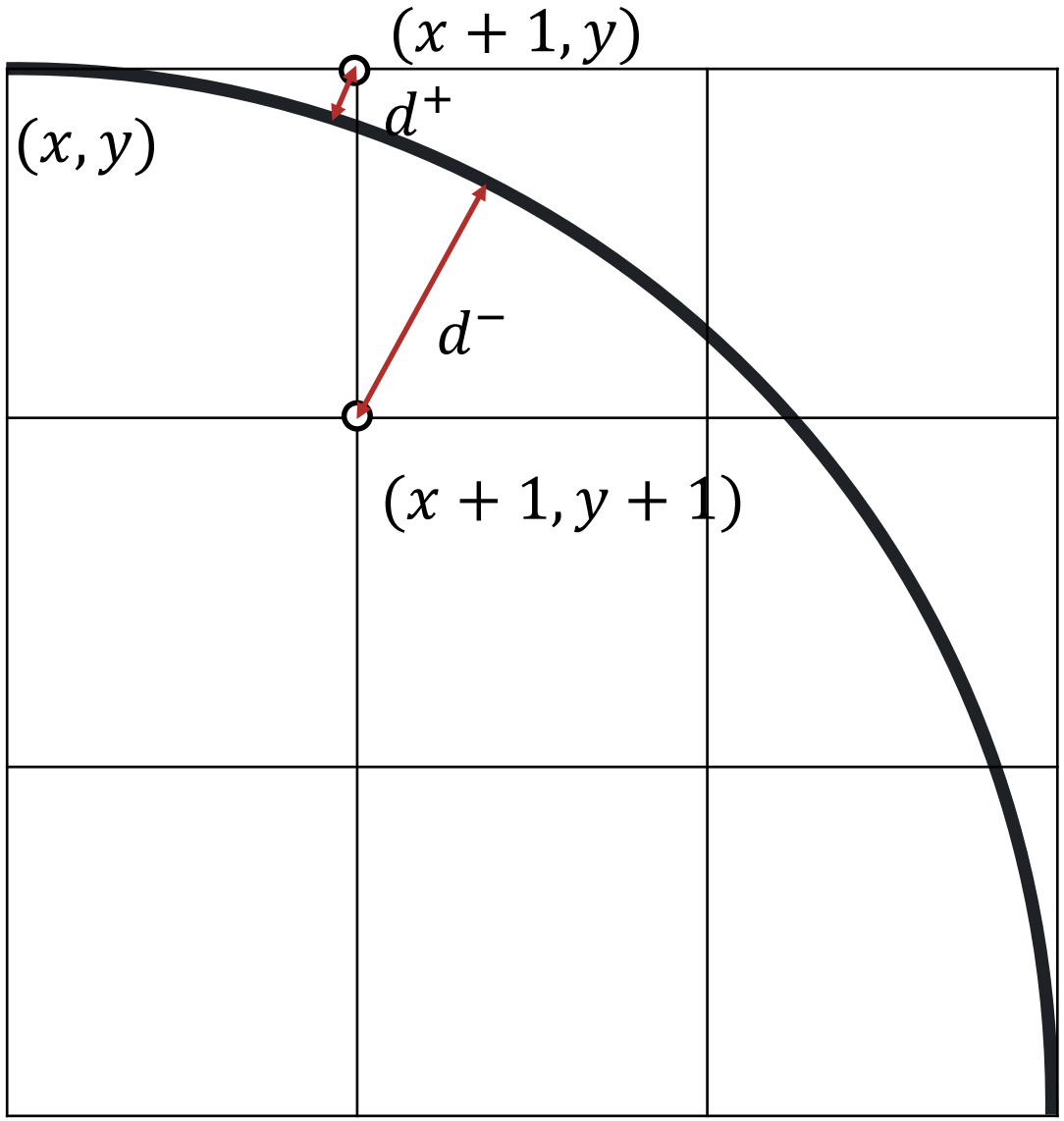
Bresenham-Algorithmus für Kreise

- Hier:
  - $(x_M, y_M) = (0,0)$
  - Startpunkt ist Rasterpunkt
  - $r \in \mathbb{N}$
  - 2. Oktant
- Gehe nach unten, wenn $d^+ > d^-$

$d^+ \quad =$

$\quad\quad =$

$d^- \quad =$



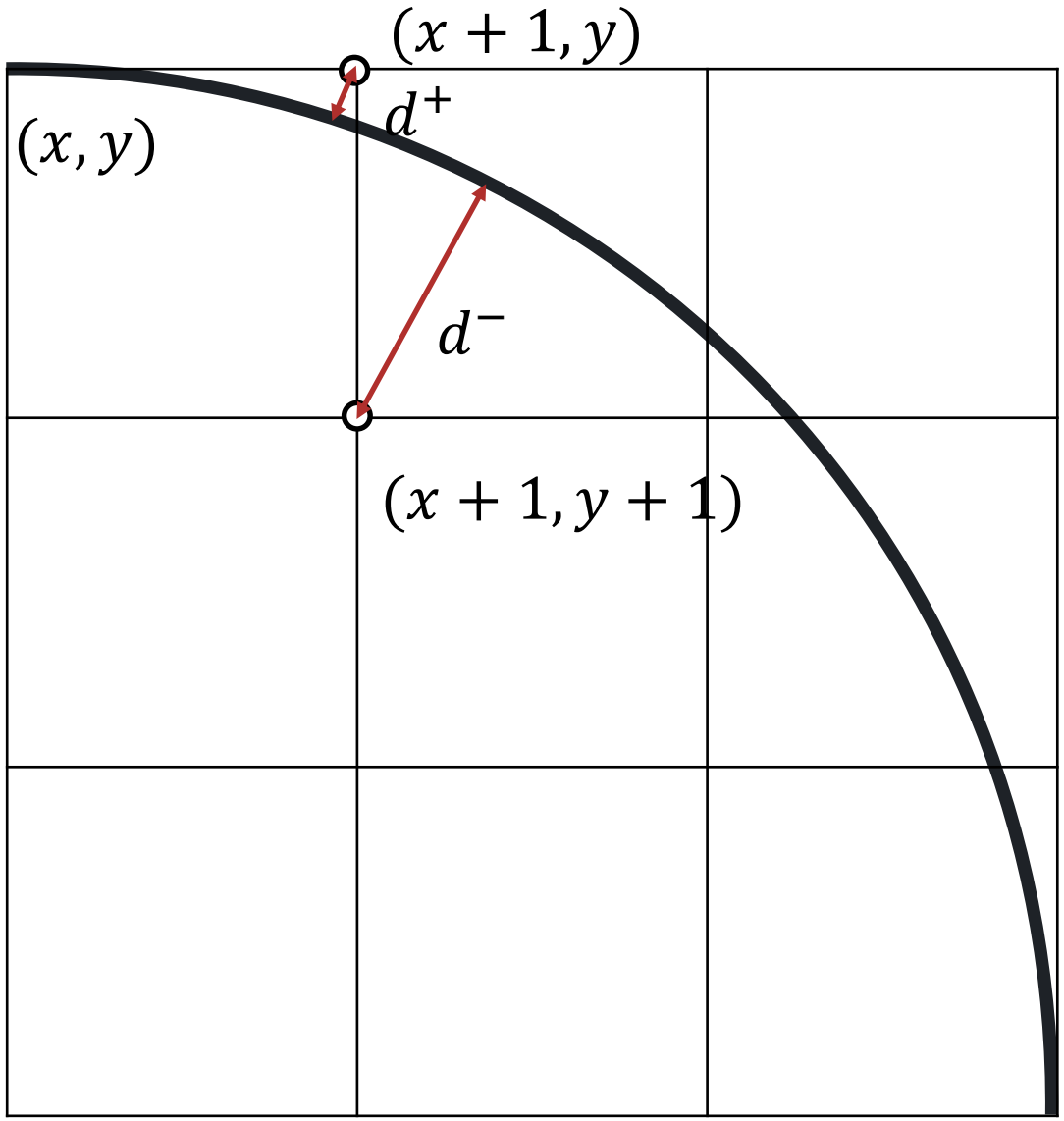
3.3 Rasterung von Kreisen

Bresenham-Algorithmus für Kreise

- Hier:
  - $(x_M, y_M) = (0,0)$
  - Startpunkt ist Rasterpunkt
  - $r \in \mathbb{N}$
  - 2. Oktant
- Gehe nach unten, wenn $d^+ > d^-$

$$d^+ = |(x_i + 1)^2 + y_i^2 - r^2|$$
$$= (x_i + 1)^2 + y_i^2 - r^2$$

$$d^- = |(x_i + 1)^2 + (y_i - 1)^2 - r^2|$$
$$= r^2 - (x_i + 1)^2 - (y_i - 1)^2$$



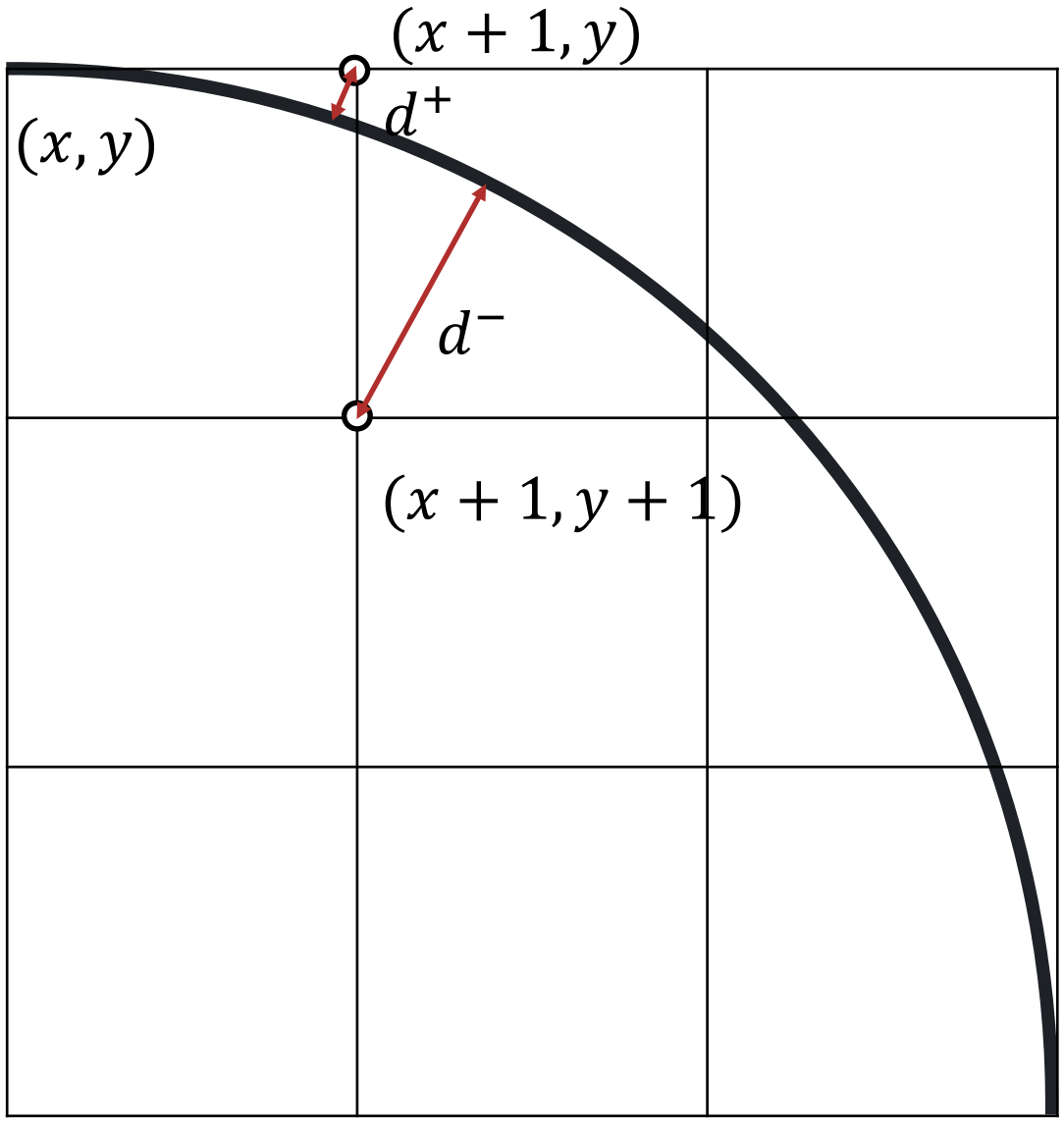
3.3 Rasterung von Kreisen

Bresenham-Algorithmus für Kreise

- Beste Approximation über **Entscheidungsgröße E** mittels minimalem Residuum $E = d^+ - d^-$
- Gehe nach unten, wenn $d^+ > d^- \Leftrightarrow E > 0$

$E \leq 0: x++,$

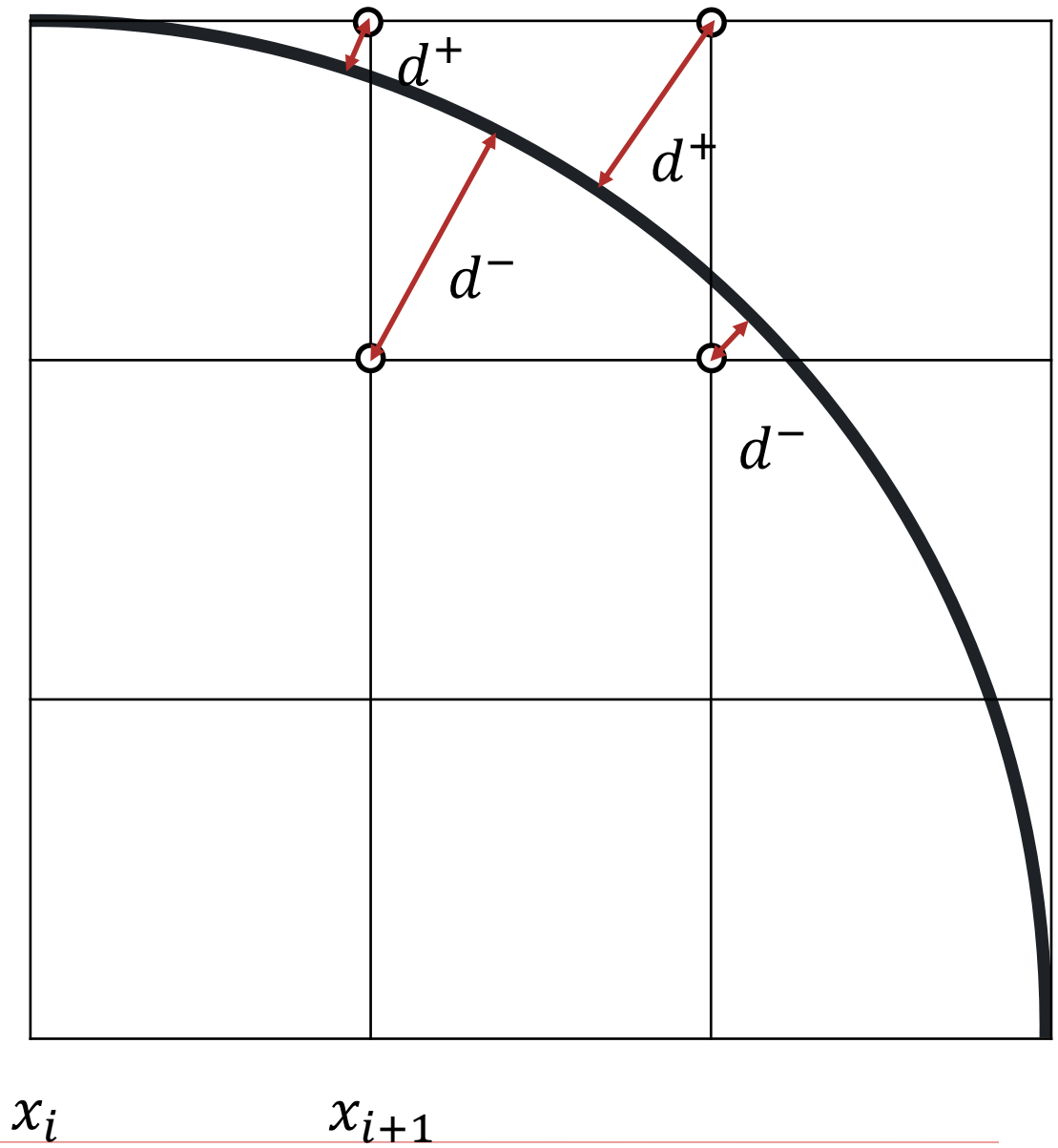
$E > 0: x++, y--,$



3.3 Rasterung von Kreisen

Bresenham-Algorithmus für Kreise

- Zur schnelleren Berechnung kann E **inkrementell** bestimmt werden



3.3 Rasterung von Kreisen

Bresenham-Algorithmus für Kreise

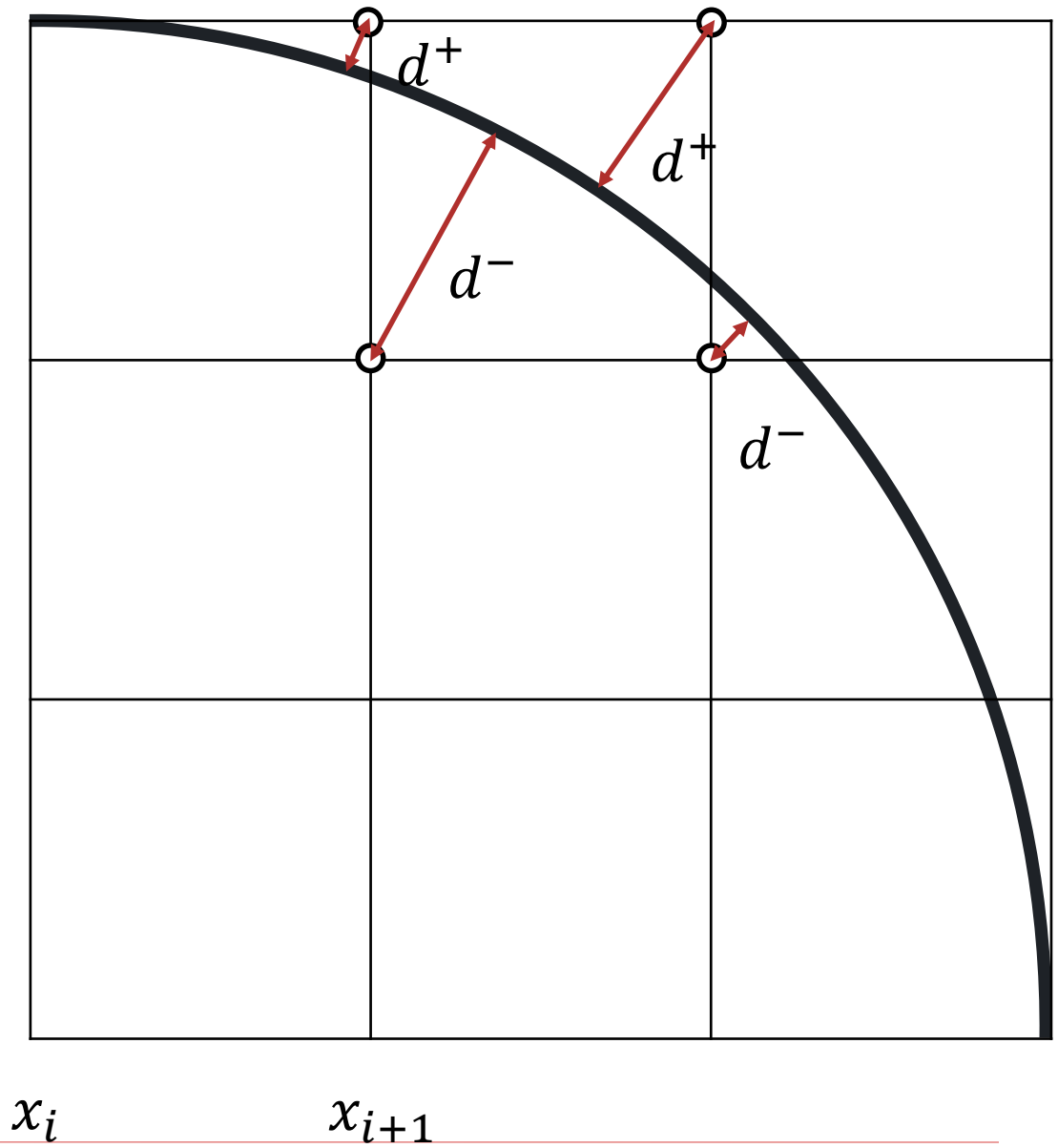
- Zur schnelleren Berechnung kann E **inkrementell** bestimmt werden
- $E \leq 0: x++$

$$
\begin{aligned}
d_{i+1}^{+} &= (x_{i+1}+1)^2 + y_{i+1}^2 - r^2 \\
&= (x_i + 1 + 1)^2 + y_i^2 - r^2 \\
&= (x_i+1)^2 + 2(x_i+1) + 1 + y_i^2 - r^2 \\
&= d_i^{+} + 2x_i + 3 \\
&= d_i^{+} + 2x_{i+1} + 1
\end{aligned}
$$

$$
\begin{aligned}
d_{i+1}^{-} &= r^2 - (x_{i+1}+1)^2 - (y_{i+1}-1)^2 \\
&= r^2 - (x_i + 1 + 1)^2 - (y_i - 1)^2 \\
&= r^2 - (x_i+1)^2 - 2(x_i+1) - 1 - (y_i-1)^2 \\
&= d_i^{-} - 2x_i - 3 \\
&= d_i^{-} - 2x_{i+1} - 1
\end{aligned}
$$

E_{i+1}

:



3.3 Rasterung von Kreisen

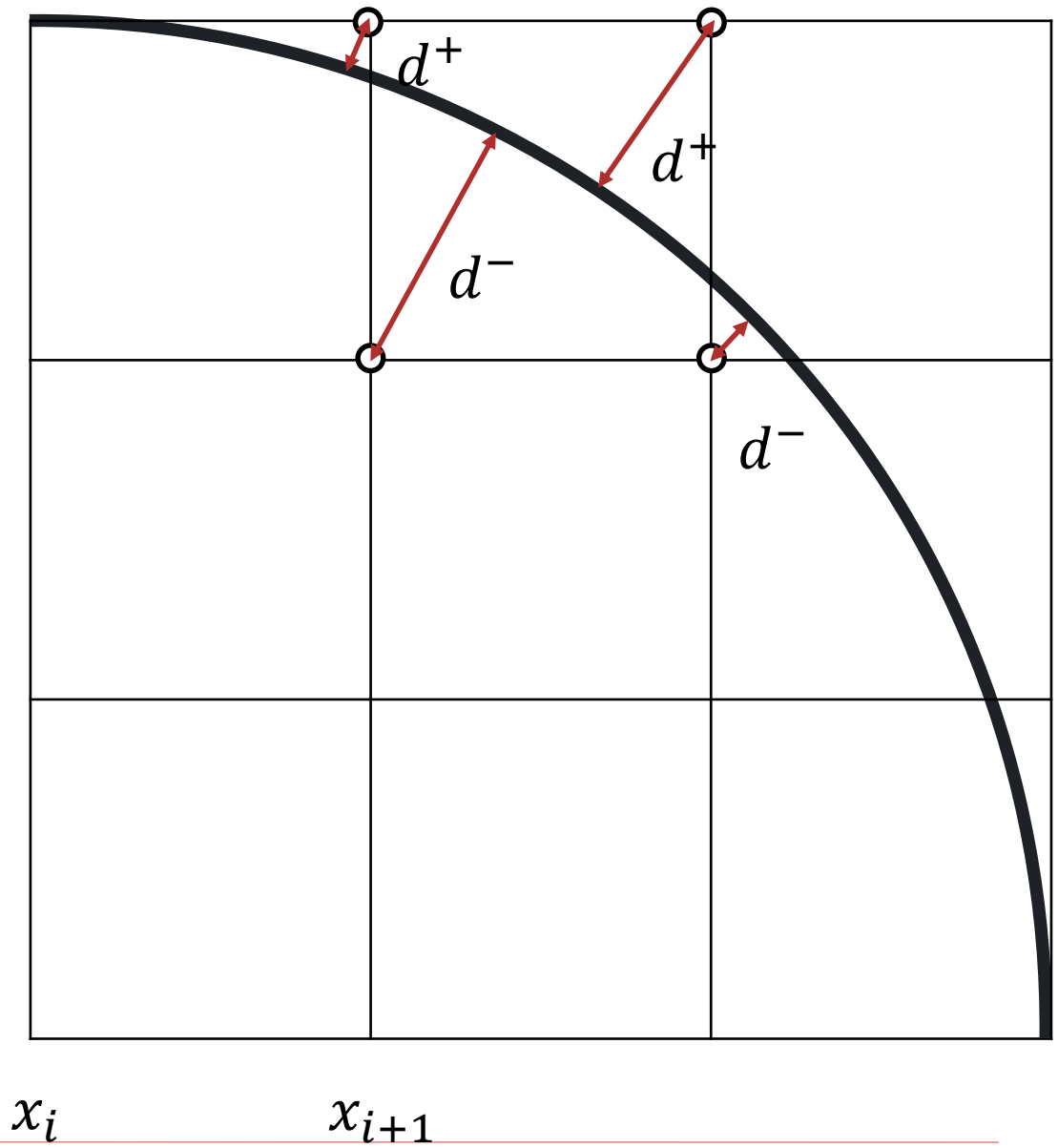
Bresenham-Algorithmus für Kreise

- Zur schnelleren Berechnung kann E **inkrementell** bestimmt werden
- $E \leq 0: x++$

$$
\begin{aligned}
d_{i+1}^{+} &= (x_{i+1}+1)^2 + y_{i+1}^2 - r^2 \\
&= (x_i+1+1)^2 + y_i^2 - r^2 \\
&= (x_i+1)^2 + 2(x_i+1) + 1 + y_i^2 - r^2 \\
&= d_i^{+} + 2x_i + 3 \\
&= d_i^{+} + 2x_{i+1} + 1
\end{aligned}
$$

$$
\begin{aligned}
d_{i+1}^{-} &= r^2 - (x_{i+1}+1)^2 - (y_{i+1}-1)^2 \\
&= r^2 - (x_i+1+1)^2 - (y_i-1)^2 \\
&= r^2 - (x_i+1)^2 - 2(x_i+1) - 1 - (y_i-1)^2 \\
&= d_i^{-} - 2x_i - 3 \\
&= d_i^{-} - 2x_{i+1} - 1
\end{aligned}
$$

$$
\begin{aligned}
E_{i+1} &= d_{i+1}^{+} - d_{i+1}^{-} \\
&= d_i^{+} + 2x_{i+1} + 1 - d_i^{-} + 2x_{i+1} + 1 \\
&= E_i + 4x_{i+1} + 2
\end{aligned}
$$



3.3 Rasterung von Kreisen

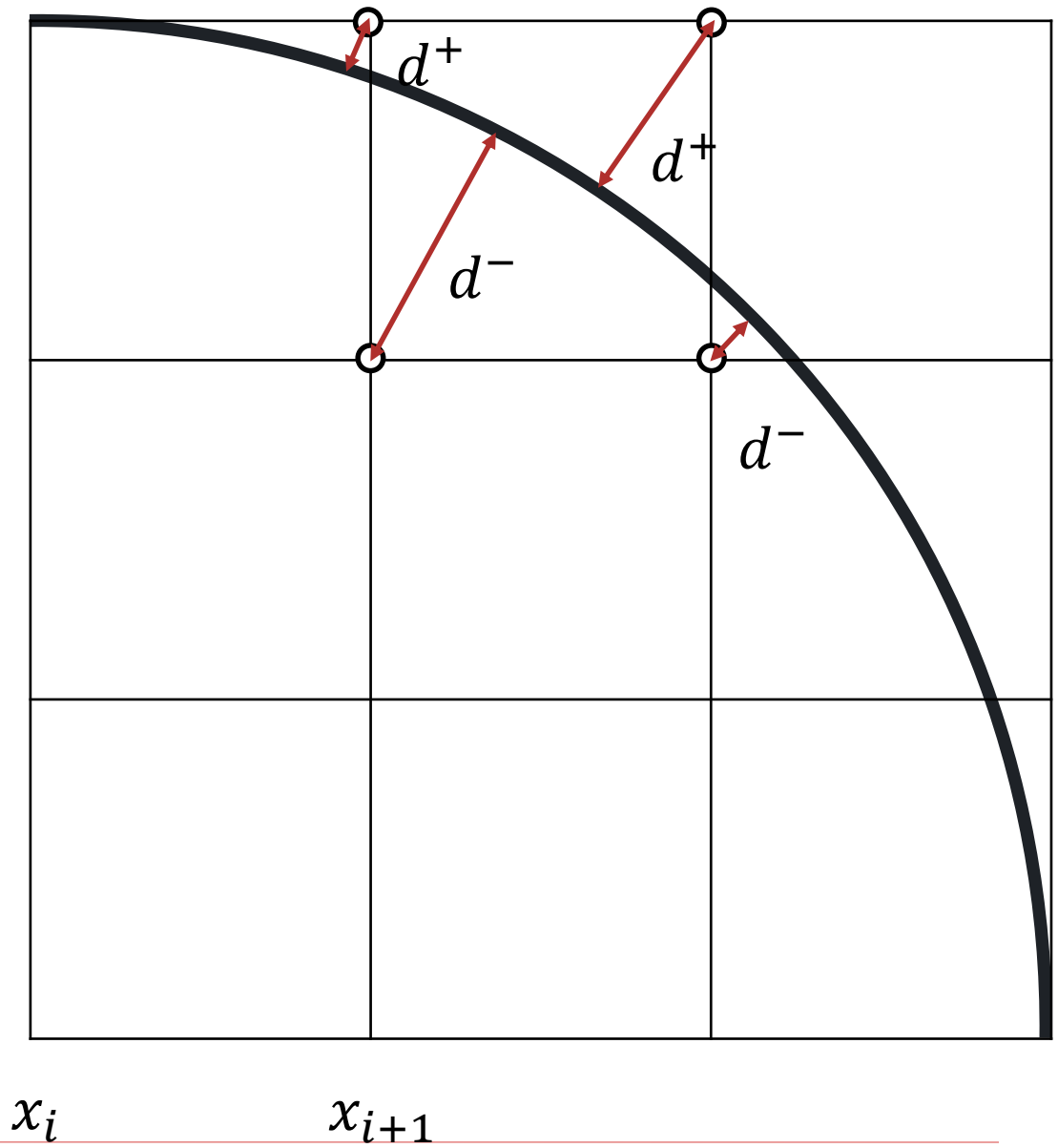
Bresenham-Algorithmus für Kreise

- Zur schnelleren Berechnung kann E **inkrementell** bestimmt werden
- $E > 0: x++, y--$

$$
\begin{aligned}
d_{i+1}^{+} &= (x_{i+1}+1)^2 + y_{i+1}^2 - r^2 \\
&= (x_i+1+1)^2 + (y_i-1)^2 - r^2 \\
&= (x_i+1)^2 + 2(x_i+1) + 1 + y_i^2 - 2y_i + 1 - r^2 \\
&= d_i^{+} + 2x_i - 2y_i + 4 \\
&= d_i^{+} + 2x_{i+1} - 2y_{i+1}
\end{aligned}
$$

$$
\begin{aligned}
d_{i+1}^{-} &= r^2 - (x_{i+1}+1)^2 - (y_{i+1}-1)^2 \\
&= r^2 - (x_i+1+1)^2 - (y_i-1-1)^2 \\
&= r^2 - (x_i+1)^2 + 2(x_i+1) + 1 - (y_i-1)^2 + 2(y_1-1) - 1 \\
&= d_i^{-} - 2x_i + 2y_i - 6 \\
&= d_i^{-} - 2x_{i+1} + 2y_{i+1} - 2
\end{aligned}
$$

$$
\begin{aligned}
E_{i+1} &= d_{i+1}^{+} - d_{i+1}^{-} \\
&= d_i^{+} + 2x_{i+1} - 2y_{i+1} - d_i^{-} + 2x_{i+1} - 2y_{i+1} + 2 \\
&= E_i + 4(x_i - y_i) + 2
\end{aligned}
$$



3.3 Rasterung von Kreisen

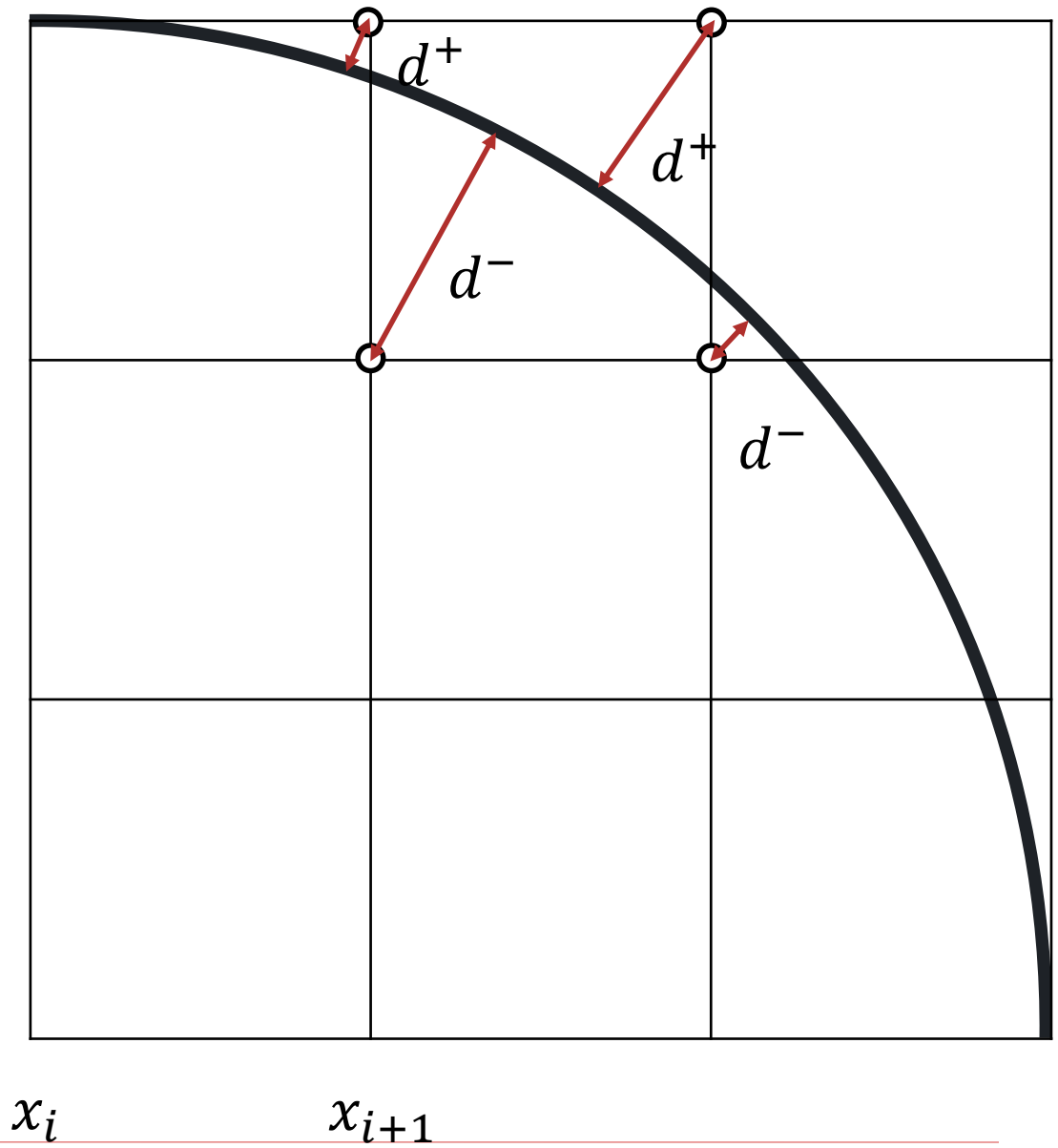
Bresenham-Algorithmus für Kreise

- Aktualisierung:
- $E \leq 0: x++,$

$$E_{i+1} = E_i + 4x_i + 2$$

- $E > 0: x++, y--,$

$$E_{i+1} = E_i + 4(x_i - y_i) + 2$$



3.3 Rasterung von Kreisen

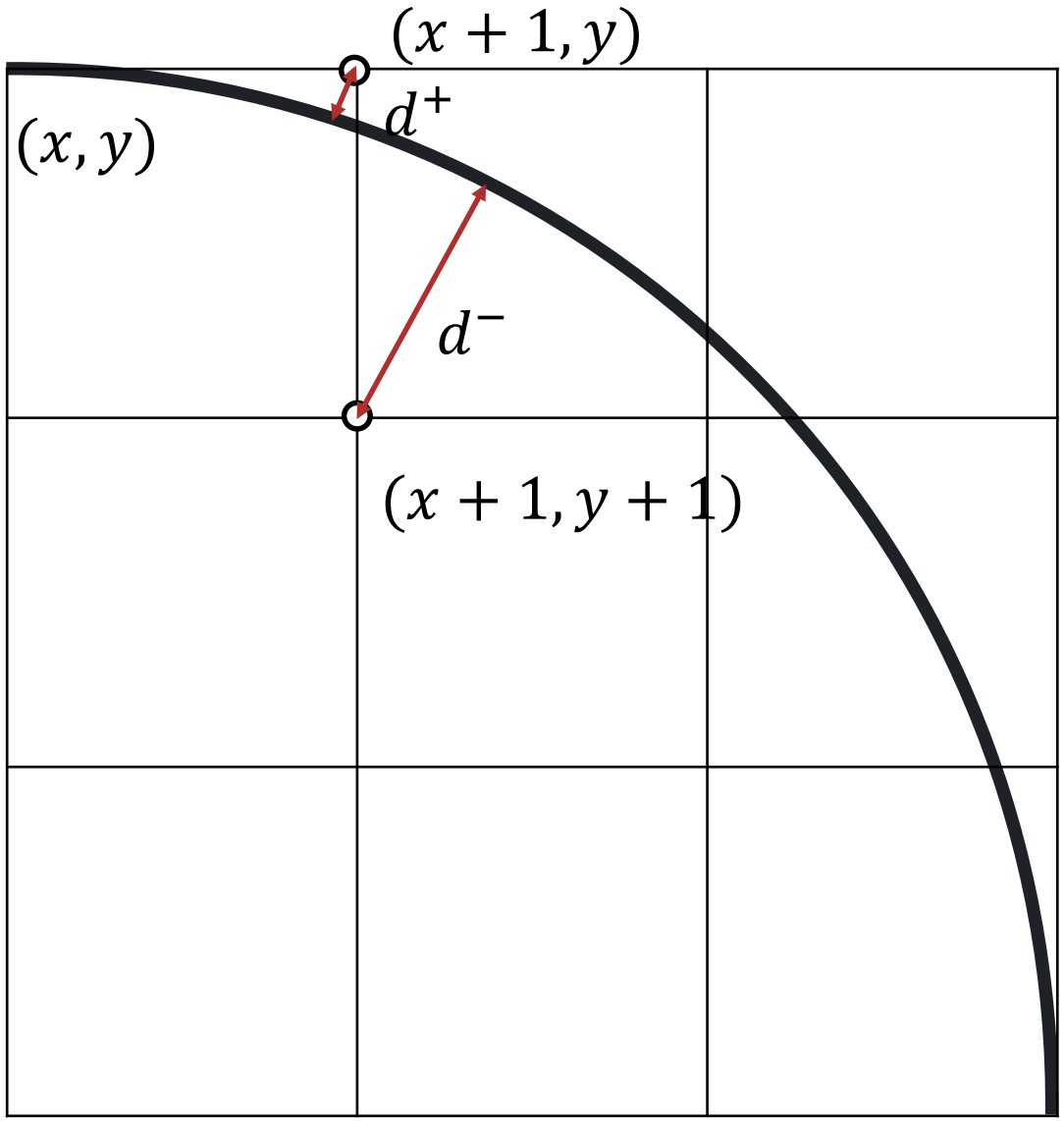
Bresenham-Algorithmus für Kreise

- Startwerte:
  - $x_1 = 0$
  - $y_1 = r$

d_1^+

d_1^-

E_1



3.3 Rasterung von Kreisen

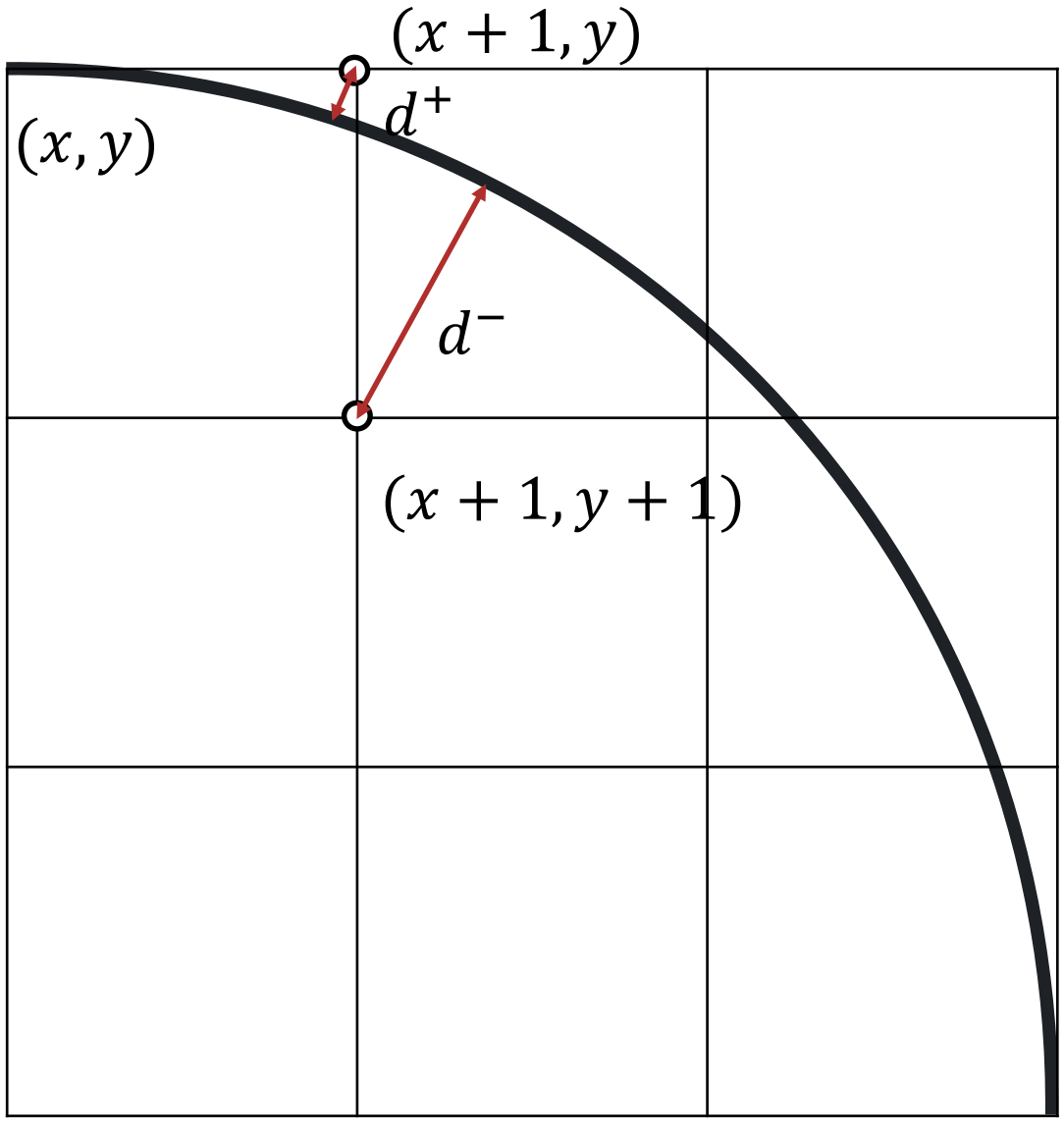
Bresenham-Algorithmus für Kreise

- Startwerte:
  - $x_1 = 0$
  - $y_1 = r$

$$d_1^+ = (x_1 + 1)^2 + y_1^2 - r^2 = 1 + r^2 - r^2 = 1$$

$$d_1^- = r^2 - (x_1 + 1)^2 - (y_1 - 1)^2 = r^2 - 1 - r^2 + 2r - 1 = 2r - 2$$

E_1



3.3 Rasterung von Kreisen

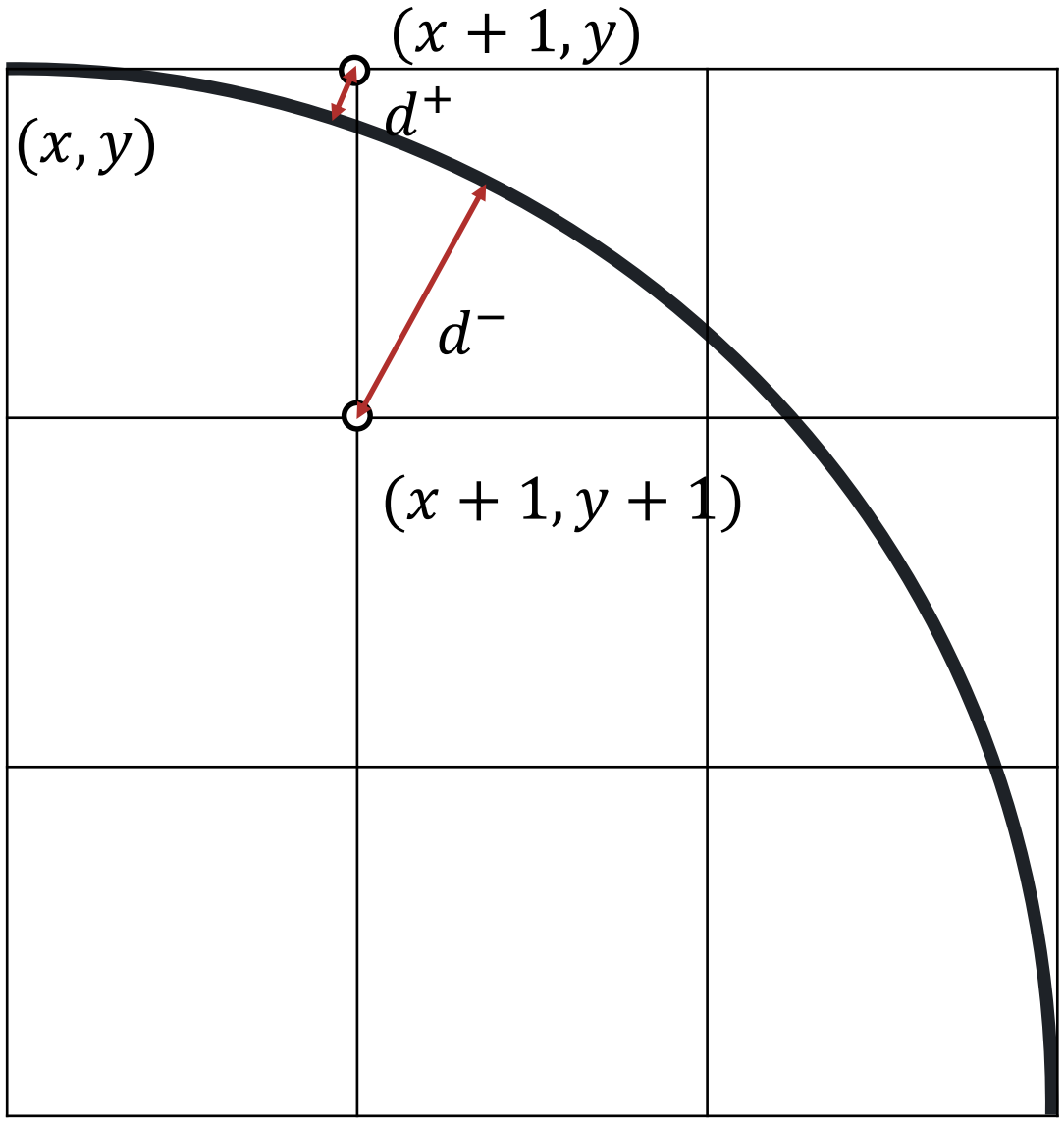
Bresenham-Algorithmus für Kreise

- Startwerte:
  - $x_1 = 0$
  - $y_1 = r$

$$
\begin{aligned}
d_1^+ &= (x_1 + 1)^2 + y_1^2 - r^2 \\
&= 1 + r^2 - r^2 \\
&= 1
\end{aligned}
$$

$$
\begin{aligned}
d_1^- &= r^2 - (x_1 + 1)^2 - (y_1 - 1)^2 \\
&= r^2 - 1 - r^2 + 2r - 1 \\
&= 2r - 2
\end{aligned}
$$

$$
\begin{aligned}
E_1 &= d_1^+ - d_1^- \\
&= 1 - 2r + 2 \\
&= 3 - 2r
\end{aligned}
$$



3.3 Rasterung von Kreisen

Bresenham-Algorithmus für Kreise

```
x = 0
y = r
E = 3 - 2*r
zeichneAchtPunkte(x,y)
solange x < y wiederhole:
   x = x + 1
   falls E > 0 dann
      y = y - 1
      E = E + 4(x-y) + 2
   sonst E = E + 4x + 2
   zeichneAchtPunkte(x,y)
ende solange
```

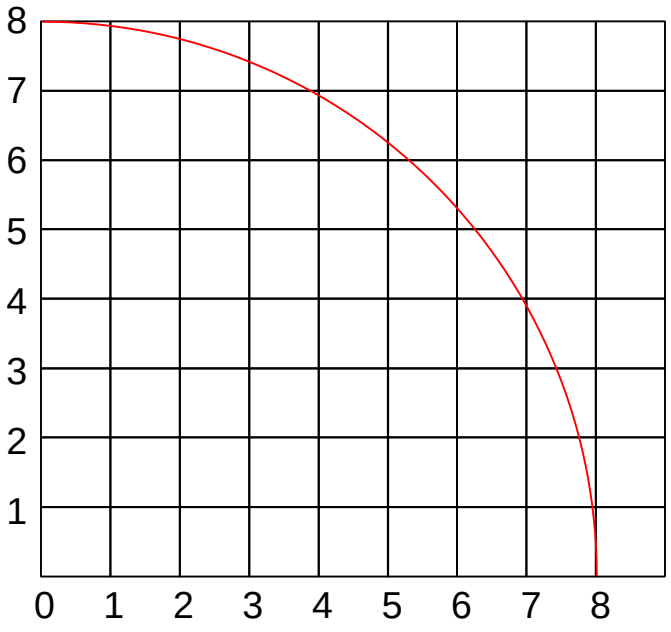
- 2. Oktant
- nur ganzzahlige Operatoren

3.3 Rasterung von Kreisen

Bresenham-Algorithmus für Kreise

```
x = 0
y = r
E = 3 - 2*r
zeichneAchtPunkte(x,y)
solange x < y wiederhole:
   x = x + 1
   falls E > 0 dann
      y = y - 1
      E = E + 4(x-y) + 2
   sonst E = E + 4x + 2
   zeichneAchtPunkte(x,y)
ende solange
```

– Beispiel: $r = 8$



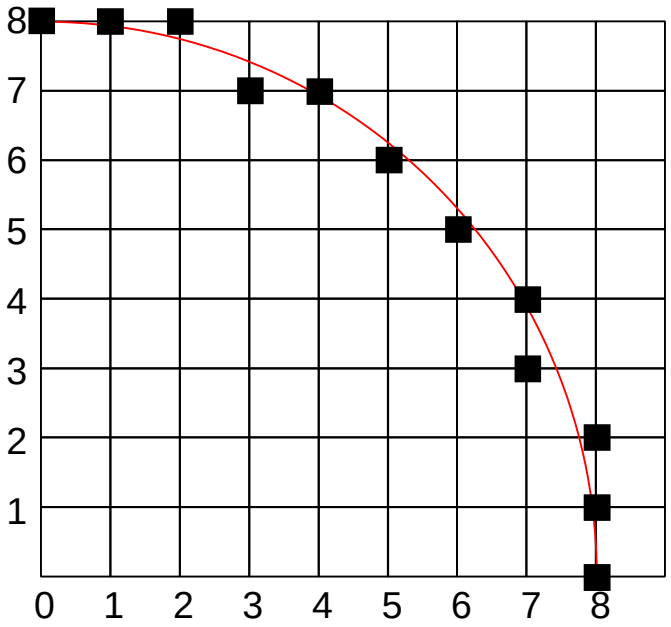
x	y	E	plot

3.3 Rasterung von Kreisen

Bresenham-Algorithmus für Kreise

```
x = 0
y = r
E = 3 - 2*r
zeichneAchtPunkte(x,y)
solange x < y wiederhole:
   x = x + 1
   falls E > 0 dann
      y = y - 1
      E = E + 4(x-y) + 2
   sonst E = E + 4x + 2
   zeichneAchtPunkte(x,y)
ende solange
```

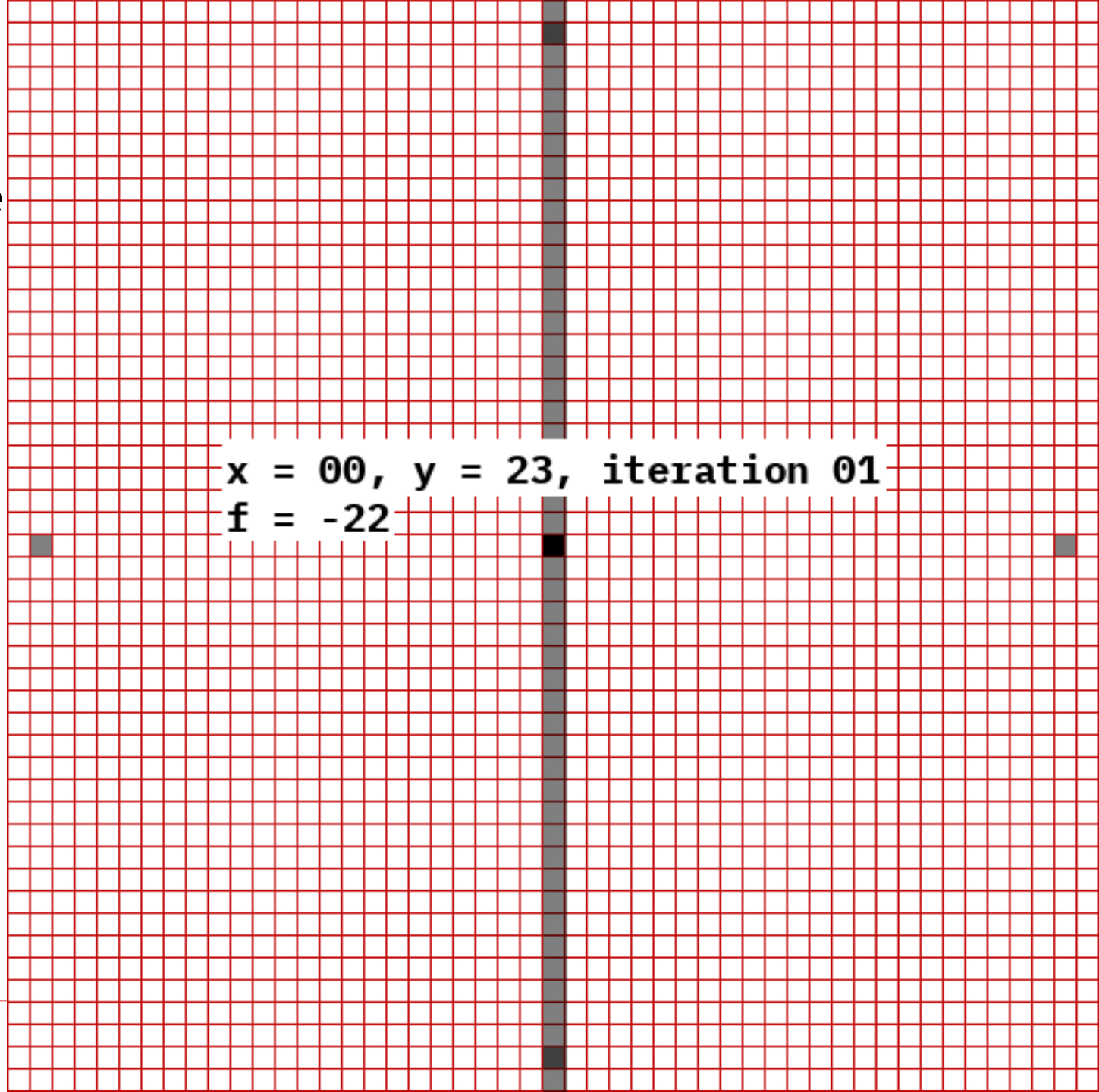
– Beispiel: $r = 8$



x	y	E	plot
0	8	-13	(0, 8)
1	8	-7	(1, 8)
2	8	3	(2, 8)
3	7	-11	(3, 7)
4	7	7	(4, 7)
5	6	5	(5, 6)
6	5	11	(6, 5)

3.3 Rasterung von Kreisen

Bresenham-Algorithmus für Kreise



3.3 Rasterung von Kreisen

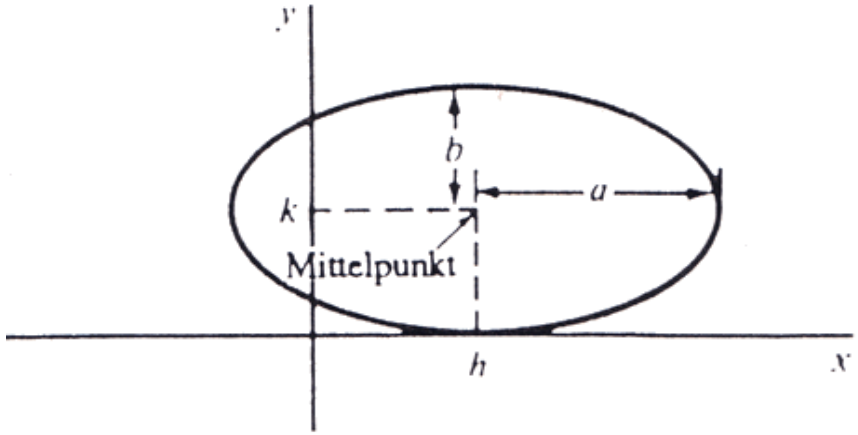
Ellipsendarstellung

- Zur Rasterung betrachtet man die Ellipsen
  - in **Normalform**
  - mit zu den **Koordinatenachsen parallelen Hauptachsen**

  → Algorithmus von Kappel

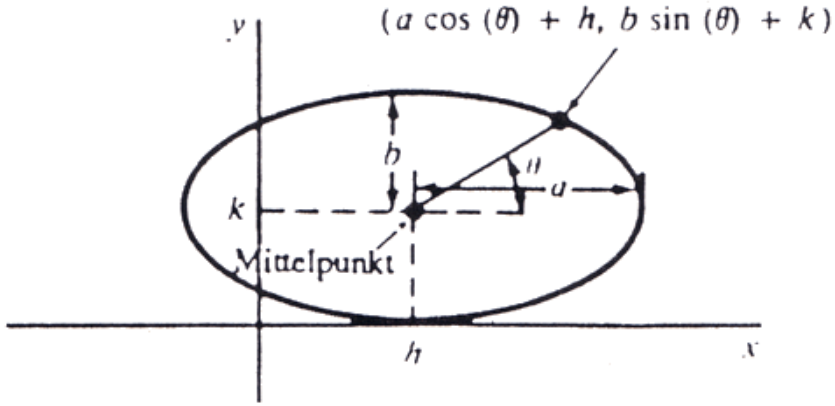
- Ellipsenerzeugung über die implizite Darstellung

$$\frac{(x-h)^2}{a^2} + \frac{(y-k)^2}{b^2} = 1$$



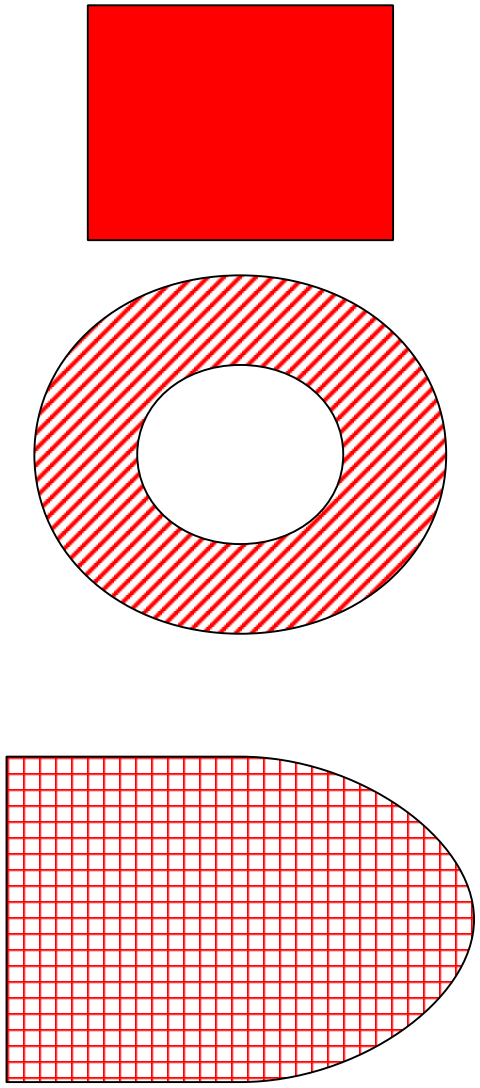
- Ellipsenerzeugung über die Parameterdarstellung

$$x = h + a\cos\theta \qquad y = k + b\sin\theta$$



3.4 Füllalgorithmen

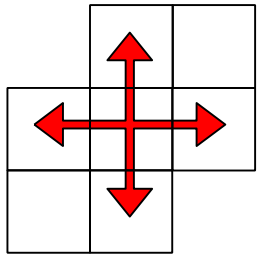
- Ziel: Füllen bzw. Einfärben eines begrenzten Bereiches oder Gebietes mit einer Füllfarbe oder einem Muster bzw. einer Schraffur
- Beispiele:
  - Balkendiagramme
  - Flächen
  - Körper
- Beschreibung der zu füllenden Gebiete erfolgt geometrisch,
  - Durch Ecken, Strecken, Polygone, Kreise (randdefiniert)
  - Durch Pixel (inhaltsdefiniert)



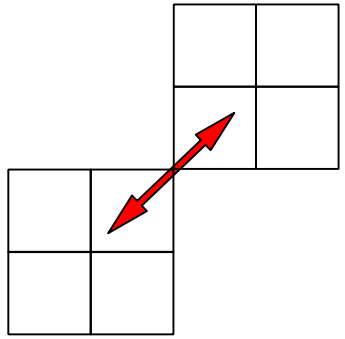
3.4 Füllalgorithmen

Zusammenhang von Gebieten

- 4-fach zusammenhängend (nur Horizontal- und Vertikalbewegung)



- 8-fach zusammenhängend (zusätzlich Diagonalbewegung möglich)



3.4 Füllalgorithmen

- Bemerkungen
  - Füllalgorithmen mit **8 Freiheitsgraden** (Bewegungsrichtungen) können **auch 4-fach** zusammenhängende Gebiete füllen.
  - Füllalgorithmen mit **4 Freiheitsgraden können keine 8-fach** zusammenhängenden Gebiete füllen.

- Problem:
  4-fach zusammenhängende Gebiete mit gemeinsamen Ecken

- Techniken zum Rastern eines Polygons / Füllen eines Gebietes
  - Scan-Line-Methode
  - Saatkorn-Methode
  - Hybrid-Methoden

3.4 Füllalgorithmen

Scan-Line-Methode

- Andere Bezeichnungen
  - Rasterzeilen-Methode
  - Scan-Conversion
- Arbeitet zeilenweise von oben nach unten
- Ein Pixel der aktuellen Zeile (Scan-Line) wird nur dann gezeichnet, wenn es innerhalb des Polygons liegt
- Definition der Gebiete
  - Geometrisch
  - Pixelweise

3.4 Füllalgorithmen

Scan-Line-Methode

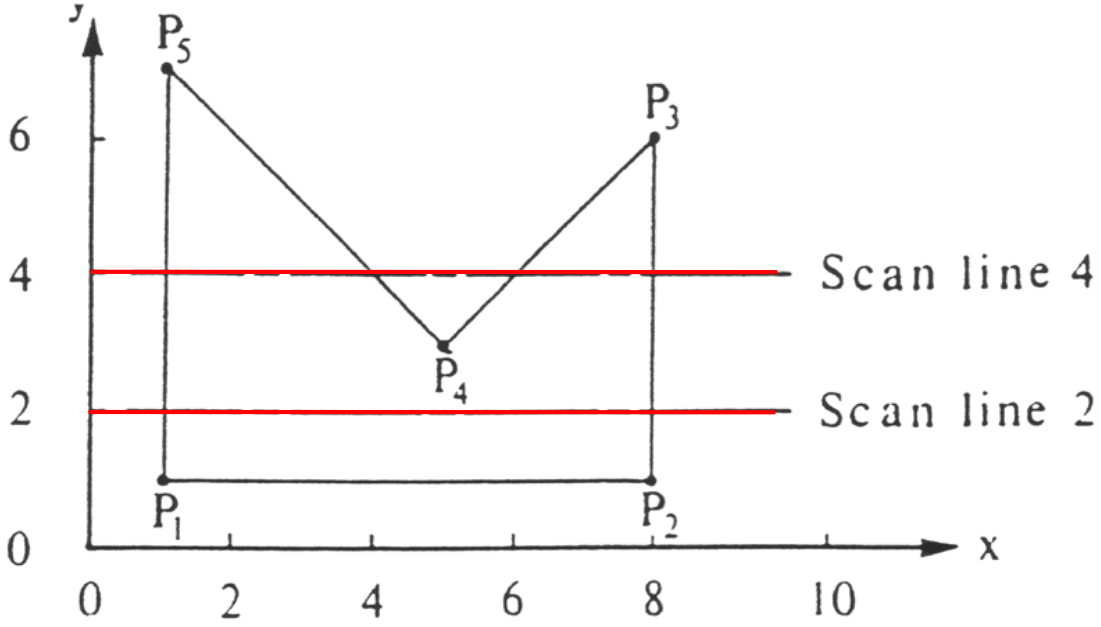
```
– // einfachster Ansatz:
– for (y = ymin; y <= ymax; y++) {
  – // row, Zeile
  – for (x = xmin; x <= xmax; x++) {
    – // column, Spalte
    – if (Inside(polygon, x, y) {
      –  SetPixel(x,y);
    – }
  – }
– }
```

- Eigenschaften
  - Sehr langsam
  - Verbesserung der Laufzeit durch Ausnutzung von Kohärenz

3.4 Füllalgorithmen

Scan-Line-Methode

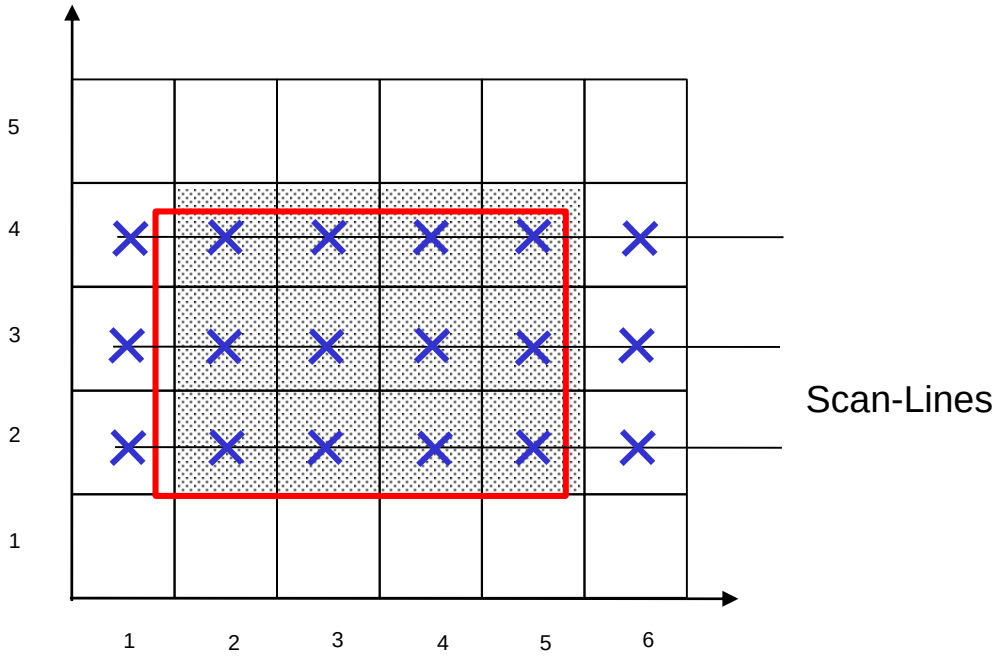
- Ausnutzung von Zeilenkohärenz
  - Benachbarte Pixel auf einer Zeile besitzen höchstwahrscheinlich die gleichen Intensitätswerte.
  - Pixelcharakteristik (Intensität) ändert sich nur dort, wo ein Schnittpunkt einer Polygonkante mit einer Scan Line vorliegt, d.h. der Bereich zwischen zwei Schnittpunkten gehört zum Polygon oder nicht.
- Schnitt mit Polygon
  - $y = 2$: für $x = 1{,}8$
  - $y = 4$: für $x = 1{,}4{,}6{,}8$



3.4 Füllalgorithmen

Scan-Line-Methode

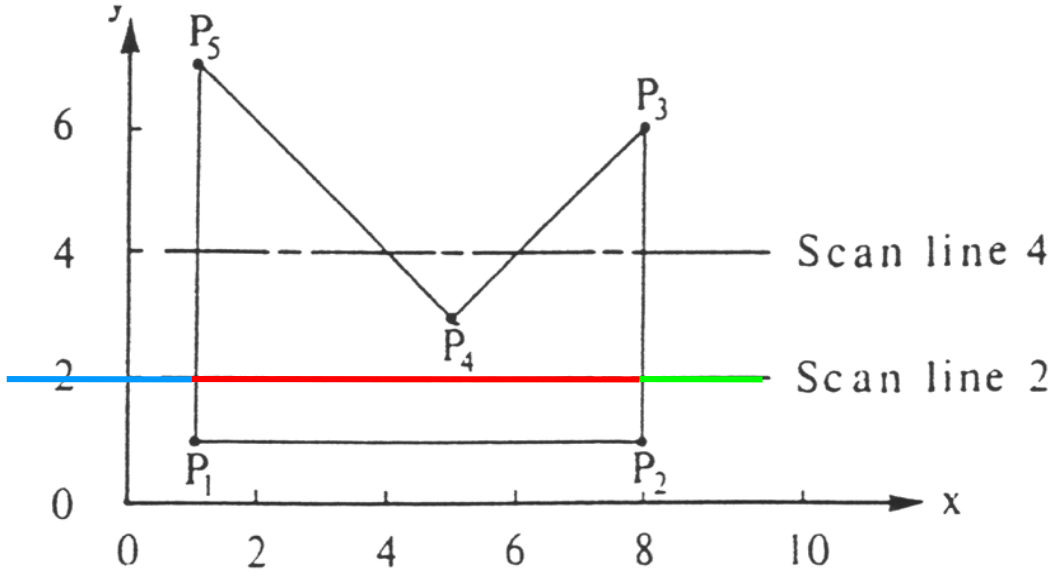
- Ausnutzung von Zeilenkohärenz
  - Benachbarte Pixel auf einer Zeile besitzen höchstwahrscheinlich die gleichen Intensitätswerte.
  - Pixelcharakteristik (Intensität) ändert sich nur dort, wo ein Schnittpunkt einer Polygonkante mit einer Scan Line vorliegt, d.h. der Bereich zwischen zwei Schnittpunkten gehört zum Polygon oder nicht.
- Hier sind die Pixelzeilen und -spalten nummeriert, nicht die Koordinaten der Achsen!



3.4 Füllalgorithmen

Scan-Line-Methode

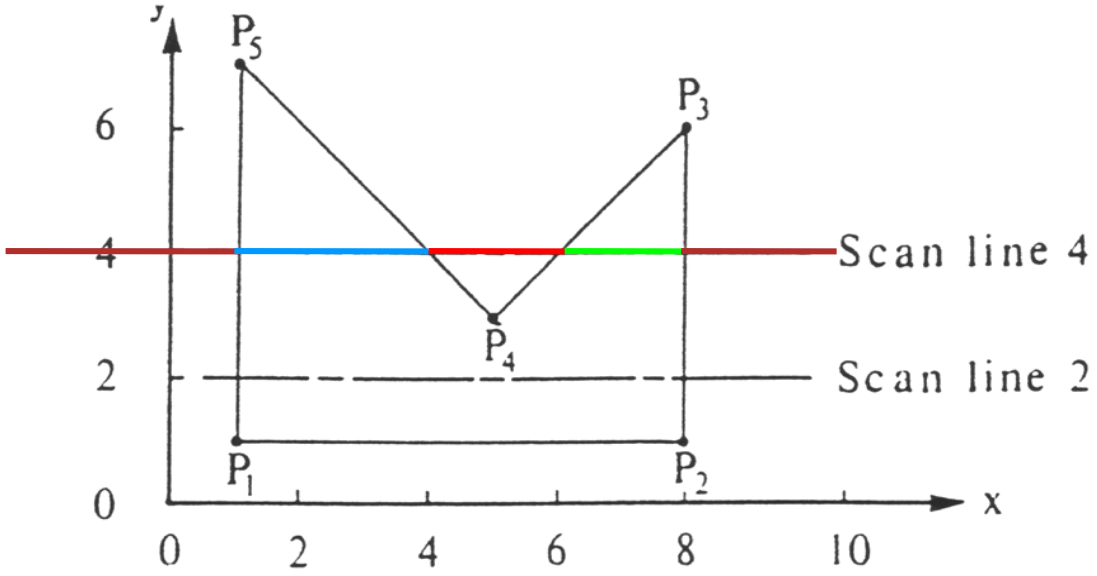
- Scan-Line $y = 2$
- Unterteilung in 3 Bereiche:
  - $x < 1$ außerhalb des Polygons
  - $1 \leq x \leq 8$ innerhalb des Polygons
  - $x > 8$ außerhalb des Polygons



3.4 Füllalgorithmen

Scan-Line-Methode

- Scan-Line $y = 4$
- Unterteilung in 5 Bereiche:
  - $x < 1$ außerhalb des Polygons
  - $1 \leq x \leq 4$ innerhalb des Polygons
  - $4 < x < 6$ außerhalb des Polygons
  - $6 \leq x \leq 8$ innerhalb des Polygons
  - $8 < x$ außerhalb des Polygons

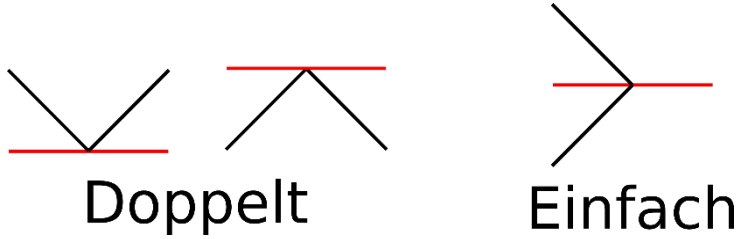


3.4 Füllalgorithmen

Scan-Line-Methode

- Probleme treten auf, wenn die Scan-Line das Polygon in einer Ecke schneidet
  → Betrachte **lokale Extrema**
- Lokale Extrema:
  - y-Werte der **Endpunkte** der in dieser Ecke beginnenden Polygonseiten sind **beide größer** oder **beide kleiner** als der y-Wert der **Schnittecke.**
  - Fallunterscheidung:
    - Ist die Ecke ein lokales Extremum, so zählt der **Schnitt zweifach.**
    - Ist die Ecke kein lokales Extremum, so zählt der **Schnitt nur einfach.**

- Einfacher Kanten-Listen-Algorithmus: Ordered Edge List Algorithm
- Funktionsweise:
  - Preprocessing
  - Scan Conversion



3.4 Füllalgorithmen

Scan-Line-Methode

- **Preprocessing**
  - Ermittle für jede Polygonkante die **Schnittpunkte mit den Scan-Lines** in der Pixelmitte
    - Bresenham
    - DDA-Algorithmus
  - Ignoriere dabei **horizontale Kanten**
  - **Speichere** jeden Schnittpunkt (x, y) in einer Liste
  - **Sortiere die Liste** dann von oben nach unten und von links nach rechts

- **Scan-Conversion**
  - Betrachte jeweils zwei direkt **aufeinander folgende Schnittpunkte** (x_1, y_1) und (x_2, y_2) der Liste
    - Listenelemente 1 und 2
    - Listenelemente 3 und 4
    - …
  - Aufgrund des Preprocessing gilt für die Scan-Line y

    $y = y_1 = y_2$ und $x_1 \leq x_2$
  - Zeichne alle Pixel auf der Scan-Line y, für die gilt:

    $x_1 \leq x < x_2$ mit ganzzahligem x

3.4 Füllalgorithmen

Scan-Line-Methode: Preprocessing

(1,7), (1,7)

(1,6), (2,6), (8,6), (8,6)

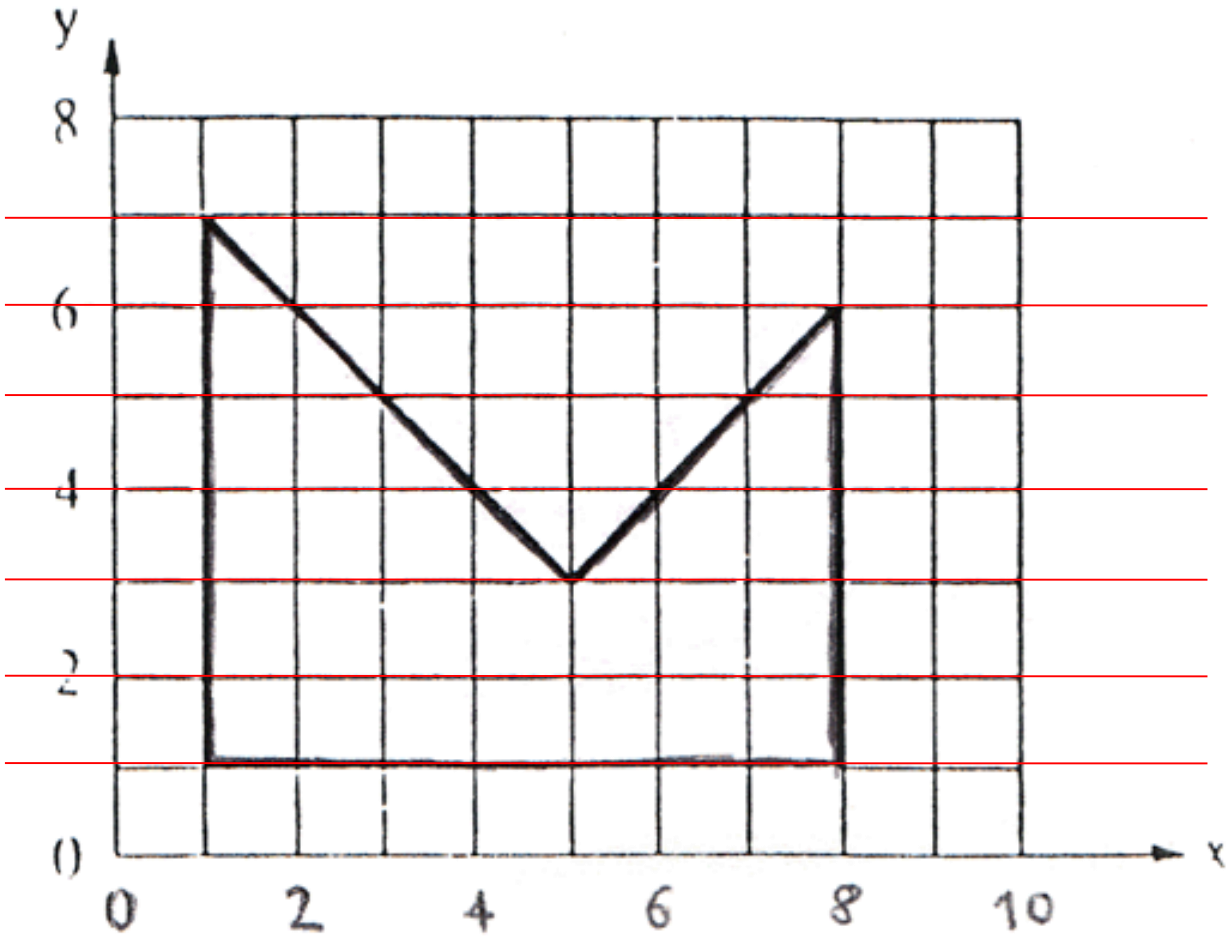
(1,5), (3,5), (7,5), (8,5)

(1,4), (4,4), (6,4), (8,4)

(1,3), (5,3), (5,3), (8,3)

(1,2), (8,2)

(1,1), (8,1)



3.4 Füllalgorithmen

Scan-Line-Methode: Scan-Conversion

(1,7), (1,7)

(1,6), (2,6), (8,6), (8,6)

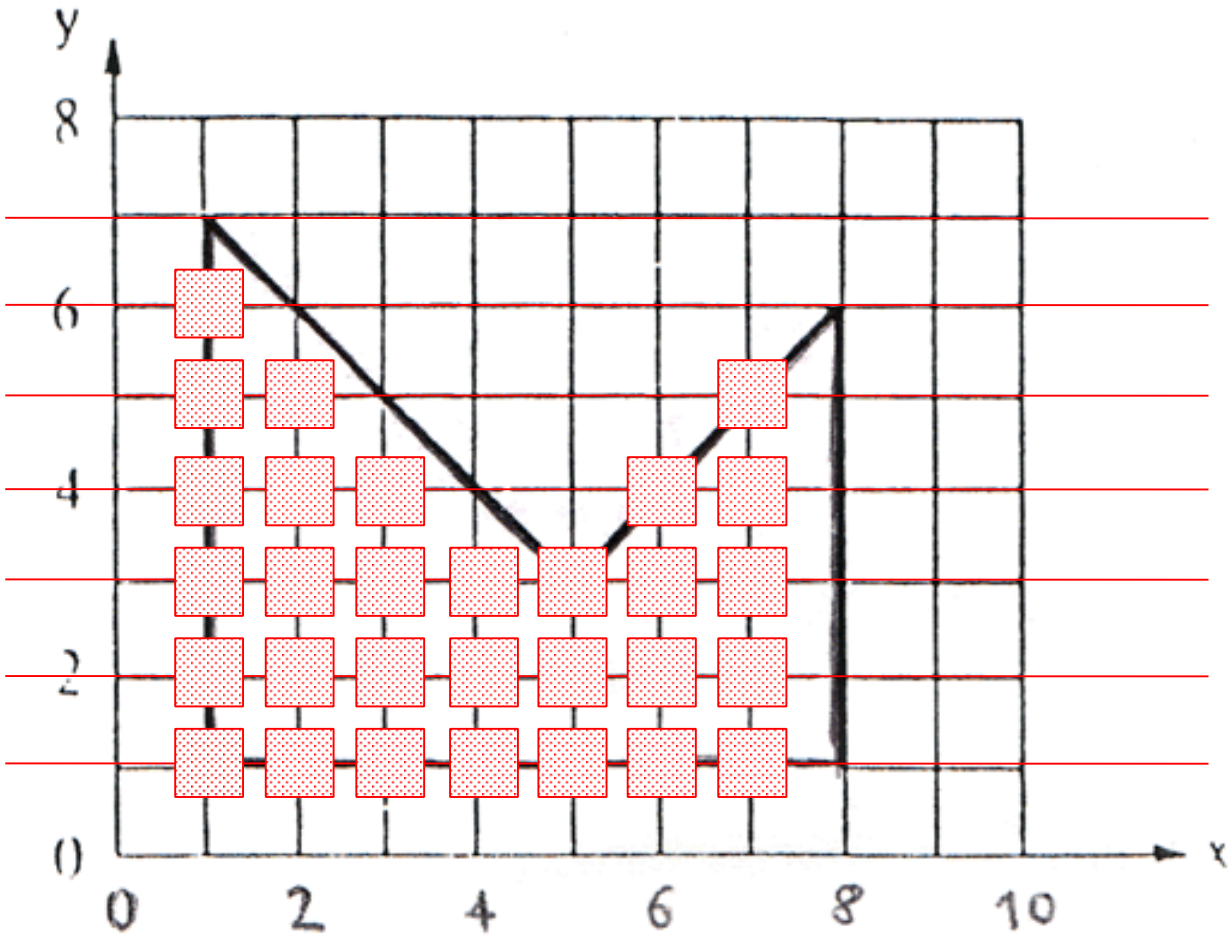
(1,5), (3,5), (7,5), (8,5)

(1,4), (4,4), (6,4), (8,4)

(1,3), (5,3), (5,3), (8,3)

(1,2), (8,2)

(1,1), (8,1)



3.4 Füllalgorithmen

Saatpunkt-Methode / Seed-Fill

- Seed-Fill-Methoden füllen das Gebiet ausgehend von einem **Ausgangspixel** (Saatpunkt, Seed)
- Eignet sich für **pixelweise definierte Gebiete**, also für Rastergeräte

- Man unterscheidet nach der Art der Gebietsdefinition:
  i. **Boundary-Fill-Algorithmus** für randdefinierte Gebiete
  ii. **Flood/Interior-Fill-Algorithmus** für inhaltsdefinierte Gebiete

3.4 Füllalgorithmen

Saatpunkt-Methode / Seed-Fill

i. Boundary-Fill-Algorithmus
   - Input
     - Startpunkt (Saatpunkt)
     - Farbe der **Begrenzungskurve**
     - Füllfarbe oder Muster
   - Algorithmus
     - Wiederhole
       - Vom Startpixel ausgehend werden rekursiv Nachbarpixel umgefärbt
     - Bis Pixel mit **der Farbe der Begrenzungskurve oder bereits umgefärbte Pixel** erreicht werden

3.4 Füllalgorithmen

Saatpunkt-Methode / Seed-Fill

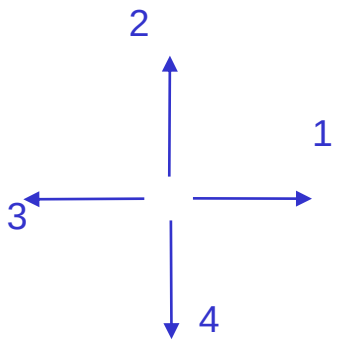
i. Boundary-Fill-Algorithmus
   - Input
     - Startpunkt (Saatpunkt)
     - Farbe der **Begrenzungskurve**
     - Füllfarbe oder Muster
   - Algorithmus
     - Wiederhole
       - Vom Startpixel ausgehend werden rekursiv Nachbarpixel umgefärbt
     - Bis Pixel mit **der Farbe der Begrenzungskurve oder bereits umgefärbte Pixel** erreicht werden

ii. Flood/Interior-Fill-Algorithmus
   - Input
     - Startpunkt (Saatkorn)
     - Farbe der **umzufärbenden Pixel**
     - Füllfarbe oder Muster
   - Algorithmus
     - Wiederhole
       - Vom Startpixel ausgehend werden rekursiv Nachbarpixel umgefärbt
     - Bis Pixel mit **abweichender Farbe** erreicht werden

3.4 Füllalgorithmen

Saatpunkt-Methode / Seed-Fill

- Einfacher Saatkorn-Algorithmus
  - 4 Bewegungsrichtungen
  - randdefiniertes Gebiet
  - FILO/LIFO-Prinzip (Stack)
- Eventuell werden Pixel mehrfach im Stack abgelegt (und gefärbt)

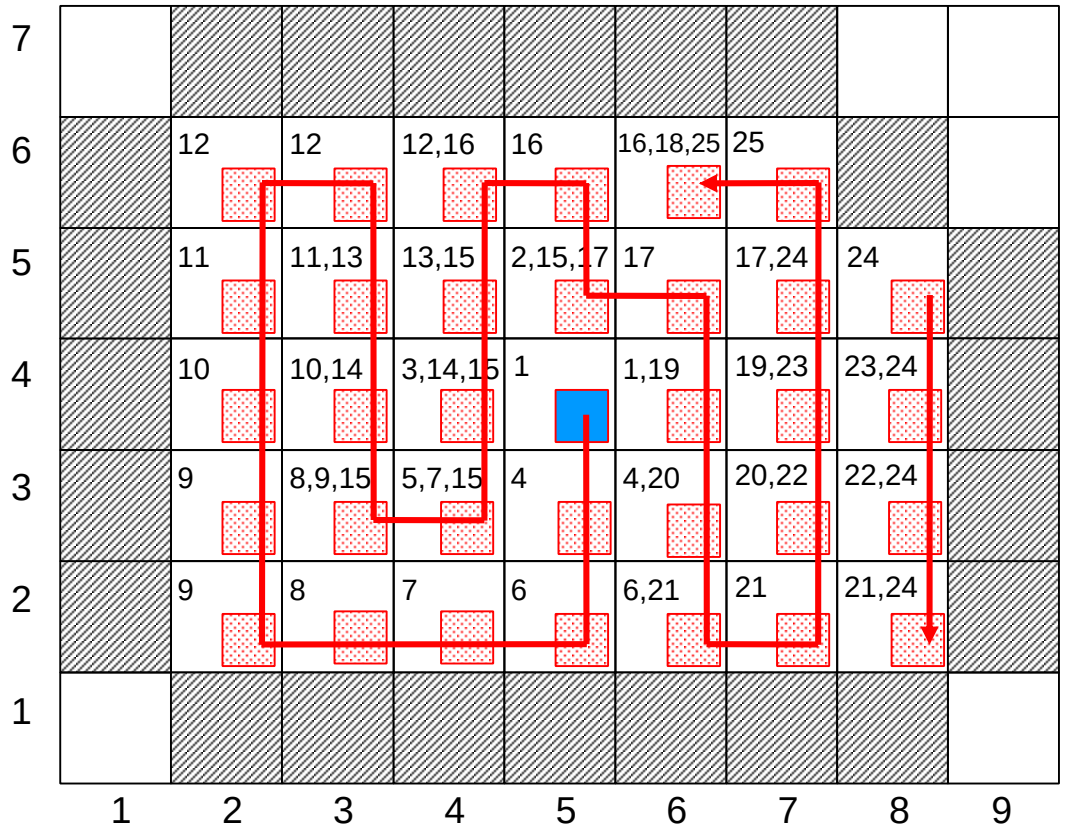


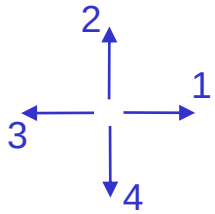

```
Empty(stack);
Push(stack, seed-pixel);

while(stack not empty) {
  pixel = Pop(stack);
  setColor(pixel, FillColor);
  for (each of the 4-connected pixels p_i) {
    if (! ((p_i == boundary_pixel)
        || (colorOf(p_i) == FillColor))) {
      Push(stack, p_i);
    }
  }
}
```

3.4 Füllalgorithmen

Saatpunkt-Methode / Seed-Fill

– Zahlen: Position der Pixel im Stack

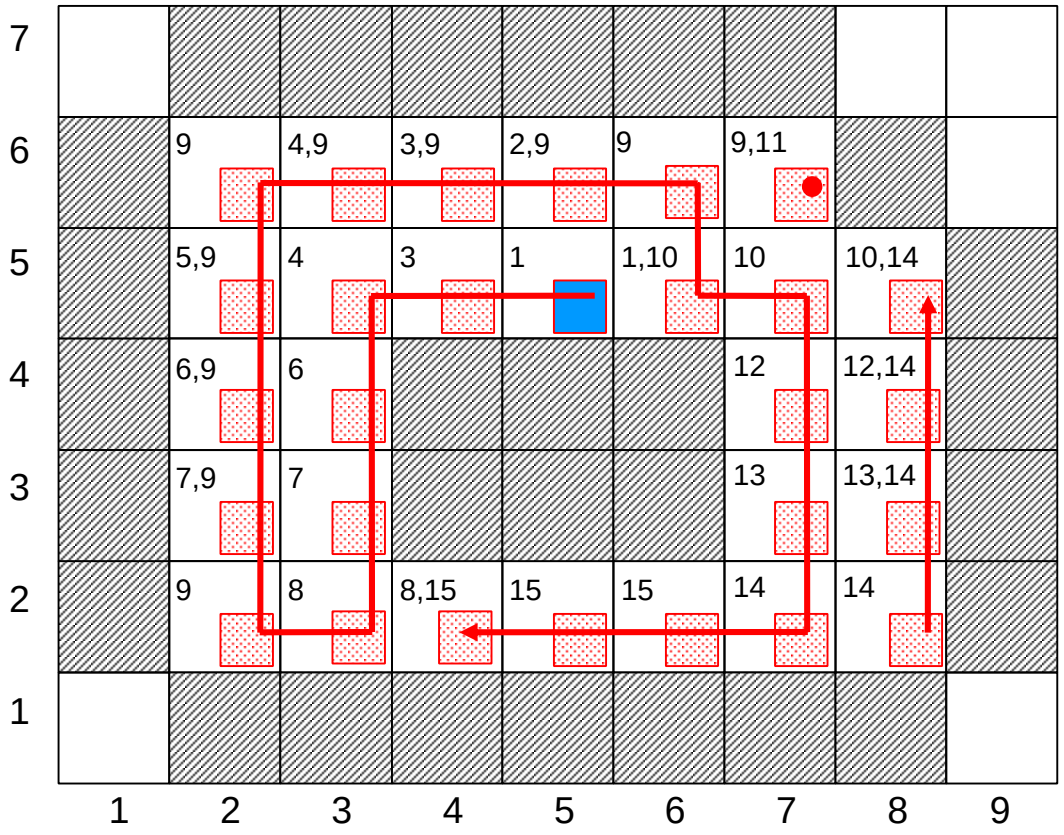


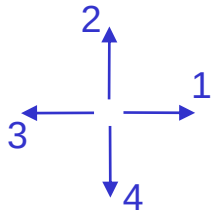


3.4 Füllalgorithmen

Saatpunkt-Methode / Seed-Fill

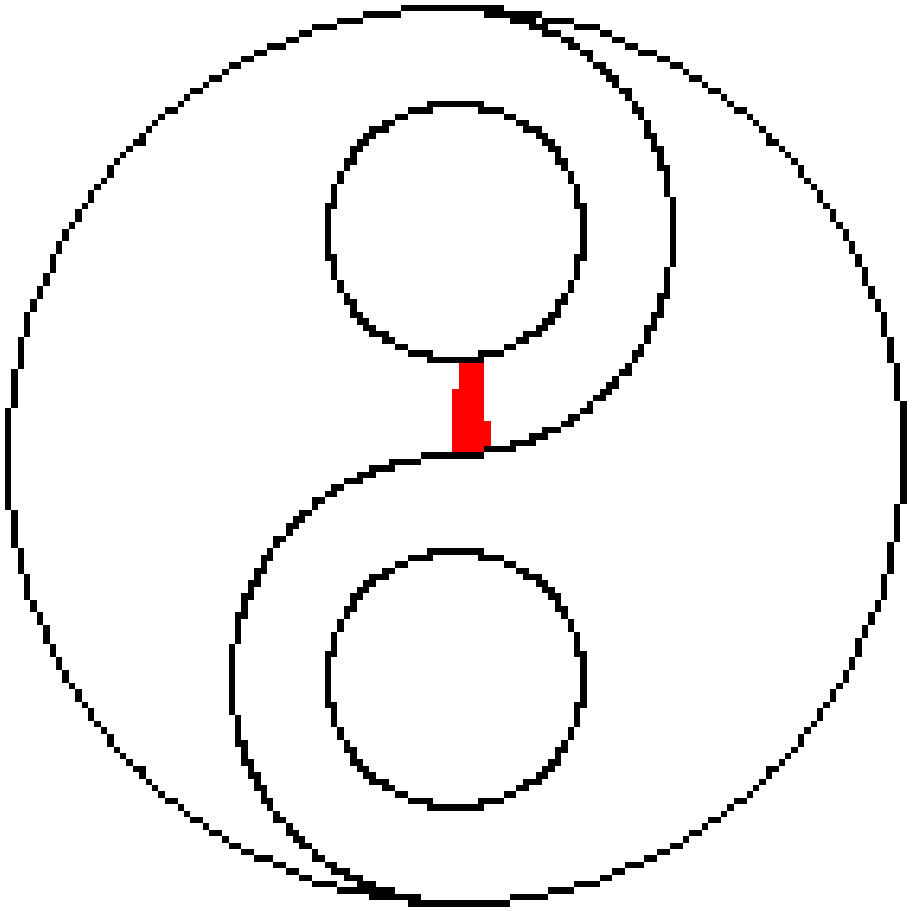
– Gebiet mit Loch (Zahlen: Position der Pixel im Stack)





3.4 Füllalgorithmen

Saatpunkt-Methode / Seed-Fill



3.4 Füllalgorithmen

- Hybrid-Methoden
  - verwenden die Ideen der Scan-Line- und Saatpunkt-Methoden gemeinsam
  - Scan-Line-Seed-Fill-Algorithmus

3.5 Aliasing

Aliasing (Signaltheorie)

- Allgemein versteht man unter Aliasing-Effekten
  - die **fehlerhafte Rekonstruktion** eines (kontinuierlichen) Ausgangssignals durch eine Abtastung mit **zu geringer Frequenz** (vgl. Nyquist-Theorem)
- Im Frequenzbereich **bandbegrenzte Signale** müssen mit **mehr als der doppelten Grenzfrequenz** abgetastet werden, um eine exakte Rekonstruktion zu ermöglichen

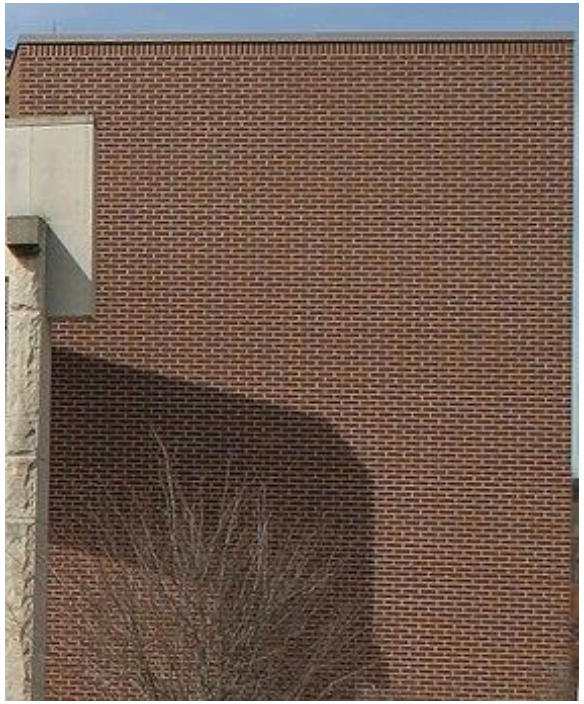
Aliasing (Computergraphik)

- Visuelle Artefakte durch
  - **Unterabtastung** (z.B. bei einem Schachbrettmuster)
  - **Rasterkonvertierungseffekte** (z.B. Treppeneffekte bei schrägen Linien)
  - **Örtliches und zeitliches Aliasing** (z.B. scheinbar rückwärts drehende Räder)

3.5 Aliasing

Anti-Aliasing-Verfahren

- Methoden, um Aliasing-Effekten entgegenzuwirken:
  - Überabtastung
  - Filterung
- Ein echtes „Beseitigen“ ist oft (schon theoretisch) **nicht möglich,**
  - wenn die Signale **nicht bandbegrenzt** sind.
  - Eine höhere Abtastfrequenz - also **Überabtastung** vermindert die Alias-Effekte, beseitigt sie aber nicht vollständig.
- Bei Artefakten der **Rasterkonvertierung** spricht man von „Verfahren zur (Bild-)**Kantenglättung**“.



3.5 Aliasing

Aliasing-Artefakte in der Computergraphik:

- Textur-Artefakte (z.B. Schachbrettmuster)
- Treppeneffekte beim Rastern von Kurven: **Jagged Edges**
- Verschwinden von Objekten, die kleiner als ein Pixel sind
- Verschwinden von langen, dünnen Objekten
- Detailverlust bei komplexen Bildern
- „Aufblinken“ kleiner Objekte bei Bewegungen / Animationen: **Popping**

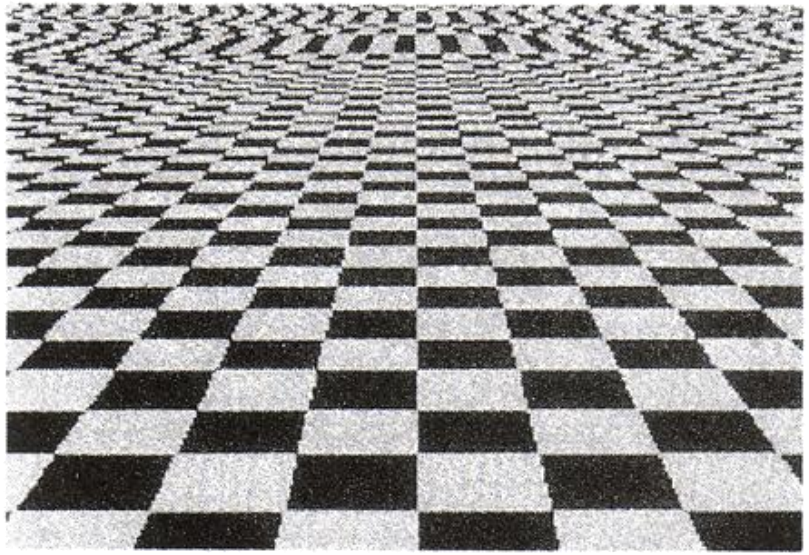
- Üblicherweise treten visuelle Artefakte auf, wenn die **Periodizität** (zum Beispiel Kachelmuster) in der Textur die **Größenordnung von Pixeln** erreicht.

- Es gilt: „Echtes“ Aliasing kann in Computergraphikbildern mittels Überabtastung nicht entfernt, sondern nur verbessert werden (**nicht bandbegrenzt**).

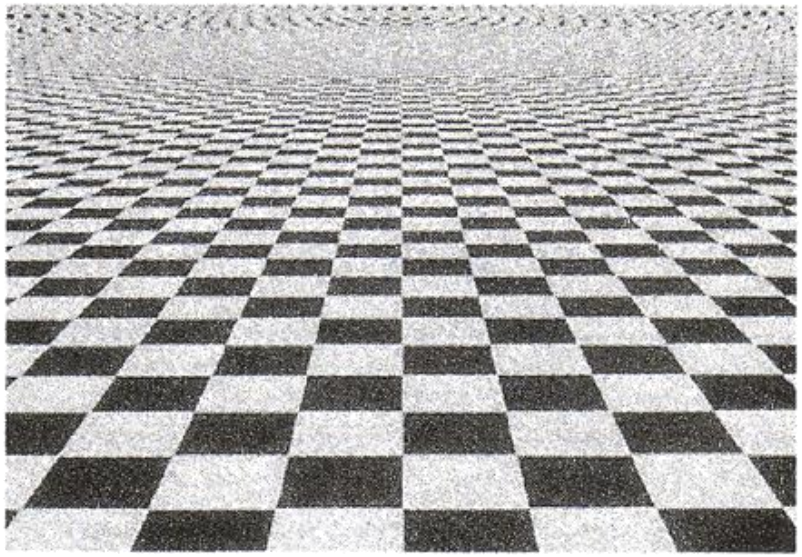
3.5 Aliasing

Textur-Artefakte, unendliches Schachbrettmuster

- Am oberen Ende werden die Quadrate zunächst immer kleiner und dann wieder größer (Aliasing)
- Dies ist ein Ergebnis zu grober Abtastung



- Mittels **zweifacher Überabtastung** (Abtastung mit doppelter Frequenz, d.h. vierfacher Rechenaufwand) können die **Artefakte verringert** werden
- Sie treten aber bei höheren Frequenzen immer noch auf (hier: später, weiter oben)



3.5 Aliasing

Treppeneffekte, Jagged Edges, Jaggies

- Die bisher beschriebenen Verfahren zur Rasterung von Geraden und Kurven **erzeugen Treppeneffekte.**
- Zeichnen von Punkten nur **an Rasterpositionen** möglich
- Diese entsprechen im Allgemeinen **nicht den tatsächlichen** Positionen (Sollpositionen).
- Um solchen Aliasing-Effekten entgegenzuwirken, werden **mehrere Intensitäten** zur Erhöhung der visuellen Auflösung benutzt.
- Zum Beispiel verwendet eine Variante des Bresenham-Algorithmus für Geraden (im ersten Oktanten) für jeden x-Wert **zwei Pixel mit Grautönen entsprechend eines Abstandmaßes** zur zu zeichnenden Strecke.

3.5 Aliasing

Treppeneffekte, Jagged Edges, Jaggies

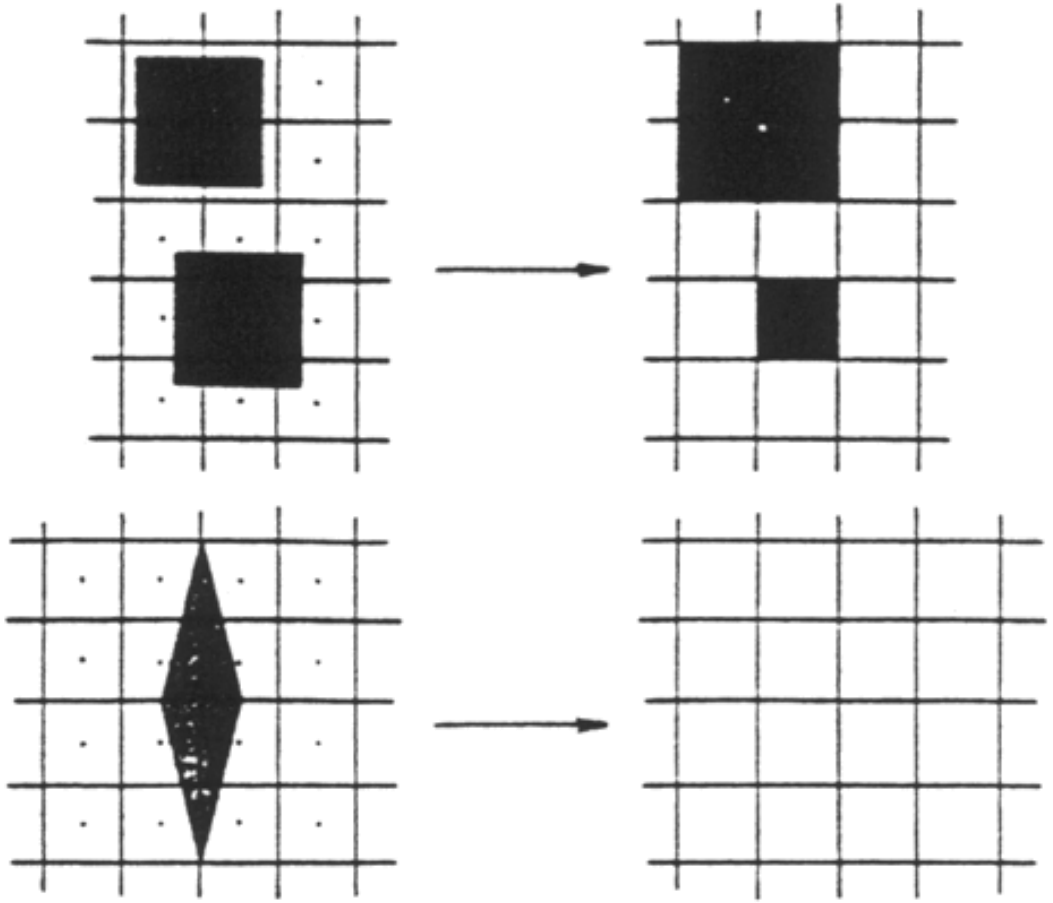
- Treppeneffekte bei einer gerasterten Geraden
- Graustufen, um Treppeneffekte zu verringern

Gegenüberstellung von ungeglättetem und geglättetem Vektor.

Gegenüberstellung von ungeglättetem und geglättetem Vektor.

3.5 Aliasing

Aliasing bei Polygonen



3.5 Aliasing

Anti-Aliasing Verfahren: Oversampling

- Ein einfaches globales Anti-Aliasing-Verfahren
- Andere Bezeichnungen
  - Überabtastung
  - Supersampling
- Jedes Pixel wird mit einer höheren Auflösung berechnet, als es schließlich dargestellt wird.
- Das Pixel erhält als eigentliche Grauwertintensität bzw. Farbwert einen gewichteten Durchschnitt der an ihm beteiligten Subpixelwerte.
- Dieses Vorgehen entspricht allgemein einem Filterprozess, dessen theoretische Grundlagen in der Digitalen Signalverarbeitung begründet liegen.

3.5 Aliasing

Anti-Aliasing Verfahren (Filterkerne [Crow, 1981])

3 x 3

Faktor: 1/16

1	2	1
2	4	2
1	2	1

5 x 5

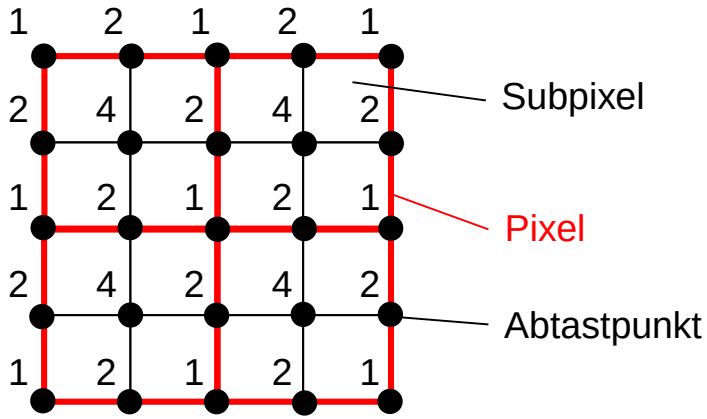
Faktor: 1/81

1	2	3	2	1
2	4	6	4	2
3	6	9	6	3
2	4	6	4	2
1	2	3	2	1

7 x 7

Faktor: 1/256

1	2	3	4	3	2	1
2	4	6	8	6	4	2
3	6	9	12	9	6	3
4	8	12	16	12	8	4
3	6	9	12	9	6	3
2	4	6	8	6	4	2
1	2	3	4	3	2	1

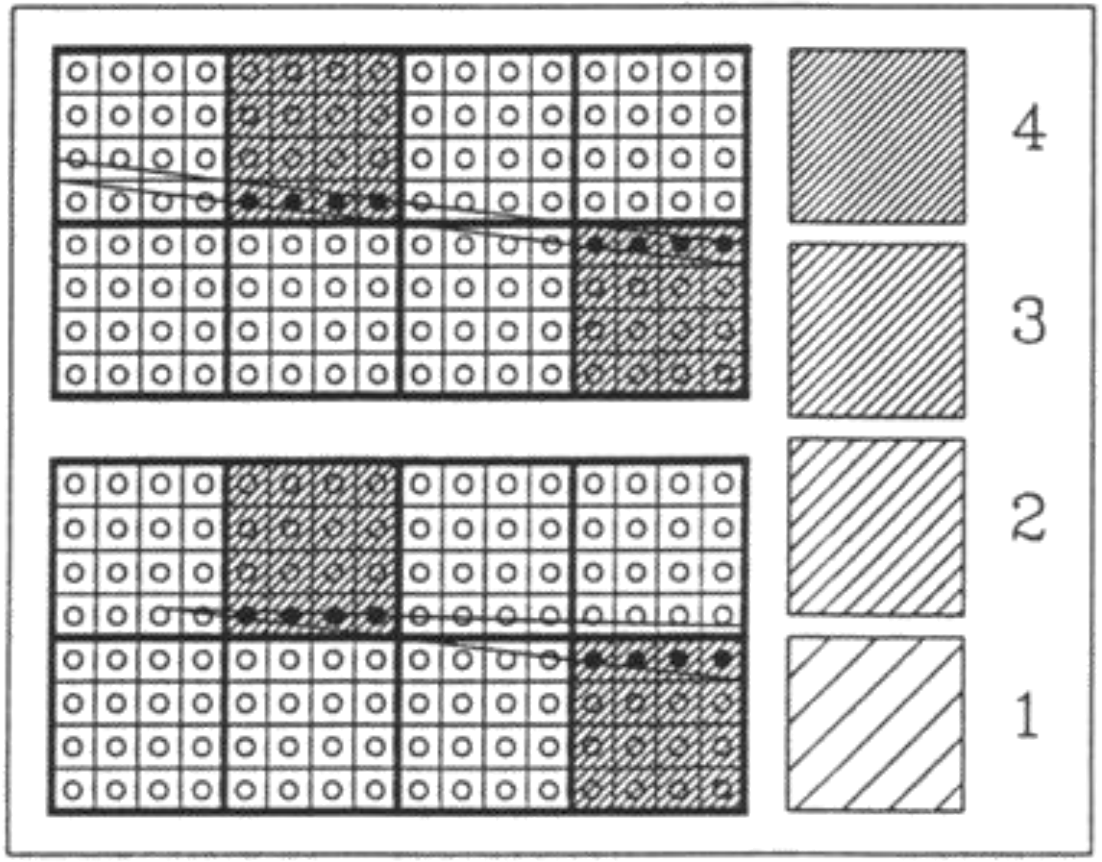


3.5 Aliasing

Anti-Aliasing Verfahren

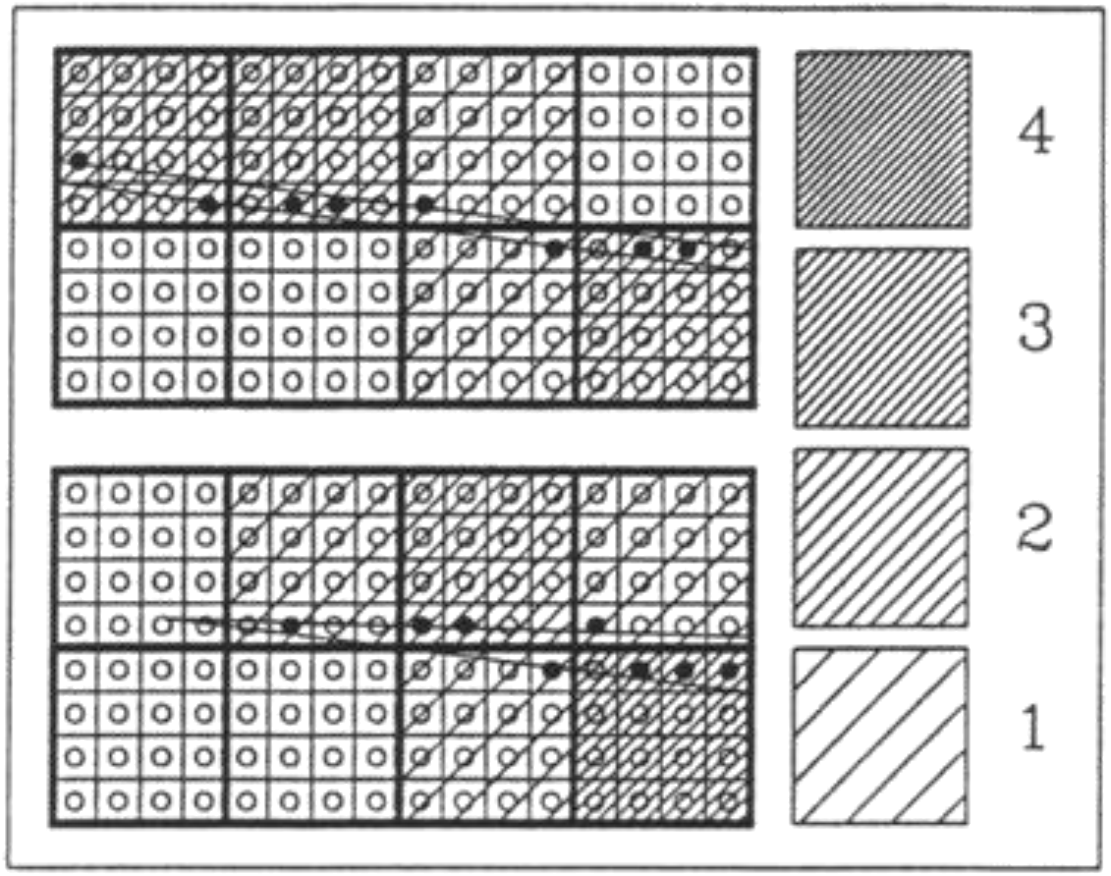
- Bei **Linien und spitzen Dreiecken (dünnen Polygonen)** kann es trotz Supersamplings zu überraschenden Effekten kommen.

- Gesetzte Subpixel sind schwarz.
- Die Schraffur zeigt den Grauwert abhängig von der Anzahl der gesetzten Subpixel.
- Oben:
  Linie verschwindet stellenweise,
  da keine Subpixel getroffen werden!
- Unten:
  Analog für das Dreieck



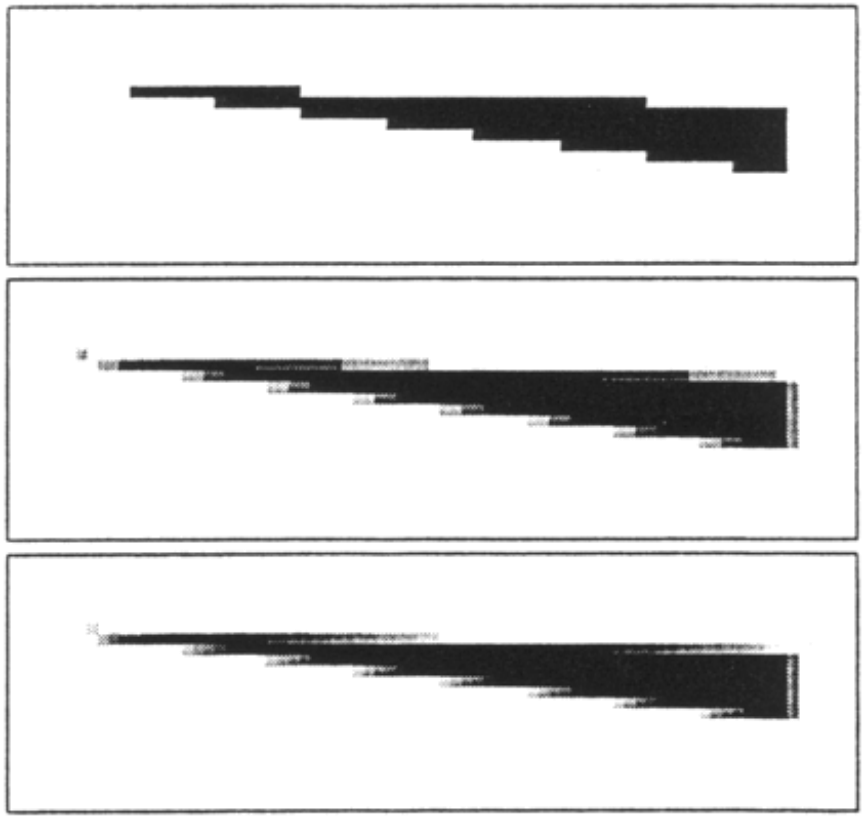
Anti-Aliasing Verfahren

- Abhilfe für dieses Problem schafft erst eine (korrekte) **Berechnung der überdeckten Fläche** im Pixel.
- Die praktische Anwendung dieser Methode schließt allerdings eine **exakte analytische Berechnung** der wirklich im Pixel überdeckten Fläche **aus.**
- Es existieren hierzu verschiedenste **Näherungsverfahren.**
  - Die überdeckte Fläche lässt sich auch durch eine Anzahl entsprechend gesetzter Subpixel angenähert darstellen.



3.5 Aliasing

Anti-Aliasing Verfahren (Leistungsfähigkeit der vorgestellten Verfahren)



Spitzes Dreieck ohne Glättung

Spitzes Dreieck mit Oversampling

Spitzes Dreieck mit korrekter Berechnung der überdeckten Fläche

Anti-Aliasing Verfahren: Stochastische Methoden

- Beim stochastischen Abtasten wird das Oversampling mittels Monte-Carlo-Methoden durchgeführt, d.h. die Grauwerte (Intensität) werden an einigen zufälligen Punkten im Pixel ermittelt und das Ergebnis gemittelt.
- Auch bei der Berechnung der vom Polygon im Pixel überdeckten Fläche können Monte-Carlo-Methoden eingesetzt werden.
- Stochastische Methoden
  - erhöhen die Effizienz (schnellere Berechnung)
  - unterdrücken systematische Probleme (z.B. dünne Linie)
  - erlauben adaptives Supersampling
  - neigen aber z.B. zu Problemen bei Animationen, da Objekte flimmern können

Quellen

- Computergraphik,
  Universität Leipzig
  (Prof. D. Bartz)

- Graphische Datenverarbeitung I,
  Universität Tübingen
  (Prof. W. Straßer)

- Graphische Datenverarbeitung I,
  TU Darmstadt
  (Prof. M. Alexa)

- Computergraphik,
  TU Kaiserslautern
  (Prof. H. Hagen)

UNIVERSITÄT LEIPZIG

Polygonale Repräsentation

COMPUTERGRAPHIK

Inhaltsverzeichnis

4. Polygonale Repräsentation

4.1 Einleitung

4.2 Polygonale Repräsentation

4.3 Level-of-Detail Ansätze

4.1 Überblick

Ziel dreidimensionaler Computergraphik

- Erzeugung (zweidimensionaler) Darstellungen einer Szene oder eines Objektes ausgehend von Beschreibungen oder Modellen
- Die Art und die Verwendung der Computer-internen Repräsentation eines Objektes hängt dabei von vielen Einflussfaktoren ab.

4.1 Überblick

Einflussfaktoren

- Das Objekt kann real oder nur in der Computerdarstellung existieren.
- Die Erstellung des Objektes ist eng mit seiner Visualisierung verknüpft (Interaktive CAD-Systeme).
  - Modellierung und Visualisierung als Werkzeuge beim Herstellungsprozess
  - Visualisierung
    - 2D
    - 3D
  - Herstellung
    - 3D Drucker
    - Ansteuerung einer Fräse

- Genauigkeit
  - **Exakte** Beschreibung von Geometrie und Form in CAD-Applikationen
  - Für einen Renderer **ausreichende approximative** Beschreibung
  - Bei interaktiven Anwendungen können für ein Objekt
    - Gleichzeitig **mehrere interne** Repräsentationen existieren.
    - Repräsentationen bei Bedarf **dynamisch erzeugt** werden.

    $\Rightarrow$ Level-of-Detail (LOD) Verfahren

4.1 Überblick

Aspekte von Modellierung und Repräsentation

- Erzeugung von 3D Geometriedaten
  - CAD-Interface
  - Laser-Scanner (Reverse Eng.)
  - Analytische Techniken (z.B. Sweeping)
  - Bild (2D) - und Video (3D) -Analyse
- Repräsentation
  - Effizienter Zugriff
  - Konvertierung
- Repräsentationen
  - Polygonnetze (für Rendering am häufigsten genutzt)
  - Finite Elemente (FEM)
  - Constructive Solid Geometry (CSG)
  - B-Rep. („Boundary-Representation“ für CAD-Modelle)
  - Implizit (Isoflächen)
  - Surface Elements (Surfels = Punkte & Normalen)

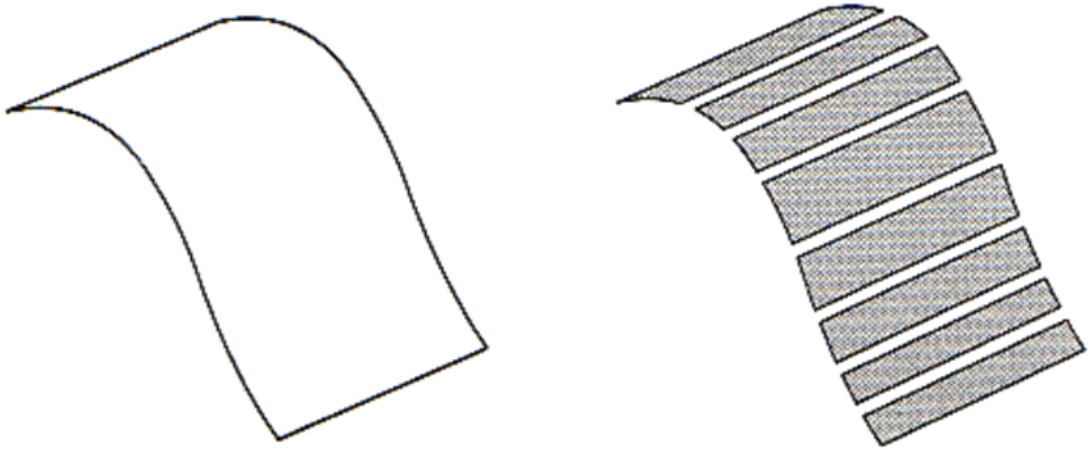
4.1 Überblick

Aspekte von Modellierung und Repräsentation

- Manipulation: Formänderung der Objekte (Editing)
  - Boolesche Operationen
  - Lokale Glättung
  - Interpolation bestimmter Features (Randkurven)
  - „Eingravieren“ geometrischer Details

4.2 Polygonale Repräsentation

- Ein Objekt wird durch ein Netz polygonaler Facetten (oft Dreiecke) repräsentiert.
  $\Rightarrow$ stückweise lineare Approximation
- Die polygonalen Facetten stellen im Allgemeinen eine Approximation gekrümmter Flächen dar, welche das Objekt begrenzen.
- Klassische Repräsentationsform dreidimensionaler Objekte in der Computergraphik



4.2 Polygonale Repräsentation

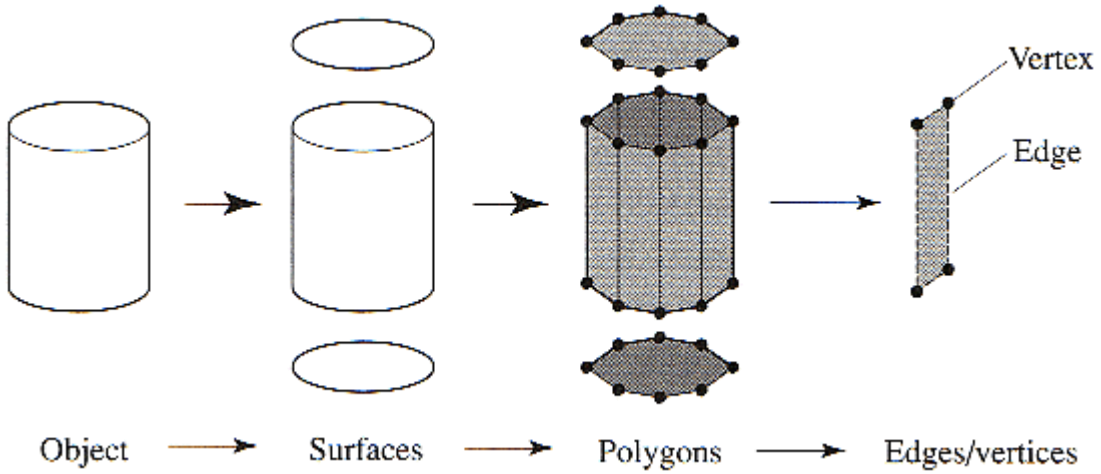
- Genauigkeit der Approximation kann gewählt werden:
  - Anzahl der Polygone
  - Größe der Polygone

- Fragen
  - Welche Polygonauflösung benötigt man für eine genaue Darstellung?
  - Welche Polygonauflösung benötigt ein Renderer, um die stückweise lineare Approximation glatt erscheinen zu lassen?
  - Wie ist der Zusammenhang zwischen Polygonanzahl des Objektes und seiner Größe in der finalen Darstellung?
    - Oft verwendete Grundregel: Polygonauflösung an die lokale Krümmung der Fläche binden

4.2 Polygonale Repräsentation

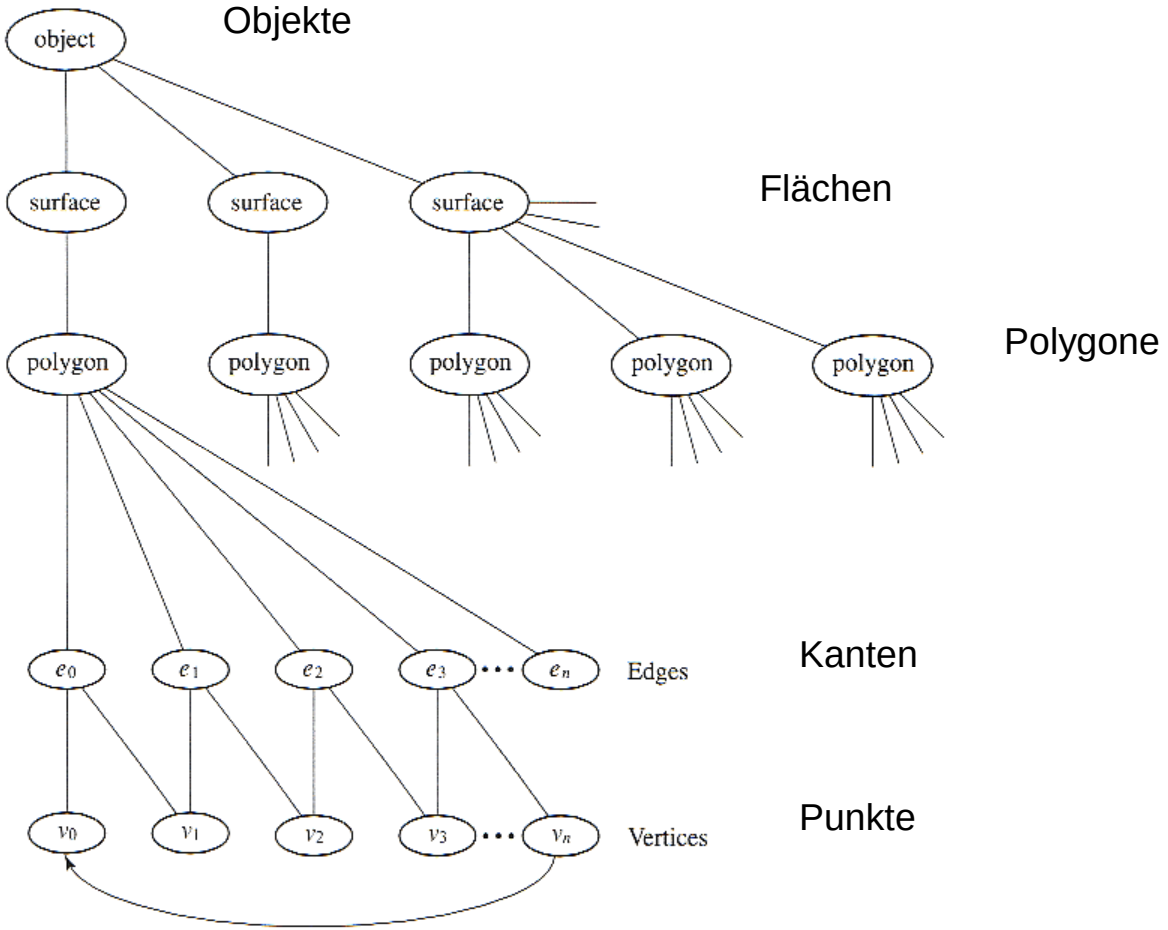
Repräsentationshierarchie (konzeptionell)

- Objekt setzt sich aus Oberflächen zusammen.
- Oberfläche setzt sich aus Polygonen zusammen.
- Polygon besteht aus
  - Eckpunkten (vertices)
  - Kanten (edges)



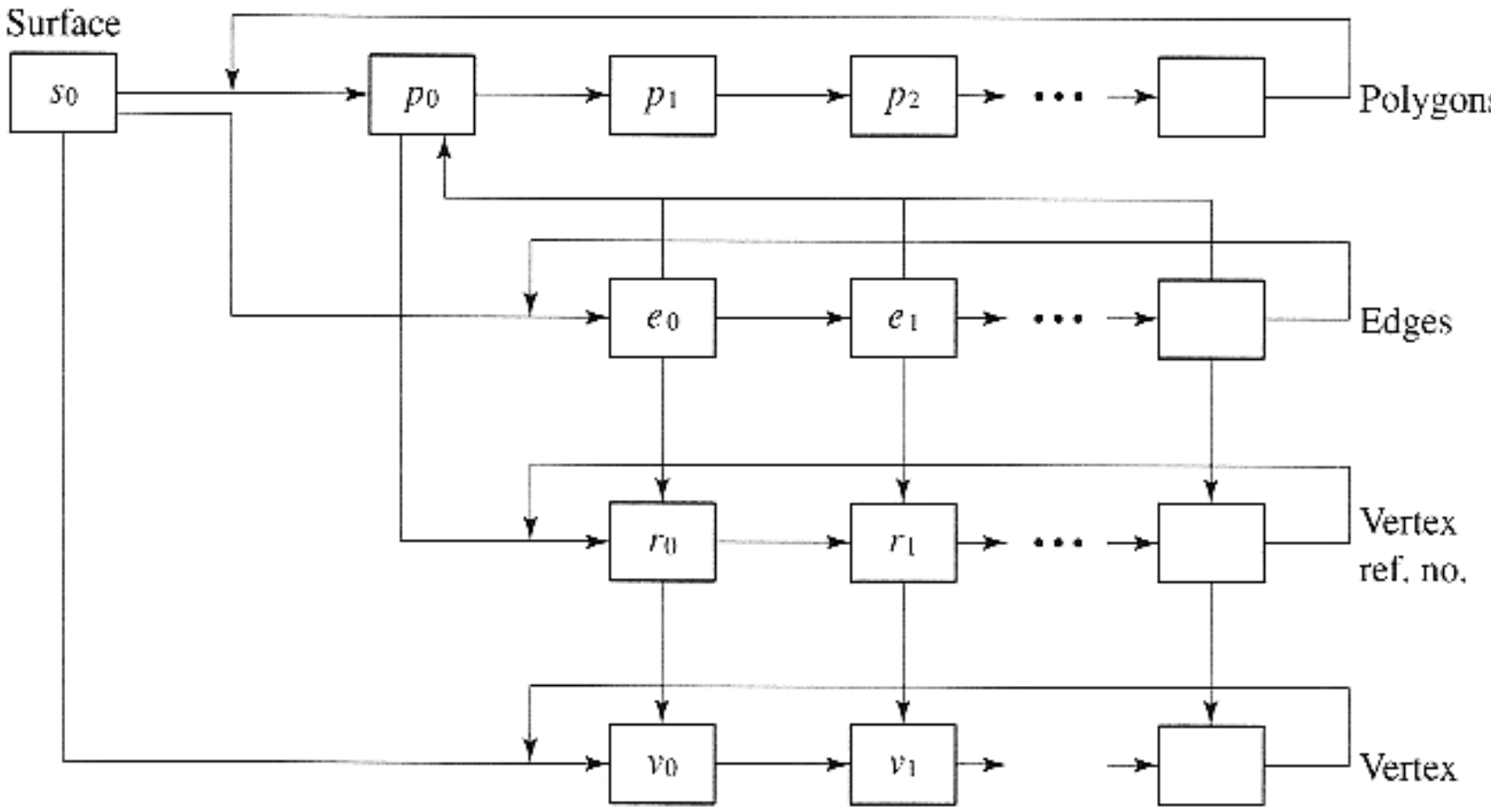
4.2 Polygonale Repräsentation

Repräsentationshierarchie (topologisch)



4.2 Polygonale Repräsentation

Repräsentationshierarchie (Datenstruktur)



Indizes auf Eckpunkte

Eckpunkte werden pro Oberfläche nur einmal gespeichert!

4.2 Polygonale Repräsentation

Kanten

- Offensichtlich existieren in der approximierenden polygonalen Darstellung zwei Arten von Kanten:
  - Scharfe Kanten (Feature Lines)
    Þ diese sollen als Kanten sichtbar bleiben
  - Virtuelle Kanten (im Inneren glatter Flächen)
    Þ diese sollte der Renderer „verschwinden“ lassen
- 70er Jahre: Schattierungsalgorithmen (Interpolative Shading)
  - Flat/Uniform
  - Gouraud
  - Phong Shading

4.2 Polygonale Repräsentation

Datenstruktur

Kann neben der Geometrie spezielle Attribute für Anwendungen und Rendering beinhalten

- Flächenattribute:
  - Repräsentation (Dreieck, Polygon, Freiformfläche)
  - Koeffizienten
  - Polygonnormalen
  - Eigenschaften: planar, konvex, „hat Löcher“
  - Verweis auf Eckpunkte (und ggf. Kanten)
- Kantenattribute:
  - Länge
  - Art: Randkante, Feature Line, virtuelle Kante
  - Ggf. Verweis auf zugehörige Polygone und Eckpunkte
- Eckpunktattribute:
  - Eckpunktnormale (gemittelte Polygonnormalen)
  - Farbe
  - Texturkoordinaten
  - Ggf. Verweis auf Polygone und Kanten

4.2 Polygonale Repräsentation

Erzeugung polygonaler Objekte: **manuelle** Verfahren

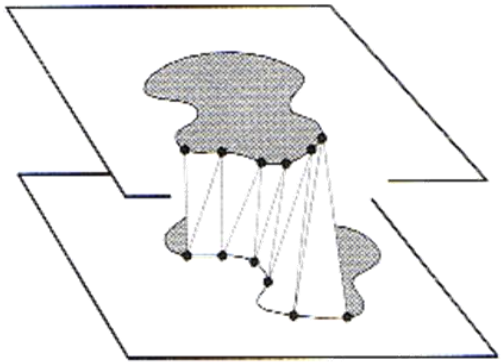
- Verschieben von (Gruppen von) Eckpunkten mittels dreidimensionaler Eingabegeräte oder Schnittstellen
  - Komplex, schwer handhabbar
  - Nur für einfache Objekte bzw. für einfache „Manipulationen“ geeignet
- 3D-Digitizer
  - Manuelles Anbringen von Punkten auf Objekten, die mittels Digitalisierer zu Polygon-Eckpunkten werden sollen
  - Beispiel:
    Netze über Objektoberflächen „ziehen“
    $\Rightarrow$ erste 3D-Darstellungen von Karosserien (1974)

4.2 Polygonale Repräsentation

Erzeugung polygonaler Objekte: **automatische** Verfahren

- Laserscanner
  - Objekt wird rundherum scheibchenweise mit einem Laserstrahl abgetastet; dieser misst den Abstand zur Objektoberfläche
  - Aus den gemessenen 2D-Konturen werden mittels eines „skinning“-Algorithmus, der geeignet benachbarte Punkte verbindet, Dreiecksflächen erzeugt (Abb. (a))

(a)



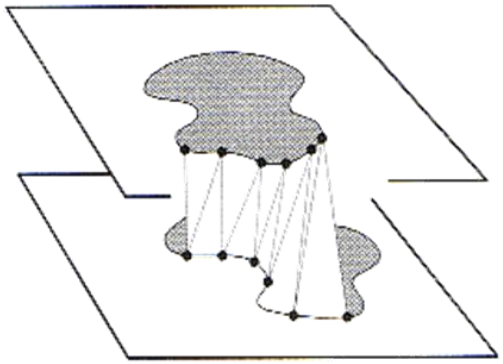
(b)

4.2 Polygonale Repräsentation

Erzeugung polygonaler Objekte: **automatische** Verfahren

- Laserscanner
  - Anwendungen:
    - Reverse Engineering
    - Virtuelle Bekleidung
    - etc.
  - Probleme:
    - Ist das Objekt stellenweise „zu konkav“, gibt es Flächen, die vom Laserstrahl nicht erfasst werden können.
    - Dieser Ansatz tendiert dazu, (zu) viele Dreiecke zu erzeugen!
      (Abb. (b): 400.000 Dreiecke)

(a)



(b)

4.2 Polygonale Repräsentation

Erzeugung polygonaler Objekte: **mathematische** Verfahren

- Erzeugung von polygonalen Darstellungen aus analytischen Kurven und Flächen
  $\Rightarrow$ CAD-Anwendungen

- Vorteile:
  - Benutzer arbeitet mit high-level Objektbeschreibung.
  - Objektform ist direkt mit mathematisch exakter Objektbeschreibung gekoppelt.

- Beispiele:
  - Parameterflächen (stückweise Polynome)
  - Rotationsflächen
  - Sweep-Flächen

4.2 Polygonale Repräsentation

Erzeugung polygonaler Objekte: **prozedurale** Verfahren

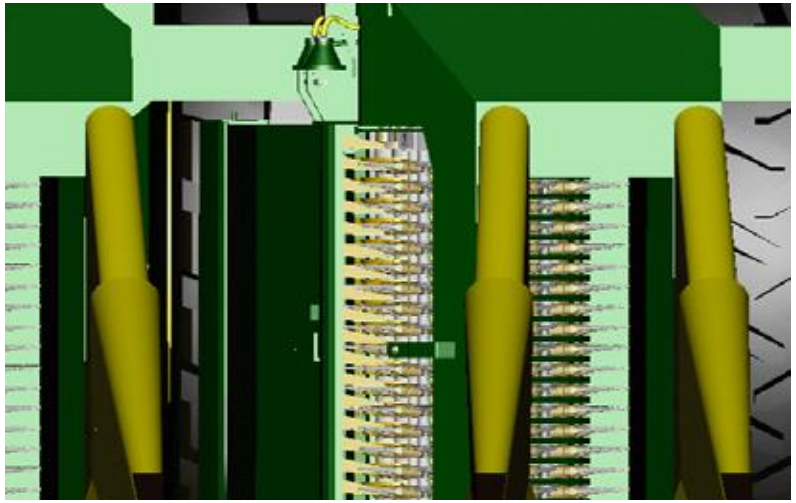
- Erzeugung polygonaler Objekte durch Fraktale
  - Fraktale (Fractals) gehen in ihrem theoretischen Ansatz auf die Mandelbrot-Geometrie zurück.
  - Werden u. a. für die Modellierungen von geographischen Höhenfeldern (Terrain Models) eingesetzt
  - Fraktale finden aufgrund ihrer Effizienz z.B. Anwendung in professionellen Flugsimulatoren für das Pilotentraining.
- Erzeugung polygonaler Objekte durch Ersetzungssysteme, z. B. Grammatiken
  - Lindenmayer-/L-Systeme zur Beschreibung von biologischen Entwicklungen
  - Werden u. a. für die Modellierungen von Bäumen, Pflanzen, etc. eingesetzt

4.3 Level-of-detail Ansätze

- Motivation
  - Allgemein tendieren Verfahren zur Erzeugung polygonaler Modelle dazu, „zu viele“ Polygone zu produzieren.
- Probleme
  - In den überwiegenden Fällen ist das Verhältnis (Polygonanzahl des Objektes) / (projizierte Fläche des Objekts) viel zu groß.
  - Overhead bei der Speicherung, Übertragung, Bearbeitung und Visualisierung „unnötiger“ Polygone
- Lösung
  - Verschiedene polygonale Auflösungen der Objektrepräsentation: Level of Detail (LOD)
  - Diese werden als sogenannte „Detail Pyramid“ / Multiresolution-repräsentation verwaltet.

4.3 Level-of-detail Ansätze

– Von ca. 10,5 Millionen Dreiecken werden nur 550.000 dargestellt.



4.3 Level-of-detail Ansätze

- Diskrete LODs, Pyramide mit verschiedenen Netzen, abhängig von der Entfernung
  - Popping
  - Alpha blending, morphing
- Kontinuierliche LODs, vordefinierte Einzeloperationen, Entfernen Kante für Kante
  - Flexibel
  - Teurer
  - Kann minimales Modell finden
  - Blickpunktabhängige Modelle möglich
- Statische LODs
  Feste, vordefinierte Auflösungsstufen
  - Kosten Speicher
- Dynamische LODs,
  Online Berechnung
  - Kosten Rechenzeit

Quellen

- Computergraphik, Universität Leipzig (Prof. Dr. D. Bartz)
- Graphische Datenverarbeitung I, TU Kaiserslautern (Prof. Dr. H. Hagen)
- Graphische Datenverarbeitung I, Universität Tübingen (Prof. Dr. W. Straßer)
- Graphische Datenverarbeitung I, TU Darmstadt (Prof. Dr. M. Alexa)

UNIVERSITÄT LEIPZIG

Transformationen

COMPUTERGRAPHIK

Inhaltsverzeichnis

5. Transformationen

5.1 Koordinatentransformationen

5.2 Transformationen in der Ebene

5.3 Transformationen im Raum

5.1 Koordinatentransformationen

Koordinatensysteme

- Koordinatensystem des Objekts
  - Oft über geometrische Eigenschaften des Objektes festgelegt
    - Ausgezeichnete Richtungen
    - Symmetrien
- Koordinatensystem des Geräts
  - Bildschirm
  - Bildfenster
  - Nullpunkt in der linken, oberen Ecke
  - x- und y-Achsen parallel zu den Bildrändern

5.1 Koordinatentransformationen

Koordinatensysteme

Weltkoordinaten (3D, $\mathbb{R}^3$)

$\Downarrow$ **1)**

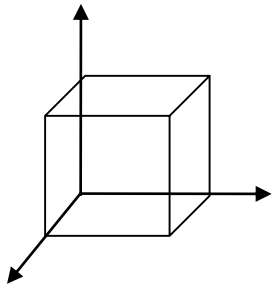
Beobachterkoordinaten (3D, $\mathbb{R}^3$)

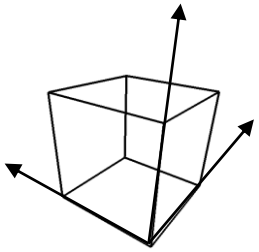
$\Downarrow$ **2)**

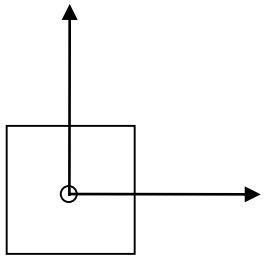
Normalisierte Koordinaten (3D, $[-1; 1]^3$)

$\Downarrow$ **3)**

Bildschirmkoordinaten (2D)







5.1 Koordinatentransformationen

- Grundlage der Bildgestaltung auf dem Bildschirm oder dem Ausgabegerät sind Koordinatentransformationen im
  - $\mathbb{R}^2$
  - $\mathbb{R}^3$
- Transformieren das Objektsystem in das Gerätesystem
- Koordinatentransformationen:
  - Verschiebungen (translation)
  - Drehungen (rotation)
  - Skalierungen (scaling)
- Voraussetzung: Orthonormierte (kartesische) Koordinatensysteme

5.1 Koordinatentransformationen

Allgemeine Vorgehensweise bei der Koordinatentransformation

1) Bildschirm oder Ausgabegerät mit einem Koordinatensystem versehen
2) Objekt mit einem Koordinatensystem versehen
3) Objekt- und Objektkoordinatensystem mittels Parallel- oder Zentralprojektion in Bildebene abbilden ($\mathbb{R}^3 \rightarrow \mathbb{R}^2$ Transformation)
4) Anpassung des Koordinatensystems der Bildebene an das Koordinatensystem des Bildschirmes: Koordinatentransformation ($\mathbb{R}^2 \rightarrow \mathbb{R}^2$Transformation)

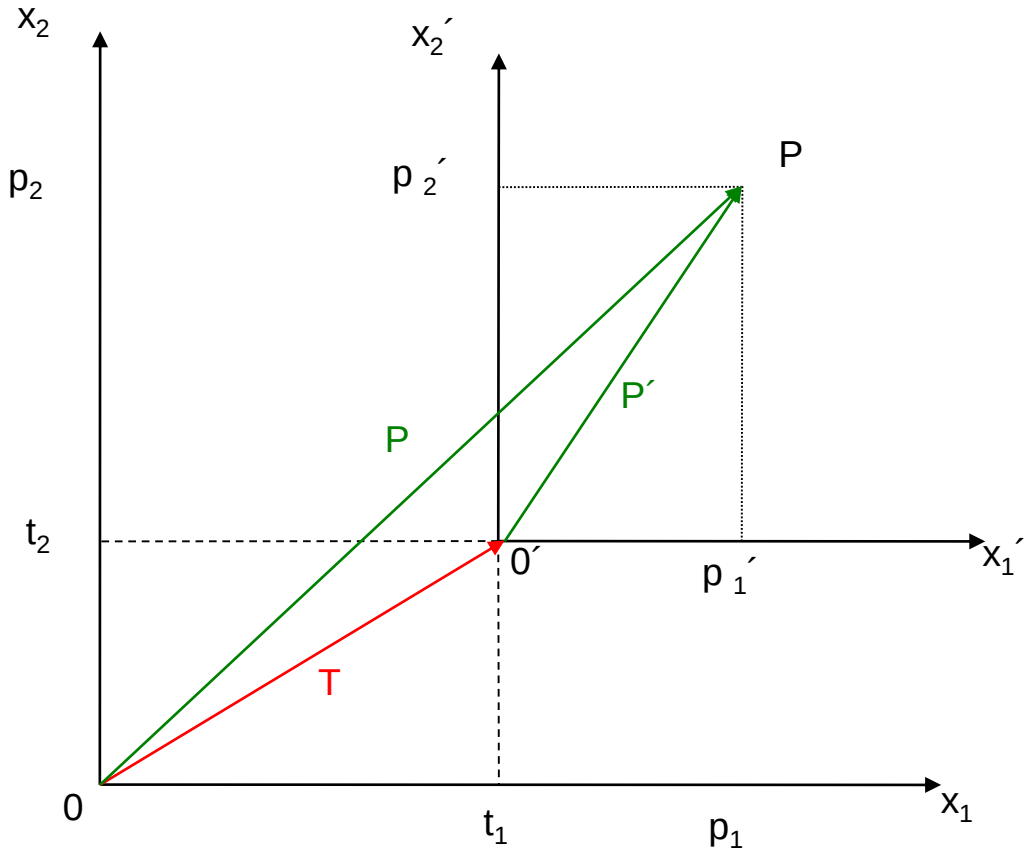
5.2 $\mathbb{R}^2 \rightarrow \mathbb{R}^2$ Koordinatentransformationen

- Gegeben seien im Folgenden die beiden Koordinatensysteme
  - S durch $(0; x_1, x_2)$
    (z. B. Gerätesystem)
  - S' durch $(0'; x_1', x_2')$
    (z. B. Objektsystem)

5.2 $\mathbb{R}^2 \rightarrow \mathbb{R}^2$ Koordinatentransformationen

Verschiebung (translation)

- Die einfachste Transformation zwischen dem System S' und S ist eine Verschiebung.
- Voraussetzung:
  die beiden (gerichteten) Koordinatenachsen sind jeweils parallel zueinander

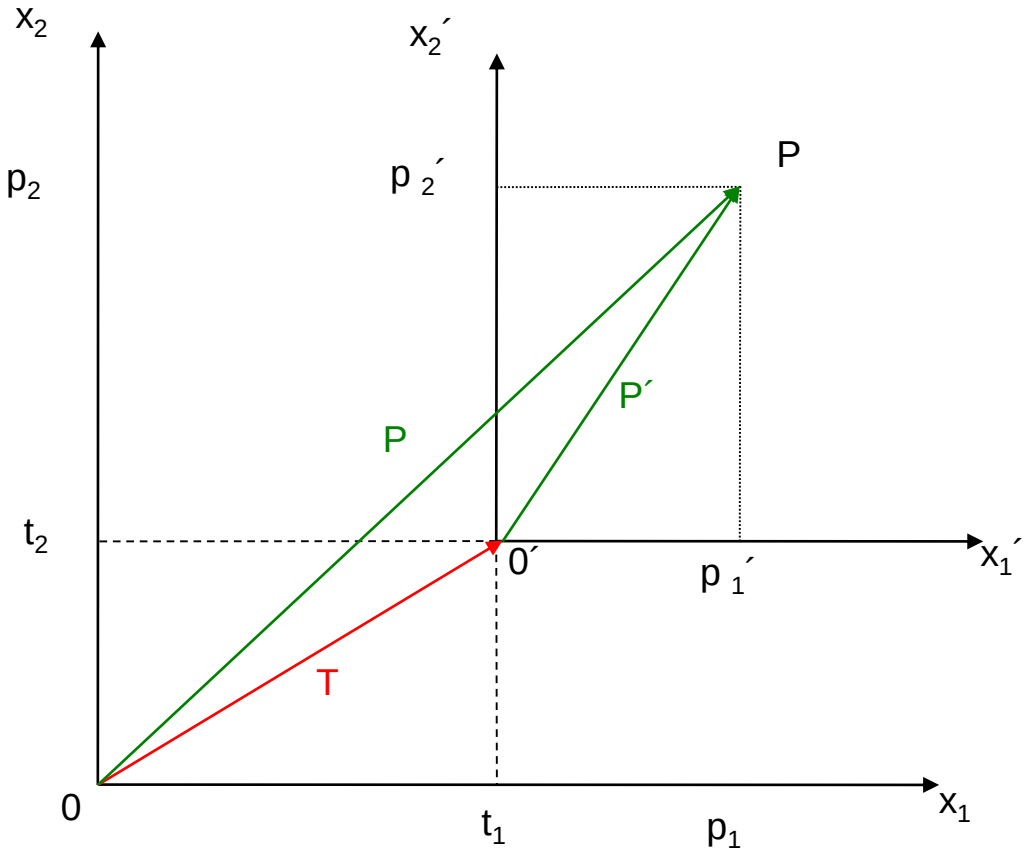


5.2 $\mathbb{R}^2 \rightarrow \mathbb{R}^2$ Koordinatentransformationen

Verschiebung (translation)

- Seien $(t_1, t_2) = 0'$ die Koordinaten des Ursprungs $0'$ von S' im System S
- Der Punkt P hat die Koordinaten
  - (p_1', p_2') in S'
  - $(t_1 + p_1', t_2 + p_2')$ in S
- Also:

$$\begin{aligned} p_1 &= t_1 + p_1' \\ p_2 &= t_2 + p_2' \end{aligned}$$

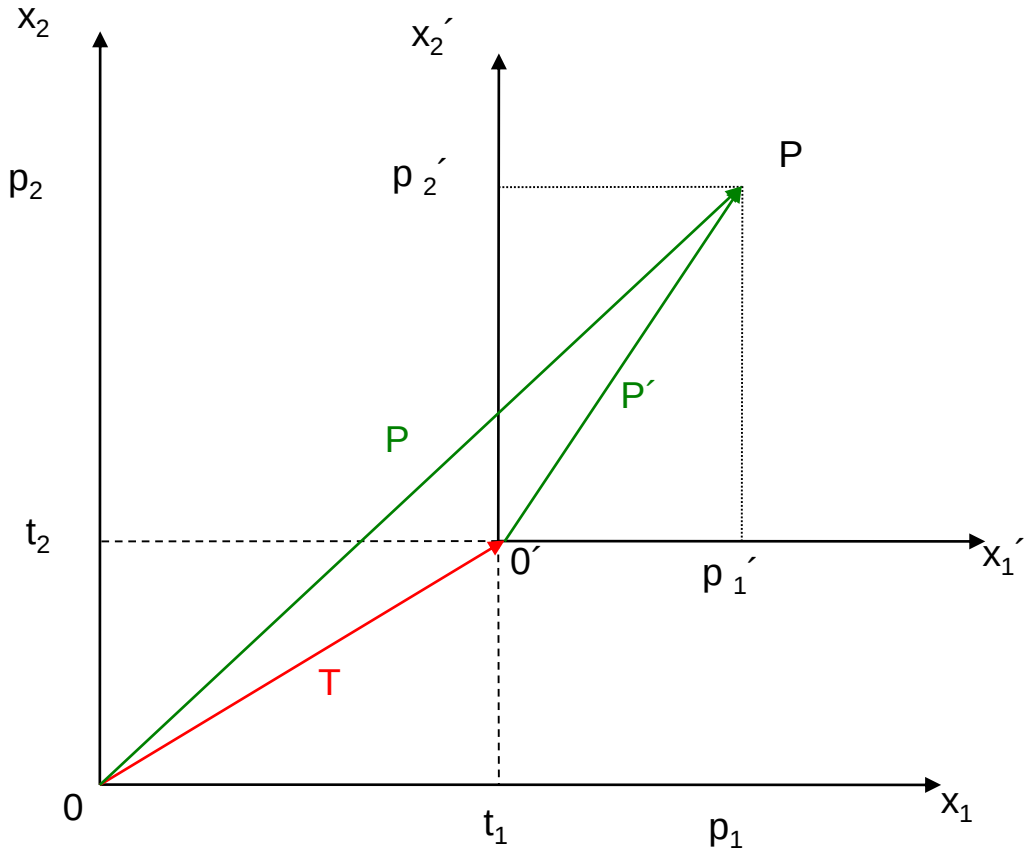


5.2 $\mathbb{R}^2 \rightarrow \mathbb{R}^2$ Koordinatentransformationen

Verschiebung (translation)

$$\begin{pmatrix} p_1 \\ p_2 \end{pmatrix} = \begin{pmatrix} t_1 \\ t_2 \end{pmatrix} + \begin{pmatrix} p_1' \\ p_2' \end{pmatrix}$$

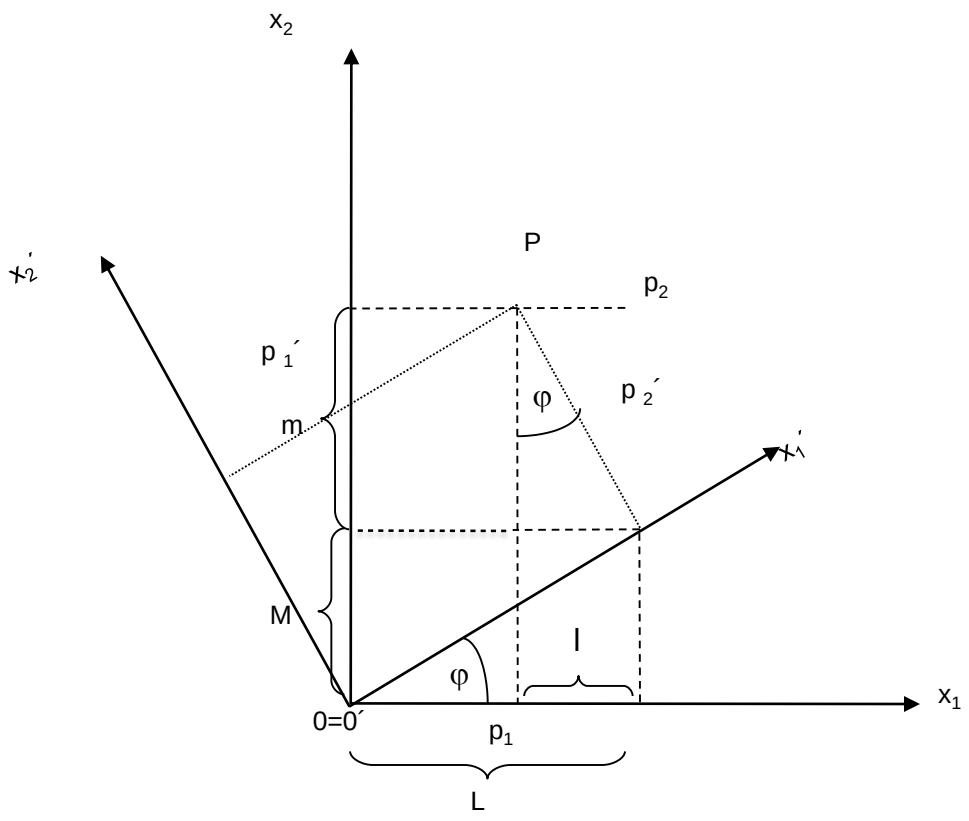
$$P = T + P'$$



5.2 $\mathbb{R}^2 \rightarrow \mathbb{R}^2$ Koordinatentransformationen

Drehung (rotation)

- Wir betrachten die Drehung eines Systems S' gegen das System S um
  - den gemeinsamen Ursprung
    $$0 = 0'$$
  - den Winkel φ

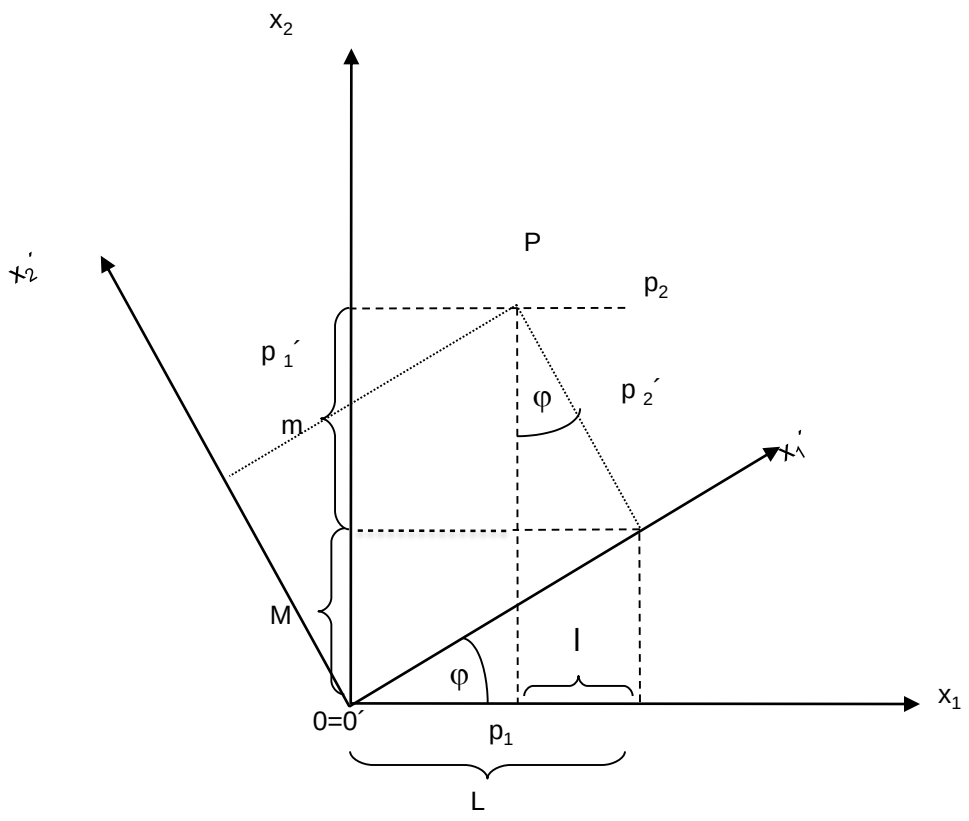


5.2 $\mathbb{R}^2 \rightarrow \mathbb{R}^2$ Koordinatentransformationen

Drehung (rotation)

$$\frac{L}{p_1'} = \cos(\phi) \quad \frac{M}{p_1'} = \sin(\phi)$$

$$\frac{l}{p_2'} = \sin(\phi) \quad \frac{m}{p_2'} = \cos(\phi)$$

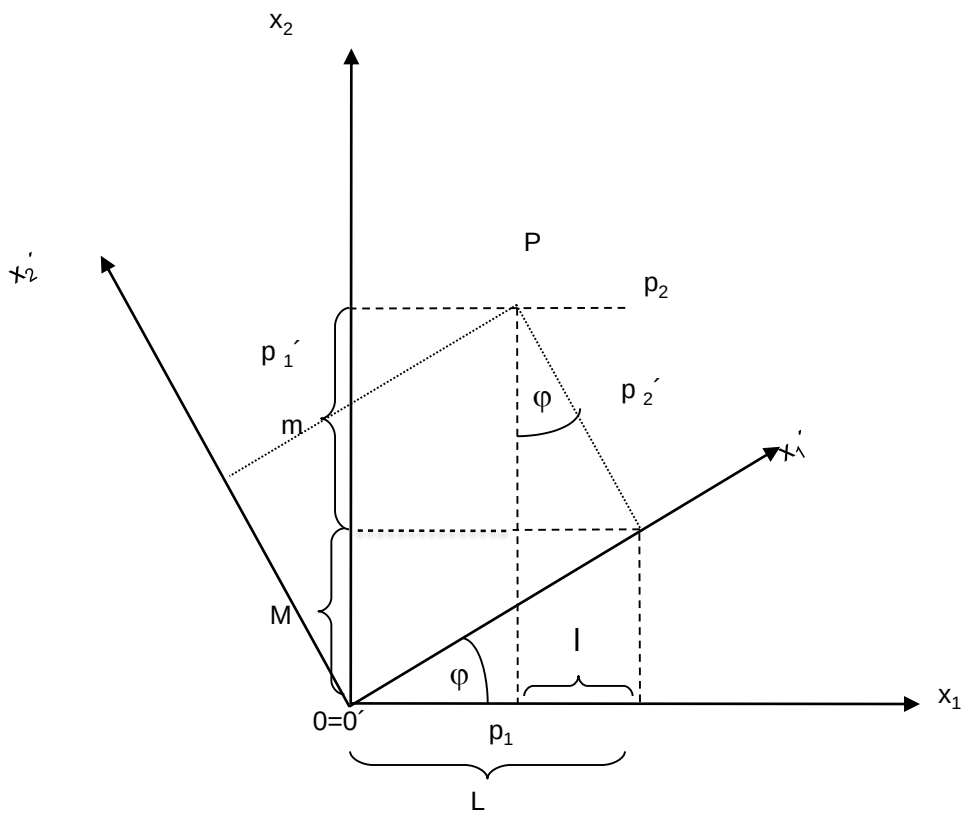


5.2 $\mathbb{R}^2 \rightarrow \mathbb{R}^2$ Koordinatentransformationen

Drehung (rotation)

$$\frac{L}{p_1'} = \cos(\phi) \quad \frac{M}{p_1'} = \sin(\phi)$$
$$\frac{l}{p_2'} = \sin(\phi) \quad \frac{m}{p_2'} = \cos(\phi)$$

$$p_1 = L - l = p_1'\cos(\phi) - p_2'\sin(\phi)$$
$$p_2 = M + m = p_1'\sin(\phi) + p_2'\cos(\phi)$$



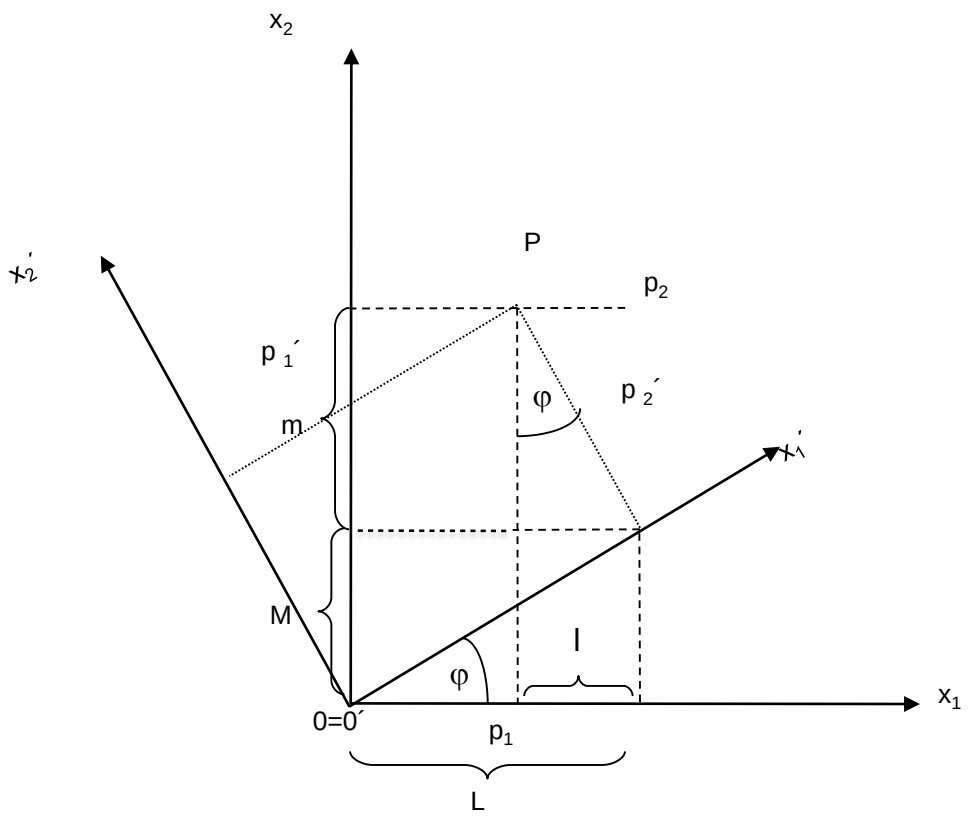
5.2 $\mathbb{R}^2 \rightarrow \mathbb{R}^2$ Koordinatentransformationen

Drehung (rotation)

$$\frac{L}{p_1'} = \cos(\phi) \quad \frac{M}{p_1'} = \sin(\phi)$$
$$\frac{l}{p_2'} = \sin(\phi) \quad \frac{m}{p_2'} = \cos(\phi)$$

$$p_1 = L - l = p_1'\cos(\phi) - p_2'\sin(\phi)$$
$$p_2 = M + m = p_1'\sin(\phi) + p_2'\cos(\phi)$$

$$\begin{pmatrix} p_1 \\ p_2 \end{pmatrix} = \begin{pmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{pmatrix} \cdot \begin{pmatrix} p_1' \\ p_2' \end{pmatrix}$$



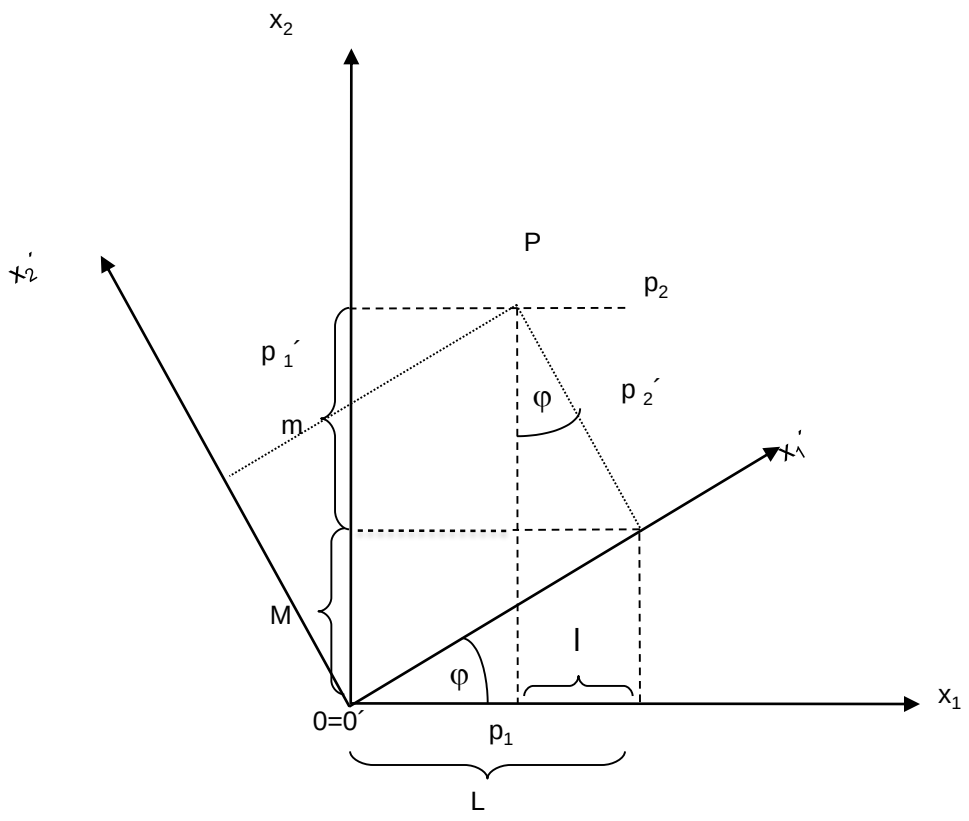
5.2 $\mathbb{R}^2 \rightarrow \mathbb{R}^2$ Koordinatentransformationen

Drehung (rotation)

$$\begin{pmatrix} p_1 \\ p_2 \end{pmatrix} = \begin{pmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{pmatrix} \cdot \begin{pmatrix} p_1' \\ p_2' \end{pmatrix}$$

$$P = R \cdot P'$$

– Bemerkung
  R ist orthonormal: $R^{-1} = R^T$



5.2 $\mathbb{R}^2 \to \mathbb{R}^2$ Koordinatentransformationen

Drehung (rotation): Interpretation

1) Punkt wird gedreht

R transformiert

- die Koordinatendarstellung (p_1', p_2') bezüglich des Systems S
- in die Koordinatendarstellung (p_1, p_2) bezüglich des selben Systems S

dies entspricht:

- einem globalen Koordinatensystem S
- auf die Koordinaten (p_1', p_2') von P wirkt die Matrix R

2) Koordinatensystem wird gedreht

R transformiert

- das lokale Koordinatensystem S
- in das lokale Koordinatensystem S'

dies entspricht:

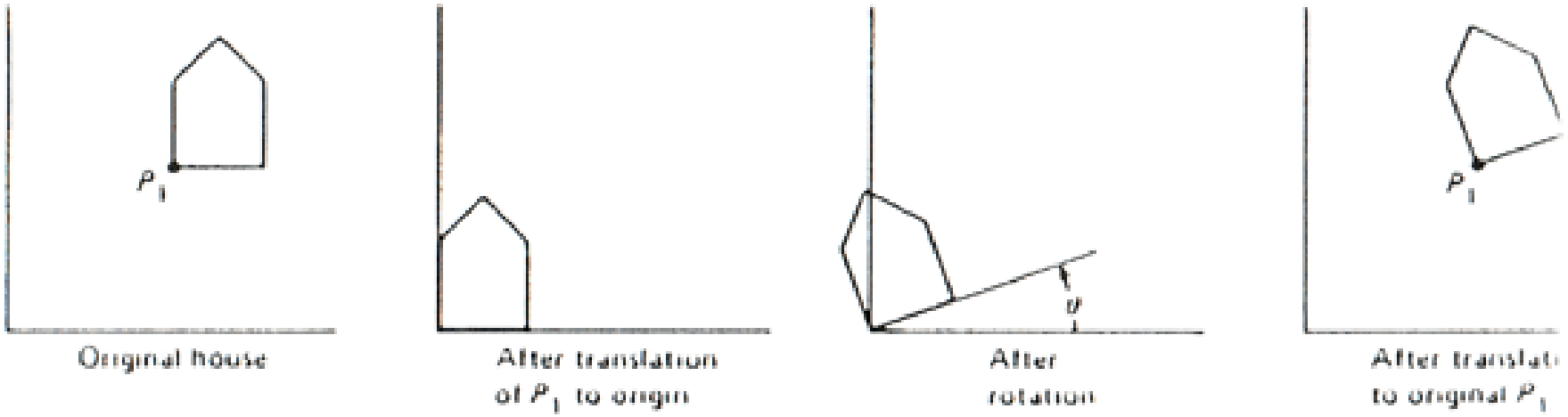
- P wird bezüglich des Koordinatensystems S' mit den Koordinaten (p_1', p_2') definiert

5.2 $\mathbb{R}^2 \rightarrow \mathbb{R}^2$ Koordinatentransformationen

Drehung (rotation)

- Bei der Rotation um einen beliebigen Punkt P_1 müssen noch zwei Translationen hinzugenommen werden

1) Verschiebung von P_1 in den Ursprung
2) Drehung um den Ursprung
3) Verschiebung von P_1 in die ursprüngliche Position



5.2 $\mathbb{R}^2 \rightarrow \mathbb{R}^2$ Koordinatentransformationen

Skalierung (scaling)

- Soll das System S' „vergrößert“ oder „verkleinert“ werden, so muss eine Skalierung durchgeführt werden:
  - $p_1 = s_1 \cdot p_1'$
  - $p_2 = s_2 \cdot p_2'$

$$\begin{pmatrix} p_1 \\ p_2 \end{pmatrix} = \begin{pmatrix} s_1 & 0 \\ 0 & s_2 \end{pmatrix} \cdot \begin{pmatrix} p_1' \\ p_2' \end{pmatrix}$$

$$P = S \cdot \mathrm{P}'$$

Scherung

- $p_1 = p_1' + \lambda_1 \cdot p_2'$
- $p_2 = p_2'$

$$\begin{pmatrix} p_1 \\ p_2 \end{pmatrix} = \begin{pmatrix} 1 & \lambda_1 \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} p_1' \\ p_2' \end{pmatrix}$$

$$P = \Lambda \cdot \mathrm{P}'$$

5.2 $\mathbb{R}^2 \rightarrow \mathbb{R}^2$ Koordinatentransformationen

- Beliebige lineare Transformationen können beschrieben werden als Kombination aus
  - einer Skalierung
  - einer Scherung und
  - einer Rotation

5.2 $\mathbb{R}^2 \rightarrow \mathbb{R}^2$ Koordinatentransformationen

Affine Transformationen

- Lassen sich als Kombination einer linearen Abbildung und einer Translation schreiben:
  $P = A \cdot P' + T$
- Die bisher genannten Transformationen sind Beispiele affiner Transformationen:
  - Translation
  - Rotation
  - Skalierung
  - Scherung

Affine Invarianz von Teilungsverhältnissen:

- Für eine affine Transformation F und die Punkte P und Q gilt immer:

$$F(\lambda \cdot P + (1 - \lambda) \cdot Q)$$
$$= \lambda \cdot F(P) + (1 - \lambda) \cdot F(Q)$$

$$0 \leq \lambda \leq 1$$

5.2 $\mathbb{R}^2 \to \mathbb{R}^2$ Koordinatentransformationen

Affine Transformationen

- Diese Beziehung zeigt
  - dass das Bild einer Strecke (Strecke von Q nach P) unter einer affinen Abbildung F wieder eine Strecke ist
  - dass Teilungsverhältnisse $\lambda : (1 - \lambda)$ unter F invariant bleiben
- Es genügt, die Endpunkte Q und P auf der Strecke abzubilden.
- Zwischenpunkte erhält man durch Interpolation von $F(Q)$ und $F(P)$.
- Man beachte, dass unter affinen Abbildungen parallele Linien parallel bleiben.

5.2 $\mathbb{R}^2 \to \mathbb{R}^2$ Koordinatentransformationen

Affine Transformationen

- Reflektion an der Gerade $y = x$:

$$F = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

- Reflektion an der Gerade $y = -x$:

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

- Reflektion an der x-Achse:

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

- Reflektion an der y-Achse:

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

- Reflektion am Ursprung:

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

5.2 $\mathbb{R}^2 \rightarrow \mathbb{R}^2$ Koordinatentransformationen

Zusammengesetzte Transformationen

- Bemerkung:
  - Die Matrizenmultiplikation ist nicht kommutativ.
  - Bei hintereinander geschalteten Matrizen-multiplikationen muss darauf geachtet werden, dass die Reihenfolge der Matrizen der Reihenfolge der Rotationen entspricht.

P’ = M_n * (M_n-1 * … * (M_3 * (M_2 * (M_1 * P)))…)

5.2 $\mathbb{R}^2 \rightarrow \mathbb{R}^2$ Koordinatentransformationen

Homogene Koordinaten

- Homogene Koordinaten entstammen der projektiven Geometrie.
- An dieser Stelle soll jedoch eine andere Motivation verwendet werden.
- Die Hintereinanderschaltung von Rotation, Translation und Skalierung führt auf die Abbildungsgleichung:

$$P = S \cdot (T + R \cdot P')$$

- Müssen mehrere solcher Transformationen hintereinander ausgeführt werden, so stört die Addition in der Gleichung.
- Da heutige Computergraphikhardware insbesondere auch Matrixmultiplikationen unterstützt, ist es günstig, Transformationen ausschließlich mittels Matrixmultiplikationen auszuführen, also:

$$P = M_n \cdot \dots \cdot M_1 \cdot P'$$

5.2 $\mathbb{R}^2 \to \mathbb{R}^2$ Koordinatentransformationen

Homogene Koordinaten

- Dies erreicht man durch folgenden Übergang auf die nächst höhere Dimension:
  - Das Tripel (x, y, w), $w \neq 0$ stellt die homogenen Koordinaten des Punktes $\left(\frac{x}{w}, \frac{y}{w}\right) \in \mathbb{R}^2$ dar.
  - Da es unendlich viele solcher Darstellungen desselben Punktes gibt, verwendet man die so genannte Standarddarstellung mit $w = 1$.
  - Also besitzt ein Punkt $P = (x, y) \in \mathbb{R}^2$ als homogene Koordinaten $(x, y, 1)$

Bemerkung:

- Für Punkte im $\mathbb{R}^3$ gilt eine analoge Konstruktion

5.2 $\mathbb{R}^2 \rightarrow \mathbb{R}^2$ Koordinatentransformationen

Homogene Koordinaten

- Verschiebung

$$\begin{pmatrix} 1 & 0 & t_1 \\ 0 & 1 & t_2 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

- Skalierung

$$\begin{pmatrix} s_1 & 0 & 0 \\ 0 & s_2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

- Drehung um den Ursprung

$$\begin{pmatrix} \cos\varphi & -\sin\varphi & 0 \\ \sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

5.2 $\mathbb{R}^2 \rightarrow \mathbb{R}^2$ Koordinatentransformationen

Homogene Koordinaten

– Drehung um Z

$$\begin{pmatrix} 1 & 0 & Z_x \\ 0 & 1 & Z_y \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \cos\varphi & -\sin\varphi & 0 \\ \sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & -Z_x \\ 0 & 1 & -Z_y \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

5.3 $\mathbb{R}^3 \rightarrow \mathbb{R}^3$ Koordinatentransformationen

– Verschiebung

$$\begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

– Skalierung

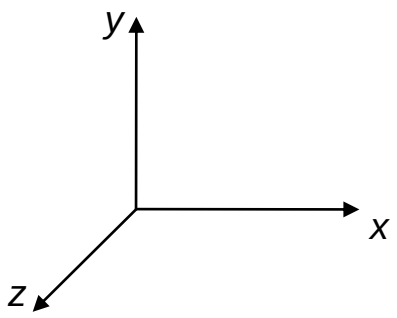
$$\begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

5.3 $\mathbb{R}^3 \rightarrow \mathbb{R}^3$ Koordinatentransformationen

Drehungen

- Im 3-dimensionalen Raum gibt es mehrere Achsen, um die gedreht werden kann
  - x-Achse
  - y-Achse
  - z-Achse
  - Beliebige Achse im Raum
- Für die ersten drei Fälle wird die Richtung der Achse als von einem negativen Wert zum Ursprung angenommen.

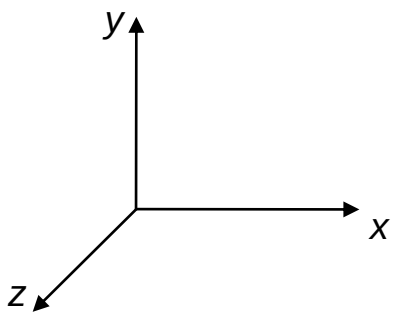
- Rechtshändiges Koordinatensystem



5.3 $\mathbb{R}^3 \rightarrow \mathbb{R}^3$ Koordinatentransformationen

Drehungen

- Es wird gegen den Uhrzeigersinn gedreht (mathematisch positiv).
- Rechtshändiges Koordinatensystem

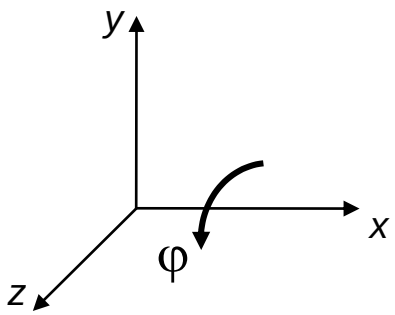


5.3 $\mathbb{R}^3 \rightarrow \mathbb{R}^3$ Koordinatentransformationen

Drehungen

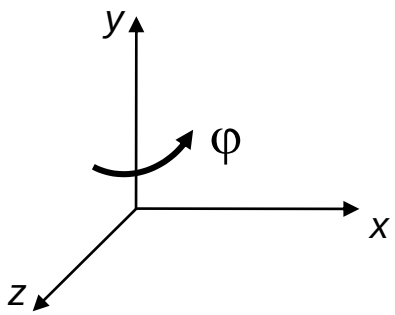
– x-Achse

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\varphi & -\sin\varphi & 0 \\ 0 & \sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$



– y-Achse

$$\begin{pmatrix} \cos\varphi & 0 & \sin\varphi & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\varphi & 0 & \cos\varphi & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

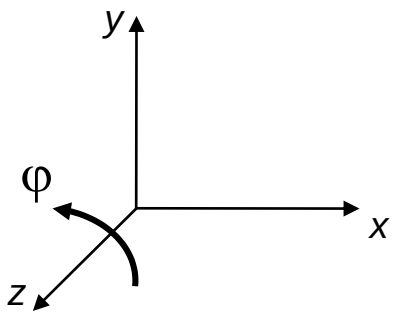


5.3 $\mathbb{R}^3 \to \mathbb{R}^3$ Koordinatentransformationen

Drehungen

– z-Achse

$$\begin{pmatrix} \cos\varphi & -\sin\varphi & 0 & 0 \\ \sin\varphi & \cos\varphi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$



5.3 $\mathbb{R}^3 \rightarrow \mathbb{R}^3$ Koordinatentransformationen

Drehung um eine beliebige Achse

- Jede Rotation um eine beliebige Achse kann aus Rotationen um die einzelnen Koordinatenachsen zusammengesetzt werden ($\Rightarrow$ Euler)

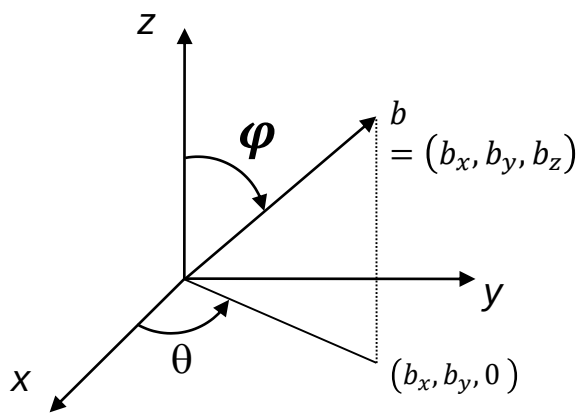
- Wir entwickeln
  - die Rotation $R_G(\alpha)$
  - für die Drehung eines Punktes P
  - um eine beliebig orientierte Achse G im Raum
  - um einen Winkel α

5.3 $\mathbb{R}^3 \to \mathbb{R}^3$ Koordinatentransformationen

Drehung um eine beliebige Achse

- Sonderfall:
  die Drehachse G
  - geht durch den Ursprung
  - wird von dem Vektor
    $$b = (b_x, b_y, b_z), \|b\| = 1$$
    generiert
  - G: $\lambda \cdot b, \lambda \in \mathbb{R}$

$$\begin{aligned} b_x &= \sin\varphi \cdot \cos\theta \\ b_y &= \sin\varphi \cdot \sin\theta \\ b_z &= \cos\varphi \end{aligned}$$



5.3 $\mathbb{R}^3 \rightarrow \mathbb{R}^3$ Koordinatentransformationen

Drehung um eine beliebige Achse durch den Ursprung

- Gesucht sind nun die Koordinaten eines Punktes P
  nach einer Drehung um die Achse G
  um den Winkel α

- Vorgehensweise:
  1) Der Punkt P wird so transformiert, dass die Drehachse mit der z-Achse zusammenfällt
  2) Die Drehung um α verwendet die Rotationsmatrix $R_z(\alpha)$
  3) Die Transformation wird rückgängig gemacht

- Bemerkung:
  Ist G mit der z-Achse identisch, so entfallen die Schritte 1) und 3) (Transformationen)

- Man geht in mehreren Teilschritten vor

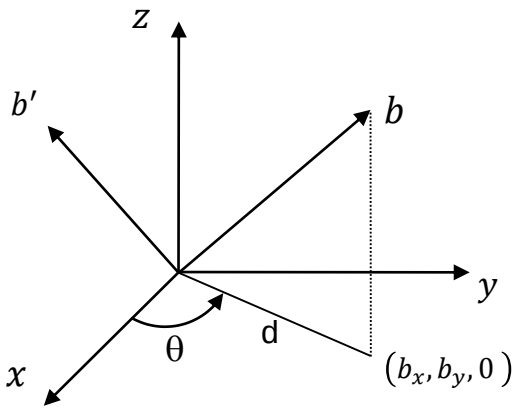
5.3 $\mathbb{R}^3 \to \mathbb{R}^3$ Koordinatentransformationen

Drehung um eine beliebige Achse durch den Ursprung

Schritt 1:

$$d^2 = b_x^2 + b_y^2$$

- Der Vektor b wird in die (z, x)-Ebene gedreht (b')
- Aus P entsteht dabei $P' = R_z(-\theta) \cdot P$



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

$$= \frac{1}{d} \cdot \begin{pmatrix} b_x & b_y & 0 & 0 \\ -b_y & b_x & 0 & 0 \\ 0 & 0 & d & 0 \\ 0 & 0 & 0 & d \end{pmatrix}$$

5.3 $\mathbb{R}^3 \to \mathbb{R}^3$ Koordinatentransformationen

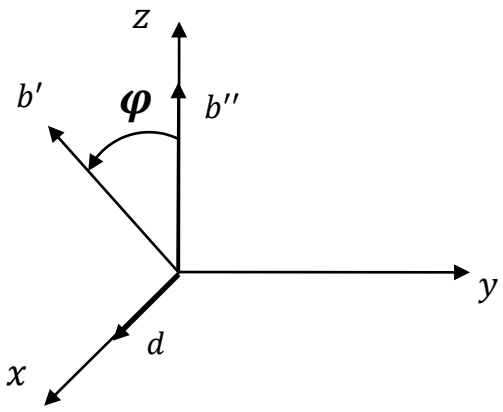
Drehung um eine beliebige Achse durch den Ursprung

Schritt 2:

- Der Vektor b' wird so gedreht, dass er mit der z-Achse zusammenfällt
- Aus P' entsteht dabei $P'' = R_y(-\varphi) \cdot P'$

$$R_y(-\varphi) = \begin{pmatrix} \cos\varphi & 0 & -\sin\varphi & 0 \\ 0 & 1 & 0 & 0 \\ \sin\varphi & 0 & \cos\varphi & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} b_z & 0 & -d & 0 \\ 0 & 1 & 0 & 0 \\ d & 0 & b_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



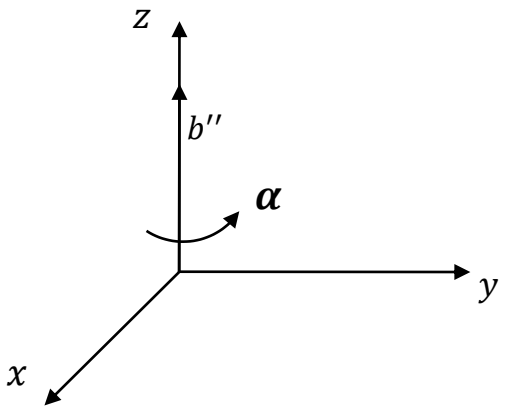
5.3 $\mathbb{R}^3 \rightarrow \mathbb{R}^3$ Koordinatentransformationen

Drehung um eine beliebige Achse durch den Ursprung

Schritt 3:

- P'' wird mit Winkel α um die Z-Achse gedreht
- Aus P'' entsteht dabei $P''' = R_z(\alpha) \cdot P''$

$$R_z(\alpha) = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 & 0 \\ \sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



5.3 $\mathbb{R}^3 \rightarrow \mathbb{R}^3$ Koordinatentransformationen

Drehung um eine beliebige Achse durch den Ursprung

Schritt 4:

– Inverse Rotation zu Schritt 2

$$R_y(\varphi) = \begin{pmatrix} \cos\varphi & 0 & \sin\varphi & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\varphi & 0 & \cos\varphi & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} b_z & 0 & d & 0 \\ 0 & 1 & 0 & 0 \\ -d & 0 & b_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Schritt 5:

– Inverse Rotation zu Schritt 1

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

$$= \frac{1}{d} \cdot \begin{pmatrix} b_x & -b_y & 0 & 0 \\ b_y & b_x & 0 & 0 \\ 0 & 0 & d & 0 \\ 0 & 0 & 0 & d \end{pmatrix}$$

5.3 $\mathbb{R}^3 \to \mathbb{R}^3$ Koordinatentransformationen

Drehung um eine beliebige Achse durch den Ursprung

Ergebnis:

- Gesamttransformation:

$$R_b(\alpha) = R_z(\theta) \circ R_y(\varphi) \circ R_z(\alpha) \circ R_y(-\varphi) \circ R_z(-\theta)$$

Allgemeiner Fall:

- Die Drehachse ist eine allgemeine Gerade
  - $G: a + \lambda \cdot b, \lambda \in \mathbb{R}$
  - $a = (a_x, a_y, a_z)$
  - $b = (b_x, b_y, b_z), \|b\| = 1$

$$R_G(\alpha) = T(a) \circ R_z(\theta) \circ R_y(\varphi) \circ R_z(\alpha) \circ R_y(-\varphi) \circ R_z(-\theta) \circ T(-a)$$

Zusammengefasste Transformationsmatrizen

- Durch die Verschiebung vieler Objekte mit einer Gesamtmatrix spart man Rechenkosten.
- Diese entspricht einer sequenziellen Multiplikation des Punktes P mit den einzelnen Transformationsmatrizen.
- Ausnutzung der Assoziativität der Matrizenmultiplikation
  - (M1 * M2) * M3 = M1 * (M2 * M3)

Statt

P’ = M_n * (M_n-1 * … * (M_3 * (M_2 * (M_1 * P)))…)

Schreibt man

P’ = (M_n * M_n-1 * … * M_3 * M_2 * M_1) * P

UNIVERSITÄT LEIPZIG

Projektionen

COMPUTERGRAPHIK

Inhaltsverzeichnis

6 Projektionen

6.1 Einleitung

6.2 Perspektivische Projektionen

6.3 Parallele Projektionen

6.4 Perspektivische Projektion -- Berechnung

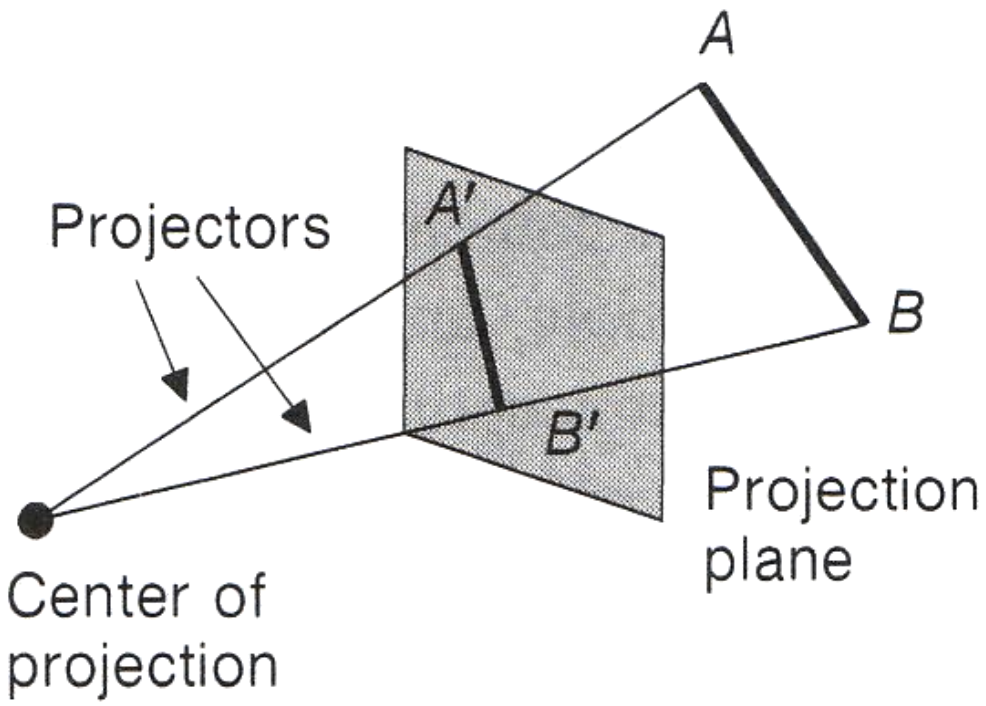
6.5 Unmögliche Strukturen

6.1 Einleitung

- Eine Projektion ist eine Abbildung
  - aus einem Raum der Dimension n
  - in einen Raum der Dimension $m < n$
- Objekte werden im $n = 3$ dimensionalen Raum dargestellt
- Bildschirm ist $m = 2$ dimensionaler Raum.

6.1 Einleitung

- Ein Raumpunkt wird entlang eines Projektionsstrahls auf eine vorgegebene Projektionsebene abgebildet.
  - Projektionsstrahl:
    - Projektionszentrum
    - Raumpunkt
  - Projizierter Raumpunkt: Schnittpunkt des Projektionsstrahls mit der Projektionsebene

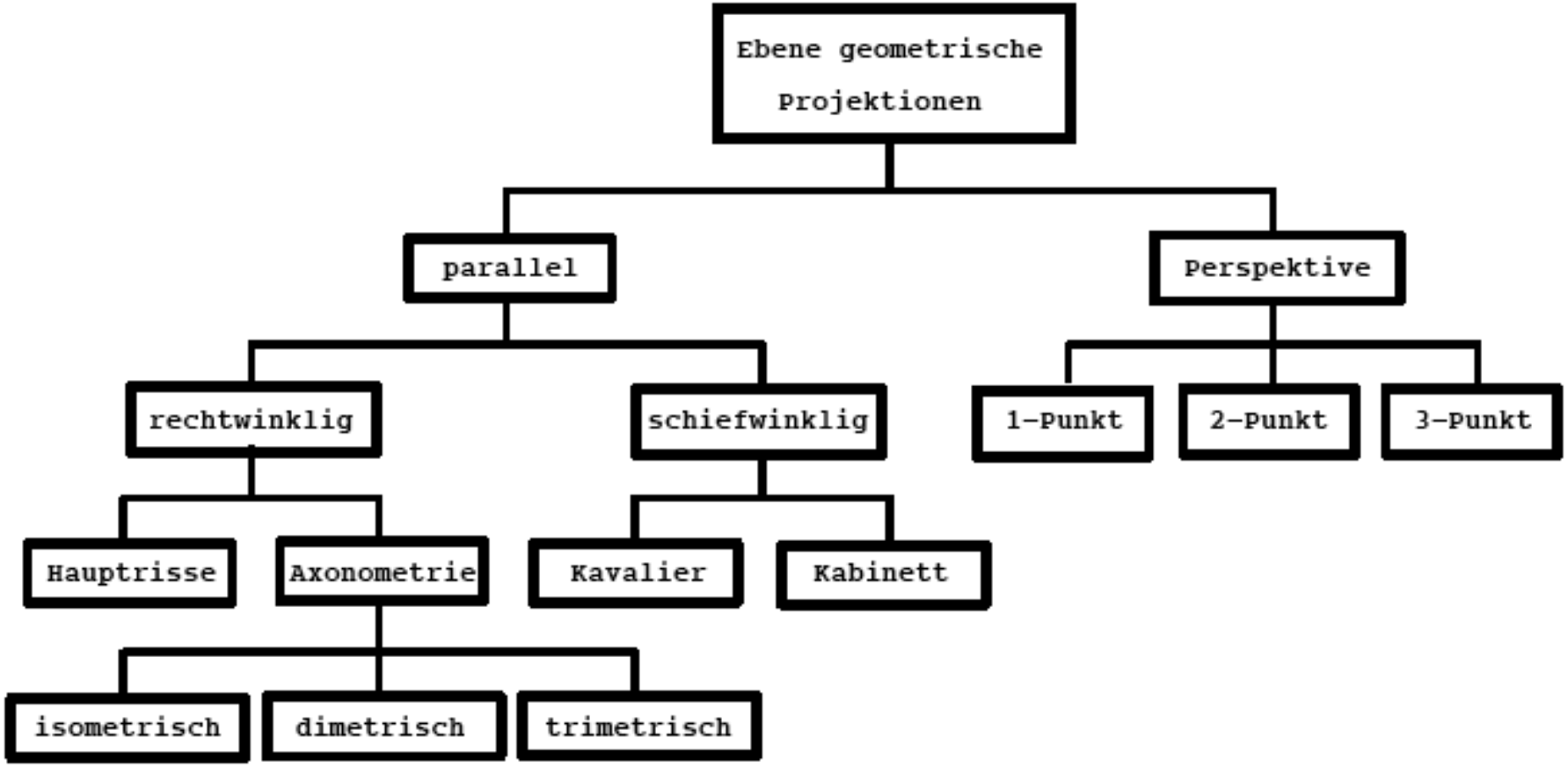


6.1 Einleitung

- Geometrisch planare Projektionen:
  - Perspektivische Projektion (Zentralprojektion)
  - Parallelprojektion
    - Projektionszentrum liegt in einem unendlich fernen Punkt
- Im Rahmen der projektiven Geometrie stellt die Parallelprojektion einen Spezialfall der Zentralprojektion dar.
- Dies lässt sich bei der praktischen Umsetzung der Projektionen als Matrizen gewinnbringend anwenden.

6.1 Einleitung

Klassifikation der gängigen Projektionsarten

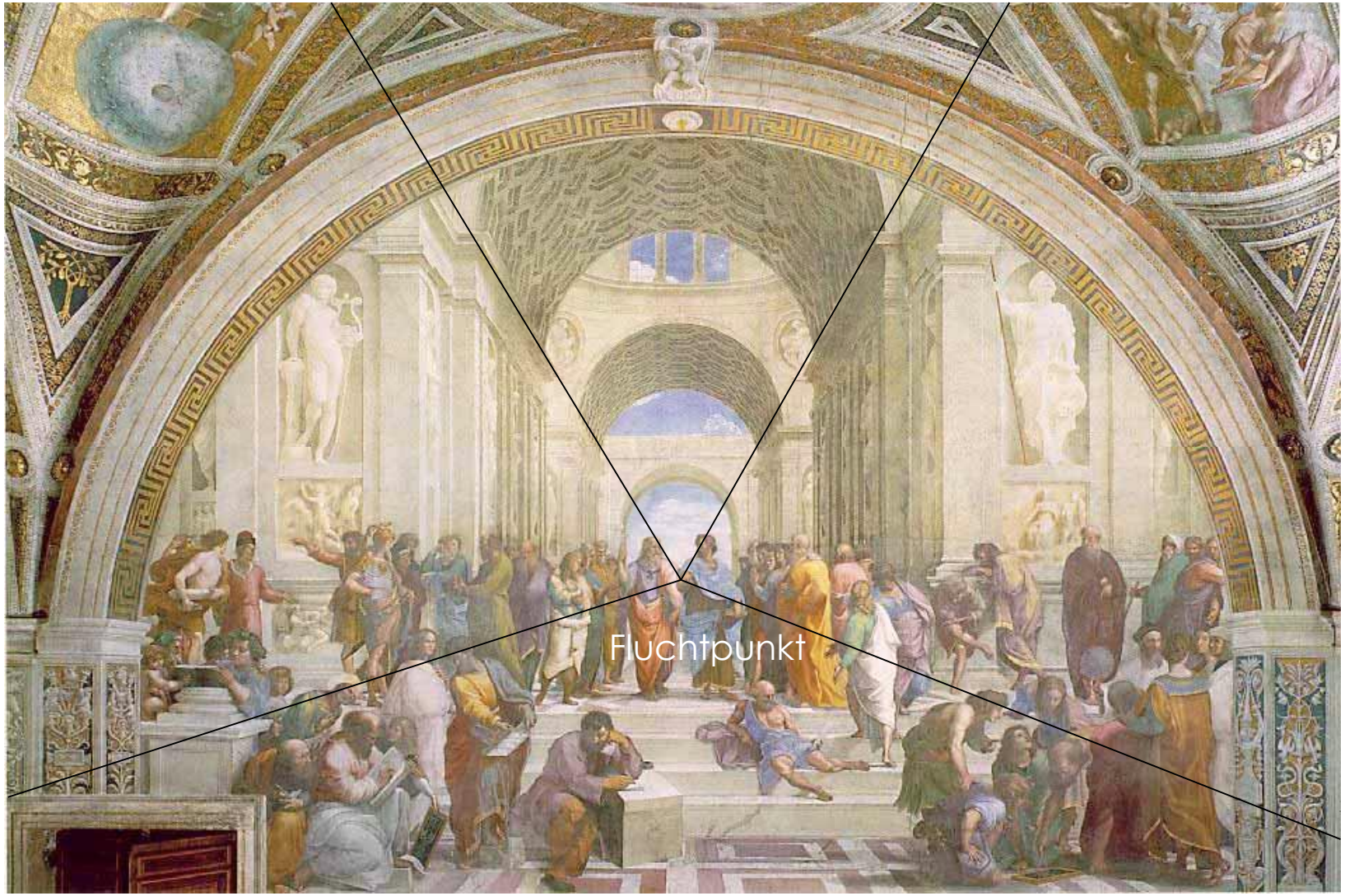


6.2 Perspektivische Projektion

- Alle Projektionsstrahlen laufen durch das Projektionszentrum.
- Projektionszentrum fällt mit dem Auge des Beobachters zusammen.
- Das Verfahren erzeugt eine optische Tiefenwirkung.
- Geht in seinen Anfängen bis in die Malerei der Antike zurück.

6.2 Perspektivische Projektion

– Raffael
  Schule von Athen



6.2 Perspektivische Projektion

Eigenschaften

- Je zwei parallele Geraden, die nicht parallel zur Projektionsebene sind, treffen sich in einem Punkt, dem Fluchtpunkt.
- Es gibt unendlich viele Fluchtpunkte, je einen pro Richtung nicht parallel zur Projektionsebene.
- Hervorgehoben werden die Fluchtpunkte der Hauptachsen
  - Geraden, die parallel zur x-Achse verlaufen, treffen sich im x-Fluchtpunkt.
  - Für die anderen Hauptachsen wird dies ähnlich definiert.

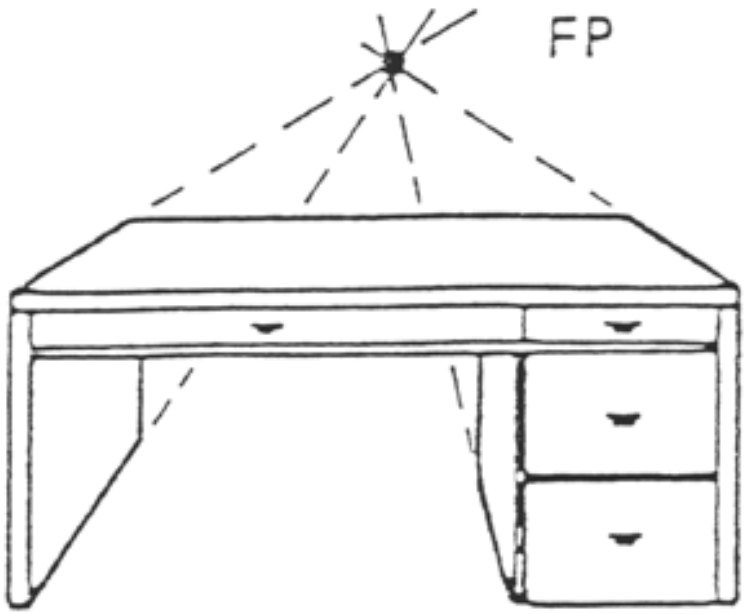
6.2 Perspektivische Projektion

Klassifikation

- Nach der Anzahl der Hauptachsen, die von der Projektionsebene geschnitten werden:
  - 1-Punkt-Perspektiven
  - 2-Punkt-Perspektiven
  - 3-Punkt-Perspektiven

6.2 Perspektivische Projektion

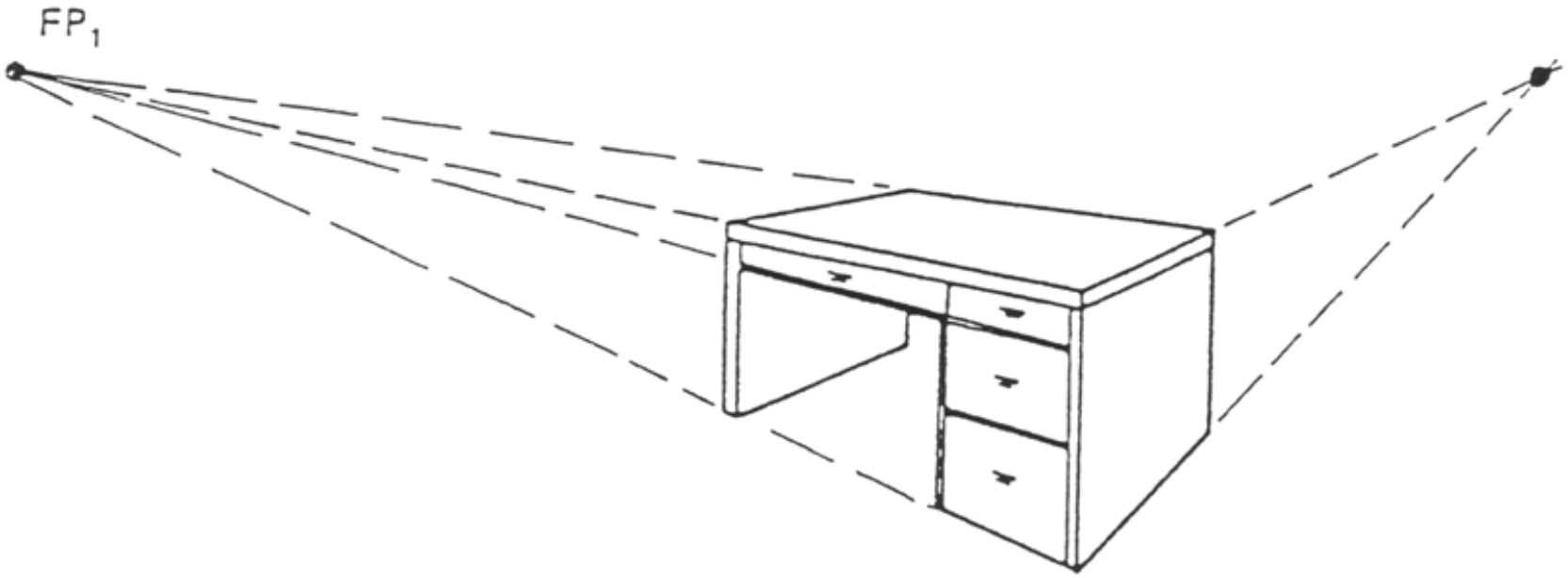
Beispiel



1-Punkt-Perspektive

6.2 Perspektivische Projektion

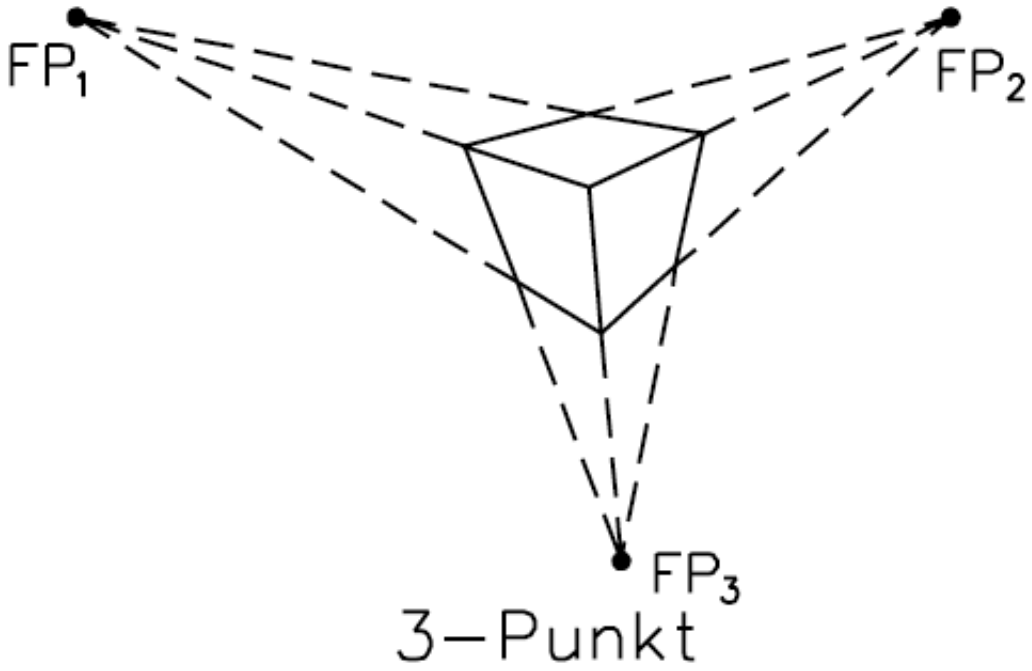
Beispiel



2-Punkt-Perspektive

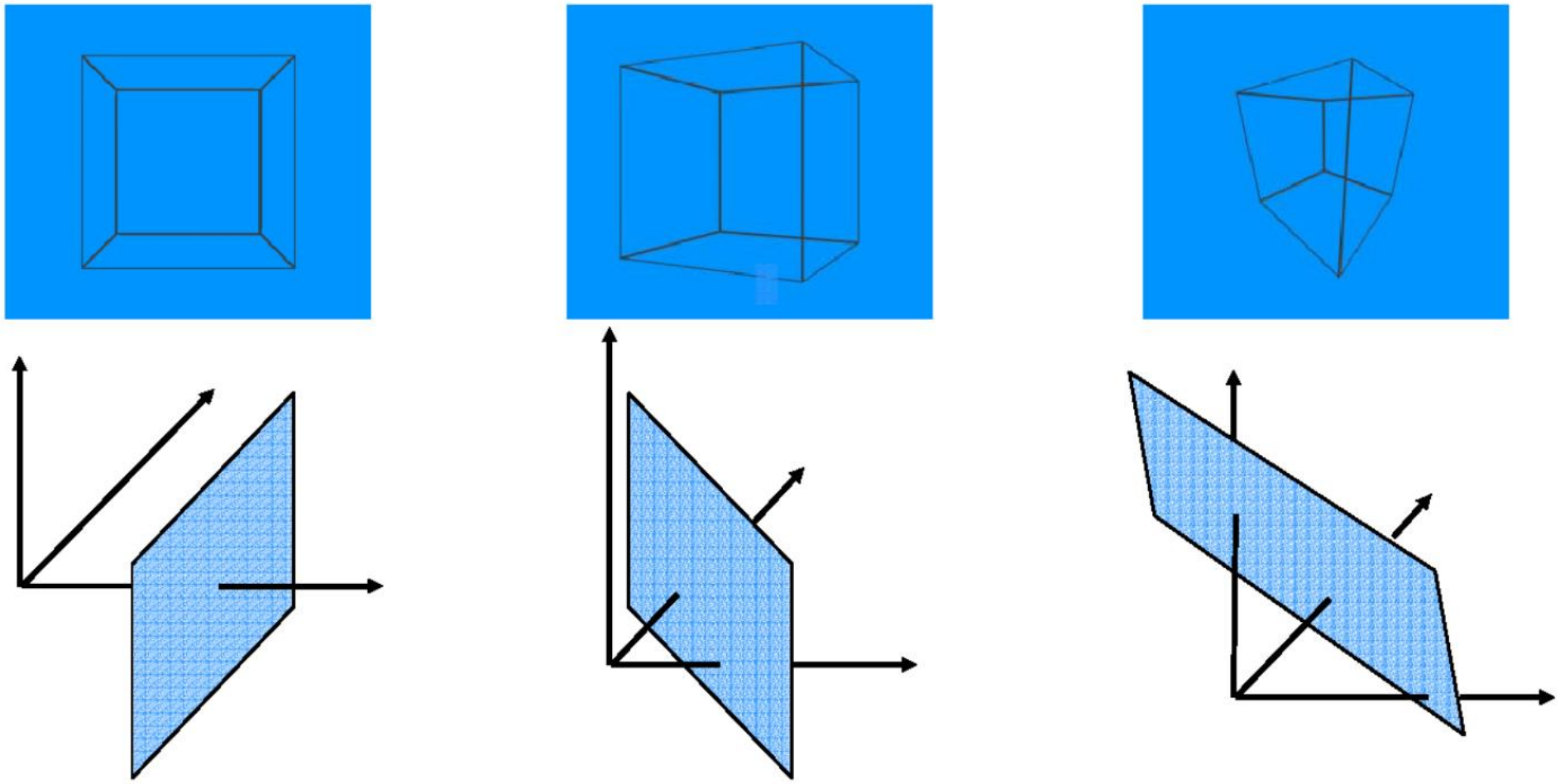
6.2 Perspektivische Projektion

Beispiel



6.2 Perspektivische Projektion

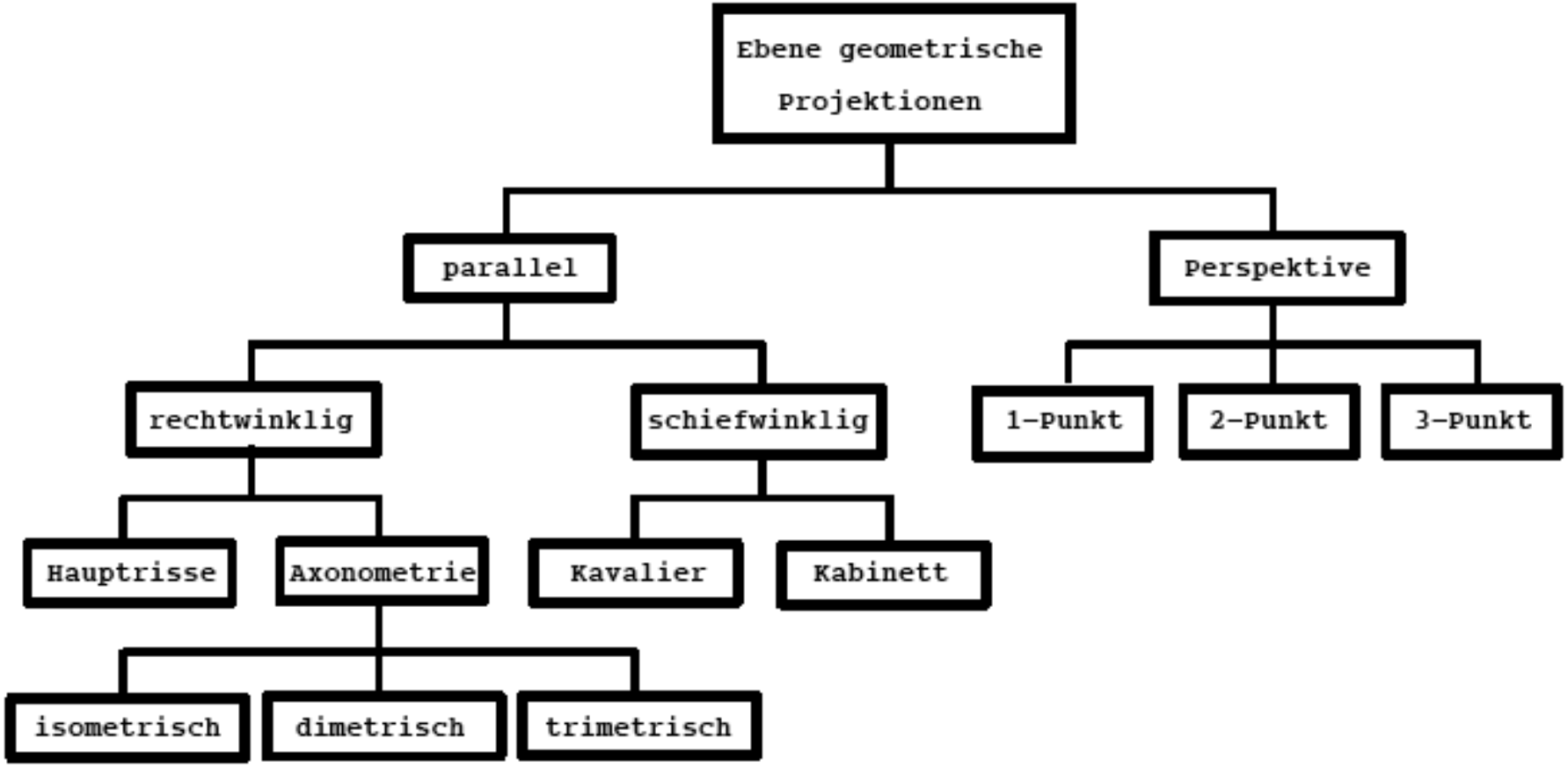
Beispiel



M. Haller, FH Hagenberg, 2002

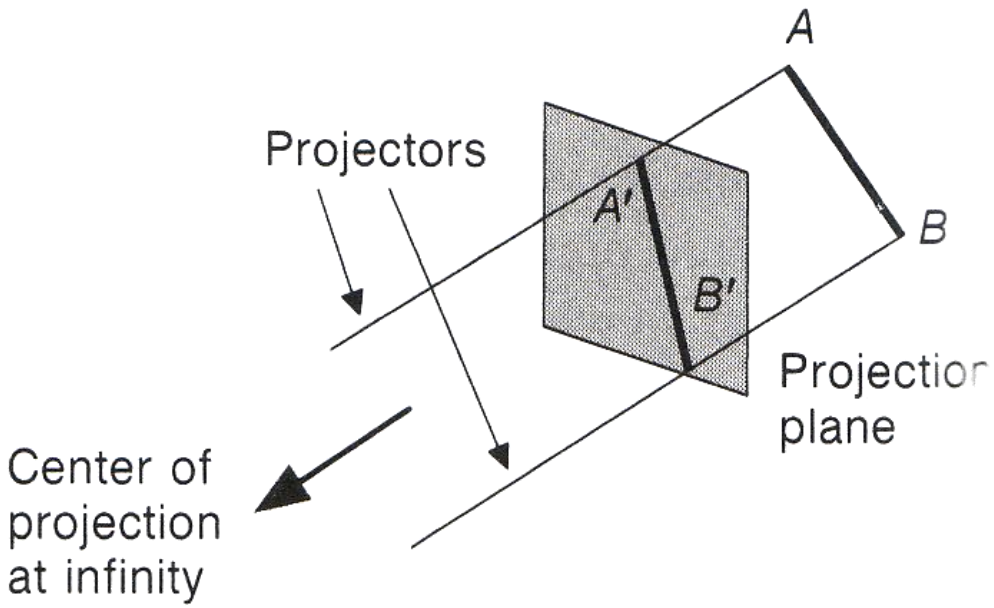
6.1 Einleitung

Klassifikation der gängigen Projektionsarten



6.3 Parallelprojektion

- Bei der Parallelprojektion ist das Projektionszentrum im Unendlichen.
- Alle Projektionsstrahlen verlaufen parallel in einer Richtung.
- Die Parallelprojektion ist
  - … weniger realistisch als die perspektivische Projektion.
  - … besser, um exakte Maße aus dem projizierten Bild zu bestimmen.



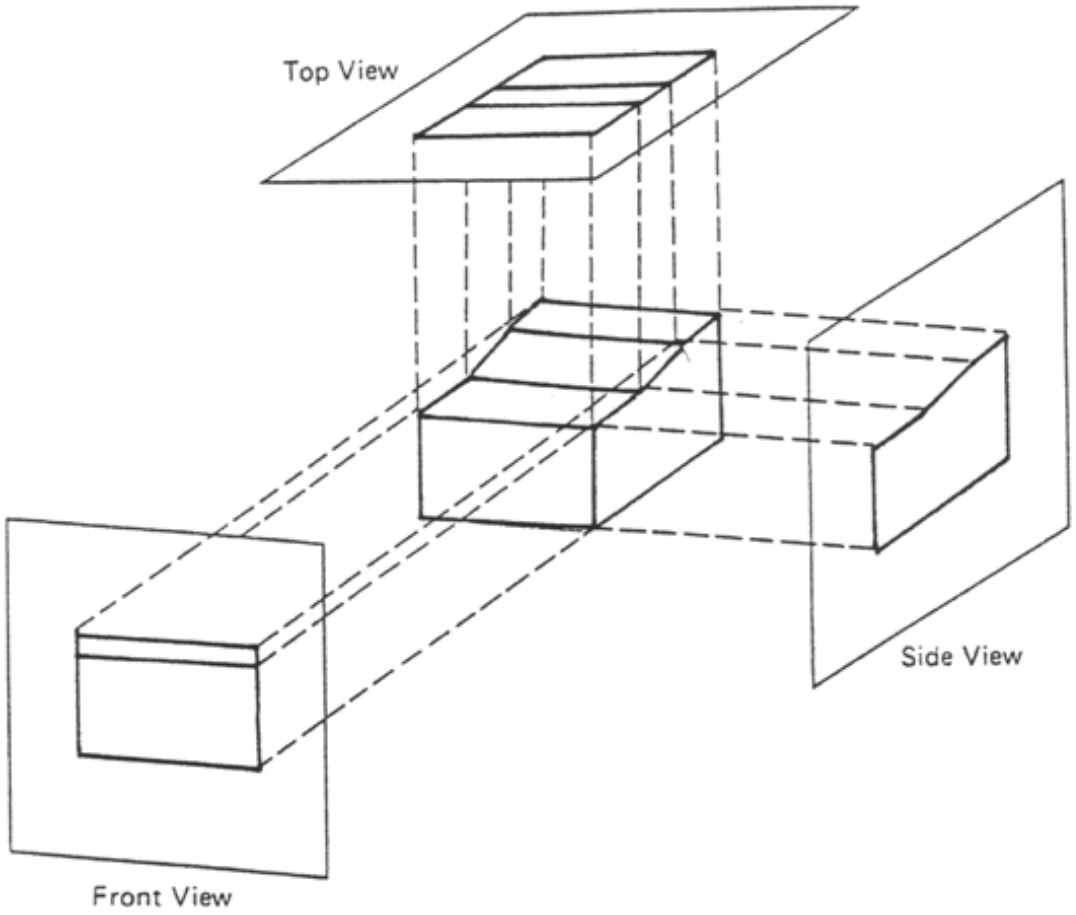
6.3 Parallelprojektion

- Orthographische Projektion:
  - Die Projektionsstrahlen stehen senkrecht gegen die Projektionsebene.
  - Projektionsrichtung fällt mit der Ebenennormalen zusammen.
- Schiefe Projektion:
  - Die Projektionsstrahlen stehen schief gegen die Projektionsebene.

6.3 Parallelprojektion

Orthographische Projektion: Hauptrisse

- Grundriss (Top View)
- Aufriss (Front View)
- Kreuzriss (Side View)
- Die Projektionsebene schneidet nur eine Hauptachse
- Die Normale der Projektionsebene ist parallel zu einer der Hauptachsen

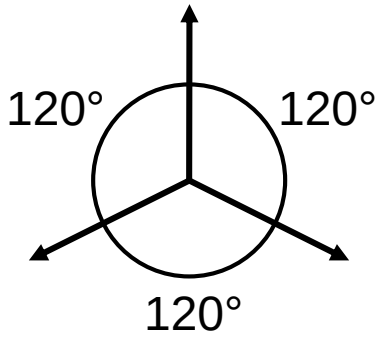


6.3 Parallelprojektion

Orthographische Projektion: Axonometrie

- Die Projektionsebene ist nicht orthogonal zu einer der Koordinatenachsen.
- Parallele Linien werden auf parallele Linien abgebildet.
- Winkel bleiben nicht erhalten.
- Abstände können längs der Hauptachsen gemessen werden (i.A. in jeweils einem anderen Maßstab).

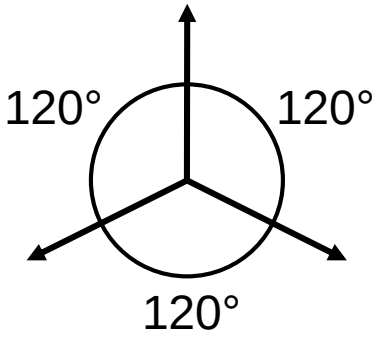
- Häufigstes Fall: isometrische Axonometrie
- Die Projektionsebene bildet mit allen Hauptachsen den gleichen Winkel.
  - Gleichmäßige Verkürzung aller Koordinatenachsen
  - Es gibt nur acht mögliche isometrische Projektionen.



6.3 Parallelprojektion

Beispiel Isometrische Darstellung

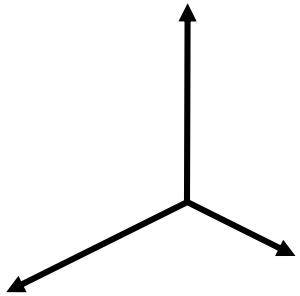
Age of Empires II

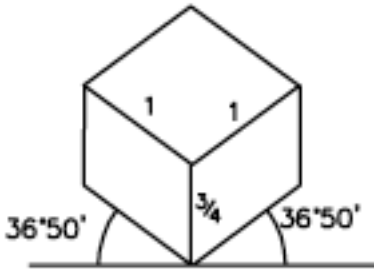


6.3 Parallelprojektion

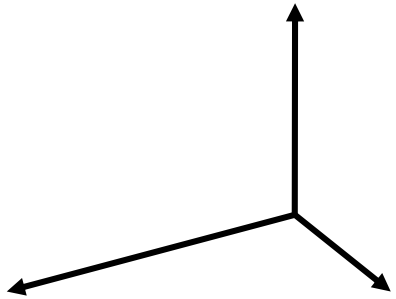
Orthographische Projektion

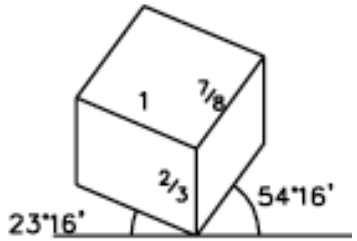
- Dimetrische Projektion
  - Projektionsebene hat mit zwei Hauptachsen den gleichen Winkel.
  - Skalierung ist in zwei Achsenrichtungen gleich.





- Trimetrische Projektion
  - Projektionsebene hat mit jeder Achse einen anderen Winkel.
  - Skalierungen sind in allen drei Achsenrichtungen verschieden.

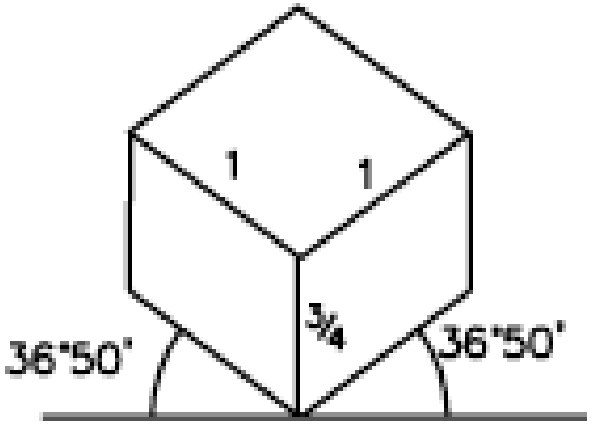




6.3 Parallelprojektion

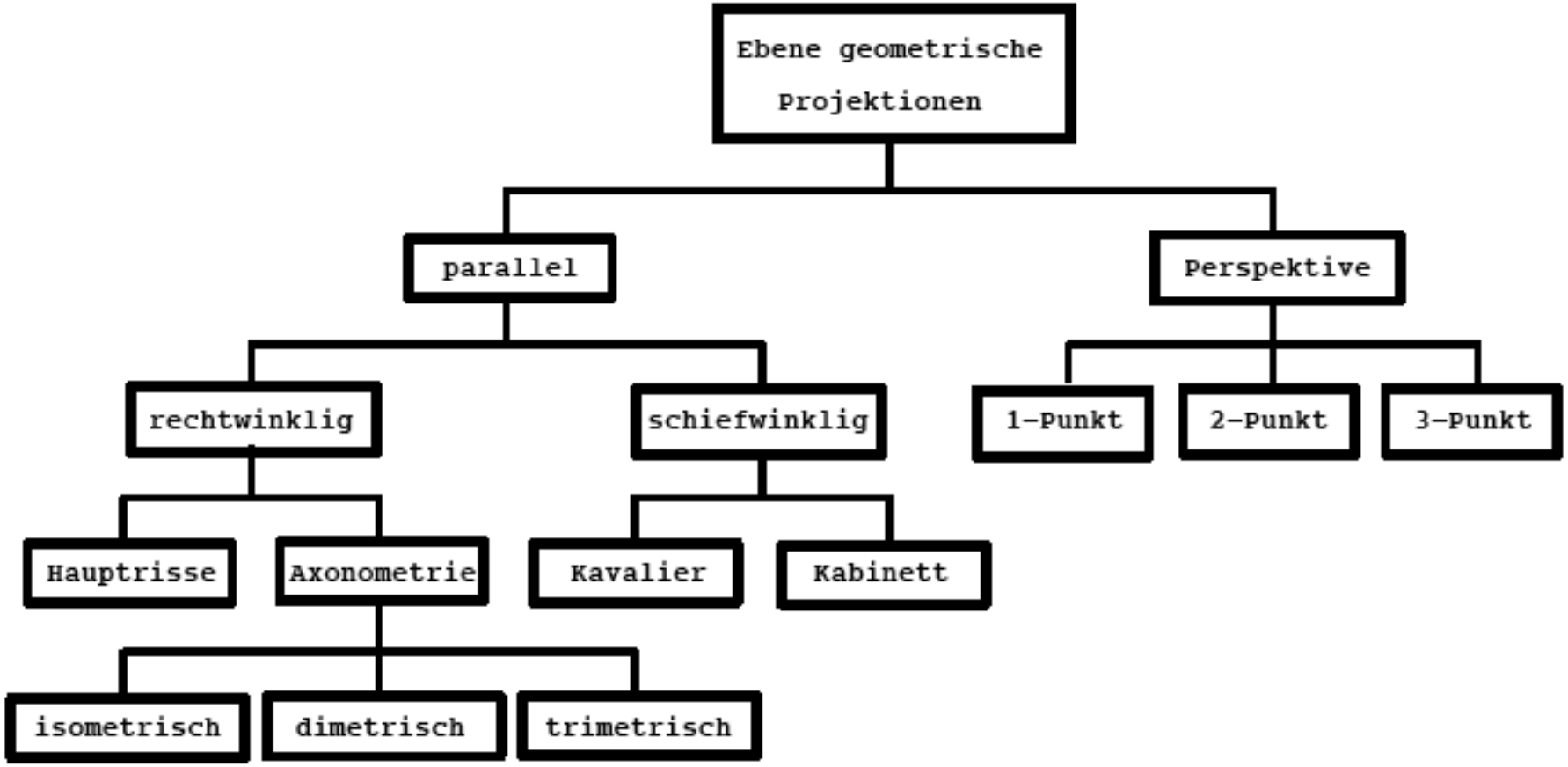
Beispiel Dimetrische Darstellung

Sim City 2000



6.1 Einleitung

Klassifikation der gängigen Projektionsarten



6.3 Parallelprojektion

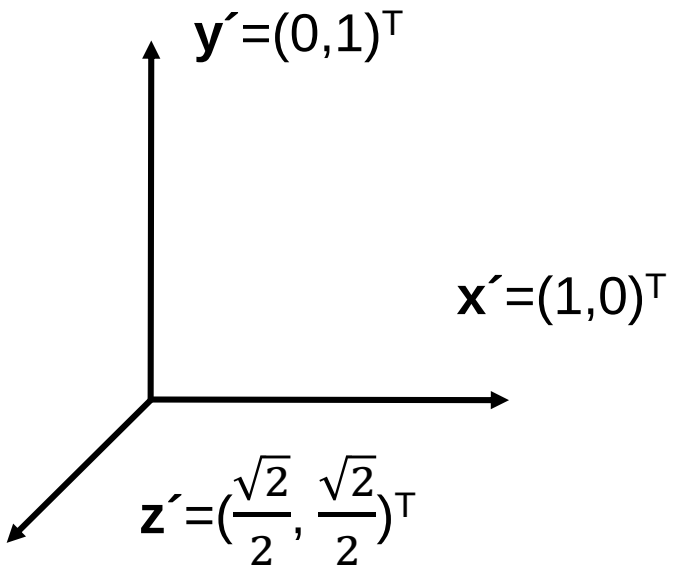
Schiefe Parallelprojektion

– Projektionsrichtung unterscheidet sich von der Normale der Projektionsebene.

6.3 Parallelprojektion

Schiefe Parallelprojektion: Kavalierprojektion

- Der Winkel zwischen Projektionsrichtung und Bildebene beträgt 45°.
- Die Länge der Projektion einer Linie, die senkrecht zur Bildebene steht, bleibt unverändert.
- Es gibt unendlich viele Kavalierprojektionen, eine für jede Richtung in der Bildebene.

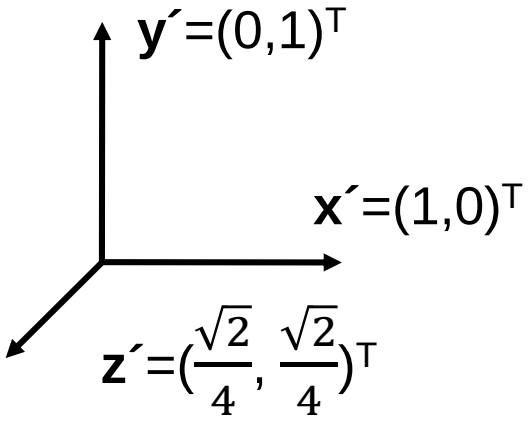


projizierte Einheitsvektoren

6.3 Parallelprojektion

Schiefe Parallelprojektion: Kabinettprojektion

– Länge der Projektion einer zur Projektionsebene senkrechten Linie soll die Hälfte ihrer Originallänge werden.



projizierte Einheitsvektoren

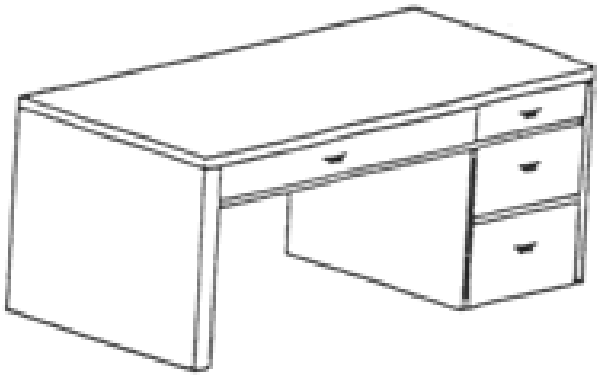
6.3 Parallelprojektion

Beispiel schiefwinklige Projektion

Sim City

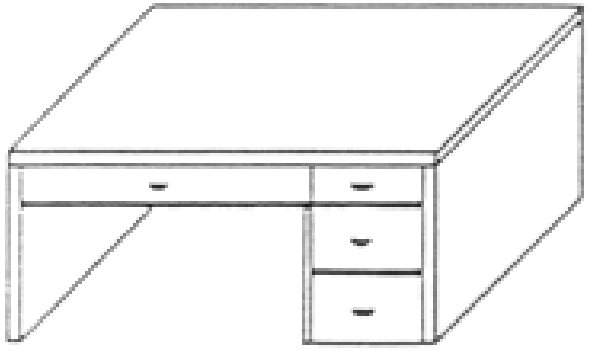
6.3 Parallelprojektion

Beispiele



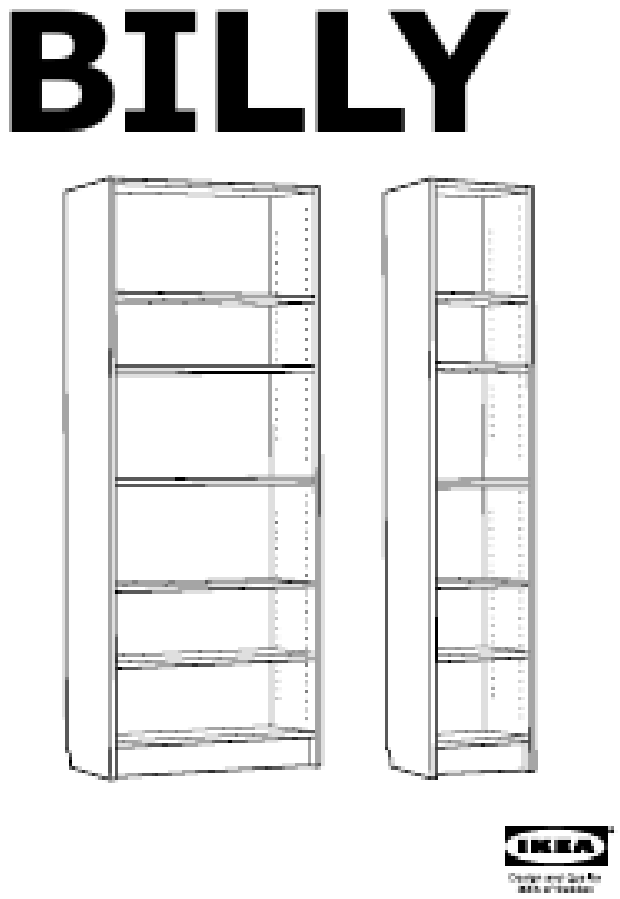
isometrisch: 1:1:1

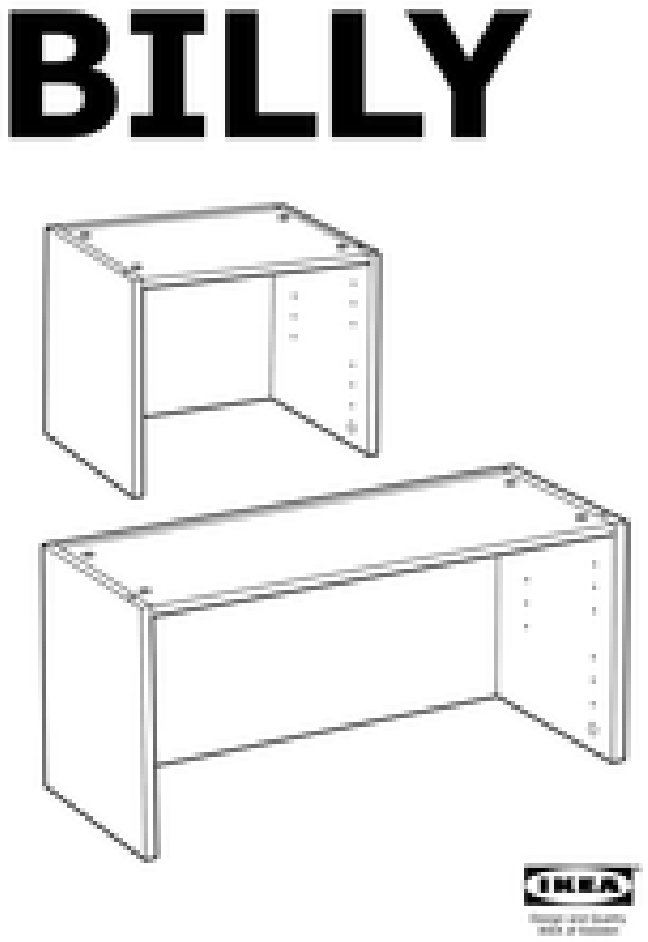
Kabinettprojektion



Kavalierprojektion

6.3 Ikea





6.4 Perspektivische Projektion – Berechnung

- Die Berechnung der perspektivischen Projektion erfolgt je nach Anwendung in unterschiedlichsten Konfigurationen.
- Diese können mittels geeigneter Transformationen des Koordinatensystems erreicht werden.

- Beispiel:
  - Projektionszentrum Z und der Augpunkt fallen zusammen.
  - Beide liegen
    - auf der positiven z-Achse
    - mit Abstand $d > 0$ zum Ursprung

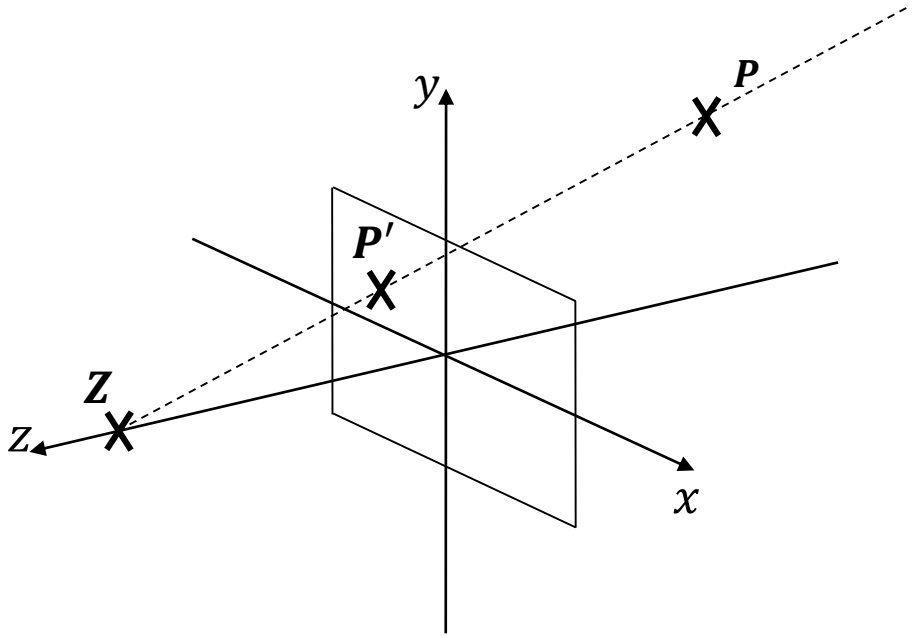
    $\rightarrow Z = (0,0,d)$.
  - Blickrichtung ist die negative z-Achse.
  - Bildebene liegt in der (x, y)-Ebene.

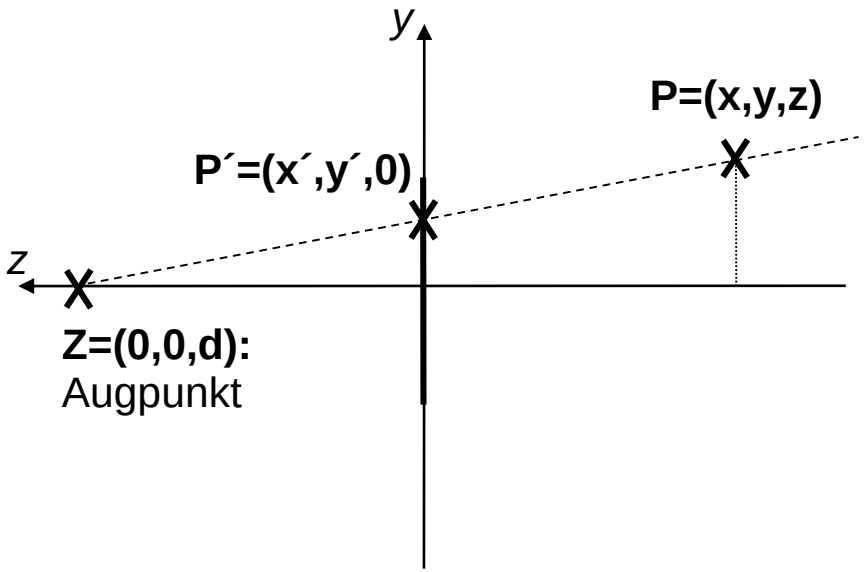
6.4 Perspektivische Projektion – Berechnung

– Aus dem Strahlensatz folgt:

$$\frac{x'}{d} = \frac{x}{d-z} \quad \Longrightarrow \quad x' = \frac{x \cdot d}{d-z}$$

$$\frac{y'}{d} = \frac{y}{d-z} \quad \Longrightarrow \quad y' = \frac{y \cdot d}{d-z}$$





6.4 Perspektivische Projektion – Berechnung

$$\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} \frac{x \cdot d}{d - z} \\ \frac{y \cdot d}{d - z} \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} x \cdot d \\ y \cdot d \\ 0 \\ d - z \end{pmatrix}$$

$$\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} d & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & d \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}^T = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{d} & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}^T = M \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}^T$$

6.4 Perspektivische Projektion – Berechnung

Zerlegung der perspektivischen Projektion

- Perspektivische Transformation M_T $(\mathbb{R}^3 \Rightarrow \mathbb{R}^3)$
- Parallele Projektion M_P auf die Ebene $z = 0$ $(\mathbb{R}^3 \Rightarrow \mathbb{R}^2)$

$$M = M_P \cdot M_T \Rightarrow \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{d} & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -\frac{1}{d} & 1 \end{pmatrix}$$

6.4 Perspektivische Projektion – Berechnung

Erweiterung

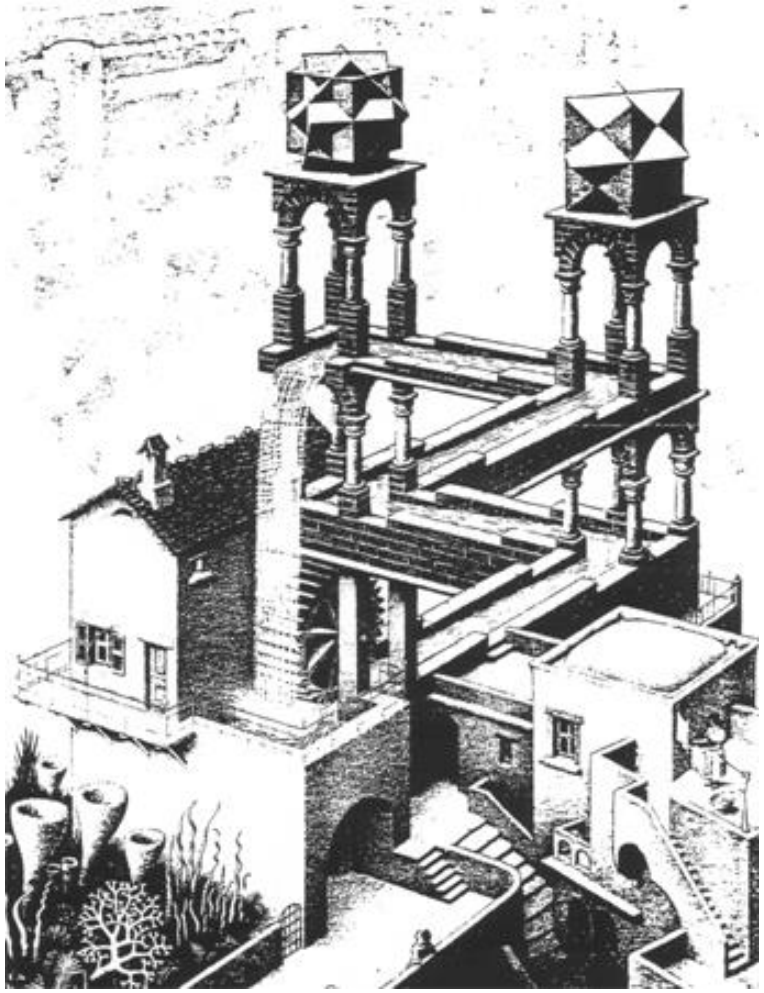
- In der Bildebene wird ein Sichtfenster (View Window) spezifiziert:
  - Breite b
  - Höhe h
  - Verhältnis Breite zu Höhe: aspect ratio
  - Das Sichtfenster ist symmetrisch um den Ursprung angeordnet
- Die Projektoren durch die Ecken der Bildebene definieren das so genannte Sichtvolumen (Viewing-Frustum)
- Zusätzlich begrenzen zwei zur Bildebene parallele Ebenen das Sichtvolumen in z-Richtung
  - Nahclipebene mit znah
  - Fernclipebene mit zfern

6.4 Perspektivische Projektion – Berechnung

Erweiterung

- Das Sichtvolumen begrenzt den Teil des Raums, der dargestellt werden soll
  $\Rightarrow$ Clipping

6.5 Unmögliche Strukturen



UNIVERSITÄT LEIPZIG

Viewport und Clipping

COMPUTERGRAPHIK

Inhaltsverzeichnis

7 Viewport und Clipping

7.1 Viewport

7.2 Clipping

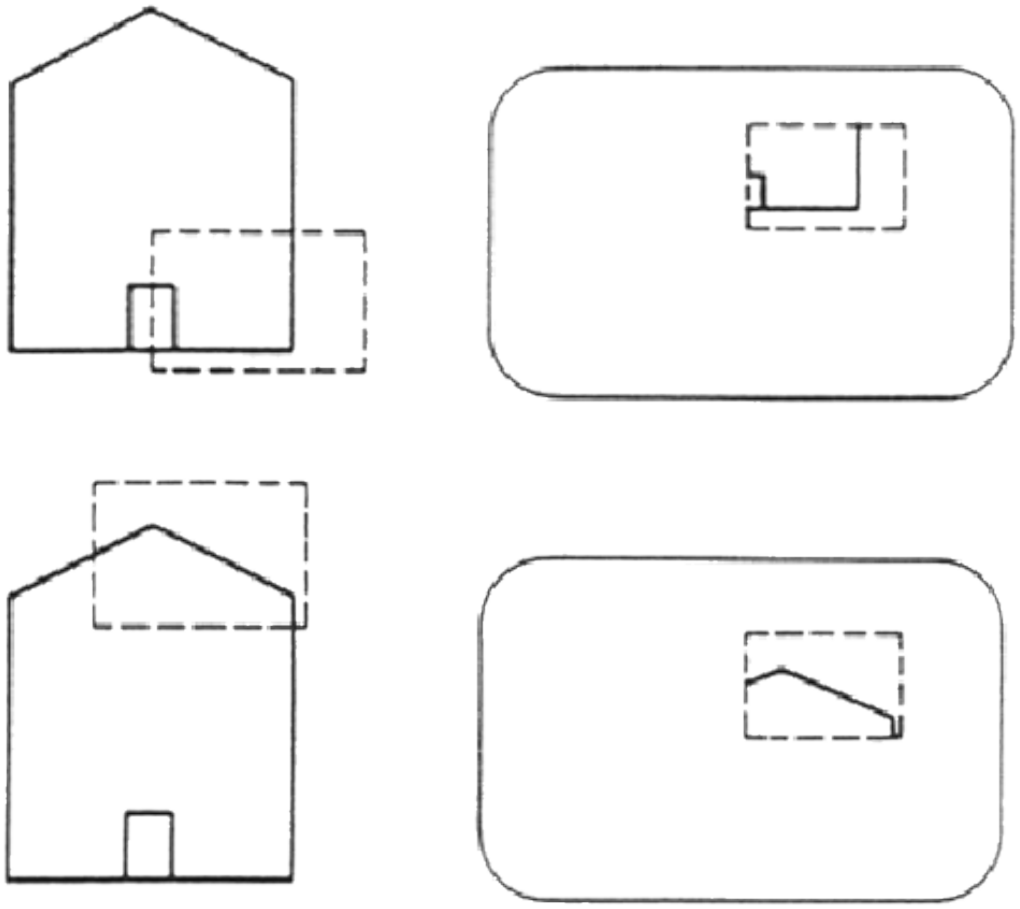
7.1 Viewport

- Window:
  - Definiert Sichtfenster in der Bildebene
  - Definiert, welcher Teilbereich der Szene abgebildet werden soll
  - Auch „View Window“ genannt
- Viewport:
  - Definiert Bildschirmbereich in dem der Inhalt eines Windows dargestellt werden soll

7.1 Viewport

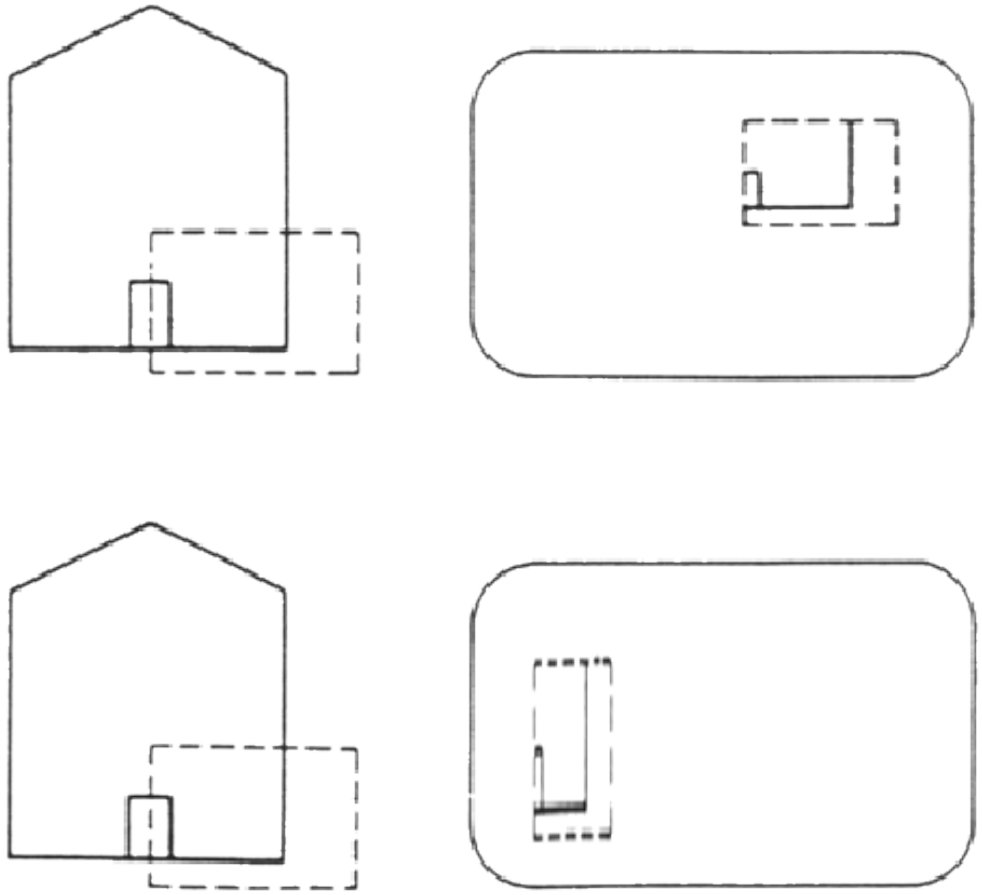
- In der Regel sind sowohl Window als auch Viewport an den Koordinatenachsen ausgerichtete rechteckige Gebiete
- Window-Viewport-Transformation (Windowing Operation):
  - elementare Verschiebungen und Skalierungen

7.1 Viewport



Verschiedene Fenster, dieselben Viewports

7.1 Viewport

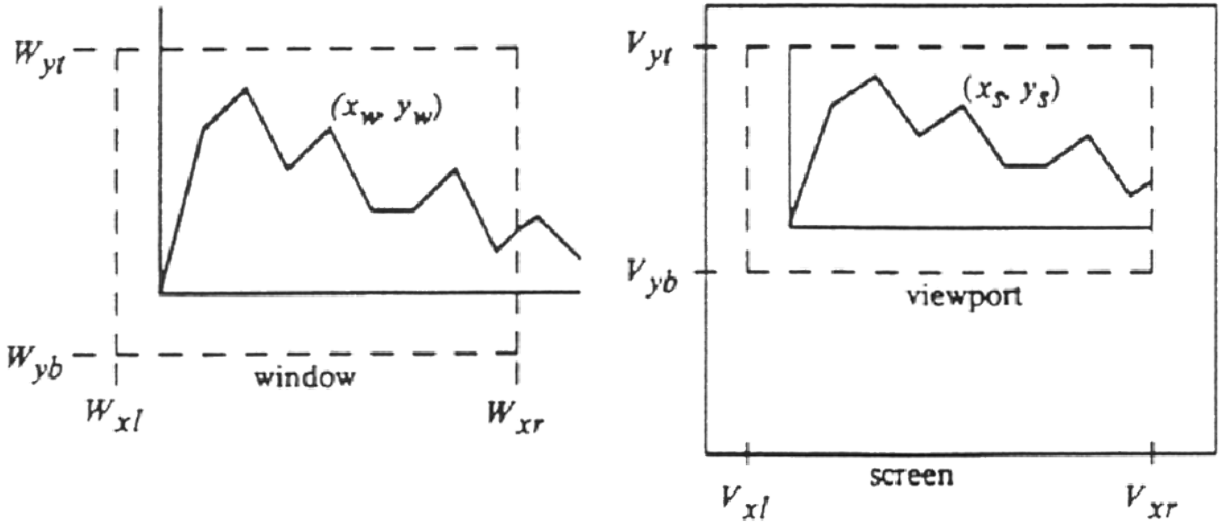


Dieselben Fenster, verschiedene Viewports

7.1 Viewport

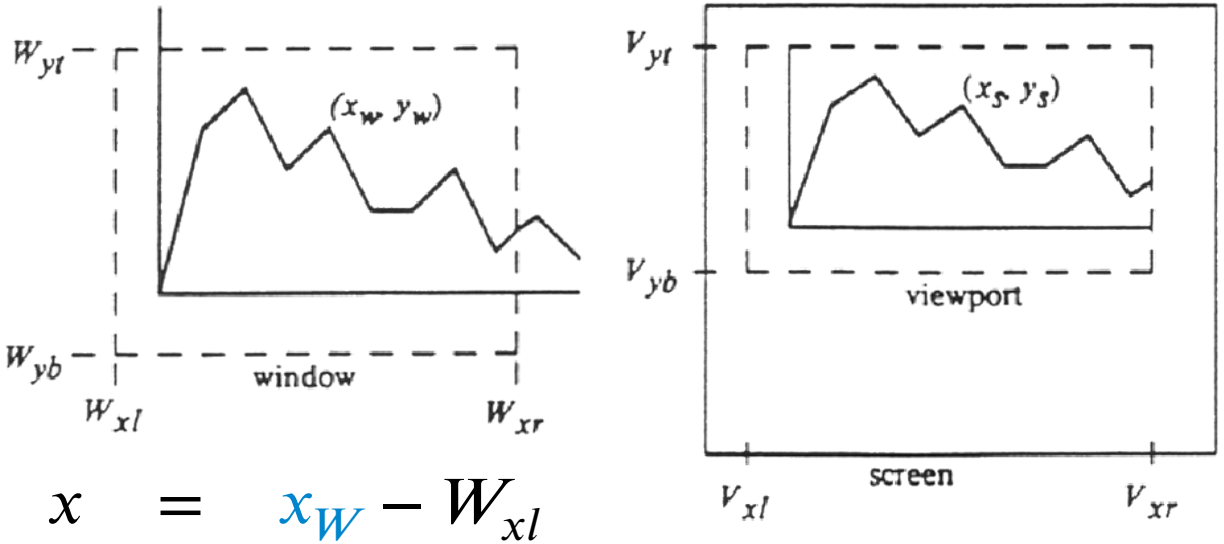
- Koordinaten des Windows:
  $W_{xl}, W_{xr}, W_{yb}, W_{yt}$
- Punktkoordinaten im Fenster (window):
  x_W, y_W

- Koordinaten des Viewports im Bildschirmkoordinatensystem
  $V_{xl}, V_{xr}, V_{yb}, V_{yt}$
- Punktkoordinaten auf dem Bildschirm (screen):
  x_S, y_S



7.1 Viewport

Transformation



1) Verschiebung
in den Koordinatenursprung

$$x = x_W - W_{xl}$$
$$y = y_W - W_{yb}$$

2) Skalierung
auf die gewünschte Größe

$$x' = \frac{V_{xr} - V_{xl}}{W_{xr} - W_{xl}} \cdot x$$
$$y' = \frac{V_{yt} - V_{xyb}}{W_{yt} - W_{yb}} \cdot y$$

3) Verschiebung
an die gewünschte Stelle

$$x_S = x' + V_{xl}$$
$$y_S = y' + V_{yb}$$

7.1 Viewport

Transformation

Zusammenfassung zu

$$x_S = a \cdot x_W + b$$
$$y_S = c \cdot y_W + d$$

$\Rightarrow$ Punkttransformation durch zwei Multiplikationen und zwei Additionen

$a,\ b,\ c,\ d$ fest für alle Punkttransformationen

$$a := \frac{V_{xr} - V_{xl}}{W_{xr} - W_{xl}}$$
$$c := \frac{V_{yt} - V_{xyb}}{W_{yt} - W_{yb}}$$

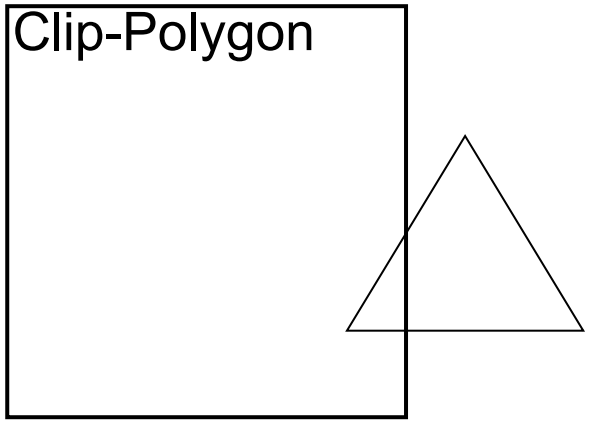
$$b = V_{xl} - a \cdot W_{xl}$$
$$d = V_{yb} - c \cdot W_{yb}$$

7.2 Clipping

Clip-Polygon

- Objekte in der Bildebene werden innerhalb eines Fensters dargestellt
- Alle außerhalb des Fensters liegenden Objektteile werden abgeschnitten:
  Clipping am Fensterrand

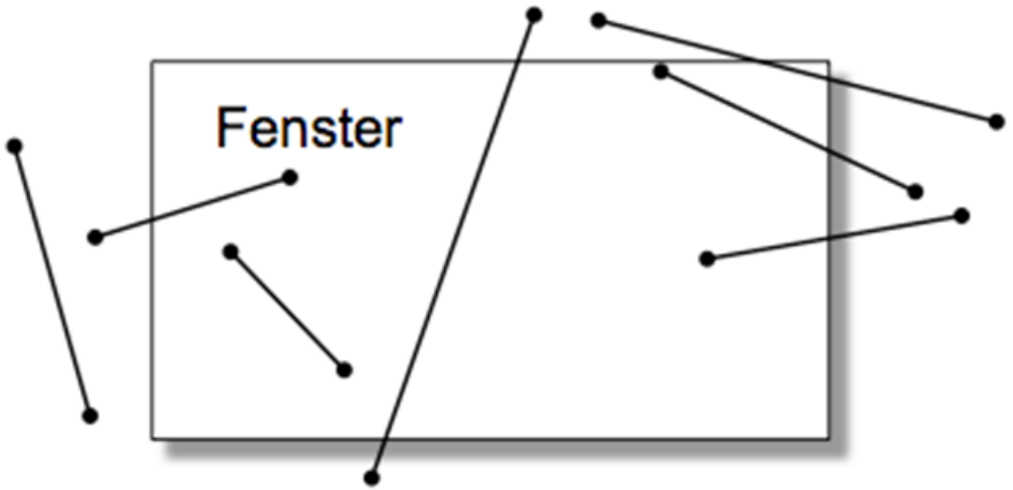
- Das Fenster wird Clip-Polygon genannt
  - typischerweise Rechtecke
  - andere Geometrie möglich
  - Nichtkonvexe oder nicht einfache Polygone sind problematisch



7.2 Clipping

Linien

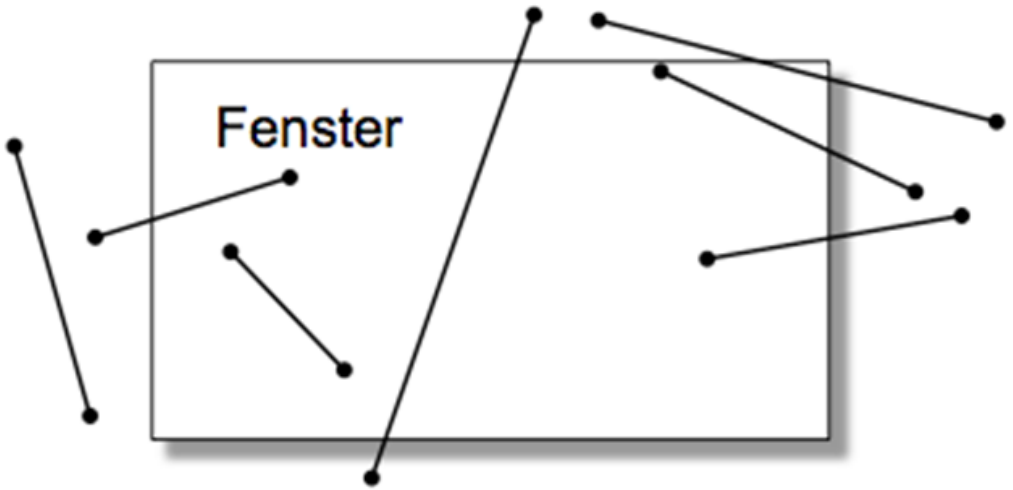
- Clip-Polygon
  - rechteckig
  - achsenparallel
- Fallunterscheidung
  1. beide Endpunkte innerhalb des Fensters
     ⇒ Linie zeichnen
  2. beide Endpunkte oberhalb oder unterhalb oder links oder rechts des Fensters
     ⇒ Linie nicht zeichnen



7.2 Clipping

Linien

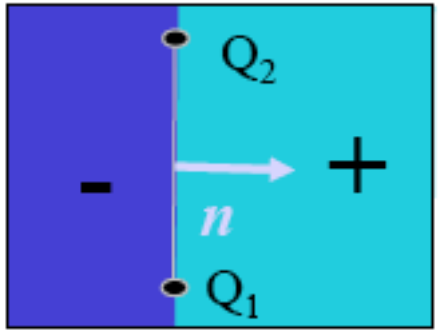
3. Sonst
   - Berechne die Schnittpunkte der Linie mit dem Fensterrand anhand der Geradengleichungen
   - Bestimme daraus die sichtbare Strecke



7.2 Clipping

Liang-Barsky-Algorithmus

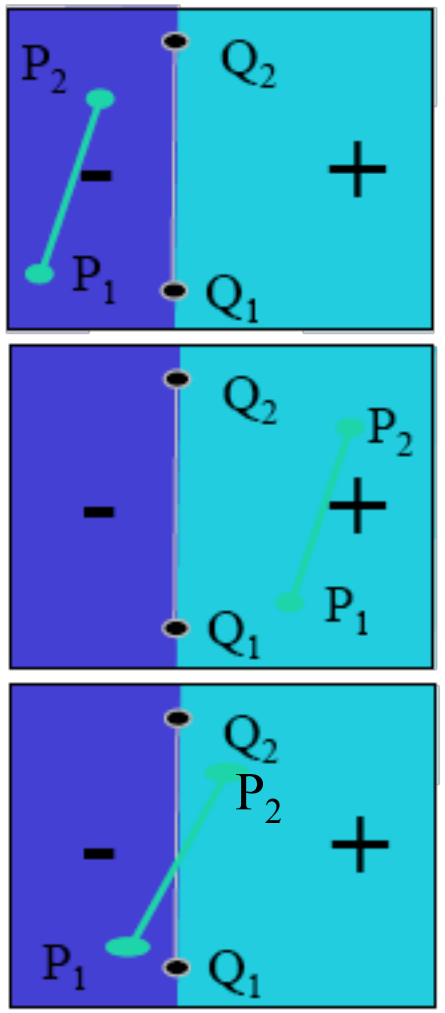
- Fensterkanten als implizite Gerade
  $Q_1 = (x_1, y_1), Q_2 = (x_2, y_2)$
  $n = (\Delta y, -\Delta x) = ((y_2 - y_1), -(x_2 - x_1))$
  $P = (x, y)$
- Normale n zeigt ins Innere
- Entscheidungsgröße
  $E(P) = \mathrm{n} \circ (P - Q_1) = n \circ P - n \circ Q_1$
  ist positiv, wenn P im Inneren liegt



7.2 Clipping

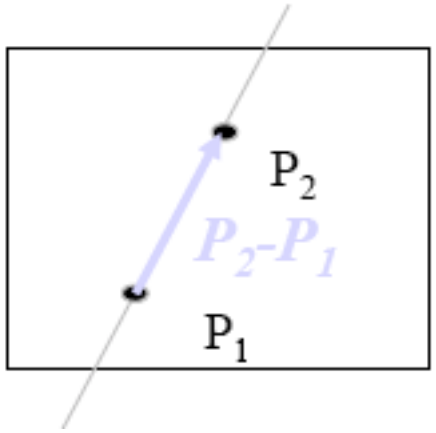
Liang-Barsky-Algorithmus

- Fallunterscheidung:
  1) P_1 und P_2 liegen außen
     $E(P_1) \leq 0,\ E(P_2) \leq 0$
  2) P_1 und P_2 liegen innen
     $E(P_1) \geq 0,\ E(P_2) \geq 0$
  3) P_1 und P_2 liegen auf verschiedenen Seiten
     $E(P_1) < 0,\ E(P_2) > 0$
     $E(P_1) > 0,\ E(P_2) < 0$



7.2 Clipping

Liang-Barsky-Algorithmus

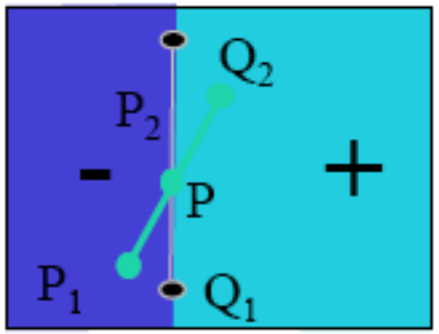


- Fall 3: Schnittpunkt P muss berechnet werden
- Parametrische Darstellung der Liniensegmente
  $P = l(t) = P_1 + t \cdot \left(P_2 - P_1\right)$
- in implizite Gleichung einsetzen:

  $E(P) = 0$

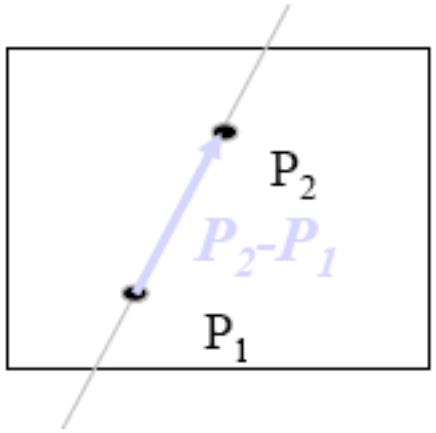
  $\Leftrightarrow\ P \circ n - Q_1 \circ n = 0$

  $\Leftrightarrow$



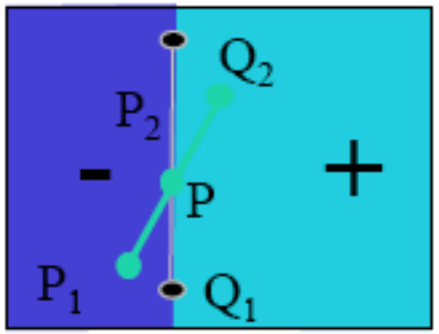
7.2 Clipping

Liang-Barsky-Algorithmus



- Fall 3: Schnittpunkt P muss berechnet werden
- Parametrische Darstellung der Liniensegmente
  $P = l(t) = P_1 + t \cdot (P_2 - P_1)$
- in implizite Gleichung einsetzen:

$$E(P) = 0$$
$$\Leftrightarrow\ P \circ n - Q_1 \circ n = 0$$
$$\Leftrightarrow\ \left(P_1 + t \cdot (P_2 - P_1)\right) \circ n - Q_1 \circ n = 0$$
$$\Leftrightarrow\ t = \frac{Q_1 \circ n - P_1 \circ n}{(P_2 - P_1) \circ n}$$



- zurück in die parametrische Gleichung

$$P = P_1 + \frac{Q_1 \circ n - P_1 \circ n}{(P_2 - P_1) \circ n} \cdot (P_2 - P_1)$$

7.2 Clipping

Cohen-Sutherland Line-Clipping Algorithmus

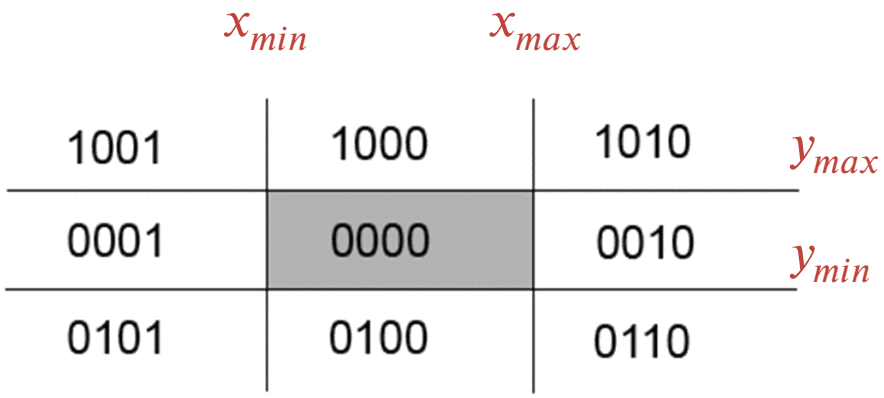
- Kern des Algorithmus ist ein schnelles Verfahren zur Bestimmung der Kategorie einer Linie
  - innerhalb
  - außerhalb
  - schneidend
- Gegeben:
  - Ein Fenster: $(x_{min}, y_{min}, x_{max}, y_{max})$ dessen begrenzende Geraden (Halbebenen) die Bildebene in neun Regionen unterteilen
- Jeder Region ist ein eindeutiger 4-Bit-Code (Outcode) zugeordnet, der Auskunft über deren Lage in Bezug auf das Fenster gibt
- Im 3D sind es 27 Regionen (3^3) und ein 6-Bit-Outcode

7.2 Clipping

Cohen-Sutherland Line-Clipping Algorithmus

- Outcode-Bits werden gesetzt, falls der Eckpunkt in der jeweiligen Region liegt

Bit	Ort	Bedingung
0	links des Fensters	$x < x_{min}$
1	rechts des Fensters	$x > x_{max}$
2	unterhalb des Fensters	$y < y_{min}$
3	oberhalb des Fensters	$y > y_{max}$

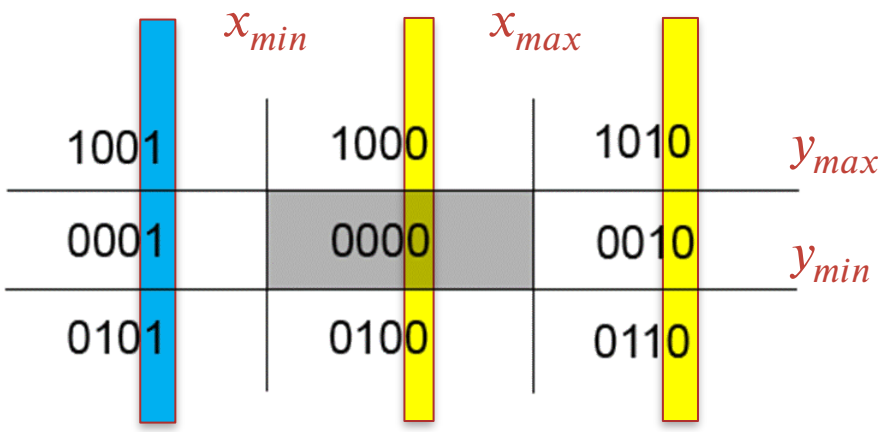


7.2 Clipping

Cohen-Sutherland Line-Clipping Algorithmus

- Outcode-Bits werden gesetzt, falls der Eckpunkt in der jeweiligen Region liegt

Bit	Ort	Bedingung
0	**links des Fensters**	$x < x_{min}$
1	rechts des Fensters	
2	unterhalb des Fensters	
3	oberhalb des Fensters	

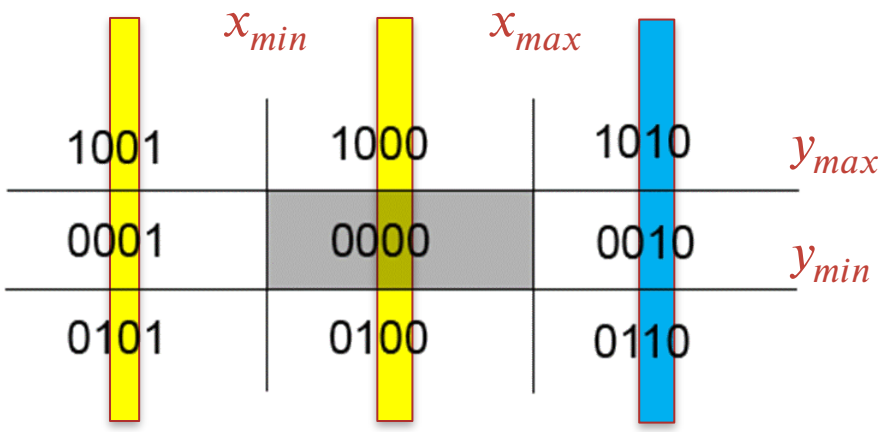


7.2 Clipping

Cohen-Sutherland Line-Clipping Algorithmus

- Outcode-Bits werden gesetzt, falls der Eckpunkt in der jeweiligen Region liegt

Bit	Ort	Bedingung
0	links des Fensters	$x < x_{min}$
1	**rechts des Fensters**	$x > x_{max}$
2	unterhalb des Fensters	
3	oberhalb des Fensters	

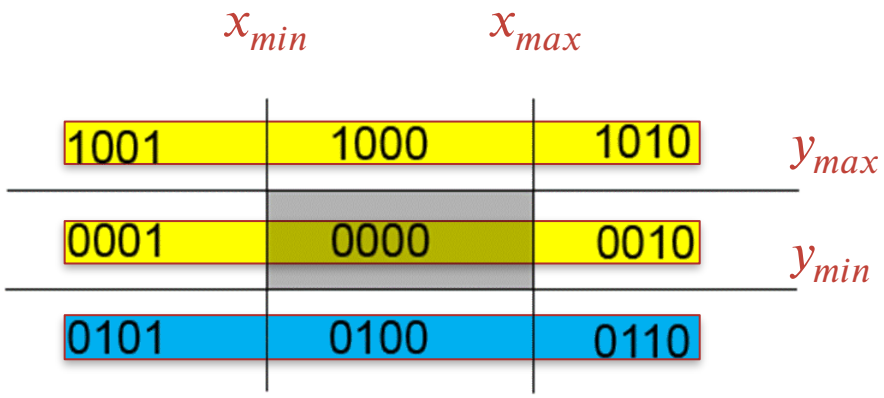


7.2 Clipping

Cohen-Sutherland Line-Clipping Algorithmus

- Outcode-Bits werden gesetzt, falls der Eckpunkt in der jeweiligen Region liegt

Bit	Ort	Bedingung
0	links des Fensters	$x < x_{min}$
1	rechts des Fensters	$x > x_{max}$
2	**unterhalb des Fensters**	$y < y_{min}$
3	oberhalb des Fensters	

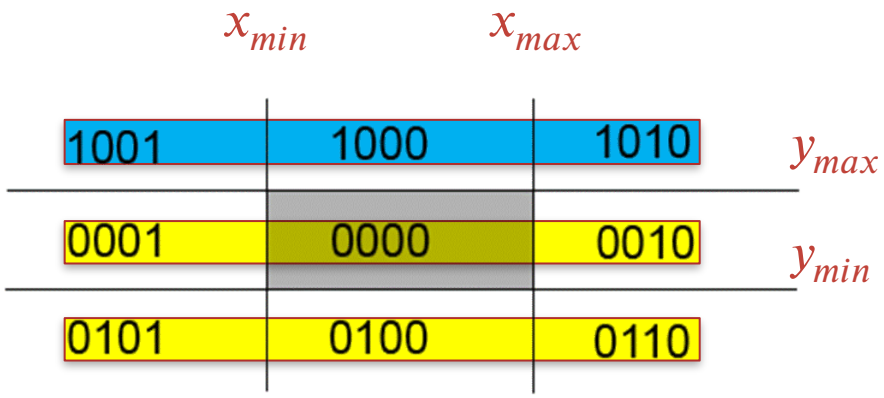


7.2 Clipping

Cohen-Sutherland Line-Clipping Algorithmus

- Outcode-Bits werden gesetzt, falls der Eckpunkt in der jeweiligen Region liegt

Bit	Ort	Bedingung
0	links des Fensters	$x < x_{min}$
1	rechts des Fensters	$x > x_{max}$
2	unterhalb des Fensters	$y < y_{min}$
3	**oberhalb des Fensters**	$y > y_{max}$



7.2 Clipping

Cohen-Sutherland Line-Clipping Algorithmus

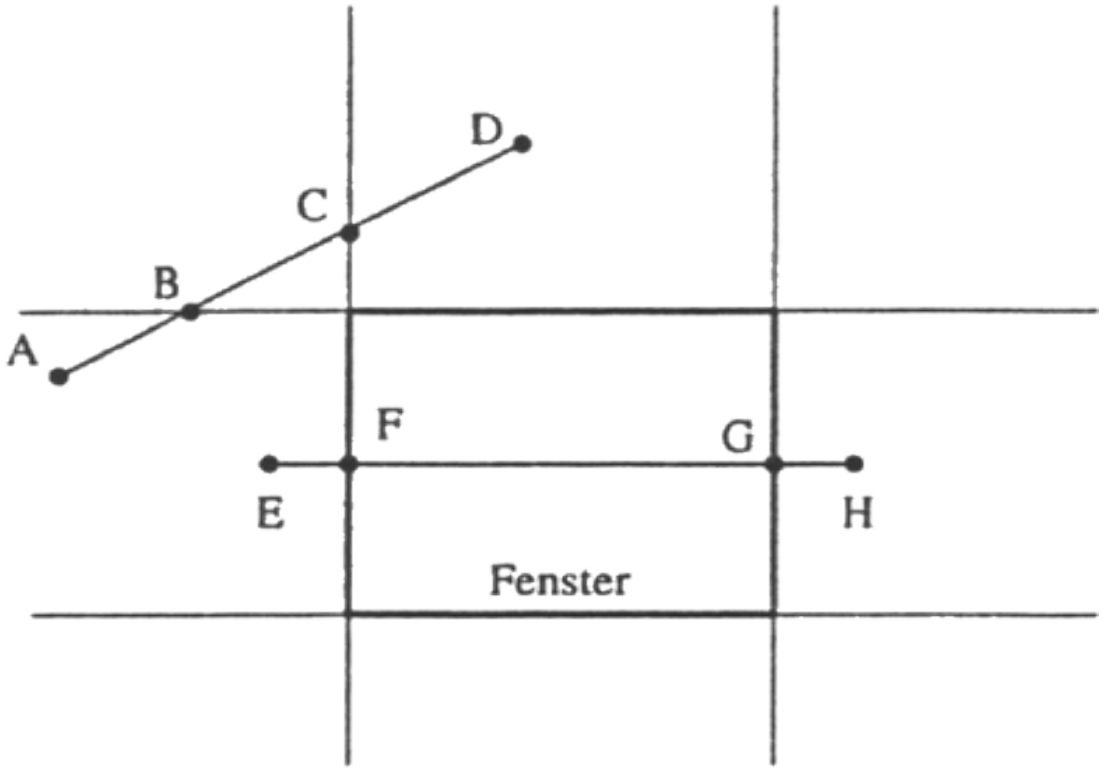
- Bestimme Outcodes für Endpunkte einer Linie
  - Linie liegt vollständig außerhalb des Fensters (Trivial Reject):
    Durchschnitt der Codes beider Endpunkte ist von 0 verschieden
    (AND-Verknüpfung $\neq 0$)
  - Linie liegt komplett im Fenster (Trivial Accept):
    Beide Endpunkte besitzen den 4-Bit-Code 0000
    (OR-Verknüpfung $= 0$)
- Sonst:
  - Schneide die Linie nacheinander mit den das Fenster begrenzenden Geraden
    → es entstehen zwei Linien
  - Außerhalb des Fensters liegende Teile können sofort entfernt werden

7.2 Clipping

Cohen-Sutherland Line-Clipping Algorithmus

Linie AD

- Codes A: 0001 und D: 1000
- 0001 AND 1000 = 0000 (kein reject)
- 0001 OR 1000 = 1001 (kein accept)
  - Schnitt mit linker Fenstergrenze liefert
    - Punkte A und C liegen links: eliminiere AC
- Codes C: 1000 und D: 1000
- 1000 AND 1000 = 1000 (reject)
    - Punkte C und D liegen oberhalb: eliminiere CD

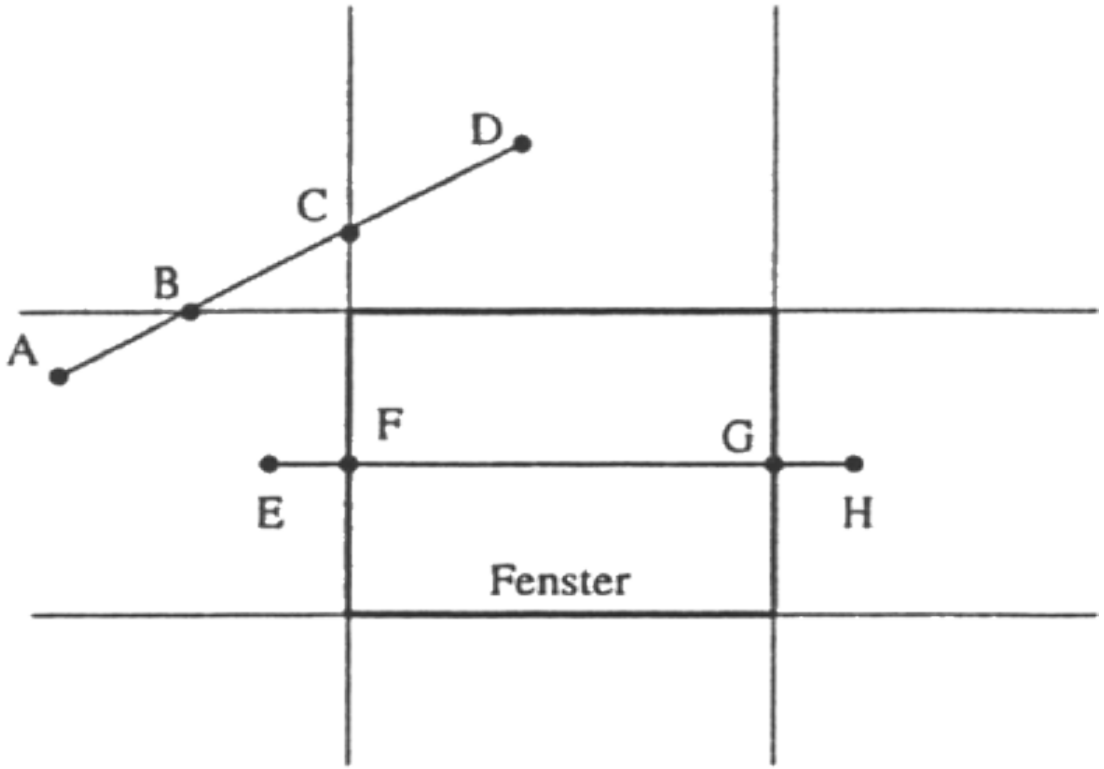


7.2 Clipping

Cohen-Sutherland Line-Clipping Algorithmus

Linie EH

- Codes E: 0001 und H: 0010
- 0001 AND 0010 = 0000 (kein reject)
- 0001 OR 0010 = 1010 (kein accept)
  - Schnitt mit linker Fenstergrenze liefert F
    - eliminiere EF

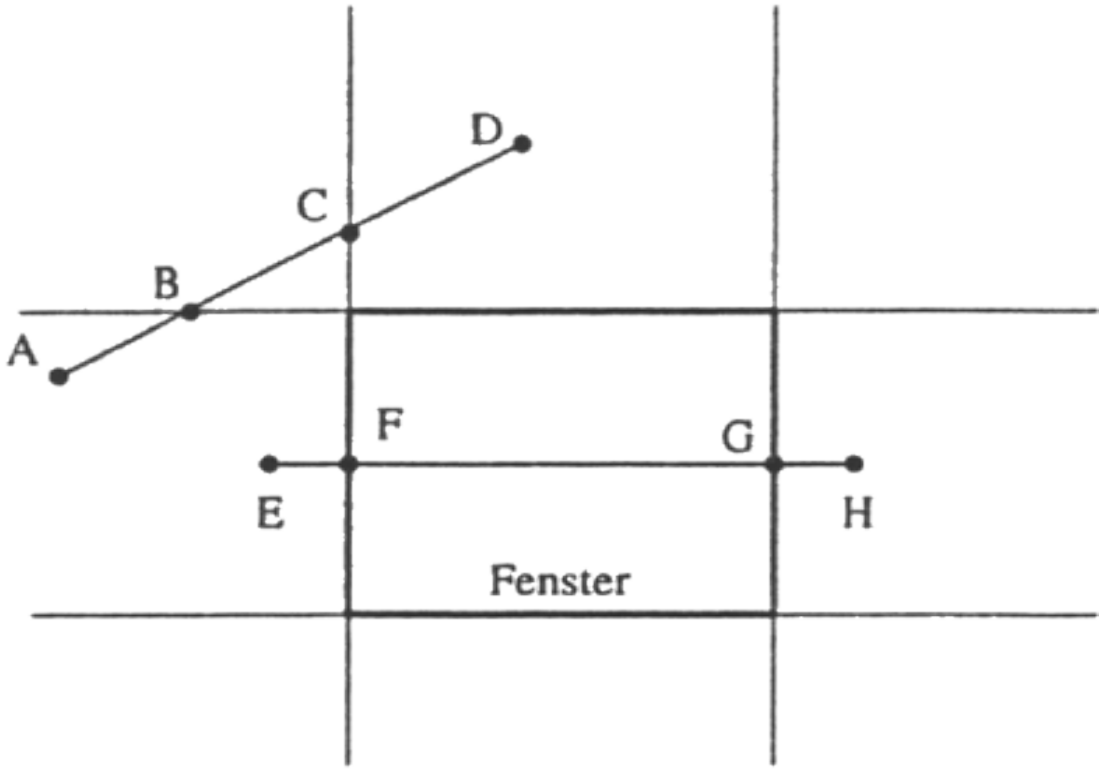


7.2 Clipping

Cohen-Sutherland Line-Clipping Algorithmus

Linie EH

- Codes F: 0000 und H: 0010
- 0000 AND 0010 = 0000 (kein reject)
- 0000 OR 0010 = 0000 (kein accept)
  - Schnitt von FH mit der rechten Fenstergrenze liefert G
    - eliminiere GH
- Codes F: 0000 und G: 0000
- 0000 AND 0010 = 0000 (kein reject)
- 0000 OR 0000 = 0000 (accept)
  - FG wird gezeichnet



7.2 Clipping

Cohen-Sutherland Line-Clipping Algorithmus: Spezialfälle & Beschleunigungen

- Bei senkrecht oder waagrecht verlaufenden Linien muss nur gegen die y- bzw. x-Grenzen getestet und geschnitten werden
- Falls genau ein Endpunkt innerhalb des Fensters liegt, gibt es nur einen Schnitt mit dem Fensterrand
- Einige Schnittoperationen führen nicht zu Schnittpunkten am Fensterrand
  - Jedes Bit korrespondiert genau zu einem der Fensterränder
  - Betrachte nur Fensterränder deren zugehöriges Bit in den zwei Endpunkt-Codes unterschiedlich gesetzt ist

7.2 Clipping

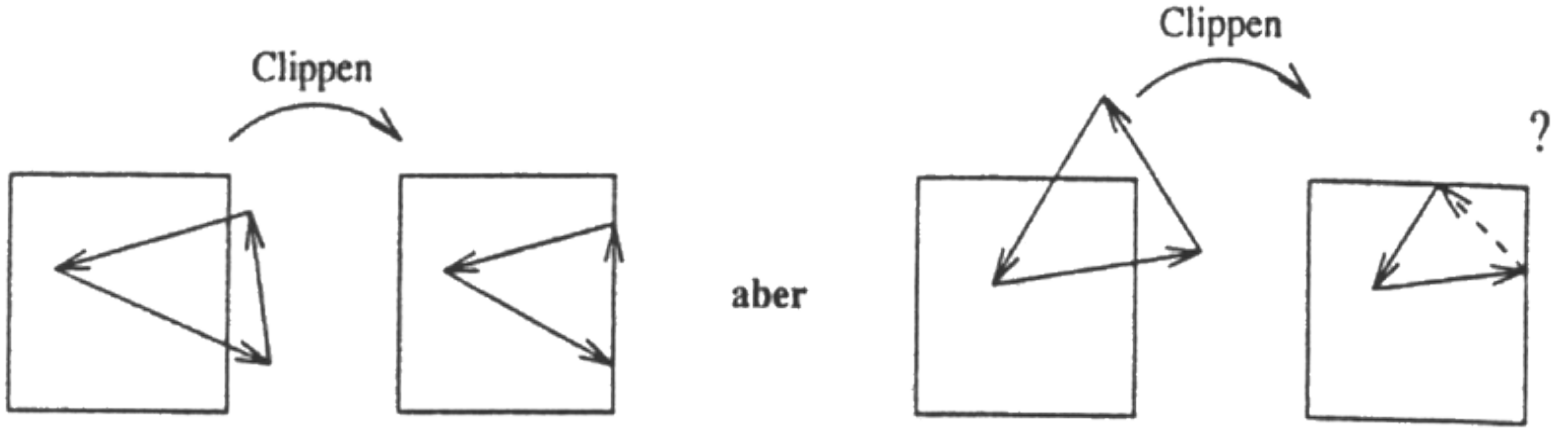
Cohen-Sutherland Line-Clipping Algorithmus: Spezialfälle & Beschleunigungen

- Vermeidung der aufwändigen Schnittpunktberechnung durch Bisektionsmethode
  - Linien, die weder ganz außerhalb, noch ganz innerhalb des Fensters liegen, werden so lange unterteilt, bis ihre Länge kleiner als ein Pixel ist
  - Bei 2^10=1024 Pixel in einer Zeile bzw. Spalte erfordert dies maximal 10 Unterteilungen (⇒ Mittelpunktalgorithmus)
  - In Hardware ist diese Variante schneller als eine direkte Schnittberechnung
    - schnelle Division durch 2 (Bitshift)
    - Parallelisierbarkeit

7.2 Clipping

Polygone

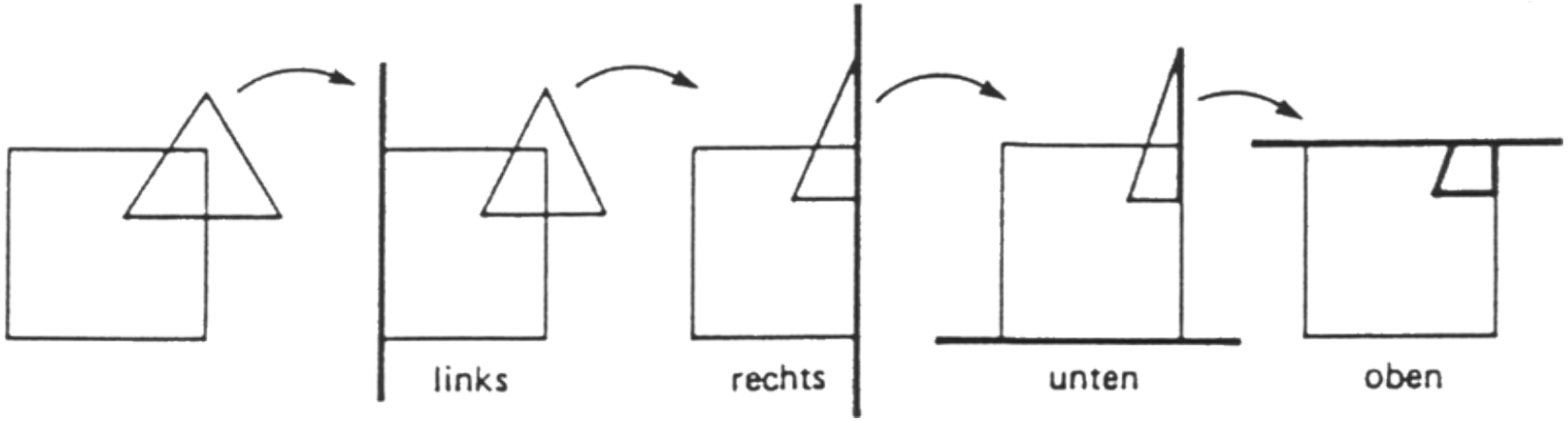
- Polygone begrenzen Flächen
  - Polygon-Clipping muss wieder geschlossene Polygone liefern
  - Teile des Fensterrandes müssen eingebaut werden
- Ein einfacher Algorithmus würde jede Seite gegen die Fenster clippen
  - Wenn eine Seite das Fenster verlässt, wird der Austrittspunkt mit dem Wiedereintritt verbunden
  - Ecken können zu Problemen führen



7.2 Clipping

Sutherland-Hodgman Polygon-Clipping Algorithmus

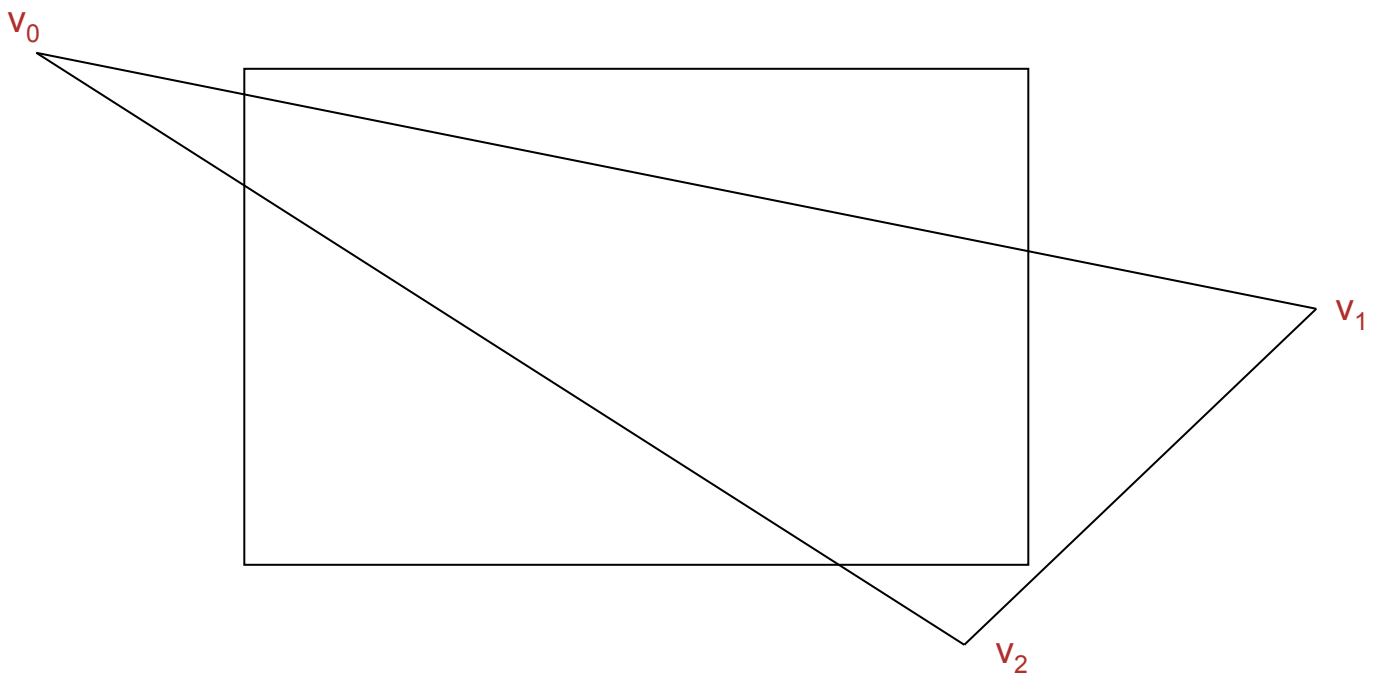
- Problem:
  Clippen jeder Polygonseite gegen alle 4 Fensterseiten
- Vollständiges Clippen des Polygons gegen eine Fensterseite nach der anderen führt zum Ziel
- Die Zwischenergebnisse müssen gespeichert werden



7.2 Clipping

Sutherland-Hodgman Polygon-Clipping Algorithmus

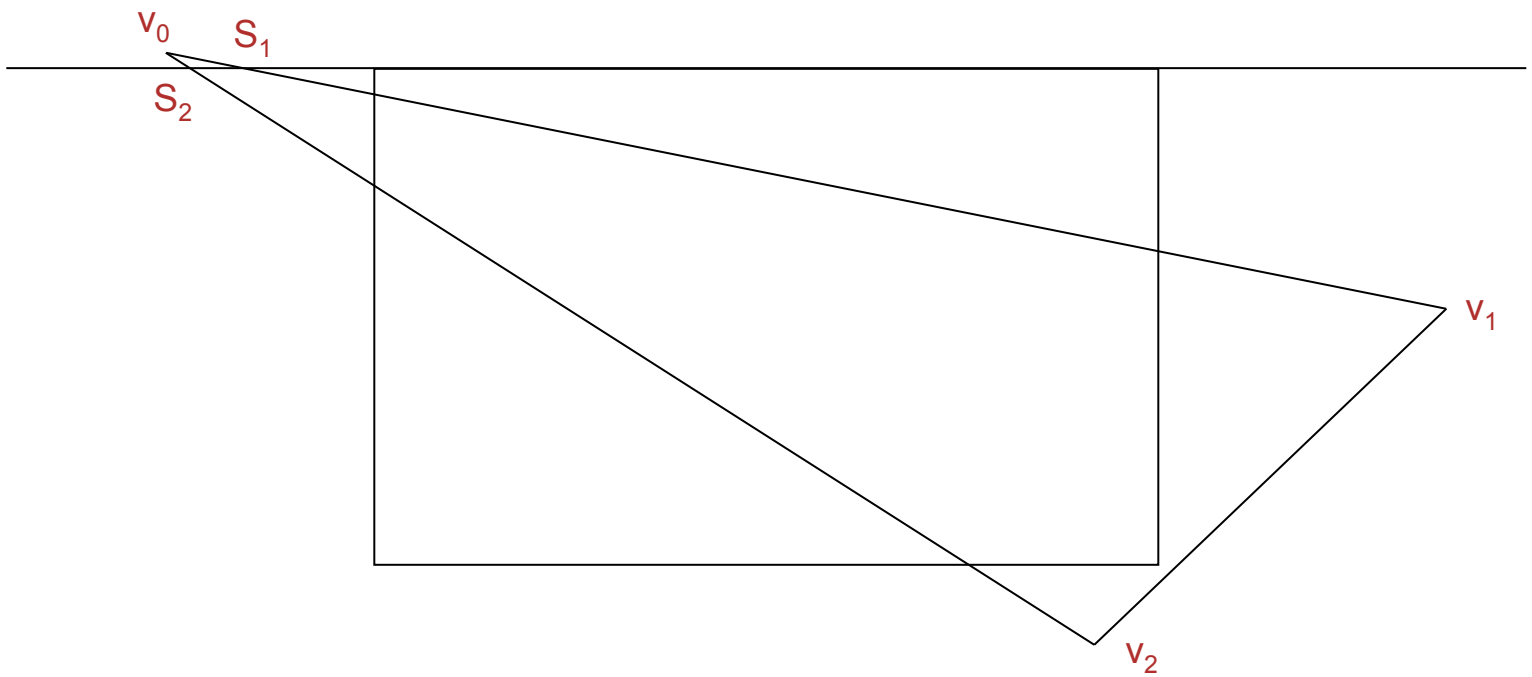
– Ausgangssituation: aktuelles Polygon $\{v_0, v_1, v_2\}$



7.2 Clipping

Sutherland-Hodgman Polygon-Clipping Algorithmus

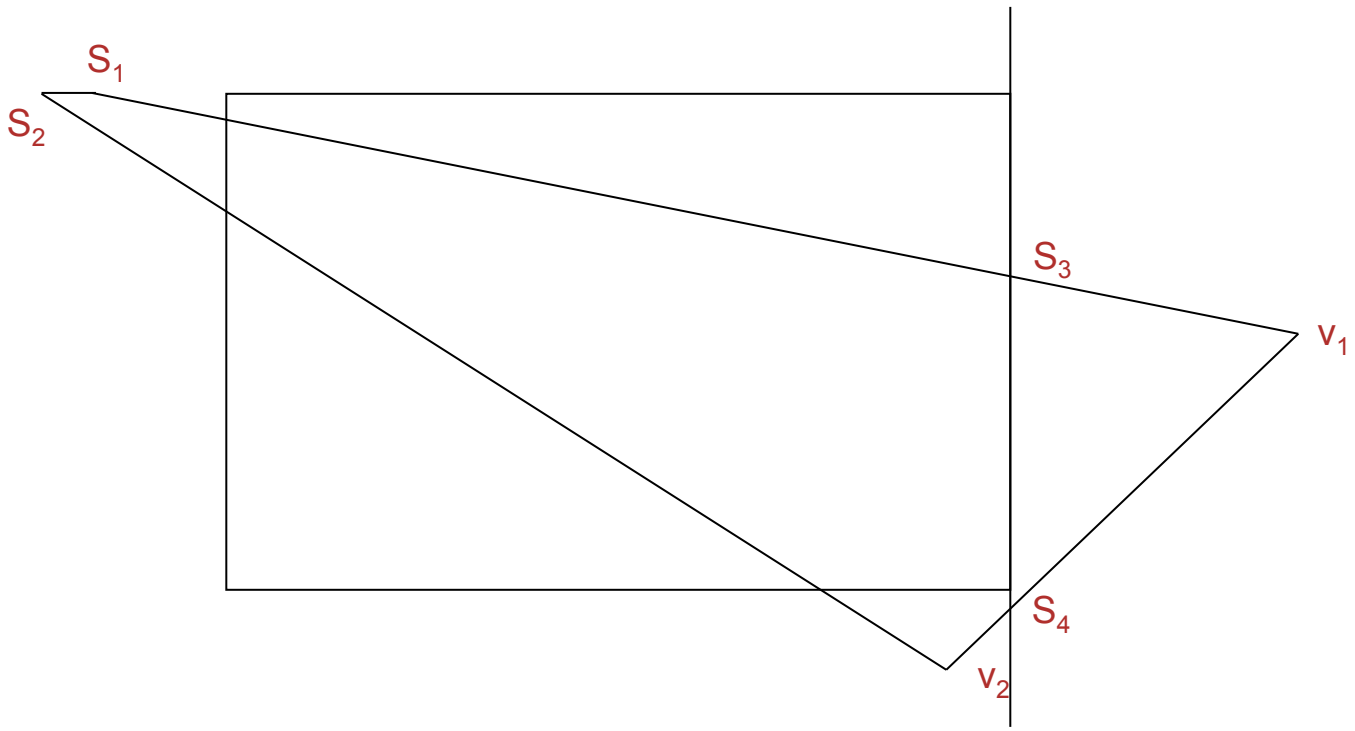
– Clip oben: aktuelles Polygon $\{s_1, v_1, v_2, s_2\}$



7.2 Clipping

Sutherland-Hodgman Polygon-Clipping Algorithmus

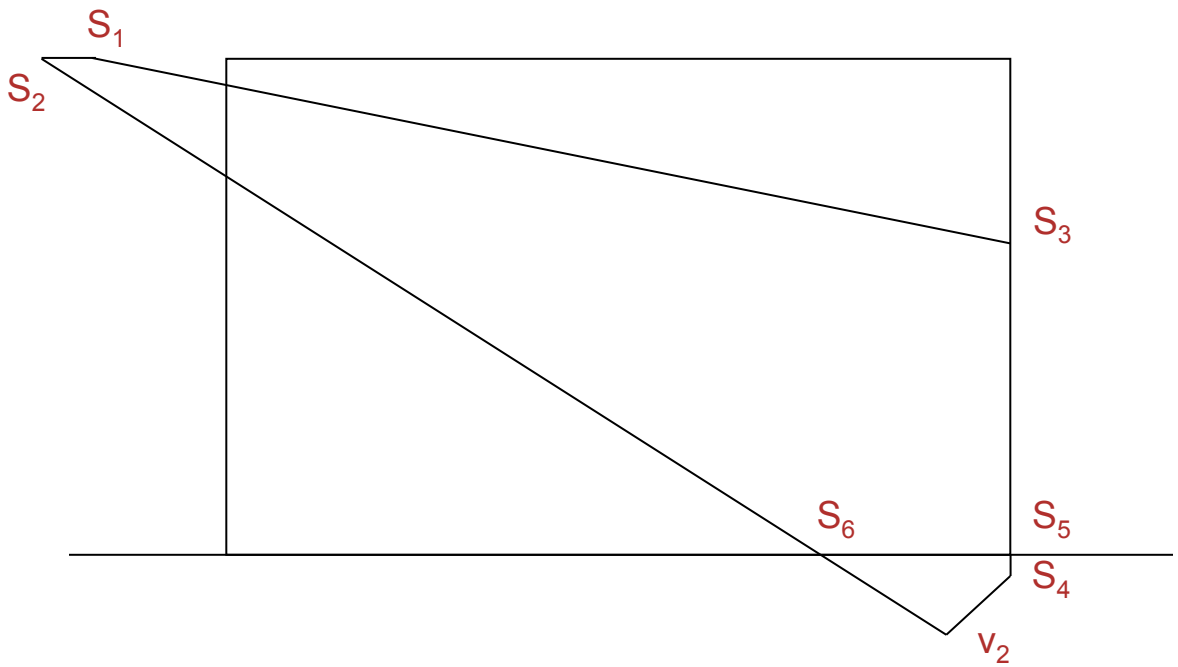
- Clip rechts: aktuelles Polygon $\left\{ s_1, s_3, s_4, v_2, s_2 \right\}$



7.2 Clipping

Sutherland-Hodgman Polygon-Clipping Algorithmus

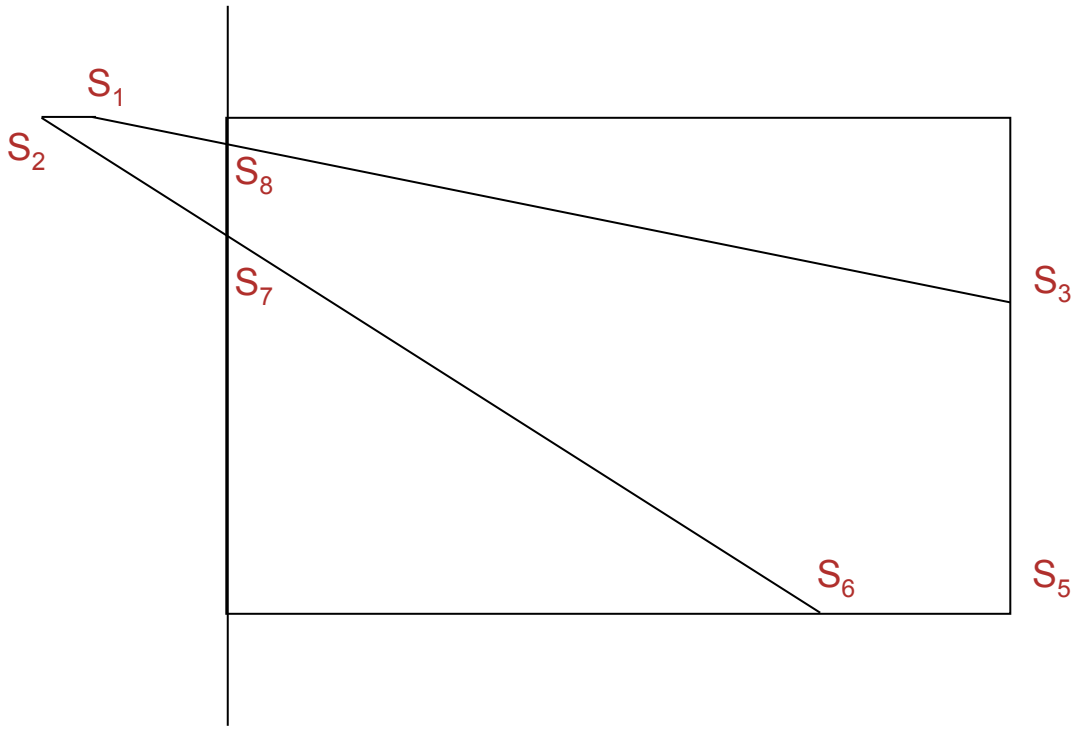
- Clip unten: aktuelles Polygon $\left\{ s_1, s_3, s_5, s_6, s_2 \right\}$



7.2 Clipping

Sutherland-Hodgman Polygon-Clipping Algorithmus

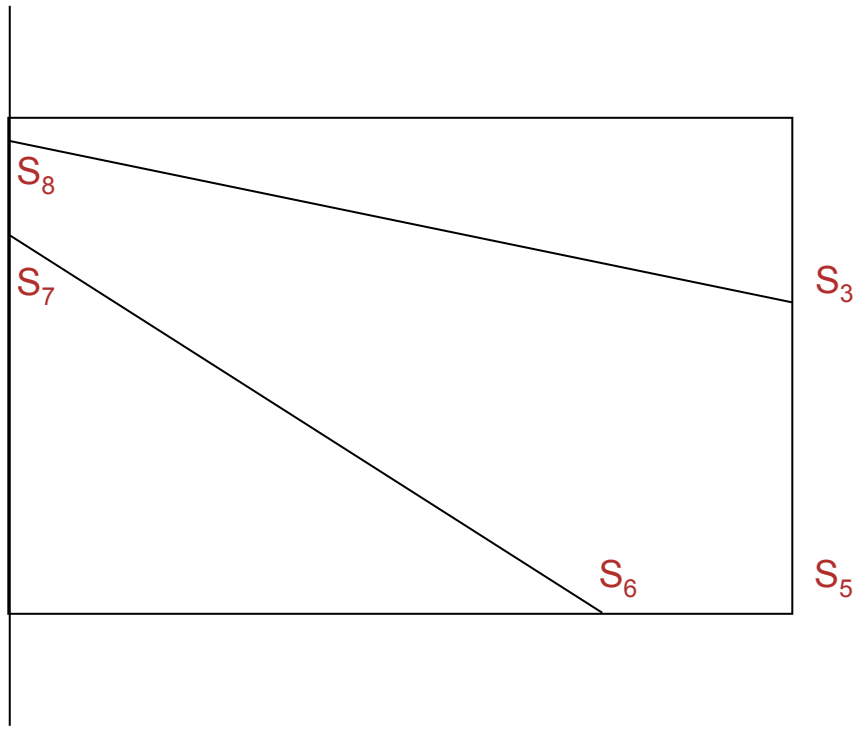
- Clip links: aktuelles Polygon $\left\{ s_8, s_3, s_5, s_6, s_7 \right\}$



7.2 Clipping

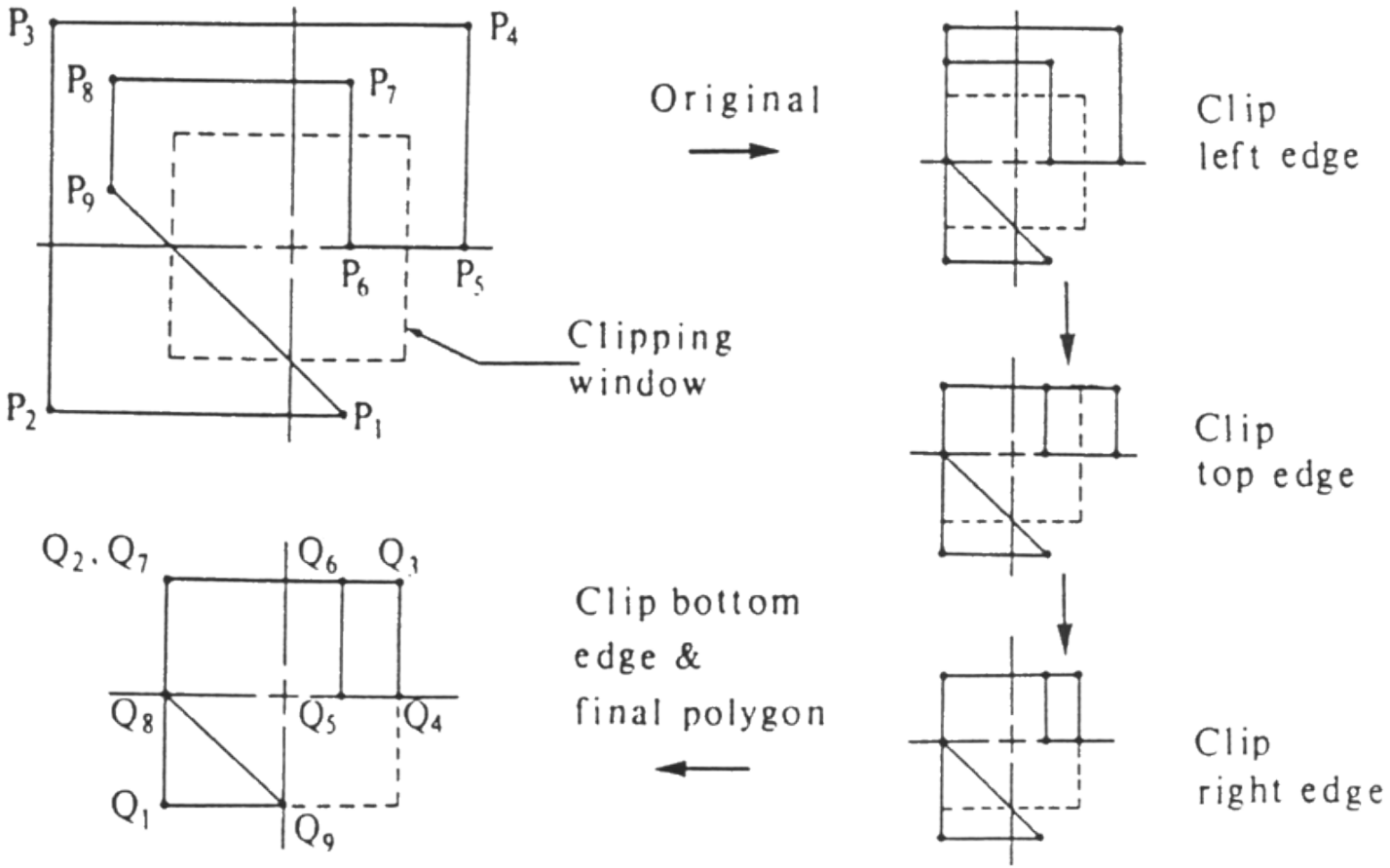
Sutherland-Hodgman Polygon-Clipping Algorithmus

- Ergebnis-Polygon $\left\{ s_8, s_3, s_5, s_6, s_7 \right\}$



7.2 Clipping

Sutherland-Hodgman Polygon-Clipping Algorithmus

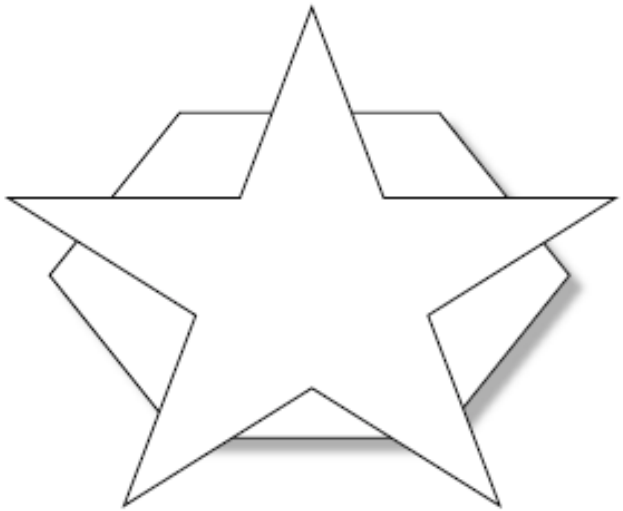


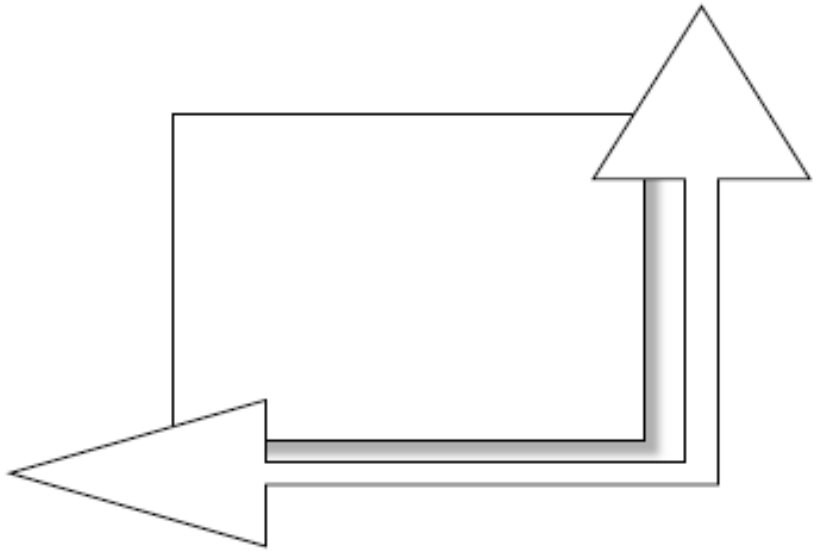
7.2 Clipping

Sutherland-Hodgman Polygon-Clipping Algorithmus

- Fenster konvexes Polygon und
- Kandidat einfaches Polygon

⇒ Ergebnis ist immer ein geschlossener Kantenzug





7.2 Clipping

Andere Polygon-Clipping-Verfahren

- Vatti-Algorithmus
  Scan-Line
  [CACM 35 1992]

- Greiner/Hormann
  [ACM TOG 17(2),1998]

7.2 Clipping

Clipping in 3D

- Clipping nach Transformation in das normalisierte Sichtvolumen (NDC)

- Clippingverfahren lassen sich einfach in 3D übertragen:
  Clippen an sechs Halbebenen statt vier

- Alternativ:
  Clipping nach Projektion in die zweidimensionale Bildebene
  → w-Clipping

UNIVERSITÄT
LEIPZIG

Sichtbarkeit

COMPUTERGRAPHIK

Inhaltsverzeichnis

8 Sichtbarkeit

- 8.1 Problemstellung
- 8.2 Projektion
- 8.3 Strahlverfolgung
- 8.4 Raumteilung
- 8.5 Culling

8.1 PROBLEMSTELLUNG

- Sichtbarkeitsberechnung
  - Ihr Ziel ist die (möglichst) exakte Bestimmung der von einem gegebenen Blickpunkt aus sichtbaren bzw. unsichtbaren Teile der darzustellenden Szene
- Anforderungen
  - Wünschenswert ist **hohe Interaktionsrate**, so dass Eingaben des Benutzers sich direkt auf die Darstellung auswirken
  - In den meisten Fällen ist die **Echtzeitausgabe** der Szene nötig
- Einteilung der Verfahren:
  - **Objektraumverfahren**
    - prinzipiell geräteunabhängig
    - Rechengenauigkeit ist die Maschinengenauigkeit
  - **Bildraumverfahren**
    - geräteabhängig
    - Rechengenauigkeit ist die Auflösung des Ausgabegerätes

8.1 PROBLEMSTELLUNG

- Bestimmung der sichtbaren Objekte
  - Welches Objekt liegt dem Betrachter (Bildebene) am nächsten?
  - Notwendig für die korrekte Darstellung
  - Abhängig von der Darstellungsart
- Wireframe-Darstellung:
  - Darstellung der Begrenzungskanten von Flächen
  - Entfernung verdeckter Kanten (Hidden Line Removal, HLR)
- Flächendarstellung
  - Darstellung der Flächen
  - Hidden Surface Removal (HSR)
- Transparenzen müssen ggf. berücksichtigt werden

8.2 Projektion

- Dreidimensionale Szene wird auf die Bildebene projiziert
  - Unterschiedliche Objektteile werden auf dieselbe Stelle abgebildet
    - (x, y)-Koordinate in der Bildebene
  - Sichtbar sind diejenigen Objektpunkte, die dem Auge des Betrachters am nächsten gelegen sind
    - Tiefenrelation der Szene (z-Koordinate)
- Beschleunigung
  - Berechnung nicht sichtbarer Teile der Szene (Culling)
  - Nutzt Heuristiken
  - In der Regel keine exakte Lösung

8.2 Projektion

Kohärenz: Ausnutzung lokaler Ähnlichkeiten

- Objektkohärenz:
  schneiden sich zwei Objekte nicht, so müssen auch ihre Flächen nicht auf Schnitt getestet werden

- Flächenkohärenz:
  Eigenschaften benachbarter Punkte auf einer Fläche ändern sich oft nur unwesentlich

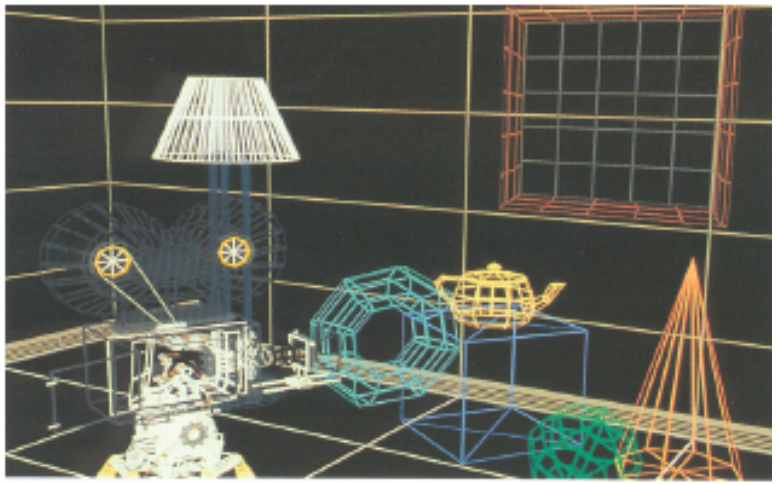
- Tiefenkohärenz:
  Die Tiefe z an der Position (x, y) einer Fläche kann oft inkrementell berechnet werden

- Zeit-/Framekohärenz:
  Oft ändern sich nur wenige Anteile eines Frames

8.2 Projektion

Wireframe-Darstellung

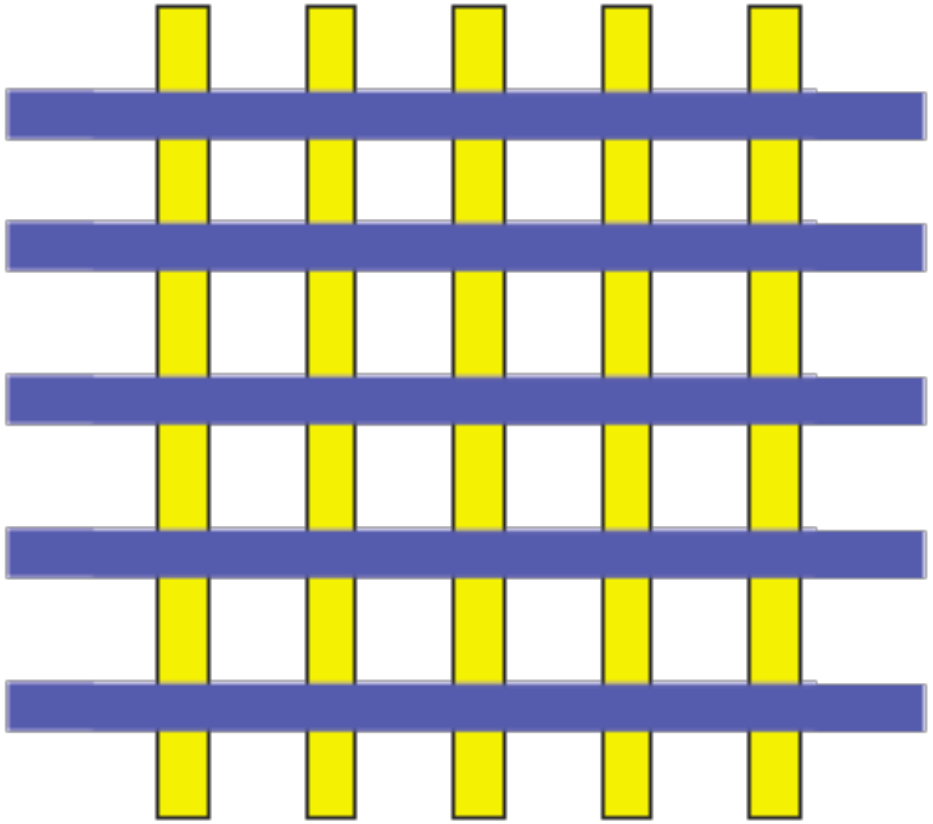
- Kanten der Flächen werden dargestellt
- Eigentlich verdeckte Kanten scheinen durch
- Hidden-Line-Removal: Entfernen verdeckter Kanten



8.2 Projektion

Sichtbarkeit von Polygonen

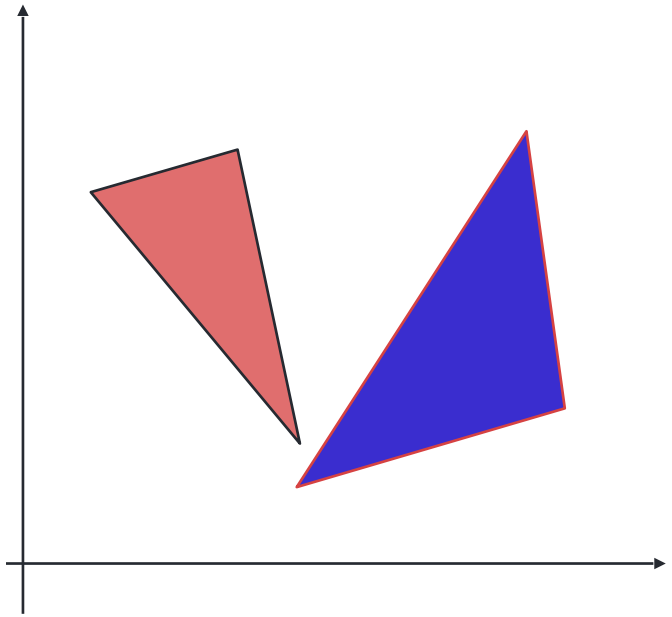
- Projektion auf die Bildebene
- Zerlegung in Pixel
- Pixel werden im Framebuffer abgelegt
- Letztes gerastertes Polygon besetzt Pixelpositionen im Framebuffer

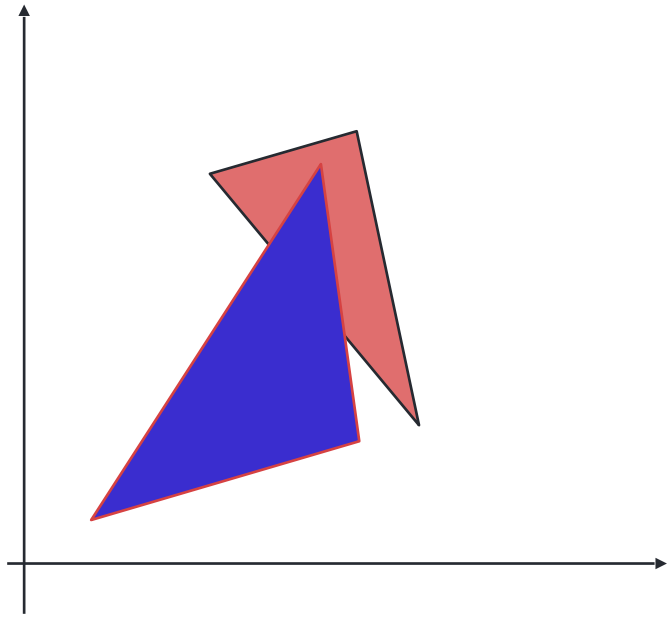


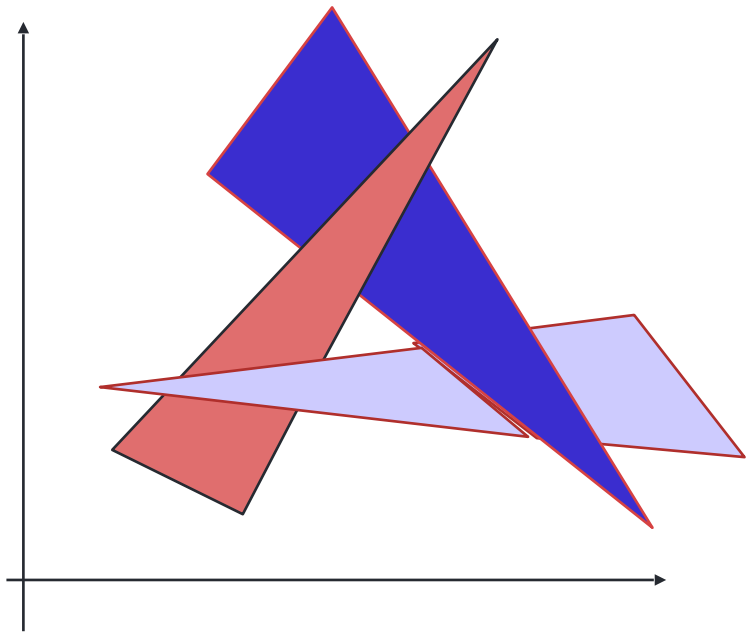
8.2 Projektion

Sichtbarkeit von Polygonen

- Welches Polygon müsste gesehen werden?
  Das dem Beobachter am nächsten Gelegene!
- Klare Aussage, welches das nächste ist, ist nicht immer gegeben.



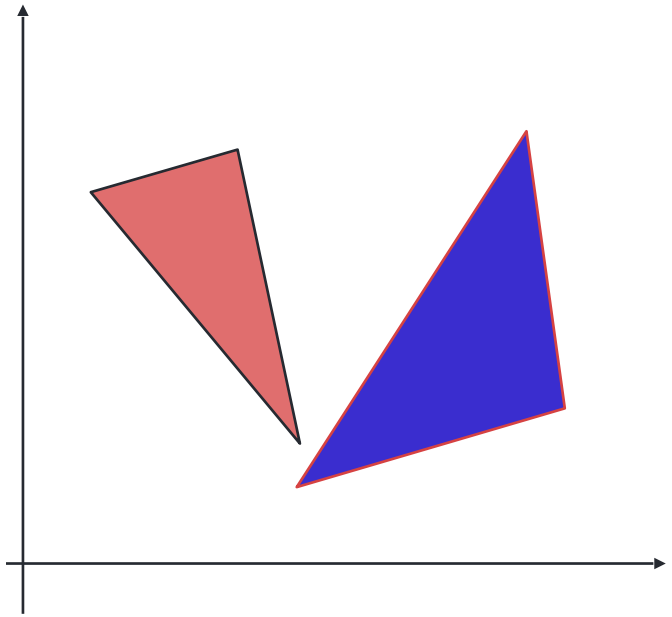


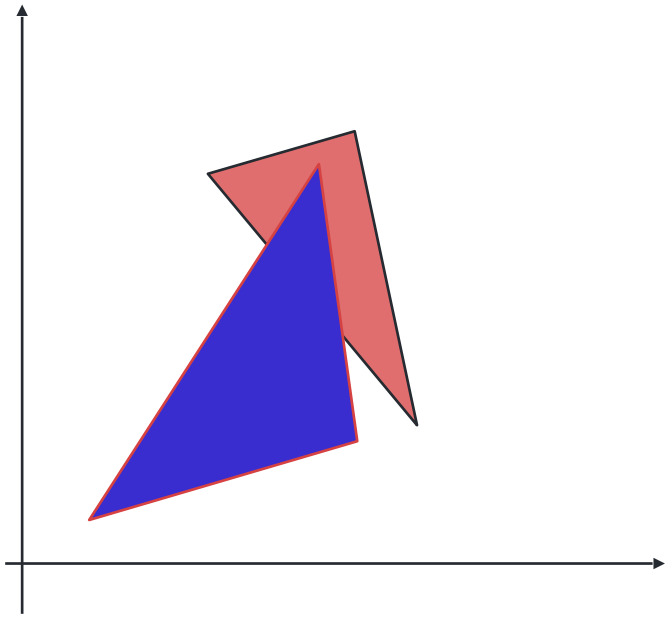


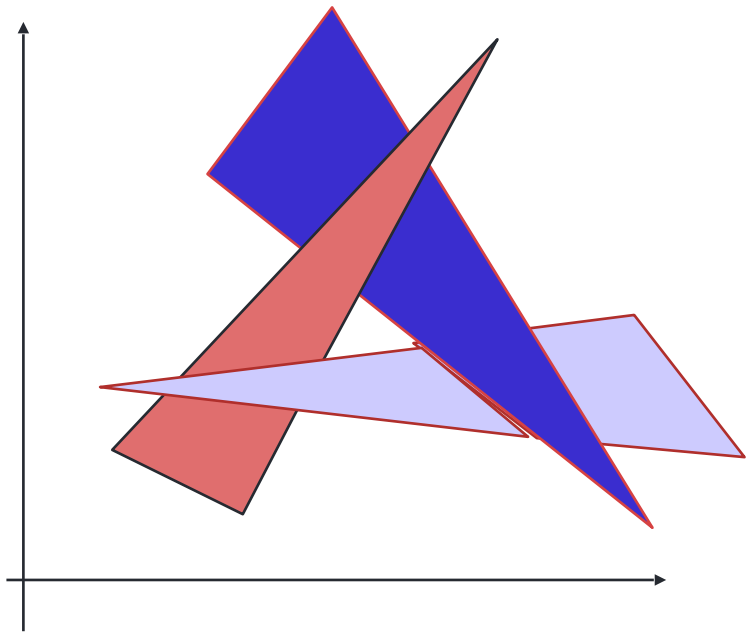
8.2 Projektion

Sichtbarkeit von Polygonen

- Sichtbarkeit: Sortierproblem
- Gleiches muss bei der Berechnung von Transparenz berücksichtigt werden







8.2 Projektion

Painter‘s Algorithmus

- Idee: Male Polygone von hinten nach vorne
- Erfordert Tiefensortierung!
- Wenn Tiefenwerte (z-Wert) der Polygone sich überlappen, müssen Polygone geschnitten werden
  - n^2 mögliche Teile!
  - Beginne mit dem (Teil-)Polygon mit größtem z-Wert
- Komplexität:
  - $O\left(n^2\right)$
  - n ist Anzahl der Dreiecke

8.2 Projektion

Erste Verfahren

- Erste Lösung des Hidden-Line-Problems:
  Roberts, 1963:
  Objektraumverfahren für konvexe Objekte

- Appels Algorithmus (1967):
  Berechnet sichtbare Kanten/ Konturen (NPR)

- Area Subdivision (divide-and conquer):
  Warnock, 1969: Ausnutzung von Flächenkohärenz durch Quadtrees

- Sample Spans:
  Watkins, 1970: Ausnutzung von Rasterzeilenkohärenz

8.2 Projektion

Erste Verfahren

- Depth List:
  Newell et al., 1972:
  Prioritätslistenalgorithmus im Objektraum

- Weiler-Atherton-Algorithmus (1977):
  Sortiert Polygone näherungsweise in der Tiefe

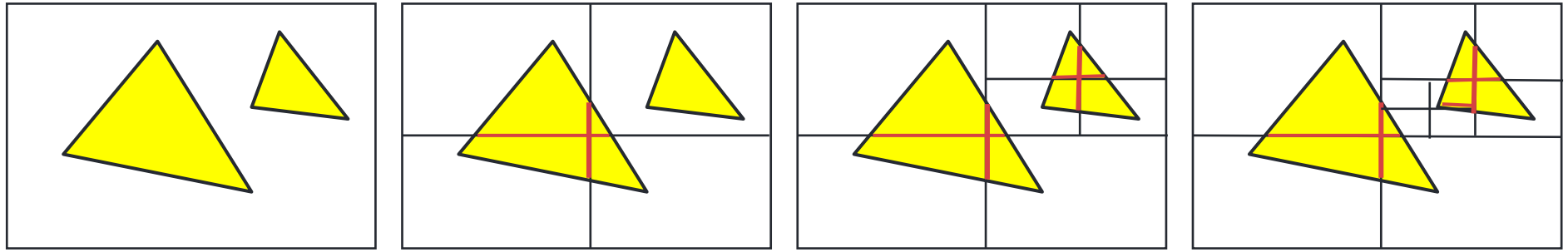
8.2 Projektion

Divide and Conquer [Warnock 1969]

- Einfache Fälle
  - Naheste Fläche überdeckt gesamten Rasterbereich
  - Es gibt maximal eine Fläche im Rasterbereich
- Ansonsten rekursive Unterteilung bis nur noch einfache Fälle auftreten
- Aufwand:
  - $O(n \cdot p)$
  - p: Anzahl der Pixel
  - n: Anzahl der Polygone
- Ggf. muss Unterteilung bis auf Pixelebene durchgeführt werden
  - Fast immer bei $n > p$ (große Szenen)
  - Entspricht dann dem Z-Buffer Algorithmus
    - aber mit Overhead

8.2 Projektion

Divide and Conquer [Warnock 1969]



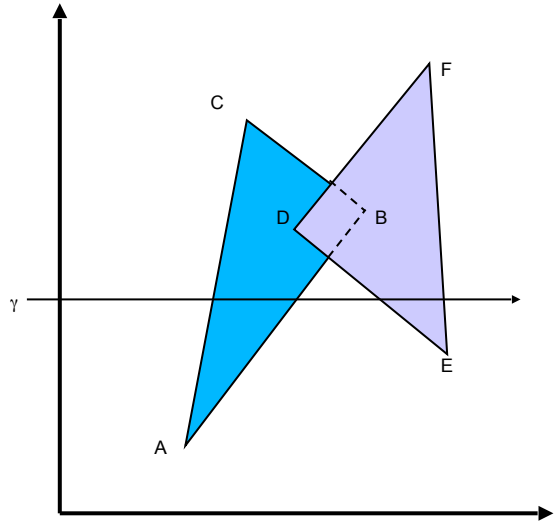
8.2 Projektion

Scanline/Sample Span [Watkins 1970]

- Suche nach Kanten entlang der y-Scanline γ
- Sortiere Kanten nach Tiefenwerten
- Datenstruktur
  - Kantentabellen
  - Polygontabellen
  - ActiveEdge-Tabellen (AET)
- Aktualisierung der AET für jede neue Scanline γ

- Scanline γ in AET:
  - P_1, P_2 Polygone

AC	AB	DE	FE
Ein	Aus	Ein	Aus



8.2 Projektion

Z-Buffer-Algorithmus

- [Straßer 1974, Catmull 1974]
- Bestimmt Sichtbarkeit von Pixeln (Bildraum)
- Geeignet für die Bildausgabe auf Rastergeräten
- Einfache Hardware-Unterstützung
- Arbeitsweise
  - Sucht für jeden Pixel bei Rasterung nach Polygon mit kleinstem (am weitesten vorne liegendem) z-Wert
  - Zusätzlicher Speicher
    - Z-Buffer
    - Speichere in jedem Pixel den bisher kleinsten aufgetretenen z-Wert

8.2 Projektion

Z-Buffer-Algorithmus

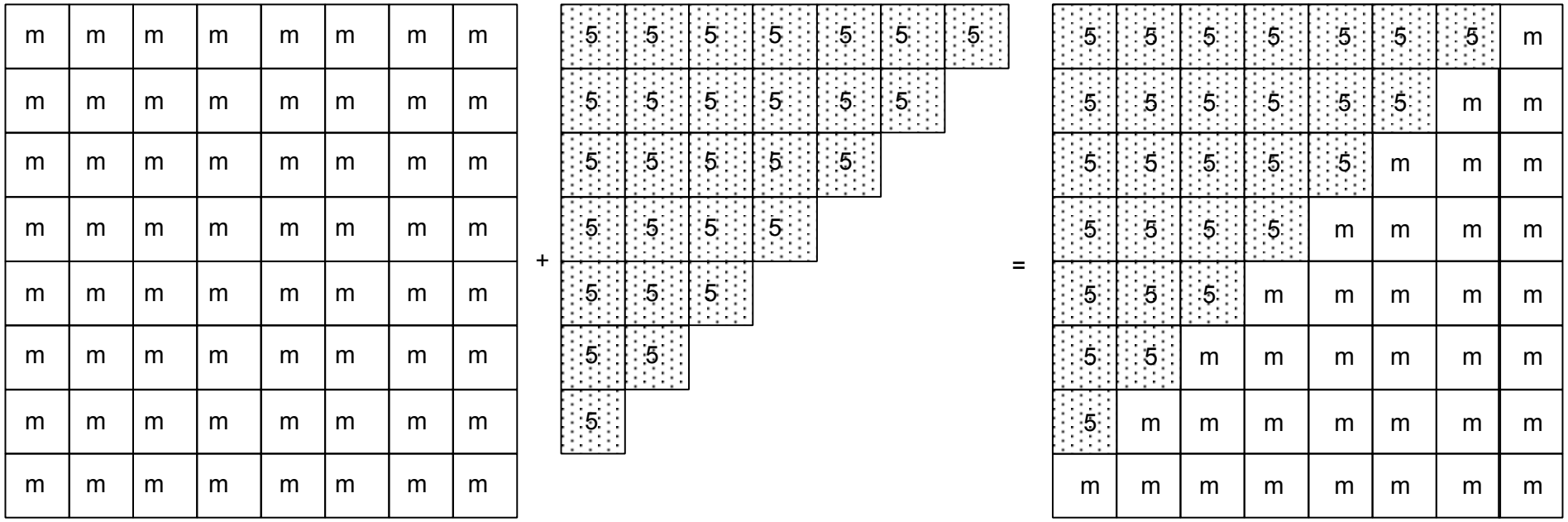
- Initialisiere Framebuffer (Farbbuffer) mit Hintergrundfarbe
- Initialisiere Z-Buffer mit maximalem z-Wert
- Scan-Conversion aller Polygone in beliebiger Reihenfolge
  - Berechne z-Wert für jedes Pixel (x, y) im Polygon
  - $z < z_{Buffer}$:
    - zeichne Polygonfarbe in Farbbuffer bei (x, y) ein
    - setze $z_{Buffer} = z$
- Am Ende enthält
  - der Farbbuffer das gewünschte Bild
  - der z-Buffer dessen Tiefenverteilung

8.2 Projektion

Z-Buffer-Algorithmus

- z-Werte codiert durch Zahlen: kleinere Zahl → näher am Auge

- Initialisiere Z-Buffer mit z-Wert m (ganz hinten)
- Addiere ein Polygon mit konstantem z-Wert 5



8.2 Projektion

Z-Buffer-Algorithmus

- Addiere ein Polygon, welches das 1. Polygon schneidet
- Artefakte bei Pixeln mit gleichem z-Wert beider Polygone

5	5	5	5	5	5	5	m
5	5	5	5	5	5	m	m
5	5	5	5	5	m	m	m
5	5	5	5	m	m	m	m
5	5	5	m	m	m	m	m
5	5	m	m	m	m	m	m
5	m	m	m	m	m	m	m
m	m	m	m	m	m	m	m

+

8					
7	8				
6	7	8			
5	6	7	8		
4	5	6	7	8	
3	4	5	6	7	8

=

5	5	5	5	5	5	5	m
5	5	5	5	5	5	m	m
5	5	5	5	5	m	m	m
5	5	5	5	m	m	m	m
5	5	5	8	m	m	m	m
4	5	6	7	8	m	m	m
3	4	5	6	7	8	m	m
m	m	m	m	m	m	m	m

8.2 Projektion

Z-Buffer-Algorithmus

- Zur Berechnung von $z(x, y)$ für ebene Polygone (z.B. Dreiecke) entlang einer Scan-Line
  - Ebene: $Ax + By + Cz + D = 0$
  - $z(x, y) = -\dfrac{D + Ax + By}{C}$
  - $$\begin{aligned} z(x + \Delta x, y) &= -\frac{D + A(x + \Delta x) + By}{C} \\ &= z(x, y) - \Delta x \cdot \frac{A}{C} \end{aligned}$$
- Nur eine Subtraktion ist notwendig
  - $\dfrac{A}{C}$ ist konstant
  - $\Delta x = 1$

8.2 Projektion

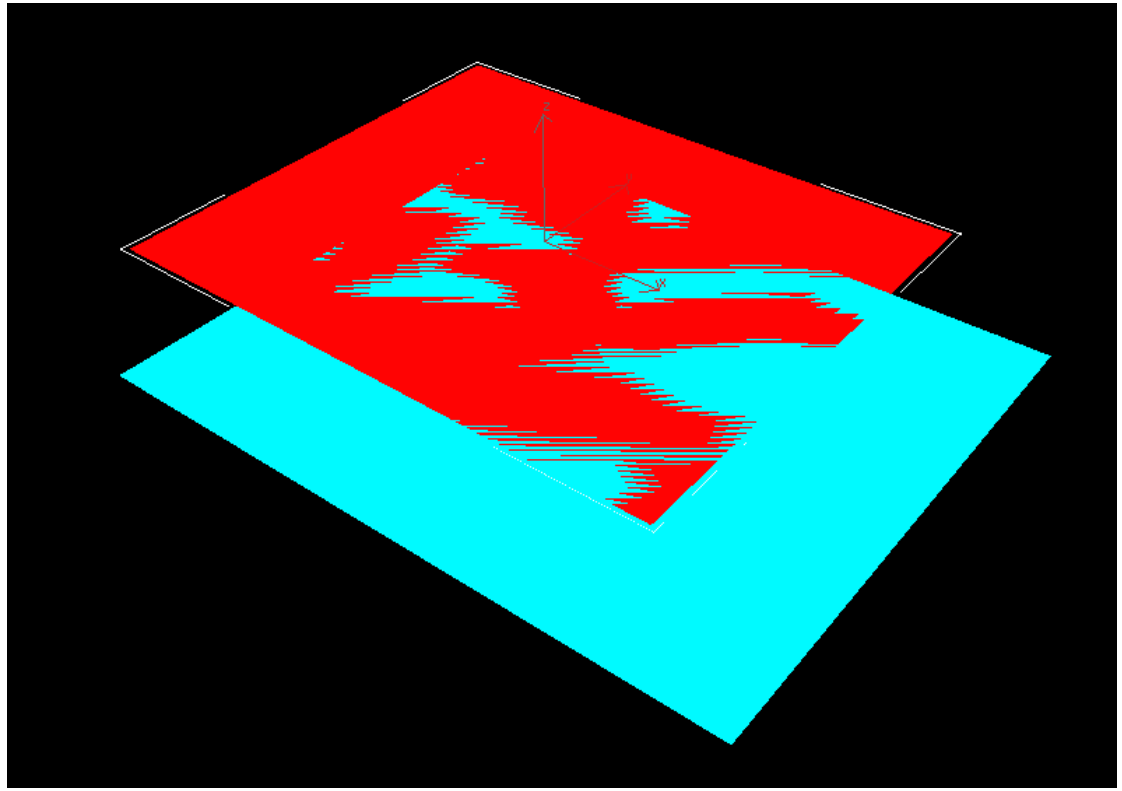
Z-Buffer-Algorithmus: Vorteile

- sehr einfache Implementierung des Algorithmus
- keine besondere Reihenfolge oder Sortierung notwendig
- keine Komplexitätsbeschränkung der Bildszene
- unabhängig von der Repräsentation der Objekte; es muss nur möglich sein, zu jedem Punkt der Oberfläche einen z-Wert zu bestimmen
- Auflösung des Z-Buffers bestimmt Diskretisierung der Bildtiefe
  - 20 Bit $\rightarrow 2^{20}$ Tiefenwerte unterscheidbar

8.2 Projektion

Z-Buffer-Algorithmus: Nachteile

- problematisch sind weit entfernte Objekte mit kleinen Details (perspektivische Transformation)
- Aufwändige Modifikationen notwendig für
  - Transparenz (Alpha-Buffering)
  - Antialiasing

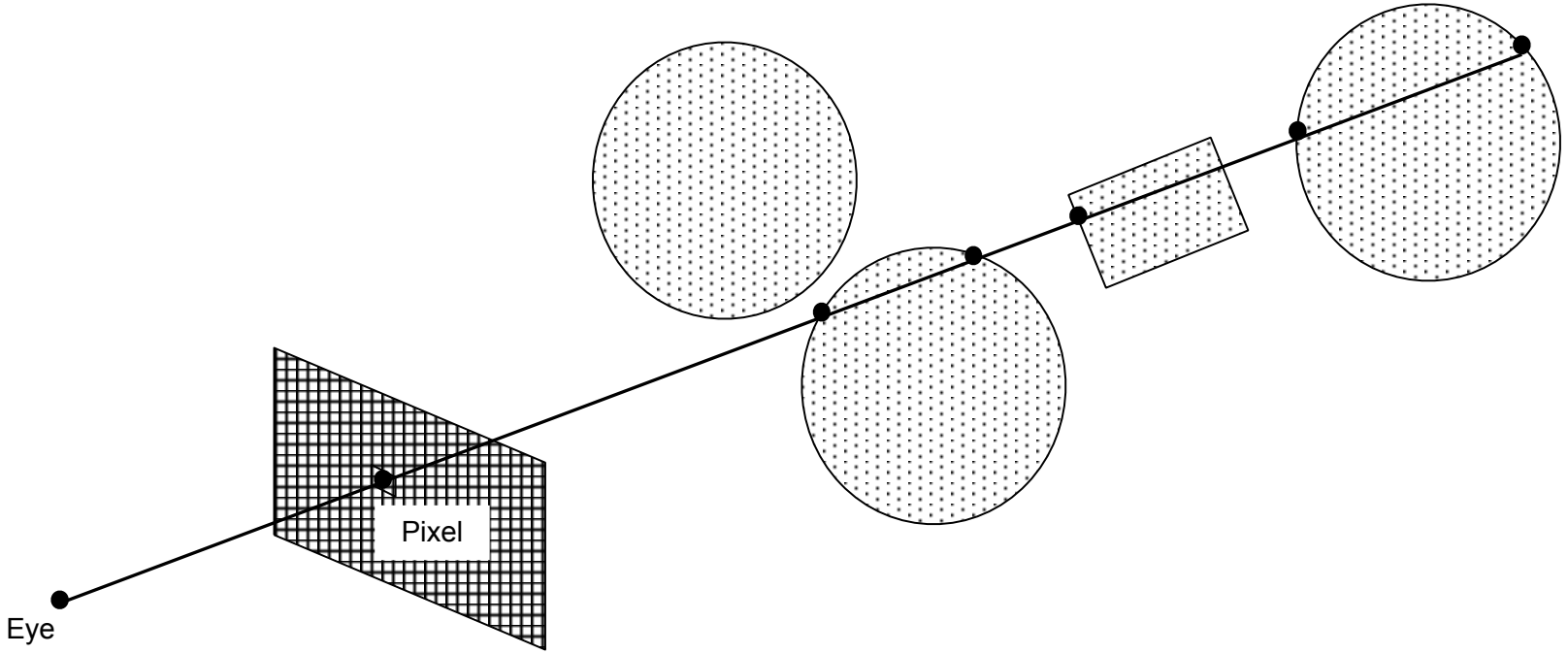


8.3 Strahlverfolgung

- RayCasting
  - Sendet (to cast) Strahlen (ray)
    - vom Auge
    - durch alle Pixel der Bildebene
    - in den Objektraum
  - Polygon opak
    - addiere Farbwert hinzu
    - stopp
  - Polygon transparent
    - addiere Farbwert hinzu
    - stopp
- RayTracing
  - RayCasting
  - Farbwert
    - Wie RayCasting
    - Verfolge reflektierte Strahlen
    - Verfolge gebrochene Strahlen
  - globales Beleuchtungsmodell

8.3 Strahlverfolgung

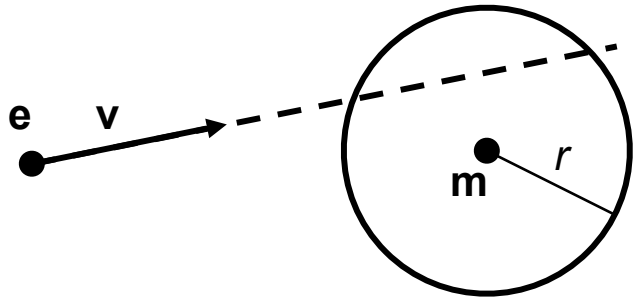
- Berechne Schnittpunkte mit allen Objekten der Szene
- Objekt mit dem nächst gelegenem Schnittpunkt ist in diesem Pixel sichtbar



8.3 Strahlverfolgung

Schnitt des Strahls mit einer impliziten Kugel

- Strahl
  - e: Augpunkt
  - v: Sichtrichtung (Pixel $-$ e)
  - t: Strahlparameter
  - $r(t) = e + t \cdot v$
- Kugel
  - $\|x - m\|^2 - r^2 = 0$

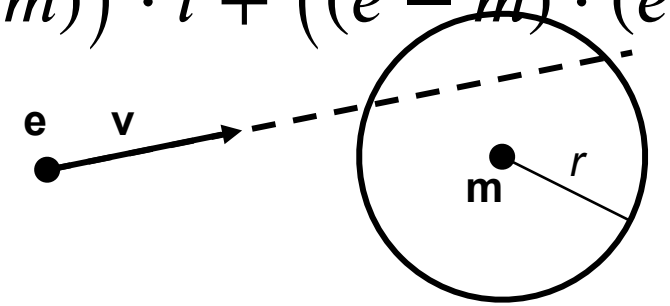


8.3 Strahlverfolgung

Schnitt des Strahls mit einer impliziten Kugel

- Einsetzen des Strahls $r(t)$ für x liefert:

$$\begin{aligned}
\left\| e + t \cdot v - m \right\|^2 - r^2 &= 0 \\
\left\| t \cdot v + (e - m) \right\|^2 - r^2 &= 0 \\
t^2 \cdot v \cdot v + 2 \cdot t \cdot v \cdot (e - m) + (e - m) \cdot (e - m) - r^2 &= 0 \\
(v \cdot v) \cdot t^2 + \big(2 \cdot v \cdot (e - m)\big) \cdot t + \big((e - m) \cdot (e - m) - r^2\big) &= 0
\end{aligned}$$

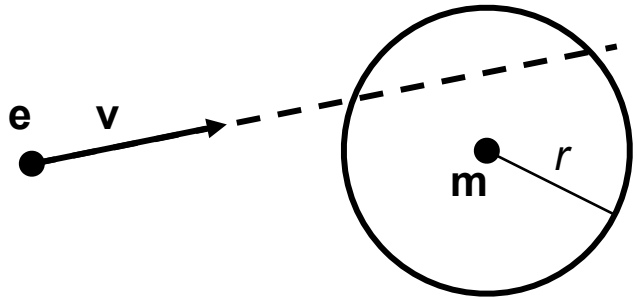


8.3 Strahlverfolgung

Schnitt des Strahls mit einer impliziten Kugel

- Lösen der quadratischen Gleichung nach t liefert die Parameter von maximal 2 Schnittpunkten
  $s_{1,2} = r(t_{1,2}) = e + t_{1,2} \cdot v$

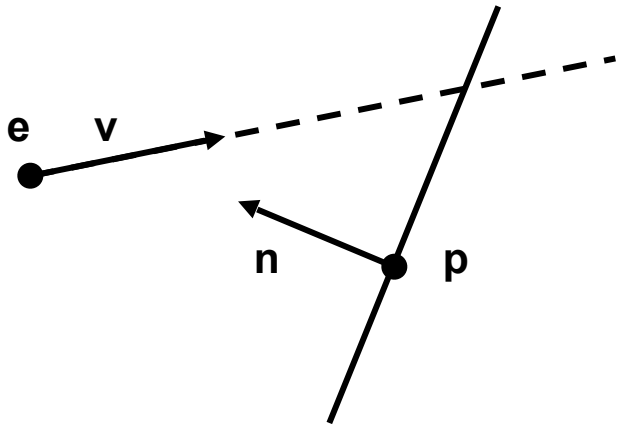
- Schnittpunkt mit kleinstem $t > 0$ liegt dem Augpunkt am nächsten



8.3 Strahlverfolgung

Schnitt des Strahls mit einer Ebene

- Strahl
  - e: Augpunkt
  - v: Sichtrichtung (Pixel $-$ e)
  - t: Strahlparameter
  - $r(t) = e + t \cdot v$
- Ebene
  - p: Punkt der Ebene
  - n: Normalenvektor nach außen
  - $\left(x - p\right) \cdot n = 0$



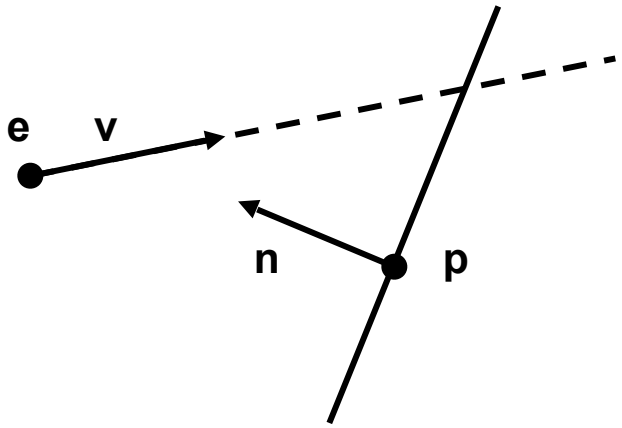
8.3 Strahlverfolgung

Schnitt des Strahls mit einer Ebene

- Einsetzen des Strahls $r(t)$ für x liefert:

$$\begin{aligned} \left(e + t \cdot v - p\right) \cdot n &= 0 \\ t \cdot v \cdot n + \left(e - p\right) \cdot n &= 0 \\ t &= \frac{\left(p - e\right) \cdot n}{v \cdot n} \end{aligned}$$

- Schnittpunkt
  $s = r(t) = e + t \cdot v$



8.3 Strahlverfolgung

Schnitt des Strahls mit einer Ebene

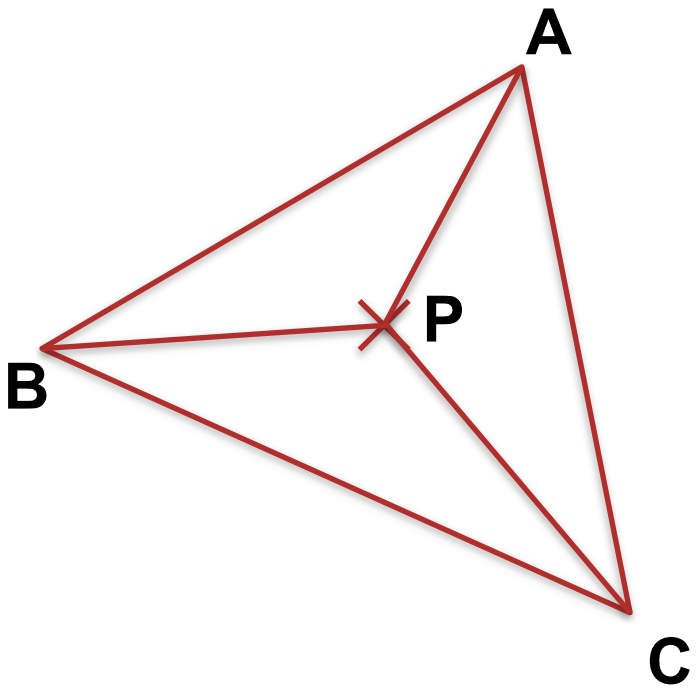
- Beim Schneiden von Polygonen ist noch die Gültigkeit des Schnittpunktes zu verifizieren
- Dieser muss innerhalb des Polygons liegen

- Dreiecke
  - Bestimme Summe der Flächeninhalte der Teildreiecke
    - Ist diese Summe größer als der Flächeninhalt des ursprünglichen Dreiecks, so liegt der Punkt außerhalb
    - Achtung: Rundungsfehler!
  - Alternativ
    Bestimme die baryzentrischen Koordinaten des Dreiecks (a, b, c)

8.3 Strahlverfolgung

Baryzentrische Koordinaten

- Gegeben: Dreieck $(A,\ B,\ C)$
- Gesucht: Koordinaten von P bezüglich des Dreiecks $(A,\ B,\ C)$
- Ansatz:
  $P = \alpha \cdot A + \beta \cdot B + \gamma \cdot C$
- Nebenbedingung:
  $\alpha + \beta + \gamma = 1$
  (Normalisierung)



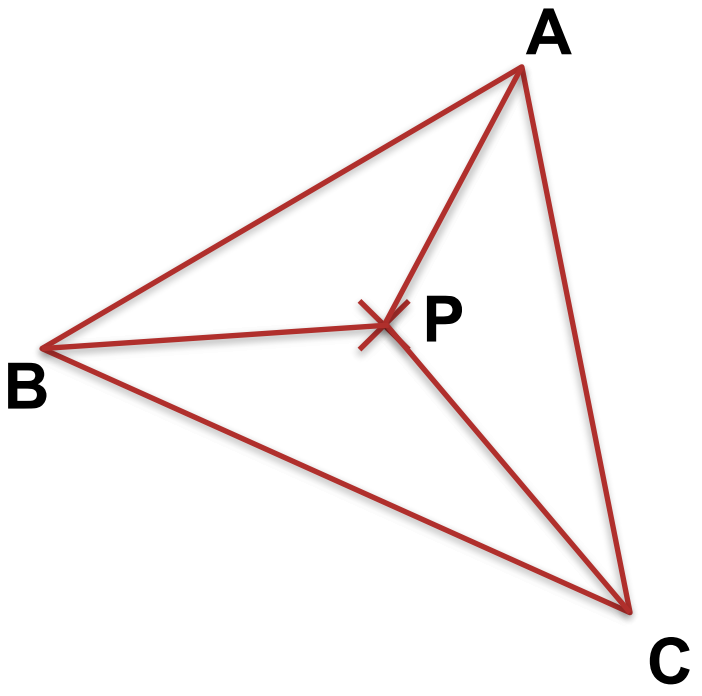
8.3 Strahlverfolgung

Baryzentrische Koordinaten

- $P = \alpha \cdot A + \beta \cdot B + \gamma \cdot C$
- Folgerungen:

$$A = (1,0,0)$$
$$B = (0,1,0)$$
$$C = (0,0,1)$$

  - $\alpha = 0,\ \beta + \gamma = 1$: Kante $B - C$
  - $\beta = 0, \alpha + \gamma = 1$: Kante $A - C$
  - $\gamma = 0, \alpha + \beta = 1$: Kante $A - B$
  - $0 \leq \alpha, \beta, \gamma \leq 1$: P liegt innerhalb
  - Sonst: P liegt außerhalb



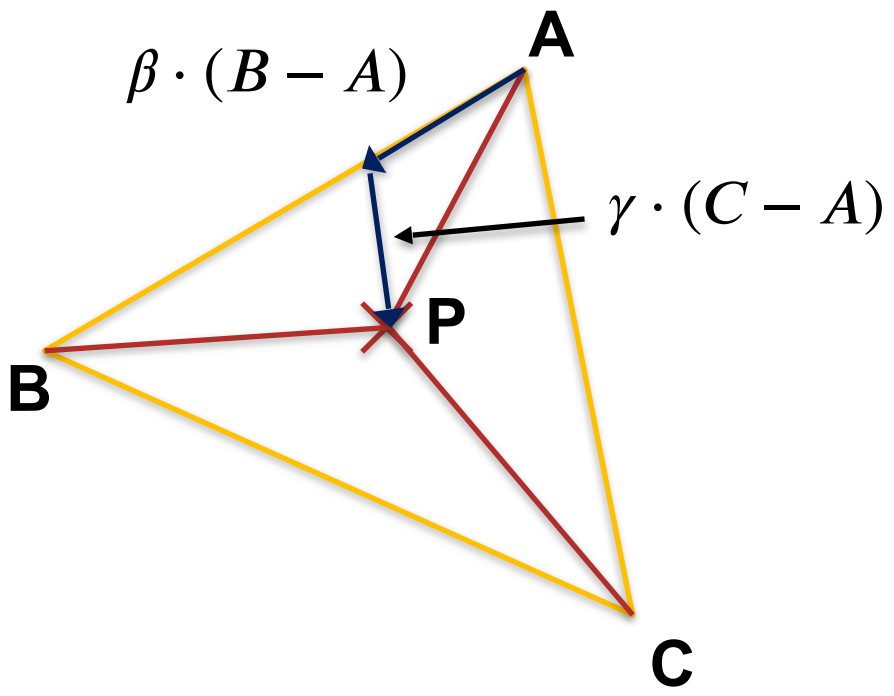
8.3 Strahlverfolgung

Baryzentrische Koordinaten

- $P = \alpha \cdot A + \beta \cdot B + \gamma \cdot C$

$$\begin{aligned} P &= \alpha \cdot A + \beta \cdot B + \gamma \cdot C \\ &= \left(1 - \beta - \gamma\right) \cdot A + \beta \cdot B + \gamma \cdot C \\ &= A + \beta \cdot (B - A) + \gamma \cdot (C - A) \end{aligned}$$

- Berechnung von $\left(\alpha, \beta, \gamma\right)$



$$\alpha = \frac{\Delta(P, B, C)}{\Delta(A, B, C)}$$

$$\Delta(P, B, C) = \left((B - P) \times (C - P)\right) \circ \left(\frac{(B - A) \times (C - A)}{\left\|(B - A) \times (C - A)\right\|} \right)$$

Flächeninhalt des Parallelogramms

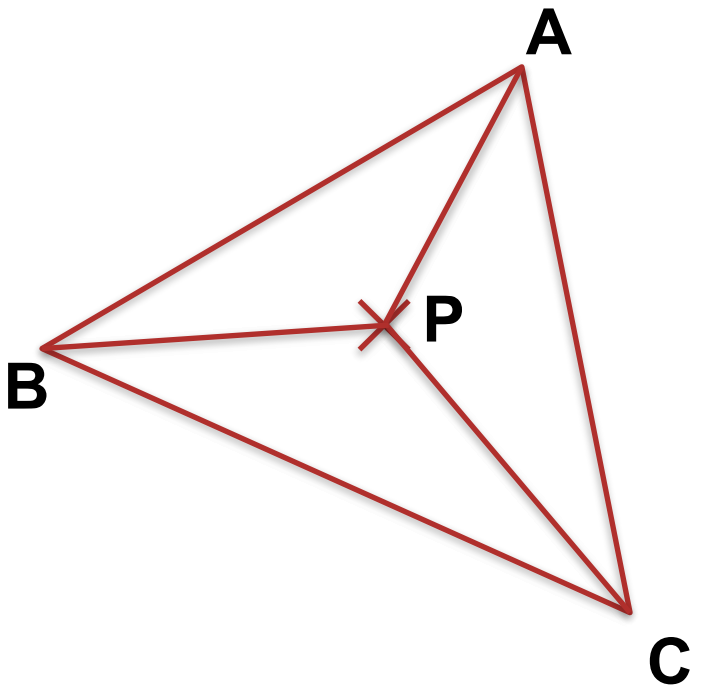
Einheitsvektor kodiert Orientierung als Vorzeichen

Δ entspricht nicht dem Flächeninhalt des Dreiecks, sondern dem doppelten orientierten Flächeninhalt

8.3 Strahlverfolgung

Baryzentrische Koordinaten

- Effizientere Berechnung und Algorithmen:
  Christer Ericson.
  Real Time Collision Detection.
  Morgan Kaufman – Elsevier, 2005.



8.3 Strahlverfolgung

- Nachteile
  - 1 Strahl pro Pixel
    - Full HD: ca. 2 Millionen Strahlen
  - Schnitttests
    - jeder Strahl
    - jedes Objekt
  - Für typische Szenen:
    - bis zu 95% der Rechenzeit
- Beschleunigung
  - Vermeidung von unnötigen Schnitttests
    - Hierarchien von Bounding Volumes
      - Bounding Spheres
      - Bounding Boxes
      - Axis-Aligned Bounding Boxes
    - Raumteilung
    - Occlusion Culling

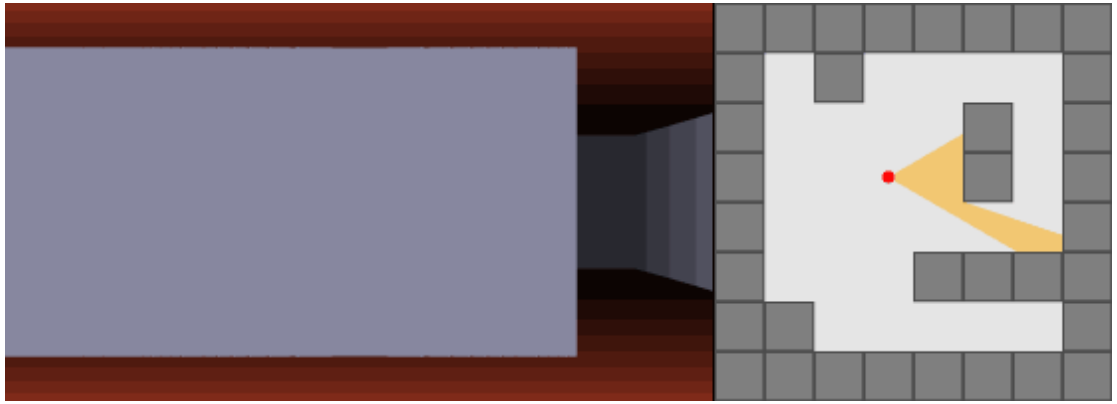
8.3 Strahlverfolgung

Hierarchien von Bounding Volumes

- Baumartige Strukturen von Bounding Volumes
  - Blätter: Objekte der Szene (Geometrie)
  - Innere Knoten: Bounding Volumes um die Objekte der Unterbäume
- Schneidet ein Strahl das Bounding Volume eines inneren Knotens nicht, so entfällt ein Test der untergeordneten Teilbäume
- Generierung guter Hierarchien ist schwierig
  - Geometrisch:
    Nach Szenenausdehnung unterteilen
  - Dichte:
    Nach Polygonschwerpunkten sortieren und unterteilen
  - Verwendung des Szenengraphen möglich

8.3 Strahlverfolgung

- Raycasting kann auch zur Darstellung von 3D Darstellungen genutzt werden
- 2D Blöcke mit Objekte werden mit dem Boden und Decke dargestellt
- Szene wird dann aus der Position des Betrachters berechnet



https://commons.wikimedia.org/wiki/File:Simple_raycasting_with_fisheye_correction.gif

8.3 Strahlverfolgung

- Pro Bildspalte wird ein Strahl berechnet
- Die Textur des getroffenen Objektes wird dann spaltenweise in die Szene eingefügt
  - Basierend auf deren Position
  - Anteil Boden und Decke wird addiert
  - -> Performancevorteile gegenüber „echtem“ 3D

https://upload.wikimedia.org/wikipedia/it/5/55/Wolfenstein_3D_Screenshot.png

8.4 Raumteilungsverfahren

- Space-Subdivision-Techniques
- Der Objektraum wird in disjunkte Teilräume zerlegt
- Für jedes Objekt wird bestimmt, in welchen Teilräumen es sich befindet
- Standardverfahren
  - Unterteilung des Raumes durch ein festes, regelmäßiges Gitter in Zellen identischer Geometrie
  - 2D: Pixel (Picture Element)
    - Quadrate
  - 3D: Voxel (Volume Element)
    - Würfel

8.4 Raumteilungsverfahren

- Vorteile
  - Es ist sehr einfach zu bestimmen, welche Objekte (teilweise) in einem Voxel liegen
  - Ist das Voxel leer wird keine Schnittberechnung durchgeführt
  - Sonst: teste nur Objekte die in diesem Voxel liegen
- Nachteile
  - Auflösung von n Voxeln in jeder Dimension: n^3 Voxel

  $\Rightarrow$ günstigere Repräsentation durch Octrees

8.4 Raumteilungsverfahren

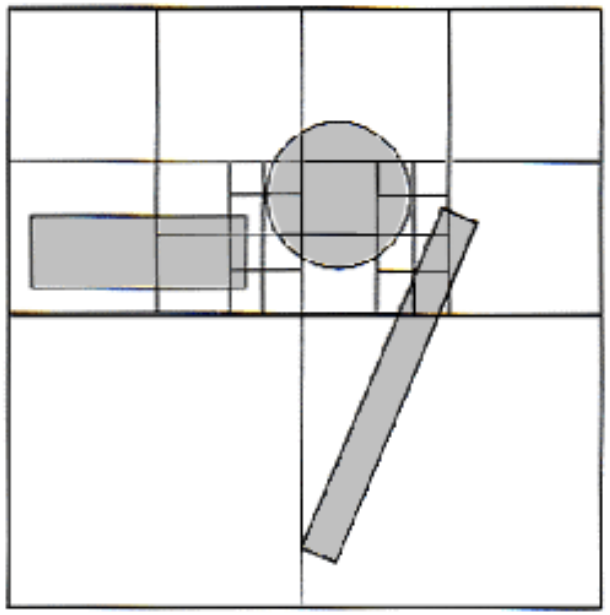
Quadtrees

- Erstes Auftreten: Ende der 1960er Jahren
- Octrees sind eine Erweiterung von Quadtrees
  - Ende 1970er, Anfang 1980er
- Unterteilung einer Ebene
- Erzeuge Quadrat, welches alle Objekte enthält (Wurzel)
- Rekursive Zerlegung
  - Enthält das Quadrat zu viele Objekte
    - Zerlege das Quadrat in vier Teile
    - Sortiere Objekte in Teilquadrate ein
  - Sonst → Stopp
- Jeder innere Knoten des Baumes besitzt genau vier Kinder

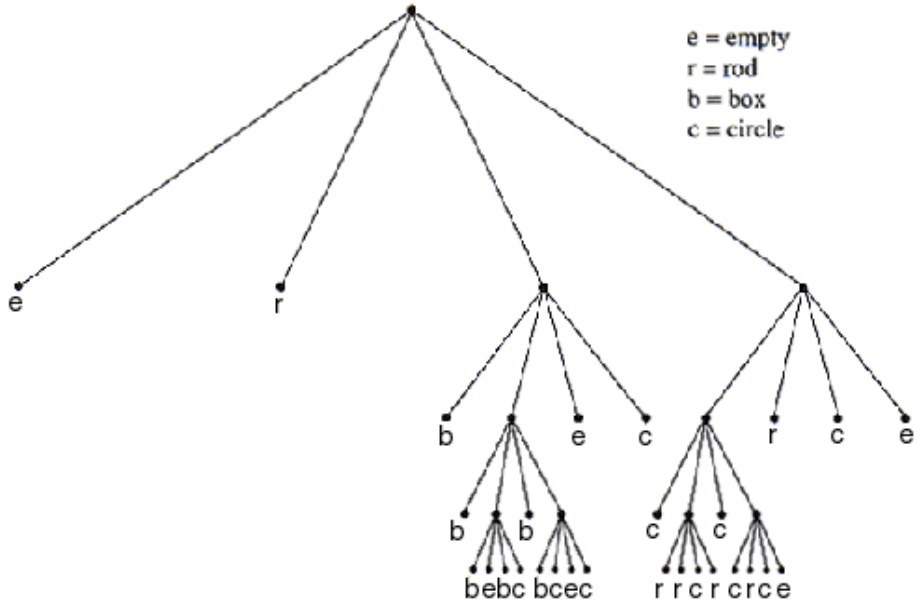
8.4 Raumteilungsverfahren

Quadtrees

- Unterteilung des Raumes, bis die Zellen maximal eine Objektreferenz enthalten



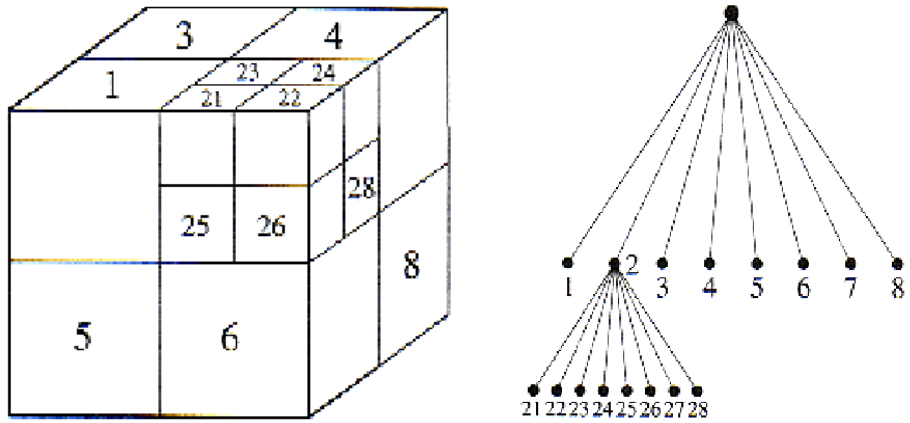
- Zugehörige Quadtree-Datenstruktur, vier Generationen



8.4 Raumteilungsverfahren

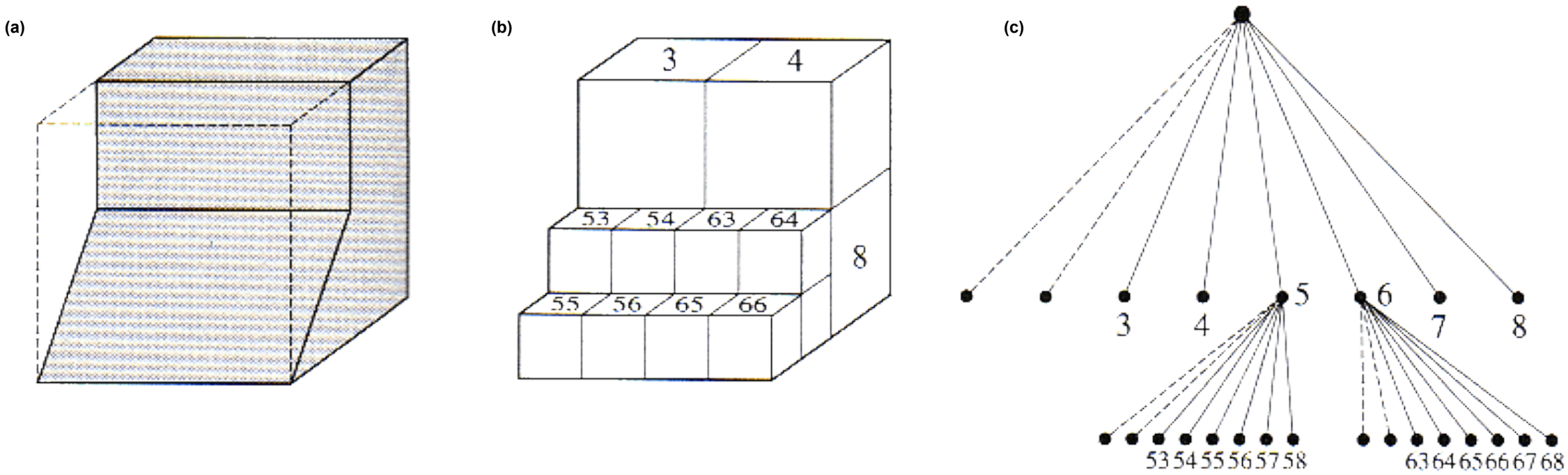
Octrees

- Unterteilung des 3D-Raums analog dem Quadtree
- Erzeuge Würfel, der alle Objekte enthält (Wurzel)
- Rekursive Zerlegung
  - Enthält der Würfel zu viele Objekte
    - Zerlege den Würfel in acht Teile
    - Sortiere Objekte in Teilwürfel ein
  - Sonst → Stopp
- Jeder innere Knoten des Baumes besitzt genau acht Kinder



8.4 Raumteilungsverfahren

Octrees



8.4 Raumteilungsverfahren

n-ary trees

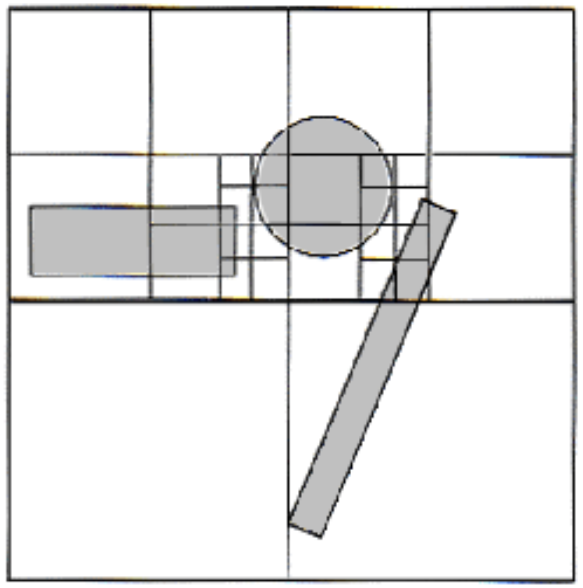
- Verallgemeinerung auf n Dimensionen
- Muss keine feste Unterteilungsstrategie repräsentieren

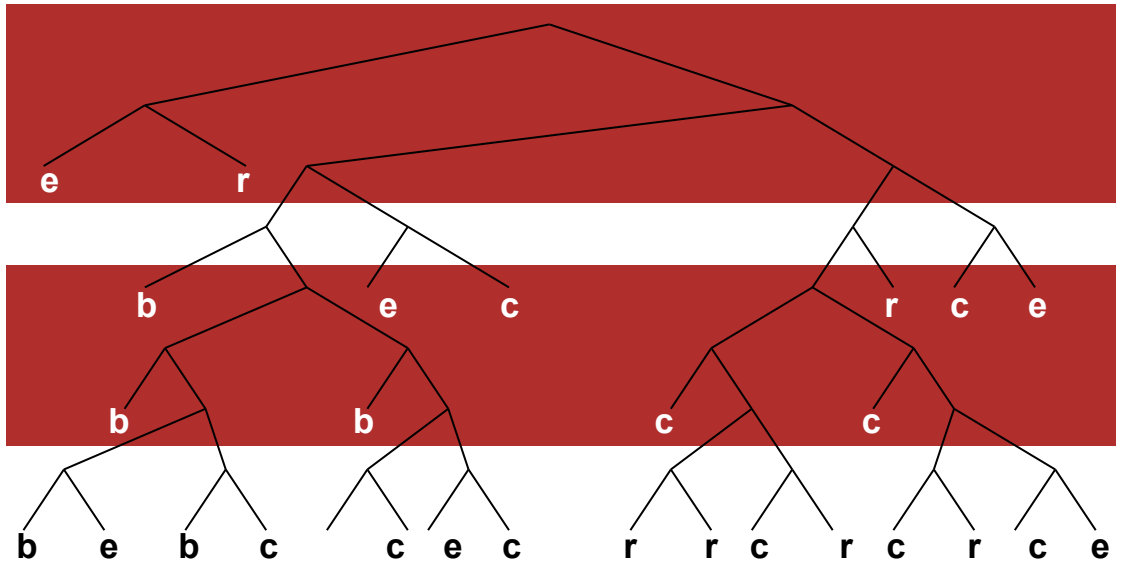
- Für den Fall $n = 1$
  - kD-Tree
  - Elternknoten hat bis zu zwei Kinder
- Für den Fall $n = 2$
  - Quadtree
- Für den Fall $n = 3$
  - Octree

8.4 Raumteilungsverfahren

kD-Tree

- Unterteilung des Raumes, bis die Zellen maximal eine Objektreferenz enthalten
- zugehörige kD-Tree-Datenstruktur
  - acht Generationen



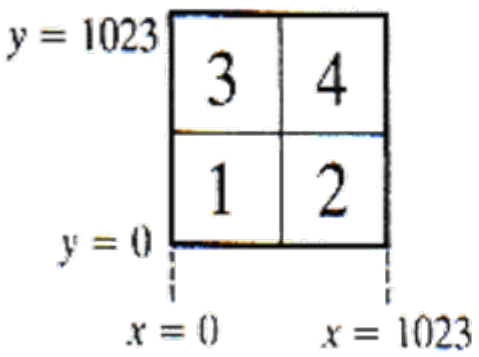


e = empty
r = rod
b = box
c = circle

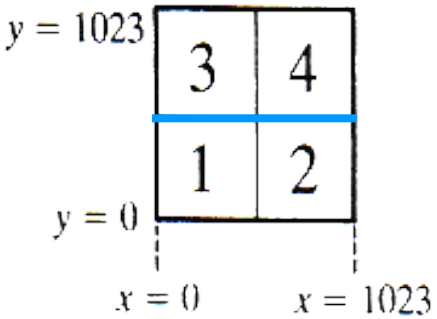
8.4 Raumteilungsverfahren

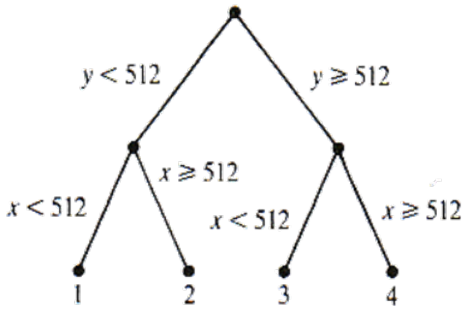
Unterteilung einer 2D-Szene

- Quadtree



- kD-Tree

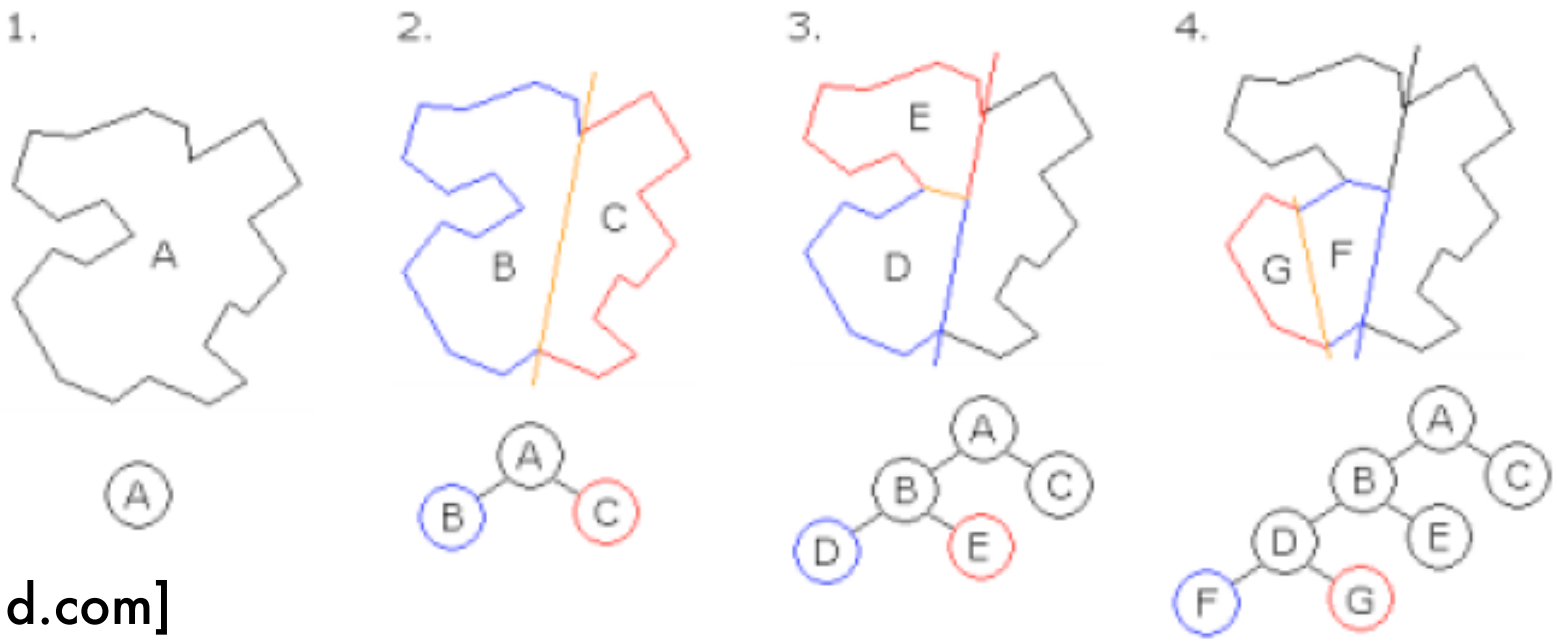




8.4 Raumteilungsverfahren

Binary Space-Partitioning Trees (BSP-Bäume)

- Octrees, Quadtrees und kD-Trees
  - Achsen-parallele Unterteilung
  - gleichzeitig oder abwechselnd
- BSP-Trees
  - Unterteilung durch beliebige (Hyper-)Ebene in zwei Unterräume



[www.beyond3d.com]

8.4 Raumteilungsverfahren

Binary Space-Partitioning Trees (BSP-Bäume)

- Können zur Unterteilung von Szenen verwendet werden
- Sind an kein Raster gebunden
- Über die relative Lage der Regionen zum Betrachter kann die Tiefenstaffelung der Objekte bestimmt werden
  - Welche Objekte können sichtbar sein?
- Eine ideale Wahl der Unterteilungsebene für BSP-Bäume liefert die PCA (Principal Component Analysis)
- Angenommen, eine komplexe Szene ist gegeben durch die Punktewolke $P_i \in \mathbb{R}^3, (i = 1, \ldots, n)$ (z.B. Objektmittelpunkte oder Eckpunkte der Polygone)
- PCA liefert ein orthogonales Koordinatensystem e_1, e_2, e_3 dessen Ausrichtung der Punktwolke entspricht

8.4 Raumteilungsverfahren

Binary Space-Partitioning Trees (BSP-Bäume)

- Wähle als Ursprung den Mittelwert

$$c = \frac{1}{n} \sum_{i=1}^{n} P_i$$

- Konstruiere Matrix der Punktwolke

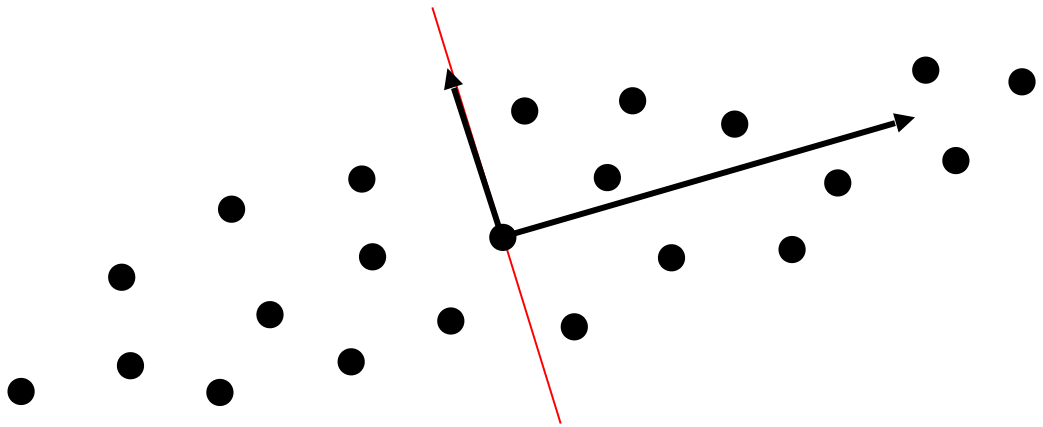
$$A = \begin{pmatrix} P_{1,x} - c_x & P_{1,y} - c_y & P_{1,z} - c_z \\ P_{2,x} - c_x & P_{2,y} - c_y & P_{2,z} - c_z \\ \vdots & \vdots & \vdots \\ P_{n,x} - c_x & P_{n,y} - c_y & P_{n,z} - c_z \end{pmatrix}$$

- Berechne
  - $B = \frac{1}{n-1} A^T A$
  - $b_{ij} = \frac{1}{n-1} \sum_{k=1}^{n} a_{ki} a_{kj}$
- B hat
  - Reelle Eigenwerte $\lambda_1, \lambda_2, \lambda_3$
  - Eigenvektoren e_1, e_2, e_3
  - $\lambda_i \cdot e_i = B \cdot e_i$

8.4 Raumteilungsverfahren

Binary Space-Partitioning Trees (BSP-Bäume)

- Die Eigenvektoren bilden zusammen mit dem Ursprung c das gesuchte System
- Die Ausdehnung der Punktwolke in Richtung e_i ist proportional zu $\sqrt{\lambda_i}$



8.4 Raumteilungsverfahren

Principal Component Analysis (PCA)

- Andere Namen
  - Karhunen-Loève Transformation (KLT, Bildverarbeitung)
  - Empirical orthogonal functions (Meteorologie, Geophysik)
  - Hotelling Transformation
  - Proper orthogonal decomposition (POD)
  - Deutsch: Hauptkomponentenanalyse
- Ziele
  - Berechne eine reduzierte Menge von Dimensionen
    - Orthogonal
    - Linear
  - Ordne die Dimensionen absteigend bezüglich der Varianz
- Varianz
  - Maß für den Informationsgehalt einer Variable
- Mathematisch
  - Suche eine neue Basis des Vektorraumes

8.4 Raumteilungsverfahren

Principal Component Analysis (PCA)

- Konstruktion
  - Gegeben: $X = (X_1, \ldots, X_m), X_j = \left(x_{1,j}, \ldots, x_{n,j}\right)^T$
  - Mittelwert: $\mu = (\mu_1, \ldots, \mu_m), \mu_j = \frac{1}{n} \cdot \sum_{i=1}^{n} x_{i,j}$
  - Zentrierte Werte: $Y = \left(y_{ij}\right), y_{ij} = x_{ij} - \mu_j$
  - Kovarianz-Matrix: $C = \frac{1}{n-1} \cdot Y^T Y$
    - Symmetrisch → spektrale Zerlegung möglich
  - Spektrale Zerlegung: $C = U \Lambda U^T, U^T U = I_m$
  - Eigenwert Matrix: $\Lambda = \begin{pmatrix} \lambda_1 & \ldots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \ldots & \lambda_m \end{pmatrix}$
  - Eigenvektor Matrix: $U = (u_1, \ldots, u_m)$
  - Ordne die Eigenwerte absteigend: $i > j \rightarrow \lambda_i > \lambda_j$

8.4 Raumteilungsverfahren

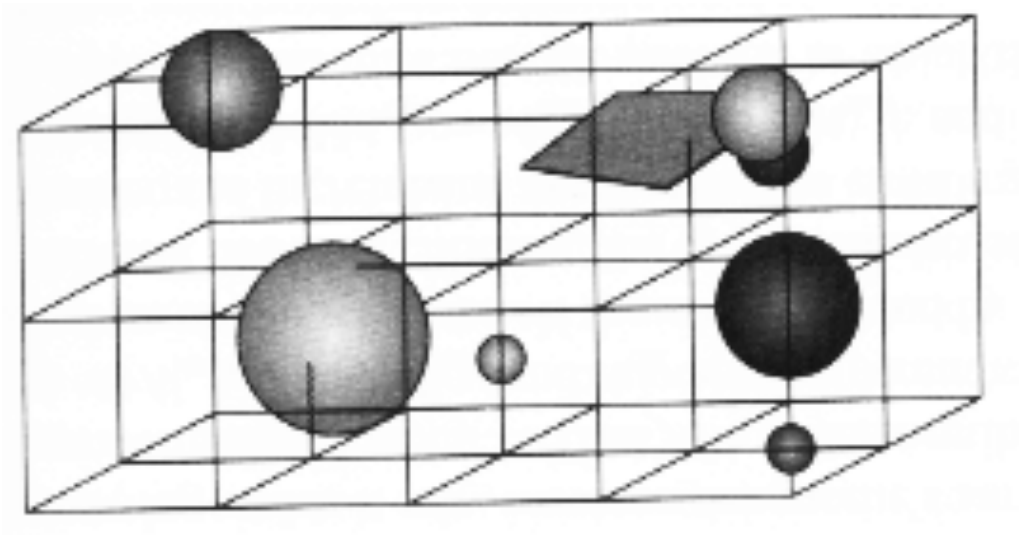
Principal Component Analysis (PCA)

- Eigenschaften:
  - $\lambda_1,\ u_1$: größte Varianz
  - $\lambda_2,\ u_2$: zweitgrößte Varianz
  - …
  - Die ersten p Eigenwerte beschreiben einen Großteil der Varianz
  - Principal components mit einer Varianz nahe 0 können verwendet werden, um Ausreißer zu identifizieren
- Wird auch häufig in der Scientific Visualization verwendet
- Einschränkungen:
  - Das Ergebnis ist abhängig von der Skalierung der einzelnen Variablen (Attribute)
  - Ausreißer haben einen großen Einfluss auf das Ergebnis
  - Lineare Methode
  - Interpretation der Basis (Ergebnisse)

8.4 Raumteilungsverfahren

Ray-Casting

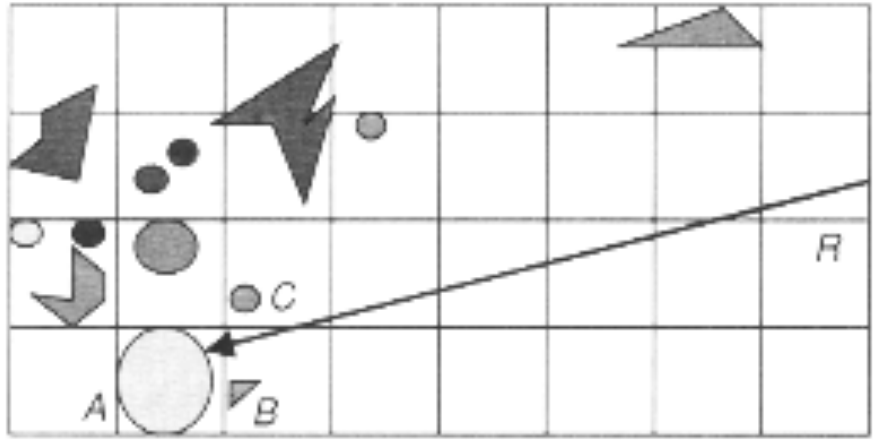
- Für das Raycasting wird der Raum zunächst unterteilt
- Dann werden die Objekte in die Teilräume einsortiert



8.4 Raumteilungsverfahren

Ray-Casting

- Der Strahl läuft zum ersten Teilraum
- Ist der Teilraum leer
  - keine Schnittberechnung notwendig
- Enthält der Teilraum Objekte
  - Schnitttest mit allen enthaltenen Objekten
  - Schnitt
    - Fertig
  - Kein Schnitt
    - Betrachte nächsten Teilraum in Richtung der Strahls



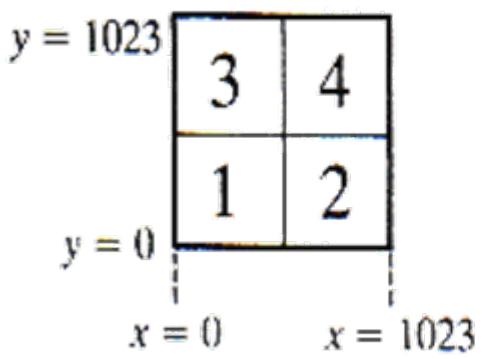
8.1 PROBLEMSTELLUNG

- Sichtbarkeitsberechnung
  - Ihr Ziel ist die (möglichst) exakte Bestimmung der von einem gegebenen Blickpunkt aus sichtbaren bzw. unsichtbaren Teile der darzustellenden Szene
- Anforderungen
  - Wünschenswert ist **hohe Interaktionsrate**, so dass Eingaben des Benutzers sich direkt auf die Darstellung auswirken
  - In den meisten Fällen ist die **Echtzeitausgabe** der Szene nötig
- Einteilung der Verfahren:
  - **Objektraumverfahren**
    - prinzipiell geräteunabhängig
    - Rechengenauigkeit ist die Maschinengenauigkeit
  - **Bildraumverfahren**
    - geräteabhängig
    - Rechengenauigkeit ist die Auflösung des Ausgabegerätes

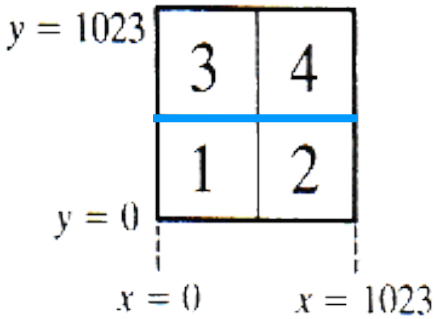
8.4 Raumteilungsverfahren

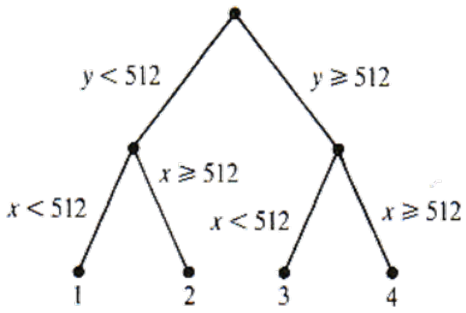
Unterteilung einer 2D-Szene

- Quadtree



- kD-Tree

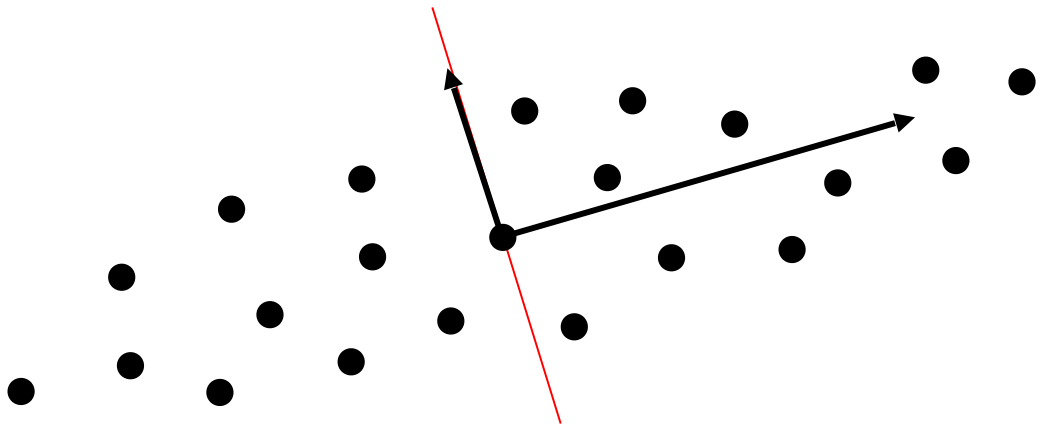




8.4 Raumteilungsverfahren

Binary Space-Partitioning Trees (BSP-Bäume)

- Die Eigenvektoren bilden zusammen mit dem Ursprung c das gesuchte System
- Die Ausdehnung der Punktwolke in Richtung e_i ist proportional zu $\sqrt{\lambda_i}$



8.5 Culling

- Begriff aus dem Englischen: Auslese, Ausduennen
- Entfernung nicht sichtbarer Objekte
- Große Szenen: Anzahl sichtbarer Objekte $\ll$ Anzahl aller Objekte
- Zielsetzung
  - Zeitersparnis bei der Schnittpunktberechnung
  - Entfernung offensichtlich nicht sichtbarer Objekte

8.5 Culling

- heuristische, grobe Antwort
- Test muss erheblich „billiger“ sein als Test der Sichtbarkeit
- Objektraum- und Bildraumverfahren
- Bounding Volumes

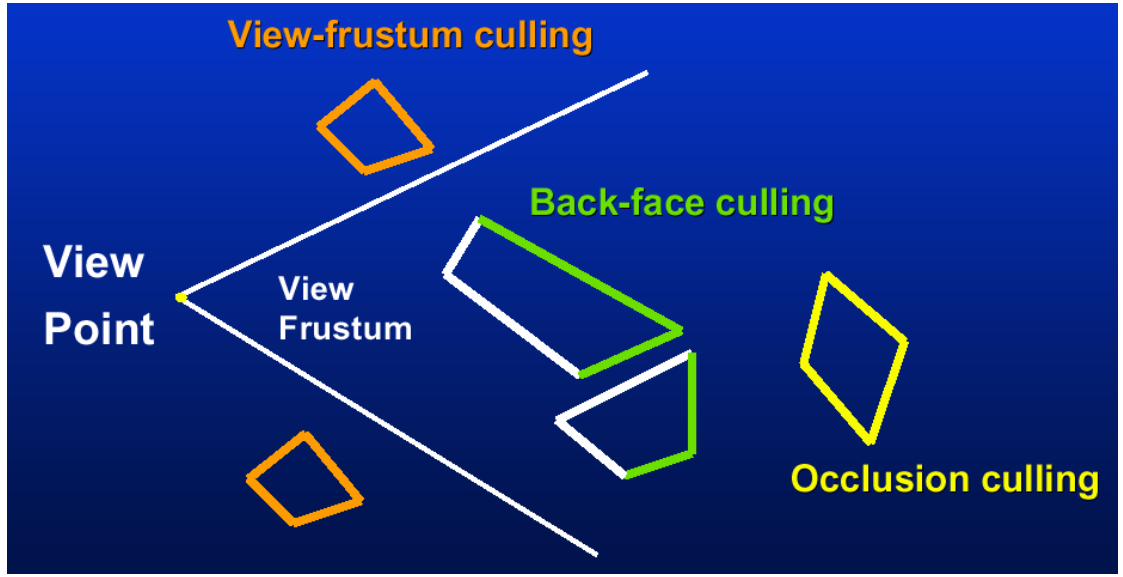
8.5 Culling

Arten

- Backface-Culling
  Normale zeigt nach hinten
  (wir sehen die Rückseite)

- View-Frustum-Culling
  Das Objekt liegt außerhalb des Blickfeldes

- Occlusion-Culling
  Das Objekt liegt hinter einem anderem Objekt

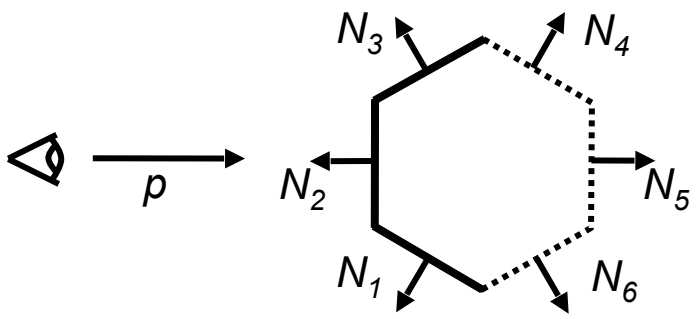


8.5 Culling

Backface-Culling

- Rückseiten von undurchsichtigen Objekten nicht sichtbar
  - Orientierung über Normalen kodiert: konsistente Berechnung wichtig
  - Bei Inkonsistenzen: Löcher in Objekten
- Wird von OpenGL unterstützt
- Sehr einfache Operation

- Klassifikation der Rückseiten
  - Normalenvektoren N_i aller Flächen
  - Normalenvektor N_i zeigt in Blickrichtung: Rückseite
    - Berechne Skalarprodukt aus Blickrichtungsvektor p und Normale N_i
    - Verwirf, wenn $p \cdot N_i > 0$



8.5 Culling

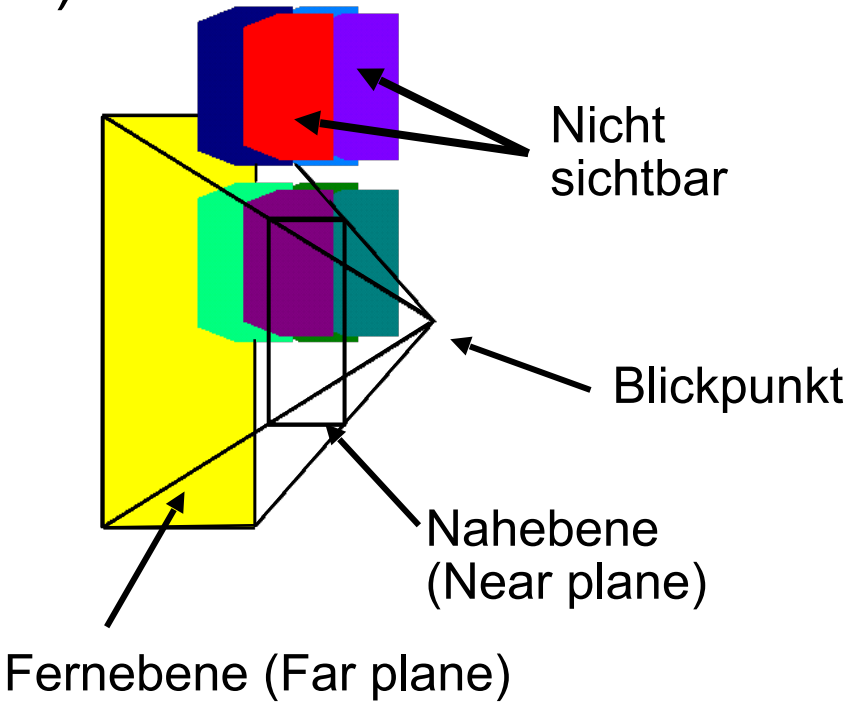
Backface-Culling: Eigenschaften

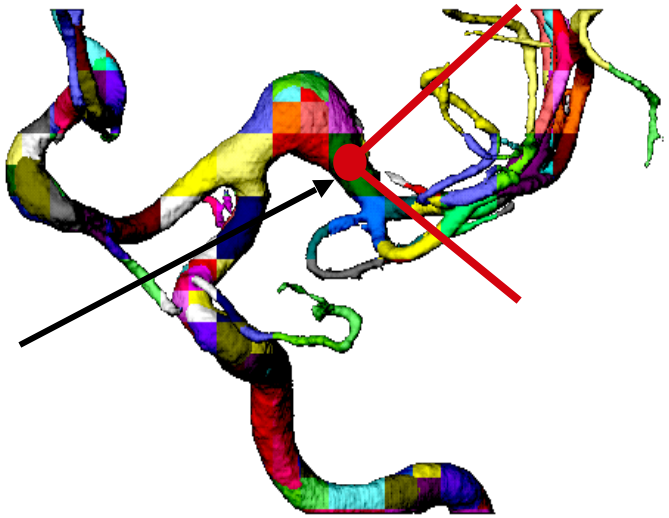
- Anzahl der Objektpolygone wird durch Entfernen der Rückseiten durchschnittlich um etwa die Hälfte reduziert
- Aufwand zur Berechnung des Skalarprodukts sehr gering
- Besteht die Szene nur aus einem einzelnen konvexen Polyeder, so löst Backface-Culling bereits das Sichtbarkeitsproblem!
- Bei konkaven Polyedern oder Szenen, an denen mehrere Objekte beteiligt sind, kann es zu Selbst- und/oder Fremdverdeckung kommen
  - Hier werden aufwändigere Verfahren benötigt

8.5 Culling

View-Frustum-Culling

- Blickfeldtest
- Liegt Objekt (Bounding Volume) im Sichtvolumen (View Frustum)?

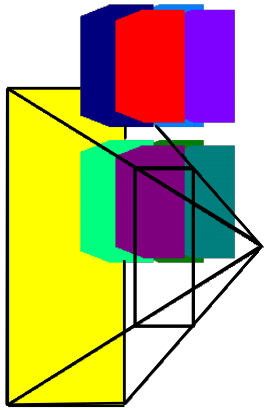


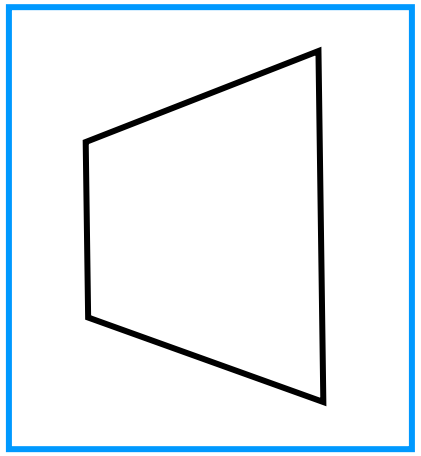


8.5 Culling

View-Frustum-Culling

- Berechnung
  - Schneide Bounding Volume mit dem View-Frustum
  - Einfach nach perspektivischer Transformation (Einheitswürfel)
  - Sonderfall: Bounding Volume umfasst View-Frustum
  - Hierarchischer oder linearer Test





8.5 Culling

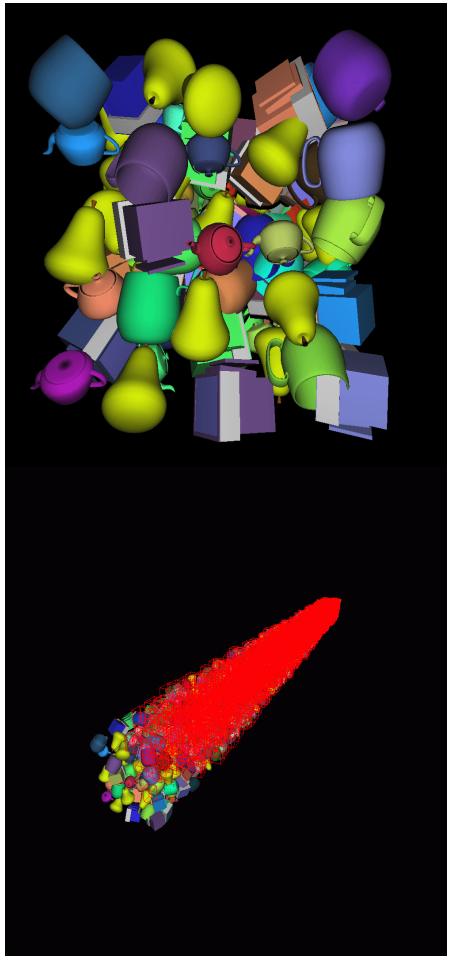
Occlusion-Culling

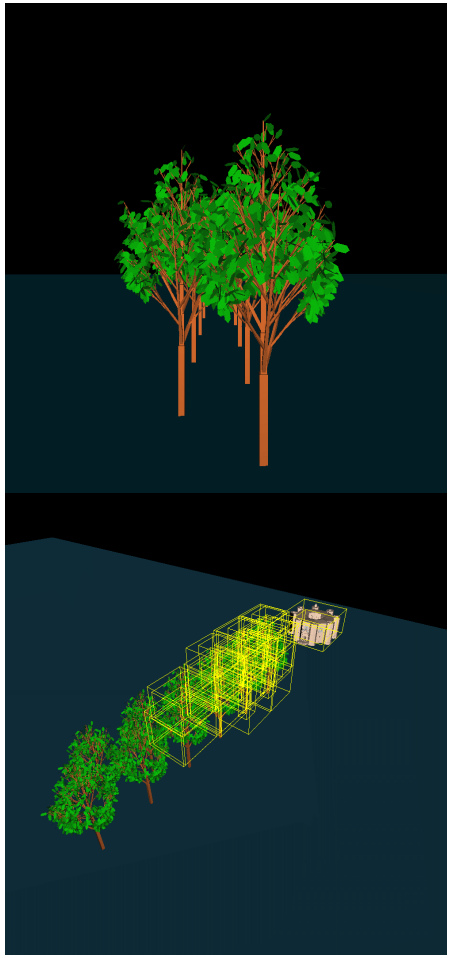
- Ist Geometrie von Geometrie verdeckt?

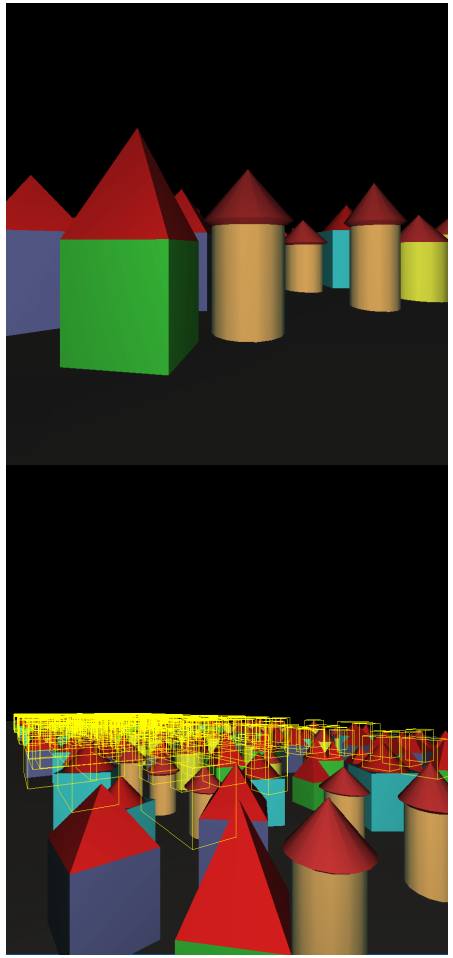
8.5 Culling

Occlusion-Culling

- Gelbe/rote Bounding Volumes sind verdeckt







8.5 Culling

Occlusion-Culling

- Verdeckungstest
  - Wie wird Verdeckung erkannt?
  - Wo ändert sich etwas, wenn man das Objekt darstellen würde
- Objektraum und Bildraumverfahren
  - Hierarchical Z-Buffer (Greene 1993)
  - Hierarchical Occlusion Maps (Zhang 1997)
  - Virtual Occlusion Buffer (Bartz 1999)

8.5 Culling

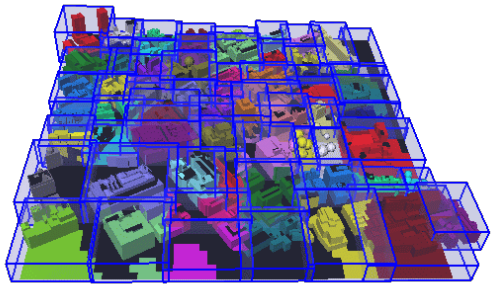
Occlusion-Culling

- Objekte von vorne nach hinten verarbeiten (Tiefensortierung)
- Verwende Bounding Volumes
  - Wenn die Hülle nicht sichtbar ist, dann auch nicht die in ihr eingeschlossene Geometrie
- Konservativ
  - Nicht exakt, aber immer auf der sicheren Seite
  - Objekte, die auf jeden Fall nicht sichtbar sind
  - Sonst darstellen
- Quantitativ/Approximativ
  - Nicht konservativ
  - Wenn nur ein geringer Anteil sichtbar ist, als verdeckt behandeln

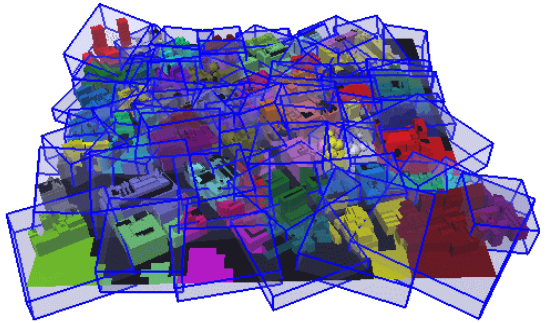
8.5 Culling

Occlusion-Culling

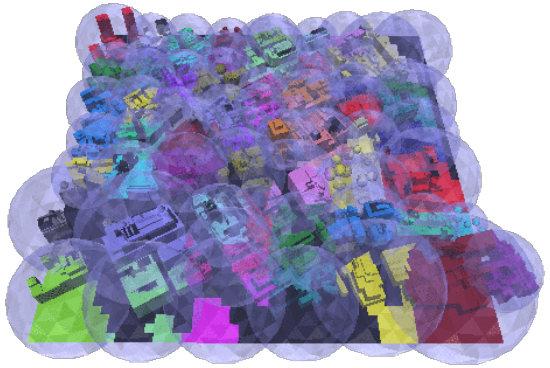
- Optimiere Hüllvolumen: Reduziere Falsch-Positive [Bartz 2001]



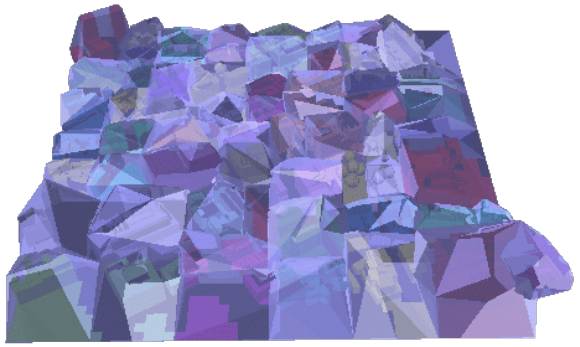
AABB



OBB



Bounding Spheres

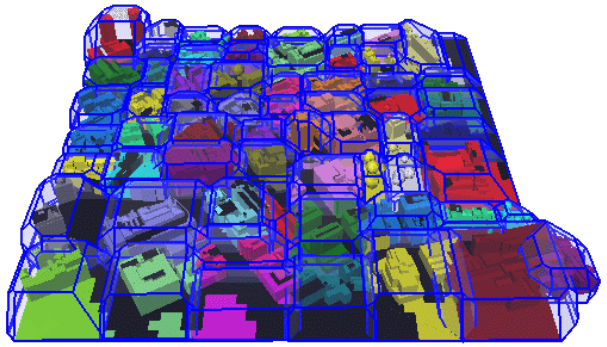


Konvexe Hüllen

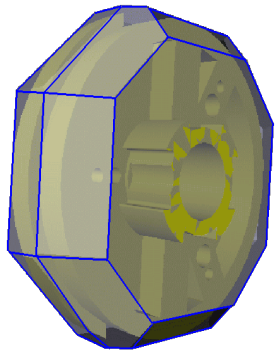
8.5 Culling

Occlusion-Culling

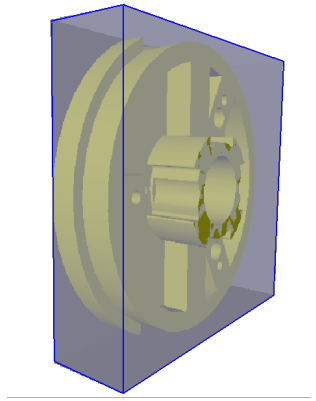
- Optimiere Hüllvolumen: Reduziere Falsch-Positive [Bartz 2001]



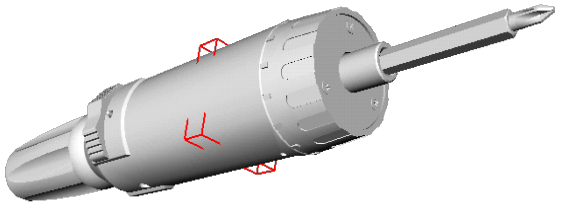
k-DOPs



26-dop



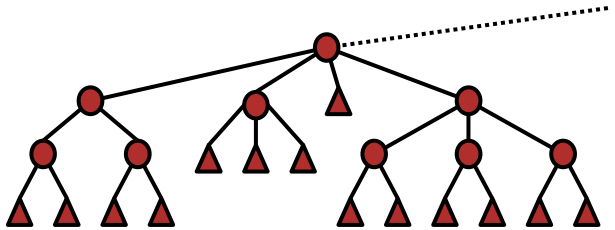
AABB



8.5 Culling

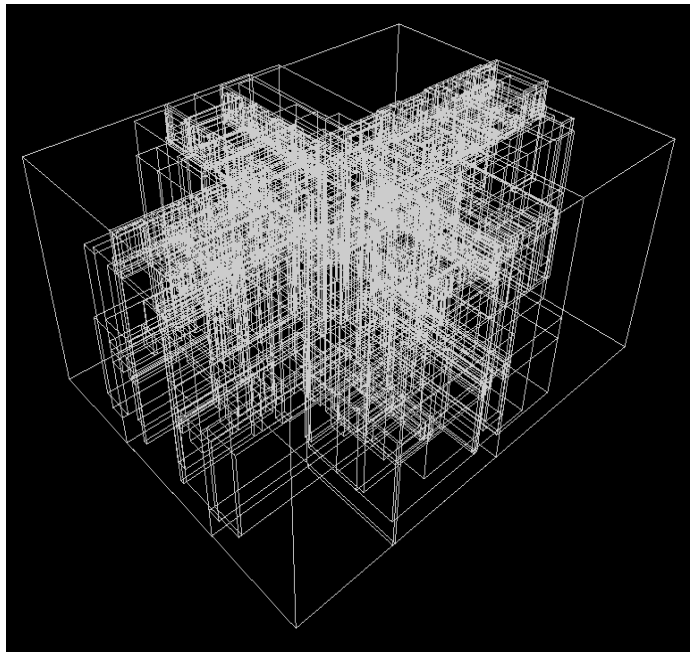
Occlusion-Culling

- Hierarchische Organisation
  - AABB-Hierarchie
  - Kathedrale



Innere Knoten

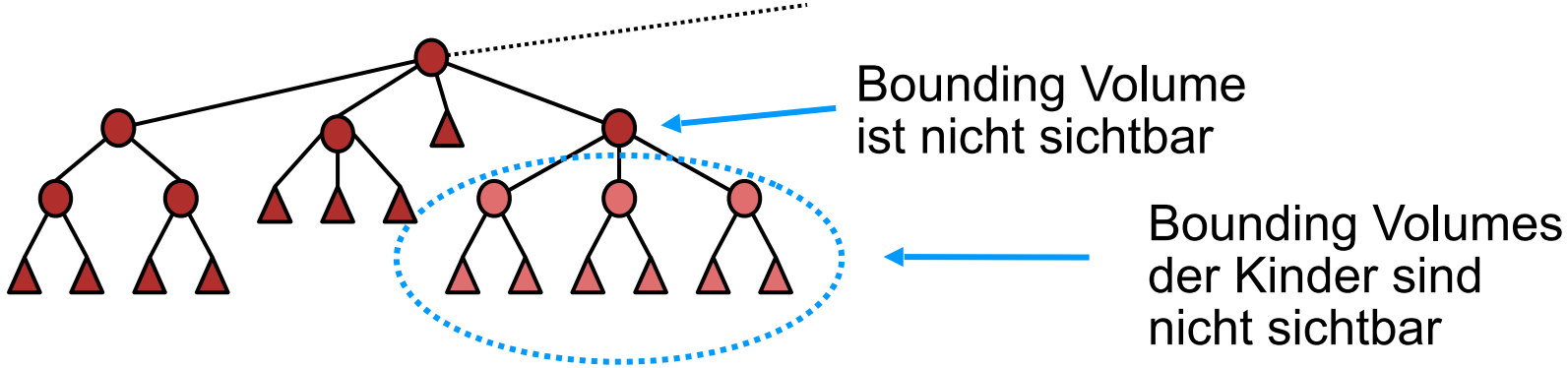
Geometrieblätter



8.5 Culling

Occlusion-Culling

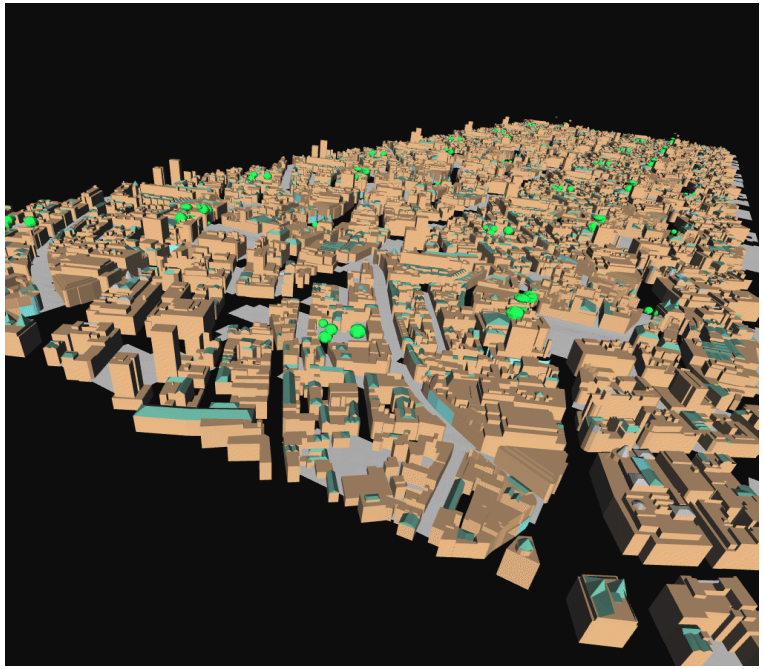
- Hierarchisches Culling testet Baum beginnend an der Wurzel
  - Nicht sichtbar: entferne Knoten
  - Sichtbar: stelle Objekte im Teilbaum dar
    teste Kindknoten

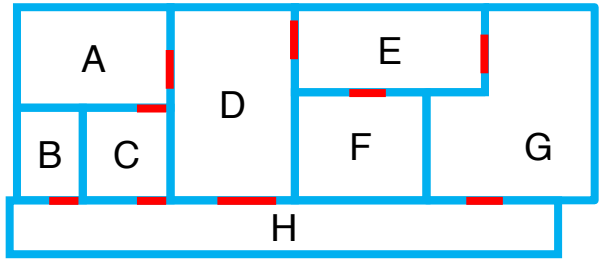


8.5 Culling

Occlusion-Culling

- Verdeckung in Stadtszenen
- Häuser haben feste Zimmer und Öffnungen: Cells and Portals
- Sichtbarkeit durch Portale bestimmt: Potentially Visible Sets (PVS)

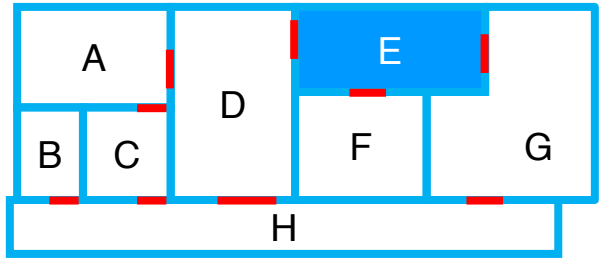


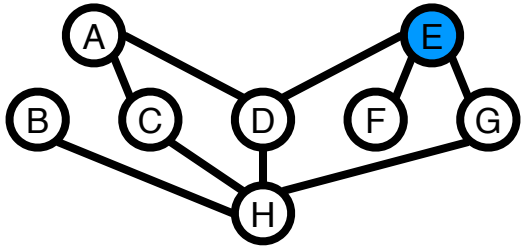


8.5 Culling

Occlusion-Culling

- Zellen und Portale
  - [Teller 1992, Luebke 1995]
  - Berechne Nachbarschaftsgraph zwischen Zellen (durch Portale)
  - Zelle ist nur dann sichtbar, wenn sie durch (eine Folge von) Portale(n) sichtbar ist: Sichtlinie

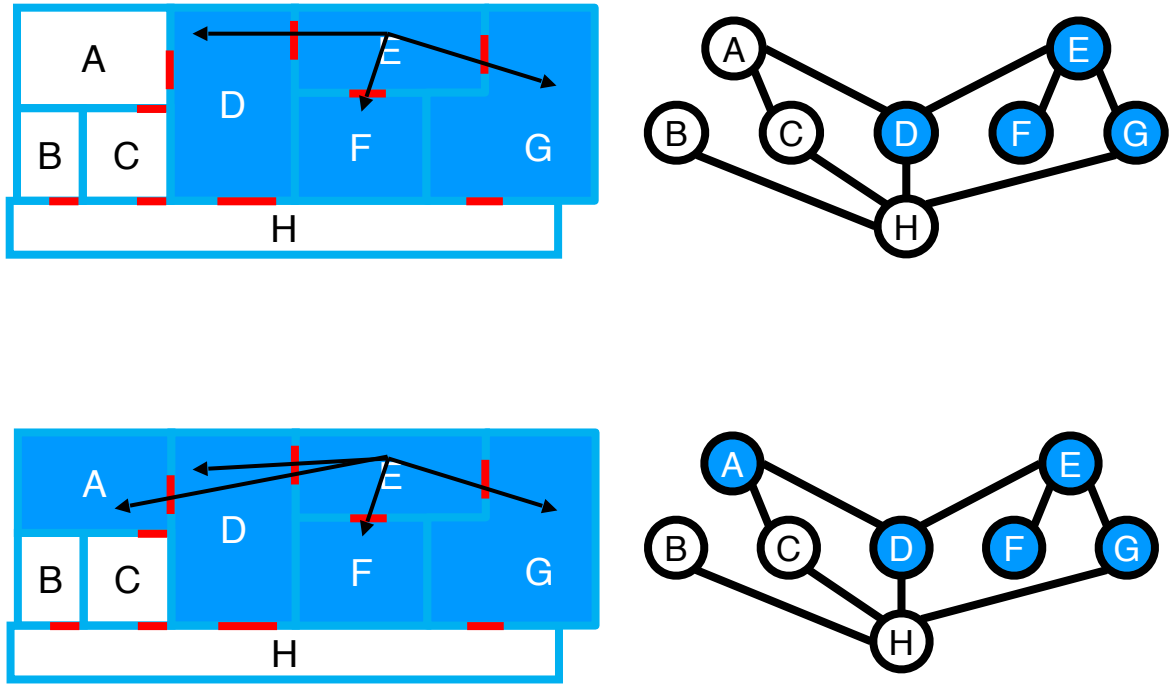




8.5 Culling

Occlusion-Culling

- Zellen und Portale
  - Speichere für jede Zelle alle durch Portale verbundenen Zellen in einem Graphen
  - Assoziiere alle Objekte in einer Zelle mit ihrem Knoten im Graphen
  - Ist vom Betrachter aus ein Portal sichtbar, muss die zugehörige Zelle mit berechnet werden
  - Iterative Analyse



8.5 Culling

Occlusion-Culling

- Für allgemeines OC sind Softwareverfahren zu teuer
  - 1998: Einführung HP OpenGL Extension Occlusion-Flag (fx4)
  - 2001: NVIDIA OpenGL Extension Occlusion-Query (NV 20)
  - 2003: Occlusion-Queries in OpenGL 1.5
- Hardware-Tests
  - Occlusion-Flag:
    Binäre Antwort, ob etwas sichtbar wird
  - Occlusion-Queries:
    Quantitative Antwort, wieviel sichtbar wird

8.5 Culling

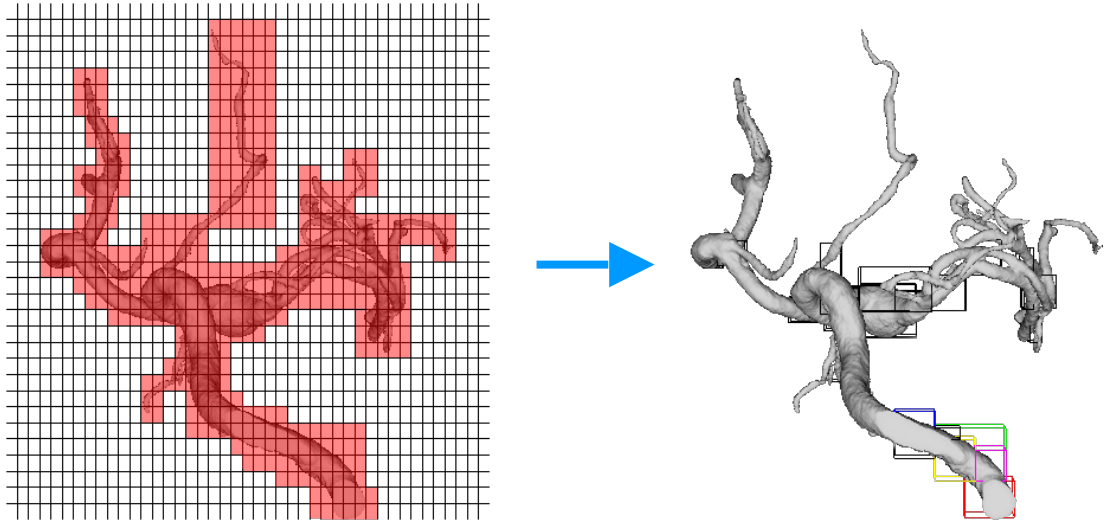
Occlusion-Culling

- Stop-and-Wait-Paradigma
  - Pro Frame, pro Objekt:
    - Teste Bounding Volume
    - Warte Occlusion-Antwort ab
    - Wenn sichtbar, zeichne assoziierte Geometrie
    - Sonst: entferne Objekt (culling)
  - Warten (Pipeline-Flush) führt zu Pipeline-Stall
- Multiple Queries
  - Gebündelte Queries
  - Wartet nicht auf Zwischenergebnisse
    - Kein Pipeline-Stall
  - Nutzt aber auch keine Zwischenergebnisse aus
  - Gesparte Stalls viel billiger als zu viel gerenderte Geometrie
    - 1/10, 1/20
  - Also: Vorsortierung in gegenseitig unabhängige Objekte (sich nicht gegenseitig verdeckend)

8.5 Culling

Occlusion-Culling

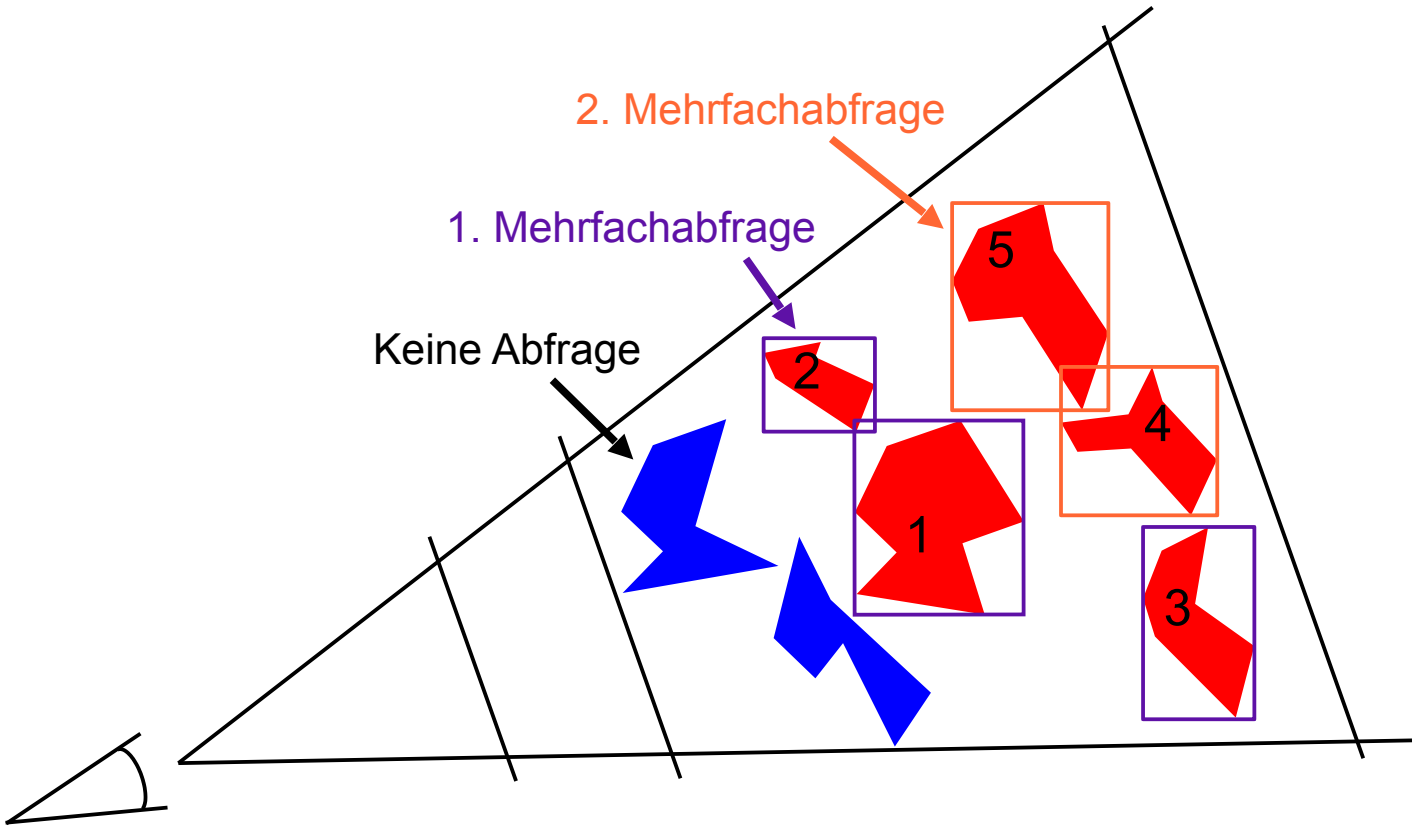
- Vorsortierung in gegenseitig unabhängige Objekte
  - OccupancyMap Hierarchie [Staneker 2003]
    - Nutze Kohärenz
    - Reduziere Falsch-Positive
    - Günstiger Vorab-Overlap-Test
    - Spart auch redundante Occlusion-Queries



8.5 Culling

Occlusion-Culling

- Überlappfreie Vorsortierung



Quellen

- Computergraphik, Universität Leipzig
  (Prof. D. Bartz)
- Graphische Datenverarbeitung I, Universität Tübingen
  (Prof. W. Straßer)
- Graphische Datenverarbeitung I, TU Darmstadt
  (Prof. M. Alexa)

UNIVERSITÄT LEIPZIG

Beleuchtung

COMPUTERGRAPHIK

Inhaltsverzeichnis

9 Beleuchtung

- 9.1 Farbe
- 9.2 Modelle
- 9.3 Lokale Beleuchtungsmodelle
- 9.4 Schattierung
- 9.5 Globale Beleuchtung

9.1 Farbe

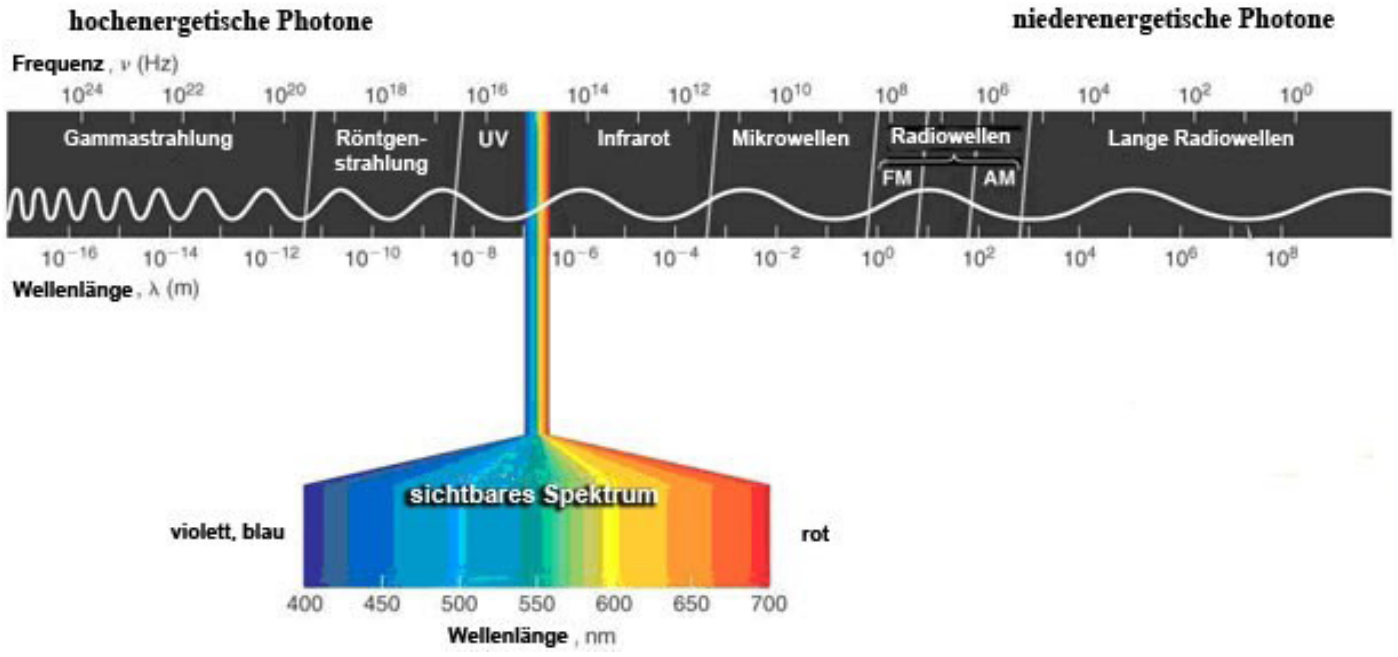
- Farbe ist ein wesentlicher Bestandteil realistischer Computergraphik
- Die Berechnung der Beleuchtung basiert darauf, für jeden darzustellenden Punkt einen Farbwert zu bestimmen

- Fragestellungen:
  - Wie kann man eine bestimmte Farbe exakt spezifizieren?
    ⇒ Farbräume, Farbmodelle
  - Wie exakt kann eine exakt spezifizierte Farbe auf einem Ausgabegerät angezeigt werden?
    ⇒ geräteabhängige Farbräume
  - Wie viele verschiedene Farben können durch eine Graphikhardware bereitgestellt werden?
    ⇒ technische Realisierbarkeit

9.1 Farbe

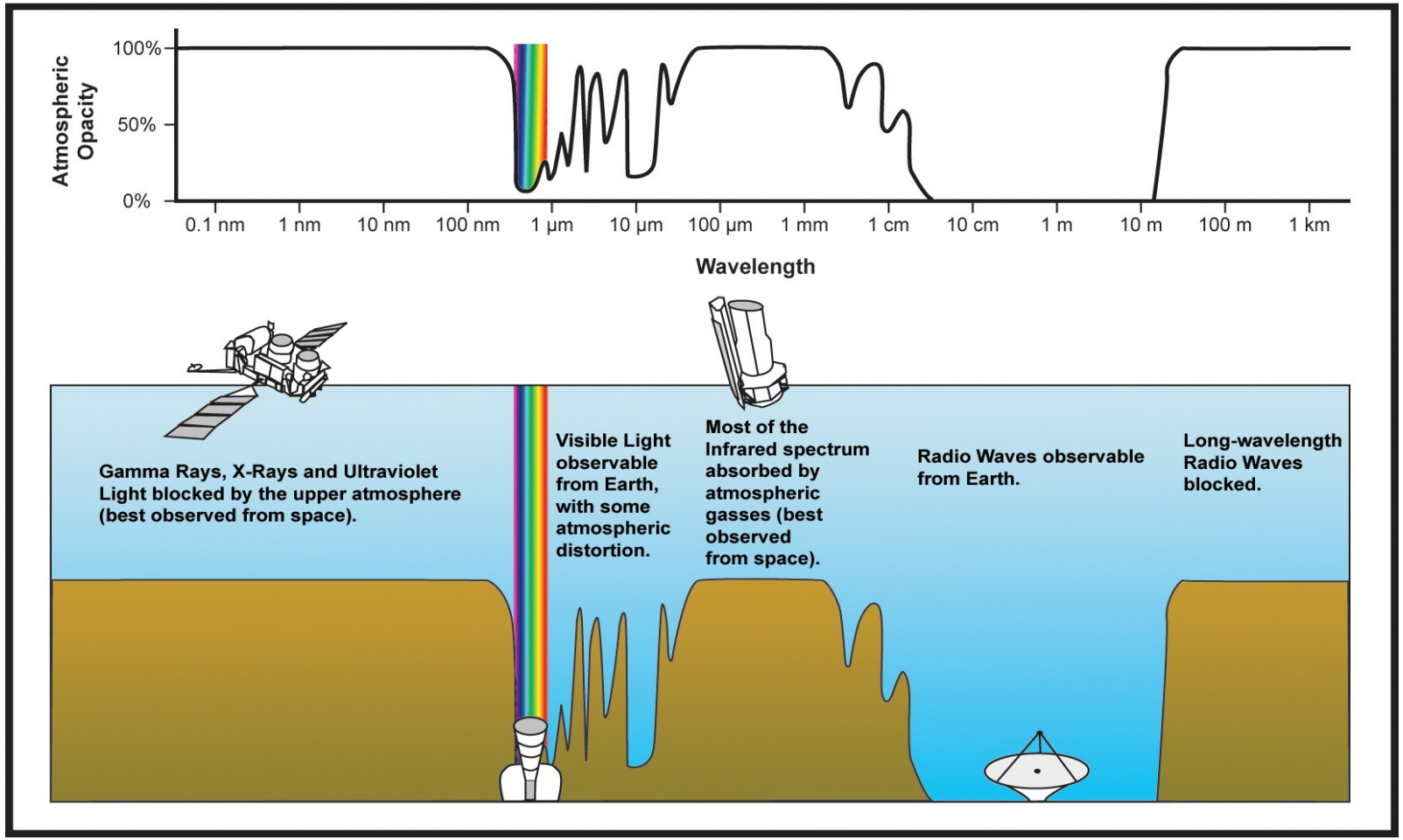
Licht und das sichtbare Spektrum

- Das sichtbare Lichtspektrum (400 – 700 nm) ist ein kleiner Teil des elektromagnetischen Spektrums.
- Manche Tiere können
  - im Infrarotbreich sehen (Schlangen)
  - ultraviolettes Licht wahrnehmen (Insekten)



9.1 Farbe

Elektomagnetische Wellen auf der Erde

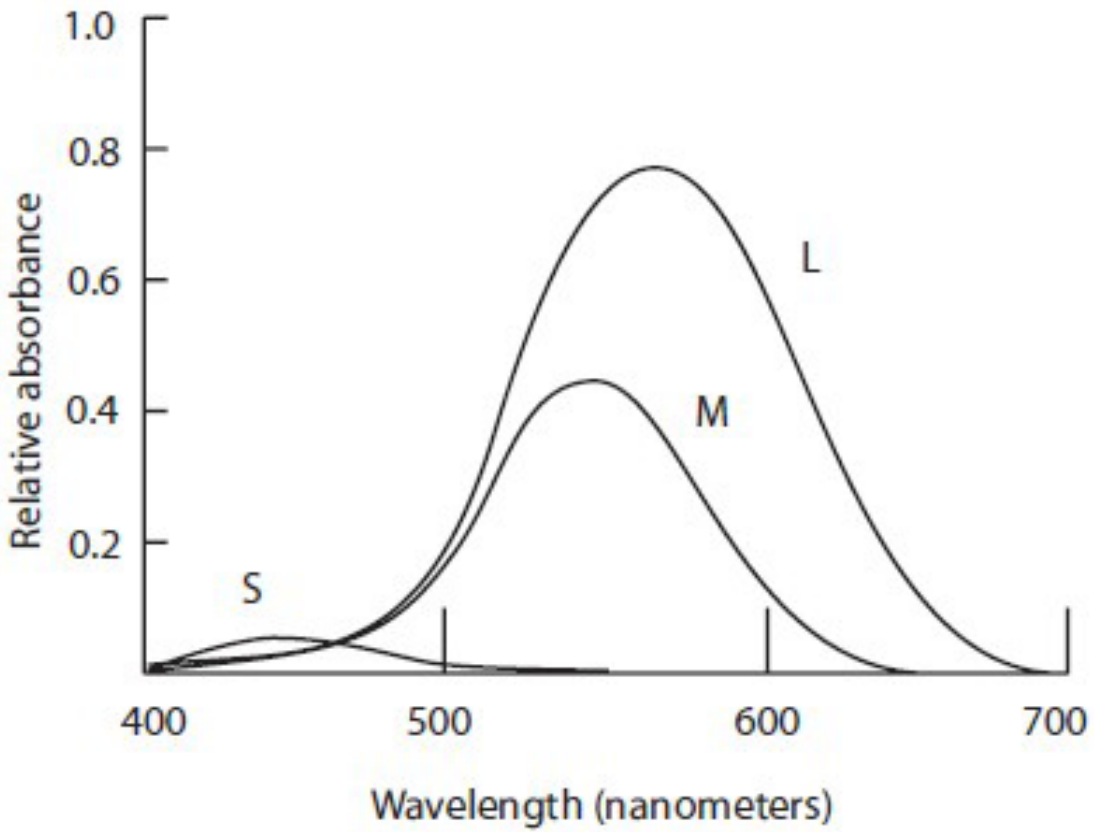


[http://en.wikipedia.org/wiki/File:Atmospheric_electromagnetic_transmittance_or_opacity.jpg]

9.1 Farbe

Dreifarbenlehre

- Der Mensch besitzt drei Typen von Zapfen
  - Jeder Zapfen hat seine maximale Sensitivität bei einer anderen Wellenlänge
  - S = **blau**, M = **grün**, L= **rot**
  - Ein Farbsystem bestehend aus drei Grundtönen reicht aus, um alle vom Menschen wahrnehmbaren Farben darzustellen
    → Dreifarbenlehre



9.1 Farbe

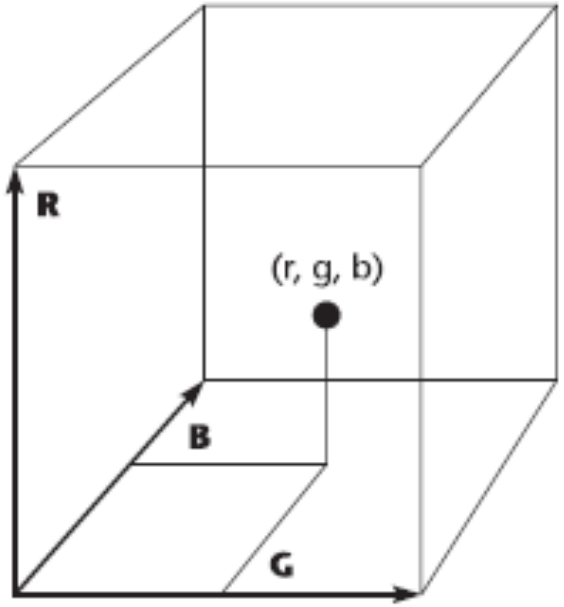
Dreidimensionaler Farbraum

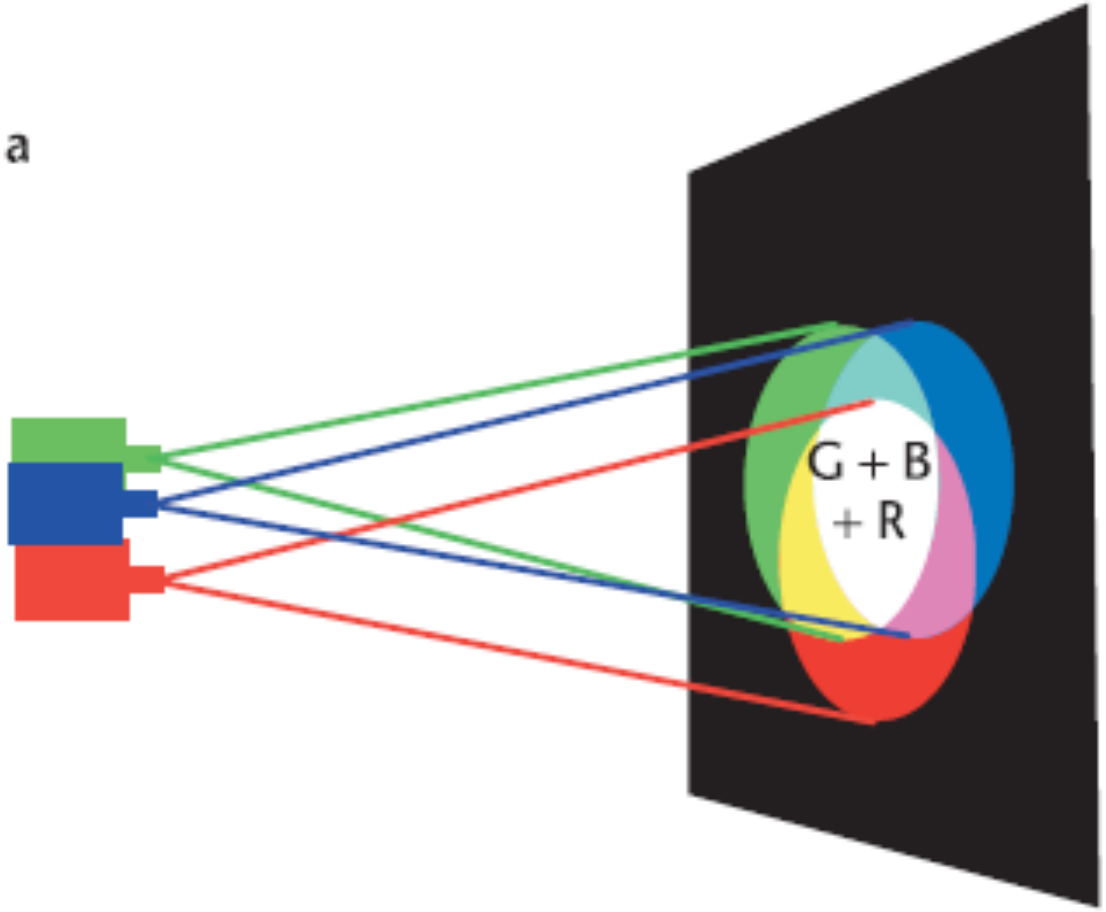
- Je nach Anforderung können verschiedene standardisierte Farbräume (Farbmodelle) verwendet werden
  - RGB
    - Additives Farbmodell
    - Monitore
    - Computergraphik
  - CMY
    - Subtraktives Farbmodell
    - Drucker
  - HSV
    - zerlegt Farbe in Farbwert, Helligkeit und Sättigung
  - CIE
    - internationaler Standard zur Farbspezifikation

9.1 Farbe

RGB

- Man kann eine beliebige (Licht-)Farbe C mittels der Grundfarben Rot (R), Grün (G) und Blau (B) als
  $C = rR + gG + bB$
  ausdrücken





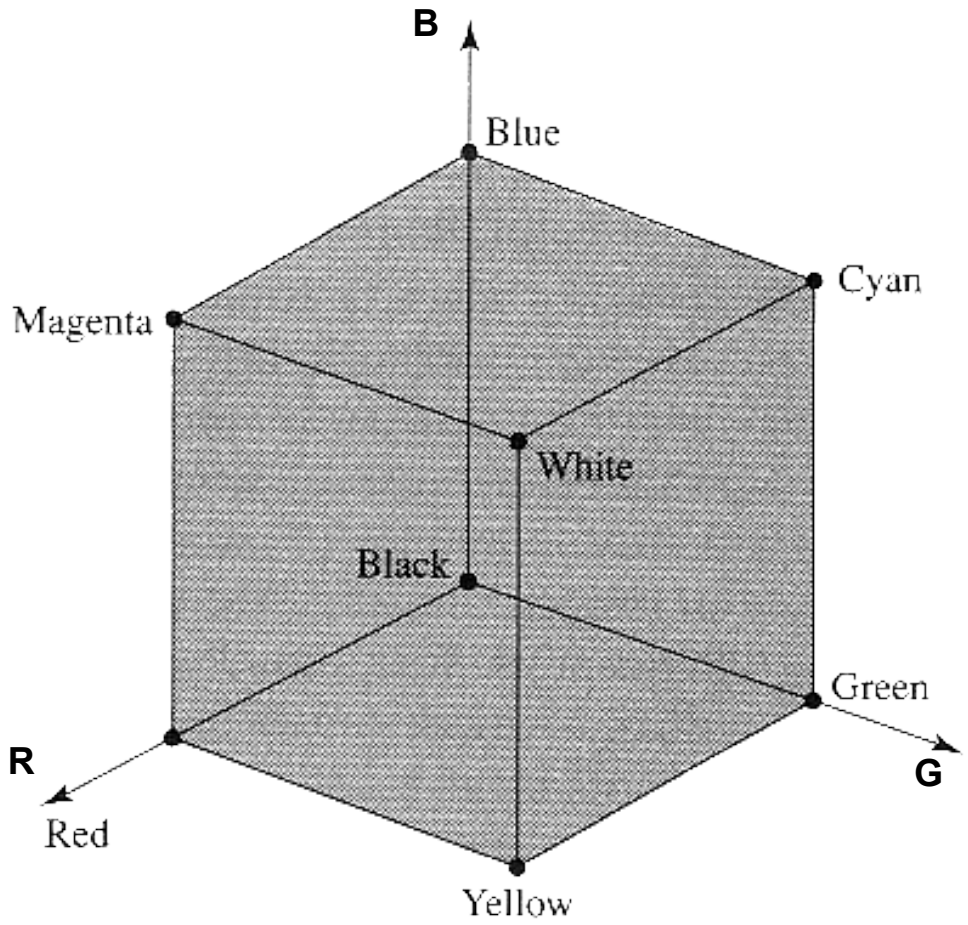
9.1 Farbe

RGB

- Farben in den Eckpunkten des Würfels
  - (0, 0, 0) = Schwarz
  - (1, 0, 0) = Rot
  - (0, 1, 0) = Grün
  - (0, 0, 1) = Blau
  - (1, 1, 0) = Gelb
  - (1, 0, 1) = Magenta
  - (0, 1, 1) = Cyan
  - (1, 1, 1) = Weiß

9.1 Farbe

RGB



9.1 Farbe

RGB

- Das Modell ist bezüglich der Farbwahrnehmung nicht linear
- Bei typischer Farbauflösung von 8 Bit pro Grundfarbe
  - Im Farbwürfel existieren Regionen
    - in denen benachbarte Punkte für das Auge denselben Farbeindruck hervorrufen
    - die Farben benachbarter Punkte für das Auge sehr unterschiedlich sind
- Für den Anwender kann es schwierig sein
  - zu einer gewünschten Farbe ein entsprechendes (r, g, b)-Tupel zu ermitteln
  - Eine Farbe etwas abzuschwächen (erfordert ungleichmäßige Änderungen von r, g und b)

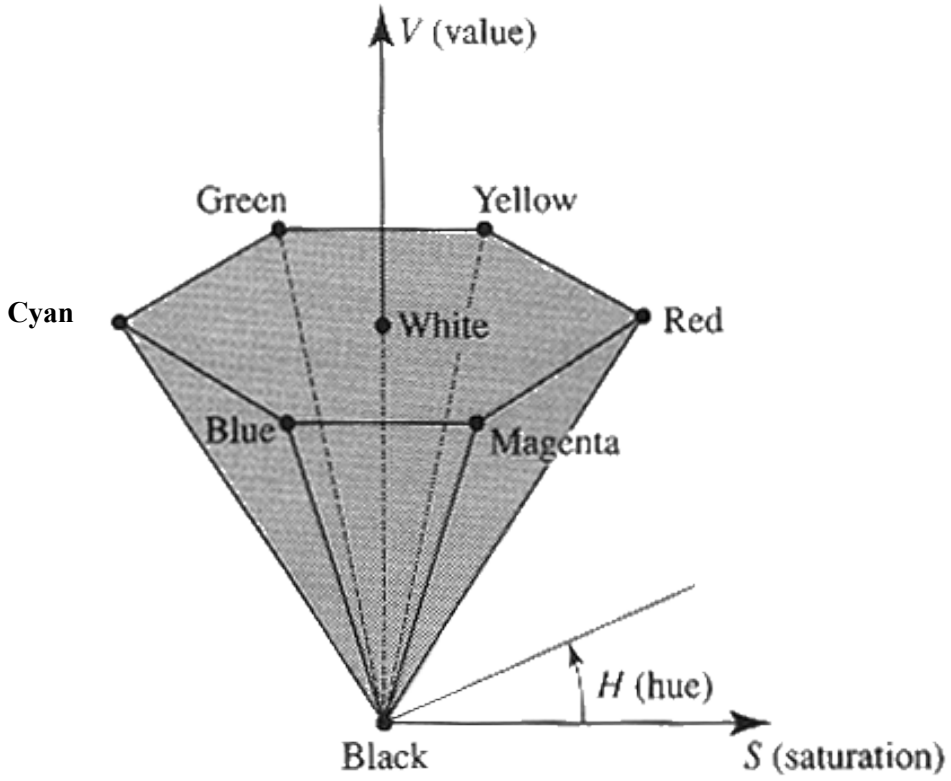
9.1 Farbe

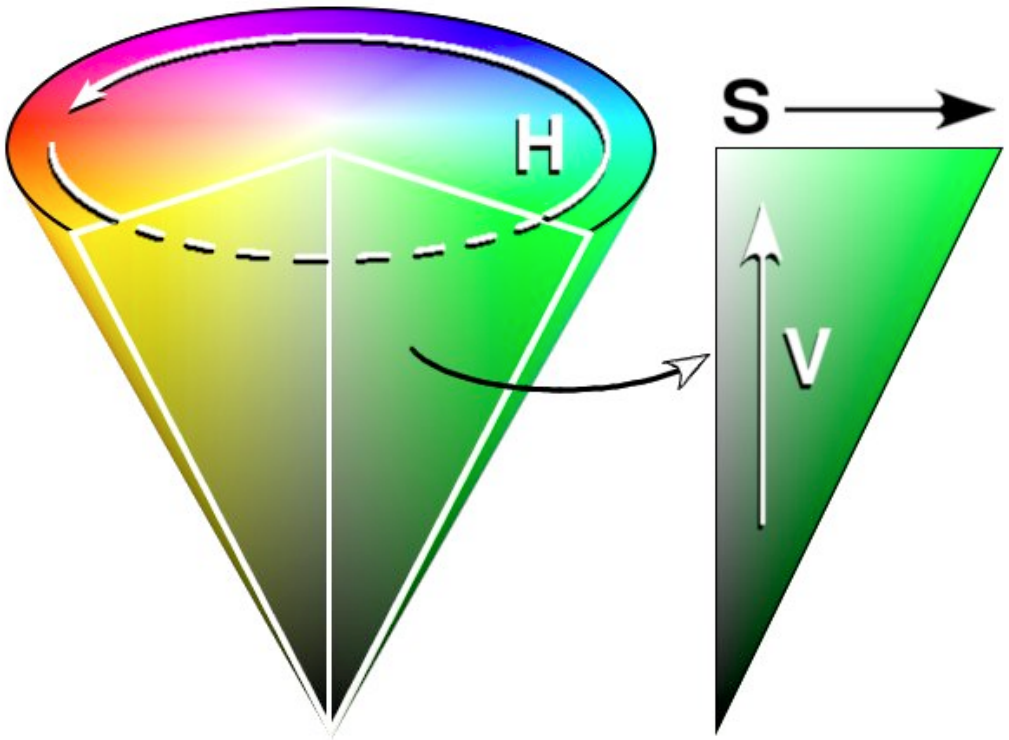
HSV

- Dieses Modell wurde zur Unterstützung einer intuitiven Farbauswahl entwickelt (wahrnehmungsorientiertes Farbmodell)
- Farbkörper im 3D-Farbraum ist Pyramide mit sechseckiger Grundfläche
- verwendet Polarkoordinaten:
  - Hue: Farbe („Farbfamilie“) als „Farbwinkel“ in Grad
    $0° \leq H < 360°$
  - Saturation: Sättigung
    $0 \leq S \leq 1$
    (Verkleinerung addiert Weiß)
  - Value: Helligkeit
    $0 \leq V \leq 1$
    (Verkleinerung addiert Schwarz)

9.1 Farbe

HSV


V (value)
Green
Yellow
Cyan
White
Red
Blue
Magenta
H (hue)
Black
S (saturation)


H
S
V

9.1 Farbe

HSV – RGB

- Die Grundfläche der HSV-Pyramide entsteht aus dem RGB-Würfel durch Projektion entlang der Raumdiagonalen von Weiß nach Schwarz auf eine dazu senkrecht stehende Ebene
- Im HSV-Modell besitzen Komplementärfarben eine Winkeldifferenz von $180°$ im H -Wert
- Korrespondierende Punkte

RGB	Farbe	HSV
(1, 0, 0)	Rot	(0, 1, 1)
(1, 1, 0)	Gelb	(60, 1, 1)
(0, 1, 0)	Grün	(120, 1, 1)
(0, 1, 1)	Cyan	(180, 1, 1)
(0, 0, 1)	Blau	(240, 1, 1)
(1, 0, 1)	Magenta	(300, 1, 1)

9.1 Farbe

CMY

- Cyan-Magenta-Yellow-(CMY)-Farbraum
- Oft kommt es bei der Farbwiedergabe auf Papier zu kleineren Differenzen
- Drucker verwenden häufig noch
  - Schwarz
  - zusätzliche Farben

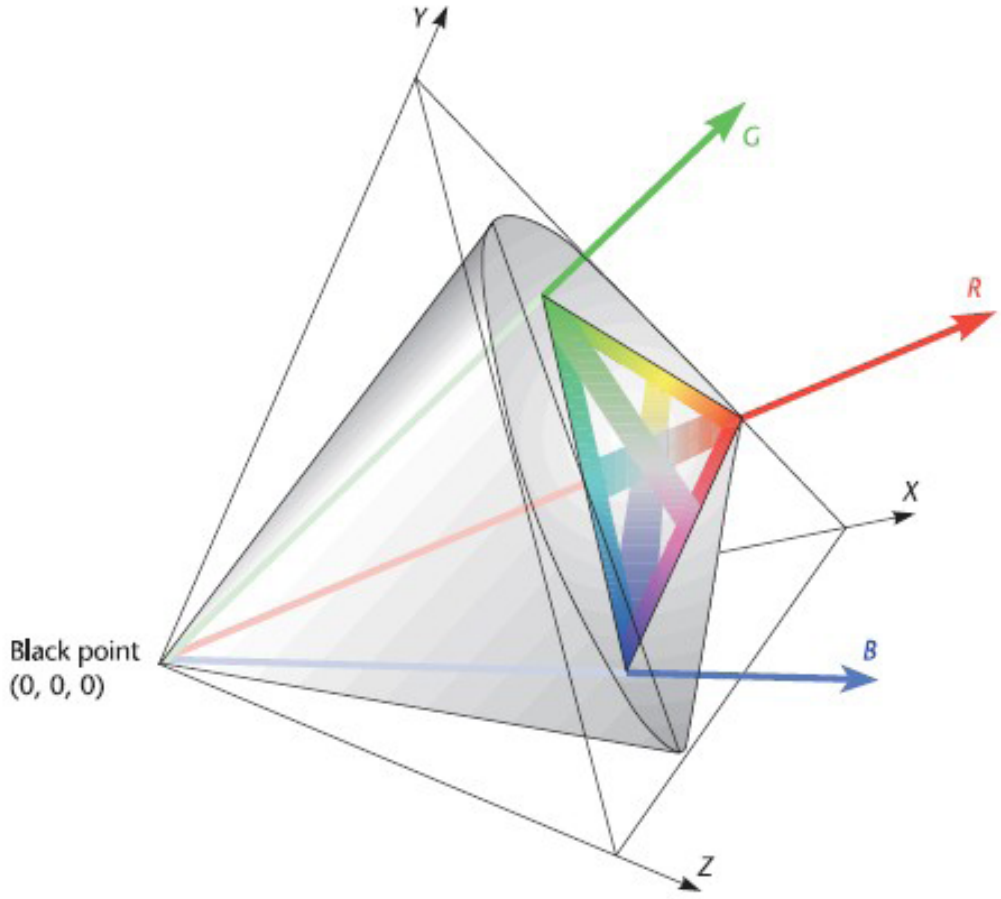
- Berechnung

$$\begin{pmatrix} C \\ M \\ Y \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} - \begin{pmatrix} R \\ G \\ B \end{pmatrix}$$

9.1 Farbe

CIE Standard für Farben

- Das Commission Internationale de l'Eclairage (CIE) System verwendet drei abstrakte Primärfarben X, Y, Z
- Wahrnehmbare Farben sind als graues Volumen dargestellt (die Messungen stammen von vor 1931)
- Die Farben, welche durch drei farbige Lichter in rot, grün und blau erzeugt werden können, sind durch die einbeschriebene Pyramide gekennzeichnet



9.1 Farbe

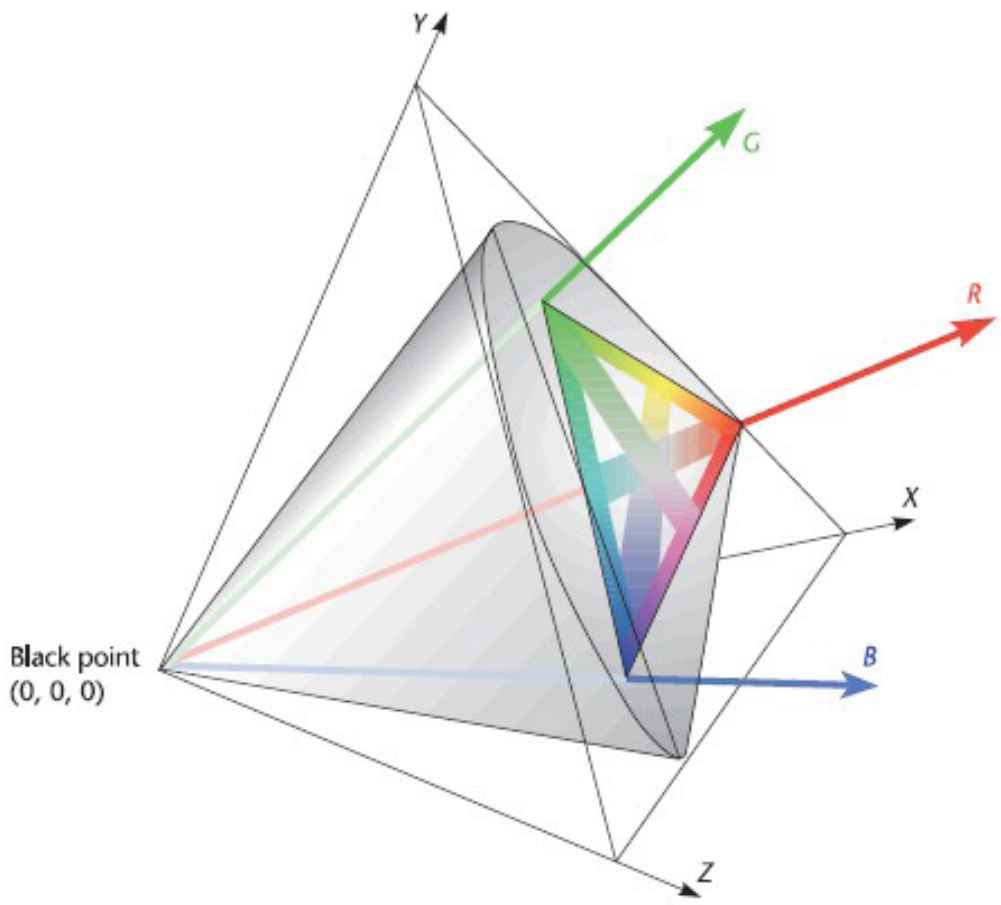
CIE Standard für Farben

- Farben in XYZ Koordinaten sind nur schwer zu verstehen
- Leichter verständlich ist die Repräsentation in Farbwertanteilen (chromaticity coordinates)

$$x = \frac{X}{X+Y+Z}$$

$$y = \frac{Y}{X+Y+Z}$$

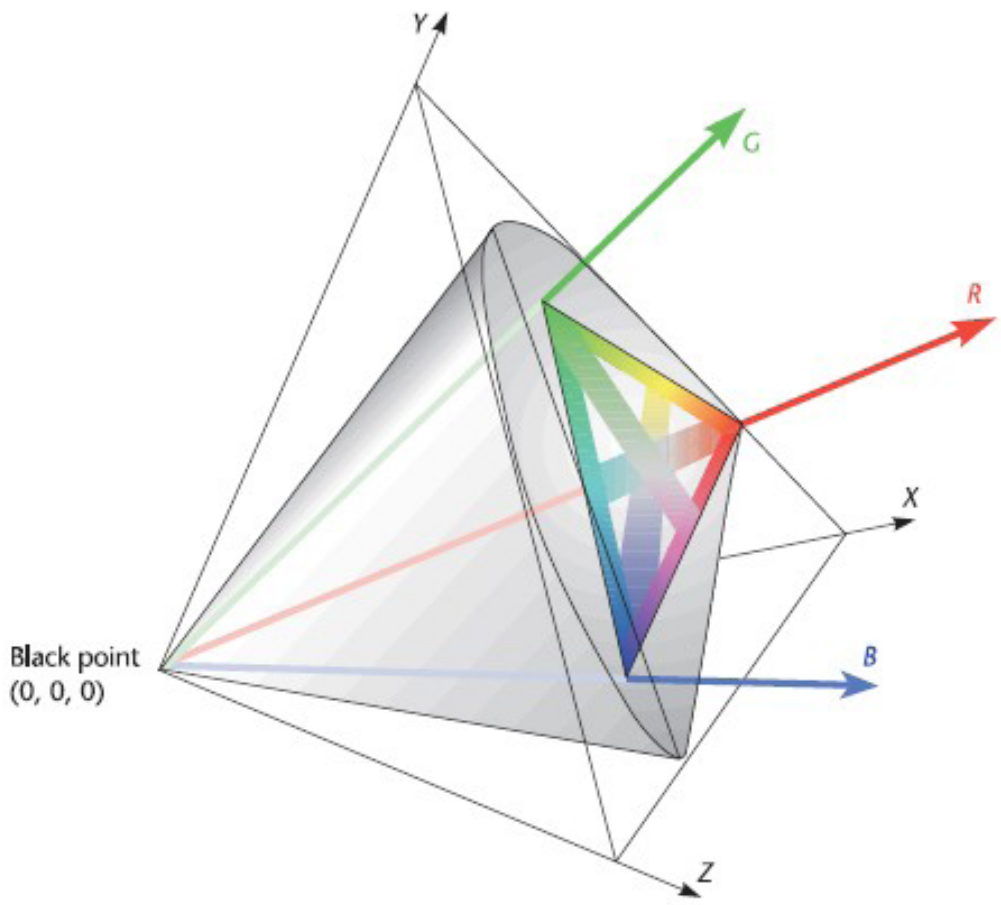
$$z = \frac{Z}{X+Y+Z}$$



9.1 Farbe

CIE Standard für Farben

- Wegen x + y + z = 1, reicht es die x und y Werte anzugeben
- Üblicherweise werden Farben in Form von (x, y, Y) spezifiziert, wobei Y die Helligkeit angibt

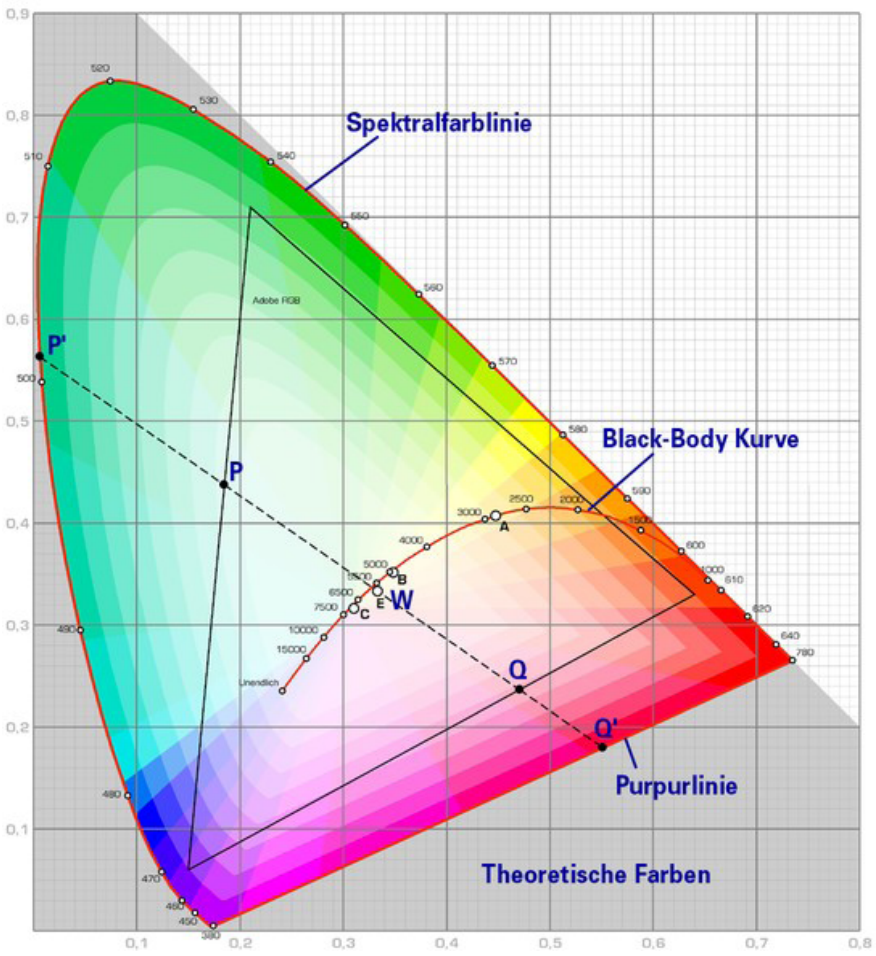


9.1 Farbe

CIE Normfarbtafel (Eigenschaften)

- Wenn zwei farbige Lichtquellen als Punkte im Diagramm dargestellt werden, liegen die Farben, welche als Mischung dieser beiden Lichtquellen erzeugt werden können, auf einer Linie

- Alle Farben innerhalb eines Dreiecks, dessen Eckpunkte drei farbige Lichtquellen repräsentieren, können mit Hilfe dieser Lichtquellen erzeugt werden

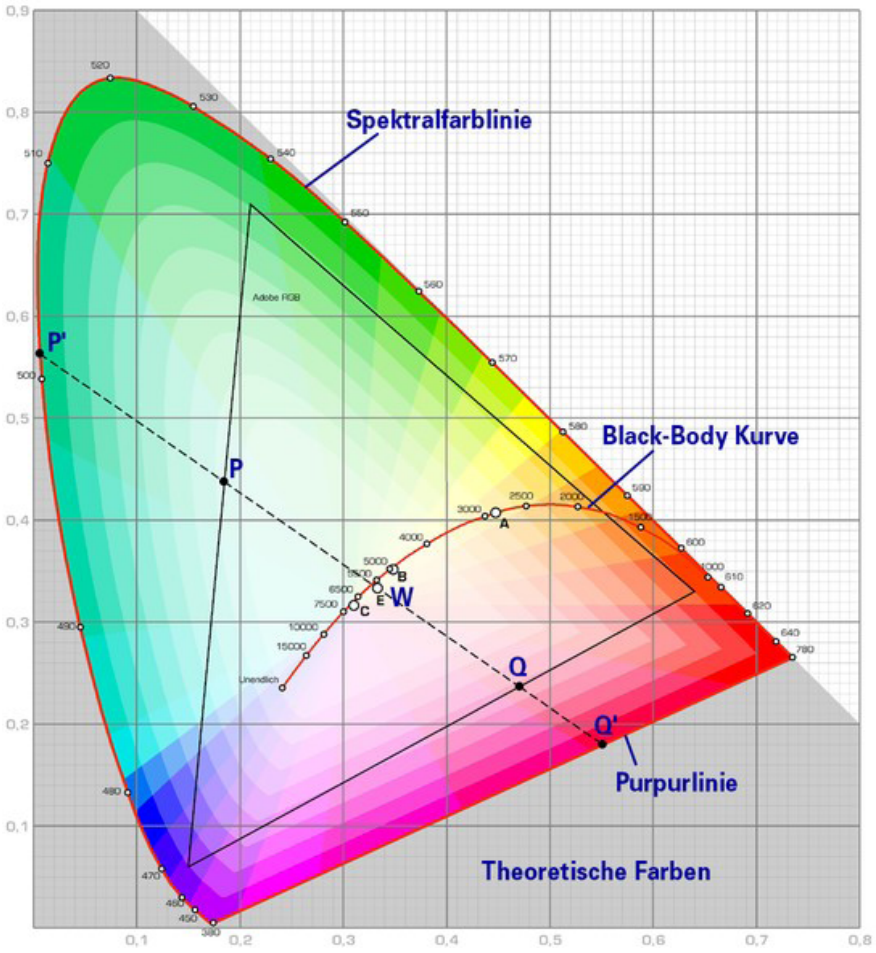


9.1 Farbe

CIE Normfarbtafel (Eigenschaften)

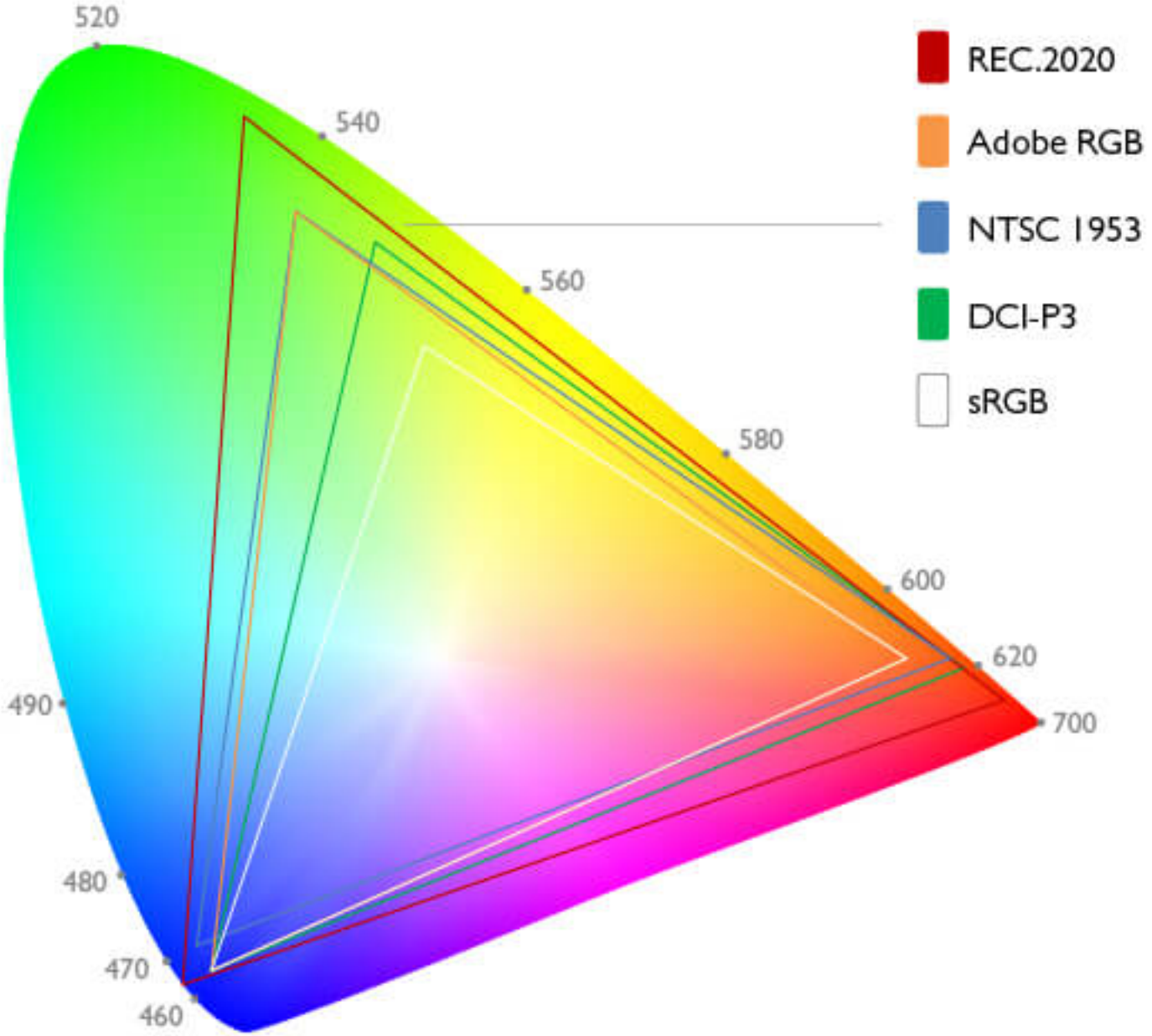
- Die hufeisenförmige Spektralfarblinie enthält alle Farben mit den höchsten Sättigungsgraden in den einzelnen Farbtönen. Man sieht sie, wenn man Licht betrachtet, welches nur aus einer Wellenlänge besteht

- Die Purpurline verbindet die beiden Enden der Spektralfarblinie



9.1 Farbe

Gamut (Farbbereich)



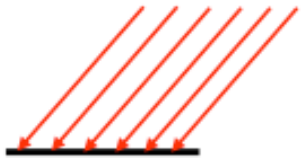
9.2 Modelle

- Beleuchtungsmodelle und Reflexionsmodelle
  - dienen zur Bestimmung der Farbe an einem Punkt eines Objektes
  - Inkonsistente Terminologie in der Literatur
    - Illumination-Model
    - Lighting-Model
    - Reflection-Model
    - NICHT: Shading-Model
- Schattierungsmodelle
  - dienen zur Bestimmung der Farbe eines Pixels auf dem Monitor

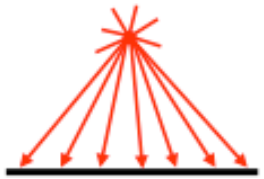
9.2 Modelle

Lichtquellen

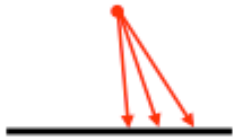
- Gerichtete Lichtquellen
  - Position im Unendlichen $(x, y, z, 0)^T$



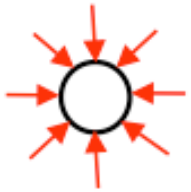
- Punktlichtquellen
  - Endliche Position $(x, y, z, 1)^T$



- Spotlichtquellen
  - Cutoff-Winkel, Position, Richtung



- Ambientes Licht
  - Gleichförmig aus allen Richtungen



9.2 Modelle

Beleuchtungsmodelle

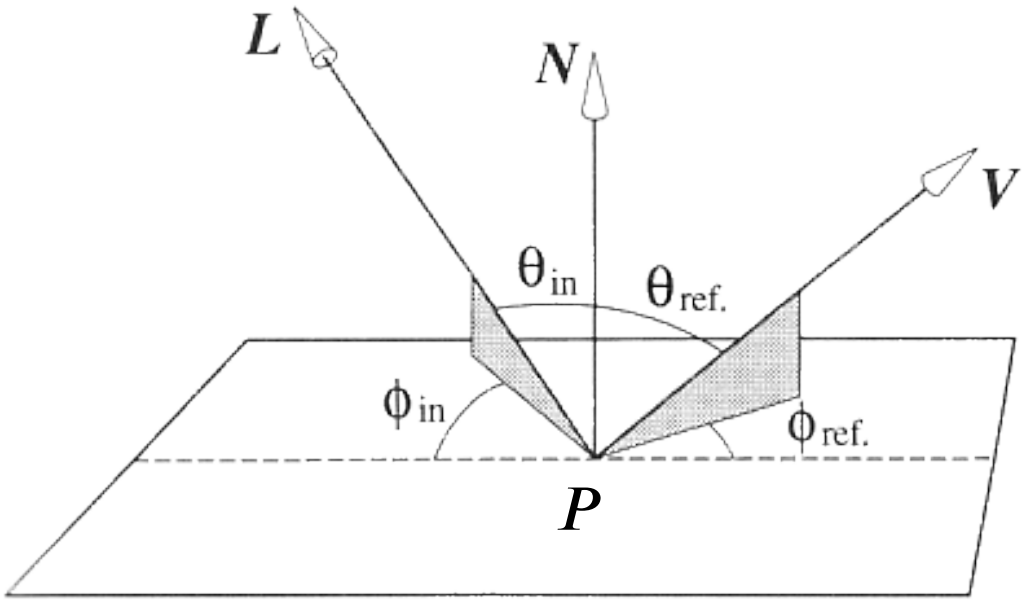
- Lokal
  - nur direkte Beleuchtung
  - Berechnung der Intensität (Farbe) eines Punktes in Abhängigkeit von
    - direktem Lichteinfall einer Lichtquelle
- Global
  - direkte und indirekte Beleuchtung
  - Berechnung der Intensität (Farbe) eines Punktes in Abhängigkeit von
    - direktem Lichteinfall einer Lichtquelle
    - indirekt einfallendem Licht
      - Reflexion
      - Transmission
  - benutzen oft erweiterte lokale Beleuchtungsmodelle

9.3 Lokale Beleuchtungsmodelle

Geometrische Betrachtung

- P: Punkt auf Objektoberfläche
- N: Flächennormalenvektor in P
- L: Vektor von P zu einer Punktlichtquelle, normiert
- V: Vektor von P zum Augpunkt (ViewingPoint), normiert

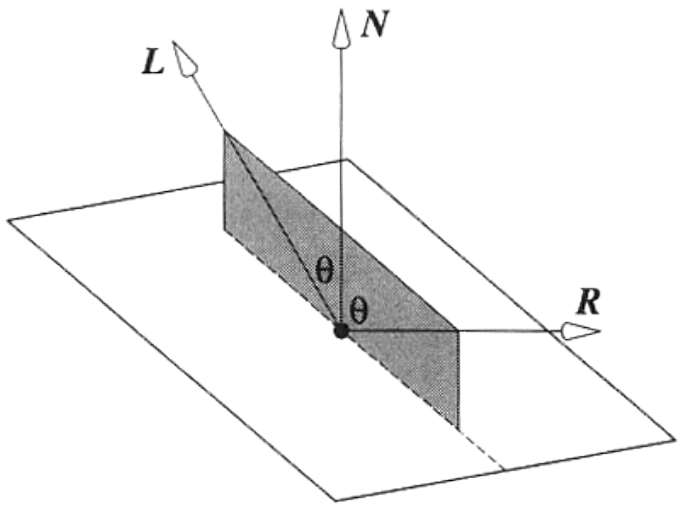
- ϕ_{in}, Θ_{in}: Einfallswinkel
- ϕ_{ref}, Θ_{ref}: Reflektionswinkel

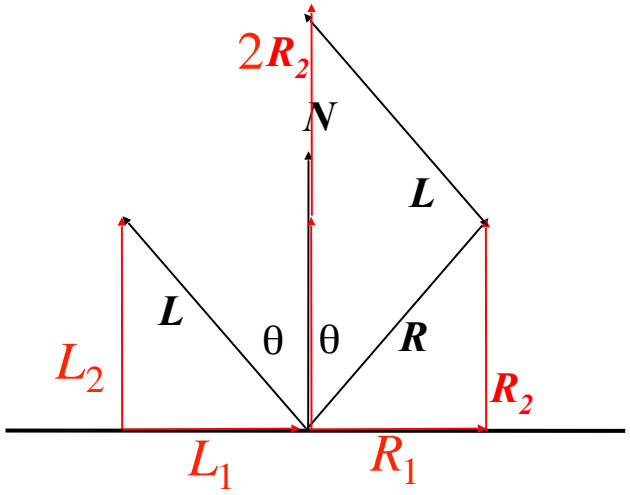


9.3 Lokale Beleuchtungsmodelle

Geometrische Betrachtung

- R : Vektor des reflektierten Lichts
- Es gilt
  - L und R liegen in einer Ebene
  - $\Theta = \Theta_{in} = \Theta_{ref}$



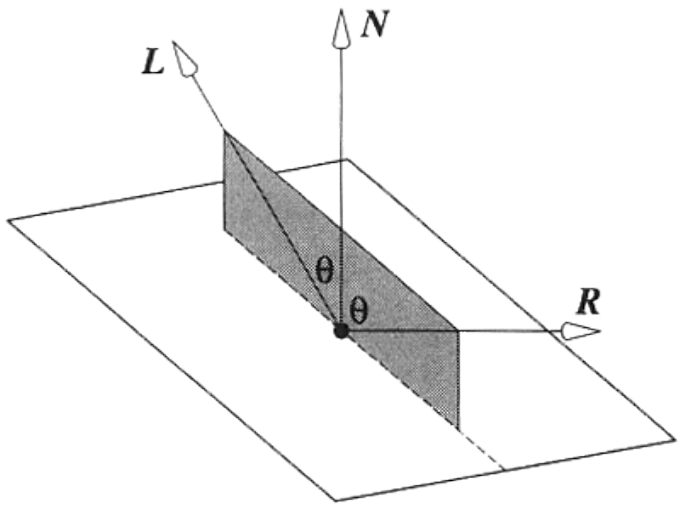


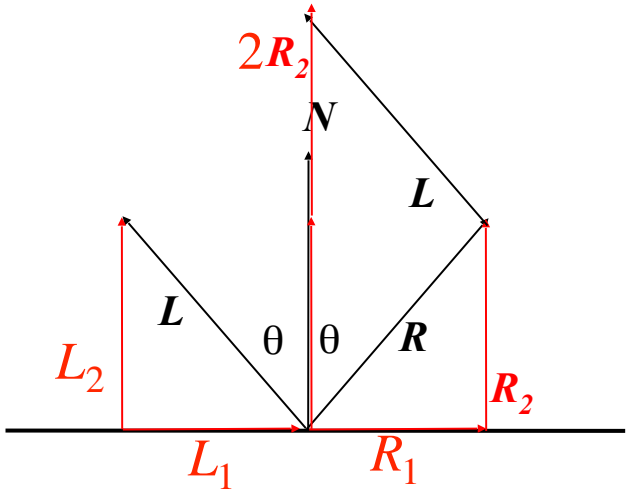
9.3 Lokale Beleuchtungsmodelle

Geometrische Betrachtung

- R : Vektor des reflektierten Lichts
- Es gilt
  - L und R liegen in einer Ebene
  - $\Theta = \Theta_{in} = \Theta_{ref}$
- $L = -L_1 + L_2$
- $R = R_1 + R_2$
- $R_2 = L_2 = (L \cdot N)N$

- $R_1 = L_1 = L_2 - L$
- $R = 2R_2 - L = 2(L \cdot N)N - L$





9.3 Lokale Beleuchtungsmodelle

Einschub Fast Inverse Square Root

- Das Berechnen von Einheitsvektoren war lange Zeit sehr teuer
  - Benötigt Reziproke einer Gleitkommazahl
- FISR Methode konnte dies aber approximieren!

9.3 Lokale Beleuchtungsmodelle

Einschub Fast Inverse Square Root

```
float Q_rsqrt( float number )
{
    long i;
    float x2, y;
    const float threehalfs = 1.5F;

    x2 = number * 0.5F;
    y  = number;
    i  = * ( long * ) &y;                       // evil floating point bit level hacking
    i  = 0x5f3759df - ( i >> 1 );               // what the fuck?
    y  = * ( float * ) &i;
    y  = y * ( threehalfs - ( x2 * y * y ) );   // 1st iteration
//  y  = y * ( threehalfs - ( x2 * y * y ) );   // 2nd iteration, this can be removed

    return y;
}
```

- Die Methode wird John Carmack zu geschrieben, ist aber wohl älter
- Wurde zuerst in Quake 3 eingesetzt
- Heutzutage nur noch in FPGAs oder Hardware verwendet

9.3 Lokale Beleuchtungsmodelle

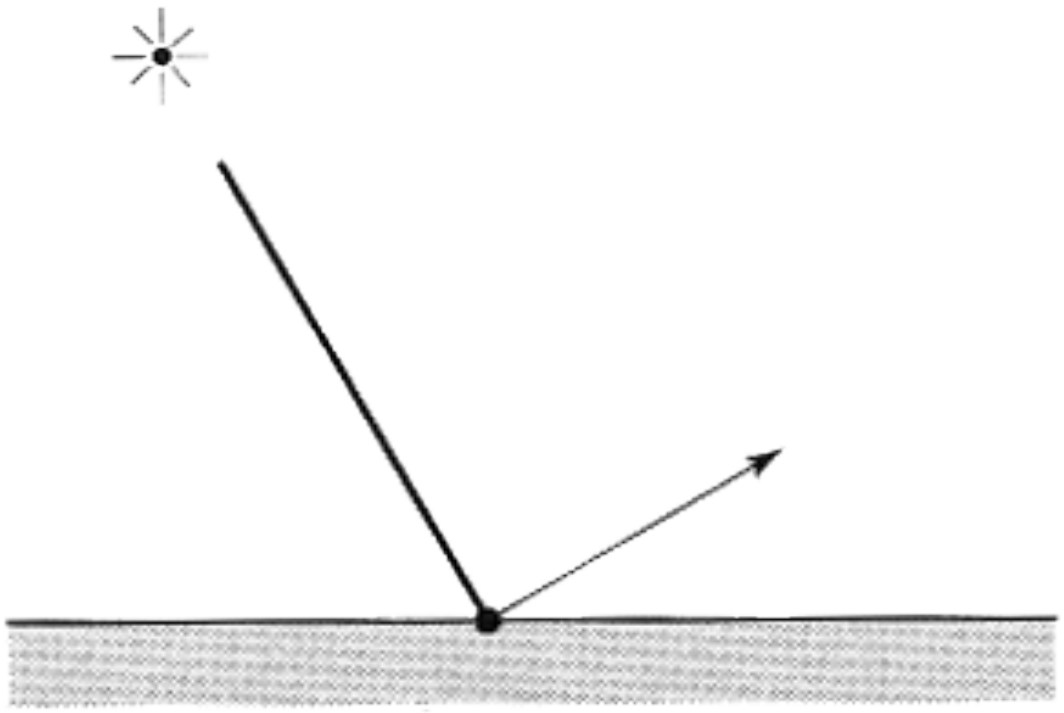
Phong-Beleuchtungsmodell (Bui-Tuong Phong), 1975

- ACHTUNG
  - Nicht verwechseln mit Phong Shading
- Am häufigsten verwendetes lokales Beleuchtungsmodell
- Empirisches Modell
  - ohne physikalische Basis
  - mit guten praktischen Resultaten
- Das Modell simuliert folgende physikalischen Reflexionsphänomene:
  - Unvollkommen spiegelnde Reflexion
  - Perfekte/vollkommene diffuse Reflexion

9.3 Lokale Beleuchtungsmodelle

Phong-Beleuchtungsmodell (Bui-Tuong Phong), 1975

- Perfekte/vollkommene spiegelnde Reflexion
  - Ein Lichtstrahl wird, ohne sich aufzustreuen, perfekt nach dem Reflexionsgesetz reflektiert
  - Oberfläche: idealer Spiegel
    - existiert in der Realität nicht

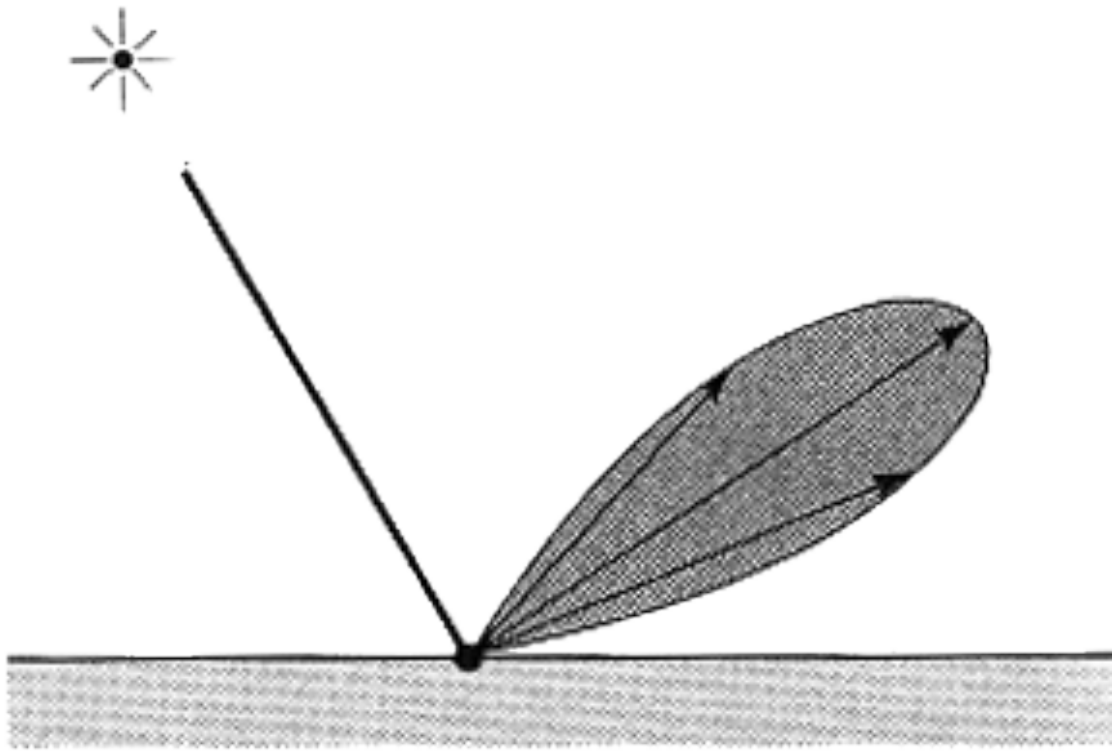


(a)

9.3 Lokale Beleuchtungsmodelle

Phong-Beleuchtungsmodell (Bui-Tuong Phong), 1975

- Unvollkommene spiegelnde Reflexion
  - Der Lichtstrahl wird bei der Reflexion „aufgespalten“
    - es entsteht ein Reflexionskonus um die ausgezeichnete Reflexionsrichtung
  - Oberfläche: unvollkommener Spiegel, raue Oberfläche
  - Oberflächenelement ist mikroskopisch aus vielen kleinen perfekten Spiegeln mit leicht unterschiedlichen Ausrichtungen zusammengesetzt

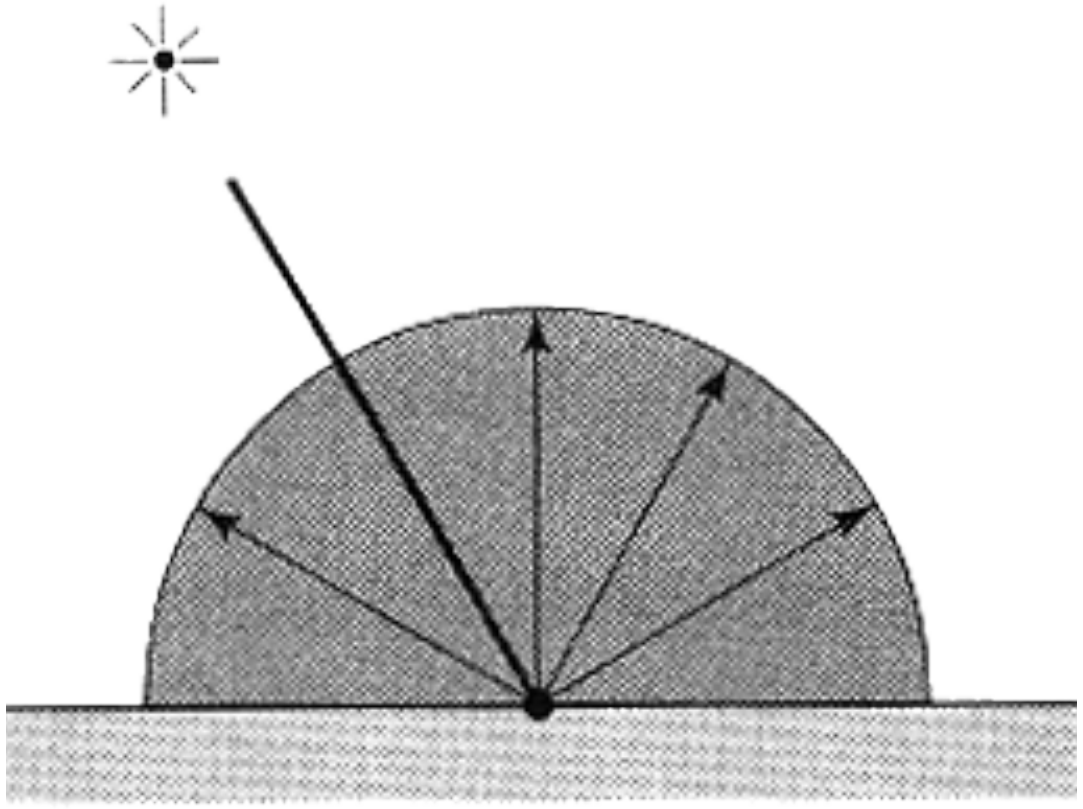


(b)

9.3 Lokale Beleuchtungsmodelle

Phong-Beleuchtungsmodell (Bui-Tuong Phong), 1975

- Perfekte/vollkommene diffuse Reflexion
  - Der Lichtstrahl wird bei der Reflexion perfekt gestreut, d.h. mit gleichmäßiger Intensität in alle Richtungen
  - Oberfläche: ideale matte Oberfläche
    - existiert in der Realität nicht
    - annäherungsweise: feine Lage Puder



(c)

9.3 Lokale Beleuchtungsmodelle

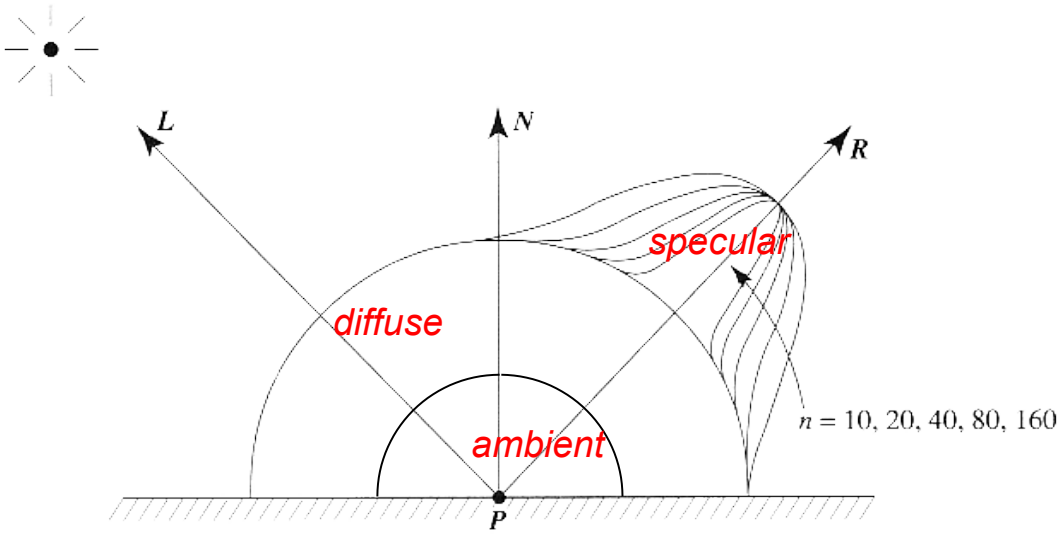
Phong-Beleuchtungsmodell (Bui-Tuong Phong), 1975

- Ambientes Licht: Hilfskonstruktion
  - Konstant gewählt
  - Simuliert die globale bzw. die indirekte Beleuchtung
  - Grundbeleuchtung notwendig
    - Einige Objekte erhalten kein Licht von Lichtquelle(n): würden in dem Modell schwarz dargestellt
    - In der Realität werden solche Objekte indirekt beleuchtet
  - Ersatz für komplexere Berechnungen

9.3 Lokale Beleuchtungsmodelle

Phong-Beleuchtungsmodell (Bui-Tuong Phong), 1975

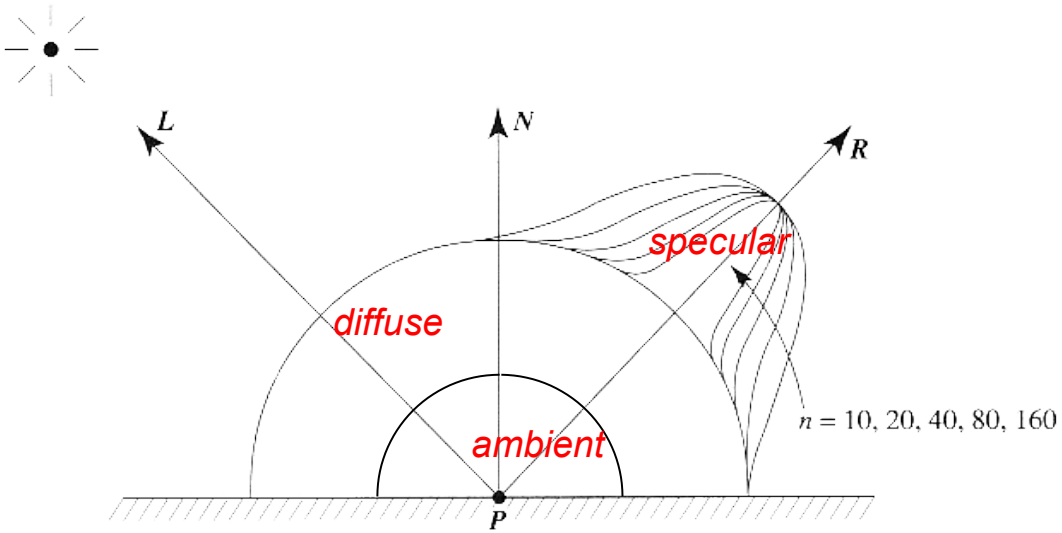
- Das reflektierte Licht (Intensität) eines Oberflächenpunktes wird dann aus den drei Reflexionskomponenten linear zusammengesetzt:
  $$I = k_s \cdot I_s + k_d \cdot I_d + k_a \cdot I_a$$
- I_s: spiegelndes Licht (specular)
- I_d: diffuses Licht
- I_a: ambientes Licht (Umgebung)



9.3 Lokale Beleuchtungsmodelle

Phong-Beleuchtungsmodell (Bui-Tuong Phong), 1975

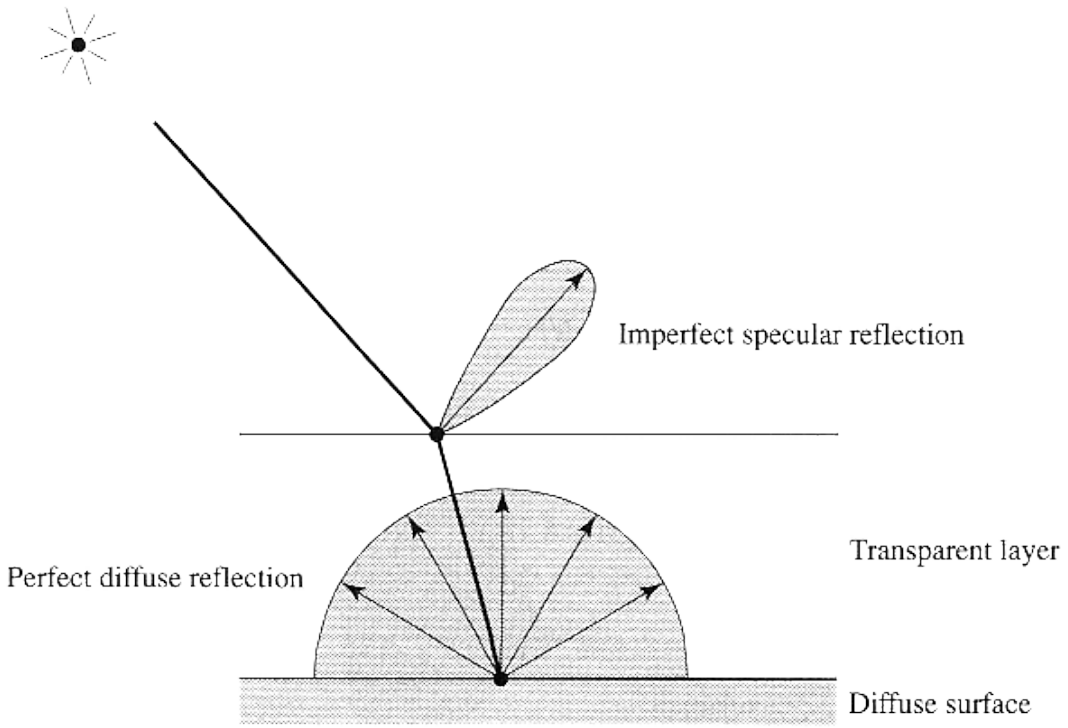
- $I = k_s \cdot I_s + k_d \cdot I_d + k_a \cdot I_a$
- Die Physik der Oberfläche wird über die Verhältnisse der einzelnen Komponenten modelliert
- Es gilt: $k_s + k_d + k_a = 1$



9.3 Lokale Beleuchtungsmodelle

Phong-Beleuchtungsmodell (Bui-Tuong Phong), 1975

- Die lineare Kombination von spiegelnder und diffuser Reflexion entspricht zum Beispiel der Physik polierter Oberflächen
  - Poliertes Holz
    - Transparente Schicht: spiegelnd
    - Oberfläche: diffus



9.3 Lokale Beleuchtungsmodelle

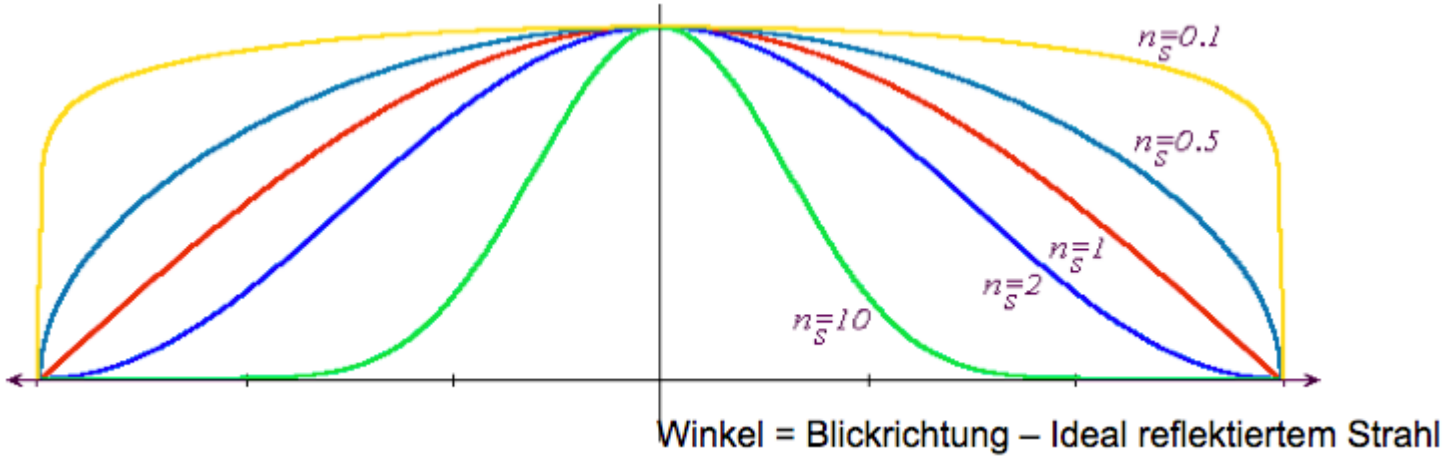
Phong-Beleuchtungsmodell (Bui-Tuong Phong), 1975

- Spiegelnde Reflexion, Term $k_s \cdot I_s$:
  - Physikalisch gesehen besteht die spiegelnde Reflexion aus einem Abbild der Lichtquelle, welches über einem Teil der Oberfläche „verschmiert“ ist: Highlight
  - Highlight kann vom Betrachter nur gesehen werden, wenn seine Betrachtungsrichtung (V) nahe der Reflexionsrichtung (R) liegt.
- Simulation durch:
  - $I_s = I_i \cdot \cos^n(\Omega) = I_i \cdot (R \circ V)^n$
  - I_i Intensität des einfallenden Lichts der Lichtquelle i
  - Ω: Winkel zwischen V und R
  - n: simuliert Perfektionsgrad der Oberfläche
    - $n \to \infty$ heißt perfekter Spiegel
      - reflektiertes Licht nur in Richtung R
      - keine Streuung

9.3 Lokale Beleuchtungsmodelle

Phong-Beleuchtungsmodell (Bui-Tuong Phong), 1975

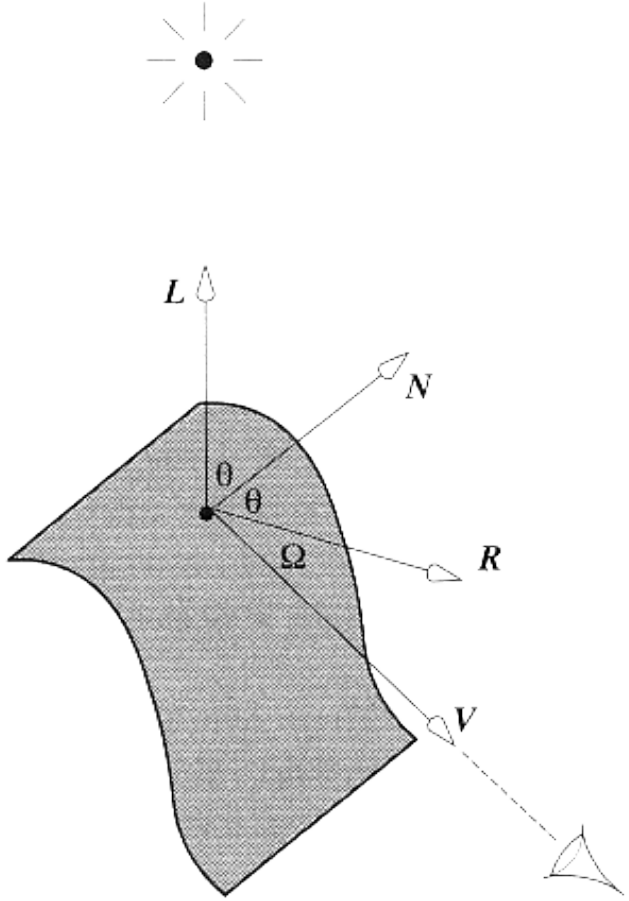
- Spiegelnde Reflexion:
  - Term wird kleiner mit zunehmenden Winkel zwischen Blickwinkel und (ideal) reflektiertem Strahl



9.3 Lokale Beleuchtungsmodelle

Phong-Beleuchtungsmodell (Bui-Tuong Phong), 1975

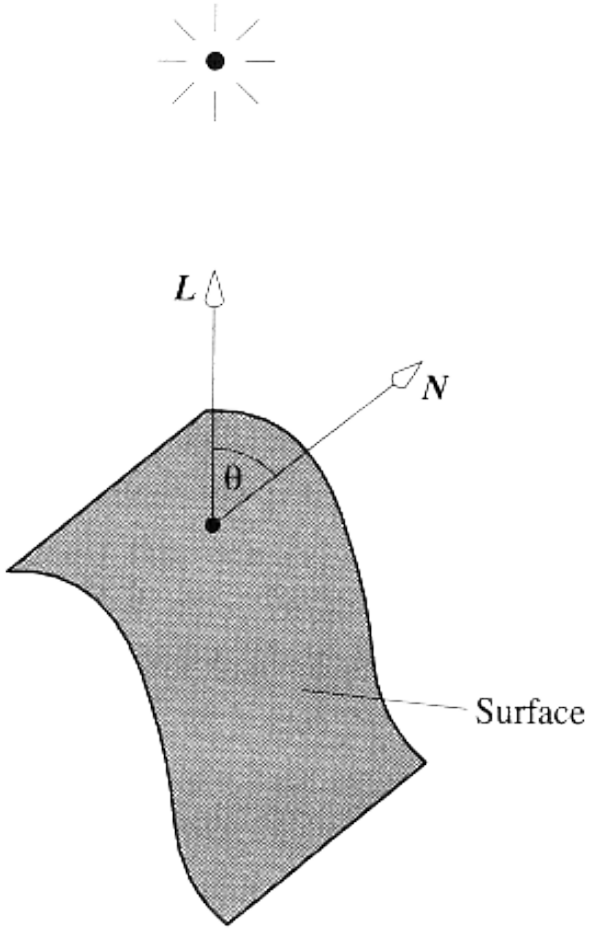
- Spiegelnde Reflexion Bemerkung
  - Für verschiedene L entsteht (bis auf seine Ausrichtung um R) immer der gleiche Reflexions-Intensitätskonus
  - Dies entspricht nicht der realen Abhängigkeit von Spiegelungen von der Ausrichtung des Lichtvektors zum Normalenvektor
  - Gravierender Mangel des Modells



9.3 Lokale Beleuchtungsmodelle

Phong-Beleuchtungsmodell (Bui-Tuong Phong), 1975

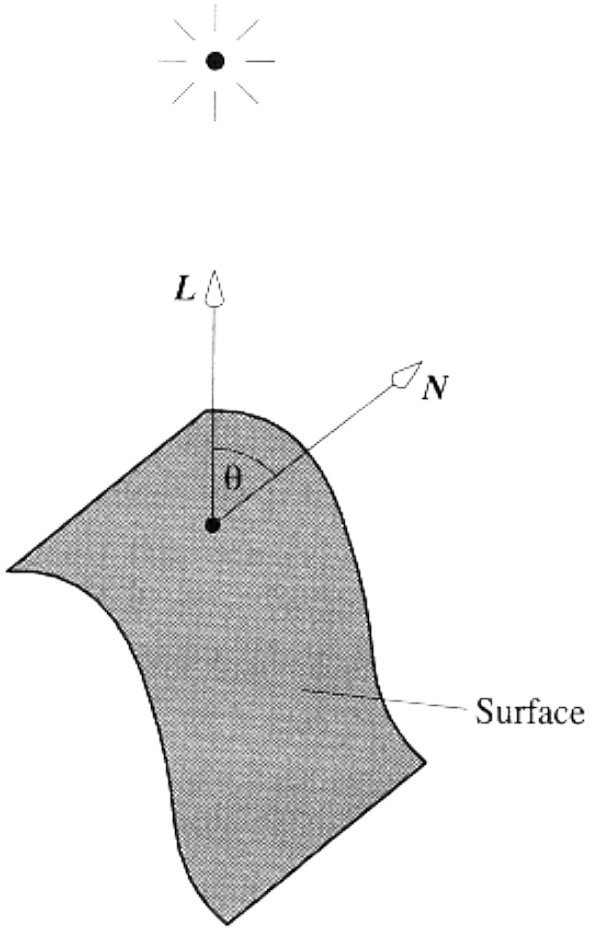
- Diffuse Reflexion, Term $k_d \cdot I_d$:
  - $I_d = I_i \cdot \cos\Theta = I_i \cdot (L \circ N)$
  - I_i: Intensität des einfallenden Lichts der Lichtquelle i
  - Θ: Winkel zwischen Punktnormale N und Lichtvektor L



9.3 Lokale Beleuchtungsmodelle

Phong-Beleuchtungsmodell (Bui-Tuong Phong), 1975

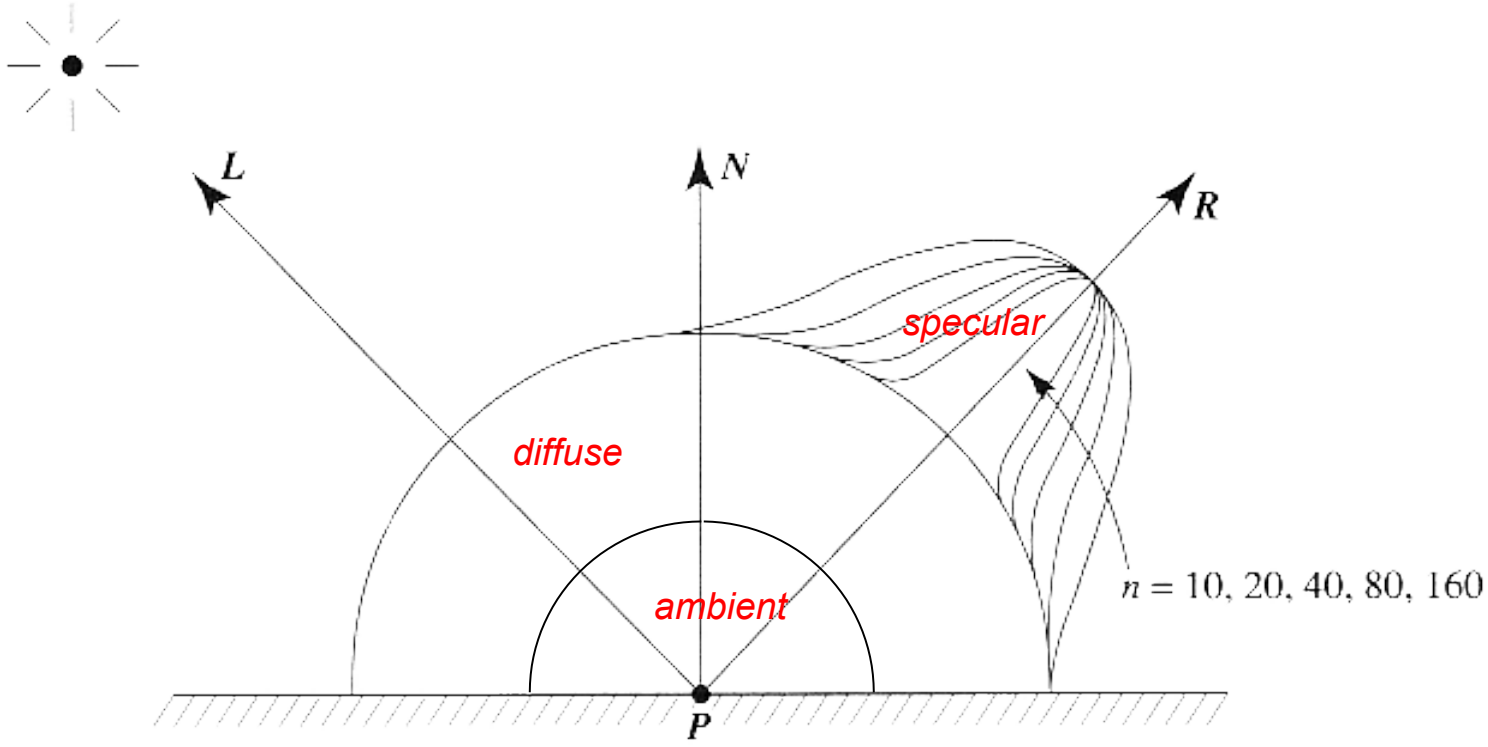
- Diffuse Reflexion
  - Die diffuse Komponente des Phong-Modells modelliert das Kosinusgesetz von Lambert
    - Bei ideal diffusen (matten) Oberflächen ist die Intensität des (in alle Richtungen gleich) reflektierten Lichtes eine Funktion des Kosinus zwischen Oberflächennormale und Lichtvektor



9.3 Lokale Beleuchtungsmodelle

Phong-Beleuchtungsmodell (Bui-Tuong Phong), 1975

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



9.3 Lokale Beleuchtungsmodelle

Phong-Beleuchtungsmodell (Bui-Tuong Phong), 1975

- Beispiel

zunehmendes k_s

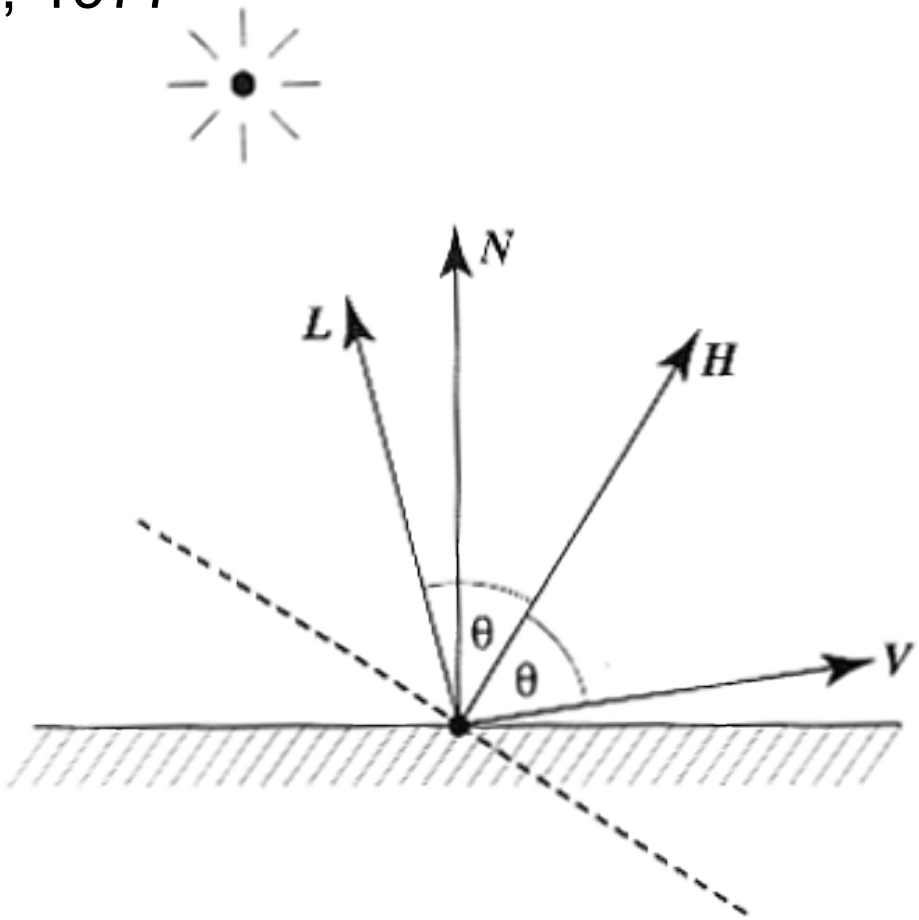
k_a konstant

zunehmendes n

9.3 Lokale Beleuchtungsmodelle

Blinn-Phong Beleuchtungsmodell (James F. Blinn), 1977

- Es kann statt dem Reflexionsvektor ein konstanter Vektor eingesetzt werden
  - $H = \dfrac{L + V}{\|L + V\|}$
  - Man betrachtet statt $R \circ V$ jetzt $H \circ V$
    - unterscheidet sich von $R \circ V$
    - verhält sich auf die gleiche Weise
  - Damit ergibt sich:
    $I = k_s \cdot I_i \cdot (H \circ V)^n + k_d \cdot I_i \cdot (L \circ N) + k_a \cdot I_a$
  - Gute Annaehrung, wenn Betrachter und Lichtquelle weit entfernt sind
  - Schneller



9.3 Lokale Beleuchtungsmodelle

Blinn-Phong Beleuchtungsmodell (James F. Blinn), 1977

- Für farbige Objekte (Lichtquellen) wird das Modell getrennt auf die Farbkomponenten r, g, b angewendet

$$
\begin{aligned}
I_r &= k_{s,r} \cdot I_i \cdot (N \circ H)^n + k_{d,r} \cdot I_i \cdot (L \circ N) + k_{a,r} \cdot I_a \\
I_g &= k_{s,g} \cdot I_i \cdot (N \circ H)^n + k_{d,g} \cdot I_i \cdot (L \circ N) + k_{a,g} \cdot I_a \\
I_b &= k_{s,b} \cdot I_i \cdot (N \circ H)^n + k_{d,b} \cdot I_i \cdot (L \circ N) + k_{a,b} \cdot I_a
\end{aligned}
$$

- $k_{d,\{r,g,b\}}$: Farbe des Objektes
- $k_{s,\{r,g,b\}}$: Farbe der Lichtquelle (weiß: $k_{s,r} = k_{s,g} = k_{s,b}$)
- $k_{a,\{r,g,b\}}$: Farbe des Umgebungslichtes

9.3 Lokale Beleuchtungsmodelle

Phong-Beleuchtungsmodell (Bui-Tuong Phong), 1975

- Gravierende Mängel des Modells
  - Gegenseitige Reflektionen und Spiegelungen der Flächen werden durch den konstanten ambienten Term nur unzureichend beschrieben
  - Objektoberflächen wirken „plastikhaft“, z. B. lässt sich kein blankes Metall modellieren
- Alternative
  - Physikalisch basierte lokale Beleuchtungsmodelle
    - Simulation der BRDF (Reflexionsfunktion)
  - Mapping-Verfahren
  - …

9.3 Lokale Beleuchtungsmodelle

BRDF

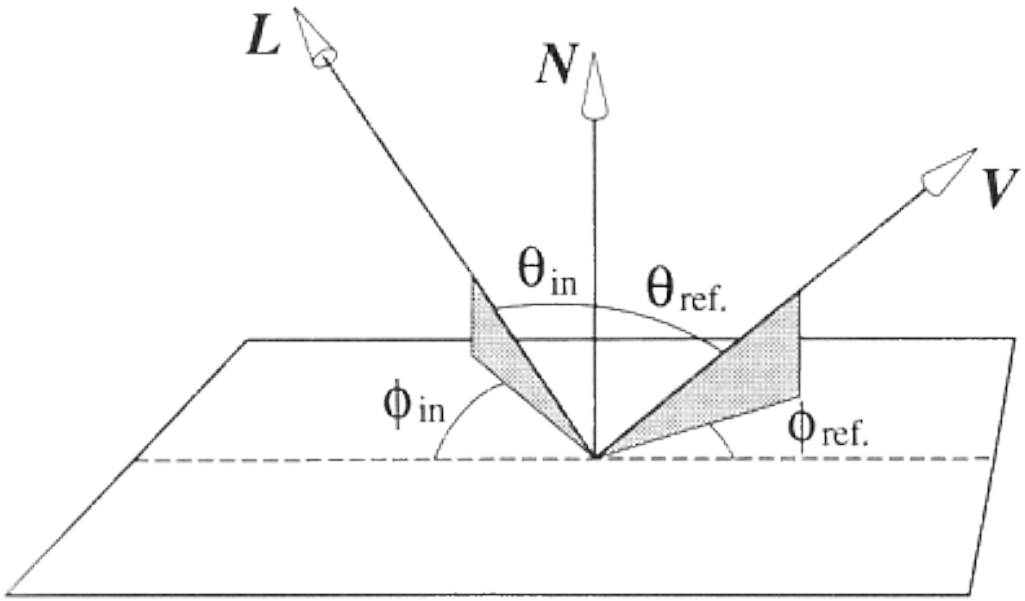
- Bi-directional Reflection Distribution Function: BRDF
  - Allgemein wird das von einem Punkt einer Oberfläche reflektierte Licht durch eine BRDF beschrieben
  - Dies betont insbesondere die Abhängigkeit des in einer beliebigen Richtung reflektierten Lichts von der Richtung des einfallenden Lichts

9.3 Lokale Beleuchtungsmodelle

BRDF

- Sind die Richtungen von L und V gegeben, so wird der Zusammenhang zwischen den Intensitäten durch folgende Formel beschrieben

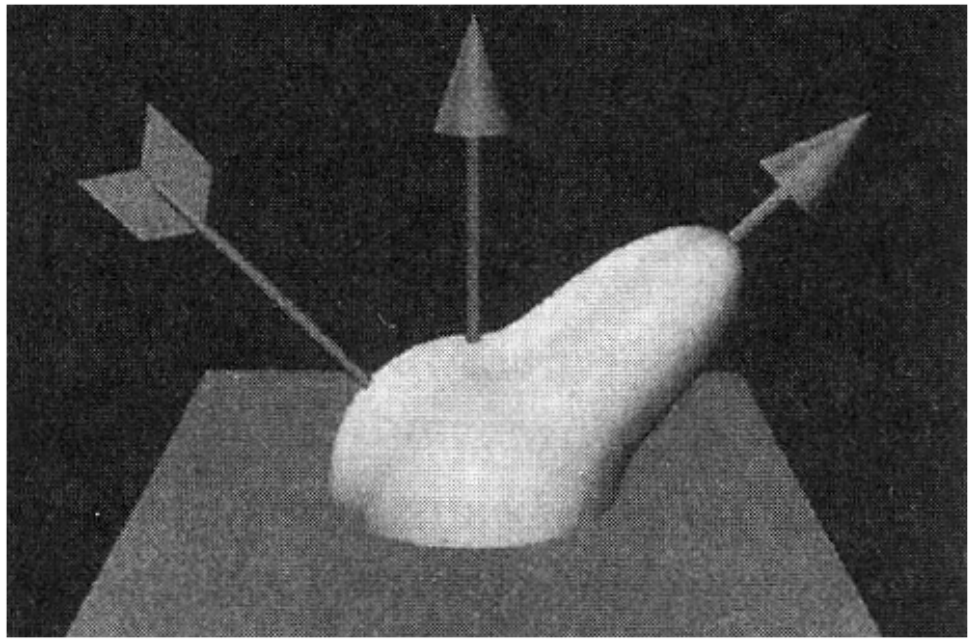
$$
\begin{aligned}
BRDF &= f\left(\Theta_{in}, \phi_{in}, \Theta_{ref}, \phi_{ref}\right) \\
&= f(L, V)
\end{aligned}
$$

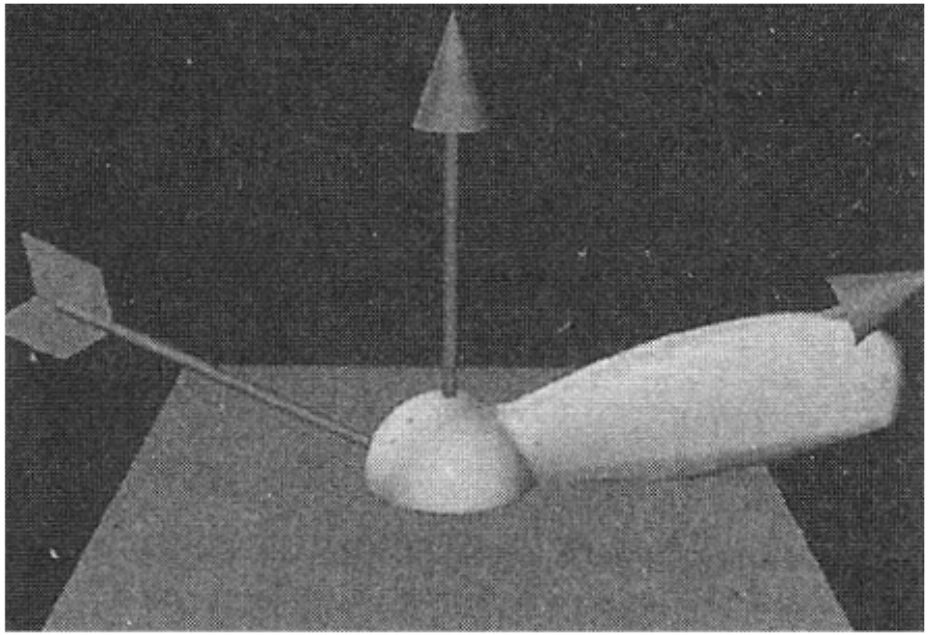


9.3 Lokale Beleuchtungsmodelle

BRDF

- Darstellung von (nach Blinn (1977)) erzeugten BRDFs für zwei verschiedene Richtungen des einfallenden Lichtes



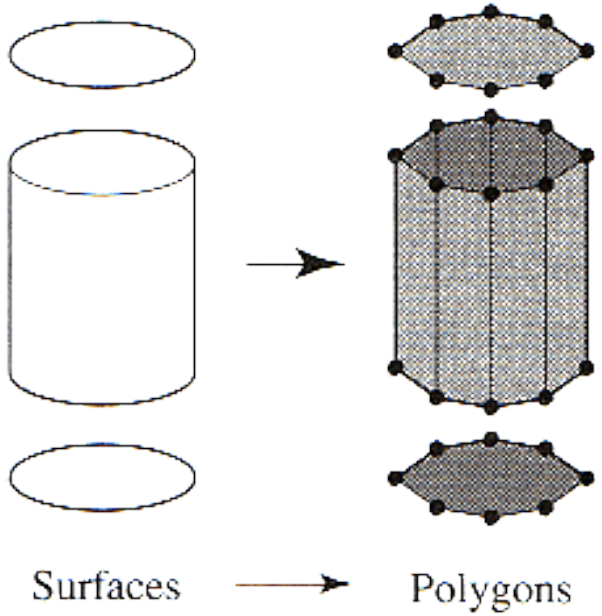


9.3 Lokale Beleuchtungsmodelle

- Nachteile „rein“ lokaler Beleuchtungsmodelle
  - Spiegeln Idealfall eines einzelnen von einer einzigen Punktlichtquelle beleuchteten Objektes in der Szene wider
  - Betrachten ausschließlich direkte Beleuchtung (bis auf Hilfskonstruktionen)
  - Interaktion mit anderen Objekten nicht modelliert (d. h. keine indirekte Beleuchtung, kein Schattenwurf!)
- Besser: globale Beleuchtungsverfahren

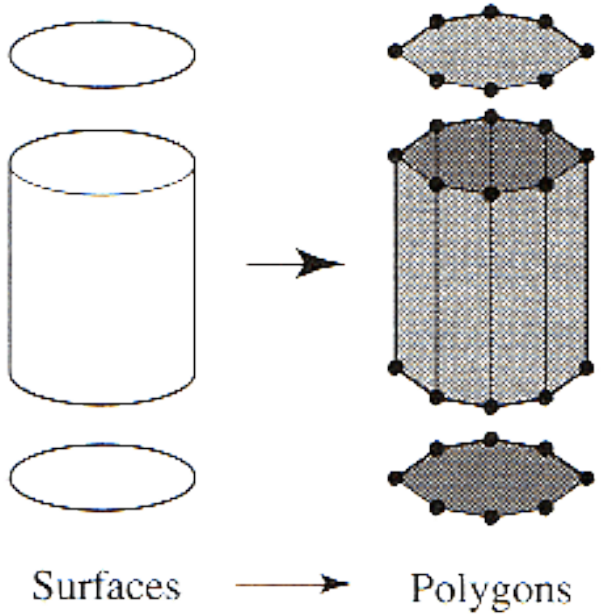
9.4 Schattierung

- Gegeben:
  - Polygonale Repräsentation eines Objekts
  - Punkt P auf der Oberfläche des Objekts
- Gesucht:
  - Farbe der Oberfläche am Punkt P
- Probleme:
  - Glattheit an den Kanten zwischen zwei Polygonen
  - Glattheit an den gemeinsamen Eckpunkten mehrerer Polygone



9.4 Schattierung

- Genauer: Bestimmung der Vektoren, welche für das Beleuchtungsmodell benötigt werden
  → Schattierungsmodell (Shading Model)



9.4 Schattierung

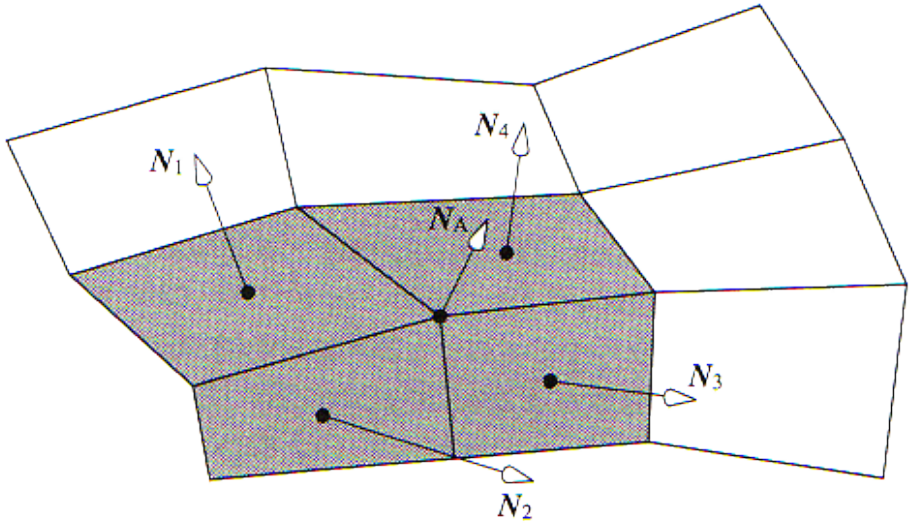
Varianten

- Konstante Farbe pro Polygon
  - Flat/uniform Shading
- Interpolation
  - Der Farben: Gouraud-Shading
  - Der Normalen: Phong-Shading

9.4 Schattierung

Flat/Uniform Shading

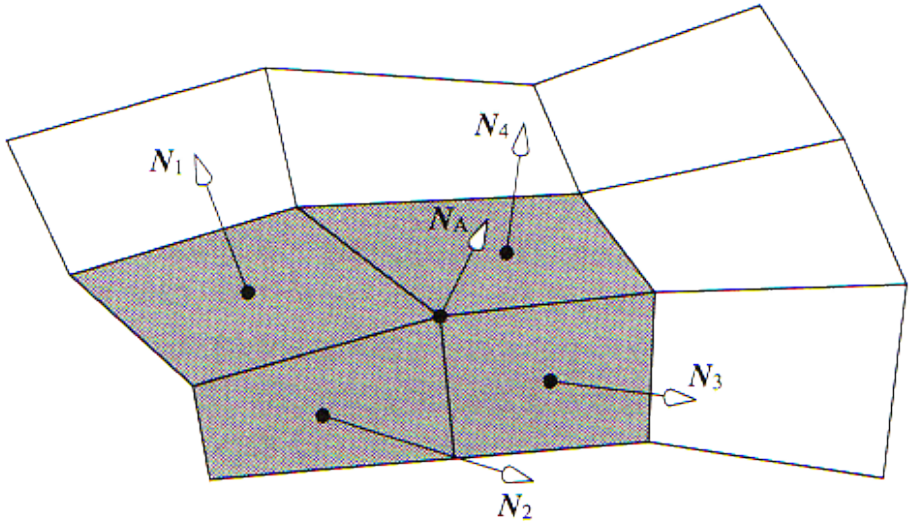
- Grundlage der Berechnung ist die Polygon- oder Oberflächennormale im Objektraum (N_1, N_2, N_3, N_4)
- Pro Polygon / Facette wird das verwendete Beleuchtungsmodell genau einmal in einem ausgewählten Oberflächenpunkt ausgewertet
- Die dort ermittelte Lichtintensität wird auch allen weiteren Punkten der Polygonoberfläche zugewiesen



9.4 Schattierung

Flat/Uniform Shading

- Als ausgewählte Punkte werden die Polygonschwerpunkte oder Polygoneckpunkte gewählt
- Voraussetzung: Planarität des Polygons
  - Für Dreiecke immer erfüllt

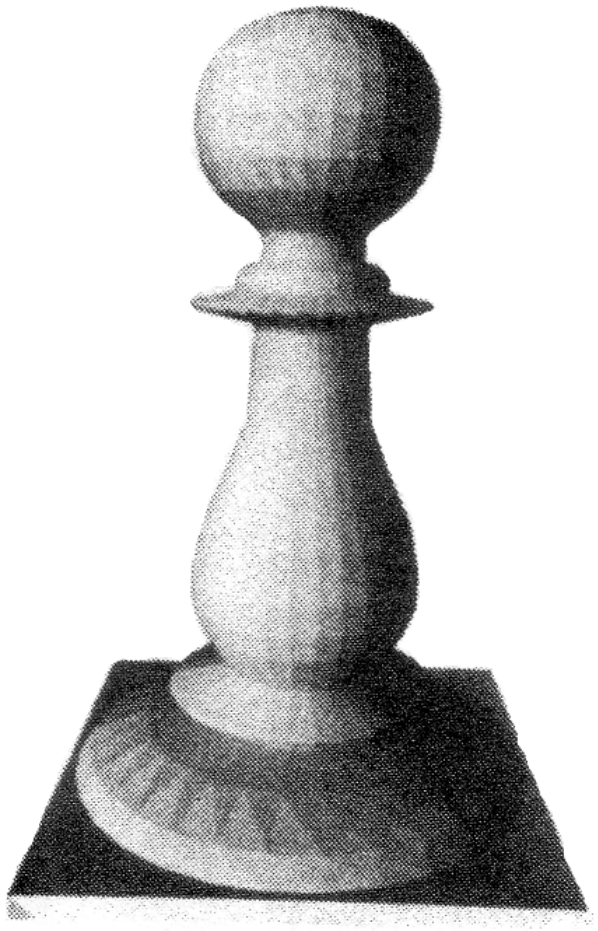


9.4 Schattierung

Flat/Uniform Shading

- Vorteile
  - Einfaches, kostengünstiges Verfahren
  - Keine Interpolation

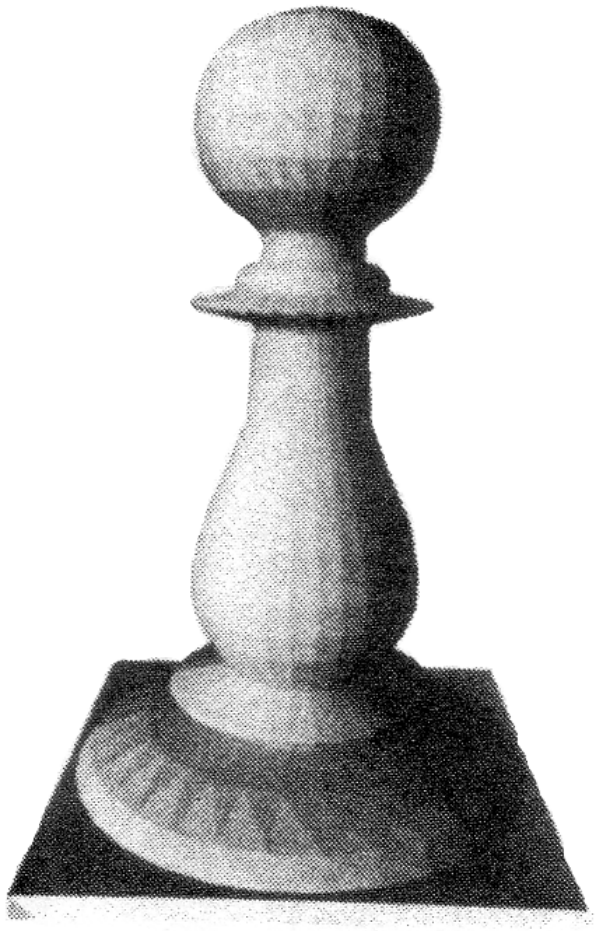
- Nachteile
  - Unstetiger Intensitätsverlauf über die Polygonkanten
  - Kanten in Polygonnetzen bleiben bei der Darstellung sichtbar
  - Objekte werden facettiert dargestellt



9.4 Schattierung

Flat/Uniform Shading

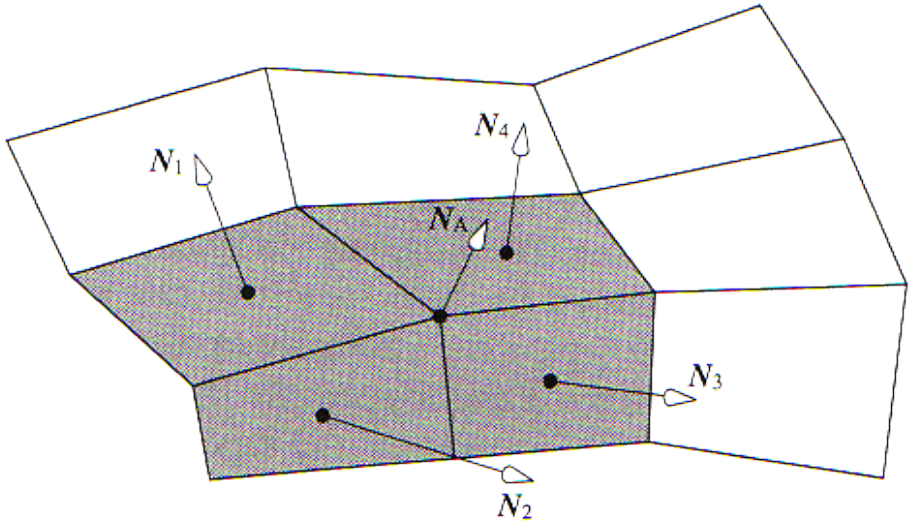
- Mögliche Anwendung
  - Visualisierung der polygonalen Auflösung für Qualitätsdarstellung



9.4 Schattierung

Gouraud- und Phong-Shading

- Grundlage der Berechnung sind die Eckpunktnormalen in den gemeinsamen Polygoneckpunkten (N_A)
- Eine Eckpunktnormale wird als (gewichtetes) Mittel der Polygonnormalen aller angrenzenden Polygone mit dem entsprechenden gemeinsamen Eckpunkt berechnet
- Normalisierung notwendig



9.4 Schattierung

Gouraud- und Phong-Shading

- Die Farbe an einem Punkt auf der Oberfläche ergibt sich dann durch Interpolation
  - Komplex für beliebige Polygone
  - Einfach für Dreiecke
- Ziel
  - Glätten (virtueller) Kanten zwischen einzelnen Polygonfacetten (Das Polygonnetz stellt die Approximation einer gekrümmten Oberfläche dar)

9.4 Schattierung

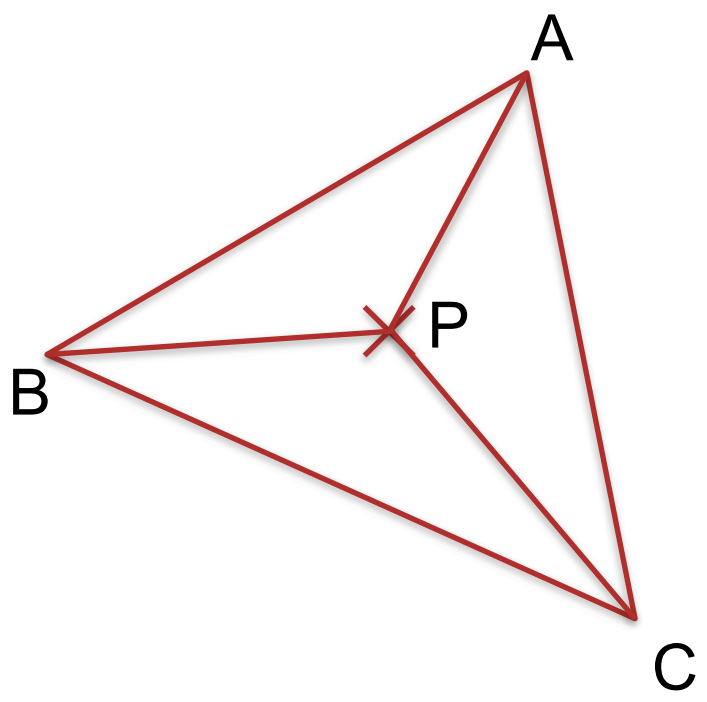
Gouraud-Shading

- Die Auswertung des Beleuchtungsmodells erfolgt ausschließlich in den Polygoneckpunkten unter Ausnutzung der Eckpunktnormalen
- Intensitätswerte am Punkt P werden interpoliert
  - Verwendung von baryzentrischen Koordinaten
- Gegeben: Baryzentrische Koordinaten (α, β, γ) eines Punktes P bezüglich eines Dreiecks $(A,\ B,\ C)$
- Berechnung der Intensität $I_P = \alpha \cdot I_A + \beta \cdot I_B + \gamma \cdot I_C$
- I_P: Intensität Punkt P
- I_A: Intensität Punkt A
- I_B: Intensität Punkt B
- I_C: Intensität Punkt C

9.4 Schattierung

Baryzentrische Koordinaten

- Gegeben: Dreieck $(A,\ B,\ C)$
- Gesucht: Koordinaten von P bezüglich des Dreiecks $(A,\ B,\ C)$
- Ansatz:
  $P = \alpha \cdot A + \beta \cdot B + \gamma \cdot C$
- Nebenbedingung:
  $\alpha + \beta + \gamma = 1$
  (Normalisierung)



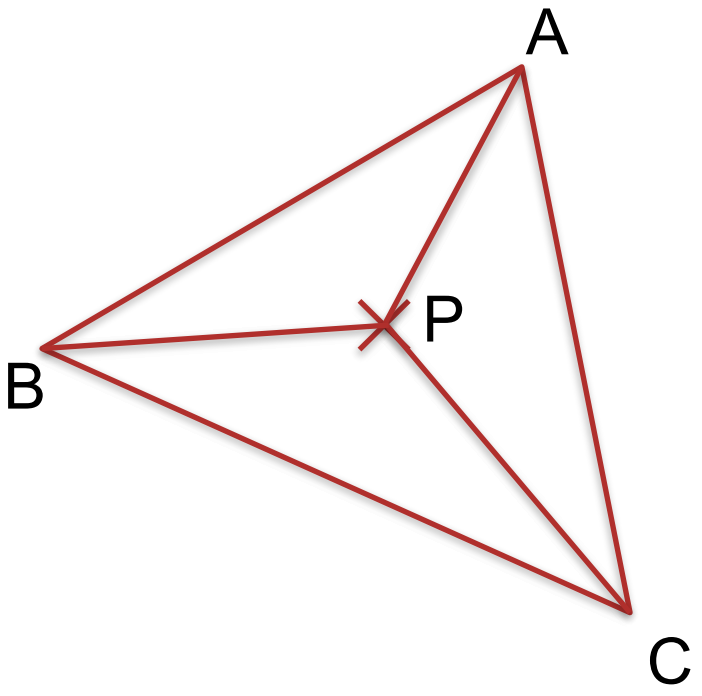
9.4 Schattierung

Baryzentrische Koordinaten

- $P = \alpha \cdot A + \beta \cdot B + \gamma \cdot C$
- Folgerungen:

$$A = (1,0,0)$$
$$B = (0,1,0)$$
$$C = (0,0,1)$$

  - $\alpha = 0, \beta + \gamma = 1$: Kante $B - C$
  - $\beta = 0, \alpha + \gamma = 1$: Kante $A - C$
  - $\gamma = 0, \alpha + \beta = 1$: Kante $A - B$
  - $0 \leq \alpha, \beta, \gamma \leq 1$: P liegt innerhalb
  - Sonst: P liegt außerhalb



9.4 Schattierung

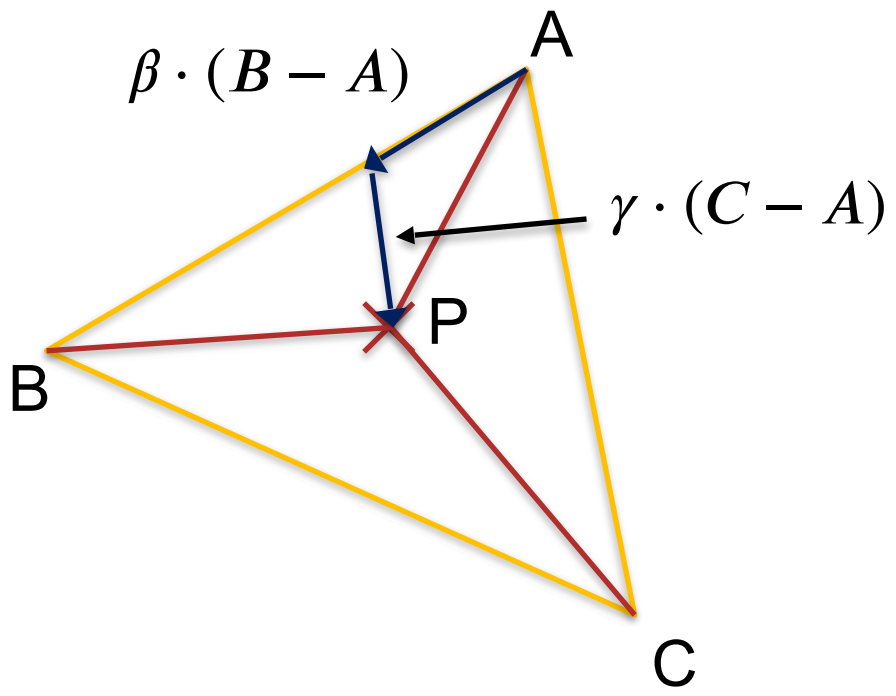
Baryzentrische Koordinaten

- $P = \alpha \cdot A + \beta \cdot B + \gamma \cdot C$

$$\begin{aligned} P &= \alpha \cdot A + \beta \cdot B + \gamma \cdot C \\ &= \left(1 - \beta - \gamma\right) \cdot A + \beta \cdot B + \gamma \cdot C \\ &= A + \beta \cdot (B - A) + \gamma \cdot (C - A) \end{aligned}$$

- Berechnung von $\left(\alpha, \beta, \gamma\right)$

$$\alpha = \frac{\Delta(P, B, C)}{\Delta(A, B, C)}$$

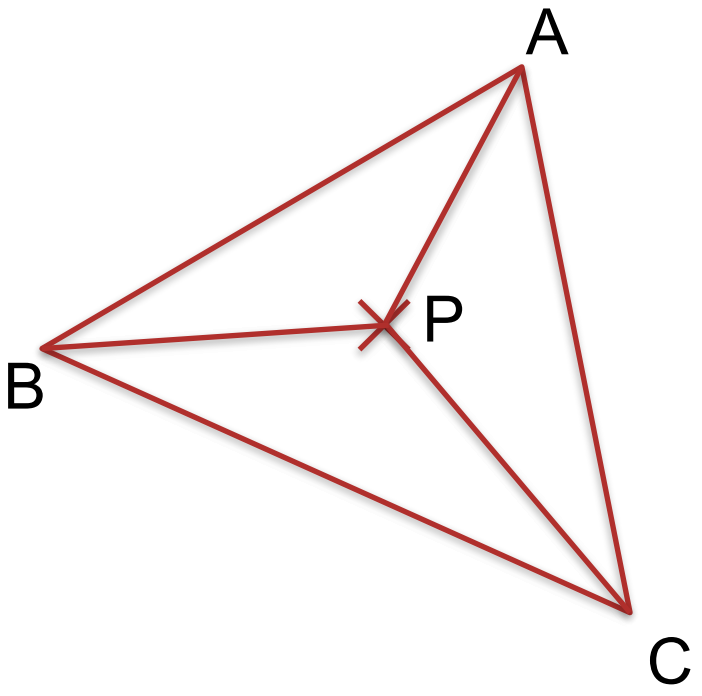


$$\Delta(P, B, C) = ((B - P) \times (C - P)) \circ \left(\frac{(B - A) \times (C - A)}{\left\|(B - A) \times (C - A)\right\|} \right)$$

9.4 Schattierung

Baryzentrische Koordinaten

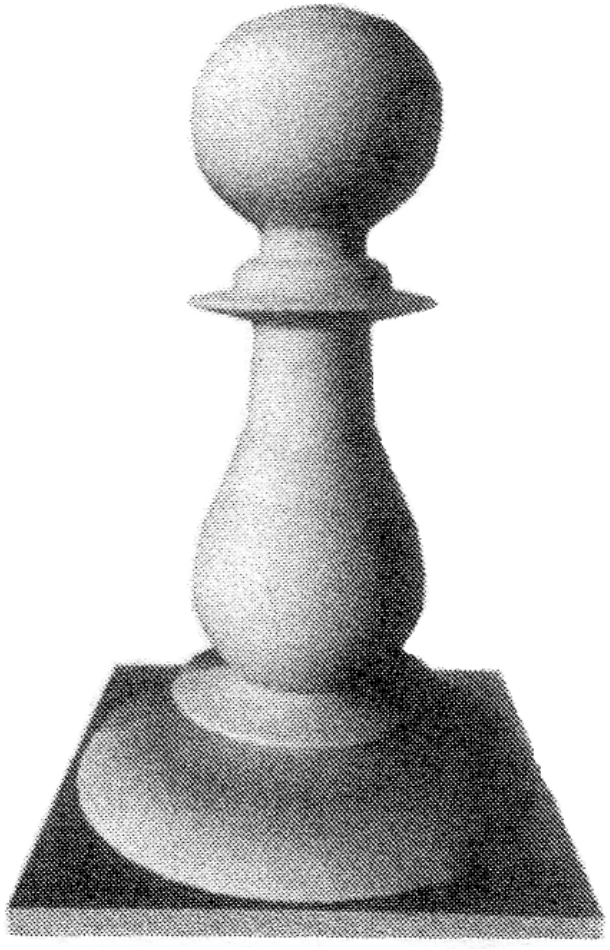
- Effizientere Berechnung und Algorithmen:
  [Christer Ericson, Real Time Collision Detection, Morgan Kaufman – Elsevier, 2005]



9.4 Schattierung

Gouraud-Shading

- Nachteil
  - Highlights werden durch Abtastfehler „verschluckt“ oder erscheinen polygonförmig (statt rund)
- Gängig: Kombination von Gouraud-Shading und Beleuchtungsmodell mit ausschließlich diffuser Reflexionskomponente
- War lange Zeit Standard in der Graphikhardware



9.4 Schattierung

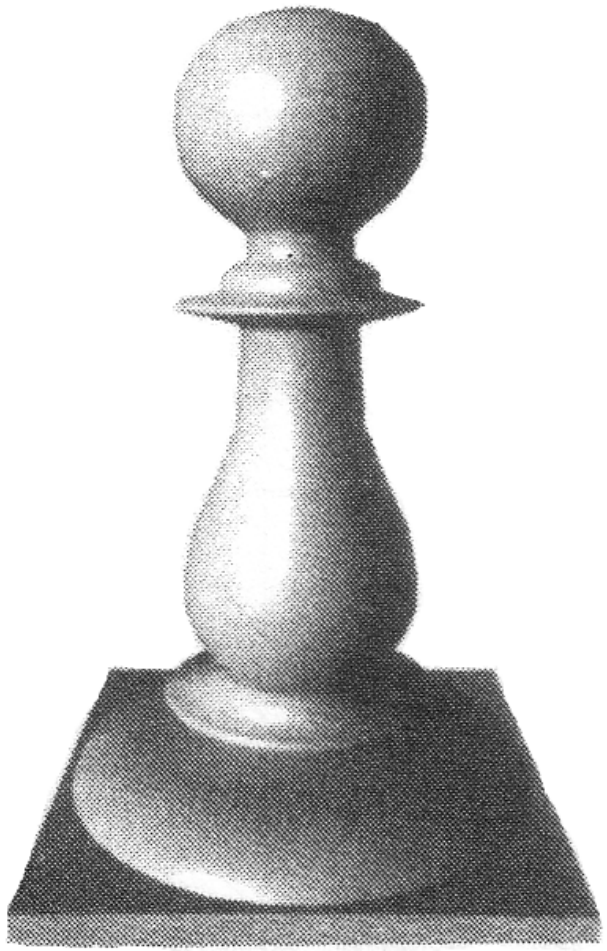
Phong-Shading

- Die Auswertung des Beleuchtungsmodells erfolgt direkt am Punkt P
- Die Oberflächennormale am Punkt P wird durch Interpolation der Stützpunktnormalen ermittelt
- Normalisierung nötig
- Auswertung des Beleuchtungsmodells für jedes Pixel nötig

9.4 Schattierung

Phong-Shading

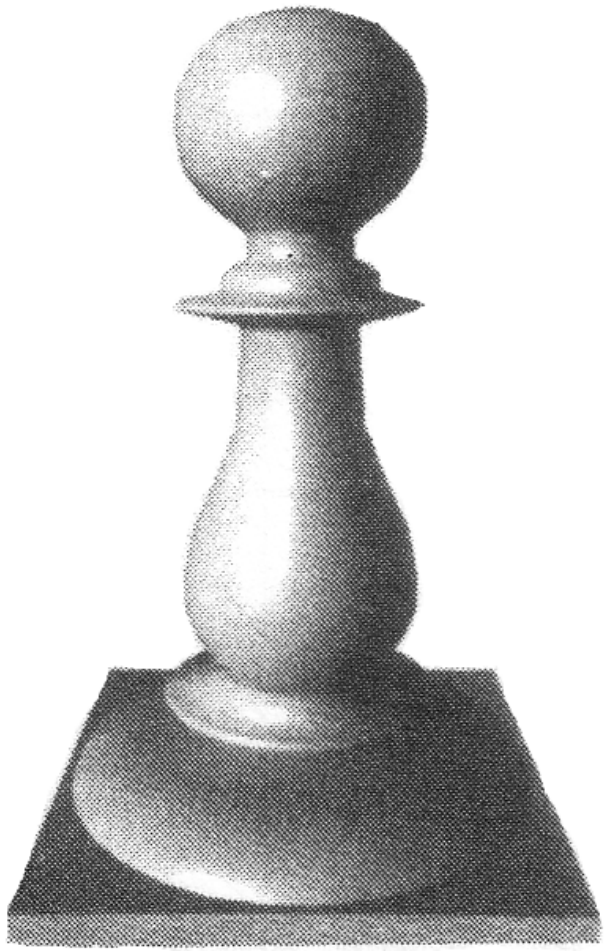
- Vorteile:
  - Intensitätsverlauf über die Polygonkanten ist stetig und glatt
  - Gute Annäherung des Erscheinungsbilds der realen gekrümmten Oberfläche durch interpolierte Normalen
  - Highlights werden adäquat dargestellt



9.4 Schattierung

Phong-Shading

- Nachteile
  - Rechenaufwändiges Verfahren
- Wird (mittlerweile) von der Graphikhardware unterstützt



9.4 Schattierung

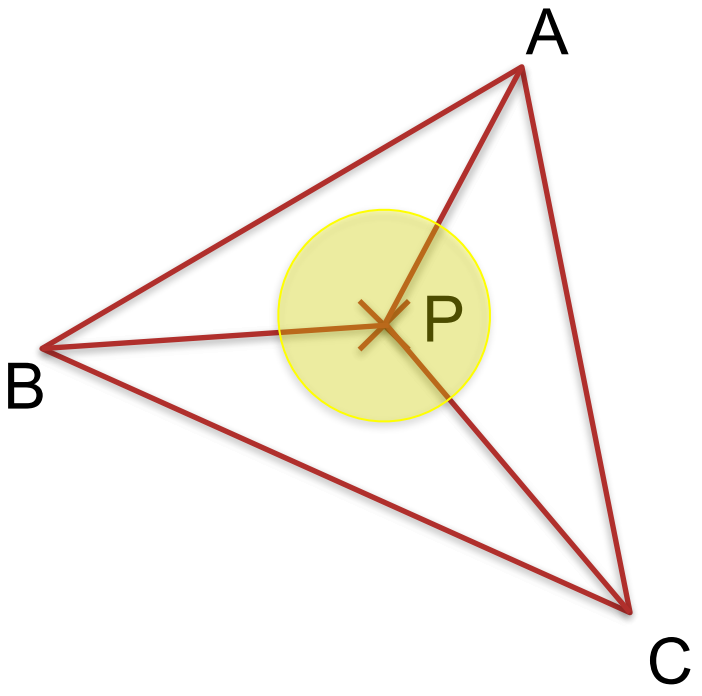
Vergleich

- Gouraud-Shading
  - Ermittle Intensitäten an den Eckpunkten des Dreiecks
    $\rightarrow$ Intensitäten in A, B, C
  - Interpoliere Intensitäten
    $\rightarrow$ Intensität in P
- Phong-Shading
  - Interpoliere Normalen der Eckpunkte des Dreiecks
    $\rightarrow$ Normale in P
  - Ermittle Intensität in P

9.4 Schattierung

Vergleich

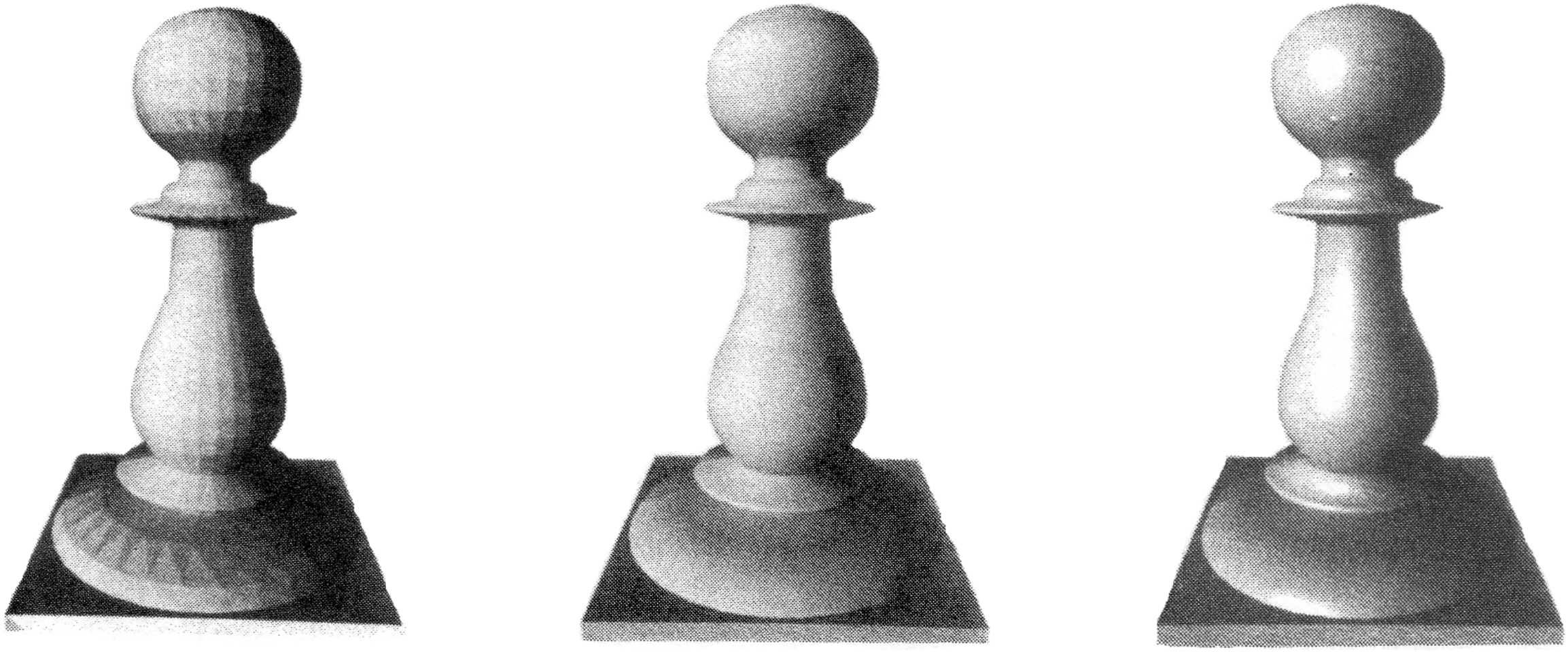
- Gouraud-Shading
  - Punkt P ist nur dann beleuchtet, wenn mindestens einer der Eckpunkte beleuchtet ist
    → Spotlights funktionieren nur über den Eckpunkten, aber nicht im Innern des Dreiecks



- Phong-Shading
  - hat dieses Problem nicht, da die Intensität direkt in P berechnet wird

9.4 Schattierung

Vergleich aller drei Verfahren



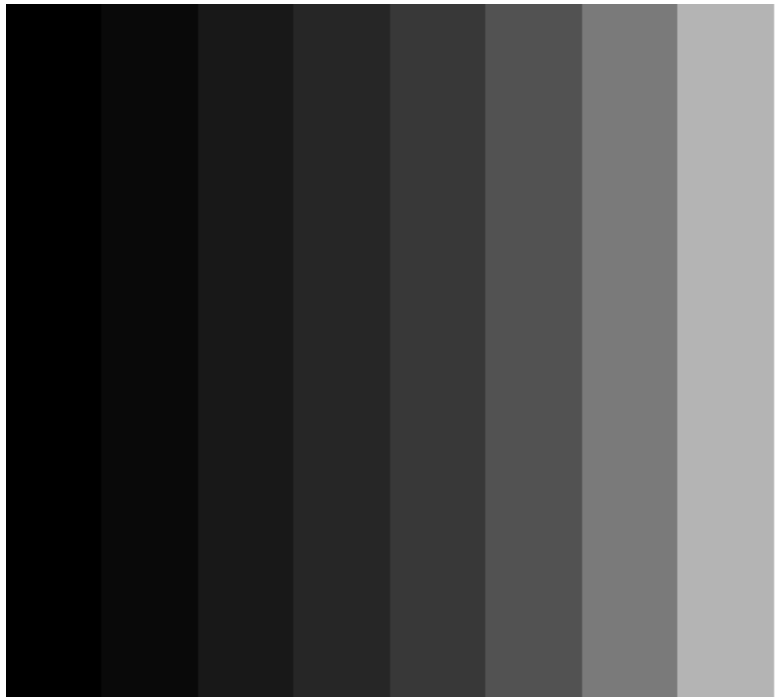
9.4 Schattierung

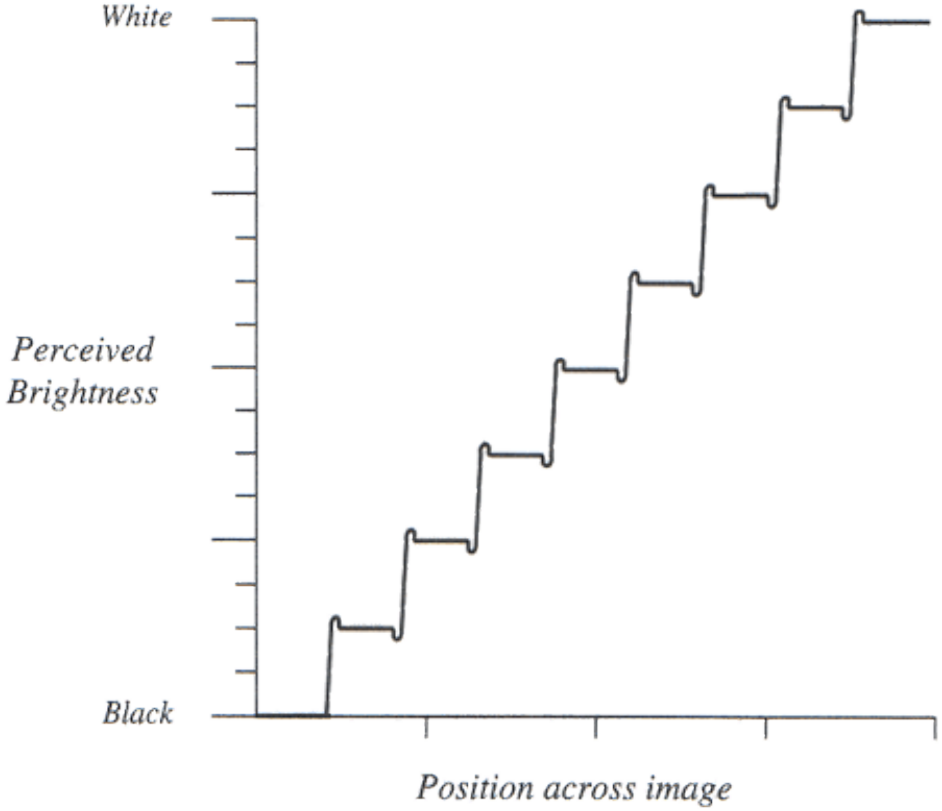
Mach Band-Effekt

- Interaktion der Lichtrezeptoren im Auge betont Intensitätsänderungen
- Dieser low-level Mechanismus der Kantenbetonung bei Intensitätsübergängen verhilft unserer visuellen Wahrnehmung zu einer automatischen Konturenschärfe

9.4 Schattierung

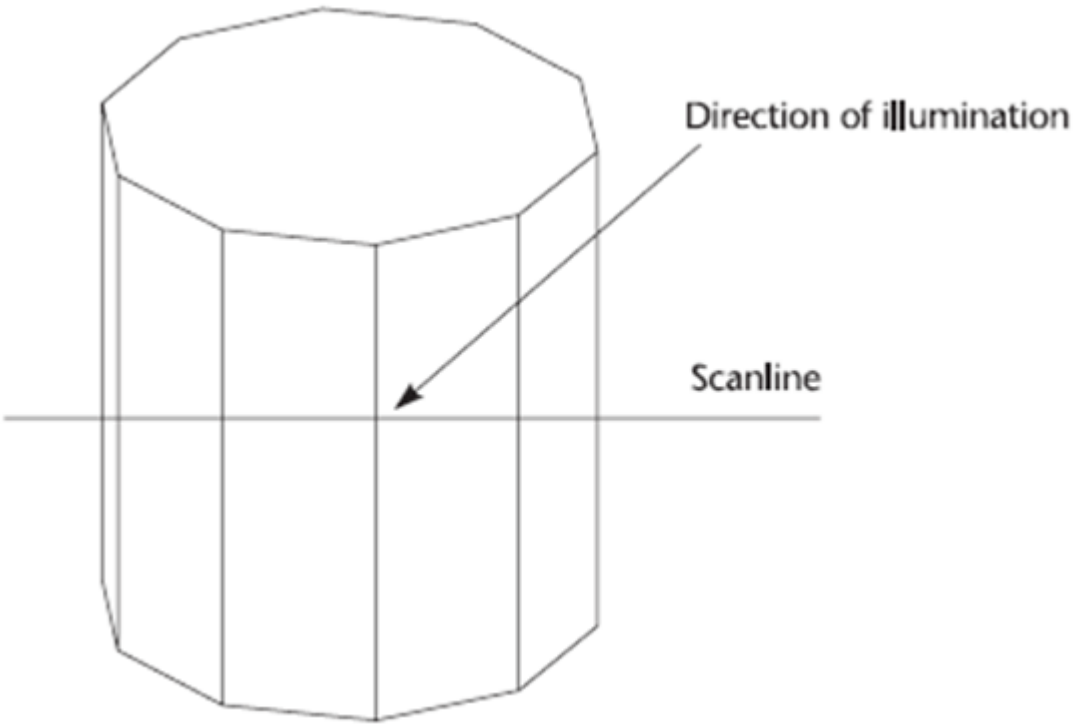
Mach Band-Effekt

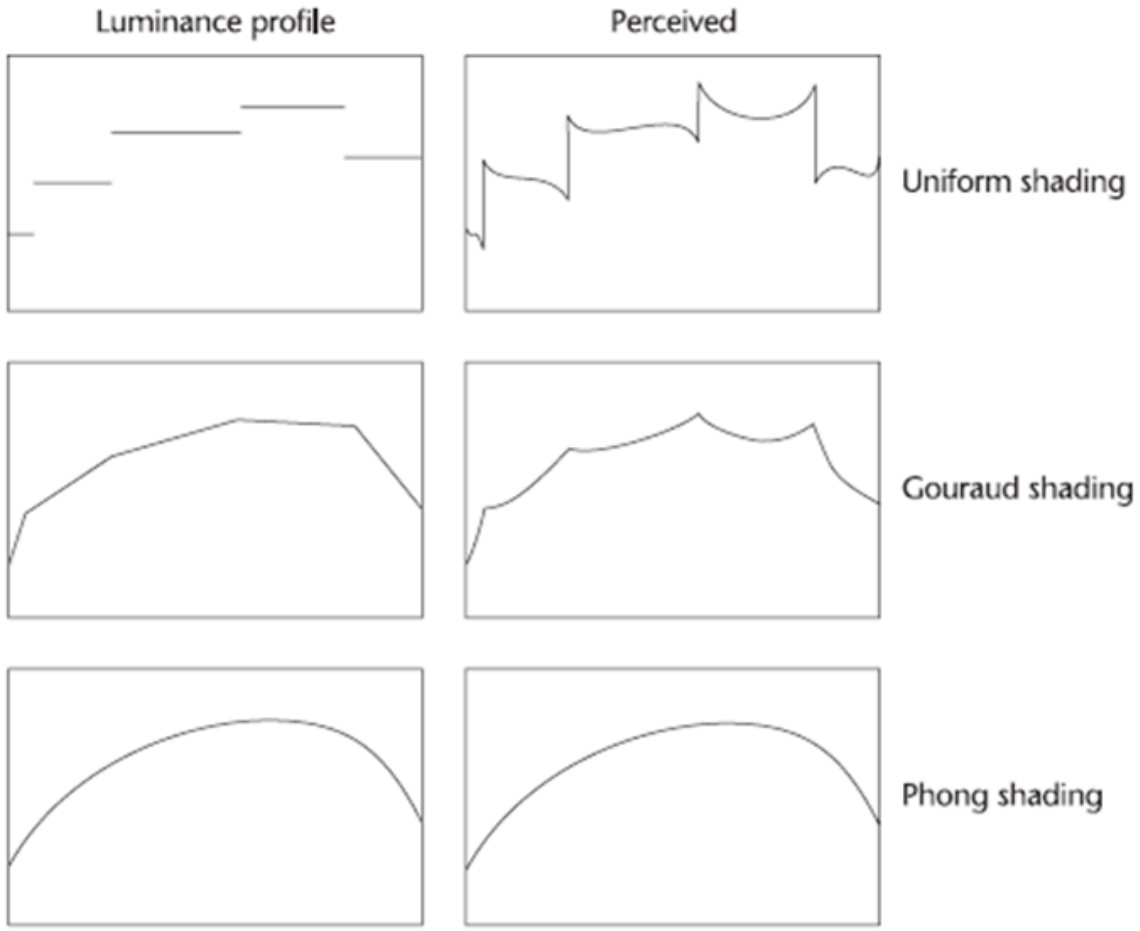




9.4 Schattierung

Mach Band-Effekt & Shading





9.4 Schattierung

Mach Band-Effekt & Shading

- Flat/Uniform Shading
  - unstetige Intensitätswechsel
  - *sehr starke* Mach Band-Effekte

- Gouraud-Shading
  - stetige Intensitätswechsel
  - abhängig von Polygonalisierung *starke* Mach Band-Effekte

- Phong-Shading
  - glatte Intensitätswechsel
  - *reduzierte* Mach Band-Effekte

9.5 Globale Beleuchtung

- Für photorealistische(re) Darstellung ist globale Beleuchtung notwendig
  - Physikalisch basierte Modellierung der Lichtausbreitung
  - Beachtet auch Wechselwirkung zwischen Objekten

9.5 Globale Beleuchtung

- Lokal, nur diffus

- Lokal, diffus und spiegelnd

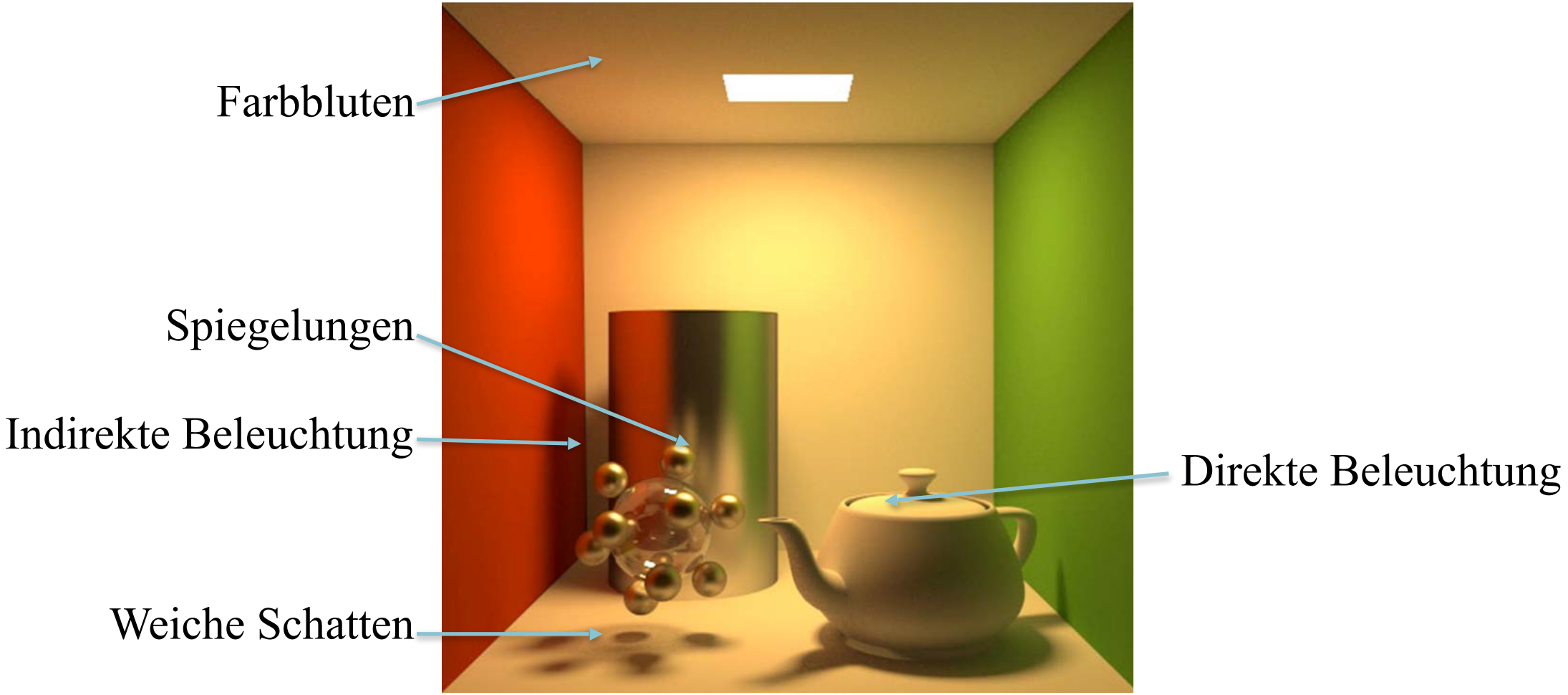
9.5 Globale Beleuchtung

- Global
  - Mehrfachreflexion
  - Weiche Schatten

- Lokal, diffus und spiegelnd

9.5 Globale Beleuchtung

Beleuchtungsphänomene



9.5 Globale Beleuchtung

Beleuchtungsphänomene

- Kaustiken

9.5 Globale Beleuchtung

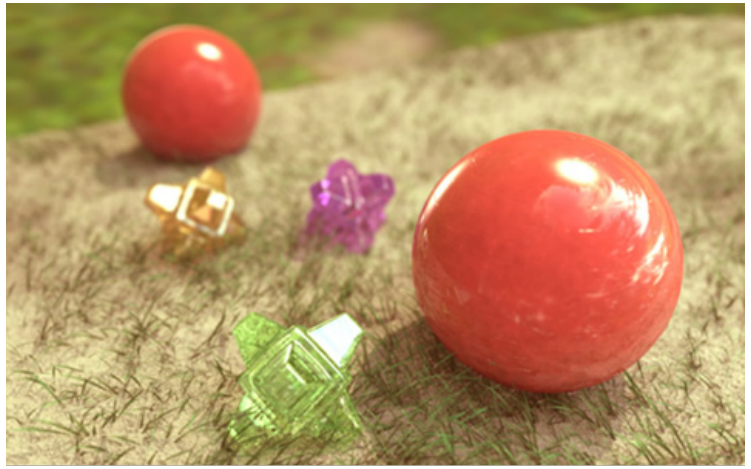
- Ray-Tracing
  - Whitted, 1979
  - Modelliert Lichtausbreitung nach Gesetzen der idealen Spiegelung und Brechung
  - Für spiegelnde und transparente Oberflächen geeignet
  - Blickrichtungsabhängig
- Radiosity-Verfahren
  - Goral & Greenberg, 1984
  - Lichtausbreitung basierend auf Energieerhaltung
  - Vorberechnung aller Interaktionen des Lichts mit Objekten der Szene
  - Diffuse Reflexion
  - Unabhängig von der Blickrichtung

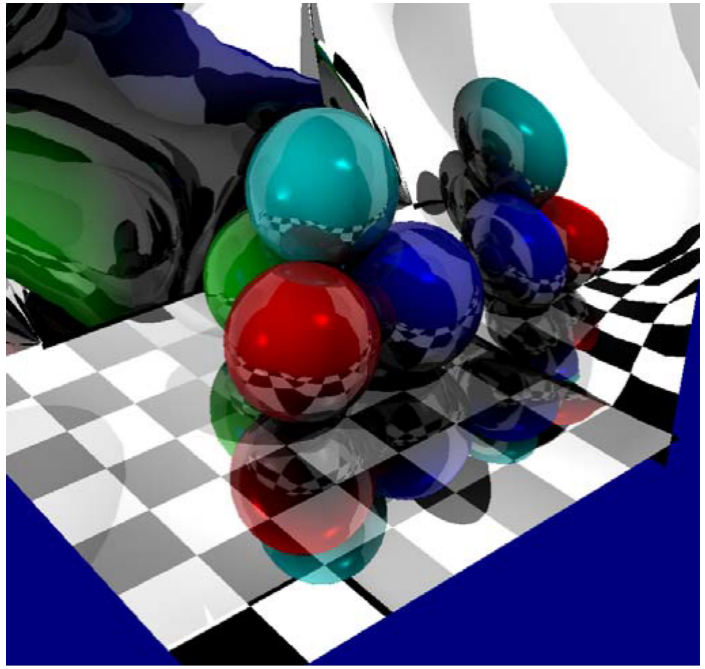
9.5 Globale Beleuchtung

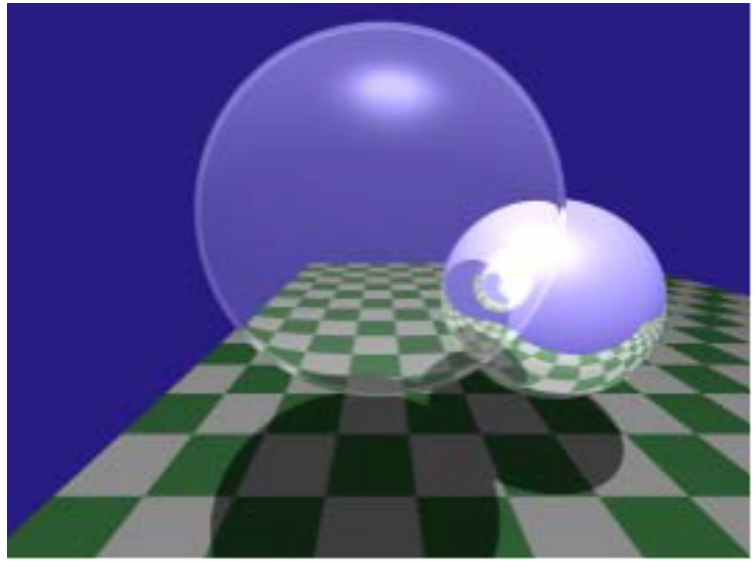
Ray-Tracing

- Besonders geeignet für Szenen mit
  - hohem spiegelnden
  - hohem transparenten

  Flächenanteil



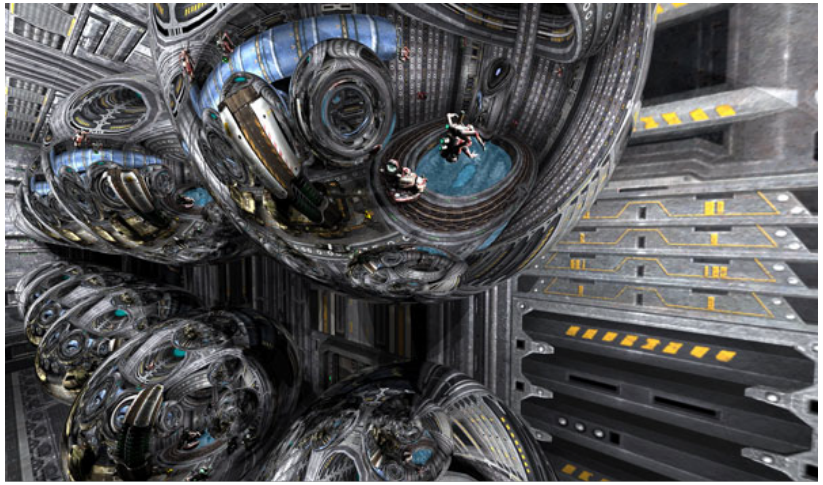




9.5 Globale Beleuchtung

Ray-Tracing

- Weitere Beispiele



9.5 Globale Beleuchtung

Ray-Tracing

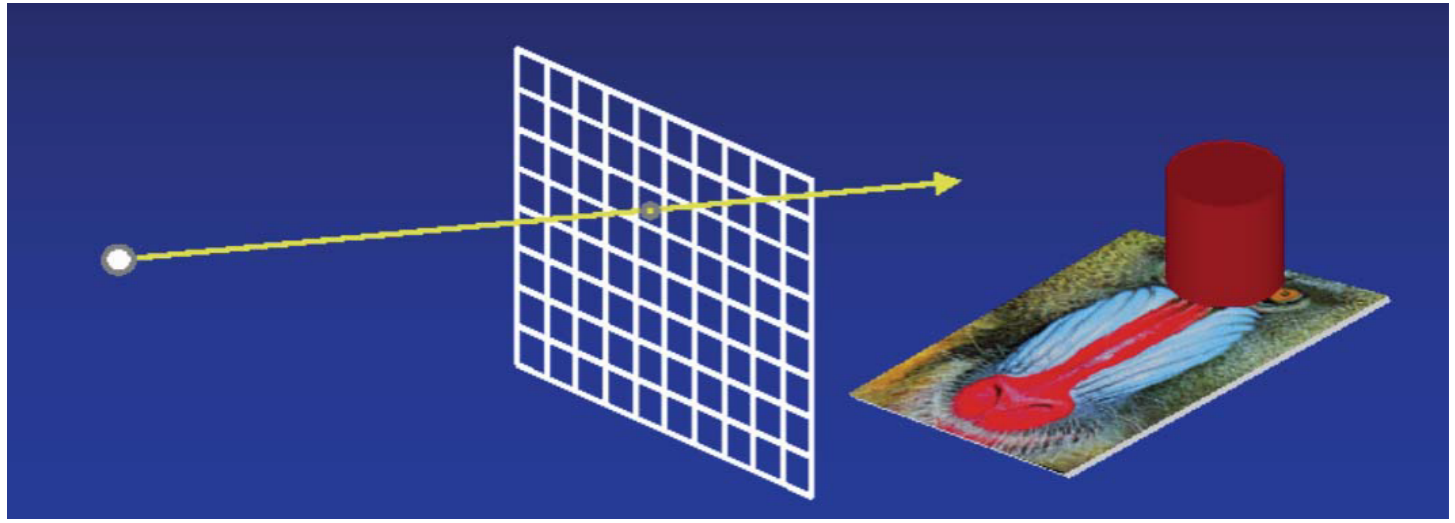
- Grundprinzip des rekursiven Raytracing
  - Verfolgung der Lichtstrahlen von der Quelle bis zum Auge
  - Aber: Nur wenige Strahlen erreichen das Auge
- Umkehrung des Verfahrens (Reziprozität der Reflexion)
  - Vom Augpunkt wird durch jedes Pixel des Bildschirms ein Strahl in die Szene gesendet
  - Bei Schnittpunkten mit Objekten werden
    - ideal reflektierte
    - ideal gebrochene

    Strahlen weiterverfolgt
  - Rekursive Auswertung des lokalen Beleuchtungsmodells (Schattierung)

9.5 Globale Beleuchtung

Ray-Tracing

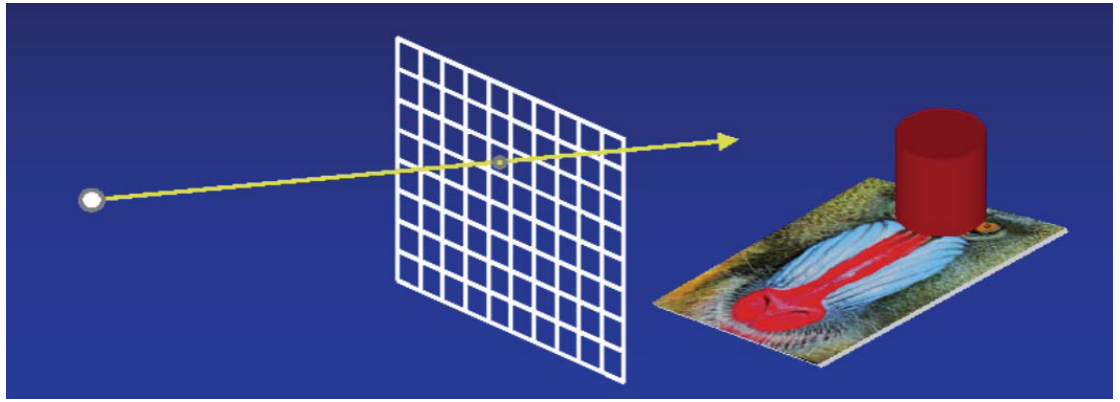
- Vom Augpunkt werden Strahlen durch die Pixel in die Szene geschickt

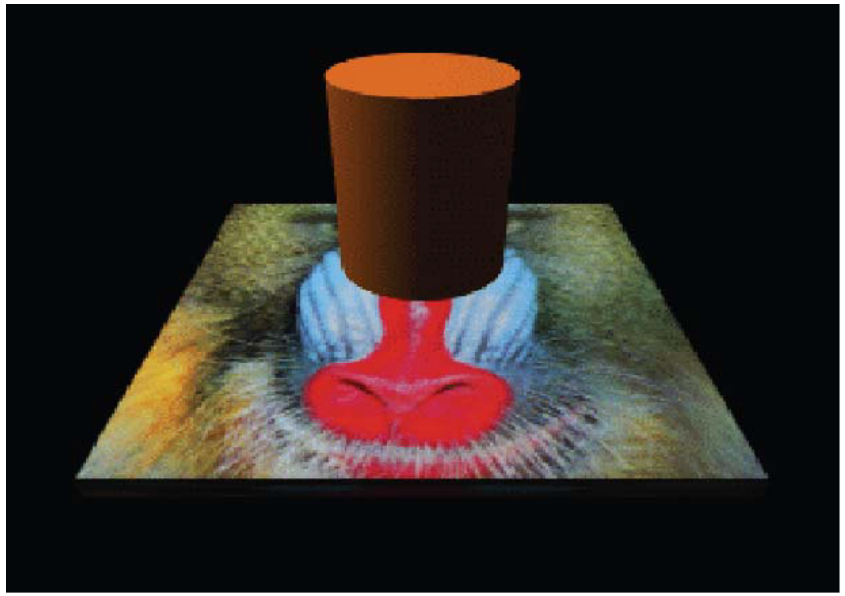


9.5 Globale Beleuchtung

Ray-Tracing

- Schneidet ein Strahl mehr als ein Objekt, wird das nächstliegende gezeichnet
- Sonst erscheint die Hintergrundfarbe

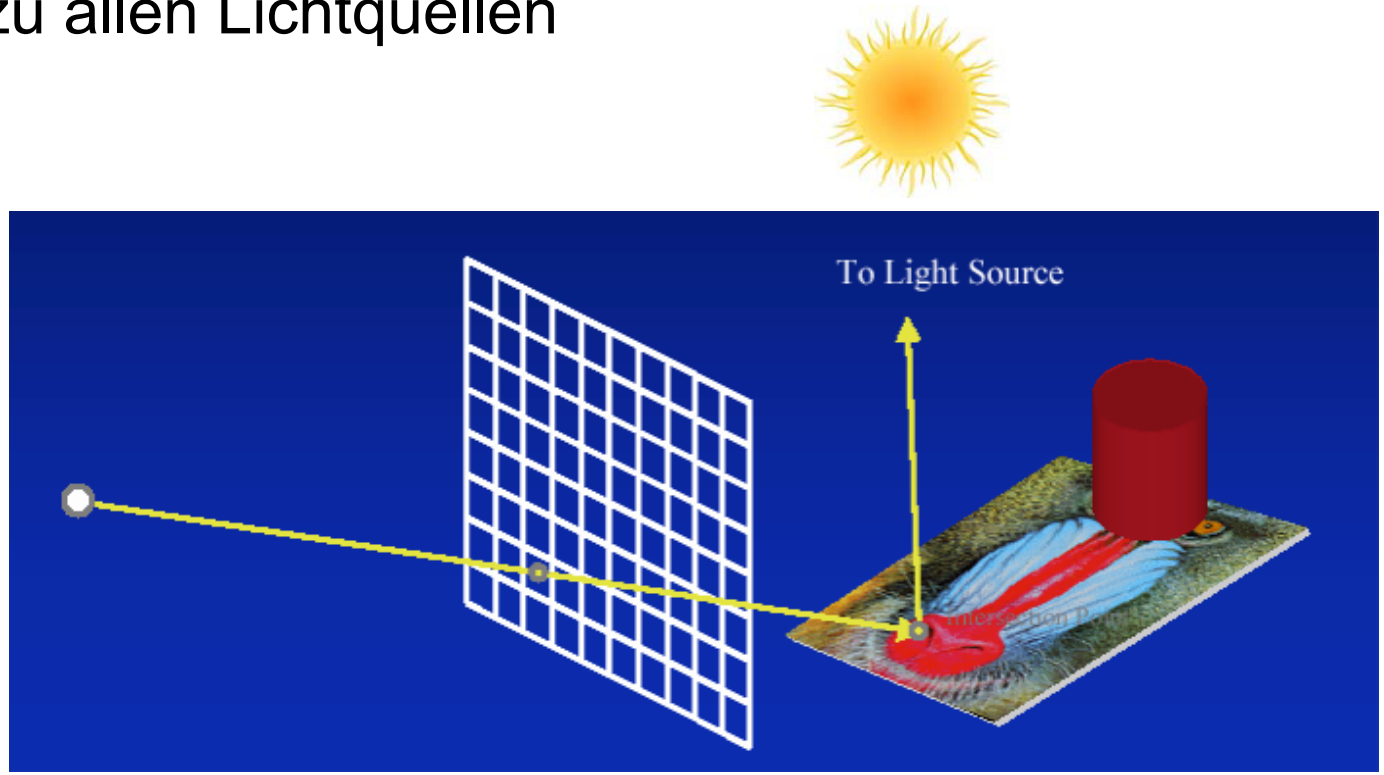




9.5 Globale Beleuchtung

Ray-Tracing

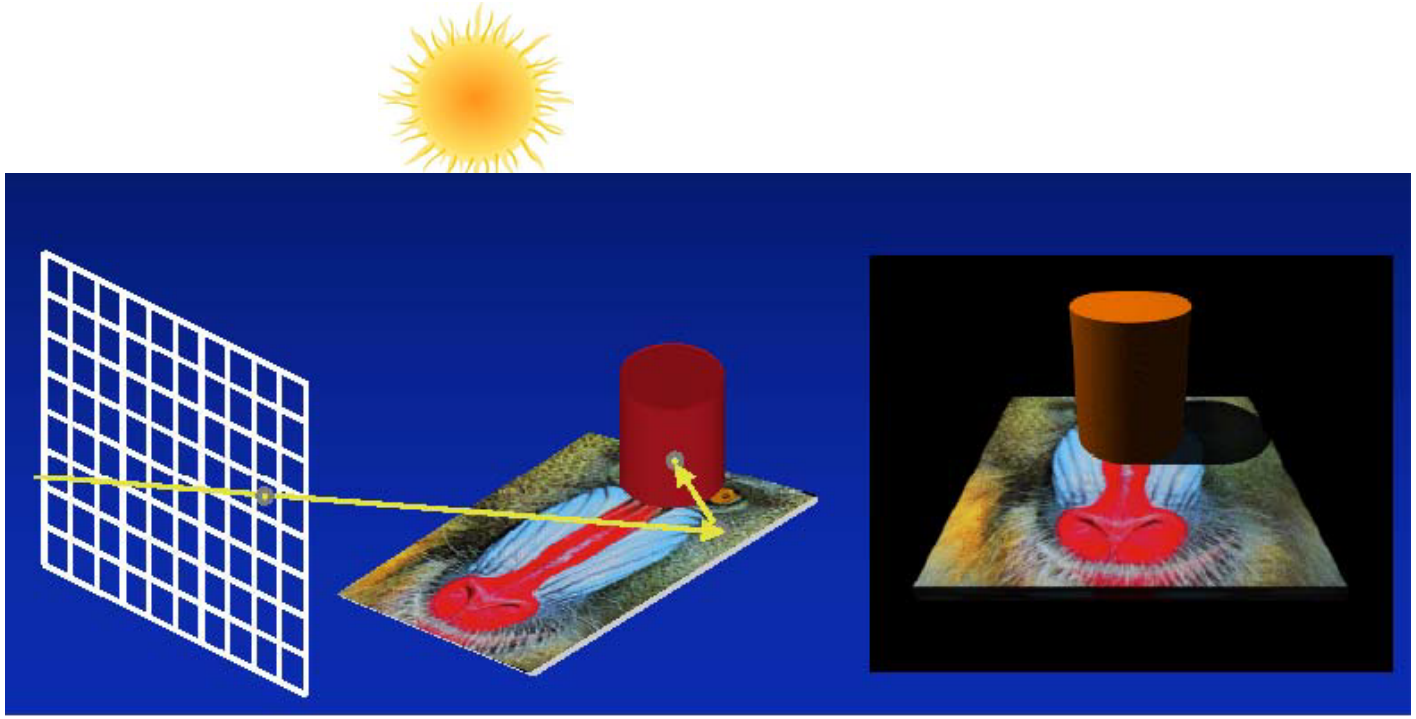
- Schneidet ein Strahl ein Objekt, so werden Schattenstrahlen vom Schnittpunkt zu allen Lichtquellen gesendet



9.5 Globale Beleuchtung

Ray-Tracing

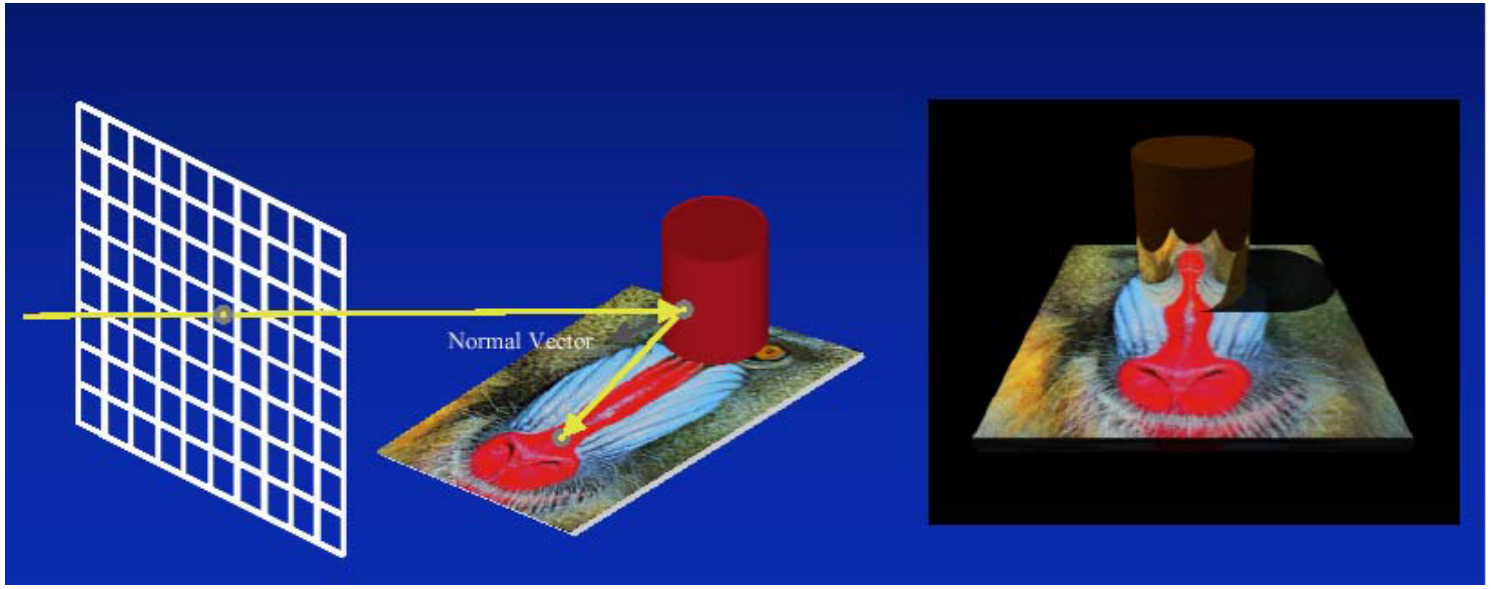
- Treffen die Schattenstrahlen auf ein Objekt, so liegt der Punkt im Schatten



9.5 Globale Beleuchtung

Ray-Tracing

- Ist die Oberfläche spiegelnd, wird ein reflektierter Strahl verfolgt



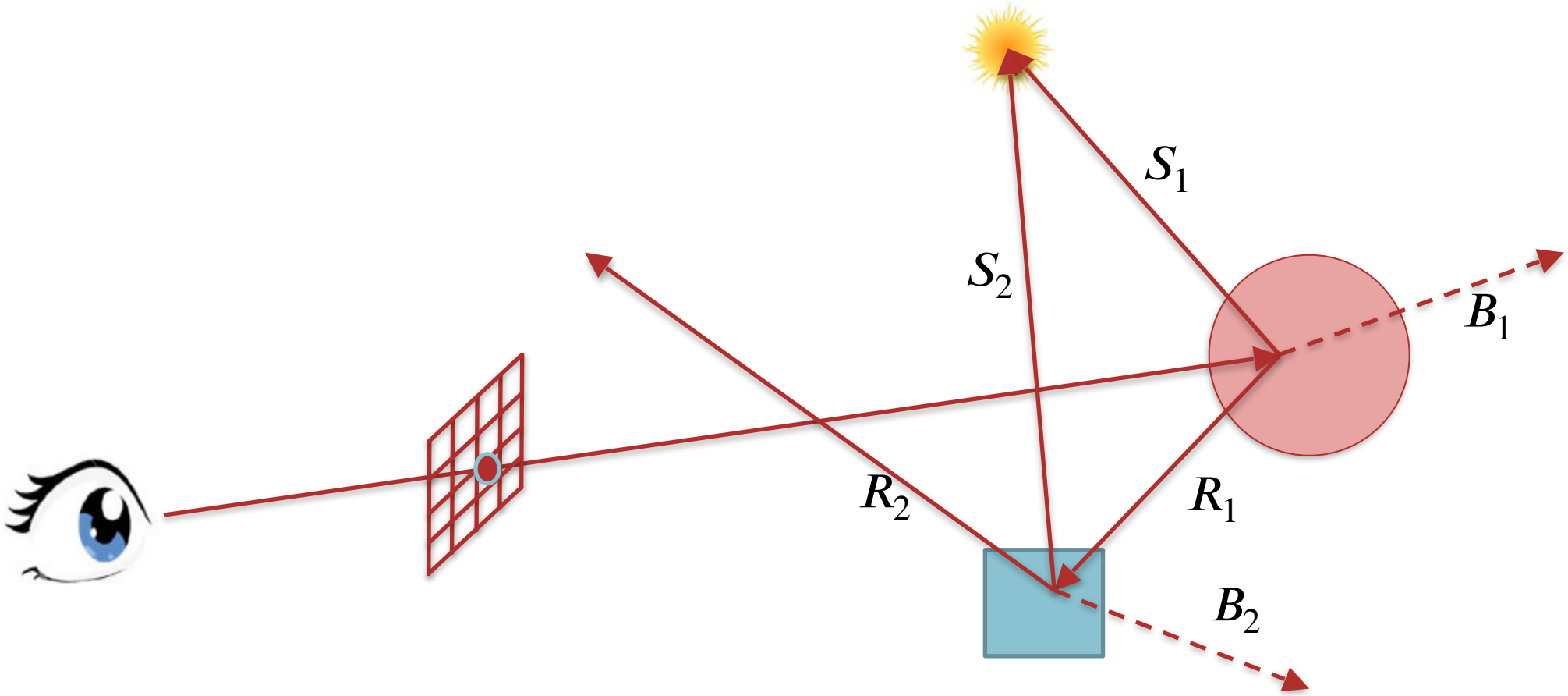
9.5 Globale Beleuchtung

Ray-Tracing: Algorithmus

- Für jeden Strahl
  - Bestimme Schnittpunkt mit nächstliegendem Objekt
  - Für jede Lichtquelle
    - Sende Schattenstrahl zur Lichtquelle
    - Falls Schattenstrahl kein Objekt schneidet, werte Phong-Beleuchtungsmodell im Schnittpunkt aus
  - Verfolge ideal reflektierten Lichtstrahl und addiere Leuchtdichte aus dieser Richtung (Rekursion)
  - Gib summierte Leuchtdichte zurück

9.5 Globale Beleuchtung

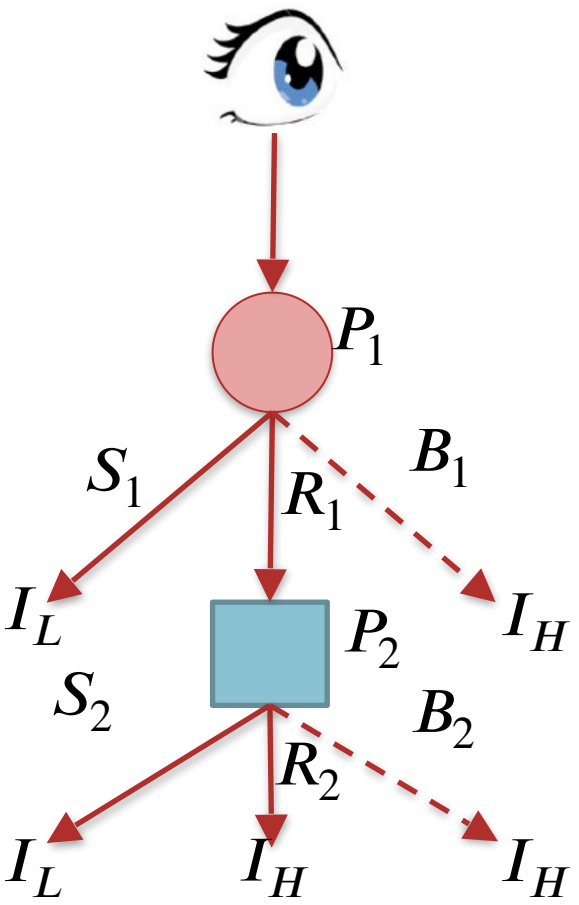
Ray-Tracing



9.5 Globale Beleuchtung

Ray-Tracing

- Strahlenbaum
  - Beschreibung der rekursiven Strahlverfolgung
  - Traversierung zur Bestimmung der Pixel-Intensität, berücksichtigt:
    - Lichtquellenintensität I_L
    - Hintergrundintensität I_H
    - Materialkoeffizienten r
    - Verdeckung von Lichtquellen V

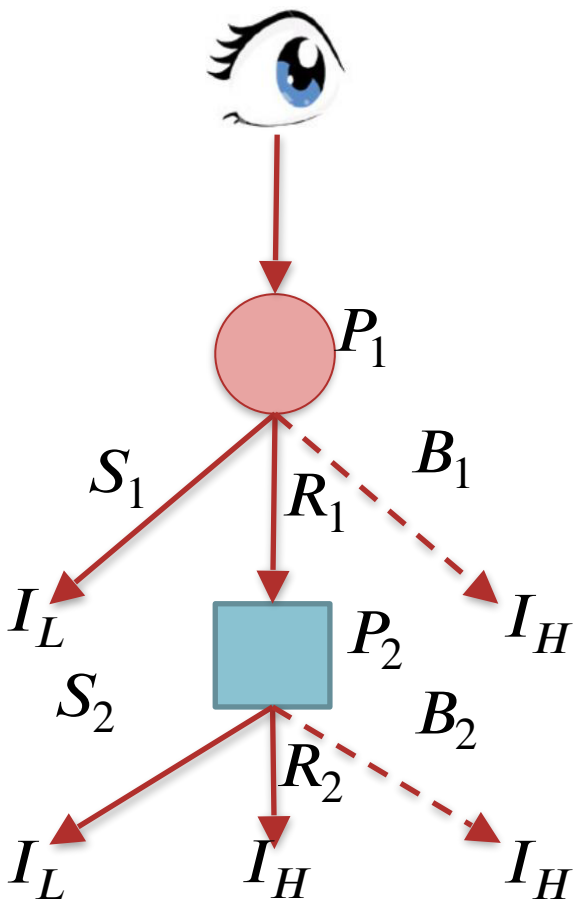


9.5 Globale Beleuchtung

Ray-Tracing

$$I_{ges} = V(P_1, L) \cdot I_{phong,P_1} + r_{r,P_1} \cdot I_{R_1} + r_{b,P_1} \cdot I_{B_1}$$

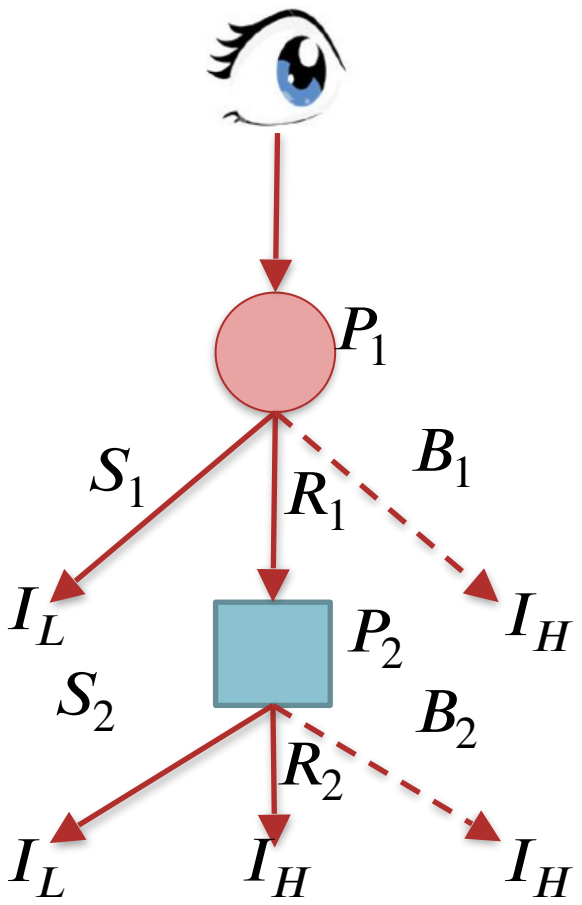
$$I_{R_1} = V(P_2, L) \cdot I_{phong,P_2} + r_{r,P_2} \cdot I_{R_2} + r_{b,P_2} \cdot I_{B_2}$$



9.5 Globale Beleuchtung

Ray-Tracing

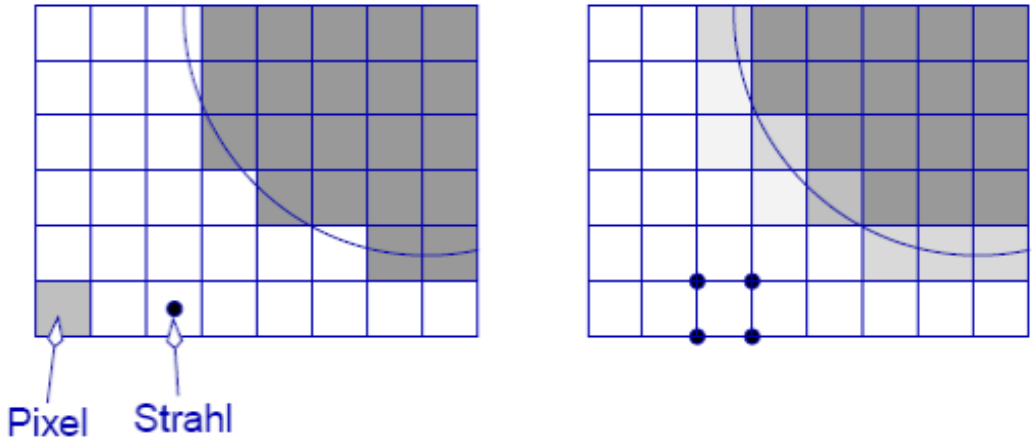
- Abbruchbedingung der Rekursionen
  - Verfolgte Strahlen schneiden keine Objekte mehr (verlassen die Szene, das Frustrum)
  - Vorgegebene maximale Baumtiefe ist erreicht
  - Farbanteil an der Pixelfarbe eines zu verfolgenden Teilstrahls ist zu gering (*adaptive Rekursionstiefenkontrolle*)



9.5 Globale Beleuchtung

Ray-Tracing

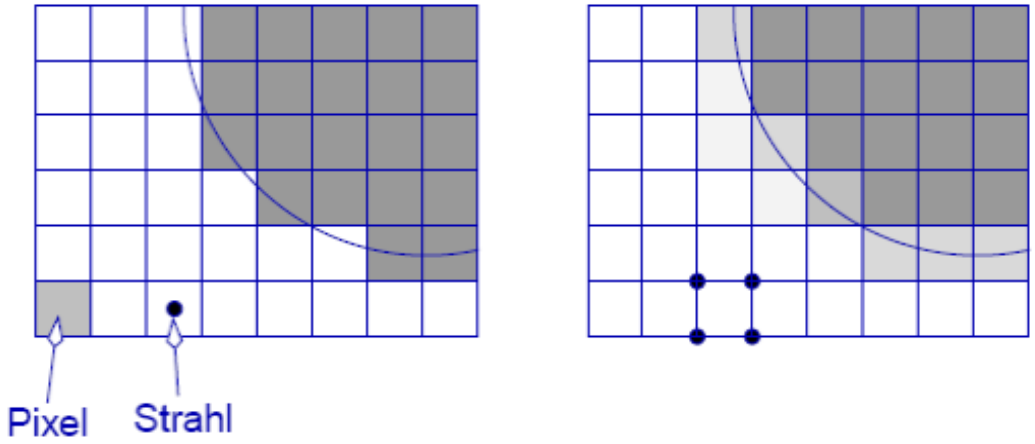
- Problem Aliasierung
  - Strahlen ohne Ausdehnung verursachen Abtastfehler



9.5 Globale Beleuchtung

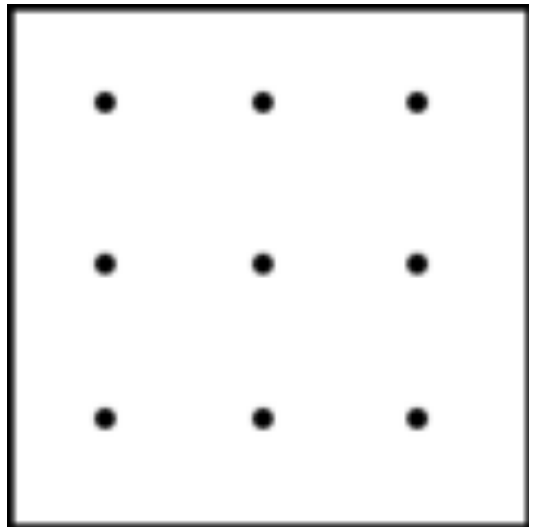
Ray-Tracing

- Problem Aliasierung
  - Strahlen ohne Ausdehnung verursachen Abtastfehler

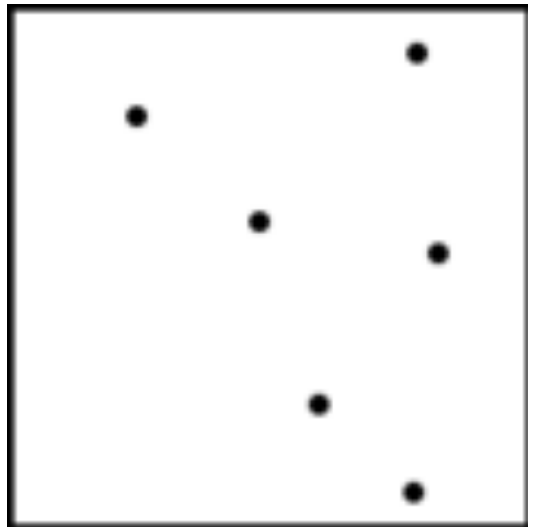


- Lösung
  - Supersampling:
    - Mehrere Strahlen pro Pixel
    - Mittelung der Farbwerte

Gitter



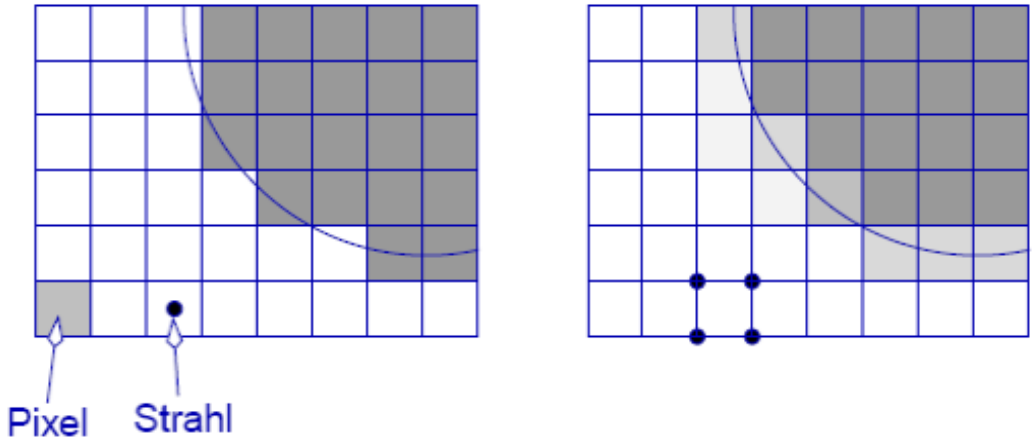
Zufällig



9.5 Globale Beleuchtung

Ray-Tracing

- Problem Aliasierung
  - Strahlen ohne Ausdehnung verursachen Abtastfehler



- Lösung
  - Adaptives Supersampling
    - An Grenzen mehr Strahlen als auf homogenen Flächen
    - Vorgehen:
      - Senden einiger Strahlen pro Pixel
      - Unterscheiden sich die Werte, werden weitere gesendet
    - Stochastische Verteilung möglich

9.5 Globale Beleuchtung

Ray-Tracing

- Zeitaufwand
  - Schnittpunktberechnungen Strahl-Objekt aufwändig
  - Viele Berechnungen $O(n \cdot m)$
    - n: Anzahl der Objekte
    - m: Anzahl der Strahlen
  - Rekursion in der Strahlenverfolgung
  - Anzahl der Lichtquellen
  - Flächige Lichtquellen
- Beschleunigung
  - Weniger Schnittpunktberechnungen
    - Hüllkörperhierarchien (z.B. Octrees)
    - Light Buffer
    - Bounding Boxes
      - Schneller Ausschluss von Schnitten
  - Weniger Strahlen
    - Adaptive Rekursionstiefenkontrolle
    - Adaptives Anti-Aliasing

9.5 Globale Beleuchtung

Ray-Tracing

- Vorteile
  - Hervorragend geeignet für
    - Spiegelungen
    - Punktlichtquellen
  - Gute Simulation des physikalischen Beleuchtungsvorgangs (Strahlengang)
  - Sichtbarkeitsproblem automatisch gelöst
  - Implizite Schattenberechnung
  - Große Wirklichkeitsnähe
- Nachteile
  - Weniger geeignet für
    - diffuse Reflektion
    - Flächige Lichtquellen
  - Großer bis sehr großer Rechenaufwand
  - Ideale Reflexion und Brechung entspricht nicht der Realität und lässt gerenderte Szenen unecht erscheinen
  - Keine Unterscheidung nach Farbe (λ) bei Reflexion und Brechung

9.5 Globale Beleuchtung

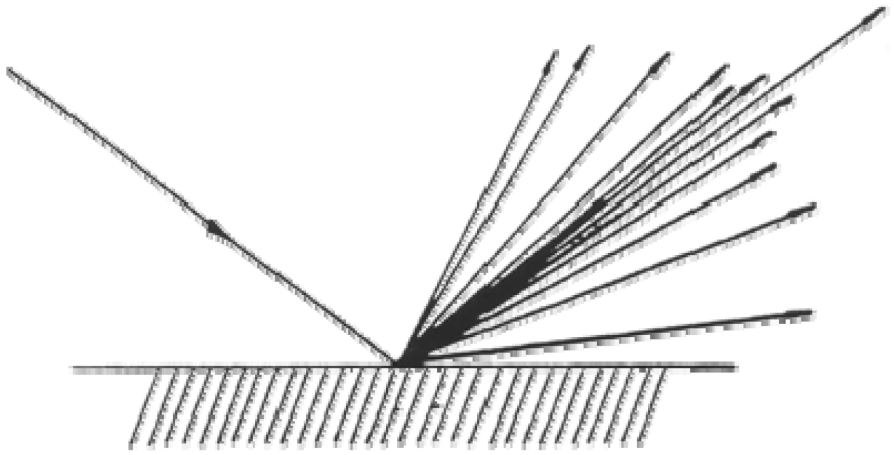
Distributed Ray-Tracing

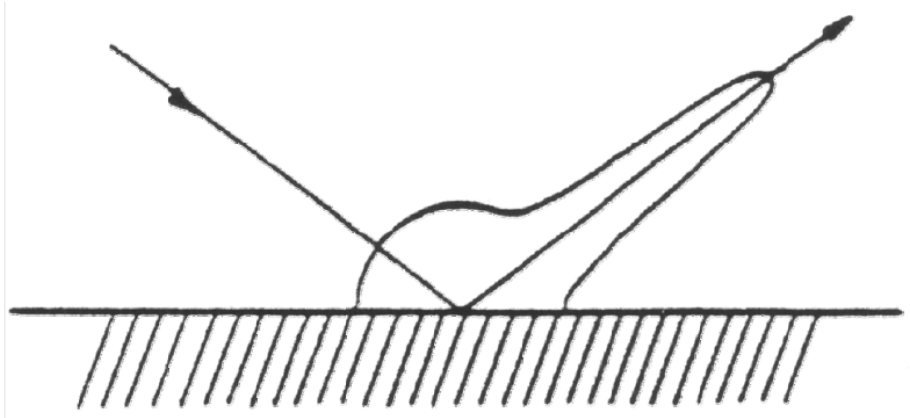
- Verbesserung:
  - Distributed ray tracing
    - Auch: distribution ray tracing
    - stochastic ray tracing
- Ziel: Realistisch unscharfe Effekte
  - Verschwommene Reflexionen und Brechungen
  - Weiche Schatten
  - Tiefenunschärfe
  - Bewegungsunschärfe

9.5 Globale Beleuchtung

Distributed Ray-Tracing

- Verschwommene Reflexionen
  - Vervielfachung der reflektierten Strahlen
  - Stochastische Verteilung der Strahlrichtungen
  - Anschließende Mittelung der Farbwerte
  - Auch für Brechung anzuwenden

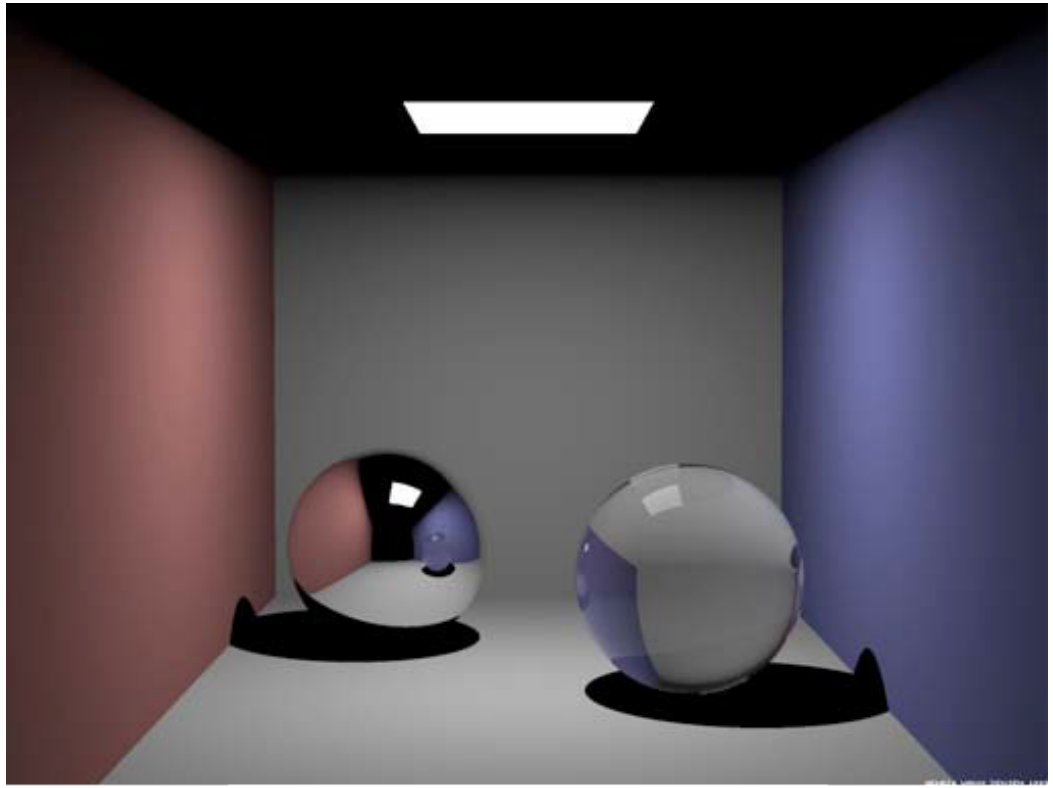


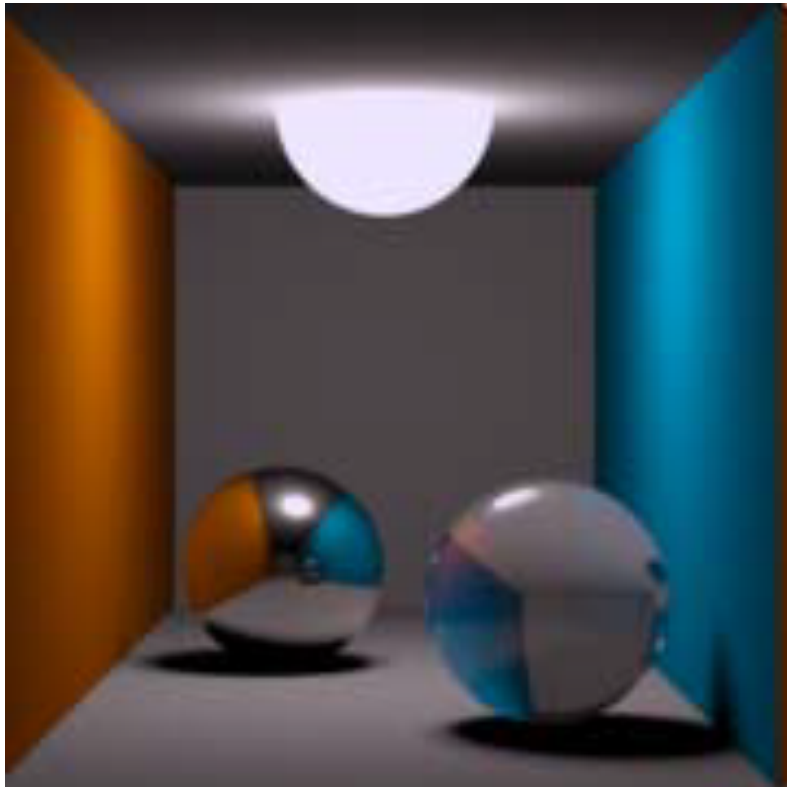


9.5 Globale Beleuchtung

Distributed Ray-Tracing

- Verschwommene Reflexionen

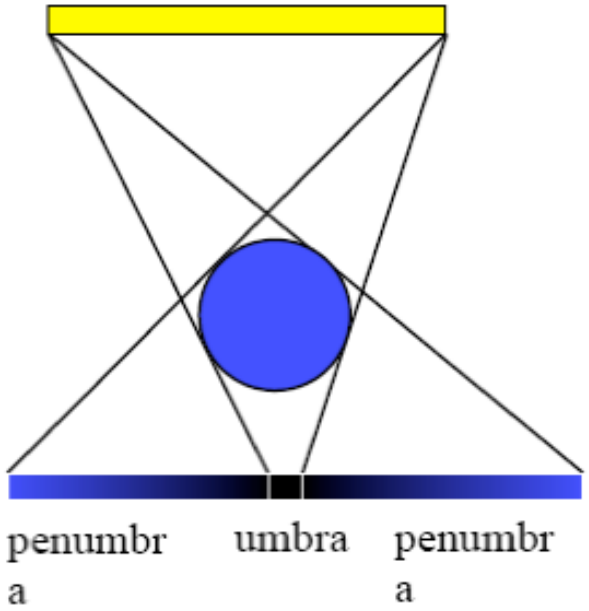




9.5 Globale Beleuchtung

Distributed Ray-Tracing

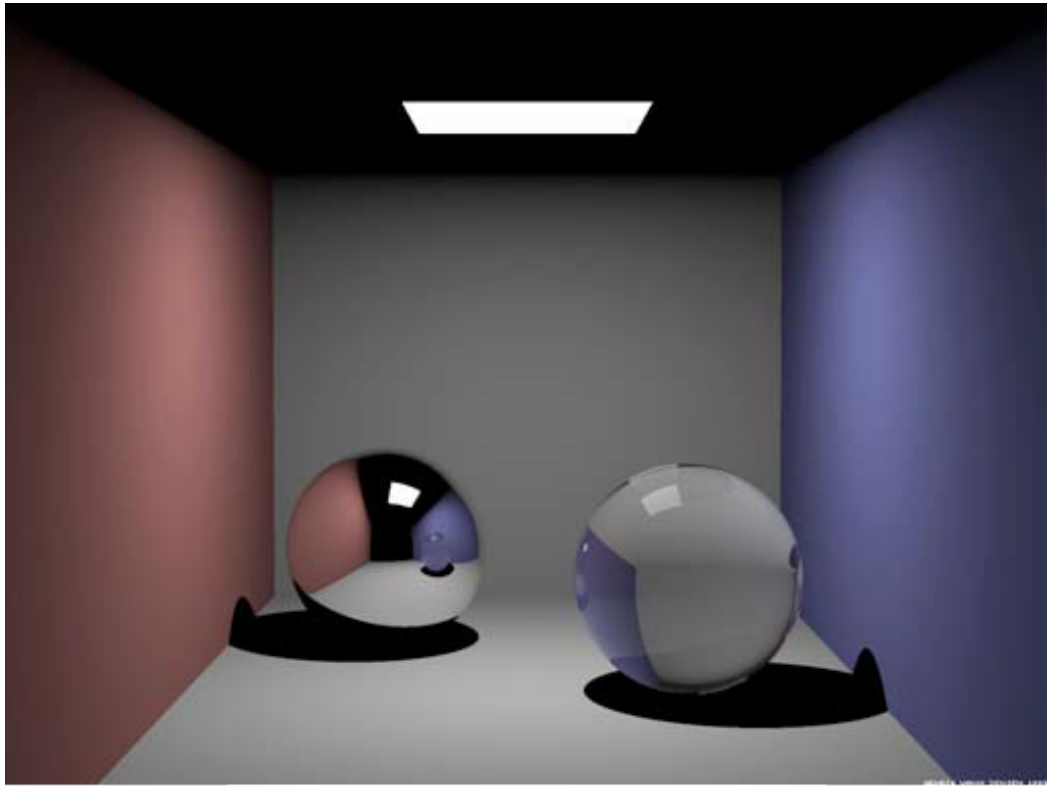
- Weiche Schatten
  - Verwendung von Flächenlichtquellen
  - Simuliert durch Vielzahl punktförmiger Lichtquellen
  - Stochastische Verteilung der Schattenstrahlen
  - Stärke des Schattens entspricht Verhältnis von verdeckten zur Gesamtzahl an Schattenstrahlen

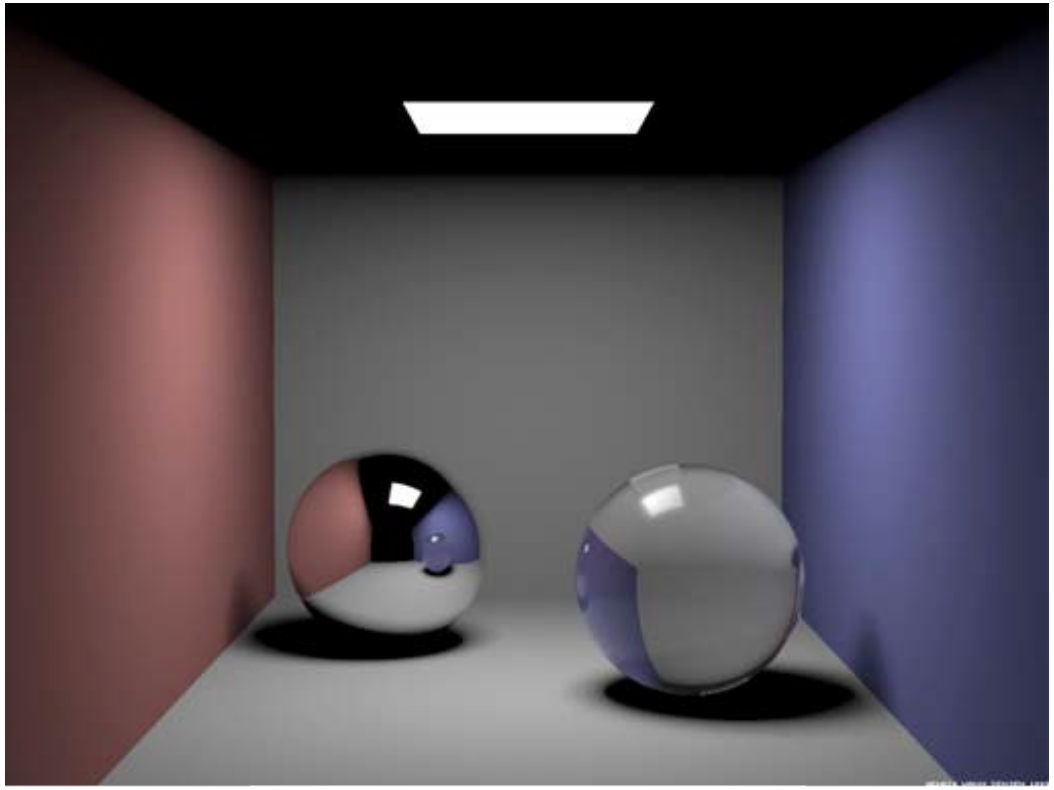


9.5 Globale Beleuchtung

Distributed Ray-Tracing

- Weiche Schatten

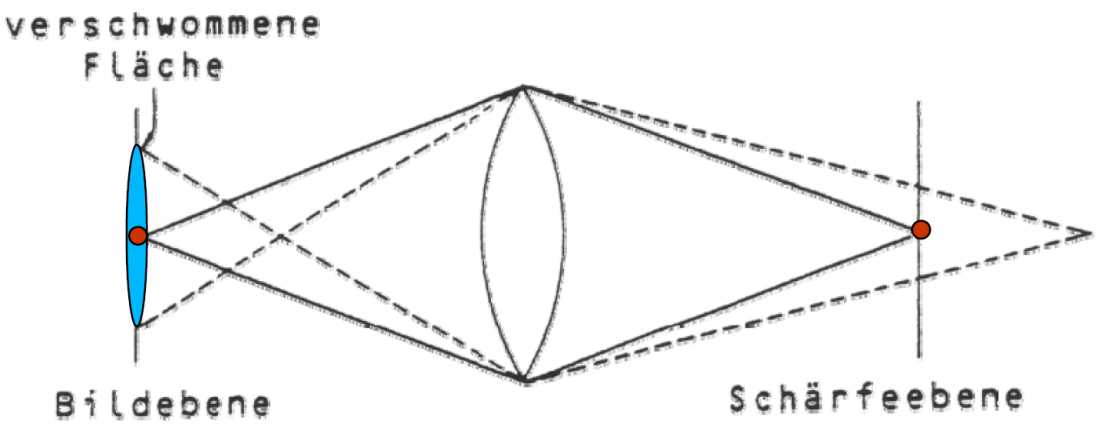




9.5 Globale Beleuchtung

Distributed Ray-Tracing

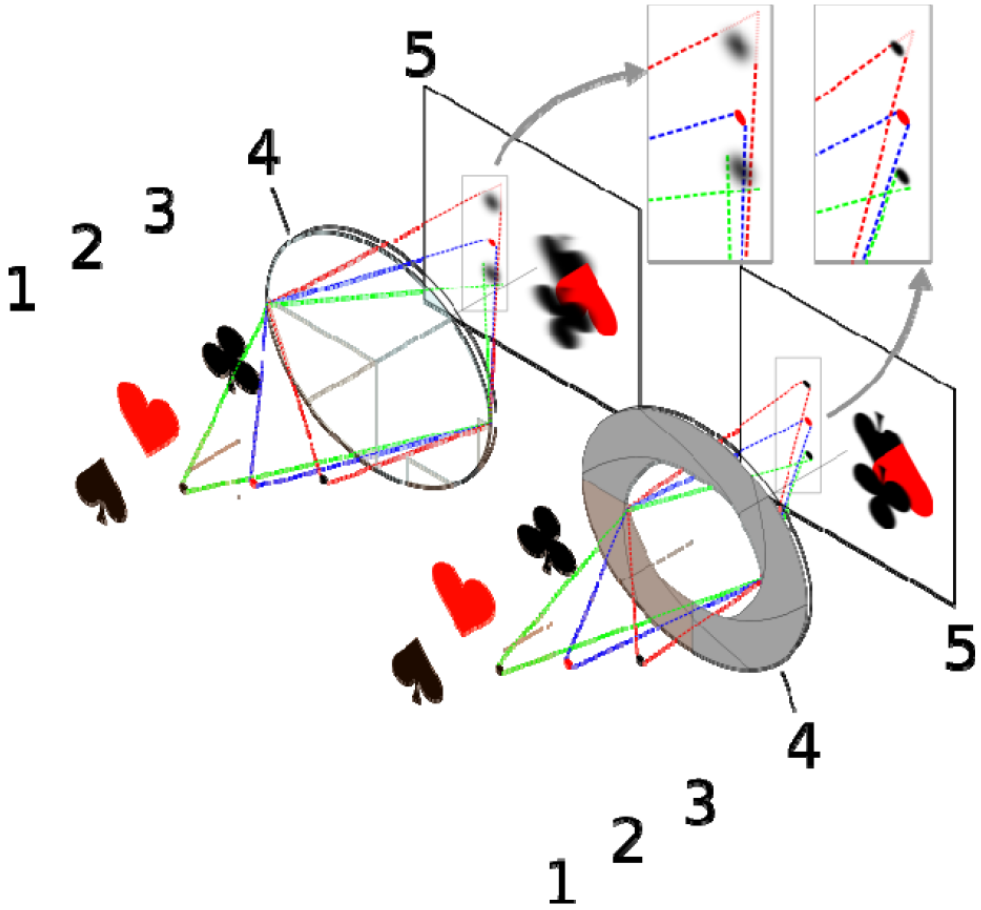
- Tiefenunschärfe
  - Simulation der Blendenöffnung einer Kamera
  - Punkte außerhalb der Schärfeebene erscheinen verschwommen (wenn größer als circle of confusion)
  - Korrekte Berechnung der Linsenbrechung mit stochastischer Strahlverteilung über Linsenoberfläche



9.5 Globale Beleuchtung

Distributed Ray-Tracing

- Tiefenunschärfe
  - Blendenöffnung bestimmt Bereich der Szene, der scharf abgebildet wird (depth of field)



9.5 Globale Beleuchtung

Distributed Ray-Tracing

- Tiefenunschärfe

9.5 Globale Beleuchtung

Distributed Ray-Tracing

- Bewegungsunschärfe
  - Zeitliche Verteilung der Augstrahlen
  - Senden von Strahlen durch gleichen Punkt mit zeitlicher Verzögerung
  - Filter bestimmt Verschlusszeit und damit Stärke des Motion Blur

9.5 Globale Beleuchtung

Radiosity-Verfahren

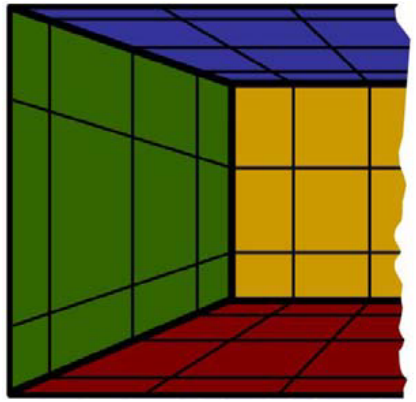
- Modelliert rein diffuse Reflexion
- Ausbreitung des Lichts unter Beachtung des Energiegleichgewichts in einem geschlossenem System
- Gleichgewicht des
  - durch Lichtquellen zugeführten
  - durch Oberflächen absorbierten

  Lichtstroms
- Modellierung von unterschiedlichem Einfall des ambienten (indirekten) Lichts besser als beim klassischen Raytracing
  - Modellierung von Flächen-Lichtquellen
  - Modellierung von flächigen Schatten
- Gut geeignet für Lichtverteilung in Gebäuden

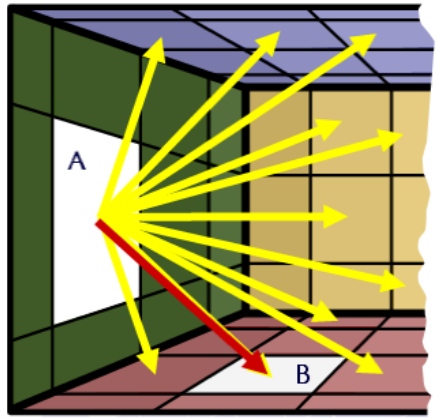
9.5 Globale Beleuchtung

Radiosity-Verfahren

- Gegeben:
  - Szene in Flächensegmente unterteilt
  - Geometrische Informationen über die gegenseitige Lage aller Objekte
  - Lichttechnische Kenngrößen der Objekte



- Gesucht:
  - Radiosity für jede Fläche
  - Dient dann zur Farbgebung beim Rendern der Szene



9.5 Globale Beleuchtung

Radiosity-Verfahren

- $B(x) = E(x) + \rho(x) \int_{x' \in S} B(x') \frac{1}{\pi r^2} \cos\phi_x \cos\phi_{x'} \cdot V(x, x')\, \mathrm{d}x'$
- $B(x)$ gesamte vom Punkt x abgestrahlte Energie als Leistung pro Flächeneinheit, genannt Bestrahlungsstärke (Radiosity)
- $E(x)$ im Punkt x abgegebene Eigenstrahlung
- $\rho(x)$ Reflexionsfaktor im Punkt x
- S: alle Oberflächen der Szene
- r = Abstand der Punkte x und x'
- $\phi(x)$ Winkel zwischen der Normalen im Punkt x und der Verbindungslinie zwischen den Punkten x und x'
- $V(x, x') = \begin{cases} 1, & \text{wenn } x \text{ von } x' \text{ aus sichtbar} \\ 0, & \text{sonst} \end{cases}$

9.5 Globale Beleuchtung

Radiosity-Verfahren

- $$B(x) = E(x) + \rho(x) \int_{x' \in S} B(x') F(x, x') \, \mathrm{d}x'$$
- $B(x)$ gesamte vom Punkt x abgestrahlte Energie als Leistung pro Flächeneinheit, genannt Bestrahlungsstärke (Radiosity)
- $E(x)$ im Punkt x abgegebene Eigenstrahlung
- $\rho(x)$ Reflexionsfaktor im Punkt x
- S: alle Oberflächen der Szene
- $F(x, x')$ Formfaktor, Anteil an der von x' abgegebenen Energie, die x erreicht

9.5 Globale Beleuchtung

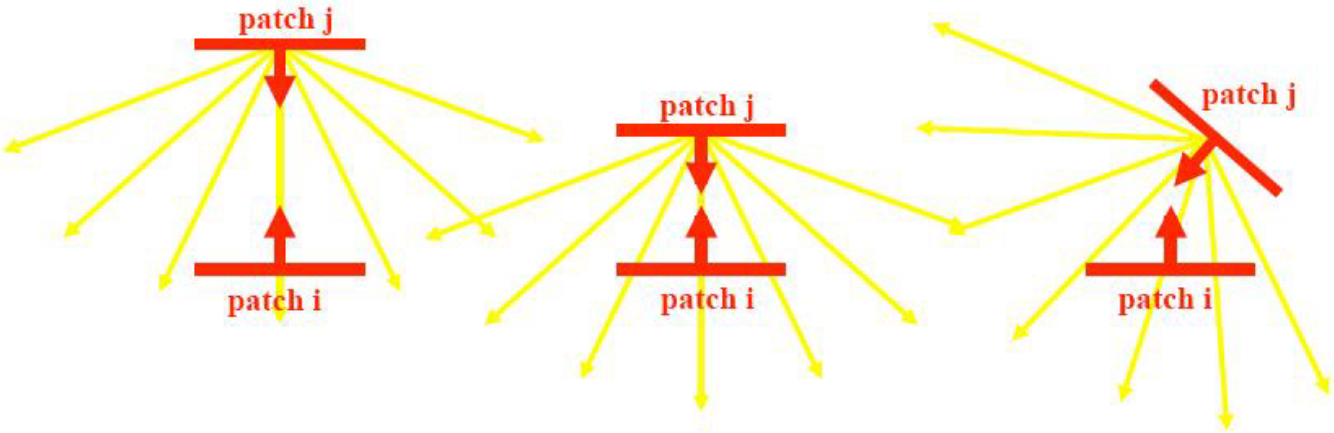
Radiosity-Verfahren

- Diskretisierung
- Statt infinitesimaler Flächenelemente dx:
  - Finite-Elemente-Methode
  - Kleine, zusammenhängende Teilflächen (Facetten oder Patches) A_i
- Aus Integralgleichung wird lineares Gleichungssystem
- $$B_i = E_i + \rho_i \sum_{j=1}^{n} B_j F_{ij}$$
- B_i gesamte von Fläche A_i abgestrahlte Energie
- E_i von der Teilfläche i abgegebene Eigenstrahlung
- F_{ij} Anteil an der von Fläche j abgegebenen Energie, die auf Fläche i auftrifft
- n Anzahl der Teilflächen

9.5 Globale Beleuchtung

Radiosity-Verfahren

- Formfaktoren F_{ij}
  - Anteil des Lichtstroms von A_i, der A_j erreicht
  - Nur abhängig von
    - Geometrie
      - Abstand
      - Orientierung
      - Größe
    - Sichtbarkeit



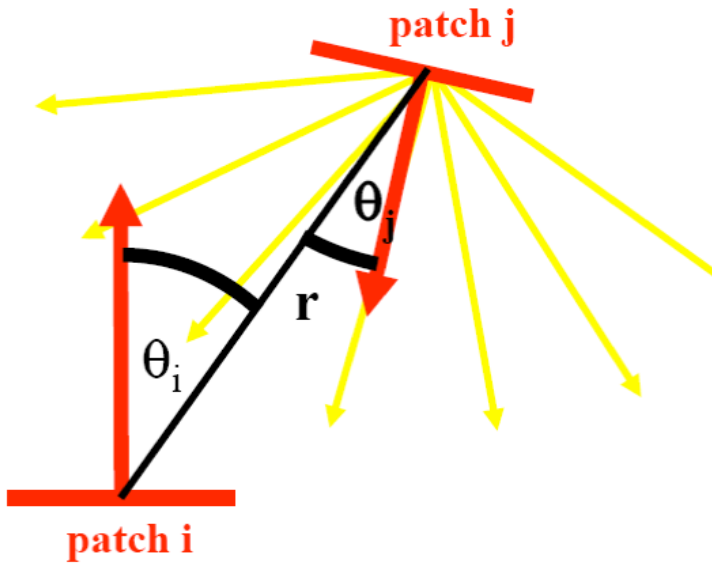
9.5 Globale Beleuchtung

Radiosity-Verfahren

- Berechnung der Formfaktoren F_{ij}

$$F_{i,j} = \frac{1}{A_i} \int_{v \in A_i} \int_{u \in A_j} \frac{1}{\pi r^2} \cos\phi_u \cos\phi_v V(u,v)\, \mathrm{d}A_j \mathrm{d}A_i$$

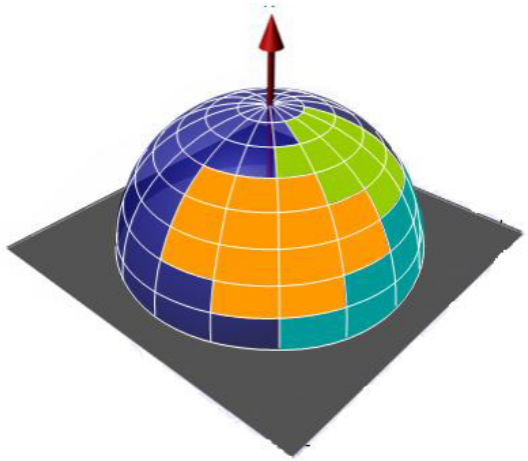
- r Abstand
- ϕ Winkel zwischen der Normalen und der Verbindungslinie
- $V(u,v) = \begin{cases} 1, & \text{wenn } i \text{ von } j \text{ aus sichtbar} \\ 0, & \text{sonst} \end{cases}$
- Einfachste Approximation
- $F_{i,j} \approx \frac{\cos\phi_u \cos\phi_v}{\pi r^2} A_i V_{ij}$

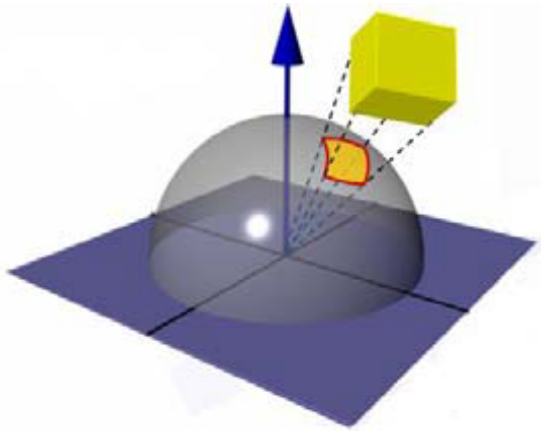


9.5 Globale Beleuchtung

Radiosity-Verfahren

- Approximation der Formfaktoren F_{SE_i}
  1. Senderfläche S in Patches S_i unterteilen
  2. Hemisphäre für jedes Patch in Normalenrichtung erstellen
  3. Empfängerflächen E_i auf Hemisphäre projizieren (Verdeckung beachten)
  4. Für jedes E_i Anteil an Hemisphäre bestimmen, der von ihr bedeckt wird
  5. Beiträge von E_i über alle Patches S_i mitteln





9.5 Globale Beleuchtung

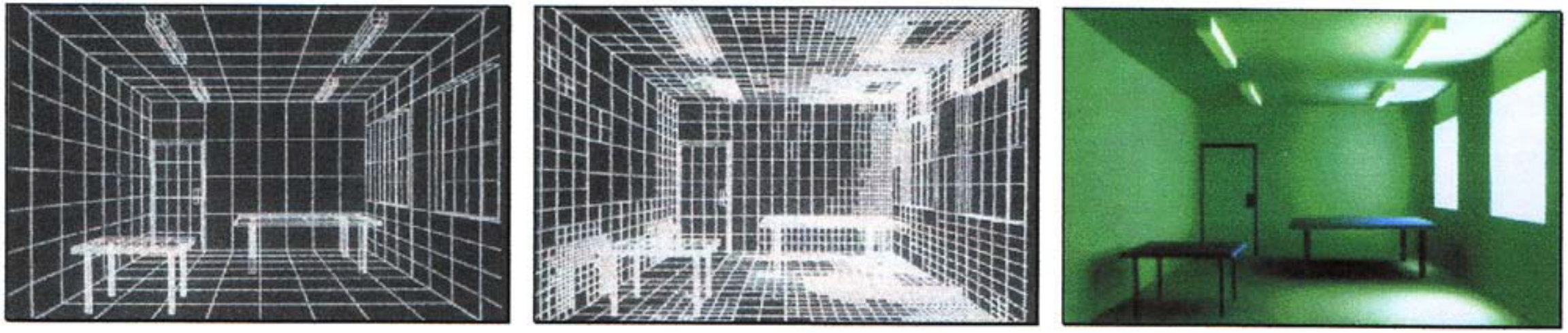
Radiosity-Verfahren

- Lösung des Radiosity-Gleichungssystems
- $$B_i = E_i + \rho_i \sum_{j=1}^{n} B_j F_{ij}$$
- Matrixschreibweise
  $$B = E + \rho F B \Rightarrow (I - \rho F) B = E$$
- Dieses System wird getrennt für die verschiedenen Frequenzen des Lichts (z.B. RGB) gelöst
- Die Koeffizientenmatrix ist diagonaldominant, lösbar mit
  - Iterativem Verfahren, wie Gauss-Seidel
  - Progressive refinement
- *„Gathering“ bzw. „Sammeln“*

9.5 Globale Beleuchtung

Radiosity-Verfahren

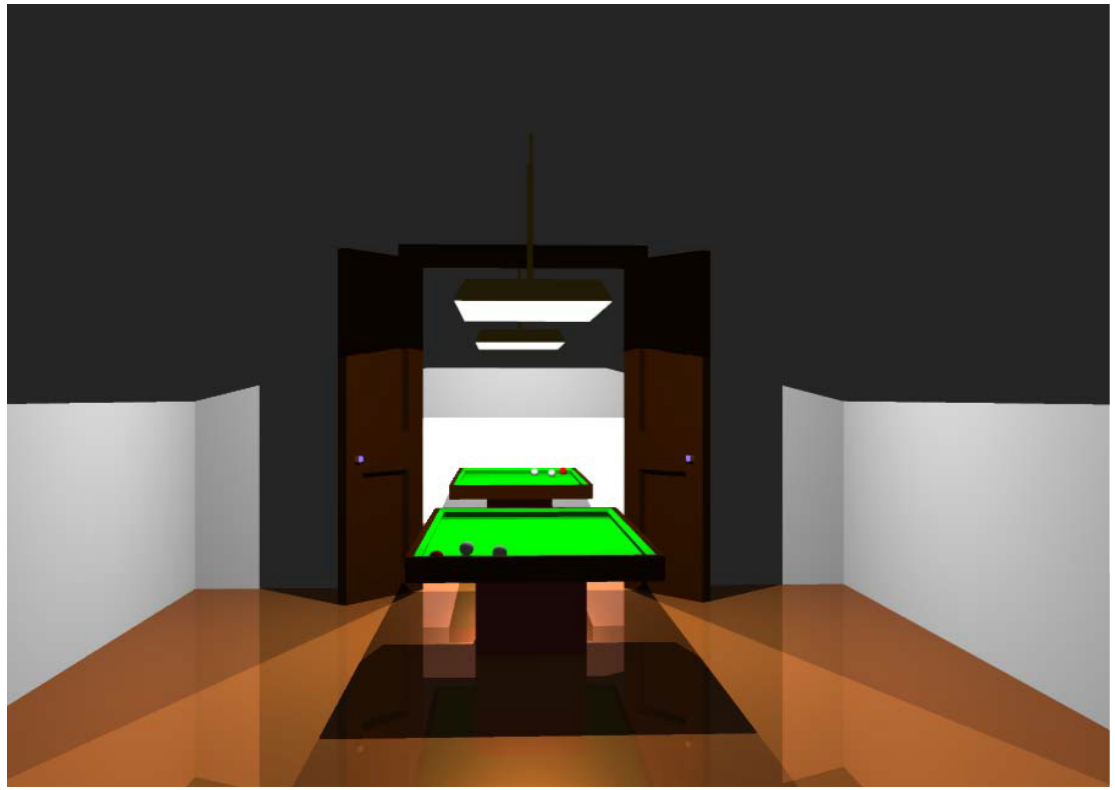
- Je feiner die Aufteilung der Szene in Patches, desto besser das Ergebnis
- Aufwand steigt mit Anzahl n der Patches, z.B. Anzahl der Formfaktoren $O\left(n^2\right)$



9.5 Globale Beleuchtung

Ergebnisse

- Raytracing



- Radiosity

9.5 Globale Beleuchtung

Radiosity-Verfahren

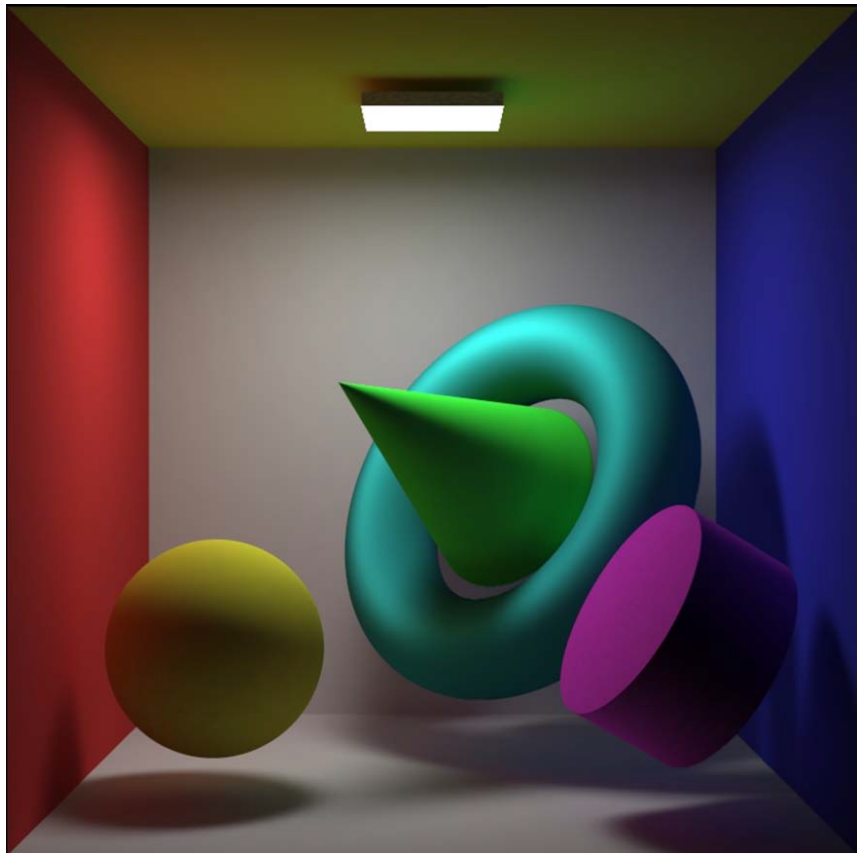
- Vorteile
- Blickpunktunabhängigkeit
- Einfache indirekte, ideal diffuse Lichtreflexion

- Nachteile
- nur diffuse Reflexion
- keine analytischen Primitive
- Hoher Speicher- und Zeitbedarf

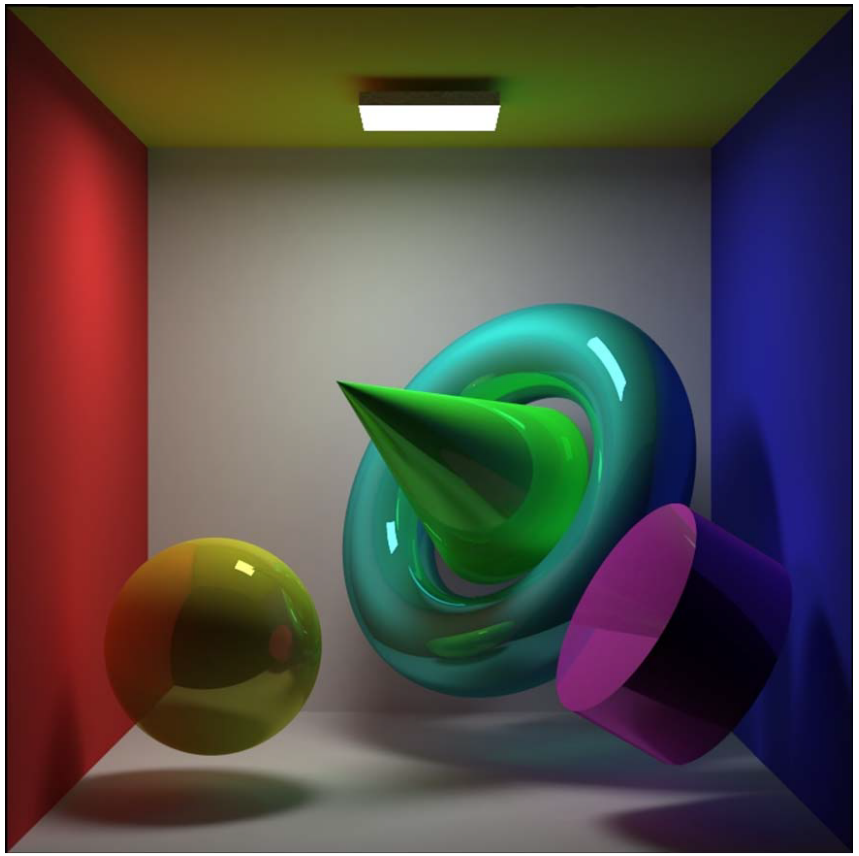
9.5 Globale Beleuchtung

Ergebnisse

- Radiosity



- Raytacing & Radiosity



9.5 Globale Beleuchtung

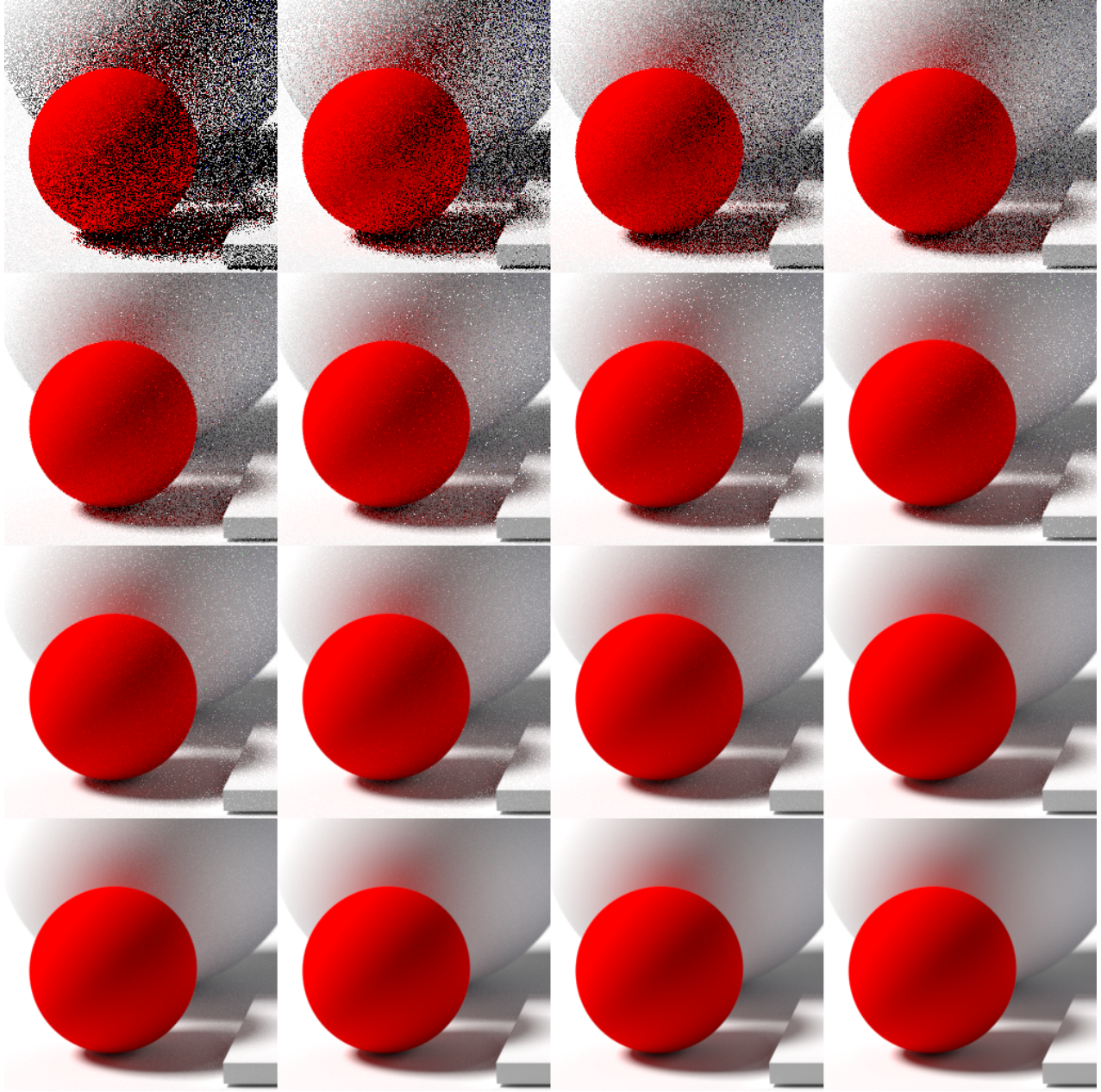
Path Tracing

- Kajiya's Rendergleichung (James Kajiya 1986)
- $L(x, \vec{\omega}) = L_e(x, \vec{\omega}) + \int_{\Omega} f_r(x, \vec{\omega}', \vec{\omega}) L(x, \vec{\omega}')(\vec{\omega}' \cdot \vec{n}) \mathrm{d}\vec{\omega}'$
- $L(x, \vec{\omega})$ Energiefluss: wie viel Licht wird von x aus in Richtung ω abgestrahlt
- $L_e(x, \vec{\omega})$ Emissionsterm: wie viel Licht wird von x aus in Richtung ω ausgestrahlt
- $f_r(x, \vec{\omega}', \vec{\omega})$ Streuungsterm (BRDF) mit Einfallswinkel ω' und Reflexionswinkel ω
- n Normalenvektor der Oberfläche im Punkt x
- Ω Gesamtheit aller Winkel der Kugel über der Oberfläche

9.5 Globale Beleuchtung

Path Tracing

- Kajiya's Rendergleichung (James Kajiya 1986)
- Wird approximativ durch Monte Carlo Integration gelöst
- zufällige Richtung bei der Aussendung und jeder Reflexion oder Brechung
- anschließsende Mittelung
- sehr aufwendig (ca. 5000 Strahlen pro Pixel nötig)
- physikalisch



Quellen

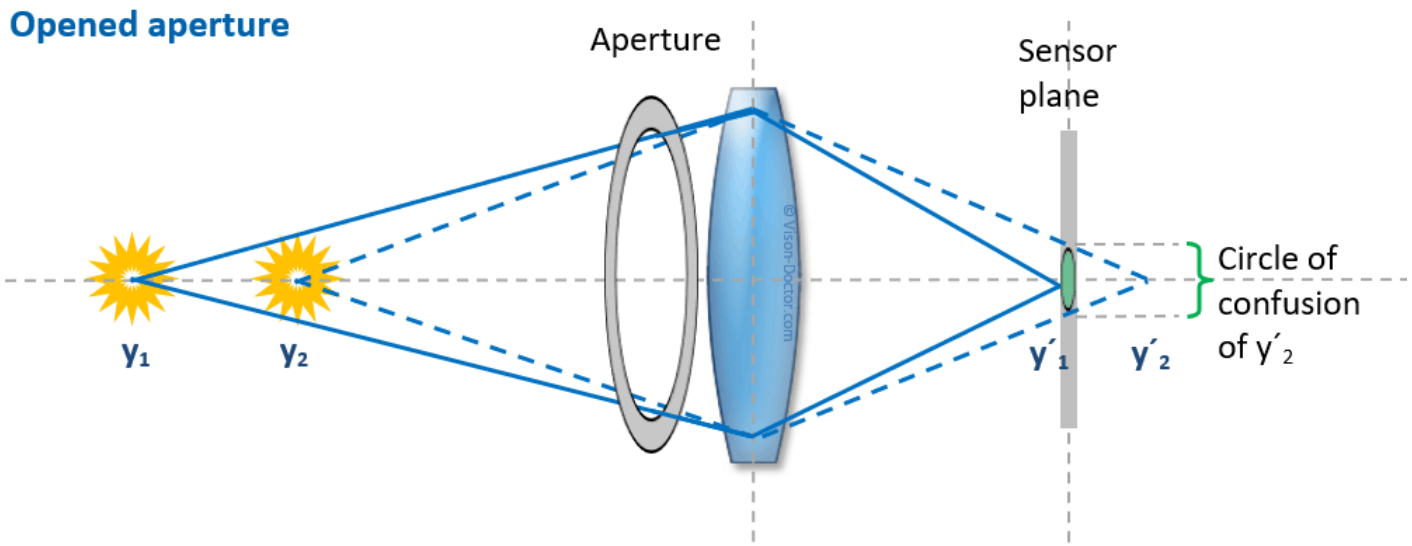
- Computergraphik, Universität Leipzig
  (Prof. D. Bartz)
- Graphische Datenverarbeitung I, Universität Tübingen
  (Prof. W. Straßer)
- Graphische Datenverarbeitung I, TU Darmstadt
  (Prof. M. Alexa)
- Wikipedia: http://en.wikipedia.org

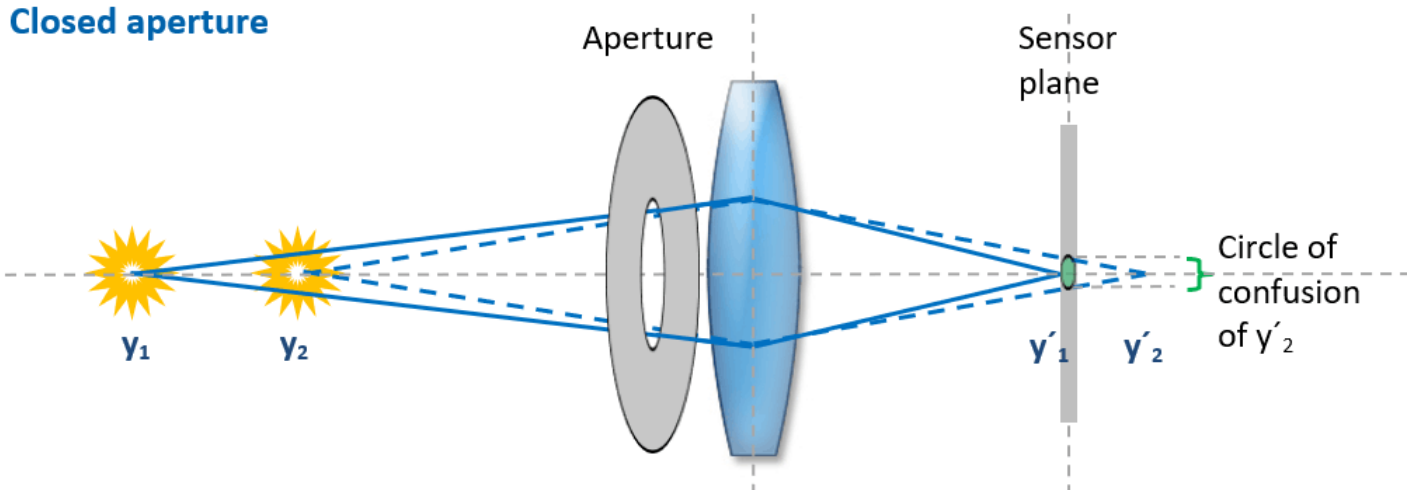
UNIVERSITÄT
LEIPZIG

Texturen

COMPUTERGRAPHIK

TIEFENUNSCHÄRFE





INHALTSVERZEICHNIS

10 Texturen

10.1 Texture-Mapping

10.2 Bump-Mapping

10.3 Environment-Mapping

10.4 Weitere Verfahren

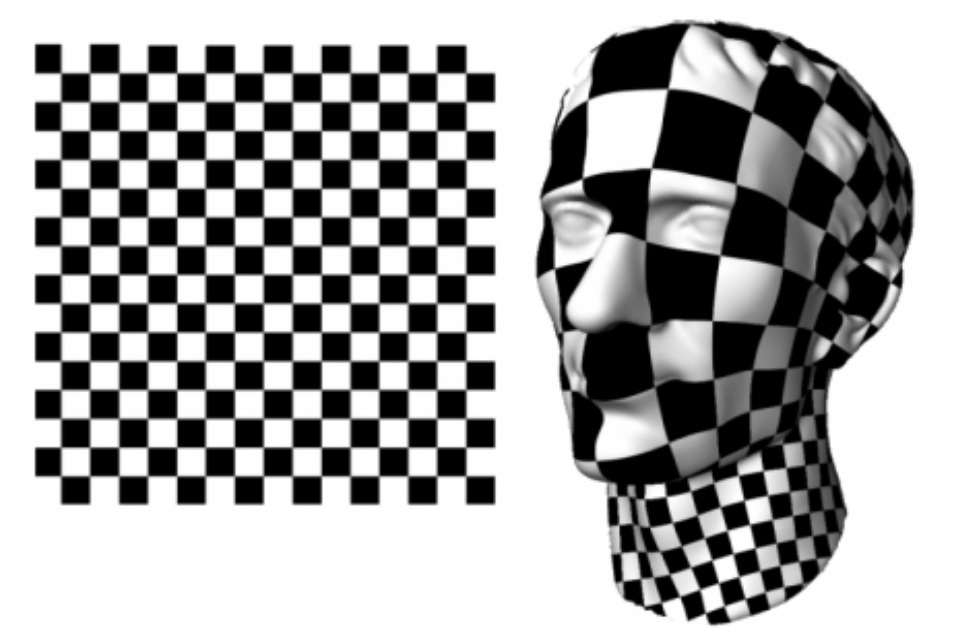
10. ABBILDUNGSVERFAHREN

Motivation

- Reale Umgebung verfügt über ein großes Spektrum geometrischer Formen und physikalischer Materialien
  - Maserungen und Muster von Oberflächen
  - Strukturen unebener Flächen
  - Hintergrund und Spiegelungen mit hohem Detailgrad
- Exakte Nachbildung dieser Objekte ist meist zu aufwändig.

10.1 TEXTURE-MAPPING

- Texture-Mapping
  - Aufbringen von 2D-Texturen auf eine 3D-Oberfläche
  - Ermöglicht komplexe Gestaltung einfacher Objekte

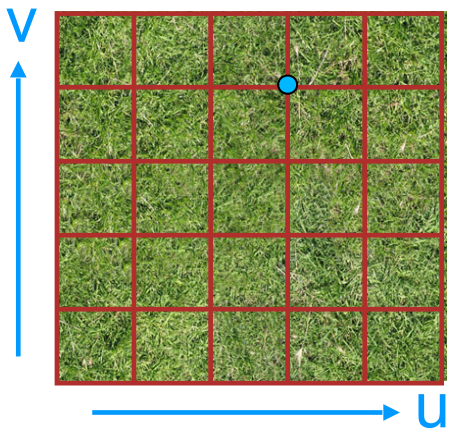


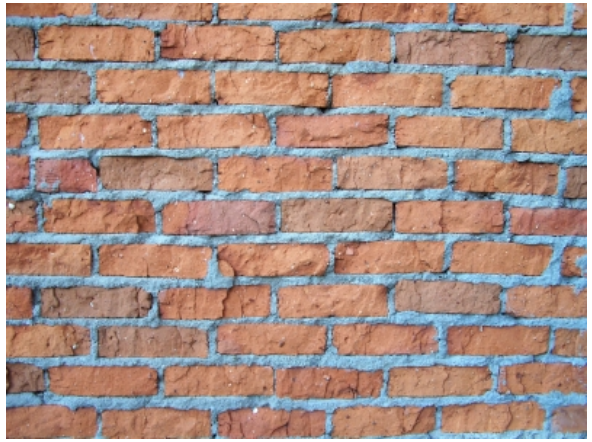
- Beispiele
  - Blick aus dem Fenster
  - Spiegelbild
  - Parkettboden

10.1 TEXTURE-MAPPING

Texturen

- Funktionen, die Punkte des (u,v)-Texturraums auf (r,g,b)-Werte abbilden:
  $$\left(r, g, b\right) = C_{textur}(u, v)$$
- $(u, v) \in \left[0; 1\right]$





10.1 TEXTURE-MAPPING

Diskrete Texturen

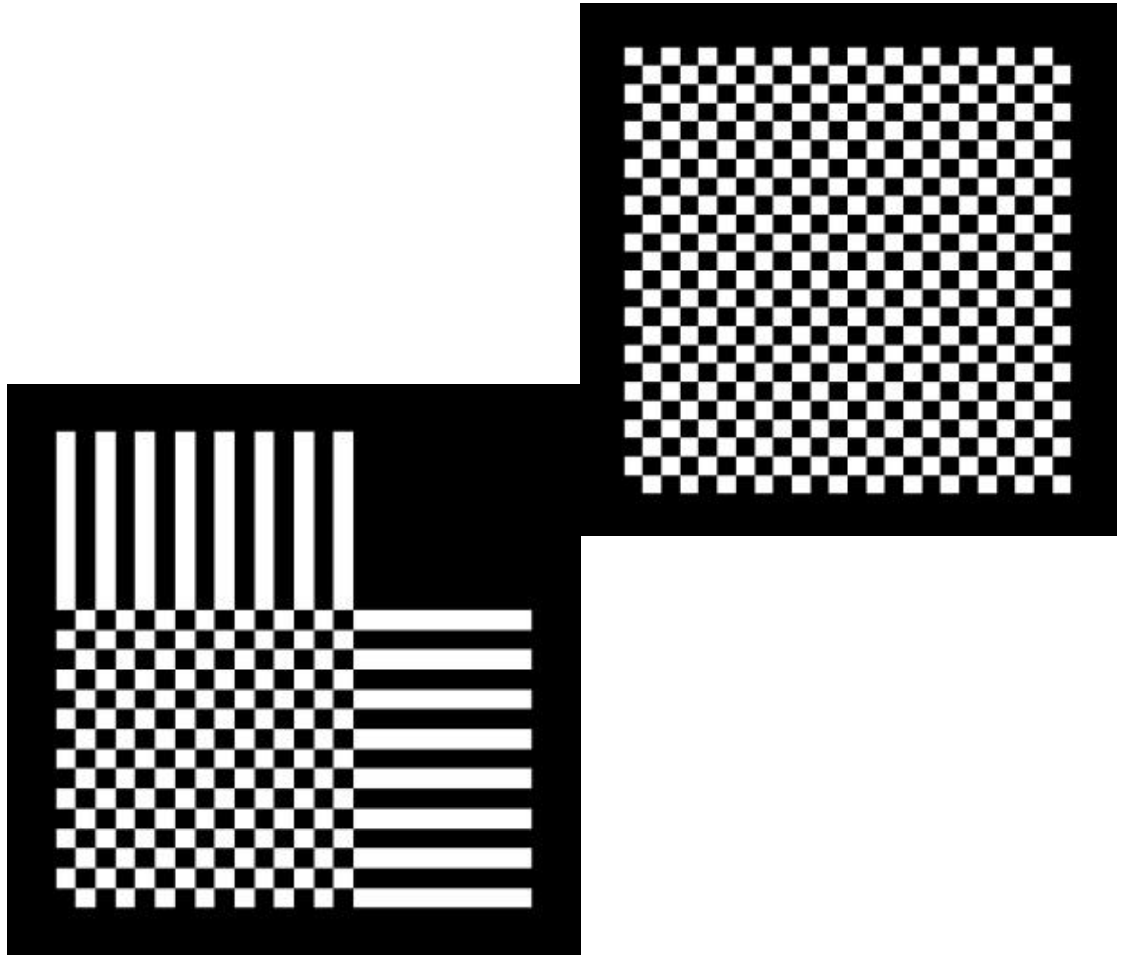
- Diskret
  - Werden als Vektorfelder C abgespeichert
  - Ein Vektor enthält Farbkomponenten und wird als Texel bezeichnet.
- Vorteile
  - Vorrat unerschöpflich
    - Smartphone
    - Fotoapparat
    - Scanner
    - Downloads
  - Photorealismus möglich
- Nachteile
  - Hoher Speicherbedarf
  - Unstimmiger Kontext von Szene und Textur (Schattenwurf…)
  - Anfällig für Artefakte und Aliasing
  - Rekonstruktion der Texturwerte notwendig

10.1 TEXTURE-MAPPING

Diskrete Texturen

- Sonderregeln, falls $(u, v) \notin [0; 1]^2$
- Periodisch (Repeat)
  $C(u) = \left(u - \lfloor u \rfloor\right)$
- Abschneiden (Clamp)

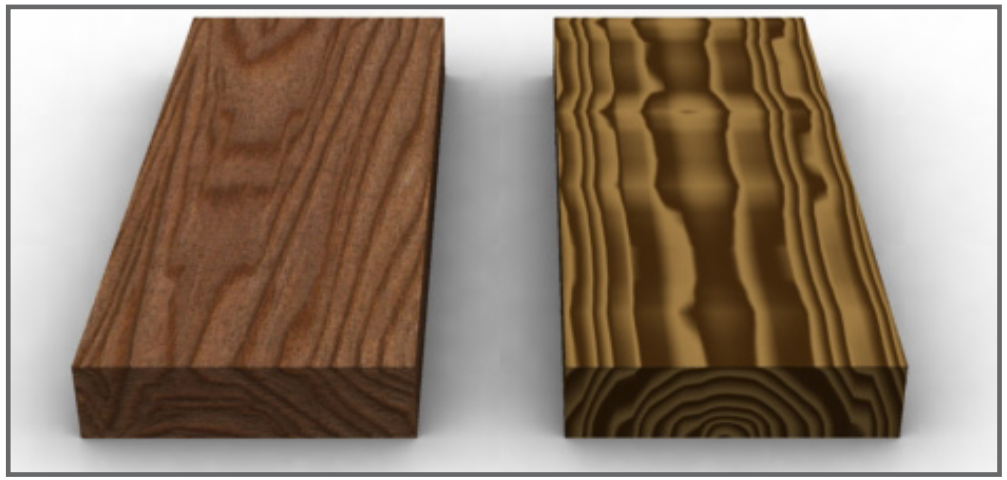
$$C(u) = \begin{cases} u, \ u \in [0; 1] \\ 0, u < 0 \\ 1, u > 1 \end{cases}$$



10.1 TEXTURE-MAPPING

Prozedurale Texturen

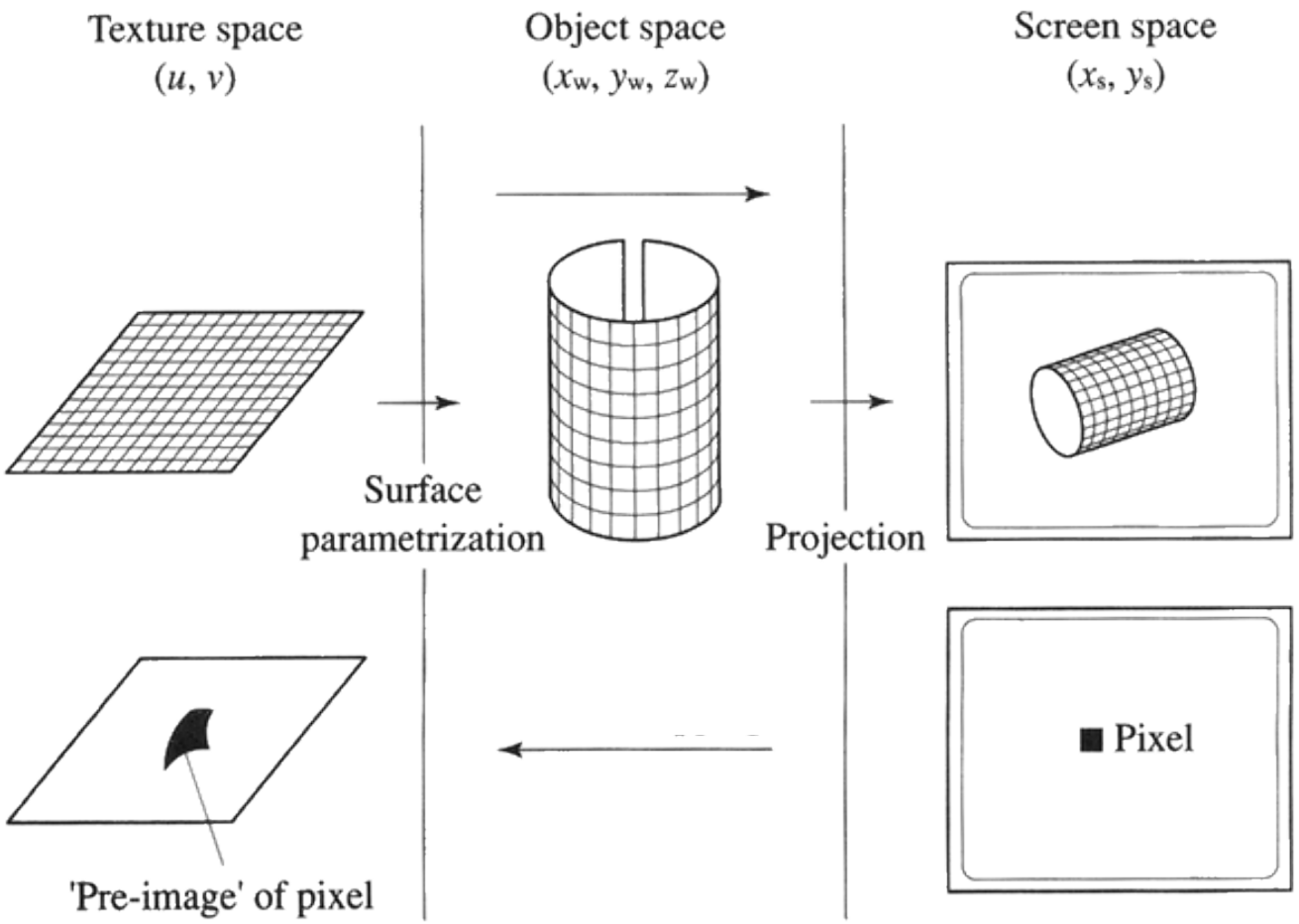
- Prozedural
  - Bei jedem Aufruf von $C_{textur}(u, v)$ wird
    - eine mathematische Formel ausgewertet
    - ein Algorithmus ausgeführt
- Vorteile
  - Minimaler Speicheraufwand
  - Texturen im gesamten Raum definiert
  - Auflösungsunabhängig
- Nachteile
  - Mathematische Beschreibung komplexer Texturen ist schwierig
  - Nicht photorealistisch



diskret prozedural

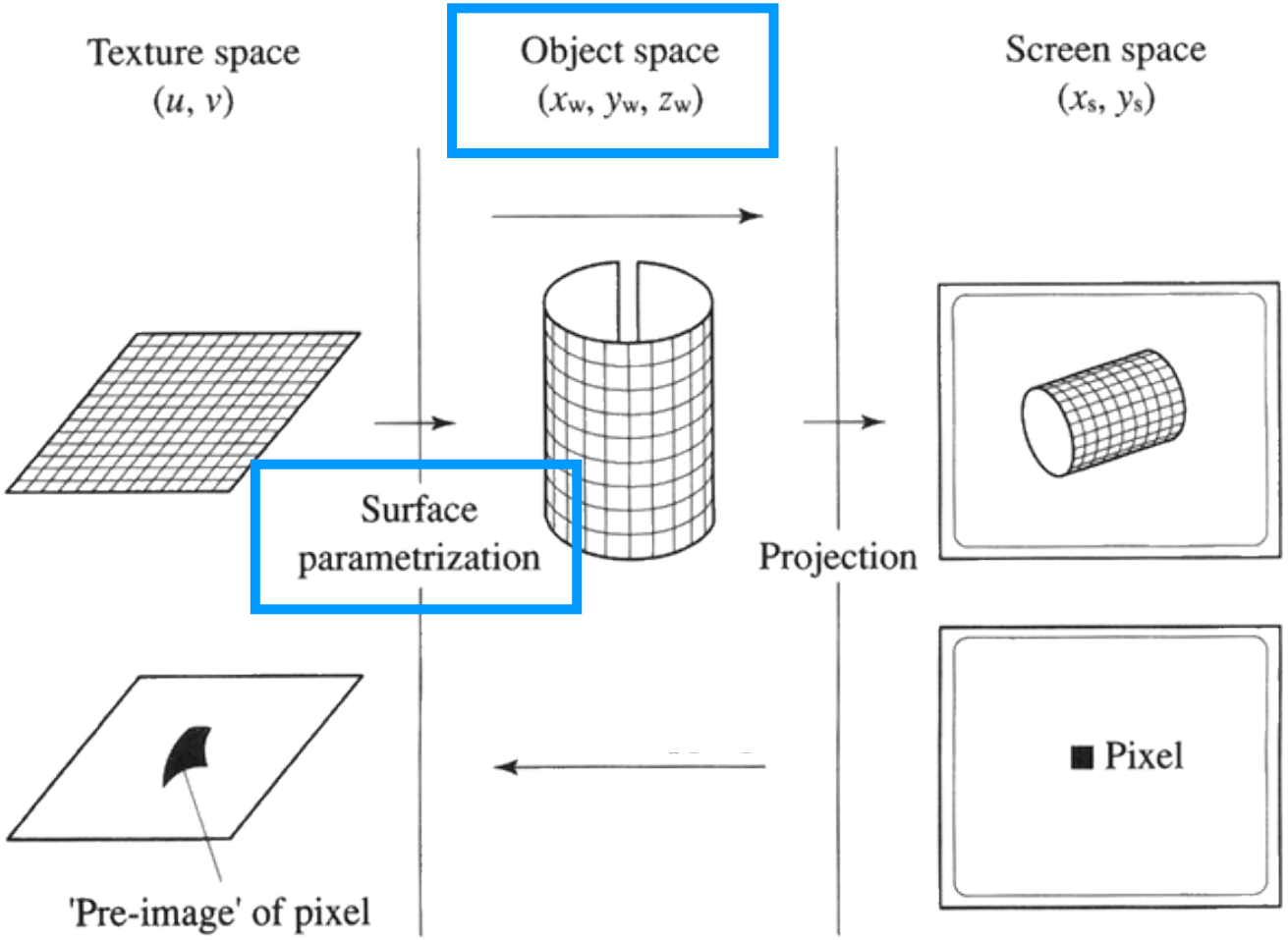
10.1 TEXTURE-MAPPING

Allgemeine Vorgehensweise



10.1 TEXTURE-MAPPING

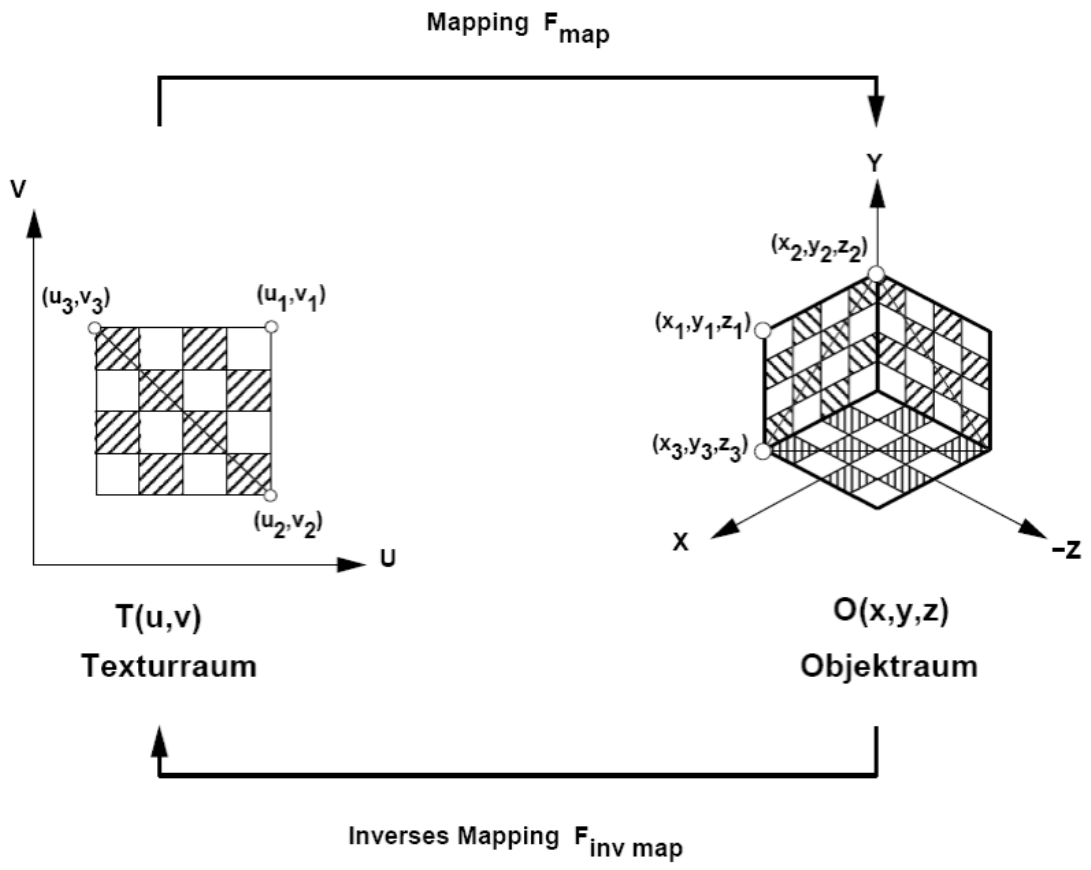
Schritt 1: Abbildung der Textur auf das Objekt



10.1 TEXTURE-MAPPING

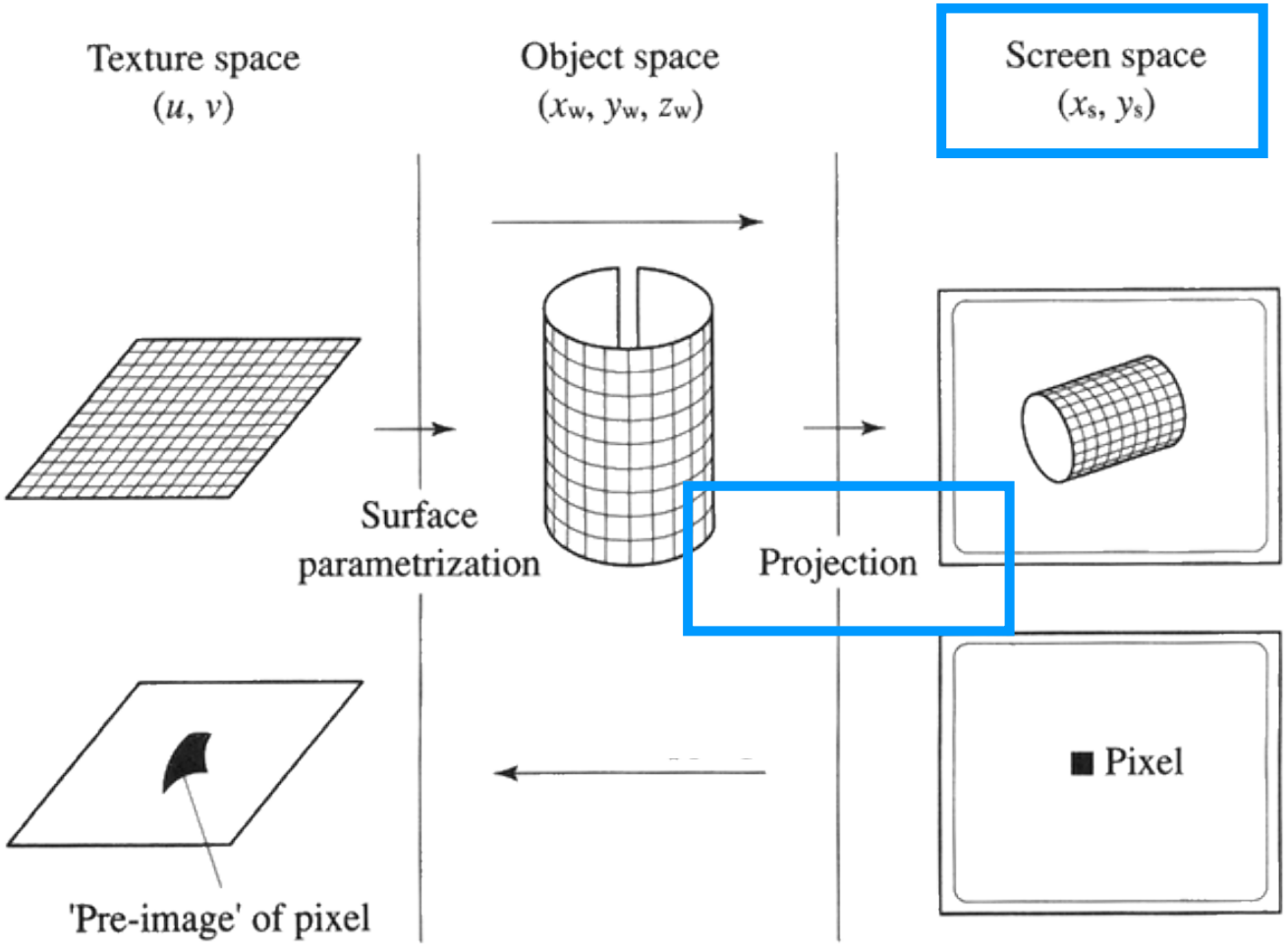
Schritt 1: Abbildung der Textur auf das Objekt

- Forward Mapping
  $$(x, y, z) = F_{map}(u, v)$$
- Zur Visualisierung muss das inverse Mapping Problem gelöst werden:
  $$(u, v) = F_{map}^{-1}(x, y, z)$$
- Texturierung entspricht mathematisch der Hintereinanderausführung von inversem Mapping und Textur-Funktion
  $$(r, g, b) = C_{texture}\left(F_{map}^{-1}(x, y, z)\right)$$



10.1 TEXTURE-MAPPING

Schritt 2: Transformation in den Bildschirmbereich

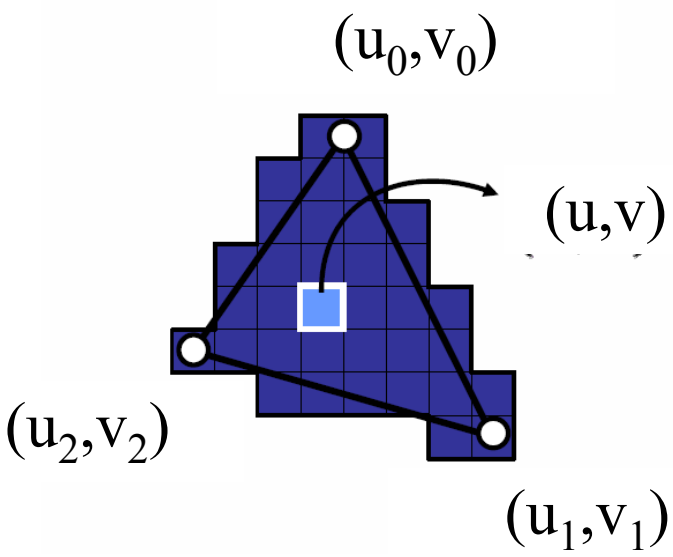


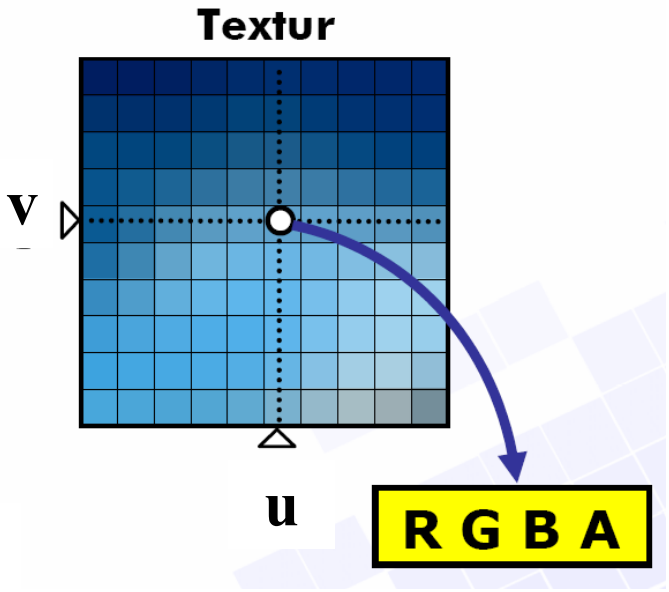
10.1 TEXTURE-MAPPING

Interpolation der Texturkoordinaten

1. Für jedes Pixel:
   - Interpolation der Texturkoordinaten

2. Texture-Lookup:
   - Interpolation der Texturwerte
   - Nearest Neighbour
   - Bilineare Interpolation



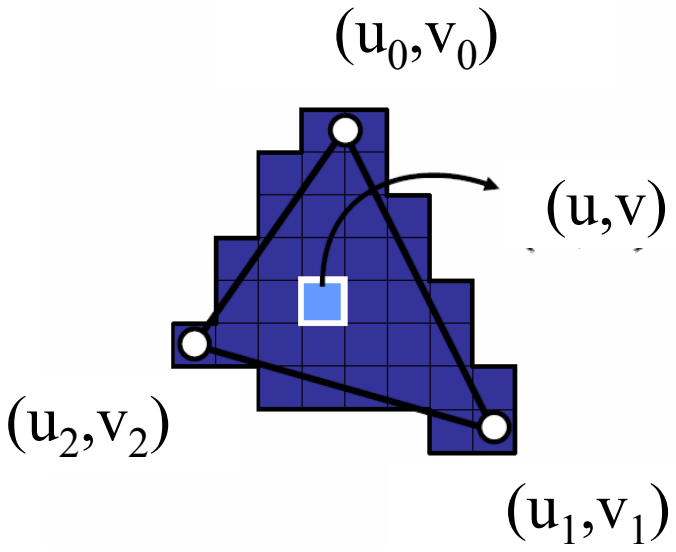


10.1 TEXTURE-MAPPING

Interpolation der Texturkoordinaten

- Abbildung Texturraum auf Objektraum
- Transformation in den Bildraum mit perspektivischer Transformation

- Inverses Mapping erweitert auf das Problem
  - Gegeben: Pixel
  - Gesucht: Punkt auf der Textur



- Information
  - Für jede Polygonecke eine Texturkoordinate

- Lösung:
  - Interpolation der Koordinaten

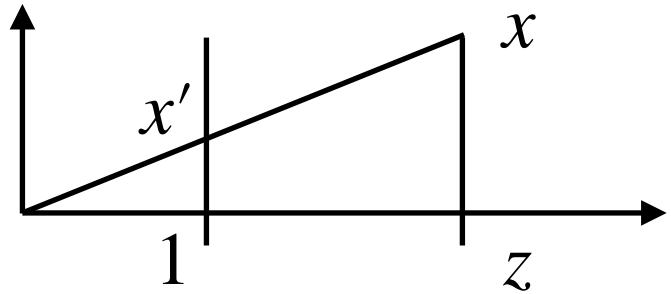
10.1 TEXTURE-MAPPING

Interpolation der Texturkoordinaten

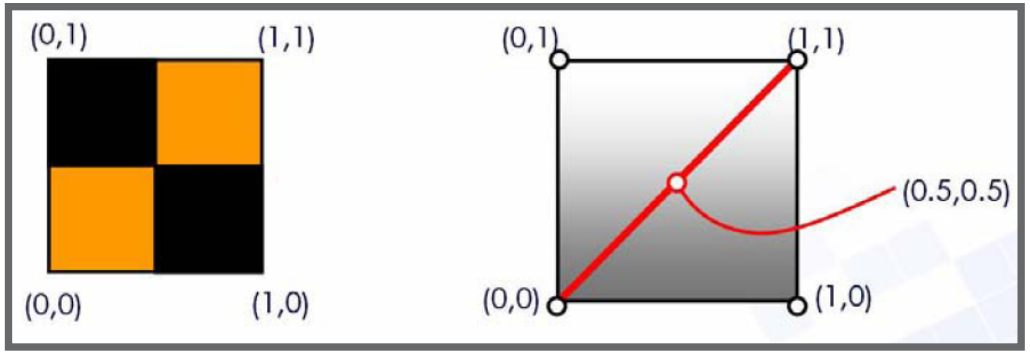
- Erste Idee:
  - Transformiere das Objekt und die Texturzuordnung bleibt dieselbe
  - Stimmt für die Eckpunkte
  - Problem bei der Interpolation, also der Zuordnung der Texturkoordinaten im Inneren eines Polygons bzw. Dreiecks
  - Texturkoordinaten müssen ebenfalls perspektivisch transformiert werden

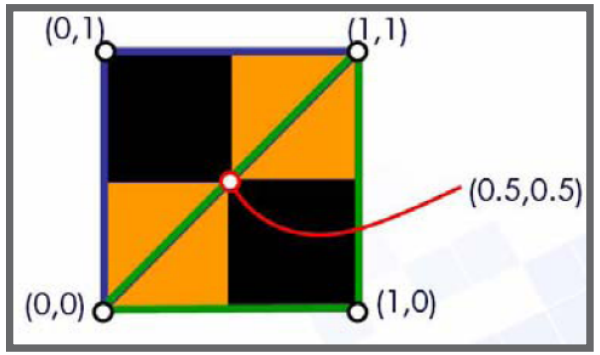
10.1 TEXTURE-MAPPING

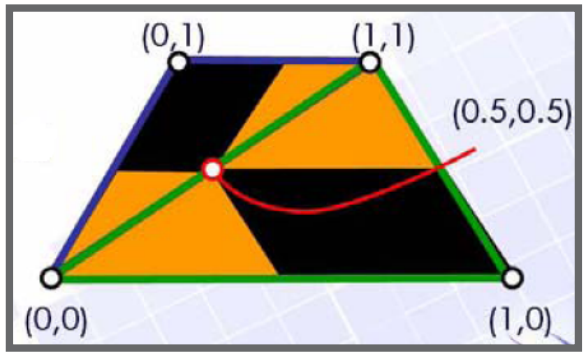
Interpolation der Texturkoordinaten

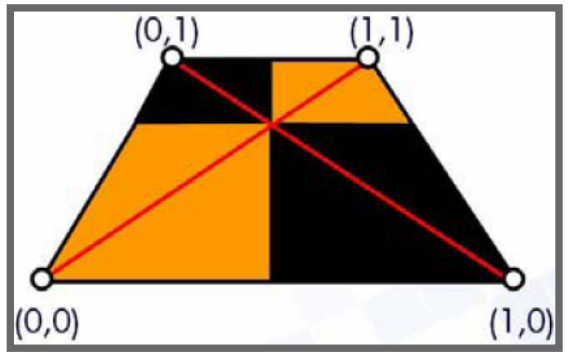


- Perspektivische Texturierung
  - Lineare Interpolation der Texturkoordinaten liefert falsches Ergebnis
  - Texturkoordinaten müssen ebenfalls perspektiv transformiert werden

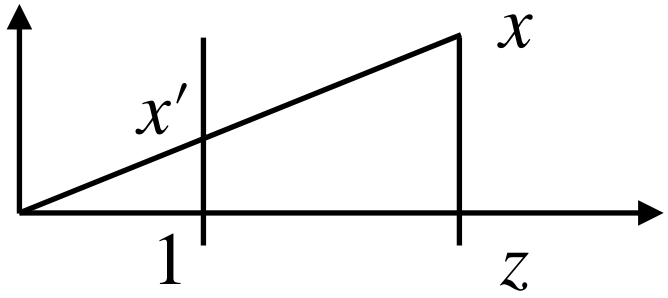








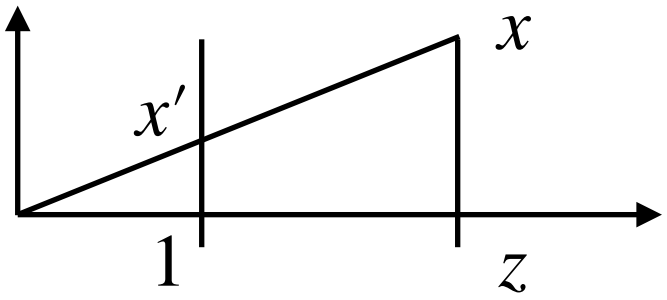
10.1 TEXTURE-MAPPING



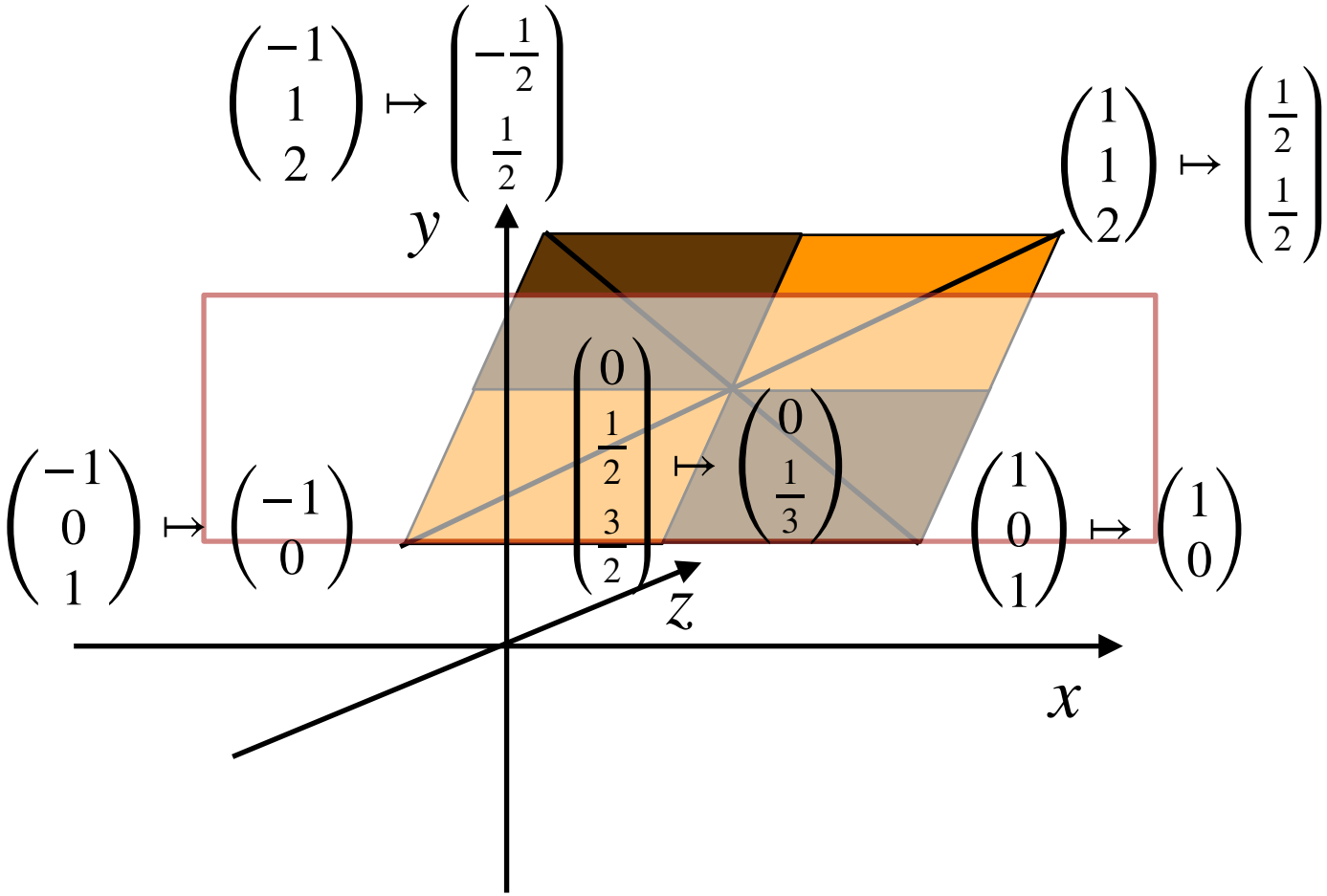
Interpolation der Texturkoordinaten

- Perspektivische Texturierung
  - Perspektivische Transformation von Punkt $p = (x, y, z, 1)$ in homogenen Koordinaten ergibt:
    $p' = (x, y, z, z)$
  - Das entspricht dem Punkt im Raum:
    $p' = (\frac{x}{z}, \frac{y}{z}, \frac{z}{z})$
  - Lineare Interpolation der Texturkoordinaten liefert falsches Ergebnis
- Lösung
  - Transformation der Texturkoordinaten $t = (u, v, 1)$ in 2D homogenen Koordinaten ergibt $t = (u, v, z)$
  - Lineare Interpolation der Texturkoordinaten innerhalb des Polygons
  - Textur-Abfrage für jedes Pixel mit rücktransformierten Texturkoordinaten
    $t' = (\frac{u}{z}, \frac{v}{z})$

10.1 TEXTURE-MAPPING

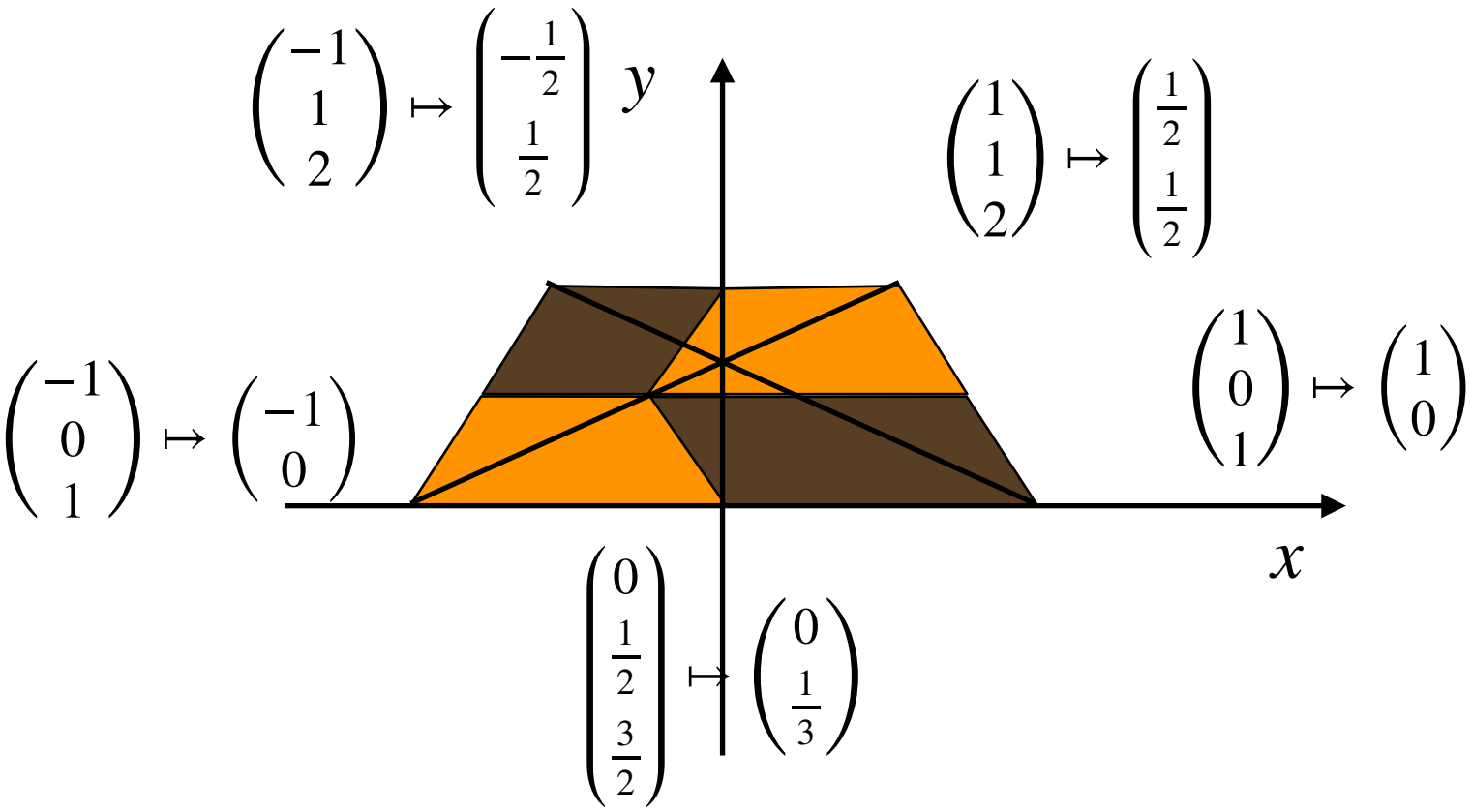


Interpolation der Texturkoordinaten



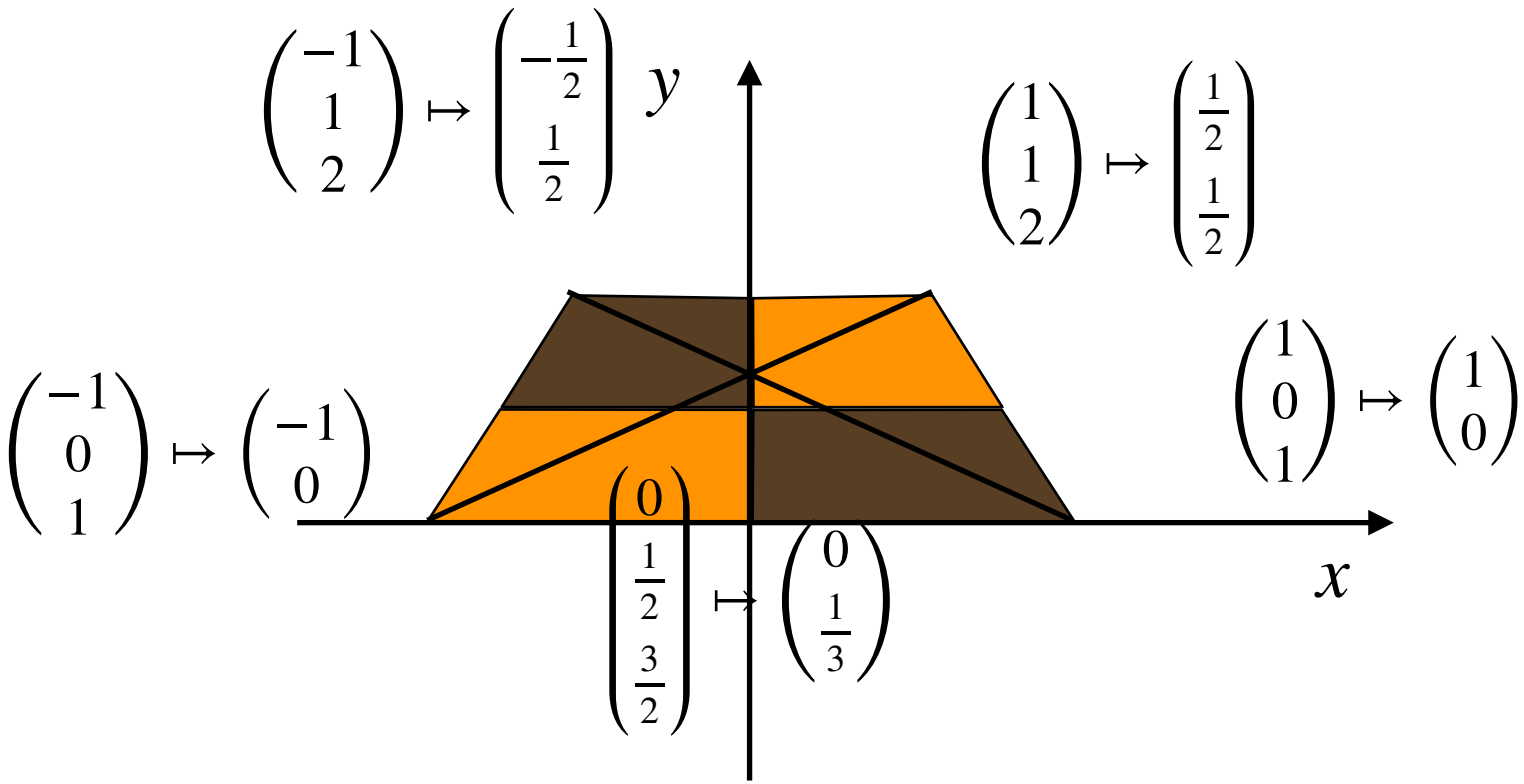
10.1 TEXTURE-MAPPING

Interpolation der Texturkoordinaten: Linear



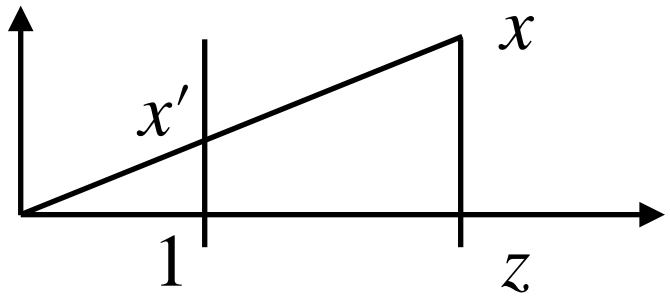
10.1 TEXTURE-MAPPING

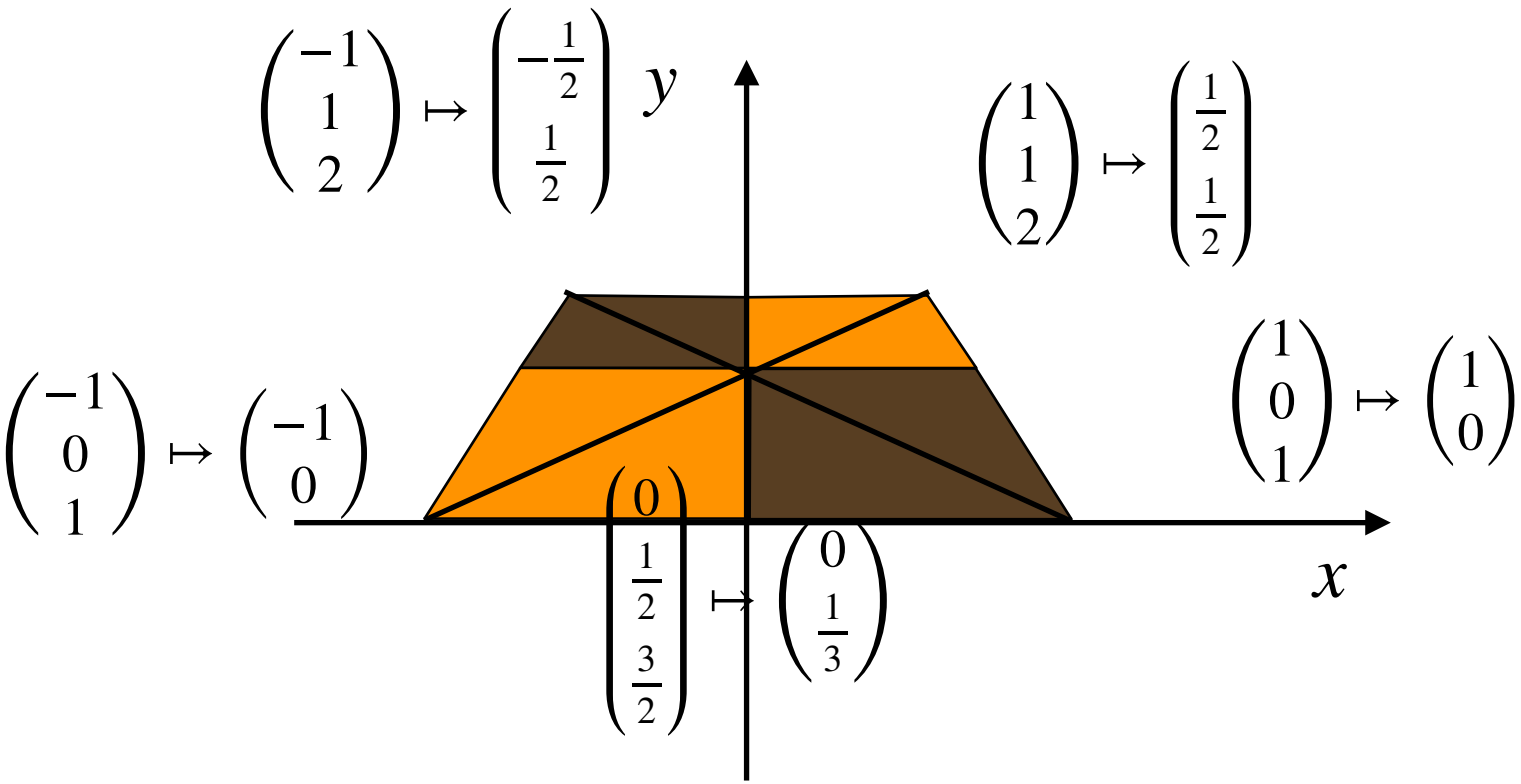
Interpolation der Texturkoordinaten: Bilinear



10.1 TEXTURE-MAPPING

Interpolation der Texturkoordinaten: Perspektivisch

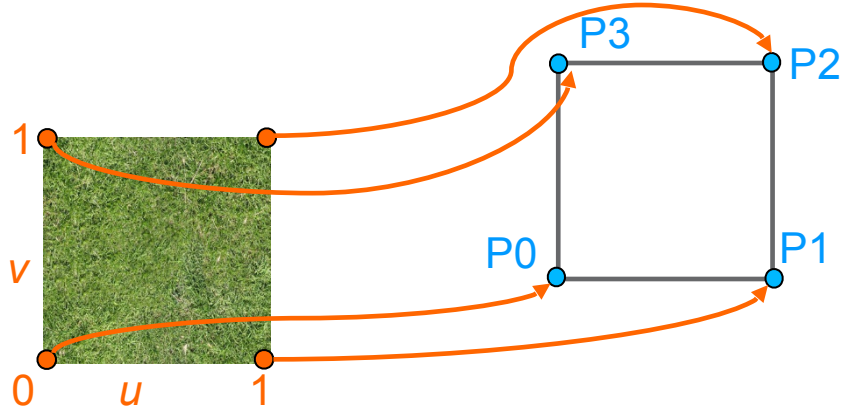




10.1 TEXTURE-MAPPING

Zuordnung von Polygonecken und Texturkoordinaten

- Bei einfachen Objekten u.U. manuell

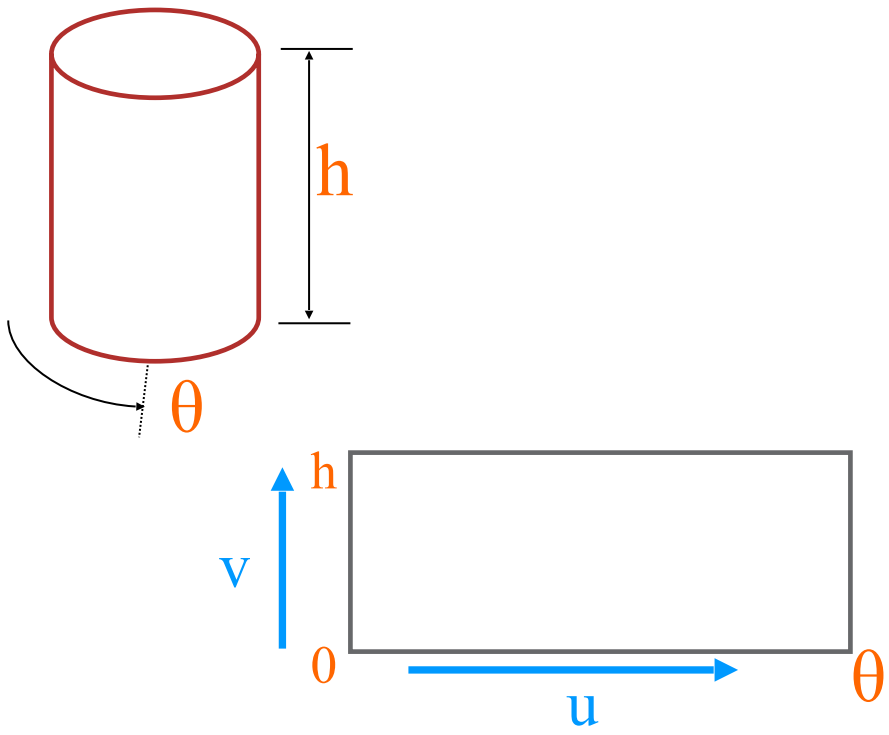


- Bei komplexeren Objekten: Two-Part Mapping
  - S-Mapping (Surface-Mapping) Abbildung der Textur auf eine einfache, virtuelle Zwischenfläche (z.B. Quader, Zylinder, Kugeln)
  - O-Mapping (Object-Mapping) Übertragung von der umhüllenden Zwischenfläche auf das zu texturierende Objekt

10.1 TEXTURE-MAPPING

Zylinder-Mapping

- Geeignet für rotationssymmetrische Objekte
- Diskontinuität an der Naht (parallel zur Achse)
- Parametrisierung $(u, v) \mapsto (\Theta, h)$



10.1 TEXTURE-MAPPING

Zylinder-Mapping

- Geeignet für rotationssymmetrische Objekte
- Diskontinuität an der Naht (parallel zur Achse)
- Parametrisierung $(u, v) \mapsto \left(\Theta, h\right)$

10.1 TEXTURE-MAPPING

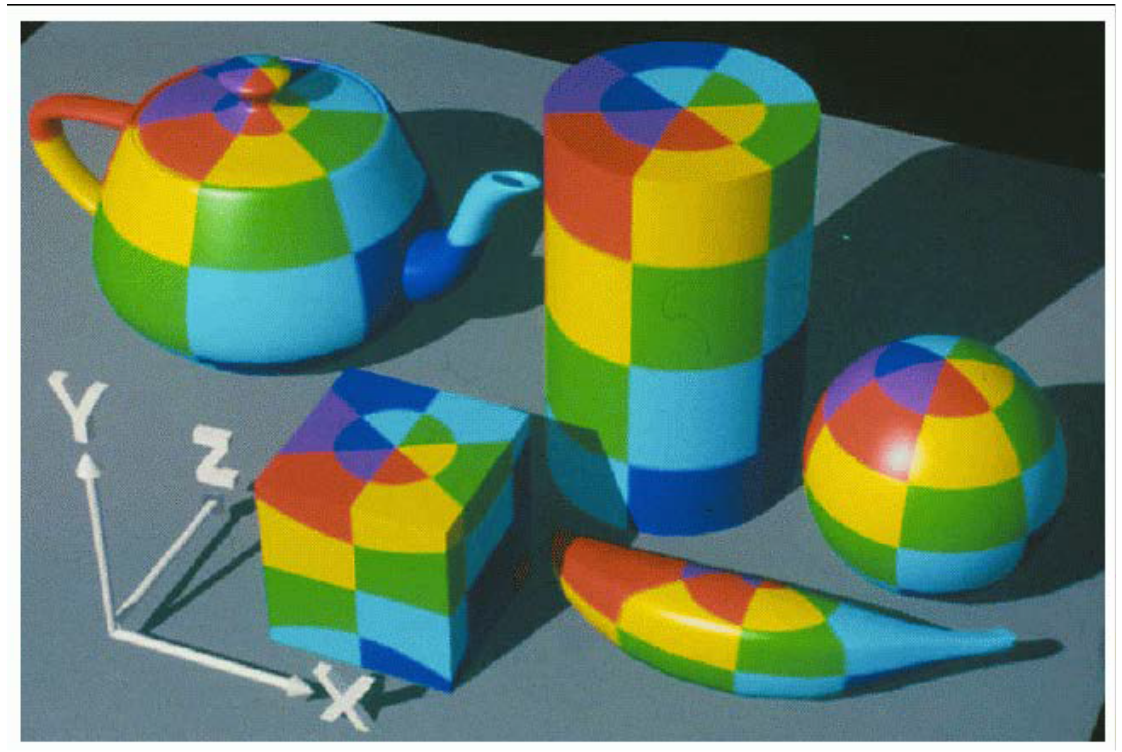
Kugel-Mapping

- Geeignet für Kugel-symmetrische Objekte
- Verzerrung, vor allem an den Polen
- Parametrisierung durch Kugelkoordinaten $(u, v) \mapsto (\Phi, \Theta)$

10.1 TEXTURE-MAPPING

Kugel-Mapping

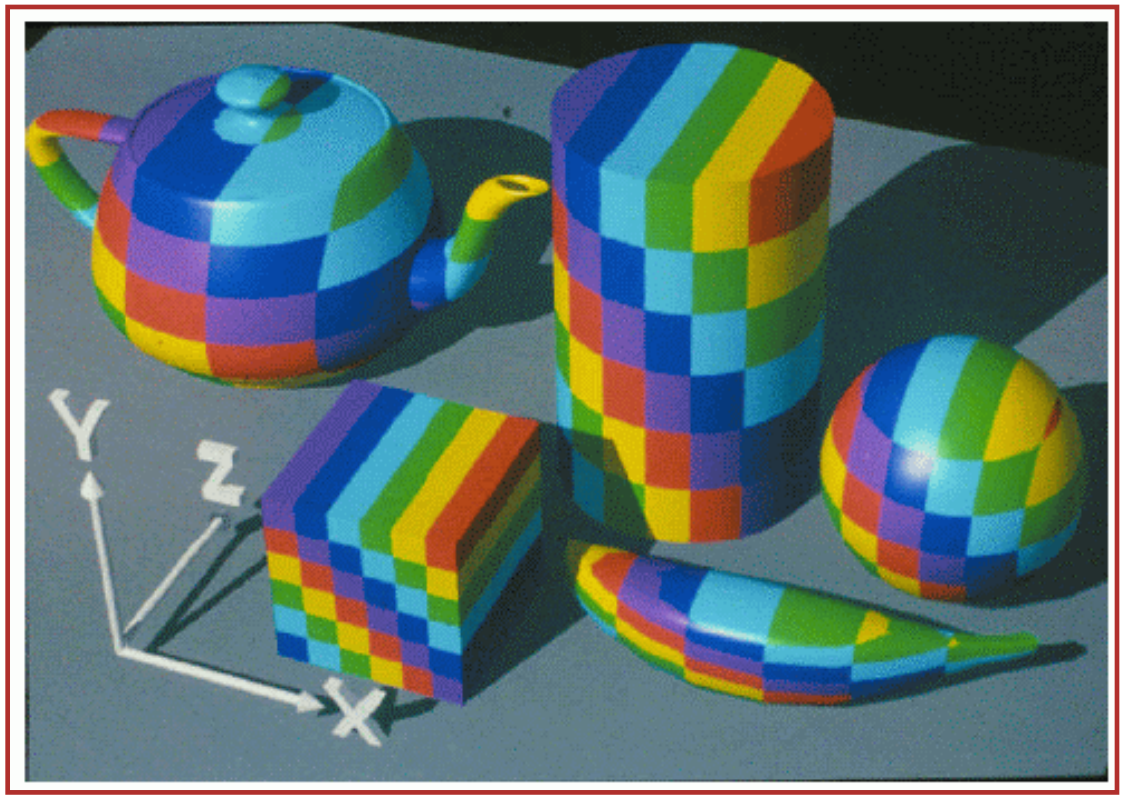
- Geeignet für Kugel-symmetrische Objekte
- Verzerrung, vor allem an den Polen
- Parametrisierung durch Kugelkoordinaten $(u, v) \mapsto (\Phi, \Theta)$



10.1 TEXTURE-MAPPING

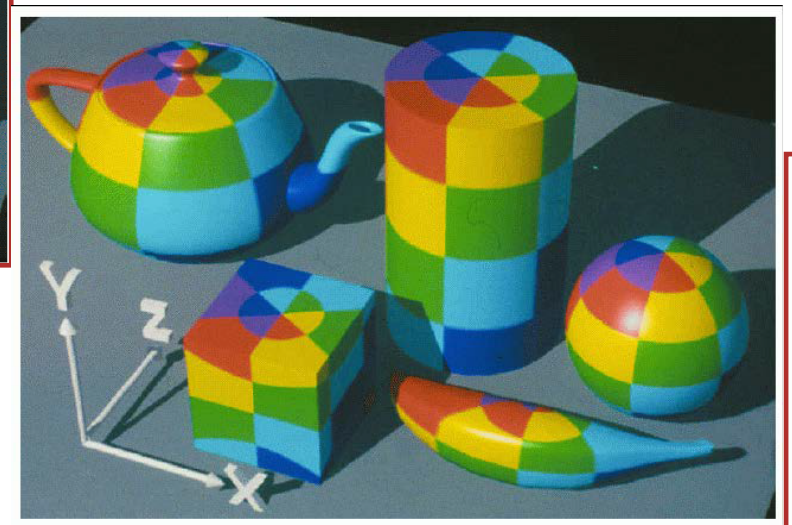
Box-Mapping

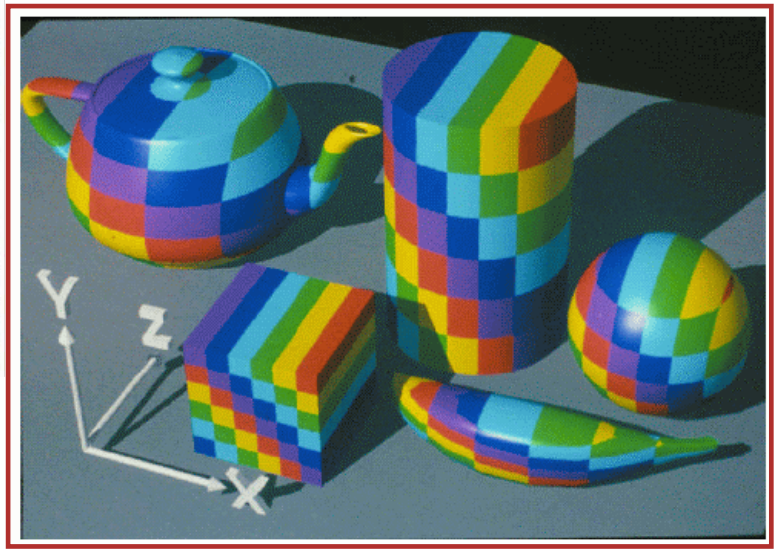
- Quader als umhüllende Fläche
  - meist achsenparallele Bounding-Box des Objekts
- Mögliche Parametrisierung
  - u-Achse:
    Längste Kante des Quaders
  - v-Achse:
    Zweitlängste Kante des Quaders



10.1 TEXTURE-MAPPING

Zylinder – Kugel – Box-Mapping

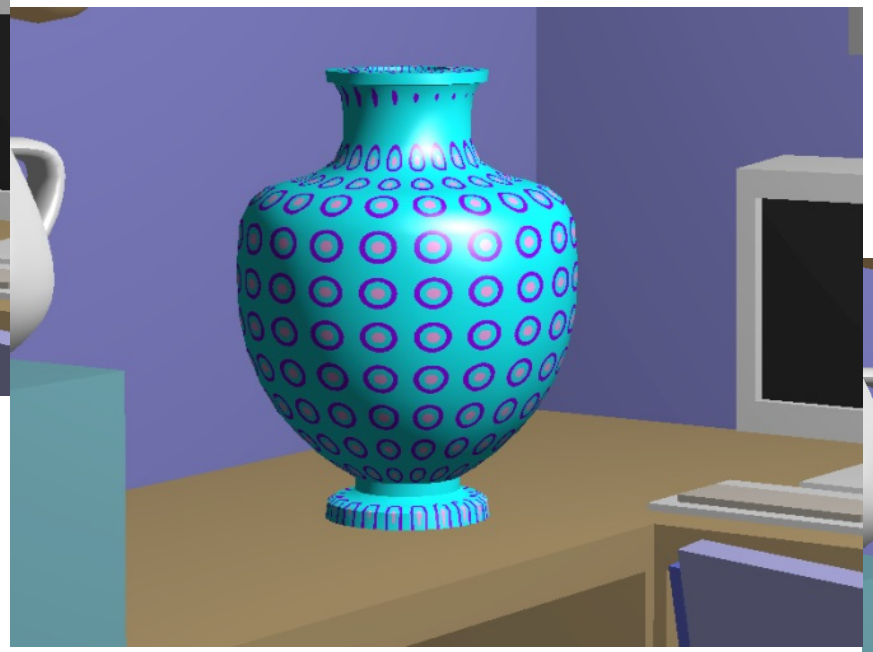




10.1 TEXTURE-MAPPING

Verwendung verschiedener Zwischenobjekte

Box



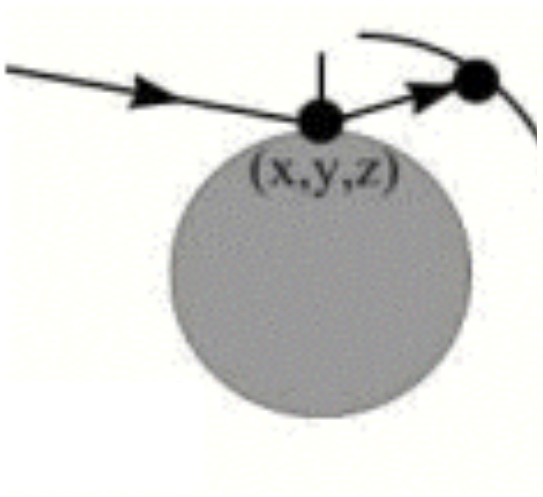
Kugel

Zylinder

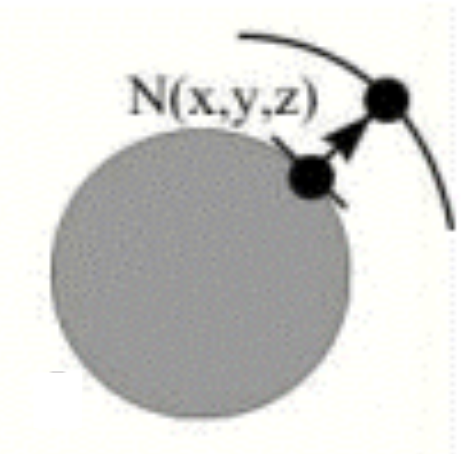
10.1 TEXTURE-MAPPING

Techniken des O-Mapping

Reflexionsstrahl

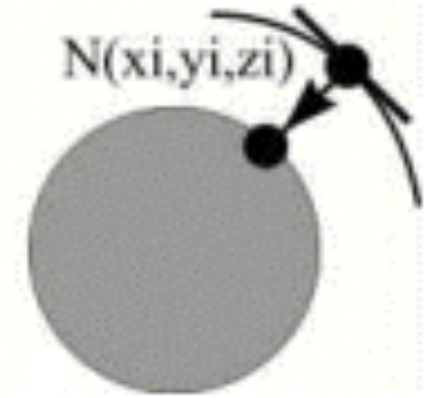


Normalenvektor



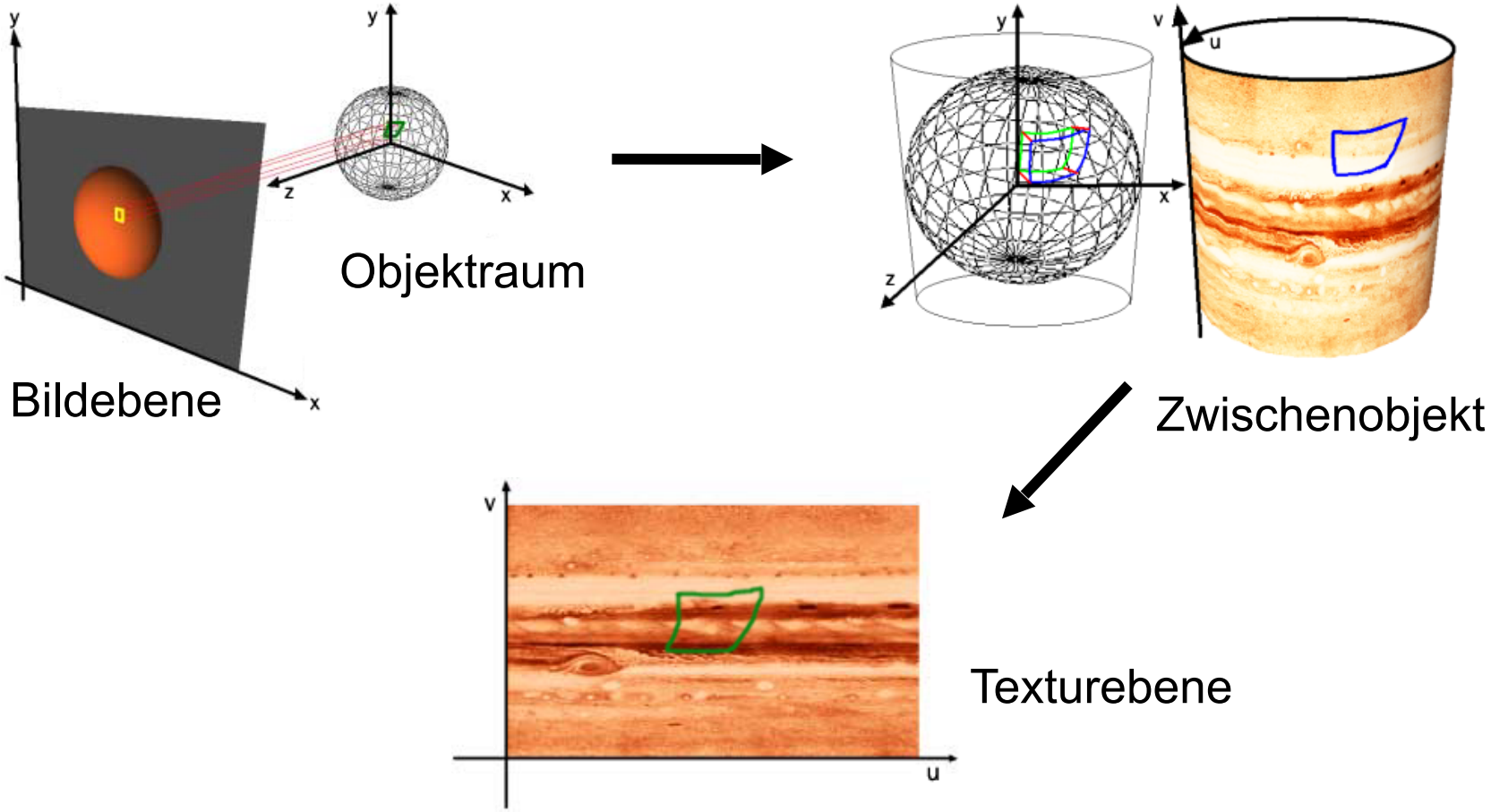
Objektzentrum

Hilfsobjektnormale



10.1 TEXTURE-MAPPING

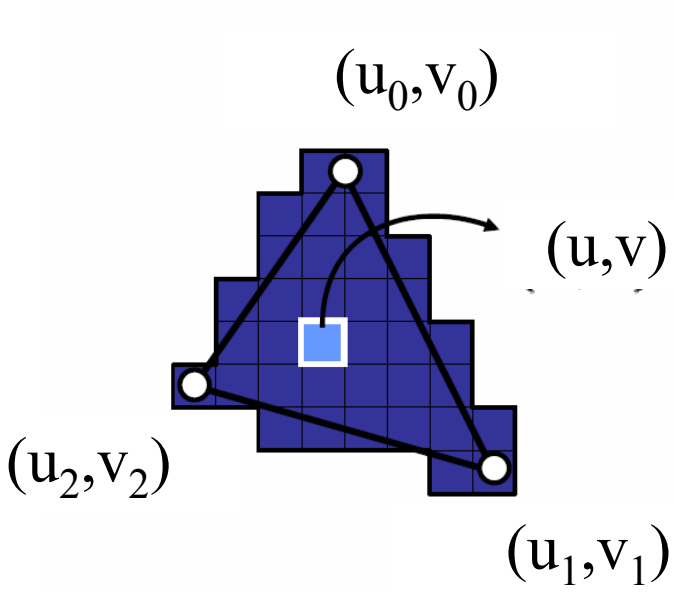
Inverse Abbildung mit Zwischenobjekt



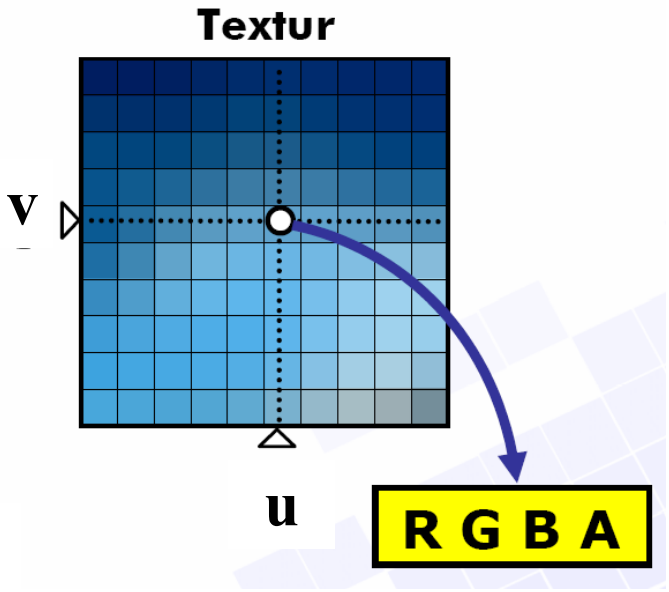
10.1 TEXTURE-MAPPING

Abtastprobleme

- Welche Probleme können bei der Abbildung von Pixel- auf Textur-Raster auftreten?



Pixel-Raster

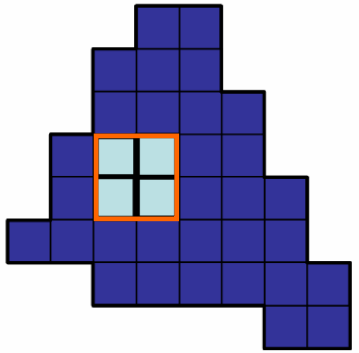


Textur-Raster

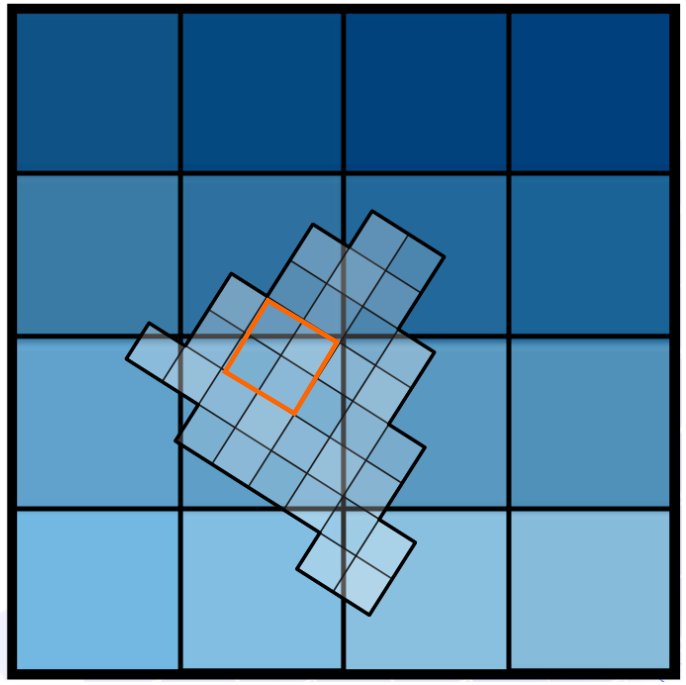
10.1 TEXTURE-MAPPING

Abtastprobleme

- Magnification / Oversampling
  - Pixel-Raster ist feiner als Textur-Raster
  - Ein Texel wird auf mehrere Pixel abgebildet.
  - Texturen erscheinen daher verschwommen.
- Abhilfe
  - Auflösung der Textur erhöhen



Pixel-Raster

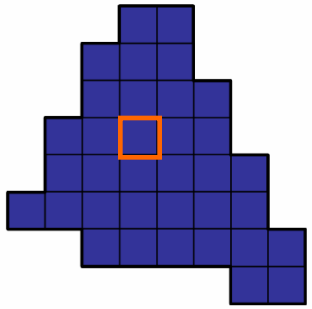


Textur-Raster

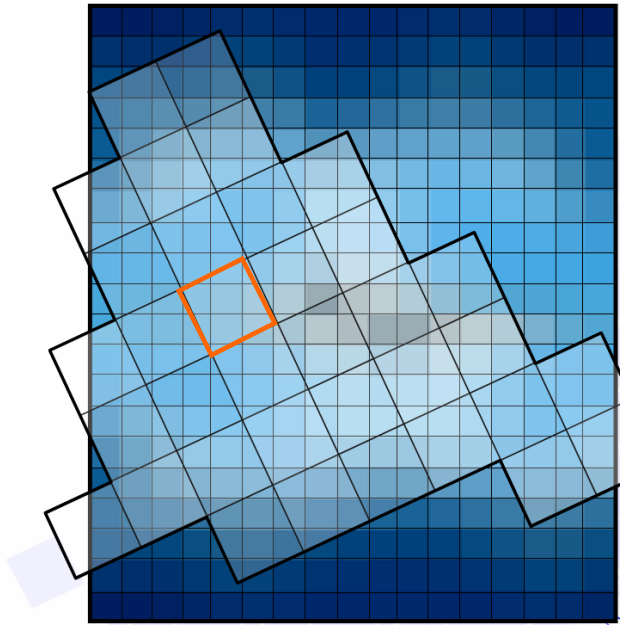
10.1 TEXTURE-MAPPING

Abtastprobleme

- Minification / Undersampling
  - Pixel-Raster ist gröber als Textur-Raster
  - Ein Pixel wird auf mehrere Texel abgebildet
  - Aliasing, da Abtastfrequenz zu gering
- Abhilfe
  - Nach Fläche gewichteten Mittelwert der Texel bestimmen
  - Exakte, aber zu aufwändige Lösung



Pixel-Raster



Textur-Raster

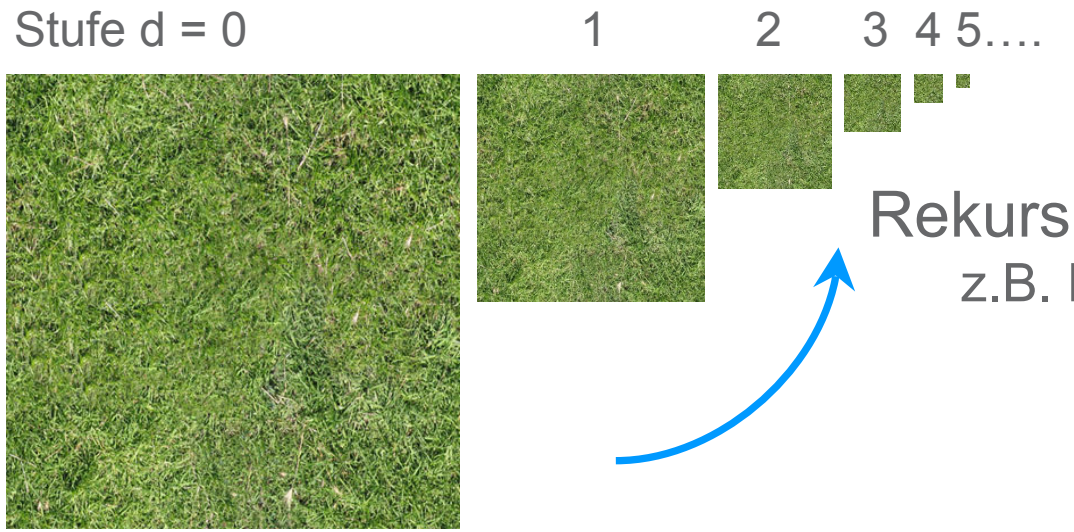
10.1 TEXTURE-MAPPING

Mip-Mapping

- Auflösung der Textur sollte der im Bildraum entsprechen
- Problematisch, wenn Auflösung im Bildraum variiert
- Lösung
  - Textur in verschiedenen Auflösungen bereithalten
  - Quadratische Texturen mit Kantenlänge 2^k (Zweierpotenz)

10.1 TEXTURE-MAPPING

Mip-Mapping



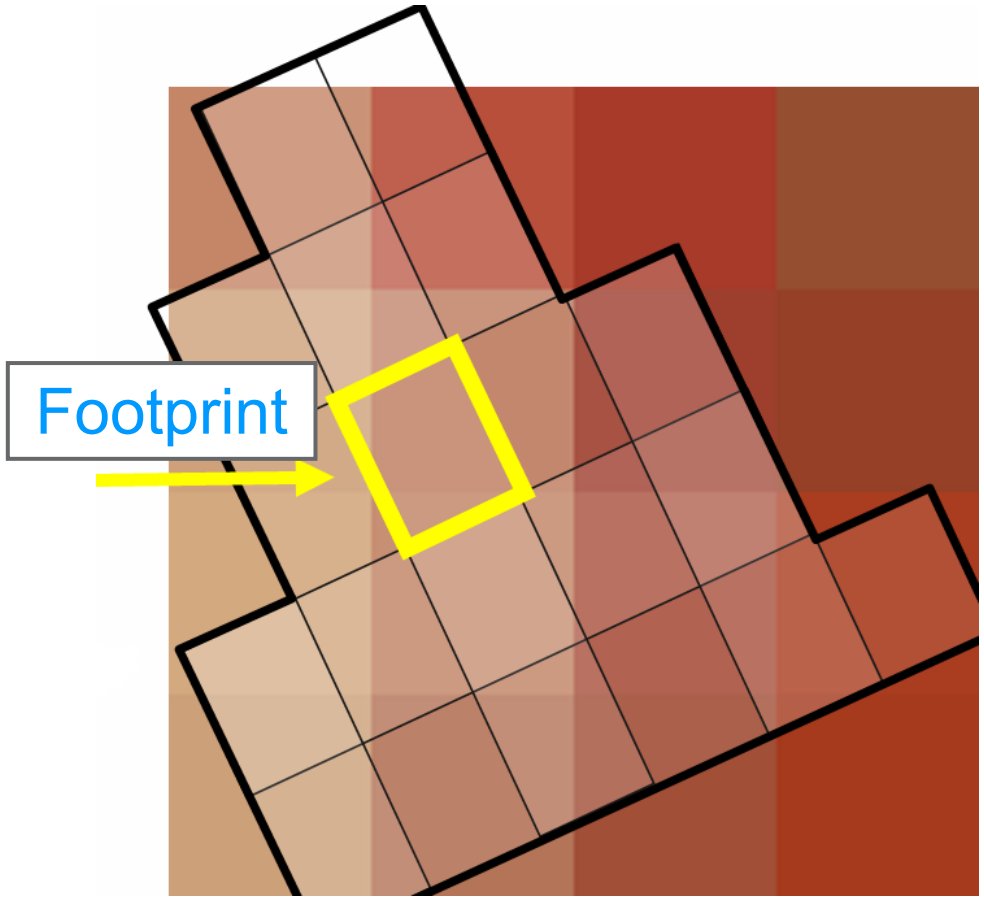
Rekursives Downsampling

z.B. Mitteln von 4 Pixeln

10.1 TEXTURE-MAPPING

Mip-Mapping

- Verfahren
  - Bestimme Footprint des Pixels auf der Textur
  - Längste Kante bestimmt Mipmap-Stufe d
  - Interpoliere Texel aus Textur der Stufe d
- Ergebnis
  - Antialiasing, da der verwendete Texel immer in etwa der Größe des Footprints entspricht



10.1 TEXTURE-MAPPING

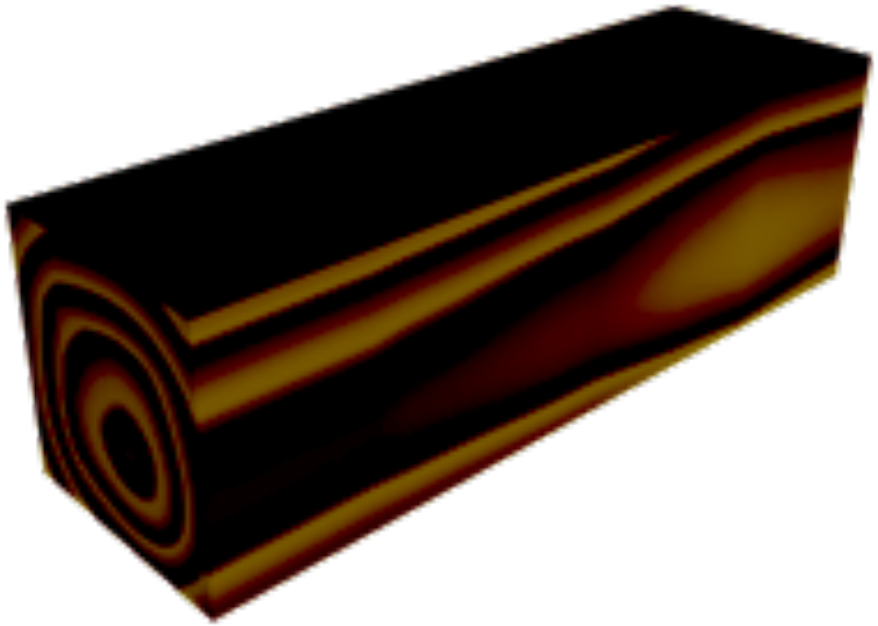
3D-Texturen / Festkörpertexturen

- Handhabung analog zu 2D-Texturen
- (Anschaulicher) Unterschied zu 2D-Texturen
  - Objekt wird aus Textur ausgeschnitten statt damit beklebt
  - Prozedurale Ansätze erlauben wirklichkeitsgetreue 3D-Muster

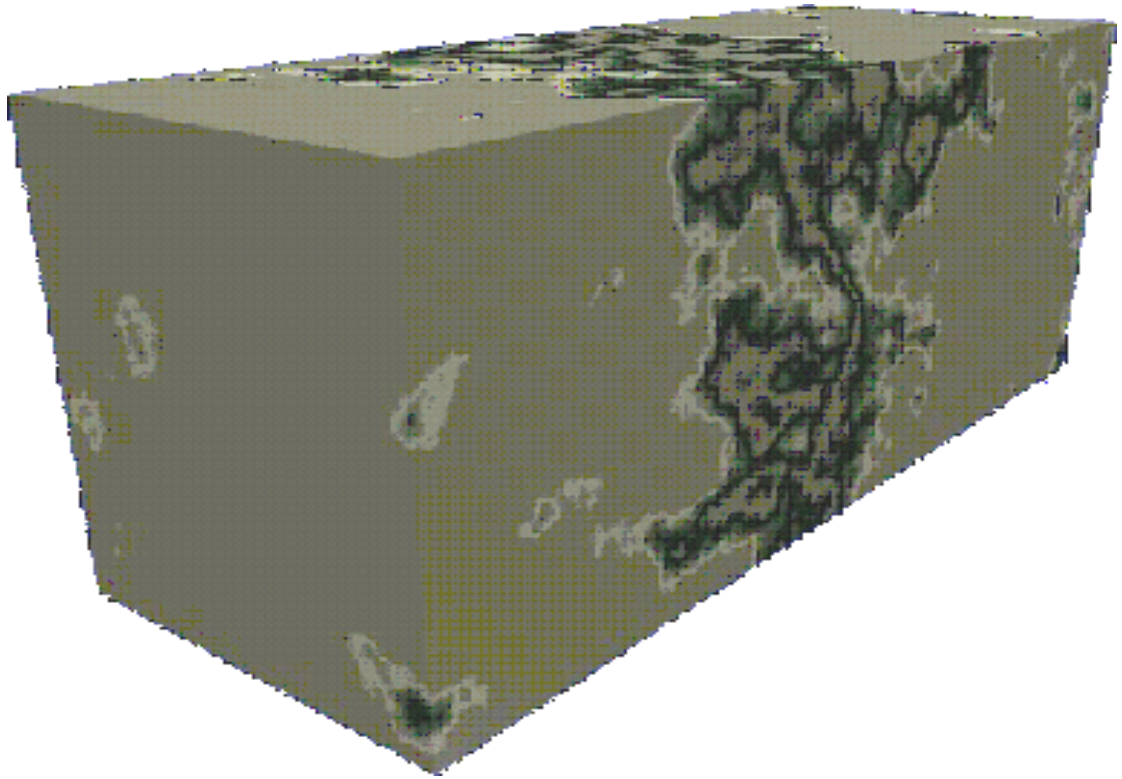
10.1 TEXTURE-MAPPING

3D-Texturen / Festkörpertexturen

- Holzmaserung

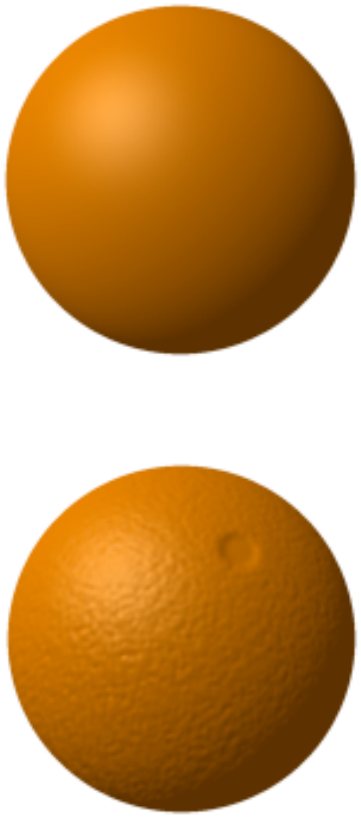


- Perlin Marmor



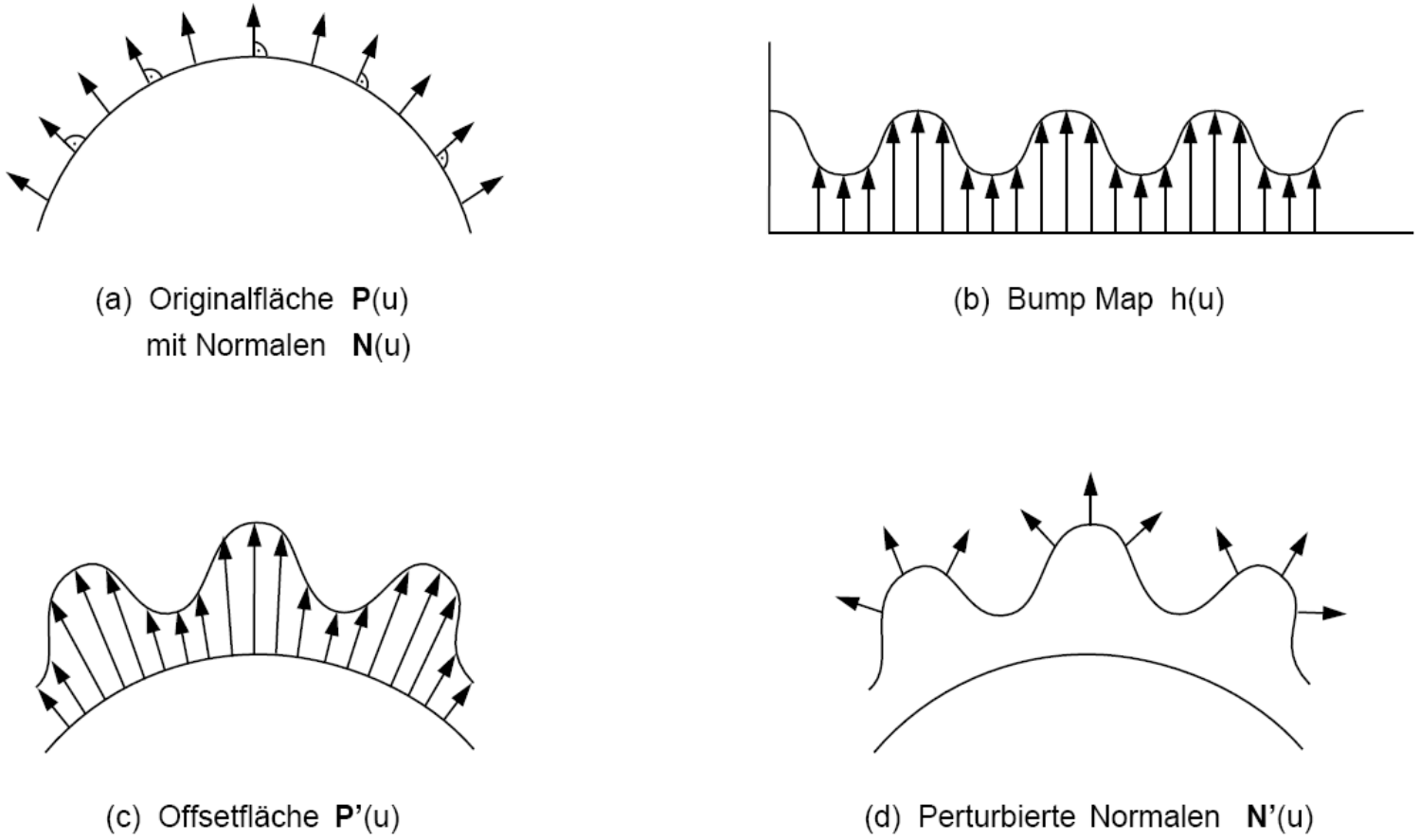
10.2 BUMP-MAPPING

- James Blinn 1978
- Ziel
  - Rauere und plastischere Erscheinung eines Objekts
  - Keine Veränderung der Geometrie
- Simulation von Oberflächenunebenheiten durch
  - Manipulation der Normalenvektoren
  - Struktureffekt nur durch Beleuchtung



10.2 BUMP-MAPPING

Exakte Modellierung



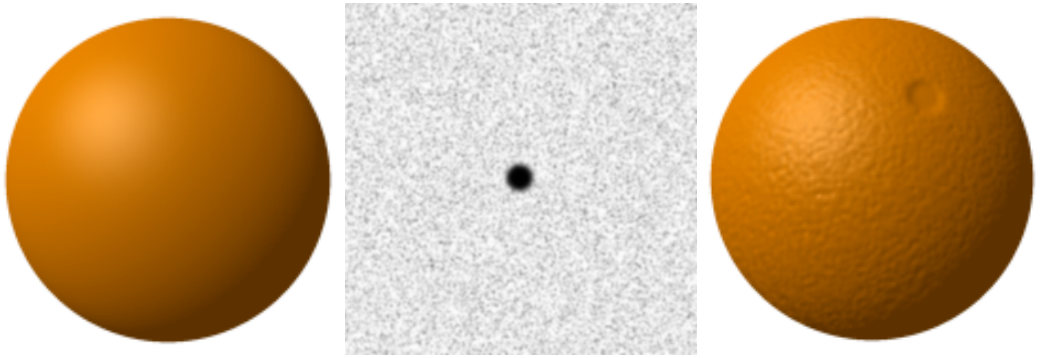
10.2 BUMP-MAPPING

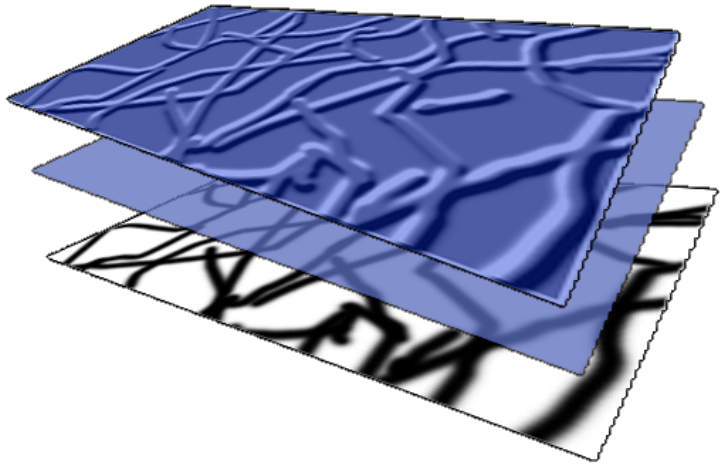
- Prinzip
  - Für die Beleuchtung ist nur die Normale relevant
  - Die Lage des Punktes auf der Fläche spielt keine Rolle
  - Originalgeometrie mit veränderten Normalen kombinieren
- Anmerkungen
  - Bump-Mapping nur mit Verfahren möglich, die Beleuchtung explizit in jedem Flächenpunkt auswerten
    - Phong-Shading
    - Raytracing
  - Nicht
    - Gouraud
  - Schatten und Silhouetten der Objekte bleiben glatt

10.2 BUMP-MAPPING

- Umsetzung
  - Prozedurale Veränderung der Normalen
  - Normalen als RGB-Textur gegeben (Normal Map)
  - Höhenfeld als 2D-Skalarfeld (Grauwerttextur) gegeben
    - Addition des Höhenfeldes auf die Geometrie
    - Berechnung der Normalen über Richtungsableitungen dieser Offsetfläche

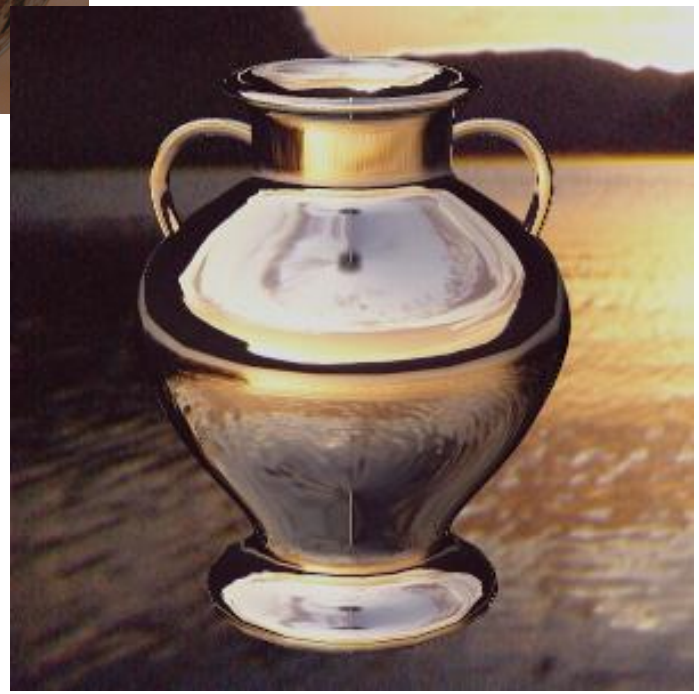
      $N' = P'_u \times P'_v$





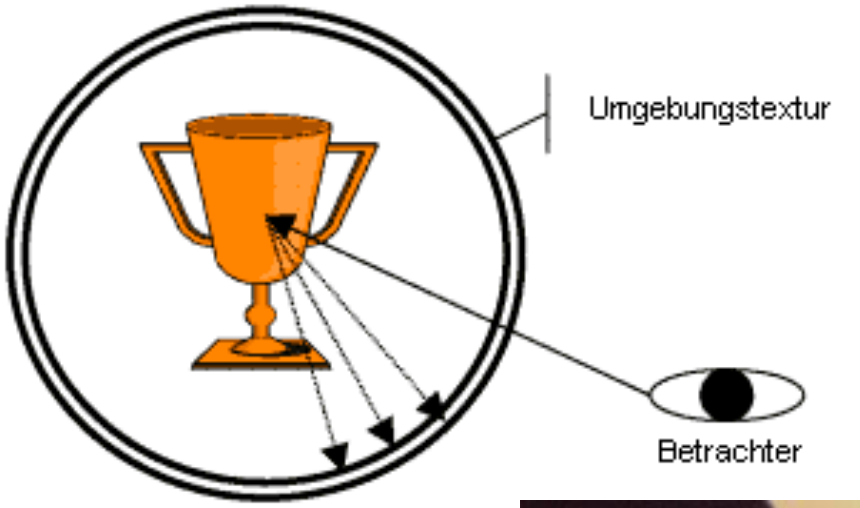
10.3 ENVIRONMENT-MAPPING

- Modellierung von Interobjekt-Reflexionen
- Abbildung einer komplexen Umgebung auf ein spiegelndes Objekt mit Hilfe von Texturen
- Günstige Alternative zum Raytracing
- Keine Verdeckungsrechnung
- Anwendung im Real-Time Rendering



10.3 ENVIRONMENT-MAPPING

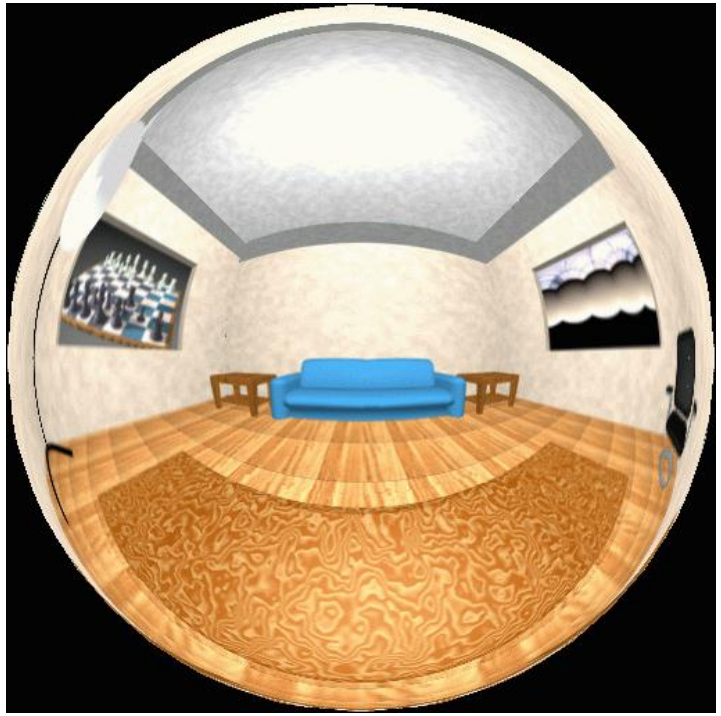
- Verfahren
  - Zwischenobjekt (Würfel, Kugel) umhüllt Objekt
  - Die Umgebungstextur wird auf die Innenseite des Zwischenobjekts aufgetragen
  - Die Texturkoordinaten sind abhängig vom Reflexionsvektor



10.3 ENVIRONMENT-MAPPING

Sphere Mapping

- Unterstützt von OpenGL
- Abbildung einer umhüllenden Kugel auf eine 2D-Textur



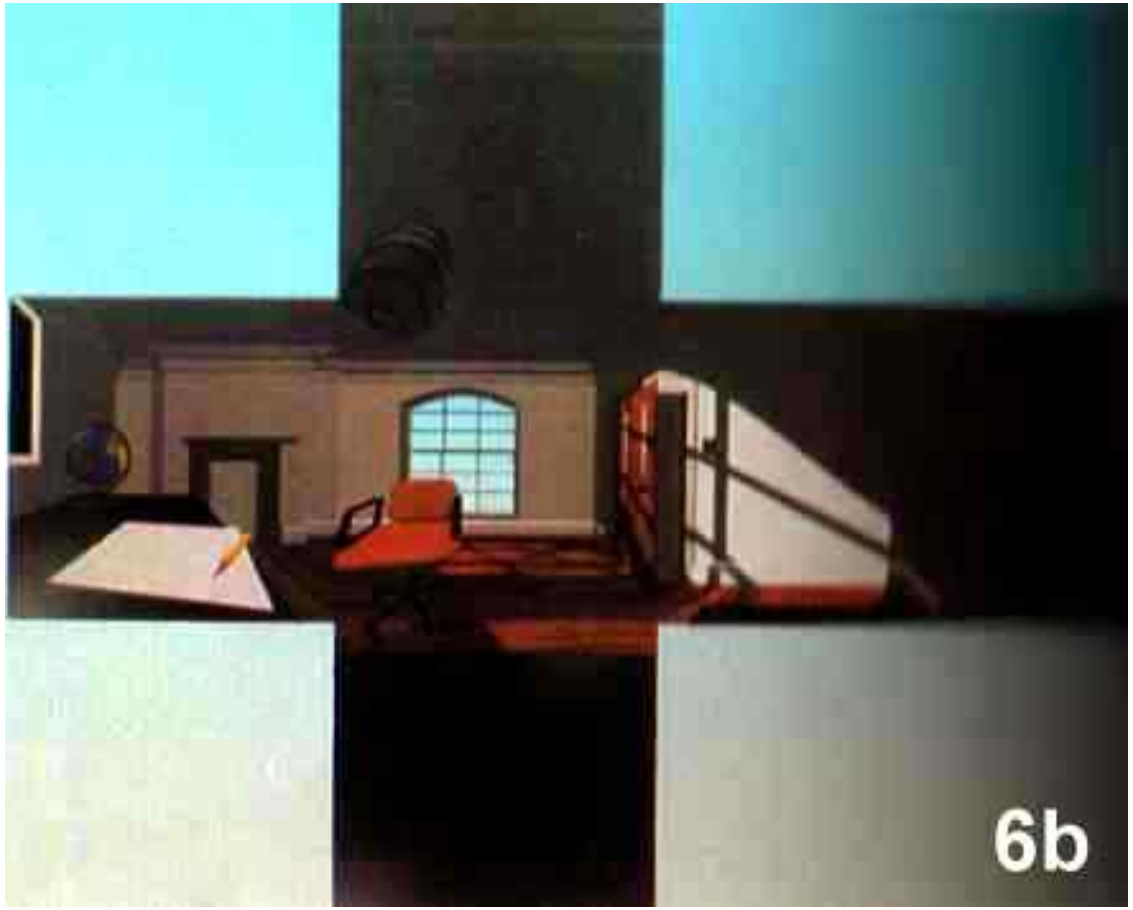
10.3 ENVIRONMENT-MAPPING

Sphere Mapping

- Probleme
  - Ebenfalls schwierige Erstellung
    - Rendering
    - Foto einer spiegelnden Kugel
    - Umrechnung aus ebenen Texturen (Würfel)
  - Abtastrate über Textur nicht regelmäßig
  - Interpolation der Texturkoordinaten führt zu Fehlern, vor allem am Rande der Kugel

10.3 ENVIRONMENT-MAPPING

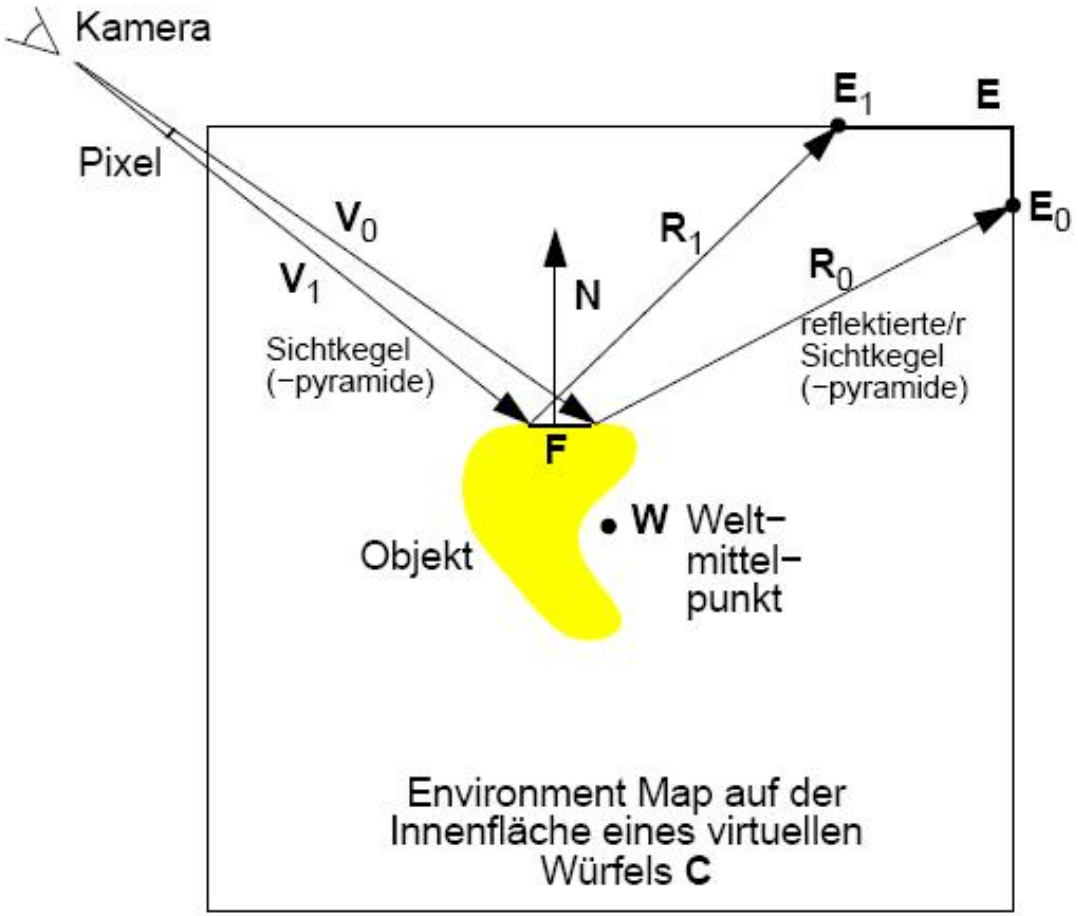
Verwendung eines Würfels



10.3 ENVIRONMENT-MAPPING

Verwendung eines Würfels

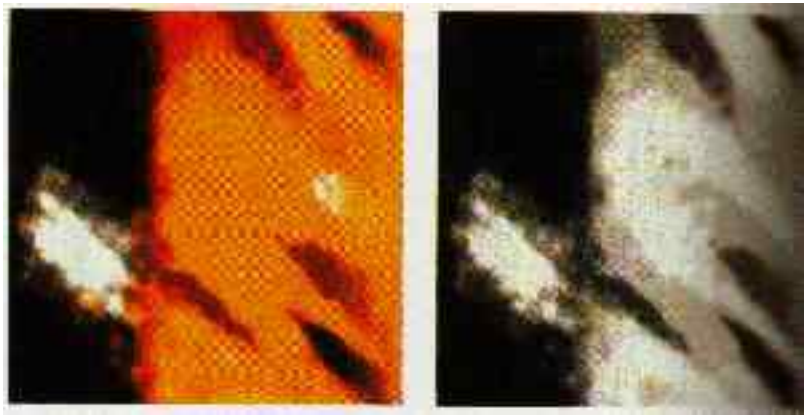
- Einfachere Erstellung der Textur
- Anti-Aliasing durch Sichtpyramide
- Parametrisierung
  - besser als bei einer Kugel
  - immer noch schlecht



10.4 WEITERE VERFAHREN

Chrome-Mapping

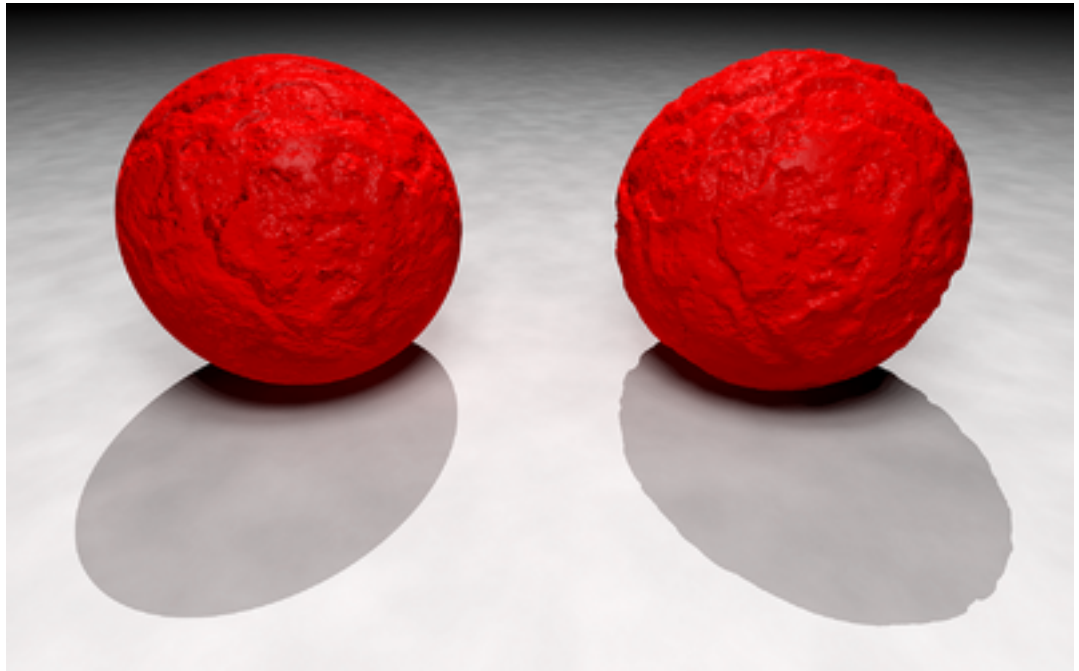
- Abbildung eines willkürlichen Musters (chrome map) auf eine reflektierende Oberfläche
- Künstliche Unschärfe
- Textur fest im Raum (daher geeignet für Animationen)



10.4 WEITERE VERFAHREN

Displacement Mapping

- Ziel
  - Plastischere Erscheinung der Oberfläche
- Unterschied zum Bump-Mapping:
  - Verschieben der Oberflächenpunkte gemäß Höhenfeld entlang der Normalen
  - Meist Kombination mit Textur
- Vorteil
  - Schatten & Silhouette erscheinen nicht mehr glatt



10.4 WEITERE VERFAHREN

Displacement Mapping

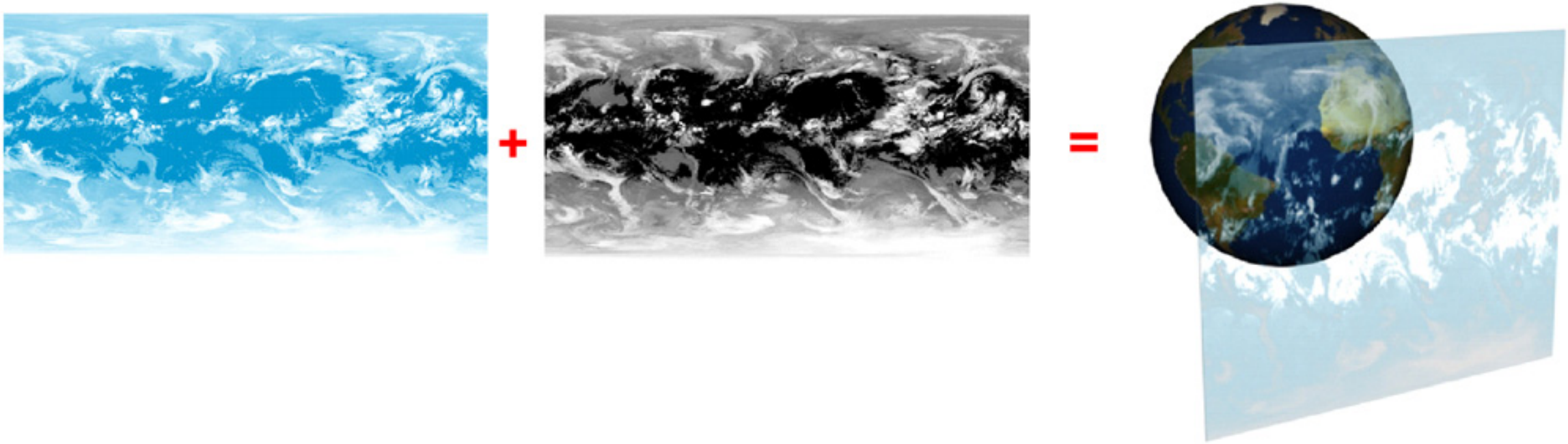
10.4 WEITERE VERFAHREN

Opacity-Mapping / Transparency-Mapping

- Ziel
  - Transparenz von Objekten lokal kontrollieren
- Opacity-Map
  - Grauwerte entsprechen Alpha-Wert
  - Von schwarz (opak) bis weiß (transparent)

10.4 WEITERE VERFAHREN

Opacity-Mapping / Transparency-Mapping



ZUSAMMENFASSUNG

- Erzielte Effekte
  - Veränderte Maserungen und Muster einer glatten Fläche
    - Texture-Mapping
    - Chrome-Mapping
  - Modellierung rauer Oberflächen
    - Bump-Mapping
    - Displacement-Mapping
  - Abbildung komplexer Umgebung auf Objekte
    - Environment-Mapping
- Kombination der Verfahren möglich
- Echtzeit durch Hardwareunterstützung kein Problem

QUELLEN

- Computergraphik, Universität Leipzig
  (Prof. D. Bartz)
- Graphische Datenverarbeitung I, Universität Tübingen
  (Prof. W. Straßer)
- Graphische Datenverarbeitung I, TU Darmstadt
  (Prof. M. Alexa)
- Computergraphik, Uni Siegen (Prof. Klomfaß)