Achtung: Dieses Dokument ist *nur* eine Mitschrift der Vorlesung "Computergrafik" SoSe2010. Sie wurde während der Vorlesung angefertigt. Es wird aber seitens des Autors keine Garantie auf Vollständigkeit und Richtigkeit des Inhalts gegeben.

Mitschrift
Computergrafik

gehalten von Prof. Dr. Günther Rote

9. Juni 2010

Inhaltsverzeichnis

1 Einführung

1.1 Organisatorisches

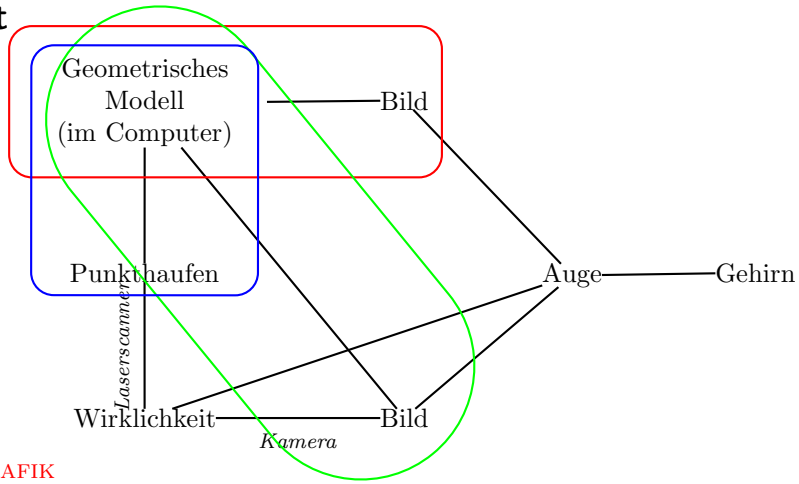
1.1.1 Übungsblätter:

- Ausgabe: Mittwoch, Abgabe: Freitag
- Abgabe in Zweiergruppen
- 60% der Punkte müssen erreicht werden
- min. einmal Vorrechnen

1.1.2 Programmierung

- Aufgaben in Java gestaltet
- mit OpenGL-Interface
- auf Nachfrage kann auch C/C++ verwendet werden

1.2 Übersicht



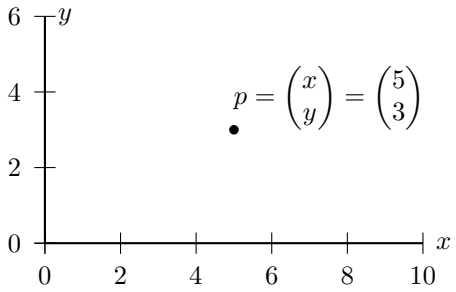
- Computergrafik
- Bildbearbeitung / Bilderkennung
- Geometrisches Rechnen / Geometrische Modellierung

1.2.1 Fahrplan

- Koordinatiensysteme, geometrische Transformationen
- Licht und Farben
- Rasterung
- Beleuchtung und Schattierung
- rendering-pipeline: vom Modell bis zum gerasterten Bildbearbeitung
- geometrische Modellierung: Kurven, Flächen und Splines
- Kein Anwendungskurs für OpenGL, JOGL, Javaview etc.!

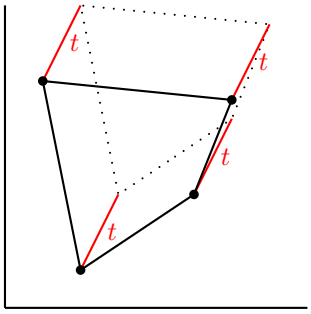
2 Koordinatensysteme, geometrische Transformationen

2.1 kartesische Koordinaten



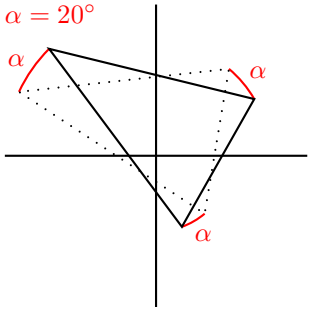
2.2 Geometrische Transformationen

- *Tranlation*: $p \mapsto p + t \qquad t \in \mathbb{R}^2$, Translationsvektor



- *Rotation* (um den Ursprung $\begin{pmatrix} 0 \\ 0 \end{pmatrix}$):

$$p \mapsto M \cdot p \qquad M = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix}, \text{ Rotationsmatrix}$$



- *Rotation* um den Punkt c: $p \mapsto M(p - c) + c = Mp + (c - Mc), \qquad c \mapsto c$
- *gleichförmige Skalierung*:

$$p \mapsto \lambda \cdot p = \begin{pmatrix} \lambda & 0 \\ 0 & \lambda \end{pmatrix} \cdot p, \qquad \lambda \neq 0$$

$$\lambda = 1 \qquad p \mapsto -p = \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix} \cdot p = \text{Spiegelung am Ursprung} = \text{Rotation um } 180°$$

- *Ungleichförmige Skalierung*:

$$M = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix} \qquad p \mapsto M \cdot p$$

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \lambda_1 x \\ \lambda_2 y \end{pmatrix}$$

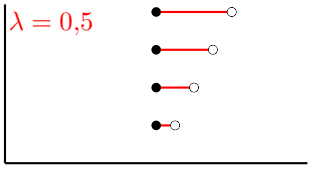
$M = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}$ resultiert in der Spiegelung an der x-Achse

$M = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$ resultiert in der Spiegelung an der y-Achse

- *Scherung*

$$M = \begin{pmatrix} \overset{\text{Scherung auf der } x\text{-Achse}}{\begin{pmatrix} 1 & \lambda \\ 0 & 1 \end{pmatrix}} & \left(\text{oder} & \overset{\text{Scherung auf der } y\text{-Achse}}{\begin{pmatrix} 1 & 0 \\ \lambda & 1 \end{pmatrix}} \right) \end{pmatrix}$$

$$\begin{pmatrix} x \\ y \end{pmatrix} \mapsto \begin{pmatrix} 1 & \lambda \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x + \lambda y \\ y \end{pmatrix}$$



Flächeninhalt:

- Translationen, Rotationen, Scherungen und Spiegelungen ändern den Flächeninhalt nicht.
- Skalierung ändert den Flächeninhalt um den Faktor $\lambda_1 \cdot \lambda_2$

Definition Eine Verknüpfung mehrerer dieser Transformationen bildet eine **affine Transformation**. Allgemein ist diese:

$$p \mapsto M \cdot p = b, \qquad M \in \mathbb{R}^{2\times 2}, b \in \mathbb{R}^2, \det M \neq 0$$

Der Flächeninhalt ändert sich um den Faktor $\det M$

Definition Die Verknüpfung von Translation, Rotation und Spiegelung heißt **starre Bewegung** oder **Isometrie**. Allgemein ist diese:

$$p \mapsto Mp + t \text{ mit } \textbf{orthogonaler Matrix } M \text{ (d. h. } \det M = \pm 1)$$

die Isometrien zerfallen:

- **orientierungserhaltende** ($\det M = 1$) und
- **orientierungsumkehrende** ($\det M = -1$) Isometrien

2.3 Homogene Koordinaten

Definition **Homogene Koordinaten**: Statt $p = \begin{pmatrix} x \\ y \end{pmatrix}$ verwendet man eine dritte Koordinate $p = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$

Konvention Die Koordinaten $\begin{pmatrix} x \\ y \\ z \end{pmatrix}$ und $\begin{pmatrix} \lambda x \\ \lambda y \\ \lambda z \end{pmatrix}$ stellen denselben Punkt dar ($\lambda \neq 0$)

Der Punkt $\begin{pmatrix} x \\ y \\ z \end{pmatrix}$ mit $z \neq 0$ hat die kartesischen Koordinaten $\begin{pmatrix} \frac{x}{z} \\ \frac{y}{z} \\ z \end{pmatrix}$

2.3.1 Allgemeine affine Transformation in homogenen Koordinaten

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} \mapsto \underbrace{\begin{pmatrix} m_{11} & m_{12} & b_1 \\ m_{21} & m_{22} & b_2 \\ 0 & 0 & 1 \end{pmatrix}}_{M'} \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} m_{11}x + m_{12}y + b_1 \\ m_{21}x + m_{22}y + b_2 \\ 1 \end{pmatrix}$$

Die Matrizen M' und $\lambda M'$ beschreiben dieselbe Transformation ($\lambda \neq 0$)

$$p \mapsto M'p \text{ mit } M' = \begin{pmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ 0 & 0 & m_{33} \end{pmatrix} \text{ und } \det M' \neq 0$$

$$\det M' \neq 0 \Leftrightarrow m_{33} \neq 0 \wedge \begin{vmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{vmatrix} \neq 0$$

$\Rightarrow$ o. B. d. A. kann man auch $m_{33} = 1$ annehmen (Dann kann man die dritte Zeile auch weglassen).

2.4 Die projektive Ebene

Definition Die (reelle) **projektive Ebene** P^2 besteht aus den Äquivalenzklassen von Punkten $\begin{pmatrix} x \\ y \\ z \end{pmatrix} \neq \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$, wobei $\begin{pmatrix} x \\ y \\ z \end{pmatrix}$ und $\begin{pmatrix} \lambda x \\ \lambda y \\ \lambda z \end{pmatrix}$ denselben Punkt darstellen ($\lambda \neq 0$)

2.4.1 Geraden in der projektiven Ebene

Gerade in $\mathbb{R}^2$ (karthesische Koordinaten):

$$y = ax + b \text{(Gerade darf nicht senkrecht sein)}$$

$$ax + by = -c$$

$$\Updownarrow$$

Gerade in Homogenen Koordinaten

$$\begin{pmatrix} x \\ y \end{pmatrix} \longrightarrow \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

$$ax + by + c = 0 \Leftrightarrow \begin{pmatrix} a \\ b \\ c \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = 0$$

Allgemeine Gleichung einer Geraden in P^2

$$\begin{pmatrix} a \\ b \\ c \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix} = 0 \Leftrightarrow ax + by + cz = 0 \qquad \begin{pmatrix} a \\ b \\ c \end{pmatrix} \neq \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

Wenn $\begin{pmatrix} x \\ y \\ z \end{pmatrix}$ die Gleichung erfüllt, dann erfüllt auch $\begin{pmatrix} \lambda x \\ \lambda y \\ \lambda z \end{pmatrix}$ die Gleichung. $\begin{pmatrix} a \\ b \\ c \end{pmatrix}$ und $\begin{pmatrix} \lambda a \\ \lambda b \\ \lambda c \end{pmatrix}$ stellen dieselbe Gerade dar.

projektive Punkte $\begin{pmatrix} x \\ y \\ z \end{pmatrix} \neq \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$ Skalierung egal.

projektive Gerade $\begin{pmatrix} a \\ b \\ c \end{pmatrix} \neq \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$ Skalierung egal

Satz Punkt $\begin{pmatrix} x \\ y \\ z \end{pmatrix}$ liegt auf der Geraden $\begin{pmatrix} a \\ b \\ c \end{pmatrix}$:

$$\begin{pmatrix} a \\ b \\ c \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix} = 0$$

Satz Zwei verschiedene Geraden schneiden sich in genau einem Punkt.

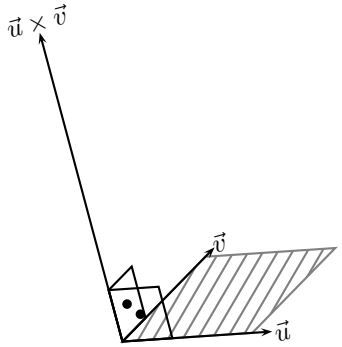
Beweis Gerade $\forall\lambda : \begin{pmatrix} a_1 \\ b_1 \\ c_1 \end{pmatrix}, \begin{pmatrix} a_2 \\ b_2 \\ c_2 \end{pmatrix} \neq \lambda \begin{pmatrix} a_1 \\ b_1 \\ c_1 \end{pmatrix}$. Schnittpunkt:

$$\begin{aligned} a_1x + b_1y + c_1z &= 0 \\ a_2x + b_2y + c_2z &= 0 \end{aligned}$$

Koeffizientenmatrix $A = \begin{pmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \end{pmatrix}$, $\operatorname{rg} A = 2$
$\Rightarrow$ Lösungsmenge ist eindimensional

$$L = \left\{ \lambda \begin{pmatrix} x_0 \\ y_0 \\ z_0 \end{pmatrix} \middle| \lambda \in \mathbb{R} \right\} \text{ ist ein projektiver Punkt}$$

$\begin{pmatrix} x_0 \\ y_0 \\ z_0 \end{pmatrix}$ kann als $\begin{pmatrix} a_1 \\ b_1 \\ c_1 \end{pmatrix} \times \begin{pmatrix} a_2 \\ b_2 \\ c_2 \end{pmatrix} = \begin{pmatrix} \begin{vmatrix} b_1 & b_2 \\ c_1 & c_2 \end{vmatrix} \\ \begin{vmatrix} c_1 & c_2 \\ a_1 & a_2 \end{vmatrix} \\ \begin{vmatrix} a_1 & a_2 \\ b_1 & b_2 \end{vmatrix} \end{pmatrix}$ berechnet werden (Kreuzprodukt)



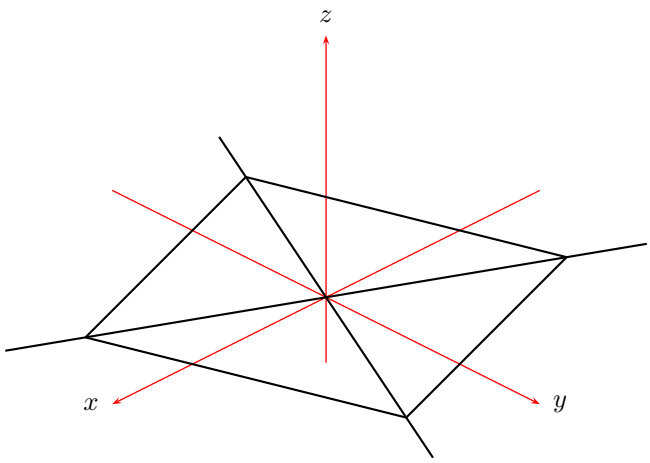
Satz Durch zwei verschiedene Punkte gibt es genau eine Geraden

Beweis gleich wie oben: $\begin{pmatrix} a \\ b \\ c \end{pmatrix}$ mit $\begin{pmatrix} x \\ y \\ z \end{pmatrix}$ vertauschen.

Dualitätsprinzip Man kann in einem Satz der projektiven Geometrie der Ebene „Punkte“ und „Geraden“ vertauschen und es bleibt ein gültiger Satz.

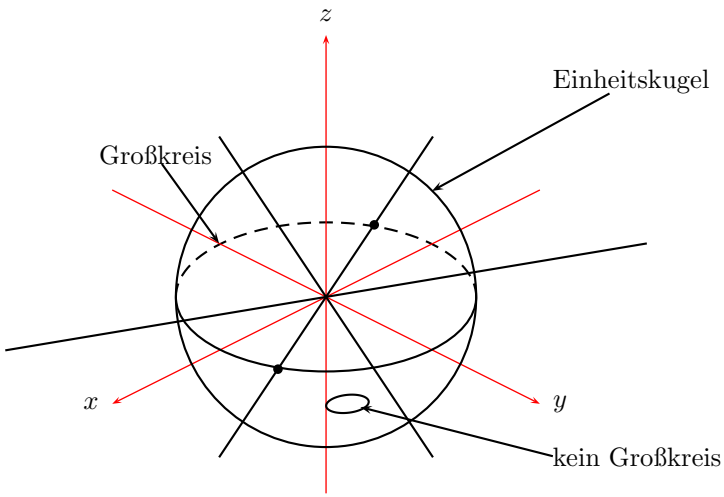
2.4.2 Modelle der projektiven Ebene

1. *Räumliches Modell der projektiven Ebene* $\left\{ \lambda \begin{pmatrix} x_0 \\ y_0 \\ z_0 \end{pmatrix} \middle| \lambda \in \mathbb{R} \right\}$... Geraden durch den Ursprung im $\mathbb{R}^3$ entsprechen den projektiven Punkten.



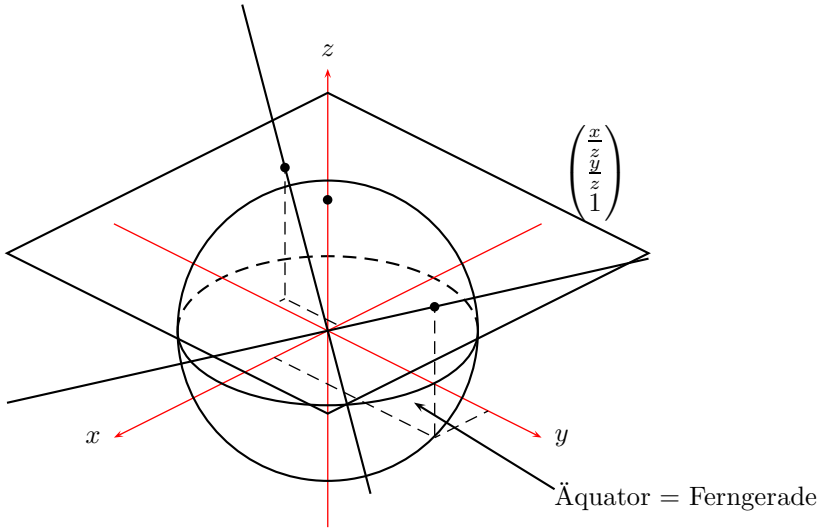
projektive Gerade ≡ Ebene durch den Ursprung

2. *Kugelmodell der projektiven Ebene* entsteht durch Schnitt des räumlichen Modells mit der Einheitskugel $S^2 = \left\{ \begin{pmatrix} x \\ y \\ z \end{pmatrix} \middle| x^2 + y^2 + z^2 = 1 \right\}$



projektiver Punkt ≡ Paar gegenüberliegender Punkte auf der Einheitskugel
projektive Gerade ≡ Großkreise

2.4.3 Projektive Punkte zu karthesische Koordinaten



Schnitt der Geraden $\begin{pmatrix} x \\ y \\ z \end{pmatrix} \cdot \lambda$ im $\mathbb{R}^3$ mit Ebene $z = 1$: $z \cdot \lambda = 1 \Rightarrow \lambda = \frac{1}{z}$

$\rightarrow \begin{pmatrix} x \cdot \frac{1}{z} \\ y \cdot \frac{1}{z} \\ 1 \end{pmatrix}$

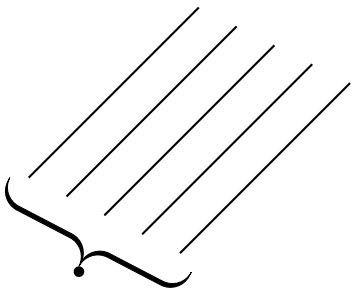
Satz Die Punkte $\begin{pmatrix} x \\ y \\ z \end{pmatrix}$ mit $z = 0$ haben *keine* Entsprechung in der euklidischen Ebene: Jede projektive Gerade hat als Bild in der euklidischen Ebene eine Gerade, mit einer Ausnahme: die Gerade $\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$

Definition Die Punkte des projektiven Raumes, die keine euklidische Entsprechung haben, heißen **Fernpunkte**. Die Gerade $\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$ **Ferngerade**.

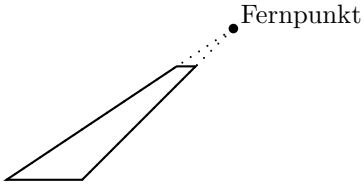
Satz Zwei Geraden der euklidischen Ebene sind genau dann *parallel*, wenn ihr Schnittpunkt ein Fernpunkt ist.

Satz Die Punkte, die auf der Ferngeraden liegen, sind genau die Fernpunkte

Satz Es gibt zu jeder Schaar paralleler Geraden genau einen Fernpunkt.



Anschaulich ist ein Fernpunkt äquivalent zu perspektivischen Sammelpunkten:



2.5 Allgemeine projektive Transformationen

$\begin{pmatrix} x \\ y \\ z \end{pmatrix} \mapsto M \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix}$ mit $M = \mathbb{R}^{3\times 3}, \det M \neq 0$
(Punkte bleiben Punkte, Geraden bleiben Geraden, Inzidenz bleibt erhalten)

Definition Affine Transformationen sind jene Transformationen, bei denen die Fernpunkte Fernpunkte bleiben.

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} \mapsto M \begin{pmatrix} x \\ y \\ 0 \end{pmatrix} = \begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} \stackrel{!}{=} \begin{pmatrix} x' \\ y' \\ 0 \end{pmatrix}$$

$$M = \begin{pmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ \cancel{m_{31}}0 & \cancel{m_{32}}0 & m_{33} \end{pmatrix}$$

$$\forall x, y : m_{31}x + m_{32}y + m_{33} \cdot 0 = 0 \Rightarrow m_{31} = m_{32} = 0$$

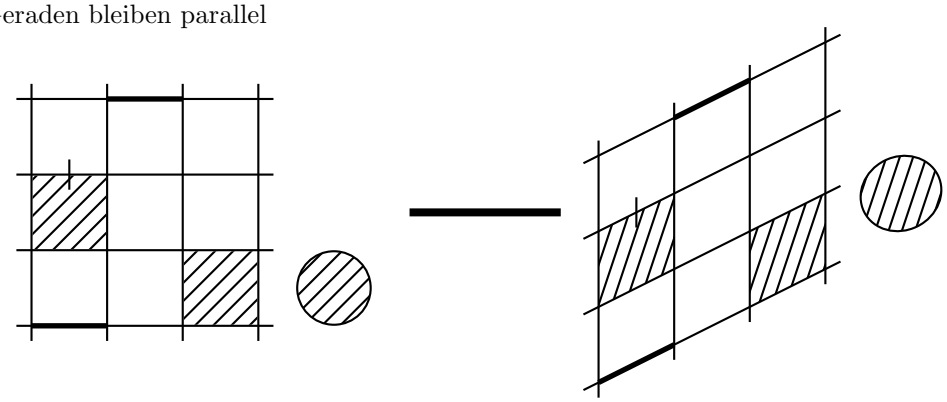
$\det M \neq 0$

$$\det M = \underbrace{m_{33}}_{\neq 0} \cdot \underbrace{\begin{vmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{vmatrix}}_{0} \qquad \Rightarrow \text{o. B. d. A. } m_33 = 1$$

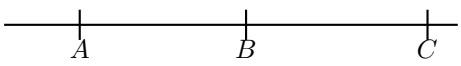
$$\begin{pmatrix} x \\ y \end{pmatrix} \mapsto \overbrace{\begin{pmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}}^{\text{lineare Transformation}} + \overbrace{\begin{pmatrix} m_{13} \\ m_{23} \end{pmatrix}}^{+\text{ Translation}}$$

Affine Transformation:

- parallele Geraden bleiben parallel



- erhalten das Teilverhältnis auf parallelen Geraden



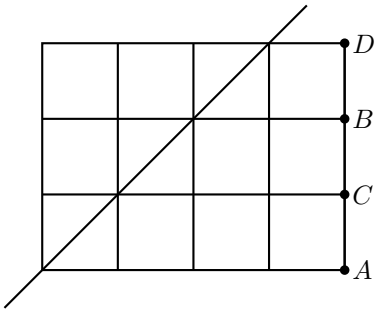
$$\frac{\overrightarrow{AB}}{\overrightarrow{AC}}$$

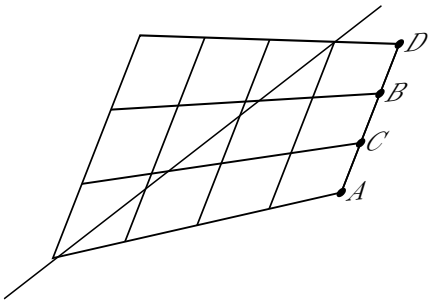
Starre Bewegungen (Isometrien, euklidische Transformationen):
$M = \begin{pmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{pmatrix}$ ist orthogonal $M^T = M^{-1}$ erhalten Längen, Winkel und Flächen

Doppelverhältnis

$$\boxed{\frac{\frac{\overrightarrow{AC}}{\overrightarrow{BC}}}{\frac{\overrightarrow{AD}}{\overrightarrow{BD}}}} = \text{Doppelverhältnis}$$

Bemerkung projektive Transformationen erhalten das sogenannte Doppelverhältnis



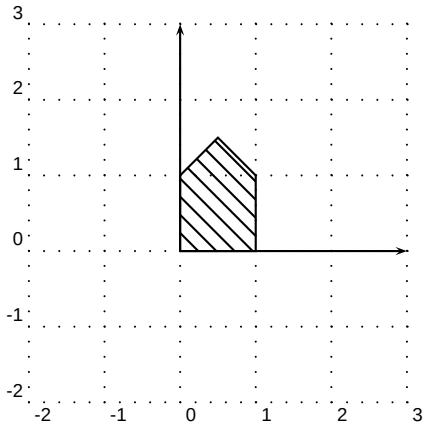


Ausblick projektiver Raum; wird beschrieben durch homogene Koordinaten $\begin{pmatrix} x \\ y \\ z \\ u \end{pmatrix} \neq \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$. Karthesische Koordinaten $\begin{pmatrix} x \\ y \\ z \end{pmatrix}$ entsprechen homogenen Koordinaten $\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$ oder $\begin{pmatrix} \lambda x \\ \lambda y \\ \lambda z \\ \lambda \end{pmatrix}$ $(\lambda \neq 0$, bel.).

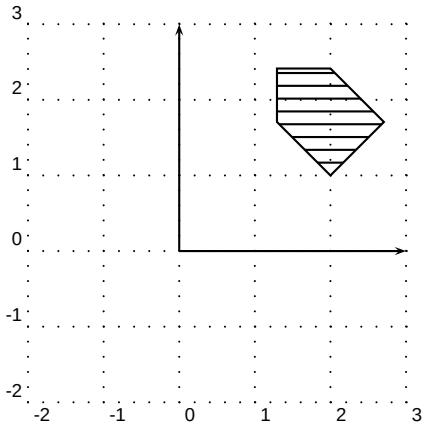
Bemerkung 1 Transformation $x \mapsto Mx$ kann man auf zwei Arten interpretieren:

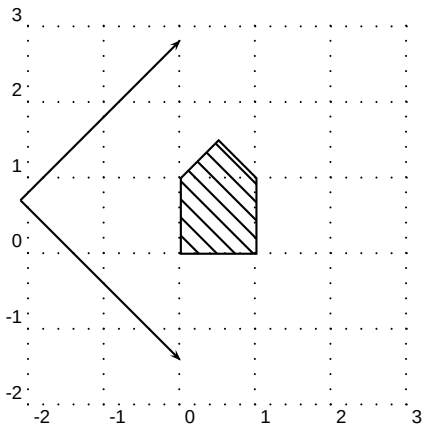
a) Wende die transformation M auf Objekte an. Objekte werden bewegt, Standpunkt/Koordinatensystem bleibt fest.

b) Drücke die unveränderte Lage eine Objektes in einem neuen Koordinatensystem aus.



$x \mapsto Mx$





$x \mapsto Mx$

Rechnerisch macht dies keinen Unterschied.

Bemerkung 2 geometrische Transformationen können verknüpft; Reihenfolge ist wichtig!

$$y = M_1 x \qquad z = M_2 y \qquad z = \underbrace{M_2 M_1}_{\text{Matrizenmultiplikation}} x$$

Inverse Transformation wird dur die inverse Matrix ausgedrückt:

$$x = M_1^{-1} y$$

Bemerkung 3 *Bei uns* stehen Koordinaten in *Spaltenvektoren* $\begin{pmatrix} x \\ y \\ z \end{pmatrix}$ $\begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix}$

$\Rightarrow$ Transformation $\equiv$ Multiplikation mit einer Matrix *von links*

$$Mx = y$$

Alternative: Zeilenvektoren
$\Rightarrow$ Transformation $\equiv$ Multiplikation mit einer *von rechts* mit der *transponierten Matriz*

$$y^t = x^t M^t = (Mx)^t$$

Diese Schreibweise ist an sich intuitiver (da die Rechnung in der Reihenfolge der Anwendung aufgeschrieben wird), aber mathematisch unüblich:

$$M_2 M_1 x = z \Longleftrightarrow x^t M_1^t M_2^t = z^t$$

2.6 Transformation im dreidimensionalen Raum

2.6.1 Affine Transformation im dreidimensionalen Raum

- allgemeine affine Transformationen:

$$\begin{pmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ 0 & 0 & 0 & 1 \end{pmatrix} \text{ mit } \begin{vmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{vmatrix} \neq 0$$

- Isometrien (starre Bewegungen):

$$\begin{pmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{pmatrix} \text{ ist eine orthogonale Matrix}$$

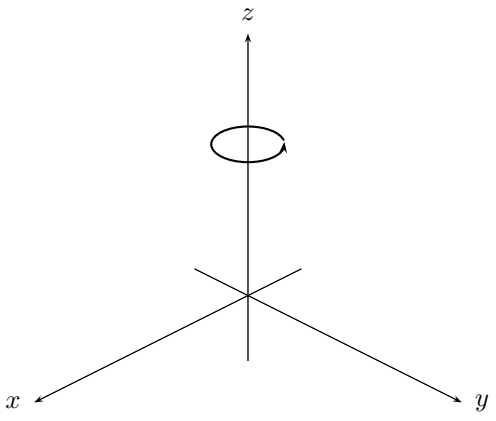
  a) orientierungserhaltende $\det M = +1$ [Rotation um eine Achse (+ Translation)]

  b) orientierungsumkehrende $\det M = -1$ [Spiegelung an einer Ebene, Spiegelung an einem Punkt, Drehspiegelung ...]

Beispiele

- Drehung um die z-Achse:

$$M = \begin{pmatrix} \cos\varphi & -\sin\varphi & 0 & 0 \\ \sin\varphi & \cos\varphi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



- Spiegelung an der xy-Ebene:

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

- Spiegelung am Nullpunkt:

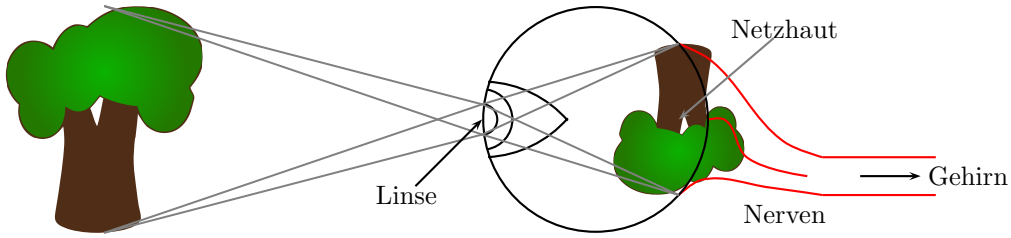
$$M = \begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

2.6.2 projektive Transformationen im dreidimensionalen Raum

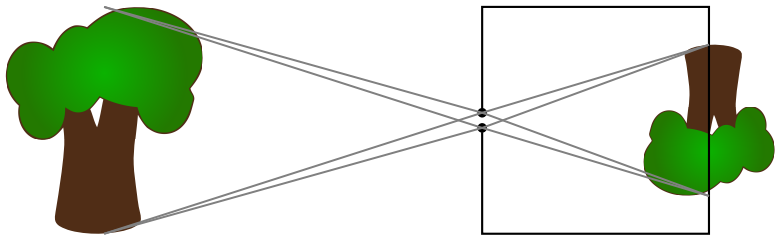
$x \mapsto Mx, \qquad M \in \mathbb{R}^{4\times 4}, \det M \neq 0$

2.7 Projektionen und Perspektive

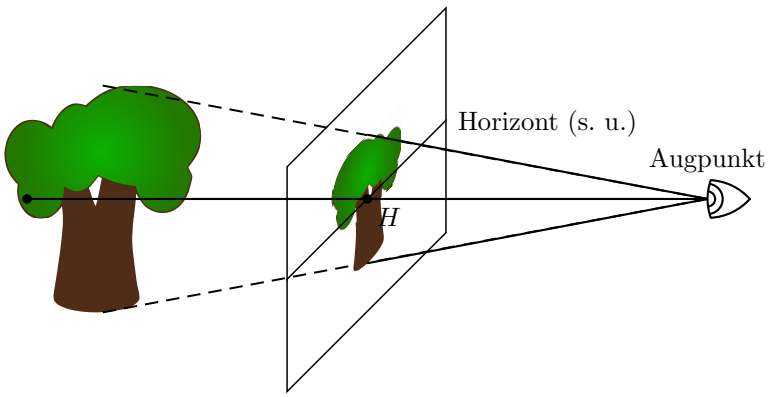
- Sehen mit dem menschlichen Auge



- Lochkamera



- Projektionen

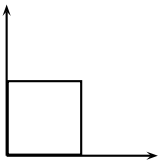


H = Hauptpunkt (s. u.)

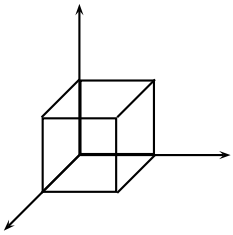
Projektionen Vebinde gegebene Punkte mit einem festen *Projektionszentrum* (kann auch ein Fernpunkt sein) und schneide die Strahlen mit einer Ebene (= *Projektionsebene*)

1. Projektionszentrum im Endlichen: Zentralprojektion
2. Projektionszentrum ein Fernpunkt: Parallelprojektion
   (Parallele Geraden bleiben parallel)

a) Wenn die Projektion senkrecht auf den Projektionsstrahlen steht, spricht man von *orthographischer Projektion*

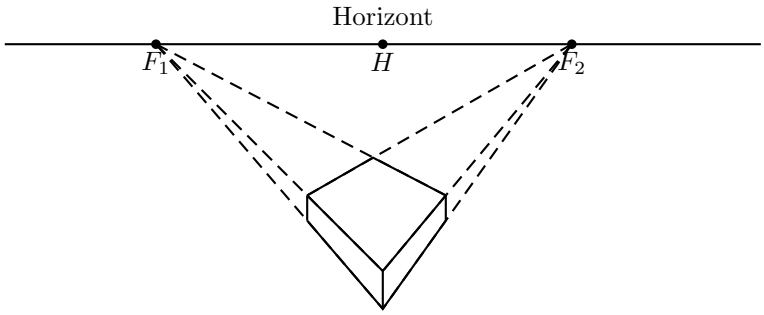


b) andernfalls von *schiefer Projektion*



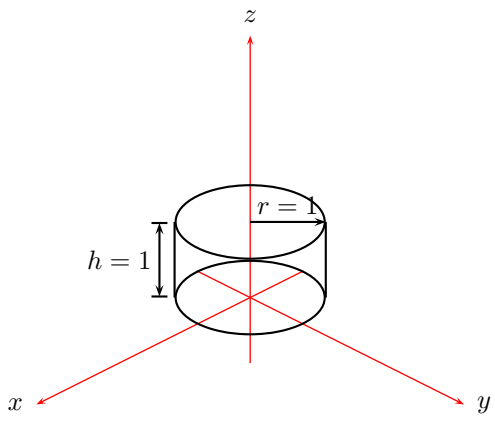
zu 1. Zentralprojektion Der *Hauptpunkt* ist der Punkt der Projektionsebene, der dem Auge am nächsten liegt. Ein projeziertes Bild vermittelt den exakten wirklichkeitsgetreuen Eindruck genau dann, wenn man sich so davor stellt, das das Auge direkt vor dem Hauptpunkt H liegt und den richtigen Abstand d und im richtigen Abstand zum Bild, mit dem das Bild berechnet wurde

- Parallele Geraden können in der Projektion zu schneidenden Geraden werden
- Das Bild des entsprechenden Fernpunktes heißt *Fluchtpunkt* (vanishing point)
- Die Fluchtpunkte der horizontalen Gerade liegen auf dem *Horizont* (die Fluchtgerade durch die alle horizontalen Ebenen gehen).
- Wenn die Projektionsgerade senkrecht ist, dann liegt der Hauptpunkt auf dem Horizont
  $\Rightarrow$ Senkrechte Geraden bleiben dann parallel (und senkrecht)



2.8 Koordinaten in der Praxis

- **Objektkoordinaten**

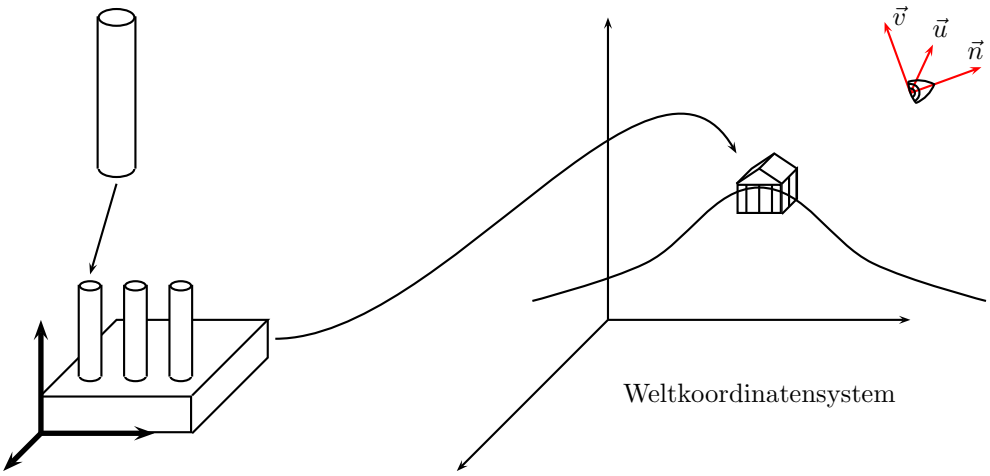


„Standardzylinder“ $x^2 + y^2 \leq 1, 0 \leq z \leq 1$

Durch die affinde Transformationen wird die Form des Zylinders angepasst und der Zylind an die passende Stelle (in einem größeren Modell / in der Umgebung) gesetzt

$$M = \begin{pmatrix} r & & & \\ & r & & \\ & & h & \\ & & & 1 \end{pmatrix} \text{...Skalierung } \rightarrow \text{ Radius } r \text{ Höhe } h$$

Translation (+Rotation) von mehreren Kopien. Zylinder wird Teil eines größeren Objektes mit einem eigenen Koordinatensysteme



- **Weltkoordinaten**
  Ein globales Koordinatensystem, das für alle Berechnungen als Referenz dient.
- **Augenkoordinaten** (Kamerakoordinaten)
  - Ursprung = Augpunkt
  - 3 orthogonale Achsen:
    $\vec{n}$ = „Blickrichtung“ vom Objekt zum Betrachter
    $\vec{u}$ = „Horizontale Richtung“ von links nach rechts
    $\vec{v}$ = „Senkrechte Richtung“ von unten nach oben

Die Projektionsebeneist orthogonal zu $\vec{n}$. Auf der Projektionsebene wir ein rechteckiges Bild erzeugt, dessen Kanten an $\vec{u}$ und $\vec{v}$ ausgerichtet sind.

$$\begin{pmatrix} x_{\text{Welt}} \\ y_{\text{Welt}} \\ z_{\text{Welt}} \\ w_{\text{Welt}} \end{pmatrix} \mapsto M_{AW} \begin{pmatrix} x_{\text{Welt}} \\ y_{\text{Welt}} \\ z_{\text{Welt}} \\ w_{\text{Welt}} \end{pmatrix} = \begin{pmatrix} x_{\text{Auge}} \\ y_{\text{Auge}} \\ z_{\text{Auge}} \\ w_{\text{Auge}} \end{pmatrix}$$

Weltkoordinaten x, y, z bilden ein Rechtssystem. Augenkoordinaten u, v, n bilden ein Rechtssystem.
$\Rightarrow M_{AW}$ ist Rotation+Translation

$$M_{AW}^{-1} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} x_{\text{Auge}} \\ y_{\text{Auge}} \\ z_{\text{Auge}} \\ 1 \end{pmatrix}$$

$$M_{AW}^{-1} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} x_u \\ y_u \\ z_u \\ 0 \end{pmatrix} = \text{Vektor } \vec{u} \text{ in Weltkoordinaten}$$

$$M_{AW}^{-1} \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} x_v \\ y_v \\ z_v \\ 0 \end{pmatrix} = \text{Vektor } \vec{v} \text{ in Weltkoordinaten}$$

$$M_{AW}^{-1} \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} x_n \\ y_n \\ z_n \\ 0 \end{pmatrix} = \text{Vektor } \vec{n} \text{ in Weltkoordinaten}$$

$$M_{AW}^{-1} = \begin{pmatrix} \boxed{\begin{matrix} x_u & x_v & x_n \\ y_u & y_v & y_n \\ z_u & z_v & z_n \end{matrix}} & \begin{matrix} x_{\text{Auge}} \\ z_{\text{Auge}} \\ y_{\text{Auge}} \end{matrix} \\ \begin{matrix} 0 & 0 & 0 \end{matrix} & 1 \end{pmatrix} \quad \text{orthogonal}$$

$$M_{AW} = \begin{pmatrix} x_u & y_u & z_u & * \\ x_v & y_v & z_v & * \\ x_n & y_n & z_n & * \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$M_{AW} \begin{pmatrix} x_{\text{Auge}} \\ y_{\text{Auge}} \\ z_{\text{Auge}} \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

Der Punkt $\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$ in Augenkoordinaten wird auf $A + \vec{n}$ in Weltkoordinaten abgebildet.

$$M_{AW}^{-1} \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} x_{\text{Auge}} + x_n \\ y_{\text{Auge}} + y_n \\ z_{\text{Auge}} + z_n \\ 1 \end{pmatrix} \qquad M_{AW}^{-1} \left[\begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \end{pmatrix} - \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \right] = \begin{pmatrix} x_n \\ y_n \\ z_n \\ 0 \end{pmatrix}$$

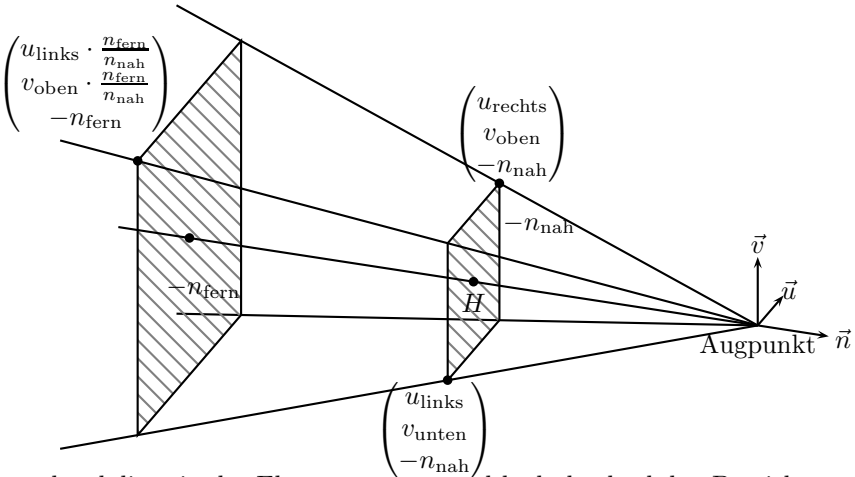
$A = \begin{pmatrix} x_u & x_v & x_n \\ y_u & y_v & y_n \\ z_u & z_v & z_n \end{pmatrix}$ Spalten sind die kartesischen Weltkoordinaten von $\vec{u}, \vec{v}, \vec{n}$

$$M_{AW}^{-1} = \left(\begin{array}{ccc|c} & A & & \begin{matrix} x_{\text{Auge}} \\ z_{\text{Auge}} \\ y_{\text{Auge}} \end{matrix} \\ \hline 0 & 0 & 0 & 1 \end{array}\right)$$

$$M_{AW} = \left(\begin{array}{ccc|c} & A^T & & -A^T \begin{pmatrix} x_{\text{Auge}} \\ z_{\text{Auge}} \\ y_{\text{Auge}} \end{pmatrix} \\ \hline 0 & 0 & 0 & 1 \end{array}\right)$$

Probe:

$$\left(\begin{array}{ccc|c} & A & & \begin{matrix} x_{\text{Auge}} \\ z_{\text{Auge}} \\ y_{\text{Auge}} \end{matrix} \\ \hline 0 & 0 & 0 & 1 \end{array}\right) \cdot \left(\begin{array}{ccc|c} & A^T & & -A^T \begin{pmatrix} x_{\text{Auge}} \\ z_{\text{Auge}} \\ y_{\text{Auge}} \end{pmatrix} \\ \hline 0 & 0 & 0 & 1 \end{array}\right) = \left(\begin{array}{c|c} A \cdot A^T = I' & 0 \\ \hline 0 & 1 \end{array}\right) = I \qquad \square$$

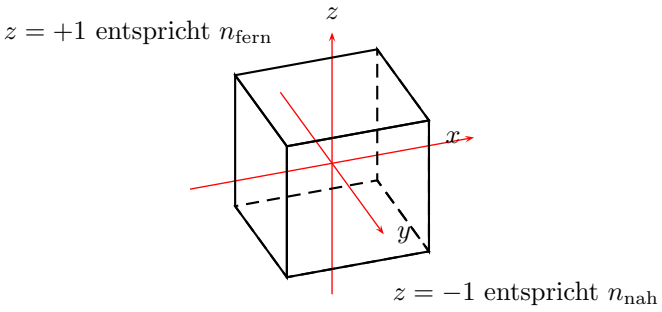


Projektionsrechteck liegt in der Ebene $n = n_{\text{nah}}$ und bedeckt dord den Bereich

$$[u_{\text{links}}, u_{\text{rechts}}] \times [v_{\text{unten}}, v_{\text{oben}}]$$

Der Sichtbare Bereich ist alles was hinter diesem Rechteck liegt. Zusätzlich wird alles abgeschnitten, was hinter der Ebene $n = n_{\text{fern}}$ liegt.
$\Rightarrow$ Pyramidenstumpf (view frustum)

- **Normalisierte Gerätekoordinaten** (normalized device coordinates, NDC) Der sichtbare Pyramidenstumpf wird durch projektive Transformation auf den Würfel $[-1,+1]^3$ transformatiert. x, y, z bilden ein Linkssystem.



$$\begin{pmatrix} * \\ * \\ -n_{\text{nah}} \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} * \\ * \\ -1 \\ +1 \end{pmatrix} \cdot \lambda \qquad \text{(Ebene } z = -1\text{)}$$

$$\begin{pmatrix} * \\ * \\ -n_{\text{fern}} \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} * \\ * \\ 1 \\ 1 \end{pmatrix} \cdot \lambda_2$$

Fernpunkt auf der z-Achse:

$$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} 0 \\ 0 \\ * \\ 0 \end{pmatrix}$$

Horizontale Linien (Richtung u) bleiben parallel und horizontal (Richtung x):

$$\begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \mapsto \begin{pmatrix} * \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Vertikale Linien

$$\begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} \mapsto \begin{pmatrix} 0 \\ * \\ 0 \\ 0 \end{pmatrix}$$

Transformation

$$\begin{pmatrix} u \\ v \\ n \\ w \end{pmatrix} \mapsto \begin{pmatrix} * & 0 & * & 0 \\ 0 & * & * & 0 \\ 0 & 0 & * & * \\ 0 & 0 & -1 & 0 \end{pmatrix} \begin{pmatrix} u \\ v \\ n \\ w \end{pmatrix} = \begin{pmatrix} * & 0 & * & 0 \\ 0 & * & * & 0 \\ 0 & 0 & a & b \\ 0 & 0 & -1 & 0 \end{pmatrix} \begin{pmatrix} u \\ v \\ n \\ w \end{pmatrix}$$

$$\begin{pmatrix} * \\ * \\ -n_{\text{nah}} \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} * \\ * \\ -1 \\ 1 \end{pmatrix} \cdot \lambda = \begin{pmatrix} * \\ * \\ -n_{\text{nah}}a + b \\ n_{\text{nah}} \end{pmatrix} \Rightarrow -n_{\text{nah}}a + b = -n_{\text{nah}}$$

$$\begin{pmatrix} * \\ * \\ -n_{\text{fern}} \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} * \\ * \\ 1 \\ 1 \end{pmatrix} \cdot \lambda = \begin{pmatrix} * \\ * \\ -n_{\text{fern}}a + b \\ n_{\text{fern}} \end{pmatrix} \Rightarrow -n_{\text{fern}}a + b = n_{\text{fern}}$$

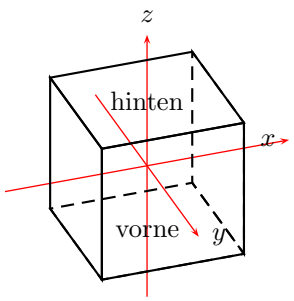
$$
\begin{aligned}
-n_{\text{nah}}a + b &= -n_{\text{nah}} \\
-n_{\text{fern}}a + b &= n_{\text{fern}} \\
\hline
a(-n_{\text{nah}} + n_{\text{fern}}) &= -n_{\text{nah}} - n_{\text{fern}} \\
a &= -\frac{n_{\text{fern}} + n_{\text{nah}}}{n_{\text{fern}} - n_{\text{nah}}} \\
\Rightarrow \qquad b &= -\frac{2 \cdot n_{\text{fern}} \cdot n_{\text{nah}}}{n_{\text{fern}} - n_{\text{nah}}}
\end{aligned}
$$

$$
\Rightarrow M = \begin{pmatrix}
\frac{2n_{\text{nah}}}{u_{\text{rechts}} - u_{\text{links}}} & 0 & \frac{u_{\text{rechts}} + u_{\text{links}}}{u_{\text{rechts}} - u_{\text{links}}} & 0 \\
0 & \frac{2n_{\text{nah}}}{v_{\text{oben}} - v_{\text{unten}}} & \frac{v_{\text{oben}} + v_{\text{unten}}}{v_{\text{oben}} - v_{\text{unten}}} & 0 \\
0 & 0 & -\frac{n_{\text{fern}} + n_{\text{nah}}}{n_{\text{fern}} - n_{\text{nah}}} & -\frac{2n_{\text{fern}} \cdot n_{\text{nah}}}{n_{\text{fern}} - n_{\text{nah}}} \\
0 & 0 & -1 & 0
\end{pmatrix}
$$

$$
M^{-1} = \begin{pmatrix}
\frac{u_{\text{rechts}} - u_{\text{links}}}{2n_{\text{nah}}} & 0 & 0 & \frac{u_{\text{rechts}} - u_{\text{links}}}{2n_{\text{nah}}} \\
0 & \frac{v_{\text{oben}} - v_{\text{unten}}}{2n_{\text{nah}}} & 0 & \frac{v_{\text{oben}} - v_{\text{unten}}}{2n_{\text{nah}}} \\
0 & 0 & 0 & -1 \\
0 & 0 & \frac{1}{2n_{\text{fern}}} - \frac{1}{2n_{\text{nah}}} & \frac{1}{2n_{\text{fern}}} + \frac{1}{2n_{\text{nah}}}
\end{pmatrix}
$$

- **Rasterkoordinaten** - Koordinaten auf dem Bildschirm
  von normalisierten Gerätekoordinaten (NDC) zu Rasterkoordinaten:

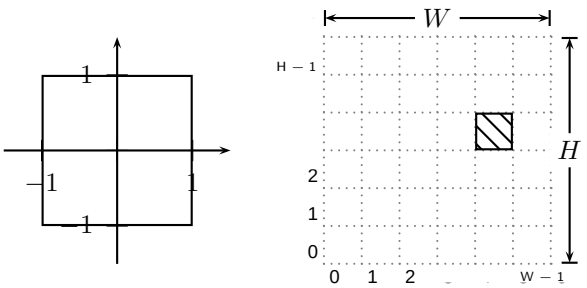
zur Erinnerung NDC Linkssystem



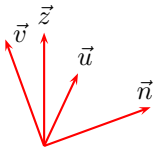
$[-1, 1] \times [-1, 1] \times [-1, 1]$

1. Projektion: z-Koordinaten weglassen. (z gibt Informationen über die Tiefe, größerer z-Wert ist weiter hinten)

2. Skalierung des x-y-Quadrates und Runden auf WxH-Gitter



Pixelkoordinaten: $\begin{pmatrix} \lfloor (x+1) \cdot \frac{W}{2} \rfloor \\ \lfloor (y+1) \cdot \frac{H}{2} \rfloor \end{pmatrix}$

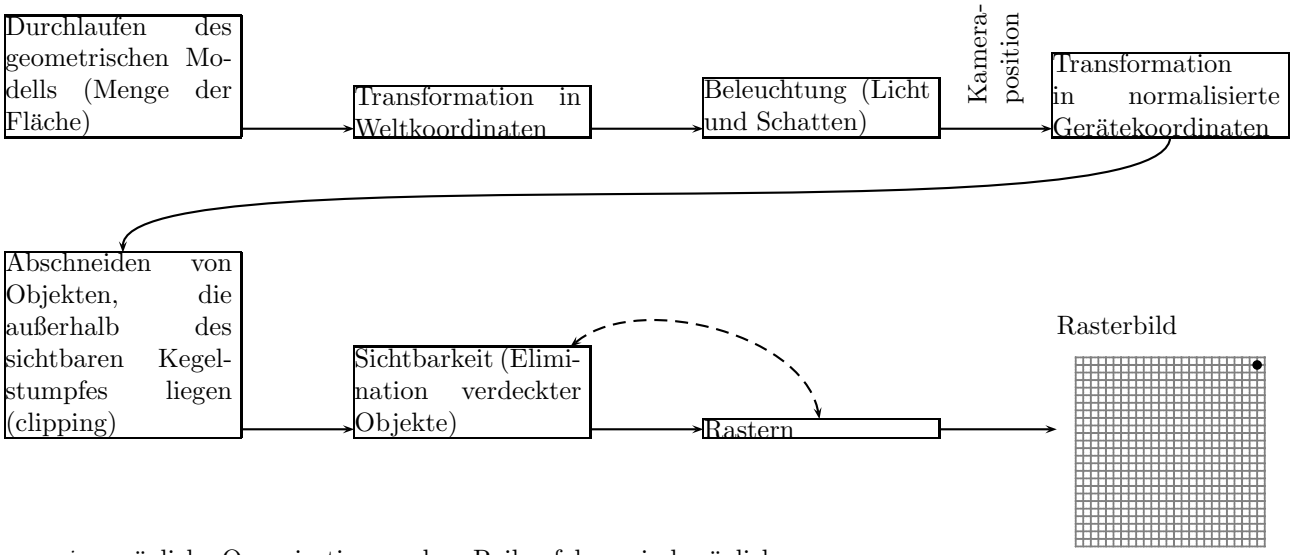


Berechnung des Augkoordinatensystems
Gegeben ist der Einheitsvektor $\vec{n}$ (und der Augpunkt)

Setze $\vec{u}_0 := \vec{z} \times \vec{n}$ $\qquad$ $\vec{z} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$ senkrecht nach oben

$$\vec{u} := \frac{\vec{u}_0}{||\vec{u}_0||}$$
$$\vec{v} := \vec{n} \times \vec{u}$$

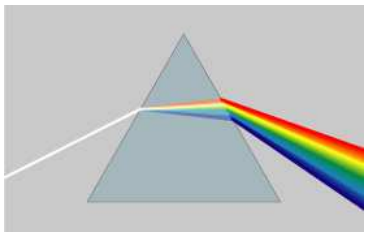
2.9 „rendering pipeline“ – vom geometrischen Modell zum Rasterbild



nur *eine* mögliche Organisation; andere Reihenfolgen sind möglich

3 Licht und Farben

Definition (sichtbares) **Licht** sind elektromagnetische Wellen verschiedener Wellenlänge (ca. zwischen 400–700 nm) Die Wellenlänge λ entscheidet über die **Farbe**. Das meiste Licht ist eine Mischung von verschiedenen Wellenlängen.

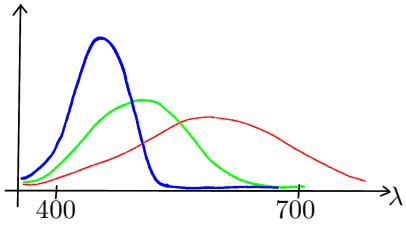


Wenn Licht auf einen Gegenstand trifft, dann wird es in unterschiedlichem Maß zurückgeworfen, je nach Wellenlänge Wenn Licht einen filter durchdringt, ist es analog (subtraktive Farbmischung).

3.1 Farbensehen im menschlichen Auge

Es gibt drei Arten von lichtempfindlichen *Zapfen* (R, G, B)

Empfindlichkeit – $e_R(\lambda), e_G(\lambda), e_B(\lambda)$



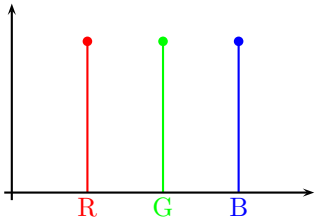
Erregung der „roten“ Zapfen bei einer Lichtquelle mit Intensitätsfunktion $f(\lambda)$

$$r = \int f(\lambda) \cdot e_R(\lambda)\ \mathrm{d}y \qquad f(\lambda)$$

analog „grün“: $g = \int f(\lambda) \cdot e_G(\lambda)\ \mathrm{d}y$
analog „blau“: $b = \int f(\lambda) \cdot e_B(\lambda)\ \mathrm{d}y$

- Verschiedene Lichtquellen mit verschiednen spektraler Zusammensetzung erzeugen den gleichen Farbeindruck, wenn sie die gleichen (r, g, b)-Werte hervorbringen.
- Dreidimensionaler Farbraum, aber nicht alle (r, g, b)-Werte erreichbar
  (Wenn $g > 0 \Rightarrow r > 0$ oder $b > 0$, $(r, g, b) = (0, 1, 0)$ gibt es nicht)
- Wenn man $f(\lambda)$ mit einem Skalar $c > 0$ multipliziert, dann ändert sich nur die Helligkeit, nicht die Farbe. Entsprechend wird (r, g, b) mit einem Skalar multipliziert.
- Normalisierung auf $r + b + g = 1$ führt auf einen zweidimensionalen Farbraum, bei dem die Helligkeit konstant ist.

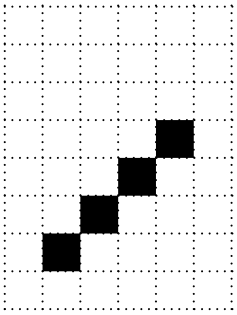
In der Computergrafik tut man so, als ob es nur *drei* verschiedene Wellenlängen (Grundfarben) gibt: Rot, Grün und Blau



4 Rasterung von Strecken und Kreisen

4.1 Strecken

- Rendering pipeline wurde durchlaufen
- Koordinaten sind ganzzahlige Pixelkoordinaten



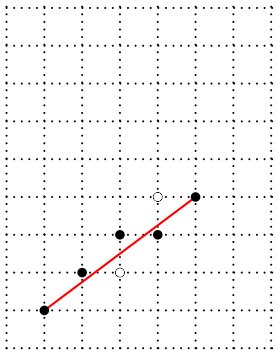
Problemstellung Gegeben sind zwei Punkte $p_1 = (x_1, y_1)$ und $p_2 = (x_2, y_2)$. Zeichne Strecke zwischen p_1 und p_2. x_1, y_1, x_2, y_2 sind ganzzahlig.

Vereinfachung Strecken die durch den Nullpunkt gehen $(0, 0), p_1 = (x_1, y_1)$

- Beschränkung auf Strecken im ersten Quadranten des Koordinatensystems. Alle anderen Strecken (im Quadranten II, III, IV) können durch Speigelung an x- oder y-Achse erzeugt werden
- Beschränkung aufersten Oktanten, alle anderen Strecken erhält man wie oben durch Vertauschen von x- und y-Koordinate

Vereinfachte Problemstellung Gegeben ist ein Punkt $p_1 = (x_1, y_1)$ mit ganzzahligen Koordinaten. $x_1 \geq y_1, x_1 \geq 0, y_1 \geq 0$. Zeichne Strecke zwischen $(0, 0)$ und (x, y).

Geradengleichung: $g(x) = \frac{y_1}{x_1} x$ Steigung: $m = \frac{y_1}{x_1}, 0 \leq m \leq 1$



Idee Werte für jeden Wert i zwischen 0 und x_1 die Funktion g aus. Runde das Ergebnis $g(i)$ und nimm diesen Wer als j-Wert

```
double m = y1/x1;
for (i = 0; i <= x1, i++) {
        j = round(i*m); // auf– bzw. abrunden
        setPixel(i,j);
}
```

- erstetze `j= round(i * m);` durch `y=y+m_j; j = round(y);`
- im i-ten Schritt $y_i = m \cdot i$

- im $(i+1)$-ten Schritt $y_{i+1} = m \cdot (i+1)$
- $y_{i+1} - y_i = m(i+1-1) = m$
- $0 \leq m \leq 1 \Rightarrow y_{i+1} \leq y_i + 1$
- Wert von j steigt pro Schleifendurchlauf um höchstens 1.

$$y_{i+1} = y_i + m \qquad m = \frac{y_1}{x_1}$$
$$j = round(y_i)$$
$$y_i = j + r_i \qquad \text{Rest: } -\frac{1}{2} \leq r_i \leq \frac{1}{2}$$
$$y_{i+1} = y_i + m = j + \underbrace{r_i + m}_{r_{i+1}}$$

```
double m = y1/x1;
double r = 0;
int j = 0;
for (i = 0; i <= x1; i+1) {
        r = r + m;      // y = y + m;
        if (r >= 1/2) { // j = round(y)
                j = j+1,
                r = r - 1;
        }
        setPixel(i,j);
}
```

z. B:

$$y = 0.6 \qquad r_{alt} = 0.6$$
$$y = j_{alt} + r_{alt} \qquad j_{neu} = j_{alt} + 1$$
$$y = j_{neu} + r_{neu} \qquad r_{neu} = r_{alt} + 1$$

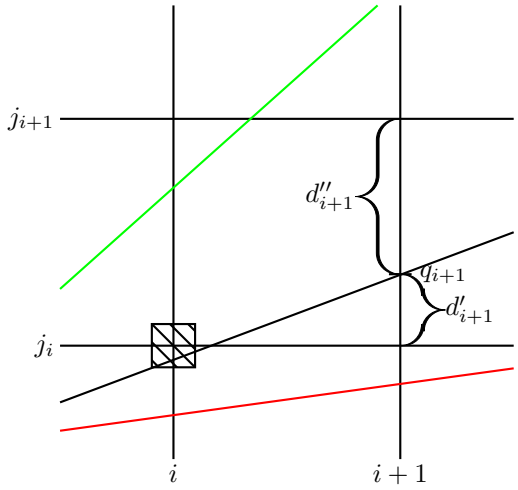
Immer noch ein Problem: **double**-Werte sind zu ungenau
Wir wissen, dass x_1 und y_1 ganzzahlig sind.
$\Rightarrow$ Multiplikation mit $2x_1$ liefert **int**-Werte

```
int m = 2*yi;
int r = 0;
int j = 0;
setPixel(0,0);
for(int i = 1; i <= x1; i++) {
        r = r + m;
        if (r >= x1) {
                j = j+1;
                r = r - 2*x1;
        }
        setPixel(i,j);
}
```

$$r_{i+1} + 2y_1 \geq x_1 \Leftrightarrow r \geq x_1 - 2y_1$$

```
int j = 0;
int m = 2*y1;
int r = 2*y1 - x1;
int c1 = m - 2*x1;
int c2 = m;
for (int i = 1; i <= x1; i++) {
        if (r >= 0) {
                j++;
                r = r + c1;
        } else {
                r = r + c2;
        }
        setPixel(i,j);
}
```

4.1.1 Bresenham-Algorithmus



Wähle eine Spalte $i+1$ das Pixel, das am nächsten an q_{i+1} liegt, also

$$\begin{aligned} d'_{i+1} \leq d''_{i+1} &\Leftrightarrow \text{ wähle } j_{i+1} = j_i \\ d''_{i+1} < d'_{i+1} &\Leftrightarrow \text{ wähle } j_{i+1} = j_i + 1 \end{aligned}$$

oder äquivalent

$$\begin{aligned} d'_{i+1} - d''_{i+1} \leq 0 &\Leftrightarrow j_{i+1} = j_i \\ d'_{i+1} - d''_{i+1} > 0 &\Leftrightarrow j_{i+1} = j_i + 1 \end{aligned}$$

y-Koordinaten von q_{i+1} $\quad \frac{y_1}{x_1}(i+1)$

$$\begin{aligned} d'_{i+1} &= \frac{y_1}{x_1}(i+1) - j_i & d''_{i+1} &= j_i + 1 - \frac{y_1}{x_1}(i+1) \\ d'_{i+1} - d''_{i+1} &= 2\frac{y_1}{x_1}(i+1) - 2j_i - 1 & &\mid \text{multipliziere mit } x_1 \end{aligned}$$

(ändert nichts an der Bedingung $\begin{matrix}\leq \\ <\end{matrix} 0$)
Wir erhalten eine Fehler-/Entscheidungsvariable für $(i+1)$-te Spalte

$$\begin{aligned} e_{i+1} &= x_i(d'_{i+1} - d''_{i+1}) = 2y_1(i+1) - 2j_i x_1 - x_1 \\ e_{i+1} &\leq 0 \Leftrightarrow j_{i+1} = j_i \\ e_{i+1} &> 0 \Leftrightarrow j_{i+1} = j_i + 1 \end{aligned}$$

Betrachte Differenz zwischen aufeinanderfolgenden Entscheidungsvariablen

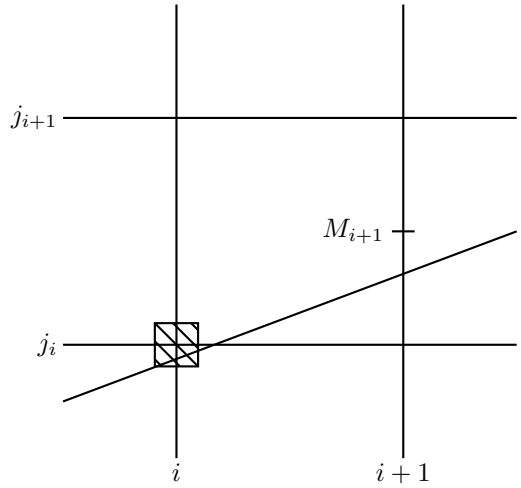
$$\begin{aligned} e_{i+1} - e_i &= \underline{2y_1(i+1)} - 2j_i x_1 - \underline{x_1} - \underline{2y_1} i + 2j_{i-1}x_1 + \underline{x_1} \\ e_{i+1} - e_i &= 2y_1 - 2x_1 \quad \underbrace{(j_i - j_{i-1})}_{= \begin{cases} 0, & \text{falls } e_i \leq 0 \\ 1, & \text{falls } e_i > 0 \end{cases}} \end{aligned}$$

Also

$$\begin{aligned} e_i > 0, j_i = j_{i-1} + 1 \text{ und } e_{i+1} = e_i + 2y_1 - 2x_1 \\ e_i \leq 0, j_i = j_{i-1} \text{ und } e_{i+1} = e_i + 2y_1 \end{aligned}$$

Anfangswerte: $i = 0, j_0 = 0, e_1 = 2y_1 - x_1$

4.1.2 Midpoint Line Algorithmus



$$M_{i+1} = \begin{pmatrix} i+1 \\ j_i + \frac{1}{2} \end{pmatrix}$$

Wenn M_{i+1} über der Strecke liegt $\Rightarrow$ wähle $j_{i+1} = j_i$ Wenn M_{i+1} unter der Strecke liegt $\Rightarrow$ wähle $j_{i+1} = j_i + 1$

$$\text{Geradengleichung: } y = \frac{y_1}{x_1}x \Leftrightarrow y_1 x - x_1 y = 0$$
$$(x,y) \text{ liegt über der Geraden } \Leftrightarrow y > \frac{y_1}{x_1}x \Leftrightarrow y_1 x - x_1 y < 0$$
$$(x,y) \text{ liegt unter der Geraden } \Leftrightarrow y < \frac{y_1}{x_1}x \Leftrightarrow y_1 x - x_1 y > 0$$

Setze $F(x,y) = xy_1 - x_1 y$, dann gilt also:

- (x,y) über Geraden $\Leftrightarrow F(x,y) < 0$
- (x,y) auf Geraden $\Leftrightarrow F(x,y) = 0$
- (x,y) unter Geraden $\Leftrightarrow F(x,y) > 0$

Entscheidungsvariable

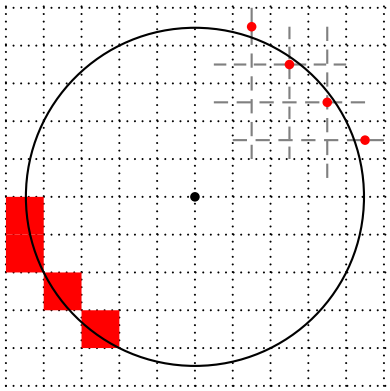
$$d_{i+1} = F(M_{i+1}) = (i+1)y_1 - x_i\left(j_i + \frac{1}{2}\right) \leq 0 \Leftrightarrow j_{i+1} = j_i$$
$$d_{i+1} = F(M_{i+1}) = (i+1)y_1 - x_i\left(j_i + \frac{1}{2}\right) = 0 \Leftrightarrow j_{i+1} = j_i + 1$$

$$\begin{aligned} d_{i+1} - d_i &= y_1(i+1) - x_1\left(j_i + \frac{1}{2}\right) - y_1 i + x_1\left(j_{i+1} + \frac{1}{2}\right) \\ &= y_1 - x_1 \quad \underbrace{(j_i - j_{i-1})} \\ &= \begin{cases} 0, & \text{falls } d_i \leq 0 \\ 1, & \text{falls } d_i > 0 \end{cases} \\ d_1 &= y_1 - \frac{x_1}{2} \end{aligned}$$

Für Ganzzahligkeit multiplitziere mit 2

$$e_i = 2d_i \Rightarrow e_{i+1} - e_i = 2y_1 - 2x_1(j_i - j_{i-1})$$
$$e_1 = 2y_1 - x_1$$

4.2 Kreise



Annahme

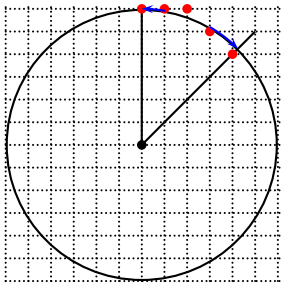
- Radius r ist ganzzahlig
- Mittelpunkt (c_x, c_y) ist ein Gitterpunkt

Kreisgleichung:

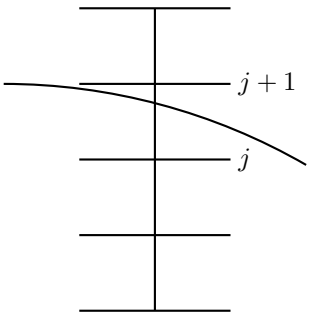
$$(x - c_x)^2 + (y - c_y)^2 = r^2$$

Betrachte den Fall, wo der Mittelpunkt (0,0) ist (anschließend alles um (c_x, c_y) verschieben). Wir zeichnen den Bereich $x \geq 0, y \geq 0, y \geq x$ auf diesem Achtelkreis zeichnen wir auf jeder senkrechten Gittergeraden *einen* Punkt.

Ausnutzung der Symmetrie:



$x = i$ ist fest, Kreis verläuft zwischen (i, j) und $(i, j + 1)$.



Welchgen dieser beiden Punkte soll man auswählen?

(1). Wähle den Punkt, der kleineren *Abstand* vom Kreis hat

(2). Berechne den Schnittpunkt mit der Geraden $x = i$ und r und runde zum nächsten Gitterpunkt.

- (Äquivalent: vergleiche den *senkrechten Abstand* zum Kreis)
- (Äquivalent: Liegt $(i, j + \frac{1}{2})$ über oder unter dem Kreisbogen?)

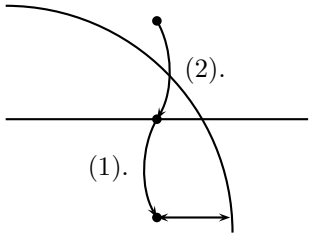
(3). Wähle den Punkt der die *Kreisgleichung* $x^2 + y^2 = r^2$ am besten erfüllt.

$$|x^2 + y^2 - r^2| \to \text{MIN}$$

Für Geraden sind alle drei Bedingungen äquivalent
Für (3). gibt es beliebig viele Varianten:

$$\sqrt{x^2+y^2} = r \Rightarrow |r - \sqrt{x^2+y^2}| \to \text{MIN führt auf (1)}$$

$$(x^2+y^2)^2 = r^4 \Rightarrow |r^4 - (x^2+y^2)^2| \to \text{MIN führt auf (1)}$$



Satz Wenn der Mittelpunkt (c_x, c_y) und der Radius r ganzzahlig sind, dann sind Bedingung (1),(2) und (3) äquivalent.
Algebraische Formulierung von (1), (2), (3):

- Punkt $(i, j+1)$ liegt außerhalb
$$i^2 + (j+1)^2 \geq r^2$$
- Punkt (i, j) liegt innerhalb
$$i^2 + (j+1)^2 < r^2$$

(1). $r - \sqrt{i^2+j^2} \gtreqless \sqrt{i^2+(j+1)^2} - r$

(2). $i^2 + \left(j+\frac{1}{2}\right)^2 \gtreqless r^2$

(3). $r^2 - (i^2+j^2) \gtreqless i^2 + (j+1)^2 - r^2$

$$\begin{aligned}
(3) \iff \quad & 2r^2 \gtreqless i^2 + j^2 + 2j + 1 + i^2 + j^2 \\
& = 2i^2 + 2j^2 + 2j + 1 \\
\iff \quad & i^2 + j^2 + j - r^2 + \frac{1}{2} \gtreqless 0 \qquad \Rightarrow \text{ „=“ kommt nicht vor für } i,j \in \mathbb{Z} \\
(2) \iff \quad & i^2 + j^2 + j + \frac{1}{4} - r^2 \gtreqless 0
\end{aligned}$$

Für $i, j, r \in \mathbb{Z}$ sind (2) und (3) äquivalent:

$\overbrace{i^2+j^2+j-r^2}^{g(i,j+1),\text{ s. Algorithmus}}$	≥ 0	$\Rightarrow$ Zeichne (i,j)
	≤ -1	$\Rightarrow$ Zeichne $(i,j+1)$

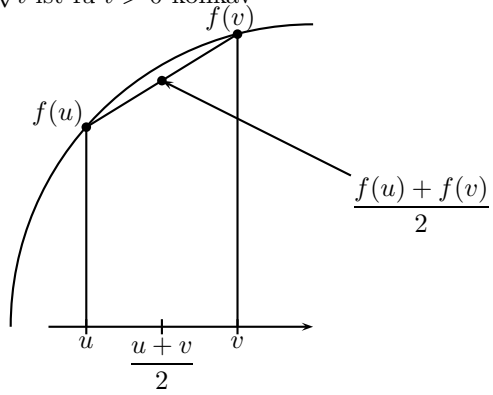
Behauptung

(a). $i^2 + \left(j+\frac{1}{2}\right)^2 \geq r^2 \Rightarrow \sqrt{i^2+(j+1)^2} - r > r - \sqrt{i^2+j^2}$

(b). $i^2 + j^2 + j + \frac{1}{2} \leq r^2 \Rightarrow \sqrt{i^2+(j+1)^2} - r < r - \sqrt{i^2+j^2}$

Daraus folgt, dass Regel (1) mit den beiden anderen Regeln (2), (3) konsistent ist.

Beweis (von (b).) Die Funktion $f(t) = \sqrt{t}$ ist fü $t > 0$ konkav



$$\text{(daraus folgt: } f\left(\frac{u+v}{2}\right) \geq \frac{f(u)+f(v)}{2}\text{)}$$

Eine differenzierbare Funktion f ist konkav $\Leftrightarrow$ f' monoton fallen $\Leftrightarrow$ $f'' \leq 0$

$$f'(t) = \frac{1}{2\sqrt{t}} \searrow \text{ konkav}$$

$$\begin{aligned} u &= i^2 + j^2 \\ v &= i^2 + (j+1)^2 = i^2 + j^2 + 2j + 1 \\ \frac{u+v}{2} &= i^2 + j^2 + j + \frac{1}{2} \\ r &\underset{\text{N.V.}}{\geq} \sqrt{i^2 + j^2 + j + \frac{1}{2}} \geq \frac{1}{2}\left[\sqrt{i^2} + \sqrt{i^2 + (j+1)^2}\right] \\ 2r &\geq \sqrt{i^2 + j^2} + \sqrt{i^2 + (j+1)^2} \end{aligned}$$

Beweis (von (a).) Funktion $h(y) = \sqrt{i^2 + y^2}$ ist *konvex*

$$h'(y) = \frac{1 \cdot 2y}{2\sqrt{i^2 + y^2}} = \frac{1}{\sqrt{\frac{i^2}{y^2} + 1}} \nearrow \text{ monoton wachsend}$$

$$g\left(\frac{u+v}{2}\right) \leq \frac{1}{2}(g(u) + g(v)) \qquad u = j, v = j+1$$

$$r^2 \underset{\text{N.V.}}{\leq} \sqrt{i^2 + \left(j + \frac{1}{2}\right)^2} \leq \frac{1}{2}\left(\sqrt{i^2 + j^2} + \sqrt{i^2 + (j+1)^2}\right)$$

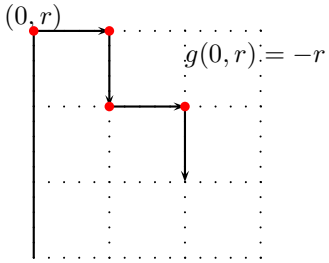
Algorithmus beginnt mit $(0, r)$ und zeichnet Punkte von links nach rechts.

- letzter gezeichneter Punkt $= (i-1, j)$
- soll nächster (i, j) oder $(i, j-1)$ gezeichnet werden?

$$g(i,j) = i^2 + (j-1)^2 + (j-1) - r^2 = i^2 + j^2 - 2j\cancel{+1} + j\cancel{-1} - r^2$$

$$g(i,j) := i^2 + j^2 - j - r^2 \begin{cases} \geq 0 & \Rightarrow \text{ zeichne } (i, j-1) \\ \leq -1 & \Rightarrow \text{ zeichne } (i,j) \end{cases}$$

$$\begin{aligned} g(i+1,j) - g(i,j) &= (i+1)^2 - i^2 = 2i + 1 \\ g(i,j-1) - g(i,j) &= (j-1)^2 - (j-1) - j^2 + j \\ &= j^2 - 2j + 1 - j + 1 - j^2 + j = -2j + 2 \end{aligned}$$



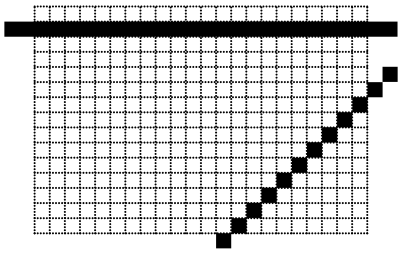
```
i := 0; j := r, g := −r; // Invariante: g = g(i,j)

loop
        SetPixel(i,j)
        g := g + 2i + 1; i := i + 1
        if g ≥ 0 then j := j − 1; g := g − 2j
until j < i

SetPixel(cx+i,cy+j)
SetPixel(cx+j,cy+i)
SetPixel(cx−i,cy+j)
SetPixel(cx−j,cy+i)
SetPixel(cx−i,cy−j)
SetPixel(cx−j,cy−i)
SetPixel(cx+i,cy−j)
SetPixel(cx+j,cy−i)
```

Die Pixel in de n 4 Himmelsrichtungen werden doppelt gezeichnet.

4.3 Schwachstellen der Rasterung (Aliasing)

- merkbare Sprünge bei fast achsenparallelen Geraden

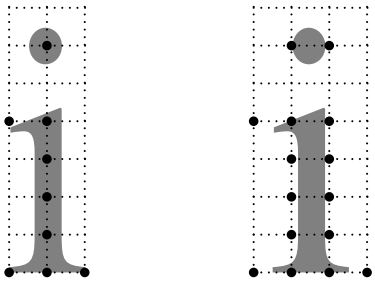
- unterschiedliche Helligkeitsverteilung in Abhängigkeit von der Steigung



  - horizontale/vertikale Linie hat 1 Pixel pro Längeneinheit
  - schräge Linie (45°) hat 1 Pixel pro $\sqrt{2}$ Längeneinheiten

  $\Rightarrow$ schräge Linien erscheinen dünner

- Buchstaben könne verschieden breit werden

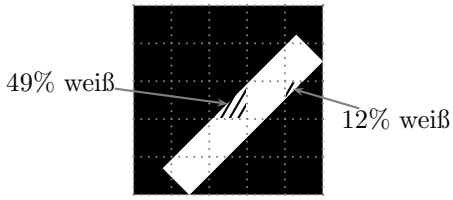


Lösung (Antialiasing) Verschiedene Graustufen für Pixel, statt nur schwarz und weiß

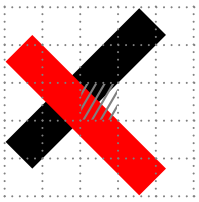
4.4 Antialiasing

verschiedene Ansätze:

1. Betrachte den Bildpunkt als quadratische Fläche und nicht als Punkt, betrachte eine Kurve (1-dimensional) als Fläche (2-dimensional). z. B. Strecke als Rechtece

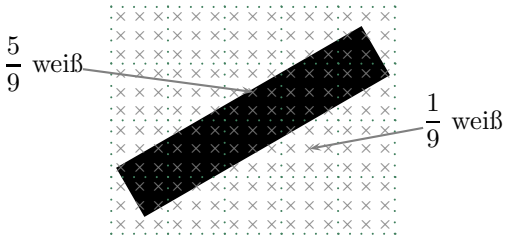


Pixel wird entsprechend hell- oder dunkelgrau gefärbt. (im Allgemeinen als proportionale Mischung aller Farben, die im Pixelquadrat vorkommen).



Aufwendig zu rechnen.

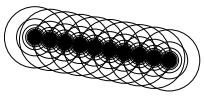
2. Supersampling: Man rechnet mit einer höheren Pixeldichte als tatsächlich vorhanden.



die verfeinerten Pixel werden „binär“ zugeordnet (in Fläche und außerhalb) und entsprechend gesetzt. Die Werte der tatsächlichen Pixel ergeben sich als Mittelwert der in ihnen enthaltenen verfeinerten Pixel.

3. Glättung:

Idee



Helligkeitswerte „strahlen“ auf die Nachbarn ab.

a) berechne die Pixelweite zunächst ohne Antialiasing.

b) Verteile die Helligkeit von jedem Pixel auf seinen Nachbarn nach einem festen Schema.

z. B.

$\frac{1}{36}$	$\frac{4}{36}$	$\frac{1}{36}$
$\frac{4}{36}$	$\frac{16}{36}$	$\frac{4}{36}$
$\frac{1}{36}$	$\frac{4}{36}$	$\frac{1}{36}$

3. lässt sich auch mit 2. kombinieren

5 Helligkeit und Farbe in der Computergrafik

5.1 Helligkeit

Definition Helligkeit in der schwarz/weiß-Grafik bezeichnet einen Grauwert auf der Skala zwischen schwarz und weiß

$$\text{schwarz} = 0, \qquad \text{weiß} = 1 \tag{5.1}$$

$$\text{oder schwarz} = 0, \qquad \text{weiß} = 25 \quad \text{(8-Bit-Integer)} \tag{5.2}$$

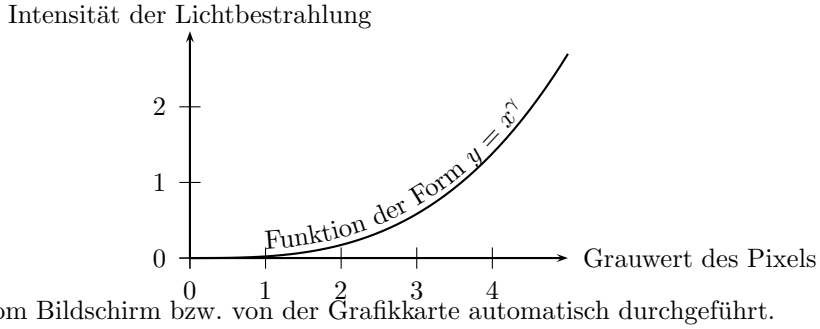
Problem tatsächliche Mischung 50% weiß und 50% schwarz → sehr hellesgrau

Weber-Fechner'sches Gesetz beschreibt die Nichtlinearität der Sinneswahrnehmungen (für Optik und Akustik gleichermaßen). Vergrößerung der Energie um einen *konstanten Faktor* wird als Vergrößerung des Reizes in *konstanten Schritten* wahrgenommen.

Beispiel Eine Vergrößerung von 10 auf 12 Energieeinheiten wird genauso groß wahrgenommen wie eine Vergrößerung von 5000 auf 6000.

Beispiel Lautstärke wir in deziBel (dB) gemessen: Logarithmus aus der Schallenergie (oder Druck?). (Logarithmus aus konstanten Faktoren konstante Differenzen).

Beispiel Eine Oktave in der Musik (z. B. Abstand zwischen tiefen C und hohen C) entspricht einer Verdoppelung der Schallfrequenz.



„γ-Korrektur", wird vom Bildschirm bzw. von der Grafikkarte automatisch durchgeführt.

5.2 Additive Farbsysteme

5.2.1 Das RGB-Farbsystem

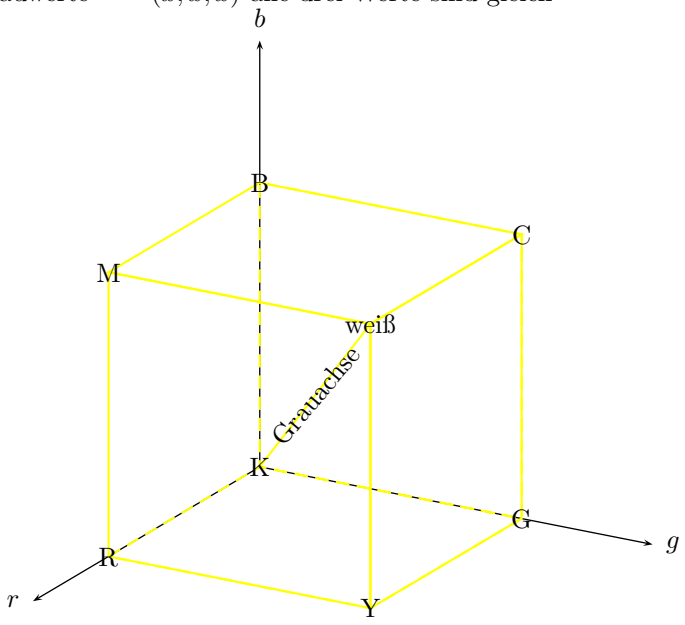
In der Computergrafik geht man von einem 3-Komponenten-Modell aus: Farbe ist aus 3 Grundfarben zusammengemischt

rot (R), grün (G), blau (B)

(In Wirklichkeit: unendlich viele Grundfarben, für jede sichtbare Wellenlänge eine)

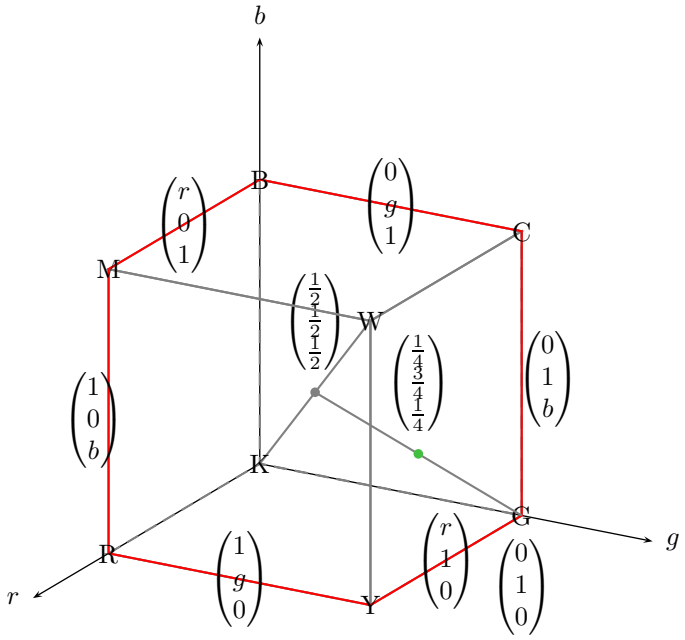
Auf einem Bildschirm sind Lichtpunkte (Phosphore) in drei Farben R, G, B nahe aneinander gitterförmig angeordnet, Die Bildpunkte werden unabhängig von einander angesteuert.
Eine Farbe in der Computergrafik ist durch 3 RGB-Werte zwischen 0 und 1 (0,1, ..., 255) charakterisiert. 24 bit pro Pixel, $2^24 \approx 16$ Millionen Farben

Farbe	(r, g, b)
rot	(1,0,0)
grün	(0,1,0)
blau	(0,0,1)
gelb (Y)	(1,1,0)
magenta (M)	(1,0,1)
cyan (C)	(0,1,1)
schwarz (K)	(0,0,0)
weiß	(1,1,1)
Grauwerte	(x, x, x) alle drei Werte sind gleich



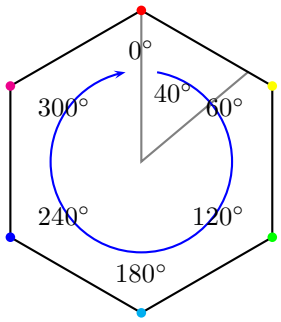
Farbwürfel $[0, 1]^3$

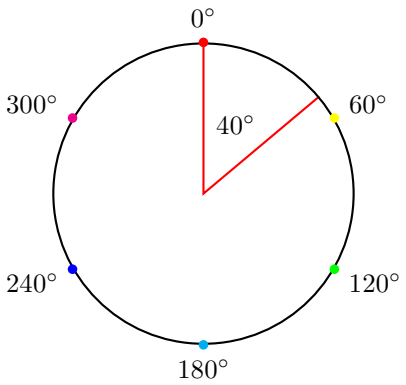
- Das RGB-System ist für die intuitive Behandlung von Farben nicht geeignet
- die Grauwerte bilden die Grauachse K-Weiß im Farbwürfel.
- die übrigen Ecken bilden das Farbsechseck RYGCBM. Auf diesem Sechseck liegen die „reinsten“/„stärksten“ Farben, alle anderen Farben kann man durch beimischen von Grau konstruieren.



Diese „stärksten Farben“ sind die Farben din mindestens eine Komponente 0 und mindestens eine Komponente 1 haben (Farben ohne Grau/Weiß/Schwarz-Anteil).

5.2.2 Farbsechseck bzw. Farbkreis





- Die Punkte auf diesem Sechseck werden häufig durch einen Winkel $(0° - 360°)$ parametrisiert.
- Startpunkt willkürlich (R = 0°, Y = 60°, ...)
- Dieser Parameter heißt „Farbton“, „Unbuntart“ (engl. *hue* (H)).

$$40° \text{ entspricht dann } \frac{1}{3} \cdot \mathrm{R} + \frac{2}{3} \cdot \mathrm{Y} \left(\frac{1}{3} \cdot 0° + \frac{2}{3} \cdot 60° \right)$$
$$= \frac{1}{3} \cdot (1,0,0) + \frac{2}{3} \cdot (1,1,0) = \left(1, \frac{2}{3}, 0\right)$$

Wir erhalte ein neues Farbsystem HSV

5.2.3 Andere Farbsysteme

HSV-System

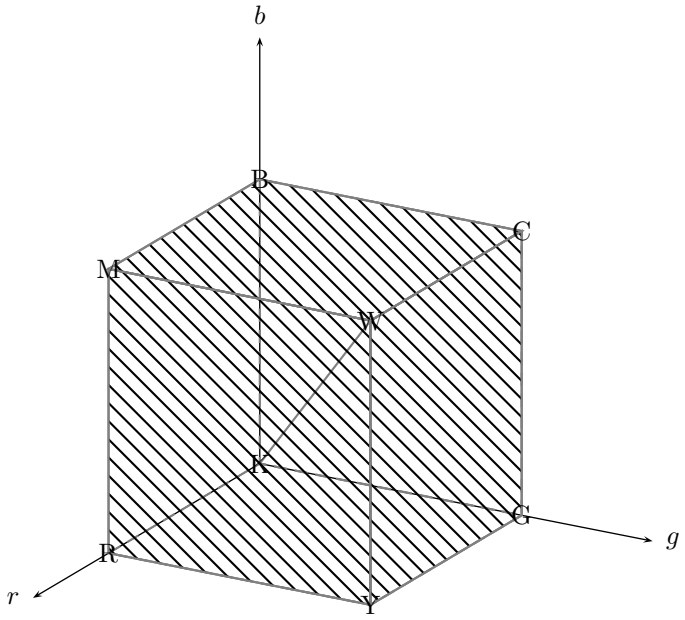
hue (Farbton) $0° \leq H \leq 360°$

saturation (Sättigung) $0 \leq S \leq 1$

value ($\approx$ Helligkeit) $0 \leq B \leq 1$

$V = 1$ sind die Farben auf den drei Deckseiten des Würfels: mindest einer der drei Werte ist 1

$$max(r, g, b) = 1$$



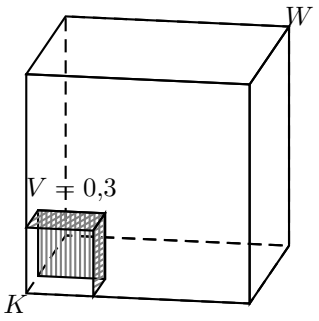
Umrechnung HSV → RGB:

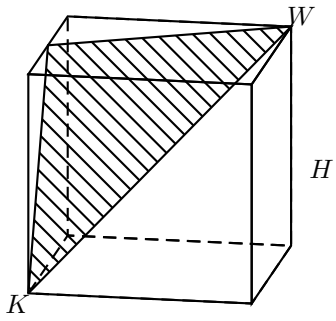
$\begin{pmatrix} r'' \\ g'' \\ b'' \end{pmatrix}$ sei die reine Farbe, die H entspricht.

$$\begin{pmatrix} r' \\ g' \\ b' \end{pmatrix} = S \begin{pmatrix} r'' \\ g'' \\ b'' \end{pmatrix} + (1 - S) \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

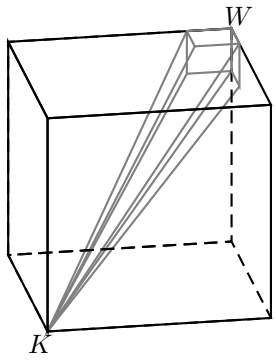
$$\text{Ergebnis } \begin{pmatrix} r \\ g \\ b \end{pmatrix} = V \begin{pmatrix} r' \\ g' \\ b' \end{pmatrix}$$

Umrechnung RGB → HSV





H konst.



$S = 0{,}2$

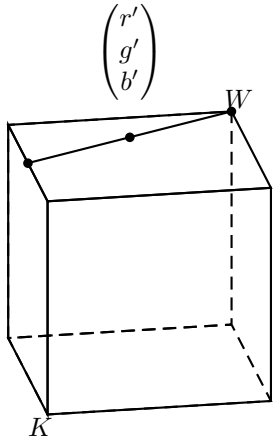
$$V = \max(r, g, b)$$

$$\begin{pmatrix} r' \\ g' \\ b' \end{pmatrix} = \begin{pmatrix} r \\ g \\ b \end{pmatrix} \cdot \frac{1}{V}$$

$$S = 1 - \min(r', g', b')$$

$$\begin{pmatrix} r'' \\ g'' \\ b'' \end{pmatrix} = \left[\begin{pmatrix} r' \\ g' \\ b' \end{pmatrix} - (1 - S) \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \right] \cdot \frac{1}{S}$$

reine Farbe → Umwandlung in H



$\begin{pmatrix} r'' \\ g'' \\ b'' \end{pmatrix}$ ist tatsächlich eine reine Farben

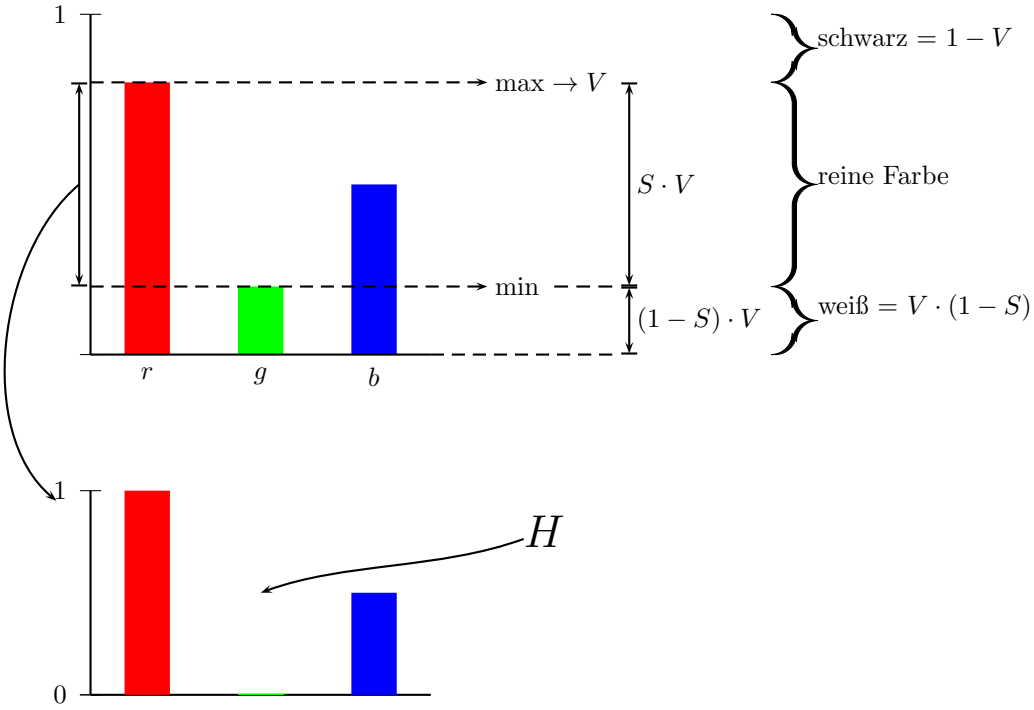
1. Wir wissen $\max(r', g', b') = 1$ z. B. $r' = 1$

$$r'' = [1 - (1 - S)1] \cdot \frac{1}{S} = S\frac{1}{S} = 1$$

2. Nehmen wir nun an $(r', g', b') = g' = 1 - S$

$$g'' = [g' - (1 - S)1] \cdot \frac{1}{S} = [-1 + S + 1 - S]\frac{1}{S} = 0$$

Für $V = 0$ setze S, H beliebig.
Für $V \neq 0$, $S = 0$, setze H beliebig.



Nachteil:

R $= (1, 0, 0)$

Y $= (1, 1, 0)$

W $= (1, 1, 1)$

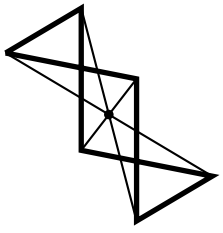
haben denselben V-Wert.

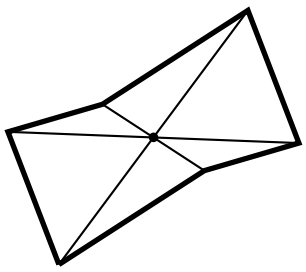
HSL-System

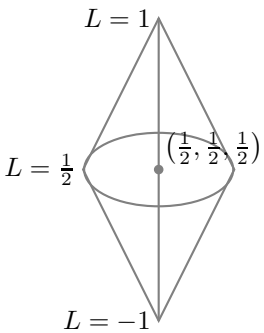
hue (Farbton) $0° \leq H \leq 360°$

saturation (Sättigung) $0 \leq S \leq 1$

lightness (oder „luminance“) $L = \frac{1}{2}$ enthält das reine Farbensechseck und den Graupunkt $\left(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}\right)$







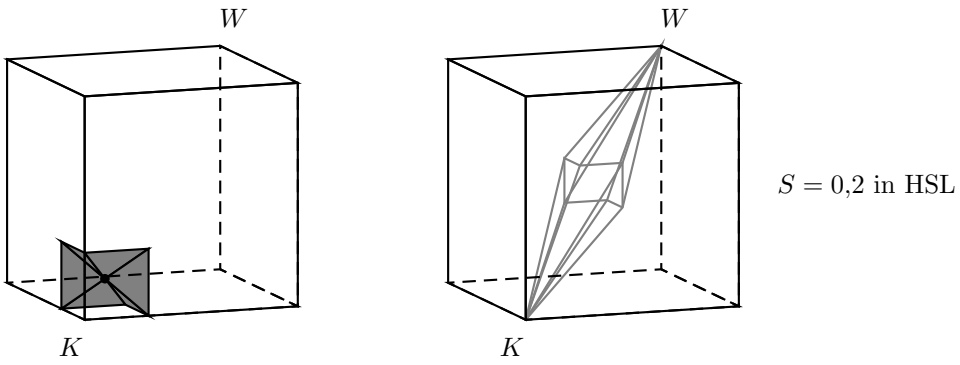
Umrechnung HSL → RGB:

$\begin{pmatrix} r'' \\ g'' \\ b'' \end{pmatrix}$ sei die reine Farbe, die H entspricht.

$$\begin{pmatrix} r' \\ g' \\ b' \end{pmatrix} = S \begin{pmatrix} r'' \\ g'' \\ b'' \end{pmatrix} + (1-S) \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

$$\text{Für } 0 \le L \le \frac{1}{2}: \begin{pmatrix} r \\ g \\ b \end{pmatrix} = \begin{pmatrix} r' \\ g' \\ b' \end{pmatrix} \cdot 2L$$

$$\text{Für } \frac{1}{2} \le L \le 1: \begin{pmatrix} r \\ g \\ b \end{pmatrix} = \begin{pmatrix} r' \\ g' \\ b' \end{pmatrix} \cdot (2-2L) + \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \cdot (2L-1)$$



$S = 0{,}2$ in HSL

$S = 1$ Farben auf allen 6 Seitenflächen des Würfels

5.2.4 YCrCb/YPrPb-Farbsystem

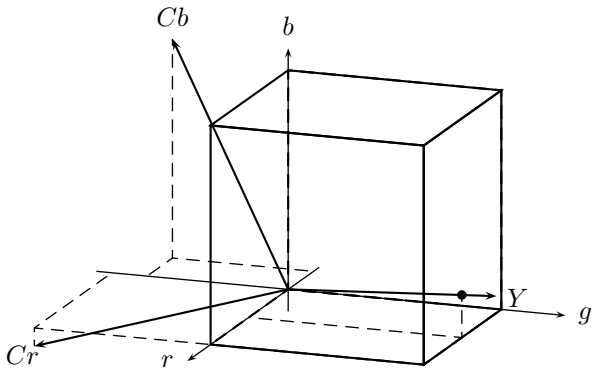
Einsatz beim Farbfernsehen

$$Y = 0{,}299 \cdot R + 0{,}587 \cdot G + 0{,}114 \cdot B \qquad \text{(Helligkeit)}$$

$$\left.\begin{aligned} Cr &= \ldots \\ Cb &= \ldots \end{aligned}\right\} \text{lineare Ausdrücke in } R,\ G,\ B \text{ (Chromanz, Farbigkeit)}$$

$$0 \le R, G, B \le 1$$

$$0 \le Y \le 1 - \frac{1}{2} \le Cr, Cb \le +\frac{1}{2}$$



5.3 Subtraktive Farbmischung (z. B. beim Drucken)

3 Grundfarben

C = cyan, B, G wird durchgelassen, R wird absorbiert

M = magenta, B, R wird durchgelassen, G wird absorbiert

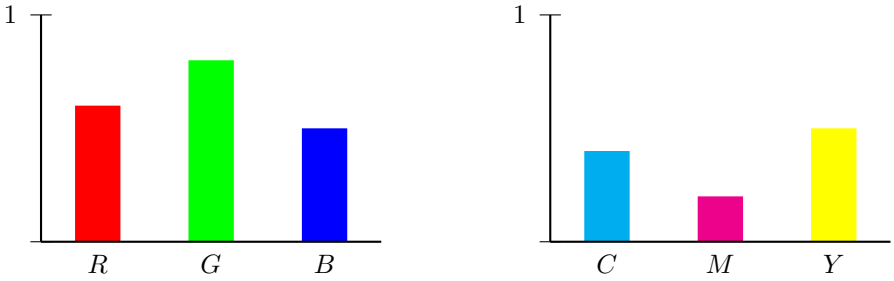
Y = gelb, R, G wird durchgelassen, B wird absorbiert

$C + M$ nur Blau bleibt übrig
$C + M + Y$ = schwarz

5.3.1 CMY-System

$0 \leq C, M, Y \leq 1$

$$C := 1 - R$$
$$M := 1 - G$$
$$Y := 1 - B$$



5.3.2 CMYK-System (Vierfarbdruck)

Zusätzlich K =schwarz. (Das Schwarz von K wird dunkler als von $C + M + Y$ oder um Druckfarbe zu sparen.)

$$K' := \min(C, M, Y)$$
$$C' := C - K'$$
$$M' := M - K'$$
$$Y' := Y - K'$$

(möglichst viel Farbe durch K ersetzen)

5.4 Digitale Farbdarstellung

heutzutage:

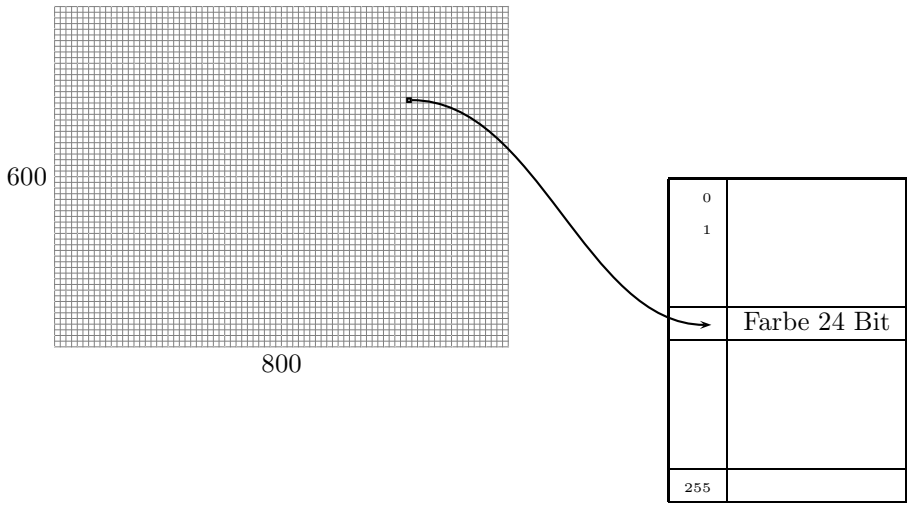
- 8 Bit pro Farbkanal (r, g, b) ... 24 Bit pro Bildpunkt
  $\Rightarrow 2^{24} = 16$ Mio. Farben (True Color)

- 4-Kanal-Darstellung (r, g, b, α) α ist für Transparenz:
  - $\alpha = 0$... durchsichtig; Farbe wird vom Hintergrund genommen,
  - $\alpha = 1$... Farbe wird von (r, g, b) bestimmt
  - $0 < \alpha < 1$... teilweise transparent

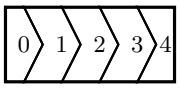
  $\rightarrow$ 32 bit pro Pixel

5.4.1 Farbpaletten

Es wird nicht für jeden Pildpunkt eine unabhängige Farbe gespeichert (24 Bit) sonder ein Index in einer Tabelle (8 Bit, = Farbpaletten).



Animation ganz einfach und schnell möglich.

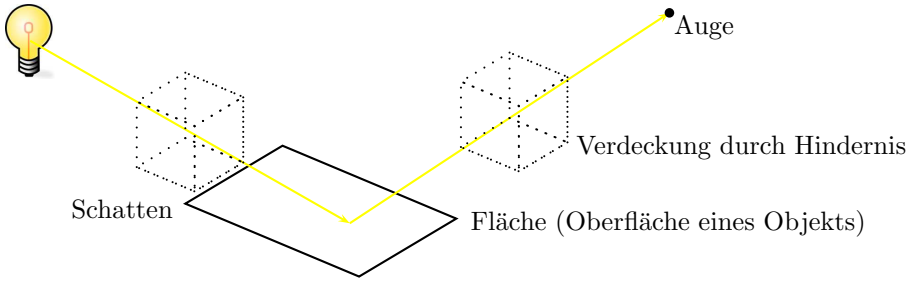


0	#FF0000	0	#FF0000
	#000000		#FF0000
	#000000		#000000
	#000000		#000000
4	#000000	4	#000000

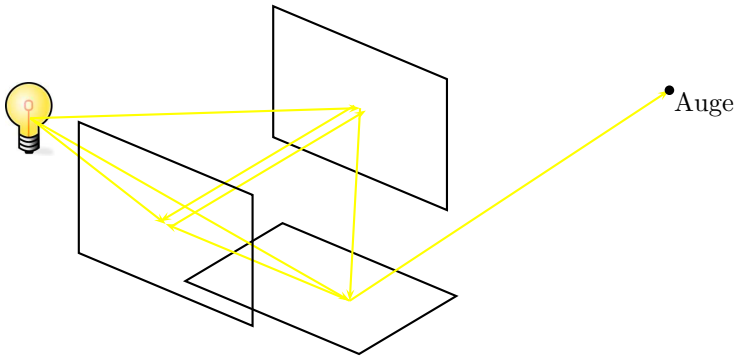
Heutzutage eher historisch.

6 Beleuchtung

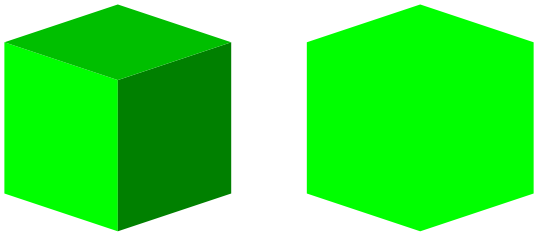
Standardfall:



In Wirklichkeit sehr komplex, in der Computergrafik aber eher selten angewandt:



Beleuchtung ist wichtig um einen räumlichen Eindruck zu erzeugen



Flächenbeschaffenheit:

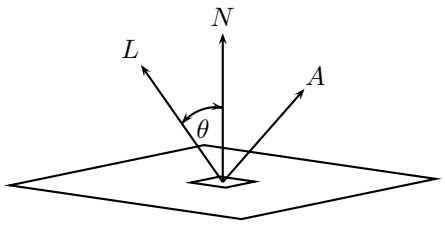
1) diffuse Reflexion; „sieht aus allen Richtungen gleich aus“
2) spiegelndes Reflexion;

} treten auch kombiniert auf

6.1 Diffuse Reflexion

L, N, A sind Einheitsvektoren:

N...Normalvektor der Fläche
L...Vektor zur Lichtquelle
A...Vektor zum Auge
θ...Einfallswinkel (zwischen L und N)



Wie erscheint ein Punkt (Flächenstück) für das Auge?

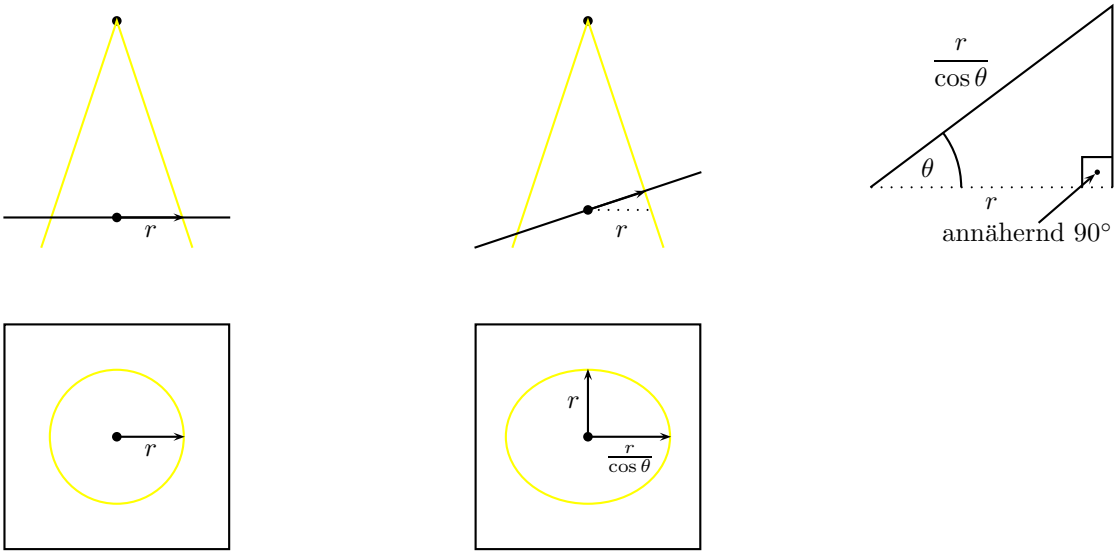
$$I = \text{Intensität}$$

zusätzliche Daten:

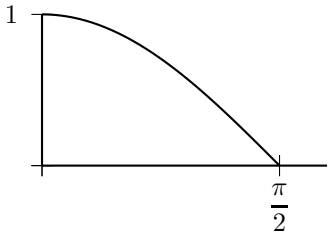
r_L...Abstand zur Lichtquelle
I_L...Intensiät der Lichtquelle
K_D...diffuser Reflexionskoeffizient der Fläche

Abhängigkeit von θ (**Lampert'sches Gesetz**)

$$I = \cos\theta \cdot (\dots) \qquad \cos\theta\langle N, L\rangle$$



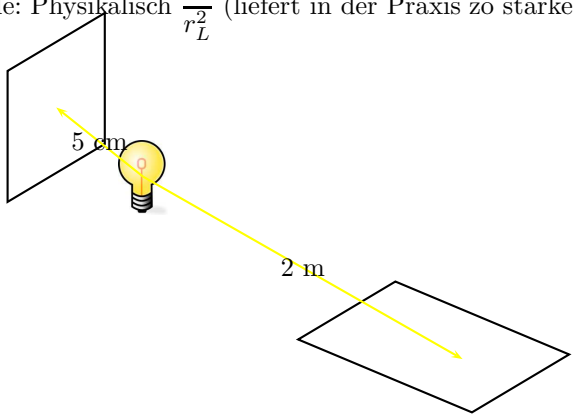
Begründung: Die Menge an Lichtenergie, die auf eine Fläche auftrifft, hängt vom räumlichen Winkel ab, unter dem die Fläche von der Lichtquelle aus erscheint.



Die gleiche Menge an Licht, die vorher auf eine Fläche von $r^2\pi$ gefallen ist, fällt jetzt auf eine Fläche von $r \cdot r\frac{1}{\cos\theta} \cdot \pi$..., pro Flächeneinheit kommt um den Faktor $\cos\theta$ beim Auge weniger an.

$\Rightarrow$ I hängt *nicht von* A ab. (Das ist spezifisch für die diffuse Reflexion)

Abstand zur Lichquelle: Physikalisch $\frac{1}{r_L^2}$ (liefert in der Praxis zo starke Kontraste)



Mit der Formel $\frac{1}{r_L^2}$ würde das einen Faktor 400 bedeuten. Man nimmt stattdessen eine Formel der Gestalt:

$$\frac{1}{C_1 + C_2 r_L^2 + C_3 r_L^2} \qquad \text{für passende Konstanten } C_1, C_2, C_3$$

- Formel für diffuse Reflexion

$$\boxed{I = I_L \frac{1}{C_1 + C_2 r_L^2 + C_3 r_L^2} \cos\theta \cdot K_D} \qquad \left(0 \le \theta \le \frac{\pi}{2}\right)$$

- Diese Formel wird für jede Farbkomponente R, G, B angewandt

$$I^R = {I^R}_L \frac{1}{C_1 + C_2 r_L^2 + C_3 r_L^2} \cos\theta \cdot {K^R}_D$$
$$I^G = {I^G}_L \frac{1}{C_1 + C_2 r_L^2 + C_3 r_L^2} \cos\theta \cdot {K^G}_D$$
$$I^B = {I^B}_L \frac{1}{C_1 + C_2 r_L^2 + C_3 r_L^2} \cos\theta \cdot {K^B}_D$$

- In Wirklichkeit müsste man das für jede Wellenlänge λ getrennt ausrechnen.

$$\cos\theta \text{ durch } \max(0, \langle N, L\rangle)$$

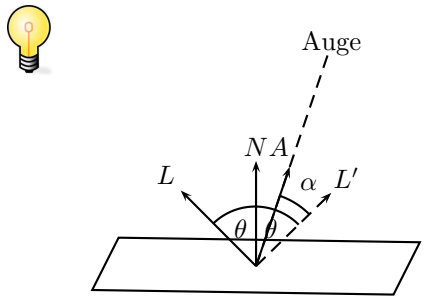
- Für unendlich ferne *Lichtquellen* lässt man den Term $\frac{1}{C_1 + C_2 r_L^2 + C_3 r_L^2}$ weg. I_L hat eine andere Bedeutung. L ist dann konstant.

- Man betrachtet auch „diffuses Licht“. Es kommt aus allen Richtungen gleich stark.

$$I = I_{L,D} \cdot K_D \qquad \text{für } R, G, B \text{ getrennt}$$

- Um die Helligkeit eines Flächenstücks (in R, G, B) zu berechnen, werden die Ergebnisse der verschiedenen Lichtquellen und der diffusen Beleuchtung aufsummiert.

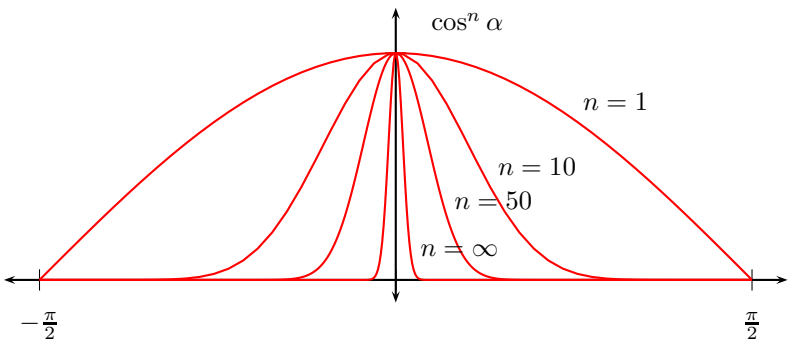
6.2 Spiegelnde Reflexion

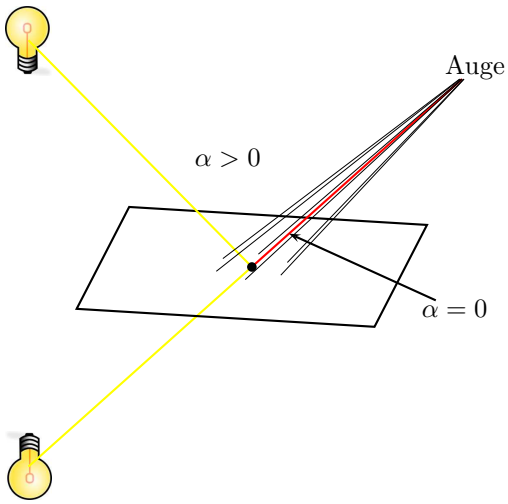


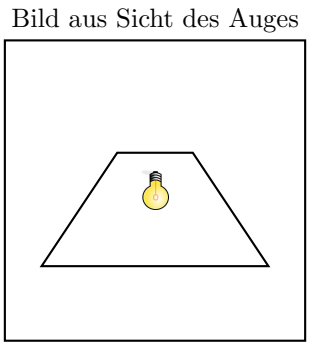
L' ist der an N gespiegelte Vektor, die der „idealen Spiegelung“

$$\alpha = \measuredangle(A, L')$$

Modell von Bui-Tong Phong: Intensität ist proportional zu $\cos^n \alpha$







Formel für spiegelnde Reflexion:

$$I^X = I_L^X \cdot \frac{1}{C_0 + C_1 r + C_2 r^2} \cdot K_S \cdot \cos^n \alpha, \qquad X = R, G, \text{ oder } B$$

I_L... Intensität der Lichtquelle (I_L^R, I_L^G, I_L^B)

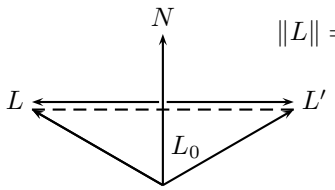
$\frac{1}{C_0 + C_1 r + C_2 r^2}$... Abhängigkeit von der Entfernung der Lichtquelle

K_S... spiegelnde Reflexionskoeffizient *unabhängig von der Farbe!*

n... Exponent: gibt an wie glatt die Spiegelung ist

$\frac{1}{C_0 + C_1 r + C_2 r^2}$... Abhängigkeit von der Entfernung der Lichtquelle

I^R, I^G, I^B... Ergebnis-Intensität aus Sicht des Auges



$\|L\| = \|N\| = \|A\| = 1$

L_0... Projektion von L auf N

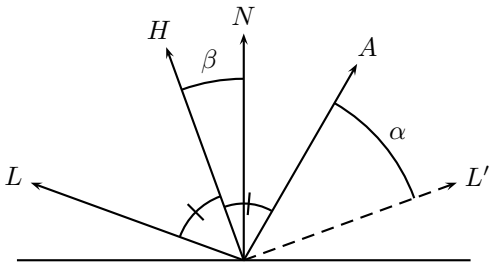
$$\begin{aligned} &= \langle L, N\rangle \cdot N \\ L' &= -(L - L_0) + L_0 = 2L_0 - L, \|L'\| = 1 \\ \cos\alpha &= \langle A, L'\rangle = \langle A, 2L_0\rangle - \langle A, L\rangle \\ &= 2\langle L, N\rangle\langle N, A\rangle - \langle A, L\rangle \\ &= \cos\alpha \end{aligned}$$

Näherungsmodell (einfacher zu rechnen)
H der Vektor, der symmetrisch zwischen A und L liegt.

$$H_0 = A + L \qquad\qquad H = \frac{H_0}{\|H_0\|}$$

Statt $\cos\alpha$ nimmt man $\cos\beta, \beta = \measuredangle(H, N)$

$$\cos\beta = \langle H, B\rangle \qquad \beta = 0 \Leftrightarrow \alpha = 0$$



$\beta = \frac{\alpha}{2}$, wenn L, N, A in einer Ebene liegen. Im Raum ist das nicht immer der Fall.

6.3 Gesamtbeleuchtung

- mehrere Lichtquellen an bestimmten Orten (bzw. aus bestimmten Richtungen) mit Intensität I_L^R, I_L^G, I_L^B
- eine diffuse Lichtquelle mit Intensität I_D^R, I_D^G, I_D^B

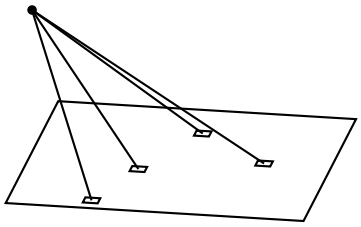
Für jede Fläche:

- diffuse Reflexionskoeffizienten K_D^R, K_D^G, K_D^B
- spiegelnde Reflexionskoeffizienten K_S, Exponent n
- möglicherweise eine Eigenleuchtintensität I_E^R, I_E^G, I_E^B (z. B. Leuchtschirm, glühendes Ofenrohr, flächig leuchtende Lichtquelle)

Für jedes Flächenstück addiert man alle Beleuchtungskomponennten zusammen (falls die Fäche sichtbar vom Auge ist).

$$\begin{aligned} I^R = \sum & \text{ diffuse Reflexionen von allen Lichtquellen, die das Flächenstück beleuchten.} \\ & + I_D^R \cdot K^R \\ & + \text{spiegelnde Reflexionen von allen Lichtquellen, die das Flächenstück beleuchten.} \\ & + I_E^R \end{aligned}$$

analog für I^G, I^B



eigentlich liefert die Rechnung in jedem Punkt der Fläche ein anderes Ergebnis.

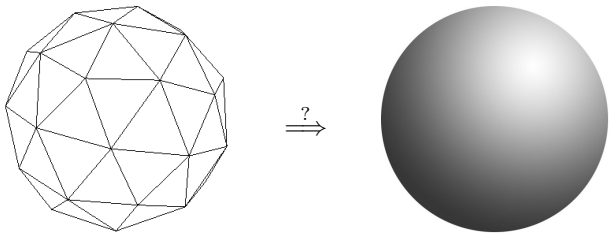
Problem Die Intensität kan in einzelnen Punkten des Bildes > 1 werden. 2 Möglichkeiten

1. herunterskalieren aller Werte
   z. B. $(r; g; b) = (3; 1; 0{,}5) \to \left(1, \frac{1}{3}, \frac{1}{6}\right)$
2. zu große Werte werden auf 1 gesetzt (dadurch können auch Farben verfälscht werden)
   z. B. $(r; g; b) = \underset{\text{(sehr rotes orange)}}{(3; 1; 0{,}5)} \to \underset{\text{(helles gelb)}}{(1, 1, 0{,}5)}$

6.4 Schattierung (Shading)

Alle folgenden Rechnungen in Weltkoordinaten

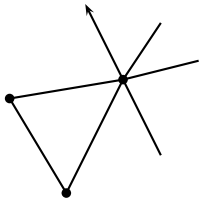
Definition Unter *Shading* versteht man die Anwendung der Beleuchtungsregeln auf jeden Punkt einer Fläche, sodass sich eine abgestufte Farbverteilung ergibt, die realistisch aussieht.



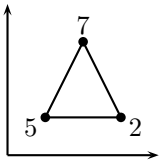
- Gekrümmte Flächen können durch ein genügend feines *Dreiecks*netz (oder Viereckснetz) approximiert werden.
- Große flache Flächen können in kleinere Dreicke zerlegt werden.

Gouraud-Schattierung für ein Dreiecksnetz:

- Berechne die Beleuchtung für jede Ecke eines Dreiecks , mit einem Normalvektor, der für jede Ecke fest ist (unabhängig von dem Dreieck zu dem es gehört). z. B. der Normalvektor der glatten Fläche, die durch das Dreiecksnetz approximiert wird.

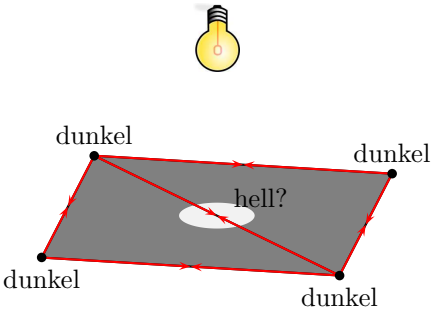


- Interpoliere die Beleuchtung linear auf jedem Dreieck



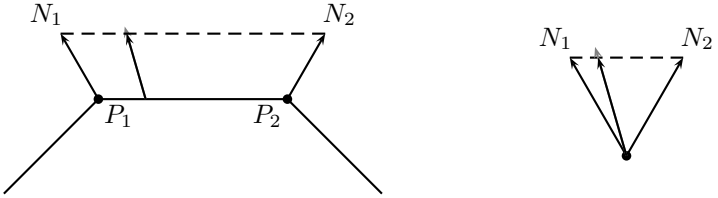
$$f(x,y) = ax + by + c$$

Die Gouraud-Schattierung, versagt z. B. bei Glanzlichtern (können verschwinden oder unnatürlich vergrößert werden) und sehr großen Flächen:



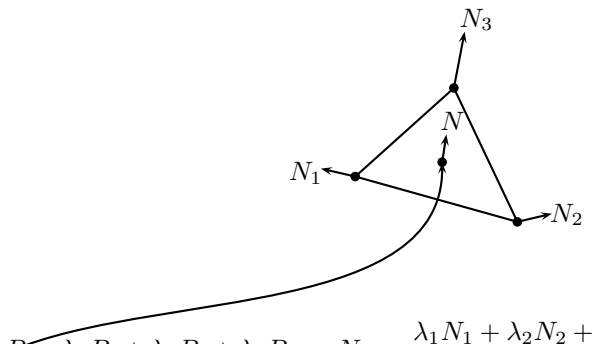
Phong-Schattierung:
~~diffuse Lichtquelle~~ *Umgebungslicht* (ambient light)

- An den Ecken werden die Normalen genommen. Dazwischen werden die Normalenrichtungen angenähert interpoliert und mit diesen Normalen in jedem Punkt die Beleuchtungsrechnung durchgeführt



Am Punkt $P = \lambda P_2 + (1-\lambda)P_1$ wird der folgende Normalvektor genommen:

$$N = \frac{\lambda N_2 + (1-\lambda)N_1}{\|\lambda N_2 + (1-\lambda)N_1\|}$$



$$P = \lambda_1 P_1 + \lambda_2 P_2 + \lambda_3 P_3 \to N = \frac{\lambda_1 N_1 + \lambda_2 N_2 + \lambda_3 N_3}{\|\lambda_1 N_1 + \lambda_2 N_2 + \lambda_3 N_3\|}, \qquad (\lambda_1 + \lambda_2 + \lambda_3 = 1)$$

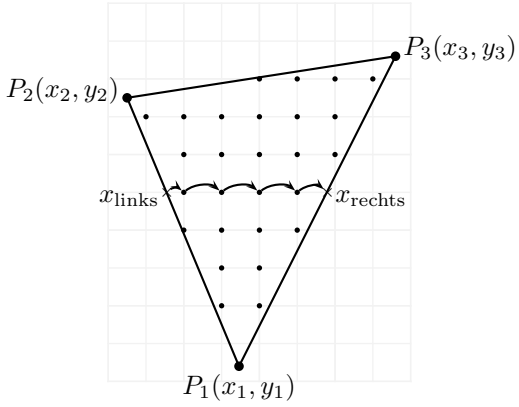
Aufwendiger zu rechnen als Gouraud-Schattierung

6.4.1 Ausfüllen einer gerasterten Fläche

Annahme Fläche ist ein Dreieck

$$y_1 \leq y_2 \leq y_3$$

P_2 links von P_1P_3



in P_1, P_2, P_3 sind Intensitätswerte I_1, I_2, I_3 berechnet worden, diese sollen linear interpoliert werden. (z. B. bei Gouraud-Schattierung) eigentlich sollte man die Interpolation in Weltkoordinten machen, und für R, G, B getrennt.

`Setpixel(`x`,y,`I`)`

ganzzahlig

Wir füllen das Dreieck zeilenweise

```
FuelleZeile(x_links, x_rechts, y, I_links, ΔI)
                                  // ΔI = I(x+1,y) − I(x,y) = unabhaengig von x und y,
                                  // weil I eine lineare Funktion ist
        x := ⌈x_links⌉
        I := I_links + ΔI · (x − x_links)
        while x ≤ x_rechts
                          Setpixel(x,y,I)
                          x := x + 1
                          I := I + ΔI
```

äußere Schleife:

1. untere Hälfte

$$\Delta x_{\text{links}} = \frac{x_2 - x_1}{y_2 - y_1} \qquad \text{Ziel: auf der Geraden } P_1P_2 \text{ gilt } x = y_1 + (y - y_1) \cdot \Delta x_{\text{links}}$$

$$\Delta x_{\text{links}} = \frac{x_3 - x_1}{y_3 - y_1}$$

$$\Delta I_{\text{links}} = \frac{I_2 - I_1}{y_2 - y_1} \qquad \text{Ziel: auf der Geraden } P_1P_2 \text{ gilt } I = I_1 + (y - y_1) \cdot \Delta I_{\text{links}}$$

$$\Delta I = \frac{(I_1 + \Delta I_{\text{rechts}}) - (I_1 + \Delta I_{\text{links}})}{(x_1 + \Delta x_{\text{rechts}}) - (x_1 + \Delta x_{\text{links}})} = \boxed{\frac{\Delta I_{\text{rechts}} - \Delta I_{\text{links}}}{\Delta x_{\text{rechts}} - \Delta x_{\text{links}}}} \qquad \text{mit } \Delta I_{\text{rechts}} = \frac{I_3 - I_1}{y_3 - y_1}$$

$y = \lceil y_1 \rceil$; $x_{\text{links}} := x_1 + (y - y_1) \cdot \Delta x_{\text{links}}$
$x_{\text{rechts}} := x_1 + (y - y_1) \cdot \Delta x_{\text{rechts}}$
$I_{\text{links}} := I_1 + (y - y_1) \cdot \Delta I_{\text{links}}$

```
while y ≤ y2
        FuelleZeile(x_links, x_rechts, y, I_links, ΔI)
        y := y + 1
        x_links := x_links + Δx_links
        x_rechts := x_rechts + Δx_rechts
        I_links := I_links + ΔI_links
```

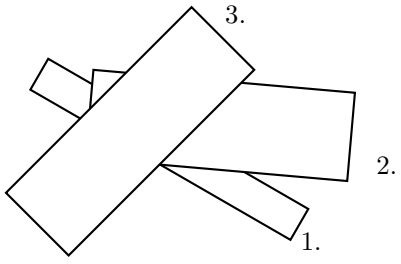
2. obere Hälfte

$\Delta x_{\text{links}} := \frac{x_3 - x_2}{y_3 - y_2}$; $\Delta I_{\text{links}} := \frac{I_3 - I_2}{y_3 - y_2}$
$y := y + 1$; $x_{\text{rechts}} := x_{\text{rechts}} + \Delta_{\text{links}}$
$x_{\text{links}} := x_2 + (y - y_2) \cdot \Delta x_{\text{links}}$
$I_{\text{links}} := I_2 + (y - y_2) \cdot \Delta I_{\text{links}}$

```
while y ≤ y3
        FuelleZeile(x_links, x_rechts, y, I_links, ΔI)
        y := y + 1
        x_links := x_links + Δx_links
        x_rechts := x_rechts + Δx_rechts
        I_links := I_links + ΔI_links
```

6.5 Entfernen verdeckter und teilweise verdeckter Flächen

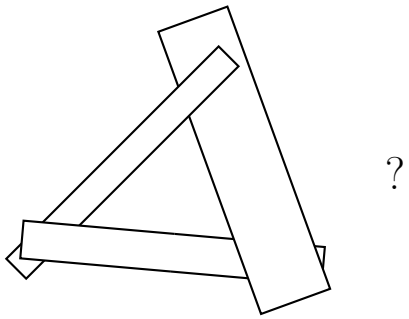
1. Painter's Algorithmus
2. Tiefenpuffer
3. Überstreichen (swap)



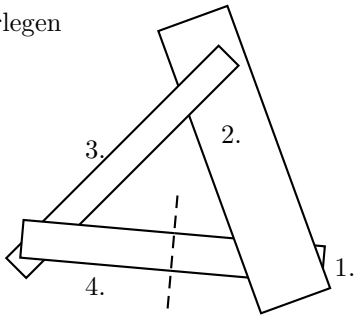
6.5.1 Painter's Algorithmus

Finde eine Reihenfolge der Flächen nach vorn, „male“ die Flächen in dieser Reihenfolge.

Problem zyklische Abhängigkeiten:



Lösung Flächen in kleinere Flächen zerlegen



6.5.2 Tiefenpuffer (z-Puffer)

Jeder Bildpunkt speichert zusätzlich einen z-Wert (größere Werte sind weiter hinten)

`Setpixel(`x`,`y`,`z`,`I`) bzw.` `Setpixel(`x`,`y`,`z`,`I^R`,`I^G`,`I^B`)`

Vergleiche z mit dem für (x, y) gespeicherten z-Wert, wenn kleiner, dann überschreibe z, I, andernfalls ignoriere.

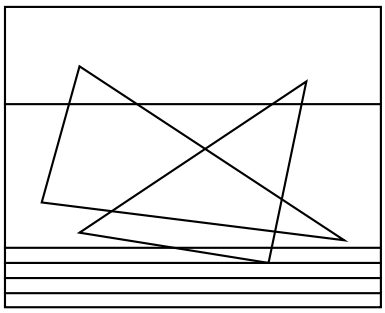
Beim Zeichnen eines ebenen Flächenstückes ist z linear von x und y abhängig. z interpoliert zwischen den 3 Ecken $P_i(x_i, y_i, z_i), i = 1, 2, 3$

Nachteil (von beiden Verfahren) Aufwand ist proportional zu gezeichneten Gesamtfläche und nicht zur sichtbaren Fläche (nicht sichtbare Flächen werden auch gezeichnet bzw. zu zeichnen versucht).

6.5.3 Darstellung einer Szene durch Überstreichen (Scanline-Algorithmus)

Vereinfachte Annahme Szene besteht auzs Dreiecken.

- Der Algorithums beruht auf der gleichen Idee wie der Algorithmus zum Ausfüllen gerasterter Flächen (s. Abschnitt 6.4.1)
  - Überstreiche Szenee mit horizontalen Geraden (von unten nach oben, mit steigendem y)
    - Für jede Rasterzeile gehe Pixel von links nach rechts durch (mit steigendem x)
- Prinzip: Der Algorithmus aus Abschnitt 6.4.1 wird für alle Dreiecke parallel angewendet



Vorverarbeitung Wir erstellen eine Kantenliste (Edge List, EL), sortiert nach den Koordinaten des unteren Endpunktes der Kante, zuerst nach der y-Koordinate sortiert, bei gleicher y-Koordinate nach x-Koordinate (horizontal Kanten brauchen nicht gespeichert werden).

Bearbeitung einer Zeile Wir benötigen zur Abarbeitung einer Zeile die Menge der von der Zeile geschnittenen Dreiecke und für diese dann die nötigen Parameter, wie sie für die Fülle-Zeile-Funktion aus Abschnitt 6.4.1 benutzt wurden.

Wir verwalten die Daten in einer *„Aktive-Kanten-Liste“* (AEL). Darin enthalten sind alle Kanten, die die aktuelle Scanline schneiden. Sie sind sortiert nach der x-Koordinate des Schnittpunktes mit der Scanline. Zu jeder Kante speichern wir die Informationen, die wir benötigen. Um das Dreieck für diese Zeile zu rastern und die entsprechenden Werte für die Zeile auszurechnen.

z. B.

- zugehöriges Dreieck
- linke oder rechte Begrenzung
- y-Koordinate des oberen Endpunktes (um die Kante au der AEL entfernen zu können)
- x-Koordinate Schnittpunktes mit der Scanline (entspricht x_{links} bzw. x_{rechts})
- Δx (entspricht Δx_{links} bzw. Δx_{rechts})

die folgenden Daten brauchen nur in einer linken Begrenzung gespeichert werden:

- $z_{\text{links}}, I_{\text{links}}, \Delta z_{\text{links}}, \Delta I_{\text{links}}, \Delta z, \Delta I$ (Notation s. Abschnitt 6.4.1)

und entsprechende Werte für andere zu interpolierende Daten.

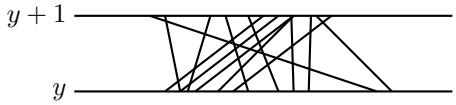
Änderung der AEL beim Übergang von y auf $y+1$ wie bei der äußeren Schleife in Abschnitt 6.4.1.

- inkrementiere $x, z_{\text{links}}, I_{\text{links}}$ um $\Delta x, \Delta z_{\text{links}}, \Delta I_{\text{links}}$

Außerdem kann es nötig sein Kanten aus der Liste zu löschen (Kante endet zwischen y und $y+1$), neue Kanten aufzunehmen (Kante beginnt zwischen y und $y+1$) oder die Reihenfolge der Kanten zu ändern (Kanten schneiden sich zwischen y und $y+1$)

1. **Kante endet zwischen y und $y+1$**
   Vergleiche y-Koordinate des oberen Endpunktes einer Kante in der AEL mit der aktuellen Scanline-Koordinate
2. **Kante beginnt zwischen y und $y+1$**
   Vergleiche y-Koordinate des unteren Endpunktes der ersten noch nicht verwendeten Kante in der EL mit aktuellen Scanline-Koordinaten.
3. **Kanten schneiden sich zwischen y und $y+1$**
   prinzipiell vergleiche die neuen x-Werte zweier Kanten die in der alten AEL benachbart waren

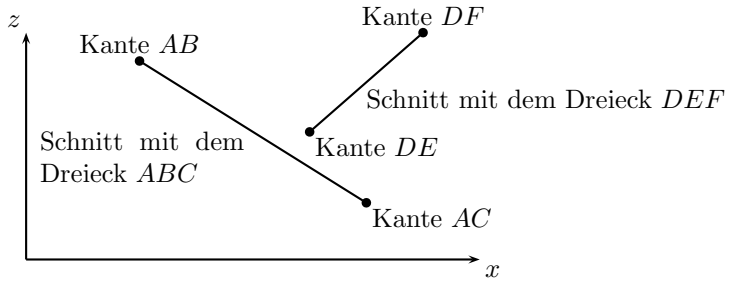
Problem Die Scanline wird nicht Kontinierlich



Es können sich auch nichtbenachbarte Kanzen zwischen y und $y+1$ schneiden, wenn die dazwischen liegenden Kanten ebenfalls geschnitten werden. Wir müssen also nicht nur bei y benachbarte Kanten vergleichen, sondern auch die nache einer Kreuzung entstehenden neuen Nachbarn.

Dieser Ansatz läuft auf ein Buble-Sort- oder Insertion-Sort-Prinzip hinaus. Dann ist Neusortieren mit effizienten Sortierverfahren.

Füllen einer Zeile Zeile wird von links nach rechts durchlaufen. Beachten wir dabei die Tiefeninformation entspricht dies einem Überstreichen einer Ebene $y = \textbf{const}$ mit einer Geraden $x = \textbf{const}$



in dieser Ebene haben wir Kante, die den Schnitten von Dreiecken mit der Ebene entsprechen. Der Dreiecksschnitt, dessen z-Koordinate für ein gegebenes x am kleinsten ist, bestimmt die Färbung des Pixels.

Analog zur AEL legen wir eine aktive Dreiecksliste (ATL, auch active span list) an, om der die von der aktiven Geraden $x = \textbf{const}$ geschnittenen Dreiecksschnitte nach z-Koordinate sortiert sind und die zugehörigen Werte für die Interpolation gespeichert sind. Das sind z, I und Δz, sowie ΔI und x_{rechts}.

Modifizierung der ATL beim Übergang von x auf $x+1$

1. Dreieck einfügen: x_{links} liegt zwischen x und $x+1$ für eine linke Kante in der AEL
2. Dreieck löschen: x_{rechts} liegt zwischen x und $x+1$
3. Zwei Dreiecke schneiden sich (z-Werte tauschen Reihenfolge)

zu zeichnen ist die aktuelle Farbe des Dreiecks, das in der ATL aktuell an der ersten Stelle steht.

Varianten, Sonderfälle, ...

1. Es ist häufig sinnvoll, sobald zwei Kanten in der AEL benachbart werden, den Schnittpunkt zu berecnhen und diesen, falls er existiert, in einem *Event-Schedule* einzufügen. Dieser verwaltet die „Zeit“-punkte, an denen sich die Reihenfolge der Kanten ändert in einer Prioritätswarteschlange.
2. Vereinfachungen bei sich nicht schneidenden Dreiecken
3. Kombination mit dem Tiefenpuffer
   z. B. tiefenpuffer für eine Zeile des Bildes
   - Überstreiche Szene zeilenweise
   - Vor jedem Wechsel in eine neue Zeile leere Tiefenpuffer
   - für jede Zeile: Zeichne alle Dreiecke in der aktuellen AEL (in beliebiger Reihenfolge) nach dem Tiefenpuffer-Prinzip

   Analog: Tiefenpuffer für ein Pixel

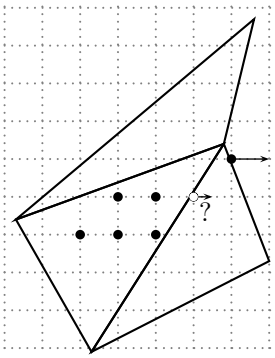
Größe des Tiefenpuffers	Struktur des Algorithmus

Breite × Höhe × Genauigkeit	`Für alle Dreiecke`
	`  Für alle Zeilen`
	`  Für alle Spalten`

Breite × Genauigkeit	`Für alle Zeilen`
	`  Für alle Dreiecke`
	`  Für alle Spalten`

Genauigkeit	`Für alle Zeilen`
	`  Für alle Spalten`
	`  Für alle Dreiecke`

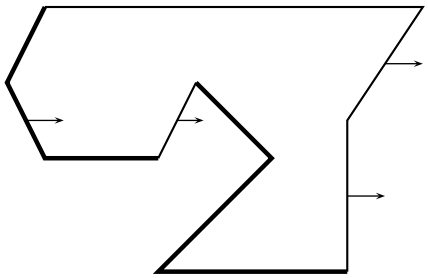
6.6 Anzeigen von benachbarten Flächenstücken (Dreiecken)



Gehört der Rand eines Dreiecks zum Dreieck?
Zu welchem Dreieck?

Gesucht Eine konsistente Regel, die bei mehreren aneienanderstoßenden Flächen feslegt, zu welcher Fläche jeder Bildpunkt gehört.

Eine Möglichkeit Ein Pixel, das auf dem Rand liegt wird (in Gedanken) horizontal nach rechts verschoben. Es gehört zu der Fläche, wo es dann landet. Wenn es dabei auf einer horizontalen Kante liegt, dann wird es zur oberen Fläche gerechnet.



Vorraussetzung Exakte Arithmetik

6.7 Lineare Interpolation in Weltkoordinaten

Interpolation von Helligkeitswerden (Gouraud-Schattierung), von Normalenrichtungen (Phong-Schattierung), bei Texturen immer in *Weltkoordinaten*! Nicht NDC.

Gegeben Eine Strecke P_1P_2 in NDC (oder Bildschirmkoordinaten) als Bild einer Strecke P_1P_2 in Weltkoordinaten.

An den Endpunkten sind Werte I_1, I_2 gegeben
(Intensitätswerte $I_1^R, I_1^G, I_1^B, ...$, oder Normalenrichtungen $N_1^x, N_1^y, N_1^z, ...$)
Wir wollen diese Werte für die dazwischen liegenden Punkte der Strecke in Weltkoordinaten interpolieren.

Wir kennen die Transformationsmatrix $M = M^{\text{Welt,NDC}}$:

$$M \begin{pmatrix} x_{\text{Welt}} \\ y_{\text{Welt}} \\ z_{\text{Welt}} \\ w_{\text{Welt}} \end{pmatrix} = \begin{pmatrix} x_{\text{NDC}} \\ y_{\text{NDC}} \\ z_{\text{NDC}} \\ w_{\text{NDC}} \end{pmatrix} \qquad P_1 = \begin{pmatrix} x_1^{\text{NDC}} \\ y_1^{\text{NDC}} \\ z_1^{\text{NDC}} \\ 1 \end{pmatrix} \qquad P_2 = \begin{pmatrix} x_2^{\text{NDC}} \\ y_2^{\text{NDC}} \\ z_2^{\text{NDC}} \\ 1 \end{pmatrix}$$

$$P_1 \text{ in homogenen Weltkoordinaten} = M^{-1} \begin{pmatrix} x_1^{\text{NDC}} \\ y_1^{\text{NDC}} \\ z_1^{\text{NDC}} \\ 1 \end{pmatrix} = \begin{pmatrix} x_1^{\text{Welt}} \\ y_1^{\text{Welt}} \\ z_1^{\text{Welt}} \\ w_1^{\text{Welt}} \end{pmatrix} \hat{=} \begin{pmatrix} x_1^{\text{Welt}}/w_1^{\text{Welt}} \\ y_1^{\text{Welt}}/w_1^{\text{Welt}} \\ z_1^{\text{Welt}}/w_1^{\text{Welt}} \end{pmatrix} \text{ in kartesischen Weltkoordinaten}$$

Annahme: $I : \mathbb{R}^3 \to \mathbb{R}$ linear $I(P_1) = I_1 \qquad I(P_2) = I_2$

Betrachte eine „homogene Erweiterung“: $\hat{I} : \mathbb{R}^4 \to \mathbb{R}$ von I

$$\hat{I}\left(\begin{pmatrix} \lambda x \\ \lambda y \\ \lambda z \\ \lambda w \end{pmatrix}\right) = \lambda \cdot \hat{I}\left(\begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix}\right) \qquad \hat{I}\left(\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}\right) = I\left(\begin{pmatrix} x \\ y \\ z \end{pmatrix}\right)$$

1. Die Funktion $\hat{I}$ ist linear;
2. Aus $\hat{I}$ kann man I ausrechnen:

$$I\left(\text{Punkt } \begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix} \text{ in homogenen Koordinaten}\right) = \frac{1}{w} \cdot \hat{I}\left(\begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix}\right)$$

$$I\left(\text{Punkt in homogenen Koordinaten } \begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix}\right) =$$

$$I\left(\text{Punkt in homogenen Koordinaten } \begin{pmatrix} x/w \\ y/w \\ z/w \\ 1 \end{pmatrix}\right) = \hat{I}\left(\begin{pmatrix} x/w \\ y/w \\ z/w \\ 1 \end{pmatrix}\right) = \frac{1}{w} \cdot \hat{I}\left(\begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix}\right)$$

$$I\left(\text{Punkt in karthesischen Koordinaten } \begin{pmatrix} x/w \\ y/w \\ z/w \end{pmatrix}\right) =$$

1. Berechne P_1 und P_2 in homogenen Weltkoordinaten:

$$\begin{pmatrix} x_1^{\text{Welt}} \\ y_1^{\text{Welt}} \\ z_1^{\text{Welt}} \\ w_1^{\text{Welt}} \end{pmatrix} = M^{-1} \cdot \begin{pmatrix} x_1^{\text{NDC}} \\ y_1^{\text{NDC}} \\ z_1^{\text{NDC}} \\ 1 \end{pmatrix} \text{ und } \begin{pmatrix} x_2^{\text{Welt}} \\ y_2^{\text{Welt}} \\ z_2^{\text{Welt}} \\ w_2^{\text{Welt}} \end{pmatrix} = M^{-1} \cdot \begin{pmatrix} x_2^{\text{NDC}} \\ y_2^{\text{NDC}} \\ z_2^{\text{NDC}} \\ 1 \end{pmatrix}$$

2. Berechne $\hat{I}_1$ und $\hat{I}_2$ an diesem Punkt in $\mathbb{R}^4$

$$I(P_1) = I_1 = \frac{1}{w} \cdot \hat{I}_1$$

$$\boxed{\hat{I}_1 = w_1^{\text{Welt}} \cdot I_1,\ \hat{I}_2 = w_2^{\text{Welt}} \cdot I_2}$$

3. Lineare Interpolation $P(\lambda) = \lambda_2 + (1-\lambda)P_1$ in NDC

Gesucht $I(\lambda) = I(P(\lambda))$

$$\begin{pmatrix} x(\lambda) \\ y(\lambda) \\ z(\lambda) \\ 1 \end{pmatrix} = \lambda \begin{pmatrix} x_2 \\ y_2 \\ z_2 \\ 1 \end{pmatrix} + (1-\lambda) \begin{pmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{pmatrix}$$

$$\boxed{\hat{I} = \lambda \cdot \hat{I}_2 + (1-\lambda)\hat{I}_1}$$

$$\boxed{w^{\text{Welt}}(\lambda) = \lambda \cdot w_2^{\text{Welt}} + (1-\lambda)w_1^{\text{Welt}}} \qquad I(\lambda) = \frac{\hat{I}(\lambda)}{w^{\text{Welt}}(\lambda)}$$

Zusammenfassung $w_1^{\text{Welt}}, w_2^{\text{Welt}}$ = letzte Komponente von $M^{-1} \begin{pmatrix} \cdots \end{pmatrix}$

$$\boxed{I(\lambda)} = \frac{\overbrace{w_2^{\text{Welt}} \cdot I_2}^{\hat{I}_2} \cdot \lambda + \overbrace{w_1^{\text{Welt}} \cdot I_1}^{\hat{I}_1} \cdot (1-\lambda)}{w_2^{\text{Welt}}\lambda + w_1^{\text{Welt}}(1-\lambda)}$$

Einbau in Füllalgorithmus zusätzlich

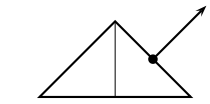
```
FuelleGerade(x1, x2, y, z1, z2, w1, w2, I1, I2)
        Î1 = w1 I1
        Î2 = w2 I2
        ΔÎ = ...
        Δw = ...
        ⋮
        while x < x2
                        Setpixel(x, y, ..., I)
                        x := x + 1
                        z := z + Δz
                        Î := Î + ΔI  // Zaehler
                        w := w + Δw       // Nenner
                        I = Î / w
```

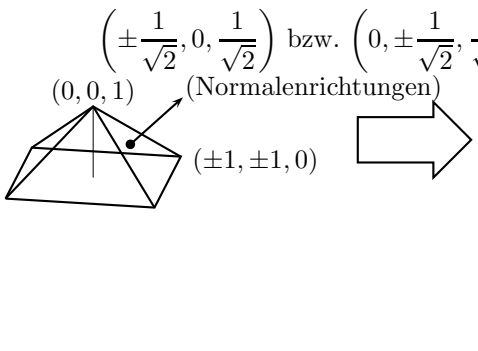
analog für mehrere Größen I^R, I^G, I^B nimmt man $\hat{I}^R$, $\hat{I}^G$, $\hat{I}^B$, w und Δw braucht man nur einmal.

Bemerkung Die z-Koordinate kann man direkt in NDC interpolieren

6.8 Transformation von Normalvektoren bei affinen Transformationen

z. B. Objektkoordinaten auf Weltkoordinaten





$$T = \begin{pmatrix} b/2 & 0 & 0 & g_x \\ 0 & b/2 & 0 & g_y \\ 0 & 0 & h & g_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$T = \left(\begin{array}{c|c} A & b \\ \hline 0 & 1 \end{array}\right) \qquad \begin{pmatrix} x \\ y \\ z \end{pmatrix} \mapsto \underbrace{A \begin{pmatrix} x \\ y \\ z \end{pmatrix} + b}$$

Ebene im Urbildraum

$$E = \left\{ \vec{x} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \;\middle|\; \vec{n}^T \cdot \vec{x} = c \right\} \qquad \vec{n}\text{...Normalenvektor, } c \in \mathbb{R}$$

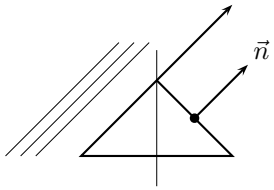
$$T(E) = \{\underbrace{A \cdot \vec{x} + b}_{=\bar{x}} \mid \vec{n}^T \cdot \vec{x} = c\}$$

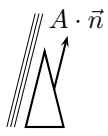
$$Ax + b = \bar{x} \qquad x = A^{-1}(\bar{x} - b)$$

$$\begin{aligned} T(E) &= \left\{\bar{x} \mid n^T A^{-1} (\bar{x} - b) = c\right\} \\ &= \left\{\bar{x} \mid n^T A^{-1} \bar{x} = c + n^T A^{-1} b\right\} \\ &= \left\{\bar{x} \;\middle|\; \underbrace{\left(\left(A^{-1}\right)^T n\right)^T}_{\text{Normalvektor der Ebene } T(E)} \bar{x} = \bar{c}\right\} \end{aligned}$$

Bei einer Affinen Transformation mit 3×3-Transformationsmatrix A muss man Nornalvektoren nicht mit A, sondern von links nach rechts mit $(A^{-1})^T$ multiplizieren und anschließend *normieren*. (Vektoren als Spalten betrachten)

Falls A orthogonal ist, dann ist $A^T = A^{-1}$, $(A^{-1})^T = A$





7 Texturen

(Abbildungen folgen später)
Aufbringen eines Musters auf eine Fläche.

Das Muster – die Textur – ist als zweidimensionales *Bild* gegeben (Pixelarray), und eine Abbildung von diesem Bild auf eine Fläche im Raum.

Die Bildfläche muss nicht unbedingt flach sein:

1. gekrümmt

2. Dreiecksgitter

   Dann muss man das Bild in ein dreicksgitter mit derselben kombinatorischen Struktur zerlegen. Die Abbildung erfolgt dann durch lineare Interpolation auf jedem Dreieck.

Bei beiden Lösungen existiert das Problem der Verzerrung.

Einfaches Modell für Architektur: Fassade wird auf die Hasuwand aufgeklebt.

Problem Schatten sind möglicherweise nicht richtig.
Bei der Darstellung benötigt man die umkehrabbildung von der Fläche in Weltkoordinaten (bzw. von der proyezierten Fläche in NDC) auf die Textur.
Bei der linearen Interpolation von *Dreiecken* verwendet man gerne *baryzenztische Koordinaten*

$$(\lambda_1, \lambda_2, \lambda_3) \text{ mit } \lambda_1 + \lambda_2 + \lambda_3 = 1$$

stellt den Punkt $\lambda_1 A_1 + \lambda_2 A_2 + \lambda_3 A_3$ dar.

$$\lambda_1, \lambda_2, \lambda_3 \geq 0 \Leftrightarrow \text{ Punkt im Dreieck}$$

Die Ecken des Dreiecks auf dem Objekt haben barizentrische Koordinaten

$$(\lambda_1, \lambda_2, \lambda_3) = (1,0,0), (0,1,0), (0,0,1)$$

dazwischen wird linear *interpoliert* (in Weltkoordinaten). Dann nimmt man in der Textur den Punkt an der Stelle $\lambda_1 B_1 + \lambda_2 B_2 + \lambda_3 B_3$.

Möglcihe Probleme dadurch, das Auflösung des erzeugten bilder nicht mit der Auflösung der Textur übereinstimmt.

a) Auflösung der Textur zu klein
   ⇒ Bild wirkt grob gepixelt (mögliche Lösung: Interpolation)

b) Auflösung der Textur ist größer
   ⇒ Es kann zu Aliasing-Effekten kommen. (mögliche Auswege s. Abschnitt 4.4)

8 Schatten

Für die Berecnhung von chatten kann man die gleichen Methoden verwenden, wie zur Bestimmung sichtbarer Flächen:

- von der Lichtquelle aus „sichtbar“ $\Rightarrow$ beleuchtet.
- von der Lichtquelle aus „verdeckt“ $\Rightarrow$ Schatten.

1. Berechne (für jede Lichtquelle unabhängig) in einem getrennten Puffer das „Bild“, dass von der Lichtquelle aus „gesehen“ wird.
   z. B. mit dem Tiefenpuffer `PutPixel(`x`,`y`,`z`,`I`)`

   Index (Nummer) der Fläche, die gerade dargestellt wird.

2. Erzeuge das Bild, dass vom Auge aus sichtbar ist Pixel für Pixel.

   P sei der Punkt (in Weltkoordinaten), der gerade dargestellt wird. Er gehört zum Objekt mit Nummer n. Bestimme das Bild von P im Bildpuffer, der zur Lichtquelle gehört und vergleiche den Index, der dort gespeichert ist.

   Wenn $= n \Rightarrow$ Punkt P ist von dieser Lichtquelle beleuchtet.

9 Clipping

Eine Strecke auf ein Rechteck zurechtstutzen.

9.1 Algorithmus von Cohen-Sutherland

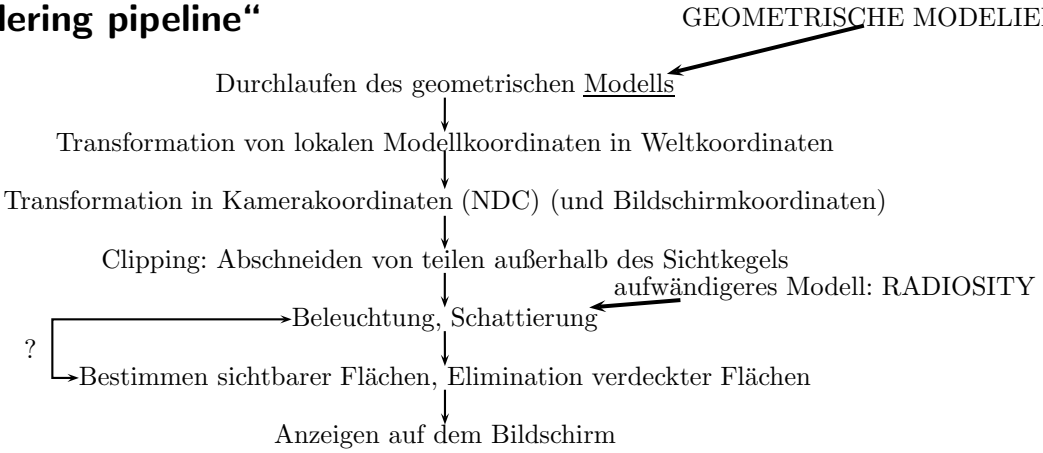
$\mathbb{R}^2 \to \{0,1\}^4$ (4 Bits)

$$x < x_{\min}$$
$$x > x_{\max}$$
$$y < y_{\min}$$
$$y < y_{\max}$$

1. Wenn A_1 und A_2 ein gemeinsames 1-Bit haben.
   $\Rightarrow$ Strecke A_1A_2 schneidet Rechteck nicht.
2. Wenn beide Codes 0000 sind.
   $\Rightarrow$ Strecke liegt zur Gänze im Inneren.
3. Bestimme eine Stelle, wo sich beide Codes unterscheiden,
4. Schneide die Gerade mit der entsprechenden horizontalen oder vertikalen Geraden.
5. Ersetze den Endpunkt mit dem 1-Bit durch den Schnittpunkt.
6. Wiederhole

10 Geometrische Modellierung

10.1 „Rendering pipeline“



nur eine typische Reihenfolge!

Alle diese Operationen werden heute von Hardwareausgeführt

Alternative für Beleuchtung, Schattierung und Bestimmung sichtbarer Flächen: Raytracing (Strahlverfolgung)

10.2 Geometrische Modellierung

10.2.1 Wie kann man geometrische Formen darstellen?

Möglichkeit 1) *geometrische Grundformen* (Kugel, Zylinder, Polygone) werden zusammengesetzt

Grundmodell: triangulierte Flächen.

Mit triangulierten Flächen lässt sich im Prinzip alles genügend fein approximieren

(flache Polygone mit mehr als 3 Ecken sind in der Regel auch zugelassen)

Möglichkeit 2) *Implizite Flächen* Fläche ist durch eine Gleichung gegeben

$$(x^2 + y^2)^2 + 3z^3xy - 28xz^2 = 0$$

Möglichkeit 3) parametrische Fläche
z. B. Kugel:

$$\begin{aligned} x &= \cos\varphi \cos\vartheta \\ y &= \sin\varphi \cos\vartheta \\ z &= \sin\vartheta \end{aligned}$$

$$0 \le \varphi \le 2\pi \qquad -\pi \le \vartheta \le \pi$$

Operation	**Dreicksgitter**	**implizite Fläche**	**parametrische Fläche**
Erzeugen eines Punktes auf der Fläche	sehr leicht	schwierig	sehr leicht
Durchlaufen aller Punkte der Fläche	sehr leicht	sehr schwierig	leicht
Schnitt der Fläche mit einem Sichtstrahl (z. B. beim Raytracing)	durchschnittlich[1] (Modell durchlaufen)	leicht (Lösen eines Gleichungssystems in einer Variablen)	sehr schwierig (Gleichungssystem mit 3 Variablen)
geometrische Manipulationen	sehr leicht	schwierig	durchschnittlich
Speicherverbrauch	ist abhängig von der Genauigkeit	kompakt	kompakt

Umwandlung von impliziten oder paramtrischen Flächen in ein Dreicksgitter: „Gittererzeugung“ („Meshing“)

10.2.2 Darstellung polyedrischer Flächen bzw. Polygongittern

1. **primitiv:** Liste von Dreiecken, Vierecken, etc.

$$D_1 : (x_1, y_1, z_1), (x_2, y_2, z_2), (x_3, y_3, z_3)$$
$$D_2 : (x_4, y_4, z_4), (x_5, y_5, z_5), (x_6, y_6, z_6)$$
$$\vdots$$

   Problem: in einem typischen Dreiecksgitter kommen die gleichen Punkte mehrfach vor. Es ist Verschwendung, die gleichen Daten mehrfach zu speichern bzw. zu übertragen.

2. Trennung der Koordinaten von der kombinatorischen Struktur:

$$P_1 : (x_1, y_1, z_1) \qquad D_1 : 1, 2, 4$$
$$P_2 : (x_2, y_2, z_2) \qquad D_2 : 2, 3, 4$$
$$\vdots \qquad\qquad \vdots$$

   Flächen sind typischerweise *orientiert*: Dreieck P_1, P_2, P_3 und Dreieck P_1, P_3, P_2 sind Vorder- und Rückseite des gleichen Dreiecks im Raum, un sie können unterschiedliche Materialeigenschaften haben.

   Um noch mehr zu sparen, werden Dreiecke zu alternierenden Dreiecksstreifen bzw. Dreiecksfächern zusammengefasst. (und Vierecke zu Viereckssstreifen)

   Für n Dreicke benötigt man nur $n + 2$ Punkte. Für isolierte Dreicke werden $3n$ Punkte benötigt.

Datenstruktur zum Speichern der kombinatorischen Struktur einer polyedrischen Fläche

- Doubly-connected edge list (*DECL*, doppelt-verkettete Kantenliste)
- Quad-edge data structure
- „winged-edge“ data structure

Jede Kante wird durch zwei *Halbkanten* repräsentiert, eine in jede Richtung

Jede Halbkante hat 3 Zeiger:

- $l(e), r(e), ...$ linke und rechte Nachbarkante mit demselben Startknoten

[1] weder leicht noch schwer

- $u(e)$... Umkehrkante

e	$r(e)$	$l(e)$	$u(e)$
1	2	3	4
2	$\emptyset$	1	5
3	1	$\emptyset$	6
4	7	8	1
$\vdots$	$\vdots$	$\vdots$	$\vdots$

- Jede Kante liegt an bis zu 23 Flächen an, eine links und eine rechts.
- Durchlaufen aller Kanten, die von einem Knoten ausgehen
- Wir benötigen eine erste Kante e_1

$$\begin{aligned} e_2 &= r(e_1) \\ e_3 &= r(e_2) \\ &\vdots \end{aligned}$$

bis $e_i = e_1$ oder $e_i = \emptyset$

Durchlaufen aller Kanten einer Fläche, die rechts von e_1 liegt.

$$\begin{aligned} e_2 &= l(u(e_1)) \\ e_3 &= l(u(e_2)) \\ &\vdots \end{aligned}$$

bis $e_i = e_1$

$$\begin{aligned} l(r(e)) &= e && \text{falls } r(e) \neq 0 \\ r(l(e)) &= e && \text{falls } l(e) \neq 0 \\ u(u(e)) &= e \end{aligned}$$

10.3 Darstellung von geometrischen Modellen in Grafikbibliotheken

typische Funktionsaufrufe in einer Grafikbibliothek wie OpenGL

```
startPolygon();
        setzeDiffuseFarbe(r,g,b);
        // ... weitere Materialeigenschaften
        Knoten(x1,y1,z1);
        Knoten(x2,y2,z2);
        Knoten(x3,y3,z3);
endPolygon();
startPolygon();
        Knoten(x1,y1,z1);
        Knoten(x6,y6,z6);
        Knoten(x5,y5,z5);
        Knoten(x3,y3,z3);
        Knoten(x2,y2,z2);
endPolygon();
// ...
```

Modellkoordiaten $\rightarrow$ Weltkoordinaten $\rightarrow$ ...

CT = „current transformation matrix“
aktuelle Transformationsmatrix

typischer Ablauf:

- Am Anfang wird auf CT die Identität initialisiert.
- eine Folge von geometrischen Transformationen; die entsprechenden Transformationen werden *von rechts* an CT dranmultipliziert.
- Dazwischen werden die aktuellen Werte von CT auf einem Stapel gespeichert (Push) bzw. wiederhergestellt (Pop)

```
setze af Einheitsmatrix                          // CT = I
        multiply(T3)                             // CT = T3
                multiply(T2)                     // CT = T3 · T2
                Beschreibe Turm
                        push
                                multiply(T1)     // CT = T3 · T2 · T1
                                beschreibe Zinne...
                        pop
                        push
                                multiply(T1)     // CT = T3 · T2 · T1'
                                beschreibe Zinne...
                        pop
                        ⋮
```

10.4 Constructive Solid Geometry (CSG)

10.4.1 Grundlagen

Objekte werden als gefülltes Volumen betrachten, nicht als Randflächen.

- **Grundkörper:** Würfel, Zylinder; konvexe Polyeder
- **Boolesche Operationen:** $\cup$, $\cap$, $\setminus$ (eigentlich Mengenoperationen) z. B:

Man möchte nur reguläre Mengen betrachten. Eine Menge K ist regulär, wenn sie der Abschluss ihres Inneren ist:

$$K = \overline{K^\circ}$$

zur Erinnerung: Das Innere der Menge A ist definiert mit

$$A^\circ = \{x \in A \mid \exists \varepsilon > 0 : \text{ Kreis mit Radius } \varepsilon \text{ um } x \subseteq A\}$$

Der Abschluss von A ist definiert mit

$$\overline{A} = \mathbb{R}^2 \setminus (\mathbb{R}^2 - A)^\circ$$

(bzw. mit $\mathbb{R}^{\not 2}$ im Raum)

Der Rand von A ist definiert mit

$$\partial A := \overline{A} - A^\circ$$

$\Rightarrow$ regularisierte Boolesche Operationen

$$A \cup^* B := \overline{(A \cup B)^\circ}$$
$$A \cap^* B := \overline{(A \cap B)^\circ}$$
$$A \setminus^* B := \overline{(A \setminus B)^\circ}$$

10.4.2 Darstellung in der Computergrafik

$$A \cup^* [(B \cap^* C) \setminus^* (D \cap^* E)]$$

Umwandlung in eine expliziete Darstellung des Randes ist aufwändig.

Man mss Snitte zwischen den Grundobjekten bestimmen. Anschließend muss manberechnen, welche der ausgeschnittenen Oberflächenstücke den Rand des Ergebnisses bilden.

Schnitt mit einer Geraden (z. B. einem Sichtstrahl beim Raytraycing) ist relativ einfach:

- Schneide den Strahl mit allen Grundbausteinen. sortiere die Schnittpunkte, bestimme den ersten Schnittpunkt, wo der Strahl in den Körper eintritt.

11 Strahlverfolgung (Raytracing)

gut geeignet für spiegelnde und transparente Körper.

A) **„Forward raytracing“** Schicke Lichtstrahlen (Lichtteilchen) von der Lichtquelle aus und verfolge ihren Weg.

   Die allermeisten Teilchen werden irgendwo absorbiert, bevor sie ins Auge gelangen.
   $\Rightarrow$ extrem aufwändig.

B) **„Backward raytracing“**

   1. Bestimme Helligkeit mit einem normalen Beleuchtungsmodell
   2. Verfolge Sichtstrahlen vom auge ausgehend zurück, solange sie nur spiegelnde oder transparenter Flächen treffen.

   Bei Spiegelung kann man entweder den Strahl vor dem Auftreffen (Auge) oder nach dem Auftreffen (Objekt) an der Spiegelungseben spiegeln $\rightarrow$ aus dem geknickten Sichtstrahl wird eine Gerade.

11.1 Gesetzmäßigkeiten

11.1.1 Reflexion

$\alpha = \beta$: Einfallswinkel = Ausfallswinkel

11.1.2 Brechung

Übergang zwischen zwei Medien M_1 und M_2, die verschiedene optische Dichte D_1 bzw. D_2 haben.

$$\frac{\cos\alpha}{\cos\beta} = \underbrace{\frac{D_2}{D_1}}_{\text{fest für 2 gegebene Materialien}}$$

Die Brechung tritt fast nie in Reinform auf. Es tritt meist immer Reflexion gleichzeitig auf
Wenn $D_1 > D_2$ ist, dann kann in der Formel $\cos\beta > 1$ herauskommen.
$\rightarrow$ Dann wird der Strahl reflektiert (Totalreflexion)