

# Computer Graphics Zusammenfassung

Lucien Zürcher & Melvin Werthmüller

January 9, 2019

## Contents

<b>1 Farbe</b>	<b>3</b>	4.5 Regionen füllen . . . . .	8
1.1 Was ist Farbe? . . . . .	3	4.5.1 FloodFill . . . . .	8
1.2 Farbe eines Objektes . . . . .	3	4.6 Zeichnen von gefüllten Polygonen . . . . .	8
1.3 Licht besteht aus? . . . . .	3	4.6.1 Scanlinien Algorithmus . . . . .	8
1.4 Das Auge . . . . .	3	4.6.2 Zeichnen von Dreiecken . . . . .	9
1.5 Wie sehen wir Farbe? . . . . .	3	4.7 Anti-Aliasing . . . . .	9
1.6 Wahrnehmung . . . . .	3		
1.7 Farbsysteme . . . . .	3		
1.8 Additives Farbsystem . . . . .	4		
1.9 Subtraktives Farbsystem . . . . .	4		
1.10 Farben Konvertieren . . . . .	4		
1.10.1 RGB -> CMYK . . . . .	4		
1.10.2 CMYK -> RGB . . . . .	4		
1.10.3 RGB -> HSV . . . . .	4		
1.10.4 HSV -> RGB . . . . .	4		
1.11 Gamma Korrektur . . . . .	4		
1.12 Normfarbtafel / CIE Farbsystem . . . . .	5		
1.13 Helligkeitswahrnehmung . . . . .	5		
1.14 Nibs (Lichtdichte) . . . . .	5		
1.15 Mach bending . . . . .	5		
1.16 Farbtäuschung . . . . .	5		
1.17 HD,UHD,UK . . . . .	5		
1.18 Was ist HDR? . . . . .	5		
1.19 Begriffe . . . . .	5		
<b>2 Halbtontechnik</b>	<b>5</b>		
2.1 Verfahren der Halbtontechnik . . . . .	5		
2.2 Quantisierung . . . . .	5		
2.3 Dithering . . . . .	5		
2.3.1 Dithermatrizen . . . . .	5		
2.3.2 Dithering bei gleich bleibender Auflösung . . . . .	6		
2.3.3 Dispersed Dot Dithering . . . . .	6		
2.3.4 Error Diffusion . . . . .	6		
<b>3 Viewing</b>	<b>6</b>		
3.1 Planare geometrische Projektionen . . . . .	6		
3.1.1 parallele Projektionen . . . . .	6		
3.1.2 perspektivische Projektionen . . . . .	7		
3.1.3 perspektivische Projektion berechnen . . . . .	7		
3.2 Vertex Transformationen . . . . .	7		
<b>4 Scan Konvertierung</b>	<b>7</b>		
4.1 Linie Rastern . . . . .	7		
4.2 Mittelpunktschema . . . . .	7		
4.3 Kreis Rastern . . . . .	8		
4.4 Mittelpunktschema Kreis . . . . .	8		
<b>5 Visibilität</b>	<b>9</b>		
5.1 Backface Culling (WebGL) . . . . .	9		
5.2 Tiefensortierung (Painters Algorithms) . . . . .	9		
5.3 z-Buffer (WebGL) . . . . .	10		
5.4 Warnock Algorithmus . . . . .	10		
5.5 Objektraum & Bildraum . . . . .	10		
<b>6 Clipping</b>	<b>10</b>		
6.1 Verfahren . . . . .	10		
6.2 Clipping von Linien . . . . .	10		
6.2.1 Clipping von Linien nach Cohen-Sutherland . . . . .	10		
6.3 Clipping von Polygonen mit dem Algorithmus von Sutherland-Hodgeman . . . . .	11		
<b>7 Beleuchtung &amp; Schattierung</b>	<b>11</b>		
7.1 Energy . . . . .	11		
7.2 Beleuchtungsmodelle . . . . .	11		
7.2.1 Diffuse Reflektion (Lambert Modell) . . . . .	12		
7.2.2 Spiegelnde Reflektion (Phong Modell) . . . . .	12		
7.3 Scattering . . . . .	12		
7.4 Farbe . . . . .	12		
7.5 Lichtabschwächung . . . . .	12		
7.6 Lichtquellen . . . . .	12		
7.7 Schattierung . . . . .	12		
<b>8 Raytracing</b>	<b>13</b>		
<b>9 Rendering Equation(s)</b>	<b>13</b>		
9.1 BRDF (Bidirectional Reflectance Distribution Function) . . . . .	13		
<b>10 WebGL</b>	<b>13</b>		
10.1 OpenGL Merkmale . . . . .	13		
10.2 Grafikpipeline . . . . .	13		
10.3 Programmierbare Shaders . . . . .	13		
10.3.1 Vertex Processing / Vertex Shader . . . . .	13		
10.3.2 Fragment Processing / Fragment Shader . . . . .	13		
10.4 Datenfluss . . . . .	13		

10.5 Attribute und Uniform Variablen mit Shaders verbinden . . . . .	13
10.6 Attribut Variablen und Buffer definieren .	14
<b>11 Vektoren</b>	<b>14</b>
11.1 Addition . . . . .	14
11.2 Multiplikation mit Skalar . . . . .	14
11.3 Nullvektor . . . . .	14
11.4 Vektorinverses . . . . .	14
11.5 Vektoren Gleichheit . . . . .	14
11.6 Skalarprodukt . . . . .	14
11.7 Skalarprodukt im beliebigem Koordinatensystem . . . . .	14
11.8 Orthogonal . . . . .	15
11.9 Länge des Vektors . . . . .	15
11.10 Einheitsvektor . . . . .	15
11.11 Euklidische Distanz . . . . .	15
11.12 Gerade im 2/3D . . . . .	15
11.13 Hessische Normalform . . . . .	15
11.14 Hessische Normalform Ebene . . . . .	15
11.15 Achsenabschnitt . . . . .	15
11.16 Projektion eines Vektors . . . . .	15
11.17 Vektorprodukt / Kreuzprodukt . . . . .	15
11.18 Vektorprodukt Anwendung . . . . .	16
11.19 Matrixprodukt . . . . .	16
11.20 Spatprodukt . . . . .	16
11.21 Translation 2D . . . . .	16
11.22 Skalierung 2D . . . . .	16
11.23 Rotation 2D . . . . .	16
11.24 Vektor Rechenregeln . . . . .	16
11.25 Rechenregel Skalarprodukt . . . . .	16
11.26 Vektorprodukt Rechenregeln . . . . .	16
11.27 Spatprodukt Rechenregeln . . . . .	17
11.28 Begriffe . . . . .	17
<b>12 Projektive Geometrie</b>	<b>17</b>
12.1 Homogener Vektor . . . . .	17
12.2 Punkt auf Gerade . . . . .	17
12.3 Schnittpunkt Geraden . . . . .	17
12.4 Parallelle Geraden . . . . .	17
12.5 Unendlicher homogener Vektor . . . . .	17
12.6 Projektive Ebene (homogener Vektor) . .	17
12.7 Projektive Transformation . . . . .	17
12.8 Transformationen kombinieren . . . . .	17
12.9 Translation 2D . . . . .	18
12.10 Nullpunkt Rotation 2D . . . . .	18
12.11 Rotation um Punkt A . . . . .	18
12.12 Spiegelung mit Gerade durch Ursprung .	18
12.13 Spiegelung mit Gerade g . . . . .	18
12.14 Transformation des Koordinatensystems	18
12.15 Abstand Punkt zur Gerade . . . . .	18
12.16 Transformationen 2D . . . . .	18
<b>13 Transformation</b>	<b>18</b>
13.1 homogene Koordinaten . . . . .	18
13.2 Ebene im Raum . . . . .	18
13.3 Projektive Transformation . . . . .	19
13.4 Transformationen . . . . .	19
13.5 Translation . . . . .	19
13.6 Rotation . . . . .	19
13.7 Rotation um beliebige Achse . . . . .	19
13.8 Rotation um eine Achse durch den Ursprung . . . . .	19
13.9 Parallele Projektion . . . . .	20
13.10 Parallele Projektionsmatrix . . . . .	20
13.11 Perspektivische Projektion . . . . .	20
13.12 Perspektivische Projektionmatrix . . . .	20
13.13 Sichtvolumen Clipping . . . . .	20
<b>14 Curves</b>	<b>21</b>
14.1 Kurve in der Ebene . . . . .	21
14.2 Kurve im Raum . . . . .	21
14.3 Spirale entlang des Zylinders . . . . .	21
14.4 Splines Übersicht (Interpolation versus Approximation) . . . . .	21
14.5 Methode unbestimmte Koeffizienten (Interpolation) . . . . .	21
14.6 Lagrange Methode (Interpolation) . . . .	21
14.7 Lineare Bézier spline (Approximation) .	21
14.8 Quadric Bézier spline (Approximation) .	22
14.9 Cubic Bézier Spline (Approximation) .	22
14.10 Bernsteinpolynome . . . . .	22
14.11 Bézier-Kurven . . . . .	22
14.12 B-Spline-Kurven . . . . .	22
14.13 NURBS (Non Uniform Rational Basic Splines) . . . . .	22
<b>15 Flächen in Kartesische Koordinaten</b>	<b>22</b>
15.1 Freiformflächen . . . . .	23
<b>16 Appendix</b>	<b>23</b>
16.1 Radians . . . . .	23

## 1 Farbe

### 1.1 Was ist Farbe?

- **Physikalisch**, Lichtzusammensetzung, Elektromagnetischestrahlen
- **Physiologisch**, Wahrnehmung und Interpretation

Farbe besteht aus:

- Farbton/Farbe
- Farbstich/Sättigung
- Helligkeit

### 1.2 Farbe eines Objektes

Ein Objekt nimmt Farbe auf und strahlt Farbe ab. Die Farbe des Objektes ist definiert durch die abgestrahlte Farbe.

- Beleuchtung (Illumination)
- Reflektion (Reflection)
- Farbsignal (Color Signal)

### 1.3 Licht besteht aus?

Licht besitzt verschiedene Wellenlängen, Kombinationen dieser Frequenzen ergeben eine Farbe.

- Sichtbares Licht (380nm - 780nm)
- Infrarot (780nm+)
- Ultraviolet (-380nm)

$Inm = 10\text{Å}(\text{Ångström})$

$1\text{Å} = \phi\text{Atom}$

### 1.4 Das Auge

Das Auge besteht aus; **Iris** (Kreisring mit radialen Muskel und Lichteinschränken), **Linse** (Fokussieren), **Pupille** (Öffnung, durch Iris kontrolliert), **Photorezeptoren** (Nehmen das Licht wahr) und **Retina** (Farb- und Lichtaufnahme am Rand des Auges)

Die Retina besteht aus  $75-100 \cdot 10^6$  Stäbchen/rods (Lichtintensität) und  $6-7 \cdot 10^6$  Zäpfchen/cones (Farbe). Die Fovea ist der dichteste Platz.

### 1.5 Wie sehen wir Farbe?

Durch die 3 Arten von Zäpfchen:

Kurz (S)	Mittel (M)	Lang (L)
Blau	Grün	Rot
440nm	530nm	560nm
1	: 5	: 10

Weiss ist eine Kombination von Wellenlängen. Es gibt verschiedene Verteilung für Weiss.

## 1.6 Wahrnehmung

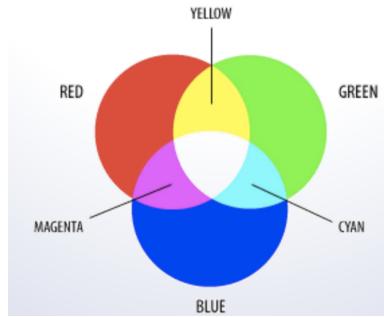
Grün 530nm wird am intensivsten wargenommen

Die Helligkeitswahrnehmung zwischen Stäbchen und Zäpfchen ist unterschiedlich

## 1.7 Farbsysteme

Nicht alle existierenden Farben (CIE) sind mit allen System darstellbar!

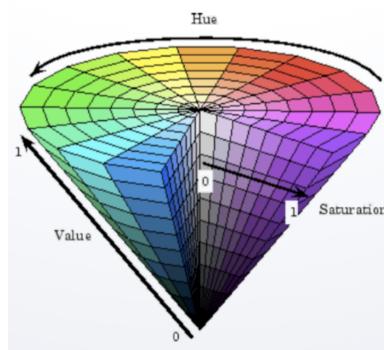
- **RGB** (Monitor, Spotlights, Pointilismus), additiv,  $C = (\text{Rot}, \text{Grün}, \text{Blau})$



- **CMY** (Drucken), subtraktiv,  $C = (\text{Cyan}, \text{Magenta}, \text{Yellow})$  Komplementärfarbe von RGB  $(C, M, Y) = (1, 1, 1) - (R, G, B)$

- **CMYK**, CMY Mit Schwarz erweitert,  $K = \min(\text{Cyan}, \text{Magenta}, \text{Yellow})$   $C = C - K, M = M - K, Y = Y - K$

- **HSV**, Farbton (Hue) / Reinheit, Sättigung (Saturation) / Intensität (Value)



- **YUV** (Alte Fernseher, Y= Helligkeit, UV = 1/4 Auflösung Farbkorrektur)  $Y = 0.229 * R + 0.587G + 0.114 * B$ ,  $U = 0.436(B - Y)/(1 - 0.114)$ ,  $V = 0.615(R - Y)/(1 - 0.299)$

- **CIE-Lab**, absolutes Farbsystem Achsen system mit Helligkeit (L) als Y-Achse und X/Z-Achse definieren Farbunterschiede
  - a: rot – grün Achse
  - b: gelb – blau Achse

## 1.8 Additives Farbsystem

Farben addieren

$(1,1,1) = \text{Weiss}, (0,0,0) = \text{Schwarz}$

## 1.9 Subtraktives Farbsystem

Farben absorbieren / filtern

$(0,0,0) = \text{Weiss}, (1,1,1) = \text{Schwarz}$

## 1.10 Farben Konvertieren

Zu Grau:  $I = 0.229 * R + 0.587G + 0.114 * B$

$$RGB \leftrightarrow CMY: x \begin{pmatrix} C \\ M \\ Y \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} - \begin{pmatrix} R \\ G \\ B \end{pmatrix}$$

### 1.10.1 RGB -> CMYK

divide by 255 to get range from 0-1

$$R' = R/255$$

$$G' = G/255$$

$$B' = B/255$$

$$K = 1 - \max(R', G', B')$$

$$C = (1 - R' - K)/(1 - K)$$

$$M = (1 - G' - K)/(1 - K)$$

$$Y = (1 - B' - K)/(1 - K)$$

### 1.10.2 CMYK -> RGB

$$R = 255 \cdot (1 - C) \cdot (1 - K)$$

$$G = 255 \cdot (1 - M) \cdot (1 - K)$$

$$B = 255 \cdot (1 - Y) \cdot (1 - K)$$

### 1.10.3 RGB -> HSV

divide by 255 to get range from 0-1

$$R' = R/255$$

$$G' = G/255$$

$$B' = B/255$$

$$C_{\max} = \max(R', G', B')$$

$$C_{\min} = \min(R', G', B')$$

$$\Delta = C_{\max} - C_{\min}$$

#### Hue calculation

$$H = \begin{cases} 0^\circ & \text{falls } \Delta = 0 \\ 60^\circ \cdot (\frac{G' - B'}{\Delta} \bmod 6) & \text{falls } C_{\max} = R' \\ 60^\circ \cdot (\frac{B' - R'}{\Delta} + 2) & \text{falls } C_{\max} = G' \\ 60^\circ \cdot (\frac{R' - G'}{\Delta} + 4) & \text{falls } C_{\max} = B' \end{cases}$$

#### Saturation calculation

$$S = \begin{cases} 0 & \text{falls } C_{\max} = 0 \\ \frac{\Delta}{C_{\max}} & C_{\max} \neq 0 \end{cases}$$

#### Value calculation

$$V = C_{\max}$$

### 1.10.4 HSV -> RGB

$$0 \leq H \leq 360, 0 \leq S \leq 1, 0 \leq V \leq 1$$

$$C = V \cdot S$$

$$X = C \cdot (1 - |(H/60^\circ) \bmod 2 - 1|)$$

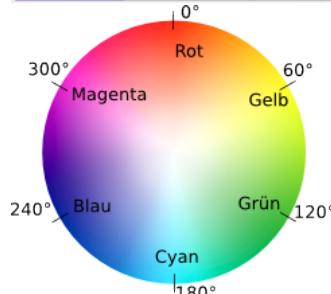
$$m = V - C$$

$$(R', G', B') = \begin{cases} (C, X, 0) & \text{falls } 0^\circ \leq H \leq 60^\circ \\ (X, C, 0) & \text{falls } 60^\circ \leq H \leq 120^\circ \\ (0, C, X) & \text{falls } 120^\circ \leq H \leq 180^\circ \\ (0, X, C) & \text{falls } 180^\circ \leq H \leq 240^\circ \\ (X, 0, C) & \text{falls } 240^\circ \leq H \leq 300^\circ \\ (C, 0, X) & \text{falls } 300^\circ \leq H \leq 360^\circ \end{cases}$$

$$(R, G, B) = ((R' + m) \cdot 255, (G' + m) \cdot 255, (B' + m) \cdot 255)$$

HSV -> RGB:

Farbe	H	S	V	R	G	B
Schwarz	-	-	0 %	0 %	0 %	0 %
Rot	0°	100 %	100 %	100 %	0 %	0 %
Gelb	60°	100 %	100 %	100 %	100 %	0 %
Braun	24,3°	75 %	36,1 %	36 %	20 %	9 %
Weiß	-	0 %	100 %	100 %	100 %	100 %
Grün	120°	100 %	100 %	0 %	100 %	0 %
Dunkelgrün	120°	100 %	50 %	0 %	50 %	0 %
Cyan	180°	100 %	100 %	0 %	100 %	100 %
Blau	240°	100 %	100 %	0 %	0 %	100 %
Magenta	300°	100 %	100 %	100 %	0 %	100 %
Orange	30°	100 %	100 %	100 %	50 %	0 %
Violett	270°	100 %	100 %	50 %	0 %	100 %



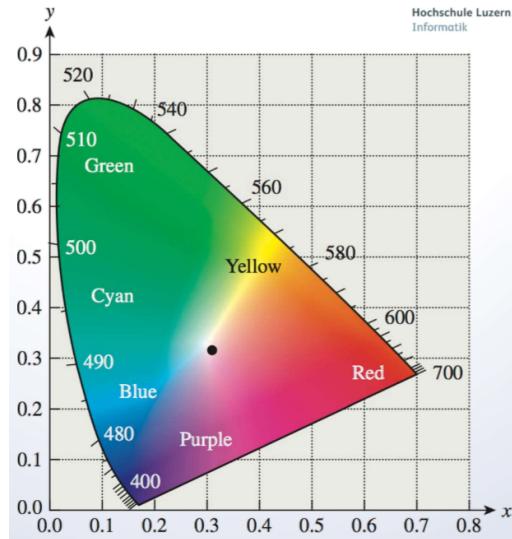
### 1.11 Gamma Korrektur

Erreichen von gleichmässiger Verteilung der Helligkeit / Kontrast. Das Empfinden der Helligkeit ist nicht linear.

Korrektur der Helligkeit des Bildes mit Gamma Wert. Wichtig für Bildschirme einstellen. Beim einstellen der Monitore Grauwerte mit echten Werten vergleichen (Gamma Test Pattern). 50% schwarz und 50% weiss einer Fläche (z.B. Punkte) sollte gleich sein wie 50% grau

## 1.12 Normfarbtafel / CIE Farbsystem

- kann alle Farben darstellen
- Spektralfarben = Farbe am Rand mit Wellenlänge
- Farben zwischen zweie Farben mischbar
- Komplementärfarben gehen durch Weiss
- Achtung: Keine Spektralfarbe am Rande zwischen UV und Infrarot



## 1.13 Helligkeitswahrnehmung

Helligkeit wird logarithmisch wahrgenommen, Webers Law

$$\frac{\Delta I}{I} = C$$

$$\log(I + \Delta I) - \log(I) = Const$$

Helligkeitsunterschied im dunkeln nehmen wir stärker wahr

## 1.14 Nibs (Lichtdichte)

Gibt Helligkeitsdichte für Auge an. 10nits werden stärker wargenommen denn 100nits. Heisst, weniger Licht wird stärker wargenommen.

## 1.15 Mach bending

Optische Illusion, bei zwei verschiedenen Grauwerten nebeneinander unterschieden sich diese vermeidlich stärker.

## 1.16 Farbtäuschung

Farbe wird abhängig durch Umgebung anders wahrgenommen (Dunkler, Heller). Optische Illusionen

## 1.17 HD,UHD,UK

Unterscheiden sich durch Pixelauflösung.

## 1.18 Was ist HDR?

**High Dynamic Range**, speichert zusätzlichen Wert um Helligkeitsunterschiede besser unterscheiden zu können (RGB-Pixelwerte proportional zum Licht). Detailreichere dunkel und helle Spots, weniger Verlust durch Farben mit weniger Helligkeitsunterschiede.

## 1.19 Begriffe

Natürliches Licht	Gemisch aus verschiedenen Lichtwellen / Frequenzen
Spektralfarben	reine Farbfrequenz; Alle Farben am Rand des CIE-Farbsystems
Spektrum	Alle Frequenzen und deren Verteilung
Spektralverteilung	Charakterisiert die Farbe, definiert durch Frequenzen (Bsp. Verschiedenes Weiss)
Komplementärfarben	Addieren ergeben Grau, gegenüberliegende Farben im CIE-Farbsystem durch Weiss

## 2 Halbtontechnik

### 2.1 Verfahren der Halbtontechnik

Da nur Schwarz und Weiss gedruckt werden kann, werden die verschiedenen Stufen durch Intensitätsstufen dargestellt. Dafür gibt es drei Verfahren:

- Quantisierung
- Dithering
- Error Diffusion

### 2.2 Quantisierung

Höhere Auflösung auf niedrige Auflösung durch Runden der Pixelfarbwerte auf verfügbare Farben.

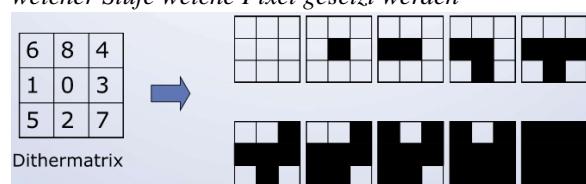
Bsp. 16Bit -> 8Bit (Runden der Werte)

### 2.3 Dithering

Wenn der Drucker eine grösse Auflösung besitzt, jedoch weniger Farbstufen kann Dithering verfahren verwendet werden. -> Farben mit höherer Auflösung durch kleinere Punkte simulieren.

#### 2.3.1 Dithermatrizen

Kann als Matrix dargestellt werden. Matrix gibt an, auf welcher Stufe welche Pixel gesetzt werden



Es gibt zwei Regeln; **Gesetzter Pixel bleibt gesetzt und Strukturen in der Ditheringmatrix vermeiden.** Es soll möglichst ein Kreis approximiert werden.

### 2.3.2 Dithering bei gleich bleibender Auflösung

Handhabung, wenn die Bildgröße/Auflösung gleich bleibt

- Clustered dot dithering: Mittelwert von  $n \times n$  Region mit Ditheringmatrix ersetzen.
- Dispersed Dot Dithering

### 2.3.3 Dispersed Dot Dithering

Bayer Matrizen können hierfür verwendet werden, wodurch die Methode Bayer Dithering genannt wird.

$2 \times 2$  Bayer Matrix

0	2
3	1

$4 \times 4$  Bayer Matrix

0	8	2	10
12	4	14	6
3	11	1	9
15	7	13	5

$$k = \frac{W_{max}}{n \cdot n + 1}$$

$W_{max}$ : Anzahl Werte des Pixels (256 bei 8Bit)

$n$ : Größe der Matrix ( $2 \times 2 \Rightarrow n = 2$ )

$k$ : Faktor für Umrechnung

$$I_{new} = \frac{I_{old}}{k}$$

Für jeden Pixel den neuen Wert ausrechnen, danach mit Bayermatrix den Wert vergleichen. Pixel setzen wenn  $I(x,y)_{new} > D_{ij}$

$$i = x \text{ modulo } n$$

$$j = y \text{ modulo } n$$

### 2.3.4 Error Diffusion

Anstatt Kreise, Punkte verschiedener Dichte anordnen. Das Bild wird dabei sequenziell durchlaufen; links -> rechts, oben -> unten

Error Diffusion verteilt den Fehler auf die umliegenden Pixel (zu hell -> dunkler machen / zu dunkel -> heller machen)

	X	7/16
1/16	5/16	3/16

Gewichtungsmatrix

Beispiel:

X	191	140	113
244	221	105	100

$$191 - 255 = -64, \text{ da Pixel Schwarz (255), Fehler: } -64$$

X	X	140 + (7/16 * -64)	113
244 + (1/16 * -64)	221 + (5/16 * -64)	105 + (3/16 * -64)	100

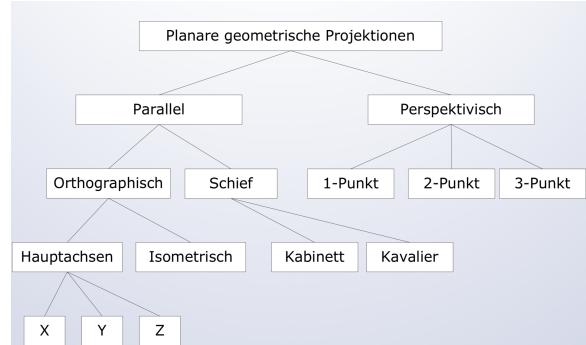
Wenn Wert > 128 = 255, ansonsten Wert <= 128 = 0

## 3 Viewing

von 3D in 2D abbilden

### 3.1 Planare geometrische Projektionen

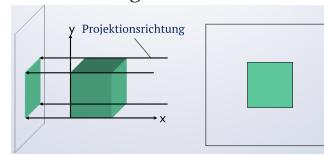
Bild auf eine Ebene projizieren



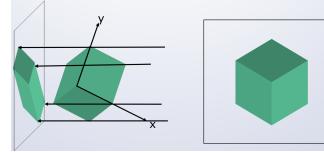
#### 3.1.1 parallele Projektionen

- Geraden und Parallelen bleiben erhalten
- konstante Verkürzung in eine Richtung
- keine Winkeltreue

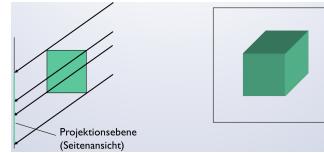
**Orthographische Projektion** von einer Seite / rechtwinklig



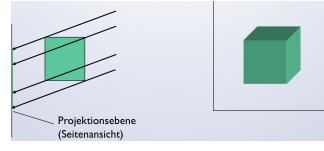
**Isometrische Projektion** Orthographische auf die Ebene (1,1,1)



**Kavaliersprojektion** Parallelprojektion mit 45 Grad. Linien rechtwinklig zur Ebene haben natürliche Länge

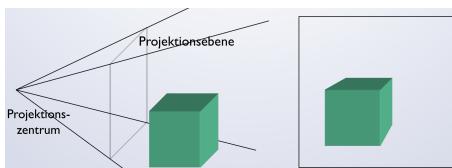


**Kabinettprojektion** Parallelprojektion mit 63.4 Grad. Linien rechtwinklig zur Ebene haben halbe Länge



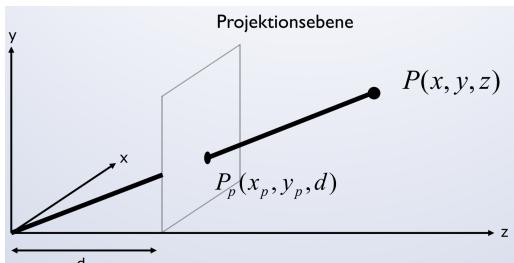
### 3.1.2 perspektivische Projektionen

- Simuliert Kamera oder Auge
- Spezifiziert durch Projektionszentrum und Projektionsebene
- Geraden bleiben erhalten
- Parallelen schneiden sich in einem Punkt (Fluchtpunkt)
- Objektgrösse nimmt proportional zum Abstand vom Projektionszentrum ab

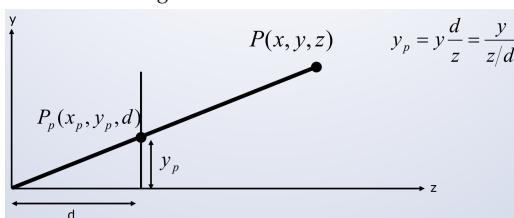


### 3.1.3 perspektivische Projektion berechnen

Projektionsebene parallel zur x-y Ebene bei  $z=d$  wenn  
Projektionszentrum =  $(0, 0, 0)$



Ansicht entlang x-Achse



Projektionsmatrix in homogene Koordinaten

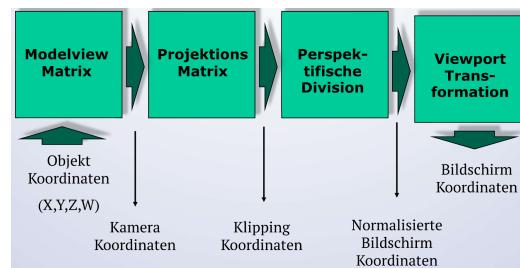
$$M_{per} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{1}{d} & 0 \end{bmatrix}$$

Für  $d \leftarrow \infty$  entsteht eine Parallelprojektionsmatrix

$$M'_{orth} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

### 3.2 Vertex Transformationen

Matrix im Vertexshader benötigt:



```
attribute vec3 aVertexPosition;
uniform mat4 uModelViewMatrix;
uniform mat4 uProjectionMatrix;
void main() {
    vec4 position = vec4(aVertexPosition, 1.0);
    gl_Position =
        uProjectionMatrix *
        uModelViewMatrix *
        position;
}
```

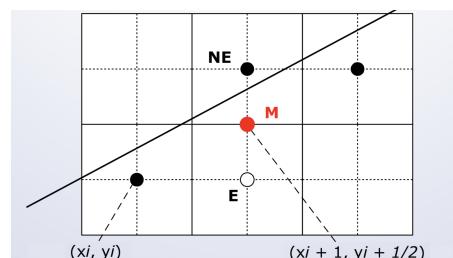
## 4 Scan Konvertierung

### 4.1 Linie Rastern

Eine Linie von  $(x_0, y_0)$  nach  $(x_1, y_1)$  rastern. Da in Pixel konvertiert werden muss. Methoden:

- Brute Force; über jeden Pixel und runden
- Brute Force inkrementell (DDA = Digital Differential Analyzer);  $y_{i+1} = m * x_{i+1} + B = y_i + m$ , Nachteil Gleitkommazahlen und Runden (aufwändig)
- Mittelpunktschema; Nächsten Punkt wird Kalkuliert durch if/ else

### 4.2 Mittelpunktschema



Ist eine inkrementelle methode zum Rastern. Mittelpunkt wird betrachtet um nächsten Punkt zu finden. ( $M = (x_i + 1, y_i + \frac{1}{2})$ ,  $E = (x_i + 1, y_i)$ ,  $NE = (x_i + 1, y_i + 1)$ )

Initialisierung:

$$\begin{aligned} D_x &= x_1 - x_0; \\ D_y &= y_1 - y_0; \\ D_E &= 2 * D_y; \\ D_{NE} &= 2 * (D_y - D_x); \\ d &= 2 * D_y - D_x; \end{aligned}$$

$y = y_0$

Wegen 1/2 alles mal 2, damit gerade Zahlen

Für jeden nächsten d Wert, wenn  $d \leq 0$ , dann  $d = d + D_{NE}$  ansonsten  $d = d + D_{NE}$  und  $y$  inkrementieren. Jeweils den Punkt  $P(x, y)$  zeichnen.  $x$  jedesmal inkrementieren.

Berechnung:

```
WritePixel(x0, y0)
for x = x0+1 to x1 do
    if d <= 0 then
        d = d + DE;
    else
        d = d + DNE;
        y = y + 1;
    end
    WritePixel(x, y);
end
```

### 4.3 Kreis Rastern

Selbe Methode wie bei den Linien kann für Kreise angewendet werden.

### 4.4 Mittelpunktschema Kreis

Funktion für Kreis:  $F(x, y) = x^2 + y^2 - R^2$

```
x = 0
y = R
d = 1 - R
D_E = 2 * x + 3
D_NE = 2 * (x - y)d + 5
```

Für jeden nächsten d Wert wiederholen bis  $y > x$ , wenn  $d < 0$ , dann  $d = d + D_E$  ansonsten  $d = d + D_{NE}$  und  $y$  dekrementieren. Jeweils den Punkt  $P(x, y)$  zeichnen.  $x$  jedesmal inkrementieren.

```
WritePixel(x, y);
while y > x do
    if d < 0 then
        d = d + 2*x + 3;
    else
        d = d + 2*(x-y) + 5;
        y = y - 1;
    end
    x = x + 1;
    WritePixel(x, y);
end
```

### 4.5 Regionen füllen

Entweder durch 4-er oder 8-er Zusammenhang definiert



4-er Zusammenhang



8-er Zusammenhang

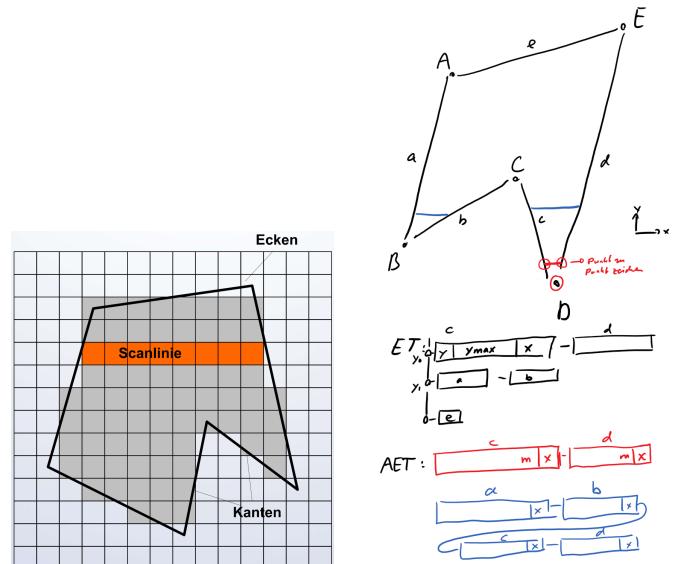
#### 4.5.1 FloodFill

Füllen durch selben abfrage ob selbe Farbe (Photoshop Zauberstab)

```
proc FloodFill(int x, int y,
Color oldColor, Color newColor)
    if ReadPixel(x,y) == oldColor then
        WritePixel(x,y,newColor);
        FloodFill(x, y-1, oldColor, newColor);
        FloodFill(x-1,y, oldColor, newColor);
        FloodFill(x, y+1, oldColor, newColor);
        FloodFill(x+1, y, oldColor, newColor);
    end
end
```

### 4.6 Zeichnen von gefüllten Polygonen

#### 4.6.1 Scanlinien Algorithmus



- Zeilenweise färben (Spans entlang der y Koordinate -> Scanlinien)
- Kantentabelle (edgetable ET)  
sortierte Kanten (Linien) des Polygon nach min y (innerhalb davon nach max x)
- Einträge werden gespeichert durch  $[x|l|m|y_{min}|y_{max}]$
- Tabelle aktiver Kanten (AET)  
Momentane Kanten für Scanlinie (sortiert nach x asc) zum Zeichnen vom einer Kante zur nächsten (immer zwei)

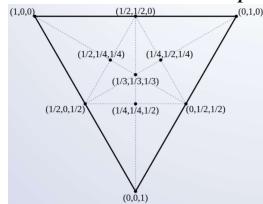
```

Erzeuge ET (Kantentabelle)
Initialisiere AET = empty
(Aktive Kanten)
y=0
repeat
  Addiere und entferne alle Kanten
    ET(y) zu AET
  Sortiere AET nach x
  Zeichne Spans
  y=y+1
  Entferne Kanten mit ymax = y aus AET
  Aktualisiere den x Wert aller
    Kanten in AET
until AET == empty and ET == empty

```

## 4.6.2 Zeichnen von Dreiecken

*Brute Force: Alle Pixel probieren, ob im Dreieck*

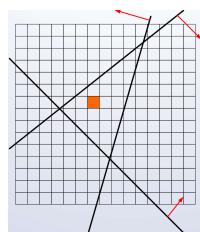


Baryzentrische Koordinaten

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

$$\alpha + \beta + \gamma = 1$$

Wenn Werte zwischen 0 und 1, dann ist Punkt P innerhalb von ABC. Wert einer Koordinate entspricht dem Verhältnis der Fläche eines Teildreiecks mit Ecke P zur Fläche von ABC



2. Bruteforce Option

Unterteilung in 3 Geraden.  
Zeichnen wenn auf der richtigen  
Seite aller Geraden

- Warnock Algorithmus
- Raycasting / Raytracing

## 5.1 Backface Culling (WebGL)

- Rückseiten nicht zeichnen
- Test durch Normalvektor (Normalvektor der Fläche Richtung Kamera, dann zeichnen)

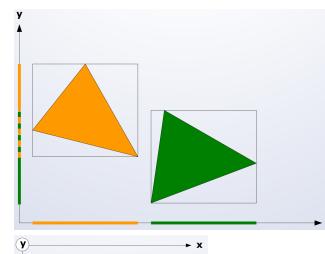
## 5.2 Tiefensortierung (Painters Algorithms)

*Zeichnen der Polygone in einer Reihenfolge zuletzt gezeichnete überdecken die ersten Sortierung*

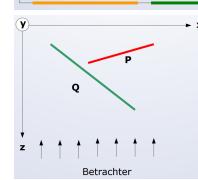
Sortierung:

1. Sortierung nach minimaler y-Koordinate (in Kamera Koordinaten)
2. wenn in z-überlagern -> weitere Bedingungen beachten
3. Falls keine Bedingung erfüllt -> Polygone vertauschen und nochmals Prüfen

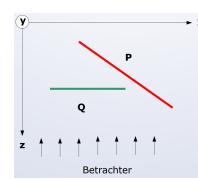
*Wenn [P,Q] Reihenfolge, austauschen wenn KEINE der folgenden Bedingungen zutrifft:*



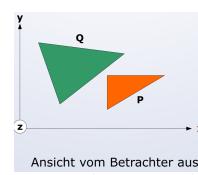
1. Überlagern sich die x-Ausdehnungen nicht?  
(vollständig nebeneinander)



2. Überlagern sich die y-Ausdehnungen nicht?  
(vollständig hintereinander)



3. Wird das hintere Objekt P vom Vorderen Q komplett überdeckt?



4. Liegt Q ganz auf der Betrachterseite von P? (Polygon P ist in der Sortierung vor Q)

Ansicht vom Betrachter aus

5. Überlappen sich die Polygone nicht auf der Projektion in die xy Ebene?

*Bei ungültiger Reihenfolge müssen die Polygone geschnitten werden*

Eigenschaften:

- + auch für transparente Objekte möglich
- + einfach für Spezialfälle (2.5D)
- ineffizient für viele Objekte O(n<sup>2</sup>)
- nicht Hardware unterstützt

## 5 Visibilität

*Algorithmen für verdeckte Linien und Flächen*

- Backface Culling
- Tiefensortierung
- Z-Buffer

### 5.3 z-Buffer (WebGL)

- Pro Pixel Tiefe zusätzlich zur Farbe speichern
- Beim Zeichnen prüfen, ob näher liegt

```
// Initialisierung
for y = 0 to YMAX do
    for x = 0 to XMAX do
        color[x,y] = Background
        depth[x,y] = MAXDEPTH
    end
end

// Polygone zeichnen
for alle Polygone q do
    for alle pixel p(x,y) in q do
        z = CalculateDepth(q,x,y);
        if z > depth[x,y] then
            depth[x,y] = z;
            color[x,y] = Color of q
        end
    end
end
```

Tiefe berechnung aus Ebenegleichung:

$$z = \frac{-D - Ax - By}{C}$$

Inkrementelle Berechnung entlang einer Scanlinie:

$$Z_{neu} = \frac{-D - A(X_{alt} + 1) - By}{C} = Z_{alt} - \frac{A}{C}$$

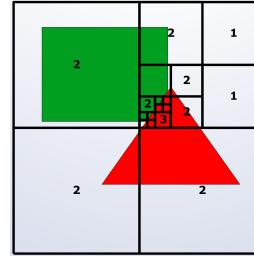
Eigenschaften:

- + Hardware unterstützt
- + Polygone können in beliebiger Reihenfolge gezeichnet werden
- + Berechenzeit  $O(n)$ , häufig konstant ab einer gewissen Anzahl Polygone
- Rundungsprobleme
- Gleiche z-Werte problematisch (Überlagerungsfehler)
- grosser Speicherbedarf

### 5.4 Warnock Algorithmus

Rekursive Unterteilung des Bildbereichs. Falls "einfach" zu zeichnen -> Zeichnen, sonst unterteilt in 4

Bereich ist zeichenbar wenn:



- Der Bereich kein Polygon enthält
- Der Bereich nur ein Polygon enthält
- Der Bereich ein Polygon enthält, das am nächsten liegt und den Bereich vollständig ausfüllt
- Der Bereich nur aus einem Pixel besteht

### 5.5 Objektraum & Bildraum

**Objektraum** definiert Raum aller Objekte (für alle Objekte)

**Bildraum** definiert Raum aller Pixel (für alle Pixel in Bild)

## 6 Clipping

Objekte außerhalb des Viewingbereichs nicht darstellen

- Vermeiden von unnötiger Rasterisierung
- Ressourcen schonen

### 6.1 Verfahren

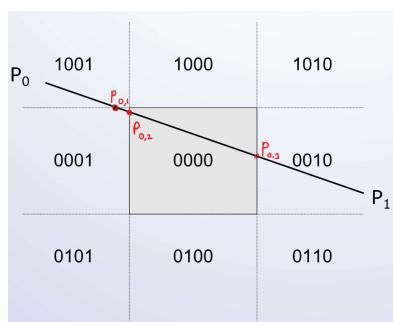
- Scissoring  
alle Pixel berechnen & darstellen wenn im Fenster
- Temporärer Buffer  
zeichnen in einem temporären Buffer, welcher dann Kopiert wird
- Analytische Berechnung derjenigen Teile, die im Innern des Fensters liegen

### 6.2 Clipping von Linien

*Brute Force Algorithmus: Alle Schnittpunkte berechnen, falls mind. ein Endpunkt außerhalb Kappungsrechteck*

#### 6.2.1 Clipping von Linien nach Cohen-Sutherland

1. Die ganze Ebene wird in Bereiche unterteilt
2. Durch Vergleichen der Bereiche in denen die Endpunkte liegen, wird entschieden ob die Linie einfach akzeptiert oder abgelehnt werden kann
3. Ist dies nicht der Fall, wird die Linie in zwei Teile geteilt, wovon eines abgelehnt wird, mit dem anderen wird zu Schritt 2 zurückgekehrt

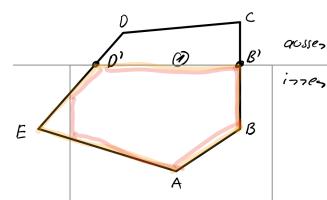


code( $P_0$ )	1001
code( $P_1$ )	0010
AND	0000
code( $P_{0,2}$ )	0000
code( $P_1$ )	0010
AND	0000
code( $P_{0,3}$ )	0000
code( $P_1$ )	0000
&	0000

Resultat Linie:  
code( $P_{0,2}$ ) 0000  
code( $P_{0,3}$ ) 0000

Polygon 1:  
ABCDE  
AB → B  
BC → B'  
CD → -  
DE → D'E  
EA → A

Polygon 2:  
BB'D'EA

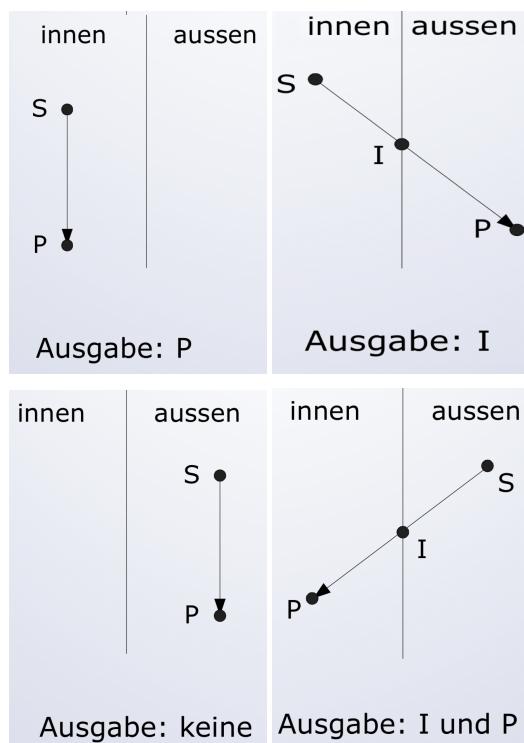


Bitweise AND

- Alles 0 → Schnittpunkte berechnen gemäss Bit, dann neue Teillinie bis beide Punkte innerhalb (0000)
- falls eine 1 → ablehnen

### 6.3 Clipping von Polygonen mit dem Algorithmus von Sutherland-Hodgeman

durch das Clipping entstehen evt. neue Formen mit mehr oder weniger Ecken!



Vorgehensweise: Verbindung zwischen zwei Punkten wird gemäss Regeln untersucht. 0-2 Eckpunkte des "neuen" Polygon als Ausgabe

Beispiel nur an einer Kante:

## 7 Beleuchtung & Schattierung

Eigenschaften eines Objektes:

- Spiegelung
- Glanz / Matt
- Transparenz
- Glatt / Rau (Oberflächenstruktur)
- Farbe
- Struktur / Textur
- Brechnung des Lichts
- Anisotrop (Richtungsabhängig)

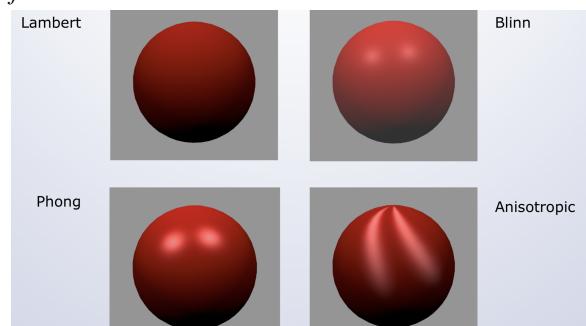
### 7.1 Energy

Die Energy einer beleuchteten Fläche ist proportional zum Cosinus zwischen Lichtrichtung und Flächennormale

Fällt das Licht schräg auf die Fläche, muss eine grössere Fläche mit Licht beleuchtet werden → Fläche erscheint dünker

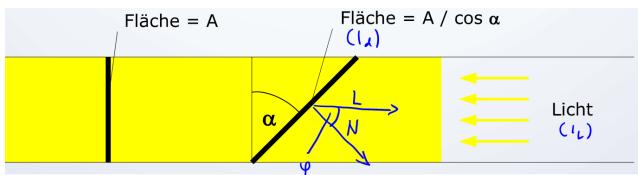
### 7.2 Beleuchtungsmodelle

Reale Beleuchtung ist sehr aufwändig, daher wird einfacheres Modell verwendet.



### 7.2.1 Diffuse Reflektion (Lambert Modell)

- Gleichmässige Abstrahlung des Lichts in alle Richtungen
- Eigenschaften eines matten, nicht glänzenden Materials
- Schiefe Fläche = senkrechte Fläche /  $\cos(\alpha)$
- reflektierte Intensität = Intensität der Lichtquelle \* Reflektierungsfaktor \*  $\cos(\alpha)$   
 $\cos(\alpha) = \text{Flächennormale(vektor)} \cdot \text{Richtung zur Lichtquelle(vektor)}$
- Reflektierungsfaktor = Farbe des Materials



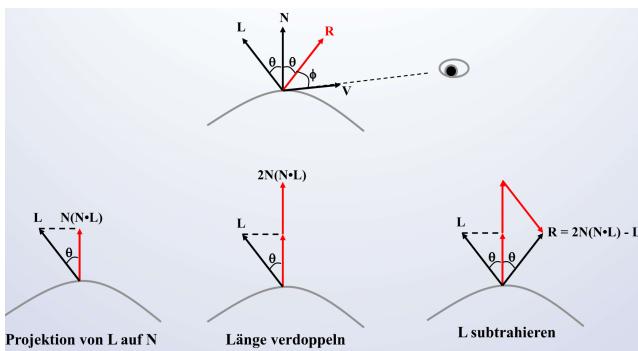
$\vec{N}$ : normalisiert Flächennormale  
 $\vec{L}$ : Richtung zur Lichtquelle  
 $I_d$ : reflektierte Intensität  
 $I_L$ : Intensität der Lichtquelle  
 $k_d$ : Reflektionsfaktor

### 7.2.2 Spiegelnde Reflektion (Phong Modell)

Intensität Spiegelung = Intensität der Lichtquelle \* Reflektierungsfaktor \*  $\cos^n$ (Winkel zwischen Beleuchtung & Reflektionsrichtung)

n = Streuung des Lichts (hohes n -> Streuung nimmt schnell ab)

Reflektierungsfaktor = Farbe der Reflektion



$$I_d = I_L \cdot k_d \cdot \cos(\phi)^n_s$$

$$\cos(\phi) = \vec{V} \cdot \vec{L}$$

$$R = 2\vec{N}(\vec{N} \cdot \vec{L}) - \vec{L}$$

$\vec{N}$ : normalisiert Flächennormale  
 $\vec{L}$ : Richtung zur Lichtquelle normalisiert

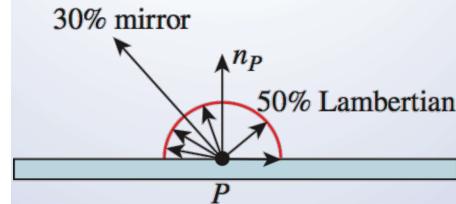
$I_d$ : reflektierte Intensität

$I_L$ : Intensität der Lichtquelle

$k_d$ : Reflektionsfaktor

### 7.3 Scattering

Licht wird teils auch reflektiert (gespiegelt)



### 7.4 Farbe

Reflektionskontanten ( $k_d, k_s$ ) hängen von der Wellenlänge ab. Diese werden beim einfachen Beleuchtungsmodell mit Konstanten für jeweils Rot Grün Blau definiert. Bsp:  $k_s = (0.35, 0.35, 0.35)$ ,  $k_d = (0.6, 0, 0)$ .

$k_d$  : definiert die Farbe des Objektes

$k_s$  : definiert die Farbe der Lichtquelle

### 7.5 Lichtabschwächung

Lichtabschwächung nimmt mit dem Quadrat des Abstandes ab. Jedoch zu schnell zu dunkel, daher beliebtes Modell:

$$f_{Att} = \frac{1}{(c_1 + c_2 d + c_3 d^2)}$$

d: Distanz  $c_n$ : Konstanten

### 7.6 Lichtquellen

- Ambiente Lichtquelle (Licht von überall)
- Direktionale Lichtquelle
- Punkt Lichtquelle
- Spot Lichtquelle (alles ausserhalb schwarz)
- Verteilte Lichtquelle (Licht mit Ausdehnung)

### 7.7 Schattierung

- Konstanter Schattierung pro Fläche eine Beleuchtung (Farbe) berechnen
- Gouraud Schattierung Beleuchtung an den Ecken berechnen dazwischen interpolieren VertexShader
- Phong Schattierung Beleuchtungsberechnung pro Pixel FragmentShader

## 8 Raytracing

"schönere" Bilder erzeugen -> Photorealistisch

- Strahl durch Pixel -> Schnitt mit nächsten Objekt aus Betrachtung
- Rekursives Raytracing für zusätzliche Effekte  
*Spiegelung, Transparenz mir Reflektion, Schatten*
- Aufwendige Technik *Um sinnlose Schnitte zu vermeiden -> Objekte zusammenfassen (Bounding Box) und damit den Schnittpunkt berechnen Schnittpunkt mit "Grid" (auf dem Bild) und dann mit Objekten in dieser Zelle*

## 9 Rendering Equation(s)

### 9.1 BRDF (Bidirectional Reflectance Distribution Function)

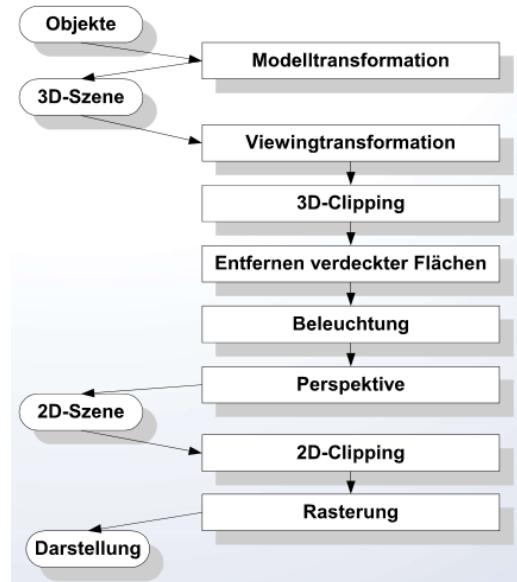
- Anstatt Diffuse (Gleichmässig) oder Specular (Abnehmend) Reflexion an einem Punkt
- Radiosity: Beleuchtete Fläche als neue Lichtquelle (indirekt)
- Fläche wird in "Patches" unterteilt
- "Patches" mit konstanter Farbe / Lichtintensität
- "Patches" als neue Lichtquelle für andere "Patches"
- Berechnungsvarianten -> *Progressive Radiosity: Berechnung und frühe Darstellung einer approximativen Lösung -> Hierarchical Radiosity: Erhöhung der Anzahl Patches wo es notwendig ist*

## 10 WebGL

### 10.1 OpenGL Merkmale

- Low Level Graphics API
- Verschiedene Plattformen
- 1.0/2.0 Fixe Funktionspipeline
- Vorlage für WebGL

## 10.2 Grafikpipeline



### 10.3 Programmierbare Shaders

Shaders werden für die Berechnung der zu zeichnenden Objekte verwendet. Das Programm wird direkt auf der Grafikkarte ausgeführt.

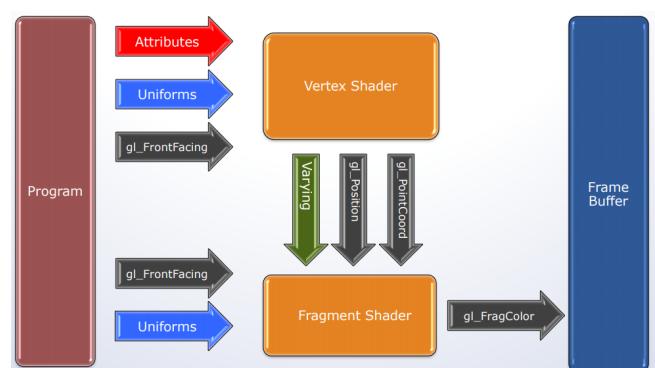
#### 10.3.1 Vertex Processing / Vertex Shader

Berechnen der **Positionen** der Vertexe (Punkte) und Werte für den folgenden Fragmentschader.

#### 10.3.2 Fragment Processing / Fragment Shader

Berechnet die **Farbe** der einzelnen Pixel.

## 10.4 Datenfluss



### 10.5 Attribute und Uniform Variablen mit Shaders verbinden

#### 1. Attribute

```

ctx.aVertexPositionId =
gl.getAttribLocation(ctx.shaderProgram,
  "aVertexPosition")
  
```

## 2. Uniform

```
ctx.uColorId =
gl.getUniformLocation(ctx.shaderProgram,
"uColor"))
```

## 10.6 Attribut Variablen und Buffer definieren

### Erzeugen

#### 1. Buffer erzeugen

```
var buffer = gl.createBuffer()
```

#### 2. Array Buffer auf Buffer setzen

```
gl.bindBuffer(gl.ARRAY_BUFFER,
buffer)
```

#### 3. Daten füllen

```
glBufferData(gl.ARRAY_BUFFER,
new Float32Array(vertices),
gl.STATIC_DRAW)
```

### Zeichnen

#### 1. Buffer binden

#### 2. Attribut und/oder uniform setzen

```
gl.vertexAttribPointer(index,
size, type, normalized, stride,
offset)
z.B. gl.vertexAttribPointer(ctx.aVertexPositionIndex,
2, gl.FLOAT, false, 0, 0)
```

#### 3. Attribut als Array setzen

```
gl.enableVertexAttribArray(index)
```

#### 4. Zeichnen

```
gl.drawArrays(mode, first, count)
```

## 11 Vektoren

- Skalarprodukt
- Matrixprodukt
- × Kreuzprodukt

### 11.1 Addition

$$\vec{a} + \vec{b} = \begin{bmatrix} a_1 \\ a_2 \\ a_n \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_n \end{bmatrix} = \begin{bmatrix} a_1 + b_1 \\ a_2 + b_2 \\ a_n + b_n \end{bmatrix}$$

### 11.2 Multiplikation mit Skalar

$$\lambda \vec{a} = \lambda \begin{bmatrix} a_1 \\ a_2 \\ a_n \end{bmatrix} = \begin{bmatrix} \lambda a_1 \\ \lambda a_2 \\ \lambda a_n \end{bmatrix}$$

$\lambda \in \text{Skalar}$

### 11.3 Nullvektor

$$\vec{0} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

### 11.4 Vektorinverses

$$-\vec{a} = -\begin{bmatrix} a_1 \\ a_2 \\ a_n \end{bmatrix} = \begin{bmatrix} -a_1 \\ -a_2 \\ -a_n \end{bmatrix}$$

Vektor mit negativen Komponenten

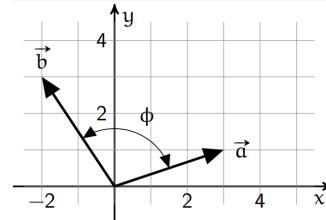
### 11.5 Vektoren Gleichheit

$$\vec{a} = \begin{bmatrix} 3 \\ 2 \end{bmatrix} = \vec{b}$$

Vektoren sind gleich, wenn Komponenten gleich

### 11.6 Skalarprodukt

$$\vec{a} \bullet \vec{b} = \begin{bmatrix} a_1 \\ a_2 \\ a_n \end{bmatrix} \bullet \begin{bmatrix} b_1 \\ b_2 \\ b_n \end{bmatrix} = a_1 b_1 + a_2 b_2 + \dots + a_n b_n$$



$$\vec{a} \bullet \vec{b} = |\vec{a}| \cdot |\vec{b}| \cdot \cos \phi$$

$$|\vec{a}| = \sqrt{a_1^2 + a_2^2 + \dots + a_n^2}$$

$$|\vec{b}| = \sqrt{b_1^2 + b_2^2 + \dots + b_n^2}$$

$$\cos \phi = \frac{\vec{a} \bullet \vec{b}}{|\vec{a}| \cdot |\vec{b}|}$$

### 11.7 Skalarprodukt im beliebigem Koordinatensystem

$$\vec{a} = a_1 \vec{e}_1 + a_2 \vec{e}_2 + a_3 \vec{e}_3 = [a_1 a_2 a_3]^T$$

$$\vec{b} = b_1 \vec{e}_1 + b_2 \vec{e}_2 + b_3 \vec{e}_3 = [b_1 b_2 b_3]^T$$

$$\vec{a} \bullet \vec{b} = [a_1 a_2 a_3] \begin{bmatrix} g_{11} & g_{12} & g_{13} \\ g_{21} & g_{22} & g_{23} \\ g_{31} & g_{32} & g_{33} \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = \mathbf{a}^T \mathbf{G} \mathbf{b}$$

Matrix  $\mathbf{G}$  wird **metrisch Tensor** genannt

## 11.8 Orthogonal

$$\vec{e}_x \bullet \vec{e}_y = 0$$

$$\vec{a} \bullet \vec{b} = 0 \Leftrightarrow \vec{a} \perp \vec{b}$$

Senkrecht zueinander, wenn Skalarprodukt zweier Einheitsvektoren 0 ergibt.

## 11.9 Länge des Vektors

$$|\vec{a}| = \sqrt{\vec{a} \bullet \vec{a}} = \sqrt{a_1^2 + a_2^2 + \dots + a_n^2}$$

## 11.10 Einheitsvektor

$$e_v = \frac{1}{\|v\|} \bullet v = \frac{1}{\sqrt{v \cdot v}} \bullet v$$

$$(i = e_1, j = e_2, k = e_3)$$

$$\vec{e}_x = [1, 0, 0]^T$$

$$\vec{e}_y = [0, 1, 0]^T$$

$$\vec{e}_z = [0, 0, 1]^T$$

## 11.11 Euklidische Distanz

$$\bar{AB} = \sqrt{(b_1 - a_1)^2 + (b_2 - a_2)^2 + \dots + (b_n - a_n)^2}$$

## 11.12 Gerade im 2/3D

- Punkt-Punktf orm mit Vektoren 2/3D

$$\vec{r} = \vec{r}_1 + t(\vec{r}_1 - \vec{r}_0), t \in \mathbb{R}$$

$\vec{r}_1$ : Punkt,  $\vec{r}_2$ : Punkt

- Punkt-Richtungsform mit Vektoren 2/3D

$$\vec{r} = \vec{r}_0 + t\vec{r}_1, t \in \mathbb{R}$$

$\vec{r}_0$ : Punkt,  $\vec{r}_1$ : Richtungsvektor

- Achsenabschnitt-Steigungsform

$$y = mx + b$$

b: Achsenabschnitt, m: Steigung

- Punkt-Richtungsform

$$(y - y_0) = m(x - x_0)$$

$(x_0, y_0)$ : Punkt, m: Steigung

- Allgemeine Geradengleichung

$$ax + by + c = 0$$

a, b, c  $\in \mathbb{R}$

Aus Punkten A(x1, y1) und B(x2, y2)

$$\frac{y_2 - y_1}{x_2 - x_1} = \frac{y - y_1}{x - x_1}$$

## 11.13 Hessische Normalform

Vektorielle Schreibweise der Hessischen Normalform

$$\vec{n} \bullet (\vec{x} - \vec{x}_0) = 0$$

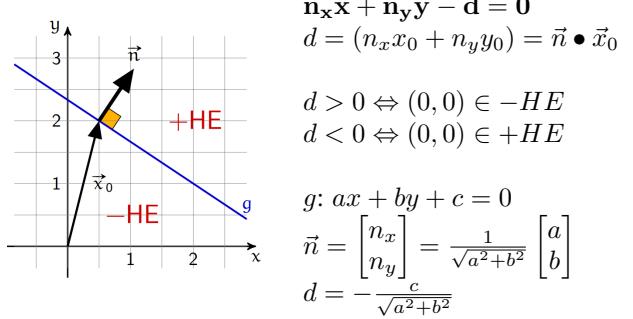
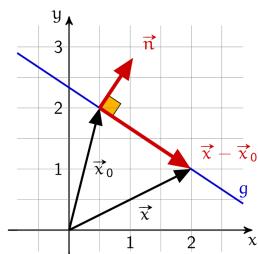
da  $\vec{n} \perp (\vec{x} - \vec{x}_0)$

$$\Rightarrow n_x(x - x_0) + n_y(y - y_0) = n_x x + n_y y - (n_x x_0 + n_y y_0)$$

$$\text{Abstand vom Ursprung: } d = (n_x x_0 + n_y y_0) = \vec{n} \bullet \vec{x}_0$$

$\vec{n}$  muss normalisiert sein:

$$|\vec{n}| = 1 \Rightarrow \frac{1}{\sqrt{n_x^2 + n_y^2}} \bullet \vec{n}$$



## 11.14 Hessische Normalform Ebene

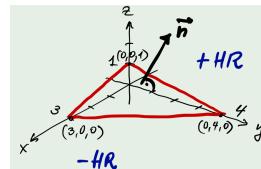
$$\epsilon : ax + by + cz + d = 0$$

$$n_x x + n_y y + n_z z - D = 0; \text{ HNF der Ebene } \epsilon \in \mathbb{R}^3$$

$$\vec{n} = \begin{bmatrix} n_x \\ n_y \\ n_z \end{bmatrix} = \frac{1}{\sqrt{a^2 + b^2 + c^2}} \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

$$D = -\frac{d}{\sqrt{a^2 + b^2 + c^2}}$$

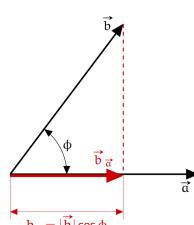
## 11.15 Achsenabschnitt



Gegeben sind 3 Punkte  $p_x = x, p_y = y, p_z = z$  die ergeben eine Ebenengleichung:

$$\frac{x}{p_x} + \frac{y}{p_y} + \frac{z}{p_z} - 1 = 0$$

## 11.16 Projektion eines Vektors



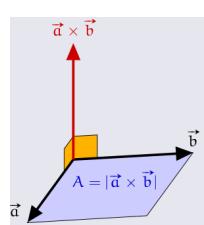
$\vec{b}$  Richtung  $\vec{a}$ :

$$\vec{b}_a = \frac{\vec{a} \bullet \vec{b}}{|\vec{a}|^2} \vec{a}$$

$b_a$  mal Einheitsvektor  $\vec{a}$

$$\vec{b}_a = b_a \frac{1}{|\vec{a}|} \vec{a} = |\vec{a}| |\vec{b}| \cos \phi \frac{1}{|\vec{a}|} \vec{a} = \frac{\vec{a} \bullet \vec{b}}{|\vec{a}|^2} \vec{a}$$

## 11.17 Vektorprodukt / Kreuzprodukt

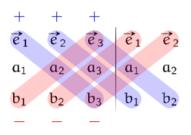


$\vec{a} \times \vec{b}$  steht senkrecht auf beiden Vektoren

$\vec{a}, \vec{b}$  und  $\vec{a} \times \vec{b}$  sind ein Rechtssystem

$\vec{a} \times \vec{b}$  entspricht der Fläche des aufgespannten Parallelogramms (A):

$$A = \sqrt{(a_2 b_3 - a_3 b_2)^2 + (a_3 b_1 - a_1 b_3)^2 + (a_1 b_2 - a_2 b_1)^2}$$



$$\vec{a} \times \vec{b} = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} \times \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} a_2 b_3 - a_3 b_2 \\ a_3 b_1 - a_1 b_3 \\ a_1 b_2 - a_2 b_1 \end{bmatrix}$$

$$= \begin{bmatrix} \vec{e}_1 & \vec{e}_2 & \vec{e}_3 \\ a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \end{bmatrix}$$

$$= (a_2 b_3 - a_3 b_2) \vec{e}_1 + (a_3 b_1 - a_1 b_3) \vec{e}_2 + (a_1 b_2 - a_2 b_1) \vec{e}_3$$

Regel von Sarrus

## 11.18 Vektorprodukt Anwendung

- Lorentz-Kraft**  $\vec{F} = q(\vec{v} \times \vec{B})$   
 $\vec{v}$ : Geschwindigkeit,  $B$ : Magnetfeld,  $q$ : Ladung
- Geschwindigkeit**  $\vec{v} = q(\vec{w} \times \vec{x})$   
 $\vec{x}$ : Punkt,  $w$ : Winkelgeschwindigkeit,  $\vec{w}$ : Drehachse
- Drehmoment**  $\vec{M} = \vec{r} \times \vec{F}$   
 $\vec{F}$ : Kraft,  $\vec{r}$ : Punkt / Koordinatenursprung
- Normalvektor**  $\vec{n} = \vec{a} \times \vec{b}$   
 $\vec{a}$  und  $\vec{b}$  liegen auf der Ebene.
- Kollinearität** kollinear (d.h. linear abhängig) wenn Vektorprodukt verschwindet

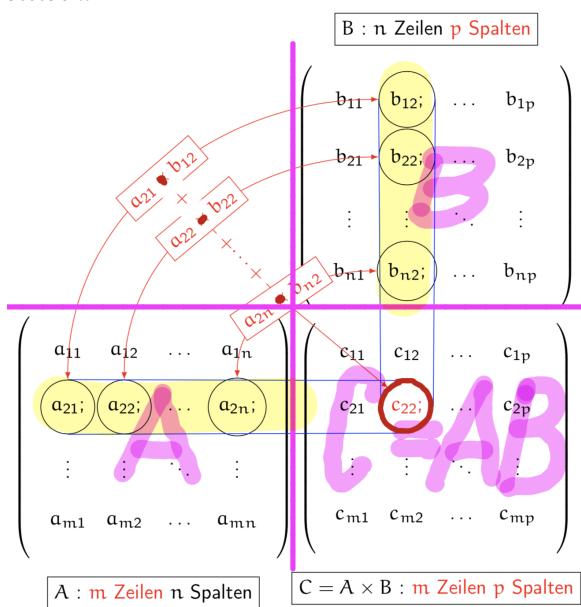
## 11.19 Matrixprodukt

falksche Schema

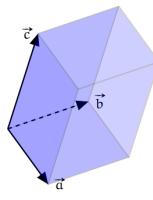
$$\vec{a} \cdot \vec{b} = \begin{bmatrix} a_1 & a_3 \\ a_2 & a_4 \end{bmatrix} \cdot \begin{bmatrix} b_1 & b_3 \\ b_2 & b_4 \end{bmatrix} =$$

$$\begin{bmatrix} a_1 b_1 + a_3 b_2 & a_1 b_3 + a_3 b_4 \\ a_2 b_1 + a_4 b_2 & a_2 b_3 + a_4 b_4 \end{bmatrix}$$

In Worten für die erste Zeile:  $a_1$  mal die erste Zeile +  $a_3$  mal die zweite Zeile etc.. Wobei die Spalten unabhängig bleiben.



## 11.20 Svatprodukt

Svatprodukt  $[\vec{a}, \vec{b}, \vec{c}]$ Ist der Skalar der Vektoren  $\vec{a}, \vec{b}, \vec{c}$ 

$$[\vec{a}, \vec{b}, \vec{c}] = \vec{a} \bullet (\vec{b} \times \vec{c})$$

Svatprodukt entspricht Volumen wenn in einem Rechtsystem, dann:  $V_{Spat} = |[\vec{a}, \vec{b}, \vec{c}]|$

$$|[\vec{a}, \vec{b}, \vec{c}]| = a_1 b_2 c_3 + a_2 b_3 c_1 + a_3 b_1 c_2 - a_3 b_2 c_1 - a_1 b_3 c_2 - a_2 b_1 c_3$$

Komplanar (linear abhängig) wenn  $[\vec{a}, \vec{b}, \vec{c}] = 0$

## 11.21 Translation 2D

$$\vec{x}' = \vec{x} + \vec{t} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} t_1 \\ t_2 \end{bmatrix} = \begin{bmatrix} x_1 + t_1 \\ x_2 + t_2 \end{bmatrix}$$

## 11.22 Skalierung 2D

$$\vec{x}' = \begin{bmatrix} \vec{s}_x x \\ \vec{s}_y y \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

## 11.23 Rotation 2D

$$\vec{x}' = \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos(\phi) & -\sin(\phi) \\ \sin(\phi) & \cos(\phi) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \mathbf{R} \vec{x}$$

Inverse Matrix:  $\mathbf{R}^{-1} = \mathbf{R}^T$ 

## 11.24 Vektor Rechenregeln

$\vec{a} + \vec{b} = \vec{b} + \vec{a}$	Kommutativgesetz
$\vec{a} + (\vec{b} + \vec{c}) = (\vec{a} + \vec{b}) + \vec{c}$	Assoziativgesetz
$\vec{a} + \vec{0} = \vec{a}$	Existenz Neutralelement $\vec{0}$
$\vec{a} + (-\vec{a}) = \vec{0}$	Existenz Inverses $-\vec{a}$
$\lambda(\vec{a} + \vec{b}) = \lambda\vec{a} + \lambda\vec{b}$	
$(\lambda + \mu)\vec{a} = \lambda\vec{a} + \mu\vec{a}$	
$(\lambda\mu)\vec{a} = \lambda(\mu\vec{a}) = \mu(\lambda\vec{a})$	
$1\vec{a} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \vec{a} = \vec{a}$	

## 11.25 Rechenregel Skalarprodukt

$$\vec{a} \bullet \vec{b} = \vec{b} \bullet \vec{a}$$

$$\vec{a} \bullet (\vec{b} + \vec{c}) = \vec{a} \bullet \vec{b} + \vec{a} \bullet \vec{c}$$

$$\lambda(\vec{a} \bullet \vec{b}) = (\lambda\vec{a}) \bullet \vec{b} = \vec{a} \bullet (\lambda\vec{b})$$

## 11.26 Vektorprodukt Rechenregeln

$\vec{a} \times \vec{b} = -\vec{b} \times \vec{a}$	Anti-Kommutativgesetz
$\vec{a} \times (\vec{b} + \vec{c}) = \vec{a} \times \vec{b} + \vec{a} \times \vec{c}$	Distributivgesetz
$\lambda(\vec{a} \times \vec{b}) = (\lambda\vec{a}) \times \vec{b} = \vec{a} \times (\lambda\vec{b})$	

### 11.27 Spatprodukt Rechenregeln

$[\vec{a}, \vec{b}, \vec{c}] = -[\vec{b}, \vec{a}, \vec{c}]$	zwei Vektoren vertauschen entspricht Vorzeichenwechsel
$[\vec{a}, \vec{b}, \vec{c}] = [\vec{c}, \vec{a}, \vec{b}]$	Zyklisches Vertauschen keine Änderung
$[\lambda\vec{a}, \mu\vec{b}, \nu\vec{c}] = \lambda\mu\nu[\vec{a}, \vec{b}, \vec{c}]$	Multiplikation
$[\vec{a} + \vec{b}, \vec{c}, \vec{d}] = [\vec{a}, \vec{c}, \vec{d}] + [\vec{b}, \vec{c}, \vec{d}]$	Addition

### 11.28 Begriffe

<b>Ortsvektor</b>	Vom Ursprung zum Punkt
<b>Richtungsvektor</b>	Eine Richtung im Raum
<b>Einheitsvektor</b>	Eine Einheit in eine beliebige Richtung
<b>Linearkombination</b> <i>kollinear</i>	Ein Vektor, der ein Vielfaches eines Einheitsvektors ist. $\vec{c} = \lambda\vec{a} + \mu\vec{b}$
<b>Linear Unabhängig</b> <i>komplanar</i>	Vektoren sind unabhängig wenn $\lambda_1\vec{a}_1 + \lambda_2\vec{a}_2 + \dots + \lambda_n\vec{a}_n = \vec{0}$ $\Leftrightarrow \lambda_1 = \lambda_2 = \dots = \lambda_n = 0$
<b>Skalar</b> <b>Rechtssystem</b>	Ist eine reelle oder komplexe Zahl Koordinatensystem aufgebaut wie die rechte Hand wobei; der Zeigfinger X-Achse ( $\vec{e}_x$ ), Mittelfinger Y-Achse ( $\vec{e}_y$ ) und Daumen Z-Achse ( $\vec{e}_z$ )

## 12 Projektive Geometrie

### 12.1 Homogener Vektor

$$\vec{r} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \text{ Homogener 2D Vektor}$$

$$(x, y) = \left( \frac{x_1}{x_3}, \frac{x_2}{x_3} \right)$$

### 12.2 Punkt auf Gerade

$$ax + by + c = 0 \Leftrightarrow \vec{g} \bullet \vec{r} = 0$$

$$\vec{r} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \vec{g} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

$A(x, y)$  (homogenisiert  $\vec{r}$ ) liegt dann auf Geraden  $\vec{g}$

### 12.3 Schnittpunkt Geraden

$$\vec{r} = \vec{g}_1 \times \vec{g}_2$$

$$\vec{g}_1 = \begin{bmatrix} a_1 \\ b_1 \\ c_1 \end{bmatrix}, \vec{g}_2 = \begin{bmatrix} a_2 \\ b_2 \\ c_2 \end{bmatrix}$$

$\vec{r}$  ist der homogene Schnittpunkt  $[x_1, x_2, x_3]$

$x_3 = 0$ , dann sind die Geraden parallel, und kein Schnittpunkt möglich (division durch 0)

### 12.4 Parallelle Geraden

$$\vec{r} = \vec{g}_1 \times \vec{g}_2 = (c_1 + c_2) \begin{bmatrix} b_1 \\ -a_1 \\ 0 \end{bmatrix}$$

$$(a_1, b_1) = (a_2, b_2), \text{ dann sind die Geraden parallel}$$

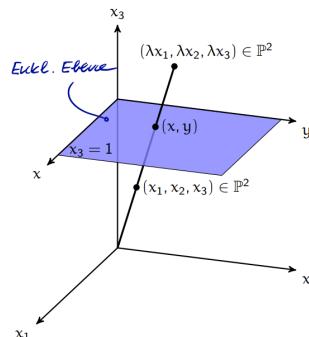
### 12.5 Unendlicher homogener Vektor

$$\vec{r} = \begin{bmatrix} x \\ y \\ 0 \end{bmatrix}, \vec{g}_\infty = \begin{bmatrix} 0 \\ 0 \\ c \end{bmatrix}$$

alle idealen, uneigentlichen oder unendlich fernen Punkte  $\vec{r}$  liegen auf der Geraden  $\vec{g}_\infty$

$$\vec{g}_\infty \bullet \vec{r} = \vec{g}_\infty = \begin{bmatrix} 0 \\ 0 \\ c \end{bmatrix} \bullet \begin{bmatrix} x \\ y \\ 0 \end{bmatrix} = 0$$

### 12.6 Projektive Ebene (homogener Vektor)



Ein Punkt auf der euklidischen Ebene entspricht dem Wert eines dehomogenisierten Punktes  $(x, y)$ .

$x_3 = 1$  ergibt die euklidische Ebene.

$\lambda(x_1, x_2, x_3)$  sind Punkte auf einer Geraden die den Punkt  $(\frac{x_1\lambda}{x_3\lambda}, \frac{x_2\lambda}{x_3\lambda}) = (x, y)$  definieren.

### 12.7 Projektive Transformation

Abbildungen  $h : \mathbb{P}^2 \mapsto \mathbb{P}^2$  mit Eigenschaften:

- $h$  ist eindeutig (bijektiv) und daher umkehrbar
- $h$  Transformationen sind geradentreu (geraden auf geraden abbilden)
- **Homogene Matrix** ist bis auf eine Konstante bestimmt ( $k\mathbf{H} = \mathbf{H}; k > 0$ )

$$\vec{r}' = h(\vec{r}) = \mathbf{H}\vec{r}, \mathbf{H} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}$$

### 12.8 Transformationen kombinieren

$$\vec{r}' = h(\vec{r}) = (h_2 \circ h_1)(\vec{r}) = h_2(h_1(\vec{r}))$$

$$\mathbf{H} = \mathbf{H}_2 \mathbf{H}_1, \vec{r}' = \mathbf{H}\vec{r} = \mathbf{H}_2 \cdot \mathbf{H}_1 \vec{r}$$

$$h_1 : \mathbb{P}^2 \mapsto \mathbb{P}^2, h_2 : \mathbb{P}^2 \mapsto \mathbb{P}^2$$

$h = h_2 \circ h_1$  entspricht erst  $h_1$  dann  $h_2$

## 12.9 Translation 2D

$$\vec{r}' = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \vec{r} = \mathbf{T}\vec{r}$$

Verschiebung durch  $\vec{t} = \begin{bmatrix} t_x \\ t_y \end{bmatrix}$ ,  $\mathbf{T}^{-1}$  entspricht  $-\vec{t}$  in  $\mathbf{T}$

## 12.10 Nullpunkt Rotation 2D

$$\vec{r}' = \begin{bmatrix} \cos(\phi) & -\sin(\phi) & 0 \\ \sin(\phi) & \cos(\phi) & 0 \\ 0 & 0 & 1 \end{bmatrix} = \mathbf{R}\vec{r}$$

Rotation mit  $\phi$ ,  $\mathbf{R}^{-1}$  entspricht sin vertauschen

## 12.11 Rotation um Punkt A

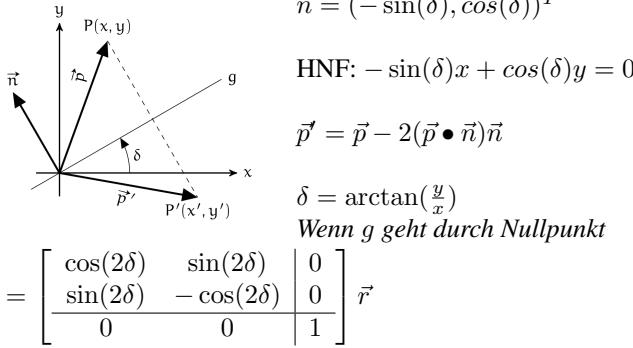
Punkt:  $A(t_x, t_y)$

1. Translation A zum Nullpunkt verschiebt ( $\mathbf{T}$ )
2. Nullpunkt Rotation 2D mit Winkel  $\Phi$
3. Translation A zurück ( $\mathbf{T}^{-1}$ )

$$\mathbf{R} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos(\Phi) & -\sin(\Phi) & 0 \\ \sin(\Phi) & \cos(\Phi) & 0 \\ 0 & 0 & 1 \end{bmatrix} = \mathbf{T}^{-1} \mathbf{R}_0 \mathbf{T}$$

## 12.12 Spiegelung mit Gerade durch Ursprung

$$\vec{n} = (-\sin(\delta), \cos(\delta))^T$$



## 12.13 Spiegelung mit Gerade $g$

1. gerade ins Zentrum Transformieren ( $\mathbf{T}$  errechnen)
2. Spiegelung mit Gerade durch Ursprung ( $\mathbf{S}$ )
3. zurück Transformieren ( $\mathbf{T}^{-1}$ )

$$\mathbf{M} = \mathbf{T}^{-1} \mathbf{S} \mathbf{T}$$

## 12.14 Transformation des Koordinatensystems

$$\begin{bmatrix} \cos(-\Phi) & -\sin(-\Phi) & 0 \\ \sin(-\Phi) & \cos(-\Phi) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Rotation des Koordinatensystems um  $\Phi$

Bei einer Transformation des Koordinatensystems handelt es sich um eine Inverse Matrix der normalen Transformation

## 12.15 Abstand Punkt zur Gerade

$$d = ax + by + c$$

$d$ : Distanz,  $P(x, y)$ : Punkt,  $g : ax + by + c = 0$ :

## 12.16 Transformationen 2D

$$t = \begin{bmatrix} x \\ y \end{bmatrix}, 0^T = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \mathbf{RMC} = 2 \times 2 \text{ Matrix}$$

Euklidisch (starre Bewegung)

$$D = \begin{bmatrix} \mathbf{R} & t \\ 0^T & 1 \end{bmatrix}$$

Abstand zwischen zwei Punkten, alle Winkel ( $R^{-1} = R^T$ )

Ähnlichkeit

$$S = \begin{bmatrix} k \cdot \mathbf{M} & t \\ 0^T & 1 \end{bmatrix}$$

Winkel zwischen zwei Punkten, alle Winkel

Affin

$$A = \begin{bmatrix} \mathbf{C} & t \\ 0^T & 1 \end{bmatrix}$$

Parallelität, Verhältnis zwischen Flächeninhalt

Allgemein

$$\mathbf{H} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}$$

Geraden bleiben Geraden

## 13 Transformation

### 13.1 homogene Koordinaten

jeder Punkt  $P(x, y, z)$  des Raumes  $\mathbb{R}^3$  besitzt eine 4-komponenten Vektor  $\vec{r}$

$$\vec{r} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}, x = \frac{x_1}{x_4}, y = \frac{x_2}{x_4}, z = \frac{x_3}{x_4}$$

$$(x, y, z) = \left( \frac{x_1}{x_4}, \frac{x_2}{x_4}, \frac{x_3}{x_4} \right)$$

### 13.2 Ebene im Raum

Ebene  $\epsilon$  im Raum  $\mathbb{R}^3$

$\epsilon : ax + by + cz + d = 0$  Hessische Normalform

$$\vec{w} = \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}, \text{Punkt: } \vec{r} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Ebenengleichung:

$$\vec{w} \bullet \vec{r} = w^T \cdot r = ax + by + cz + d = 0$$

### 13.3 Prokektive Transformation

Die homogene Matrix  $\mathbf{H}$  ist nur bis auf einen konstanten Faktor bestimmt, heisst, alle Vielfachen von  $\mathbf{H}$  sind auch gültig

$\eta : \mathbb{P}^3 \mapsto \mathbb{P}^3$  stellt eine **projektiven Transformation** dar

$$\eta(r) = \mathbf{H} \cdot r = \begin{bmatrix} h_{11} & h_{12} & h_{13} & h_{14} \\ h_{21} & h_{22} & h_{23} & h_{24} \\ h_{31} & h_{32} & h_{33} & h_{34} \\ h_{41} & h_{42} & h_{43} & h_{44} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

### 13.4 Transformationen

$$t = \begin{bmatrix} x \\ y \\ z \end{bmatrix}, 0^T = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \text{RMC} = 3 \times 3 \text{ Matrix}$$

**Euklidisch** (starre Bewegung)

$$D = \begin{bmatrix} \mathbf{R} & t \\ 0^T & 1 \end{bmatrix}$$

Abstand zwischen zwei Punkten, alle Winkel ( $R^{-1} = R^T$ )

**Ähnlichkeit**

$$S = \begin{bmatrix} k \cdot \mathbf{M} & t \\ 0^T & 1 \end{bmatrix}$$

Winkel zwischen zwei Punkten, alle Winkel

**Affin**

$$A = \begin{bmatrix} \mathbf{C} & t \\ 0^T & 1 \end{bmatrix}$$

Parallelität, Verhältnis zwischen Volumeninhalt

**Allgemein**

$$\mathbf{H} = \begin{bmatrix} h_{11} & h_{12} & h_{13} & h_{14} \\ h_{21} & h_{22} & h_{23} & h_{24} \\ h_{31} & h_{32} & h_{33} & h_{34} \\ h_{41} & h_{42} & h_{43} & h_{44} \end{bmatrix}$$

Geraden bleiben Geraden

### 13.5 Translation

$$\mathbf{T}(\vec{t}) = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

### 13.6 Rotation

$$\mathbf{R}_z = \begin{bmatrix} \cos(\Phi_z) & -\sin(\Phi_z) & 0 & 0 \\ \sin(\Phi_z) & \cos(\Phi_z) & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{R}_y = \begin{bmatrix} \cos(\Phi_y) & 0 & \sin(\Phi_y) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\Phi_y) & 1 & \cos(\Phi_y) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{R}_z = \left[ \begin{array}{ccc|c} 1 & 0 & 0 & 0 \\ 0 & \cos(\Phi_x) & -\sin(\Phi_x) & 0 \\ 0 & \sin(\Phi_x) & \cos(\Phi_x) & 0 \\ \hline 0 & 0 & 0 & 1 \end{array} \right]$$

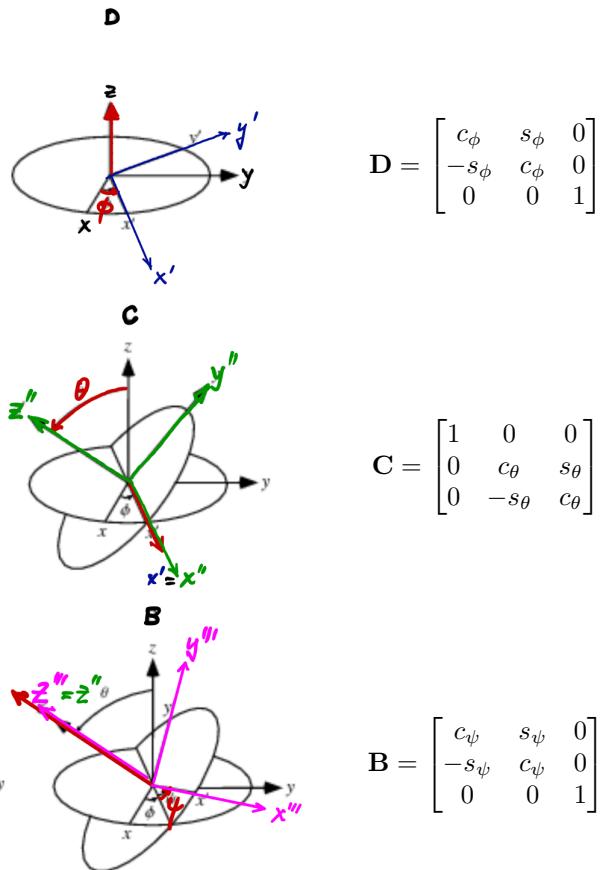
Bei Rotation um selbe Achse gilt Kommutativgesetz ( $\mathbf{R}_z(\Phi_{z,1} + \Phi_{z,2}) = \mathbf{R}_z(\Phi_{z,1})\mathbf{R}_z(\Phi_{z,2}) = \mathbf{R}_z(\Phi_{z,2})\mathbf{R}_z(\Phi_{z,1})$ )

Inverse entspricht  $\mathbf{R}^{-1}(\Phi) = \mathbf{R}(-\Phi)$ , wobei  $\cos(-\Phi) = \cos(\Phi)$

### 13.7 Rotation um beliebige Achse

- 1) Rotation um  $\phi$  um z-Achse (Matrix D)
- 2) Rotation um den Winkel  $\theta \in [0, \pi]$  (um frühere X-Achse) (Matrix C)
- 3) Rotation um den gegebenen Winkel  $\psi$  (Matrix B)

$$c_\alpha = \cos \alpha, s_\alpha = \sin \alpha, \alpha \in \phi, \theta, \psi$$



$$\mathbf{M} = \mathbf{D}^{-1}\mathbf{C}^{-1}\mathbf{B}\mathbf{C}\mathbf{D}$$

Bei Rotation um eine Gerade, 1. Transformation **D** & **C** Matrix (mit Winkel der Gerade), dann eigentliche Transformation mit gegebenem Winkel, dann zurück-transformiere **C**<sup>-1</sup> & **D**<sup>-1</sup>

### 13.8 Rotation um eine Achse durch den Ursprung

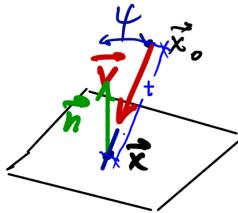
$$\text{Rotation um einen Einheitsvektor } \vec{a} = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

$$\mathbf{Q} = (\cos \theta)I + (1 - \cos \theta) \begin{bmatrix} a_1^2 & a_1 a_2 & a_1 a_3 \\ a_1 a_2 & a_2^2 & a_2 a_3 \\ a_1 a_3 & a_2 a_3 & a_3^2 \end{bmatrix} - \sin \theta \begin{bmatrix} 0 & a_3 & -a_2 \\ -a_3 & 0 & a_1 \\ a_2 & -a_1 & 0 \end{bmatrix}$$

$$\begin{bmatrix} -d & 0 & 0 & 0 \\ 0 & -d & 0 & 0 \\ 0 & 0 & -d & 0 \\ a & b & c & 0 \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \\ z_0 \\ 1 \end{bmatrix}$$

### 13.9 Parallelle Projektion

Projektion auf Ebene  $\epsilon : ax + by + cz + d = 0$   
 Die Ebene ist definiert durch Normalvektor  $\vec{n} = [a, b, c]^T$  (Ebenen Normalenvektor)  
 Projektionsrichtung definiert durch Normalenvektor  $\vec{v} = [v_x, v_y, v_z]^T$  (Projektionsrichtung)



$$\vec{x} = \vec{x}_0 + t\vec{v} = \vec{x}_0 + t^*\vec{v}$$

$$t = \frac{ax_0 + by_0 + cz_0 + d}{av_x + bv_y + cv_z}$$

$$t^* = \frac{ax_0 + by_0 + cz_0 + d}{\cos(\psi)}$$

$$av_x + bv_y + cv_z = \vec{v} \bullet \vec{n} = |\vec{v}| |\vec{n}| \cos(\psi) = \cos(\psi)$$

$$\vec{x} = \vec{x}_0 + t\vec{v}, \text{ komponentenweise } \begin{bmatrix} x = x_0 + tv_x \\ y = y_0 + tv_y \\ z = z_0 + tv_z \end{bmatrix}$$

Wobei  $\vec{x}_0$  Punkt dem projizierten  $\vec{x}$  Punkt auf Ebene entspricht.  $\psi$  ist der Winkel zwischen  $\vec{n}$  und  $\vec{v}$

### 13.10 Parallelle Projektionsmatrix

$$\begin{bmatrix} x^* \\ y^* \\ z^* \end{bmatrix} = \mathbf{H} \begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix} =$$

$$\frac{1}{c_\psi} \begin{bmatrix} (c_\psi - av_x) & -bv_x & -cv_x & -dv_x \\ -av_y & (c_\psi - bv_y) & -cv_y & -dv_y \\ -av_z & -bv_z & (c_\psi - cv_z) & -dv_z \\ 0 & 0 & 0 & c_\psi \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \\ z_0 \\ 1 \end{bmatrix}$$

$$\cos(\psi) = c_\psi$$

### 13.11 Perspektivische Projektion

Fall wenn Zentrum O im Nullpunkt

$\epsilon : ax + by + cz + d = 0$ , Ebene

Beliebigen Punkt  $A_0(x_0, y_0, z_0)$  mit Projektionspunkt  $A^*(x^*, y^*, z^*)$  in Ebene  $\epsilon$

$$\begin{bmatrix} x^* \\ y^* \\ z^* \end{bmatrix} = \begin{bmatrix} \lambda x_0 \\ \lambda y_0 \\ \lambda z_0 \end{bmatrix}$$

$$\lambda = -\frac{d}{ax_0 + by_0 + cz_0}$$

$$(ax_0 + by_0 + cz_0) \cdot \begin{bmatrix} x^* \\ y^* \\ z^* \\ 1 \end{bmatrix} = \begin{bmatrix} -dx_0 \\ -dy_0 \\ -dz_0 \\ ax_0 + by_0 + cz_0 \end{bmatrix} =$$

### 13.12 Perspektivische Projektionmatrix

$$\mathbf{H} = \begin{bmatrix} -d & 0 & 0 & 0 \\ 0 & -d & 0 & 0 \\ 0 & 0 & -d & 0 \\ a & b & c & 0 \end{bmatrix}$$

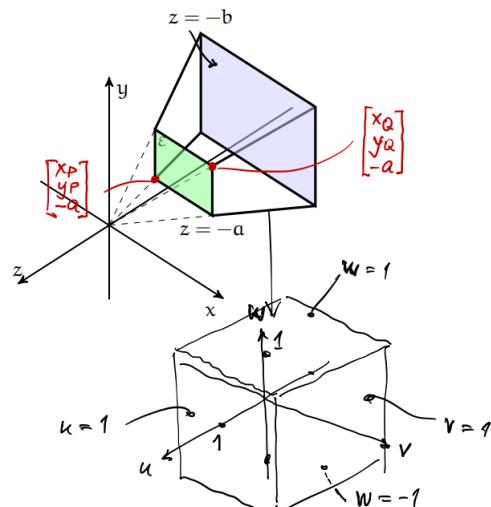
Projektionszentrum muss dann im zentrum liegen.

Wenn Ebene nicht mit Nullpunkt im Zentrum, dann zum Zentrum transferieren. Wichtig, die perspektivische Ebene muss transferiert werden. Bsp: x-y-Ebene hat  $\epsilon : z = 0$  dies mit der Transformation multiplizieren. Bei Zentrum der x-y-Ebene  $Z(2, 4, -3)$  entspricht  $\epsilon^* : z = 3$ , da  $d = 3$  wenn 0 Punkt verschoben:  $[-2, -4, 3, 1] \begin{bmatrix} 0 \\ 0 \\ 1 \\ d \end{bmatrix} = 0$

### 13.13 Sichtvolumen Clipping

Das kanonische Sichtvolumen ist ein Würfel mit  $P(\pm 1, \pm 1, \pm 1)$

Defür sind vorne und hinten, sowie zwei Punkte bestimmend Grösse gegeben



$P$  links unten,  $Q$  rechts oben  
 $z$  vorne  $z = -a$ ,  $z$  hinten  $z = -b$

$$\mathbf{T} = \begin{bmatrix} \frac{2a}{x_Q - x_P} & 0 & \frac{x_Q + x_P}{x_Q - x_P} & 0 \\ 0 & \frac{2a}{y_Q - y_P} & \frac{y_Q + y_P}{y_Q - y_P} & 0 \\ 0 & 0 & \frac{-b+a}{b-a} & -2 \frac{ba}{b-a} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

## 14 Curves

### 14.1 Kurve in der Ebene

#### Explizite Darstellung

$\gamma : [a, b] \rightarrow \mathbb{R}, x \mapsto y = f(x)$

Kreis:

oberer Halbkreis  $\sqrt{r^2 - x^2}$

unterer Halbkreis  $-\sqrt{r^2 - x^2}$

(Implizite Darstellung nach  $y$  auflösen -> Explizite Darstellung)

#### Implizite Darstellung

$F(x, y) = 0$

Kreis:  $x^2 + y^2 - r^2 = 0$

#### Parameterdarstellung

$\gamma : [a, b] \rightarrow \mathbb{R}^2, t \mapsto X(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix}$

Punkte miteinander verbunden, einzeln angegeben

Kreis:  $\begin{bmatrix} r \cos t \\ r \sin t \end{bmatrix}$

-> einzelne Funktionen für  $x$  &  $y$

### 14.2 Kurve im Raum

Parameterdarstellung mit 3 Komponenten

$\gamma : [a, b] \rightarrow \mathbb{R}^3, t \mapsto X(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \end{bmatrix}$

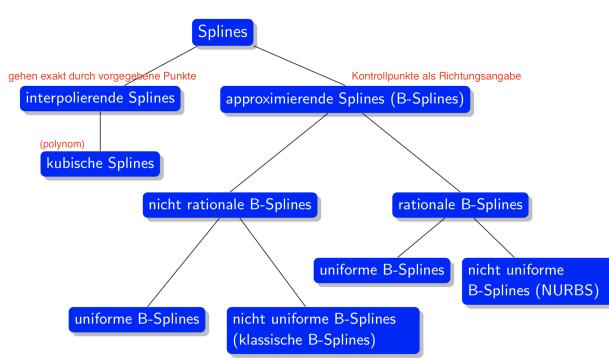
### 14.3 Spirale entlang des Zylinders

$$x^2 + y^2 = r^2$$

$\gamma : [0, 4\pi] \rightarrow \mathbb{R}^3, t \mapsto X(t) = \begin{bmatrix} r \cos t \\ r \sin t \\ ht/(2\pi) \end{bmatrix}$

Grundriss ergibt Kreis, Höhe Linear

### 14.4 Splines Übersicht (Interpolation versus Approximation)



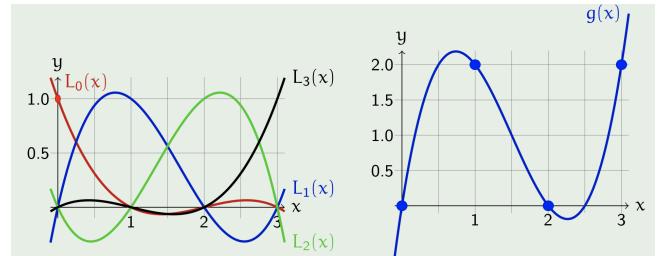
### 14.5 Methode unbestimmte Koeffizienten (Interpolation)

TODO Beschreibung  $P_3(x) = c_0 + c_1 x + c_2 x^2 + c_3 x^3$

$$\begin{bmatrix} 1 & x_0 & x_0^2 & x_0^3 \\ 1 & x_1 & x_1^2 & x_1^3 \\ 1 & x_2 & x_2^2 & x_2^3 \\ 1 & x_3 & x_3^2 & x_3^3 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{bmatrix}$$

$$c_0 = c_1 = c_2 = c_3 = 1$$

### 14.6 Lagrange Methode (Interpolation)



Polynome 3. Grades, welche an allen Stützstellen Null sind ausser an einer, wo der Wert Eins (1) sein soll:

$$L_0(x) = \frac{(x-x_1)(x-x_2)(x-x_3)}{(x_0-x_1)(x_0-x_2)(x_0-x_3)}$$

$$L_1(x) = \frac{(x-x_0)(x-x_2)(x-x_3)}{(x_1-x_0)(x_1-x_2)(x_1-x_3)}$$

$$L_2(x) = \frac{(x-x_0)(x-x_1)(x-x_3)}{(x_2-x_0)(x_2-x_1)(x_2-x_3)}$$

$$L_3(x) = \frac{(x-x_0)(x-x_1)(x-x_2)}{(x_3-x_0)(x_3-x_1)(x_3-x_2)}$$

$$g(x) = 0 \cdot L_0(x) + 2 \cdot L_1(x) + 0 \cdot L_2(x) + 2 \cdot L_3(x)$$

Addition der einzelnen Polynome ergibt die gesuchte Kurve durch die vorgegebenen Punkte

TODO aufräumen:  $l_0(x) = (x - x_1)(x - x_2) \dots$

$$L_0(x) = \frac{l_0(x)}{l_0(x_0)} = \frac{(x-x_1)(x-x_2)\dots}{(x_0-x_1)(x_0-x_2)\dots}$$

$$P_n(x) = y_0 L_0(x) + y_1 L_1(x) + \dots + y_n L_n()$$

Funktionen an den Stützstellen entweder Null oder Eins  $l_k(x) = \prod_{i=0, i \neq k}^n (x - x_i)$

$$L_k(x) = \frac{l_k(x)}{l_k(x_k)}$$

### 14.7 Lineare Bézier spline (Approximation)

$$P(0) = \text{Anfang}$$

$$P(1) = \text{Ende}$$

$$P(t) = (1-t)P_0 + P_1 \quad (0 \leq t \leq 1)$$

Gewichteter Durchschnitt der Kontrollpunkte

$$P(t) = (P_1 - P_0)t + P_0$$

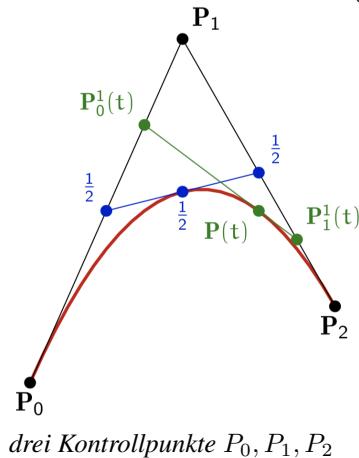
Polynom in  $t$

$$P(t) = [P_0, P_1] \begin{bmatrix} -1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} t \\ 1 \end{bmatrix} \quad (0 \leq t \leq 1)$$

Matrixform

## 14.8 Quadric Bézier spline (Approximation)

Verhältniss der Linien ist immer gleich

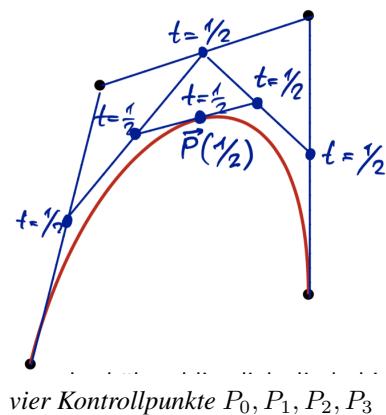


$$P_0^1(t) = (1-t)P_0 + P_1 \\ P_1^1(t) = (1-t)P_0 + P_1$$

$$P(t) = (1-t)^2 P_0 + 2(1-t)t P_1 + t^2 P_2$$

## 14.9 Cubic Bézier Spline (Approximation)

Punkt = Punkt auf Linie zwischen zwei Hilfslinien



$$\text{Mit } P_0^1, P_1^1 \text{ und} \\ P_2^1(t) = (1-t)P_2 + tP_3$$

$$P_1^2(t) = (1-t)P_0^1(t) + tP_1^1(t) \\ P_2^2(t) = (1-t)P_1^1(t) + tP_2^1(t)$$

$$P(t) = (1-t)^3 P_0 + 3(1-t)^2 t P_1 + 3(1-t)t^2 P_2 + t^3 P_3$$

## 14.10 Bernsteinpolynome

TODO

## 14.11 Bézier-Kurven

TODO Kurve immer innerhalb der konvexen Hülle des Linienzuges der Kontrollpunkte

## 14.12 B-Spline-Kurven

TODO

- lokale Funktionen nur in der Nähe von  $P_j$  verschieden von 0
- Einfluss vom Punkt auf die gesamte Kurve ist kleiner

## 14.13 NURBS (Non Uniform Rational Basic Splines)

TODO

- homogenisiert
- rationale B-Spline-Kurve
- Punkte mit Gewichtung
- invariant bei den Transformationen

## 15 Flächen in Kartesische Koordinaten

- Eine Fläche  $S$  ist eine **Regelfläche** falls durch jeden ihrer Punkte eine Gerade existiert, die vollständig in  $S$  liegt.
- **doppelte Regelfläche** falls durch jeden ihrer Punkte zwei unterschiedliche Geraden existieren

### Explizite Darstellung

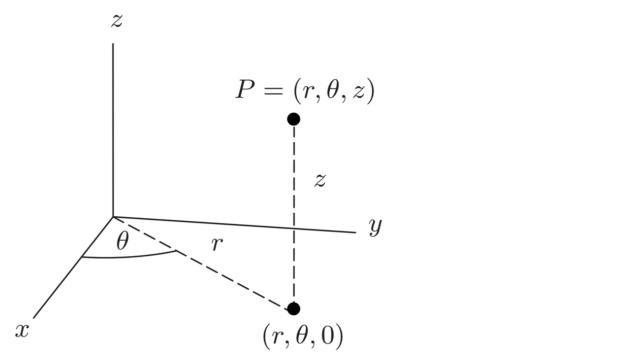
$$\text{Sattel: } z = f(x, y) = x^2 - y^2$$

### Implizite Darstellung

$$\text{Kugel: } (x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2 = R^2 \\ x_0, y_0, z_0 = \text{Zentrum}$$

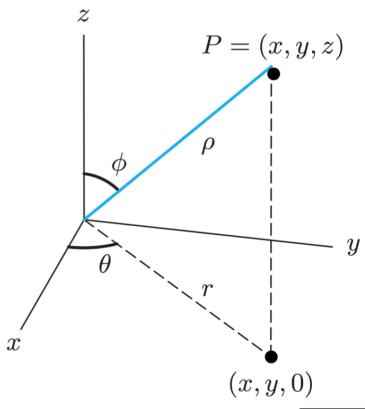
### Parameterdarstellung

Zylinderkoordinaten:



Kugelkoordinaten:

$$(\rho, \phi, \theta)$$



$$\begin{aligned} x &= \rho \sin \phi \cos \theta \\ y &= \rho \sin \phi \sin \theta \\ z &= \rho \cos \phi \end{aligned} \quad \iff \quad \begin{aligned} \rho &= \sqrt{x^2 + y^2 + z^2} \\ \tan \theta &= y/x \\ \cos \phi &= z/\rho \end{aligned}$$

Rotationsfläche:  $x = u$   $y = f(x) * \cos(v)$   $z = f(x) * \sin(v)$

## 15.1 Freiformflächen

TODO

- Tensorproduktfläche  $S(u,v) = \text{Fläche gegeben von Kurven } F(u) \& G(v)$   
*Für jeden u Wert gibt es eine Kurve -> "Gitterpunkte"*

## 16 Appendix

### 16.1 Radians

Winkel $\alpha^\circ$	Bogenmaß	Sinus	Kosinus
$0^\circ$	0	$\frac{1}{2}\sqrt{0} = 0$	$\frac{1}{2}\sqrt{4} = 1$
$30^\circ$	$\frac{\pi}{6}$	$\frac{1}{2}\sqrt{1} = \frac{1}{2}$	$\frac{1}{2}\sqrt{3}$
$45^\circ$	$\frac{\pi}{4}$	$\frac{1}{2}\sqrt{2} = \frac{1}{\sqrt{2}}$	$\frac{1}{2}\sqrt{2} = \frac{1}{\sqrt{2}}$
$60^\circ$	$\frac{\pi}{3}$	$\frac{1}{2}\sqrt{3}$	$\frac{1}{2}\sqrt{1} = \frac{1}{2}$
$90^\circ$	$\frac{\pi}{2}$	$\frac{1}{2}\sqrt{4} = 1$	$\frac{1}{2}\sqrt{0} = 0$
$180^\circ$	$\pi$	0	-1
$270^\circ$	$\frac{3\pi}{2}$	-1	0
$360^\circ$	$2\pi$	0	1

$$\cos^2(\alpha) = \frac{1}{1+\tan^2(\alpha)}$$

$$\sin^2(\alpha) = \frac{\tan^2(\alpha)}{1+\tan^2(\alpha)}$$

$$\sin(\alpha + \beta) = \sin \alpha \cos \beta + \cos \alpha \sin \beta$$

$$\cos(\alpha + \beta) = \cos \alpha \cos \beta - \sin \alpha \sin \beta$$

$$\cos(\alpha) = \cos(-\alpha)$$

$$\sin(-\alpha) = -\sin(\alpha)$$