

Computer Graphics Zusammenfassung

Lucien Zürcher & Melvin Werthmüller

January 7, 2019

Contents

1 Farbe	3	5 Visibilität	8
1.1 Was ist Farbe?	3	5.1 Backface Culling (WebGL)	8
1.2 Farbe eines Objektes	3	5.2 Tiefensortierung (Painters Algorithms)	8
1.3 Licht besteht aus?	3	5.3 z-Buffer (WebGL)	8
1.4 Das Auge	3	5.4 Warnock Algorithmus	8
1.5 Wie sehen wir Farbe?	3		
1.6 Wahrnehmung	3		
1.7 Farbsysteme	3		
1.8 Additives Farbsystem	4		
1.9 Subtraktives Farbsystem	4		
1.10 Farben Konvertieren	4		
1.11 Gamma Korrektur	4		
1.12 Normfarbtafel / CIE Farbsystem	4		
1.13 Helligkeitswahrnehmung	4		
1.14 Nibs (Lichtdichte)	4		
1.15 Mach bending	4		
1.16 Farbtäuschung	4		
1.17 HD,UHD,UK	4		
1.18 Was ist HDR?	5		
1.19 Begriffe	5		
2 Halbtontechnik	5	6 Clipping	9
2.1 Verfahren der Halbtontechnik	5	6.1 Verfahren	9
2.2 Quantisierung	5	6.2 Clipping von Linien	9
2.3 Dithering	5	6.2.1 Clipping von Linien nach Cohen-Sutherland	9
2.3.1 Dithermatrizen	5	6.3 Clipping von Polygonen mit dem Algorithmus von Sutherland-Hodgeman	9
2.3.2 Dithering bei gleich bleibender Auflösung	5		
2.3.3 Dispersed Dot Dithering	5		
2.3.4 Error Diffusion	5		
3 Viewing	6	7 Beleuchtung & Schattierung	9
3.1 Planare geometrische Projektionen	6	7.1 Beleuchtungsmodelle	10
3.1.1 parallele Projektionen	6	7.1.1 Diffuse Reflektion (Lambert Modell)	10
3.1.2 perspektivische Projektionen	6	7.1.2 Spiegelnde Reflektion (Phong Modell)	10
4 Scan Konvertierung	6	7.2 Lichtquellen	10
4.1 Linie Rastern	6	7.3 Schattierung	10
4.2 Mittelpunktschema	6		
4.3 Kreis Rastern	7		
4.4 Mittelpunktschema Kreis	7		
4.5 Regionen füllen	7		
4.5.1 FloodFill	7		
4.6 Zeichnen von gefüllten Polygonen	7		
4.6.1 Scanlinien Algorithmus	7		
4.6.2 Zeichnen von Dreiecken	7		
4.7 Anti-Aliasing	8		
5 Visibilität	8	8 Raytracing	10
6 Clipping	9	9 Rendering Equation(s)	10
7 Beleuchtung & Schattierung	9	9.1 BRDF (Bidirectional Reflectance Distribution Function)	10
8 Raytracing	10	10 WebGL	10
9 Rendering Equation(s)	10	10.1 OpenGL Merkmale	10
10 WebGL	10	10.2 Grafikpipeline	11
11 Vektoren	12	10.3 Programmierbare Shaders	11
11.1 Addition	12	10.3.1 Vertex Processing / Vertex Shader	11
11.2 Multiplikation mit Skalar	12	10.3.2 Fragment Processing / Fragment Shader	11
11.3 Nullvektor	12	10.4 Datenfluss	11
11.4 Vektorinverses	12	10.5 Attribute und Uniform Variablen mit Shaders verbinden	11
11.5 Vektoren Gleichheit	12	10.6 Attribut Variablen und Buffer definieren	11
11.6 Skalarprodukt	12		
11.7 Skalarprodukt im beliebigem Koordinatensystem	12		

11.8 Orthogonal	12	14.4 Methode unbestimmte Koeffizienten	18
11.9 Länge des Vektors	12	14.5 Lagrange Methode	18
11.10 Einheitsvektor	12	14.6 Lineare Bézier spline	18
11.11 Euklidische Distanz	12	14.7 Quadric Bézier spline	18
11.12 Gerade im 2/3D	12	14.8 Cubic Bézier Spline	18
11.13 Hessische Normalform	12	14.9 Bernsteinpolynome	18
11.14 Hessische Normalform Ebene	13		
11.15 Achsenabschnitt	13		
11.16 Projektion eines Vektors	13	15 Appendix	18
11.17 Vektorprodukt	13	15.1 Radians	18
11.18 Vektorprodukt Anwendung	13		
11.19 Spatprodukt	13		
11.20 Translation 2D	13		
11.21 Skalierung 2D	13		
11.22 Rotation 2D	13		
11.23 Vektor Rechenregeln	13		
11.24 Rechenregel Skalarprodukt	14		
11.25 Vektorprodukt Rechenregeln	14		
11.26 Spatprodukt Rechenregeln	14		
11.27 Begriffe	14		
12 Projektive Geometrie	14		
12.1 Homogener Vektor	14		
12.2 Punkt auf Gerade	14		
12.3 Schnittpunkt Geraden	14		
12.4 Parallele Geraden	14		
12.5 Unendlicher homogener Vektor	14		
12.6 Projektive Ebene (homogener Vektor)	14		
12.7 Projektive Transformation	14		
12.8 Transformationen kombinieren	15		
12.9 Translation 2D	15		
12.10 Nullpunkt Rotation 2D	15		
12.11 Rotation um Punkt A	15		
12.12 Spiegelung mit Gerade durch Ursprung	15		
12.13 Spiegelung mit Gerade g	15		
12.14 Transformation des Koordinatensystems	15		
12.15 Abstand Punkt zur Gerade	15		
12.16 Transformationen 2D	15		
13 Transformation	15		
13.1 homogene Koordinaten	15		
13.2 Ebene im Raum	15		
13.3 Projektive Transformation	16		
13.4 Transformationen	16		
13.5 Translation	16		
13.6 Rotation	16		
13.7 Rotation um beliebige Achse	16		
13.8 Rotation um eine Achse durch den Ursprung	17		
13.9 Parallelprojektion	17		
13.10 Parallelprojektionsmatrix	17		
13.11 Perspektivische Projektion	17		
13.12 Perspektivische Projektionsmatrix	17		
13.13 Sichtvolumen Clipping	17		
14 Curves	18		
14.1 Kurve in der Ebene	18		
14.2 Kurve im Raum	18		
14.3 Spirale entlang des Zylinders	18		

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

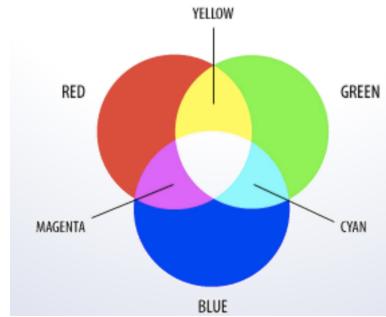
Grün 530nm wird am intensivsten wargenommen

Die Helligkeitswahrnehmung zwischen Stäbchen und Zäpfchen ist unterschiedlich

1.7 Farbsysteme

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

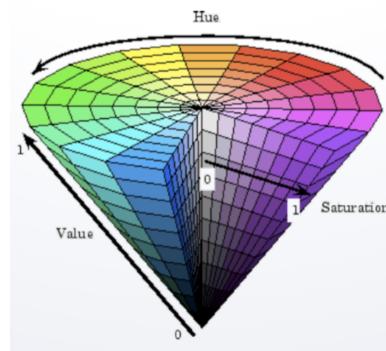
- **RGB** (Monitor, Spotlights, Pointilismus), additiv, $C = (\text{Rot}, \text{Grün}, \text{Blau})$



- **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

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

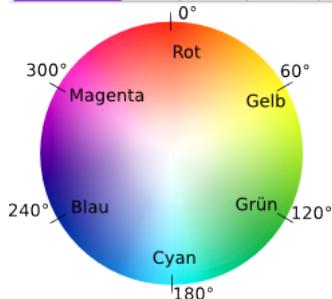
1.10 Farben Konvertieren

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

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

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

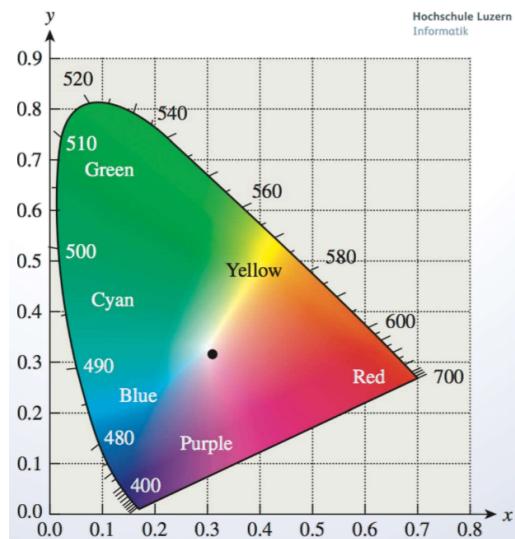
1.12 Normfarbtafel / CIE Farbsystem

- 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

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

$$\log(I + \Delta I) - \log(I) = \text{Const}$$

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 Helligkeitsunterschieden.

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 tiefere Auflösung durch Runden der Pixelfarbwerte auf verfügbare Farben.

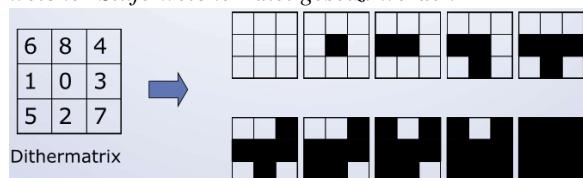
Bsp. 16Bit -> 8Bit (Runden der Werte)

2.3 Dithering

Wenn der Drucker eine grössere 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 x 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 x 2 Bayer Matrix

0	2
3	1

4 x 4 Bayer Matrix

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

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

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

n : Grösse der Matrix (2 x 2 => $n = 2$)

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

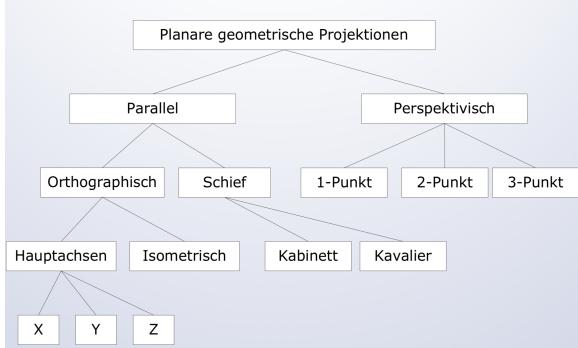
Wenn Wert > 128 = 255, ansonsten Wert <= 128 = 0

3 Viewing

von 3D in 2D abbilden

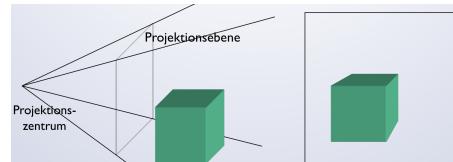
3.1 Planare geometrische Projektionen

Bild auf eine Ebene projizieren



3.1.2 perspektivische Projektionen

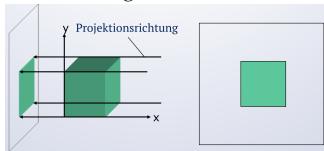
- Simuliert Kamera oder Auge
- Spezifiziert durch Projektionszentrum und Projektionsebene
- Geraden bleiben erhalten
- Parallelen schneiden sich in einem Punkt (Fluchtpunkt)
- Objektgröße nimmt proportional zum Abstand vom Projektionszentrum ab



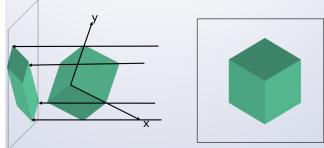
3.1.1 parallele Projektionen

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

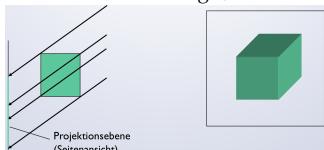
Orthographische Projektion von einer Seite / rechtwinklig



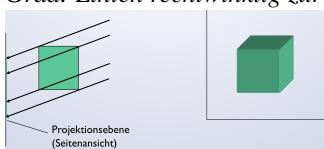
Isometrische Projektion Orthographische auf die Ebene



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



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



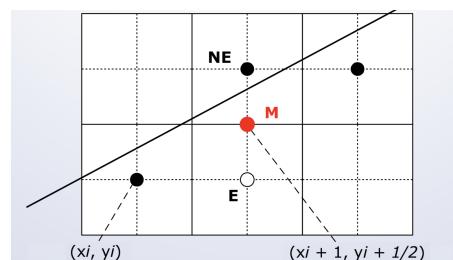
4 Scan Konvertierung

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:

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

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
DE = 2 * x + 3
DNE = 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 x)
- Tabelle aktiver Kanten (AET)
Momentane Kanten für Scanlinie (sortiert nach x
asc) zum Zeichnen vom einer Kante zur nächsten
(immer zwei)

Erzeuge ET

Initialisiere AET = empty

y=0

repeat

Addiere 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

- Unterteilung in 3 Geraden und dann auf der richtigen Seite aller Geraden

4.7 Anti-Aliasing

Treppenmuster vermeiden

- Prefiltering -> Farbintensität proportional zur Fläche
- Supersampling -> Bild mit höherer Auflösung berechnen und Farbmittelwerte als Übergänge zeichnen

5 Visibilität

Algorithmen für verdeckte Linien und Flächen

5.1 Backface Culling (WebGL)

- Rückseiten nicht zeichnen
- Test durch Normalvektor (Normalvektor gegen hinten und nicht nach vorne)

5.2 Tiefensortierung (Painters Algorithms)

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

Sortierung:

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

Bedingungen:

1. Überlagern sich die x-Ausdehnungen nicht? (vollständig nebeneinander)
2. Überlagern sich die y-Ausdehnungen nicht? (vollständig hintereinander)
3. Wird das hintere Objekt vom Vorderen komplett überdeckt?
4. Liegt Q ganz auf der Betrachterseite von P? (Polygon P ist in der Sortierung vor Q)
5. Überlappen sich die Polygone nicht auf der Projektion in die xy Ebene?

TODO Beispiel

Eigenschaften:

- + auch für transparente Objekte möglich
- + einfach für Spezialfälle (2.5D)
- - ineffizient für viele Objekte O(n²)
- - nicht Hardware unterstützt

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
```

Berechnung aus Ebenengleichung:

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

Inkrementelle Berechnung entlang einer Scanlinie:

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

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
- - (grosser Speicherbedarf)

5.4 Warnock Algorithmus

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

Einfach falls:

1. Der Bereich kein Polygon enthält
2. Der Bereich nur ein Polygon enthält
3. Der Bereich ein Polygon enthält, das am nächsten liegt und den Bereich vollständig ausfüllt
4. Der Bereich nur aus einem Pixel besteht

6 Clipping

Objekte ausserhalb des Viewingbereichs nicht darstellen

- Vermeiden von unnötiger Rasterisierung
- Ressourcen schonen

6.1 Verfahren

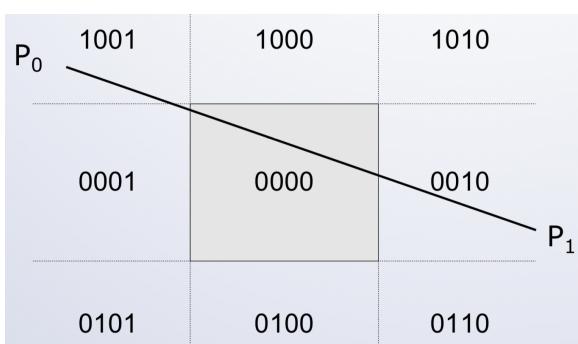
- Scissoring
alle Pixel berechnen & darstellen wenn im Fenster
- Temporärer Buffer
zeichnen in einem temporären Buffer, welcher dann Kopiert wird
- Analytische Berechnung derjenigen Teile, die im Innen des Fensters liegen

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



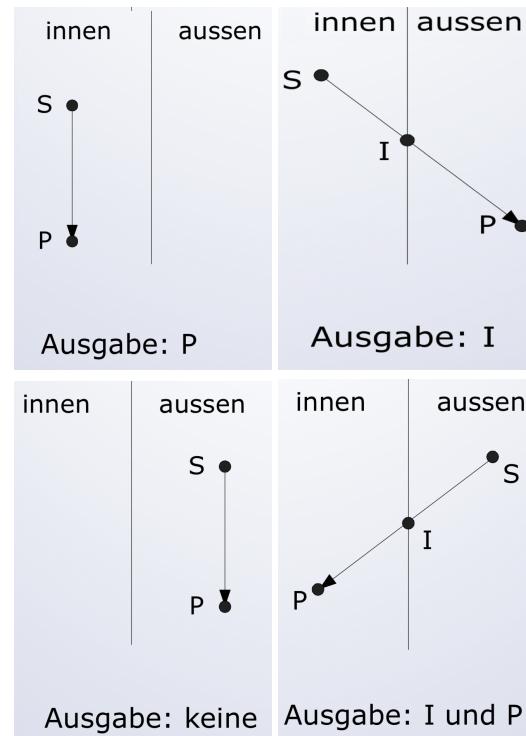
Bitweise AND

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

TODO Beispiel

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

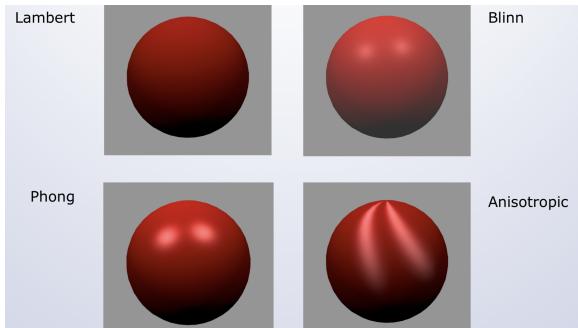
TODO Beispiel

7 Beleuchtung & Schattierung

Eigenschaften eines Objektes:

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

7.1 Beleuchtungsmodelle



TODO Formeln

7.1.1 Diffuse Reflektion (Lambert Modell)

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

7.1.2 Spiegelnde Reflektion (Phong Modell)

- Intensität Spiegelung = Intensität der Lichtquelle * Reflektierungsfaktor * \cos^n (Winkel zwischen Betrachtung & Reflektionsrichtung)
 n = Streuung des Lichts (hohes n -> nimmt schnell ab)
Reflektierungsfaktor = Farbe der Reflektion

7.2 Lichtquellen

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

7.3 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)

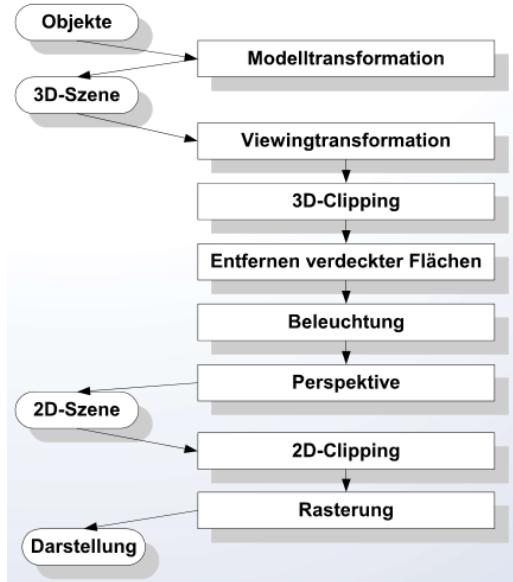
- Anstatt Diffuse (Gleichmässig) oder Specular (Abnehmend) Reflektion an einem Punkt
- Radiosity: Beleuchtete Fläche als neue Lichtquelle (indirekt)
- Fläche wird in "Patches" unterteilt
- "Patches" mit konstanter Farbe / Lichtintensität
- "Patches" als neue Lichtquelle für andere "Patches"
- Berechnungsvarianten -> *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 Platformen
- 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.

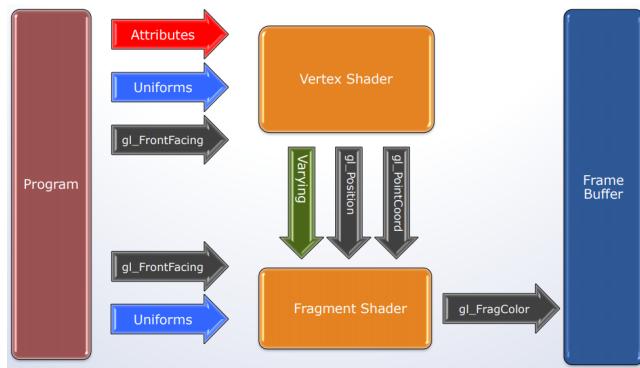
10.3.1 Vertex Processing / Vertex Shader

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

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.aVertexPosition, 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

11.1 Addition

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

11.2 Multiplikation mit Skalar

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

$\lambda \in \text{Skalar}$

11.3 Nullvektor

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

11.4 Vektorinverses

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

Vektor mit negativen Komponenten

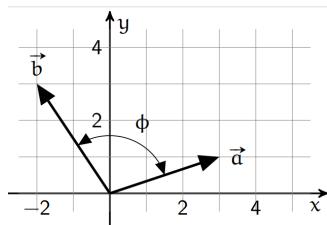
11.5 Vektoren Gleichheit

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

Vektoren sind gleich, wenn Komponenten gleich

11.6 Skalarprodukt

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



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

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

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

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

11.7 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

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

11.10 Einheitsvektor

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

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

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

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

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

11.11 Euklidische Distanz

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

11.12 Gerade im 2/3D

• Punkt-Punktform mit Vektoren 2/3D

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

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

• 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

$$y = mx + b$$

y : Achsenabschnitt, m : Steigung

• Punkt-Richtungsform

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

(x_0, y_0) : Punkt, m : Steigung

• Allgemeine Geradengleichung

$$ax + by + c = 0$$

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

11.13 Hessische Normalform

Vektorielle Schreibweise der Hessischen Normalform

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

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

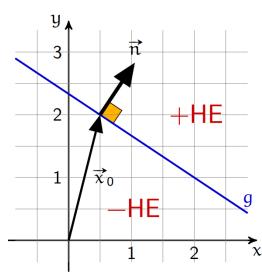
$$\Rightarrow n_x(x - x_0) + n_y(y - y_0) = n_x x + n_y y - (n_x x_0 + n_y y_0)$$

Abstand vom Ursprung: d

$$d = (n_x x_0 + n_y y_0) = \vec{n} \bullet \vec{x}_0$$

\vec{n} muss normalisiert sein:

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



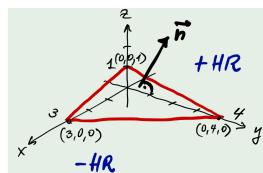
$$\begin{aligned} n_x x + n_y y - d &= 0 \\ d &= (n_x x_0 + n_y y_0) = \vec{n} \cdot \vec{x}_0 \\ d > 0 &\Leftrightarrow (0,0) \in -HE \\ d < 0 &\Leftrightarrow (0,0) \in +HE \\ g: ax + by + c &= 0 \\ \vec{n} &= \begin{bmatrix} n_x \\ n_y \end{bmatrix} = \frac{1}{\sqrt{a^2+b^2}} \begin{bmatrix} a \\ b \end{bmatrix} \\ d &= -\frac{c}{\sqrt{a^2+b^2}} \end{aligned}$$

11.14 Hessische Normalform Ebene

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

$$\begin{aligned} n_x x + n_y y + n_z z - D &= 0; \text{HNF der Ebene } \epsilon \in \mathbb{R}^3 \\ \vec{n} &= \begin{bmatrix} n_x \\ n_y \\ n_z \end{bmatrix} = \frac{1}{\sqrt{a^2+b^2+c^2}} \begin{bmatrix} a \\ b \\ c \end{bmatrix} \\ D &= -\frac{d}{\sqrt{a^2+b^2+c^2}} \end{aligned}$$

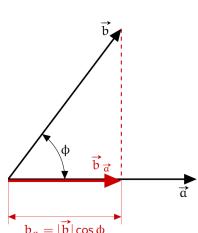
11.15 Achsenabschnitt



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



$$\begin{aligned} \vec{b} &\text{ Richtung } \vec{a}: \\ \vec{b}_{\vec{a}} &= \frac{\vec{a} \cdot \vec{b}}{|\vec{a}|^2} \vec{a} \end{aligned}$$

$$\begin{aligned} b_a &\text{ mal Einheitsvektor } \vec{a} \\ \vec{b}_{\vec{a}} &= b_a \frac{1}{|\vec{a}|} \vec{a} = |\vec{a}| |\vec{b}| \cos \phi \frac{1}{|\vec{a}|} \vec{a} \\ &= \frac{\vec{a} \cdot \vec{b}}{|\vec{a}|^2} \vec{a} \end{aligned}$$

11.17 Vektorprodukt

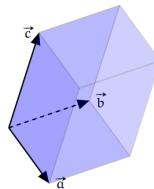
$$\begin{aligned} \vec{a} \times \vec{b} &\text{ steht senkrecht auf beiden Vektoren} \\ \vec{a}, \vec{b} \text{ und } \vec{a} \times \vec{b} &\text{ sind ein Rechtsystem} \\ \vec{a} \times \vec{b} &\text{ entspricht der Fläche des aufgespannten Parallelogramms (A):} \\ A &= \sqrt{(a_2 b_3 - a_3 b_2)^2 + (a_3 b_1 - a_1 b_3)^2 + (a_1 b_2 - a_2 b_1)^2} \\ \vec{a} \times \vec{b} &= \begin{bmatrix} \vec{e}_1 & \vec{e}_2 & \vec{e}_3 \\ a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \end{bmatrix} = \begin{bmatrix} a_2 b_3 - a_3 b_2 \\ a_3 b_1 - a_1 b_3 \\ a_1 b_2 - a_2 b_1 \end{bmatrix} \\ &= (a_2 b_3 - a_3 b_2) \vec{e}_1 + (a_3 b_1 - a_1 b_3) \vec{e}_2 + (a_1 b_2 - a_2 b_1) \vec{e}_3 \end{aligned}$$

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 Spatprodukt



$$\begin{aligned} \text{Spatprodukt } [\vec{a}, \vec{b}, \vec{c}] &= \vec{a} \bullet (\vec{b} \times \vec{c}) \\ \text{Spatprodukt entspricht Volumen wenn in einem Rechtssystem, dann: } V_{\text{Spat}} &= |[\vec{a}, \vec{b}, \vec{c}]| \\ |[\vec{a}, \vec{b}, \vec{c}]| &= a_1 b_2 c_3 + a_2 b_3 c_1 + a_3 b_1 c_2 - a_3 b_2 c_1 - a_1 b_3 c_2 - a_2 b_1 c_3 \\ \text{Komplanar (linear abhängig) wenn } [\vec{a}, \vec{b}, \vec{c}] &= 0 \end{aligned}$$

11.20 Translation 2D

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

11.21 Skalierung 2D

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

11.22 Rotation 2D

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

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

11.23 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 $-\vec{a}$
$\lambda(\vec{a} + \vec{b}) = \lambda\vec{a} + \lambda\vec{b}$	
$(\lambda + \mu)\vec{a} = \lambda\vec{a} + \mu\vec{a}$	
$(\lambda\mu)\vec{a} = \lambda(\mu\vec{a}) = \mu(\lambda\vec{a})$	
$1\vec{a} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \vec{a} = \vec{a}$	

11.24 Rechenregel Skalarprodukt

$$\begin{aligned}\vec{a} \bullet \vec{b} &= \vec{b} \bullet \vec{a} \\ \vec{a} \bullet (\vec{b} + \vec{c}) &= \vec{a} \bullet \vec{b} + \vec{a} \bullet \vec{c} \\ \lambda(\vec{a} \bullet \vec{b}) &= (\lambda\vec{a}) \bullet \vec{b} = \vec{a} \bullet (\lambda\vec{b})\end{aligned}$$

11.25 Vektorprodukt Rechenregeln

$$\begin{array}{l|l} \vec{a} \times \vec{b} = -\vec{b} \times \vec{a} & \text{Anti-Kommutativgesetz} \\ \vec{a} \times (\vec{b} + \vec{c}) = \vec{a} \times \vec{b} + \vec{a} \times \vec{c} & \text{Distributivgesetz} \\ \lambda(\vec{a} \times \vec{b}) = (\lambda\vec{a}) \times \vec{b} = \vec{a} \times (\lambda\vec{b}) & \end{array}$$

11.26 Spatprodukt Rechenregeln

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

11.27 Begriffe

Ortsvektor	Vom Ursprung zum Punkt
Richtungsvektor	Eine Richtung im Raum
Einheitsvektor	Eine Einheit in eine beliebige Richtung
Linearkombination <i>kollinear</i>	Ein Vektor, der ein Vielfaches eines Einheitsvektors ist.
Linear Unabhängig <i>komplanar</i>	$\vec{c} = \lambda\vec{a} + \mu\vec{b}$ Vektoren sind unabhängig wenn $\lambda_1\vec{a}_1 + \lambda_2\vec{a}_2 + \dots + \lambda_n\vec{a}_n = \vec{0}$ $\Leftrightarrow \lambda_1 = \lambda_2 = \dots = \lambda_n = 0$
Skalar Rechtssystem	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

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

\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

$$\begin{aligned}\vec{r} &= \vec{g}_1 \times \vec{g}_2 = (c_1 + c_2) \begin{bmatrix} b_1 \\ -a_1 \\ 0 \end{bmatrix} \\ (a_1, b_1) &= (a_2, b_2), \text{ dann sind die Geraden parallel}\end{aligned}$$

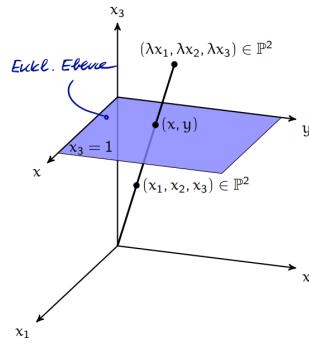
12.5 Unendlicher homogener Vektor

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

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

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

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_2 dann h_1

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 ϕ , \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 Φ
3. Translation A zurück (\mathbf{T}^{-1})

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

12.12 Spiegelung mit Gerade durch Ursprung

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

$$\text{HNF: } -\sin(\delta)x + \cos(\delta)y = 0$$

$$\vec{p}' = \vec{p} - 2(\vec{p} \bullet \vec{n})\vec{n}$$

$$\delta = \arctan\left(\frac{y}{x}\right)$$

Wenn g geht durch Nullpunkt

$$\vec{r}' = \begin{bmatrix} \cos(2\delta) & \sin(2\delta) & 0 \\ \sin(2\delta) & -\cos(2\delta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \vec{r}$$

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 Φ

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

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

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 ϵ im Raum \mathbb{R}^3

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

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

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

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

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

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

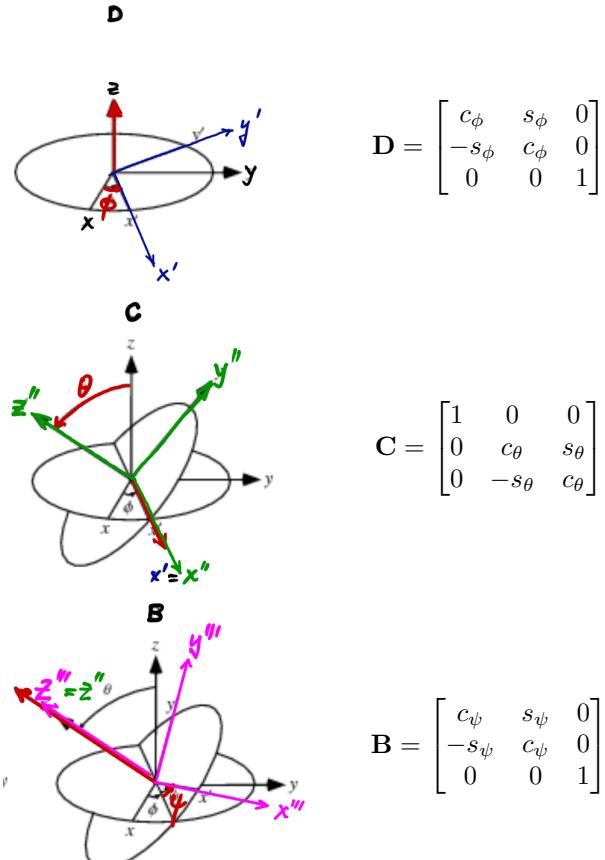
Bei Rotation um selbe Achse gilt Kommutativgesetz ($\mathbf{R}_z(\Phi_{z,1} + \Phi_{z,2}) = \mathbf{R}_z(\Phi_{z,1})\mathbf{R}_z(\Phi_{z,2}) = \mathbf{R}_z(\Phi_{z,2})\mathbf{R}_z(\Phi_{z,1})$)

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

13.7 Rotation um beliebige Achse

- 1) Rotation um ϕ 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 ψ (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 \mathbf{D}

& C Matrix (mit Winkel der Gerade), dann eigentliche Transformation mit gegebenem Winkel, dann zurücktransformiere C^{-1} & D^{-1}

13.8 Rotation um eine Achse durch den Ursprung

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

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

13.9 Parallele Projektion

Projektion auf Ebene $\epsilon : ax + by + cz + d = 0$
Die Ebene ist definiert durch Normalenvektor $\vec{n} = [a, b, c]^T$ (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. ψ ist der Winkel zwischen \vec{n} und \vec{v}

13.10 Parallelprojektionsmatrix

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

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

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

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 ϵ

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

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

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

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

13.12 Perspektivische Projektionsmatrix

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

Projektionszentrum muss dann im Zentrum liegen.

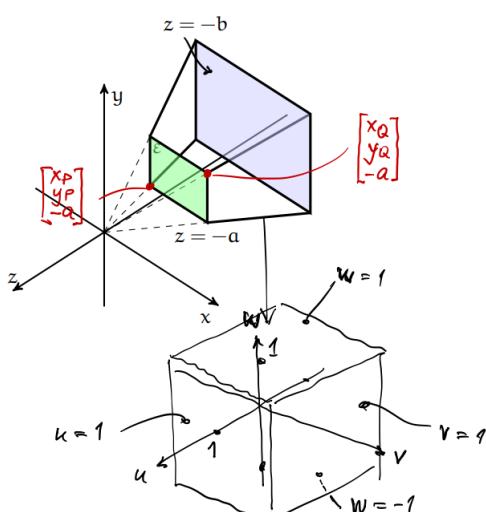
Wenn Ebene nicht mit Nullpunkt im Zentrum, dann zum Zentrum transferieren. Wichtig, die perspektivische Ebene muss transferiert werden. Bsp: x-y-Ebene hat $\epsilon : z = 0$ dies mit der Transformation multiplizieren. Bei Zentrum der x-y-Ebene $Z(2, 4, -3)$ entspricht $\epsilon^* : z = 3$, da $d = 3$ 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 gegebenen



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:

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

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

Implizite Darstellung

$$F(x, y) = 0$$

$$\text{Kreis: } x^2 + y^2 - r^2 = 0$$

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

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

14.2 Kurve im Raum

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

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 Methode unbestimmte Koeffizienten

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

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

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

14.5 Lagrange Methode

$$l_0(x) = (x - x_1)(x - x_2) \dots$$

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

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

$$l_k(x) = \prod_{i=0, i \neq k}^n (x - x_i)$$

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

14.6 Lineare Bézier spline

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

Gewichteter Durchschnitt der Kontrollpunkte

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

Polynom in t

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

Matrixform

14.7 Quadric Bézier spline

drei Kontrollpunkte P_0, P_1, P_2

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

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

$$P(t) = (1-t)^2 P_0 + 2(1-t)tP_1 + t^2 P_2$$

14.8 Cubic Bézier Spline

vier Kontrollpunkte P_0, P_1, P_2, P_3

$$\text{Mit } P_0^1, P_1^1 \text{ und}$$

$$P_2^1(t) = (1-t)P_2 + tP_3$$

$$P_0^2(t) = (1-t)P_0^1(t) + tP_1^1(t)$$

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

$$P(t) = (1-t)^3 P_0 + 3(1-t)^2 tP_1 + 3(1-t)t^2 P_2 + t^3 P_3$$

14.9 Bernsteinpolynome

15 Appendix

15.1 Radians

Winkel α°	Bogenmass	Sinus	Kosinus
0°	0	$\frac{1}{2}\sqrt{0} = 0$	$\frac{1}{2}\sqrt{4} = 1$
30°	$\frac{\pi}{6}$	$\frac{1}{2}\sqrt{1} = \frac{1}{2}$	$\frac{1}{2}\sqrt{3}$
45°	$\frac{\pi}{4}$	$\frac{1}{2}\sqrt{2} = \frac{1}{\sqrt{2}}$	$\frac{1}{2}\sqrt{2} = \frac{1}{\sqrt{2}}$
60°	$\frac{\pi}{3}$	$\frac{1}{2}\sqrt{3}$	$\frac{1}{2}\sqrt{1} = \frac{1}{2}$
90°	$\frac{\pi}{2}$	$\frac{1}{2}\sqrt{4} = 1$	$\frac{1}{2}\sqrt{0} = 0$
180°	π	0	-1
270°	$\frac{3\pi}{2}$	-1	0
360°	2π	0	1

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

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

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

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

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

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