Computergrafik

Mitschrift von

Markus Vieth   Steffen Eiden   Lukas Birklein

12. Januar 2017

Vorwort

Dieses Skript basiert auf unserer Mitschrift der Vorlesung Computergrafik und VR im WS 2016/17 an der JGU Mainz (Dozent: Prof. Dr. E. Schömer).
Es handelt sich nicht um eine offizielle Veröffentlichung der Universität.
Wir übernehmen keine Gewähr für die Fehlerfreiheit und Vollständigkeit des Skripts.
Fehler können unter Github gemeldet werden. Die aktuelle Version dieses Skriptes ist ebenfalls auf Github zu finden.

Inhaltsverzeichnis

Vorwort i

1 Koordinatensysteme 1
- 1.1 Normalisiertes Koordinatensystem . . . 1
- 1.2 Bildschirmkoordinaten $\leftrightharpoons$ Weltkoordinaten . . . 1
- 1.3 Homogene Koordinaten . . . 2
- 1.4 Transformationen . . . 3
  - 1.4.1 Rotation . . . 3
  - 1.4.2 + Verschiebung . . . 3
  - 1.4.3 Skalierung . . . 3
  - 1.4.4 Translation . . . 3
  - 1.4.5 Hintereinanderausführung von Translation, Rotation und Skalierung . . . 4
- 1.5 Invertierung von M . . . 4
- 1.6 Qt . . . 5

2 VBO 6
- 2.1 Baryzentrische Koordinaten . . . 7
- 2.2 Texturen . . . 8
  - 2.2.1 Mipmap . . . 8

3 3D-Objekte 10
- 3.1 Orthogonalprojektion . . . 10
- 3.2 Perspektivische Projektion . . . 11
- 3.3 Objekte drehen aka Blickwinkel ändern . . . 14

4 Qt Formen 15
- 4.1 OFF-Format . . . 16

5 Beleuchtung 17
- 5.1 Ein einfaches Beleuchtungsmodel . . . 17
  - 5.1.1 Lambertsches Gesetz . . . 17

6 Einschübe 18
- 6.1 Berechnung des Normalenvektors für parametrisierte Flächen . . . 18
- 6.2 Tiefenbuffer . . . 18

7 Beleuchtung (Fortsetzung) 19
- 7.1 Flatshading . . . 19

8 Einschub: virtual Trackball 21
- 8.1 Formel von Rodriques . . . 21
  - 8.1.1 dyadisches Produkt . . . 22

9 Beleuchtung (Fortsetzung) 23
9.1 Phong Lichtmodell 23
9.1.1 Phong 23

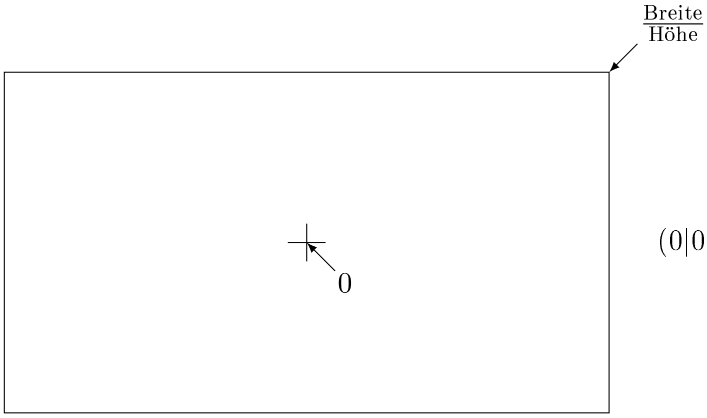
10 Oberflächen 25
10.1 Texturen 25
10.2 Cube-Mapping 26

11 Volume Rendering mit 3D-Texturen 27
11.1 DVT 27
11.1.1 Lambert-Beer-Gesetz 27
11.2 Raycasting mittels front-to-back rendering 28

12 Einschub: Algebraische Flächen 29

1 Koordinatensysteme

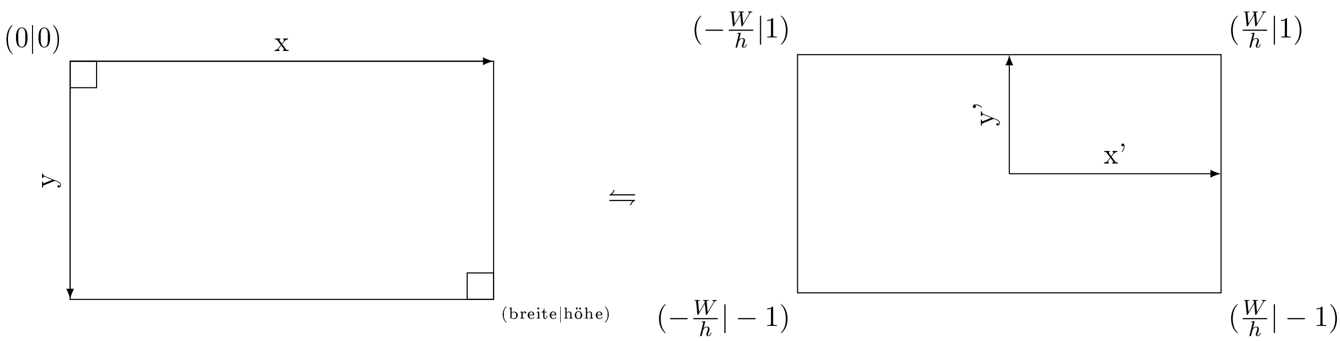
1.1 Normalisiertes Koordinatensystem



(0|0) in der Mitte

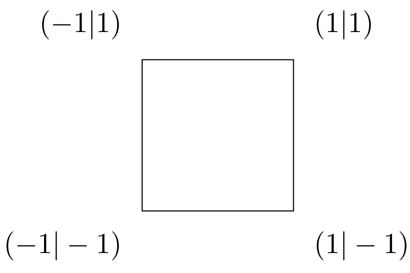
`glViewport`: Ausschnitt wo gezeichnet wird.

1.2 Bildschirmkoordinaten $\leftrightharpoons$ Weltkoordinaten



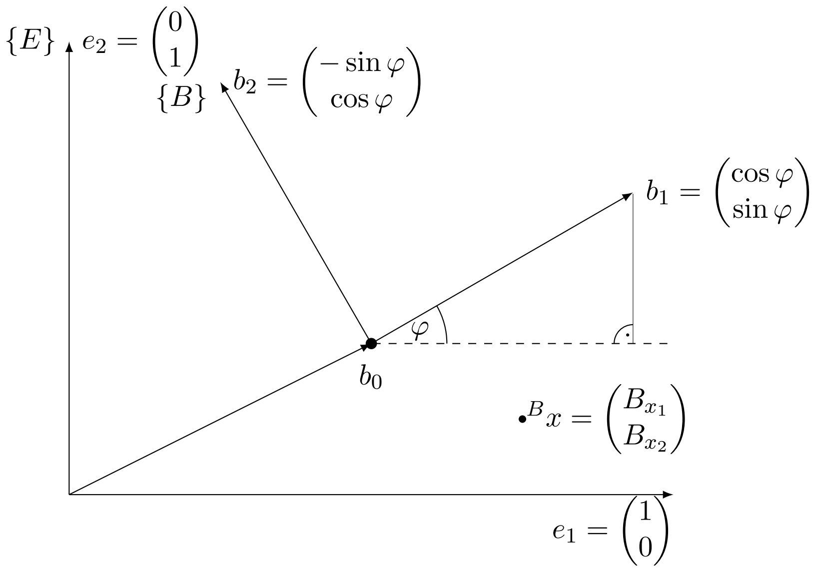
(a) Bildschirmkoordinaten

(b) Weltkoordianten/Normalisiert



$$\boxed{x' = ax + b}$$

$$\boxed{y' = cy + d}$$



$\boldsymbol{b_0}$**:** E-Koordinaten des Ursprungs von System B

$\boldsymbol{b_1, b_2}$ E-Koordinaten der Basisvektoren von System B

$$ {}^Ex = b_0 + {}^Bx_1 \cdot b_1 + {}^Bx_2 \cdot b_2 $$

$$ [b_1, b_2] = R \in \mathbb{R}^{2\times 2}, \quad |b_1| = |b_2| = 1, \quad \overbrace{b_1^T \cdot b-2}^{\text{Standardskalarprodukt}} = 0 $$

$$ R^T \cdot R = \mathbb{E}, \det(R) = 1 \text{ (Rechtssystem)} $$

$$ \Rightarrow {}^Ex = b_0 + {}^ER \cdot {}^Bx $$

1.3 Homogene Koordinaten

$$ {}^Ex^1 = \begin{pmatrix} {}^Ex_1 \\ {}^Ex_2 \\ 1 \end{pmatrix} = \begin{pmatrix} {}^Ex \\ 1 \end{pmatrix} \in \mathbb{R}^3 $$

$$ = \left(\begin{array}{c|c} {}^ER_B & b_0 \\ \hline 0 & 1 \end{array}\right) \begin{pmatrix} B_x \\ 1 \end{pmatrix} $$

$$ \begin{pmatrix} \cos\varphi & -\sin\varphi & b_{0_1} \\ \sin\varphi & \cos\varphi & b_{0_1} \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} {}^Bx_1 \\ {}^Bx_2 \\ 1 \end{pmatrix} $$

Allgemein:

$$ {}^Ex^1 = {}^EM_B \cdot {}^B\hat{x} $$

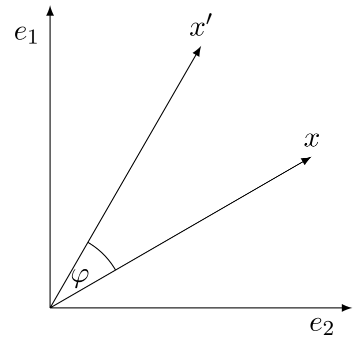
1. ${}^EM_B$ beschreibt die Transformationsmatrix von Koordinaten aus System B in das System E

2. ${}^EM_B$ kann auch interpretiert werden, als die (starre) Transformation, die E in B überführt.

1.4 Transformationen

1.4.1 Rotation

z.B.



$$x' = R \cdot x$$

1.4.2 + Verschiebung

$$x' = Rx + z$$

$\rightsquigarrow \hat{x}' = Mx$ mit $M = \left(\begin{array}{c|c} R & t \\ \hline 0 & 1 \end{array}\right)$

$$x' = R(x + t)$$

1.4.3 Skalierung

$$\underbrace{S = \begin{pmatrix} s_1 & 0 \\ 0 & s_2 \end{pmatrix}}_{\substack{\text{homogenisiert} \\ \hat{S} = \left(\begin{array}{c|c} S & 0 \\ \hline 0 & 1 \end{array}\right)}} \quad x' = Sx = \begin{pmatrix} x_1' = s_1 x_1 \\ x_2' = s_2 x_2 \end{pmatrix}$$

mit $s_2 = -1$ Spiegelung um x_1

1.4.4 Translation

Homogen:

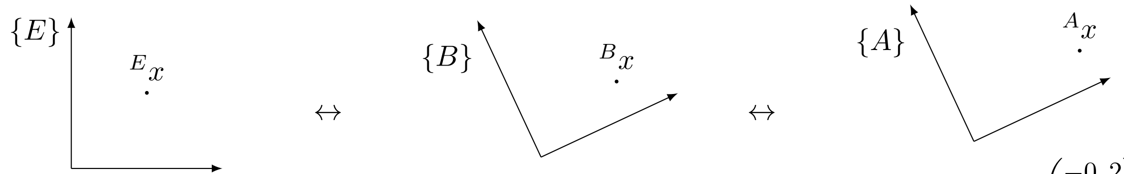
$$\hat{T} = \left(\begin{array}{cc|c} 1 & 0 & t_1 \\ 0 & 1 & t_2 \\ \hline 0 & 0 & 1 \end{array}\right)$$

1.4.5 Hintereinanderausführung von Translation, Rotation und Skalierung

z.B.

$$x''' = S(R(x+t))$$
$$\rightsquigarrow \;\; x' = x+t; \quad x'' = Rx'; \quad x''' = Sx''$$
$$x''' = \underset{\xleftarrow{\text{Leserichtung}}}{\hat{S}\hat{R}\hat{T}\hat{x}} \qquad \text{alles homogenisiert}$$

$\hat{}$ wird oft weggelassen, wenn klar.



(a) Ausgang (b) Rotation um 25° (c) Translation um $\begin{pmatrix} -0,2 \\ -0,1 \end{pmatrix}$

$$^{B}M_{A}\,{}^{A}x = {}^{B}x$$
$$^{E}x = {}^{E}M_{B}\,{}^{B}x$$
$$\Rightarrow {}^{E}x = \underbrace{{}^{E}M_{B}\,{}^{B}M_{A}}_{^{E}M_{A}}\,{}^{A}x$$

z.B.

gegeben

$$^{E}M_{B} \quad {}^{E}M_{A}$$

gesucht

$$^{B}M_{A}$$

$$^{E}M_{A} = {}^{E}M_{B}\,{}^{B}M_{A}$$
$$\Rightarrow {}^{B}M_{A} = {}^{E}M_{B}^{-1} \cdot {}^{E}M_{A}$$
$$^{E}M_{B}^{-1} = {}^{B}M_{E}$$

1.5 Invertierung von M

Sei $M = \left(\begin{array}{c|c} R & t \\ \hline 0 & 1 \end{array}\right)$ $\qquad \underset{\substack{\uparrow \\ \text{Drehung um } -\varphi}}{R^{-1}} = R^T$ u.a. auch, da Rotation $(R^T R = 1)$

$$Mx = Rx + t = x'$$
$$\rightsquigarrow R^T(x'-t) = x$$
$$\rightsquigarrow R^T x' - R^T t$$
$$\rightsquigarrow M^{-1} = \left(\begin{array}{c|c} R^T & -R^T t \\ \hline 0 & 1 \end{array}\right)$$

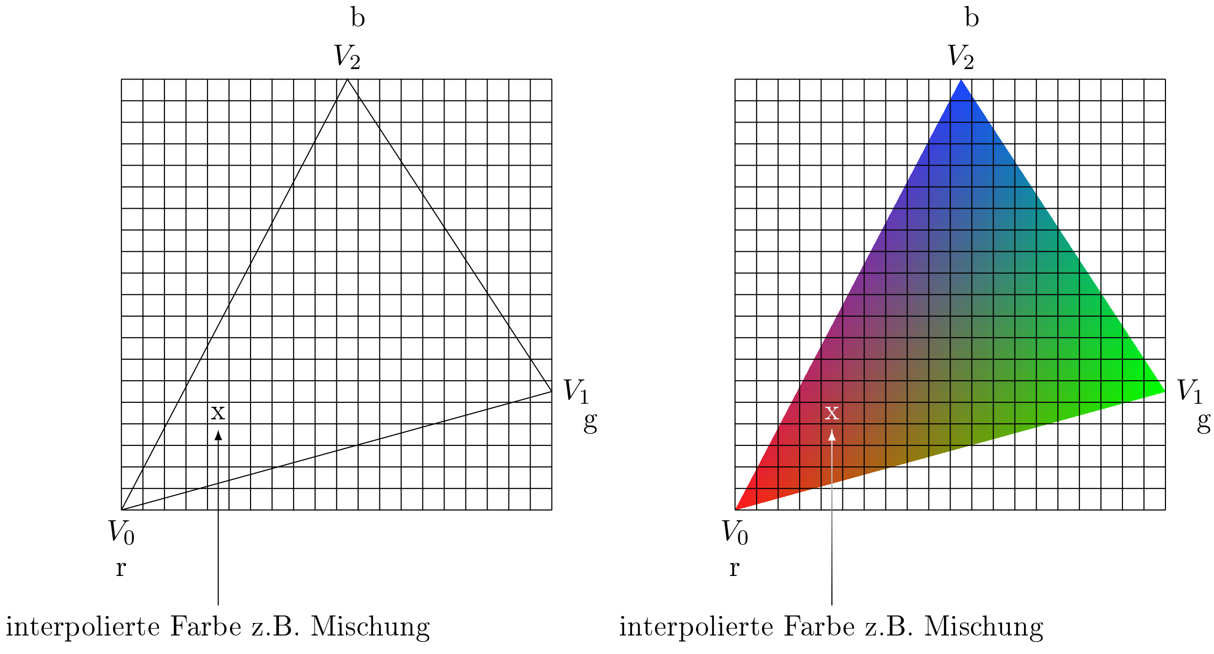
1.6 Qt

```
QMatrix4x4 M;
M.setToIdentity();    // $M = \mathbb{1}$
M.rotate($\varphi$, 0, 0, 1); // Rotation um die z-Achse
                      // $M = M \cdot R$
M.scale($s_1$, $s_2$);       // $M = M \cdot S$
M.translate($t_1$, $t_2$);   // $M = M \cdot T$
// $M = \underbrace{\mathbb{1} \cdot R \cdot S \cdot T}_{\xleftarrow{\text{Leserichtung für Transformation}}}$
// Leter Befehl wird zuerst ausgeführt! LIFO!
Mx'
```

2 VBO

	Kartesische Koord.	Farben	Textur-Koord.	Normal
v_0	x_0, y_0, z_0, w_0	r_0, g_0, b_0	s_0, t_0	u_0, v_0, w'_0
v_1				
$\vdots$				
v_{n-1}				

Entweder alles in einen Buffer $\Rightarrow$ Array of Structs
Oder für jede Klasse einen eigenen Buffer $\Rightarrow$ Struct of Array



(a) Beispiel Raster (b) Beispiel Mischung der Farben

Shader nimmt die Attribute von den Randpunkten und prozessiert diese auf die Pixel im inneren des Dreiecks.

2.1 Baryzentrische Koordinaten

$\boxed{\text{varying}}$ im Vertexshader

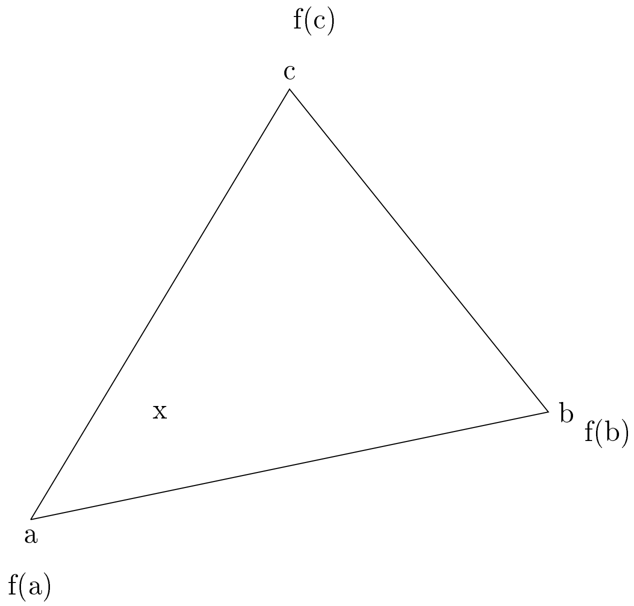


Abbildung 2.2: Baryzentrisches Koordinatensystem

$$\begin{pmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ 1 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 \\ 1 \end{pmatrix}$$

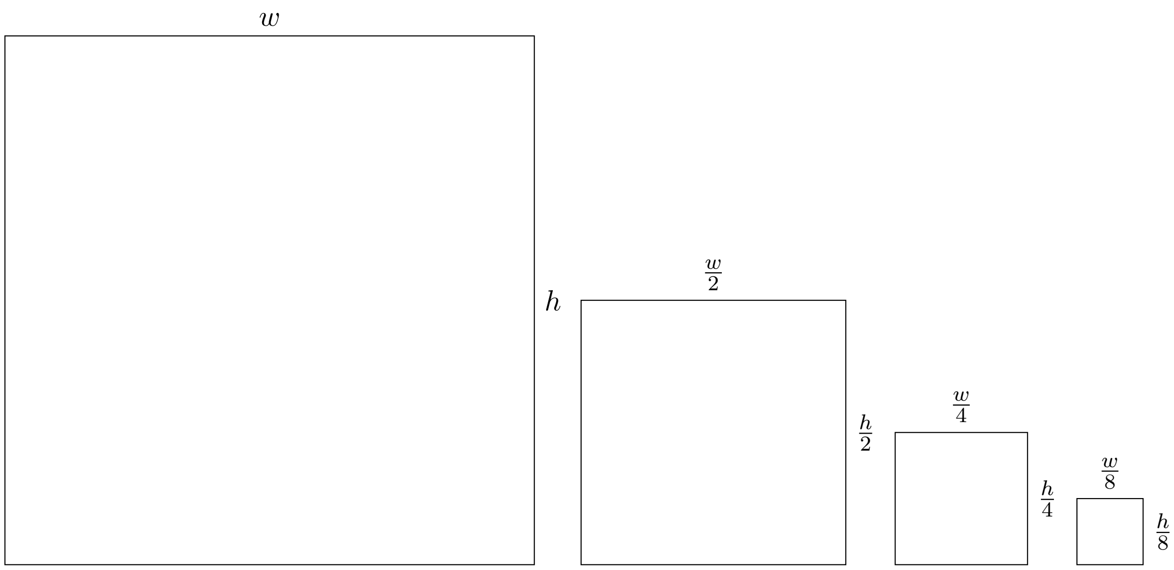
$$x = \alpha \cdot a + \beta \cdot b + \gamma \cdot c \wedge \alpha + \beta + \gamma = 1$$

$$\Rightarrow f(x) = \alpha \cdot f(a) + \beta \cdot f(b) + \gamma \cdot f(c)$$

(wird bei Qt durch das Keyword „Varying“ ausgelöst)

2.2 Texturen

2.2.1 Mipmap



$$S = \sum_{i=0}^{\infty} (\frac{1}{4})^i = \frac{1}{1 - \frac{1}{4}} = \frac{4}{3}$$

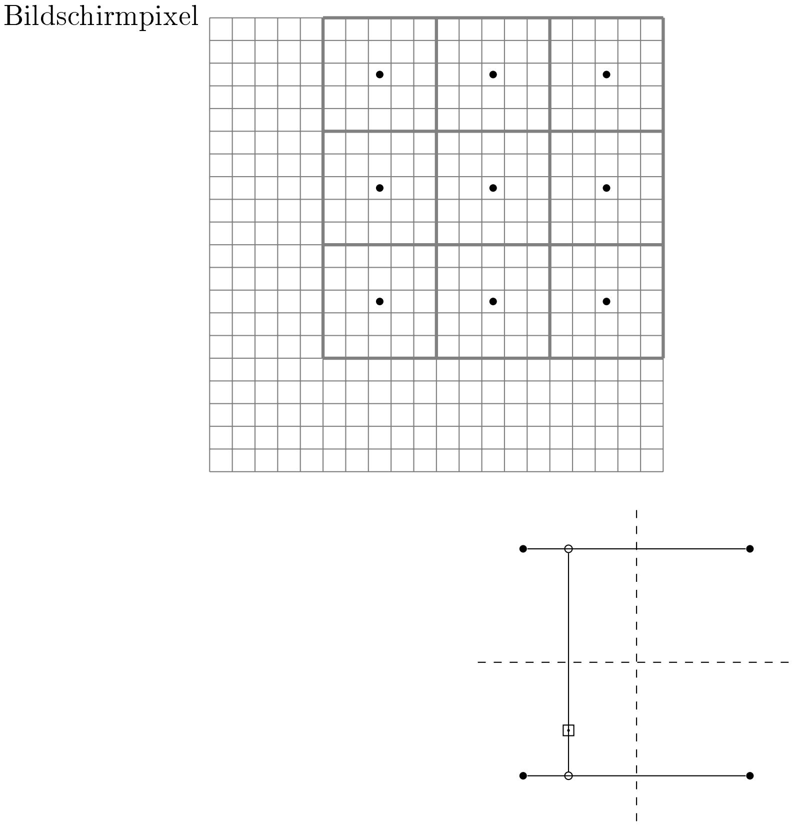
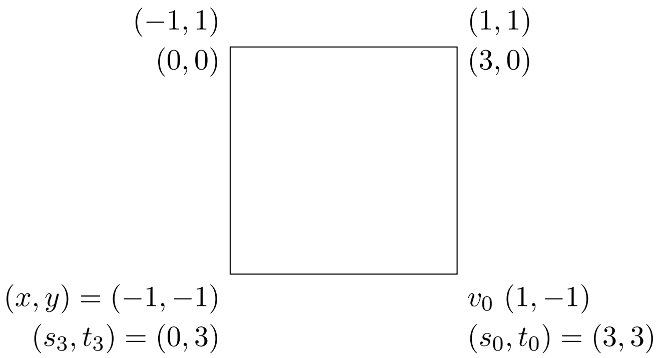


Abbildung 2.3: bilineare Interpolation



In der Regel sind Texturkoordinaten in $[0, 1]^2$, wenn größer wird sie periodisch verwendet.

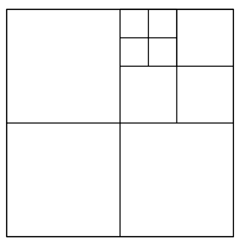
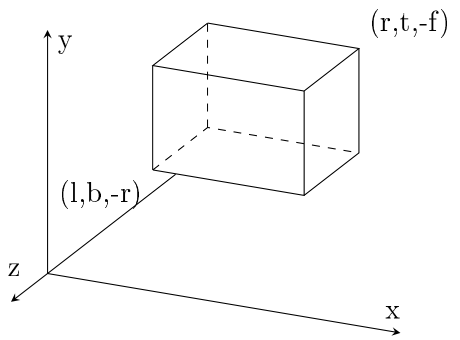


Abbildung 2.4: Maps Mipmapping

Denkanstoß: Welche Kette von Transformationen braucht man um zu einem Fixpunkt zu zoomen?

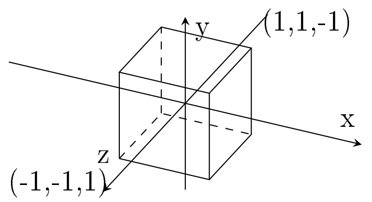
3 3D-Objekte

3.1 Orthogonalprojektion



$$\begin{aligned} x &\in [l, r] \\ y &\in [b, t] \\ z &\in [-f, -n] \end{aligned}$$

Sichtquader $\rightarrow$ Einheitsquader



$$\begin{aligned} x' &\in [-1, 1] \\ y' &\in [-1, 1] \\ z' &\in [-1, 1] \end{aligned}$$

$$\begin{gathered} x' = \alpha \cdot x + \beta \\ l \mapsto -1,\ r \mapsto 1 \end{gathered}$$

$$\begin{array}{rr} (1) & -1 = \alpha \cdot l + \beta \\ (2) & 1 = \alpha \cdot r + \beta \\ \hline (2) - (1) & 2 = \alpha \cdot r - \alpha \cdot l \Rightarrow \alpha = \dfrac{2}{r-l} \\ & 1 = \dfrac{2 \cdot r}{r-l} + \beta \\ & \beta = 1 - \dfrac{2r}{r-l} = \dfrac{r-l-2r}{r-e} = -\dfrac{r+l}{r-l} \end{array}$$

Analog für y' und z'

$$\underbrace{\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix}}_{\text{NDS}} = \underbrace{\begin{pmatrix} \frac{2}{r-l} & 0 & 0 & \frac{r+l}{r-l} \\ 0 & \frac{2}{t-b} & 0 & \frac{t+b}{t-b} \\ 0 & 0 & \frac{-2}{f-n} & -\frac{f+n}{f-n} \\ 0 & 0 & 0 & 1 \end{pmatrix}}_{O} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

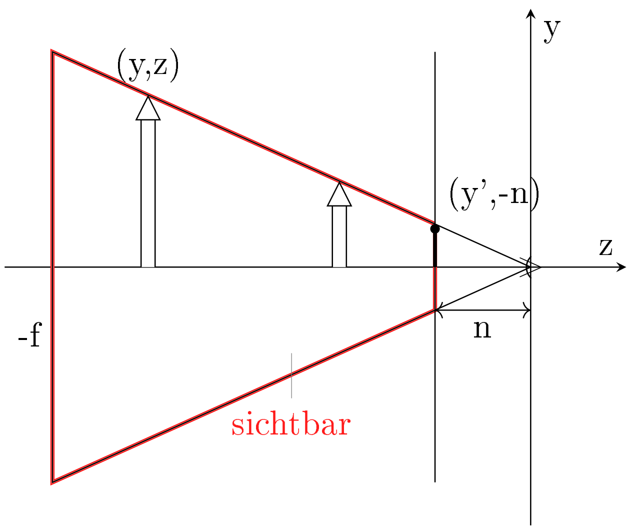
$$z' = -\frac{2}{f-n} z - \frac{f+n}{f-n}$$

$$z = n \quad z* = \frac{2n-(f+n)}{f-n} = \frac{n-f}{f-n} = -1$$

$$-n \mapsto -1, \quad -f \mapsto 1$$

`Qmatrix4x4.ortho(l,n,b,t,n,f);` liefert O

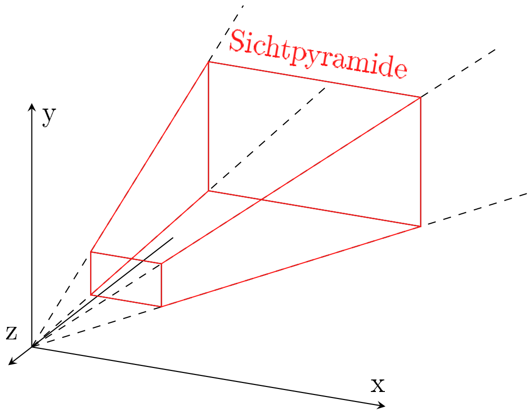
3.2 Perspektivische Projektion



$$\frac{y'}{-n} = \frac{y}{z}$$

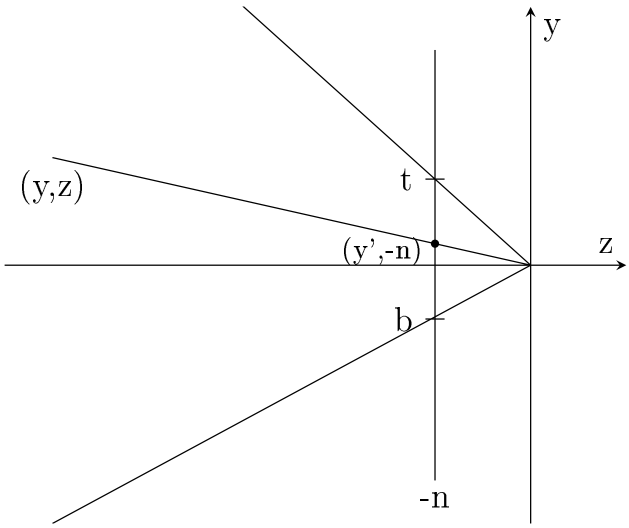
$$y' = -\frac{n \cdot y}{z}$$

Sichtpyramide $\rightarrow$ Einheitswürfel



$$
\begin{aligned}
y' &= -\frac{n \cdot y}{z} \\
&\pitchfork \\
[b,t] &\mapsto [-1,1] \\
y'' &= \alpha \cdot y' + \beta \\
y'' &= \frac{2}{t \cdot b} \cdot y' - \frac{t+b}{t-b} \\
&\boxed{y'' = \frac{2n}{t-b} \cdot \frac{y}{-z} - \frac{t+b}{t-b}}
\end{aligned}
$$

analog für x'', z''



$$
\underbrace{\begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix}}_{\text{homogene Koord.}} \xrightarrow[\text{isierung}]{\text{Dehomogen-}} \underbrace{\begin{pmatrix} \frac{x}{w} \\ \frac{y}{w} \\ \frac{z}{w} \end{pmatrix}}_{\text{Kartesiche koord.}}
$$

Homogenisierungsmatrix:

$$\begin{pmatrix} x'' \\ y'' \\ z'' \\ w'' \end{pmatrix} = \begin{pmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & -\frac{t+b}{t-b} & 0 \\ 0 & 0 & \alpha & \beta \\ 0 & 0 & -1 & 0 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

$$y'' = \frac{2n}{t-b} \cdot y + \frac{t+b}{t-b} \cdot z$$

$$w'' = -z$$

$$\frac{y''}{w''} = \frac{2n}{t-b}\frac{y}{(-z)} + \frac{t+b}{t-b}\frac{z}{(-z)}$$

$$z''' = \frac{z''}{w''} = \frac{\alpha \cdot z + \beta}{-z} = -\alpha - \frac{\beta}{z}$$

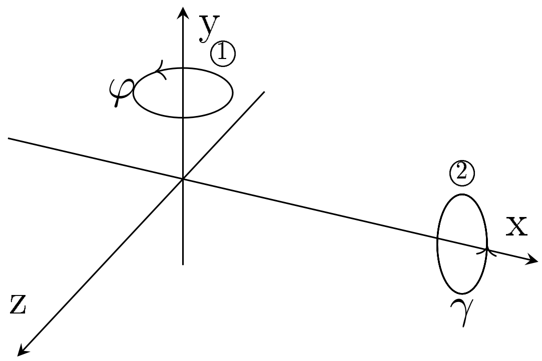
$$-n \mapsto -1, \quad -f \mapsto 15$$

$$\begin{aligned} -\alpha - \frac{\beta}{-n} &= -1 \\ -\alpha - \frac{\beta}{-f} &= 1 \\ -\alpha + \frac{\beta}{n} &= -1(1) \\ -\alpha + \frac{\beta}{f} &= 1(2) \end{aligned}$$

$$\begin{aligned} \frac{\beta}{f} - \frac{\beta}{n} &= 2(2) - (1) \\ \beta\left(\frac{1}{f} - \frac{1}{n}\right) &= 2 \\ \beta\left(\frac{n-f}{fn}\right) & \\ \beta &= \frac{-2nf}{f-n} \\ \alpha = \frac{\beta}{f} - 1 = -\frac{2n-(f-n)}{f-n} &= \frac{f+n}{f-n} \end{aligned}$$

3.3 Objekte drehen aka Blickwinkel ändern

$$p' = \underset{\substack{2 \\ \uparrow \\ \text{Drehung um} \\ \text{die Welt-x-Achse}}}{R_{\vartheta,x}} \cdot \underset{\substack{1 \\ \uparrow \\ \text{Drehung um} \\ \text{die Welt-y-Achse}}}{R_{\varphi,y}} \cdot p$$



3D-Brille

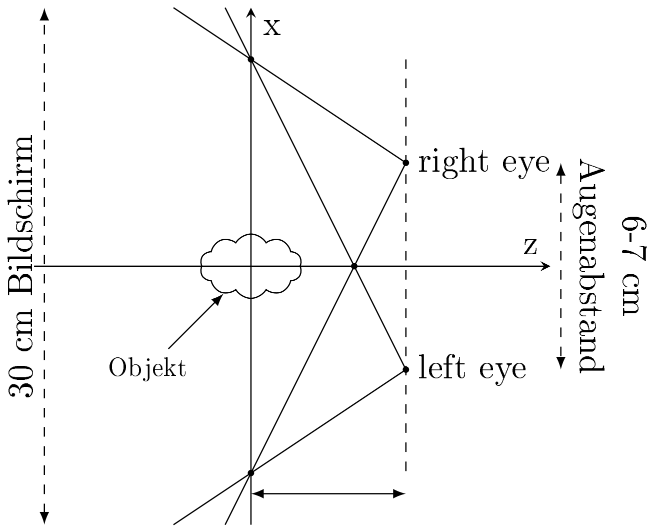
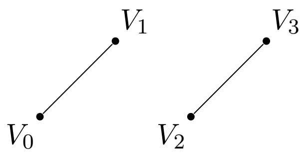
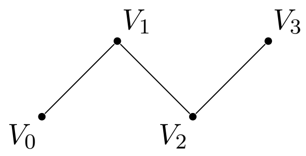


Abbildung 3.1: Streckenangaben sind beispielhaft

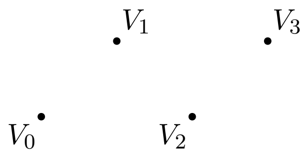
4 Qt Formen



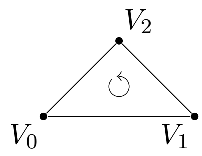
(a) `GL_LINES`



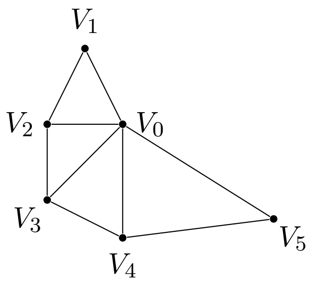
(b) `GL_LINE_STRIP`



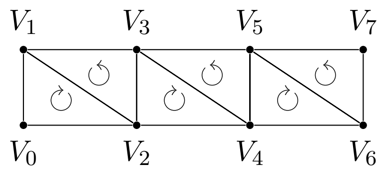
(c) `GL_POINTS`



(d) `GL_TRIANGLE`

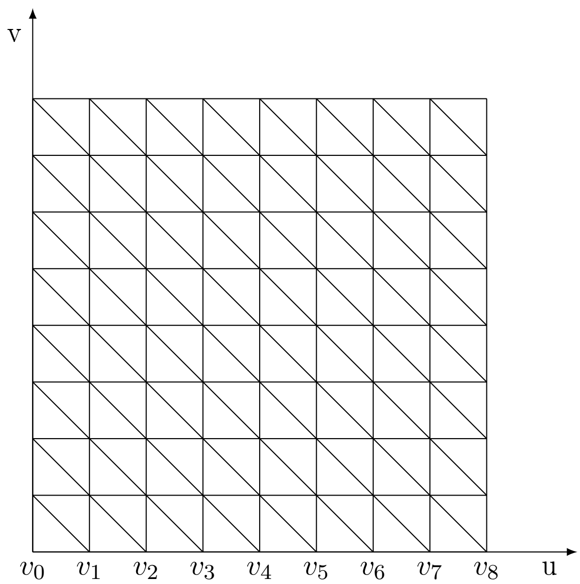


(e) `GL_TRIANGLE_FAN`



(f) `GL_TRIANGLE_STRIP`

Um den `GL_TRIANGLE_STRIP` abzuschließen wird der letzte Knoten (hier V_7) zweimal gesendet.



$$f: \mathbb{R}^2 \to \mathbb{R}^3$$
$$(u, v) \to (x, y, z)$$

Vertex Buffer Objects ← Koordinatenzusatzinformationen

Index Buffer Objects ← Indices der Vertexe

4.1 OFF-Format

n	f	e	
x_0	y_0	z_0	
x_1	y_1	z_1	
	$\vdots$		
3	i_1	i_2	i_3
	$\vdots$		

Index Face Set
`GL_ELEMENT_ARRAY_BUFFER`

5 Beleuchtung

5.1 Ein einfaches Beleuchtungsmodel

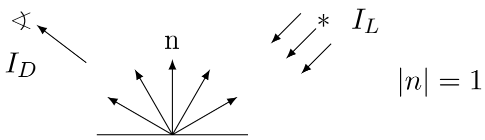
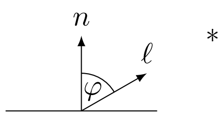


Abbildung 5.1: Diffuse Reflexion

5.1.1 Lambertsches Gesetz

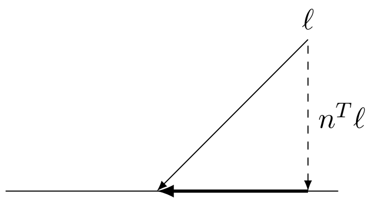


$$I_O = I_L \cdot \ell^T \cdot n$$

$$|n| = |\ell| = 1$$

ℓ zeigt zur Lichtquelle.

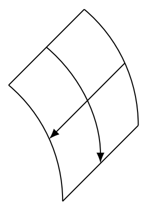
$$\cos(\varphi) = \frac{\ell^T \cdot n}{|n||\ell|} \left(= \frac{< \ell, n >}{|n||\ell|} \right)$$



6 Einschübe

6.1 Berechnung des Normalenvektors für parametrisierte Flächen

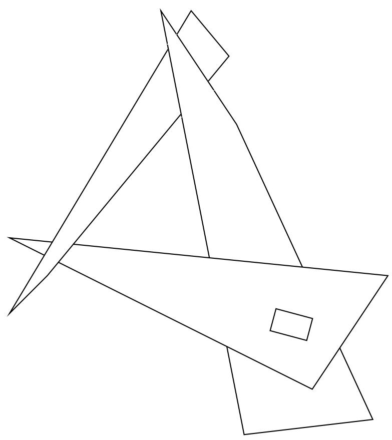
$$(u, v) \to (x, y, z) = f$$



$$\frac{\partial f(u,v)}{\partial u} \times \frac{\partial f(u,v)}{\partial v} = n$$

Im Allgemeinen gilt $|n| \neq 1$

6.2 Tiefenbuffer

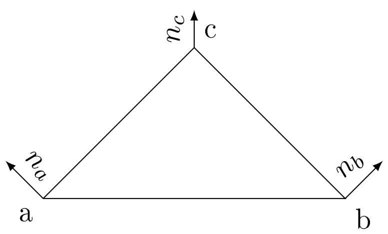


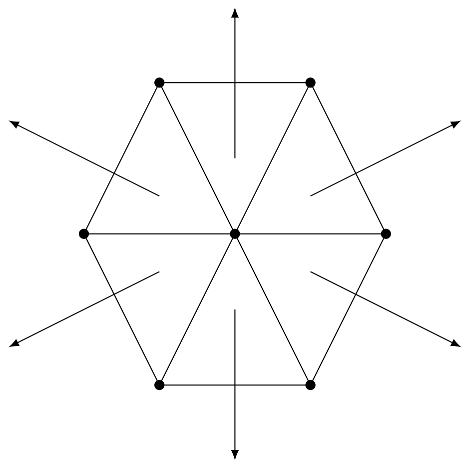
`glEnable(GL_DEPTH_TEST)` - Tiefentest aktivieren.

7 Beleuchtung (Fortsetzung)

7.1 Flatshading

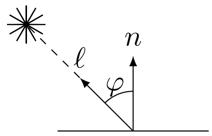
Alle Knoten einer Fläche habe die gleiche Normale
Normale ermitteln + normieren (u.U. Gewichten z.B. Fläche oder Winkel)





Lambert

$$\mathrm{I}_D = \mathrm{I}_L \cdot \left(n^T \cdot \ell\right)$$



$$x' = M \cdot x \qquad E = \{x | n^T x = n_0\} \qquad M \in \mathbb{R}^{3\times 3}$$
$$\{Mx | n^T x = n_0\}$$
$$\text{z.B. } M = R \qquad R^{-1} = R^T \qquad R: \text{ Rotation}$$

$$\text{Transformationsregel} \qquad \begin{aligned} x' &= \mathbb{R}x \\ n' &= \mathbb{R}n \end{aligned}$$

Behauptung

$$\begin{aligned} x' &= Mx \\ n' &= \left(M^{-1}\right)^T n \end{aligned} \qquad \text{für Rotationen } \left(R^{-1}\right)^T = \mathbb{R}$$

Beweis

$$\begin{aligned} n'^T x' = \left(\left(M^{-1}\right)^T n\right)^T Mx &= n^T M^{-1^{T^T}} \cdot Mx = n^T M^{-1} Mx \\ &= n^T x = n_0 \end{aligned}$$

Qt $\left(M^{-1}\right)^T$. `QMatrix4x4::normalMatrix`

8 Einschub: virtual Trackball

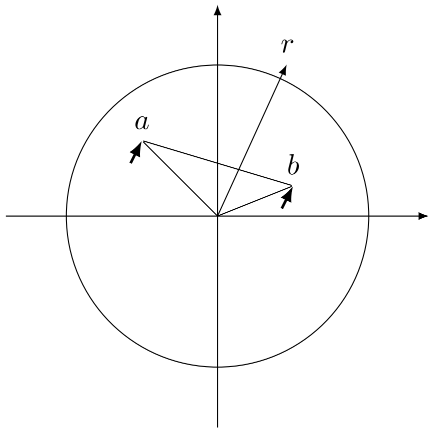


Abbildung 8.1: Virtueller Trackball

$$
\begin{aligned}
a &= \left(x_a, y_a, \sqrt{1 - x_a^2 - y_a^2}\right) \\
b &= \left(x_b, y_b, \sqrt{1 - x_b^2 - y_b^2}\right) \\
r &= a \times b \pm \text{normieren}
\end{aligned}
$$

Bewege a nach b auf einem Großkreis
Normale der Großkreisebene ist $\frac{a \times b}{|a \times b|} = r \widehat{=}$ Rotationsachse

Winkel

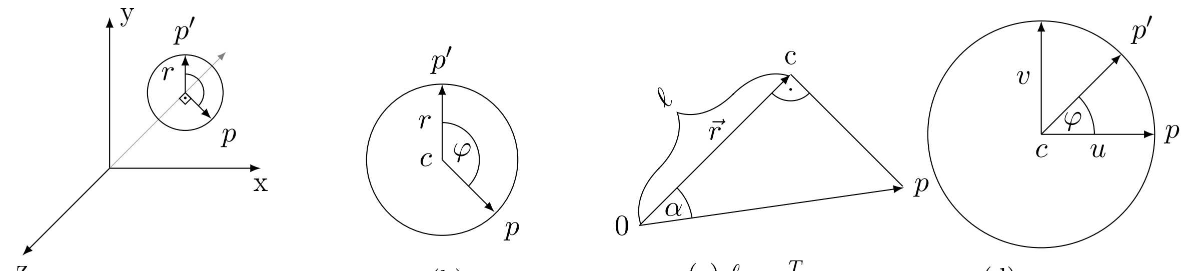
$$\cos \sphericalangle(a, b) = \frac{a \cdot b}{|a \cdot b|}$$

8.1 Formel von Rodriques

$$
\begin{aligned}
\text{Achse, Winkel} &\rightarrow \text{Rotationsmatrix} \\
(r, \varphi) &\rightarrow R \qquad\qquad |r| = 1
\end{aligned}
$$

```
QMatrix4x4 R;
```

$$\texttt{R.rotate(}\underbrace{f}_{\text{deg}}\texttt{,}\,r\texttt{);}$$



(a) $c = \left(r^T p\right) r$ (b) (c) $\ell = r^T p$ (d) $u = p - c$, $v = r \times u$

zu c

$$
\begin{aligned}
\ell &= \cos\alpha \cdot |p| \\
&= \frac{r^T p}{\underbrace{\cancel{|r|}\,\cancel{|p|}}_{=1}} \cancel{|p|}
\end{aligned}
$$

$$
\begin{aligned}
Rp = p' &= c + \cos\varphi u + \sin\varphi v \\
&= c + \cos\begin{pmatrix}1\\0\end{pmatrix} + \sin\varphi\begin{pmatrix}0\\1\end{pmatrix} \\
&= c + \cos\varphi(p-c) + \sin\varphi\,(r \times (p-c)) \\
&= (1-\cos\varphi)c + \cos\varphi p + \sin\varphi(r \times p) && \text{da } r \times c = 0 \\
&= (1-\cos\varphi)\left(rr^t\right)p + \cos\varphi p + \sin\varphi\left(r^x p\right) && (AB)C = A(BC) \\
&\boxed{= (1-\cos\varphi)rr^T + \cos\varphi I + \sin\varphi r^x} && r^T r = \begin{pmatrix} r_1^2 & r_1 r_2 & r_1 r_3 \\ r_2 r_1 & r_2^2 & r_2 r_3 \\ r_3 r_1 & r_3 r_2 & r_3^2 \end{pmatrix}
\end{aligned}
$$

8.1.1 dyadisches Produkt

$$
\left(rr^T\right)^T = r^{T^T} r^T = rr^T \qquad \Rightarrow \text{symmetrisch}
$$

$$
r \times p = Ap
$$

$$
\begin{pmatrix}r_1\\r_2\\r_3\end{pmatrix} \times \begin{pmatrix}p_1\\p_2\\p_3\end{pmatrix} = \begin{pmatrix}r_2p_3 - r_3p_2\\ r_3p_1 - r_1p_3\\ r_1p_2 - r_2p_1\end{pmatrix} = \begin{pmatrix}0 & -r_3 & r_2\\ r_3 & 0 & -r_1\\ -r_2 & r_1 & 0\end{pmatrix}\begin{pmatrix}p_1\\p_2\\p_3\end{pmatrix}
$$

$A =$ Axiator von r
$r^x := A$

Qt

$$
R - e^{\varphi r^x} \qquad e^A = \sum_{n=0}^{\infty} \frac{1}{n!} \cdot A^n
$$

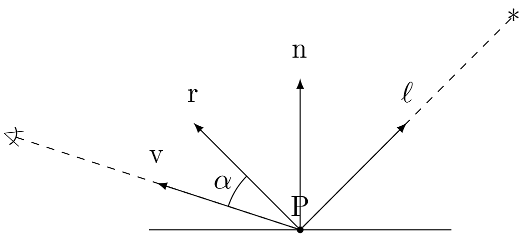
$$
r^{x^T} = -r^x
$$

$$
\mathrm{sch}(A) = \sum_{n=0}^{\infty} \frac{1}{(2n+1)!} A^{2n+1} \cdot (-1)^n
$$

Alternative: Quaternionen

9 Beleuchtung (Fortsetzung)

9.1 Phong Lichtmodell

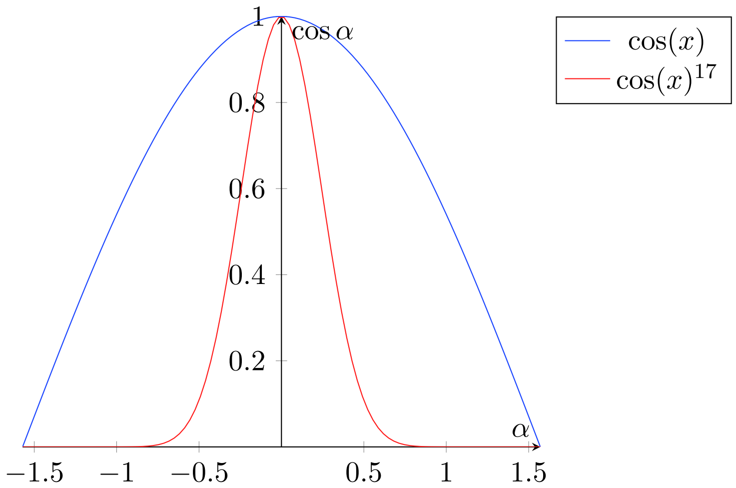


$$|n| = |\ell| = |r| = |v| = 1$$

$$r = 2n(n^T \ell) - \ell$$

S = Shininess

$$I_S = I_L (\cos\alpha)^S = I_L \left(r^T v\right)^S, \quad I_D = I_L \left(n^T \ell\right)$$



9.1.1 Phong

$$I_{\text{Color}} = I_{\text{Ambient,Color}} + I_{\text{Diffuse,Color}} + I_{\text{Specular, Color}}$$

$$\text{Color} \in \{\text{Red}, \text{Green}, \text{Blue}\}$$

```
void main() {
    vec3 normal = normalize(vNormal);
    vec3 lightDir = normalize(lighPos - vPos);
    vec3 reflectDie = reflect(lightDir, normal);
    vec3 viewDir = normalize(-vPos);

    float lambertian = max(dot(loghtDir, normal), 0.0.);
```

```
    float specular = 0.0;

    if ( lambertian > 0.0) {
        float specAngle = max(dot(reflectDir, viewDir), 0.0);
        specular = pow(specAngle, uShininess);
    }
    gl_FragColor = vec4(uAmbient + lambertian * uDiffuse + specular * uSpecular, 1.0);
}
```

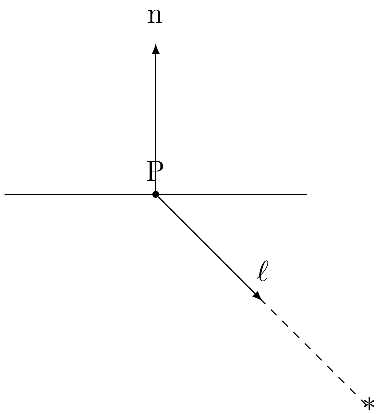
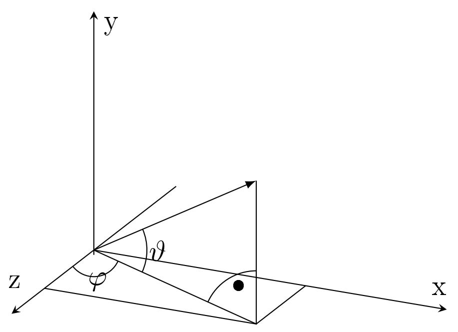


Abbildung 9.1: Zu ignorierende Lichtquelle

10 Oberflächen

10.1 Texturen



$$(\varphi, \vartheta) \mapsto \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} \cos\vartheta \cdot \sin\varphi \\ \sin\vartheta \\ \cos\vartheta \cdot \cos\varphi \end{pmatrix}$$

$$0 \leq \varphi \leq 2\pi$$

$$-\frac{\pi}{2} \leq \vartheta \leq \frac{\pi}{2}$$

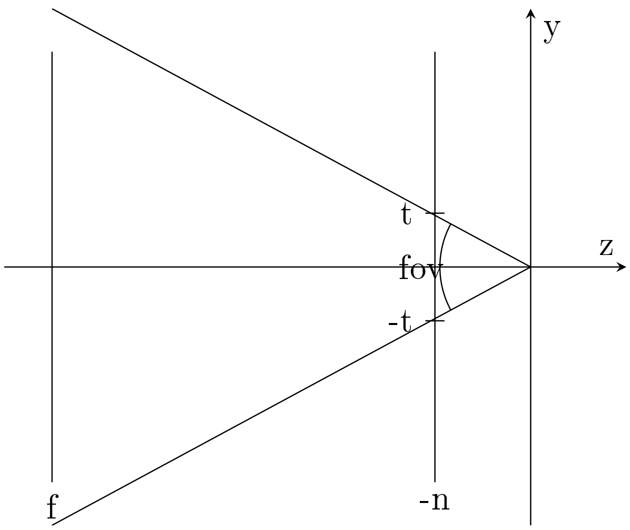
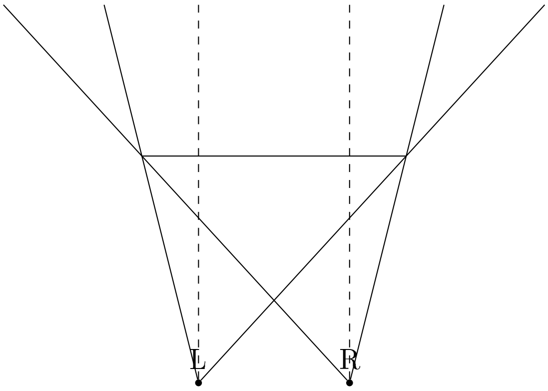
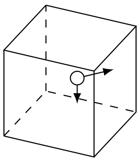


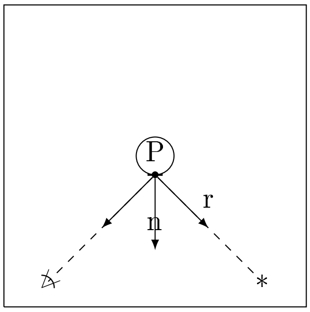
Abbildung 10.1: Field of view

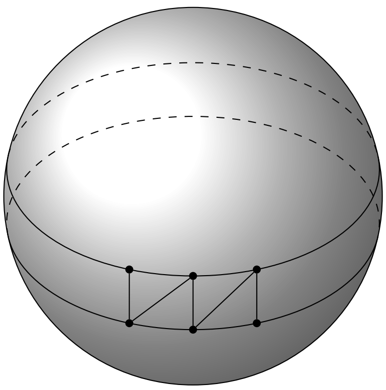
```
perspective(fov, aspectratio, n, f);
```



10.2 Cube-Mapping

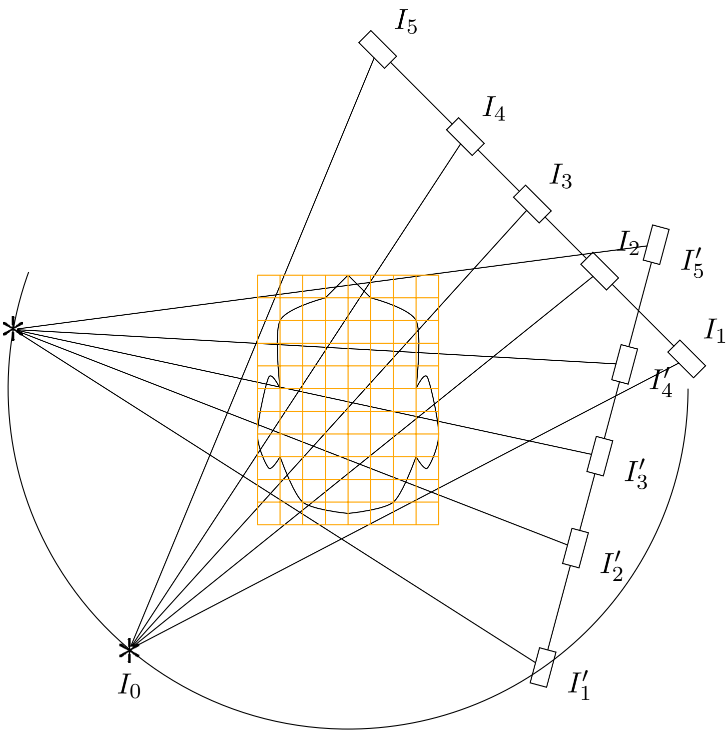


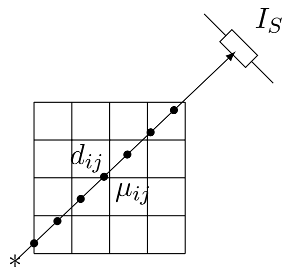




11 Volume Rendering mit 3D-Texturen

11.1 DVT





11.1.1 Lambert-Beer-Gesetz

$$I^{\text{out}} = I^{\text{in}} \cdot e^{-\mu \cdot d}$$

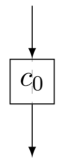
Schwächungskoeffizienten

$$\begin{aligned} I_S &= I_0 \cdot e^{\sum_{\square(i,j)\cap S \neq \emptyset} \mu_{ij} \cdot d_{ij}} \\ \ln \frac{I_0}{I_S} &= \sum_{\square(i,j)\cap S \neq \emptyset}^{\mu_{ij} \cdot d_{ij}} \\ I &= \underset{\mathbb{R}^{360.000.000 \times 128.000.000}}{\underset{\Cap}{D}} \cdot \mu \end{aligned}$$

Inverses Problem

Gegeben: Gemessene Intensitäten I_S für alle Strahlen, die die Bildebene treffen für hinreichend viele Aufnahmerichtungen.

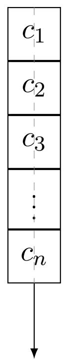
Gesucht: Schwächungskoeffizienten für alle Voxel des zu rekonstruierenden Volumens.



$$c^{\text{out}} = c^{\text{in}} \cdot (1 - \alpha_i) + c_i \alpha_i$$

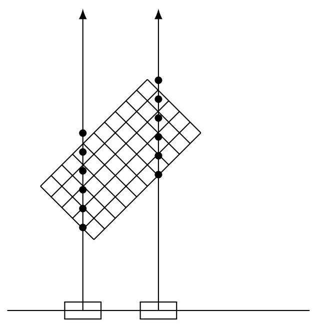
$\boldsymbol{\alpha_i} =$ Deckkraft der Farbe c_i

$\mathbf{1} - \boldsymbol{\alpha_i} \mathrel{\hat{=}}$ Transparenz



$$c^{\text{out}} = \sum_{i=1}^{n} c_i \alpha_i \prod_{j=i+1}^{n} (1 - \alpha_j)$$

11.2 Raycasting mittels front-to-back rendering

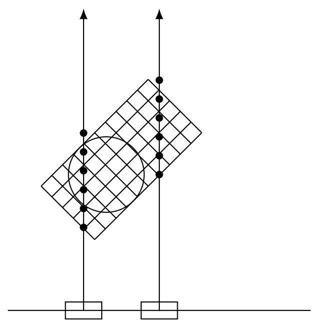


Auf der Sichtlinie werden in regelmäßigen Abständen im Körper Messpunkte gesetzt. Für jeden Messpunkt wird nun, wie vorher beschrieben, die Farbe und Leuchtkraft bestimmt. Anschließend werden diese zu einer Farbe für den entsprechenden Pixel auf dem Bildschirm zusammengefasst. Wir beginnen dabei mit dem entferntesten und enden mit dem Messpunkt am nähesten. Dadurch werden Voxel, welche hinter „soliden“ Voxeln liegen in der Farbwahl nicht betrachtet.

12 Einschub: Algebraische Flächen

$$f(x, y, z) = 0$$

$$\text{z.B.} \quad \left\{(x, y, z) \,\middle|\, x^2 + y^2 + z^2 - 1 = 0\right\}$$



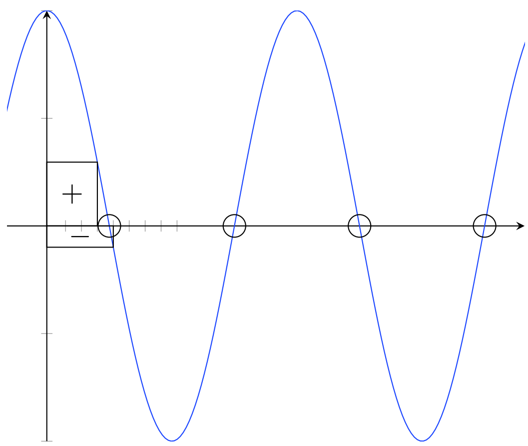


Abbildung 12.1: Finden von Nullstellen durch Suche nach Vorzeichenwechsel

Wie beim Raycasting wird der Bereich in Abschnitte unterteilt und diese einzeln untersucht, folgt auf ein + ein − (oder umgekehrt), liegt dazwischen eine Nullstelle.

$$p \quad \nabla f(p) = n$$