Computergrafik

Mitschrift von

Markus Vieth Steffen Eiden Lukas Birklein

9. Januar 2017

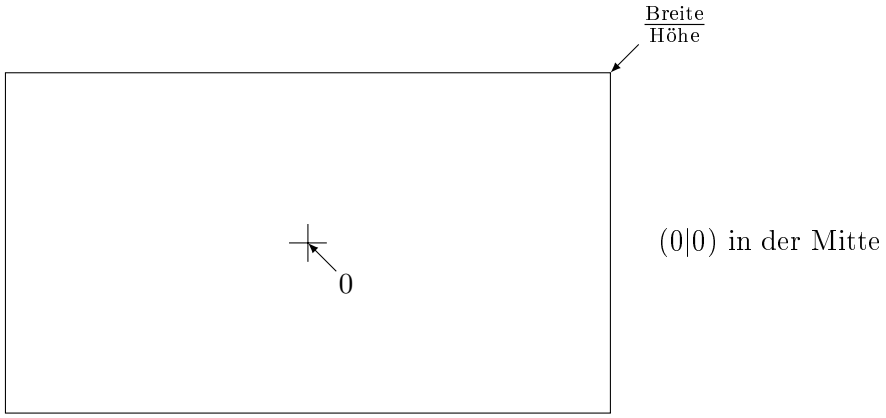
Vorwort

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

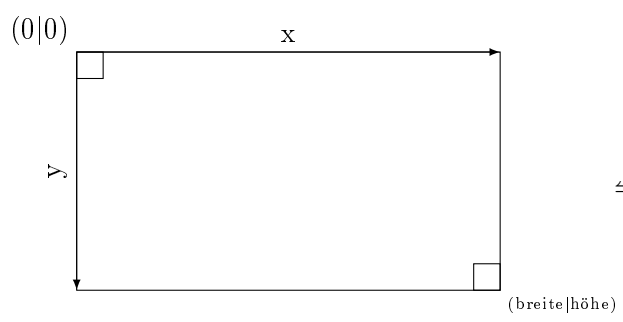
1 Koordinatensysteme

1.1 Normalisiertes Koordinatensystem

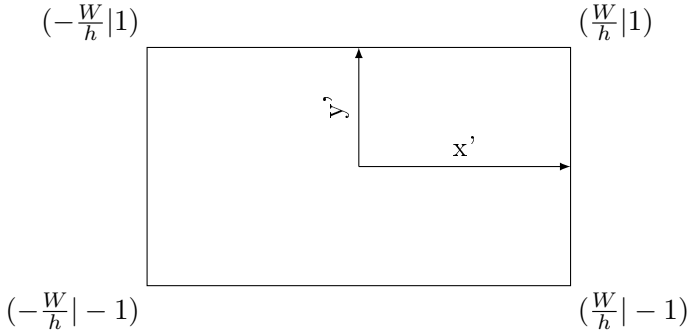


`glViewport`: Ausschnitt wo gezeichnet wird.

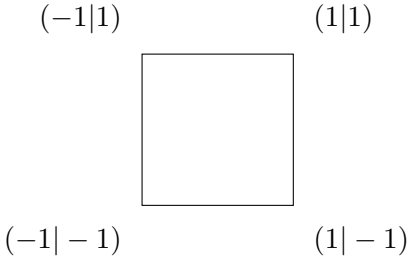
1.2 Bildschirmkoordinaten $\leftrightharpoons$ Weltkoordinaten



(a) Bildschirmkoordinaten

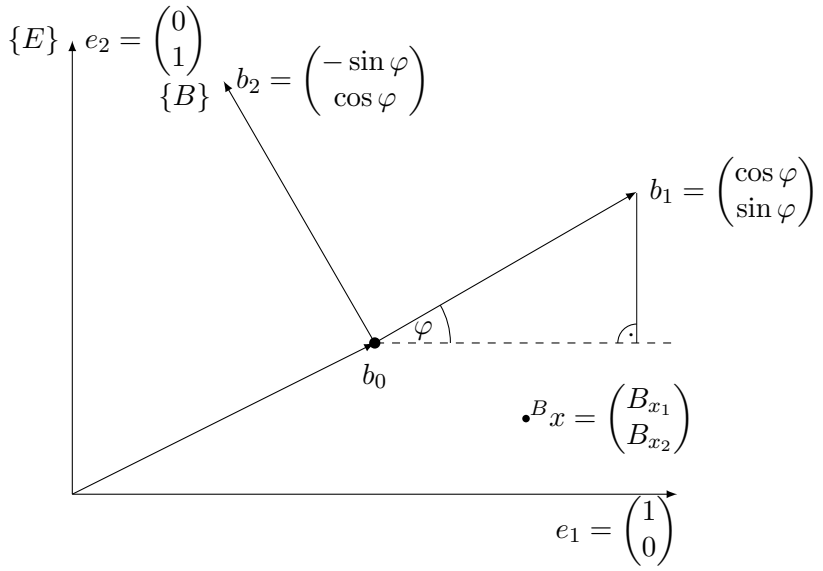


(b) Weltkoordianten/Normalisiert



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

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



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

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

$$^{E}x = b_0 + {}^{B}x_1 \cdot b_1 + {}^{B}x_2 \cdot b_2$$

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

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

$$\Rightarrow {}^{E}x = b_0 + {}^{E}R \cdot {}^{B}x$$

1.3 Homogene Koordinaten

$$^{E}x^1 = \begin{pmatrix} {}^{E}x_1 \\ {}^{E}x_2 \\ 1 \end{pmatrix} = \begin{pmatrix} {}^{E}x \\ 1 \end{pmatrix} \in \mathbb{R}^3$$

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

$$\begin{pmatrix} \cos\varphi & -\sin\varphi & b_{0_1} \\ \sin\varphi & \cos\varphi & b_{0_1} \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} {}^{B}x_1 \\ {}^{B}x_2 \\ 1 \end{pmatrix}$$

Allgemein:

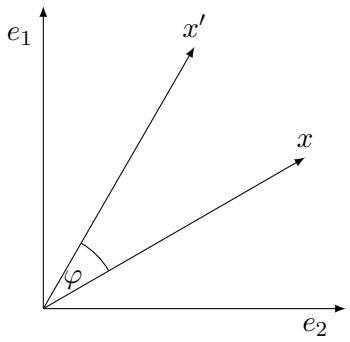
$$^{E}x^1 = {}^{E}M_B \cdot {}^{B}\hat{x}$$

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

$$\underbrace{S = \begin{pmatrix} s_1 & 0 \\ 0 & s_2 \end{pmatrix}}_{\text{homogenisiert}} \quad x' = Sx = \begin{pmatrix} x_1' = s_1 x_1 \\ x_2' = s_2 x_2 \end{pmatrix}$$

$$\hat{S} = \left(\begin{array}{c|c} S & 0 \\ \hline 0 & 1 \end{array}\right)$$

mit $s_2 = -1$ Spiegelung um x_1

1.4.4 Translation

Homogen:

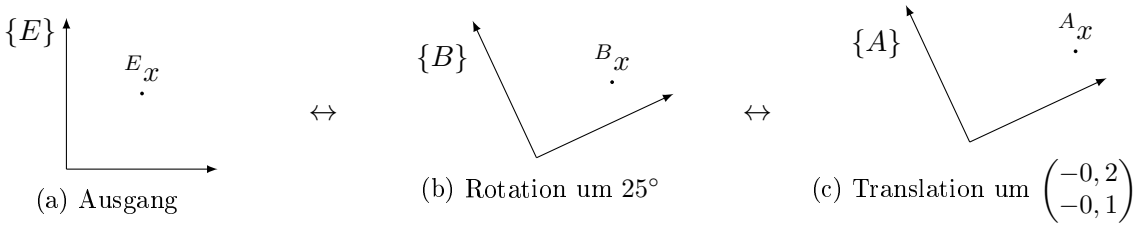
$$\hat{T} = \left(\begin{array}{cc|c} 1 & 0 & t_1 \\ 0 & 1 & t_2 \\ \hline 0 & 0 & 1 \end{array}\right)$$

1.4.5 Hintereinanderausführung von Translation, Rotation und Skalierung

z.B.

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

$\hat{}$ 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

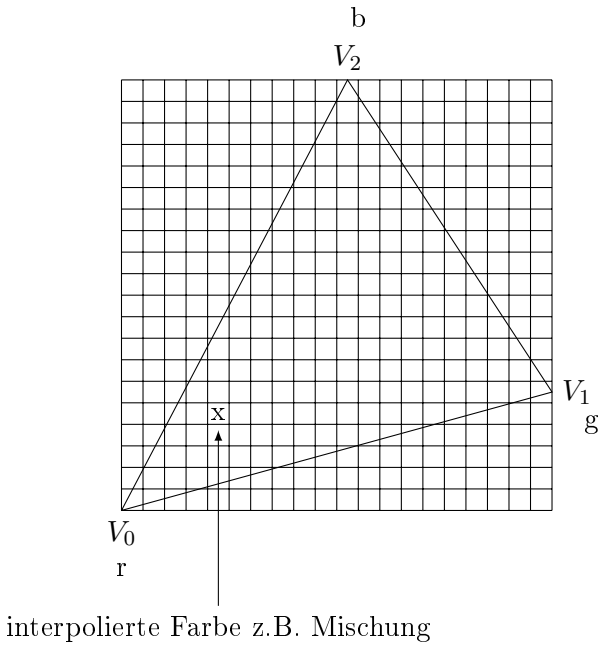
```
1 QMatrix4x4 M;
2 M.setToIdentity();    // M = 𝟙
3 M.rotate(φ, 0, 0, 1); // Rotation um die z-Achse
4                       // M = M · R
5 M.scale(s_1, s_2);    // M = M · S
6 M.translate(t_1, t_2); // M = M · T
7 // M = 𝟙 · R · S · T
8 //      ←
9 // Leter Befehl wird zuerst ausgeführt! LIFO!
  Mx'
```

Leserichtung für Transformation

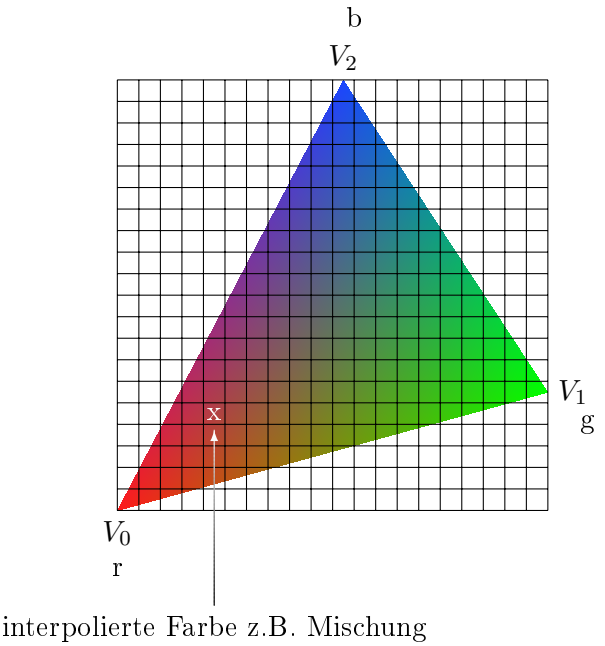
2 VBO

	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 ⇒ Array of Structs
Oder für jede Klasse einen eigenen Buffer ⇒ 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

varying im Vertexshader

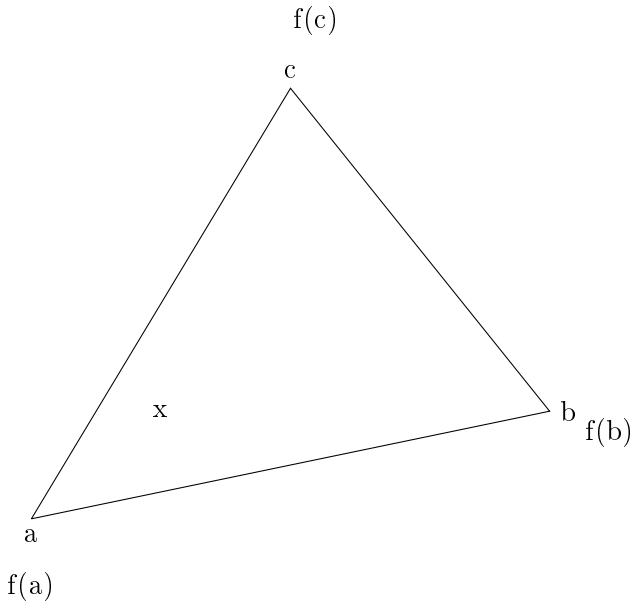


Abbildung 2.2: Baryzentrisches Koordinatensystem

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

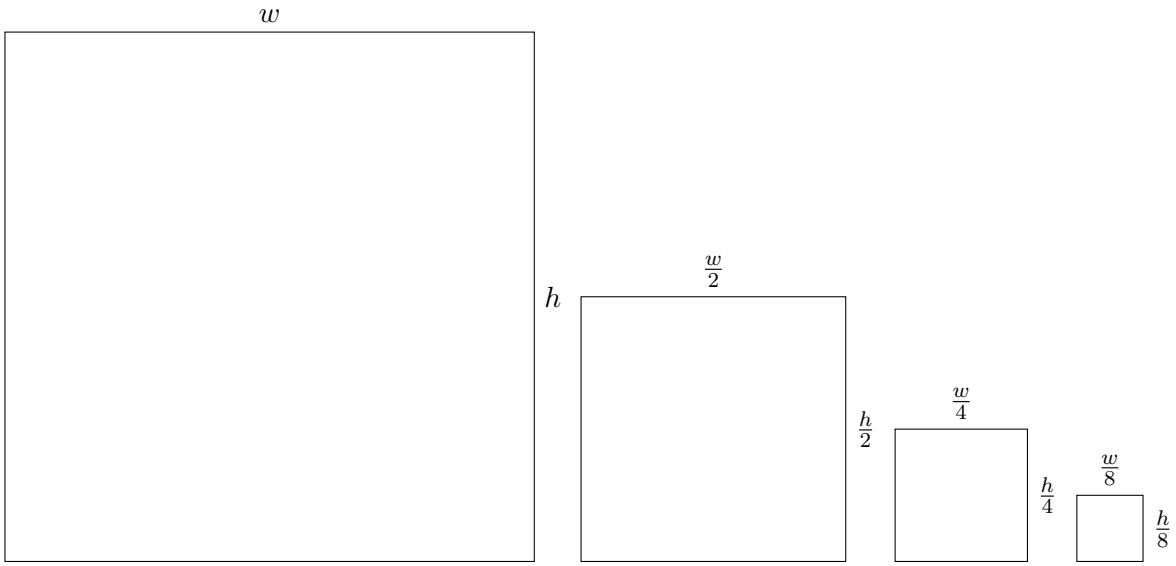
$$x = \alpha \cdot a + \beta \cdot b + \gamma \cdot c \wedge \alpha + \beta + \gamma = 1$$

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

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

2.2 Texturen

2.2.1 Mipmap



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

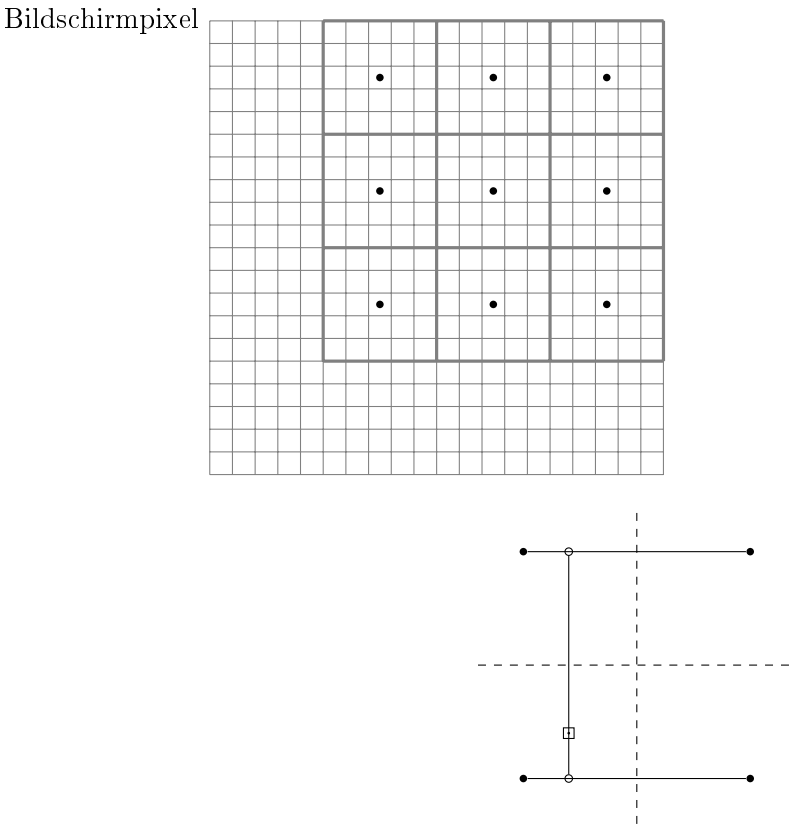
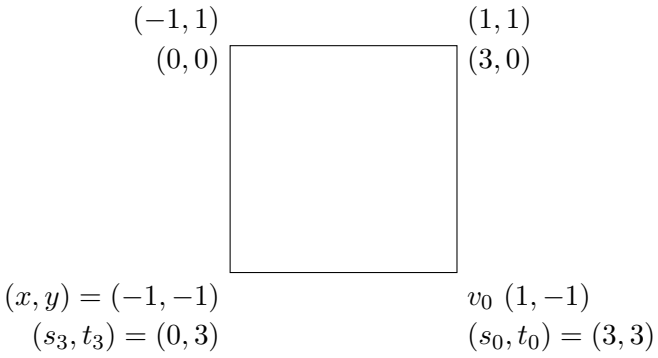


Abbildung 2.3: bilineare Interpolation



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

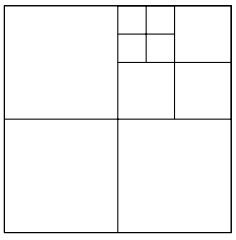
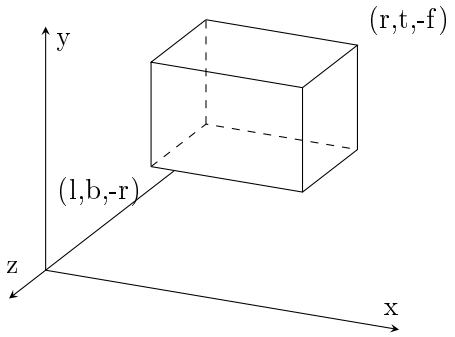


Abbildung 2.4: Maps Mipmapping

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

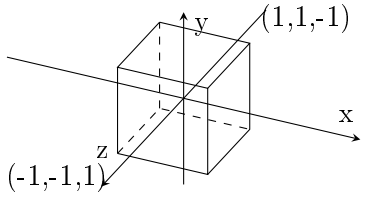
3 3D-Objekte

3.1 Orthogonalprojektion



$$x \in [l, r]$$
$$y \in [b, t]$$
$$z \in [-f, -n]$$

Sichtquader $\rightarrow$ Einheitsquader



$$x' \in [-1, 1]$$
$$y' \in [-1, 1]$$
$$z' \in [-1, 1]$$

$$x' = \alpha \cdot x + \beta$$
$$l \mapsto -1,\ r \mapsto 1$$

$$(1) \qquad -1 = \alpha \cdot l + \beta$$
$$(2) \qquad 1 = \alpha \cdot r + \beta$$
$$(2) - (1) \qquad 2 = \alpha \cdot r - \alpha \cdot l \Rightarrow \alpha = \frac{2}{r - l}$$
$$1 = \frac{2 \cdot r}{r - l} + \beta$$
$$\beta = 1 - \frac{2r}{r - l} = \frac{r - l - 2r}{r - e} = -\frac{r + l}{r - l}$$

Analog für y' und z'

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

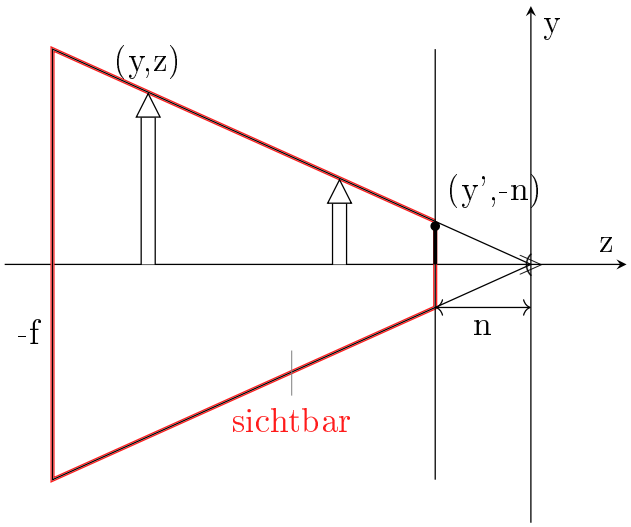
$$z' = -\frac{2}{f-n}z - \frac{f+n}{f-n}$$

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

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

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

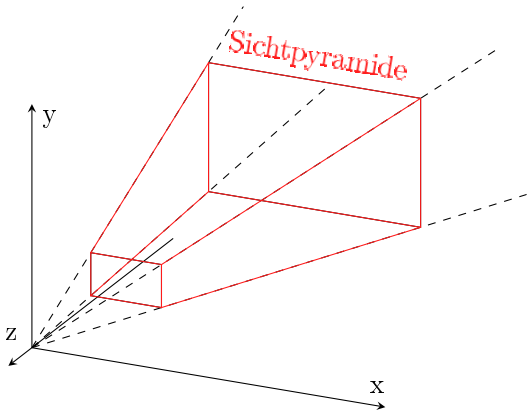
3.2 Perspektivische Projektion



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

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

Sichtpyramide → Einheitswürfel



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

$$\cap$$

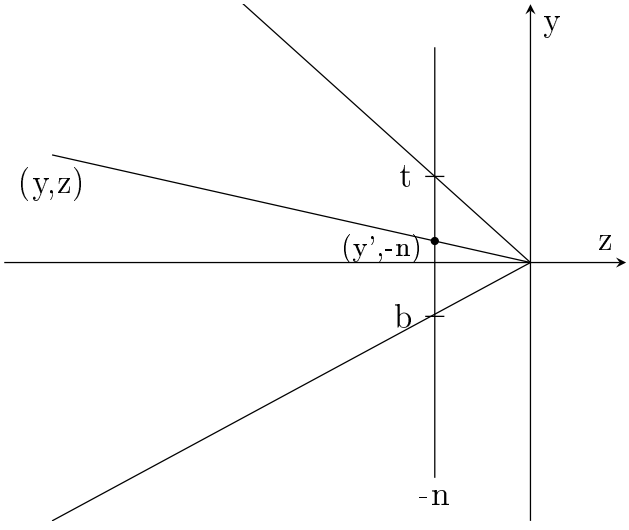
$$[b,t] \mapsto [-1,1]$$

$$y'' = \alpha \cdot y' + \beta$$

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

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

analog für x'', z''



$$\begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix} \xrightarrow{\text{Dehomogen-isierung}} \begin{pmatrix} \frac{x}{w} \\ \frac{y}{w} \\ \frac{z}{w} \end{pmatrix}$$

homogene Koord. — Kartesiche koord.

Homogenisierungsmatrix:

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

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

$$w'' = -z$$

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

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

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

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

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

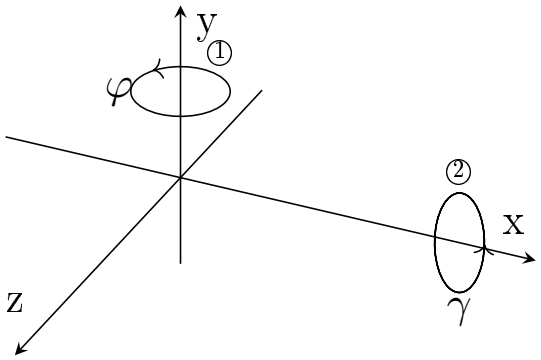
$$\alpha = \frac{\beta}{f} - 1 = -\frac{2n-(f-n)}{f-n} = \frac{f+n}{f-n}$$

3.3 Objekte drehen aka Blickwinkel ändern

$$p' = \underbrace{R_{\vartheta,x}}_{\text{②}} \cdot \underbrace{R_{\varphi,y}}_{\text{①}} \cdot p$$

$R_{\vartheta,x}$: ② ↑ Drehung um die Welt-x-Achse

$R_{\varphi,y}$: ① ↑ Drehung um die Welt-y-Achse



3D-Brille

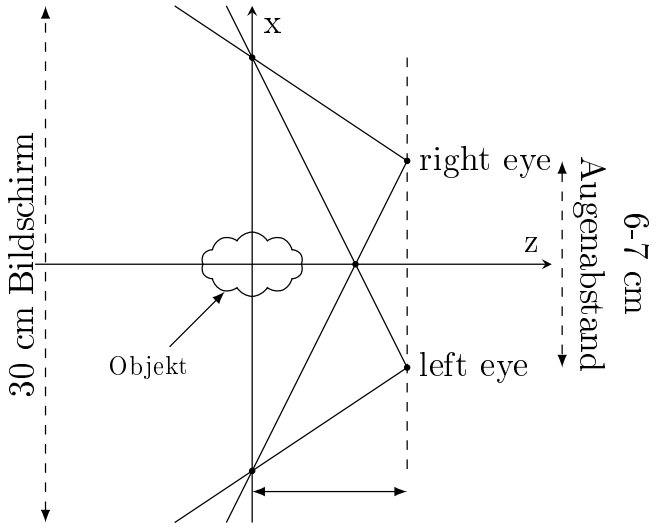
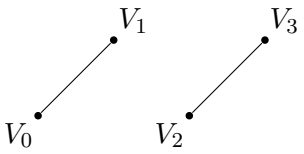
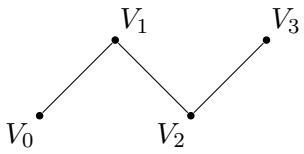


Abbildung 3.1: Streckenangaben sind beispielhaft

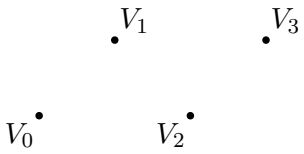
4 Qt Formen



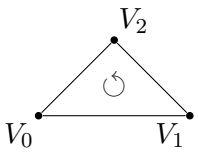
(a) GL_LINES



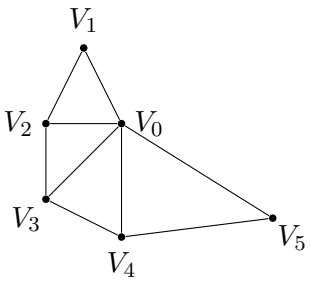
(b) GL_LINE_STRIP



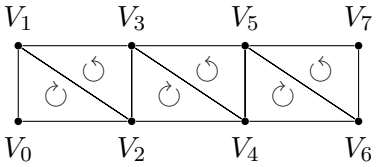
(c) GL_POINTS



(d) GL_TRIANGLE



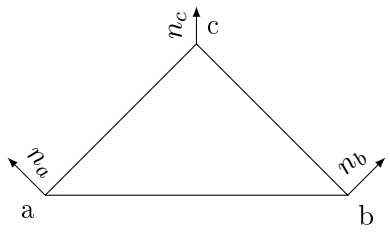
(e) GL_TRIANGLE_FAN

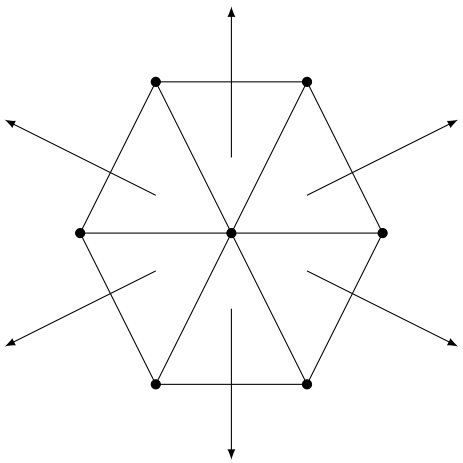


(f) GL_TRIANGLE_STRIP

5 Beleuchtung

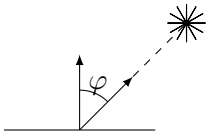
5.0.1 Smoothing



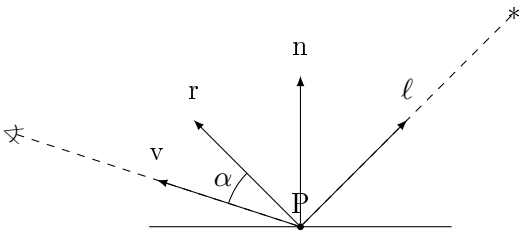


Lambert

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



5.1 Phong Lichtmodell

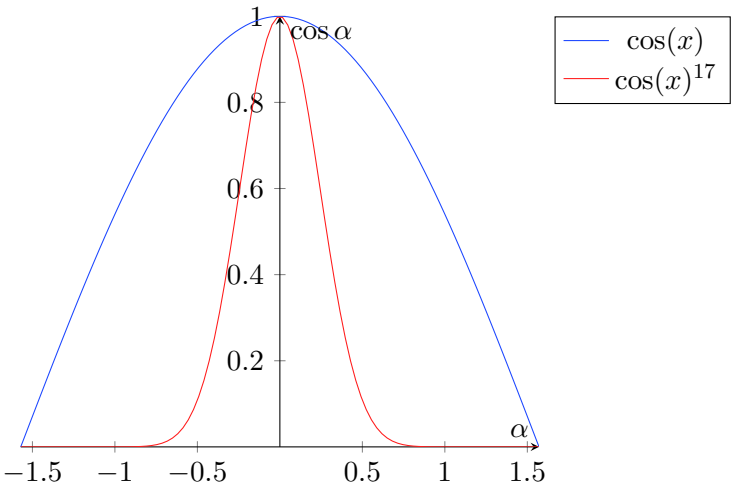


$$|n| = |\ell| = |r| = |v| = 1$$

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

S = Shininess

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



5.1.1 Phong

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

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

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

7      float lambertian = max(dot(loghtDir, normal), 0.0.);
8      float specular = 0.0;

10     if ( lambertian > 0.0) {
11         float specAngle = max(dot(reflectDir, viewDir), 0.0);
12         specular = pow(specAngle, uShininess);
13     }
14     gl_FragColor = vec4(uAmbient + lambertian * uDiffuse + specular * uSpecular, 1.0);
15 }
```

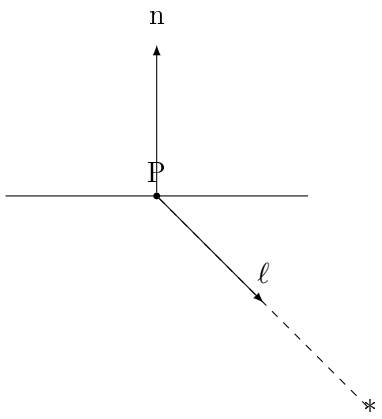
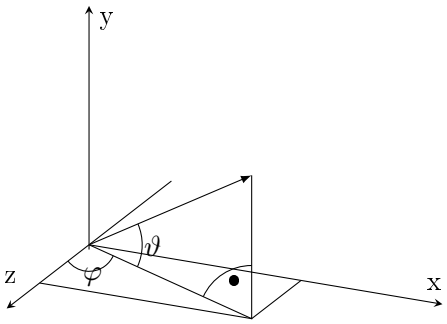


Abbildung 5.2: Zu ignorierende Lichtquelle

6 Oberflächen

6.1 Texturen



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

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

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

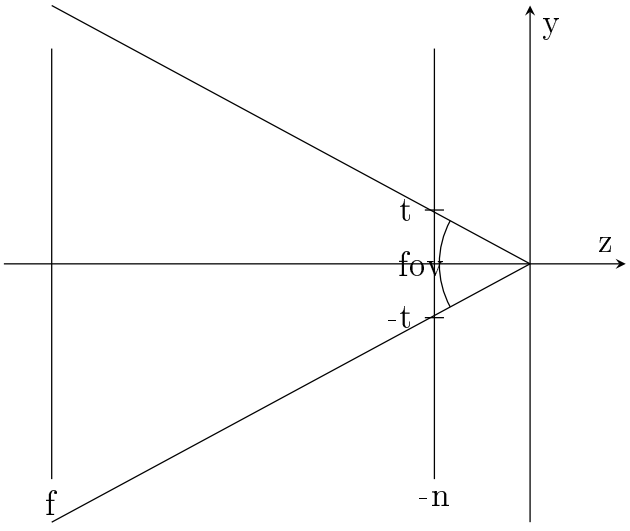
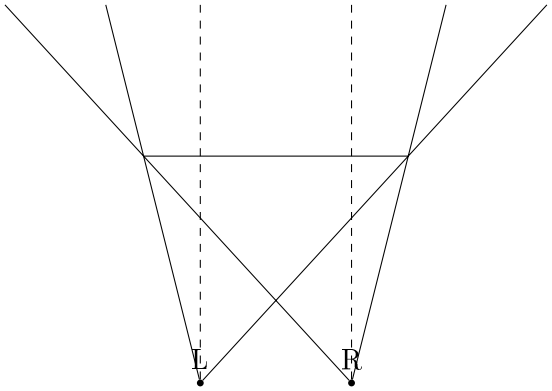
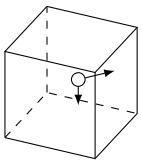


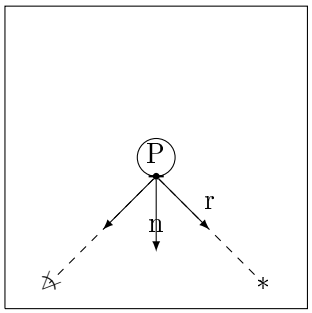
Abbildung 6.1: Field of view

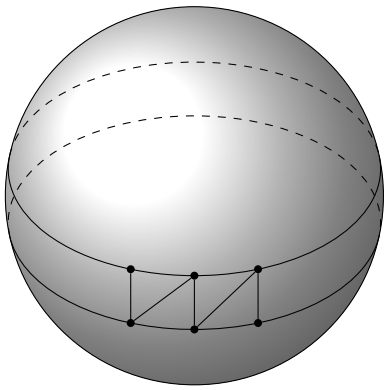
```
perspective(fov, aspectratio, n, f);
```



6.2 Cube-Mapping

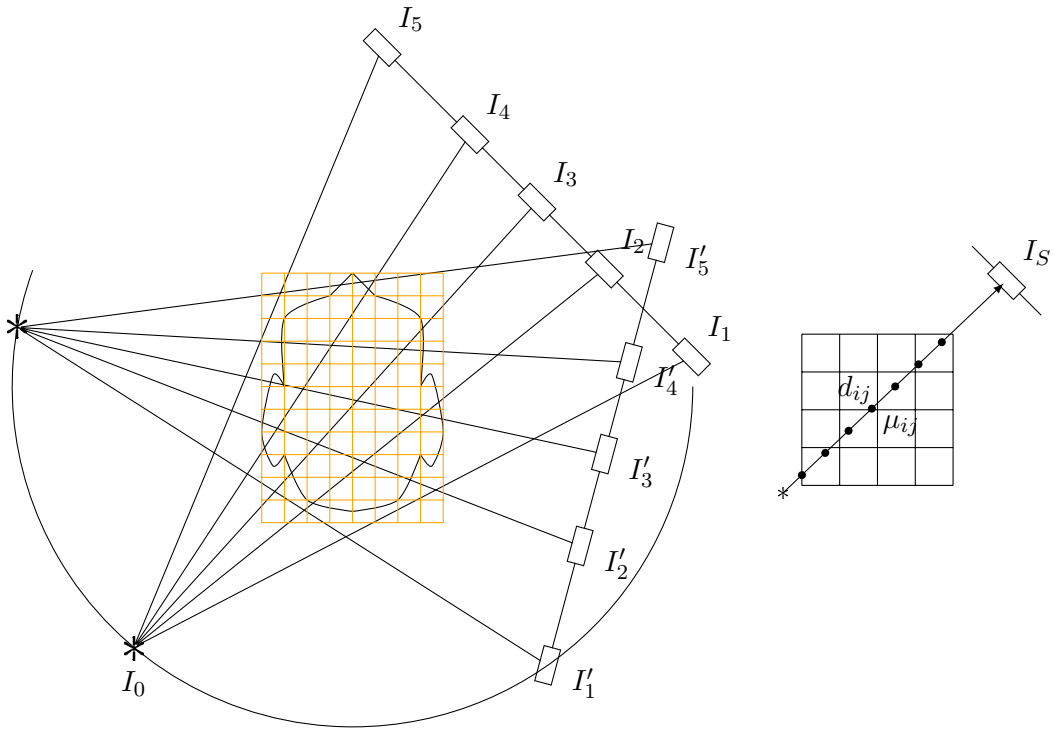






7 Volume Rendering mit 3d-Texturen

7.1 DVT



7.1.1 Lambert-Beer-Gesetz

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

Schwächungskoeffizienten

$$I_S = I_0 \cdot e^{\sum_{\square(i,j)\cap S \neq \emptyset} \mu_{ij} \cdot d_{ij}}$$

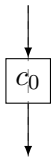
$$\ln \frac{I_0}{I_S} = \sum_{\square(i,j)\cap S \neq \emptyset}^{\mu_{ij} \cdot d_{ij}}$$

$$I = \underset{\mathbb{R}^{360.000.000 \times 128.000.000}}{\underset{\cap}{D}} \cdot \mu$$

Inverses Problem

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

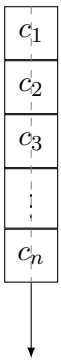
Gesucht: Schwächungskoeffizienten für alle Voxel des zu rekonstruierenden Volumens.



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

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

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



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