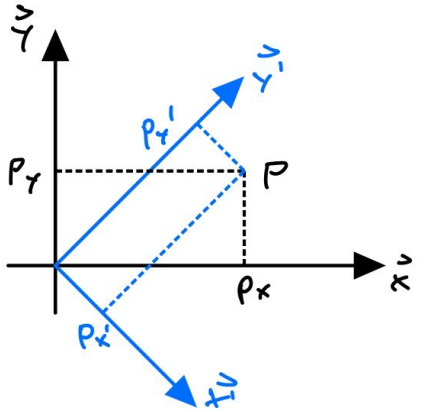
5 Kapitel 5: Kameramodell und Projektionen

5.1 Kamera (Look At-Matrix)

Kameramodell - Weltkoordinatensystems in das Kamerakoordinatensystems wandeln

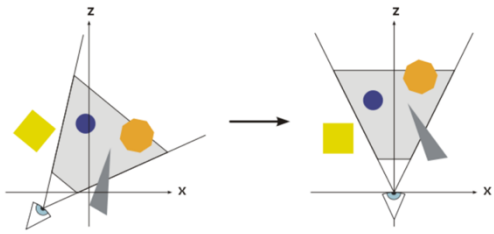
- Skalarprodukt projiziert schwarzen Vektor auf roten Vektor. Führt eine Rotation aus ohne Rotationsmatrix
- Das Skalarprodukt kann verwendet werden, da das Ergebnis eine Projektion darstellt. Die Achsen des Weltkoordinatensystems und des Kamerakoordinatensystems müssen normiert sein. Die Zeichnung ist in 2D, funktioniert in 3D genauso.
- Schwarz: Weltkoordinatensystem
  Blau: Kamerakoordinatensystem



$p_y' = p_y \cdot \vec{y}'$

$p_x' = p_x \cdot \vec{x}'$

Skalarprodukt



- Führe dieses Projektionsprinzip jetzt für alle 3 Koordinatenrichtungen durch
- Seien dabei $\mathbf{u}, \mathbf{v}, \mathbf{w}$ die aufspannenden Richtungen des neuen Koordinatensystems mit $\|\mathbf{u}\| = \|\mathbf{v}\| = \|\mathbf{w}\| = 1$ (Richtungsvektor)

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} u_x & u_y & u_z & 0 \\ v_x & v_y & v_z & 0 \\ w_x & w_y & w_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

(quasi Rotationsmatrix)

- Mit vorheriger Verschiebung:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} u_x & u_y & u_z & 0 \\ v_x & v_y & v_z & 0 \\ w_x & w_y & w_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & t_1 \\ 0 & 1 & 0 & t_2 \\ 0 & 0 & 1 & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} u_x & u_y & u_z & t_1 \\ v_x & v_y & v_z & t_2 \\ w_x & w_y & w_z & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Kameramodell - Blick-Koordinaten definieren

- Eine virtuelle Kamera ist üblicherweise definiert durch die Position Eye, einen Punkt Center auf den sie blickt, und einen Vektor Up der definiert wo oben ist.

LookAt-Matrix berechnen:

1. Neue z-Achse w: $w = Center.Eye$ $w.normalize()$
2. Neue y-Achse v: $v = Up$ $v.normalize()$
3. Neue x-Achse u: $u = v\ x\ w$
4. Neues v, da nicht zwangsläufig orthogonal zu w: $v = w\ x\ u$
5. $u.normalize()$
   $v.normalize()$

→ LookAt-Matrix Ergebnis ohne Verschiebung:

$$\begin{bmatrix} u_x & u_y & u_z & 0 \\ v_x & v_y & v_z & 0 \\ w_x & w_y & w_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

5.2 Projektionen und Projektionsmatrizen – Parallelprojektion

Projektionen: Grundlagen - Parallelprojektion

- Alles ist gleich Groß
- Die Projektionsstrahlen verlaufen parallel
- Projektionszentrum liegt in einem unendlich entfernten Punkt
- Ist ein Spezialfall der Zentralprojektion
- Weniger realistisch, erlaubt aber die Bestimmung von Maßen aus dem Bild

- Alles gleich groß, 3d schwierig zu erkennen. Aber praktisch um in 2d längen zu messen wie bei einem Bauplan

Parallelprojektion: Praktische Umsetzung - Normalized Device Coordinates

$$X_{pixel} = s_x * X_{NDC} + \frac{N_x}{2} \qquad Y_{pixel} = s_y * Y_{NDC} + \frac{N_Y}{2}$$

$X_{NDC} =$ Normalized Device Coordinate x
$X_{pixel} =$ Pixelposition x
$N_x =$ Anzahl an x-Pixel
$s_x = \frac{N_x}{2}$, ein Skalierungsfaktor um die Hälfte der Anzahl der Pixel in x-Richtung
$s_y = -s_x$

Wieso wird bei den Normalized Device Coordinates auch der z-Wert bei der Umrechnung berücksichtigt?

- Verdeckungsproblem lösen, welches Polygon genau gezeichnet werde soll. Der mit dem kleiner z-Wert
- Um von Pixel-Koordinaten, dass Objekt zurück in Weltkoordinaten zu konstruieren. Dafür werden die Werte in einem z-Buffer reingeschrieben

Berechnungsvorschrift für Umrechnung der Koordinate in Normalized Device Coordinates
Annahme: l <= r, also kein Orientierungswechsel!

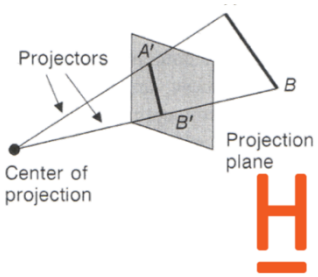
- l: left, r: right, b: bottom, t: top, n: near, f: far

$$
\begin{aligned}
& l \le x \le r && | -l \\
\Leftrightarrow\ & 0 \le x - l \le r - l && | : (r-l) \\
\Leftrightarrow\ & 0 \le \frac{x-l}{r-l} \le 1 && | \cdot 2 \\
\Leftrightarrow\ & 0 \le 2 \cdot \frac{x-l}{r-l} \le 2 && | -1 \\
\Leftrightarrow\ & -1 \le 2 \cdot \frac{x-l}{r-l} - 1 \le 1 && | \text{umformen} \\
\Leftrightarrow\ & -1 \le 2 \cdot \frac{x-l}{r-l} - \frac{r-l}{r-l} \le 1 \\
\Leftrightarrow\ & -1 \le \frac{2x - 2l - r + l}{r-l} \le 1 \\
\Leftrightarrow\ & -1 \le \frac{2x - l - r}{r-l} \le 1 \\
\Leftrightarrow\ & -1 \le \frac{2x}{r-l} - \frac{r+l}{r-l} \le 1
\end{aligned}
$$

$$
\begin{aligned}
& n \le -z \le f \\
& 0 \le -z - n \le f - n \\
& 0 \le \frac{-z-n}{f-n} \le 1 \\
& 0 \le 2 \cdot \frac{-z-n}{f-n} \le 2 \\
& -1 \le 2 \cdot \frac{-z-n}{f-n} - 1 \le 1 \\
& -1 \le 2 \cdot \frac{-z-n}{f-n} - \frac{f-n}{f-n} \le 1 \\
& -1 \le \frac{-2z - 2n - f + n}{f-n} \le 1 \\
& -1 \le \frac{-2z}{f-n} - \frac{f+n}{f-n} \le 1
\end{aligned}
$$

$$
\begin{bmatrix}
\frac{2}{r-l} & 0 & 0 & -\frac{r+l}{r-l} \\
0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\
0 & 0 & \frac{-2}{f-n} & -\frac{f+n}{f-n} \\
0 & 0 & 0 & 1
\end{bmatrix}
\cdot
\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}
$$

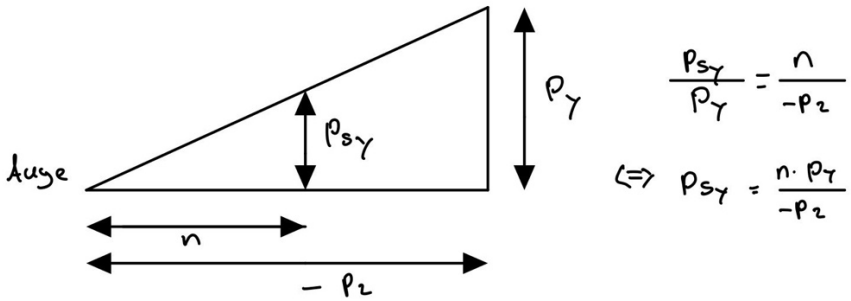
5.3 Projektionen und Projektionsmatrizen – Zentralprojektion



Zentralprojektion: Grundlagen

- Die Projektionsstrahlen gehen durch das Projektionszentrum
- Erzeugt eine Tiefenwirkung
- Sieht realistischer aus, weil es dem entspricht, was unser Auge macht

Zentralprojektion: Berechnung



$$\frac{p_{sy}}{p_y} = \frac{n}{-p_z} \quad \Leftrightarrow \quad p_{sy} = \frac{n \cdot p_y}{-p_z}$$

Matrix sieht etwas anders aus, da man durch -Pz teilt. Man könnte -Pz durch n/-Pz eingeben, aber dann müsste jeder Punkt seine eigene Matrix haben. Pz ist variabel zum Punkt, deswegen diesen Trick wie unten anwenden

$$\begin{bmatrix} n & 0 & 0 & 0 \\ 0 & n & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix} = \begin{bmatrix} n \cdot p_x \\ n \cdot p_y \\ 0 \\ -p_z \end{bmatrix} \Rightarrow \begin{bmatrix} n \cdot p_x / -p_z \\ n \cdot p_y / -p_z \\ 0 \\ 1 \end{bmatrix}$$

- **Fehlt noch:** Normieren auf Normalized Device Coordinates, d.h. $[l, r] \mapsto [-1, 1]$, $[b, t] \mapsto [-1, 1]$ und auch $[n, f] \mapsto [-1, 1]$.
- Also auch hier: für Normalized Device Coordinates darf die z-Koordinate nicht verloren gehen.

Zentralprojektion: Berechnung Normalized Device Coordinates

Berechnungsvorschrift für Umrechnung der Koordinate in Normalized Device Coordinates

- Analog für die y-Koordinate
- Vorgehen: A und B ermitteln, falls es auf der near clipping plane -1 und der far clipping plane +1 projiziert wird
- Pz mit n und f ersetzten. -n und -f ist zu beachten

$$
\begin{aligned}
l \le P_{s_x} &\le r \\
0 \le P_{s_x} - l &\le r - l \\
0 \le \frac{P_{s_x} - l}{r - l} &\le 1 \\
0 \le 2 \cdot \frac{P_{s_x} - l}{r - l} &\le 2 \\
-1 \le 2 \cdot \frac{P_{s_x} - l}{r - l} - 1 &\le 1 \\
-1 \le 2 \cdot \frac{P_{s_x} - l}{r - l} - \frac{r - l}{r - l} &\le 1 \\
-1 \le \frac{2P_{s_x} - 2l - r + l}{r - l} &\le 1 \\
-1 \le \frac{2P_{s_x} - l - r}{r - l} &\le 1 \\
-1 \le \frac{2P_{s_x}}{r - l} - \frac{r + l}{r - l} &\le 1
\end{aligned}
$$

$$
\begin{bmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & A & B \\ 0 & 0 & -1 & 0 \end{bmatrix} \cdot \begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix} = \begin{bmatrix} \dots \\ \dots \\ A \cdot P_z + B \\ -P_z \end{bmatrix} \xrightarrow{\text{normieren}} \begin{bmatrix} \dots \\ \dots \\ \dots \\ 1 \end{bmatrix}
$$

$$
\boldsymbol{P_{S_z,NDC} = \frac{P_z * A + B}{-P_z}}
$$

Es werden Terme gesucht für **A** und **B** mit

- -1 auf der near-Clipping Plane Pz = -n
  - Setzte n die Formel ein $\frac{-n*A+B}{-(-n)} = -1, \; falls \; P_z = -n$
- 1 auf der far-Clipping Plane Pz = -f
  - Setzte -f in die Formel ein $\frac{-f*A+B}{-(-f)} = +1, \; falls \; P_z = -f$

→ $\begin{matrix} (1) & \boldsymbol{-nA + B = -n} \\ (2) & \boldsymbol{-fA + B = f} \end{matrix}$ → Gleichung auslösen

- löse Gleichung (1) nach B auf: $B = -n + An$
- ersetze B in der zweiten Gleichung: $-fA - n + An = f$ und löse nach A auf:

$$-fA + An = f + n \;\Rightarrow\; -(f - n)A = f + n \;\Rightarrow\; A = -\frac{f + n}{f - n}$$

- da A jetzt bekannt ist, ist B leicht auszurechnen: ersetze A in Gleichung (1) um B zu finden:

$$B = -n + An = -n - \frac{f + n}{f - n} \cdot n = -(1 + \frac{f + n}{f - n}) \cdot n = -\frac{(f - n + f + n) \cdot n}{(f - n)} = -\frac{2fn}{f - n}$$

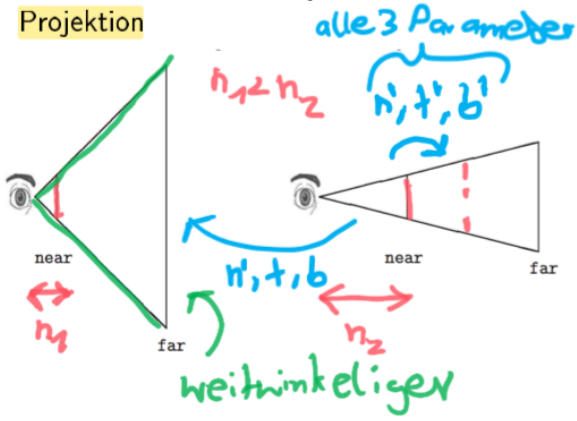
- fasse alle Ergebnisse zur Gesamt-Projektionsmatrix zusammen:

$$
\begin{bmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & -\frac{2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix}
$$

5.4 Projektionen und Projektionsmatrizend - View Frustum

- z-wert darf nicht kleiner als -1 sein, z-Werte die kleiner sind, werden nicht gezeichnet. Z-Plane verschieben, um im Objekt reinzugucken
- Problem: Projektionswinkel wird verändern, Perspektive ändern sich
- Sichtfeld wird kleiner und man sieht den roten Punkt P1 nicht mehr
- Deshalb muss die Clipping Plane auch größer werden beim Verschieben, damit der Blickwinkel gleich bleibt
- Wenn man gleichbleibende Projektion will muss bei einem gegebenen Winkel und veränderter near die anderen Parameter sich verändern

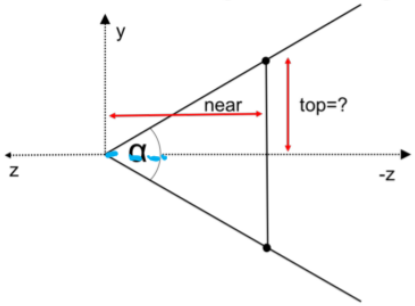
- Änderungen des Parameters `near` für die `frustum`-Methode ändert nicht nur die Sichtbarkeit von Objekten die nah am Sichtpunkt liegen, sondern die ganze Projektion



- Ein kleiner Wert von `near` (bei l, r, b, t unverändert) weitet das sichtbare Volumen stark auf (Weitwinkel-Effekt)
- Ein großer Wert von `near` (bei l, r, b, t unverändert) engt das sichtbare Volumen stark ein (Tele-Effekt)

- Symmetrie wichtig. Symmetrisch um den Ursprung top und bottom, deswegen alpha / 2
- Kamerasicht alpha bleibt immer gleich und near ist Argument, bei veränderter near soll sich top ändern

- In der Praxis ist dieser Effekt nicht erwünscht, da man z.B. durch das Verschieben der near-Clipping-Plane in eine technische Konstruktion hineinblicken möchte, indem die vorne liegenden Teile ausgeblendet werden.
- Eine Änderung des Öffnungswinkels der Kamera ist hier nicht gewünscht



- Deswegen: Berechne `left, right, bottom, top` symmetrisch um den Ursprung aus den gegebenen Öffnungswinkeln der Kamera:

$$\tan\frac{\alpha}{2} = \frac{top}{near} \Rightarrow top = near \cdot \tan\frac{\alpha}{2},$$

$$\text{bzw. } bottom = -near \cdot \tan\frac{\alpha}{2}$$

- Beta, wichtig, wenn Bild bzw. plane nicht quadratisch ist

- analog

$$\tan\frac{\beta}{2} = \frac{right}{near} \Rightarrow right = near \cdot \tan\frac{\beta}{2}, \text{ und } left = -near \cdot \tan\frac{\beta}{2}$$

für den seitlichen Öffnungswinkel der Kamera.

- Implementierung:

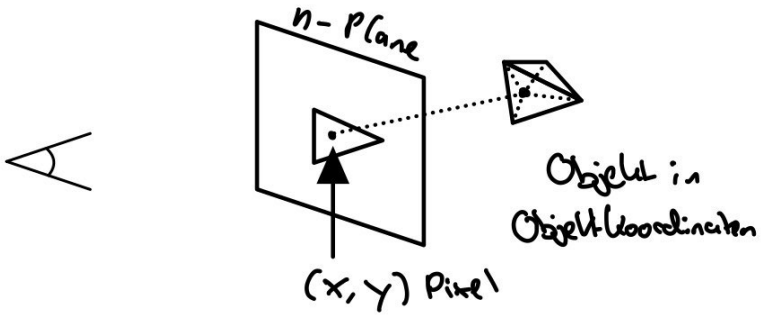
user-Parameter

```
glm::mat4 perspective(double α, double aspect, double near, double far)
{
  return frustum(−near · tan α·aspect/2 , near · tan α·aspect/2 , −near · tan α/2 , near · tan α/2 , near , far);
}
```

- wobei aspect=$\frac{\beta}{\alpha}$ das Verhältnis zwischen horizontalem und vertikalem Blickwinkel ist.

5.5 Selektion/Markierung von Objekten (Picking)

- Durch einen Mausklick wird eine Pixelposition erhalten. Beim Picking wird nach dem Punkt gesucht, der auf diesen Pixel projiziert wurde. Bzw. der Punkt in lokale Koordinaten des Objektes
- Das Vorgehen grundsätzlich:
  Pixel → NDC / projizierte Koordinaten → Kamerakoordinaten → Weltkoordinaten → Objektkoordinaten



Picking in Bild-Koordinaten

Picking - Rekonstruktion 3D-Punkt in Welt-Koordinaten

Die Objekt-ID in rgb codieren und jedes Objekt in einem zusätzlichen Select-Buffer zeichnen. Aus der Farbe kann dann die Objekt-ID rekonstruiert werden

1. Aus den Pixelkoordinaten X_{NDC}und Y_{NDC} berechnen. Z_{NDC}kann aus dem Z-Buffer gelesen werden
2. Auf den Punkt $(X_{NDC},\ Y_{NDC},\ Z_{NDC},\ 1)^T$ die Inverse der Projektionsmatrix anwenden
3. Auf den erhaltenen Punkt die Inverse der Kameramatrix anwenden
   ⇨ Punkt ist in Weltkoordinaten

Problem:

- Das lokale Koordinatensystem ist nicht rekonstruierbar, da nicht bekannt ist, aus welchem Objekt der Punkt abgebildet wurde. Deswegen braucht man einen Strahl, der durch alle lokalen Koordinatensystem geschossen wird.

Vorteile: Effizienz

Nachteile: Objekt-Zuordnung in Hierarchie des Szenengraphen nicht möglich **und** Nachbarschaften nicht bekannt

Picking in lokalen Objekt-Koordinaten

Schnittpunkt-Berechnung von Picking-Strahl und Objekten

Einen Picking Strahl erzeugen, der in allen lokalen Koordinatensystem transformiert wird. Dabei Schnittpunkttest mit den Objekten machen

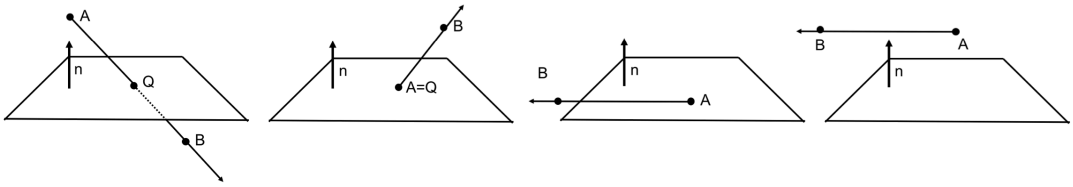
1. Berechne aus den Pixel-Koordinaten Xndc und Yndc. aus
2. Punkte der near clipping plane $\boldsymbol{A} = (X_{NDC},\ Y_{NDC}, -n,\ 1)$
   und far clipping plane $\boldsymbol{B} = (X_{NDC},\ Y_{NDC}, -f,\ 1)$ die Inverse der Projektionsmatrix anwenden
3. Danach die Inverse der Kameramatrix anwenden
4. Der Strahl $\boldsymbol{s(t)} = A + t * (B - A)$. Auf s liegen alle Punkte, die auf X_{NDC}und Y_{NDC}projiziert wurden

Vorteile: Objekt-Zuordnung in Hierarchie möglich und Nachbarschaften bekannt

Nachteile: aufwändiger, dauert länger

Strahl: $\boldsymbol{R(t) = A + t * (B - A)}$**.**

Ebenen-Darstellungen mit Strahl



1. Eindeutiger Schnittpunkt in der Ebene
2. A=Q, also unser Startpunkt ist genau in der Ebene, es gibt aber kein Durchstoss
3. Strahl liegt in der Ebene
4. Strahl liegt parallel zur Ebene aber nicht in der Ebene
5. kein Schnitt: A ∈/ P und R zeigt von Ebene weg (t < 0), ohne Bild.

Infinity Arithmetik IEEE 754

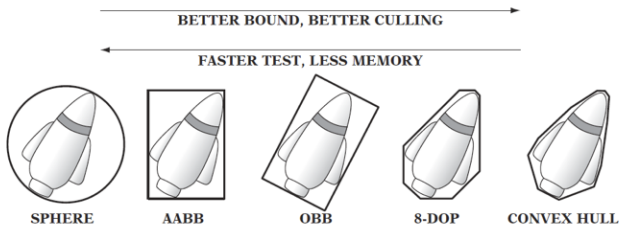
- Die Infinity Arithmetik ist ein Standard wie man mit NaN und INF-Werten rechnen kann, um damit boolesche Operationen oder Rechnungen durchzuführen.
  - z.B. 2+INF = INF, 0/0 = INF, 2+NaN = NaN
  - NaN == X → false
- Die booleschen Operationen können genutzt werden damit man bei der Schnittpunkt Berechnung nicht extra if-Abfragen benötigt, um wischen den verschiedenen Ebene-Strahl Schnitt Arten zu unterscheiden.
- Falls durch 0 geteilt wird oder 0/0 berechnet wird kann es passieren das der Strahl parallel zur Ebene oder in der Ebene liegt.

Bounding Volumes (AABB, Kugel)

Ein Bounding Volume umhüllt komplexe Objekte.
Man benutzt dies um einen billigen Schnitt-Test zu bekommen, es wird geprüft ob das BV geschnitten worden ist und dann erst geschaut ob welche faces geschnitten worden sind.
Sonst muss man durch alle faces von allen Objekten durchiterieren um zu prüfen ob ein Schnitt vorhanden ist.

Vorteile:

- Billige Schnitt-Tests, Volume selbst billig zu berechnen
- Enge Passform
- Einfach zu rotieren und transformieren
- Wenig Speicher-Verbrauch



Nachteile:

- Viel Raum in der sich kein Objekt befindet bei einfachen Formen → falscher positiver Test

Je nachdem wie einfach man das BV auswählt hat man eine schlechte Passgenauigkeit jedoch schnellere Berechnungen. Und genau so auch andersrum

Schnitt Strahl vs. AABB

- Damit ein Strahl eine AABB schneidet, muss er zu einem Zeitpunkt t gleichzeitig im x-, y-,z-Koordinatenbereich sein. Das sind die sogenannten slabs.
- Der Schnitt muss in allen Ebenen vorhanden sein. Wie in der Skizze wo sich x und y lab überlappen. Es wird nicht geprüft, wo der Schnittpunkt ist sondern nur ob es einen gibt.

- Diagramm unten ist es zweidimensional dargestellt. Mathematisch sind mehrere Ebenen-Schnitte der Ein- und Austrittspunkte zu berechnen, die einen Intervall an möglichen t-Parametern darstellen und für die x, y und z-Koordinaten darf am Ende die Schnittmenge an möglichen t's nicht leer sein für einen gültigen Schnittpunkt.