First Steps Towards Virtual Reality

Teilnehmer:

Korcan Dau	Herder-Oberschule, Berlin
Justus Eilers	Heinrich-Hertz-Oberschule, Berlin
Niels Grube	Käthe-Kollwitz-Oberschule, Berlin
Azad Gül	Immanuel-Kant-Oberschule, Berlin
Janek Reichardt	Immanuel-Kant-Oberschule, Berlin
Erik Richter	Käthe-Kollwitz-Oberschule, Berlin
Luana Riebel	Heinrich-Hertz-Oberschule, Berlin
Gábor Schultz	Herder-Oberschule, Berlin

Gruppenleiter:

Marco Bettinaglio	MNG Rämibühl, Zürich

Schon im Mittelalter versuchten Menschen ihre Umwelt und ihre Familien mit Hilfe von Gemälden für die Nachwelt festzuhalten. Doch meistens fehlten den Bildern Raumtiefe und Dynamik. Um 1410 begann Filippo Brunelleschi sich Gedanken über das Sehen an sich zu machen und entwickelte in Folge dessen die Zentralperspektive, mit deren Prinzip und technischer Umsetzung wir uns in den letzten Tagen beschäftigt haben. Auf folgenden Seiten möchten wir auf die mathematische Wichtigkeit einiger Aspekte genauer eingehen:

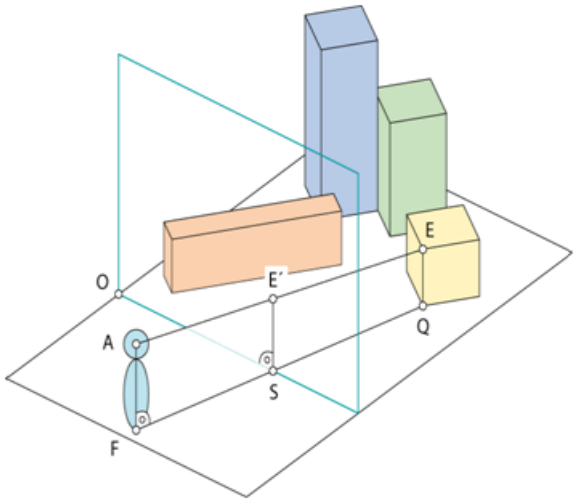
1) Grundprinzip – Wie Vektoren die Fotografie bestimmen
   - 1.1) Von Durchstoßpunkten und Augendistanz
   - 1.2) Entfernungsproblem – Die Suche nach dem Lambda
2) Fluchtpunkte – Alles nur Grenzwerte
3) Flugsimulator – Wir bringen eine Landschaft in den Computer
4) Ray Tracing – Die schattige Seite der Zentralprojektion

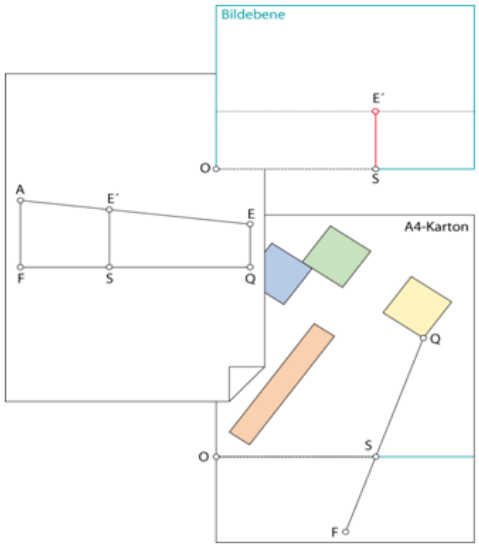
1 Wie Vektoren die Fotografie bestimmen

1.1 Von Durchstoßpunkten und Augendistanz

Zum besseren Verständnis wird die Zentralprojektion anhand einer Skizze erläutert.

Im Bild mit A gekennzeichnet ist das Auge des Betrachters, welcher einen Punkt (hier mit E bezeichnet) fixiert. Um ein Bild zu erzeugen, soll der Durchstoßpunkt E' auf der blau umrahmten Bildfläche abgebildet werden. Durch Abbildung aller Durchstoßpunkte auf der Bildfläche entsteht eine 2D-Kopie der 3D-Umgebung.





E' lässt sich finden, indem man zuerst im Grundriss eine Verbindungslinie zwischen dem Fußpunkt des Betrachters F und dem Fußpunkt des Gebäudes Q zieht. Wir erhalten zusammen mit den Strecken AF, EQ und AE ein zweidimensionales Trapez. Anhand des Grundrisses kennen wir den Punkt S. E' ist der Schnittpunkt der zu FQ senkrechten Strecke durch S mit AE. Nun übertragen wir die Länge $\overline{OS}$ auf das Bild und konstruieren eine Parallele, mit dem Abstand $\overline{SE}$‘.

Dieses Verfahren wenden wir für alle Punkte der Landschaft an und erhalten unser Bild, welches sich durch eine realistische Raumwirkung auszeichnet.

Von A aus fällen wir das Lot auf die Bildfläche. Der entstandene Punkt heißt Hauptpunkt H. Die Strecke $\overline{AH}$ heißt Augendistanz d. Betrachtet man ein Bild mit der Entfernung d (hier: Augendistanz zum Zeitpunkt der Aufnahme), so wirkt dieses besonders realistisch.

1.2 Das Entfernungsproblem – Die Suche nach dem Lambda

Um das Problem zu lösen benötigt man Vektoren, die in der folgenden Form dargestellt werden:

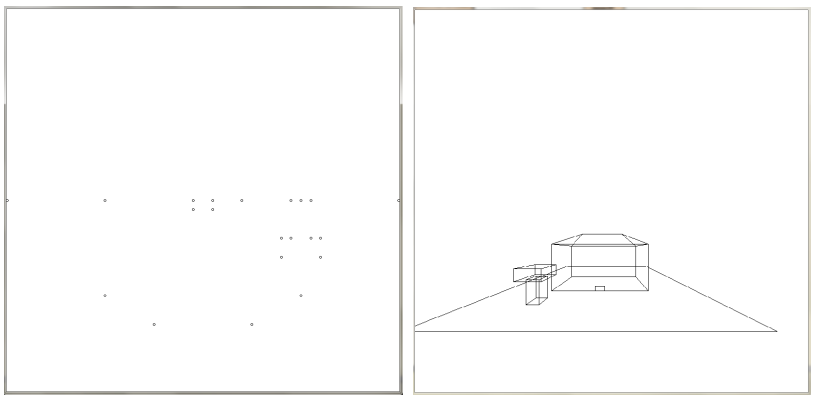
$$\overrightarrow{OA} = \begin{pmatrix} X_a \\ Y_a \\ Z_a \end{pmatrix}, \overrightarrow{OE} = \begin{pmatrix} X_e \\ Y_e \\ Z_e \end{pmatrix}.$$

Das Ziel ist nun den entsprechenden Punkt auf der Bildfläche für jeden Punkt in der Landschaft zu finden. Dies wird mithilfe von Vektorgeometrie gelöst.

Wie man bereits erkennen kann, muss für den Vektor $\overrightarrow{AE'}$, den es zu bestimmen gilt, $\overrightarrow{AE'} = A + \lambda * (\vec{A} - \vec{E})$ sein. Uns sind sowohl der Vektor A als auch der Vektor P bekannt, weswegen wir nur das Lambda ausrechnen müssen. Wir betrachten nun zuerst ausschließlich die Vektorgleichung für die x-Komponente, für welche gilt: $X_{AE'} = X_A + \lambda * (X_A - X_E)$. Daraus erhalten wir $\lambda = \dfrac{X - X_A}{X_E - X_A}$.

X_A	–	x-Koordinate von $\vec{A}$,
X	–	x-Koordinate vom Bildschirm,
$X_{AE'}$	–	x-Koordinate vom gesuchten Vektor $\overrightarrow{AE'}$,
X_E	–	x-Koordinate vom beliebig gewählten Punkt E aus der Landschaft.

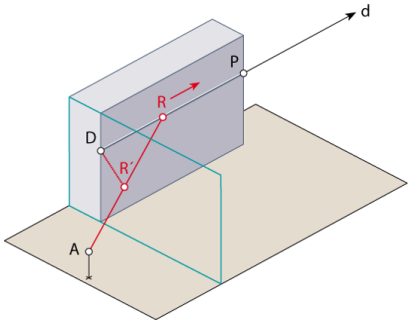
Diese Rechnung muss nun für jeden Eckpunkt ausgeführt werden, damit wir ein Bild in der folgenden Art erhalten.



Als nächsten Schritt verbinden wir nun alle diese Eckpunkte, so wie sie auch schon im Original verbunden waren, und erhalten ein Bild nach Art der Zentralprojektion.

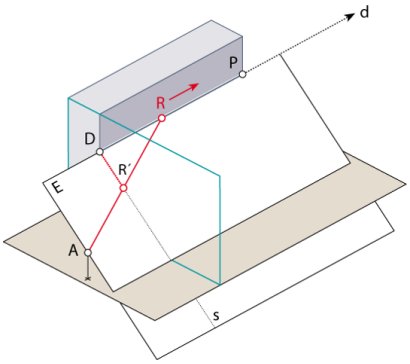
2 Fluchtpunkte – Alles nur Grenzwerte

Auf dem obenstehenden Bild ist erkennbar, dass parallele Kanten in der zentralperspektivischen Ansicht nicht parallel sind. Bei Verlängerung der Kanten kommt es zu einem Phänomen: die Geraden treffen sich in einem Punkt, dem Fluchtpunkt F.

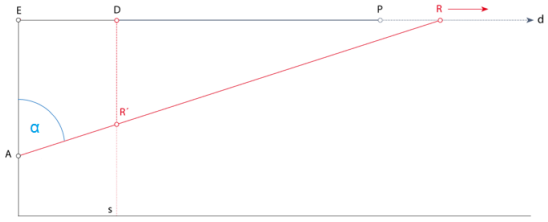


In der nebenstehenden schematischen Darstellung ist der prinzipielle Aufbau einer zentralperspektivischen Projektion in Kavalierperspektive dargestellt, bei der die Kante d orthogonal zur Bildfläche steht. Dabei ist A die Position des Betrachters. Der Punkt R, welcher auf d liegt, wird auf der Bildfläche als R' abgebildet.

Wenn man weitere Punkte der Kante d auf der Bildfläche darstellt, erkennt man, dass die abgebildete und im Raum parallel zum Boden gerichtete Kante schräg nach unten verläuft. Fügt man nun eine Hilfsebene ein, welche die Bodenfläche und die Bildebene wie im Bild rechts schneidet, kann man nun den Punkt R auf der Kante d nach hinten wandern lassen und somit die Strecke $\overrightarrow{ER}$ verlängern.

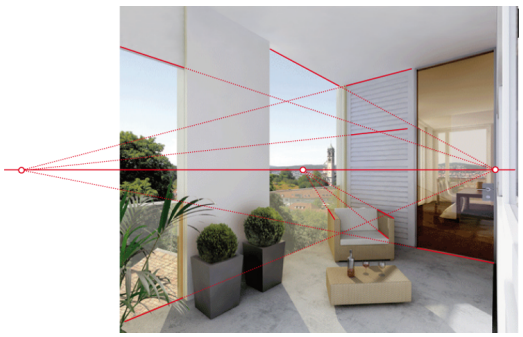


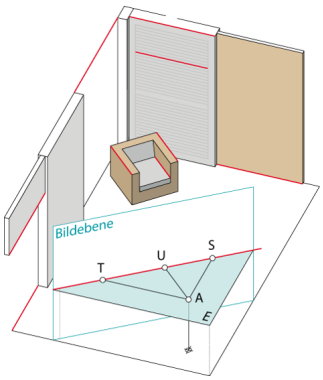
Jetzt kann man den Grenzwert $\lim_{\overline{ER}\to\infty} \alpha$ bestimmen. α ist abhängig von der Position von R und da $\overline{ER}$ die Gegenkathete und $\overline{EA}$ die Ankathete sind, auch dementsprechend darstellbar als Arkustangens von $\frac{\overline{ER}}{\overline{AE}}$. Damit ergibt sich $\lim_{\overline{ER}\to\infty} \alpha = \lim_{\overline{ER}\to\infty} \arctan(\frac{\overline{ER}}{\overline{AE}})$. Da $\overline{AE}$ konstant ist und $\overline{ER}$ gegen unendlich geht, ergibt sich nach Rechenregeln für bestimmte Divergenz: $\lim_{\overline{ER}\to\infty} \alpha = \lim_{\overline{ER}\to\infty} \arctan(\frac{\overline{ER}}{\overline{AE}}) = 90°$.



Der Fluchtpunkt F kann also auf der Bildfläche dargestellt werden und liegt auf der Geraden, die durch D und P' geht. Außerdem ergibt sich aus der Grenzwertbetrachtung für den Winkel α, dass der Fluchtstrahl, der Strahl der vom Auge A durch den Fluchtpunkt F verläuft, parallel zur Kante d im Raum ist, da sowohl α als auch der Winkel $\sphericalangle AED$ 90° beträgt.

Bei mehreren Gruppen paralleler Kanten, besitzt jede ihren eigenen Fluchtpunkt. Es fällt auf, dass alle Fluchtpunkte auf einer horizontalen Linie liegen. Die Lösung dieses Phänomens ist schnell gefunden. Da unser Betrachter A senkrecht auf der Grundfläche steht und unsere Parallelen parallel zum Boden und zu ihrer jeweiligen Fluchtlinie sind, liegen alle Fluchtpunkte auf einer Ebene.





3 Flugsimulation – Wir bringen eine Landschaft in den Computer

3.1 Erstellen eines 3D-Modells

- Um mit Hilfe von Java ein 3D-Modell erstellen zu können, ist es am Anfang nötig, dass alle Ecken und Kanten des Modells in zwei zweidimensionalen Arrays (also in Tabellen beziehungsweise Matrizen) abgespeichert werden, ein Array für die Ecken und einer für die Kanten:

  public int[ ][ ] Ecke
  public int[ ][ ] Kante

- Hierbei steht „int“ für „integer“. Es wird also lediglich angegeben, dass es ein Array für ganze Zahlen ist.

- Bei den Ecken dient der Array dazu, die x-, y- und z-Werte jeder einzelnen Ecke zu speichern:

  Ecke[0][0] = 50
  Ecke[0][1] = 50
  Ecke[0][2] = 0

- Hierbei steht die Zahl in der ersten Klammer für die Nummer der Ecke und die Zahl in der zweiten Klammer für die x-, y- und z-Werte. Für die Ecke 0 sind also sowohl der x-, als auch der y-Wert 50 und der z-Wert ist 0 (x-Wert: 0, y-Wert: 1 und z-Wert: 2).

- Bei den Kanten dient der Array dazu, den Start- und Zielpunkt jeder Kante zu speichern, also welche beiden Ecken durch diese Kante verbunden werden:

  Kante[0][0] = 0
  Kante[0][1] = 1

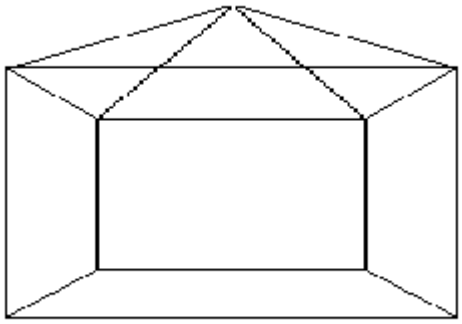
- Hierbei steht die Zahl in der ersten Klammer für die Nummer der Kante und die Zahl in der zweiten Klammer für den Start- oder Zielpunkt (Startpunkt: 0 und Zielpunkt: 1).

- Als nächstes muss der Standpunkt des Betrachters definiert werden. Dies geschieht mit einem eindimensionalen Array:

  public double[ ] Auge
  Auge[0] = 200
  Auge[1] = 0
  Auge[2] = 50

- Die Zahl in der Klammer steht für die x-, y- und z-Werte. Für das Auge ist also der x-Wert 200, der y-Wert 0 und der z-Wert 50.
- Aus den Werten der Eckpunkte werden als nächstes unter Zuhilfenahme des Streckfaktors λ, der ebenfalls berechnet werden muss, die Bildpunkte, welche auf der Bildebene liegen, berechnet. Diese werden dann in einem weiteren multidimensionalen Array gespeichert:

  public double[ ][ ] Bild
- Um nun das vollständige 3D-Modell zu erhalten ist es notwendig den Befehl p.line zu verwenden, welcher dafür sorgt, dass man aus den Daten des Bild- und des Kantenarrays durch zeichnen der Kanten zwischen den Bildpunkten eine 3D-Darstellung des Models erhält.



3.2 Bewegen des 3D-Modells:

Um dieses 3D-Modell nun in Bewegung zu versetzen, ist es nötig fünf Schritte in einer Dauerschleife auszuführen:

- 1. Schritt: Die Bildfläche muss geleert werden, damit sich die einzelnen Bewegungsschritte des 3D-Modells nicht überlagern und wir keinen komplett schwarzen Bildschirm erhalten.
- 2. Schritt: Im Anschluss muss der x-, y- und/oder der z-Wert aller Ecken verändert werden, damit das Bild sich in eine vom Benutzer definierte Richtung bewegen kann.

- 3. Schritt: Nun müssen die neue Bildpunkte mit veränderten Werten der Eckpunkte wieder berechnet werden, damit diese wieder durch die Kanten verbunden werden können.
- 4. Schritt: Anschließend kann das neue Bild gezeichnet werden.
- 5. Schritt: Zum Schluss ist es wichtig, eine Pause einzufügen, da die Bilder sonst so schnell neu gezeichnet werden, dass man im Prinzip nichts mehr erkennen kann. Die Pause darf aber auch nicht länger als eine vierundzwanzigstel Sekunde sein, da das menschliche Auge dann die einzelnen Bilder wahrnehmen könnte und es nicht wie eine flüssige Bewegung wirken würde, sondern ein wenig abgehackt.

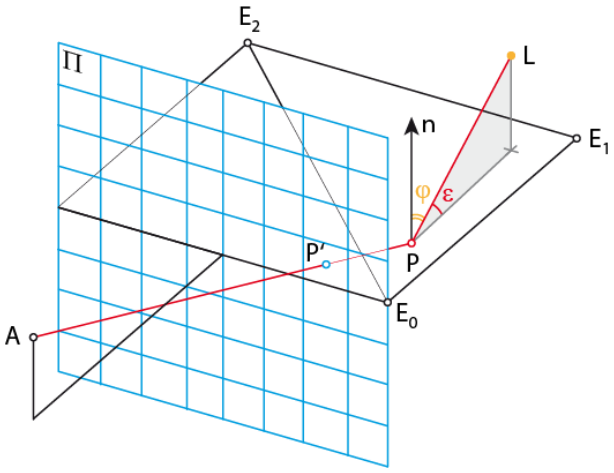
4 Ray Tracing – Die schattige Seite der Zentralprojektion

4.1 Idee

Nachdem wir uns bereits mit der Darstellung der Zentralprojektion in Java beschäftigt haben, überlegten wir uns wie diese Modelle realistischer wirken könnten. Dafür müssten zum Beispiel Schatten, Lichtintensitäten und Verdeckungen realisiert werden. Hier eignet sich das Verfahren der Strahlenverfolgung, oder zu Englisch: „Ray Tracing“. Dabei geht es um die Entwicklung eines Algorithmus, der die Position eines Punktes und die Lichtintensität an diesem Punkt berechnet.

4.2 Grundüberlegungen

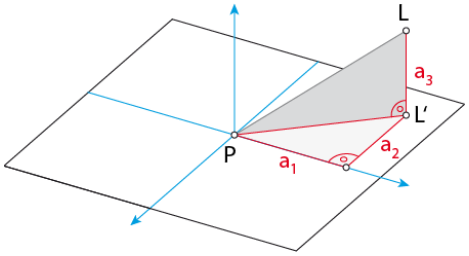
Die Grundlage ist unser bekanntes Setting, welches durch eine Lichtquelle L erweitert wird. Weiterhin teilen wir unsere Bildebene in ein Gitternetz, wobei jedes Quadrat des Gitternetzes später einem Pixel auf dem Bildschirm entsprechen wird.



Beim Ray Tracing-Verfahren wird nun vom Auge A ein Strahl durch den Mittelpunkt jedes Quadrates ausgesendet, es entsteht der Bildpunkt P', sowie der Punkt P in der Landschaft. Des Weiteren sendet die Lichtquelle L einen Lichtstrahl zu unserem fixierten Punkt P. Die Lichtintensität an diesem Punkt bestimmt die Helligkeit des Pixels am Bildschirm. Wir entwickeln folgende Überlegungen:

- $I \sim \frac{1}{r^2}$, wobei r die Entfernung von P zu L ist.
- $I \sim \sin(\varepsilon)$, wobei ε der Einfallswinkel des Lichtstrahls ist $\Rightarrow I_p(\varepsilon, r) = c \cdot \frac{\sin(\varepsilon)}{r^2} \cdot I_0$.

4.2.1 Die Berechnung von r



Wir wenden an dieser Stelle zwei Mal den Satz des Pythagoras an:

- $\sqrt{(a_1)^2 + (a_2)^2}$
- $\sqrt{\sqrt{(a_1)^2 + (a_2)^2}^2 + (a_3)^2} = \sqrt{(a_1)^2 + (a_2)^2 + (a_3)^2}$

Somit haben wir hergeleitet, wie man die Länge, also den Betrag eines Vektors bestimmt:

$$\left| \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} \right| = \sqrt{(a_1)^2 + (a_2)^2 + (a_3)^2}.$$

4.2.2 Die Berechnung von ε

Das Problem ist, dass wir die Neigung der Fläche F zu unserer Lichtquelle L nicht kennen. Jedoch definieren die zwei Vektoren $\overrightarrow{E_0E_1}$ und $\overrightarrow{E_0E_2}$ unsere Fläche F und sind somit bekannte Größen. Es lässt sich also ein dritter Vektor konstruieren, der senkrecht auf $\overrightarrow{E_1E_0}$ und $\overrightarrow{E_2E_0}$ steht. Dieser nennt sich Normalenvektor $\overrightarrow{n_F}$ der Fläche F. Allgemein bildet man das sogenannte Vektorprodukt nach:

$$\begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} \times \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{pmatrix} a_2 \cdot b_3 - a_3 \cdot b_2 \\ a_3 \cdot b_1 - a_1 \cdot b_3 \\ a_1 \cdot b_2 - a_2 \cdot b_1 \end{pmatrix}.$$

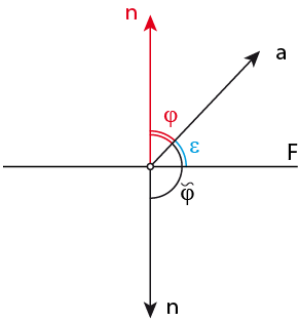
Auf unser Beispiel angewendet bedeutet das:

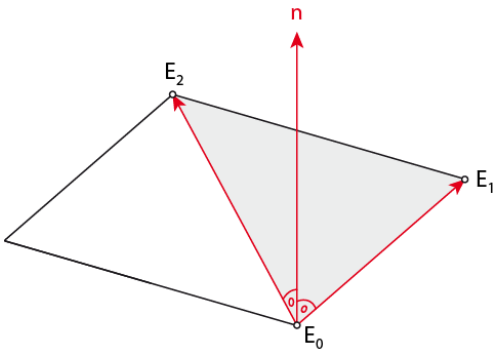
$$\overrightarrow{n_F} = \overrightarrow{E_1E_0} \times \overrightarrow{E_2E_0}.$$

Um den Winkel ε zu bestimmen, verwenden wir das Skalarprodukt. Mit diesem, lässt sich der Winkel φ bestimmen:

$$\overrightarrow{n_F} \circ (P - L) = |\overrightarrow{n_F}| \cdot |(P - L)| \cdot \cos(\widetilde{\varphi}).$$

Wir verwenden an dieser Stelle den Winkel $\widetilde{\varphi}$, um den Fall, dass $\overrightarrow{n_F}$ negativ ist, zu berücksichtigen.





Aus der Zeichnung lässt sich ε bestimmen:

- $180° - \widetilde{\varphi} = \varphi$.
- $\cos(\varphi) = \sin(\varepsilon)$ aufgrund der Phasenverschiebung.

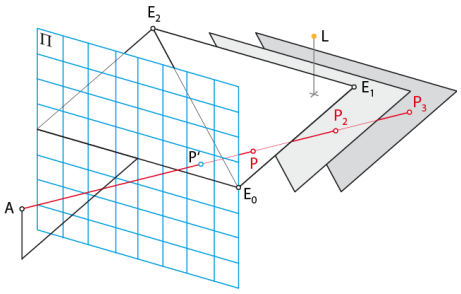
4.2.3 Wo liegt P?

Mit Hilfe der Streckungsfaktoren λ, u und v sowie dem Augpunkt A und dem Eckpunkt E_0 können wir P über bekannte Vektoren beschreiben.

- $P = A + \lambda * \overrightarrow{P'A}$.
- $P = E_0 + u * \overrightarrow{E_1E_0} + v \cdot \overrightarrow{E_2E_0}$.

Durch eine Gleichsetzung ergibt sich ein lineares Gleichungssystem, welches sich als Matrix schreiben lässt. Diese wird mit Hilfe der Gauss-Elimination gelöst und liefert einen Rückschluss auf die gesuchten Streckungsfaktoren.

4.3 Mehrere Flächen



Das Ray Tracing-Verfahren eignet sich auch, um verdeckte Punkte auszublenden.
Wir nehmen an, dass der Strahl aus dem Auge nach der Fläche F auf zwei weitere Flächen trifft. Es entstehen ebenfalls die Punkte P_2 und P_3. Diese lassen sich mit dem Streckungsfaktor λ als Verlängerung des Vektors $\overrightarrow{P'A}$ beschreiben. Wählen wir nun den Punkt, mit dem kleinsten λ, erhalten wir automatisch den Punkt, der sichtbar ist und nicht von F verdeckt wird.

Alle Bilder stammen aus dem Buch „Perspektive Verstehen“ von Marco Bettinaglio und Urs Kirchgraber, herausgegeben vom Orell Füssli Verlag Zürich, 2011.