

Michael Bender  
Manfred Brill

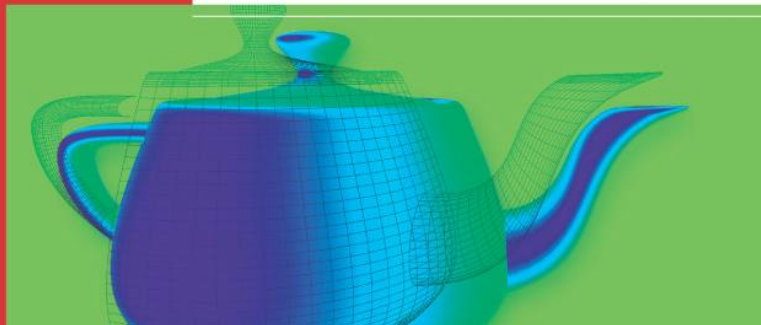
---

# Computer- grafik

Ein anwendungsorientiertes Lehrbuch

HANSER

2. Auflage



## Aufgaben und Lösungen

©Michael Bender, Manfred Brill  
Oktober 2005

Sie finden in diesem Dokument alle Aufgaben und die zugehörigen Lösungen aus

Michael Bender, Manfred Brill: *Computergrafik*

2. Auflage, Hanser Verlag, München, 2006



<http://www.vislab.de/cgbuch>

Diese Unterlagen wurden mit L<sup>A</sup>T<sub>E</sub>X erstellt.

# Kapitel 4

## Polygonale Netze

### 4.1 Polygone und Polyeder

1. Zeichnen Sie die planaren Graphen der Polyeder aus Abbildung 4.5 und überprüfen Sie Gültigkeit der Euler'schen Formel für diese Objekte!

*Lösung:*

Für den Tetraeder gilt  $4 - 6 + 4 = 2$ ; für die vierseitige Pyramide  $5 - 8 + 5 = 2$  und für das Prisma ganz rechts in Abbildung 4.5 auf Seite 198 gilt  $6 - 9 + 5 = 2$ .

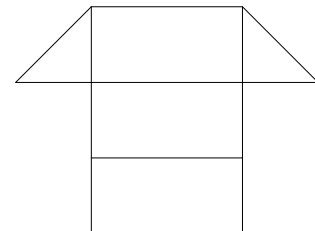
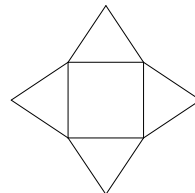
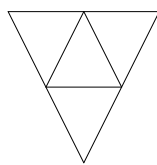


Abbildung 4.1: Die planaren Graphen der Aufgabe 1

2. Überprüfen Sie das linke Polygon aus Abbildung 4.2 mit Hilfe des Verfahrens auf Seite 197 auf Konvexität, falls die Eckpunkte die in Tabelle 4.1 angegebenen Koordinaten besitzen!

Tabelle 4.1: Die Koordinaten für Aufgabe 2

$V_0$	$V_1$	$V_2$	$V_3$	$V_4$	$V_5$
(1, 0)	(5, 0)	(5, 2.5)	(4, 2.5)	(3, 3)	(1, 1)

*Lösung:*

Der Schwerpunkt für das linke Polygon in Abbildung 4.1 ist gegeben als

$$C = \frac{1}{6}(1 + 5 + 5 + 4 + 3 + 1, 0 + 2.5 + 2.5 + 3 + 1) = \left(\frac{19}{6}, \frac{3}{2}\right).$$

Die Normale für  $V_0V_1$  ist  $\mathbf{n}_0 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ , die Normale für  $V_1V_2$  ist  $\mathbf{n}_1 = \begin{pmatrix} -1 \\ 0 \end{pmatrix}$ . Für  $V_2V_3$  ist  $\mathbf{n}_2 = \begin{pmatrix} 0 \\ -1 \end{pmatrix}$ , für  $V_3V_4$  ist die Normale gegeben durch  $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$ . Dann sind die Normalenformen gegeben durch

$$\begin{aligned} y &= 0, \\ -x - 5 &= 0, \\ -y - \frac{5}{2} &= 0, \\ x &= 0. \end{aligned}$$

Die Normalenform für die Gerade für die Kante  $V_3V_4$  ist gegeben durch

$$-2x - \frac{1}{2}y + \frac{15}{2} = 0.$$

Die Normalenform für die Gerade für die Kante  $V_4V_5$  ist gegeben durch

$$x - y = 0.$$

Setzt man  $C$  in diese Normalenformen ein, erhält man immer positives Vorzeichen; also ist dieses Polygon konvex.

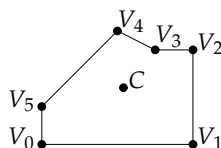


Abbildung 4.2: Das Polygon und der Schwerpunkt für Aufgabe 2

3. An den Eckpunkten eines planaren Polygons ist der Außenwinkel  $\alpha$  und Innenwinkel  $\beta$  wie in Abbildung 4.3 definiert.
- Weisen Sie nach, dass die Summe aller Innenwinkel eines  $n$ -seitigen konvexen Polygons gleich  $(n - 2)\pi$  ist!
  - Weisen Sie nach, dass die Summe aller Außenwinkel gleich  $2\pi$  ist!

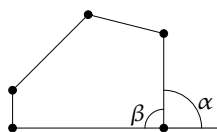


Abbildung 4.3: Innen- und Außenwinkel in einem planaren Polygon

*Lösung:*

- Wir wählen einen der Eckpunkte,  $V_i$ , beliebig aus und bilden Dreiecke mit den restlichen Eckpunkten. Dann gibt es  $n - 2$  Dreiecke wie in Abbildung 4.4. Die Summe der Innenwinkel dieser Dreiecke ist immer  $\pi$ . Dann ist die Summe der Innenwinkel gleich  $I = (n - 2)\pi$ .

- b) An einer Ecke gilt für die Summe des Innen- und Außenwinkels offensichtlich immer  $\alpha + \beta = \pi$ . Also ist  $\alpha = \pi - \beta$  und

$$\begin{aligned} E &= \sum_{i=1}^n \alpha_i = \sum_{i=1}^n (\pi - \beta_i) \\ &= n\pi - \sum_{i=1}^n \beta_i \\ &= n\pi - I \\ &= n\pi - (n-2)\pi \\ &= 2\pi. \end{aligned}$$

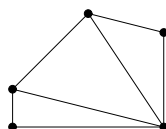


Abbildung 4.4: Dreiecke falls das Eck rechts unten als  $V_i$  gewählt wurde in Aufgabe 3

4. Ist ein Polygon mit  $n$  Eckpunkten sowohl gleichseitig als auch gleichwinklig wird es ein *reguläres  $n$ -Eck* genannt. Reguläre *Sternpolygone* entstehen durch Verbinden des jeweils übernächsten Punkts in einem regulären  $n$ -Eck. Sind  $V_0, V_1, V_2, V_3$  und  $V_4$  die Eckpunkte eines regulären 5-Ecks, dann ist ein reguläres Sternpolygone gegeben durch den geschlossenen Polygonzug  $\{(V_0, V_2), (V_2, V_4), (V_4, V_1), (V_1, V_3), (V_3, V_0)\}$ . Zeichnen Sie reguläre  $n$ -Ecke für  $n = 3, 4, 5, 6, 7, 8$  und reguläre Sternpolygone für  $n = 5, n = 7$  und  $n = 9$ !

*Lösung:*

Die Eckpunkte regulärer  $n$ -Ecke legen wir auf den Einheitskreis mit Mittelpunkt im Ursprung. Dann sind die Eckpunkte gegeben durch

$$V_j = \left( \cos j \cdot \left( \frac{2\pi}{n} \right), \sin j \cdot \left( \frac{2\pi}{n} \right) \right), j = 0, \dots, n-1.$$

Aus den Abbildungen sehen Sie, dass für großes  $n$  die regulären  $n$ -Ecken als Näherung von Kreisen auf dem Bildschirm geeignet sind.

In den Abbildungen 4.5 und 4.6 sehen Sie die  $n$ -Ecke für  $n = 3, 4, 5, 6, 7$  und  $n = 8$ .

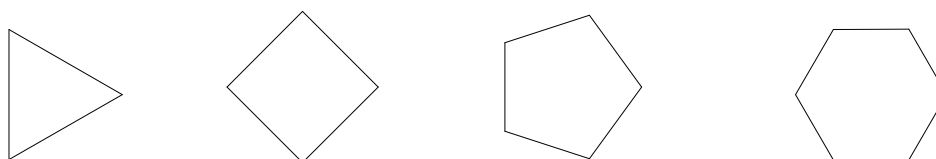


Abbildung 4.5: Reguläre  $n$ -Ecke für  $n = 3, 4, 5$  und  $n = 6$  für Aufgabe 4



Abbildung 4.6: Reguläre  $n$ -Ecke für  $n = 7$  und  $n = 8$  für Aufgabe 4

In Abbildung 4.7 finden Sie die entsprechenden regulären Sternpolygone. Eine Methode zur Konstruktion der Sternpolygone ist es, wie in der Aufgabenstellung von den regulären  $n$ -Ecken auszugehen. Die Sternpolygone sind Beispiele für nicht-einfache Polygone. Jede der

Seiten wird in 3 Teile geteilt. Die jeweils mittleren Teilen bilden ein konvexes reguläres Polygon, den sogenannten *Kern* des regulären Sternpolygons.

Dadurch wird eine weitere Konstruktion der Sternpolygone möglich. Zeichnen Sie ein reguläres  $n$ -Eck. Die Ecken des entsprechenden regulären Sternpolygons sind gegeben als Schnittpunkt der Geraden, die die Kanten des  $n$ -Ecks enthalten. Dabei wird die  $i$ -te Kante immer mit der  $(i + 2)$ -ten Kante geschnitten!

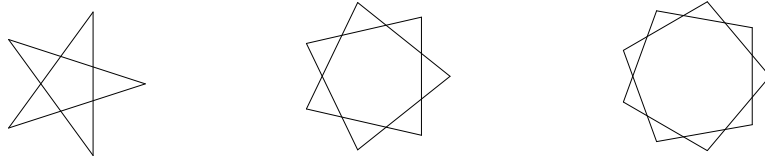


Abbildung 4.7: Die regulären Sternpolygone für  $n = 5$ ,  $n = 7$  und  $n = 9$  für Aufgabe 4

## 4.2 Datenstrukturen für polygonale Netze

- Gegeben sind die Punkte  $V_1 = (1, 1)$ ,  $V_2 = (2, 5)$ ,  $V_3 = (2, 1)$ ,  $V_4 = (3, 5)$ ,  $V_5 = (4, 3)$ ,  $V_6 = (5, 2)$ ,  $V_7 = (4, 0)$  und  $V_8 = (3, 0)$ . Skizzieren Sie das polygonale Netz für die Eckenliste  $F_1 = (1, 2, 3)$ ,  $F_2 = (3, 2, 4)$ ,  $F_3 = (3, 4, 5, 6)$ ,  $F_4 = (6, 7, 8)$ ,  $F_5 = (1, 3, 8)$ ,  $F_6 = (3, 6, 8)$ .

*Lösung:*

- Die Skizze finden Sie in Abbildung 4.8.
- Für die Kantenliste werden die Kanten neu nummeriert durch Zeiger auf ihre Eckpunkte; darüberhinaus darf man nicht vergessen, die Orientierung der Kanten anzugeben!

$$E_1 = (1, 2; 1, N), E_2 = (2, 3; 1, N), E_3 = (3, 1; 1, N),$$

$$E_4 = (2, 4; 2, N), E_5 = (3, 4; 3, 2), E_6 = (4, 5; 3, N),$$

$$E_7 = (5, 6; 3, N), E_8 = (3, 6; 6, 3), E_9 = (6, 7; 4, N),$$

$$E_{10} = (7, 8; 4, N), E_{11} = (6, 8; 6, 4), E_{12} = (3, 8; 5, 6), E_{13} = (1, 8; N, 5).$$

Jetzt können die Facetten durch Verweise auf die nummerierten Kanten angegeben werden:

$$F_1 = (1, 2, 3), F_2 = (2, 4, 5), F_3 = (5, 6, 7, 8),$$

$$F_4 = (9, 10, 11), F_5 = (3, 12, 13), F_6 = (8, 11, 12).$$

- In der Tabelle 4.2 finden Sie die 10 Ecken  $V_0, \dots, V_9$ . Skizzieren Sie das polygonale Netz für die folgende Eckenliste:  $F_0 = (0, 2, 1)$ ,  $F_1 = (0, 3, 2)$ ,  $F_2 = (2, 5, 6, 1)$ ,  $F_3 = (3, 4, 5, 2)$ ,  $F_4 = (1, 6, 8)$ ,  $F_5 = (4, 7, 6, 5)$ ,  $F_6 = (7, 9, 6)$ .

Stellen Sie für die gegebene Facettennummerierung und die folgende Kantennummerierung die Kantenliste und die doppelt verkettete Kantenliste für dieses Netz auf:  $E_0 = (0, 1)$ ,  $E_1 = (0, 2)$ ,  $E_2 = (0, 3)$ ,  $E_3 = (1, 2)$ ,  $E_4 = (2, 3)$ ,  $E_5 = (1, 8)$ ,  $E_6 = (1, 6)$ ,  $E_7 = (2, 5)$ ,  $E_8 = (3, 4)$ ,  $E_9 = (6, 8)$ ,  $E_{10} = (5, 6)$ ,  $E_{11} = (4, 7)$ ,  $E_{12} = (6, 7)$ ,  $E_{13} = (6, 9)$ ,  $E_{14} = (7, 9)$ ,  $E_{15} = (4, 5)$ .

Tabelle 4.2: Die Eckenkoordinaten für Aufgabe 2

0	1	2	3	4	5	6	7	8	9
(0,0)	(1,2)	(2,1)	(2,0)	(3,0)	(3,1)	(4,2)	(4,0)	( $\frac{5}{2}$ , 4)	(5,0)

*Lösung:*

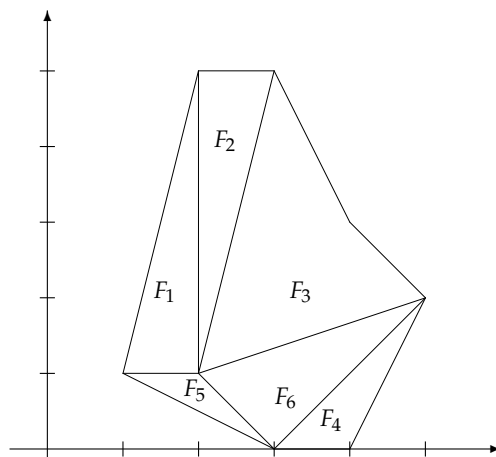


Abbildung 4.8: Die Skizze des Netzes in Aufgabe 1 a)

- (a) Die Lösung finden Sie in Abbildung 4.9. Für die Skizze benötigen Sie die Eckenkoordinaten nicht, sie sind nur zur besseren Orientierung oder falls Sie die Skizze nochmals auf einem separaten Blatt machen möchten mit angeben!
- (b) Die Nummerierung der Facetten wie in Teilaufgabe a) wird weiter verwendet. Dann ergeben sich die folgenden Kanten für die Kantenliste:

$$\begin{aligned}
 E_0 &= (0, 1; N, 0), E_1 = (0, 2; 0, 1), E_2 = (0, 3; 1, N), E_3 = (1, 2; 2, 0), \\
 E_4 &= (2, 3; 3, 1), E_5 = (1, 8; N, 4), E_6 = (1, 6; 4, 2), E_7 = (2, 5; 2, 3), \\
 E_8 &= (3, 4; 3, N), E_9 = (6, 8; 4, N), E_{10} = (5, 6; 2, 5), E_{11} = (4, 7; 5, N), \\
 E_{12} &= (6, 7; 6, 5), E_{13} = (6, 9; N, 6), E_{14} = (7, 9; 6, N), E_{15} = (4, 5; 3, 5).
 \end{aligned}$$

Jetzt können die Facetten als Zeiger auf diese Ecken dargestellt werden:

$$\begin{aligned}
 F_0 &= (1, 3, 0), F_1 = (2, 4, 1), F_2 = (3, 7, 10, 6), F_3 = (8, 15, 7, 4), \\
 F_4 &= (6, 9, 5), F_5 = (15, 11, 12, 10), F_6 = (12, 14, 13).
 \end{aligned}$$

3. Berechnen Sie die Facettennormale für das durch  $V_0 = (1, 1, 2)$ ,  $V_1 = (2, 0, 5)$ ,  $V_2 = (5, 1, 4)$  und  $V_3 = (6, 0, 7)$  gegebene Viereck mit Hilfe des Newell-Verfahrens! Ist das Viereck planar? Wenn ja, vergleichen Sie Ihr Ergebnis mit der Normale, die Sie mit Hilfe des Vektorprodukts berechnen!

*Lösung:*

Das Vektorprodukt ergibt das unnormierte Ergebnis  $(-2, 10, 4)^T$ , durch Normieren erhalten wir  $(-0.182574, 0.912871, 0.365148)^T$ .

Newell ergibt als Ergebnis den Nullvektor.

Trotzdem liegen die vier Punkte in einer Ebene. Die Matrix  $\begin{pmatrix} 1 & 1 & 2 \\ 2 & 0 & 5 \\ 5 & 1 & 4 \\ 6 & 0 & 7 \end{pmatrix}$  hat den Rang 3. Es gilt  $\mathbf{v}_3 = -\mathbf{v}_0 + \mathbf{v}_1 + \mathbf{v}_2$ .

4. Gegeben ist ein polygonales Netz mit den Eckenkoordinaten in Tabelle 4.3 und der Eckenliste  $F_1 = (0, 1, 2, 3)$ ,  $F_2 = (3, 2, 4, 5)$ ,  $F_3 = (5, 4, 6, 7)$ . Berechnen Sie die Facettennormalen  $\mathbf{n}_1$ ,  $\mathbf{n}_2$  und  $\mathbf{n}_3$  für die drei gegebenen Facetten mit Hilfe des Newell-Verfahrens! Berechnen Sie die Eckennormalen für die Ecken  $V_2$ ,  $V_3$ ,  $V_4$  und  $V_5$ !

*Lösung:*

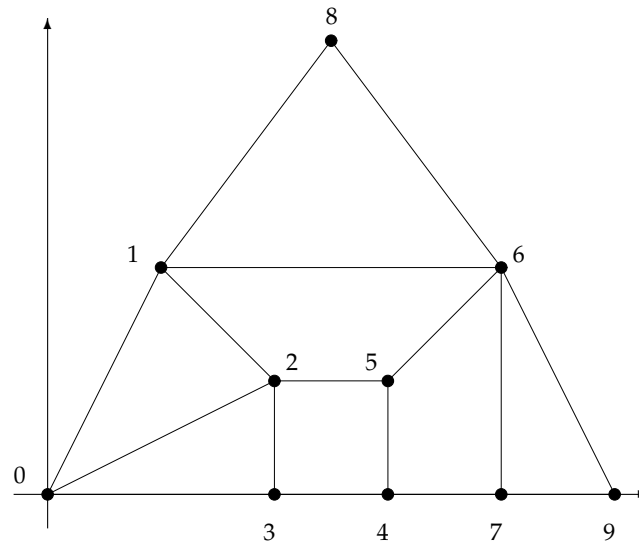


Abbildung 4.9: Die Skizze des Netzes für Aufgabe 2

Tabelle 4.3: Die Eckenkoordinaten für Aufgabe 4

	$V_0$	$V_1$	$V_2$	$V_3$	$V_4$	$V_5$	$V_6$	$V_7$
$x$	0	0	1	1	2	2	3	3
$y$	0	0	1	1	0	0	$\frac{1}{2}$	$\frac{3}{4}$
$z$	0	1	1	0	1	0	$\frac{1}{2}$	$\frac{1}{4}$

- (a) Für die Berechnungen mit dem Newell-Verfahren schreiben wir die benötigten Ecken nochmals in ein Schema.

Für  $\mathbf{n}_1$ , also die Normale auf der Facette  $F_1$  benötigen wir die Eckenkoordinaten im Schema

	$V_0$	$V_1$	$V_2$	$V_3$
$x$	0	0	1	1
$y$	0	0	1	1
$z$	0	1	1	0

Dann erhalten wir

$$\begin{aligned}
 n_x &= \frac{1}{2}((0-0)(0+1) + (0-1)(1+1) + (1-1)(1+0) + (1-0)(0+0)) \\
 &= \frac{1}{2} \cdot (-2) \\
 &= -1.
 \end{aligned}$$

$$\begin{aligned}
 n_y &= \frac{1}{2}((0-1)(0+0) + (1-1)(0+1) + (1-0)(1+1) + (0-0)(1+0)) \\
 &= \frac{1}{2} \cdot 2 \\
 &= 1.
 \end{aligned}$$

$$\begin{aligned}
 n_z &= \frac{1}{2}((0-0)(0+0) + (0-1)(0+1) + (1-1)(1+1) + (1-0)(1+0)) \\
 &= \frac{1}{2} \cdot (-1+1) \\
 &= 0.
 \end{aligned}$$

Anschließendes Normieren ergibt

$$\mathbf{n}_1 = \frac{1}{2}\sqrt{2} \begin{pmatrix} -1 \\ 1 \\ 0 \end{pmatrix} \approx \begin{pmatrix} -0,707 \\ 0,707 \\ 0 \end{pmatrix}.$$

Für  $\mathbf{n}_2$ , also die Normale auf der Facette  $F_2$  benötigen wir die Eckenkoordinaten im Schema

	$V_3$	$V_2$	$V_4$	$V_5$
$x$	1	1	2	2
$y$	1	1	0	0
$z$	0	1	1	0

Dann erhalten wir

$$\begin{aligned}
 n_x &= \frac{1}{2}((1-1)(0+1) + (1-0)(1+1) + (0-0)(1+0) + (0-1)(0+0)) \\
 &= \frac{1}{2} \cdot 2 \\
 &= 1.
 \end{aligned}$$

$$\begin{aligned}
 n_y &= \frac{1}{2}((0-1)(1+1) + (1-1)(1+2) + (1-0)(2+2) + (0-0)(2+1)) \\
 &= \frac{1}{2}(-2+4) \\
 &= \frac{1}{2} \cdot 2 \\
 &= 1.
 \end{aligned}$$

$$\begin{aligned}
 n_z &= \frac{1}{2}((1-1)(1+1) + (1-2)(1+0) + (2-2)(0+0) + (2-1)(0+1)) \\
 &= \frac{1}{2} \cdot (-1+1) \\
 &= 0.
 \end{aligned}$$

Anschließendes Normieren ergibt

$$\mathbf{n}_2 = \frac{1}{2}\sqrt{2} \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} \approx \begin{pmatrix} 0,707 \\ 0,707 \\ 0 \end{pmatrix}.$$

Für  $\mathbf{n}_3$ , also die Normale auf der Facette  $F_3$  benötigen wir die Eckenkoordinaten im Schema

	$V_5$	$V_4$	$V_6$	$V_7$
$x$	2	2	3	3
$y$	0	0	$\frac{1}{2}$	$\frac{3}{4}$
$z$	0	1	$\frac{1}{2}$	$\frac{1}{4}$

Dann erhalten wir

$$\begin{aligned}
 n_x &= \frac{1}{2}((0-0)(0+1) + (0-\frac{1}{2})(1+\frac{1}{2}) + (\frac{1}{2}-\frac{3}{4})(\frac{1}{2}+\frac{1}{4}) + (\frac{3}{4}-0)(\frac{1}{4}+0)) \\
 &= \frac{1}{2}(-\frac{1}{2} \cdot \frac{3}{2} - \frac{1}{4} \cdot \frac{3}{4} + \frac{3}{4} \cdot \frac{1}{4}) \\
 &= \frac{1}{2}(-\frac{3}{4} - \frac{3}{16} + \frac{3}{16}) \\
 &= \frac{1}{2} \cdot \left(-\frac{3}{4}\right) \\
 &= -\frac{3}{8}.
 \end{aligned}$$

$$\begin{aligned}
 n_y &= \frac{1}{2}((0-1)(2+2) + (1-\frac{1}{2})(2+3) + (\frac{1}{2}-\frac{1}{4})(3+3) + (\frac{1}{4}-0)(3+2)) \\
 &= \frac{1}{2}(-4 + \frac{5}{2} + \frac{3}{2} + \frac{5}{4}) \\
 &= \frac{1}{2} \cdot \frac{5}{4} \\
 &= \frac{5}{8}.
 \end{aligned}$$

$$\begin{aligned}
 n_z &= \frac{1}{2}((2-2)(0+0) + (2-3)(0+\frac{1}{2}) + (3-3)(\frac{1}{2}+\frac{3}{4}) + (3-2)(\frac{3}{4}+0)) \\
 &= \frac{1}{2} \cdot (0 - \frac{1}{2} + 0 + \frac{3}{4}) \\
 &= \frac{1}{2} \cdot \frac{1}{4} \\
 &= \frac{1}{8}.
 \end{aligned}$$

Anschließendes Normieren ergibt

$$\mathbf{n}_3 \approx \frac{1}{0,7395} \begin{pmatrix} -\frac{3}{8} \\ \frac{5}{8} \\ \frac{1}{8} \end{pmatrix} \approx \begin{pmatrix} -0,507 \\ 0,845 \\ 0,169 \end{pmatrix}.$$

- (b) Die Eckennormalen ergeben sich jeweils durch Mitteln der beteiligten Facettennormalen. Dabei ist dann auf Grund der Topologie  $\mathbf{n}_{V_2} = \mathbf{n}_{V_3}$  und  $\mathbf{n}_{V_4} = \mathbf{n}_{V_5}$ .

$$\begin{aligned}
 \mathbf{n}_{V_2} = \mathbf{n}_{V_3} &= \frac{1}{2}\mathbf{n}_1 + \frac{1}{2}\mathbf{n}_2 \\
 &= \frac{1}{2} \begin{pmatrix} -\frac{1}{2}\sqrt{2} \\ \frac{1}{2}\sqrt{2} \\ 0 \end{pmatrix} + \frac{1}{2} \begin{pmatrix} \frac{1}{2}\sqrt{2} \\ \frac{1}{2}\sqrt{2} \\ 0 \end{pmatrix} \\
 &= \begin{pmatrix} 0 \\ \frac{1}{2}\sqrt{2} \\ 0 \end{pmatrix} \approx \begin{pmatrix} 0 \\ 0,707 \\ 0 \end{pmatrix}.
 \end{aligned}$$

Normieren ergibt das Endergebnis

$$\mathbf{n}_{V_2} = \mathbf{n}_{V_3} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}.$$

$$\begin{aligned} \mathbf{n}_{V_4} = \mathbf{n}_{V_5} &= \frac{1}{2} \mathbf{n}_2 + \frac{1}{2} \mathbf{n}_3 \\ &\approx \frac{1}{2} \begin{pmatrix} 0,707 \\ 0,707 \\ 0 \end{pmatrix} + \frac{1}{2} \begin{pmatrix} -0,507 \\ 0,845 \\ 0,169 \end{pmatrix} \\ &\approx \begin{pmatrix} 0,1 \\ 0,776 \\ 0,0845 \end{pmatrix}. \end{aligned}$$

Normieren ergibt dann das Endergebnis

$$\mathbf{n}_{V_4} = \mathbf{n}_{V_5} \approx \begin{pmatrix} 0,127 \\ 0,765 \\ 0,009 \end{pmatrix}.$$

5. Berechnen Sie einen Normalenvektor für das Viereck mit den Punkten  $V_0 = (0,0,0)$ ,  $V_1 = (0,-a,1)$ ,  $V_2 = (1,a,1)$  und  $V_3 = (1,0,0)$  in Abhängigkeit von  $a$  und interpretieren Sie das Ergebnis!

*Lösung:*

Das Newell-Verfahren ergibt

$$\begin{aligned} n_x &= \frac{1}{2}((0+a) \cdot 1 + (-a-a) \cdot 2 + (a-0) \cdot 1 + 0 \cdot 0) = -\frac{1}{2}2a = -a, \\ n_y &= \frac{1}{2}((0-1) \cdot 0 + (1-1) \cdot 1 + (1-0) \cdot 2 + 0 \cdot 1) = 1, \\ n_z &= \frac{1}{2}(0 \cdot 0 + (-1) \cdot (-a+a) + (1-1) \cdot a + 1 \cdot 0) = 0. \end{aligned}$$

Normiert erhält man dann die Normale

$$\mathbf{n} = \frac{1}{\sqrt{1+a^2}} \begin{pmatrix} -a \\ 1 \\ 0 \end{pmatrix}.$$

Das Viereck ist offensichtlich nur im Fall  $a = 0$  planar. Die Normale in diesem Fall ist gegeben durch die  $y$ -Achse, denn dann liegen alle Punkte in der Ebene  $y = 0$ . Also ist in diesem Fall

$$\mathbf{n}_0 = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}.$$

Das Newell-Verfahren „glättet“ kleine Messfehler bei der Berechnung der Normalen. Für  $a \neq 0$  und der Verwendung der drei Punkte  $V_0$ ,  $V_1$  und  $V_2$  ergibt das Vektorprodukt das folgende Ergebnis:

$$\mathbf{n} = \frac{1}{\sqrt{1+5a^2}} \begin{pmatrix} -2a \\ 1 \\ a \end{pmatrix}.$$

Der Fehler hat sich jetzt in beide Komponenten, die  $x$  und die  $z$ -Komponente fortgepflanzt.

### 4.3 Dreiecksnetze

1. Triangulieren Sie den Würfel in Abbildung 4.2 auf Seite 192 so, dass er durch *einen* Triangle Strip darstellbar ist!



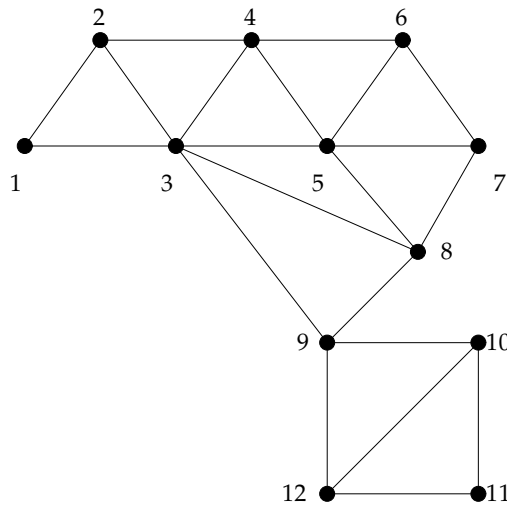


Abbildung 4.11: Die Lösung für Aufgabe 3

## 4.4 Modellieren mit Netzen

1. Welches Objekt ist durch die Parameterdarstellung

$$F(u, v) = \begin{pmatrix} \sqrt{1-v^2} \cos(u) \\ \sqrt{1-v^2} \sin(u) \\ v \end{pmatrix}, (u, v) \in [0, 2\pi) \times [-1, 1].$$

gegeben? Beschreiben Sie die die Iso-Linien!

*Lösung:*

Setzt man die gegebene Parametrisierung in die implizite Darstellung der Kugel in Standardlage ein, erhält man

$$\begin{aligned} F(u, v) &= (1 - v^2) \cos^2(u) + (1 - v^2) \sin^2(u) + v^2 - 1 \\ &= (1 - v^2)(\cos^2(u) + \sin^2(u)) + v^2 - 1 \\ &= 0. \end{aligned}$$

Also ist wieder die Einheitskugel beschrieben.

Die Isolinien für konstantes  $v$  sind wieder Breitenkreise; die Isolinien für konstantes  $u$  sind Halbkreise, die Längenkreisen entsprechen. Allerdings sind diesmal die Abstände zwischen den einzelnen Breitenkreise für Parameterwerte

$$V_j = 1 - j \cdot \frac{2}{n_L - 1}, j = 0, n_L - 1,$$

äquidistant. Bei der Tessellation der Kugel mit der Parametrisierung wie im Buch ist die Bogenlänge zwischen den einzelnen Schichten konstant. Dies ergibt eine erhebliche bessere visuelle Qualität. Wenn Sie die Darstellung in Abbildung 4.12 betrachten, fällt dies vor allem im Polbereich auf. Dort ist die Abbildung aus dem Buch sehr viel „runder“.

Eine Implementierung dieser Tessellation in OpenGL finden Sie bei den Quellcodes zu diesem Kapitel!

2. Vergleichen Sie die planaren Graphen in den Abbildungen 4.29 und 4.31! Worin unterscheiden sich die Netze, was haben sie gemeinsam?

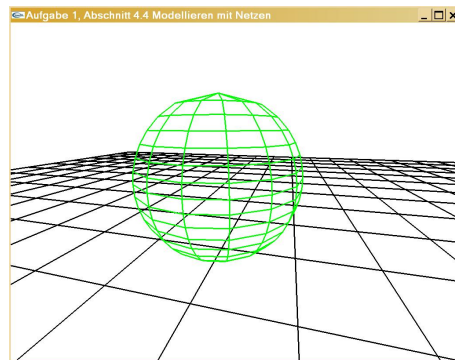


Abbildung 4.12: Das Ergebnis der Parametrisierung in Aufgabe 1

*Lösung:*

Verwendet man für Kugel und Zylinder die gleiche Auflösungen  $n_B$  und  $n_L$ , dann erhält man identische planare Graphen. Also ist die *Topologie* der beiden Tesselationen gleich. Nur die *Geometrie* der beiden Netze unterscheidet sich; die Punktkoordinaten sind durch die jeweiligen Parameterdarstellungen gegeben.

3. Stellen Sie die Eckenliste eines Zylinders und eines Kegels in Standardlage mit  $n_B = 8$  und  $n_L = 2$  auf!

*Lösung:*

### Zylinder

Wenn die Nummerierung wie im Buch vorgeschlagen wird, erhält man die 17 Punktkoordinaten als

$$V_0 = (0, 0, 1), V_{17} = (0, 0, 0),$$

$$V_k = \left( \cos\left(\frac{(k-1)\pi}{4}\right), \sin\left(\frac{(k-1)\pi}{4}\right), 1 \right), k = 1, \dots, 8$$

$$V_k = \left( \cos\left(\frac{(k-9)\pi}{4}\right), \sin\left(\frac{(k-9)\pi}{4}\right), 0 \right), k = 9, \dots, 16.$$

In Tabelle 4.6 finden Sie die Eckenliste; Abbildung 4.13 den planaren Graphen.

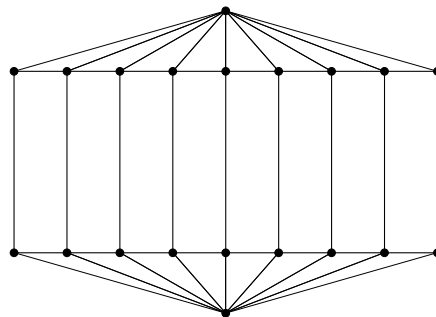


Abbildung 4.13: Der planare Graph für den Standardzylinder für  $n_B = 8$ ,  $n_L = 2$  in Aufgabe 3

**Tabelle 4.6:** Die Eckenliste für die Tesselation des Zylinders in Standardlage mit  $n_B = 8$ ,  $n_L = 2$ 

0	0, 1, 2
1	0, 2, 3
2	0, 3, 4
3	0, 4, 5
4	0, 5, 6
5	0, 6, 7
6	0, 7, 8
7	0, 8, 1
8	1, 9, 10, 2
9	2, 10, 11, 3
10	3, 11, 12, 4
11	4, 12, 13, 5
12	5, 13, 14, 6
13	6, 14, 15, 7
14	7, 15, 16, 8
15	8, 16, 9, 1
16	17, 10, 9
17	17, 11, 10
18	17, 12, 11
19	17, 13, 12
20	17, 14, 13
21	17, 15, 14
22	17, 16, 15
23	17, 9, 16

Beim Kegel fallen die Vierecke komplett weg, die Tesselation besteht aus zwei Triangle Fans. Die Punktkoordinaten der 10 Punkte sind gegeben als

$$\begin{aligned}
 V_0 &= (0, 0, 1), \\
 V_k &= \left( \cos\left(\frac{(k-1)\pi}{4}\right), \sin\left(\frac{(k-1)\pi}{4}\right), 0 \right), k = 1, \dots, 8 \\
 V_{10} &= (0, 0, 0).
 \end{aligned}$$

In Tabelle 4.7 finden Sie die Eckenliste; Abbildung 4.14 den planaren Graphen.

**Tabelle 4.7:** Die Eckenliste für die Tesselation des Kegels in Standardlage mit  $n_B = 8$ ,  $n_L = 2$ 

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

4. Beschreiben Sie das Ergebnis der Anwendung eines Tapers entlang der z-Achse mit  $f_{Taper}(x) = x + 1$  und eines Twists um die z-Achse mit  $f_{Twist}(x) = \frac{\pi}{2}(x + 1)$  auf einen Würfel mit den Eck-

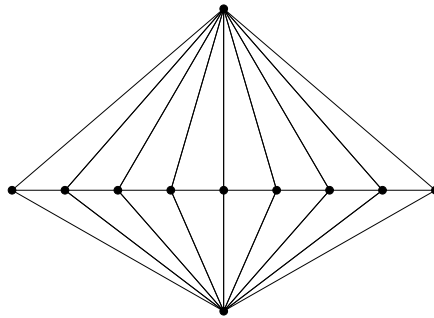


Abbildung 4.14: Der planare Graph für den Kegel für  $n_B = 8, n_L = 2$

punkten  $(\pm 1, \pm 1, \pm 1)$ !

*Lösung:*

Die Matrix eines Tapers entlang der  $z$ -Richtung ist gegeben durch

$$T_z = \begin{pmatrix} f(x_3) & 0 & 0 \\ 0 & f(x_3) & 1 \\ 0 & 0 & 1 \end{pmatrix}$$

angewandt auf Punkte  $(x_1, x_2, x_3)$ .

Für die gegebene Funktion sind die Funktionswerte für 1 und  $-1$  interessant; es gilt

$$f(1) = 2, f(-1) = 0.$$

Also werden die Punkte in der Ebene  $z = -1$  auf einen Punkte, den Ursprung, abgebildet. Für eine Funktion  $f$ , für die  $f(-1) < 0$  erfüllt wäre würde der Würfel sogar „umgestülpt“, was Sie sich an Hand der Funktion  $f(x) = x - \frac{1}{2}$  klar machen sollten.

Hier das Ergebnis für  $f(x) = x + 1$ :

$$\begin{aligned} T_z(1, 1, 1) &= (2, 2, 1), \\ T_z(-1, 1, 1) &= (-2, 2, 1), \\ T_z(-1, -1, 1) &= (-2, -2, 1), \\ T_z(1, -1, 1) &= (2, -2, 1). \end{aligned}$$

Für die Punkte in der Ebene  $z = 1$  entspricht der gegebene Taper einer gleichmäßigen Skalierung um den Faktor 2.

Die Punkte des Würfels in der Ebene  $z = -1$  werden alle auf den Ursprung abgebildet:

$$T_z(\pm 1, \pm 1, -1) = (0, 0, -1).$$

Insgesamt erhalten wir eine auf dem Kopf stehende Pyramide, denn die Taperfunktion  $f$  ist linear, die Kanten werden also nicht verbogen.

Die Twistmatrix ist gegeben durch

$$T_z = \begin{pmatrix} \cos(f(x_3)) & \sin(f(x_3)) & 0 \\ \sin(f(x_3)) & \cos(f(x_3)) & 1 \\ 0 & 0 & 1 \end{pmatrix}$$

angewandt auf Punkte  $(x_1, x_2, x_3)$ .

Für  $z = 1$  ist  $f(1) = \pi$ , die Matrix ist gegeben durch

$$\text{Twist}(1) = \begin{pmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Diese Matrix entspricht einer Drehung mit  $\varphi = \pi$ .

In der Ebene  $z = -1$  gilt  $f(-1) = 0$ , also ist dort die Twist-Matrix gegeben als die Identität:

$$\text{Twist}(-1) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Dazwischen variiert der Drehwinkel linear, in der Ebene  $z = 0$  ist der Twist eine Drehung um  $\varphi = \frac{\pi}{2}$ .

5. Weisen Sie nach, dass ein Twist volumenerhaltend ist!

*Lösung:*

Wir betrachten einen Twist um die  $z$ -Achse, für die beiden anderen Fälle ist das Ergebnis analog herzuleiten.

Die lokale Volumenveränderung wird durch die Jacobi-Matrix  $J$  beschrieben. Diese ist für den Twist um die  $z$ -Achse gegeben durch

$$J = \begin{pmatrix} \cos(f(x_3)) & -\sin(f(x_3)) & -\sin(f(x_3))f'(x_3)x_1 - \cos(f(x_3))f'(x_3)x_2 \\ \sin(f(x_3)) & \cos(f(x_3)) & \cos(f(x_3))f'(x_3)x_1 - \sin(f(x_3))f'(x_3)x_2 \\ 0 & 0 & 1 \end{pmatrix}.$$

Dann ist die Determinante gegeben als

$$\det(J)(x_1, x_2, x_3) = \begin{vmatrix} \cos(f(x_3)) & -\sin(f(x_3)) \\ \sin(f(x_3)) & \cos(f(x_3)) \end{vmatrix} = \cos^2(f(x_3)) + \sin^2(f(x_3)) = 1.$$

6. Führen Sie zwei Subdivision-Schritte für den zweidimensionalen Polygonzug  $\{(0,1), (1,0), (2,1), (3,0)\}$  durch und skizzieren Sie das Ergebnis!

*Lösung:*

In Abbildung 4.15 finden Sie eine Darstellung des Ausgangspolygonzugs.

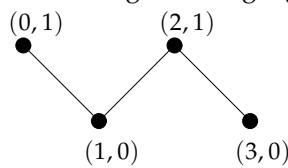


Abbildung 4.15: Der Polygonzug für Aufgabe 6

Die erste Unterteilung liefert die neuen Punkte

$$\begin{aligned} V'_1 &= V_1 = (0,1), \\ V'_2 &= \frac{1}{2}(V_1 + V_2) = \left(\frac{1}{2}, \frac{1}{2}\right), \\ V'_3 &= V_2 = (1,0), \\ V'_4 &= \frac{1}{2}(V_2 + V_3) = \left(\frac{3}{2}, \frac{1}{2}\right), \\ V'_5 &= V_3 = (2,1), \\ V'_6 &= \frac{1}{2}(V_3 + V_4) = \left(\frac{5}{2}, \frac{1}{2}\right), \\ V'_7 &= V_4 = (3,0). \end{aligned}$$

Durch das Mitteln sind die Punkte der ersten Subdivisionsstufe dann gegeben als

$$\begin{aligned} V'_1 &= \left(\frac{1}{4}, \frac{3}{4}\right), \\ V'_2 &= \left(\frac{1}{2}, \frac{1}{2}\right), \\ V'_3 &= \left(1, \frac{1}{2}\right), \\ V'_4 &= \left(\frac{3}{2}, \frac{1}{2}\right), \\ V'_5 &= 2, \frac{3}{4}, \\ V'_6 &= \left(\frac{5}{2}, \frac{1}{2}\right), \\ V'_7 &= \left(\frac{11}{4}, \frac{1}{4}\right). \end{aligned}$$

Im zweiten Subdivisionsschritt wird zuerst einmal unterteilt:

$$\begin{aligned} V''_1 &= V'_1, \\ V''_2 &= \frac{1}{2}(V'_1 + V'_2) = \left(\frac{3}{8}, \frac{5}{8}\right), \\ V''_3 &= V'_2, \\ V''_4 &= \frac{1}{2}(V'_2 + V'_3) = \left(\frac{3}{4}, \frac{3}{8}\right), \\ V''_5 &= V'_3, \\ V''_6 &= \frac{1}{2}(V'_3 + V'_4) = \left(\frac{5}{4}, \frac{3}{8}\right), \\ V''_7 &= V'_4, \\ V''_8 &= \frac{1}{2}(V'_4 + V'_5) = \left(\frac{7}{4}, \frac{5}{8}\right), \\ V''_9 &= V'_5, \\ V''_{10} &= \frac{1}{2}(V'_5 + V'_6) = \left(\frac{9}{4}, \frac{5}{8}\right), \\ V''_{11} &= V'_6, \\ V''_{12} &= \frac{1}{2}(V'_6 + V'_7) = \left(\frac{21}{8}, \frac{3}{8}\right), \\ V''_{13} &= V'_7. \end{aligned}$$

Durch das anschließende Mitteln ergibt sich das Ergebnis

$$\begin{aligned} V''_1 &= \left(\frac{5}{16}, \frac{11}{16}\right), \\ V''_2 &= \left(\frac{3}{8}, \frac{5}{8}\right), \\ V''_3 &= \left(\frac{17}{32}, \frac{1}{2}\right), \\ V''_4 &= \left(\frac{3}{4}, \frac{3}{8}\right), \\ V''_5 &= \left(1, \frac{5}{16}\right), \end{aligned}$$

$$\begin{aligned}
 V_6'' &= \left(\frac{5}{4}, \frac{3}{8}\right), \\
 V_7'' &= \left(\frac{3}{2}, \frac{1}{2}\right), \\
 V_8'' &= \left(\frac{7}{4}, \frac{5}{8}\right), \\
 V_9'' &= \left(2, \frac{11}{16}\right), \\
 V_{10}'' &= \left(\frac{9}{4}, \frac{5}{8}\right), \\
 V_{11}'' &= \left(\frac{79}{32}, \frac{1}{2}\right), \\
 V_{12}'' &= \left(\frac{21}{8}, \frac{3}{8}\right), \\
 V_{13}'' &= \left(\frac{43}{16}, \frac{5}{16}\right).
 \end{aligned}$$

In Abbildung 4.16 finden Sie eine graphische Darstellung des Ergebnisses!

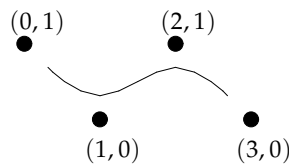


Abbildung 4.16: Die Ausgangsform und das Ergebnis nach zwei Subdivisionsschritten für Aufgabe 6

## 4.5 Level-of-Detail und Vereinfachen von Netzen

1. Beschreiben Sie eine Funktion, die für einen Punkt den Abstand zu einer Linie oder zu einem Polygonzug bestimmt!

*Lösung:*

Wir betrachten zuerst den Fall der Linie. Angenommen, der Abstand zwischen dem Punkt  $V$  und der Linie, gegeben als

$$L = \{A \mid A = V_0 + t\mathbf{v}, t \in \mathbb{R}\}$$

soll bestimmt werden. Wir nehmen an, dass der Richtungsvektor  $\mathbf{v}$  normiert ist.

Dann wird der kürzeste Abstand zwischen  $V$  und der Linie  $L$  an der Orthogonalprojektion von  $V$  auf  $L$  angenommen. Es gilt

$$\text{ortho}_V = V_0 + t_0\mathbf{v}$$

mit

$$t_0 = \langle \mathbf{v}, \mathbf{V}_0\mathbf{V} \rangle$$

Der gesuchte Abstand ist dann gegeben durch

$$\text{dist}(V, L) = \|(V_0 + t_0\mathbf{v}) - V\|.$$

Ist nicht eine Linie, sondern nur das Segment zwischen zwei Punkten  $V_1$  und  $V_2$  gegeben, dann kann der kürzeste Abstand am Rand angenommen werden. Es ist  $\mathbf{v} = \mathbf{V}_1\mathbf{V}_2$ . Auch hier sollte dieser Richtungsvektor normiert werden.

Ist  $t_0 < 0$ , dann wird der kürzeste Abstand im Punkt  $V_1$  angenommen; ist  $t_0 > 1$ , im Punkt  $V_2$ . Sonst liegt die Orthogonalprojektion zwischen  $V_1$  und  $V_2$ .

Ist die Gerade durch eine Normalenform gegeben, dann kann leicht der orientierte Abstand bestimmt werden. Mehr dazu finden Sie in Brill, Mathematik für Informatiker, Abschnitt 9.3.

Im Fall eines Polygonzugs  $P$  wird jetzt zu jedem Segment  $P_i$  der Abstand berechnet; das Segment, das den kürzesten Abstand liefert definiert den gesuchten Abstand:

$$\text{dist}(V, P) = \min_i \text{dist}(V, P_i).$$

2. Beschreiben Sie eine Funktion, die den Abstand zwischen einem Punkt und einem Dreieck bestimmt!

*Lösung:*

Möglich wäre wie in der Lösung von Aufgabe 1, die Orthogonalprojektion des Punkts  $V$  auf die Ebene zu bestimmen, die durch das Dreieck gegeben ist; und anschließend zu überprüfen, ob diese Orthogonalprojektion im Innern oder auf dem Rand des Dreiecks liegt.

Diese Überprüfung kann sinnvollerweise mit Hilfe von baryzentrischen Koordinaten erfolgen, wie sie in Kapitel 2 eingeführt wurden.

Sind  $V_1, V_2$  und  $V_3$  die Punkte des Dreiecks, dann bilden wir die zwei Richtungsvektoren  $\mathbf{v}_1 = \mathbf{V}_1\mathbf{V}_2$  und  $\mathbf{v}_2 = \mathbf{V}_1\mathbf{V}_3$ . Das Dreieck (inklusive seines Randes) ist dann gegeben als

$$D = \{A \in A^3 \mid A = V_1 + s\mathbf{v}_1 + t\mathbf{v}_2, s, t \in [0, 1], s + t \leq 1\}.$$

Damit können wir den Abstand zwischen dem Punkt  $V$  und  $D$  als Minimierungsproblem in  $s$  und  $t$  mit Nebenbedingungen formulieren:

Finde  $s, t \in [0, 1], s + t \leq 1$  so, dass

$$Q(s, t) = \|V_1 + s\mathbf{v}_1 + t\mathbf{v}_2 - V\|$$

minimal wird. Die Funktion  $Q$  ist quadratisch in  $s$  und  $t$ :

$$Q(s, t) = as^2 + 2bst + ct^2 + 2ds + 2et + f,$$

mit

$$\begin{aligned} a &= \|\mathbf{v}_1\|, \\ b &= \langle \mathbf{v}_1, \mathbf{v}_2 \rangle, \\ c &= \|\mathbf{v}_2\|, \\ d &= \langle \mathbf{v}_1, \mathbf{V}\mathbf{V}_1 \rangle, \\ e &= -\langle \mathbf{v}_2, \mathbf{V}\mathbf{V}_1 \rangle, \\ f &= \|\mathbf{V}\mathbf{V}_1\|. \end{aligned}$$

Die Diskriminante dieses Kegelschnitts ist  $ac - b^2 = \|\mathbf{v}_1 \times \mathbf{v}_2\|^2$ . Ist wirklich ein Dreieck gegeben, sind also die beiden Richtungsvektoren linear unabhängig, dann ist die Diskriminante  $> 0$ .

Das gesuchte Minimum wird an einer Nullstelle des Gradienten oder am Rand des Dreiecksbereichs mit  $s + t = 1$  angenommen. Der Gradient ist gegeben durch

$$\nabla Q(s, t) = 2 \begin{pmatrix} as + bt + d \\ bs + c + e \end{pmatrix}.$$

Eine Nullstelle des Gradienten ist gegeben durch

$$s = \frac{be - cd}{ac - b^2}, t = \frac{bd - ae}{ac - b^2}.$$

Liegen diese Werte innerhalb des gegebenen Bereichs, ist das Minimum gefunden; der kürzeste Abstand wird im Innern des Dreiecks angenommen. Andernfalls muss der Rand des zulässigen Bereichs für  $s$  und  $t$  überprüft werden und dort das Minimum bestimmt werden.

- Wie können Attribute wie Farben oder Normalenvektoren von Eckpunkten eines Netzes bei einem Edge Collapse berücksichtigt werden?

*Lösung:*

Die neuen Attribute sollten auf Grund der Attribute der beiden Dreiecke bestimmt werden, die durch den Edge Collapse verschwinden. Liegen die beiden Dreiecke nicht in einer Ebene, dann bilden die vier Punkte  $V_i, V_j, V_k$  und  $V_l$  einen Tetraeder. Wie in Kapitel 2 dargestellt, können alle Punkte im Tetraeder und auf dem Rand mit Hilfe von baryzentrischen Koordinaten beschrieben werden.

Dann können wir für den neuen Punkt  $V'_j$ , der durch den Edge Collapse entsteht, die zugehörigen baryzentrischen Koordinaten bestimmen. Die neuen Attribute werden ebenfalls mit Hilfe dieser baryzentrischen Koordinaten berechnet. Ist also

$$V'_j = \lambda_1 V_i + \lambda_2 V_j + \lambda_3 V_k + \lambda_4 V_l, \lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 = 1,$$

dann wird ein neues skalares Attribut  $a'_j$ , beispielsweise die RGB-Werte einer Farbe, durch

$$a'_j = \lambda_1 a_i + \lambda_2 a_j + \lambda_3 a_k + \lambda_4 a_l, \lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 = 1$$

berechnet. Das kann offensichtlich auch für die Normalenvektoren durchgeführt werden (die affine Kombination von Vektoren ergibt einen Vektor als Ergebnis).

Im Fall, dass eine der baryzentrischen Koordinaten negativ ist (was dem Fall entspricht, dass der neue Punkt  $V'_j$  nicht im Innern des Tetraeders liegt), sollten diese baryzentrischen Koordinaten auf 0 oder 1 abgeschnitten werden („clamp“), um sinnvolle Werte für die Attribute zu garantieren – Sie wollen bestimmt keine negativen Farbwerte produzieren ...

## 4.7 Fallstudien

### 4.7.1 Polygonale Netze in OpenGL

- Erstellen Sie für jede der Topologien `GL_TRIANGLES`, `GL_TRIANGLE_STRIP`, `GL_TRIANGLE_FAN`, `GL_QUADS`, `GL_QUAD_STRIP` und `GL_POLYGON` ein kleines OpenGL-Programm. Verändern Sie insbesondere die Default-Definition der Durchlaufrihtung mit `glFrontFace()`!

*Lösung:*

In den folgenden Bildern wurde immer eine Ausgabe der Achsen und der kanonischen Einheitsvektoren integriert. Dabei codiert rot-grün-blau die Reihenfolge  $x$ - $y$ - $z$ . In den Abbildungen ist immer links die Darstellung für `glFrontFace(GL_CCW)`, rechts `glFrontFace(GL_CW)` zu sehen.

#### **GL\_TRIANGLES**

Der Quellcode

```
glBegin(GL_TRIANGLES);
    glVertex3f(0.0, 0.0, 0.0);
    glVertex3f(1.0, 1.0, 0.0);
    glVertex3f(0.0, 1.0, 0.0);
glEnd();
```

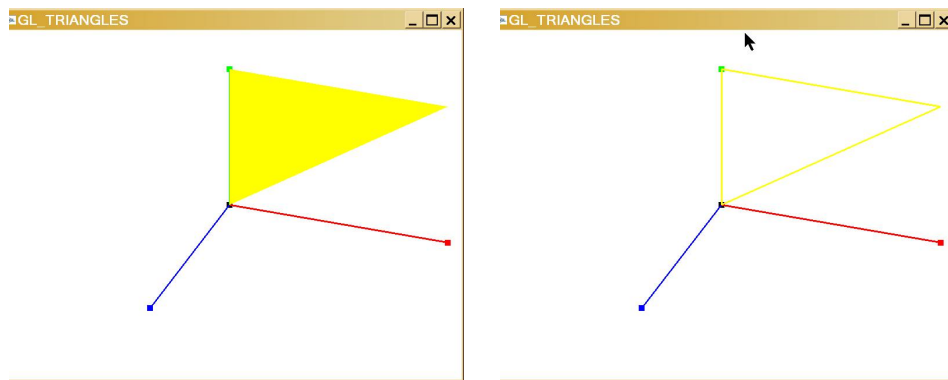


Abbildung 4.17: Ausgabe eines Dreiecks mit Hilfe von `GL_TRIANGLES`

erzeugt die Ausgabe wie in Abbildung 4.17.

### `GL_TRIANGLE_STRIP`

Der Quellcode

```
glBegin(GL_TRIANGLE_STRIP);
    glVertex3f(0.0, 1.0, 0.0);
    glVertex3f(0.0, 0.0, 0.0);
    glVertex3f(1.0, 1.0, 0.0);
    glVertex3f(1.0, 0.0, 0.0);
glEnd();
```

erzeugt die Ausgabe wie in Abbildung 4.18.

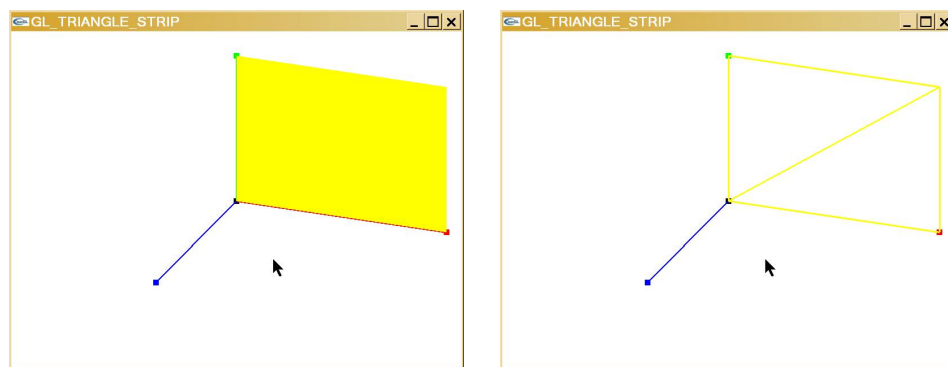


Abbildung 4.18: Ausgabe eines Triangle Strips mit Hilfe von `GL_TRIANGLE_STRIP`

### `GL_TRIANGLE_FAN`

Der Quellcode

```
glColor3f(1.0, 1.0, 0.0);
glBegin(GL_TRIANGLE_FAN);
    glVertex3f(0.0, 0.0, 0.0);
    glVertex3f(1.0, 0.0, 0.0);
    glVertex3f(1.0, 1.0, 0.0);
    glVertex3f(0.0, 1.0, 0.0);
glEnd();
```

erzeugt die Ausgabe wie in Abbildung 4.19.

### `GL_QUADS`

Der Quellcode

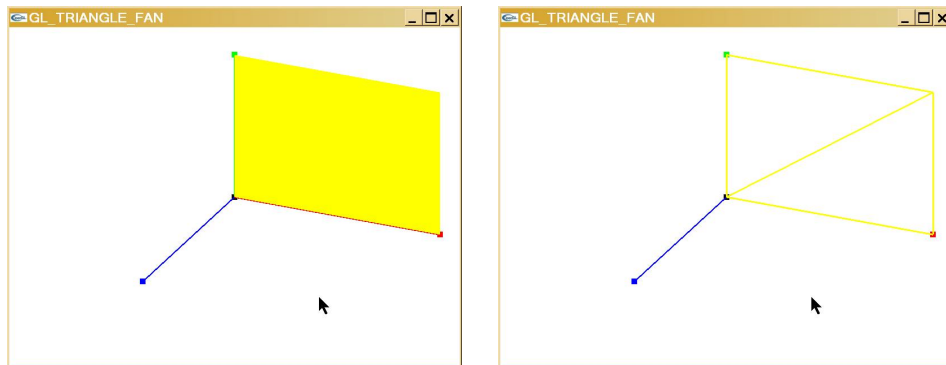


Abbildung 4.19: Ausgabe eines Triangle Fans mit Hilfe von `GL_TRIANGLES_FAN`

```
glBegin(GL_QUADS);
  glVertex3f(0.0, 0.0, 0.0);
  glVertex3f(1.0, 0.0, 0.0);
  glVertex3f(1.0, 1.0, 0.0);
  glVertex3f(0.0, 1.0, 0.0);
glEnd()
```

erzeugt die Ausgabe wie in Abbildung 4.20.

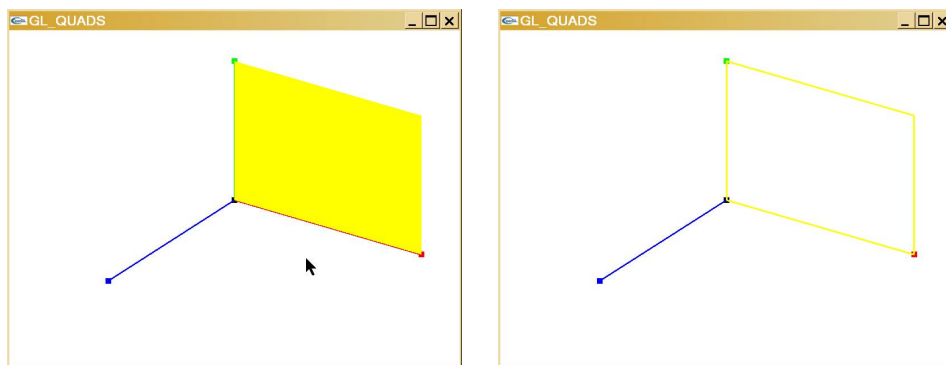


Abbildung 4.20: Ausgabe eines Vierecks mit Hilfe von `GL_QUADS`

## GL\_QUAD\_STRIP

Der Quellcode

```
glBegin(GL_QUAD_STRIP);
  glVertex3f(0.0, 1.0, 0.0);
  glVertex3f(0.0, 0.0, 0.0);
  glVertex3f(0.5, 1.0, 0.0);
  glVertex3f(0.5, 0.0, 0.0);
  glVertex3f(1.0, 1.0, 0.0);
  glVertex3f(1.0, 0.0, 0.0);
glEnd()
```

erzeugt die Ausgabe wie in Abbildung 4.21. Beachten Sie die Reihenfolge der Punkte. In Abbildung 4.22 sehen Sie, was passieren kann, falls Sie die beiden letzten Punkte wie in

```
glBegin(GL_QUAD_STRIP);
  glVertex3f(0.0, 1.0, 0.0);
  glVertex3f(0.0, 0.0, 0.0);
  glVertex3f(0.5, 1.0, 0.0);
```

```

glVertex3f(0.5, 0.0, 0.0);
glVertex3f(1.0, 0.0, 0.0);
glVertex3f(1.0, 1.0, 0.0);
glEnd()

```

vertauschen!

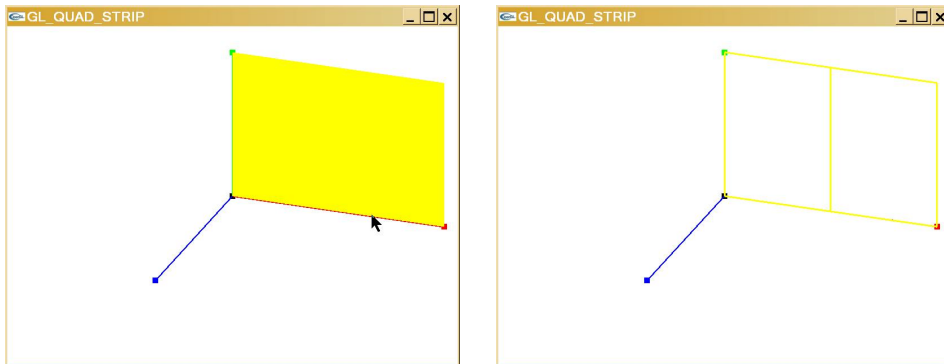


Abbildung 4.21: Ausgabe eines Quad Strips mit Hilfe von GL\_QUADS

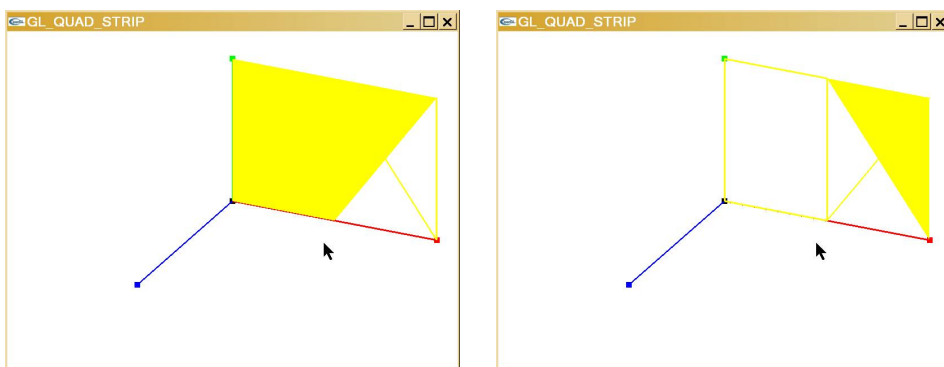


Abbildung 4.22: Ausgabe eines Quad Strips mit Hilfe von GL\_QUADS mit vertauschter Reihenfolge der letzten beiden Punkte

- Geben Sie einen Würfel in einem OpenGL-Programm mit Hilfe von Vertex Arrays und *eines einzigen* Triangle Strips aus!

*Lösung:*

Die Lösung steht in den Code-Fragmenten im Buch. Wenn man davon ausgeht, dass nur der Würfel ausgegeben wird, dann können Sie die Verwendung von Vertex Arrays in der `init`-Funktion aktivieren; die `display`-Funktion greift dann auf die entsprechenden Felder zurück und gibt sie mit Hilfe von `glDrawElements` aus. Hier nochmals einige Code-Abschnitte; den kompletten Quellcode finden Sie bei den Downloads zu diesem Kapitel:

```

void init(void)
{
    glClearColor (1.0, 1.0, 1.0, 1.0);
    glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);
    glLineWidth(2.0);

    glColor3f(0.0, 0.0, 0.0);

    glEnableClientState(GL_VERTEX_ARRAY);
}

```

```

void display(void)
{
    static GLfloat vertices[] = {1.0, 1.0, 1.0, 0.0, 1.0, 1.0,
                                0.0, 0.0, 1.0, 1.0, 0.0, 1.0,
                                1.0, 1.0, 0.0, 0.0, 1.0, 0.0,
                                0.0, 0.0, 0.0, 1.0, 0.0, 0.0};
    static GLshort topology[] = { 2, 3, 6, 7, 4, 3, 0, 2, 1, 6,
                                  5, 4, 1, 0};

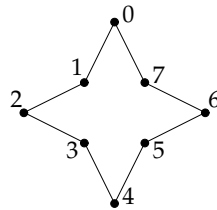
    glClear(GL_COLOR_BUFFER_BIT);

    glVertexPointer(3, GL_FLOAT, 0, vertices);
    glDrawElements(GL_TRIANGLE_STRIP, 14, GL_UNSIGNED_SHORT, topology);

    glutSwapBuffers();
}

```

3. Geben Sie mit Hilfe von Dreiecken und Vierecken das nicht-konvexe Polygon aus Abbildung 4.23 in OpenGL aus! Mit Hilfe der Funktion `glEdgeFlag` können Sie OpenGL angeben, ob die folgenden Eckpunkte zu einer Randkante gehören oder nicht. Verwenden Sie diese Funktion, um eine Darstellung *ohne* innere Kanten wie in Abbildung 4.23 zu erhalten!



0	1	2	3
(0,4)	(-2,0)	(-6,-2)	(-2,-4)

4	5	6	7
(0,-8)	(2,-4)	(6,-2)	(2,0)

Abbildung 4.23: Das Polygon und die Eckpunktkoordinaten für Aufgabe 3

*Lösung:*

Der Stern wird mit Hilfe von vier Dreiecken für die Spitzen und einem Viereck für das Innere ausgegeben. Mit Hilfe von `glEdgeFlag` kann dafür gesorgt werden, dass die *inneren* Kanten nicht mehr dargestellt werden. Hier die Quelltext der `display`-Funktion; die einzelnen Polygone werden der Übersichtlichkeit halber mit unterschiedlichen Farben markiert; die Verwendung der Randkanten kann mit Hilfe der logischen Variable `EDGE` interaktiv verändert werden:

```

void display(void)
{
    glClear(GL_COLOR_BUFFER_BIT);

    glColor3f(1.0, 0.0, 0.0);
    glBegin(GL_TRIANGLES);
        glColor3f(1.0, 0.0, 0.0);
        glEdgeFlag(EDGE);
        glVertex2f(-2.0f, 0.0f);
        glEdgeFlag(true);
        glVertex2f(2.0f, 0.0f);
        glVertex2f(0.0f, 4.0f);

    glColor3f(0.0, 1.0, 0.0);
    glVertex2f(-2.0f, 0.0f);
    glVertex2f(-6.0f, -2.0f);
    glEdgeFlag(EDGE);
    glVertex2f(-2.0, -4.0f);
}

```



### 4.7.2 Fallstudie Platonische und archimedische Körper

1. Geben Sie ausgehend von einem Hexaeder mit Eckpunkten in  $(\pm 1, \pm 1, \pm 1)$  die Eckenkoordinaten eines Oktaeders an!

*Lösung:*

Der planare Graph eines Oktaeders ist im Buch angegeben. Durch die Dualität zwischen Hexaeder und Oktaeder ergeben sich dann die folgenden Eckpunkte:

$$VO_1 = (0, 0, -1),$$

$$VO_2 = (0, 0, 1),$$

$$VO_3 = (-1, 0, 0),$$

$$VO_4 = (1, 0, 0),$$

$$VO_5 = (0, -1, 0),$$

$$VO_6 = (0, 1, 0).$$

2. Begründen Sie, dass für den abgeschnittenen Hexaeder die Seitenlänge der Achtecke durch  $\lambda = 2(\sqrt{2} - 1)$  gegeben ist!

*Lösung:*

$\lambda$  ist die Seitenlänge eines regelmäßigen Achtecks mit Innenkreisradius  $r_i = 1$ , wie in Abbildung 4.25 dargestellt. Die Seitenlänge des umschriebenen Quadrats ist 2.

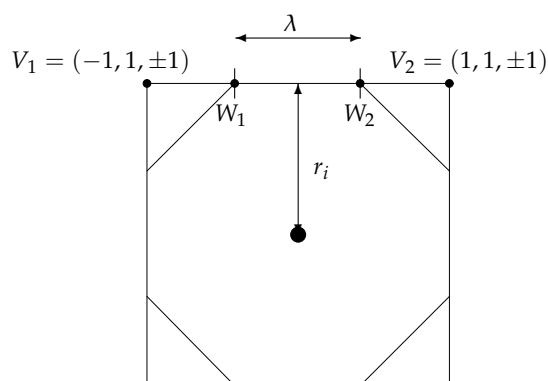


Abbildung 4.25: Die Seitenlänge  $\lambda$  in Aufgabe 2

Für  $\lambda$  gilt dann

$$\lambda = 2r_u \sin\left(\frac{\varphi}{2}\right)$$

mit dem Innenwinkel  $\varphi = 45^\circ$ ;  $r_u$  ist der Umkreisradius des Achtecks. Für diesen Umkreisradius gilt

$$r_u = \frac{r_i}{\cos\left(\frac{\varphi}{2}\right)} = \frac{1}{\sqrt{\frac{1+\cos(\varphi)}{2}}} = \frac{\sqrt{2}}{\sqrt{1 + \frac{1}{2}\sqrt{2}}}.$$

Dann ist

$$\begin{aligned}
 \lambda &= \frac{2\sqrt{2}}{\sqrt{1 + \frac{1}{2}\sqrt{2}}} \sqrt{\frac{1 - \frac{1}{2}\sqrt{2}}{2}} \\
 &= 2 \frac{\sqrt{1 - \frac{1}{2}\sqrt{2}}}{\sqrt{1 + \frac{1}{2}\sqrt{2}}} \\
 &= 2 \frac{\sqrt{1 - \frac{1}{2}}}{1 + \frac{1}{2}\sqrt{2}} \\
 &= \frac{\sqrt{2}}{1 + \frac{1}{2}\sqrt{2}} \\
 &= 2(\sqrt{2} - 1).
 \end{aligned}$$

3. Stellen Sie Ikosaeder und Dodekaeder mit Hilfe von OpenGL dar!

*Lösung:*

Den kompletten Quelltext finden Sie bei den Downloads zu diesem Kapitel; in Abbildung 4.26 sehen Sie das Ergebnis.

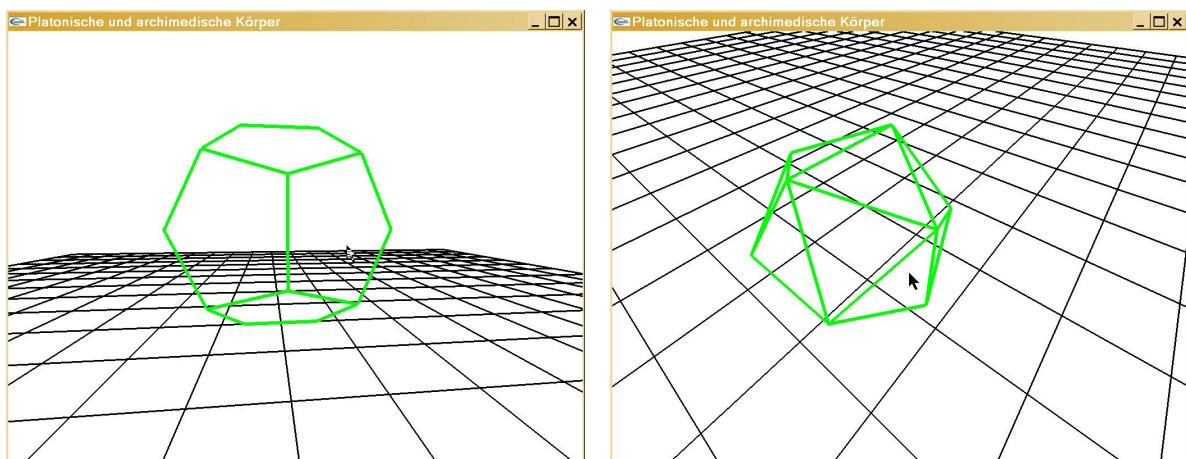


Abbildung 4.26: Dodekaeder und Ikosaeder in OpenGL für Aufgabe 3

Die Eckenliste des Dodekaeders in Feldern, die für Vertex Arrays verwendbar sind:

```

static GLfloat A = (GLfloat) 0.618034f;
static GLfloat B = (GLfloat) 1.0f+A;

// Die Eckenkoordinaten
static GLfloat v[60] =
    { 1.0,  1.0,  1.0,  1.0,  1.0, -1.0,  1.0, -1.0,  1.0,
      1.0, -1.0, -1.0, -1.0,  1.0,  1.0, -1.0,  1.0, -1.0,
     -1.0, -1.0,  1.0, -1.0, -1.0, -1.0,  A,  B,  0.0,
     -A,  B,  0.0,  A,  -B,  0.0, -A, -B,  0.0,
      B,  0.0,  A,  B,  0.0, -A, -B,  0.0,  A,
     -B,  0.0, -A,  0.0,  A,  B,  0.0, -A,  B,
      0.0,  A, -B,  0.0, -A, -B};

// Die Topologie als einzelne Fünfecke

```

```

static GLshort f1[5] = {1, 8, 0, 12, 13};
static GLshort f2[5] = {4, 9, 5, 15, 14};
static GLshort f3[5] = {2, 10, 3, 13, 12};
static GLshort f4[5] = {7, 11, 6, 14, 15};
static GLshort f5[5] = {2, 12, 0, 16, 17};
static GLshort f6[5] = {1, 13, 3, 19, 18};
static GLshort f7[5] = {4, 14, 6, 17, 16};
static GLshort f8[5] = {7, 15, 5, 18, 19};
static GLshort f9[5] = {4, 16, 0, 8, 9};
static GLshort f10[5] = {2, 17, 6, 11, 10};
static GLshort f11[5] = {1, 18, 5, 9, 8};
static GLshort f12[5] = {7, 19, 3, 10, 11};

```

Dann kann der Dodekaeder mit Hilfe der folgenden `display`-Funktion ausgegeben werden:

```

void display(void)
{
    glClear(GL_COLOR_BUFFER_BIT);

    /* Virtuelle Kamera in Funktion examine() */
    examine();

    // Gitter ausgeben
    grid(10);

    glVertexPointer(3, GL_FLOAT, 0, v);

    glDrawElements(GL_POLYGON, 5, GL_UNSIGNED_SHORT, f1);
    glDrawElements(GL_POLYGON, 5, GL_UNSIGNED_SHORT, f2);
    glDrawElements(GL_POLYGON, 5, GL_UNSIGNED_SHORT, f3);
    glDrawElements(GL_POLYGON, 5, GL_UNSIGNED_SHORT, f4);
    glDrawElements(GL_POLYGON, 5, GL_UNSIGNED_SHORT, f5);
    glDrawElements(GL_POLYGON, 5, GL_UNSIGNED_SHORT, f6);
    glDrawElements(GL_POLYGON, 5, GL_UNSIGNED_SHORT, f7);
    glDrawElements(GL_POLYGON, 5, GL_UNSIGNED_SHORT, f8);
    glDrawElements(GL_POLYGON, 5, GL_UNSIGNED_SHORT, f9);
    glDrawElements(GL_POLYGON, 5, GL_UNSIGNED_SHORT, f10);
    glDrawElements(GL_POLYGON, 5, GL_UNSIGNED_SHORT, f11);
    glDrawElements(GL_POLYGON, 5, GL_UNSIGNED_SHORT, f12);

    glutSwapBuffers();
}

```

Die Eckenliste des Ikosaeders in Feldern, die für Vertex Arrays verwendbar sind:

```

// Ecken mit Hilfe des goldenen Schnitts
static GLfloat X = (GLfloat) 0.525731112119133606;
static GLfloat Z = (GLfloat) 0.850650808352039932;

// Die Eckenkoordinaten
static GLfloat v[36] =
    {-X, 0.0, Z, X, 0.0, Z, -X, 0.0, -Z, X, 0.0, -Z,
     0.0, Z, X, 0.0, Z, -X, 0.0, -Z, X, 0.0, -Z, -X,
     Z, X, 0.0, -Z, X, 0.0, Z, -X, 0.0, -Z, -X, 0.0};

// Die Topologie
static GLshort topology[60] =
    { 1,4,0, 4,9,0, 4,5,9, 8,5,4, 1,8,4,
      1,10,8, 10,3,8, 8,3,5, 3,2,5, 3,7,2,
      3,10,7, 10,6,7, 6,11,7, 6,0,11, 6,1,0,

```

```
10,1,6, 11,0,9, 2,11,9, 5,2,9, 11,2,7};
```

Dann kann der Ikosaeder mit Hilfe der folgenden `display`-Funktion ausgegeben werden:

```
void display(void)
{
    glClear(GL_COLOR_BUFFER_BIT);

    /* Virtuelle Kamera in Funktion examine() */
    examine();

    // Gitter ausgeben
    grid(10);

    glVertexPointer(3, GL_FLOAT, 0, v);
    glDrawElements(GL_TRIANGLES, 60, GL_UNSIGNED_SHORT, topology);

    glutSwapBuffers();
}
```

4. Stellen Sie einen abgeschnittenen Hexaeder und einen abgeschnittenen Ikosaeder mit Hilfe von OpenGL dar!

*Lösung:*

Den kompletten Quelltext finden Sie bei den Downloads zu diesem Kapitel; in Abbildung 4.27 sehen Sie das Ergebnis.

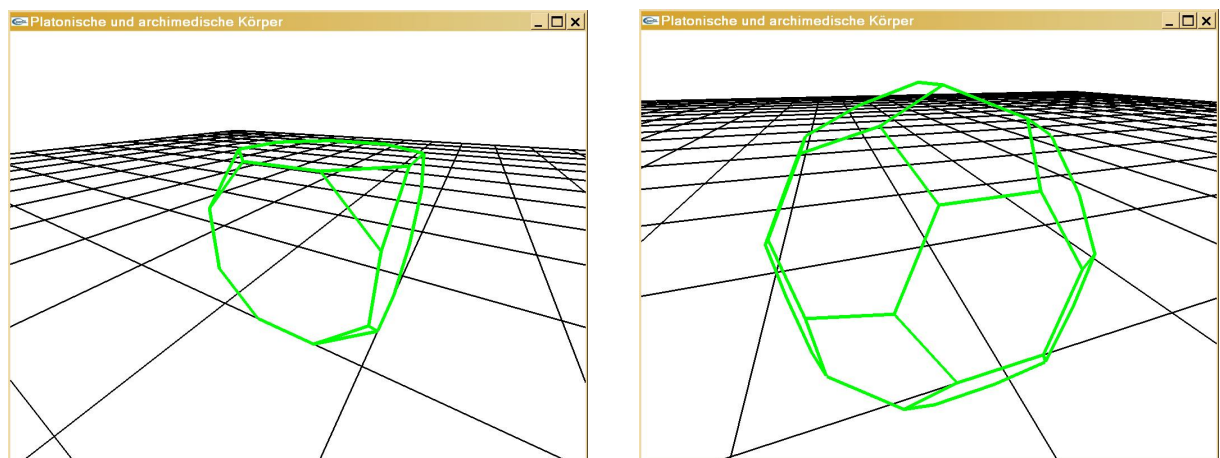


Abbildung 4.27: Abgeschnittener Hexaeder und Ikosaeder in OpenGL für Aufgabe 4

Versuchen Sie insbesondere beim abgeschnittenen Ikosaeder *nicht*, die komplette Eckenliste zu definieren und dann erst darzustellen. Schalten Sie *Backface Culling* ein und fügen Sie sukzessive Polygon für Polygon hinzu!

Die Eckenliste des abgeschnittenen Hexaeders in Feldern, die für Vertex Arrays verwendbar ist:

```
static GLfloat A = 0.292893219, B = 0.707106781,
              ZERO = 0.0, ONE = 1.0;

static GLfloat v[72] = {ZERO, A, ZERO, //0
                       A, ZERO, ZERO, //1
                       ZERO, ZERO, A, //2
                       B, ZERO, ZERO, //3
```

```

ONE, A, ZERO, //4
ONE, ZERO, A, //5
ONE, B, ZERO, //6
B, ONE, ZERO, //7
ONE, ONE, A, //8
A, ONE, ZERO, //9
ZERO, B, ZERO, //10
ZERO, ONE, A, //11
ZERO, ZERO, B, //12
A, ZERO, ONE, //13
ZERO, A, ONE, //14
B, ZERO, ONE, //15
ONE, ZERO, B, //16
ONE, A, ONE, //17
ONE, B, ONE, //18
B, ONE, ONE, //19
ONE, ONE, B, //20
A, ONE, ONE, //21
ZERO, ONE, B, //22
ZERO, B, ONE}; //23

// Die Achtecke, die aus den Vierecken des Würfels entstehen
static GLshort ACHT1[8] = {10, 9, 7, 6, 4, 3, 1, 0};
static GLshort ACHT2[8] = {3, 5, 16, 15, 13, 12, 2, 1};
static GLshort ACHT3[8] = {0, 2, 12, 14, 23, 22, 11, 10};
static GLshort ACHT4[8] = {14, 13, 15, 17, 18, 19, 21, 23};
static GLshort ACHT5[8] = {6, 8, 20, 18, 17, 16, 5, 4};
static GLshort ACHT6[8] = {9, 11, 22, 21, 19, 20, 8, 7};

// Die Dreiecke in den Ecken
static GLshort DREIECKE[24] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11,
                               12, 13, 14, 15, 16, 17, 20, 19, 18,
                               21, 22, 23};

```

Dann kann der abgeschnittene Hexaeder mit Hilfe der folgenden `display`-Funktion ausgegeben werden:

```

void display(void)
{
    glClear(GL_COLOR_BUFFER_BIT);
    /* Virtuelle Kamera in Funktion examine() */
    examine();
    // Gitter ausgeben
    grid(10);

    glLineWidth(6.0);
    glColor3f(0.0, 1.0, 0.0);

    // Die Geometrie
    glVertexPointer(3, GL_FLOAT, 0, v);

    // Die Topologie
    // Die Achtecke, die aus den Vierecken des Würfels entstehen
    glDrawElements(GL_POLYGON, 8, GL_UNSIGNED_SHORT, ACHT1);
    glDrawElements(GL_POLYGON, 8, GL_UNSIGNED_SHORT, ACHT2);
    glDrawElements(GL_POLYGON, 8, GL_UNSIGNED_SHORT, ACHT3);
    glDrawElements(GL_POLYGON, 8, GL_UNSIGNED_SHORT, ACHT4);
    glDrawElements(GL_POLYGON, 8, GL_UNSIGNED_SHORT, ACHT5);
    glDrawElements(GL_POLYGON, 8, GL_UNSIGNED_SHORT, ACHT6);
}

```

```

glDrawElements(GL_TRIANGLES, 24, GL_UNSIGNED_SHORT, DREIECKE);

glutSwapBuffers();
}

```

Die Eckenliste des abgeschnittenen Ikosaeders in Feldern, die für Vertex Arrays verwendbar ist:

```

GLfloat v[] = {
    0.0, 1.0, 0.20601133, //0
    0.0, 1.0, -0.20601133, //1
    0.33333333, 0.872677996, 0.412022659, //2
    0.666666667, 0.745355992, 0.20601133, //3
    0.20601133, 0.666666667, 0.745355992, //4
    0.412022659, 0.33333333, 0.872677996, //5
    -0.20601133, 0.666666667, 0.745355992, //6
    -0.412022659, 0.33333333, 0.872677996, //7
    -0.33333333, 0.872677996, 0.412022659, //8
    -0.666666667, 0.745355992, 0.20601133, //9
    0.33333333, 0.872677996, -0.412022659, //10
    0.666666667, 0.745355992, -0.20601133, //11
    0.20601133, 0.666666667, -0.745355992, //12
    0.412022659, 0.33333333, -0.872677996, //13
    -0.20601133, 0.666666667, -0.745355992, //14
    -0.412022659, 0.33333333, -0.872677996, //15
    -0.33333333, 0.872677996, -0.412022659, //16
    -0.666666667, 0.745355992, -0.20601133, //17
    1.0, 0.20601133, 0.0, //18
    1.0, -0.20601133, 0.0, //19
    0.872677996, 0.412022659, 0.33333333, //20
    0.745355992, 0.20601133, 0.666666667, //21
    0.872677996, 0.412022659, -0.33333333, //22
    0.745355992, 0.20601133, -0.666666667, //23
    0.666666667, -0.745355992, -0.20601133, //24
    0.33333333, -0.872677996, -0.412022659, //25
    0.666666667, -0.745355992, 0.20601133, //26
    0.33333333, -0.872677996, 0.412022659, //27
    0.872677996, -0.412022659, 0.33333333, //28
    0.745355992, -0.20601133, 0.666666667, //29
    0.872677996, -0.412022659, -0.33333333, //30
    0.745355992, -0.20601133, -0.666666667, //31
    0.0, -1.0, -0.20601133, //32
    0.0, -0.1, 0.20601133, //33
    0.20601133, -0.666666667, -0.745355992, //34
    0.412022659, -0.33333333, -0.872677996, //35
    -0.20601133, -0.666666667, -0.745355992, //36
    -0.412022659, -0.33333333, -0.872677996, //37
    -0.33333333, -0.872677996, -0.412022659, //38
    -0.666666667, -0.745355992, -0.20601133, //39
    0.20601133, -0.666666667, 0.745355992, //40
    0.412022659, -0.33333333, 0.872677996, //41
    -0.20601133, -0.666666667, 0.745355992, //42
    -0.412022659, -0.33333333, 0.872677996, //43
    -0.33333333, -0.872677996, 0.412022659, //44
    -0.666666667, -0.745355992, 0.20601133, //45
    0.20601133, 0.0, 1.0, //46
    -0.20601133, 0.0, 1.0, //47
    -0.745355992, 0.20601133, 0.666666667, //48
    -0.872677996, 0.412022659, 0.33333333, //49
}

```

```

-0.745355992,-0.20601133,0.666666667, //50
-0.872677996,-0.412022659,0.333333333, //51
0.20601133,0.0,-1.0, //52
-0.20601133,0.0,-1.0, //53
-0.745355992,0.20601133,-0.666666667, //54
-0.872677996,0.412022659,-0.333333333, //55
-0.745355992,-0.20601133,-0.666666667, //56
-0.872677996,-0.412022659,-0.333333333, //57
-1.0,0.20601133,0.0, //58
-1.0,-0.20601133,0.0}; //59
/*
 * Die Fünfecke
 */
GLshort FUENF1[5] = {0,8,6,4,2};
GLshort FUENF2[5] = {1, 10, 12, 14, 16};
GLshort FUENF3[5] = {3, 20, 18, 22, 11};
GLshort FUENF4[5] = {19, 28, 26, 24, 30};
GLshort FUENF5[5] = {25, 32, 38, 36, 34};
GLshort FUENF6[5] = {33, 27, 40, 42, 44};
GLshort FUENF7[5] = {41, 29, 21, 5, 46};
GLshort FUENF8[5] = {43, 47, 7, 48, 50};
GLshort FUENF9[5] = {35, 52, 13, 23, 31};
GLshort FUENF10[5] = {37, 56, 54, 15, 53};
GLshort FUENF11[5] = {55, 58, 49, 9, 17};
GLshort FUENF12[5] = {59, 57, 39, 45, 51};
/*
 * Die Sechsecke
 */
GLshort SECHS1[6] = {0, 1, 16, 17, 9, 8};
GLshort SECHS2[6] = {2, 3, 11, 10, 1, 0};
GLshort SECHS3[6] = {22, 23, 13, 12, 10, 11};
GLshort SECHS4[6] = {4, 5, 21, 20, 3, 2};
GLshort SECHS5[6] = {21, 29, 28, 19, 18, 20};
GLshort SECHS6[6] = {19, 30, 31, 23, 22, 18};
GLshort SECHS7[6] = {30, 24, 25, 34, 35, 31};
GLshort SECHS8[6] = {24, 26, 27, 33, 32, 25};
GLshort SECHS9[6] = {40, 27, 26, 28, 29, 41};
GLshort SECHS10[6] = {38, 32, 33, 44, 45, 39};
GLshort SECHS11[6] = {41, 46, 47, 43, 42, 40};
GLshort SECHS12[6] = {47, 46, 5, 4, 6, 7};
GLshort SECHS13[6] = {43, 50, 51, 45, 44, 42};
GLshort SECHS14[6] = {8, 9, 49, 48, 7, 6};
GLshort SECHS15[6] = {52, 53, 15, 14, 12, 13};
GLshort SECHS16[6] = {35, 34, 36, 37, 53, 52};
GLshort SECHS17[6] = {37, 36, 38, 39, 57, 56};
GLshort SECHS18[6] = {54, 55, 17, 16, 14, 15};
GLshort SECHS19[6] = {55, 54, 56, 57, 59, 58};
GLshort SECHS20[6] = {58, 59, 51, 50, 48, 49};

```

Dann kann der abgeschnittene Hexaeder mit Hilfe der folgenden `display`-Funktion ausgegeben werden:

```

void display(void)
{
    glClear(GL_COLOR_BUFFER_BIT);
    examine();
    // Gitter ausgeben
    grid(15);
}

```

```

glColor3f(0.0, 1.0, 0.0);
glLineWidth(6.0);

// Die Geometrie
glVertexPointer(3, GL_FLOAT, 0, v);

// Die Topologie
// Die Fünfecke
glDrawElements(GL_POLYGON, 5, GL_UNSIGNED_SHORT, FUENF1);
glDrawElements(GL_POLYGON, 5, GL_UNSIGNED_SHORT, FUENF2);
glDrawElements(GL_POLYGON, 5, GL_UNSIGNED_SHORT, FUENF3);
glDrawElements(GL_POLYGON, 5, GL_UNSIGNED_SHORT, FUENF4);
glDrawElements(GL_POLYGON, 5, GL_UNSIGNED_SHORT, FUENF5);
glDrawElements(GL_POLYGON, 5, GL_UNSIGNED_SHORT, FUENF6);
glDrawElements(GL_POLYGON, 5, GL_UNSIGNED_SHORT, FUENF7);
glDrawElements(GL_POLYGON, 5, GL_UNSIGNED_SHORT, FUENF8);
glDrawElements(GL_POLYGON, 5, GL_UNSIGNED_SHORT, FUENF9);
glDrawElements(GL_POLYGON, 5, GL_UNSIGNED_SHORT, FUENF10);
glDrawElements(GL_POLYGON, 5, GL_UNSIGNED_SHORT, FUENF11);
glDrawElements(GL_POLYGON, 5, GL_UNSIGNED_SHORT, FUENF12);

// Die Sechsecke
glDrawElements(GL_POLYGON, 6, GL_UNSIGNED_SHORT, SECHS1);
glDrawElements(GL_POLYGON, 6, GL_UNSIGNED_SHORT, SECHS2);
glDrawElements(GL_POLYGON, 6, GL_UNSIGNED_SHORT, SECHS3);
glDrawElements(GL_POLYGON, 6, GL_UNSIGNED_SHORT, SECHS4);
glDrawElements(GL_POLYGON, 6, GL_UNSIGNED_SHORT, SECHS5);
glDrawElements(GL_POLYGON, 6, GL_UNSIGNED_SHORT, SECHS6);
glDrawElements(GL_POLYGON, 6, GL_UNSIGNED_SHORT, SECHS7);
glDrawElements(GL_POLYGON, 6, GL_UNSIGNED_SHORT, SECHS8);
glDrawElements(GL_POLYGON, 6, GL_UNSIGNED_SHORT, SECHS9);
glDrawElements(GL_POLYGON, 6, GL_UNSIGNED_SHORT, SECHS10);
glDrawElements(GL_POLYGON, 6, GL_UNSIGNED_SHORT, SECHS11);
glDrawElements(GL_POLYGON, 6, GL_UNSIGNED_SHORT, SECHS12);
glDrawElements(GL_POLYGON, 6, GL_UNSIGNED_SHORT, SECHS13);
glDrawElements(GL_POLYGON, 6, GL_UNSIGNED_SHORT, SECHS14);
glDrawElements(GL_POLYGON, 6, GL_UNSIGNED_SHORT, SECHS15);
glDrawElements(GL_POLYGON, 6, GL_UNSIGNED_SHORT, SECHS16);
glDrawElements(GL_POLYGON, 6, GL_UNSIGNED_SHORT, SECHS17);
glDrawElements(GL_POLYGON, 6, GL_UNSIGNED_SHORT, SECHS18);
glDrawElements(GL_POLYGON, 6, GL_UNSIGNED_SHORT, SECHS19);
glDrawElements(GL_POLYGON, 6, GL_UNSIGNED_SHORT, SECHS20);

glutSwapBuffers();
}

```

### 4.7.3 Fallstudie Polygonale Netze in Alias MAYA

1. Modellieren Sie mit Hilfe der Alias MAYA PLE eine Hand wie im Text angedeutet. Verwenden Sie dabei Ihre eigene Hand als Vorbild.

*Lösung:*

Bei den Downloads zu diesem Kapitel finden Sie eine entsprechende MAYA Datei!

2. Modellieren Sie eine Hand in Subdivision-Darstellung und vergleichen Sie die Ergebnisse mit dem Netz aus Aufgabe 1!

*Lösung:*

Auch für diese Aufgabe finden Sie eine entsprechende MAYA Datei bei den Downloads zu diesem Kapitel.

Es soll an dieser Stelle nochmals darauf hingewiesen werden, dass Sie beim `smooth`-Operator (nicht nur in MAYA) extrem vorsichtig sein sollten – wenigstens falls Sie auf die Anzahl der Polygone im endgültigen Modell achten müssen!

3. Modellieren Sie mit Hilfe der Alias MAYA PLE verschiedene Schachfiguren als Subdivisions-Modell!

*Lösung:*

Auch für diese Aufgabe finden Sie eine entsprechende MAYA Datei bei den Downloads zu diesem Kapitel!