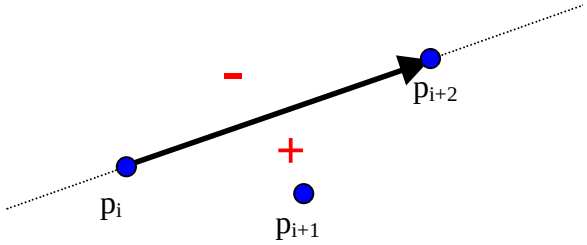
Aufgabe 12

Um eine korrekte Überprüfung des Start- bzw. Endpunktes zu ermöglichen, erweitere ich die Liste P, indem ich p_0 und p_1 als Kopie an das Ende anfüge:

$$P = \left\{ p_0, p_1, p_2, ..., p_n, p_{n+1}, p_{n+2} \;\middle|\; p_i \in R^2, 0 \le i \le n+2 \right\}$$
$$p_{n+1} = p_0$$
$$p_{n+2} = p_1$$

Danach durchläuft eine Schleife alle n Punkte und vergleicht sie mit den beiden vom Index her folgenden Punkten, d.h. p_i, p_{i+1} und p_{i+2} werden betrachtet. Als Kernidee entsteht aus p_i und p_{i+2} eine Liniengleichung, die man in ihrer impliziten Form berechnet. Setzt man anschließend p_{i+1} in diese Gleichung ein, so kann man anhand des Vorzeichens bestimmen, auf welcher Seite dieser Punkt liegt. Es ist das Erreichen von Null erlaubt, weil dann p_{i+1} genau auf der Linie liegt, jedoch spielt die Orientierung des Richtungsvektors eine wichtige Rolle, d.h. es besteht ein wesentlicher Unterschied zwischen $p_{n+2} - p_n$ und $p_n - p_{n+2}$.



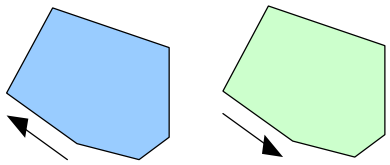
Die implizite Form der Geradengleichung ergibt sich gemäß Vorlesung durch:

$$F(x,y) = ax + by + c = 0$$
$$a = \delta y$$
$$b = -\delta x$$
$$c = y_i \cdot \delta x - x_i \cdot \delta y$$
$$\delta x = x_{i+2} - x_i$$
$$\delta y = y_{i+2} - y_i$$

Vollständig entwickelt unter der Annahme $p_k = (x_k, y_k)$:

$$\begin{aligned} F(x_{i+1}, y_{i+1}) &= ax_{i+1} + by_{i+1} + c \\ &= (y_{i+2} - y_i) \cdot x_{i+1} - (x_{i+2} - x_i) \cdot y_{i+1} + (x_{i+2} - x_i) \cdot y_i - (y_{i+2} - y_i) \cdot x_i \\ &= (y_{i+2} - y_i) \cdot (x_{i+1} - x_i) - (x_{i+2} - x_i) \cdot (y_{i+1} - y_i) \end{aligned}$$

Nur in wenigen Fällen kann man Annahmen machen, ob das Polygon im oder entgegen dem Uhrzeigersinn orientiert ist. Aus genau diesem Grunde ist es nicht bekannt, ob das Vorzeichen jedesmal positiv oder negativ sein soll.



Um dieses Problem zu umschiffen, führe ich eine Hilfsvariable v ein, die anfangs auf Null gesetzt wird. Sobald ein Punkt p_k als Ergebnis des Einsetzens in die implizite Gleichung ein Ergebnis ungleich Null liefert (i.d.R. geschieht das schon bei p_0, p_1 und p_2), setze ich

$$v = F(x_k, y_k)$$

Für jeden nachfolgenden Punkt p_j führe ich die gleiche Berechnung durch, ändere aber v nicht mehr, sondern prüfe, ob

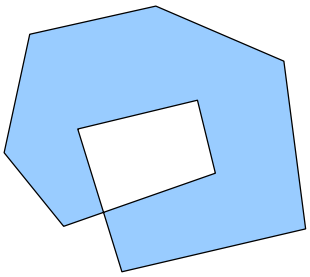
$$v \cdot F(x_j, y_j) < 0$$

zutrifft, was heißen würde, dass das Polygon nicht konvex ist, da die Vorzeichen verschieden sind, also p_j auf der falschen Seite der Gerade liegt. Ich möchte nochmals auf die im Text erwähnte Feststellung hinweisen, dass $F(x_j, y_j) = 0$ bedeutet, dass Teil der Gerade ist und daher auch als konvex zählt.

In Pseudocode ist der Algorithmus recht kurz:

```
// Initialisierung
// kopiere die ersten beiden Punkte
p[n+1] = p[0];
p[n+2] = p[1];
// Orientierung des Polygons ist noch unbekannt
v = 0;

// durchlaufe alle Punkte
for (i=0; i<=n; i++)
{
      // bestimme Lage von p[i+1] bezüglich p[i] und p[i+2]
      F = (p[i+2].y-p[i].y)*(p[i+1].x-p[i].x) –
          (p[i+2].x-p[i].x)*(p[i+1].y-p[i].y);

      // Orientierung festlegen
      if (v == 0)
            v = F;

      if (v*F < 0)
            return „nicht konvex“;
}

return „konvex“;
```

Dieses Verfahren ist leider nicht in der Lage, alle nicht konvexen Polygon korrekt zu erkennen, da es jeweils nur eine Teilmenge der Eckpunkte betrachtet. Ein Beispiel liefert diese Grafik:



Das weiße Loch in der Mitte weist auf Konkavität hin, nur kann mein Algorithmus das nicht erkennen. Ein mögliches Verfahren, um auch diese Fälle auszuschließen, besteht in der Bestimmung der konvexen Hülle aller Punkte. Sollte mindestens ein Punkt nicht in ihr enthalten sein, so ist das Polygon konkav. In der Literatur findet man etliche Verfahren zur Ermittlung der konvexen Hülle, so dass ich darauf nicht allzu weit eingehen möchte. Als einfache, wenn auch langsame, Erweiterung meines Verfahren könnte man alle Punkte, statt nur einen, auf die Lage bezüglich jeder Geraden testen.

Aufgabe 13

Der Cyrus-Beck-Algorithmus (insbesondere in der Erweiterung von Liang-Barsky) lässt sich relativ leicht auf den dreidimensionalen Raum übertragen. Als wesentliche Erweiterungen ist das Clippen gegen 6 Ebenen statt 4 Linien notwendig, was natürlich die Anzahl der benötigten Kantennormalen ebenfalls von 4 auf 6 erhöht. Zu beachten ist, dass alle Berechnungen unter Einfluss von 3 Koordinaten erfolgen müssen.

Eine Strecke im dreidimensionalen Raum hat die Form:

$$P(t) = P_0 + (P_1 - P_0) \cdot t, \quad t \in R, \quad P_0, P_1 \in R^3$$

Der Schnittpunkt mit einer Clipping-Ebene berechnet sich als:

$$t = \frac{N_i \cdot (P_0 - P_{E_i})}{-N_i \cdot (P_1 - P_0)}$$

Dabei ist N_i der Normalenvektor der entsprechenden Ebene und P_{Ei} ein Punkt derselben, günstig ist die Verwendung eines Eckpunktes. Voraussetzung für die Verwendung der Formel ist, dass die Normale eine Länge größer Null besitzt, dass die Strecke nicht zu einem Punkt degeneriert ist ($P_0 \neq P_1$) und die Strecke nicht parallel zur Clippingebene verläuft.

Alle Schnittpunkte müssen danach klassifiziert werden, ob sie „potentially entering“ (PE) oder „potentially leaving“ (PL) sind, was am Vorzeichen von $N_i \cdot (P_1 - P_0)$ zu erkennen ist: negative Werte stehen für PE, positive für PL. Anschließend betrachtet man die Strecke als Gerade, d.h. mit unendlicher Länge. Genau der Teil ist auf dem Bildschirm sichtbar, für den $t_{PE} < t_{PL}$ ist. Sollte $t_{PE} < 0$ oder $1 < t_{PL}$ sein, so muss man sich von der Geraden wieder auf die Strecke zurückbesinnen und $t_{PE} = 0$ bzw. $t_{PL} = 1$ setzen.

Die Umsetzung in Pseudocode hat dann die Gestalt:

```
D = P1-P0;
if (length(D) == 0)
     ClipPoint(P1);    // zum Punkt degeneriert
else
{
     tE = 0;
     tL = 1;

     for (each candidate intersection with a clip edge)
           // nur nichtkantenparallele Strecken
           if (Ni * D != 0)
           {
                 t = Ni*(P0-PEi) / -(Ni*(P1-P0);

                 if (Ni*D < 0)
                       tE = max(tE, t);

                 if (Ni*D > 0)
                       tL = min(tL, t);

           }
```

```
        if (tE > tL)
              return „outside clip region“;
        else
              drawline(P0+tE*(P1-P0), P0+tL*(P1-P0));
}
```

Bis dato tauchten noch keine wesentlichen Unterschiede zwischen 2D und 3D auf. Nimmt man jedoch einen Quader als Clipping Object, dessen Seiten senkrecht auf den jeweiligen Achsen stehen, so ergeben sich starke Vereinfacherungen:

Clip plane	**Normal N_i**	**P_{Ei}**	$t = \frac{N_i \cdot (P_0 - P_{E_i})}{-N_i \cdot (P_1 - P_0)}$
left: $x = x_{\min}$	$\begin{pmatrix} -1 \\ 0 \\ 0 \end{pmatrix}$	$\begin{pmatrix} x_{\min} \\ y \\ z \end{pmatrix}$	$\frac{(x_0 - x_{\min})}{(x_0 - x_1)}$
right: $x = x_{\max}$	$\begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$	$\begin{pmatrix} x_{\max} \\ y \\ z \end{pmatrix}$	$\frac{(x_0 - x_{\max})}{(x_0 - x_1)}$
bottom: $y = y_{\min}$	$\begin{pmatrix} 0 \\ -1 \\ 0 \end{pmatrix}$	$\begin{pmatrix} x \\ y_{\min} \\ z \end{pmatrix}$	$\frac{(y_0 - y_{\min})}{(y_0 - y_1)}$
top: $y = y_{\max}$	$\begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$	$\begin{pmatrix} x \\ y_{\max} \\ z \end{pmatrix}$	$\frac{(y_0 - y_{\max})}{(y_0 - y_1)}$
far: $z = z_{\min}$	$\begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix}$	$\begin{pmatrix} x \\ y \\ z_{\min} \end{pmatrix}$	$\frac{(z_0 - z_{\min})}{(z_0 - z_1)}$
close: $z = z_{\max}$	$\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$	$\begin{pmatrix} x \\ y \\ z_{\max} \end{pmatrix}$	$\frac{(z_0 - z_{\max})}{(z_0 - z_1)}$

Da es vollkommen unerheblich ist, wo genau P_{Ei} auf den clipping planes liegt, kann man zwei der drei Koordinatenelemente aus dem Start- bzw. Endpunkt der Strecke übernehmen. Zusätzlich kann man die Parallelität zu den Ebenen auf einen Test gegen Null jedes Koordinatenelementes reduzieren. Schlussendlich wirkt sich das Skalarprodukt noch sehr positiv auf Vereinfacherungen aus, da zwei Koordinaten Null sind.

Wendet man diese Ergebnisse auf den Einheitswürfel und die Strecke durch die Punkte

$$p_0 = \begin{pmatrix} -2 \\ -1 \\ 0{,}5 \end{pmatrix},\ p_1 = \begin{pmatrix} 1{,}5 \\ 1{,}5 \\ -0.5 \end{pmatrix},\ D = p_1 - p_0 = \begin{pmatrix} 3{,}5 \\ 2{,}5 \\ -1 \end{pmatrix}$$

an, so erhält man:

$$t_{left} = \frac{-2+1}{-2-1{,}5} = \frac{2}{7},\ PE$$

$$t_{right} = \frac{-2-1}{-2-1{,}5} = \frac{6}{7},\ PL$$

$$t_{bottom} = \frac{-1+1}{-1-1{,}5} = 0,\ PE$$

$$t_{top} = \frac{-1-1}{-1-1{,}5} = \frac{4}{5},\ PL$$

$$t_{far} = \frac{0{,}5+1}{0{,}5+0{,}5} = \frac{3}{2},\ PL$$

$$t_{close} = \frac{0{,}5-1}{0{,}5+0{,}5} = -\frac{1}{2},\ PE$$

$$t_E = \frac{2}{7}$$

$$t_L = \frac{4}{5}$$

Die geclippte Strecke lautet nun (gerundet auf drei Stellen nach dem Komma):

$$p_E = \begin{pmatrix} -2 \\ -1 \\ 0{,}5 \end{pmatrix} + t_E \begin{pmatrix} 3{,}5 \\ 2{,}5 \\ -1 \end{pmatrix} = \begin{pmatrix} -1 \\ -0{,}286 \\ 0{,}214 \end{pmatrix}$$

$$p_L = \begin{pmatrix} -2 \\ -1 \\ 0{,}5 \end{pmatrix} + t_L \begin{pmatrix} 3{,}5 \\ 2{,}5 \\ -1 \end{pmatrix} = \begin{pmatrix} 0{,}8 \\ 1 \\ -0{,}3 \end{pmatrix}$$

$$P_{clipped}(t) = \begin{pmatrix} -1 \\ -0{,}286 \\ 0{,}214 \end{pmatrix} + t \cdot \begin{pmatrix} 1{,}8 \\ 1{,}286 \\ -0{,}514 \end{pmatrix},\ t \in [0;1]$$