BERUFSAKADEMIE MANNHEIM
STAATLICHE STUDIENAKADEMIE

Fachrichtung Informationstechnik

Referat
Seminar Informationstechnik

Computergrafik II: Clipping

grafische Betriebssysteme sind allgegenwärtig
- ohne Clipping funktioniert dabei nichts

Mannheim, den 22.April 2004

Andreas Richter
102731
TIT02AGR

Nico Schröder
121576
TIT02AGR

Inhaltsverzeichnis

Stichwortverzeichniss

Begriff	**Erklärung**
Clipping	(vom engl. to clip: abschneiden, kappen) in der Computergrafik spricht man vom Clipping, wenn Figuren auf einen bestimmten Bereich begrenzt werden sollen
Clipping Plane	Ebene, die als Trennlinie für eine Sichtebene dient
Gerätetransformation	Abbildung von dem Normalraumfenster in den Geräteviewport
GKS	(engl. Graphical Kernel System) grafisches Kernsystem
NDC	(engl. Normalized Device Coordinates) normierter Koordinatenraum
Normierungstrans-formation	Abbildung des Weltwindows auf ein im normierten Koordinatenraum leigendes Fenster
Outcode	4-bit-Code, welcher die Lage eines Punktes im Verhältnis zum Clipping-Rechteck im Cohen-Southerland-Algorithmus beschreibt
Polymarker	Polymarker werden auch Pixel bzw. Bildpunkte benannt
Sichtvolumen	Bereich, der bei dreidimensionalen Objekten angezeigt wird
Viewport	Zielbereich, in dem Daten abgebildet werden sollen; dem Gerät näher
Window	abzubildener Bereich; dem Benutzer näher
Window-Viewport-Transformation	Window-Viewport-Transformation, Vorgang, der den Window-Inhalt in den Viewport bringt
Wraparound	Fehler, bei dem Punkte auf der einen Seite fälschlicherweise auf der gegenüberliegenden Seite auftauchen

Einleitung

Heutzutage kommt man um eine grafische Benhutzerführung kaum noch herum. Umso wichtiger ist es, daß man sich als Programmierer keine Gedanken machen muss, wie die benutzten Elemente für jedes Ausgabegerät umgewandelt werden.

Bei der Entwicklung von Programmen ist es für den Programmierer angenehmer, die grafischen Daten in Weltkoordinaten anzugeben. Weltkoordinaten heißt, dass es sich um die Zahlen handelt, die auch in den restlichen Berechnungen benutzt werden. Auch die Struktur und Portabilität der Programme wird damit verbessert. Parameter wie die Auflösung des Bildschirmes oder die Größe der Zeichenfläche können dadurch weniger leicht in die Programmierung einfließen.

Um nun diese Trennung der Programmdaten (Weltkoordinaten) einerseits und die Charakteristika der Ausgabe (Gerätekoordinaten) andererseits zu unterstützen, erlauben Grafikpakete die Definition von Teilbereichen beider Koordinatensystemen, die aufeinander abgebildet werden. Diese Teilbereiche nennt man *Window* für den abzubildenden Bereich der Weltkoordinaten und *Viewport* als Zielbereich, auf dem das Window abgebildet wird.

Bei solchen Abbildungen kann es vorkommen, dass das abzubildende Objekt aus dem Window, bzw. Viewport hinausragt. Damit diese nicht außerhalb des Viewport angezeigt werden, muss das Objekt an den Begrenzungslinien abgeschnitten werden. Diesen Vorgang nennt man *Clipping*. Wird das darzustellende Objekt nicht beschnitten, führt dies in der Regel zu Anomalien in der Darstellung, wie zum Beispiel zum sogenannten *Wraparound*.

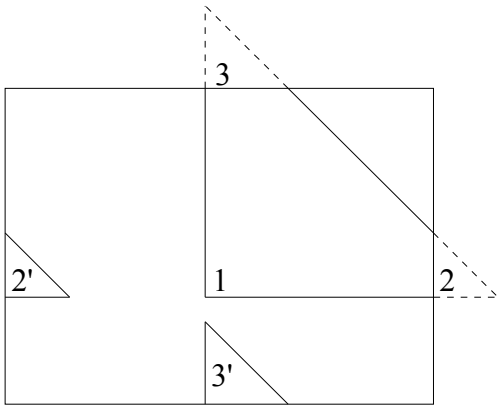

Abbildung 1: Wraparound-Fehler

Die auftretenden Fehler hängen dabei stark vom Prinzip der Berechnung der einzelnen Bildpunkte im Gerät ab. Fehler können des Weiteren auftreten, wenn sich mehrere Viewports den Bildschirm teilen. Ohne Clipping würden die Inhalte der einzelnen

Viewports sich gegenseitig beeinflussen, also falsche Bilder erzeugen, somit wird das Clipping in den meisten Anwendungen unumgänglich.

Window-Viewport-Transformation

Die Definition von Window und Viewport bietet dem Anwender den Vorteil, dass die grafischen Daten problembezogen umgewandelt werden können. Die Definition des Bereiches am Ausgabegerät, wo der Inhalt des Window dargestellt werden soll, erfolgt somit völlig entkoppelt.

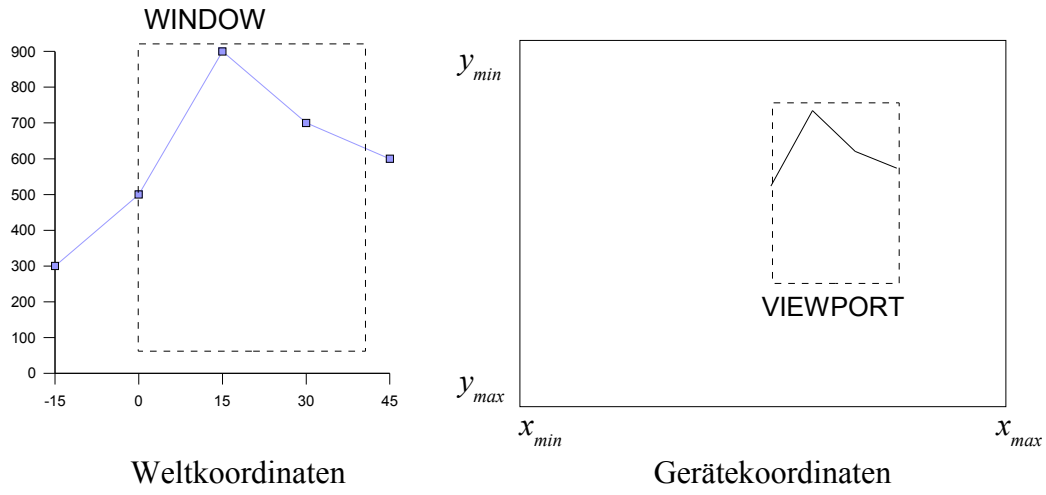

Abbildung 2: Abbildung des Window-Inhalts auf den Viewport-Inhalt

Damit nun mehrere Windows auf einem Ausgabegerät dargestellt werden können, bietet das *GKS* (grafische Kernsystem) eine zweifache *Window-Viewport-Transformation* an: Im ersten Schritt wird durch eine *Normierungstransformation* eine Abbildung des Welt-Windows auf ein im normierten Koordinatenraum liegendes Fenster durchgeführt. Dieser normierte Koordinatenraum gilt als geräteunabhängig, ist sozusagen eine abstrakte Sichtfläche. Nach dem Namen des Koordinatenraumes (*Normalized Device Coordinates (NDC)*) wird dieses Fenster auch *NDC-Viewport* genannt. Für die Abbildung unterschiedlicher Daten können mehrere Normierungstransformationen definiert werden, die dann verschiedenste Welt-Windows in NDC-Viewports umwandeln.

Im zweiten Schritt wird die Abbildung durch die *Gerätetransformation* auf das Ausgabegerät gebracht. Dabei wird das NDC-Window auf ein *Device-Viewport* transformiert. Da GKS die Ansteuerung mehrerer Ausgabegeräte unterstützen, kann für jeden Arbeitsplatz eine eigene Gerätetransformation definiert werden. So kann die Ausgabe, wie in Abbildung 3 abgebildet, einer Grafik auf dem Drucker komplett durchgeführt werden und auf dem Monitor ist nur ein Ausschnitt zu sehen.

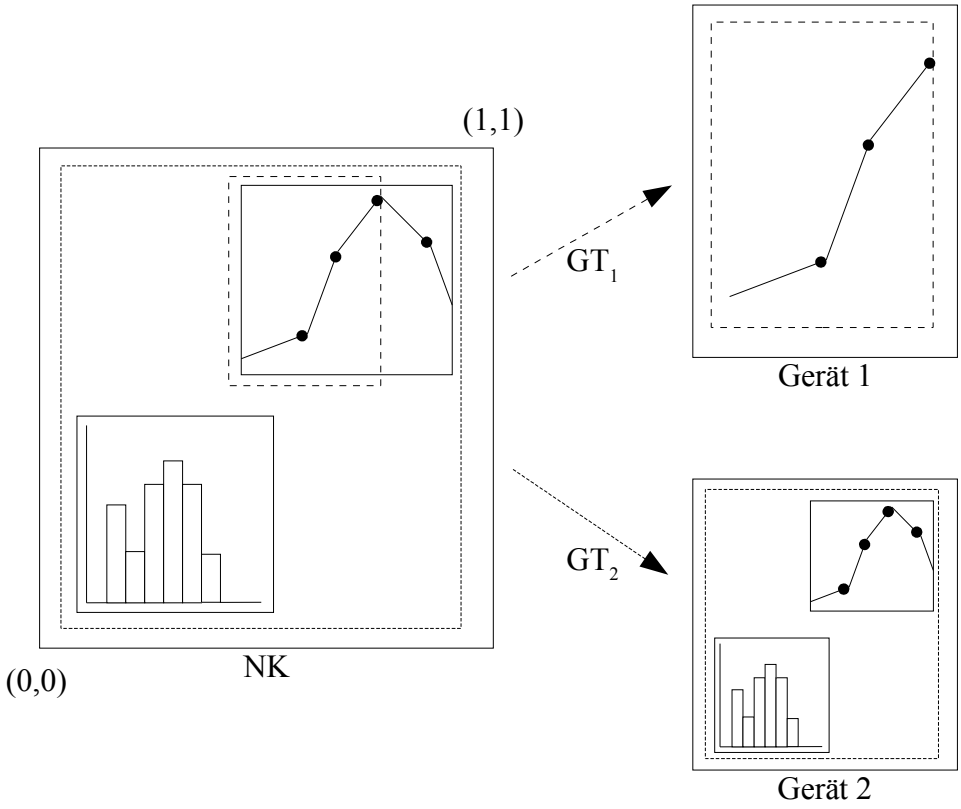

Abbildung 3: Gerätetransformation GT_1 und GT_2

Allgemein wird bei der Window-Viewport-Transformation davon ausgegangen, dass Window als auch Viewport achsenparallele Rechtecke sind. Die Normierungstransformation hat allerdings keine Einschränkungen bezüglich der Parameter für Höhe und Breite der Rechtecke, so kann es deshalb bei unterschiedlichen Seitenverhältnissen der beiden Rechtecke zu Verzerrungen der transformierten Objekte kommen.

Bei der Gerätetransformation hingegen fordert das GKS, daß die Proportionen der Objekte erhalten bleiben. Deswegen werden die Eckpunkte des Viewports modifiziert, so dass ein Teil des Viewports nicht ausgenützt wird, wenn die Proportionen von NDC-Window und Geräte-Viewpoint nicht übereinstimmen.

Mittlerweile wird nicht nur in GKS, sondern bei fast jedem Betriebssystem die Fenstertechnik intensiv genutzt. Verschiedenste Programme bieten dem Benutzer das Öffnen, bzw. Aktivieren von Fenstern an, in denen verschiedenen, zum Teil voneinander unabhängige Berechnungsergebnisse dargestellt werden.

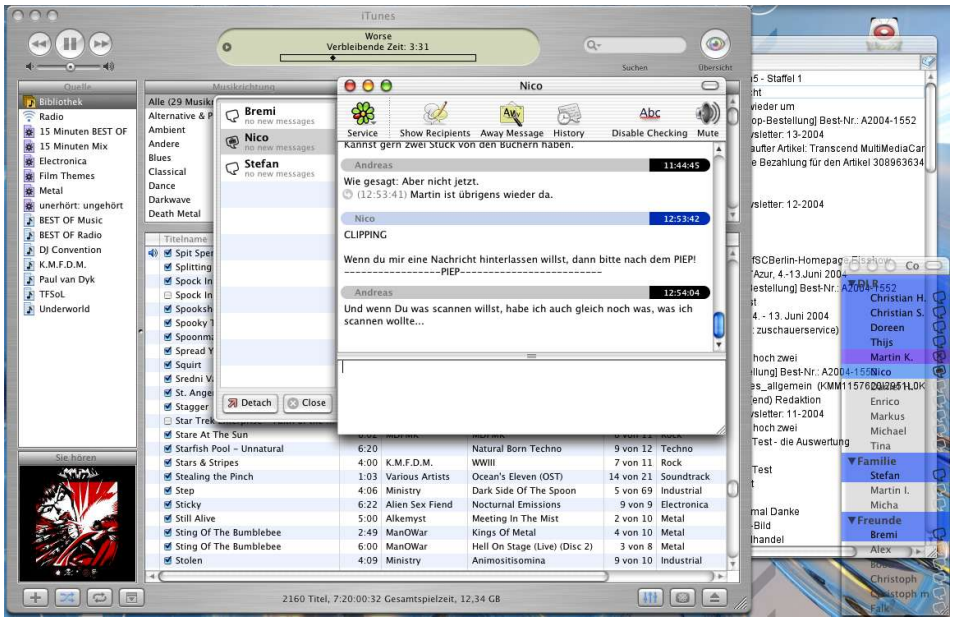

Abbildung 4: Screenshot einer typischen Arbeitssituation

Aus Platzgründen überlappen sich die meisten Fenster. Um diese Überlappung zu kontrollieren, erhält jedes Fenster eine eigene Priorität, die die Sichtbarkeit gegenüber den restlichen Fenstern regelt. Wendet der Benutzer seine Aufmerksamkeit einem Fenster zu, so wird durch Erhöhung der Priorität das Fenster ganz nach vorne geholt und das im Fenster „ablaufende“ Programm akzeptiert nachfolgend Benutzereingaben.

Durch die grafischen Oberflächen hat sich ebenfalls der Begriff Window eingebürgert, der mit dem Begriff aus dem Grafikstandard kollidiert. In der Window–Viewport–Transformation bedeuted Window einen Ausschnitt aus einem höhergelegenen Koordinatensystem (dem Benutzer näher), der auf einem niedrigergelegenen Koordinatensystems (dem Gerät näher) abgebildet wird. Ein Window in der Fenstertechnik entspricht somit eher einem Viewport. Allerdings ist zu beachten, daß in der Fenstertechnik wesentlich Aspekte der Benutzerführung integriert sind.

Clipping

Wie im vorangegangenen Abschnitt bereits beschrieben wurde, wird bei der Window–Viewport–Transformation davon ausgegangen, dass Objekte, die ganz oder teilweise außerhalb des abzubildenden Fensters liegen, durch das System entfernt oder nicht angezeigt werden. Diese Algorithmen, die das Entfernen und Abschneiden durchführen werden *Clipping-Algorithmen* genannt.

Im Zusammenhang mit der Window–Viewport–Transformation muss man sich entscheiden, ob das Clipping an der Window- oder an der Viewportgrenze erfolgen soll. Im

Allgemeinen wird die Window-Grenze verwendet, da dadurch die Transformation für im Viewport nicht enthaltene Objekte eingespart werden kann. Bei Grafiksystemen, die hardwareseitig das Clipping an den Viewport-Grenzen unterstützen, ist die Entscheidung nicht ganz so eindeutig.

Polymarkern

Die einfachste Art und Weise des Clippings stellt das Clippen von *Polymarkern* (Punkten bzw. Pixeln) dar. Dabei wird anhand zweier Ungleichungen ermittelt, ob ein Pixel im Clipping-Rechteck liegt und somit sichtbar ist:

$$X_{min} \leq X \leq X_{max} \quad \text{und} \quad Y_{min} \leq Y \leq y_{max}$$

Es werden die zu betrachtenden Bildpunktkoordinaten (X, Y) mit den Fenterkoordinaten ($X_{max}, X_{min}, Y_{max}, Y_{min}$) verglichen. Alle Bildpunkte, welche die Ungleichung nicht erfüllen, liegen außerhalb des Fensters und sind somit für den Benutzer nicht sichtbar. Diese Bildpunkte sollen nicht dargestellt werden.

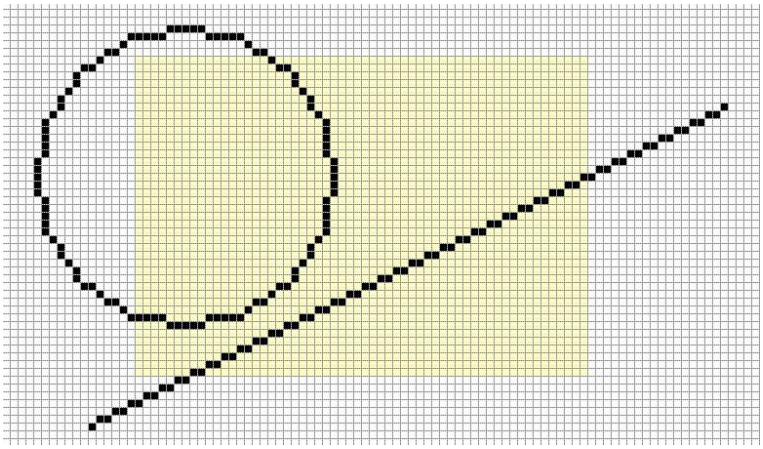

Abbildung 5: Clipping von Polymakern

Letztendlich lassen sich mit diesem Verfahren alle Clipping-Probleme lösen, da man jedes Objekt nur in Pixel umwandeln muss und dann Punkt für Punkt clippt. Dieser Vorgang bedarf einer sehr großen Rechenleistung uns kann aus zeitlichen Gründen nicht in Frage kommen. Es müssen bessere Clipping-Vorschriften für größere Bildteile gefunden werden.

Vektoren (Geraden)

Die nächste Abstraktionshöhe ist das Clippen von Vektoren, wie in Abbildung 6 dargestellt.

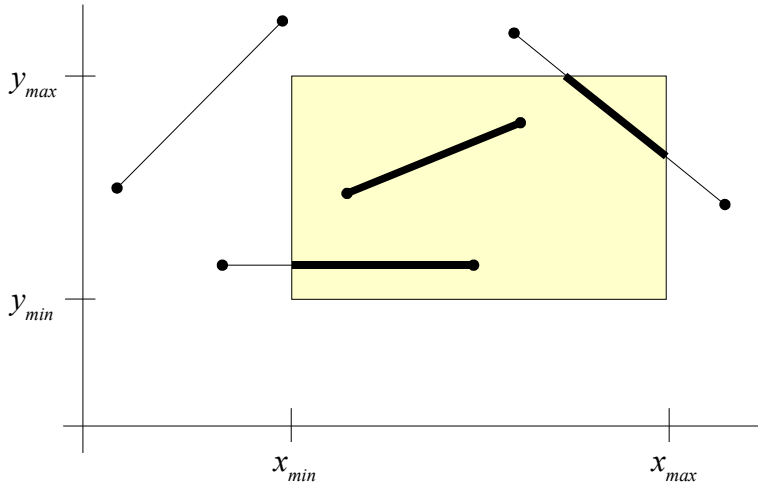

Abbildung 6: mögliche Lage einer Linie zum Clipping-Bereich; dickere Linien(segmente) zeigen das Bild nach dem Clipping

Cohen-Southerland

Ein sehr weit verbreiteter Algorithmus für das Clipping von Vektoren ist der Cohen-Southerland-Algorithmus. Dabei wird jedem Anfangs- und Endpunkt eines Vektors ein 4-bit-Code zugeordnet. Dieser Code wird entsprechend der Lage des Punktes in einer der 9 Regionen, welche durch die Fenstergrenzen gebildet werden, ermittelt. Es gelten folgende Spezifikationen:

y_{max}	1001	1000	1010
	0001	0000	0010
y_{min}	0101	0100	0110
	x_{min}		x_{max}

Codierung:

bit 1: links vom Fenster
bit 2: rechts vom Fenster
bit 3: unter dem Fenster
bit 4: über dem Fenster

Abbildung 7: Bereichscodes des Cohen-Southerland Algorithmus

- er liegt innerhalb, wenn der Code beider Punkte Null ist

- er liegt außerhalb, wenn der Durchschnitt (logisches UND) der Codes beider Punkte verschieden von Null ist

Auf Grund dieser Eigenschaft nennt man diesen 4-bit-Code auch *Outcode*.

Für den Fall, dass keine der beiden Aussagen zutrifft, schneidet der Vektor eine der Fensterkanten. Somit liegt ein Teil im sichtbaren und ein Teil im nicht sichtbaren Bereich des Clipping-Fensters. Durch den Algorithmus wird in diesem Fall der Schnittpunkt des Vektors mit einer geeigneten Fensterkante bestimmt. Nachdem dieser Schnittpunkt berechnet wurde, wird wieder die Schritt 1 des Algorithmus durchgeführt und erneut für den neuen Punkt ein Outcode vergeben.

Ob tatsächlich eine Schnittpunktberechnung durchgeführt werden muss, wird durch den Vergleich der Outcodes der Endpunkte festgestellt. Bei einem ungleichen Wert der Codes in einer Bitstelle wird ein Schnittpunkt mit der entsprechenden Fensterbgrenzung berechnet. In Abbildung 8 wird dies verdeutlicht.

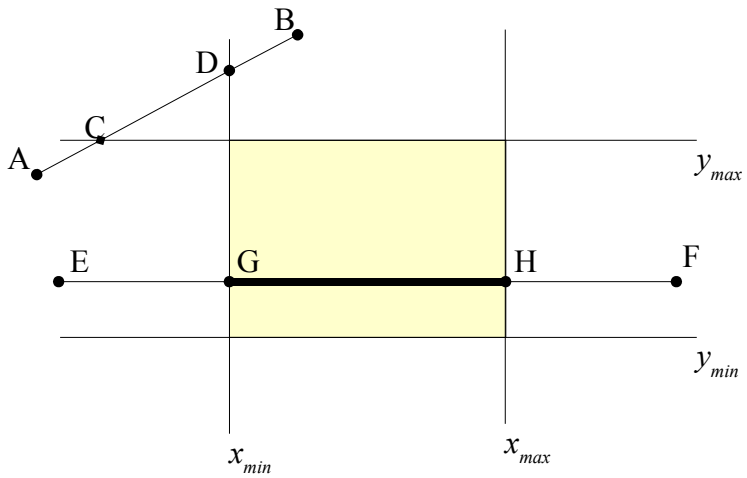

Abbildung 8: Schnittpunktberechung von Vektoren mit dem Cohen-Southerland Algortihmus

Vektor AB : Augrund der Lage der beiden Endpunkte in verschiedenen Bereichen und unterschiedlichen Outcodes lässt dieser Verktor keine Entscheidung zu. Es wird eine Schnittpunktberechnung mit der linken Fensterkante durchgeführt, diese liefert den Punkt D . Da der Vektor AD links des Fensters liegt, kann dieser Teil komplett elminiert werden. Es verbleibt der Vektor DB , dessen Endpunkte oberhalb des Fensters liegen. Somit kann dieser Vektor auch eliminiert werden.

Vektor EF : Bei diesem Vektor liefert die Schnittpunktberechnung den Punkt G . Der Teil EG kann auf Grund seiner Lage entfernt werden. Ein Endpunkttest des Teils GF ergibt keine Entscheidung. Es muss eine weitere Schnittpunktberechnung durchgeführt werden. Diese liefert den Punkt H . HF kann eliminiert werden, GH wird als sichtbar erkannt und dargestellt.

Zur Schnittpunktberechnung wird von Cohen und Southerland eine Geradengleichung der Form ($P_1(X_1, Y_1), P_2(X_{2,} Y_2)$) verwendet:

$$\frac{Y-Y_1}{X-X_1}=\frac{Y_2-Y_1}{X_2-X_1}=m$$

wobei der Fall $X_2=X_1$ (vertikale Gerade) bzw. $Y_2=Y_1$ (horizontale Gerade) besonders behandelt wird.

Die Schnittpunktkoordinaten X_{smin}, X_{smax} bzw. Y_{smin}, Y_{smax} ergeben sich dann für die Y-Begrenzungen als

$$X_{smin}=X_1+(Y_{min}-Y_1)\cdot\frac{1}{m} \quad , \quad X_{smax}=X_1+(Y_{max}-Y_1)\cdot\frac{1}{m}$$

bzw. für die X-Begrenzungen

$$Y_{smin}=Y_1+(X_{min}-X_1)\cdot m \quad , \quad Y_{smax}=Y_1+(X_{max}-X_1)\cdot m$$

die Ergebnisse können für die Berechnung der Outcodes verwendet werden.

Liang-Barsky

Der Algorithmus von Liang-Barsky geht von der Parameterdarstellung einer Geraden aus. Dieses Verfahren stellt die Linie $\overline{P_1P_2}$ in der Parameterform

$$x=x_1+\Delta x\cdot t \quad , \quad y=y_1+\Delta y\cdot t$$

mit

$$\Delta x=x_2-x_1 \quad \text{und} \quad \Delta y=y_2-y_1$$

dar. Dabei gibt der Parameter t für das Intervall $[0,1]$ genau das Geradenstück $\overline{P_1P_2}$ und für $[-\infty,+\infty]$ eine Gerade durch die Punkte P_1 und P_2 .

Werden die Fenstergrenzen mit x_{min}, x_{max} und y_{min}, y_{max} bezeichnet, so gilt für das Geradenstück im inneren des Fensters folgende Ungleichungen

$$-\Delta x \cdot t \le x_1 - x_{min} \qquad -\Delta y \cdot t \le y_1 - y_{min}$$
$$\Delta x \cdot t \le x_{max} - x_1 \qquad \Delta y \cdot t \le y_{max} - y_1$$

Diese Ungleichungen werden in eine einheitliche Form $p_i \cdot t \le q_i, i = 1...4$ gebracht. Dabei ergeben sich für die Parameter p_i und q_i die Werte

$$p_1 = -\Delta x \qquad q_1 = x_1 - x_{min}$$
$$p_2 = \Delta x \qquad q_2 = x_{max} - x_1$$
$$p_3 = -\Delta y \qquad q_3 = y_1 - y_{min}$$
$$p_4 = \Delta y \qquad q_4 = y_{max} - y_1$$

Die Ebene wird durch die Verlängerung der vier Begrenzungskanten x_{min}, x_{max} und y_{min}, y_{max} in jeweils zwei Halbebenen geteilt. Wird nun aus den Halbebenen, die als sichtbar definiert wurden der Durchschnitt gebildet, so ergibt sich daraus das Clipping-Rechteck. Die obigen vier Ungleichungen beschreiben je eine dieser sichtbaren Halbebenen. Die Begrenzungskanten werden durch die Zahlen 1 bis 4 beschrieben. Dabei gilt die Reihenfolge links, rechts, unten und oben.

Aus der Sicht der Geometrie wird die Gerade durch die Punkte P_1 und P_2 von dem Algorithmus behandelt. Für jede der vier Begrenzungskanten kann mit Hilfe der vier Ungleichungen der Wertebereich für den Parameter t ermittelt werden. Der Wertebereich gibt immer den sichtbaren Teil der Geraden im Bezug zur verwendeten Begrenzungskante an. Das bedeutet, wird der ermittelte Wert für den Parameter t in die Parameterform eingesetzt, so erhält man einen Strahl, dessen Ursprungspunkt auf der Geraden der zu behandelten Begrenzungskante liegt. Der Strahl verläuft darüber hinaus nur im als sichtbar definierten Bereich der aktuellen Halbebenen.

Für $p_i \neq 0$ hat der Schnittpunkt der Geraden mit der Begrenzungskante den Wert $t = q_i / p_i,\ i = 1,...4$ und das Vorzeichen von q_i gibt an, auf welcher Seite der Kante der Punkt P_1 liegt. Ist $q_i \ge 0$ so liegt P_1 im sichtbaren Bereich, andernfalls liegt P_1 außerhalb des Clipping-Rechtecks. Für denn Fall, dass $p_i = 0$ ist, liegt die Gerade parallel zu der i-ten Begrenzungskante. Ist $p_i < 0$, so kommt die gerichtete Gerade durch $\overline{P_1 P_2}$ vom unsichtbaren in den sichtbaren Bereich. Liegt $p_i > 0$ vor, so verlässt die Gerade den sichtbaren Bereich.

Der Algorithmus berechnet für jede Linie die Parameterwerte t_1 und t_2 , die den im Inneren des Clipping-Rechtecks liegenden Abschnitt der Geraden durch $\overline{P_1 P_2}$ festlegen, durch die Gleichungen

$$t_1 = max(\{q_i / p_i | p_i < 0,\ i = 1,...4\} \cup \{0\})$$
$$t_2 = min(\{q_i / p_i | p_i > 0,\ i = 1,...4\} \cup \{1\})$$

Gilt $t_1 > t_2$, dann liegt die Linie komplett außerhalb des Clipping-Rechtecks und muss nicht weiter behandelt werden. Sonst werden die beiden Eckpunkte des sichtbaren Geradenstückes aus den Parametern t_1 und t_2 nach folgender Gleichung berechnet:

$$x = x_1 + \Delta x \cdot t \quad , \quad y = y_1 + \Delta y \cdot t$$

Für den ersten Punkt (S_1) wird t_1 , für den zweiten Punkt (S_2) t_2 eingesetzt. Beide Punkte werden auf der Basis des Punktes P_1 berechnet. Nun kann durch eine einfache „zeichneLinie(S_1,S_2)"-Funktion der sichtbare Teil der Geraden durch $\overline{P_1 P_2}$ in das Clipping-Rechteck gezeichnet werden.

Polygone

Das Clipping von Polygonen kann theoretisch als ein sequentieles Clipping von Vektoren angeshen werden. Dabei werden die Kanten eines Polygons in Vektoren umgewandelt und dann an den Bildkanten geclippt. Allerdings kann es dabei zu Fehlern in der Darstellung kommen, wie in Abbildung 9 zu sehen ist.

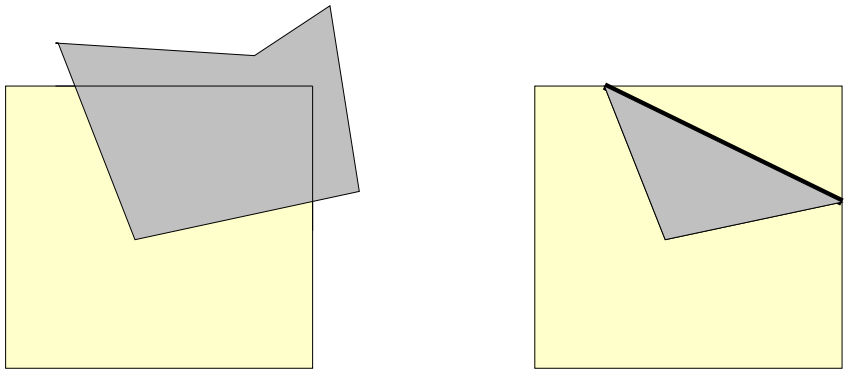

Abbildung 9: Eine mögliche falsche Verbindung beim Polygonclipping

Southerland-Hodgman

Da bei Polygonen, welche eine der Ecken des Clipping-Rechteckes einschließen Fehler auftreten, gibt es verschiedene Methoden diesem Problem entgegen zu treten. Eine recht einfache Methode ist es, dass gesamte Polygon (gesamtheit aller Vektoren) nacheinander

an je einer Fensterkanten zu clippen. Dies ist die wesentliche Idee des Southerland-Hodgeman-Algorithmus.

Dabei müssen die Ergebnisse jedes Zwischenschrittes zwischengespeichert werden. Dies kann durch einen rekursiven Aufbau des Algorithmus realisiert werden. Für die heutigen GKS gibt es eine hardwaremäßige Realisierung durch eine Pipeline von vier identischen Clipper-Stufen ohne Zwischenspeicherung.

Es wird eine Punkteliste für jedes Polygon verwaltet. Nach dem Clippen an einer der Fensterkanten werden aus dieser Punkteliste die nicht im Clipping-Rechteck befindlichen Punkte gestrichen. Falls es zu einem Schnittpunkt mit den Fensterkanten kommt, so werden diese Schnittpunkte in die Punkteliste eingetragen.

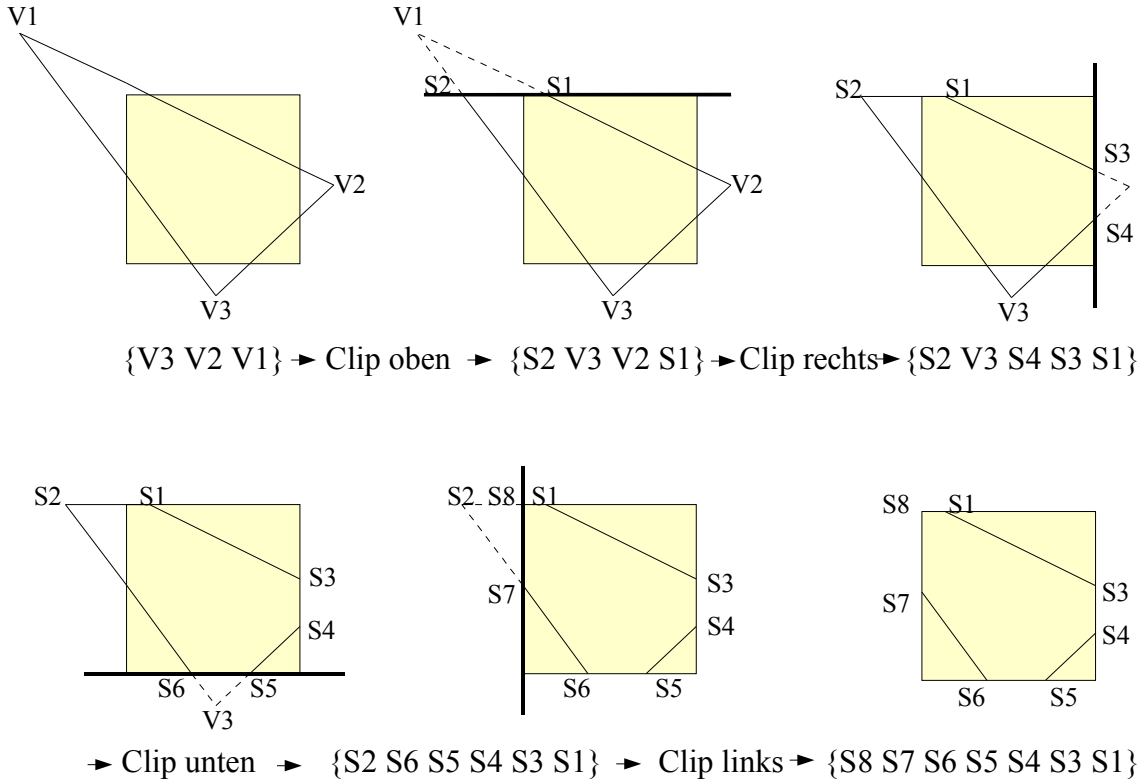

Abbildung 10: Prinzip des Southerland-Hodgeman-Clipping

Allerdings kann es auch bei diesem Algorithmus zu unschönen Effekten kommen. Werden konkave Polygone an einem Rechteck wie in Abbildung 11 geclippt, kann es zu einer unerwünschten Verbindung durch die Fensterkanten kommen. Es entstehen entartete Flächen.

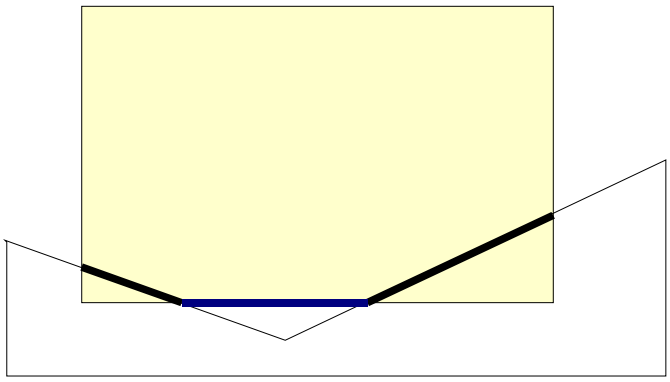

Abbildung 11: unerwünschte Verbindung beim Clipping konkaver Polygone

Diese Verbindung ist in manchen Anwendungen unerwünscht und kann durch zusätzliche Rechenschritte vermieden werden. Der Weiler-Atherton-Algorithmus, welcher sehr allgemein gehalten ist und konkave Polygone gegen beliebige konkave Fenster clippen kann, erzeugt diese Verbindungen nicht.

Text

Die einfachste Art Textzeichen, welche über den Clipping Bereich hinausragen, zu bearbeiten, besteht in dem einfachen Weglassen dieser Zeichen. Allerdings ist dies eine Lösung, die im heutigen Qualitätsverständnis abzulehnen ist. Des Weiteren bieten die bereits beschriebenen Algorithmen die Möglichkeit, auch Text zu clippen.

Die einfachste der vorgestellten Clipping-Methoden besteht darin, die Zeichen in Pixeldarstellung umzuwandeln und das Clipping von Polymarkern durchzuführen. Allerdings sollte diese Methode nur auf die zu bearbeitende Zeile beschränkt werden, um nicht in Performance-Probleme zu geraten. Liegen die Zeichen dagegen als geometrische Formen vor, so können die bereits erwähnten Clipping-Algorithmen verwendet werden.

3D

Geht man nun von zweidimensionalen Darstellungen in die Darstellung von drei dimensionalen Objekten, wird der interessierende Teil der Bildinformationen durch ein sogenanntes *Sichtvolumen* begrenzt. Hierbei gibt es zwei Unterscheidungen: Im Fall einer Paralellprojektion ist das Schichtvolumen ein in Projektionsrichtung unendlich ausgedehnter Quader. Im Fall einer perspektivischen Projektion ist das Sichtvolumen durch eine Pyramide gekennzeichnet.

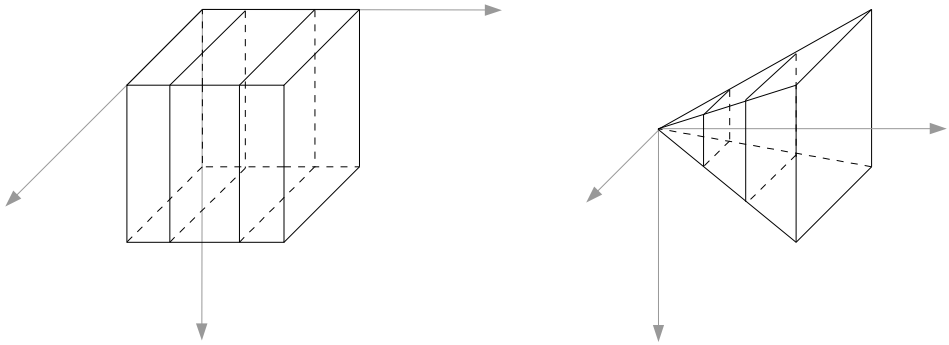

Abbildung 12: Sichtvolumen bei Paralellprojektion und perspektivischer Projektion

Beide Volumina werden allerdings aus praktischen Gründen durch eine vordere und eine hintere Ebene (sog. Clipping Plane) begrenzt, um ein endlichen Raum zu erreichen und um auftretende Wraparound-Fehler zu verhindern.

Die bereits genannten Clipping-Algorithmen werden für eben diesen Fall sinngemäß auf drei Dimensionen erweitert. Auch die Transformation wird sinngemäß erweitert, sodaß in ein normiertes Sichtvolumen transformiert wird. Dadurch wird die Implementierung dieser Algorithmen besonders einfach.

Ist das Schichtvolumen ein Einheitswürfel, so wird ein auf 3D erweiterter Cohen-Sutherland-Clipping-Algorithmus unmittelbar zum Clipping dieser Szene benutzt. Ist das Schichtvolumen eine Pyramide, so kann das parametrische Linien-Clipping, beschrieben durch den Liang-Barsky-Algorithmus, auf 3D erweitert werden. Da allerdings das Clipping in einem normierten Raum stattfindet, muß die Perspektive nach dem Clipping in einem separatem Schritt berechnet werden.

Abbildungsverzeichnis

Literaturverzeichnis

/1/ Fellner, Wolf-Dietrich — Computer Grafik
BI-Wiss.-Verl., Mannheim, Wien, Zürich, 1988

/2/ Foley, van Dam, Feiner, Hughes — Computer Graphics: Principles and Practice
Addison-Wesley, 2002

/3/ J. Encarnação, W. Straßer, R.Klein — Graphische Datenverarbeitung – 4.Auflage
R. Oldenbourg Verlag, München, Wien, 1996

/4/ http://de.wikipedia.org — Wikipedia
die freie Enzyklopädie