



# Computergraphik I

## Clipping

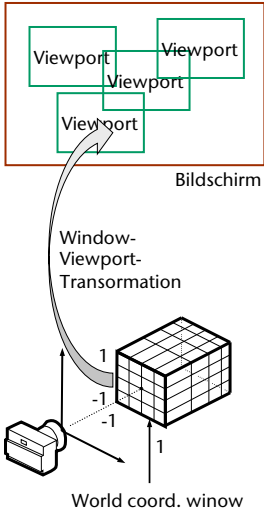


G. Zachmann  
Clausthal University, Germany  
[cg.in.tu-clausthal.de](http://cg.in.tu-clausthal.de)



## Ausschnittsbildung (Windowing und Clipping)

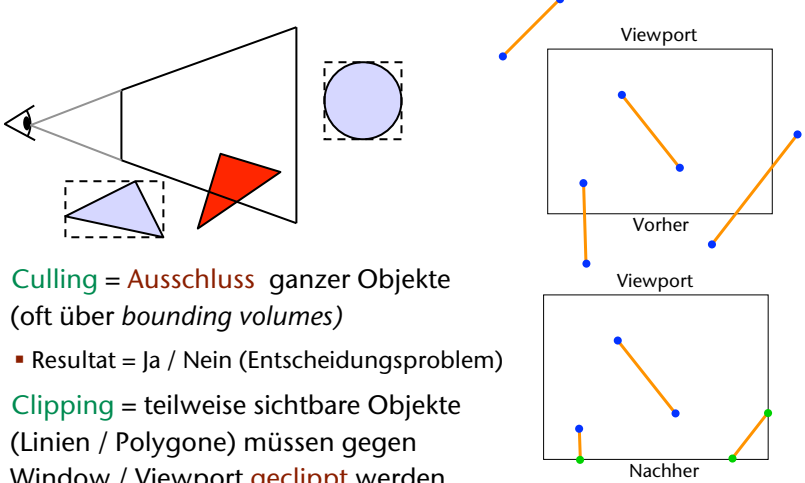
- **Viewport** = Ausgabefenster auf dem Bildschirm
  - Wird in Bildschirmkoordinaten spezifiziert
- **World coordinate window** = Fenster in die Szene (meist einfach "Window")
  - Wird in Weltkoordinaten spezifiziert
- Transformation zwischen Weltkoordinatensystem und Bildschirmkoordinatensystem → **Window-Viewport-Transformation**
- Ohne **Clipping** würden die einzelnen Viewports sich gegenseitig überschreiben



G. Zachmann Computer-Graphik 1 – WS 11/12

Clipping 2

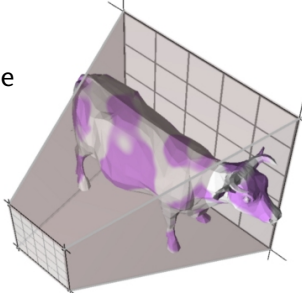
## Unterschiede: Culling und Clipping



- **Culling** = **Ausschluss** ganzer Objekte (oft über *bounding volumes*)
  - Resultat = Ja / Nein (Entscheidungsproblem)
- **Clipping** = teilweise sichtbare Objekte (Linien / Polygone) müssen gegen Window / Viewport **geclipt** werden
  - Resultat = maximales Teil-Objekt, das vollständig im Window liegt (Konstruktionsproblem)

G. Zachmann Computer-Graphik 1 – WS 11/12 Clipping 3

## Station in der Graphik-Pipeline



- Abschneiden der Objekte, die außerhalb des sichtbaren Bereiches liegen (*view frustum*)
- Frühere Graphik-Hardware führte vollständiges Clipping durch – moderne Hardware "kürzt ab"
- Trotzdem sinnvoll, Clipping-Algos kennenzulernen, da oft wiederkehrendes Problem

Modell Transformation

Illumination (Shading)

Viewing Transformation (Perspective / Orthographic)

Clipping

Projektion (in Screen Space)

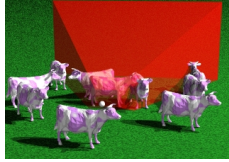
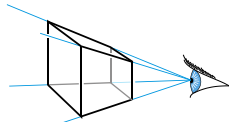
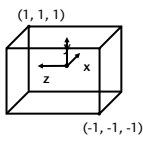
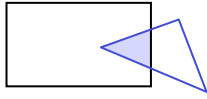
Scan Conversion (Rasterization)

Visibility / Display

G. Zachmann Computer-Graphik 1 – WS 11/12 Clipping 4

## Wann sollte geclippt werden?

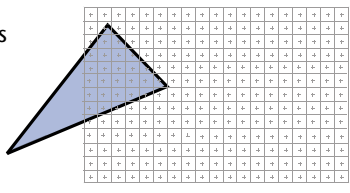
- Vor der Kamera-Transformation im 3D Raum
  - Benötigt 6 allgemeine Ebenengleichungen
- In homogenen Koordinaten nach der Kamera-Transformation im 4D, bevor durch die Perspektive geteilt wird (*Clip space*)
  - ergibt ungewöhnliche w-Werte
  - ist tatsächlich am einfachsten zu implementieren
- Im perspektivisch transformierten 3D-Screen-Space
  - Problem: Objekte in der Kameraebene
- Nach der Projektion am Viewport in 2D
- Während der Rasterisierung für jedes Pixel

G. Zachmann Computer-Graphik 1 – WS 11/12 Clipping 5

## Naives Clipping

- Idee:
  - Führe Clipping während des Rasterisierens aus
  - Teste vor dem tatsächlichen Setzen eines Pixels, ob es innerhalb des Viewports ist
- Vorteil: funktioniert für beliebige Clipping-Windows (auch mit "Löchern")
- Nachteil: evtl. werden sehr viele Pixel ausgerechnet, die dann letztlich doch nicht gezeichnet werden (im worst-case alle)



G. Zachmann Computer-Graphik 1 – WS 11/12 Clipping 6

### Motivation für Clipping vor der Projektion

- Der Fall  $p_z < eye_z$  :

$(eye_x, eye_y, eye_z)$

$z\ axis \rightarrow$

image plane

G. Zachmann Computer-Graphik 1 – WS 11/12 Clipping 9

- Der Fall  $p_z > eye_z$  :

$(eye_x, eye_y, eye_z)$

$z\ axis \rightarrow$

image plane

G. Zachmann Computer-Graphik 1 – WS 11/12 Clipping 10

Der Fall  $p_z = eye_z!$

$(eye_x, eye_y, eye_z)$  z axis  $\rightarrow$

image plane

Clipping 11

Lösung: Full Clipping

clip geometry to view frustum

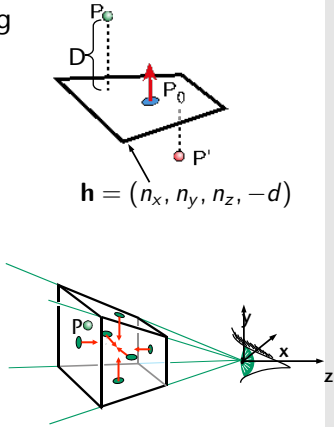
$(eye_x, eye_y, eye_z)$  z axis

image plane

Clipping 12

## Clipping in Bezug auf das View Frustum

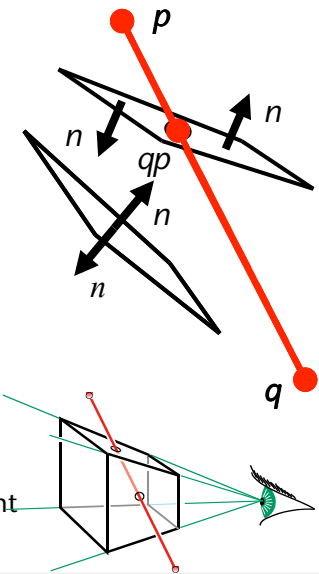
- Erinnerung: homogene Ebenengleichung
 
$$\mathbf{h} \cdot \mathbf{p} = 0$$
- Clipping eines Punktes:
  - Teste gegen alle 6 Ebenen
  - Annahme: Normalenvektoren sind nach **innen** gerichtet
  - Verwerfe Punkt P wenn für eine Ebene
 
$$\mathbf{h} \cdot \mathbf{p} < 0$$
- Klappt für beliebige konvexe Polyeder



G. Zachmann Computer-Graphik 1 – WS 11/12 Clipping 13

## Clipping eines Linien-Segmentes

- 4 Fälle:
  - Wenn  $\mathbf{h} \cdot \mathbf{p} > 0 \wedge \mathbf{h} \cdot \mathbf{q} < 0$ 
    - Ersetze q
  - Wenn  $\mathbf{h} \cdot \mathbf{p} < 0 \wedge \mathbf{h} \cdot \mathbf{q} > 0$ 
    - Ersetze p
  - Wenn  $\mathbf{h} \cdot \mathbf{p} > 0 \wedge \mathbf{h} \cdot \mathbf{q} > 0$ 
    - "pass through"
  - Wenn  $\mathbf{h} \cdot \mathbf{p} < 0 \wedge \mathbf{h} \cdot \mathbf{q} < 0$ 
    - Komplett verwerfen ("reject")
- Für das ganze Frustum: alle Ebenen auf diese Weise durchlaufen
- Das Ergebnis ist ein einzelnes Segment (warum?)



G. Zachmann Computer-Graphik 1 – WS 11/12 Clipping 15

### Ist dieses Clipping effizient?

- Was ist das Problem?
  - Die Berechnung der Schnittpunkte und aller dazugehörigen interpolierten Werte sind — in diesem Fall — unnötig!
  - Kann man dies früher erkennen?

G. Zachmann Computer-Graphik 1 – WS 11/12 Clipping 17

### Erhöhung der Effizienz: Outcodes

- Berechne die "Sidedness" jedes Vertex bzgl. jeder Ebene
  - 0 = "richtige" Seite (Vorderseite); 1 = "falsche" Seite (Rückseite)
  - Ergibt pro Vertex einen 6-Bit langen *Outcode* (4 Bit im 2D)
- Bedingung:  $out(P) \& out(Q) \neq 0 \rightarrow$  "trivial reject"
 

Bitweises UND!!
- Bsp.:
 

P	H <sub>1</sub>	H <sub>2</sub>	Q	
1010	0010	0110	0110	
1000	0000	0100	0100	H <sub>3</sub>
1001	0001	0101	0101	H <sub>4</sub>

Outcode von P : 1010

Outcode von Q : 0110

---

AND : 0010

$\rightarrow$  "trivial reject", da  $\neq 0$

G. Zachmann Computer-Graphik 1 – WS 11/12 Clipping 18

▪ Weiteres Beispiel:

Outcode von P : 1000  
 Outcode von Q : 0010

---

AND : 0000  
 → "potentially visible"

▪ In diesen Fällen macht der Test also keine Aussage!  
 ▪ Dies ist der sog. **Cohen-Sutherland-Algorithmus**

G. Zachmann Computer-Graphik 1 – WS 11/12 Clipping 19

▪ Weitere Aussage auf Basis der Outcodes:

$out(P) \mid out(Q) \neq 0 \rightarrow$  "trivial accept"  
 ↗ Bitweises ODER!!

▪ Beispiele:

Linie	out(A)	out(B)	AND	OR
AB	0000	0000	0000	0000
CD	0000	1000	0000	1000
EF	0001	1001	0001	1001
GH	0100	0010	0000	0110
IJ	0100	0010	0000	0110

G. Zachmann Computer-Graphik 1 – WS 11/12 Clipping 20

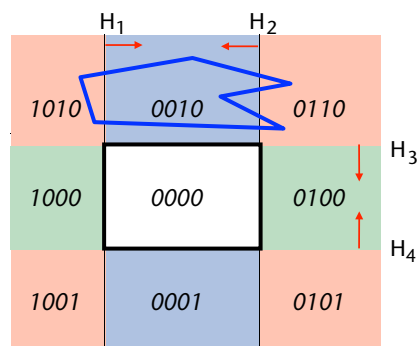


## Der Cohen-Sutherland-Algorithmus

- Genereller Trick, der hier angewandt wird:
  - Erst einfache Tests durchführen, ob Clipping nötig ist
    - Die Tests liefern evtl. keine definitive Antwort, aber dafür sind sie sehr schnell
  - Dann erst im "nötigen" Fall die (teuren) mathematischen Operationen durchführen
- Der Code für die Outcodes (2D-Fall, rechteckiges Clip-Window):

```
unsigned int outcode( int x, int y )
{
    unsigned int c = 0;
    if ( y > ymax ) c = c | 8; // 1000
    if ( y < ymin ) c = c | 4; // 0100
    if ( x > xmax ) c = c | 2; // 0010
    if ( x < xmin ) c = c | 1; // 0001
    return c;
}
```

- Das Gute an dem Test: er funktioniert für **beliebige Primitive** in **beliebigen Dimensionen** mit beliebigen, **konvexen Clip-Windows!**
- Beispiel:



Outcode of p : 1010

Outcode of q : 1010

Outcode of r : 0110

Outcode of s : 0010

Outcode of t : 0110

Outcode of u : 0010

---

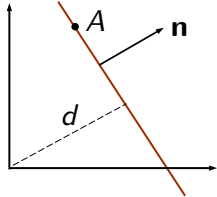
AND : 0010

→ Clipped

## Clipping in homogenen Koordinaten

- Erinnerung:
  - Ein Punkt  $P = (x, y, z) \in \mathbb{R}^3 \equiv$  Vektor  $\hat{\mathbf{p}} = \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \in \mathbb{R}^4$  in homogenen Koord.
  - Alle Vielfache des homogenen Vektors entsprechen demselben Punkt:
 
$$s \cdot \hat{\mathbf{p}} \equiv P$$
- Sei eine Ebene gegeben durch  $\mathbf{n}, A$
- Dann liegt P in der Ebene  $\Leftrightarrow$ 

$$(P - A) \cdot \mathbf{n} = P \cdot \mathbf{n} - \underbrace{A \cdot \mathbf{n}}_d \equiv \hat{\mathbf{p}} \cdot \begin{pmatrix} n_x \\ n_y \\ n_z \\ -d \end{pmatrix} = 0$$
- Der Vektor  $\hat{\mathbf{n}} = (n_x, n_y, n_z, -d)$  heißt auch **homogene Darstellung** der Ebene, oder **homogene Normale**



G. Zachmann Computer-Graphik 1 – WS 11/12 Clipping 24

- Fazit: Punkt/Vektor im 3D  $\equiv$  Vektor im 4D  
Ebene im 3D  $\equiv$  Vektor im 4D
- Bemerkung: alle Vektoren  $t \cdot \hat{\mathbf{n}}$  beschreiben die gleiche Ebene
 
$$\hat{\mathbf{p}} \cdot \hat{\mathbf{n}} = 0 = s \hat{\mathbf{p}} \cdot t \hat{\mathbf{n}}$$

G. Zachmann Computer-Graphik 1 – WS 11/12 Clipping 25

### Outcodes in homogenen Koordinaten

- Betrachte das Clip-Window  $(0,0,0) - (1,1,1)$ 
  - Die homogene Darstellung der beiden Ebenen  $x=0$  und  $x=1$  ist
 
$$\mathbf{n}_1 = (1 \ 0 \ 0 \ 0) \quad \mathbf{n}_2 = (-1 \ 0 \ 0 \ 1)$$
- Definiere die *boundary distance (BD)*

$$d_i = \mathbf{p} \cdot \mathbf{n}_i, \quad \mathbf{p} = (x, y, z, w)$$
  - Berechnung ist trivial; z.B.  $d_1 = x \quad d_2 = w - x$
  - Erstelle Tabelle aller BDs  $\rightarrow$

BD	homog. Wert	Ebene
$d_1$	$x$	$x=0$
$d_2$	$w - x$	$x=1$
$d_3$	$y$	$y=0$
$d_4$	$w - y$	$y=1$
$d_5$	$z$	$z=0$
$d_6$	$w - z$	$z=1$

- Outcodes sind auch trivial zu bestimmen:
 
$$i(\mathbf{p}) = (d_i)$$

$$= \begin{cases} 1 & , P \text{ au\ss}erhalb Ebene i \\ 0 & , P \text{ innerhalb Ebene i} \end{cases}$$

G. Zachmann Computer-Graphik 1 – WS 11/12 Clipping 26

### Trivial accept / reject

- Sei Linie PQ gegeben; in homogenen Koordinaten:  $\mathbf{p}, \mathbf{q}$
- Wie bisher auch:
  - $\text{out}(\mathbf{p}) \wedge \text{out}(\mathbf{q}) \neq 000000 \Rightarrow$  trivial reject
  - $\text{out}(\mathbf{p}) \vee \text{out}(\mathbf{q}) = 000000 \Rightarrow$  trivial accept
- Ansonsten: es mu\ss mindestens eine Bitposition  $i$  geben, wo  $\text{out}_i(\mathbf{p}) = 0$  und  $\text{out}_i(\mathbf{q}) = 1$  oder umgekehrt.
- Schneide  $X(t)$  mit dieser Clip-Window-Ebene:
 
$$X(t) = P_0 + t(Q - P) \leftrightarrow \hat{x}(t) = \mathbf{p} + t(\mathbf{q} - \mathbf{p})$$

$$\hat{x}(t) \cdot \mathbf{n}_i = [\mathbf{p} + t(\mathbf{q} - \mathbf{p})] \cdot \mathbf{n}_i = \mathbf{p}\mathbf{n}_i + t(\mathbf{q}\mathbf{n}_i - \mathbf{p}\mathbf{n}_i)$$

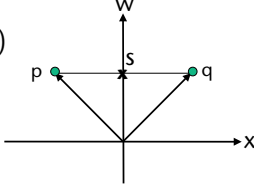
$$= d_i^p + t(d_i^q - d_i^p) \stackrel{!}{=} 0$$

$$t = \frac{d_i^p}{d_i^p - d_i^q}$$
- Bemerkung:  $t \in \{0, 1\} \Leftrightarrow \text{sign}(d_i^p) \neq \text{sign}(d_i^q)$

G. Zachmann Computer-Graphik 1 – WS 11/12 Clipping 27

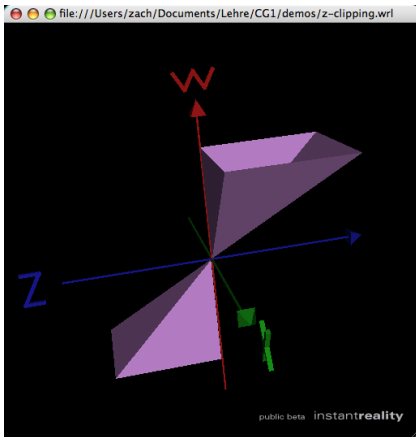
Beispiel

- $P = (-1 \ 0 \ 0)$   $Q = (1 \ 0 \ 0)$  Ebene 1:  $x = 0$
- $\mathbf{p} = (-1 \ 0 \ 0 \ 1)$   $\mathbf{q} = (1 \ 0 \ 0 \ 1)$   $\mathbf{n}_1 = (1 \ 0 \ 0 \ 0)$
- Parameter des Schnittpunktes  $\mathbf{s}$ :
 
$$t = \frac{d_1^P}{d_1^P - d_1^Q} = \frac{-1}{-1 - (+1)} = \frac{1}{2}$$
- Schnittpunkt:  $\mathbf{s} = \mathbf{p} + \frac{1}{2}(\mathbf{q} - \mathbf{p}) = (0 \ 0 \ 0 \ 1)$
- Frage: was ist mit  $\mathbf{p}' = (-2 \ 0 \ 0 \ 2)$ ? (Ist derselbe Punkt P in 3D!)
  - Parameter  $t' = \frac{2}{3}$
  - Ist das ein anderer Punkt?
  - Nein, denn Schnittpunkt
$$\mathbf{s}' = \mathbf{p}' + \frac{2}{3}(\mathbf{q} - \mathbf{p}') = \begin{pmatrix} -2 \\ 0 \\ 0 \\ 2 \end{pmatrix} + \frac{2}{3} \begin{pmatrix} 3 \\ 0 \\ 0 \\ -1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \equiv \mathbf{s}$$



G. Zachmann Computer-Graphik 1 – WS 11/12 Clipping 28

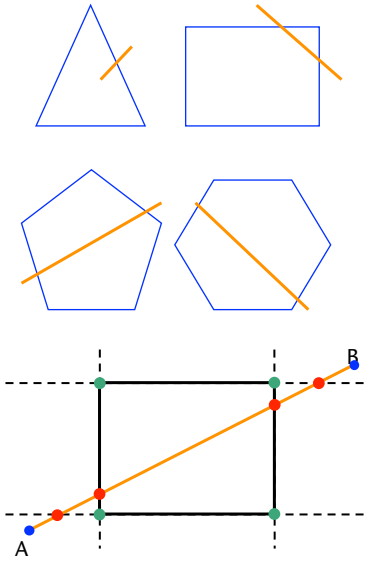
Visualisierung der Clipping-Region im 4D:



G. Zachmann Computer-Graphik 1 – WS 11/12 Clipping 29

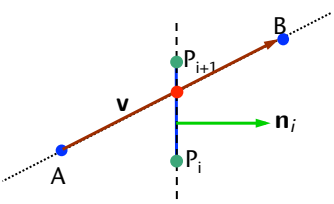
## Der Cyrus-Beck-Algorithmus [1978]

- Clippen von Linien an beliebigen, konvexen Clip-Windows
- Verwendet die Parameterdarstellung der Linien
- Im Folgenden wird wieder das Rechteck als Beispiel verwendet; der Algorithmus ist aber für beliebige (konvexe) Clip-Windows anwendbar
- Bei einem n-Eck kann es bis zu n Schnittpunkte geben
- Nur 2 davon sind echte Schnittpunkte mit dem Clip-Window



G. Zachmann Computer-Graphik 1 – WS 11/12 Clipping 30

- Berechne für jeden Clipping-Rand  $i$  das zugehörige  $t$ :

$$t = -\frac{(A - P_i) \cdot \mathbf{n}_i}{\mathbf{v} \cdot \mathbf{n}_i}$$


- Die Werte  $t < 0$  und  $t > 1$  werden ignoriert
- Jeweils genau ein  $t$  markiert den Eintrittspunkt und den Austrittspunkt der Linie

G. Zachmann Computer-Graphik 1 – WS 11/12 Clipping 31

### Entering and Leaving $t$ 's

- Idee: klassifiziere alle  $t$ 's
  - Konvention: Normalen zeigen nach *außen*
  - "Leaving", falls:
 
$$\mathbf{n}_i \cdot \mathbf{v} > 0 \Rightarrow t_i^l$$
  - "Entering", falls:
 
$$\mathbf{n}_i \cdot \mathbf{v} < 0 \Rightarrow t_i^e$$
  - Sonst: Sonderfall, der anderweitig abgefangen wird

G. Zachmann Computer-Graphik 1 – WS 11/12 Clipping 32

### Der Algorithmus

- Berechne das Maximum und Minimum
 
$$t_e = \max\{t_i^e, 0\} \quad t_l = \min\{t_i^l, 1\}$$
- Falls  $t_e > t_l \rightarrow$  Linie ist komplett außerhalb des Clip-Windows
- Sonst:  $t_e$  und  $t_l$  definieren die Enden der geclippten Linie
- Alternative Betrachtungsweise:
  - Starte mit Intervall  $[0,1]$
  - Schneide Linie der Reihe nach gegen jeden Clip-Rand
  - Falls "entering"  $\rightarrow$  schneide aktuelles Intervall unten ab (falls überhaupt)
  - Falls "leaving"  $\rightarrow$  schneide aktuelles Intervall oben ab
  - Stop, falls Intervall leer wird

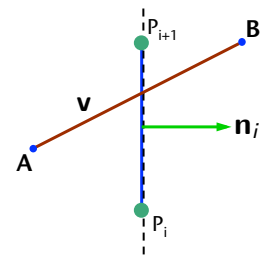
G. Zachmann Computer-Graphik 1 – WS 11/12 Clipping 33

### Sonderfälle

- Wann kann der Nenner Null werden?

$$t = -\frac{(A - P_i) \cdot \mathbf{n}_i}{\mathbf{v} \cdot \mathbf{n}_i}$$

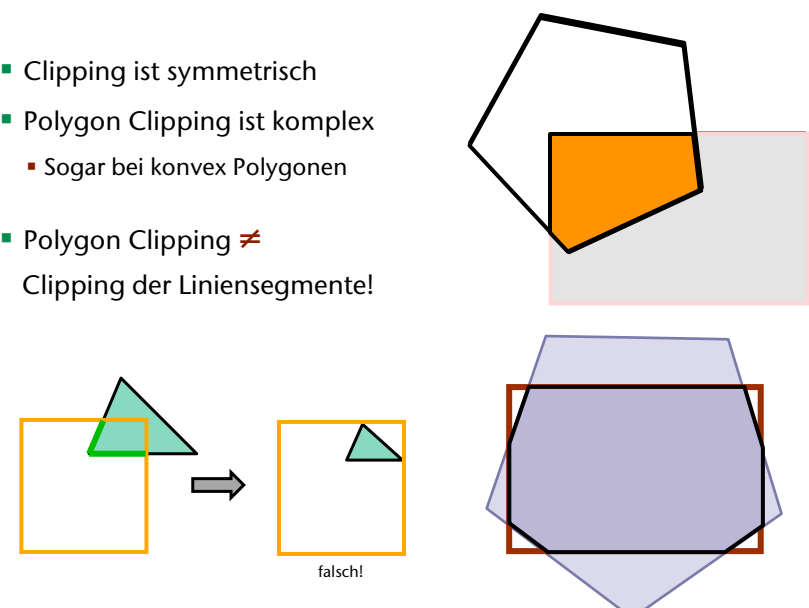
- $\mathbf{v} = 0$  :
  - Start- und Endpunkt der Linie sind identisch (muß vorher abgefangen werden)
- $\mathbf{n}_i = 0$  :
  - Nur, falls 2 Punkte des Clip-Windows identisch sind, vorher abfangen
- $\mathbf{n}_i \cdot \mathbf{v} = 0$  :
  - Zu zeichnende Linie ist parallel zu einer Kante des Clip-Objekts → kein t ausrechnen, nächsten Clip-Rand betrachten



G. Zachmann Computer-Graphik 1 – WS 11/12 Clipping 34

### Polygon Clipping in 2D

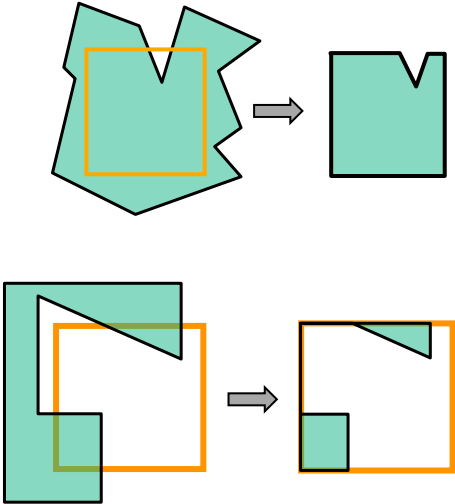
- Clipping ist symmetrisch
- Polygon Clipping ist komplex
  - Sogar bei konvex Polygonen
- Polygon Clipping  $\neq$  Clipping der Liniensegmente!



falsch!

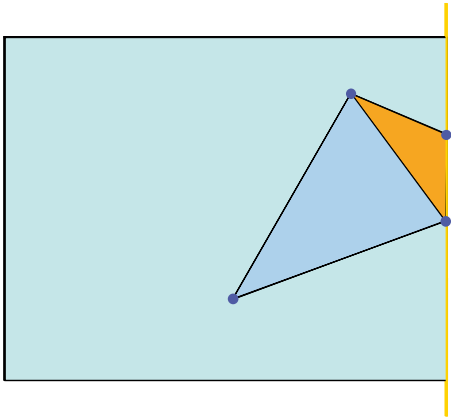
G. Zachmann Computer-Graphik 1 – WS 11/12 Clipping 35

- Polygon-Clipping kann, insbesondere bei nicht-konvexen Polygonen, unangenehm werden
- Es können sogar mehrere Polygone entstehen



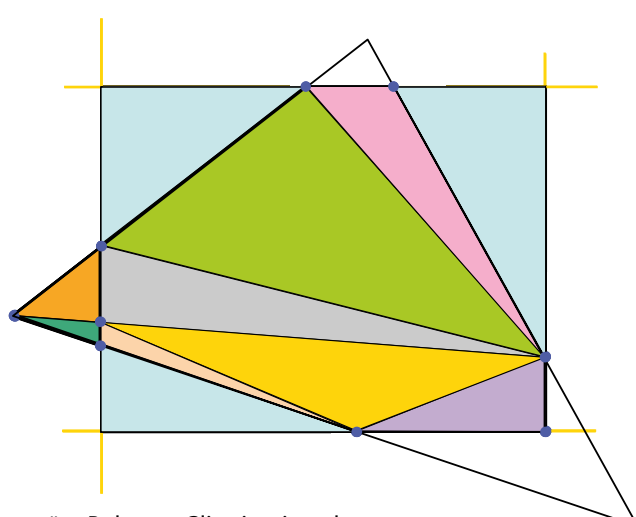
G. Zachmann Computer-Graphik 1 – WS 11/12 Clipping 36

### Naiver Clipping-Algorithmus



G. Zachmann Computer-Graphik 1 – WS 11/12 Clipping 37



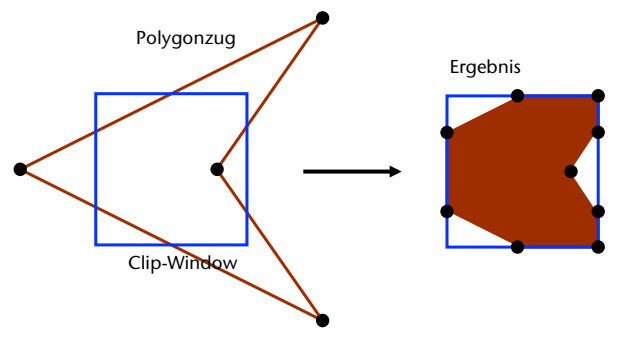


Fazit: das naive Polygon-Clipping ist schon für Dreiecke zu ineffizient (erzeugt viel zu viel Output)

G. Zachmann Computer-Graphik 1 – WS 11/12 Clipping 38

### Sutherland-Hodgman

- Clipping eines Polygonzugs gegen ein konvexes Clip-Polygon (z.B. Viewport); der Polygonzug darf konkav sein



G. Zachmann Computer-Graphik 1 – WS 11/12 Clipping 39

## Vorgehen

- Eingabe =
  - Liste der Eckpunkte in der richtigen Reihenfolge (gegen den Uhrzeigersinn)
  - Menge von Clip-Kanten, die ein konvexes Clip-Window definieren
- Betrachte eine (beliebige) Clip-Kante:
  - Nach dem Schnitt mit dieser Clip-Kante wird eine neue Liste von Eckpunkten erzeugt
  - Dieses Ergebnis ist wieder ein geschlossener Polygonzug
  - Alle Punkte des neuen Polygons befinden sich auf der "Innenseite" (der "richtigen" Seite) dieser Clip-Kante (Schleifeninvariante)
- Das wird mit allen Clip-Kanten wiederholt
  - (Im Prinzip ist die Reihenfolge der Clip-Kanten egal)

G. Zachmann Computer-Graphik 1 – WS 11/12 Clipping 40

Der Polygonzug wird der Reihe nach an den Clip-Kanten geschnitten

G. Zachmann Computer-Graphik 1 – WS 11/12 Clipping 41

### 4 Fälle

- Annahme: der Punkt A wurde bereits behandelt

Beide Punkte drinnen: → Output B

Linie „zeigt“ nach außen: → Output S

Linie „zeigt“ nach innen: → Output S, B

Beide Punkte draußen: → kein Output

G. Zachmann Computer-Graphik 1 – WS 11/12 Clipping 42

### Beispiel

Input: A B C D

Output: S<sub>1</sub> A B C S<sub>2</sub>

Man beginnt mit der Kante [letzter Punkt – erster Punkt], hier also D-A. (D wird am Ende der Schleife betrachtet)

G. Zachmann Computer-Graphik 1 – WS 11/12 Clipping 43

Input:  $S_1 A B C S_2$

Output:  $S_3 S_1 A S_4 S_5 C S_6$

G. Zachmann Computer-Graphik 1 – WS 11/12 Clipping 44

- Kleines Problem des Sutherland-Hodgeman-Algos: falls das ursprüngliche Polygon in mehrere Teile zerfällt beim Clipping, dann entsteht eine unschöne Polygonkante am Rand des Windows
- Beispiel:

G. Zachmann Computer-Graphik 1 – WS 11/12 Clipping 45

Eingabe:  
 $P_1 P_2 P_3 P_4 P_5 P_6 P_7 P_8$

G. Zachmann Computer-Graphik 1 – WS 11/12 Clipping 47

Etwas komplexeres Beispiel

Eingabe:  
 $P_1 P_2 P_3 P_4 P_5 P_6 P_7 P_8$

Ausgabe:  
 $l_1 P_2 P_3 P_4 P_5 l_2 l_3 P_7 l_4$

G. Zachmann Computer-Graphik 1 – WS 11/12 Clipping 48

Eingabe:  
 $I_1 P_2 P_3 P_4 P_5 I_2 I_3 P_7 I_4$

Ausgabe:  
 $I_1 I_5 I_6 P_4 P_5 I_2 I_3 P_7 I_4$

G. Zachmann Computer-Graphik 1 – WS 11/12 Clipping 49

Eingabe:  
 $I_1 I_5 I_6 P_4 P_5 I_2 I_3 P_7 I_4$

Ausgabe:  
 $I_1 I_5 I_6 P_4 I_9 I_{10} I_3 P_7 I_4$

G. Zachmann Computer-Graphik 1 – WS 11/12 Clipping 50

Ausgabe:  
 $I_7 I_8 I_6 P_4 I_9 I_{10} I_3 P_7 I_4$

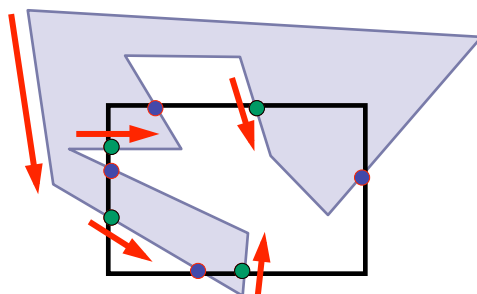
G. Zachmann Computer-Graphik 1 – WS 11/12 Clipping 51

### Der Algorithmus von Weiler-Atherton

- Strategie: "Wandere" auf dem Polygonzug oder dem Window-Rand
- Konvention (wie immer): Polygone sind CCW orientiert

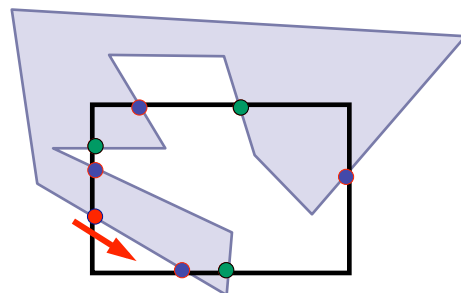
G. Zachmann Computer-Graphik 1 – WS 11/12 Clipping 56

- Berechne alle Schnittpunkte
- Markiere die Punkte, an denen das Polygon in das Clipping-Window eintritt (hier grün)



G. Zachmann Computer-Graphik 1 – WS 11/12 Clipping 57

- Solange noch ein unbearbeiteter Eintrittsschnittpunkt vorhanden ist wird das Polygon weiter umlaufen
- Die Ausgabe ist ein **oder mehrere** Listen von Punkten (Eckpunkte des Polygons, Schnittpunkte, und/oder Eckpunkte des Windows)
  - Unterschied zu Sutherland-Hodgeman: die Eingabe wird **nur einmal** abgearbeitet

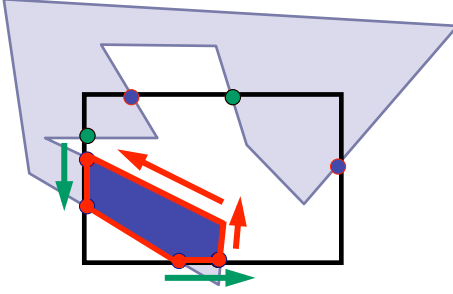


G. Zachmann Computer-Graphik 1 – WS 11/12 Clipping 58



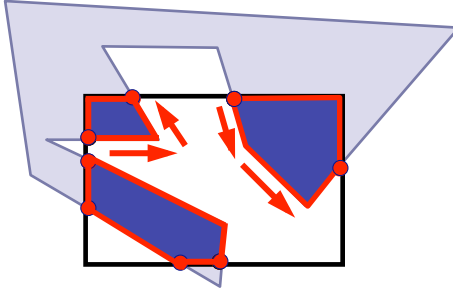
## Umlauf-Regeln

- Trifft man beim Umlauf auf einen Schnittpunkt, dann:
  - Füge Schnittpunkt zur Ausgabe hinzu
  - Falls Schnittpunkt = "Out-to-in": folge dem Polygonzug (ccw)
  - Falls Schnittpunkt = "In-to-out": folge dem Window-Rand (ccw)



G. Zachmann Computer-Graphik 1 – WS 11/12 Clipping 59

- Solange noch ein unbearbeiteter Eintrittsschnittpunkt vorhanden ist wird das Polygon weiter umlaufen



- Der Weiler-Atherton-Algorithmus erzeugt echt separate Polygonzüge für jedes sichtbare Fragment

G. Zachmann Computer-Graphik 1 – WS 11/12 Clipping 60

### Implementierung des Algorithmus

- Eingabe (hier ist die Konvention *clockwise ordering*):

The diagram illustrates the input for a clipping algorithm. On the left, a gray polygon is shown with vertices labeled 0 through 9 in a clockwise direction. A black clip window is overlaid on it, with vertices labeled a, b, c, and d. The clip window is a rectangle with vertices a (top-left), b (top-right), c (bottom-right), and d (bottom-left). The polygon's vertices are numbered 0 to 9, with 0 at the top-left corner of the clip window. The clip window's vertices are labeled a, b, c, and d. To the right, two cyclic lists are shown. The first list, labeled 'Zyklische Liste der Vertices des Polygons', contains vertices 0 through 9 in a vertical column, with arrows indicating a clockwise cycle from 0 to 1, 2, 3, 4, 5, 6, 7, 8, 9, and back to 0. The second list, labeled 'Zyklische Liste der Vertices des Clip-Windows', contains vertices a, b, c, and d in a vertical column, with arrows indicating a clockwise cycle from a to b, b to c, c to d, and d back to a.

G. Zachmann Computer-Graphik 1 – WS 11/12 Clipping 61

- Finde die Schnittpunkte und füge sie in **beide** Listen ein

The diagram shows the same polygon and clip window as in slide 61. An intersection point 'i' is now shown where the top edge of the clip window (between vertices a and b) intersects the top edge of the polygon (between vertices 0 and 1). This intersection point 'i' is added to both cyclic lists. In the polygon list, 'i' is inserted between vertices 0 and 1. In the clip window list, 'i' is inserted between vertices a and b. The cyclic lists are now: 0, i, 1, 2, 3, 4, 5, 6, 7, 8, 9 for the polygon, and a, i, b, c, d for the clip window. Arrows indicate the clockwise direction of the cycles.

G. Zachmann Computer-Graphik 1 – WS 11/12 Clipping 62

Find the intersection points and add them to **both** lists

**Add Vertex i:**

G. Zachmann Computer-Graphik 1 – WS 11/12 Clipping 63

Find the intersection points and add them to **both** lists

**Add Vertex k:**

G. Zachmann Computer-Graphik 1 – WS 11/12 Clipping 64

Find the intersection points and add them to both lists

Add Vertex j:

The diagram on the left shows two overlapping polygons. The left polygon has vertices labeled 0 through 9 in a clockwise order starting from the top-left. The right polygon has vertices labeled 1 through 8. Their intersection points are labeled a, b, c, and d. Dashed lines indicate the bounding box of the intersection. The graph on the right, titled 'Add Vertex j:', shows a directed graph with nodes 0 through 9 and nodes a, b, c, d. Blue arrows indicate the edges of the polygons. Node 0 is the start of the left polygon, and node 9 is the end. Node 1 is the start of the right polygon, and node 8 is the end. The intersection points a, b, c, and d are added as new vertices on the edges of the polygons.

G. Zachmann Computer-Graphik 1 – WS 11/12 Clipping 65

Classify each intersection point as „entering“ or „leaving“

The diagram on the left is identical to the one in slide 65. The graph on the right is also identical, but the intersection points a, b, c, and d are now classified as 'entering' or 'leaving'. Node 'a' is labeled 'entering' (in red), node 'b' is labeled 'leaving' (in green), node 'c' is labeled 'entering' (in red), and node 'd' is labeled 'leaving' (in green). The edges of the polygons are shown in blue.

G. Zachmann Computer-Graphik 1 – WS 11/12 Clipping 66

- Erzeuge das geclippte Polygon:
  - Starte bei einem "entering"-Vertex
  - Falls man auf einen "leaving"-Vertex trifft, dann wechsele auf die Liste des Clip-Polygons (blaue Zeiger)
  - Falls man auf einen "entering"-Vertex trifft, dann wechsele auf die Liste des Polygons (schwarze Zeiger)
  - Eine Polygonzug ist beendet, wenn der Startpunkt wieder erreicht ist
  - Wiederhole, solange noch unbesuchte "entering"-Vertices vorhanden sind

Clipping 67

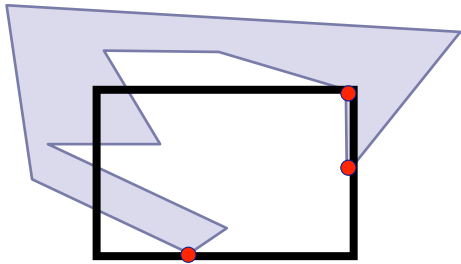
Beispiel

- Polygonzug 1 = *l*, 4, 5, *k*
- Polygonzug 2 = *j*, 9, 0, *l*

Clipping 68

## Robustheit, Präzision, Entartungen

- Die üblichen (leidigen) Fragen:
  - Was passiert wenn ein Vertex (beinahe) auf dem Rand des anderen Polygonzuges liegt?



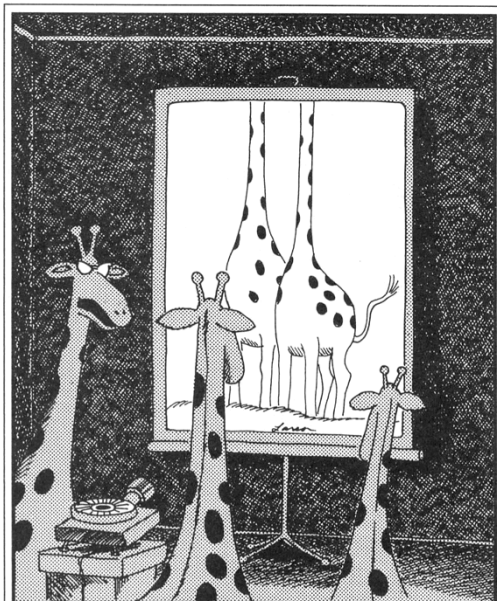
G. Zachmann Computer-Graphik 1 – WS 11/12 Clipping 69

## Absch(l)ießende Bemerkungen

- Ältere Hardware machte volles Clipping
- Moderne Hardware vermeidet Clipping weitgehend:
  - Nur bzgl. der Ebene  $z=z_0$
- Im Allgemeinen ist es nützlich, Clipping zu kennen, da es viele ähnliche geometrische Probleme und Algorithmen gibt, z.B.:
  - Zur Bestimmung, welche Objekte innerhalb eines "Picking-Frustums" liegen
  - Schnittpunkte zwischen Objekten
  - Berechnung analytischer Schatten

Modeling Transformations
Illumination (Shading)
Viewing Transformation (Perspective / Orthographic)
Clipping
Projection (to Screen Space)
Scan Conversion (Rasterization)
Visibility / Display

G. Zachmann Computer-Graphik 1 – WS 11/12 Clipping 70



“Oh, lovely — just the hundredth time you’ve managed to cut everyone’s head off.”