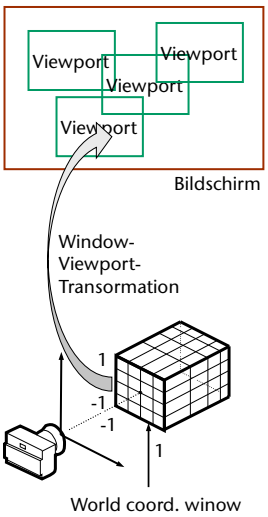
Computergraphik I
Clipping

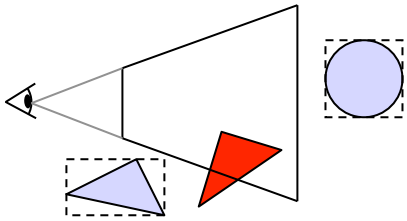
G. Zachmann
Clausthal University, Germany
cg.in.tu-clausthal.de

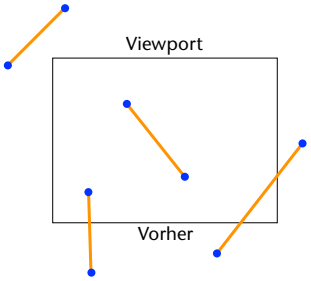
Ausschnittsbildung (Windowing und Clipping)

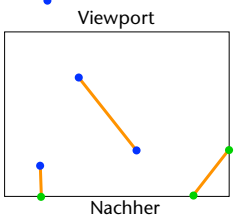
- Viewport = Ausgabefenster auf dem Bildschirm
  - Wird in Bildschirmkoordinaten spezifiziert
- World coordinate window = Fenster in die Szene (meist einfach "Window")
  - Wird in Weltkoordinaten spezifiziert
- Transformation zwischen Weltkoordinatensystem und Bildschirmkoordinatensystem → Window-Viewport-Transformation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



Unterschiede: Culling und Clipping



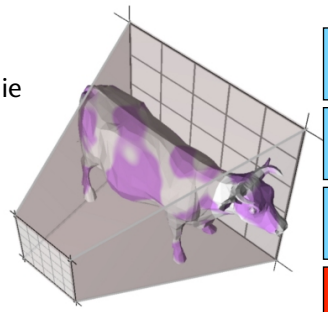




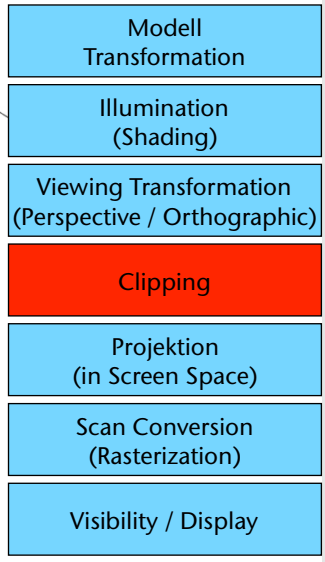
- Culling = Ausschluss ganzer Objekte (oft über *bounding volumes)*
  - Resultat = Ja / Nein (Entscheidungsproblem)
- Clipping = teilweise sichtbare Objekte (Linien / Polygone) müssen gegen Window / Viewport geclippt werden
  - Resultat = maximales Teil-Objekt, das vollständig im Window liegt (Konstruktionsproblem)

Station in der Graphik-Pipeline

- Abschneiden der Objekte, die außerhalb des sichtbaren Bereiches liegen (*view frustum*)

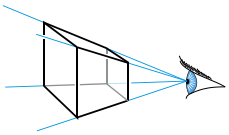


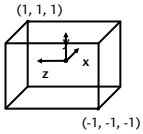
- Frühere Graphik-Hardware führte vollständiges Clipping durch — moderne Hardware "kürzt ab"
- Trotzdem sinnvoll, Clipping-Algos kennenzulernen, da oft wiederkehrendes Problem

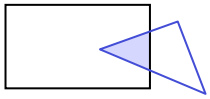


Wann sollte geclippt weden?

- 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

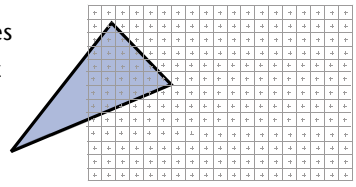




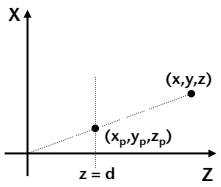


Naives Clipping

- 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)



- Perspektivische Projektion: 2 konzeptionelle Schritte
  - 4x4 Matrix
  - Homogenisierung
    - wird nicht immer benötigt
    - moderne Grafikhardware führt die meisten Operationen mit homogenen Koordinaten in 2D aus



homogenize

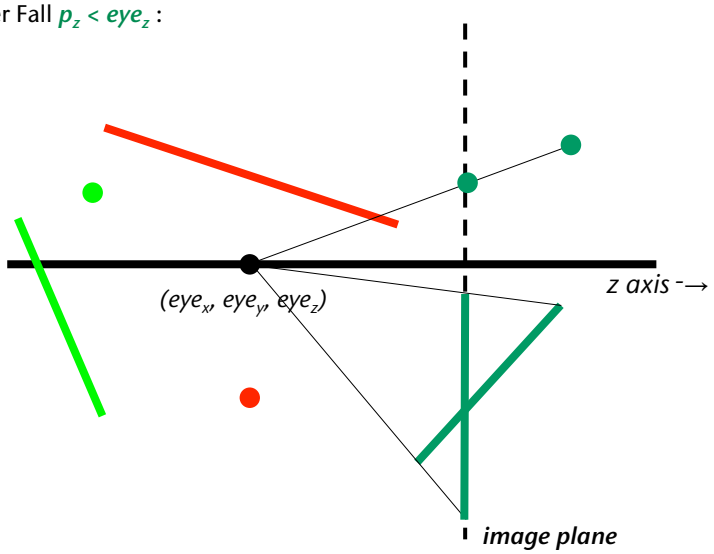
$$\begin{pmatrix} x * d / (z+d) \\ y * d / (z+d) \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ 0 \\ (z+d)/d \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1/d & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

homogenize

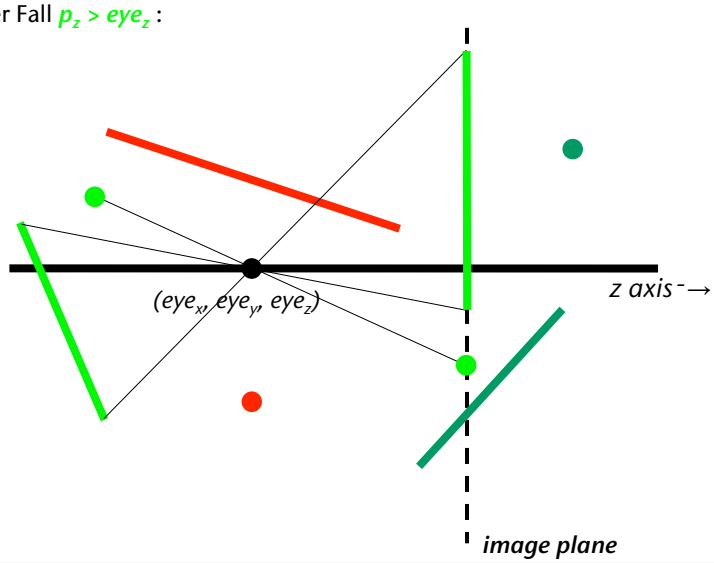
$$\begin{pmatrix} x \cdot d/z \\ y \cdot d/z \\ d/z \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ 1 \\ z/d \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1/d & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

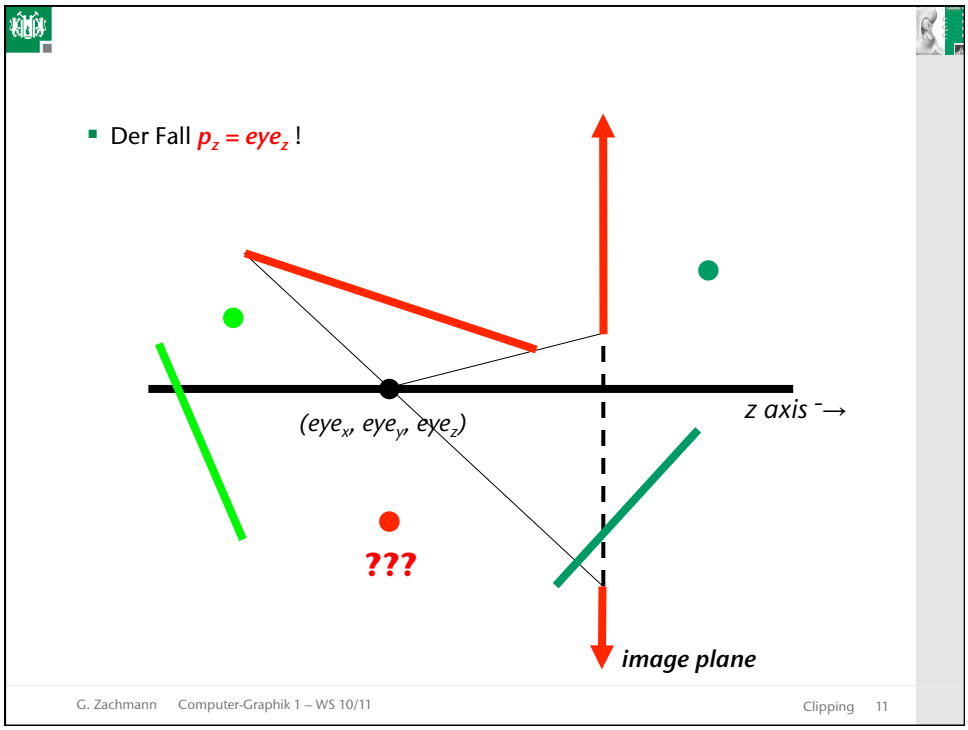
Motivation für Clipping vor der Projektion

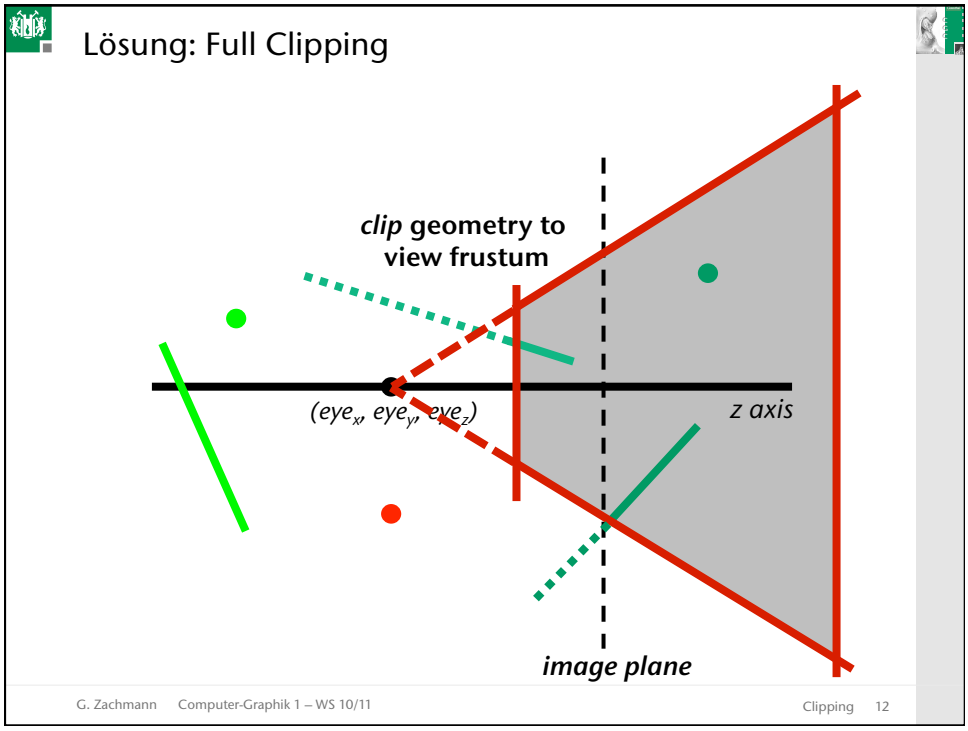
- Der Fall $p_z < eye_z$:



- Der Fall $p_z > eye_z$:




Der Fall $p_z = eye_z$!
(eye_x, eye_y, eye_z)
z axis
???
image plane
G. Zachmann Computer-Graphik 1 – WS 10/11 Clipping 11


Lösung: Full Clipping
clip geometry to view frustum
(eye_x, eye_y, eye_z)
z axis
image plane
G. Zachmann Computer-Graphik 1 – WS 10/11 Clipping 12