Projektionen

Orthografische Projektion:

- Parallele Strahlen von Objekten zur Bildfläche
  - Größen und Winkel aller Objekte bleiben erhalten
- Ausschnitt aus der Szene durch *Clipping Planes.*

$$\mathbf{P}_{ortho} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Perspektivische Projektion:

- Konvergierende Strahlen von allen sichtbaren Objekten zum Augpunkt
- Ausschnitt aus der Szene durch einen Kegelstumpf, definiert durch *Clipping planes*

$$\mathbf{P}_{persp.} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 + \frac{far}{near} & far \\ 0 & 0 & -\frac{1}{near} & 0 \end{pmatrix}$$

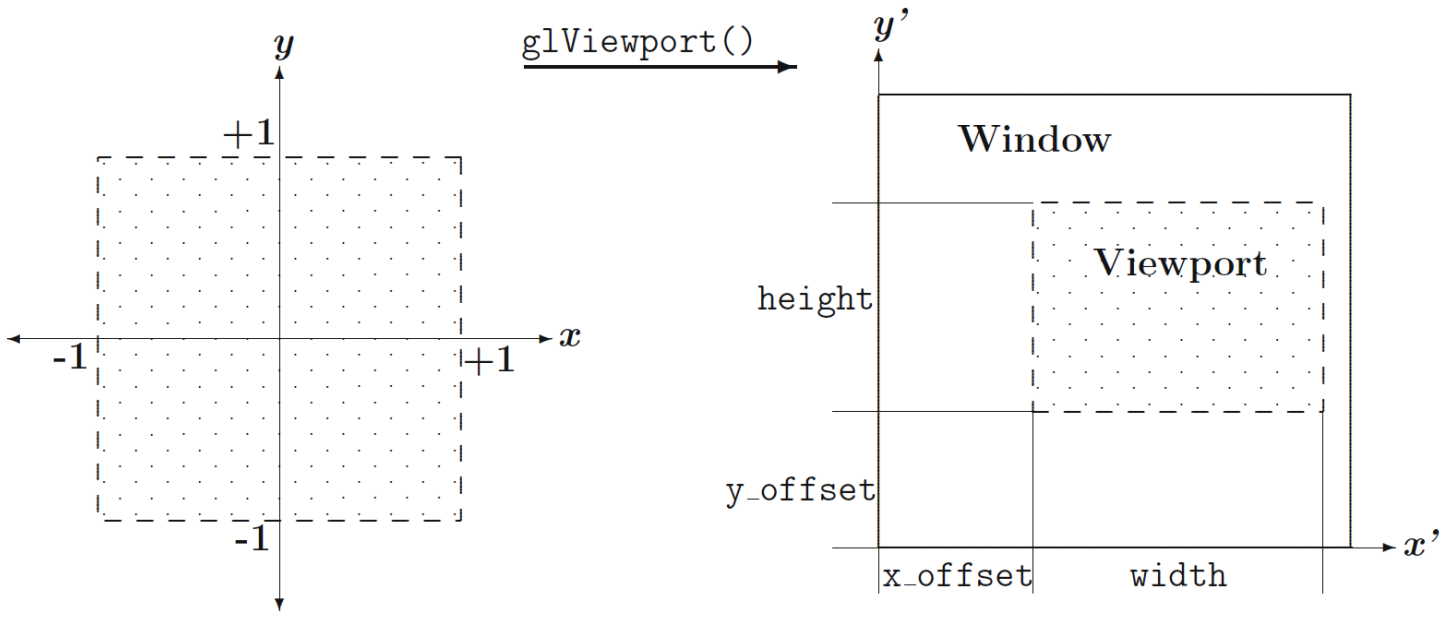
Normierung

- Abbildung der *near clipping plane* auf Bildschirmfenster: Zwischenschritt Normierung
  - → Werte auf das Intervall $[-1\ 1]$ abbilden.

$$\mathbf{N} = \begin{pmatrix} \frac{2}{right-left} & 0 & 0 & -\frac{right+left}{right-left} \\ 0 & \frac{2}{top-bottom} & 0 & -\frac{top+bottom}{top-bottom} \\ 0 & 0 & \frac{-2}{far-near} & -\frac{far+near}{far-near} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

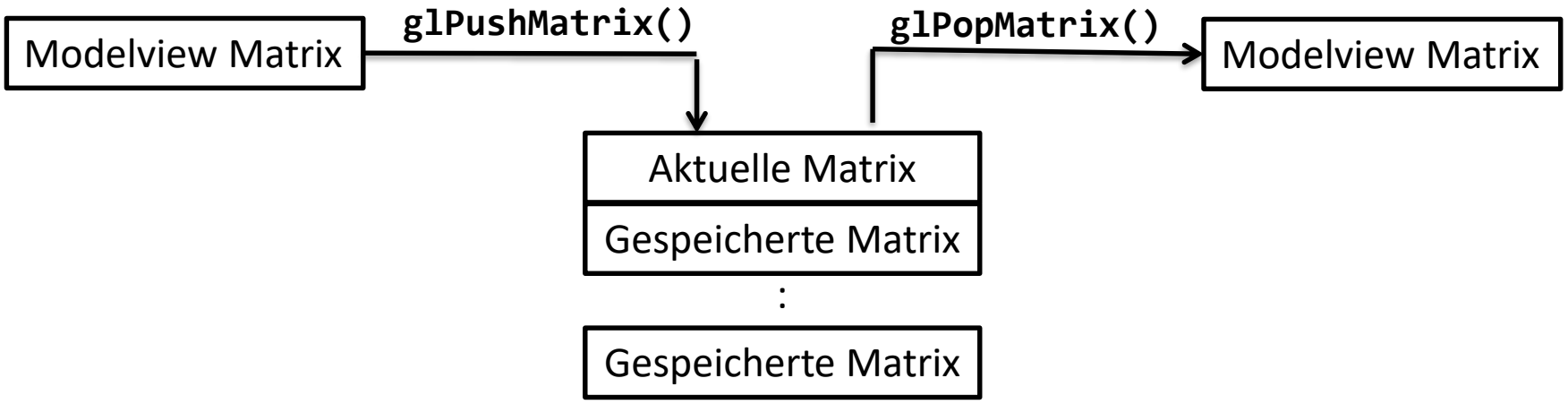
Viewport-Transformation

- Abbildung der Szene auf Ausschnitt des Bildschirms (*Viewport*)
- Viewport definiert in Pixeln
  - Startpunkt (x_{Offset}, y_{Offset})
  - Ausdehnung (width, height)



A. Nischwitz et al. „Computergrafik und Bildverarbeitung“, Band 1: Computergrafik“

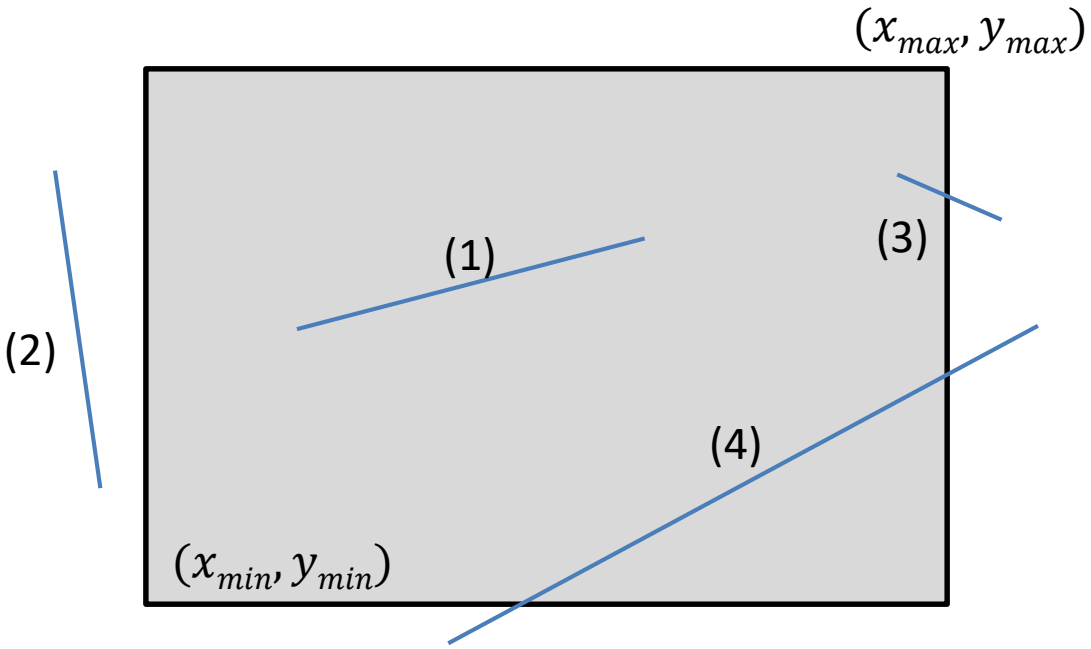
Matrizenstapel



- **`glPushMatrix:`**
  - Anfertigen einer Kopie der aktuellen Matrix + auf dem Stapel speichern.
  - Weitere Transformationen können hinzugefügt werden (Multiplikation auf bisher aktuelle Matrix).
- **`glPopMatrix:`**
  - Entfernt aktuelle Matrix und kehrt zur letzten auf dem Stapel zurück.

Clipping

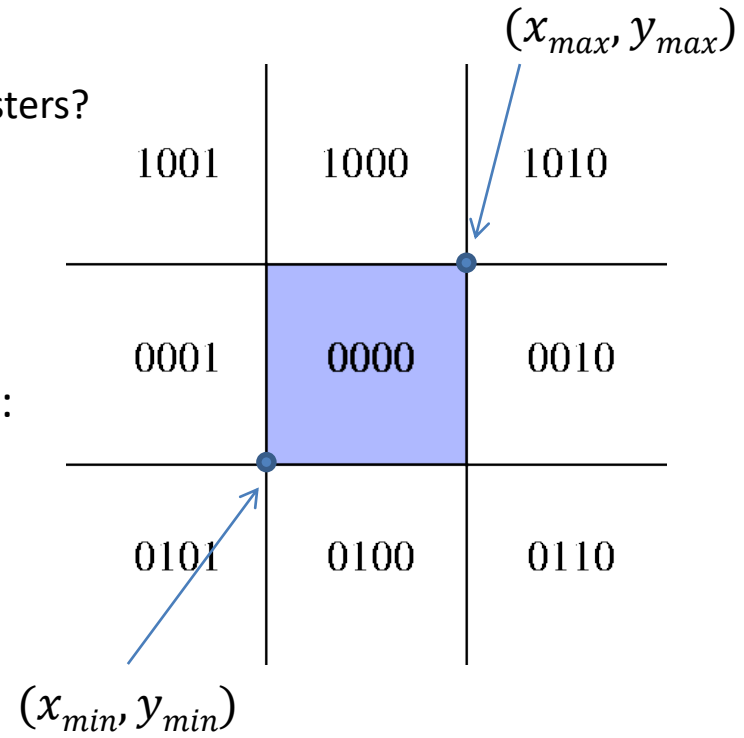
- Clipping = Zuschneiden von Objekten an einem vorgegebenen Bereich.
- Vertex-Clipping:
  - Vertex innerhalb des Darstellungsbereichs wenn $x_{min} \leq x \leq x_{max}$ und $y_{min} \leq y \leq y_{max}$
- Linien-Clipping: 4 mögliche Lagebeziehungen der Vertices



Clipping von Linien

- Algorithmus von Cohen und Sutherland:
  - Zuordnung eines binären Bereichscodes zu jedem Anfangs- und Endpunkt
  - Beide Punkte innerhalb des Fensters?
    - Ja, wenn bitweise ODER-Verknüpfung von v_1 und $v_2 = 0$
  - Beide Punkte und gesamte Linie außerhalb des Fensters?
    - Ja, wenn bitweise UND-Verknüpfung von v_1 und v_2 an einer Stelle ungleich 0
  - Wenn beide Tests nicht erfolgreich sind: weitere Schnittpunkt-Test erforderlich
- Bestimmung Schnittpunkt mit Darstellungsbereich:

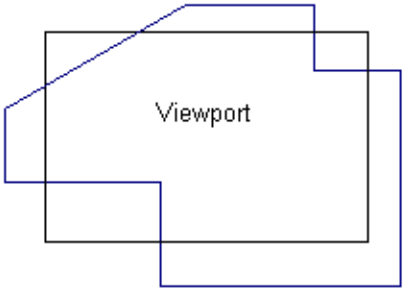
$$s_x = v_{0,x} + \frac{s_y - v_{0,y}}{v_{1,y} - v_{0,y}} (v_{1,x} - v_{0,x})$$

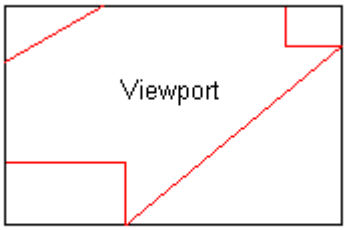


http://www-lehre.inf.uos.de/~rkunze/flashweather/Diplomarbeit/node37.html

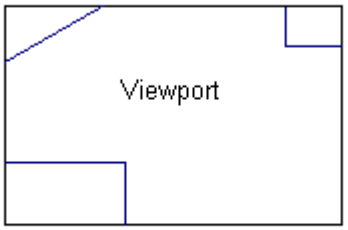
Clipping von Polygonen

- Linien-Clipping-Verfahren kann bei Anwendung auf Polygone zu falschen Ergebnissen führen





falsch



richtig

- Algorithmus von Sutherland und Hodgman:
  - Interpretation der Kanten des Darstellungsbereichs als Geraden ohne Begrenzung
  - Schrittweise Schneiden aller Kanten des Polygons an jeweils einer Geraden des Darstellungsbereichs

  - Entscheidung welche Vertices in Ausgabepolygon übernommen werden:
    (1) Beide Vertices der Kante außerhalb: keinen Vertex in Ausgabepolygon übernehmen
    (2) Gerichtete Kante von v_1 zu v_2 von außen nach innen: Schnittpunkt und v_2 übernehmen
    (3) Beide Vertices der Kante innerhalb: beide Vertices in das Ausgabepolygon übernehmen
    (4) Gerichtete Kante von v_1 zu v_2 von innen nach außen: Schnittpunkt und v_1 übernehmen

http://www.schulportal.bremerhaven.de/wypior/GraphDV/GrafDVTeil2.html

Midpoint Line Algorithmus

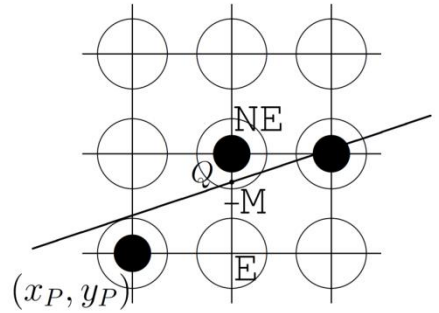
- Ablauf:
  - X-Koordinate wird schrittweise um 1 erhöht
  - Für zugehörige y-Koordinate wird festgestellt, ob sie gleich bleibt oder um 1 erhöht wird
- Implizite Formulierung der Linie:

$$F(x,y) = \Delta y \cdot x - \Delta x \cdot y + B \cdot \Delta x = 0$$

mit $\Delta y = y_1 - y_0, \Delta x = x_1 - x_0$

- Berechnet wird nun der Punkt *M* (Midpoint) als

$$F(M) = F\left(x_p + 1, y_p + \tfrac{1}{2}\right) = d$$



- Das Vorzeichen von F(M) = *d* entscheidet nun darüber ob y-Wert inkrementiert wird oder gleich bleibt:
  - $d > 0$: wähle NE
  - $d \leq 0$: wähle E
- Inkrementelle Berechnung der Entscheidungsvariable d_{new} aus der alten (konstanter Wert- kann einmal vorberechnet werden):
  - Wenn E gewählt wurde: $d_{new} = d_{old} + \Delta y$
  - Wenn NE gewählt wurde: $d_{new} = d_{old} + (\Delta y - \Delta x)$

Midpoint Line Algorithmus

- Wie kommt man auf die konstanten Inkremente?
  - Wenn E gewählt wurde:

    Betrachte nächsten Punkt:

    $$d_{new} = F\left(x_p + 2, y_p + \tfrac{1}{2}\right) = \Delta y\left(x_p + 2\right) - \Delta x\left(y_p + \tfrac{1}{2}\right) + B\Delta x$$
    $$d_{old} = F\left(x_p + 1, y_p + \tfrac{1}{2}\right) = \Delta y\left(x_p + 1\right) - \Delta x\left(y_p + \tfrac{1}{2}\right) + B\Delta x$$

    Subtraktion $d_{new} - d_{old}$ ergibt: $d_{new} = d_{old} + \Delta y$.

Erinnerung:
$F(x, y) = \Delta y \cdot x - \Delta x \cdot y + B \cdot \Delta x = 0$

  - Wenn NE gewählt wurde:

    Betrachte nächsten Punkt:

    $$d_{new} = F\left(x_p + 2, y_p + \tfrac{3}{2}\right) = \Delta y\left(x_p + 2\right) - \Delta x\left(y_p + \tfrac{3}{2}\right) + B\Delta x$$
    $$d_{old} = F\left(x_p + 1, y_p + \tfrac{1}{2}\right) = \Delta y\left(x_p + 1\right) - \Delta x\left(y_p + \tfrac{1}{2}\right) + B\Delta x$$

    Subtraktion $d_{new} - d_{old}$ ergibt: $d_{new} = d_{old} + (\Delta y - \Delta x)$.