Normal Estimation and Clipping Methods
Normal Vectors

• How to compute the normal per planar face?

\[ n(T) = \frac{(x_j - x_i) \times (x_k - x_i)}{\| (x_j - x_i) \times (x_k - x_i) \|} \]
Normal Vectors

• However, vertex normals are not well defined

• Nevertheless, assume the shape is smooth

$$n(v) = \frac{\sum_{T \in \mathcal{N}_1(v)} \omega_T n(T)}{\left\| \sum_{T \in \mathcal{N}_1(v)} \omega_T n(T) \right\|}$$

• How to choose $\omega_T$? E.g., $\omega_T = 1, |T|, \theta_T$
Normal Vectors

• How to **transform** normal vectors?

• Using the transformation matrix won’t work!
Normal Vectors

• How to transform normal vectors?

• The following should hold

\[ n \cdot t = 0 = (Gn) \cdot (Mt) \]

• Thus, \( G^T M = I \)  \( \Rightarrow \)  \( G = (M^{-1})^T \)
Clipping

- When and how to perform clipping?

- **Wireframe** rendering:
  - Clipping can be done in screen coordinates
  - Cull/Draw the triangle if it is outside/inside
  - Border triangles:
    - Line-Segment clipping or
    - Ignore pixels
Line-Segment Clipping

• Given two endpoints \((x_1, y_1)^T\) and \((x_2, y_2)^T\)

• Line segment in **parametric form**: \(\alpha \in [0,1]\)

\[
x(\alpha) = (1 - \alpha)x_1 + \alpha x_2, \\
y(\alpha) = (1 - \alpha)y_1 + \alpha y_2.
\]
Line-Segment Clipping

- Liang–Barsky [TOG ’84] Clipping

- Defer computation of line intersection as much as possible

\[
x(\alpha) = (1 - \alpha)x_1 + \alpha x_2, \quad y(\alpha) = (1 - \alpha)y_1 + \alpha y_2.
\]
Liang–Barsky Clipping

- A point is in the clip window if

\[
\begin{align*}
x_{\text{min}} & \leq x_1 + \alpha \Delta x \leq x_{\text{max}}, \\
y_{\text{min}} & \leq y_1 + \alpha \Delta y \leq y_{\text{max}}.
\end{align*}
\]

\[
x(\alpha) = x_1 + \alpha(x_2 - x_1), \\
y(\alpha) = y_1 + \alpha(y_2 - y_1).
\]
Liang–Barsky Clipping

- A point is in the clip window if

  \[ \alpha p_k \leq q_k, \quad k = 1,2,3,4 \]

- Where

  \[ p_2 = -\Delta x, \quad q_2 = x_1 - x_{\min} \]
  \[ p_4 = \Delta x, \quad q_4 = x_{\max} - x_1 \]
  \[ p_1 = -\Delta y, \quad q_1 = y_1 - y_{\min} \]
  \[ p_3 = \Delta y, \quad q_3 = y_{\max} - y_1 \]

  \[ x(\alpha) = x_1 + \alpha(x_2 - x_1), \]
  \[ y(\alpha) = y_1 + \alpha(y_2 - y_1). \]
Liang–Barsky Clipping

• Determine the final line segment

  – Find $k$ for which $p_k < 0$, then $\alpha_2 = \max\left\{0, \frac{q_k}{p_k}\right\}$.

  – Find $k$ for which $p_k > 0$, then $\alpha_4 = \min\left\{1, \frac{q_k}{p_k}\right\}$.

  – Similarly for $\alpha_1$ and $\alpha_3$
Polygon Clipping

• Follows directly from line-clipping
• Apply clipping to the edges of the polygon
• Clipping a convex polygon against a convex polygon results in at most one convex object
Polygon Clipping

• We can implement a clipping pipeline

• Each clipper clips against one window’s edge
Polygon Clipping

• Example of pipeline clipping
Clipping

• When and how to perform clipping?

• Scan Conversion
  – Perform clipping before perspective division
  – The visible volume

\[
\begin{align*}
w & \geq x \geq -w, \\
w & \geq y \geq -w, \\
w & \geq z \geq -w.
\end{align*}
\]
Clipping

• For each model
  – Determine if it is inside/outside the view volume
    • Intersect its bounding box with the view volume
  – For each boundary model
    • Perform polygon-clipping for each triangle

• Thus, extend our clipping to three dimensions
Bounding Box

• How to compute the bounding box?

• Find the minimal/maximal coordinates
3D Polygon Clipping

- Extend Liang–Barsky with the equation

\[ z(\alpha) = (1 - \alpha)z_1 + \alpha z_2 \]

- Add near and far clippers in the pipeline

- Need to adjust intersection computations...
3D Polygon Clipping

- Plane–Line intersection is given by

\[ \alpha = \frac{n \cdot (p_0 - p_1)}{n \cdot (p_2 - p_1)} \]
Suggested Readings

• Transforming Normals:

• Interactive Computer Graphics, Chapter 6

• Liang–Barsky Clipping:
  – http://en.wikipedia.org/wiki/Liang%E2%80%93Barsky_algorithm