Clipping and Rasterization

Prof. Vladlen Koltun
Computer Science Department
Stanford University
Clipping

- Cull parts of the scene that are outside the view frustum
- Done on a per-primitive basis
- Main motivation: efficiency
- This lecture covers clipping in 2D. The simple extension to 3D is in Angel, Section 7
Line Segment Clipping

- Not completely trivial. Even if both endpoints are outside view window, part of the line segment could be inside.
- In the inner loop. Must be efficient down to the instruction level.
Given: line segment \( p(\alpha) = (1 - \alpha)p_1 + \alpha p_2 \) and window \( W \).
• Intersect line spanned by \( p \) with the four axis-aligned lines that bound \( W \).
Line Segment Clipping

\( p(\alpha(x_1)) \)

\( p(\alpha(x_2)) \)

\( p(\alpha(y_1)) \)

\( p(\alpha(y_2)) \)

\( (x_1, y_1) \)

\( (x_2, y_2) \)

\[ \alpha(x_1) = \frac{x_1 - x(p_1)}{x(p_2) - x(p_1)} \]

\[ \alpha(x_2) = \frac{x_2 - x(p_1)}{x(p_2) - x(p_1)} \]

\[ \alpha(y_1) = \frac{y_1 - y(p_1)}{y(p_2) - y(p_1)} \]

\[ \alpha(y_2) = \frac{y_2 - y(p_1)}{y(p_2) - y(p_1)} \]
Line Segment Clipping

- Order the parameters by value. If the order is XXYY (or YYXX) then the whole line is outside the clipping window.
- Otherwise, the part bounded by the inner two parameter values is the intersection of the line with W.
- Take the intersection of that part with the range (0,1).
- Special case: axis-parallel segments.
Polygon Clipping

- Can produce a more complicated polygon that includes edges and vertices from the clipping rectangle.
- Begin with line segment clipping, then walk along clipping rectangle.
Rasterization
Line segment rasterization
Line segment rasterization

Digital Differential Analyzer

\[ m = \frac{y_2 - y_1}{x_2 - x_1}; \]
\[ y = y_1; \]
\[ \text{for} \ (x = x_1; \ x \leq x_2; \ x++ \) \{ \]
\[ \text{fill} (x, \text{round}(y)); \]
\[ y = y + m; \]
\[ \} \]
Line segment rasterization

Digital Differential Analyzer

\[
m = \frac{y_2 - y_1}{x_2 - x_1};
\]
\[
y = y_1;
\]
\[
\text{for } (x = x_1; x \leq x_2; x++) \{
\]
\[
\quad \text{fill}(x, \text{round}(y));
\]
\[
\quad y = y + m;
\]
\[
\}
\]
if \((y_2 - y_1 \leq x_2 - x_1)\) {
  \[m = \frac{y_2 - y_1}{x_2 - x_1}\]
  \[y = y_1;\]
  for\((x = x_1; x \leq x_2; x + +)\) {
    fill\((x, \text{round}(y))\);
    \[y = y + m;\]
  }
} \\
else {
  \[m = \frac{x_2 - x_1}{y_2 - y_1}\]
  \[x = x_1;\]
  for\((y = y_1; y \leq y_2; y + +)\) {
    fill\((\text{round}(x), y)\);
    \[x = x + m;\]
  }
}
Line segment rasterization

Bresenham’s Algorithm

- Rasterization is in the inner loop. Performance is critical. DDA performs a flop in every step.

- Assume \( x_1, y_1, x_2, y_2 \) are integers. Also assume \( x_1 \leq y_1, y_1 \leq y_2, \) and \( y_2 - y_1 \leq x_2 - x_1 \)

- Define \( D_x = x_2 - x_1 \) and \( D_y = y_2 - y_1 \)

- The line equation is \( y = \frac{D_y}{D_x}(x - x_1) + y_1 \)

- Observation: At every step, we either increment \( y \) or not. Binary decision.

- Bresenham’s algorithm maintains \( Y = y \times D_x \) and increments it by \( D_y \) in every step.
Line segment rasterization

Bresenham’s Algorithm

- Rasterization is in the inner loop. Performance is critical. DDA performs a flop in every step.

- Assume \( x_1, y_1, x_2, y_2 \) are integers. Also assume \( x_1 \leq y_1, y_1 \leq y_2, \) and \( y_2 - y_1 \leq x_2 - x_1 \)

- Define \( D_x = x_2 - x_1 \) and \( D_y = y_2 - y_1 \)

- The line equation is \( y = \frac{D_y}{D_x} (x - x_1) + y_1 \)

- Observation: At every step, we either increment \( y \) or not. Binary decision.

- Bresenham’s algorithm maintains \( Y = y \times D_x \) and increments it by \( D_y \) in every step.
Line segment rasterization

Bresenham’s Algorithm

\[
Y = 0; \\
\text{for } (x=x_1; x \leq x_2; x++) \\
\quad \text{if } (Y \leq D_x - Y) \\
\quad \quad \text{fill}(x,y); \\
\quad \text{else} \\
\quad \quad y++; \\
\quad \quad \text{fill}(x,y); \\
\quad \quad Y = Y - D_x; \\
\quad \} \\
\} \\
Y = Y + D_y;
\]
Polygon rasterization

\[ y = y_{\text{first}}; \]
initialize-left-segment();
initialize-right-segment();
while \((y \leq y_{\text{last}})\) {
    if \((y == \text{end-of-left-segment})\)
        switch-to-next-left-segment();
    if \((y == \text{end-of-right-segment})\)
        switch-to-next-right-segment();
    \(x_{\text{left}} = \text{evaluate-left-segment}()\);
    \(x_{\text{right}} = \text{evaluate-right-segment}()\);
    for \((x = x_{\text{left}}; x \leq x_{\text{right}}; x++)\)
        fill \((x,y)\);
    \(y++;\)
}
Interpolating color and depth
Perspective-Correct Interpolation

Review session
Hidden-surface removal
Painter’s algorithm

- Sort the polygons by depth, draw back to front
- Generally impossible (and incompatible with pipeline)
z-buffering