Clipping

Greg Humphreys
University of Virginia
CS 445, Fall 2003

3D Rendering Pipeline (for direct illumination)

- 3D Primitives
  - Modeling
    - 3D Modeling Coordinates
    - Lighting
    - Viewport
    - Transformations
    - 3D Camera Coordinates
    - Clipping
    - 2D Screen Coordinates
    - Scan Conversion
    - 2D Image Coordinates
    - Image

2D Rendering Pipeline

- 3D Primitives
  - 2D Primitives
    - Clip portions of geometric primitives residing outside the window
    - Transform the clipped primitives from screen to image coordinates
    - Fill pixels representing primitives in screen coordinates
  - Viewport
  - Transformation
  - Scan Conversion
  - Image
2D Rendering Pipeline

- 3D Primitives
  - Clipping
    - Clip portions of geometric primitives residing outside the window
  - Viewport Transformation
    - Transform the clipped primitives from screen to image coordinates
  - Scan Conversion
    - Fill pixels representing primitives in screen coordinates
  - Image

Clipping

- Avoid drawing parts of primitives outside window
  - Window defines part of scene being viewed
  - Must draw geometric primitives only inside window

Clipping

- Avoid drawing parts of primitives outside window
  - Points
  - Lines
  - Polygons
  - Circles
  - etc.
Point Clipping

- Is point \((x, y)\) inside the clip window?

\[
\text{inside} = (x \geq w_1) \land (x \leq w_2) \land (y \geq w_1) \land (y \leq w_2);
\]

Window

Line Clipping

- Find the part of a line inside the clip window

Before Clipping

Line Clipping

- Find the part of a line inside the clip window

After Clipping

Cohen Sutherland Line Clipping

- Use simple tests to classify easy cases first

Before Clipping

After Clipping
Cohen Sutherland Line Clipping
• Classify some lines quickly by AND of bit codes representing regions of two endpoints (must be 0)

<table>
<thead>
<tr>
<th>Bit 1</th>
<th>Bit 2</th>
<th>Bit 3</th>
<th>Bit 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1001</td>
<td>P0001</td>
<td>0011</td>
<td>0101</td>
</tr>
<tr>
<td>1000</td>
<td>0000</td>
<td>0100</td>
<td></td>
</tr>
<tr>
<td>1000</td>
<td>0001</td>
<td>0100</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0110</td>
<td></td>
</tr>
</tbody>
</table>

Bit 1 Bit 2
Cohen-Sutherland Line Clipping

• Compute intersections with window boundary for lines that can’t be classified quickly

Bit 1 Bit 2
0000 01001000
0001 01011001
0010 01101010

P10 P9 P8
P7 P4 P3
P6 P5

Bit 3
Cohen-Sutherland Line Clipping
- Compute intersections with window boundary for lines that can't be classified quickly

<table>
<thead>
<tr>
<th>Bit 1</th>
<th>Bit 2</th>
<th>Bit 3</th>
<th>Bit 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>0100</td>
<td>1000</td>
<td>1001</td>
</tr>
<tr>
<td>0001</td>
<td>0100</td>
<td>1000</td>
<td>1001</td>
</tr>
<tr>
<td>0010</td>
<td>0010</td>
<td>0100</td>
<td>1001</td>
</tr>
<tr>
<td>0010</td>
<td>0010</td>
<td>0100</td>
<td>1001</td>
</tr>
<tr>
<td>0010</td>
<td>0010</td>
<td>0100</td>
<td>1001</td>
</tr>
</tbody>
</table>

Cohen-Sutherland Line Clipping
- Compute intersections with window boundary for lines that can't be classified quickly

<table>
<thead>
<tr>
<th>Bit 1</th>
<th>Bit 2</th>
<th>Bit 3</th>
<th>Bit 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>0100</td>
<td>1000</td>
<td>1001</td>
</tr>
<tr>
<td>0001</td>
<td>0100</td>
<td>1000</td>
<td>1001</td>
</tr>
<tr>
<td>0010</td>
<td>0010</td>
<td>0100</td>
<td>1001</td>
</tr>
<tr>
<td>0010</td>
<td>0010</td>
<td>0100</td>
<td>1001</td>
</tr>
<tr>
<td>0010</td>
<td>0010</td>
<td>0100</td>
<td>1001</td>
</tr>
</tbody>
</table>
Cohen-Sutherland Line Clipping

- Compute intersections with window boundary for lines that can’t be classified quickly

```
<table>
<thead>
<tr>
<th>Bit 1</th>
<th>Bit 2</th>
<th>Bit 3</th>
<th>Bit 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1001</td>
<td>0001</td>
<td>0000</td>
<td>P_4</td>
</tr>
<tr>
<td>1000</td>
<td>0000</td>
<td>P_3</td>
<td>P_7</td>
</tr>
<tr>
<td>1010</td>
<td>0010</td>
<td>P_6</td>
<td>P_8</td>
</tr>
</tbody>
</table>
```

Cohen-Sutherland Line Clipping

- Compute intersections with window boundary for lines that can’t be classified quickly

```
<table>
<thead>
<tr>
<th>Bit 1</th>
<th>Bit 2</th>
<th>Bit 3</th>
<th>Bit 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1001</td>
<td>0001</td>
<td>0000</td>
<td>P_4</td>
</tr>
<tr>
<td>1000</td>
<td>0000</td>
<td>P_3</td>
<td>P_7</td>
</tr>
<tr>
<td>1010</td>
<td>0010</td>
<td>P_6</td>
<td>P_8</td>
</tr>
</tbody>
</table>
```
Cohen-Sutherland Line Clipping

- Compute intersections with window boundary for lines that can’t be classified quickly

Clipping

- Avoid drawing parts of primitives outside window
  - Points
  - Lines
  - Polygons
  - Circles
  - etc.

2D Screen Coordinates

Window
Polygon Clipping
- Find the part of a polygon inside the clip window?

Before Clipping

After Clipping

Sutherland Hodgeman Clipping
- Clip to each window boundary one at a time
Sutherland Hodgeman Clipping

- Clip to each window boundary one at a time

Clipping to a Boundary

- Do inside test for each point in sequence,
  Insert new points when cross window boundary,
  Remove points outside window boundary

Outside

Inside

Window Boundary

P₁

P₂

P₃

P₄

P₅

P₆
Clipping to a Boundary

- Do inside test for each point in sequence,
  Insert new points when cross window boundary,
  Remove points outside window boundary
Clipping to a Boundary

- Do inside test for each point in sequence,
- Insert new points when cross window boundary,
- Remove points outside window boundary

---

Clipping to a Boundary

- Do inside test for each point in sequence,
- Insert new points when cross window boundary,
- Remove points outside window boundary

---

Clipping to a Boundary

- Do inside test for each point in sequence,
- Insert new points when cross window boundary,
- Remove points outside window boundary

---

Clipping to a Boundary

- Do inside test for each point in sequence,
- Insert new points when cross window boundary,
- Remove points outside window boundary
2D Rendering Pipeline

- 3D Primitives → 2D Primitives
- Clipping
  - Clip portions of geometric primitives residing outside the window
- Viewport Transformation
  - Transform the clipped primitives from screen to image coordinates
- Scan Conversion
  - Fill pixels representing primitives in screen coordinates
- Image

Viewport Transformation

- Transform 2D geometric primitives from screen coordinate system (normalized device coordinates) to image coordinate system (pixels)

Viewport Transformation

- Window-to-viewport mapping

Viewport

\[
\begin{align*}
vx &= vx_1 + (wx - wx_1) \cdot (vx_2 - vx_1) / (wx_2 - wx_1) \\
vy &= vy_1 + (wy - wy_1) \cdot (vy_2 - vy_1) / (wy_2 - wy_1)
\end{align*}
\]

Summary of Transformations

- Modeling transformation
- Viewing transformations
- Viewport transformation

\[ p(x,y,z) \]
**Summary**

3D Primitives

2D Primitives

- Clipping
  - Clip portions of geometric primitives residing outside the window

- Viewport Transformation
  - Transform the clipped primitives from screen to image coordinates

- Scan Conversion
  - Fill pixels representing primitives in screen coordinates

**3D Primitives**

- Modeling
- Transformation
- Projection
- Transformation
- Clipping
- Lighting
- Image

**2D Primitives**

- Viewport Transformation
- Scan Conversion
- Image

**Next Time**

3D Primitives

- 3D Modeling Coordinates
- Lighting
- Viewport Transformation
- Projection Transformation
- Clipping
- Image

2D Primitives

- 2D Screen Coordinates
- Scan Conversion
- Image

**Notes:**

- Image Clip portions of geometric primitives residing outside the window
- Fill pixels representing primitives in screen coordinates