Clipping

Adam Finkelstein
Princeton University
COS 426, Spring 2003

3D Rendering Pipeline (for direct illumination)

3D Primitives  
3D Modeling Coordinates

Modeling Transformation

3D World Coordinates

Lighting

Viewing Transformation

3D Camera Coordinates

Projection Transformation

2D Screen Coordinates

Clipping

2D Screen Coordinates

Viewport Transformation

2D Image Coordinates

Scan Conversion

2D Image Coordinates

Image

3D Primitives

3D Modeling Coordinates

Modeling Transformation

3D World Coordinates

Lighting

Viewing Transformation

3D Camera Coordinates

Projection Transformation

2D Screen Coordinates

Clipping

2D Screen Coordinates

Viewport Transformation

2D Image Coordinates

Scan Conversion

2D Image Coordinates

Image

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

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

Clipping

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

Window

Screen Coordinates
Clipping

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

Point Clipping

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

\[
\text{inside} = \begin{cases} 
(x \geq wx1) \land (x \leq wx2) \land (y \geq wy1) \land (y \leq wy2), 
\end{cases}
\]

Line Clipping

- Find the part of a line inside the clip window

Cohen Sutherland Line Clipping

- Use simple tests to classify easy cases first
Cohen Sutherland Line Clipping

• Classify some lines quickly by AND of bit codes representing regions of two endpoints (must be 0)

  Bit 1 Bit 2
  0000 01001000
  0001 01011001
  0010 01101010

P1 P10
P9
P8
P7
P4 P3
P6
P5
P2

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

P1 P10
P9
P8
P7
P4 P3
P6
P5
P2
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>0101</td>
</tr>
<tr>
<td>0001</td>
<td>0101</td>
<td>1000</td>
<td>0100</td>
</tr>
<tr>
<td>0010</td>
<td>0110</td>
<td>1000</td>
<td>0100</td>
</tr>
</tbody>
</table>

P10 P9 P8 P7 P6 P4 P3 P5

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>0101</td>
</tr>
<tr>
<td>0001</td>
<td>0101</td>
<td>1000</td>
<td>0100</td>
</tr>
<tr>
<td>0010</td>
<td>0110</td>
<td>1000</td>
<td>0100</td>
</tr>
</tbody>
</table>

P10 P9 P8 P7 P6 P4 P3 P5

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>0101</td>
</tr>
<tr>
<td>0001</td>
<td>0101</td>
<td>1000</td>
<td>0100</td>
</tr>
<tr>
<td>0010</td>
<td>0110</td>
<td>1000</td>
<td>0100</td>
</tr>
</tbody>
</table>

P10 P9 P8 P7 P6 P4 P3 P5

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>0101</td>
</tr>
<tr>
<td>0001</td>
<td>0101</td>
<td>1000</td>
<td>0100</td>
</tr>
<tr>
<td>0010</td>
<td>0110</td>
<td>1000</td>
<td>0100</td>
</tr>
</tbody>
</table>

P10 P9 P8 P7 P6 P4 P3 P5

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>0101</td>
</tr>
<tr>
<td>0001</td>
<td>0101</td>
<td>1000</td>
<td>0100</td>
</tr>
<tr>
<td>0010</td>
<td>0110</td>
<td>1000</td>
<td>0100</td>
</tr>
</tbody>
</table>

P10 P9 P8 P7 P6 P4 P3 P5

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>0101</td>
</tr>
<tr>
<td>0001</td>
<td>0101</td>
<td>1000</td>
<td>0100</td>
</tr>
<tr>
<td>0010</td>
<td>0110</td>
<td>1000</td>
<td>0100</td>
</tr>
</tbody>
</table>

P10 P9 P8 P7 P6 P4 P3 P5

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>0101</td>
</tr>
<tr>
<td>0001</td>
<td>0101</td>
<td>1000</td>
<td>0100</td>
</tr>
<tr>
<td>0010</td>
<td>0110</td>
<td>1000</td>
<td>0100</td>
</tr>
</tbody>
</table>

P10 P9 P8 P7 P6 P4 P3 P5

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>0101</td>
</tr>
<tr>
<td>0001</td>
<td>0101</td>
<td>1000</td>
<td>0100</td>
</tr>
<tr>
<td>0010</td>
<td>0110</td>
<td>1000</td>
<td>0100</td>
</tr>
</tbody>
</table>

P10 P9 P8 P7 P6 P4 P3 P5

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>0101</td>
</tr>
<tr>
<td>0001</td>
<td>0101</td>
<td>1000</td>
<td>0100</td>
</tr>
<tr>
<td>0010</td>
<td>0110</td>
<td>1000</td>
<td>0100</td>
</tr>
</tbody>
</table>

P10 P9 P8 P7 P6 P4 P3 P5
Cohen-Sutherland Line Clipping

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

Bit 1 Bit 2

<table>
<thead>
<tr>
<th>Bit 3</th>
<th>Bit 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>0100</td>
</tr>
<tr>
<td>0001</td>
<td>0101</td>
</tr>
<tr>
<td>0010</td>
<td>0110</td>
</tr>
</tbody>
</table>

P10  P9  P8  P7  P6  P5  P4  P3  P2  P1
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></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Clipping

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

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
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
    - 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

**Viewport Transformation**

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

**Viewport Transformation**

- Window-to-viewport mapping

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

**Summary of Transformations**

- **Modeling transformation**
- **Viewing transformations**
- **Projection Transformation**
- **Viewport transformation**

- **2D Image Coordinates**
- **3D Modeling Coordinates**
- **3D Object Coordinates**

**Summary**

- **3D Primitives**
  - 2D Primitives
    - 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
Next Time

3D Primitives
Modeling Transformations
3D World Coordinates
Lighting
Modeling Transformations
3D World Coordinates
Viewing Transformations
Camera Coordinates
Projection Transformations
Camera Coordinates
Clipping
Viewport Transformations
Viewport Coordinates
Scan Conversion
Image

Scan Conversion!