Clipping

Thomas Funkhouser
Princeton University
C0S 426, Fall 2000

3D Rendering Pipeline (for direct illumination)
3D Rendering Pipeline (for direct illumination)

3D Primitives
↓
3D Modeling Coordinates
↓
Modeling Transformation
↓
3D World Coordinates
↓
Lighting
↓
3D World Coordinates
↓
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 Rendering Pipeline

3D Primitives
  ↓
2D Primitives
  ↓
Clipping
  ↓
Viewport Transformation
  ↓
Scan Conversion
  ↓
Image

- 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

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

Clipping

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

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

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

Line Clipping

- Find the part of a line inside the clip window

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
Cohen Sutherland Line Clipping

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

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

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

- Compute intersections with window boundary for lines that can’t be classified quickly
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>1010</td>
<td>1000</td>
<td>1001</td>
<td>0101</td>
</tr>
<tr>
<td>P_9</td>
<td>P_3</td>
<td>P_7</td>
<td>0001</td>
</tr>
<tr>
<td>P_6</td>
<td>P_4</td>
<td>P_8</td>
<td>0100</td>
</tr>
<tr>
<td>P_5</td>
<td></td>
<td>P_10</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></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>1010</td>
<td>1000</td>
<td>1001</td>
<td>0101</td>
</tr>
<tr>
<td>P_9</td>
<td>P_3</td>
<td>P_7</td>
<td>0001</td>
</tr>
<tr>
<td>P_6</td>
<td>P_4</td>
<td>P_8</td>
<td>0100</td>
</tr>
<tr>
<td>P_5</td>
<td></td>
<td>P_10</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
Cohen-Sutherland Line Clipping

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

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

---

1001 | 0001 | 0101
---|---|---

Bit 4

1000 | 0000 | 0100
---|---|---

Bit 3

1010 | 0010 | 0110
---|---|---

Bit 2

1010 | 0010 | 0110
---|---|---

Bit 2

---

P3 | P4 | P7
---|---|---

P5 | P6 | P9
---|---|---

P8 | P10
---|---

---

Cohen-Sutherland Line Clipping

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

---

1001 | 0001 | 0101
---|---|---

Bit 4

1000 | 0000 | 0100
---|---|---

Bit 3

1010 | 0010 | 0110
---|---|---

Bit 2

---

P3 | P4 | P7
---|---|---

P5 | P6 | P9
---|---|---

P8 | P10
---|---

---
Cohen-Sutherland Line Clipping

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

```
Bit 1  | Bit 2  | Bit 3  | Bit 4
1010  | 0010  | 0110  | 0101
1010  | 0110  | 0101  | 0101
1001  | 0001  | P_7   | P_4
1000  | 0000  | P_3   | P_4
1000  | 0100  | P_6   | P_8
1010  | 0110  | P_5   | P_6
```

Cohen-Sutherland Line Clipping

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

```
Bit 1  | Bit 2  | Bit 3  | Bit 4
1010  | 0010  | 0110  | 0101
1010  | 0110  | 0101  | 0101
1001  | 0001  | P_7   | P_4
1000  | 0000  | P_3   | P_4
1000  | 0100  | P_6   | P_8
1010  | 0110  | P_5   | P_6
```
Cohen-Sutherland Line Clipping

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

![Diagram](image-url)

Cohen-Sutherland Line Clipping

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

![Diagram](image-url)
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.
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
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

![Diagram of point clipping]

Clipping to a Boundary

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

![Diagram of point clipping]
Clipping to a Boundary

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

![Diagram of Clipping to a 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

![Diagram of Clipping to a 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

```
vx = vx1 + (wx - wx1) * (vx2 - vx1) / (wx2 - wx1);
vy = vy1 + (wy - wy1) * (vy2 - vy1) / (wy2 - wy1);
```

Summary of Transformations

```
p(x,y,z)  →  3D Object Coordinates
| Modeling Transformation  |
| 3D World Coordinates    |
| Viewing Transformation   |
| 3D Camera Coordinates   |
| Projection Transformation|
| 2D Screen Coordinates   |
| Window-to-Viewport Transformation|
| 2D Image Coordinates   |
| p'(x',y')               |
```
Summary

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

Summary

3D Primitives

Modeling Transformation

3D Modeling Coordinates

Lighting

3D World Coordinates

Viewing Transformation

3D World Coordinates

Projection Transformation

3D Camera Coordinates

Clipping

2D Screen Coordinates

Viewport Transformation

2D Screen Coordinates

Scan Conversion

2D Image Coordinates

Image

Viewing Window
Next Time

3D Primitives
  ↓
Modeling Transformation
  ↓
3D Modeling Coordinates
  ↓
Lighting
  ↓
3D World Coordinates
  ↓
Viewing Transformation
  ↓
3D World Coordinates
  ↓
Projection Transformation
  ↓
3D Camera Coordinates
  ↓
Clipping
  ↓
2D Screen Coordinates
  ↓
Viewport Transformation
  ↓
2D Screen Coordinates
  ↓
Scan Conversion
  ↓
2D Image Coordinates
  ↓
Image

Scan Conversion!