Clipping

Greg Humphreys
CS445: Intro Graphics
University of Virginia, Fall 2004

3D Rendering Pipeline (for direct illumination)

3D Primitives
 Modeling Transformation
 Lighting
 Projection
 Transformation
 Clipping
 Lighting
 Projection
 Transformation
 Viewport Transformation
 Scan Conversion
 Image

2D Rendering Pipeline

3D 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
 Image

Clipping

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

Screen Coordinates
Window
Clipping

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

Viewing Window

Point Clipping

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

\[
\text{inside} = (x \geq wx1) \&\& (x \leq wx2) \&\& (y \geq wy1) \&\& (y \leq wy2);
\]

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

- 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
Cohen-Sutherland Line Clipping

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

\[ P_1 \quad P_2 \quad P_3 \quad P_4 \quad P_5 \quad P_6 \quad P_7 \quad P_8 \quad P_9 \quad P_{10} \]

\[ Bit 1 \quad Bit 2 \quad Bit 3 \quad Bit 4 \]

\[ 0000 \quad 0001 \quad 0010 \quad 0011 \quad 0100 \quad 0101 \quad 0110 \quad 0111 \quad 1000 \quad 1001 \]

\[ P_1 \quad P_2 \quad P_3 \quad P_4 \quad P_5 \quad P_6 \quad P_7 \quad P_8 \quad P_9 \quad P_{10} \]

\[ Bit 1 \quad Bit 2 \quad Bit 3 \quad Bit 4 \]

\[ 0000 \quad 0001 \quad 0010 \quad 0011 \quad 0100 \quad 0101 \quad 0110 \quad 0111 \quad 1000 \quad 1001 \]
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?

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

```
| P5 | P4 | P3 | P2 | P1 |
```

Outside

Inside

Window Boundary

P'

P''
### 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**

### 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 Transformation Diagram](image)

  - \(vx = vx_1 + \frac{(wx - wx_1) \cdot (vx_2 - vx_1)}{(wx_2 - wx_1)}\)
  - \(vy = vy_1 + \frac{(wy - wy_1) \cdot (vy_2 - vy_1)}{(wy_2 - wy_1)}\)

### Summary of Transformations

- \(p(x,y,z)\)

  - **Modeling transformation**
  - **Viewing transformations**
  - **Viewport transformation**

### Summary

- **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**

- 3D Primitives
  - 3D Modeling Coordinates
  - 3D World Coordinates
  - Lighting
  - Viewing Transformation
  - Projection Transformation
  - Clipping
  - 2D Screen Coordinates
  - Viewport Transformation
  - 2D Image Coordinates

- **Viewing Window**
Next Time

3D Primitives
Modeling
Transformation
Lighting
Viewing
Transformation
Projection
Transformation
Clipping
Scan Conversion

2D Image Coordinates
3D Modeling Coordinates
2D World Coordinates
3D Camera Coordinates
2D Screen Coordinates
2D Screen Coordinates
2D Screen Coordinates
3D World Coordinates
3D Camera Coordinates
3D World Coordinates
Scan Conversion!