3D rendering pipeline

3D Primitives

Modeling Transformation

Lighting

Viewing Transformation

Projection Transformation

Clipping

Viewport Transformation

Scan Conversion

Image

3D Modeling Coordinates

3D World Coordinates

3D World Coordinates

3D Camera Coordinates

2D Screen Coordinates

2D Screen Coordinates

2D Image Coordinates

2D Image Coordinates
3D rendering pipeline

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

Clip portions of geometric primitives residing outside the window

2D Primitives

Transform the clipped primitives from screen to image coordinates

Clipping

Fill pixels representing primitives in screen coordinates

Viewport Transformation

Scan Conversion

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
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
  - 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?

\[\begin{align*}
\text{Inside} &= \ (x \geq wx_{min}) \ \text{and} \ \ (x \leq wx_{max}) \ \text{and} \ \ (y \geq wy_{min}) \ \text{and} \ \ (y \leq wy_{max})
\end{align*}\]
Line clipping

- Find the part of a line inside the clip window
Line clipping

- Find the part of a line inside the clip window

After Clipping
Line clipping

- Both points inside – trivially accepted
- Both points outside one edge – trivially reject
- Brute force:
  - Calculate the infinite line-edge intersection
  - Check if the intersection is on the edge/line
Cohen-Sutherland algorithm

- Use simple tests to classify easy cases first
Cohen-Sutherland algorithm

- Classify some lines quickly by AND of bit codes representing regions of two endpoints (≠ 0)
Cohen-Sutherland algorithm

- Classify some lines quickly by AND of bit codes representing regions of two endpoints (≠ 0)
Cohen-Sutherland algorithm

- Classify some lines quickly by AND of bit codes representing regions of two endpoints ($\neq 0$)
Cohen-Sutherland algorithm

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

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

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

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

```
1001
1000
1010
0001
0000
0010
0101
0100
0110
```

Bit 1  Bit 2  Bit 3  Bit 4

- $P_3$  $P_4$  $P_5'$  $P_6$
- $P_7$  $P_9$  $P_{10}$  $P_8$
Cohen-Sutherland algorithm

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

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

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

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

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

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

Bit 1: $P_5$, $P_5'$, $P_6$
Bit 2: $P_9$
Bit 3: $P_4$, $P_3$,
Bit 4: $P_7'$, $P_7$, $P_8'$, $P_{10}$
Compute intersections with window boundary for lines that can’t be classified quickly.
Cohen-Sutherland algorithm

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

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

- For each line:
  - Assign a region code to the end points
    - Ex.: $\text{Sign}(x - x_{min})$ for Bit 1
  - Accept if $\text{Or}(P_i, P_j) = 0000$
  - Reject if $\text{And}(P_i, P_j) \neq 0000$
  - For Bit i=1:4
    - Find intersection with the line
      - Look for a change in the relevant bit
    - Assign a region code to the intersection point
    - Check the two line segments
Cohen-Sutherland algorithm

- Disadvantages:
  - “Random” edge choice
  - Redundant edge-line cross calculations
Liang-Barsky algorithm

- **Parametric representation:**
  \[
  x = x_0 + u \cdot \Delta x, \quad 0 \leq u \leq 1 \\
  y = y_0 + u \cdot \Delta y, \quad 0 \leq u \leq 1 \\
  \Delta x = x_{\text{end}} - x_0, \quad \Delta y = y_{\text{end}} - y_0
  \]

- **Conditions:**
  \[
  wx_{\text{min}} \leq x \leq wx_{\text{max}} \\
  wy_{\text{min}} \leq y \leq wy_{\text{max}} \\
  \Rightarrow \\
  u_{p_k} \leq q_k, \quad k = 1, 2, 3, 4
  \]
Liang-Barsky algorithm

- For each line:
  - For each clipping boundary:
    - Reject if \( p_k = 0 \) and \( q_k \leq 0 \)
    - Calculate \( r_k = \frac{q_k}{p_k} \) (for \( p_k \neq 0 \))
  - \( u_1 = \max[0, r_k = \frac{q_k}{p_k}] \) (for \( p_k < 0 \))
  - \( u_2 = \min[1, r_k = \frac{q_k}{p_k}] \) (for \( p_k > 0 \))
  - Reject if \( u_1 > u_2 \)
  - Otherwise, take \((x(u_1), y(u_1))\) and \((x(u_2), y(u_2))\)
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
Polygon clipping

- Find the part of a polygon inside the clip window

After clipping
Polyon clipping

- Can we use the line clipping algorithm?
Can we use the line clipping algorithm?
Sutherland Hodgeman algorithm

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

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

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

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

- 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:
- Points $P_1, P_2, P_3, P_4, P_5$
- Window Boundary
- Inside
- Outside
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 with points P1, P2, P3, P4, and P5. The window boundary is shown with points P1 and P5 inside, and P3, P4 outside.]
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
Clipping to a Boundary

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

- For each polygon:
  - Accept if min & max are inside the window
  - Reject if min & max are outside the window
- For each clipping boundary:
  - For each line \((P_1, P_2)\) choose the following points:
    - (Outside, Inside): \{Intersection, \(P_2\}\)
    - (Inside, Inside): \{\(P_2\}\)
    - (Inside, Outside): \{Intersection\}
    - (Outside, Outside): \{-\}

- Output is a list of ordered vertices
2D rendering pipeline

3D Primitives

\[ \downarrow \]

2D Primitives

\[ \downarrow \]

**Clipping**

Clip portions of geometric primitives residing outside the window

\[ \downarrow \]

**Viewport Transformation**

Transform the clipped primitives from screen to image coordinates

\[ \downarrow \]

**Scan Conversion**

Fill pixels representing primitives in screen coordinates

\[ \downarrow \]

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**

\[
\begin{align*}
vx &= vx_{\text{min}} + (wx - wx_{\text{min}}) \frac{(vx_{\text{max}} - vx_{\text{min}})}{(wx_{\text{max}} - wx_{\text{min}})} \\
vx &= vy_{\text{min}} + (wy - wy_{\text{min}}) \frac{(vy_{\text{max}} - vy_{\text{min}})}{(wy_{\text{max}} - wy_{\text{min}})}
\end{align*}
\]
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