CS 418: Interactive Computer Graphics

Clipping

Eric Shaffer

Based on slides by John Hart
Graphics Pipeline

Model Coords → Model Xform → World Coords → Viewing Xform → Viewing Coords → Perspective Distortion

Homogeneous Divide → Still Clip Coords. → Clipping → Clip Coords. → Window Coordinates → Window to Viewport → Viewport Coordinates

Distortion
Why not just transform all triangles to the screen and just ignore pixels off the screen?

- Takes time to rasterize a triangle
- Very small number of triangles fall within the viewing frustum
- WebGL clips automatically
  - ...you don’t have to implement clipping
  - You should know how it works
Different rasterization engines can make different choices

- WebGL does it after the vertex shader runs
  - In 3D
  - Before performing division by the homogeneous coordinate
- Could also be done in 2D, after the division

We’ll look at a 2D clipping algorithm

- Generalizes to 3D
Outcodes

- Cohen-Sutherland
- Assign segment endpoints a bitcode $b_3b_2b_1b_0$
  - $b_0 = x < \text{left}$
  - $b_1 = x > \text{right}$
  - $b_2 = y < \text{bottom}$
  - $b_3 = y > \text{top}$
- Let $o_0 = \text{outcode}(x_0,y_0), o_1 = \text{outcode}(x_1,y_1)$
  - $o_0 = o_1 = 0$: segment visible
  - $o_0 = 0, o_1 \neq 0$: segment must be clipped
Cohen-Sutherland

Assign segment endpoints a bitcode

\[ b_3b_2b_1b_0 \]

- \( b_0 = x < \text{left} \)
- \( b_1 = x > \text{right} \)
- \( b_2 = y < \text{bottom} \)
- \( b_3 = y > \text{top} \)

Let \( o_0 = \text{outcode}(x_0, y_0), o_1 = \text{outcode}(x_1, y_1) \)

- \( o_0 = o_1 = 0: \text{segment visible} \)
- \( o_0 = 0, o_1 \neq 0: \text{segment must be clipped} \)
Outcodes

- Cohen-Sutherland
- Assign segment endpoints a bitcode
  \[ b_3b_2b_1b_0 \]
  - \( b_0 = x < \text{left} \)
  - \( b_1 = x > \text{right} \)
  - \( b_2 = y < \text{bottom} \)
  - \( b_3 = y > \text{top} \)
- Let \( o_0 = \text{outcode}(x_0,y_0) \), \( o_1 = \text{outcode}(x_1,y_1) \)
  - \( o_0 = o_1 = 0 \): segment visible
  - \( o_0 = 0, o_1 \neq 0 \): segment must be clipped
  - \( o_0 \& o_1 \neq 0 \): segment can be ignored
Cohen-Sutherland

Assign segment endpoints a bitcode

\[ b_3 b_2 b_1 b_0 \]

- \( b_0 = x < \text{left} \)
- \( b_1 = x > \text{right} \)
- \( b_2 = y < \text{bottom} \)
- \( b_3 = y > \text{top} \)

Let \( o_0 = \text{outcode}(x_0, y_0) \), \( o_1 = \text{outcode}(x_1, y_1) \)

- \( o_0 = o_1 = 0 \): segment visible
- \( o_0 = 0, o_1 \neq 0 \): segment must be clipped
- \( o_0 \& o_1 \neq 0 \): segment can be ignored
- \( o_0 \& o_1 = 0 \): segment might need clipping
Cohen-Sutherland

Assign segment endpoints a bitcode

\[ b_3 b_2 b_1 b_0 \]

- \( b_0 = x < \text{left} \)
- \( b_1 = x > \text{right} \)
- \( b_2 = y < \text{bottom} \)
- \( b_3 = y > \text{top} \)

Let \( o_0 = \text{outcode}(x_0, y_0), o_1 = \text{outcode}(x_1, y_1) \)

- \( o_0 = o_1 = 0: \text{segment visible} \)
- \( o_0 = 0, o_1 \neq 0: \text{segment must be clipped} \)
- \( o_0 \neq 0, o_1 = 0: \text{segment can be ignored} \)
- \( o_0 \neq 0, o_1 \neq 0: \text{segment might need clipping} \)
Intersecting Lines

- Parametric representation of a line segment
  
  \[ x(t) = x_0 + t \ (x_1 - x_0) \]
  \[ y(t) = y_0 + t \ (y_1 - y_0) \]
Intersecting Lines

- Parametric representation of a line segment
  \[ x(t) = x_0 + t (x_1 - x_0) \]
  \[ y(t) = y_0 + t (y_1 - y_0) \]

- Plug in clipping window edge to find \( t \)
  \[ \text{top} = y_0 + t (y_1 - y_0) \]
  \[ t = (\text{top} - y_0) / (y_1 - y_0) \]
Cohen-Sutherland Clipping

- $y = \text{top}$
- $y = \text{bottom}$
- $x = \text{left}$
- $x = \text{right}$

Diagram showing the clipping regions in the coordinate system.
Cohen-Sutherland Clipping

- First clip 0101
- Move \((x_0, y_0)\) to \((\text{left,} \ldots)\)
Cohen-Sutherland Clipping

- First clip 0101
- Move \((x_0, y_0)\) to (left,…)
- Then clip 1010
- Move \((x_1, y_1)\) to (right,…)

\[
\begin{align*}
\text{y = top} & : 1001, 1010, 0010, 0001 \\
\text{y = bottom} & : 0101, 0100, 0110, 0110 \\
\text{x = left} & : 0101, 0100, 0110, 0110 \\
\text{x = right} & : 1001, 1010, 0010, 0001
\end{align*}
\]
Cohen-Sutherland Clipping

- First clip 0001
- Move \((x_0, y_0)\) to (left,...)
- Then clip 0010
- Move \((x_1, y_1)\) to (right,...)
- Then clip 0100
- Move \((x_0, y_0)\) again, now to (...,bottom)
Cohen-Sutherland Clipping

- First clip 0101
  - Move \((x_0, y_0)\) to (left, \(\ldots\))
- Then clip 1010
  - Move \((x_1, y_1)\) to (right, \(\ldots\))
- Then clip 0100
  - Move \((x_0, y_0)\) again, now to (\(\ldots\), bottom)
- Finally clip 1000
  - Move \((x_1, y_1)\) again, now to (\(\ldots\), top)
Polygon Clipping

- Sutherland-Hodgman
- Polygon $ABC$
Polygon Clipping

- Sutherland-Hodgman
- Polygon $ABC$
- Clip left: $A_1BCA_2$
Polygon Clipping

- Sutherland-Hodgman
- Polygon $ABC$
- Clip left: $A_1BCA_2$
- Clip right: $A_1B_1B_2CA_2$
Polygon Clipping

- Sutherland-Hodgman
- Polygon $ABC$
- Clip left: $A_1BCA_2$
- Clip right: $A_1B_1B_2CA_2$
- Clip bottom: $A_1'B_1'B_2CA_2$
Polygon Clipping

- Sutherland-Hodgman
- Polygon $ABC$
- Clip left: $A_1B_1CA_2$
- Clip right: $A_1B_1'B_2CA_2$
- Clip bottom: $A_1B_1'B_2'C_2A_2$
- Clip top: $A_1B_1'B_2'C_1C_2A_2$
Concave Clipping

- Sutherland-Hodgman
- Clip segments even if they are trivially rejectible (rejectionable?)
- Outputs a single polygon that appears as multiple polygons
- Reversed edges don’t get filled
Clipping in 3D

- Clipping can be done in 3D clip coordinates
- Need to be able to compute
  - Which side of a plane a point is on
  - Line segment – Plane intersections
- Can still use Cohen-Sutherland
  - 6-bit outcodes
  - 27 different regions of space
Clipping in 3-D

- Need to keep depth (z-coordinate) of geometry for visible surface detection
- Generalize oriented screen edge to oriented clipping plane \( C = (A,B,C,D) \)
- Then any homogeneous point \( P = (x,y,z,w)^T \) classified as
  - “on” if \( C \cdot P = 0 \)
  - “in” if \( C \cdot P < 0 \)
  - “out” if \( C \cdot P > 0 \)

\[
\begin{bmatrix}
A & B & C & D
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
z \\
w
\end{bmatrix} = 0
\]

\( Ax + By + Cz + D = 0 \)
\( \uparrow \)
\( wAx + wBy + wCz + wD = 0 \)
Clipping in 3D

- Plane equation can be rewritten \( n \cdot (p - p_0) = 0 \)
  - \( n \) the normal and \( p_0 \) is a point on the plane
  - plane is formed by all points \( p \) for which equation is true