Chapter 7

Clipping

Rendering Pipeline

Geometry Processing

- Discard geometry outside viewport window

Copyright Alla Sheffer UBC 2013
Line and Polygon Clipping

Problem:
Given a 2D line/polygon and a window, clip the line/polygon to their regions that are inside the window.

- Objectives
  - Efficiency
  - (Parallelization)
- Two approaches
  - Explicit (continuous setting)
  - Implicit (discrete setting) – part of scan conversion

Convexity

Set $C \subseteq \mathbb{R}^d$ is **convex** if for any two points $p, q \in C$ and any $\alpha \in [0,1]$, $\alpha p + (1-\alpha)q \in C$

2D Projection of **convex** 3D shape is **convex**

Copyright Alla Sheffer UBC 2013
Explicit Solution: Line Segments

- Intersection of convex regions is convex
  - Why?

- \( L \) & \( D \) are convex - intersection is convex
  - single connected segment of \( L \)

- Clipping uses intersections of \( L \) with four boundary segments of window \( D \)

Basic Method

```
Clip(P_1, P_2, x_min, x_max, y_min, y_max);
if ((P_0 and P_1 inside window) then draw(P_0, P_1);
  test if segment \( (P_0, P_1) \) intersects any of the edges
  if not, return;
else let \( P_2 \) be the first intersection found
  Clip(P_0, P_2, x_min, x_max, y_min, y_max);
  Clip(P_2, P_1, x_min, x_max, y_min, y_max);
end
```

- Works, but inefficient for lines OUTSIDE \( D \)
  - Four intersection tests
- Note: need special care for vertices ON window edges
Segment-Segment Intersection

Intersection: \( x \) & \( y \) values equal in both representations - two linear equations in two unknowns \((r,t)\)

Test if resulting \( r \) & \( t \) are inside the \([0,1]\) range

\[
\begin{align*}
\frac{x_1 - x_0}{x_1 - x_0} + \frac{y_1 - y_0}{y_1 - y_0} t = 1 & \Rightarrow \frac{x_1 - x_0}{x_1 - x_0} \leq t \leq 1, \quad \text{if} \quad \frac{x_1 - x_0}{x_1 - x_0} \leq 1, \quad \text{else} \quad t \leq 1 \\
\frac{x_1 - x_0}{x_1 - x_0} + \frac{y_1 - y_0}{y_1 - y_0} (1 - t) = 1 & \Rightarrow \frac{x_1 - x_0}{x_1 - x_0} \leq t \leq 1, \quad \text{if} \quad \frac{x_1 - x_0}{x_1 - x_0} \leq 1, \quad \text{else} \quad t \leq 1
\end{align*}
\]

Intersection with axis-aligned lines

Intersection: \( x \) & \( y \) values equal in both representations - two linear equations in two unknowns \((r,t)\)

\[
\begin{align*}
x_0 + \frac{x_1 - x_0}{x_1 - x_0} x = x_0^2 \quad & \Rightarrow \frac{x_1 - x_0}{x_1 - x_0} \leq \frac{r - x_0}{x_1 - x_0} \leq 1, \quad \text{if} \quad \frac{x_1 - x_0}{x_1 - x_0} \leq 1, \quad \text{else} \quad \frac{r - x_0}{x_1 - x_0} \leq 1 \\
y_0 + \frac{y_1 - y_0}{y_1 - y_0} y = y_0^2 \quad & \Rightarrow \frac{y_1 - y_0}{y_1 - y_0} \leq \frac{r - y_0}{y_1 - y_0} \leq 1, \quad \text{if} \quad \frac{y_1 - y_0}{y_1 - y_0} \leq 1, \quad \text{else} \quad \frac{r - y_0}{y_1 - y_0} \leq 1
\end{align*}
\]

Copyright Alla Sheffer UBC 2013
Line and Polygon Clipping

**Line Clipping**

- Fast treatment of line segments that are trivially inside/outside window.

**Cohen-Sutherland Algorithm**

**Purpose:**
Fast treatment of line segments that are trivially inside/outside window.

- $P = (x, y)$ - point to be classified against window $D$

**Idea:** Assign to $P$ a binary code consisting of a bit for each edge of $D$, using lookup table:

<table>
<thead>
<tr>
<th>bit</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$y &lt; y_{min}$</td>
<td>$y \geq y_{min}$</td>
</tr>
<tr>
<td>2</td>
<td>$y &gt; y_{max}$</td>
<td>$y \leq y_{max}$</td>
</tr>
<tr>
<td>3</td>
<td>$x &gt; x_{max}$</td>
<td>$x \leq x_{max}$</td>
</tr>
<tr>
<td>4</td>
<td>$x &lt; x_{min}$</td>
<td>$x \geq x_{min}$</td>
</tr>
</tbody>
</table>

Copyright Alla Sheffer UBC 2013
**Line and Polygon Clipping**

**Line Clipping**

- \(0101\)  \(0100\)  \(0110\)
- \(0001\)  \(0000\)  \(0010\)
- \(1001\)  \(1000\)  \(1010\)

**Cohen-Sutherland Algorithm (cont’d)**

Given \(L\) from \((x_0, y_0)\) to \((x_1, y_1)\) & rectangle \(D\).

If bitwise **and** of the codes of \((x_0, y_0)\) and \((x_1, y_1)\) is not zero, or the bitwise **or** is zero,

then \(L\) can be trivially handled (it is either totally outside or totally inside \(D\)).

*Why?*

---

Copyright Alla Sheffer UBC 2013
Line and Polygon Clipping

Cohen-Sutherland Algorithm (cont’d)

\[ C - S - \text{Clip}(P_i, P_j) = (x_{min}, y_{min}, x_{max}, y_{max}) \]

(\text{assumes } x_i \leq x_j )

\[ C_0 \equiv \text{code}(P_i); \quad C_1 \equiv \text{code}(P_j); \]

if \(( (C_0 \text{ and } C_1) = \neq 0) \) then return;

if \(( (C_0 \text{ or } C_1) = \neq 0) \) then draw\((P_i, P_j)\);

else if \((\text{OutsideWindow}(P_i) ) \) then

\begin{align*}
\text{Edge} & \equiv \text{Window boundary of leftmost non-zero bit of } C_i; \\
P_i & \equiv P_i \cap \text{Edge}; \\
C - S - \text{Clip}(P_i, P_j, x_{min}, x_{max}, y_{min}, y_{max}); \\
\end{align*}

\text{end}

3D clipping

- Determine portion of line inside axis-aligned box (viewing frustum in NDC)
- Simple extension of 2D algorithms
- After projection transform
  - clipping volume always the same
    \( x_{min} = y_{min} = z_{min} = -1, \ x_{max} = y_{max} = z_{max} = 1 \)
  - boundary lines become boundary planes
    \[ \text{but bit-codes still work the same way} \]
Triangle Clipping

- How does intersection of rectangle & triangle looks like?
  - How many sides?

- How to expand clipping to triangles?
  - Hint: it is convex
  - Will sketch on the board…

Cohen-Sutherland Algorithm
for convex polygons

```
C - S - Clip( poly = P_1, P_2, x_{min}, x_{max}, y_{min}, y_{max} )
for i = 1 to n C_i := code( P_i );
if ( ( C_1 and C_2 and ... and C_n ) != 0 ) then return;
if ( ( C_1 or C_2 or ... or C_n ) == 0 ) then draw( poly );
else
  for i = 1 to n if ( OutsideWindow( P_i ) ) then
    begin
      Edge := Window boundary of leftmost non-zero bit of C_i;
      P_{i-1} := P_{i-1}, P_{i} \cap Edge;
      P_{i+1} := P_{i}, P_{i} \cap Edge;
      C - S - Clip( P_1, ..., P_{i-1}, P_{i+1}, P_{i+2}, ..., P_n, x_{min}, x_{max}, y_{min}, y_{max} )
    end
end
```

Copyright Alla Sheffer UBC 2013