Chapter 7

Clipping

Rendering Pipeline

- Discard geometry outside viewport window

Copyright Alla Sheffer UBC 2012
Line and Polygon Clipping

Line/Polygon Clipping (2D)

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 $\alpha \in [0, 1]$, $\alpha p + (1-\alpha)q \in C$

Convex

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

Copyright Alla Sheffer UBC 2012
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

```plaintext
Clip(P0, P1, x_min, x_max, y_min, y_max)
if (P0 and P1 inside window) then draw(P0, P1);
else if segment (P0, P1) intersects any of the edges
  if not, return;
  else let P2 be the first intersection found
  Clip(P0, P2, x_min, x_max, y_min, y_max);
Clip(P2, P1, 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

Copyright Alla Sheffer UBC 2012
### Segment-Segment Intersection

Given two line segments $G_1$ and $G_2$ defined by their endpoints:

- $G_1 = (x_0^1, y_0^1)$ to $(x_1^1, y_1^1)$
- $G_2 = (x_0^2, y_0^2)$ to $(x_1^2, y_1^2)$

The intersection of these segments can be found by solving the system of linear equations:

- $x^1(t) = x_0^1 + (x_1^1 - x_0^1)t$
- $x^2(r) = x_0^2 + (x_1^2 - x_0^2)r$
- $y^1(t) = y_0^1 + (y_1^1 - y_0^1)t$
- $y^2(r) = y_0^2 + (y_1^2 - y_0^2)r$

The solution $(r, t)$ must satisfy $t \in [0, 1]$ and $r \in [0, 1]$ to ensure the intersection point lies within both segments.

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.

### Intersection with axis-aligned lines

For axis-aligned lines:

- $G_1$ is defined by $(x_0^1, y_0^1)$ to $(x_1^1, y_1^1)$
- $G_2$ is defined by $(x_0^2, y_0^2)$ to $(x_1^2, y_1^2)$

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

Test if $t < 0$ or $t > 1$ no intersection on segments.

Copyright Alla Sheffer UBC 2012
### Line Clipping

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

- **Idea:** Assign to \( P = (x, y) \) - point to be classified against window \( D \), using lookup table:

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

### Cohen-Sutherland Algorithm

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

- **Idea:** Assign to \( P = (x, y) \) - point to be classified against window \( D \), using lookup table:
Line and Polygon Clipping

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 2012
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
  - xmin=ymin=zmin= -1, xmax=ymax=zmax= 1
- boundary lines become boundary planes
  - **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-
0101 0100 0110
0001 0000 0010
1001 1000 1010
```

```c-
C - S - Clip( poly = P_{p_1}, P_{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_{edge} ⇐ P_{i+1} \cap Edge;
P_{out} ⇐ P_i \cap Edge;
C - S - Clip( P_{p_1}, P_{p_2}, p_{i+1}(P_{i+1}, P_{i+2}, P_{i+3}, P_{i+4}, x_{min}, y_{min}, y_{max} ),
end
```

```c-
bit 1 0
1 \ y < y_{min} \ y \geq y_{min}
2 \ y > y_{max} \ y \leq y_{max}
3 \ x > x_{max} \ x \leq x_{max}
4 \ x < x_{min} \ x \geq x_{min}
```

Copyright Alla Sheffer UBC 2012