Cyrus-Beck/Liang-Barsky clipping

- Parametric clipping: consider line in parametric form and reason about the parameter values
- More efficient, as we don’t compute the coordinate values at irrelevant vertices

Line is:
\[
\begin{align*}
  x &= x_1 + t \Delta x \\
  y &= y_1 + t \Delta y
\end{align*}
\]

\[\Delta x = x_2 - x_1\]
\[\Delta y = y_2 - y_1\]

Segment restricts \( t \) to be inside \([0,1]\)

Computing \( t \) for intersection point

Think of \( X \) moving along the line shown. What is the condition that it is on the other line as well (i.e., intersects?)

Simplest to work from condition
\[
(P_e - X(t)) \cdot n = 0
\]

Set
\[
D = X_2 - X_1
\]

Then
\[
X(t) = X_1 + tD
\]

And condition is
\[
(P_e - (X_1 + tD)) \cdot n = 0
\]
Computing $t$ for intersection point

**Condition**

$$(P_e - (X_1 + tD)) \cdot n = 0$$

**Rearrange**

$$(P_e - X_1) \cdot n = d \cdot n$$

**And solve**

$$t = \frac{(P_e - X_1) \cdot n}{D \cdot n}$$

---

**Computing $t$ for intersection point**

From previous slide

$$t = \frac{(P_e - X_1) \cdot n}{D \cdot n}$$

This simplifies greatly for axis aligned rectangles

Consider left edge. Now $n=\vec{-1}$ and $P_e=\vec{x_{min}}$

And $t = ?$

---

**Computing $t$ for intersection point**

All four special cases can be expressed by:

$$t = \frac{q_k}{p_k}$$

Where

- $p_1 = -\Delta x, q_1 = x_1 - x_{min}$
- $p_2 = \Delta x, q_2 = x_{max} - x_1$
- $p_3 = -\Delta y, q_3 = y_1 - y_{min}$
- $p_4 = \Delta y, q_4 = y_{max} - y_1$

One can also get this special case **directly** by solving:

$$x_{min} \leq x_1 + t\Delta x \leq x_{max}$$

$$y_{min} \leq y_1 + t\Delta y \leq y_{max}$$
Cyrus-Beck/Liang-Barsky (cont)

- Next step: Use the $t$’s to determine the clip points
- Recall that only  $t$ in $\langle 0,1 \rangle$ is relevant, but we need additional logic to determine clip endpoints from multiple $t$’s inside $\langle 0,1 \rangle$.
- We imagine going from $X_1$ to $X_2$ and classify intersections as either potentially entering (PE) or potentially leaving (PL) if they go across a clip edge from outside in or inside out.
- This is easily determined from the sign of $D\cdot n$ which we have already computed.

\[ D = X_2 - X_1 \]

Cyrus-Beck/Liang-Barsky--Algorithm

- Compute incoming (PE) $t$ values, which are $q_k/p_k$ for each $p_k<0$
- Compute outgoing (PL) $t$ values, which are $q_k/p_k$ for each $p_k>0$
- Parameter value for small $t$ end of the segment is:
  \[ t_{small} = \max(0, \text{incoming values}) \]
- Parameter value for large $t$ end of the segment is:
  \[ t_{large} = \min(1, \text{outgoing values}) \]
- If $t_{small} > t_{large}$, there is a segment portion in the clip window - compute endpoints by substituting these two $t$ values (how)?
- Otherwise reject because it is outside.

Cyrus-Beck/Liang-Barsky--Notes

- Works fine if clipping window is not an axis-aligned rectangle. Computing the $t$ values is just more expensive.
- **Bibliographic note:** Original algorithm was Cyrus-Beck (close to what we have done here). A very similar algorithm was independently developed later by Liang-Barsky with some additional improvements for identifying early rejects as the $t$ values are computed.
Nicholl-Lee-Nicholl clipping

- Fast specialized method
- We will just outline the basic idea
- Consider segment with endpoints: a, b
- Cases:
  - a inside
  - a in edge region
  - a in corner region
- For each case, we generate specialized test regions for b
- Which region b is in is determined by simple “which-side” tests.
- The region b is in determines which edges need to be clipped against.
- Speed is enhanced by good ordering of tests, and caching intermediate results

Polygon clip (against convex polygon)

Sutherland-Hodgeman polygon clip

- Recall: polygon is convex if any line joining two points inside the polygon, also lies inside the polygon; implies that a point is inside if it is on the right side of each edge.
- Clipping each edge of a given polygon doesn’t make sense - how do we reassemble the pieces? We want to arrange doing so on the fly.
- Clipping the polygon against each edge of the clip window in sequence works if the clip window is convex.
- (Note similarity to Sutherland-Cohen line clipping)
Sutherland-Hodgeman polygon clip

Clip window
Polygon to clip

Clip entire polygon against one edge

Then clip it against the next

Then the next
Sutherland-Hodgeman polygon clip

Polygon to clip

Clip window

1

2

3

4

And finally, the last one.

Clipping against current clip edge

- Polygon is a list of vertices
- Think of process as rewriting polygon, vertex by vertex
- Check start vertex
  - in - emit it
  - out - ignore it
- Walk along vertices and for each edge consider four cases and apply corresponding action.

- Four cases:
  - polygon edge crosses clip edge going from out to in
    - emit crossing, next vertex
  - polygon edge crosses clip edge going from in to out
    - emit crossing
  - polygon edge goes from out to out
    - emit nothing
  - polygon edge goes from in to in
    - emit next vertex

Start vertex

class edge crossing

class edge crossing

**implies**

emit crossing, next vertex

emit crossing

emit nothing

emit next vertex
Start vertex

- Polygon edge crosses clip edge going from out to in: emit crossing, next vertex
- Polygon edge crosses clip edge going from in to out: emit crossing
- Polygon edge goes from out to out: emit nothing
- Polygon edge goes from in to in: emit next vertex

Now have

- Polygon edge crosses clip edge going from out to in: emit crossing, next vertex
- Polygon edge crosses clip edge going from in to out: emit crossing
- Polygon edge goes from out to out: emit nothing
- Polygon edge goes from in to in: emit next vertex

New start vertex

Clip against next edge

- Polygon edge crosses clip edge going from out to in: emit crossing, next vertex
- Polygon edge crosses clip edge going from in to out: emit crossing
- Polygon edge goes from out to out: emit nothing
- Polygon edge goes from in to in: emit next vertex

Now have

- Polygon edge crosses clip edge going from out to in: emit crossing, next vertex
- Polygon edge crosses clip edge going from in to out: emit crossing
- Polygon edge goes from out to out: emit nothing
- Polygon edge goes from in to in: emit next vertex
More Polygon clipping

- Notice that we can have a pipeline of clipping processes, one against each edge, each operating on the output of the previous clipper -> substantial advantage.
- Unpleasantness can result from concave polygons; in particular, polygons with empty interior.
- Can modify algorithm for concave polygons (e.g. Weiler Atherton)

Weiler Atherton

For clockwise polygon (starting outside):
- For out-to-in pair, follow usual rule
- For in-to-out pair, follow clip edge (clockwise) and then jump to next vertex (which is on the outside) and start again
- Only get a second piece if polygon is convex
Additional remarks on clipping

- Although everything discussed so far has been in terms of polygons/lines clipped against lines in 2D, all - except Nicholl-Lee-Nicholl - will work in 3D against convex regions without much change.
- This is because the central issue in each algorithm is the inside outside decision as a convex region is the intersection of half spaces.
- Inside-outside decisions can be made for lines in 2D, planes in 3D. e.g testing \( d \cdot n > 0 \)
- Hence, all (except N-L-N) can be used to clip:
  - Lines against 3D convex regions (e.g. cubes)
  - Polygons against 3D convex regions (e.g. cubes)
- N-L-N could work in 3D, but the number of cases increases too much to be practical.