Clipping references

Hearn and Baker

C-S (lines): p 317
L-B (lines): p 322
N-L (lines): p 325

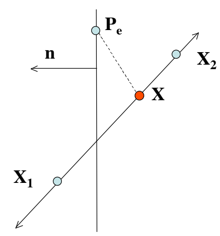
S-H (poly): p 331
W-A(poly): p 335

Foley at al.

C-S (lines): p 103
L-B (lines): p 107
N-L (lines): N.A.

S-H (poly): p 112
W-A(poly): N.A.

Computing t for intersection point



Condition

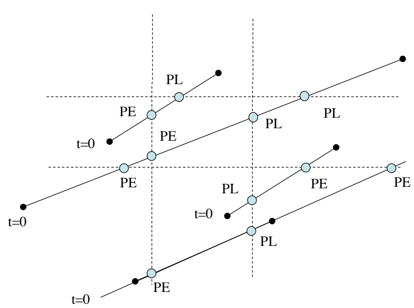
$$(\mathbf{P_e} - (\mathbf{X_1} + tD)) \bullet \mathbf{n} = \mathbf{0}$$

Rearrange

$$(\mathbf{P_e} - \mathbf{X_1}) \bullet \mathbf{n} = t\mathbf{D} \bullet \mathbf{n}$$

And solve

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



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, incoming values)
- Parameter value for large t end of the segment is:

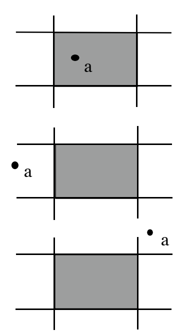
  t_{large}=min(1, outgoing values)
- If $t_{small}<t_{large}$, there is a segment portion in the clip window - compute endpoints by substituting t values (otherwise reject as it is outside).

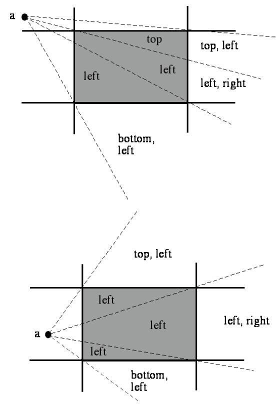
Cyrus-Beck/Liang-Barsky--Notes

- Works fine if clipping window is not an axis-aligned rectangle. Computing the t vaues 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.

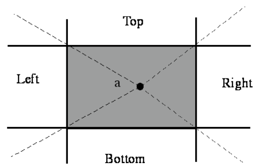
Nicholl-Lee-Nicholl clipping

- 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

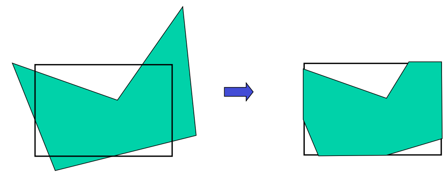




NLN clipping: Compute the area that b is in, and clip the segment ab against the edges specified.



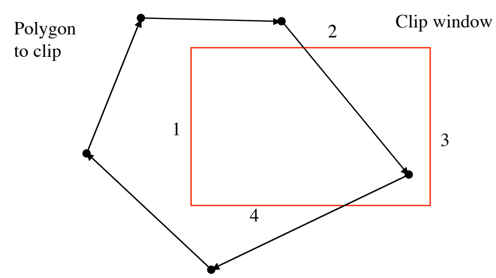
Polygon clip (against convex polygon)



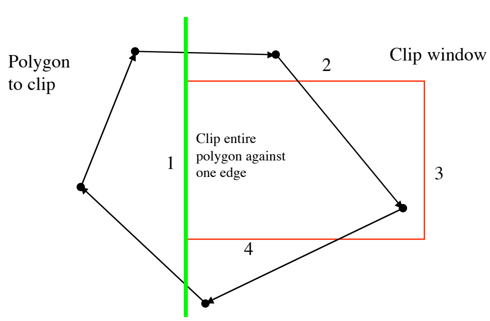
Sutherland-Hodgeman polygon clip

- 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)

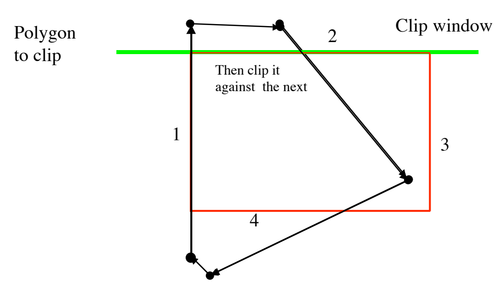
Sutherland-Hodgeman polygon clip



Sutherland-Hodgeman polygon clip

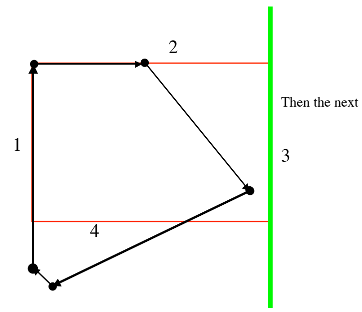


Sutherland-Hodgeman polygon clip



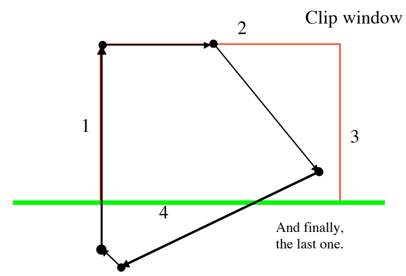
Sutherland-Hodgeman polygon clip

Polygon
to clip



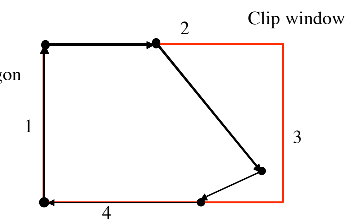
Sutherland-Hodgeman polygon clip

Polygon
to clip



Sutherland-Hodgeman polygon clip

Clipped polygon



Clipping against current clip edge

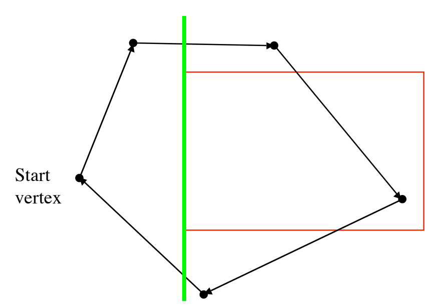
- Polygon is a list of vertices
- Think of process as rewriting polygon, vertex by vertex
- Check start vertex
  - in - emit it
  - out - ignore it
- Walk along vertices and for each edge consider four cases and apply corresponding action.

- Four cases:
  - polygon edge crosses clip edge going from out to in
    -
  - polygon edge crosses clip edge going from in to out
    -
  - polygon edge goes from out to out
    -
  - polygon edge goes from in to in
    -

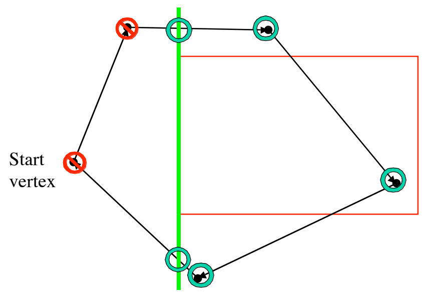
Clipping against current clip edge

- 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

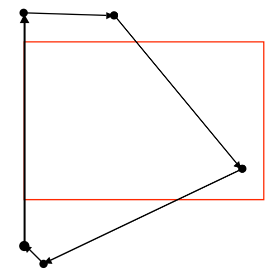


polygon edge crosses clip edge going from out to in ==> emit crossing, next vertex
polygon edge crosses clip edge going from in to out ==> emit crossing
polygon edge goes from out to out ==> emit nothing
polygon edge goes from in to in ==> emit next vertex



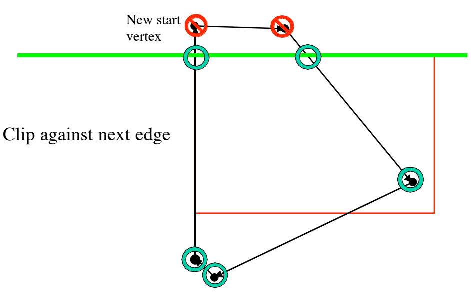
polygon edge crosses clip edge going from out to in	==> emit crossing, next vertex
polygon edge crosses clip edge going from in to out	==> emit crossing
polygon edge goes from out to out	==> emit nothing
polygon edge goes from in to in	==> emit next vertex

Now have



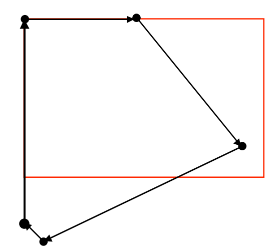
polygon edge crosses clip edge going from out to in	==> emit crossing, next vertex
polygon edge crosses clip edge going from in to out	==> emit crossing
polygon edge goes from out to out	==> emit nothing
polygon edge goes from in to in	==> emit next vertex

Clip against next edge



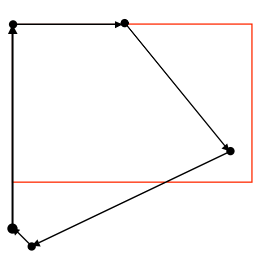
polygon edge crosses clip edge going from out to in	==> emit crossing, next vertex
polygon edge crosses clip edge going from in to out	==> emit crossing
polygon edge goes from out to out	==> emit nothing
polygon edge goes from in to in	==> emit next vertex

Now have



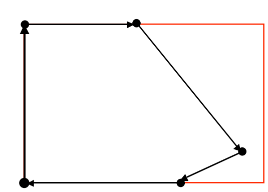
polygon edge crosses clip edge going from out to in	==> emit crossing, next vertex
polygon edge crosses clip edge going from in to out	==> emit crossing
polygon edge goes from out to out	==> emit nothing
polygon edge goes from in to in	==> emit next vertex

Clipping against
next edge (right)
gives



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

Clipping against
final(bottom)
edge gives



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

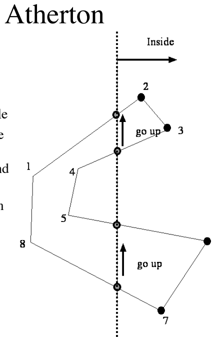
More Polygon clipping

- 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 dx•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)
- NLN could work in 3D, but the number of cases increases too much to be practical.