Clipping

- 2D elements are laid out in a convenient (often user based) coordinate system and then transformed to a frame buffer coordinate system.
- Objects that are to be drawn must lie inside frame buffer, and may have to lie inside particular region - e.g. viewport.
- We want to dodge additional expensive operations on objects or parts of objects that won’t be displayed.
- How do we ensure that the line/polygon lies inside a region?
  - (Answer) Cut them up?

Clipping in the 2D pipeline

Refer to pages:
- S (lines): p 317
- L-B (lines): p 322
- N-L (lines): p 325
- S-H (poly): p 331
- W-A(poly): p 335

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.

Clipping lines

Cohen-Sutherland clipping (lines)

- Clip line against convex region.
  - For each edge of the region, clip line against that edge:
    - line all on wrong side of any edge? throw it away (trivial reject - e.g. red line with respect to bottom edge)
    - line all on correct side of all edges? doesn’t need clipping (trivial accept - e.g. green line)
    - line crosses edge? replace endpoint on wrong side with crossing point (clip)

Have

Need
Cohen Sutherland - details

- Only need to clip line against edges where one endpoint is inside and one is outside.
- The state of the outside endpoint (e.g., in or out, w.r.t. a given edge) changes due to clipping as we proceed--need to track this.
- Use “outcode” to record endpoint in/out wrt each edge. One bit per clipping edge, 1 if out, 0 if in.

Outcode example

Outcode example

Outcode example

Cohen Sutherland - Algorithm

- Compute outcodes for endpoints
- While not trivial accept and not trivial reject:
  - clip against a problem edge (i.e. one for which an outcode bit is 1)
  - compute outcodes again
- Return appropriate data structure