CLIPPING ON A RASTER DISPLAY

- Clipping:
  - Remove points outside a region of interest.
  - Discard (parts of) primitives outside of window
- Point clipping:
  - Remove points outside window.
  - A point is either entirely inside the region or not.
- Line clipping:
  - Remove portion of line segment outside window.
- Polygon clipping:
  - Remove portion of polygon outside window

Approaches to Clipping

- Analytically
  - best for floating-point packages
  - scan convert the clipped primitives
- During scan conversion as part of span arithmetic-*scissoring*
  - good for filled or thick primitives
  - only extrema need be clipped
  - good if primitives not much larger then clip rectangle
    - not too many extra pixels generated
- As part of copy-pixel operation
  - useful when likely to have window pan over large canvas

Role of Analytical Clipping

- Floating-point packages
  - clip then scan convert
- Integer packages
  - pre-clip and scan convert or
  - clip during scan conversion
- Common strategy
  - clip lines and polygons analytically
  - clip other primitives during scan conversion

Analytic Clipping of Lines against Rectangles

- Lines are always clipped to at most one line segment
- Lines on rectangle border consider inside (displayed)

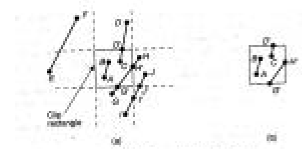


Fig. 3.38 Cases for clipping lines.

- Clipping (End)points
  - inside $x_{min} <= x <= x_{max}$, $y_{min} <= y <= y_{max}$

Clipping Rules

- Consider only end-points
- If both inside rectangle - *trivally accepted*
- One inside, one outside - need to compute intersection
- Both outside - may or may not intersect with clip rectangle

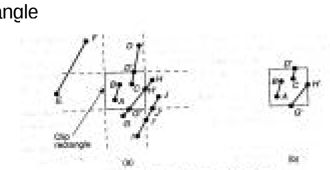


Fig. 3.38 Cases for clipping lines.

Computing Intersections

- Brute force
  - intersect each line with 4 edges of clip rectangle
  - need to solve for intersection point per edge
    - treat line and edge as infinite
    - solve for intersection
    - test if intersection is interior to line and edge
  - each intersection involves solving 2 simultaneous equations and the interior test

Computing Intersections

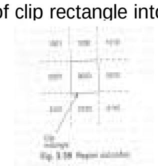
- Could use slope-intercept formula
  - really for infinite lines - does not handle vertical
- Use parametric form instead
  - $x = x_0 + t(x_1 - x_0)$, $y = y_0 + t(y_1 - y_0)$, t in [0,1]
- solve for t_{edge} and t_{line}
  - if both lie in [0,1] then real intersection
  - still need to test for lines parallel to clip rectangle edges
- Still involves much calculation

Cohen-Sutherland Algorithm

- Performs tests to avoid calculations
  - check for *trivial acceptance*
  - do *region tests* e.g. if both endpoints lie to left of rectangle *trivially rejected* etc.
  - if neither divide line segment in two at clip edge
    - trivially reject one
  - continue comparing against each clip edge
- Efficient if clip rectangle large (almost all inside) or small (almost all outside) *picking window*

Outcodes

- Divide plane of clip rectangle into 9 regions



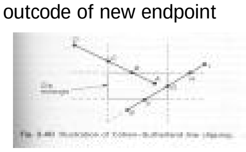
- Assign 4 bit code to each - each bit (1 or 0) indicates position wrt outside half-plane of clip edge
- Can calculate efficiently as sign bit of $(y_{max}-y)$, $(y- y_{min})$, $(x_{max}-x)$, $(x- x_{min})$

Cohen-Sutherland Procedures

- Consider endpoints of line segment
- Both outcodes 0000, line inside, *trivially accept*
- Both in outside plane of same edge, *trivially reject*
  - if logical **and** of outcode not 0 *trivially reject*
- If neither, subdivide at edge
  - throw away outside segment
- Convention : go top to bottom, right to left i.e. follow the bit order in the outcode
- Outcode property: if bit set, line crosses edge
  - makes it easy to divide segment at edge

Cohen-Sutherland Procedures

- Compute outcodes of endpoints
- Check for trivial acceptance or rejection
- if neither
  - find outside endpoint
  - test outcode to find edge crossed, compute intersection
  - clip by replacing outside point by intersection
  - compute outcode of new endpoint

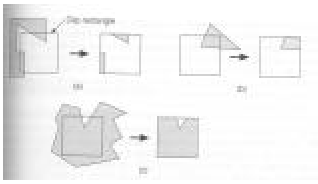


Cohen-Sutherland Procedure

- Efficiency
  - use bitwise arithmetic for outcodes
  - do not recalculate slopes
- Not most efficient
  - sometimes needless clipping e.g. clip line at clip line outside clip rectangle
  - Nicholl, Lee, Nicholl avoids this
- Advantage of Cohen-Sutherland
  - extension to 3D orthographic view volume straightforward

Clipping Polygons against Rectangles

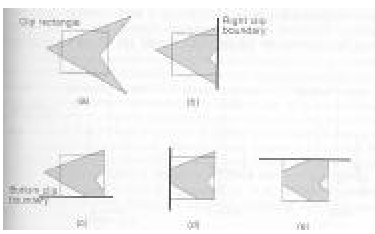
- Clipping rectangle gives (at most) rectangle
- Clipping convex polygon gives convex polygon
- Clipping concave polygon may produce more than one concave polygon
- Clipping a circle or ellipse may result in up to 4 arcs



Sutherland-Hodgman Algorithm

- Clip polygon (convex or concave) against any convex clipping polygon
- 3D: against convex polyhedral volumes defined by planes
- Accepts a set of vertices
- Clips against a single infinite clip edge and outputs another series of vertices
- Clips against next plane etc.
- At each step 0, 1 or 2 vertices are added to output list
- To test if point is inside, test sign of dot product of normal to clip boundary and the polygon edge
- For upright clip rectangle, sign of distance to boundary

Sutherland-Hodgman Algorithm



Improvements and Generalizations

- Can structure
  - so that it is reentrant
  - so no intermediate storage is required
- Pass polygon through pipeline of clippers
  - makes suitable for hardware implementation