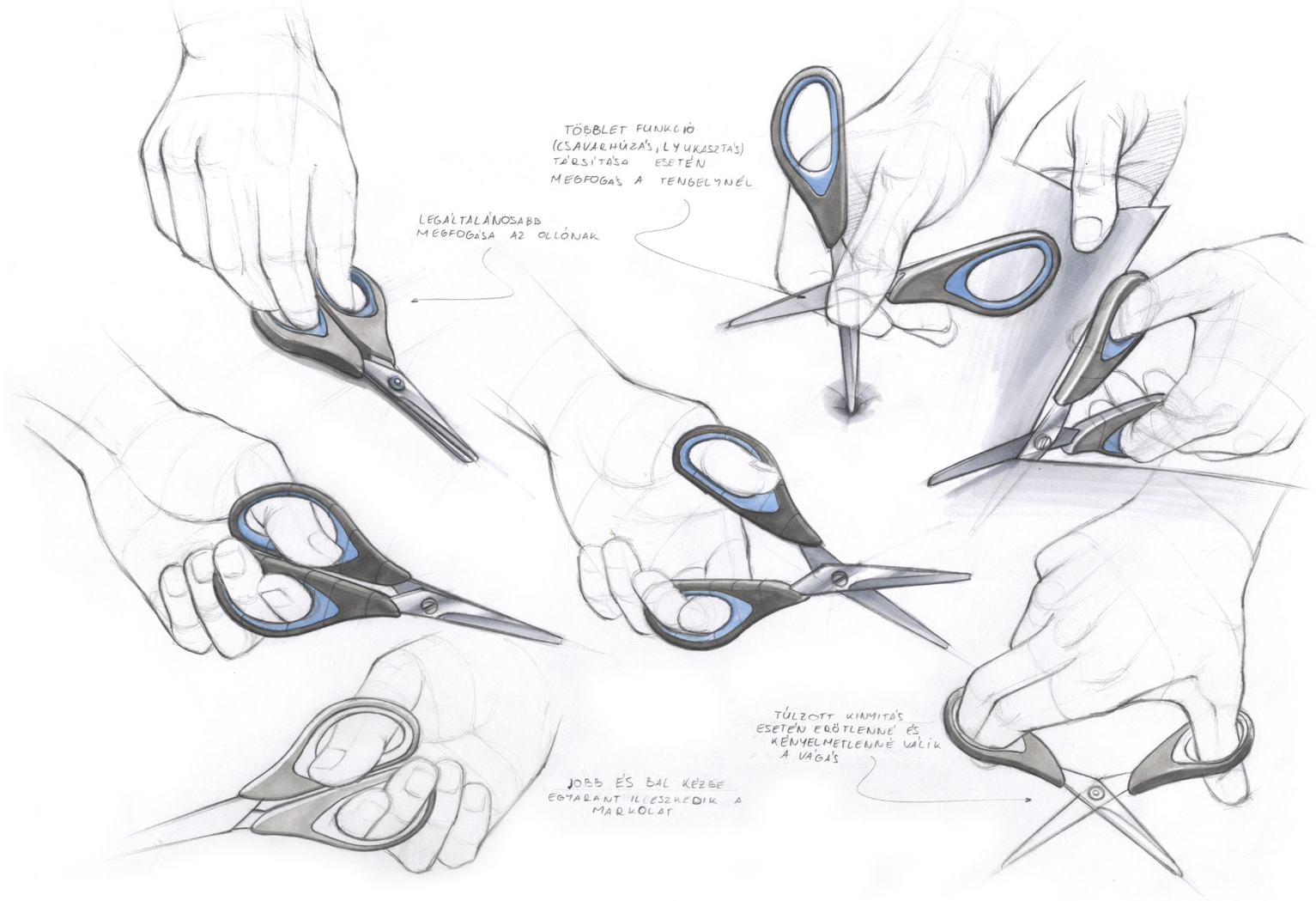
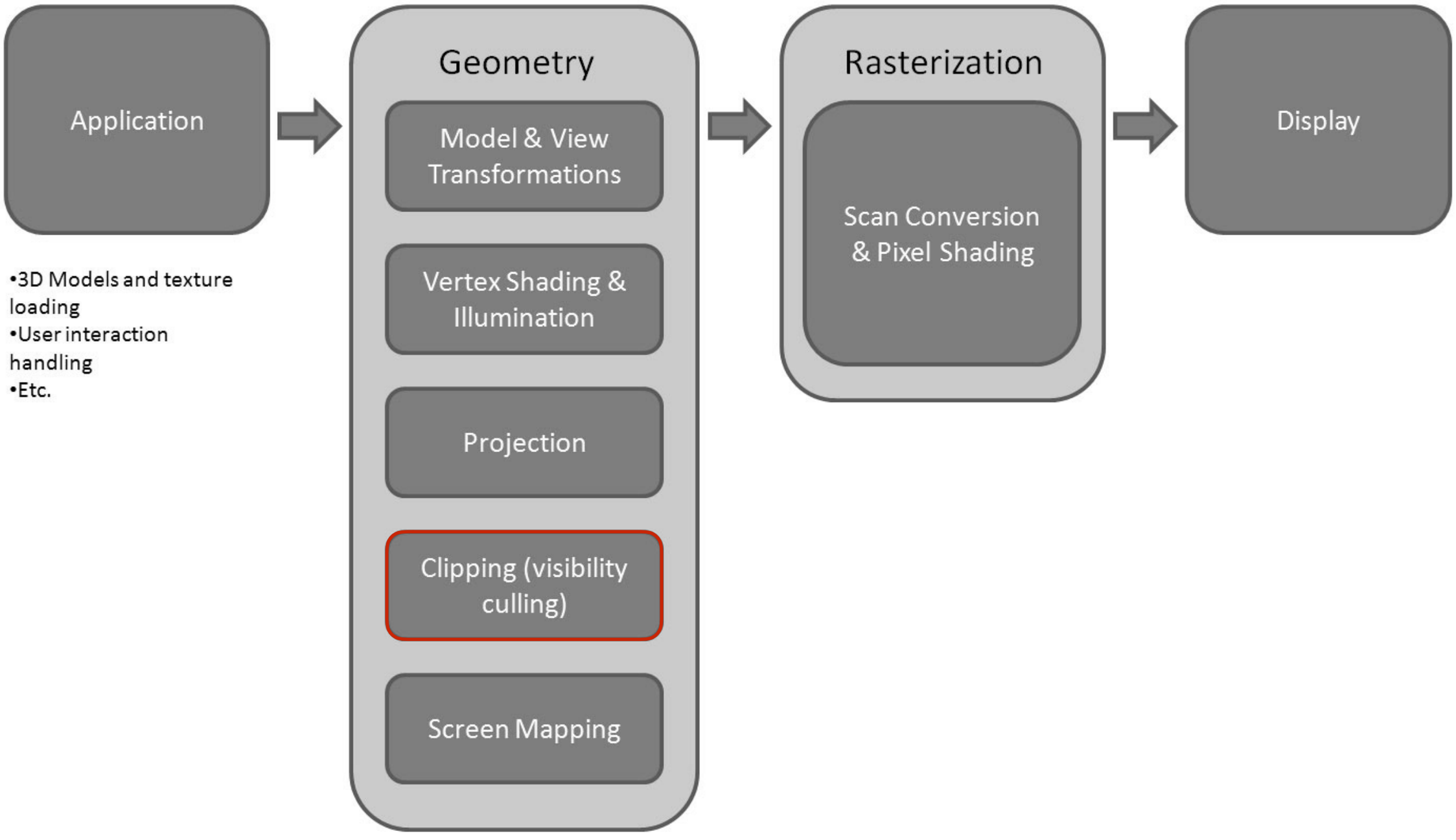


Clipping

Introduction to Computer Graphics
CSE 533 / 333

Graphics Pipeline



- 3D Models and texture loading
- User interaction handling
- Etc.

Source: http://www.gamersnexus.net/guides/2429-gpu-rendering-and-game-graphics-explained

Clipping

- Rasterization is preceded by a *clipping* operation
- Process of determining which primitives, or parts of primitives, fit within the view volume
- Performed before perspective divide

$$w \geq x, y, z \geq -w$$

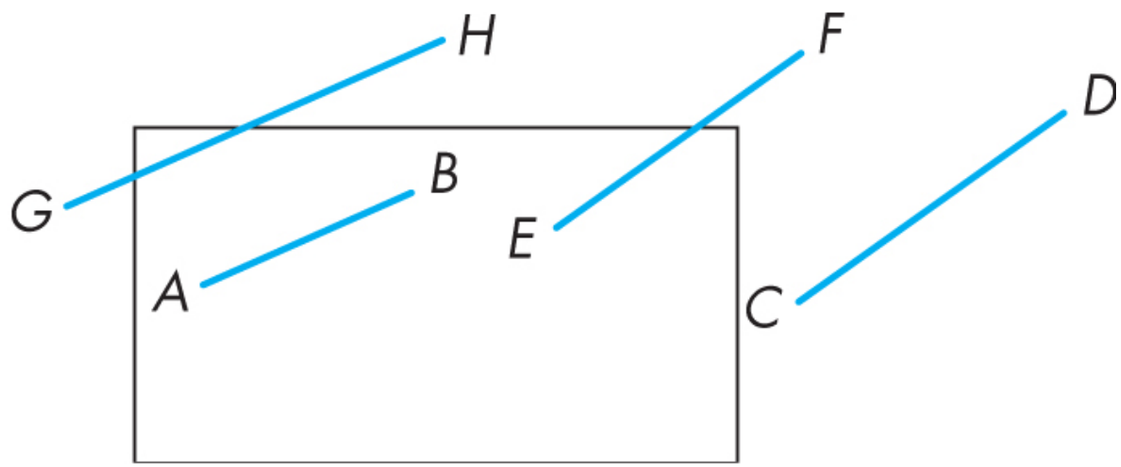
- Algorithms identical in 2D and 3D space

Line Segment Clipping

Cohen-Sutherland Clipping

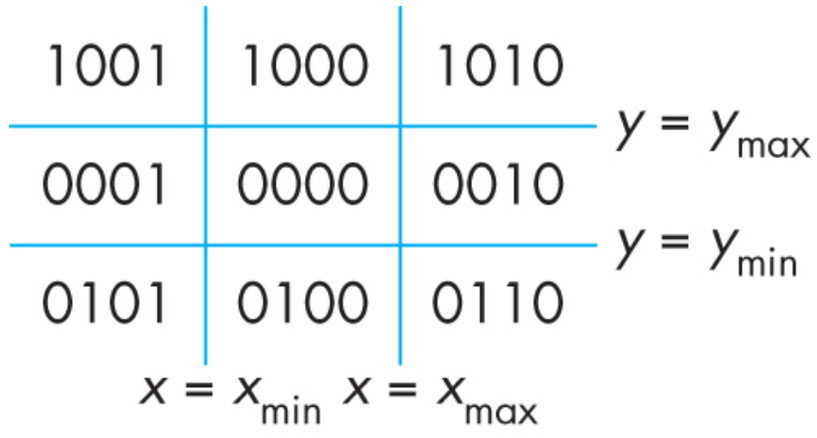
- In 2D, four possible cases w.r.t. the clipping window:

1. Both end-points inside
2. One end-point inside, other outside
3. Both end-points outside
   I. Segment outside
   II. Segment partially inside



Cohen-Sutherland Clipping

- Combines floating-point subtractions and bit operations
- Extend side of the window to infinity and partition space into nine regions each with its own unique 4-bit binary number

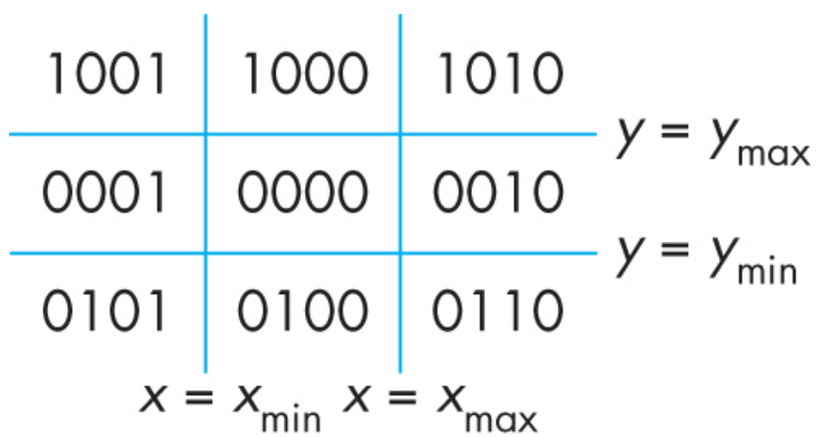


Cohen-Sutherland Clipping

- The bits in the **outcode**, $b_0b_1b_2b_3$ are given by:

$$b_0 = \begin{cases} 1 & \text{if } y > y_{max}, \\ 0 & \text{otherwise} \end{cases}$$

and similarly for b_1, b_2, and b_3.



Cohen-Sutherland Clipping

- Consider a line segment with outcodes of end-points: $o_1 = \text{outcode}(x_1, y_1), o_2 = \text{outcode}(x_2, y_2)$
- We now have four cases based on these outcomes

1. $(o_1 = o_2 = 0)$ — Rasterize
2. $(o_1 \neq 0, o_2 = 0) \vee (o_1 = 0, o_2 \neq 0)$ — Clip
3. $(o_1 \wedge o_2) \neq 0)$ — Reject
4. $(o_1 \wedge o_2) = 0$ — More checks

Cohen-Sutherland Clipping

- Works best when out of many line segments, only a few are to be displayed.
- Intersections for case 2) and 4) may be computed using standard equation of line: $y = mx + c$
  - However, one must be careful with vertical lines
  - Can be handled better by using parametric equation:

$$p(t) = p_1 + (p_2 - p_1)t$$

Liang-Barsky Clipping

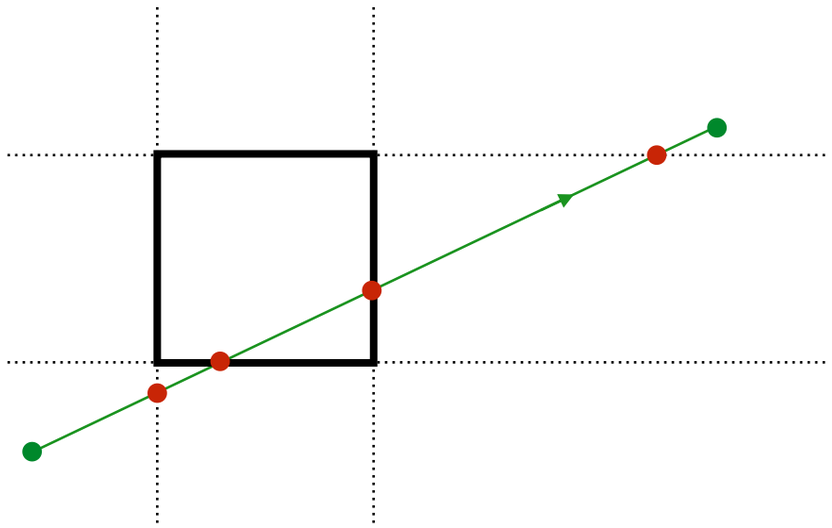
- Uses parametric form for lines

$$p(t) = p_1 + (p_2 - p_1)t$$

- $t \in [0, 1]$: move from p_1 to p_2
- $t < 0$: other side of p_1 away from p_2
- $t > 1$: other side of p_2 away from p_1

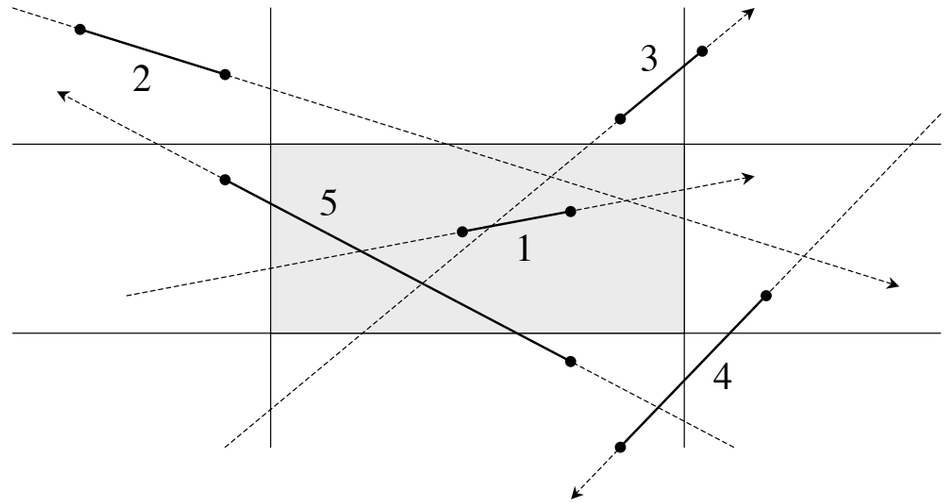
Liang-Barsky Clipping

- Any line segment will intersect the extended clipping box lines at four points
  - Unless the line segment is parallel to one of the sides (resulting in two intersections), or
  - it is coincident to one of the sides (resulting in infinite intersections).



Liang-Barsky Clipping

- For what values of *t* does a line segment enter or exit the bounds?
  - There can be at most two of each. We care about the maximum entry value and minimum exit value
  - For each line segment, check *t*:
    1. If $t_{entry} < 0$ and $t_{exit} > 1$ - *accept*
    2. If $t_{entry} > 1$ or $t_{exit} < 0$ - *reject*
    3. If $t_{entry} > t_{exit}$ - *reject*
    4. Otherwise, clip and try again



Line 1: max entry < 0, min exit > 1 — *accept*
Line 2: max entry > 1, min exit > 1 — *reject*
Line 3: max entry < 0, min exit < 0 — *reject*
Line 4: max entry > min exit — *reject*
Line 5: max entry > 0, min exit < 1, max entry < min exit — *clip*

Liang-Barsky Clipping - Algorithm

1. For a given line segment p_1 to p_2, derive parametric equation: $p(t) = p_1 + (p_2 - p_1)t$

2. For each boundary (L, R, T, B), calculate value of t for that line and that boundary

   a. Substituting in the parametric form, let:

$$\Delta x = x_2 - x_1, \Delta y = y_2 - y_1$$

$$L \leq x_1 + t\Delta x \leq R, \text{and } B \leq y_1 + t\Delta y \leq T$$

Liang-Barsky Clipping - Algorithm

b. The inequalities can be broken up as:

$$-\Delta xt \leq x_1 - L, \qquad \text{let } C = -\Delta x, q = x_1 - L$$

$$\Delta xt \leq R - x_1, \qquad \text{let } C = \Delta x, q = R - x_1$$

$$-\Delta yt \leq y_1 - B, \qquad \text{let } C = -\Delta y, q = y_1 - B$$

$$\Delta yt \leq T - y_1, \qquad \text{let } C = \Delta y, q = T - y_1$$

c. Now, for each C and its corresponding boundary:

$C < 0 \implies$ line goes out $\rightarrow$ in: *entry*

$C > 0 \implies$ line goes in $\rightarrow$ out: *exit*

$C = 0 \implies$ line is parallel to the boundary

Liang-Barsky Clipping - Algorithm

d. So, we can calculate t for each boundary by calculating q and C

- The value of C tells us if we're looking at an entry or exit point for the boundary

3. Apply conditions for accept, reject, or clip.

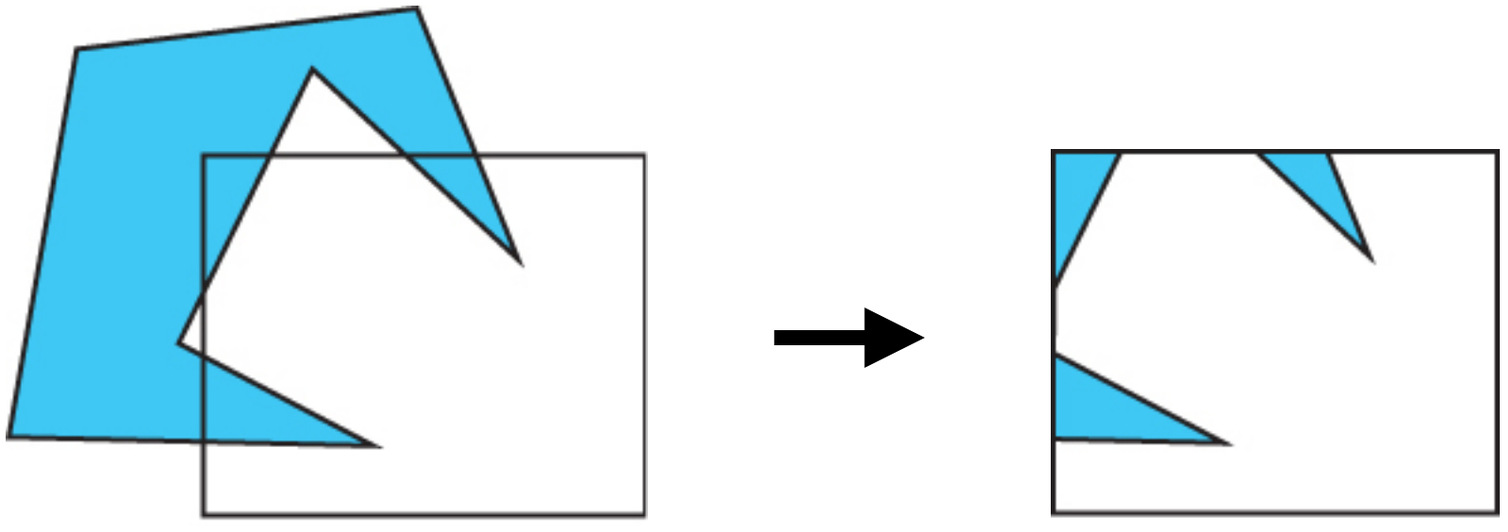
Polygon Clipping

Sutherland-Hodgman Polygon Clipping

- Polygon clipping can be build on top of line clipping algorithms
- Divide and Conquer strategy:

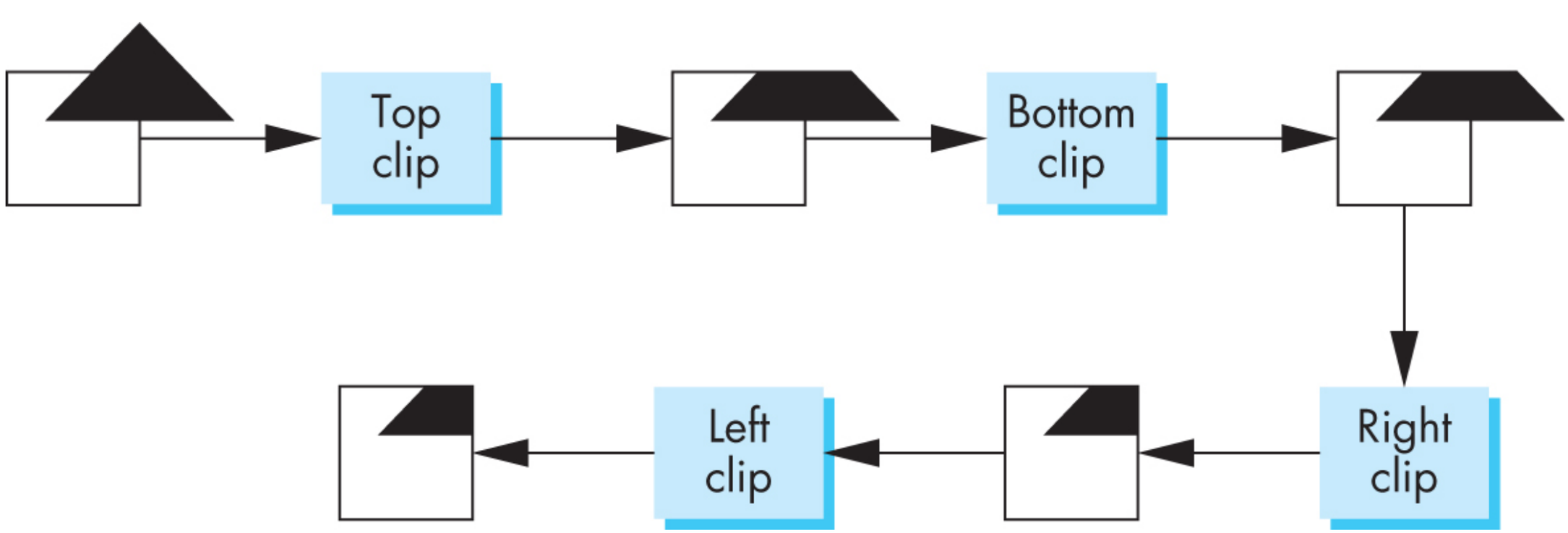
Clip each polygon edge against a clip edge and re-enter algorithm clipping the clipped polygon against next clip edge

…



Sutherland-Hodgman Polygon Clipping

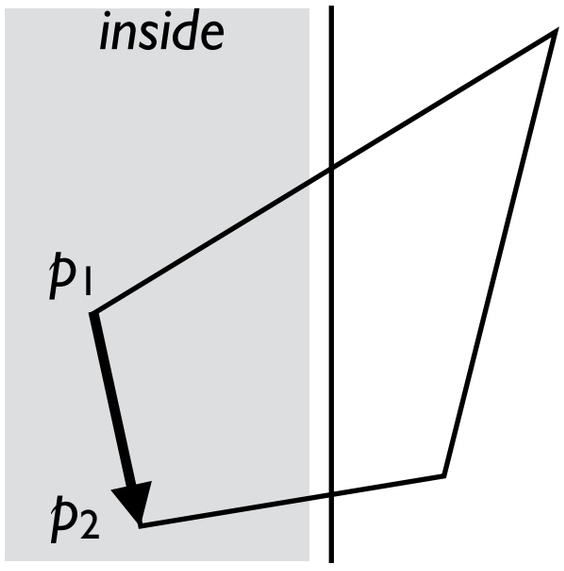
- Therefore, clipping can be performed in a pipeline using a re-entrant clipper



Sutherland-Hodgman Polygon Clipping

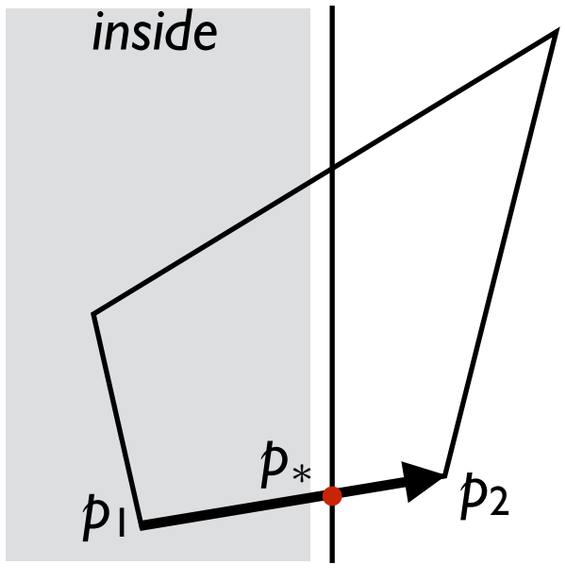
- Four cases in clipping arise:

Case I



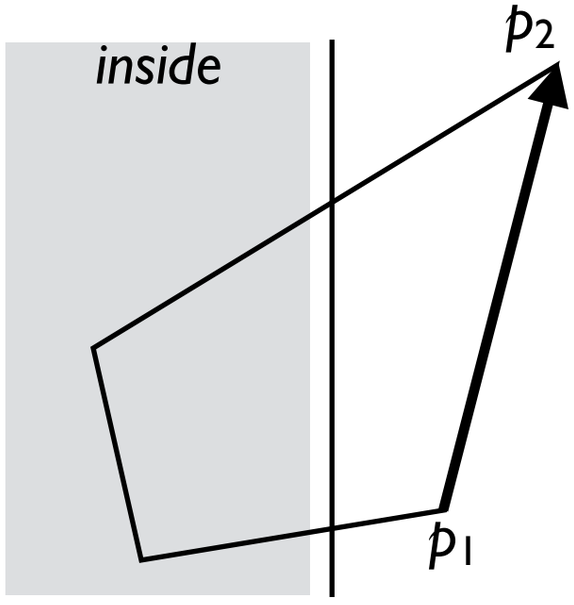
Add $<p_1,p_2>$

Case II

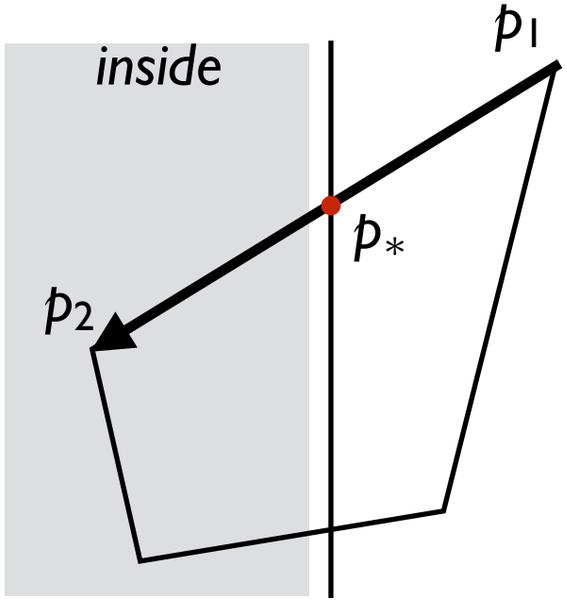


Add $<p_1,p_*>$

Case III



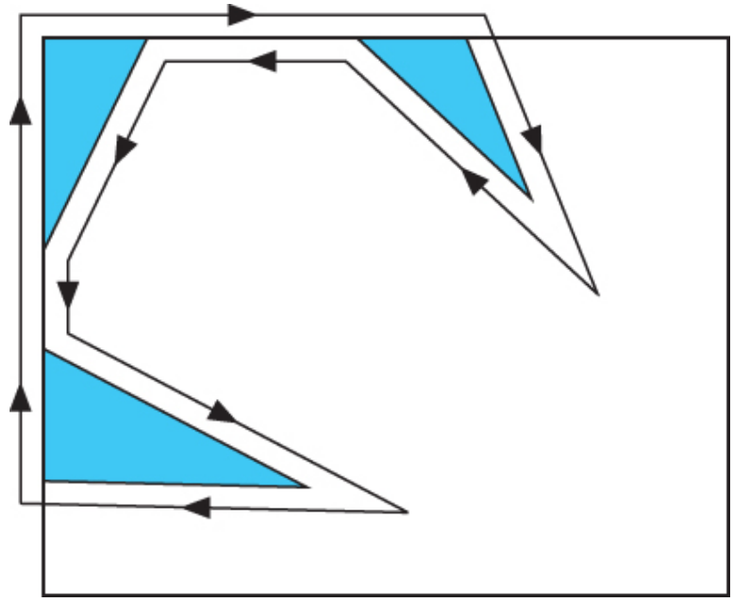
Case IV



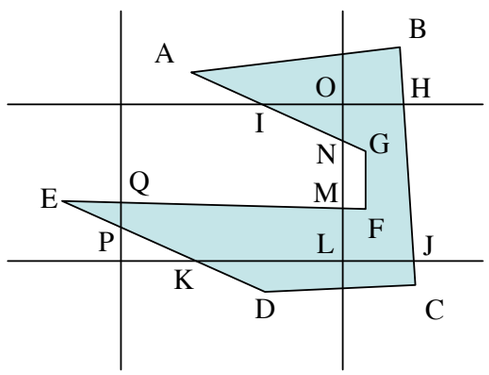
Add $<p_*,p_2>$

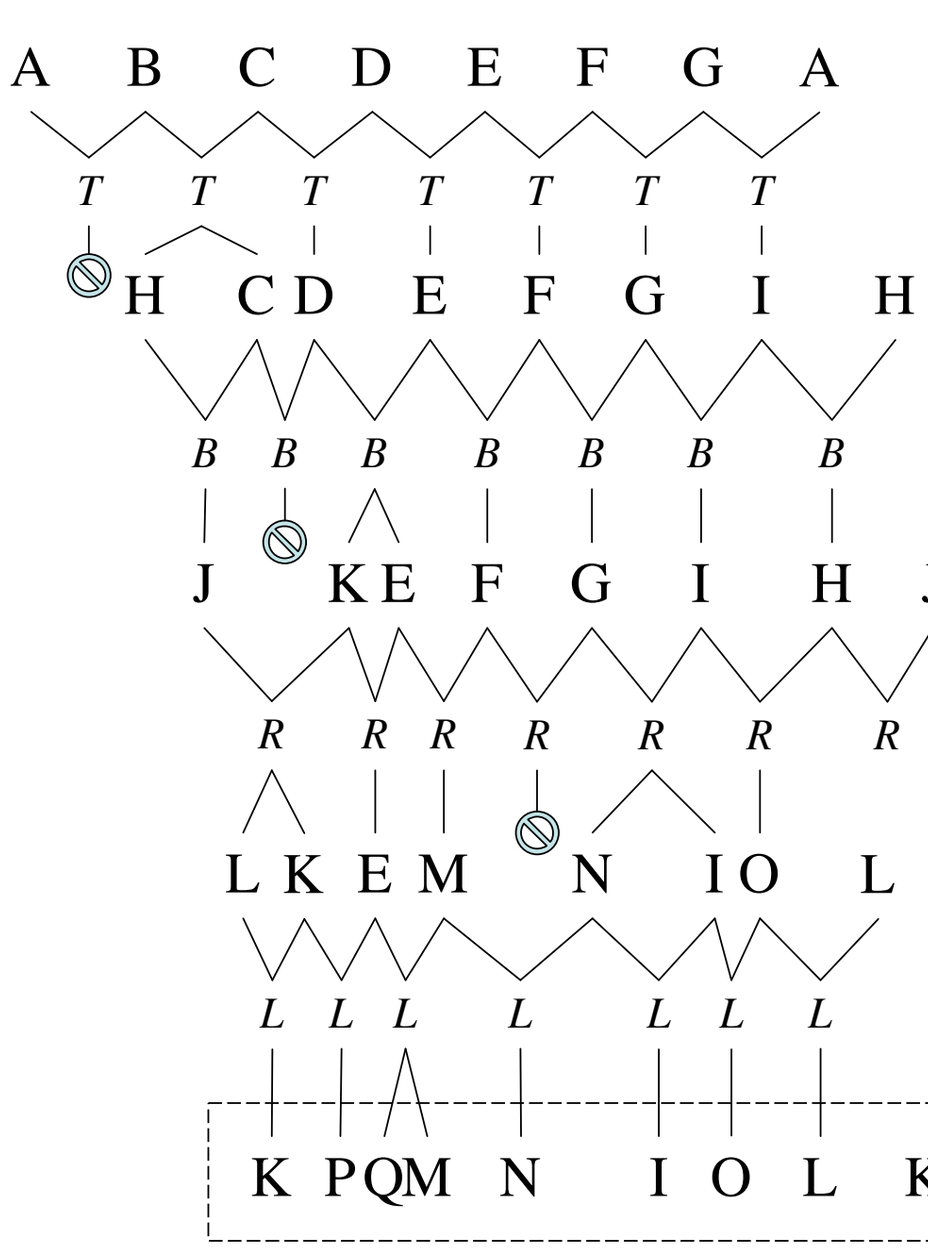
Sutherland-Hodgman Polygon Clipping

- A convex polygon will produce one polygon after clipping
- A concave polygon, can however result in several disconnected polygons
  - Difficult to implement a clipper that can increase number of objects
  - We could treat result of a clipper as a single polygon



Sutherland-Hodgman Polygon Clipping





Reading

- ICG: 6 (Notes on clipping)
- FCG: 8.1.3

ICG: Interactive Computer Graphics, E. Angel, and D. Shreiner, 6th ed.
FCG: Fundamentals of Computer Graphics, P. Shirley, M. Ashikhmin, and S. Marschner, 3rd ed.