

Computer Graphics

Prof. Feng Liu

Fall 2016

<http://www.cs.pdx.edu/~fliu/courses/cs447/>

10/24/2016

Last time

☐ Graphics Pipeline

Today

- Clipping

- In-class Middle-Term

 - Wednesday, Nov. 2

 - Close-book exam

 - Notes on 1 page of A4 or Letter size paper

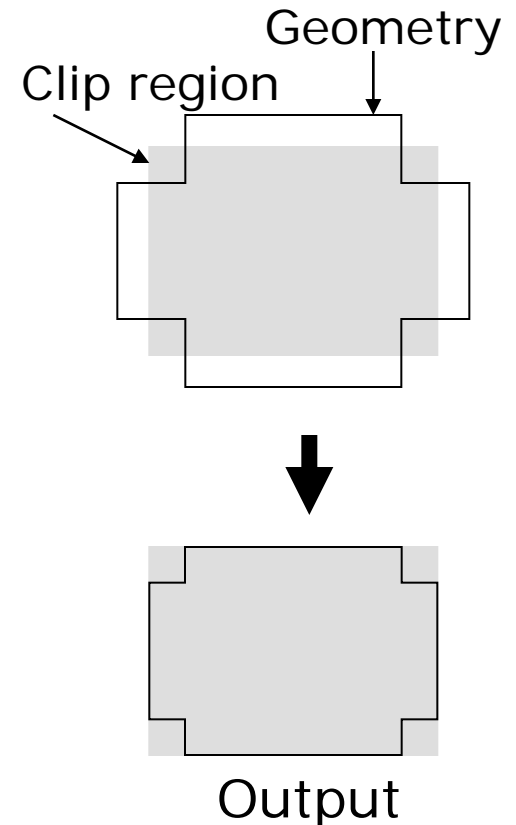
 - To-know list available online

Clipping

- Parts of the geometry to be rendered may lie outside the view volume
- *Clipping* removes parts of the geometry that are outside the view
- Best done in canonical space *before perspective divide*
 - Before dividing out the homogeneous coordinate

Clipping Terminology

- Clip region: the region we wish to restrict the output to
- Geometry: the thing we are clipping
 - Only those parts of the geometry that lie inside the clip region will be output
- Clipping edge/plane: an infinite line or plane and we want to output only the geometry on one side of it
 - Frequently, one edge or face of the clip region



Clipping

- ❑ In hardware, clipping is done in canonical space *before perspective divide*
 - Before dividing out the homogeneous coordinate
- ❑ Clipping is useful in many other applications
 - Building BSP trees for visibility and spatial data structures
 - Hidden surface removal algorithms
 - Removing hidden lines in line drawings
 - Finding intersection/union/difference of polygonal regions
 - 2D drawing programs: cropping, arbitrary clipping
- ❑ We will make explicit assumptions about the geometry and the clip region
 - Assumption depend on the algorithm

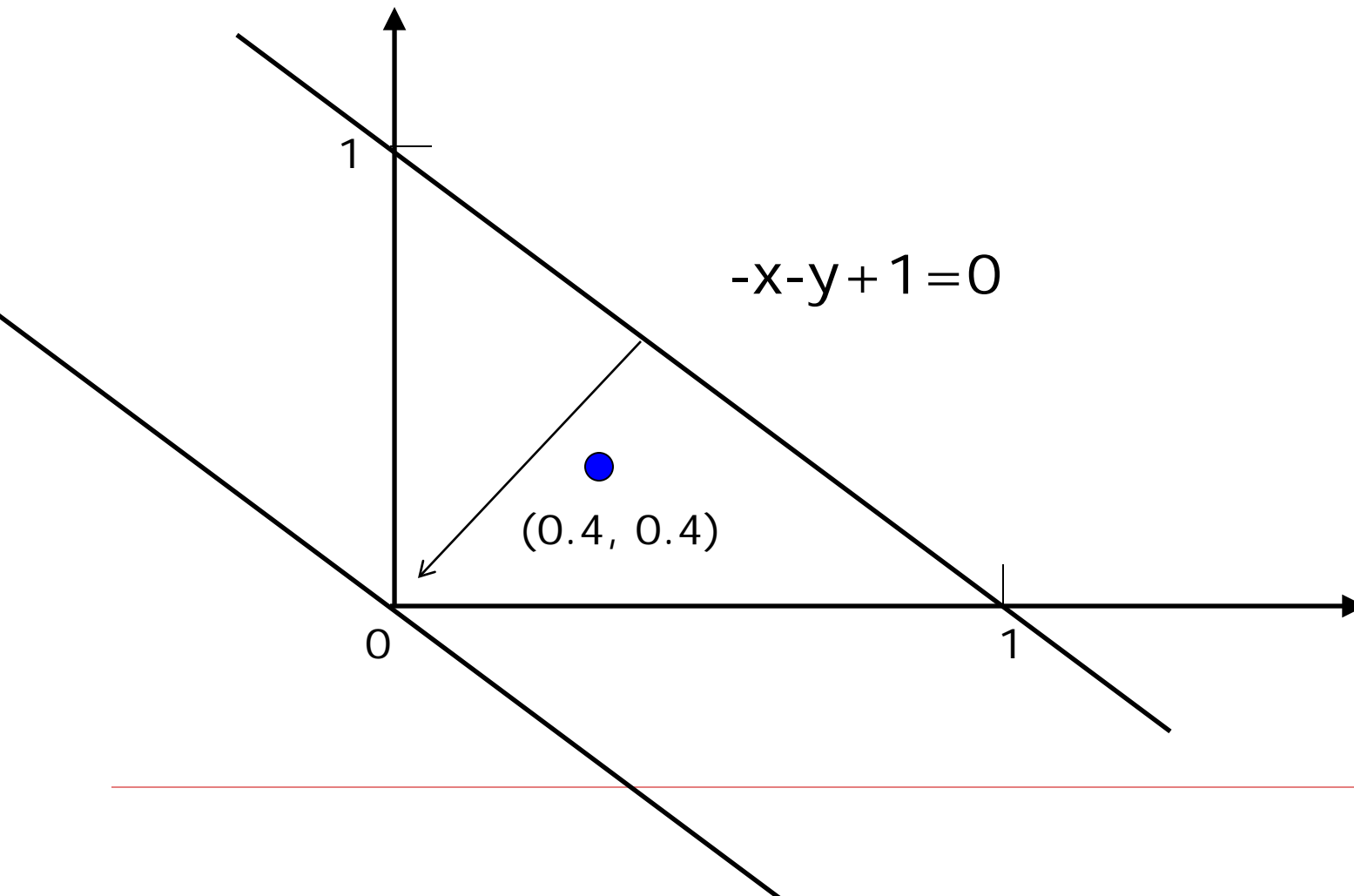
Types of Geometry

- *Points* are clipped via inside/outside tests
 - Many algorithms for this task, depending on the clip region
- Two main algorithms for clipping polygons exist
 - Sutherland-Hodgman
 - Weiler that we will not talk about in our class

Clipping Points to View Volume

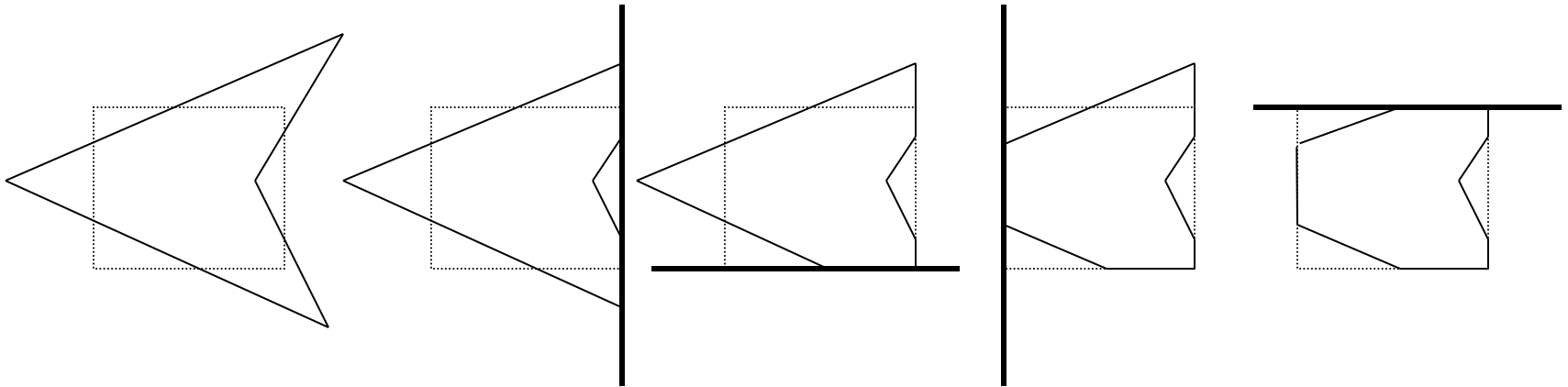
- A point is inside the view volume if it is on the “inside” of all the clipping planes
 - The normals to the clip planes are considered to point inward, toward the visible region
- Now we see why clipping is done in canonical view space
 - For instance, to check against the left plane:
 - X coordinate in 3D must be > -1
 - In homogeneous screen space, same as: $x_{screen} > -w_{screen}$
- In general, a point, p , is “inside” a plane if:
 - You represent the plane as $n_x x + n_y y + n_z z + d = 0$, with (n_x, n_y, n_z) pointing inward
 - And $n_x p_x + n_y p_y + n_z p_z + d > 0$

Clipping Point to Line



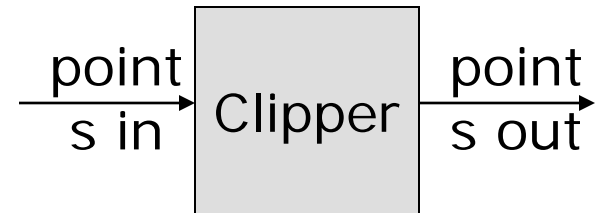
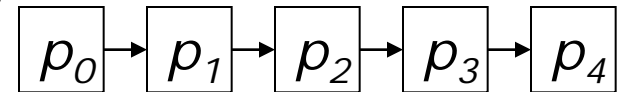
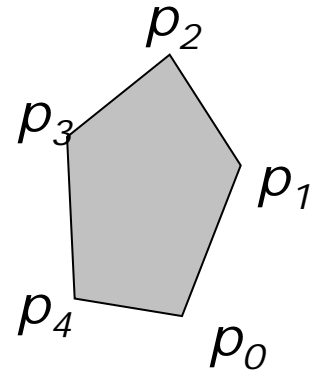
Sutherland-Hodgman Clip

- Clip polygons to convex clip regions
- Clip the polygon against each edge of the clip region in turn
 - Clip polygon each time to line containing edge
 - Only works for convex clip regions (Why? Example that breaks?)

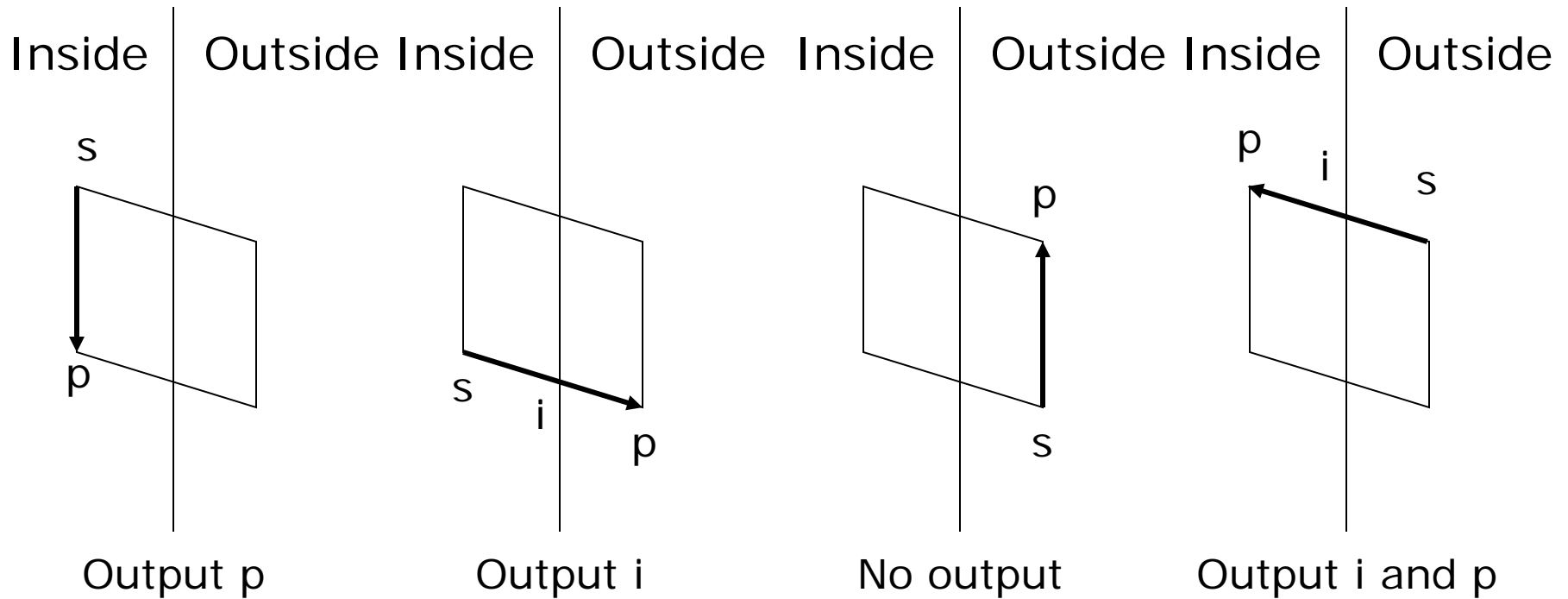


Sutherland-Hodgman Clip (2)

- To clip a polygon to a line/plane:
 - Consider the polygon as a list of vertices
 - One side of the line/plane is considered inside the clip region, the other side is outside
 - We are going to rewrite the polygon one vertex at a time - the rewritten polygon will be the polygon clipped to the line/plane
 - Check start vertex: if “inside”, *emit* it, otherwise ignore it
 - Continue processing vertices as follows...



Sutherland-Hodgman (3)



Sutherland-Hodgman (4)

- Look at the next vertex in the list, p , and the edge from the last vertex, s , to p . If the...
 - polygon edge crosses the clip line/plane going from out to in: emit crossing point, i , next vertex, p
 - polygon edge crosses clip line/plane going from in to out: emit crossing, i
 - polygon edge goes from out to out: emit nothing
 - polygon edge goes from in to in: emit next vertex, p

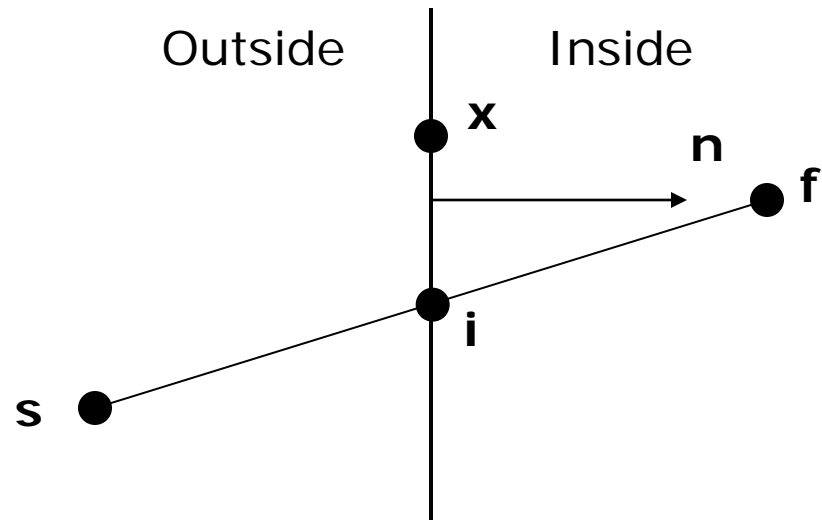
Inside-Outside Testing

- Lines/planes store a vector pointing toward the inside of the clip region - the inward pointing normal
 - Could re-define for outward pointing
- Dot products give inside/outside information
- Note that \mathbf{x} (a vector) is any point on the clip line/plane

$$\mathbf{n} \bullet (\mathbf{s} - \mathbf{x}) < 0$$

$$\mathbf{n} \bullet (\mathbf{i} - \mathbf{x}) = 0$$

$$\mathbf{n} \bullet (\mathbf{f} - \mathbf{x}) > 0$$



Finding Intersection Pts

- Use the parametric form for the edge between two points, \mathbf{x}_1 and \mathbf{x}_2 :

$$\mathbf{x}(t) = \mathbf{x}_1 + (\mathbf{x}_2 - \mathbf{x}_1)t \quad 0 \leq t \leq 1$$

- For planes of the form $x=a$:

$$\mathbf{x}_i = (a, y_1 + \frac{(y_2 - y_1)}{(x_2 - x_1)}(a - x_1), z_1 + \frac{(z_2 - z_1)}{(x_2 - x_1)}(a - x_1))$$

- Similar forms for $y=a$, $z=a$
 - Solution for general plane can also be found
-

Inside/Outside in Screen Space

- In canonical view space, clip planes are $x_s=\pm 1$, $y_s=\pm 1$, $z_s=\pm 1$
- Inside/Outside reduces to comparisons before perspective divide

$$-w_s \leq x_s \leq w_s$$

$$-w_s \leq y_s \leq w_s$$

$$-w_s \leq z_s \leq w_s$$

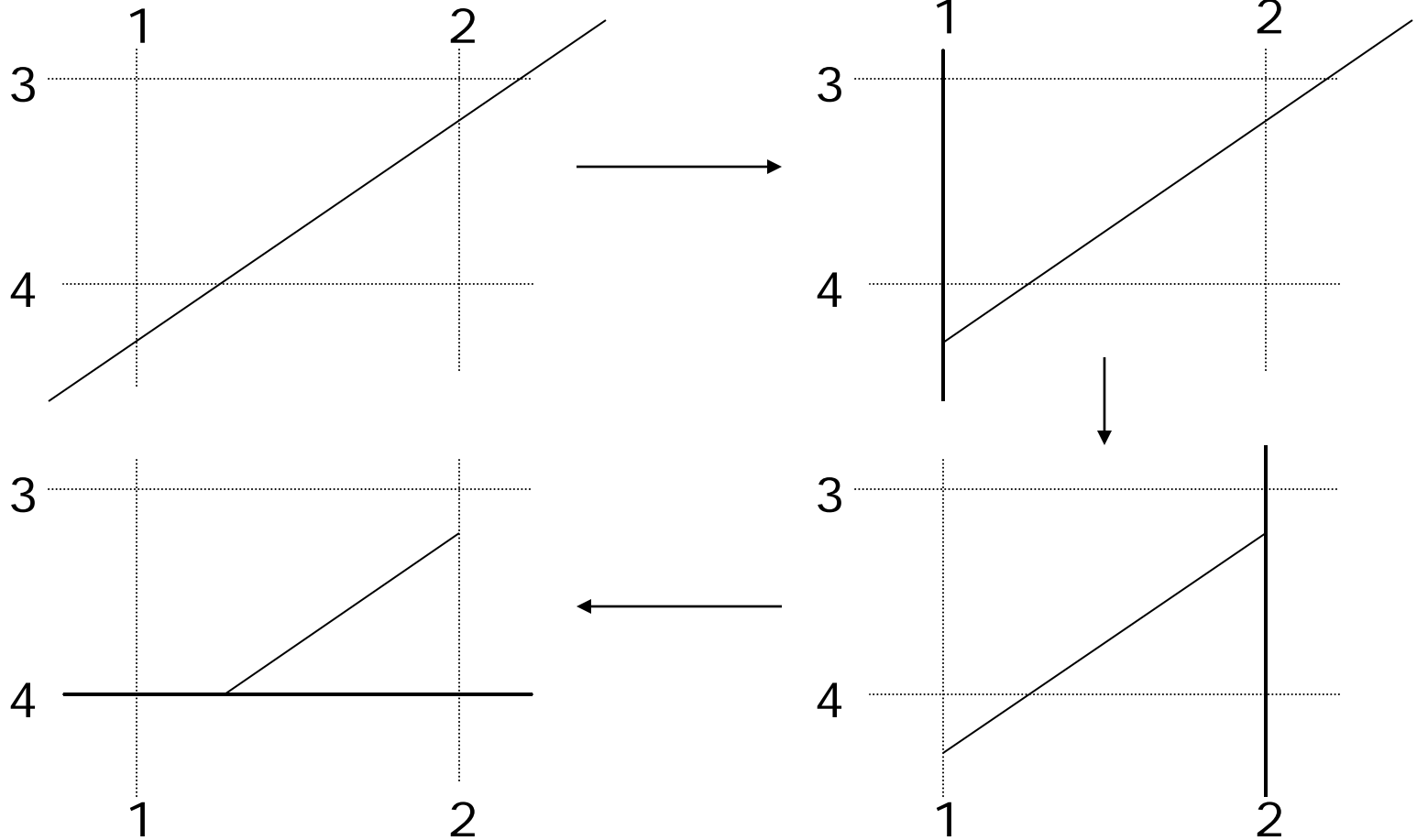
Clipping Lines

- ❑ Lines can also be clipped by Sutherland-Hodgman
 - Slower than necessary, unless you already have hardware
- ❑ Better algorithms exist
 - Cohen-Sutherland
 - Liang-Barsky
 - Nicholl-Lee-Nicholl (we won't cover this one - only good for 2D)

Cohen-Sutherland (1)

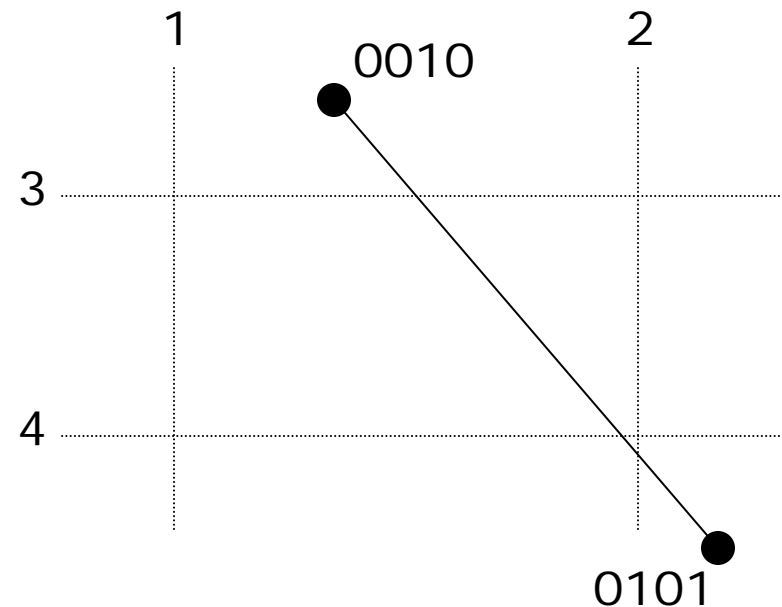
- Works basically the same as Sutherland-Hodgman
 - Was developed earlier
- Clip line against each edge of clip region in turn
 - If both endpoints outside, discard line and stop
 - If both endpoints in, continue to next edge (or finish)
 - If one in, one out, chop line at crossing pt and continue
- Works in both 2D and 3D for convex clipping regions

Cohen-Sutherland (2)



Cohen-Sutherland - Details

- ❑ Only need to clip line against edges where one endpoint is out
- ❑ Use *outcode* to record endpoint in/out wrt each edge. One bit per edge, 1 if out, 0 if in.
- ❑ Trivial reject:
 - $\text{outcode}(x1) \& \text{outcode}(x2) \neq 0$
- ❑ Trivial accept:
 - $\text{outcode}(x1) | \text{outcode}(x2) == 0$
- ❑ Which edges to clip against?
 - $\text{outcode}(x1) \wedge \text{outcode}(x2)$

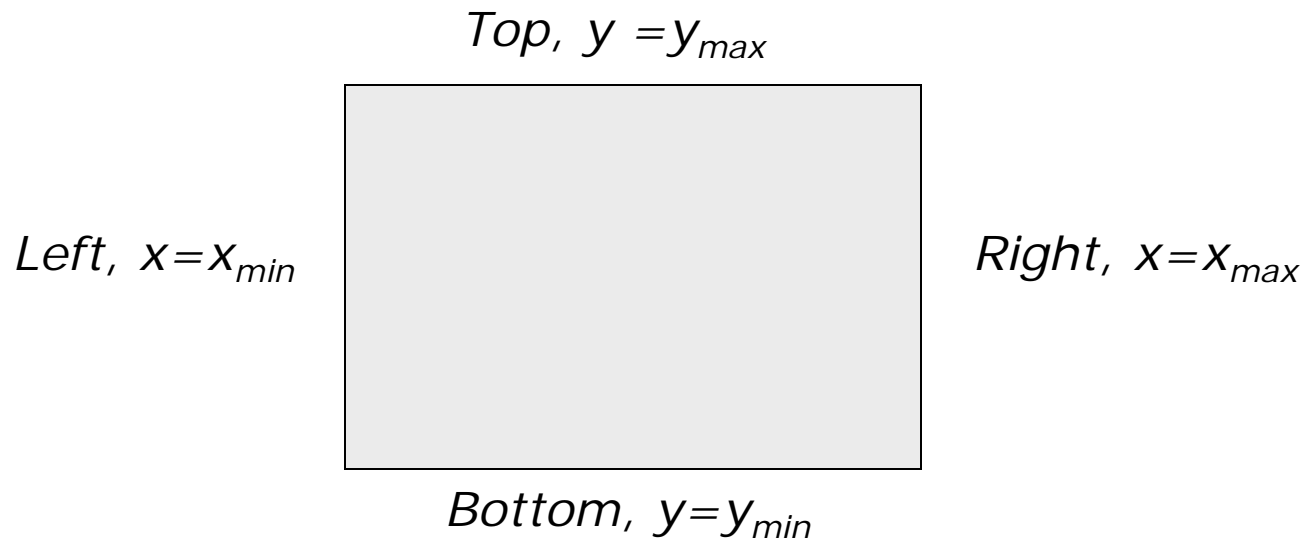


Liang-Barsky Clipping

- Parametric clipping - view line in parametric form and reason about the parameter values
 - Parametric form: $\mathbf{x} = \mathbf{x}_1 + (\mathbf{x}_2 - \mathbf{x}_1)t$
 - $t \in [0, 1]$ are points between \mathbf{x}_1 and \mathbf{x}_2
- Liang-Barsky is more efficient than Cohen-Sutherland
 - Computing intersection vertices is most expensive part of clipping
 - Cohen-Sutherland may compute intersection vertices that are later clipped off, and hence don't contribute to the final answer
- Works for convex clip regions in 2D or 3D

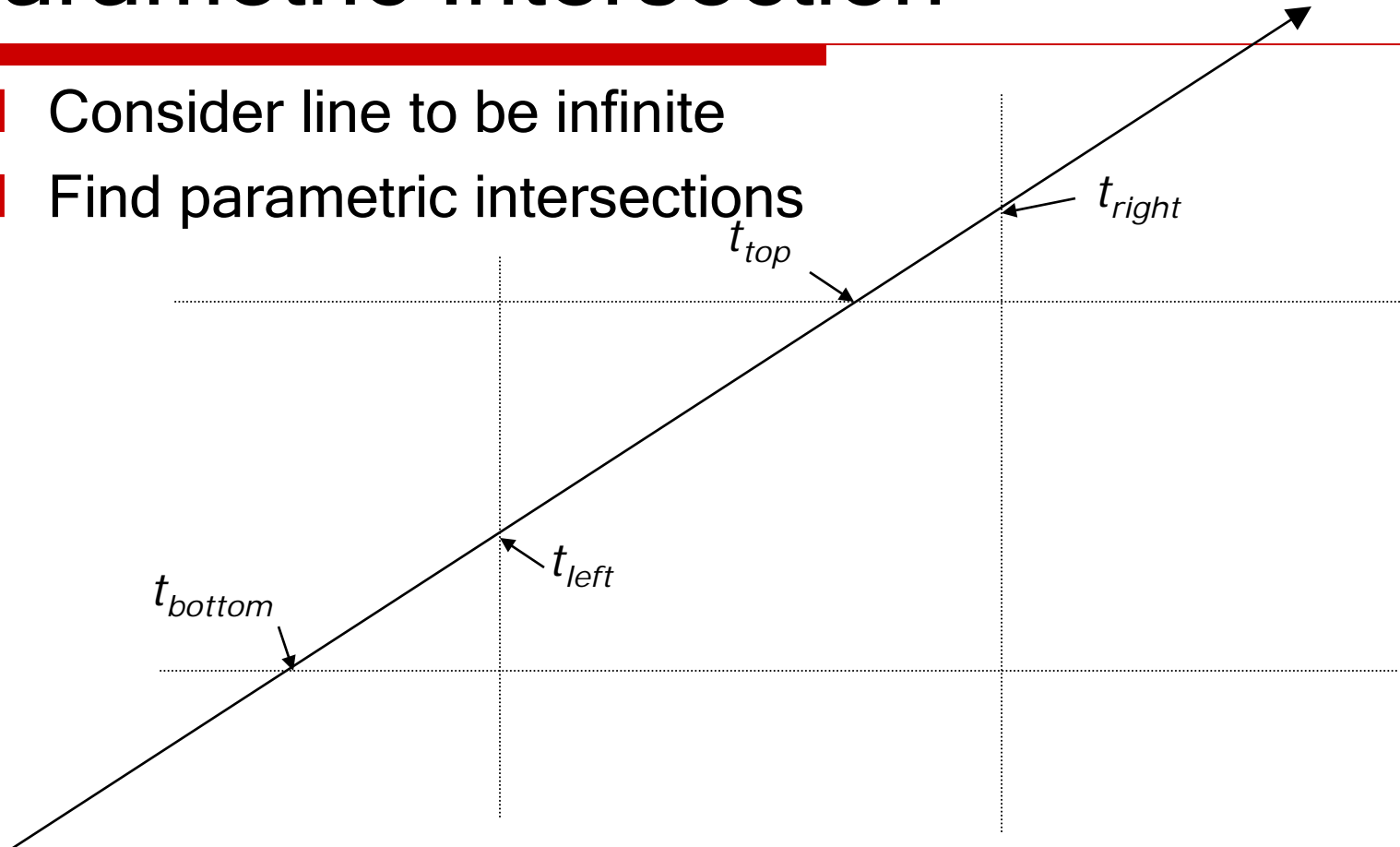
Parametric Clipping

- Recall, points inside a convex region are inside all clip planes
- Parametric clipping finds the values of t , the parameter, that correspond to points inside the clip region
- Consider a rectangular clip region



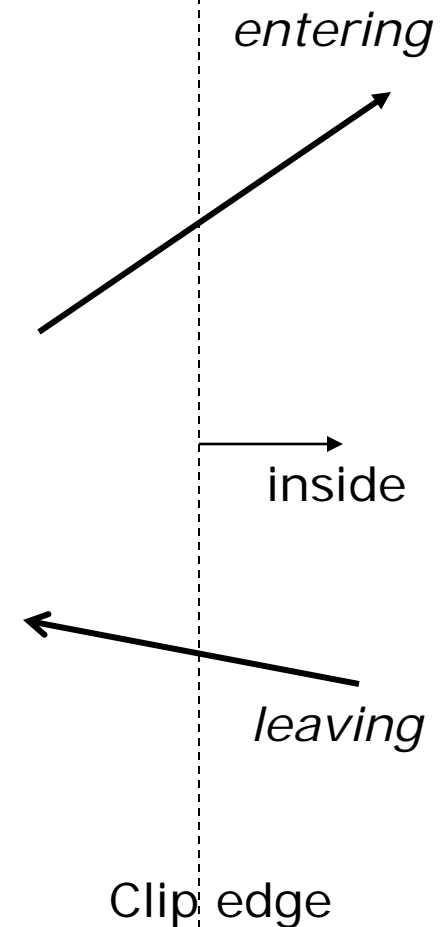
Parametric Intersection

- Consider line to be infinite
- Find parametric intersections



Entering and Leaving

- Recall, a point is inside a view volume if it is on the inside of every clip edge/plane
- Consider the left clip edge and the infinite line. Two cases:
 - $t < t_{left}$ is inside, $t > t_{left}$ is outside → *leaving*
 - $t < t_{left}$ is outside, $t > t_{left}$ is inside → *entering*
- To be inside a clip plane we either:
 - Started inside, and have not left yet
 - Started outside, and have entered



Country	Year	Value
Algeria	2010	0.00
Algeria	2011	0.00
Algeria	2012	0.00
Algeria	2013	0.00
Algeria	2014	0.00
Algeria	2015	0.00
Algeria	2016	0.00
Algeria	2017	0.00
Algeria	2018	0.00
Algeria	2019	0.00
Algeria	2020	0.00
Algeria	2021	0.00
Algeria	2022	0.00
Algeria	2023	0.00
Algeria	2024	0.00
Algeria	2025	0.00
Algeria	2026	0.00
Algeria	2027	0.00
Algeria	2028	0.00
Algeria	2029	0.00
Algeria	2030	0.00
Algeria	2031	0.00
Algeria	2032	0.00
Algeria	2033	0.00
Algeria	2034	0.00
Algeria	2035	0.00
Algeria	2036	0.00
Algeria	2037	0.00
Algeria	2038	0.00
Algeria	2039	0.00
Algeria	2040	0.00
Algeria	2041	0.00
Algeria	2042	0.00
Algeria	2043	0.00
Algeria	2044	0.00
Algeria	2045	0.00
Algeria	2046	0.00
Algeria	2047	0.00
Algeria	2048	0.00
Algeria	2049	0.00
Algeria	2050	0.00
Algeria	2051	0.00
Algeria	2052	0.00
Algeria	2053	0.00
Algeria	2054	0.00
Algeria	2055	0.00
Algeria	2056	0.00
Algeria	2057	0.00
Algeria	2058	0.00
Algeria	2059	0.00
Algeria	2060	0.00
Algeria	2061	0.00
Algeria	2062	0.00
Algeria	2063	0.00
Algeria	2064	0.00
Algeria	2065	0.00
Algeria	2066	0.00
Algeria	2067	0.00
Algeria	2068	0.00
Algeria	2069	0.00
Algeria	2070	0.00
Algeria	2071	0.00
Algeria	2072	0.00
Algeria	2073	0.00
Algeria	2074	0.00
Algeria	2075	0.00
Algeria	2076	0.00
Algeria	2077	0.00
Algeria	2078	0.00
Algeria	2079	0.00
Algeria	2080	0.00
Algeria	2081	0.00
Algeria	2082	0.00
Algeria	2083	0.00
Algeria	2084	0.00
Algeria	2085	0.00
Algeria	2086	0.00
Algeria	2087	0.00
Algeria	2088	0.00
Algeria	2089	0.00
Algeria	2090	0.00
Algeria	2091	0.00
Algeria	2092	0.00
Algeria	2093	0.00
Algeria	2094	0.00
Algeria	2095	0.00
Algeria	2096	0.00
Algeria	2097	0.00
Algeria	2098	0.00
Algeria	2099	0.00
Algeria	2100	0.00
Algeria	2101	0.00
Algeria	2102	0.00
Algeria	2103	0.00
Algeria	2104	0.00
Algeria	2105	0.00
Algeria	2106	0.00
Algeria	2107	0.00
Algeria	2108	0.00
Algeria	2109	0.00
Algeria	2110	0.00
Algeria	2111	0.00
Algeria	2112	0.00
Algeria	2113	0.00
Algeria	2114	0.00
Algeria	2115	0.00
Algeria	2116	0.00
Algeria	2117	0.00
Algeria	2118	0.00
Algeria	2119	0.00
Algeria	2120	0.00
Algeria	2121	0.00
Algeria	2122	

When are we Inside?

- We want parameter values that are inside *all* the clip planes
- Any clip plane that we started inside we must not have left yet
 - First parameter value to leave is the end of the visible segment
- Any clip plane that we started outside we must have already entered
 - Last parameter value to enter is the start of the visible segment
- If we leave some clip plane before we enter another, we cannot see any part of the line
- All this leads to an algorithm - Liang-Barsky

Liang-Barsky Sub-Tasks

1. Find parametric intersection points
 - Parameter values where line crosses each clip edge/plane
2. Find entering/leaving flags
 - For every clip edge/plane, are either entering or leaving
3. Find last parameter to enter, and first one to leave
 - Check that enter before leave
4. Convert these into endpoints of clipped segment

1. Parametric Intersection

- Segment goes from (x_1, y_1) to (x_2, y_2) :
 $\Delta x = x_2 - x_1$
 $\Delta y = y_2 - y_1$
- Rectangular clip region with x_{\min} , x_{\max} , y_{\min} , y_{\max}
- **Infinite** line intersects **rectangular** clip region edges when:

$$t_k = \frac{q_k}{p_k} \quad \text{where} \quad \begin{array}{ll} p_{\text{left}} = -\Delta x & q_{\text{left}} = x_1 - x_{\min} \\ p_{\text{right}} = \Delta x & q_{\text{right}} = x_{\max} - x_1 \\ p_{\text{bottom}} = -\Delta y & q_{\text{bottom}} = y_1 - y_{\min} \\ p_{\text{top}} = \Delta y & q_{\text{top}} = y_{\max} - y_1 \end{array}$$

2. Entering or Leaving?

- When $p_k < 0$, as t increases line goes from outside to inside - entering
 - $p_{left} = -\Delta x$
 - $p_{right} = \Delta x$
- When $p_k > 0$, line goes from inside to outside - leaving
 - $p_{bottom} = -\Delta y$
 - $p_{top} = \Delta y$
- When $p_k = 0$, line is parallel to an edge
 - Special case: one endpoint outside, no part of segment visible, otherwise, ignore this clip edge and continue

Find Visible Segment t s

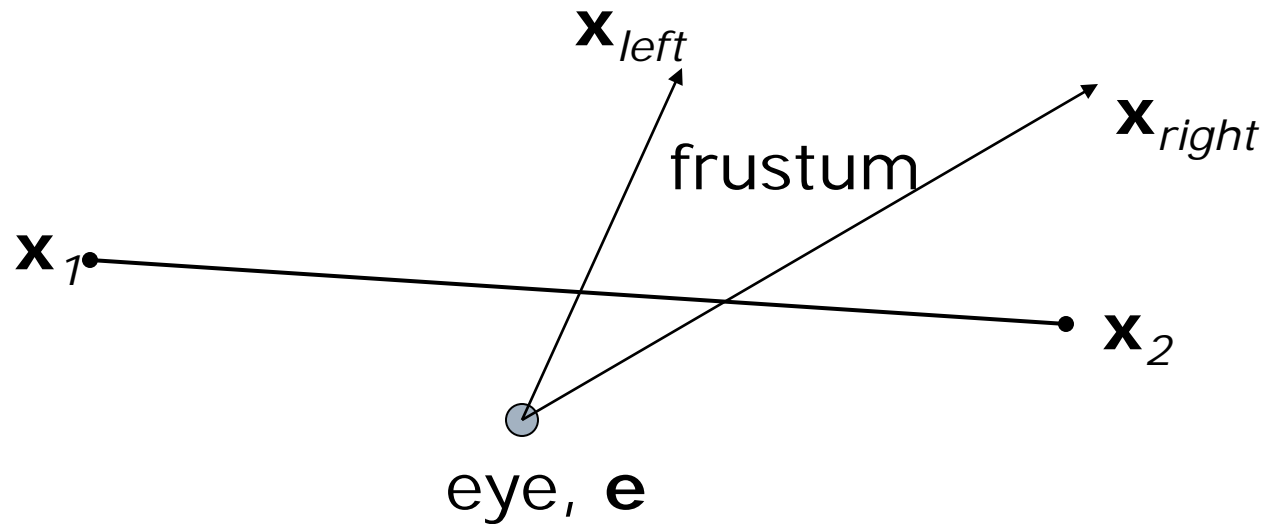
- Last parameter is enter is $t_{small} = \max(0, \text{entering } t\text{'s})$
- First parameter is leave is $t_{large} = \min(1, \text{leaving } t\text{'s})$
- If $t_{small} > t_{large}$, there is no visible segment
- If $t_{small} < t_{large}$, there is a line segment
 - Compute endpoints by substituting t values into parametric equation for the line segment
- Improvement (and actual Liang-Barsky):
 - compute t 's for each edge in turn (some rejects occur earlier like this)

Country	Year	Value
Algeria	2000	0.0000
Algeria	2001	0.0000
Algeria	2002	0.0000
Algeria	2003	0.0000
Algeria	2004	0.0000
Algeria	2005	0.0000
Algeria	2006	0.0000
Algeria	2007	0.0000
Algeria	2008	0.0000
Algeria	2009	0.0000
Algeria	2010	0.0000
Algeria	2011	0.0000
Algeria	2012	0.0000
Algeria	2013	0.0000
Algeria	2014	0.0000
Algeria	2015	0.0000
Algeria	2016	0.0000
Algeria	2017	0.0000
Algeria	2018	0.0000
Algeria	2019	0.0000
Algeria	2020	0.0000
Algeria	2021	0.0000
Algeria	2022	0.0000
Algeria	2023	0.0000
Algeria	2024	0.0000
Algeria	2025	0.0000
Algeria	2026	0.0000
Algeria	2027	0.0000
Algeria	2028	0.0000
Algeria	2029	0.0000
Algeria	2030	0.0000
Algeria	2031	0.0000
Algeria	2032	0.0000
Algeria	2033	0.0000
Algeria	2034	0.0000
Algeria	2035	0.0000
Algeria	2036	0.0000
Algeria	2037	0.0000
Algeria	2038	0.0000
Algeria	2039	0.0000
Algeria	2040	0.0000
Algeria	2041	0.0000
Algeria	2042	0.0000
Algeria	2043	0.0000
Algeria	2044	0.0000
Algeria	2045	0.0000
Algeria	2046	0.0000
Algeria	2047	0.0000
Algeria	2048	0.0000
Algeria	2049	0.0000
Algeria	2050	0.0000
Algeria	2051	0.0000
Algeria	2052	0.0000
Algeria	2053	0.0000
Algeria	2054	0.0000
Algeria	2055	0.0000
Algeria	2056	0.0000
Algeria	2057	0.0000
Algeria	2058	0.0000
Algeria	2059	0.0000
Algeria	2060	0.0000
Algeria	2061	0.0000
Algeria	2062	0.0000
Algeria	2063	0.0000
Algeria	2064	0.0000
Algeria	2065	0.0000
Algeria	2066	0.0000
Algeria	2067	0.0000
Algeria	2068	0.0000
Algeria	2069	0.0000
Algeria	2070	0.0000
Algeria	2071	0.0000
Algeria	2072	0.0000
Algeria	2073	0.0000
Algeria	2074	0.0000
Algeria	2075	0.0000
Algeria	2076	0.0000
Algeria	2077	0.0000
Algeria	2078	0.0000
Algeria	2079	0.0000
Algeria	2080	0.0000
Algeria	2081	0.0000
Algeria	2082	0.0000
Algeria	2083	0.0000
Algeria	2084	0.0000
Algeria	2085	0.0000
Algeria	2086	0.0000
Algeria	2087	0.0000
Algeria	2088	0.0000
Algeria	2089	0.0000
Algeria	2090	0.0000
Algeria	2091	0.0000
Algeria	2092	0.0000
Algeria	2093	0.0000
Algeria	2094	0.0000
Algeria	2095	0.0000
Algeria	2096	0.0000
Algeria	2097	0.0000
Algeria	2098	0.0000
Algeria	2099	0.0000
Algeria	2100	0.0000
Algeria	2101	0.0000
Algeria	2102	0.0000
Algeria	2103	0.0000
Algeria	2104	0.0000
Algeria	2105	0.0000
Algeria	2106	

- _____



In View Space



First Step

- Compute inside/outside for endpoints of the line segment
 - Determine which side of each clip plane the segment endpoints lie
 - Use the cross product
 - What do we know if $(\mathbf{x}_1 - \mathbf{e}) \times (\mathbf{x}_{left} - \mathbf{e}) > 0$?
 - Other cross products give other information
- What can we say if both segment endpoints are **outside** one clip plane?
 - Stop here if we can, otherwise...

Finding Parametric Intersection

□ Left clip edge: $\mathbf{x} = \mathbf{e} + (\mathbf{x}_{left} - \mathbf{e}) t$

□ Line: $\mathbf{x} = \mathbf{x}_1 + (\mathbf{x}_2 - \mathbf{x}_1) s$

□ Solve simultaneous equations in t and s :

$$\mathbf{e}_x + (\mathbf{x}_{left,x} - \mathbf{e}_x)t = \mathbf{x}_{1,x} + (\mathbf{x}_{2,x} - \mathbf{x}_{1,x})s$$

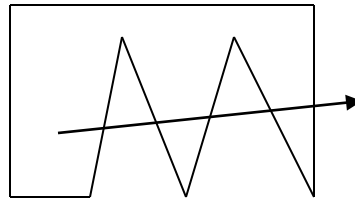
$$\mathbf{e}_y + (\mathbf{x}_{left,y} - \mathbf{e}_y)t = \mathbf{x}_{1,y} + (\mathbf{x}_{2,y} - \mathbf{x}_{1,y})s$$

□ Use endpoint inside/outside information to label as entering or leaving

□ Now we have general Liang-Barsky case

General Clipping

- Liang-Barsky can be generalized to clip line segments to arbitrary polygonal clip regions
 - Consider clip edges as non-infinite segments
 - Look at all intersecting t s between 0 and 1
- Clipping general polygons against general clip regions is quite hard: Weiler-Atherton algorithm



Next Time

- ☐ Rasterization

