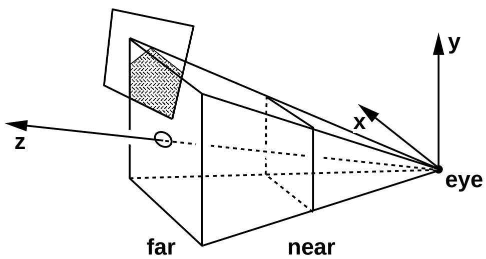
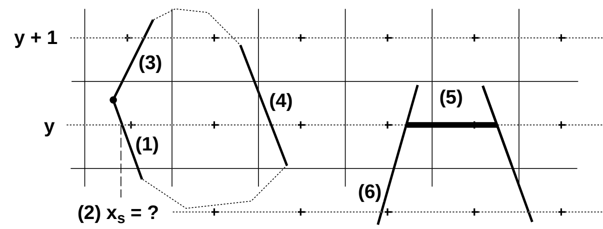
Lecture 9: 7 October 1999

Reminder:
  Asst 4 (scene modeling) due Friday 5pm
  Asst 4 Exhibition next Tuesday in class
Today:
  Recap of Scan-Conversion
  Demo of Ass't 5 `ivscan` (start early!)
  3D Clipping (H&B §6.8)
  Points, segments, convex polys, general polys
  Clipping optimizations
Next week:
  Ray Casting, Ray Tracing
3D Clipping:
  Remove portions of primitives outside frustum



Scan Conversion Recap

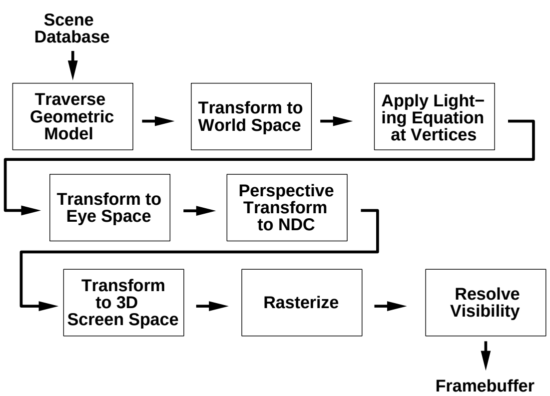
Initialize **ET** (list of scheduled events)
Initialize **AEL** to empty (no intersection)



Sweep scan-line through discrete y values $(h-1)..0$:
  (1) Insert to AEL edges that begin above scan-line
    (2) Initialize x_s, etc. using y, $\frac{dx}{dy}$, edge params
  (3) Delete from AEL edges that end above scan-line
  (4) Recall that edges occur in pairs, so...
    (5) Each matching edge pair yields one span
    Output spans for each active edge pair
    Resolve visibility with z-buffer or ASL
  For each edge in AEL
    (6) Update x, color, depth for this edge
Assignment 4 (`ivscan`) demo, Damian

Pipeline Overview

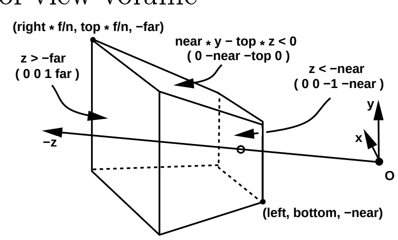
Recall classical rendering pipeline



So far, assumed polygon falls entirely within frustum
  If not, we must clip it; but where in pipeline?

3D Clipping

Orient planes so that positive halfspaces contain
  interior of view volume

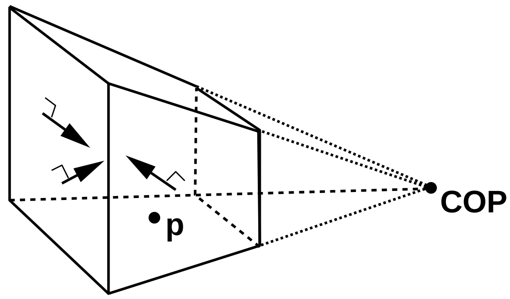


What are plane equations in Eye Space?

$$\mathbf{H}_{near} = \begin{pmatrix} 0 & 0 & -1 & \Box \end{pmatrix}$$
$$\mathbf{H}_{far} = \begin{pmatrix} 0 & 0 & 1 & \Box \end{pmatrix}$$
$$\mathbf{H}_{bottom} = \begin{pmatrix} 0 & \text{near} & \Box & 0 \end{pmatrix}$$
$$\mathbf{H}_{top} = \begin{pmatrix} 0 & -\text{near} & \Box & 0 \end{pmatrix}$$
$$\mathbf{H}_{left} = \begin{pmatrix} \Box & \text{near} & 0 & 0 \end{pmatrix}$$
$$\mathbf{H}_{right} = \begin{pmatrix} \Box & -\text{near} & 0 & 0 \end{pmatrix}$$

3D Clipping

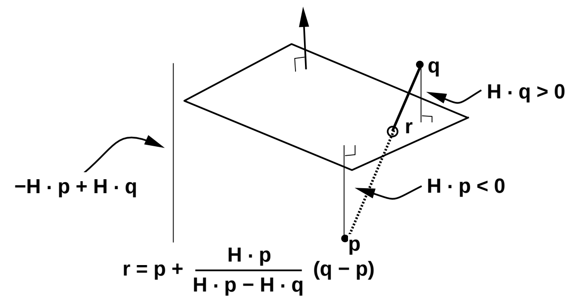
Point "clipping:"



Clip point against a single plane:
 If $\mathbf{Hp} \geq 0$, "pass through"
 If $\mathbf{Hp} < 0$, "clipped out"
Cull points whose signed distance to
 any clipping plane is negative
(Terminology: cull, reject, etc.)

3D Clipping

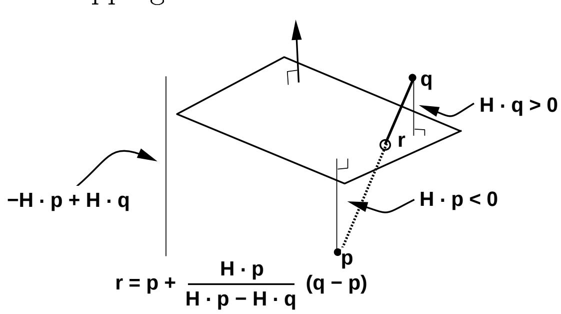
Intersecting a line (segment) with a plane



Find point **r** as a function of **p** and **q**
 Recovering internal points: *Interpolation*
Note that this can be written as $(1-t)\mathbf{p} + t\mathbf{q}$
Parameter t can be used to interpolate attributes
 When does this method fail?

3D Clipping

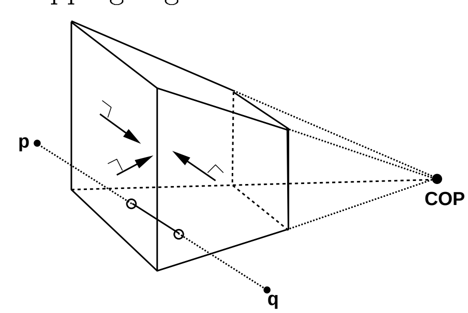
Segment "clipping"



Clip segment against a single plane
 If $\mathbf{Hp} > 0$ and $\mathbf{Hq} > 0$, "pass through"
 If $\mathbf{Hp} < 0$ and $\mathbf{Hq} < 0$, "clipped out"
 If $\mathbf{Hp} < 0$ and $\mathbf{Hq} > 0$, "clip **p** to plane"
 If $\mathbf{Hq} < 0$ and $\mathbf{Hp} > 0$, "clip **q** to plane"
Note *qualitative* contrast with point clipping
 "Triage" – will see it again later

3D Clipping

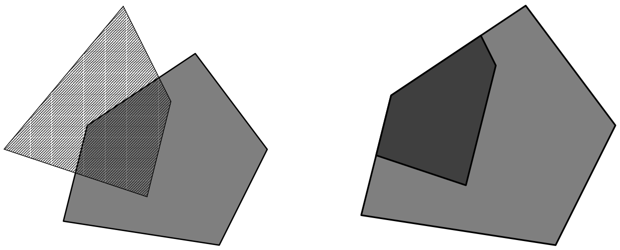
Segment "clipping" against a view frustum



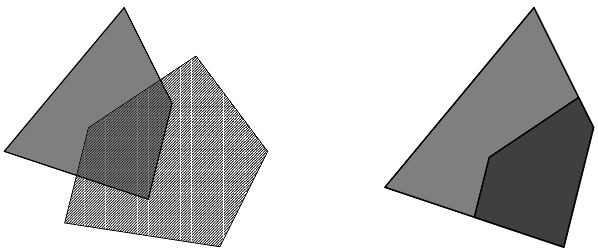
Clip segment against each frustum plane in turn
Implementation:
Clip (point **p**, point **q**, ...)
 For each frustum plane **H**
  If $\mathbf{Hp} \leq 0$ and $\mathbf{Hq} \leq 0$, "clipped out; break"
  If $\mathbf{Hp} \geq 0$ and $\mathbf{Hq} \geq 0$, "pass through"
  If $\mathbf{Hp} < 0$ and $\mathbf{Hq} > 0$, "clip **p** to **H**"
  If $\mathbf{Hq} < 0$ and $\mathbf{Hp} > 0$, "clip **q** to **H**"
Note: must *interpolate* associated attributes
 (color, normal, texture, etc.)

Clipping filled 2D polygons (H&B S6.8)

Clip "subject polygon" to "clip polygon"

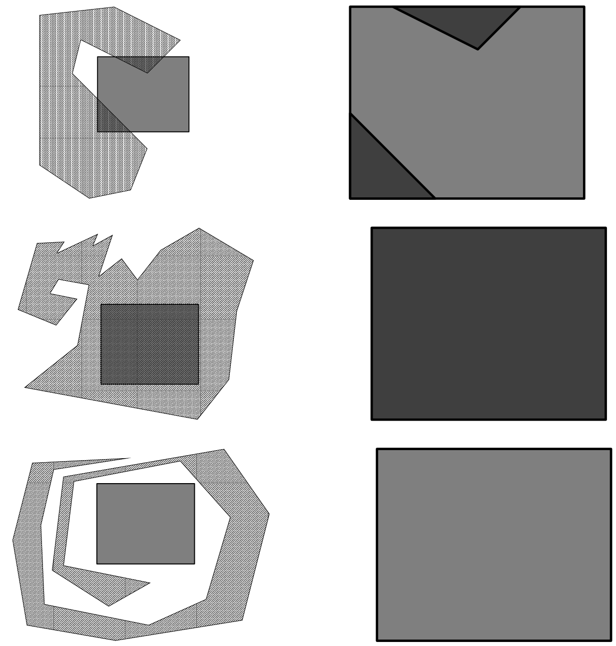


Roles interchangeable; really computing *intersection*:



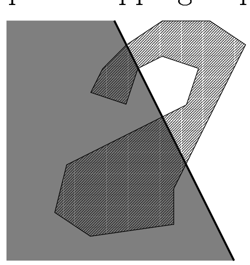
Clipping Challenges

Many complications even for convex clip regions:

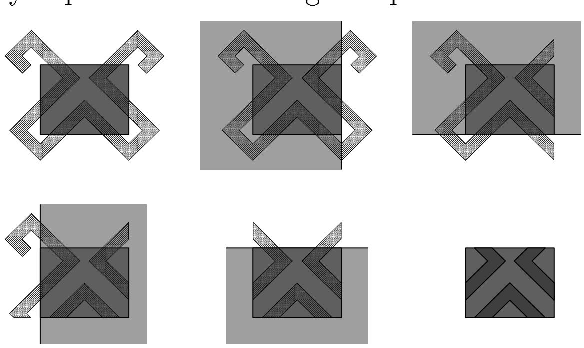


Sutherland-Hodgman Algorithm

Clips subject polygon to any convex polygon
Idea: employ half-plane clipping as primitive:

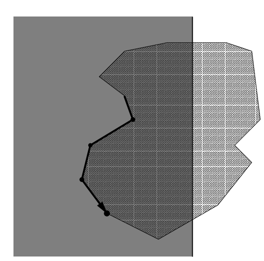


Simply clip to each bounding half-plane in turn:

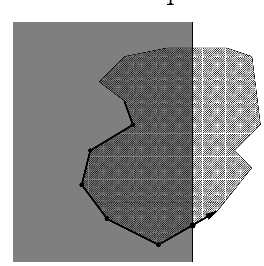


Sutherland-Hodgman Algorithm

Input: ordered list of subject polygon vertices
"Walk" around polygon, processing each edge in turn
Produces output for each edge **AB** (0, 1, or 2 vertices).
1) Edge entirely in half-plane: output **B**

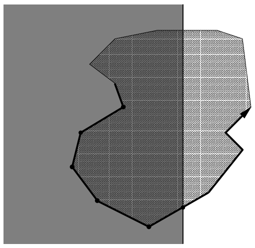


2) **A** inside, **B outside**: output exit point

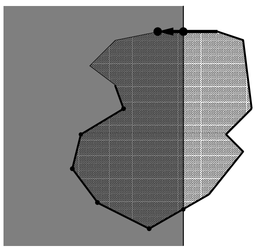


Sutherland-Hodgman Algorithm: Cases

3) **A** outside, **B outside**: output nothing



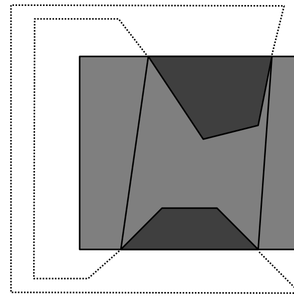
4) **A** outside, **B inside**:
output entry point, then **B** (in order)



Is the algorithm complete and correct?

Sutherland-Hodgman Problem

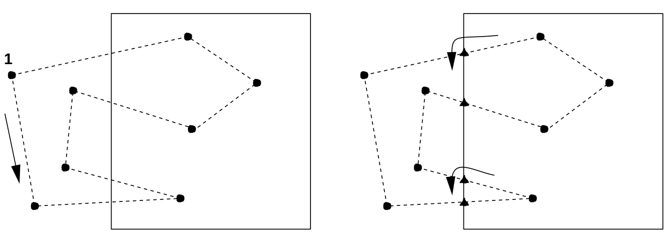
Algorithm cannot represent disconnected output!



Remove by postprocessing, or...
Handle in scan conversion
Implementation note:
How to represent input vertices?
How to represent clipped vertices?

Weiler-Atherton Clipping

Strategy: "Walk" polygon/window bdry together
Follow window boundaries between exit/entry



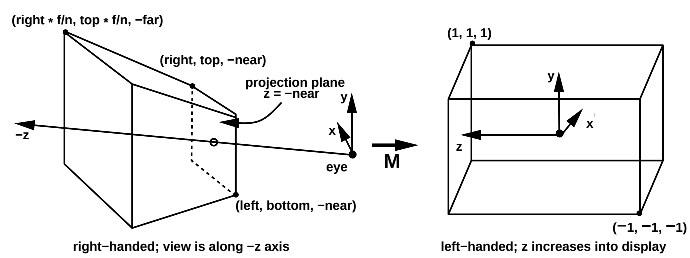
Clipping Rules:
- Out-to-in pair:
  - Record clipped point
  - Follow polygon boundary (ccw)
- In-to-out pair:
  - Record clipped point
  - Follow window boundary (ccw)

Many other clipping algorithms:
Parametric, general windows, region-region, etc.

Achieving robustness, efficiency is non-trivial!

Clipping to Canonical Volume

Transform ES to NDC (X, Y, Z coords), *then* clip



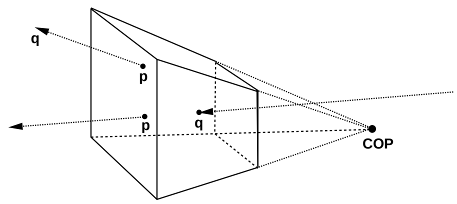
Need to write (and optimize) only *one* clipper !
Plane equations in NDC:

$$z > -1: \quad \mathbf{H}_{near} = \begin{pmatrix} 0 & 0 & 1 & 1 \end{pmatrix}$$
$$z < +1: \quad \mathbf{H}_{far} = \begin{pmatrix} 0 & 0 & -1 & 1 \end{pmatrix}$$
$$y > -1: \quad \mathbf{H}_{bottom} = \begin{pmatrix} 0 & 1 & 0 & 1 \end{pmatrix}$$
$$y < +1: \quad \mathbf{H}_{top} = \begin{pmatrix} 0 & -1 & 0 & 1 \end{pmatrix}$$
$$x > -1: \quad \mathbf{H}_{left} = \begin{pmatrix} 1 & 0 & 0 & 1 \end{pmatrix}$$
$$x < +1: \quad \mathbf{H}_{right} = \begin{pmatrix} -1 & 0 & 0 & 1 \end{pmatrix}$$

Signed distances are just $\mathbf{p}$'s coordinates times ± 1!

3D Clipping – Problem

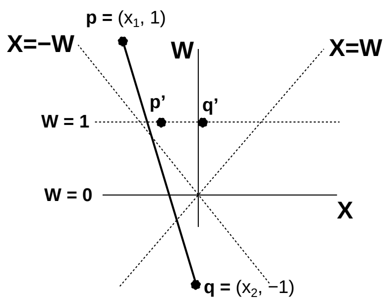
Suppose primitive reaches (or crosses) plane at ∞



Examples:

A line segment with one finite, one infinite point

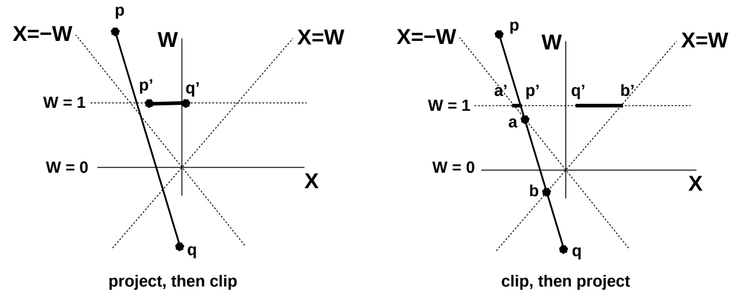
A line segment with one $W > 0$, one $W < 0$



These cases can't be handled after projection

What's going on?

Look again at 1D case



At left:

1. Project **pq** to **p'q'**
2. Clip **p'q'** to $X \in [-1..1]$

But this displays portion of line *outside* **pq**!

At right:

1. Clip to $X, W \in [-1..1]$

   This produces *two* segments, **pa** and **bq**
2. Project these to **p'a'** to **b'q'**

Transformation / Clipping Order

Transform to NDC, Project to X, Y, Z, then Clip
- Simplest scheme (most commonly implemented)
- Efficiency depends on input distribution:
  - Clips against canonical, axial planes
  - Wastes cycles transforming irrelevant geometry
  - Incorrect for exotic geometry (as shown)

Clip in NDC X, Y, Z, W space, then Project to X, Y, Z
- Correctly handles primitives with negative W
  - Must deal with primitive fan-out ($1 \rightarrow 2$)
  - Many prims can be culled before transformation

Clip in X, Y, Z World- or Eye-Space
- Complex implementation to handle $W < 0, W = 0$
- Plane inner products are slower by factor of 2-4

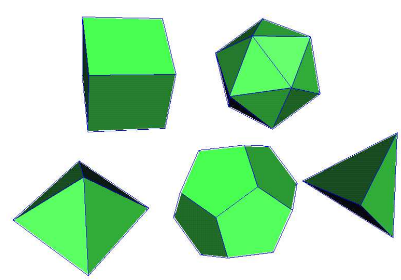
Hardware designers must commit to one approach
- SGIs, for example, perspective transform, then clip
- *Scissoring* can prevent most sequential clipping
  - *if* your architecture is massively parallel

Software renderers can be more flexible
- E.g., by adapting to changing cull statistics

Back-Face Removal

Objects modeled as closed, "watertight" containers

Vertices oriented so that face normals point "out"

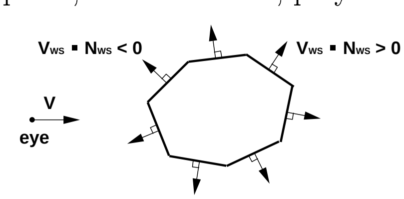


Any line of sight must first encounter an exterior face

Thus back-facing polygons need not be rendered

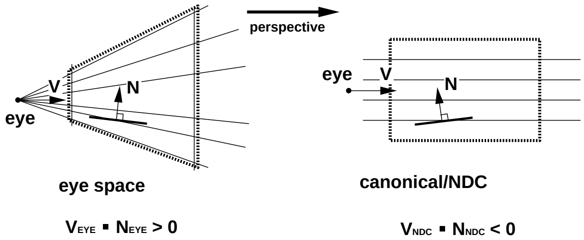
What criterion can be used to eliminate them?

Given: eye point, view direction, poly normal



Back-Face Removal

Problem: this quantity is not an invariant !



Must incorporate eye position as well

So, compute sign of $\mathbf{H} \cdot \mathbf{E}$

In which space should we perform this?

Object? (How?)

World? (How?)

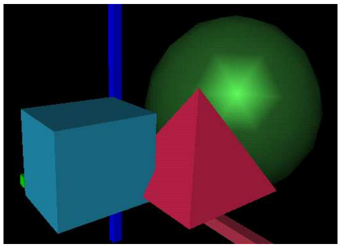
Eye? (How?)

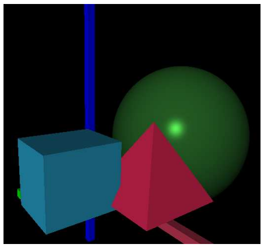
NDC? (How?)

Screen? (How?)

Next Week

Tuesday: Ray-Casting





Thursday: Recursive Ray-Tracing