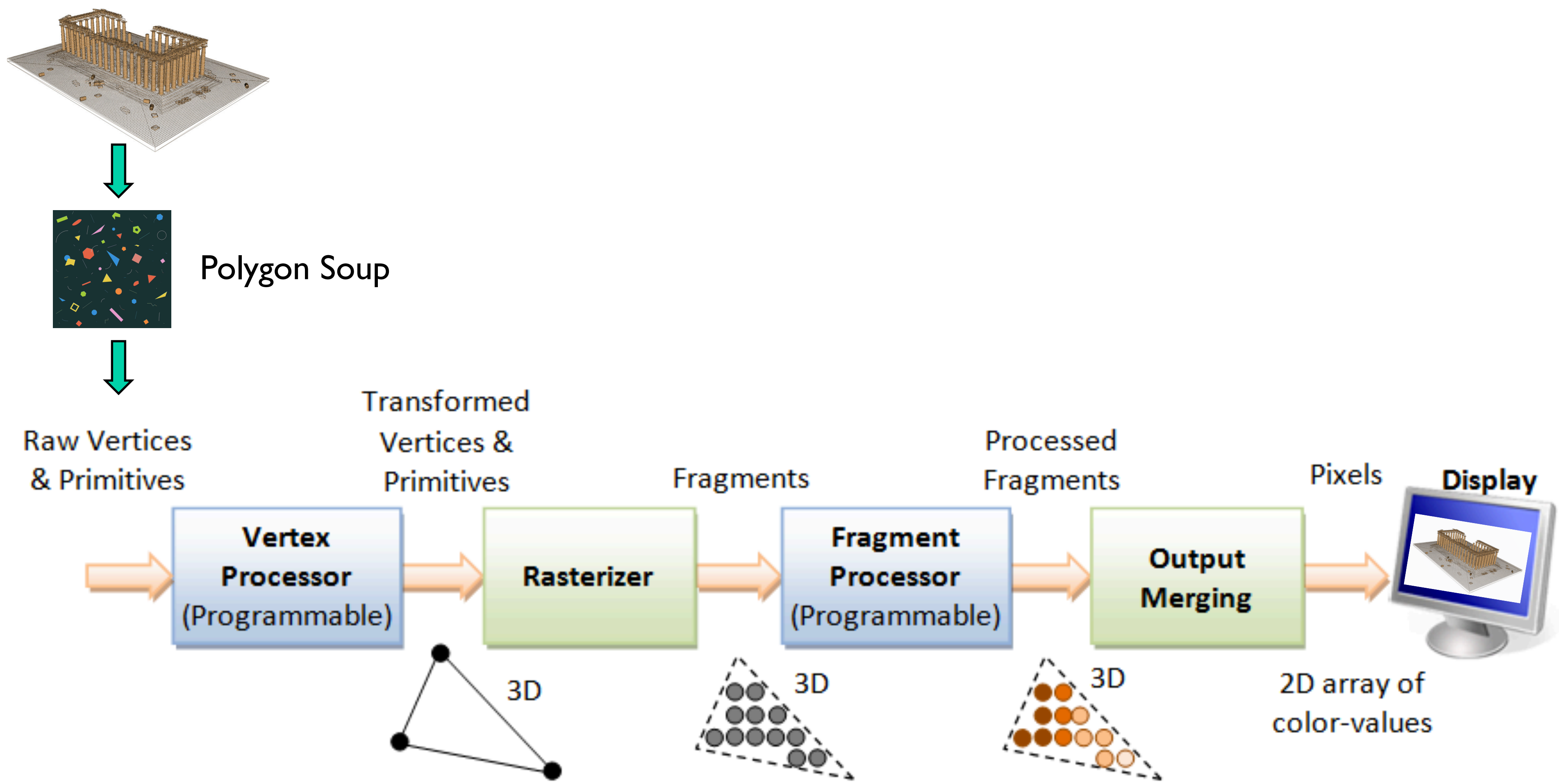
CSE 5542 - Real Time Rendering
Week 9

Post Geometry Shaders

Courtesy: E. Angel and D. Shreiner – Interactive Computer Graphics 6E © Addison-Wesley 2012

Pipeline


Polygon Soup
Raw Vertices & Primitives
Transformed Vertices & Primitives
Fragments
Processed Fragments
Pixels
Display
Vertex Processor (Programmable)
Rasterizer
Fragment Processor (Programmable)
Output Merging
3D
3D
3D
2D array of color-values

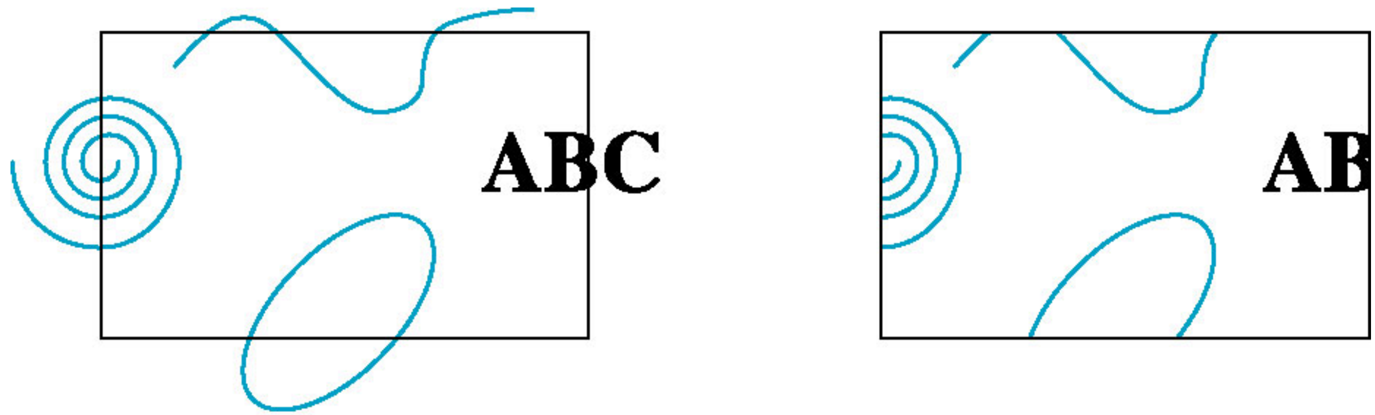
Pipeline


Vertices
Vertex Processor
Clipper and Primitive Assembler
Rasterizer
Fragment Processor
Pixels

Topics

- Clipping
- Scan conversion

Clipping

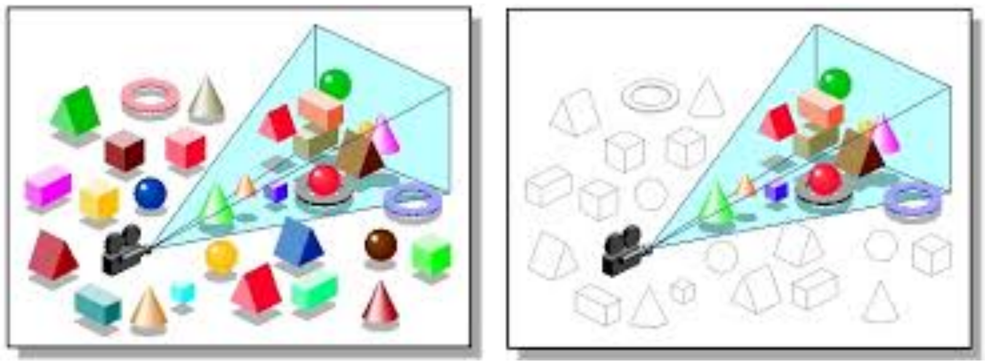


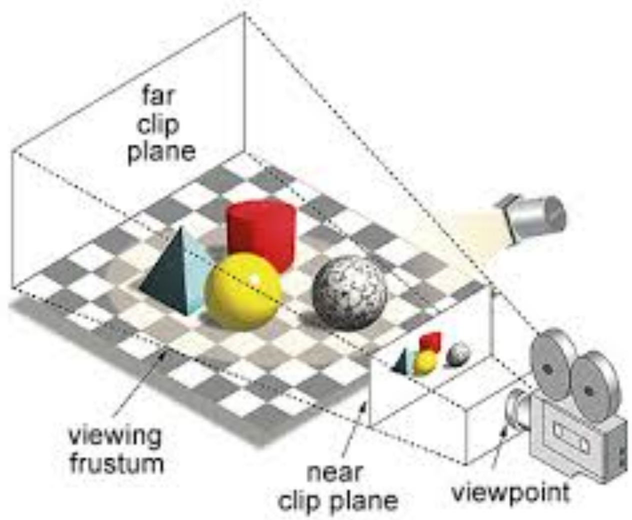
Clipping

- After geometric stage
  - vertices assembled into primitives

- Must clip primitives that are outside view frustum

Clipping



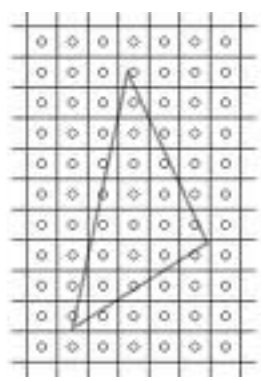


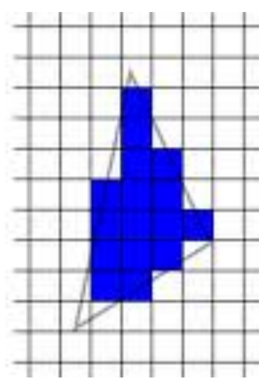
Scan Conversion

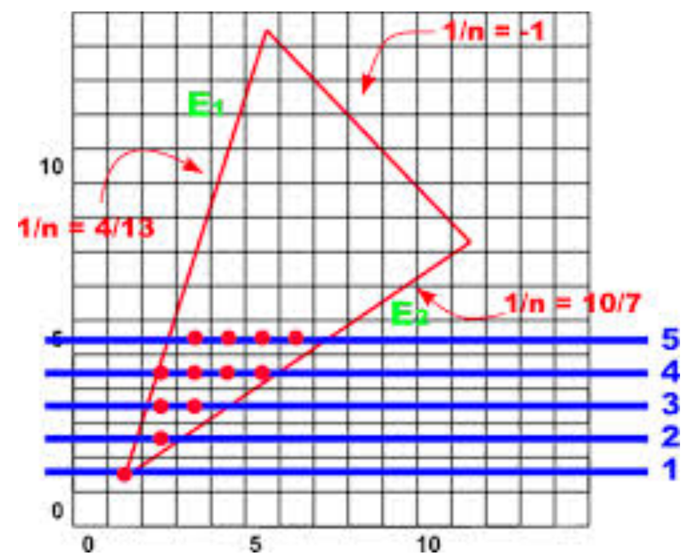
Which pixels can be affected by each primitive

– Fragment generation

– Rasterization or scan conversion







Additional Tasks

Some tasks deferred until fragment processing

- Hidden surface removal
- Antialiasing

Clipping

Contexts

- 2D against clipping window

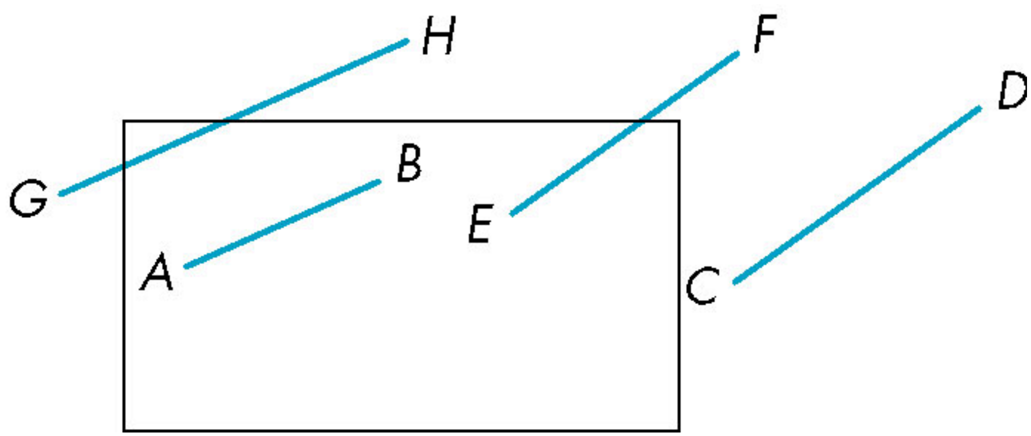
- 3D against clipping volume

2D Line Segments

Brute force:

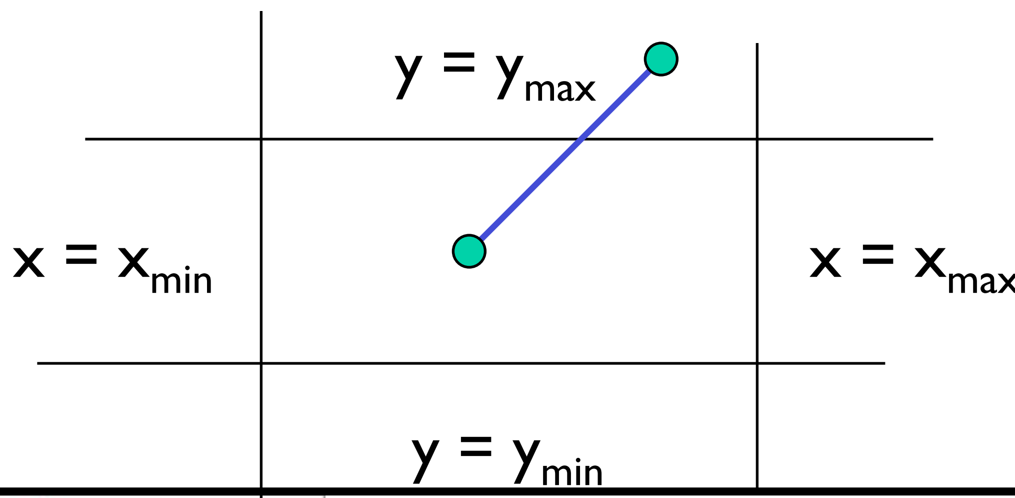
– compute intersections with all sides of clipping window

– Inefficient



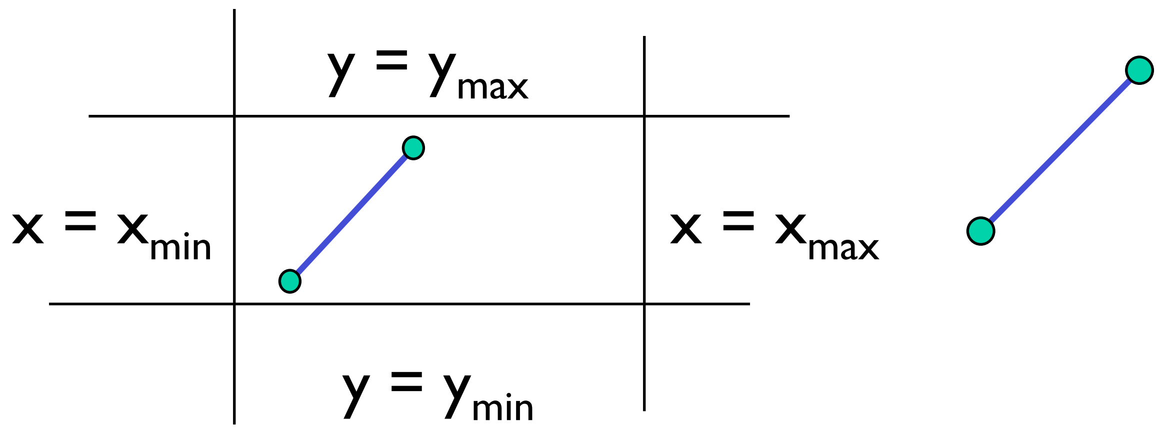
Cohen-Sutherland Algorithm

- Eliminate cases without computing intersections
- Start with four lines of clipping window



The Cases

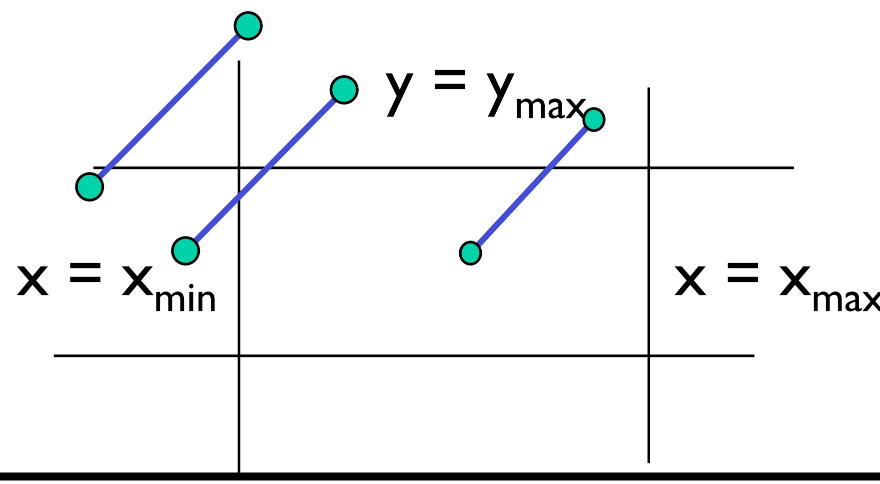
- Case 1: both endpoints of line segment inside all four lines
  - Draw (accept) line segment as is



- Case 2: both endpoints outside all lines and on same side of a line
  - Discard (reject) the line segment

The Cases

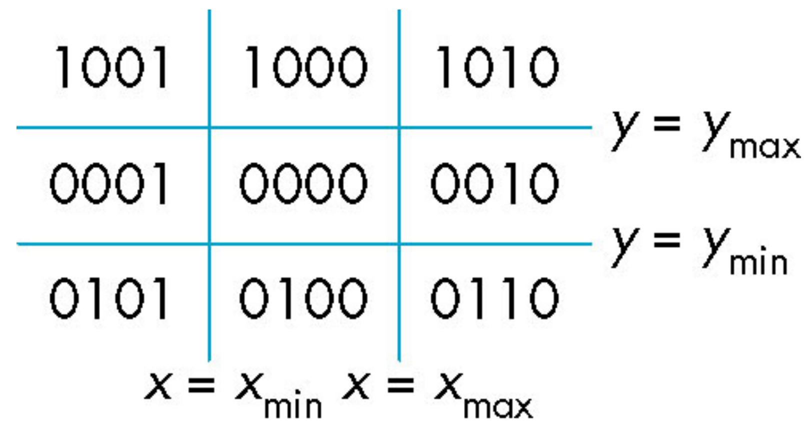
- Case 3: One endpoint inside, one outside
  - Must do at least one intersection
- Case 4: Both outside
  - May have part inside
  - Must do at least one intersection



Defining Outcodes

- For each endpoint, define an outcode $b_0 b_1 b_2 b_3$

b_0 = 1 if $y > y_{max}$, 0 otherwise
b_1 = 1 if $y < y_{min}$, 0 otherwise
b_2 = 1 if $x > x_{max}$, 0 otherwise
b_3 = 1 if $x < x_{min}$, 0 otherwise



- Outcodes divide space into 9 regions

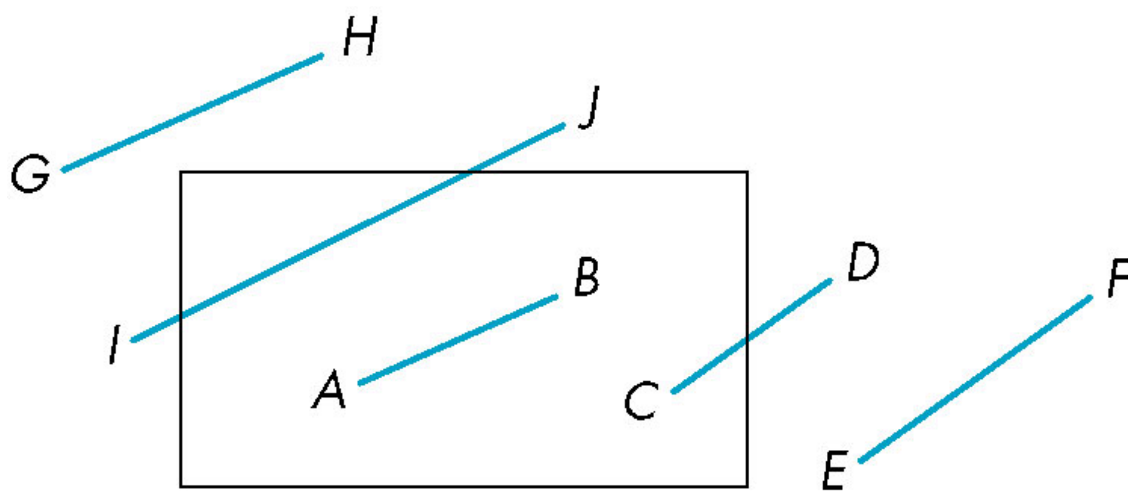
- Computation of outcode requires at most 4 comparisons

Using Outcodes

Consider the 5 cases below

AB: outcode(A) = outcode(B) = 0

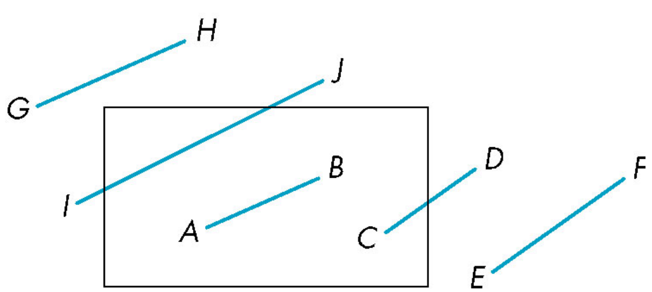
– Accept line segment

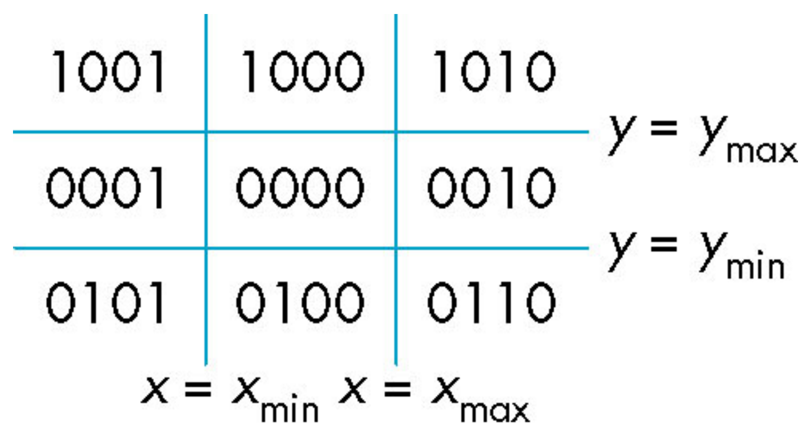


Using Outcodes

CD: outcode (C) = 0, outcode(D) $\neq$ 0

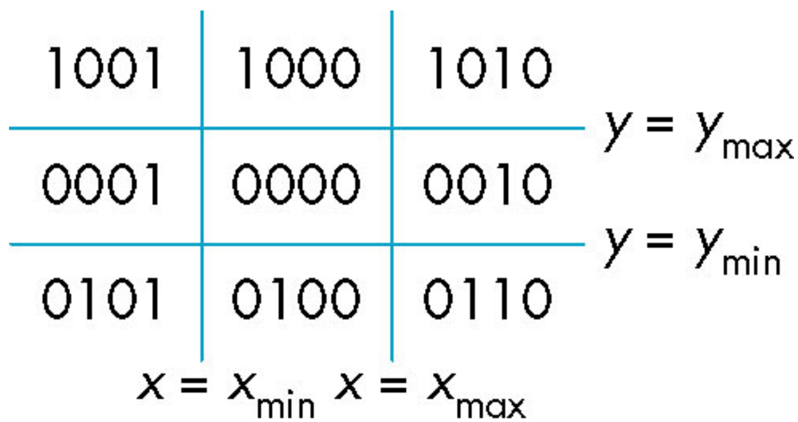
- Compute intersection
- Location of 1 in outcode(D) marks edge to intersect with





Using Outcodes

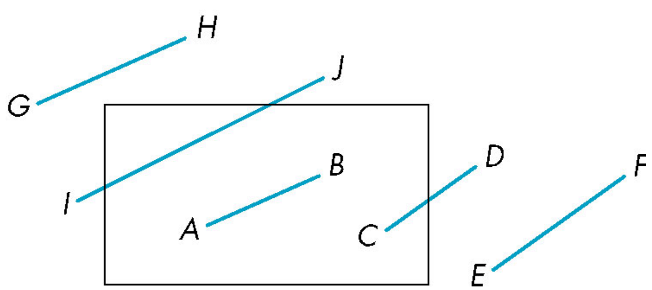
If there were a segment from A to a point in a region with 2 ones in outcode, we might have to do two intersections

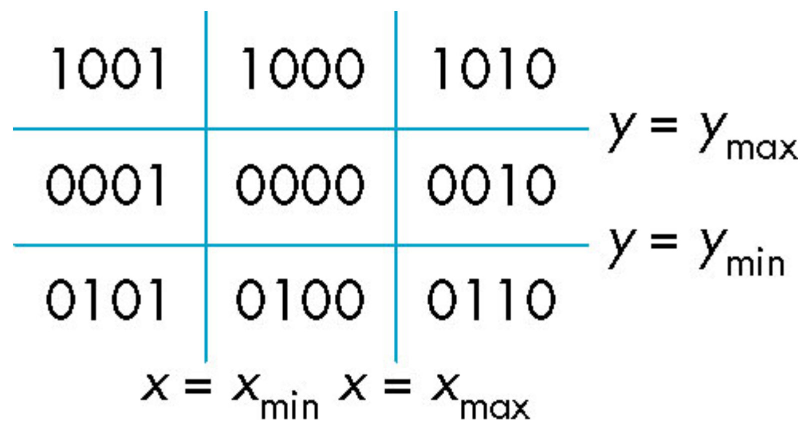


Using Outcodes

EF: outcode(E) logically ANDed with outcode(F) (bitwise) ≠ 0

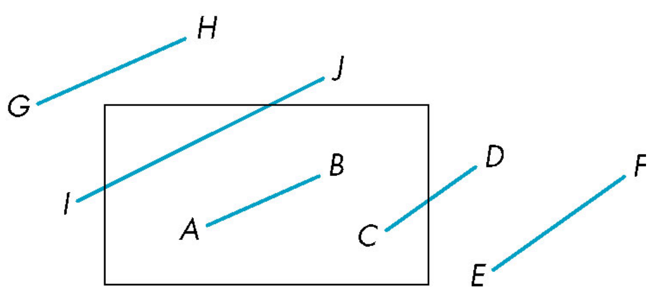
- Both outcodes have a 1 bit in the same place
- Line segment is outside clipping window
- reject

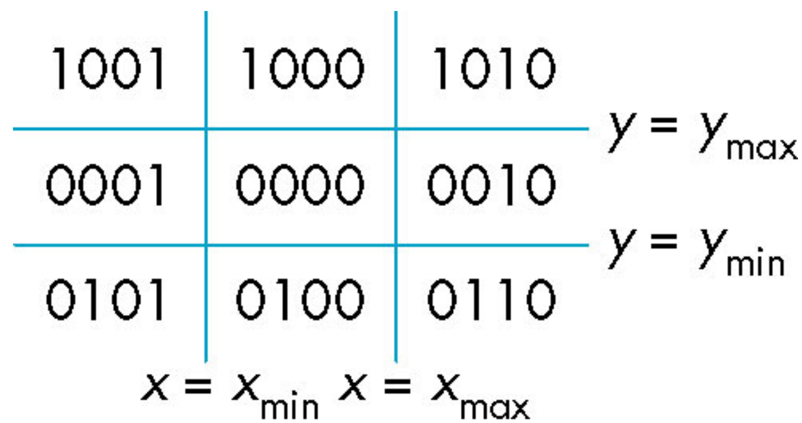




Using Outcodes

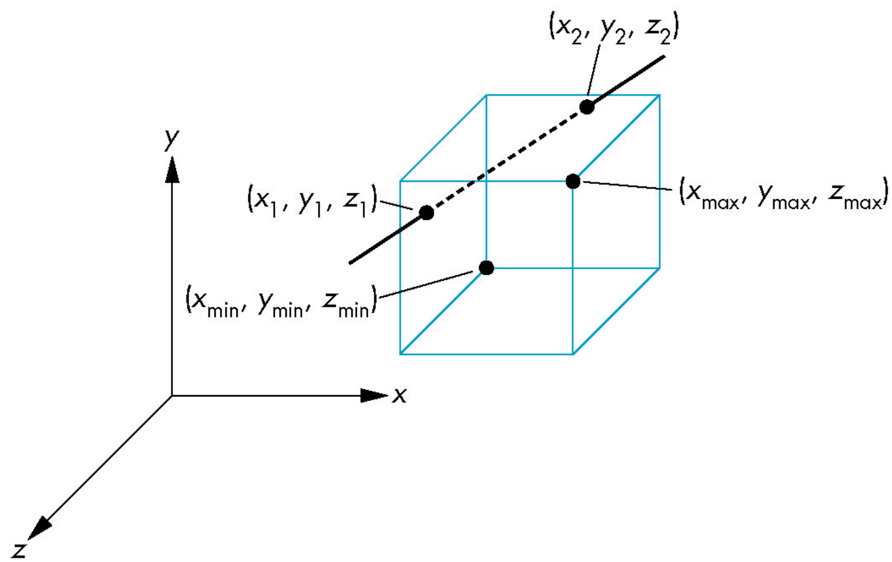
- GH and IJ
  - same outcodes, neither zero but logical AND yields zero
- Shorten line by intersecting with sides of window
- Compute outcode of intersection
  - new endpoint of shortened line segment
- Recurse algorithm

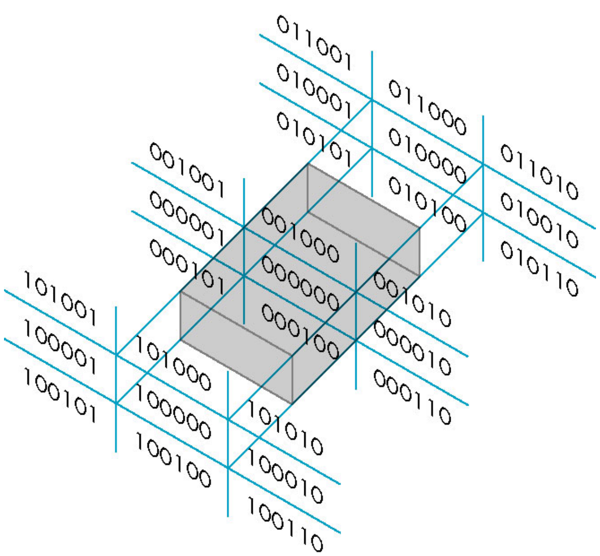




Cohen Sutherland in 3D

- Use 6-bit outcodes
- When needed, clip line segment against planes

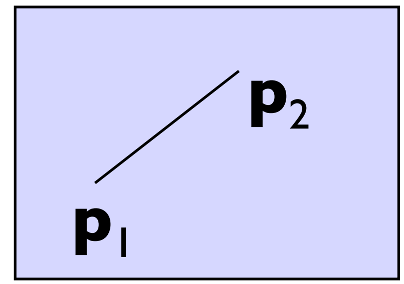




Liang-Barsky Clipping

Consider parametric form of a line segment

$$\mathbf{p}(\alpha) = (1-\alpha)\mathbf{p}_1 + \alpha\mathbf{p}_2 \quad 1 \geq \alpha \geq 0$$



Intersect with parallel slabs –

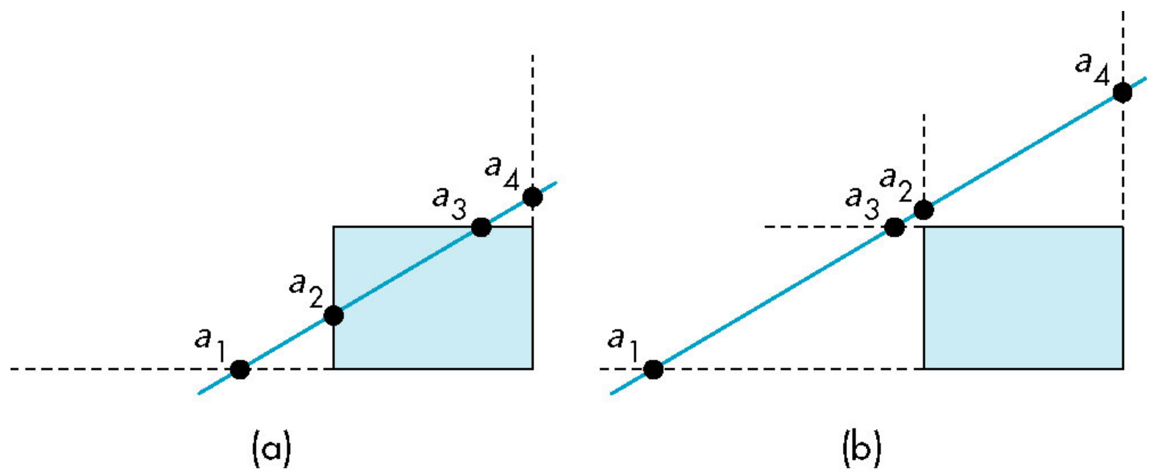
Pair for Y

Pair for X

Pair for Z

Liang-Barsky Clipping

- In (a): $a_4 > a_3 > a_2 > a_1$
  - Intersect right, top, left, bottom: shorten
- In (b): $a_4 > a_2 > a_3 > a_1$
  - Intersect right, left, top, bottom: reject



Advantages

- Can accept/reject as easily as with Cohen-Sutherland
- Using values of α, we do not have to use algorithm recursively as with C-S
- Extends to 3D