

# Computer Graphics (CS 543)

## Lecture 12a: 2D Clipping

Prof Emmanuel Agu

*Computer Science Dept.  
Worcester Polytechnic Institute (WPI)*



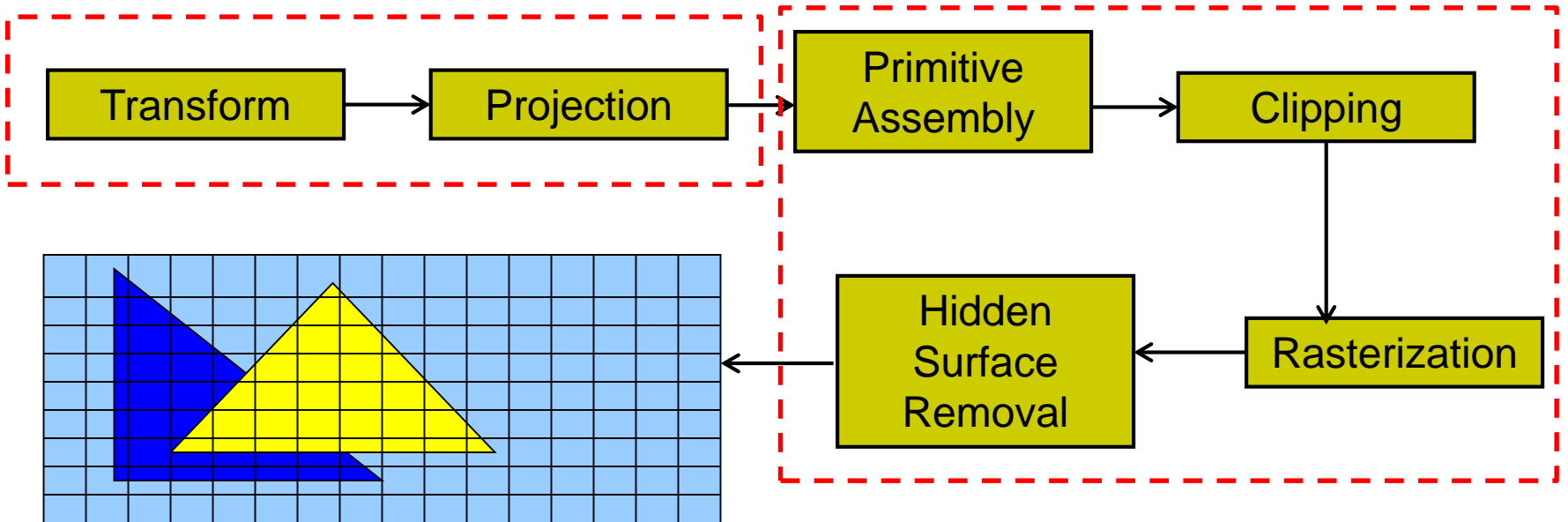


# OpenGL Stages

- After projection, several stages before objects drawn to screen
- These stages are **NOT** programmable

Vertex shader: programmable

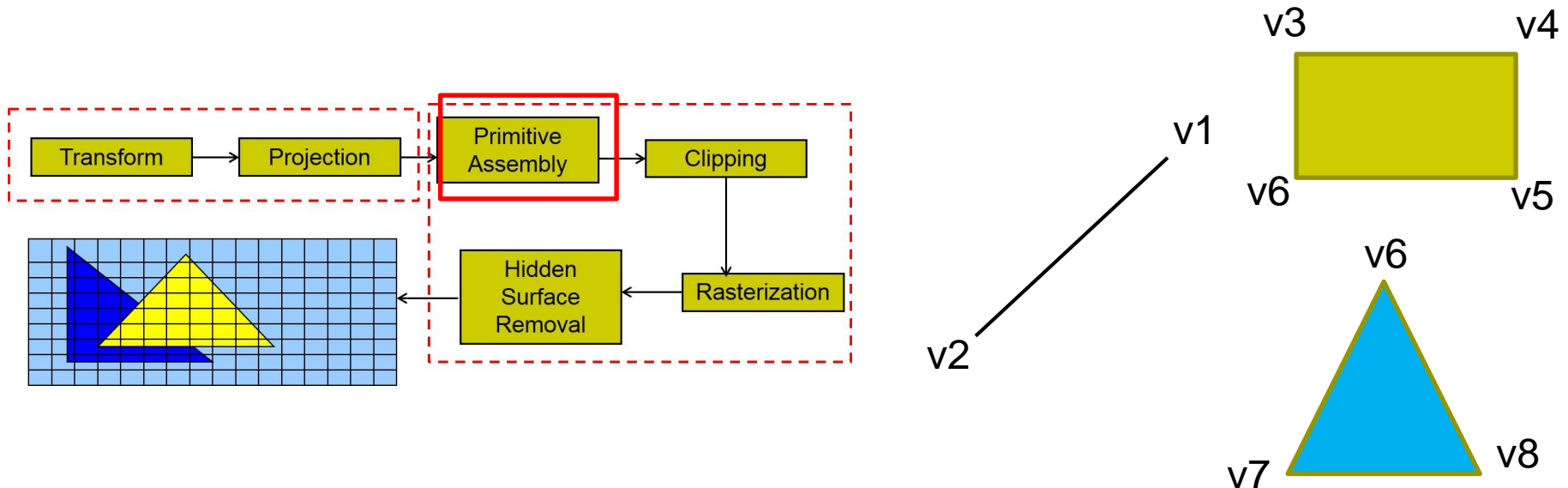
In hardware: **NOT** programmable





# Hardware Stage: Primitive Assembly

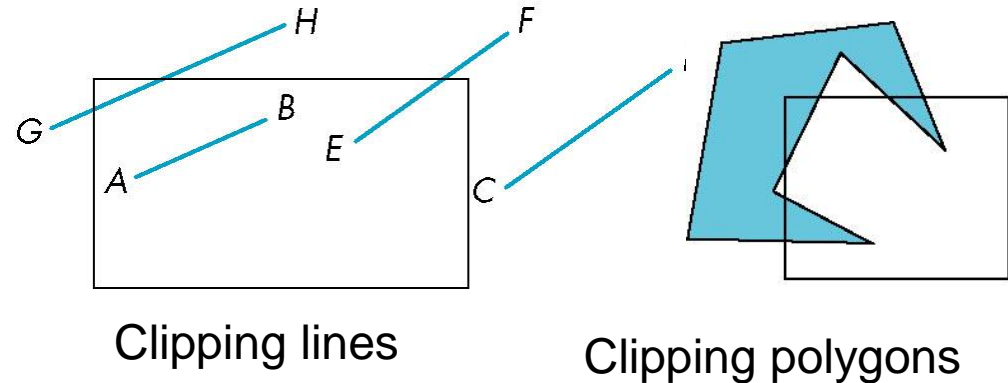
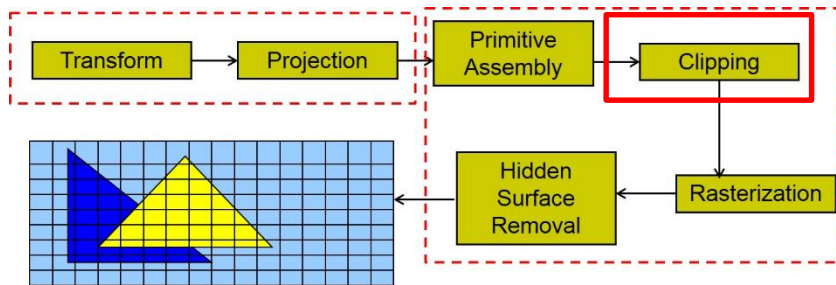
- Up till now: Transformations and projections applied to vertices individually
- **Primitive assembly:** After transforms, projections, individual vertices grouped back into primitives
- E.g. **v6, v7 and v8** grouped back into triangle





# Hardware Stage: Clipping

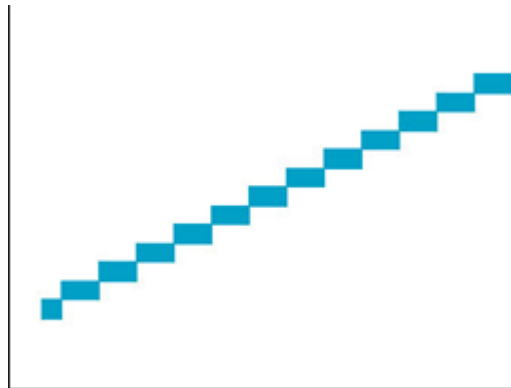
- After primitive assembly, subsequent operations are **per-primitive**
- **Clipping:** Remove primitives (lines, polygons, text, curves) outside view frustum (canonical view volume)





# Rasterization

- Determine which pixels that primitives (shapes) map to
  - Fragment generation
  - Rasterization or scan conversion

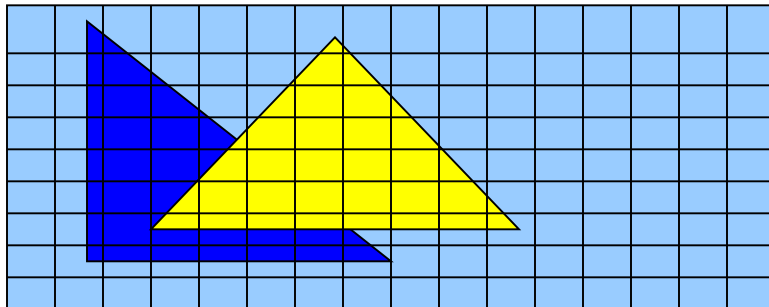




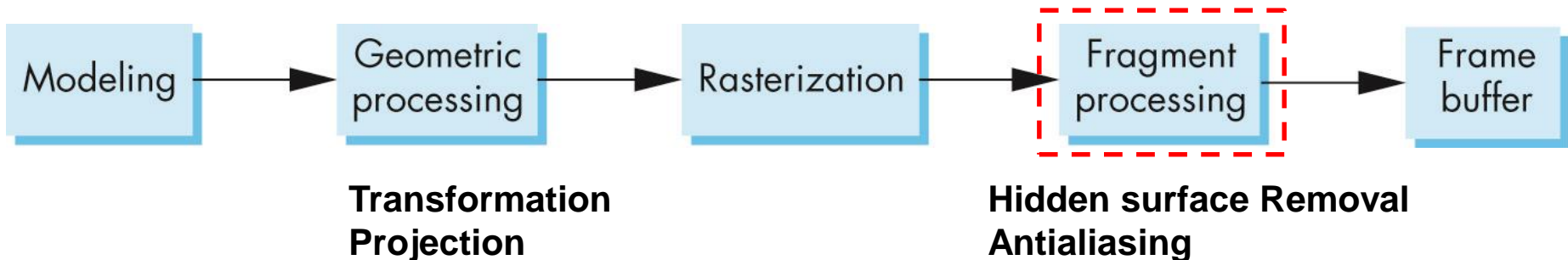
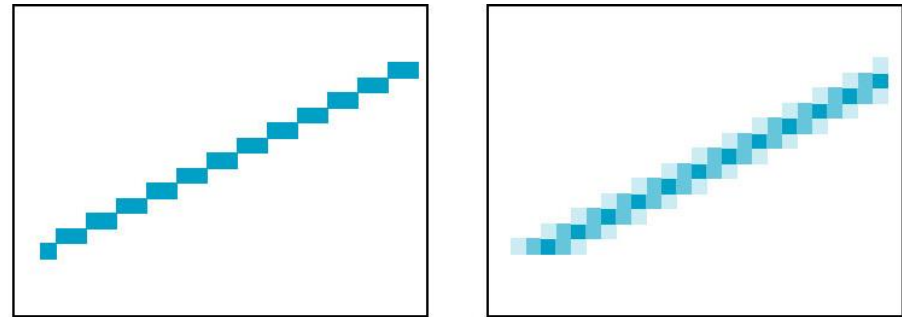
# Hidden Surface Removal

- Some tasks deferred until fragment processing

Hidden Surface Removal



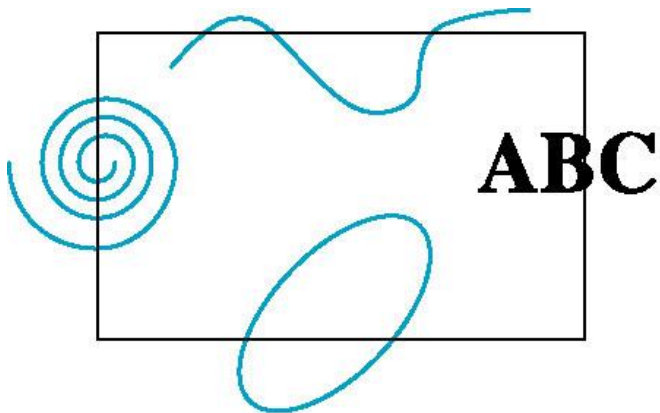
Antialiasing





# Clipping

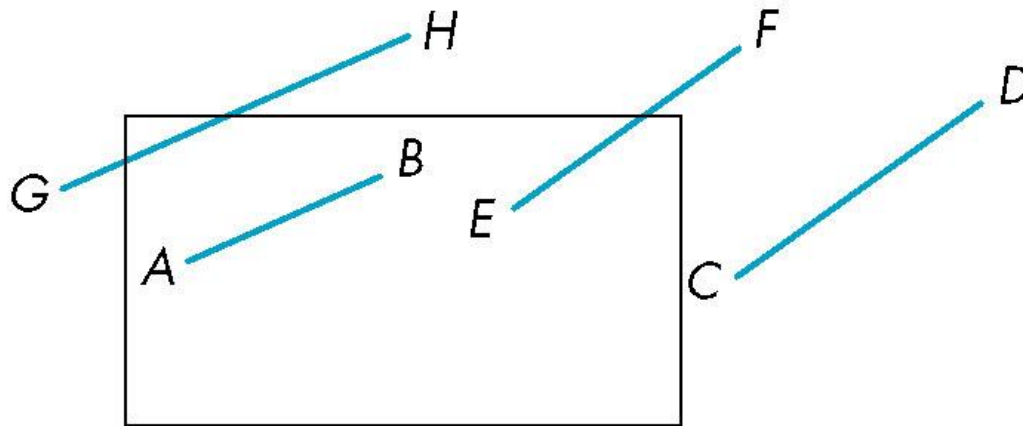
- 2D and 3D clipping algorithms
  - 2D against clipping window
  - 3D against clipping volume
- 2D clipping
  - Lines (e.g. dino.dat)
  - Polygons
  - Curves
  - Text





# Clipping 2D Line Segments

- **Brute force approach:** compute intersections with all sides of clipping window
  - Inefficient: one division per intersection

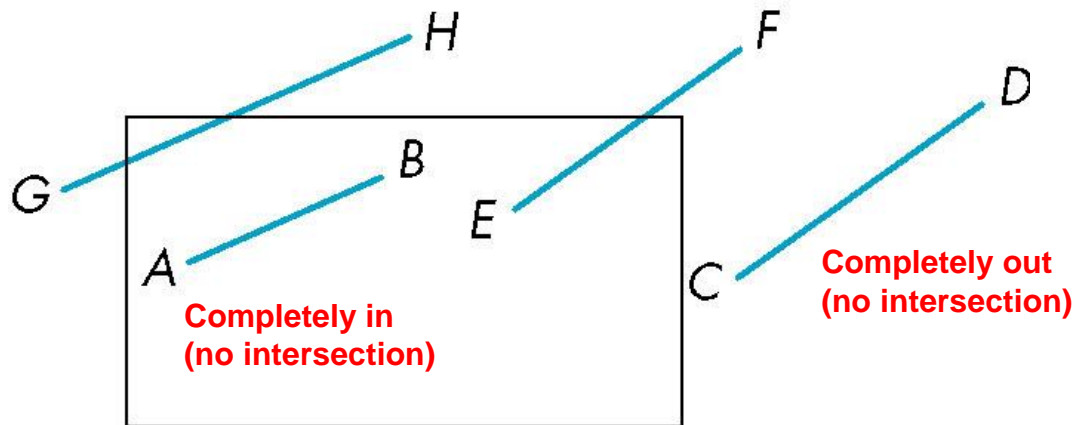






# 2D Clipping

- **Better Idea:** eliminate as many cases as possible without computing intersections
- Cohen-Sutherland Clipping algorithm



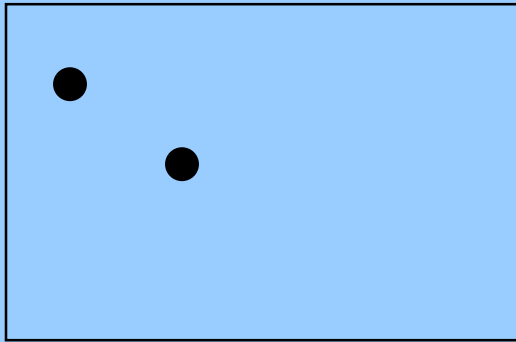
**Goal: Develop simple tests to eliminate lines like CD or AB (no intersection)**

# Clipping Points

Ref: Computer Graphics using OpenGL, Hill and Kelley, 3<sup>rd</sup> edition



$(x_{max}, y_{max})$



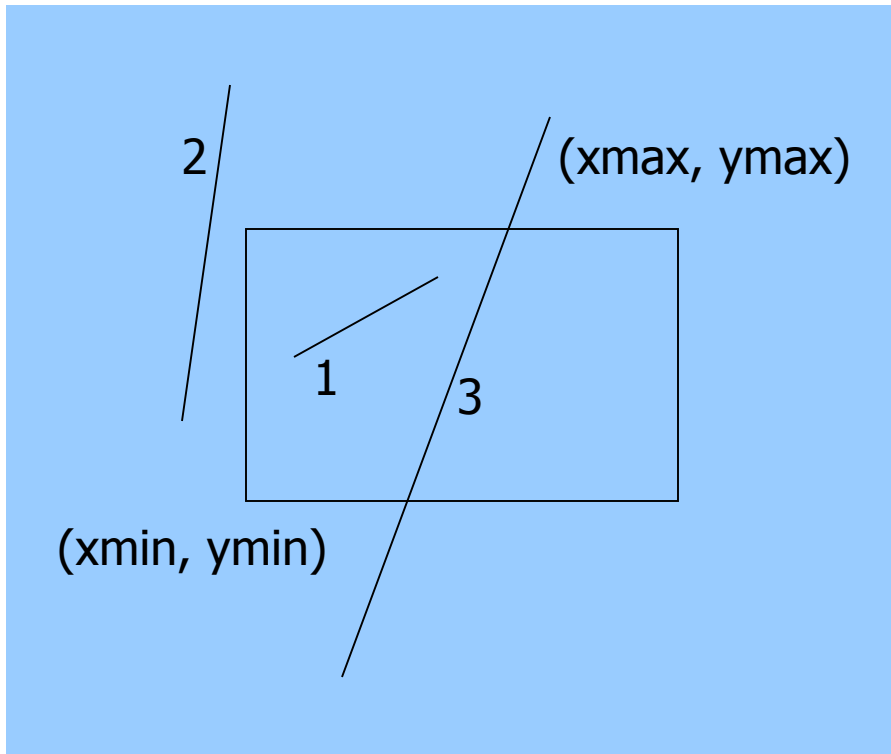
$(x_{min}, y_{min})$

Determine whether a point  $(x,y)$  is inside or outside of the world window?

If  $(x_{min} \leq x \leq x_{max})$   
**and**  $(y_{min} \leq y \leq y_{max})$

then the point  $(x,y)$  is inside  
else the point is outside

# Clipping Lines



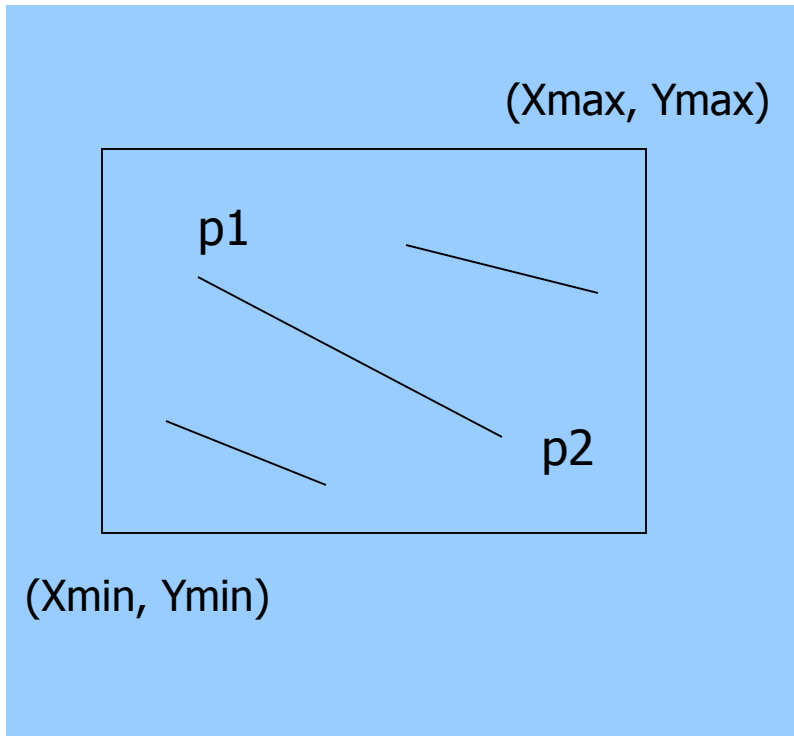
3 cases:

**Case 1:** All of line in

**Case 2:** All of line out

**Case 3:** Part in, part out

# Clipping Lines: Trivial Accept



**Case 1:** All of line in  
Test line endpoints:

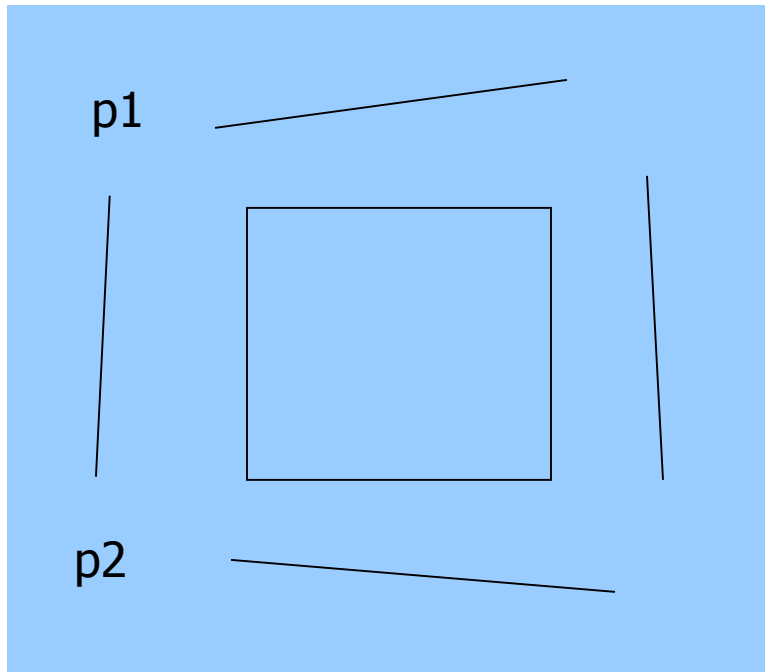
$$X_{min} \leq P1.x, P2.x \leq X_{max} \text{ and} \\ Y_{min} \leq P1.y, P2.y \leq Y_{max}$$

**Note:** simply comparing x,y values of  
endpoints to x,y values of rectangle

**Result:** trivially accept.  
Draw line in completely



# Clipping Lines: Trivial Reject



**Case 2:** All of line out

Test line endpoints:

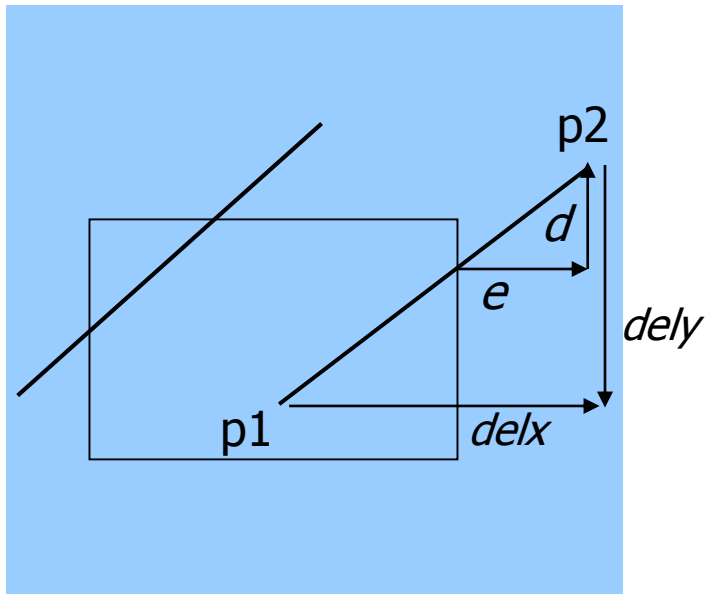
- $p1.x, p2.x \leq Xmin$  OR
- $p1.x, p2.x \geq Xmax$  OR
- $p1.y, p2.y \leq ymin$  OR
- $p1.y, p2.y \geq ymax$

**Note:** simply comparing x,y values of endpoints to x,y values of rectangle

**Result:** trivially reject.

Don't draw line in

# Clipping Lines: Non-Trivial Cases



**Case 3:** Part in, part out

Two variations:

One point in, other out

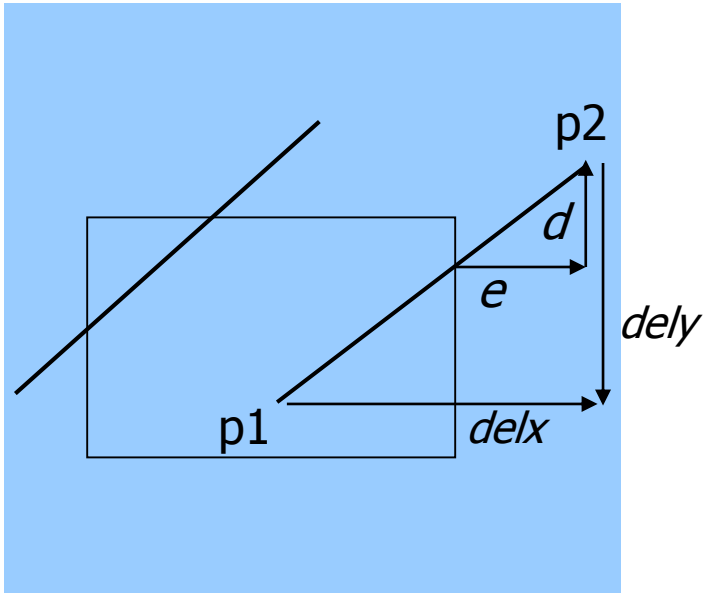
Both points out, but part of line cuts through viewport

Need to find inside segments

Use similar triangles to figure out length of inside segments

$$\frac{d}{dely} = \frac{e}{delx}$$

# Clipping Lines: Calculation example



If chopping window has  
(left, right, bottom, top) = (30, 220, 50, 240),  
what happens when the following lines are  
chopped?

(a)  $p1 = (40, 140)$ ,  $p2 = (100, 200)$

(b)  $p1 = (20, 10)$ ,  $p2 = (20, 200)$

(c)  $p1 = (100, 180)$ ,  $p2 = (200, 250)$

$$\frac{d}{dely} = \frac{e}{delx}$$

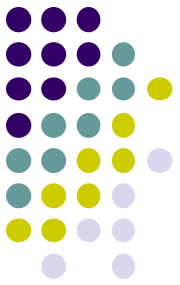
# Cohen-Sutherland pseudocode (Hill)



```
int clipSegment(Point2& p1, Point2& p2, RealRect W)
{
    do{
        if(trivial accept) return 1; // whole line survives
        if(trivial reject) return 0; // no portion survives
        // now chop
        if(p1 is outside)
            // find surviving segment
            {
                if(p1 is to the left) chop against left edge
                else if(p1 is to the right) chop against right edge
                else if(p1 is below) chop against the bottom edge
                else if(p1 is above) chop against the top edge
            }
    }
```



# Cohen-Sutherland pseudocode (Hill)



```
else // p2 is outside
    // find surviving segment
{
    if(p2 is to the left) chop against left edge
    else if(p2 is to right) chop against right edge
    else if(p2 is below) chop against the bottom edge
    else if(p2 is above) chop against the top edge
}
}while(1);
}
```



# References

- Angel and Shreiner, Interactive Computer Graphics, 6<sup>th</sup> edition
- Hill and Kelley, Computer Graphics using OpenGL, 3<sup>rd</sup> edition