CSC 240 Computer Graphics
Video 9: Line Clipping

Nick Howe
Smith College

Line Clipping

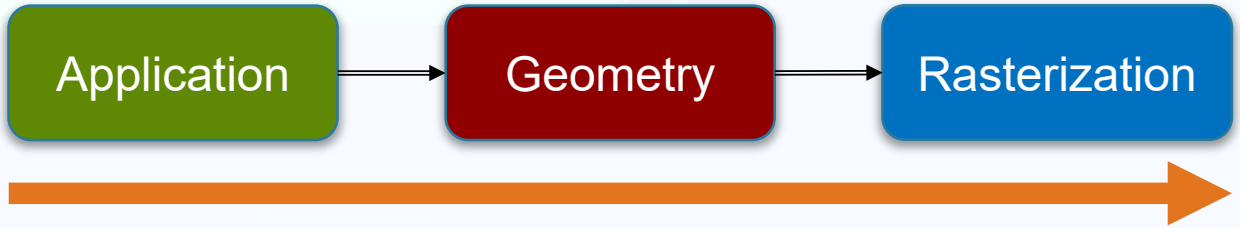
- Rendered environments can have millions of polygons
- Characters have up to 50K polygons
- Must economize wherever possible!

https://gamerant.com/fortnite-battle-royale-second-map/

http://disney.wikia.com/wiki/Vanellope_von_Schweetz

Line Clipping

Typical game rendering pipeline:



Application: input response, collision detection, etc.

Geometry: 3D → 2D → display

Rasterization: compute actual pixel colors

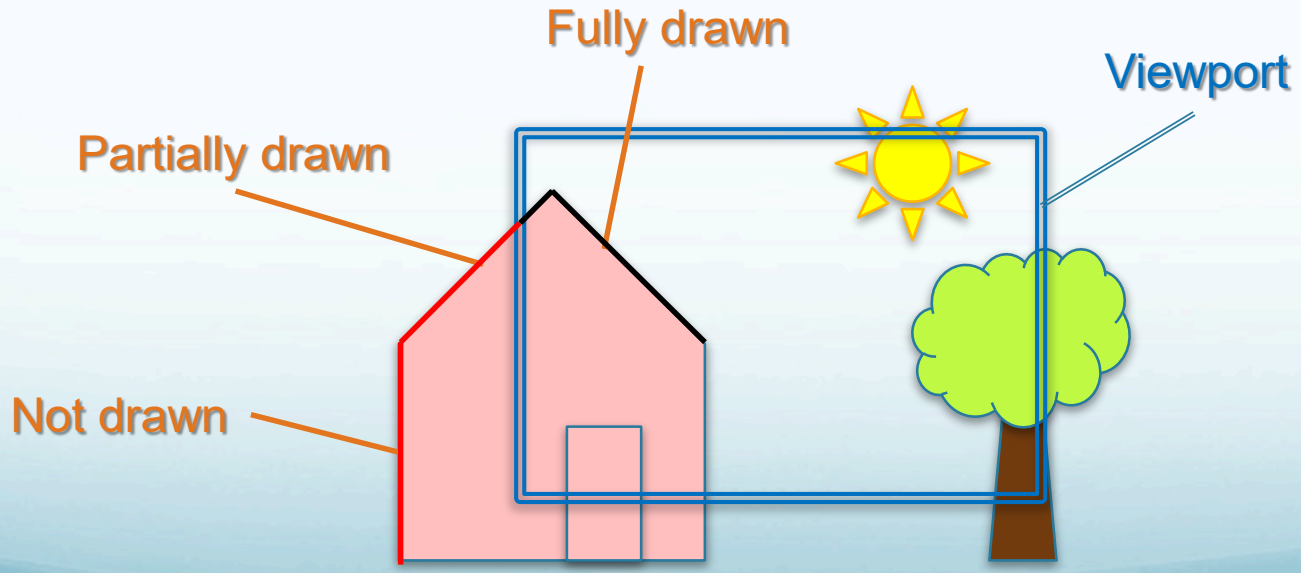
Steps get more expensive the further you go

Clipping reduces work further down the pipeline

Line Clipping

Goals:

- Draw parts of lines that are within the viewport
- Don’t draw lines that are fully outside the viewport

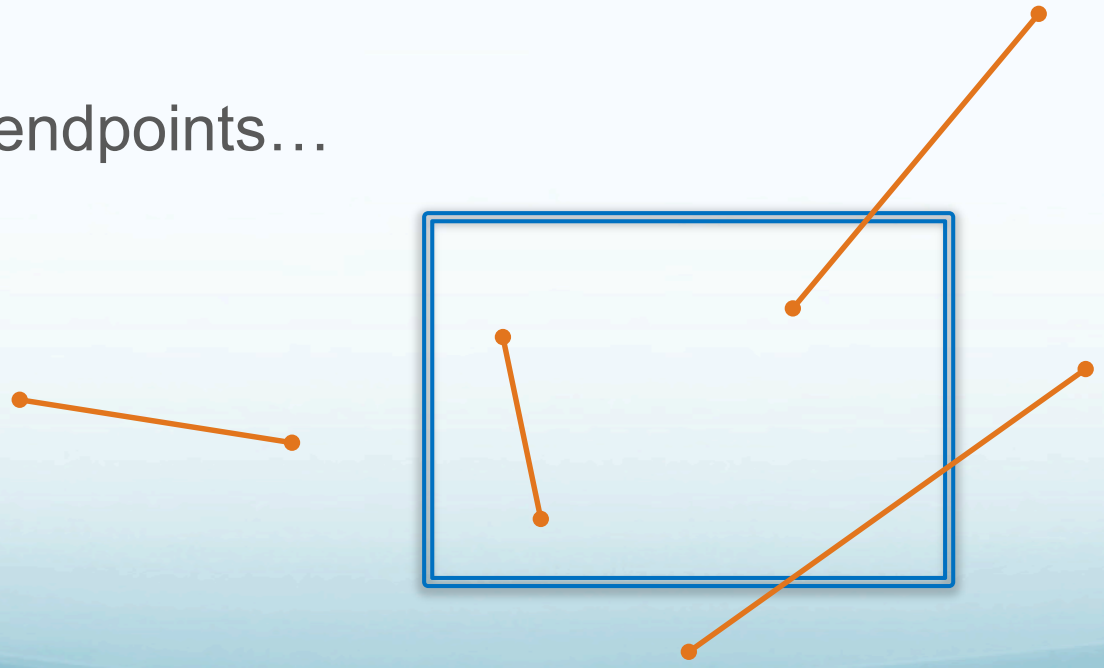


Discussion

How can we efficiently decide whether lines are

- in,
- out,
- or both?

➢ Look at the endpoints…

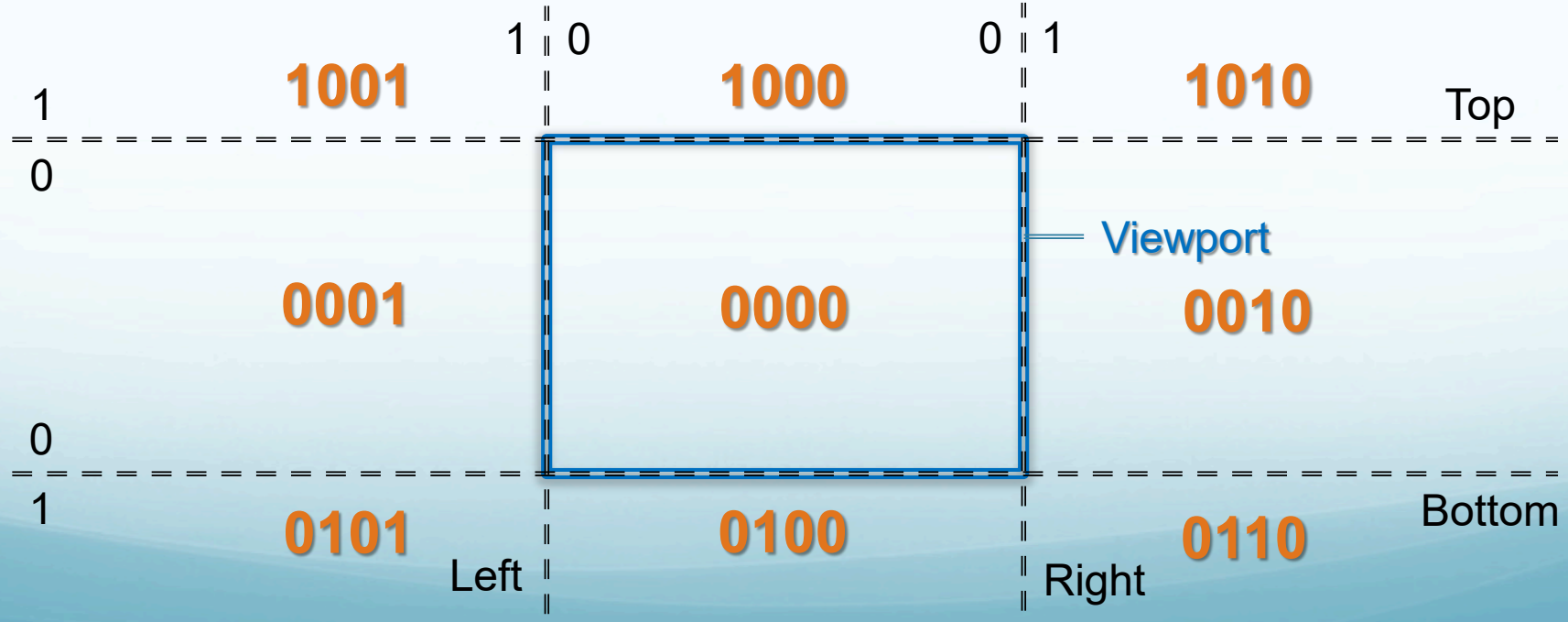


Both in:
Draw fully
One in, one out:
Draw partially
Both out:
Don't draw?
…sometimes?

Line Clipping

Usual method is called the Cohen-Sutherland algorithm

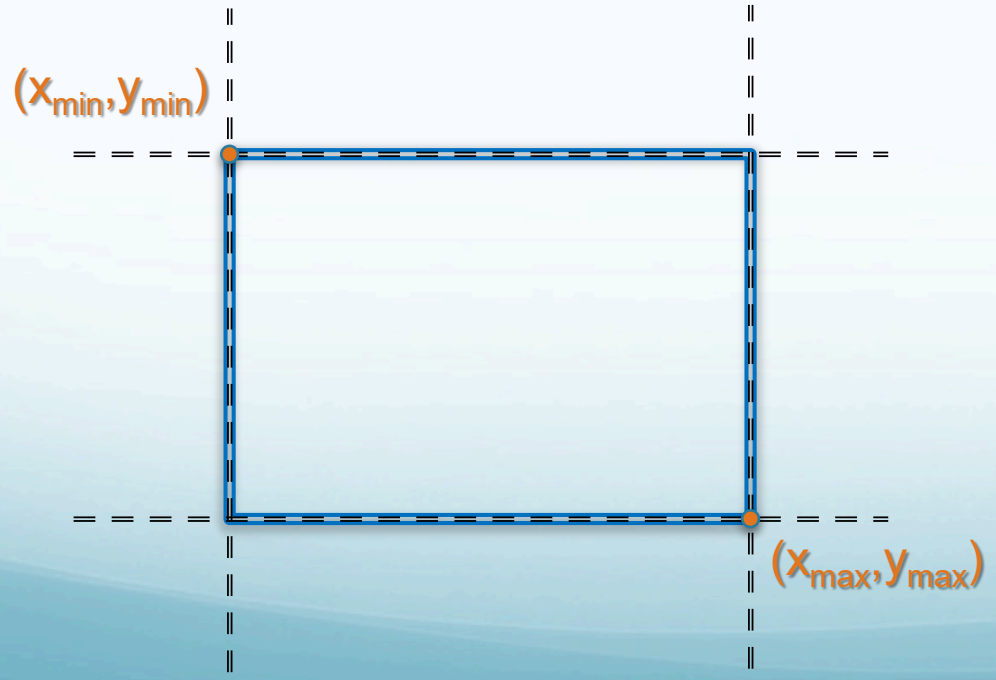
- Divide 2D plane into nine regions using 4-bit code
- Top, Bottom, Right, Left: 1 = outside, 0 = inside



Line Clipping

Viewport boundaries are (x_{min},y_{min}) to (x_{max},y_{max})

Use this to determine code of line endpoints (x,y):

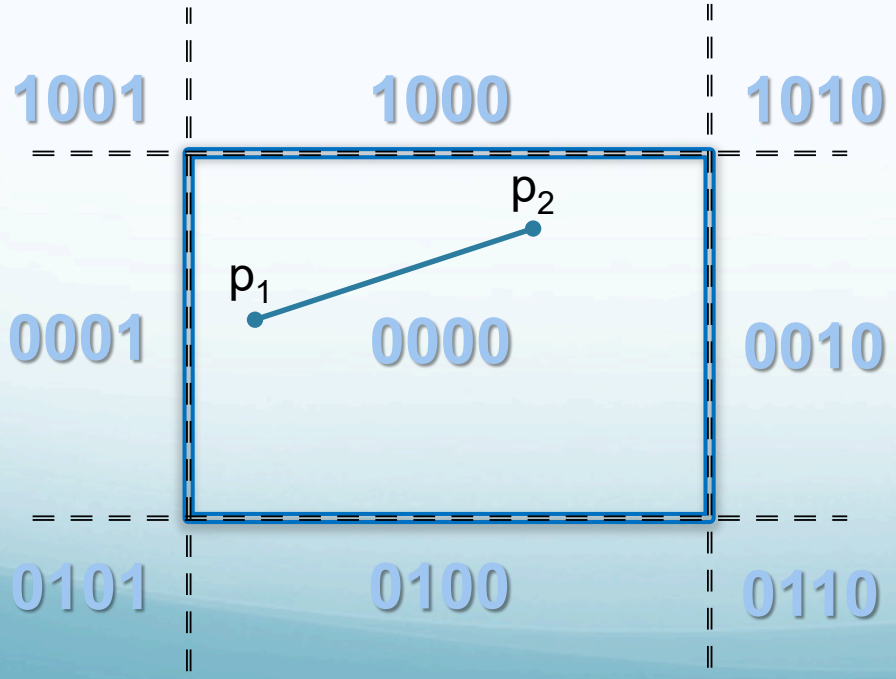



```
getCode(x,y):
    code ← 0000
    if (x < xmin) then
        code ← code|0001
    if (x > xmax) then
        code ← code|0010
    if (y < ymin) then
        code ← code|1000
    if (y > ymax) then
        code ← code|0100
    return code
```

Line Clipping

Case 1: `code1 | code2 == 0000`

- Both endpoints are within the viewport
- Render the entire line



Bitwise OR:

1100 | 1010 = 1110

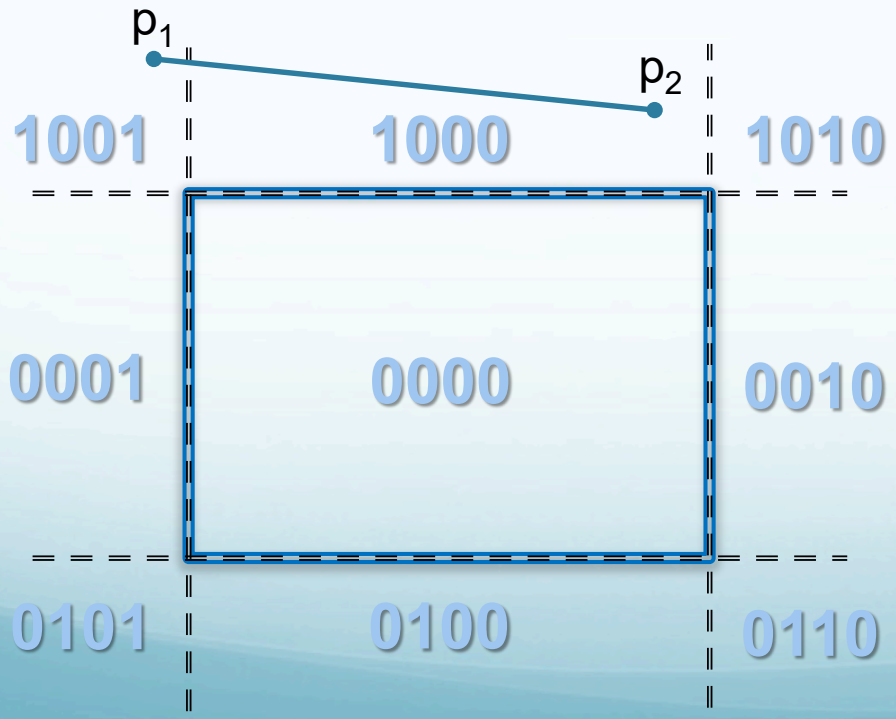
Compare each paired bit

Result is 1 iff any input is 1

Line Clipping

Case 2: `code1 & code2 != 0000`

- Entire line is outside the viewport
- Skip the entire line



Bitwise AND:

1100 & 1010 = 1000

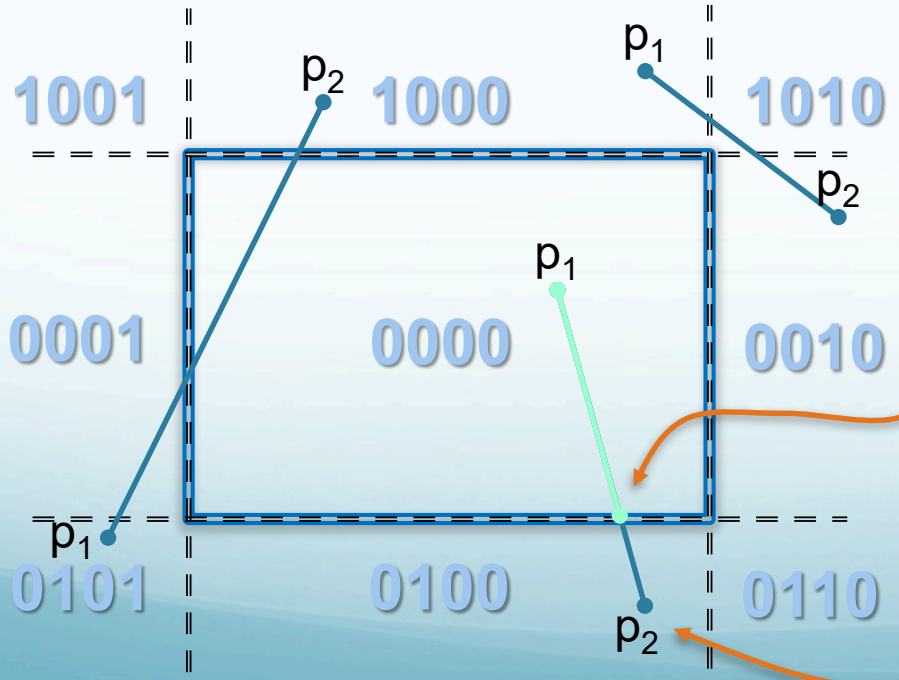
Compare each paired bit

Result is 0 iff any input is 0

Line Clipping

Case 3: neither 1 nor 2 and: `code1 & code2 == 0000`

- Line might be partially inside viewport
- Simplify problem and check again
- How to simplify? Find intersection of line with a boundary



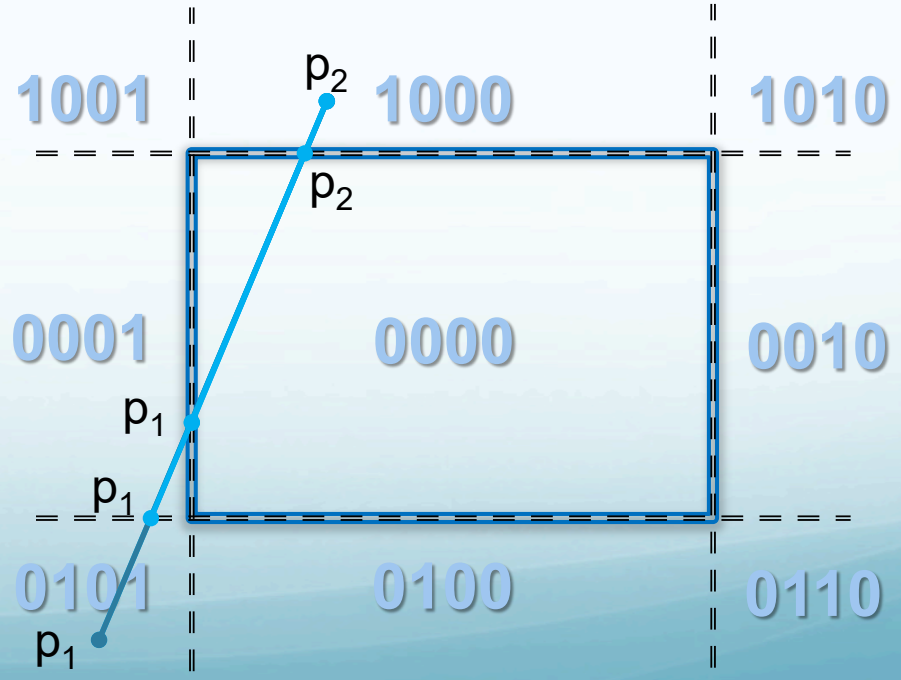
Use math to find intersection with viewport boundary
(How can we be sure that line intersects boundary?)
`code1 & code2 == 0000`

Pick an endpoint with nonzero region code

Line Clipping

Clipping process may take multiple rounds

1. p_1 = 0**1**01 → clip bottom; intersect at y = y_{max}
2. p_1 = 000**1** → clip left; intersect at x = x_{min}
3. p_2 = **1**000 → clip top; intersect at y = y_{min}



Questions

PAUSE NOW & ANSWER

1. In the Cohen-Sutherland code 0110, what does the third digit mean?

   A one in the third digit means the point is outside the viewport to the right.

2. Compute the result: 1010 & 0110 = ?

   0010 (answer is one only where both inputs are one)

3. Compute the result: 1010 | 0110 = ?

   1110 (answer is one only where either input is one)

4. Which of the following must be clipped to determine if they are visible?

   a. 0101 and 0100
   b. 1001 and 0110
   c. 1010 and 0000
   d. 0001 and 0001

Line Clipping

Math for finding intersections:

$$y - y_1 = m(x - x_1)$$

Recall: $m = \frac{y_2 - y_1}{x_2 - x_1}$

- Substitute x_{min} for x and solve for y: ——— if code & 0001

$$y = m(x_{min} - x_1) + y_1$$

- Substitute y_{min} for y and solve for x: ——— if code & 1000

$$y_{min} - y_1 = m(x - x_1)$$

$$\frac{1}{m}(y_{min} - y_1) = x - x_1$$

$$x = \frac{1}{m}(y_{min} - y_1) + x_1$$

Use same forms to substitute x_{max} or y_{max}

if code & 0100

if code & 0010

Line Clipping

Putting it all together:

```
Case 1: …                          // draw line
Case 2: …                          // skip line
Case 3:
  if code1 & 1000                  // move (x1,y1) to y=ymin
    y ← ymin
    x ← (ymin-y1)/m+x1
    clipLine(x,y,x2,y2)            // recursive call
  else if code1 & 0100 …

  else if code2 & 1000 …
```

Questions

PAUSE NOW & ANSWER

1. If you are clipping a line, and the code for p_1 is 0010, which variable are you constraining, x or y?

   The point is to the right of the viewport, so you constrain x.

2. If you are clipping a line, and the code for p_1 is 0010, what value are you constraining it to?

   The point is to the right of the viewport, so you constrain x to x_{max}.

3. Suppose that $p_1 = (120,65)$ and you are constraining $x = x_{max} = 100$. Also suppose $m = 0.5$ for this line segment. What is the new (x, y)?

$$y = m(x_{max} - x_1) + y_1 = 0.5(100 - 120) + 65 = -10 + 65 = 55$$

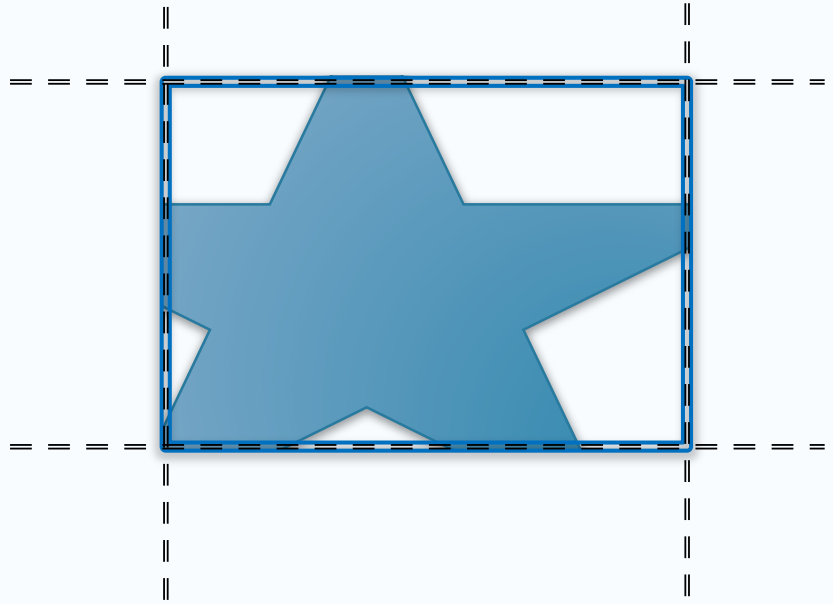
$$(x, y) = (100,55)$$

Other Clipping

Polygon Clipping

Sutherland Hodgman Algorithm clips **polygons**

- Loop over clipping rectangle boundary lines
  - Loop over vertices
    - Keep vertices on inside of boundary
    - Replace vertices outside boundary with new ones at edge



Text Clipping

Text strings may be clipped via several strategies:

- All or none string clipping
- All or none character clipping
- Exact text clipping

Review

After watching this video, you should be able to…

- Explain the motivation for line clipping
- Compute Cohen-Sutherland endpoint codes, given a point & viewport
- Use the codes to determine whether a segment is visible, and/or whether clipping is required
- Clip line segments as needed according to the viewport boundaries
- Demonstrate the Sutherland-Hodgman polygon clipping algorithm by hand