CSC 240 Computer Graphics
Line Clipping

Nick Howe
Smith College
Q. Why wouldn't question 4 be answer B and C? Doesn't C compute to 0000 as well?

A. For C, we know the line will be partially visible because one end is in the viewport. It’s true that the other end will need to be clipped.

Q. How do you get the region code orders of one and zeros what is the significant of a one vs a zero? How do we know when to clip? Which case?

A. 0 is in bounds, 1 is out of bounds. Next slide shows computation.
Endpoint Coding

Let \((x, y) = (200, -120)\), & viewport shown below

\[
\begin{align*}
(x_{\text{min}}, y_{\text{min}}) &= (-150, -100) \\
(x_{\text{max}}, y_{\text{max}}) &= (150, 100)
\end{align*}
\]

get\(\text{code}(x, y)\):

<table>
<thead>
<tr>
<th>Code: 0000</th>
</tr>
</thead>
<tbody>
<tr>
<td>200 &lt; (-150) : false</td>
</tr>
</tbody>
</table>

\[
\begin{array}{ll}
\text{if} (x < x_{\text{min}}) \text{ then} & \text{code} \leftarrow \text{code}\!\!\uparrow\!0001 \\
\text{if} (x > x_{\text{max}}) \text{ then} & \text{code} \leftarrow \text{code}\!\!\uparrow\!0010 \\
\text{if} (y < y_{\text{min}}) \text{ then} & \text{code} \leftarrow \text{code}\!\!\uparrow\!1000 \\
\text{if} (y > y_{\text{max}}) \text{ then} & \text{code} \leftarrow \text{code}\!\!\uparrow\!0100 \\
\text{return code}
\end{array}
\]
Q. How to determine whether clipping is required?

A. First check whether it is case 1. Is \( \text{code1} \mid \text{code2} = 0000 \) ?

Then check whether it is case 2. Is \( \text{code1} \& \text{code2} \neq 0000 \) ?

If neither, then clip the line and check again.

Q. Can we go over the clipping process taking multiple rounds?

A. Each clip may fix only one bit of one endpoint code. So it may take several rounds to reach the 0000 code for both endpoints.
Cohen-Sutherland Pseudocode

cohen_sutherland(p1,p2):
    code1 = getCode(p1);
    code2 = getCode(p2);
    if (code1|code2 == 0000) then // case 1
        drawLine(p1,p2)
    else if (code1 & code2 != 0000) then // case 2
        // skip line
    else if (code1 & 1000) then // case 3a
        p1new.x ← (ymin-p1.y)/m+p1.x, p1new.y ← ymin,
        cohen_sutherland(p1new,p2)
    else if (code1 & 0100) then // case 3b
        p1new.x ← (ymax-p1.y)/m+p1.x, p1new.y ← ymax,
        cohen_sutherland(p1new,p2)
Cohen-Sutherland Pseudocode, cont.

```
else if (code1 & 0010) then // case 3c
    p1new.y ← m(x_max-p1.x) + p1.y, p1new.x ← x_max,
    cohen_sutherland(p1new,p2)
else if (code1 & 0001) then // case 3d
    p1new.y ← m(x_min-p1.x) + p1.y, p1new.x ← x_min,
    cohen_sutherland(p1new,p2)
else if (code2 & 1000) then // case 3e
    p2new.x ← (y_min-p2.y)/m+p2.x, p2new.y ← y_min,
    cohen_sutherland(p1,p2new)
else if (code2 & 0100) then // case 3f
    p2new.x ← (y_max-p2.y)/m+p2.x, p2new.y ← y_max,
    cohen_sutherland(p1,p2new)
else if (code2 & 0010) then // case 3g
    p2new.y ← m(x_max-p2.x) + p2.y, p2new.x ← x_max,
    cohen_sutherland(p1,p2new)
else if (code2 & 0001) then // case 3h
    p2new.y ← m(x_min-p2.x) + p2.y, p2new.x ← x_min,
    cohen_sutherland(p1,p2new)
```
Cohen-Sutherland Line Clipping

In this problem, you are given a clipping window (defining the “viewport”) and an example line, and asked to perform the steps of the line clipping algorithm we just learned in class. Assume the origin is at the top left and y is increasing going down (like HTML canvas).

Input: viewport defined by the lines $x_{\text{min}} = 2$, $x_{\text{max}} = 10$, $y_{\text{min}} = 3$, $y_{\text{max}} = 8$.
    line defined by the points $p_1 = (1, 5)$ and $p_2 = (11, 1)$.
Output: $p_1'$ and $p_2'$, the points defining the line that should actually be drawn.

1. Draw out the viewport and the example line, labeling $p_1, p_2, x_{\text{min}}, x_{\text{max}}, y_{\text{min}},$ and $y_{\text{max}}$.
2. Write out the binary 4-digit codes for $p_1$ and $p_2$.
3. Write out what case each point falls under and show how the algorithm would update the points. What are the final $p_1'$ and $p_2'$?
4. Label $p_1'$ and $p_2'$ on your picture and make sure they agree visually with your calculations.

5. How many “rounds” of clipping are required to make this example line within the viewport?
6. Why are there eight subparts to case 3? What does each one represent/do?

7. A four-bit sequence allows for sixteen possible values, yet we only have nine regions. Which bit sequences are not used, and why do they represent nonsensical situations?

8. Which case would be activated for each of the following pairs of codes?
   a. 1010 and 0101
   b. 0000 and 1001
   c. 0110 and 0101
   d. 0001 and 0001