Answer 1: Scissoring

- Compute pixels in world grid
- Then check chosen pixels against viewport boundaries

Problem: Inefficient

1. We could go back to using floating point operations.
2. We could round the endpoint coordinates and scan convert.
3. We could do it the right way.
The right way

```
SpecialCaseBresenham(x_1, y_1, x_{first}, x_{last})
  m = y_1 / x_1
  i = x_{first}, j = round(m * x_{first})
  d = x_1(2j+1) - 2y_1(i+1)
  while i < x_{last}
    writePixel(i, j)
    if d < 0
      i += 1, j += 1, d += 2(x_1 - y_1)
    else
      i += 1, d -= 2y_1
```

Answer 2: Analytical Clipping

- Compute intersection points
- Scan convert line segment between intersection points

Intersection of lines: y = mx and x = c

Naïve approach

- Compute 4 intersection points:
  1. (XMIN, mXMIN+b)
  2. ((YMAX-b)/m, YMAX)
  3. (XMAX, mXMAX+b)
  4. ((YMIN-b)/m, YMIN)

Naïve approach cont.

- BOUND CHECK:
  1. (XMIN, mXMIN+b):
     YMIN ≤ mXMIN+b ≤ YMAX?
  2. ((YMAX-b)/m, YMAX)
  3. (XMAX, mXMAX+b)
  4. ((YMIN-b)/m, YMIN)

Problem?

Step 1 - Cohen-Sutherland

- Assign an “outcode” to each endpoint
- Example:
  Endpoint (u, v) and viewport boundaries: x = Xmin, x = Xmax, y = Ymin, y = Ymax.
  y > Y_{max} (v > Y_{max})
  x < X_{min} (u < X_{min})
  y < Y_{min} (v < Y_{min})
  x > X_{max} (u > X_{max})
Outcode Example (1)

\[(u,v):0000\]
\[(w,z):0000\]

Outcode Example (2)

\[(u,v):0101\]
\[(w,z):0110\]

Outcode Example (3)

\[(u,v):1001\]
\[(w,z):0110\]

Step 1:
Endpoints \(p_0\) and \(p_1\), outcodes \(C_0\) and \(C_1\)

\[
\begin{array}{c|c|c|c}
000 & 001 & 010 & 011 \\
010 & 011 & 100 & 101 \\
100 & 101 & 110 & 111 \\
110 & 111 & 111 & 111 \\
\end{array}
\]

- \(C_0 \land C_1 = 0\)
- \(C_0 \land C_1 = 0\)

Trivial Reject     Trivial Accept

If Step 1 fails to resolve then Step 2:
Endpoints \(p_0\) and \(p_1\), outcodes \(C_0\) and \(C_1\)

- Choose an “out” endpoint and clip
- Return to step 1 with new line segment

Example: (u,v) outcode is 0101 so clip to \(y=YMIN\)

How many clips are needed

- Before reject?
- Before accept?
Overview of Scan Conversion

Endpoints: \((x_0, y_0), (x_1, y_1)\)

First/last pixels:

- \((x_{\text{first}}, y_{\text{first}})\)
- \((x_{\text{last}}, y_{\text{last}})\)

“ON” pixels

Overview of Clipper

Endpoints: \((x_0, y_0), (x_1, y_1)\)

Opcode: compute and test

Clip out endpoint

Round

Accept

Reject

Overview of Scan Converter

Endpoints: \((x_0, y_0), (x_1, y_1)\)

First/last pixels:

- \(x_{\text{first}}/y_{\text{first}}\)
- \(x_{\text{last}}/y_{\text{last}}\)

“ON” pixels

Assignment 1

- Implement Special Case Bresenham’s 20
- Generalize to arbitrary lines 35
- Dashed lines obeying endpoint order 10
- Clip to viewport boundaries 15
- Use opcodes 10
- Use iterative opcode test 10

Anti-aliasing (Getting Rid of the Jaggies)

Fundamental Problem: pixel area/line length

Open GL

- OpenGL: Primitive graphics operations
- GLUT: Interactions with window environment

- Lab exercise coming up!