# Computer Graphics - Exercise 09

Maximilian Richter

Winter Semester 2022/23

## 1 Polygon Clipping

What is the maximum number of new vertices, if

a) a n-sided convex polygon is clipped with a line?
Answer: 2

b) a n-sided non-convex polygon is clipped with a line?
Answer: $n$

c) a n-sided convex polygon is clipped with a rectangle?
Answer: 8

d) a n-sided non-convex (possibly self-intersecting) polygon is clipped with a rectangle?
Answer: $2n$

## 2 Sutherland–Hodgman Algorithm

The following points are given and labeled:

$$P_1 = \left(-\frac{1}{2}, -\frac{1}{2}\right), \quad P_2 = \left(\frac{3}{2}, -1\right), \quad P_3 = \left(\frac{1}{2}, \frac{3}{2}\right), \quad P_4 = \left(-\frac{3}{2}, \frac{1}{2}\right).$$

These points make an polygon which is supposed to be clipped to a square with extent $(-1, -1, 1, 1)$

| **Edge** | **Case** | **Output** |
|---|---|---|
| P1, P2 | Outgoing | P2'=(1,7/8) |
| P2, P3 | Incoming | P2"=(1,1/4), P3 |
| P3, P4 | Inside | P4 |
| P4, P1 | Inside | P1 |

Table 1: Right

| **Edge** | **Case** | **Output** |
|---|---|---|
| P1, P2' | Inside | P2' |
| P2', P2" | Inside | P2" |
| P2", P3 | Outgoing | P3'=(7/10,1) |
| P3, P4 | Incoming | P3"=(1/2,1), P4 |
| P4, P1 | Inside | P1 |

Table 2: Up

| **Edge** | **Case** | **Output** |
|---|---|---|
| P1, P2' | Inside | P2' |
| P2', P2" | Inside | P2" |
| P2", P3' | Inside | P3' |
| P3', P3" | Inside | P3" |
| P3", P4 | Outgoing | P4'=(1,3/4) |
| P4, P1 | Incoming | P4"=(-1, 0), P1 |

Table 3: Left

| **Edge** | **Case** | **Output** |
|---|---|---|
| P1, P2' | Inside | P2' |
| P2', P2" | Inside | P2" |
| P2", P3' | Inside | P3' |
| P3', P3" | Inside | P3" |
| P3", P4' | Inside | P4' |
| P4', P4" | Inside | P4" |
| P4", P1 | Inside | P1 |

Table 4: Bottom

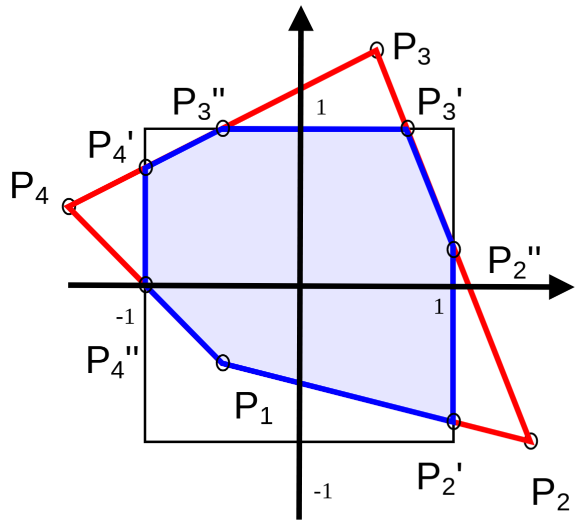


Figure 1: Clipped Polygon

# 3 Bresenham Algorithm

The Bresenham algorithm is a frequently used method in computer graphics to rasterize lines and circles. The implicit representation of a line between two given points in space is given as

$$F(x, y) = y(x_1 - x_0) + x(y_0 - y_1) + y_1 x_0 - y_0 x_1 \tag{1}$$

a) Using the line equation for $P_0 = (x_0, y_0) = (1, 2)$ and $P_1 = (x_1, y_1) = (6, 4)$ we obtain an implicit representation of the given line

$$F(x, y) = 5y - 2x - 8$$

We initialize out control variable $d$ as $2F(x_0 + 1, y_0 + \frac{1}{2})$

$$\begin{aligned} d :&= 2F(x_0 + 1, y_0 + \frac{1}{2}) \\ &= 2(5(y_0 + \frac{1}{2}) - 2(x_0 + 1)8) \\ &= 10y_0 - 4x_0 - 15 \end{aligned}$$

We use $P_0 = (1, 2)$ and yield

$$d_0 = 20 - 4 - 15 = 1$$

Since $d \geq 0$ holds, the Bresenham algorithm indicates to go east (E) and the incremented $d$ is calculated by

$$d_1 = d_0 + 2(y_0 - y_1) = 1 + 2(2 - 4) = -3$$

Since $d < 0$ holds, the Bresenham algorithm indicates to go northeast (NE) and the incremented $d$ is calculated by

$$d_2 = d_1 + 2(y_0 - y_1) + 2(x_1 - x_0) = -3 + 2(2 - 4) + 2(6 - 1) = 3$$

$d \geq 0 \rightarrow$ E

$$d_3 = d_2 + 2(y_0 - y_1) = -1$$

$d < 0 \rightarrow$ NE

$$d_4 = d_3 + 2(y_0 - y_1) + 2(x_1 - x_0) = 5$$

$d \geq 0 \rightarrow$ E

As the formulas used above are specified for the case that the slope of the line is smaller than 1, we have to exploit symmetrie and thus switch $x$ and $y$ axes for the calculations. Therefore $P_0 = (x_0, y_0) = (2, 1)$ becomes $(1, 2)$ and $P_1 = (x_1, y_1) = (5, 5)$ becomes $(5, 5)$.

$$F(x, y) = 4y - 3x - 5$$

We initialize our control variable $d$

$$\begin{aligned} d := &\, 2F(x_0 + 1, y_0 + \frac{1}{2}) \\ = &\, 2(4(y_0 + \frac{1}{2}) - 3(x_0 + 1)5) \\ = &\, 8y_0 - 6x_0 - 12 \end{aligned}$$

We use $P_0 = (1, 2)$ and yield

$$d_0 = 16 - 6 - 12 = -2$$

$d < 0 \rightarrow$ NE

$$d_1 = d_0 + 2(y_0 - y_1) + 2(x_1 - x_0) = -2 + 2(5 - 2) + 2(5 - 1) = 0$$

$d \geq 0 \rightarrow$ E

$$d_2 = d_1 + 2(y_0 - y_1) = -6$$

$d < 0 \rightarrow$ NE

$$d_3 = d_2 + 2(y_0 - y_1) + 2(x_1 - x_0) = -4$$

$d < 0 \rightarrow$ NE

$$d_4 = d_3 + 2(y_0 - y_1) + 2(x_1 - x_0) = -2$$

Depending on the stopping criterion and the chosen point, the end of the line will look slightly different.

b) For rasterization of the first line using anti-aliasing we need the signed distance a between the line and the center between the E and NE pixel. On the basis of $a$ one can then decide which pixel should be set with which intensity. $a$ can be determined by

$$a = \frac{d}{2\Delta x} \quad \text{with} \quad \Delta x = x_1 - x_0$$

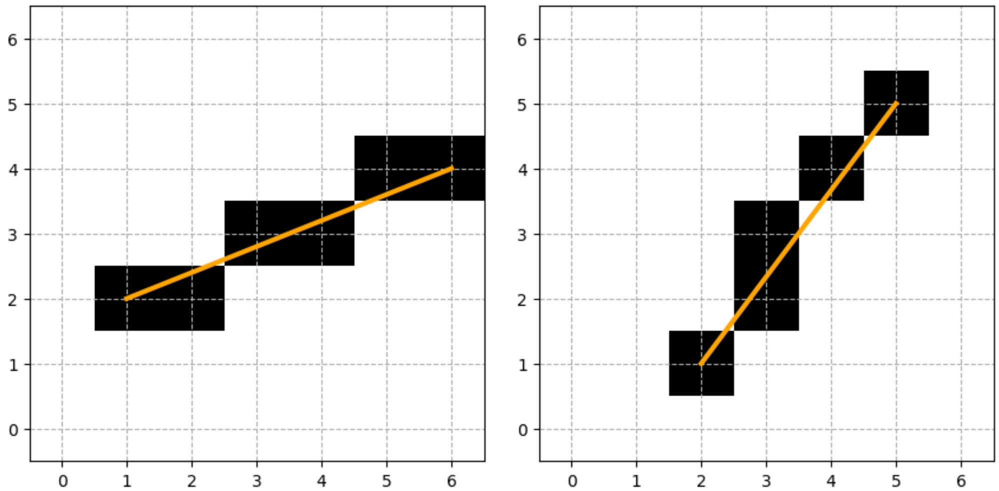


Figure 2: Result of Rasterization Line 1 (left) and Line 2 (right)

For every $d$ we calculate $a$ and use the appropriate functions to determine the fraction of the intensity for the East and North East pixel

- Step 1: $d = 1, a = -0.05 \rightarrow (x+1, y)$ with 0.55 and $(x+1, x+1)$ with 0.45
- Step 2: $d = -3, a = -0.15 \rightarrow (x+1, y)$ with 0.35 and $(x+1, x+1)$ with 0.65
- Step 3: $d = 3, a = 0.15 \rightarrow (x+1, y)$ with 0.65 and $(x+1, x+1)$ with 0.35
- Step 4: $d = -1, a = --0.05 \rightarrow (x+1, y)$ with 0.45 and $(x+1, x+1)$ with 0.55

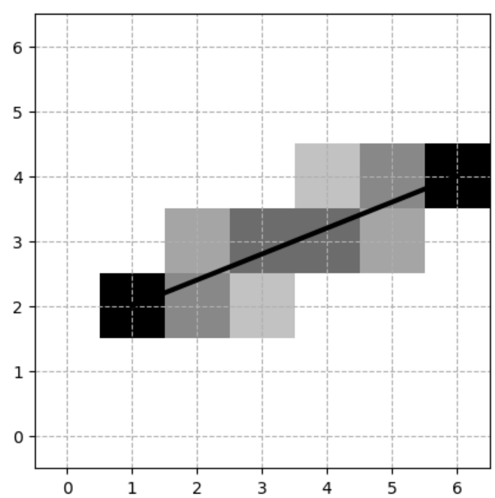


Figure 3: Result of Anti-aliasing Line 1