Scanline Fill Algorithm

- Intersect scanline with polygon edges
- Fill between pairs of intersections
- Basic algorithm:

For \( y = \text{ymin} \) to \( \text{ymax} \)

1) intersect scanline \( y \) with each edge
2) sort intersections by increasing \( x \)
3) fill pairwise (\( p_0 \rightarrow p_1, p_2 \rightarrow p_3, \ldots \))
However, we need to handle some special cases and improve the performance

**Special handling:**

a) Make sure we only fill the interior pixels

Define interior:

For a given pair of intersecting points \((X_i, Y), (X_j, Y)\)

\[-\rightarrow\] Fill ceiling\((X_i)\) to floor\((X_j)\)

important when we have polygons adjacent to each other

b) Intersection has an integer X coordinate

\[-\rightarrow\] if \(X_i\) is integer, we define it to be interior
\[-\rightarrow\] if \(X_j\) is integer, we define it to be exterior (so don’t fill)
Special handling (cont’d)

c) Intersection is an edge end point

Intersection points: \((p_0, p_1, p_2)\) ???

\(\rightarrow\) \((p_0, p_1, p_1, p_2)\) so we can still fill pairwise

\(\rightarrow\) In fact, if we compute the intersection of the scanline with edge \(e_1\) and \(e_2\) separately, we will get the intersection point \(p_1\) twice. Keep both of the \(p_1\).
Special handling (cont’d)

c) Intersection is an edge end point (cont’d)

However, in this case we don’t want to count p1 twice \((p0,p1,p1,p2,p3)\), otherwise we will fill pixels between p1 and p2, which is wrong
c) Intersection is an edge end point (cont’d)

Rule:

If the intersection is the ymin of the edge’s endpoint, count it. Otherwise, don’t.