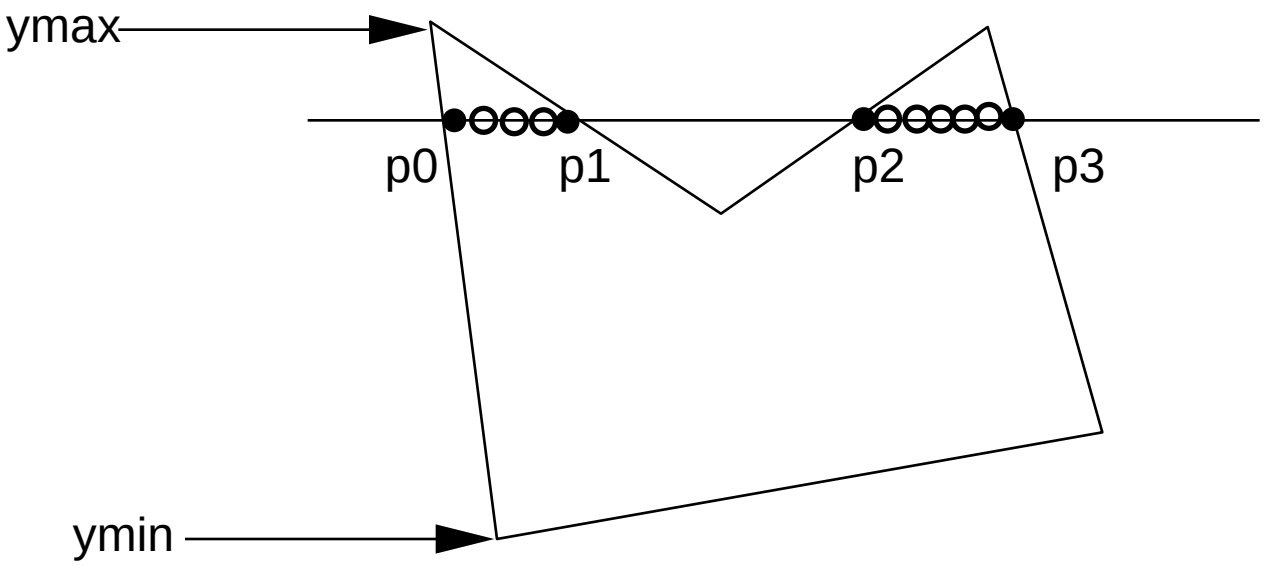
# Scanline Fill Algorithm

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

For y = ymin to ymax

1) intersect scanline y with each edge
2) sort interesections by increasing x
   [p0,p1,p2,p3]
3) fill pairwise (p0 −> p1, p2−> p3, ....)



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 intersectin points
(Xi, Y), (Xj, Y)

−> Fill ceiling(Xi) to floor(Xj)

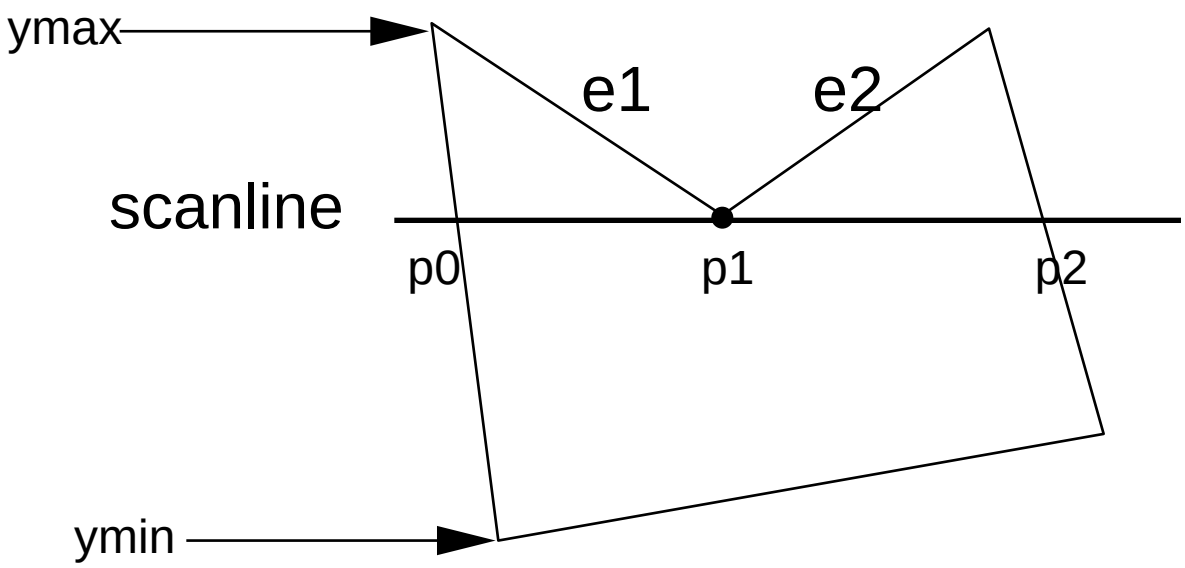
important when we have polygons adjacent to each other

b) Intersection has an integer X coordinate

−> if Xi is integer, we define it to be interior
−> if Xj is integer, we define it to be exterior
(so don’t fill)

Special handling (cont'd)

c) Intersection is an edge end point



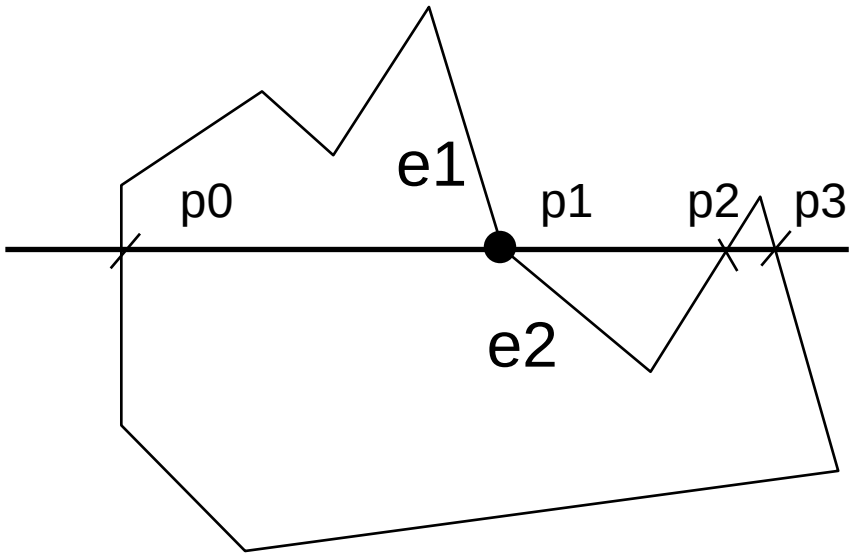

Intersection points: (p0, p1, p2) ???

−> (p0,p1,p1,p2) so we can still fill pairwise

−> In fact, if we compute the intersection of the scanline with edge e1 and e2 separately, we will get the intersection point p1 twice. Keep both of the p1.

Special handling (cont'd)

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



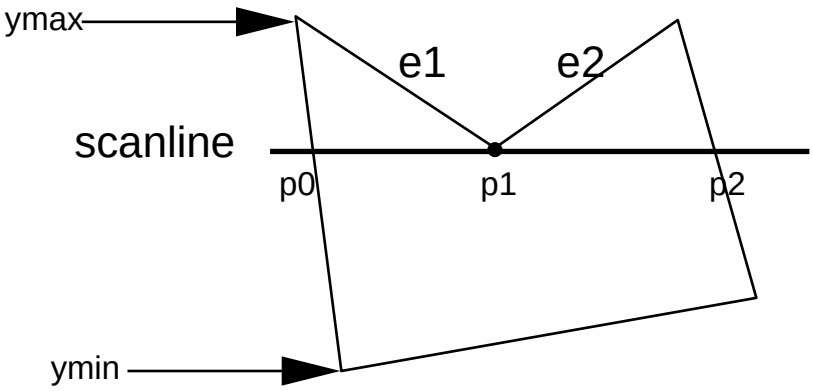

However, in this case we don't want to count p1 twice (p0,p1,p1,p2,p3), otherwise we will fill pixels between p1 and p2, which is wrong

## Special handling (cont'd)

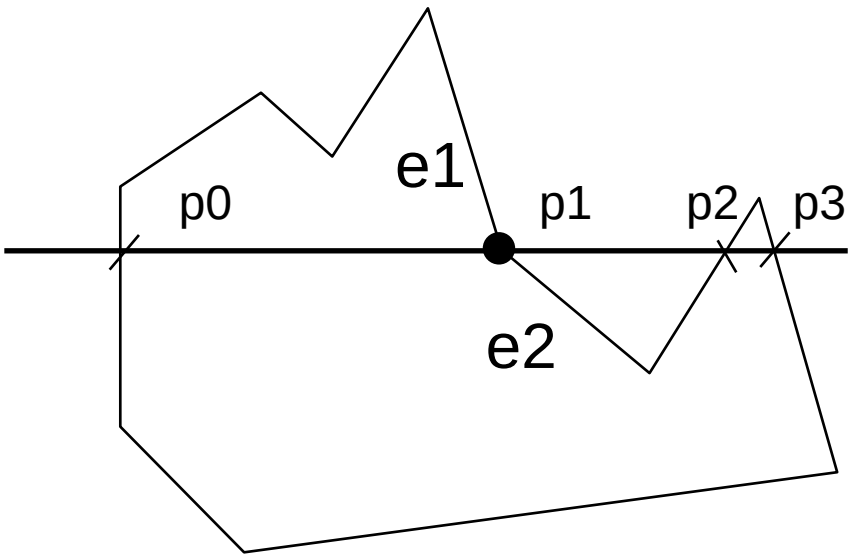
### c) Intersection is an edge end point (cont'd)

Rule:

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




Yes, count p1 for both e1 and e2




No, don't count p1 for the edge e2