ΗΥ416
ΓΡΑΦΙΚΑ ΥΠΟΛΟΓΙΣΤΩΝ

Αποκοπή

Π. ΤΣΟΜΠΑΝΟΠΟΥΛΟΥ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ & ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Liang-Barsky Algorithm

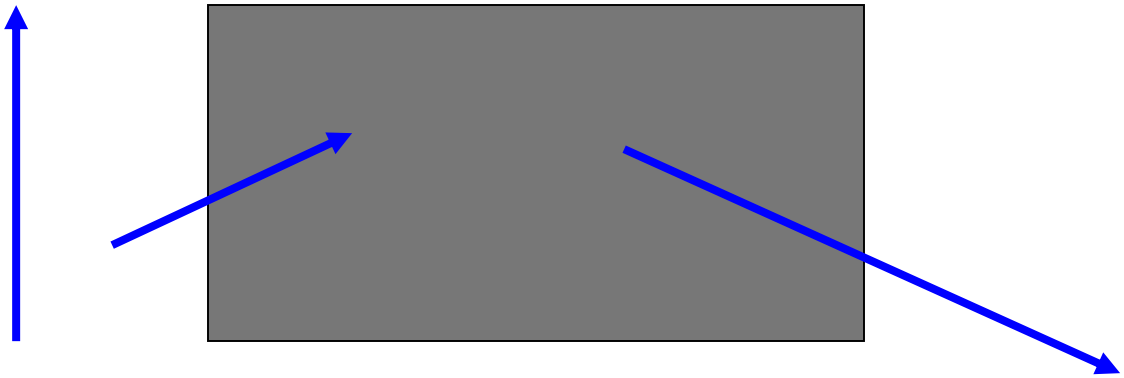
Liang-Barsky Algorithm

- Cyrus-Beck Algorithm, earlier approach
- Liang – Barsky Algorithm, later-better-independent approach
- Parametric definition of a line:
  - $x = x_1 + u\Delta x$
  - $y = y_1 + u\Delta y$
  - $\Delta x = (x_2 - x_1)$, $\Delta y = (y_2 - y_1)$, $0 <= u <= 1$
- Lines are oriented: classify lines as moving inside to out or outside to in

- Goal: find range of u for which x and y *both* inside the viewing window

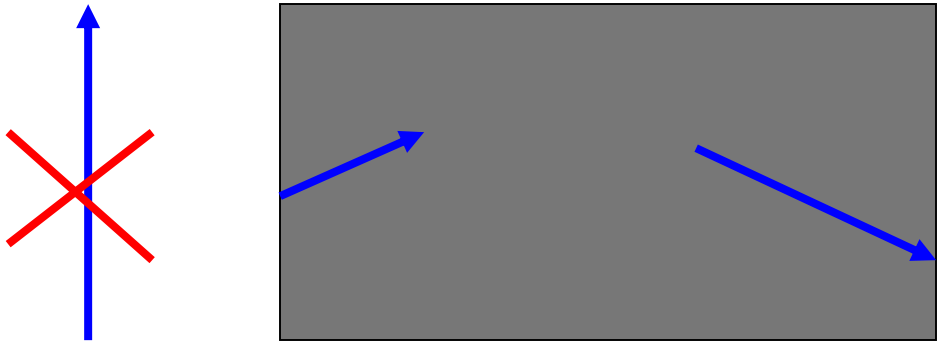
Liang-Barsky Algorithm

- For lines starting outside of boundary, update its starting point (u_1)
- For lines starting inside of boundary, update end point (u_2)
- For lines paralleling the boundaries and outside window, reject it.



Liang-Barsky Algorithm

- For lines starting outside of boundary, update its starting point (u_1)
- For lines starting inside of boundary, update end point (u_2)
- For lines paralleling the boundaries and outside window, reject it.



Liang-Barsky Algorithm

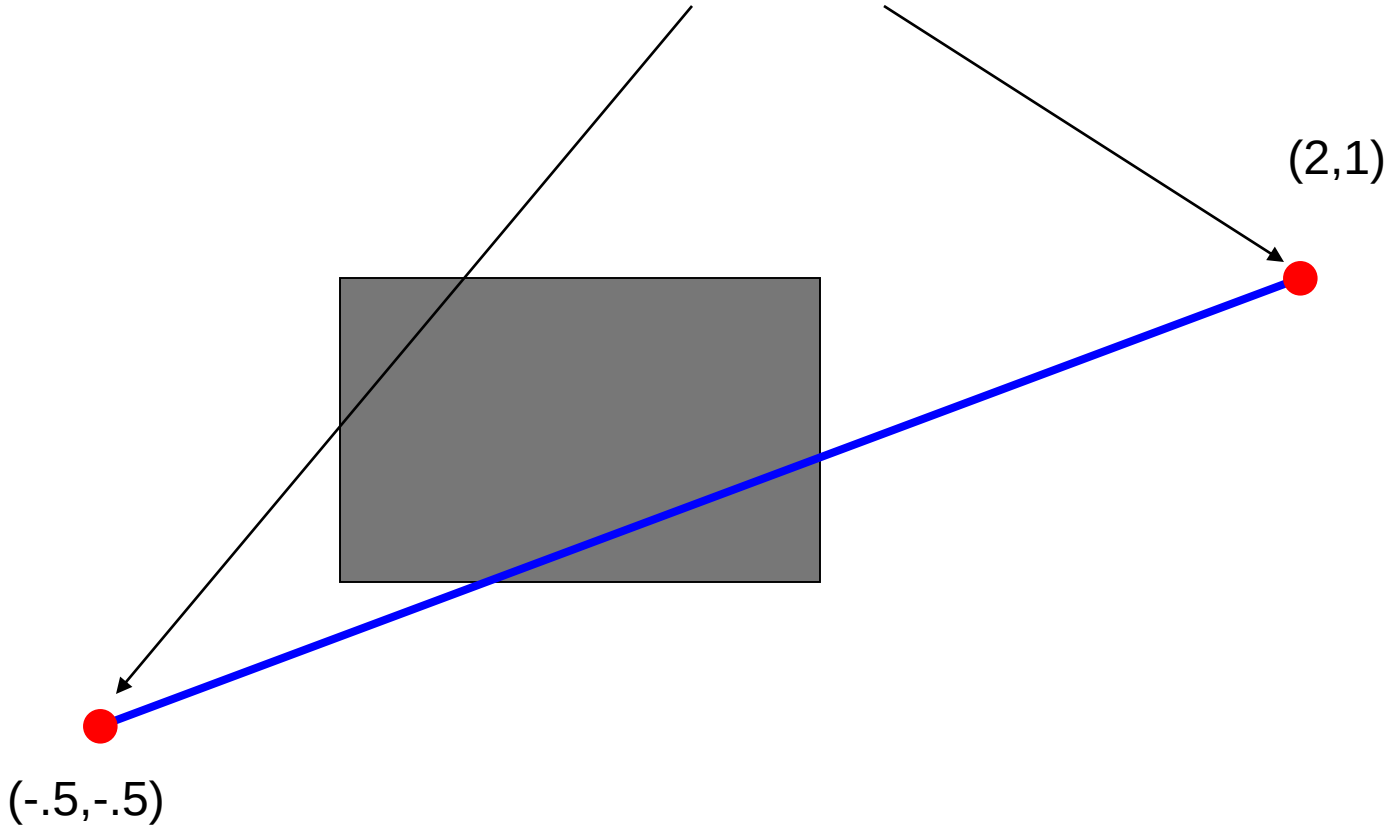
- Mathematically:
  - $x_{min} <= x_1 + u\Delta x <= x_{max}$
  - $y_{min} <= y_1 + u\Delta y <= y_{max}$
- Rearranged
  - 1: $u*(-\Delta x) <= (x_1 - x_{min})$
  - 2: $u*(\Delta x) <= (x_{max} - x_1)$
  - 3: $u*(-\Delta y) <= (y_1 - y_{min})$
  - 4: $u*(\Delta y) <= (y_{max} - y_1)$
  - gen: $u*(p_k) <= (q_k)$, k=1,2,3,4

Liang-Barsky Algorithm

- Initial values: $u_1 = 0$, $u_2 = 1$
- Rules:
  1. $p_k = 0$: the line is parallel to boundaries
     - If for that same k, $q_k < 0$, it's outside
     - Otherwise it's inside
  2. $p_k < 0$: the line starts outside this boundary
     - $r_k = q_k/p_k$
     - $u_1 = \max(0, r_k, u_1)$ /**update starting point**/
  3. $p_k > 0$: the line starts inside the boundary
     - $r_k = q_k/p_k$
     - $u_2 = \min(1, r_k, u_2)$ /** update end point**/
  4. If $u_1 > u_2$, the line is completely outside

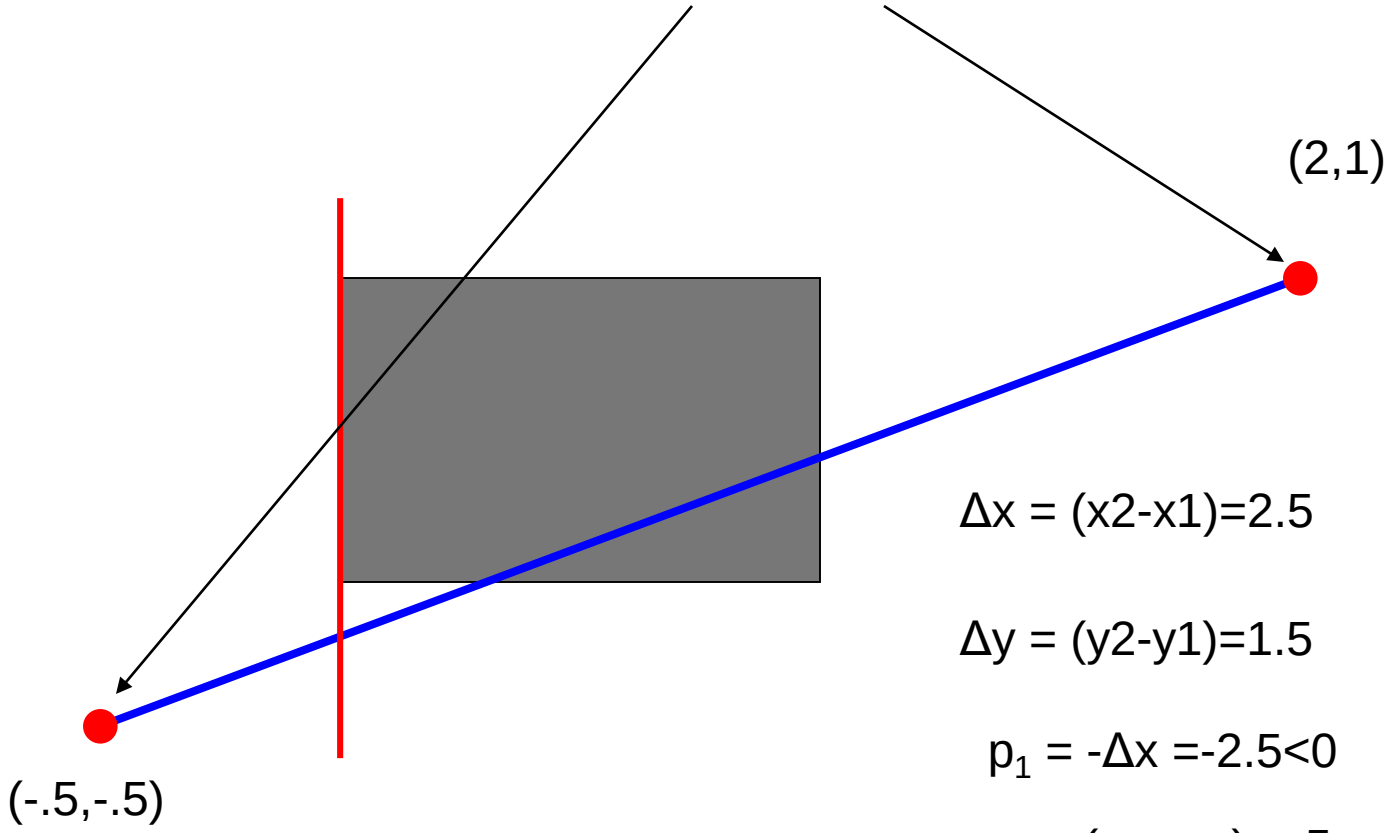
Liang-Barsky Algorithm

- Current clipped line (u_1=0,u_2=1)



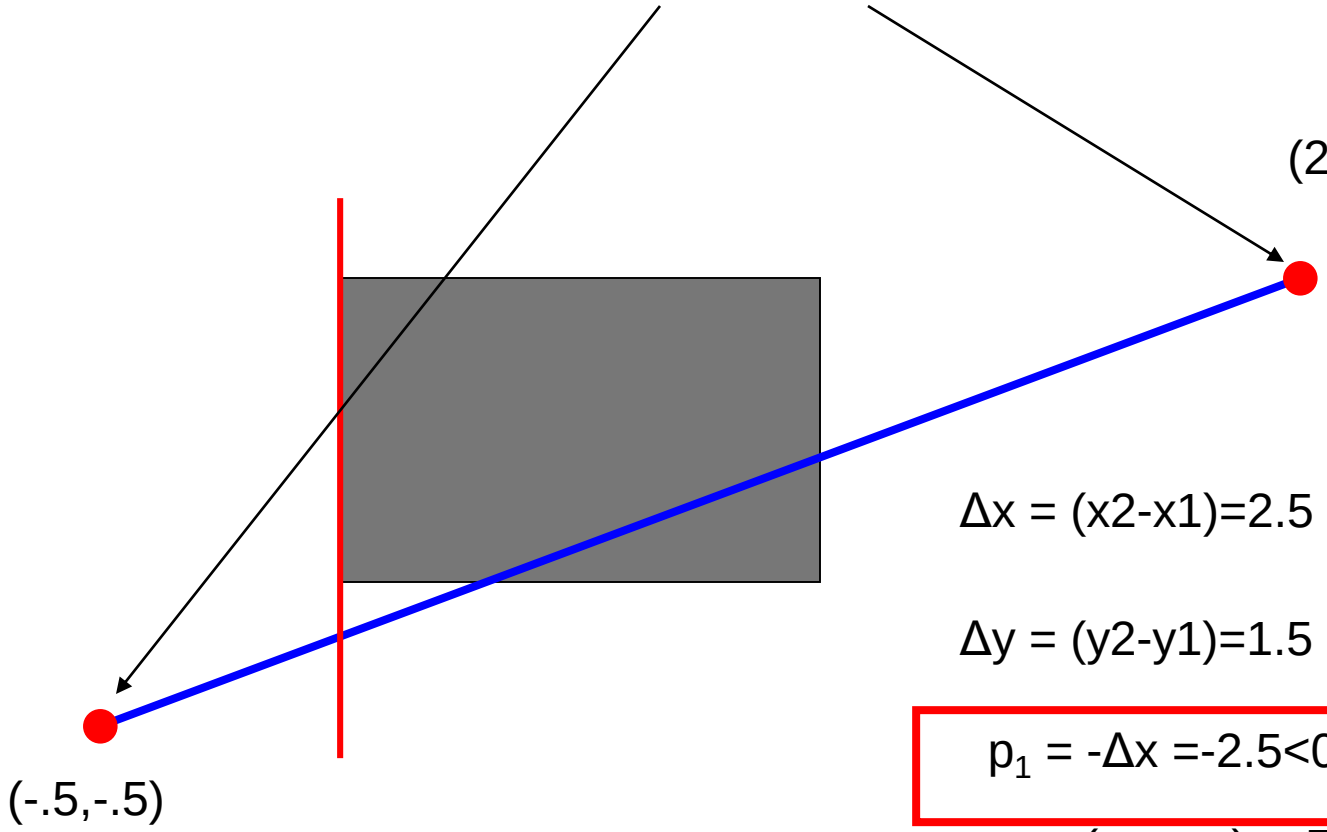
Liang-Barsky Algorithm

- Current clipped line (u_1=0,u_2=1)



Liang-Barsky Algorithm

- Current clipped line (u_1=0,u_2=1)



$\Delta x = (x2-x1)=2.5$

$\Delta y = (y2-y1)=1.5$

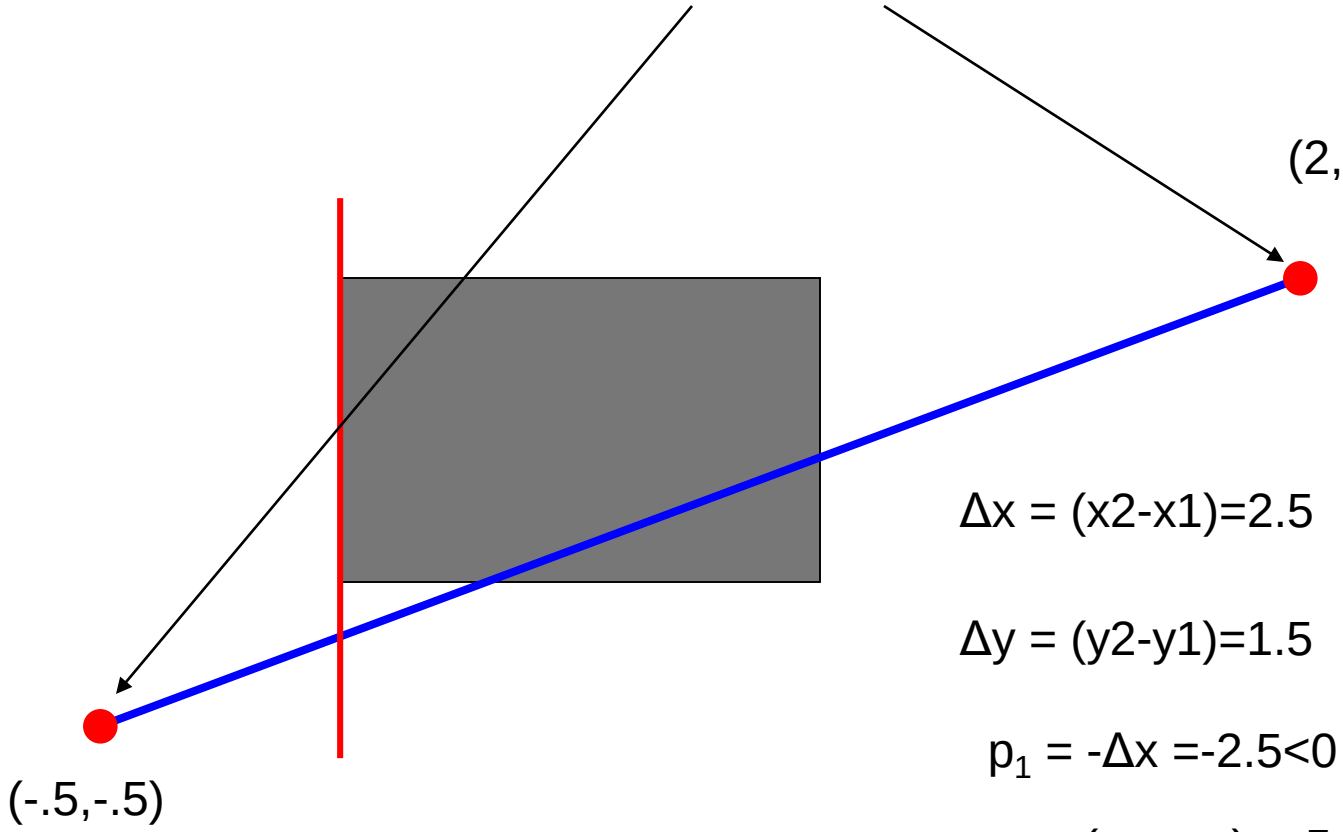
$p_1 = -\Delta x = -2.5 < 0$

$q_1 = (x_1-x_{min})=-.5$

$r_1 = q_1/p_1=0.2$

Liang-Barsky Algorithm

- Current clipped line (u_1=0,u_2=1)



$\Delta x = (x2-x1)=2.5$

$\Delta y = (y2-y1)=1.5$

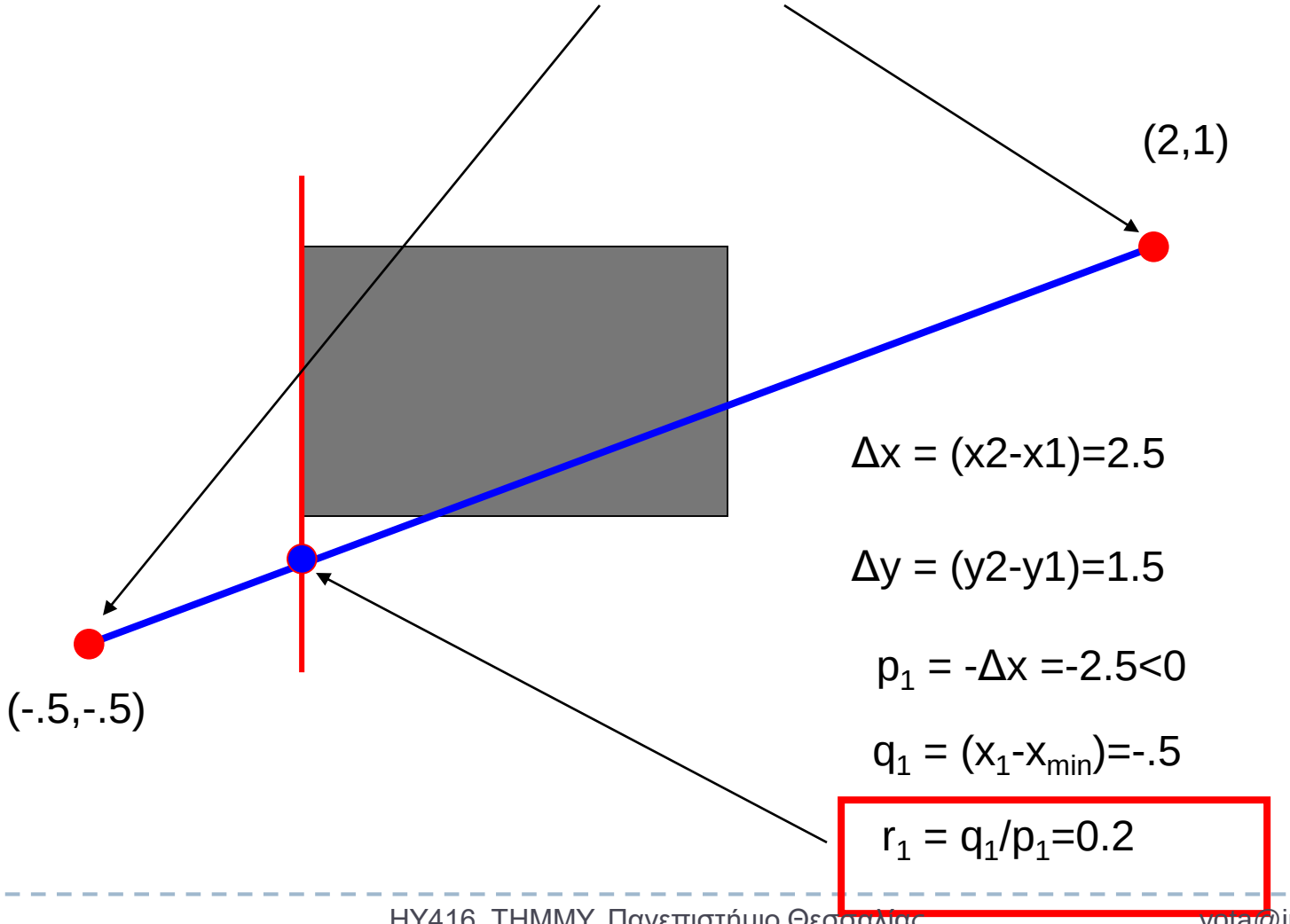
$p_1 = -\Delta x = -2.5 < 0$

$q_1 = (x_1 - x_{min}) = -.5$

$r_1 = q_1/p_1 = 0.2$

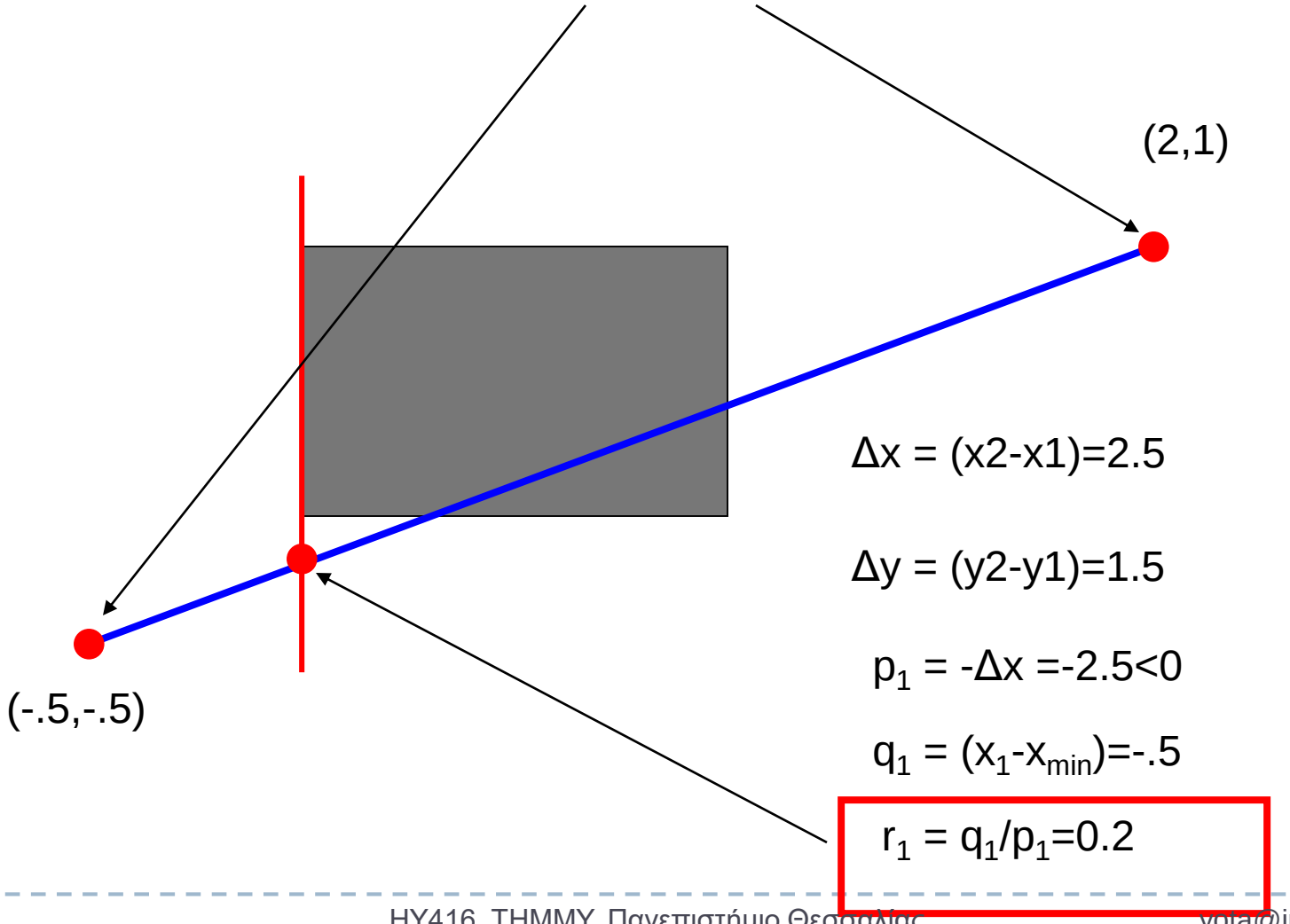
Liang-Barsky Algorithm

- Current clipped line (u_1=0,u_2=1)



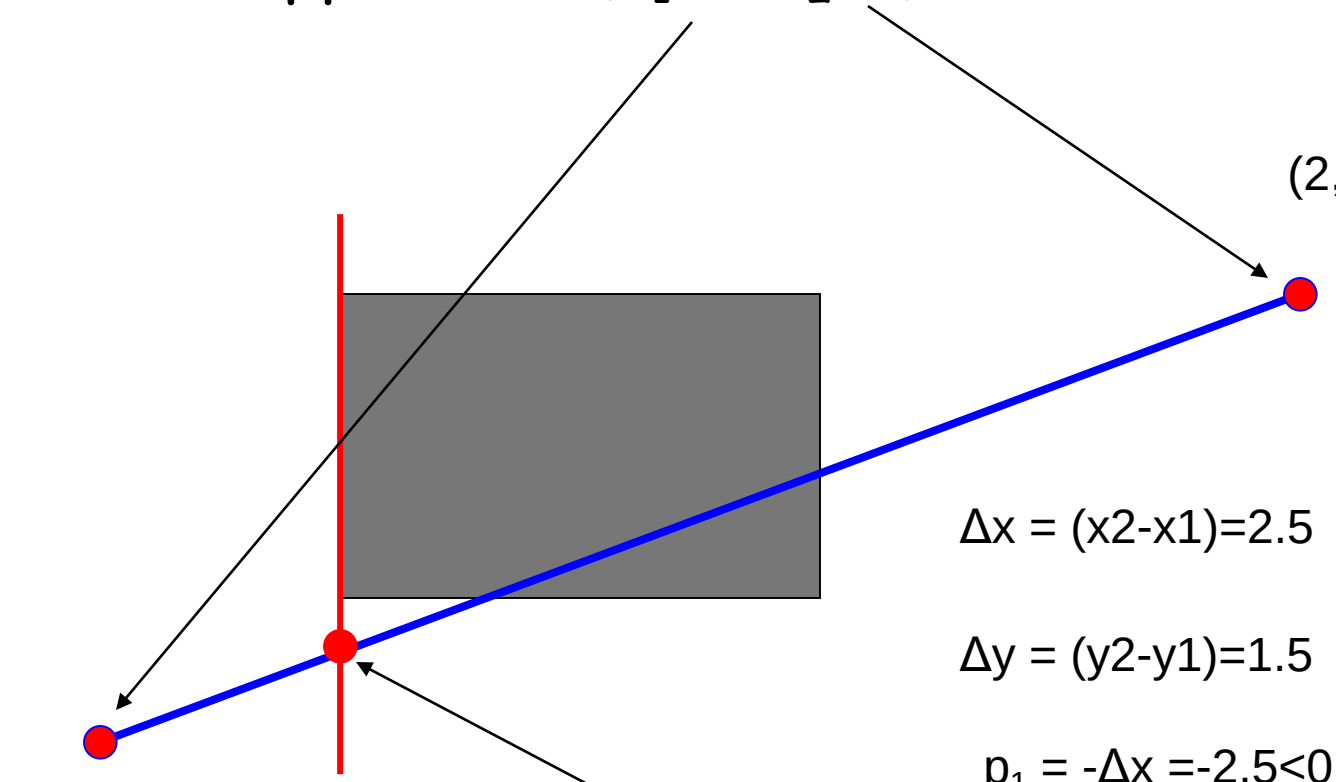
Liang-Barsky Algorithm

- Current clipped line (u_1=0,u_2=1)



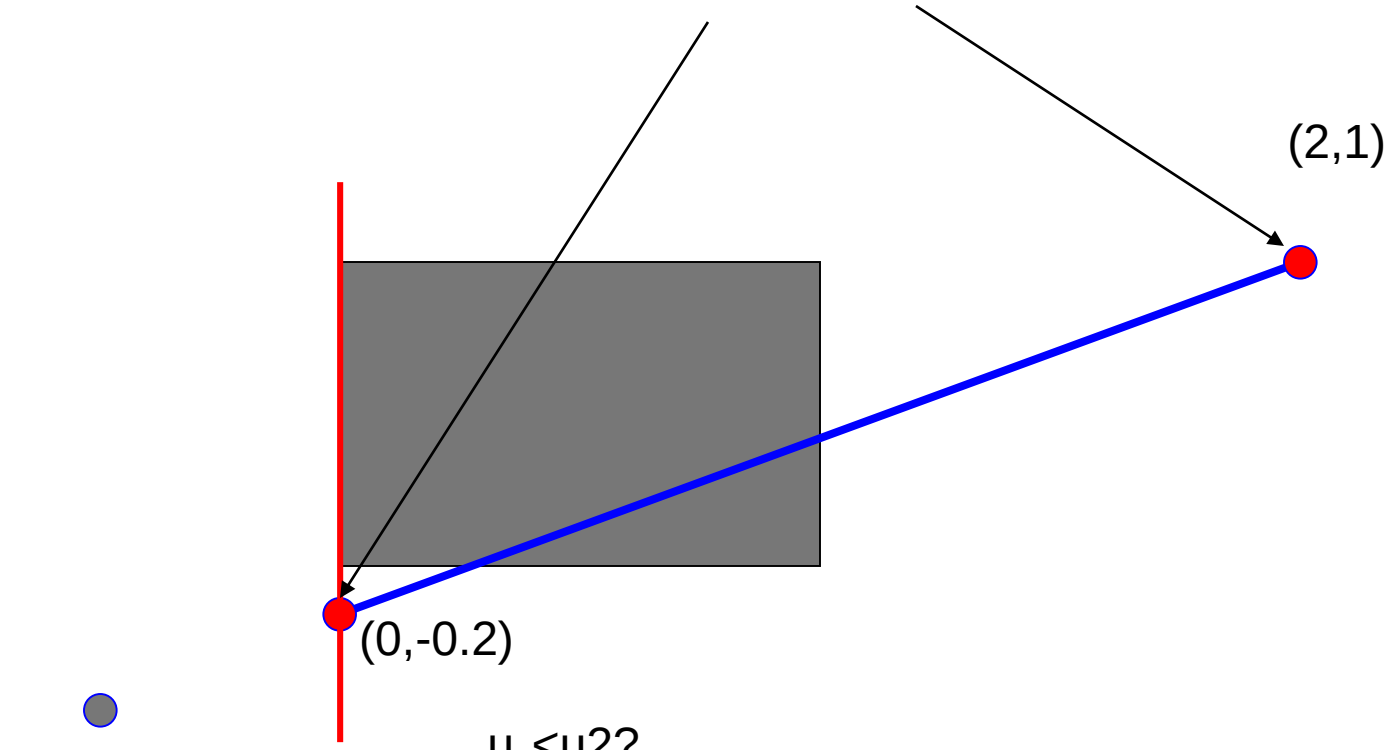
Liang-Barsky Algorithm

- Current clipped line (u_1=0,u_2=1)



Liang-Barsky Algorithm

- Current clipped line (u_1=0.2,u_2=1)



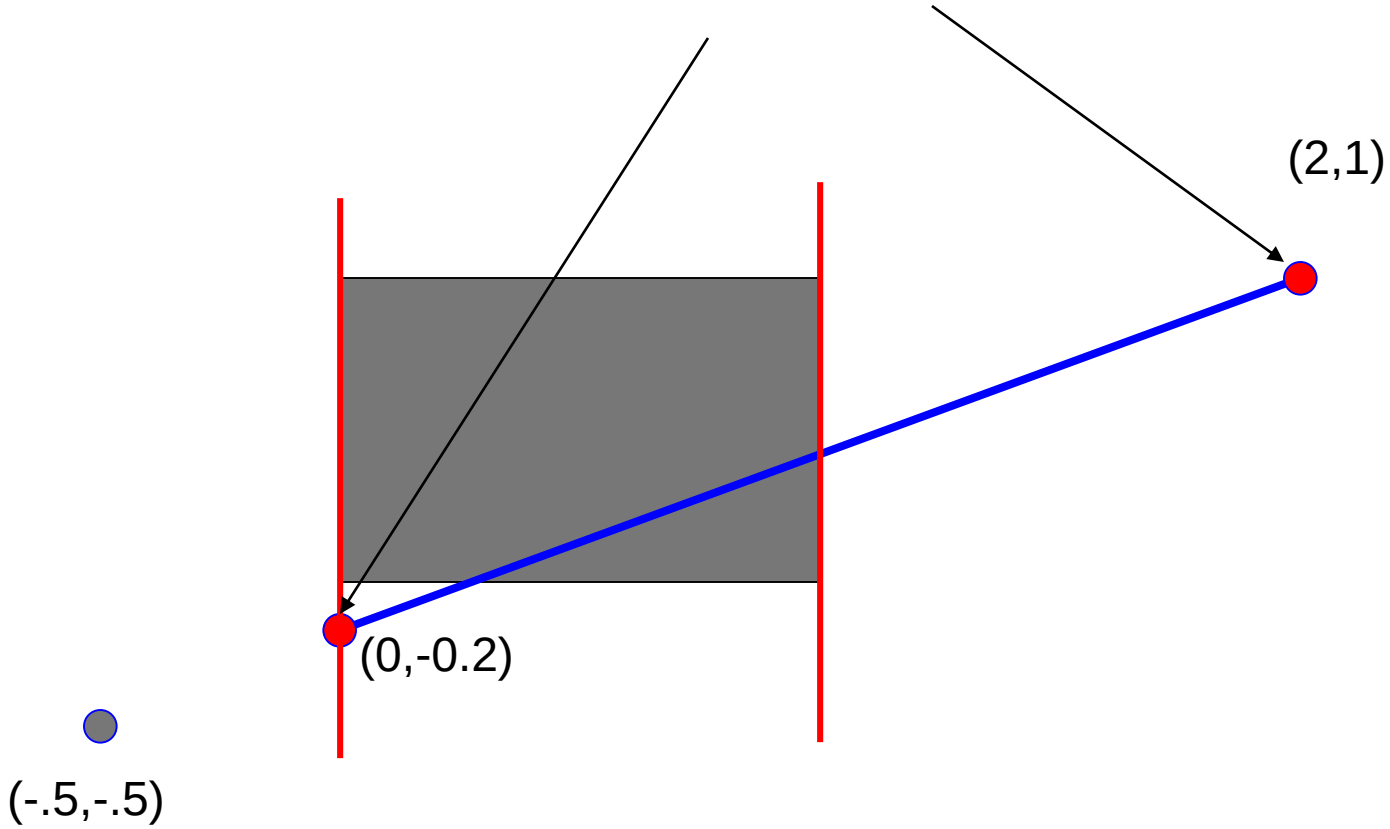
u_1<u2?

- yes: continue

- no: the line is completely outside window

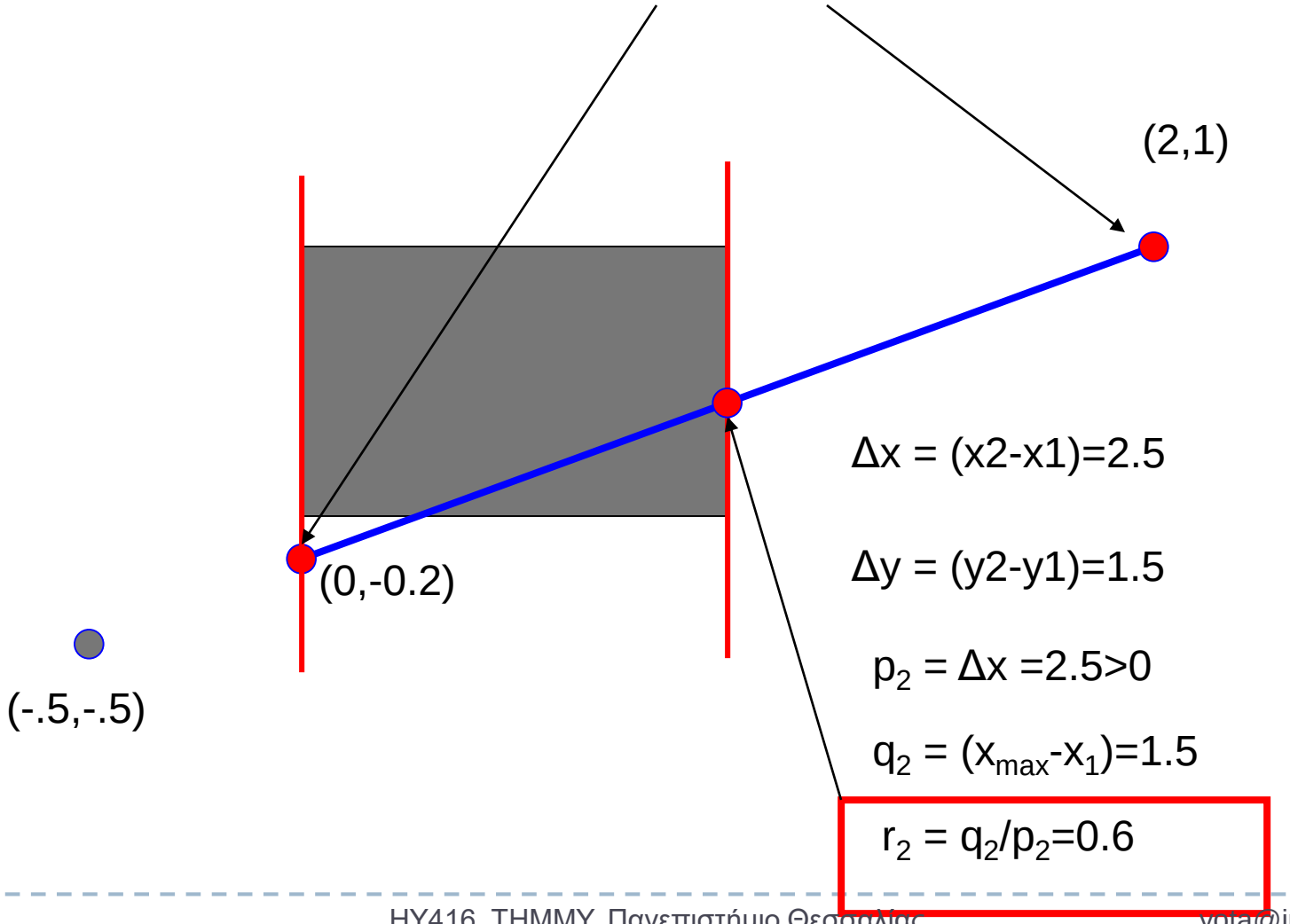
Liang-Barsky Algorithm

- Current clipped line (u_1=0.2,u_2=1)



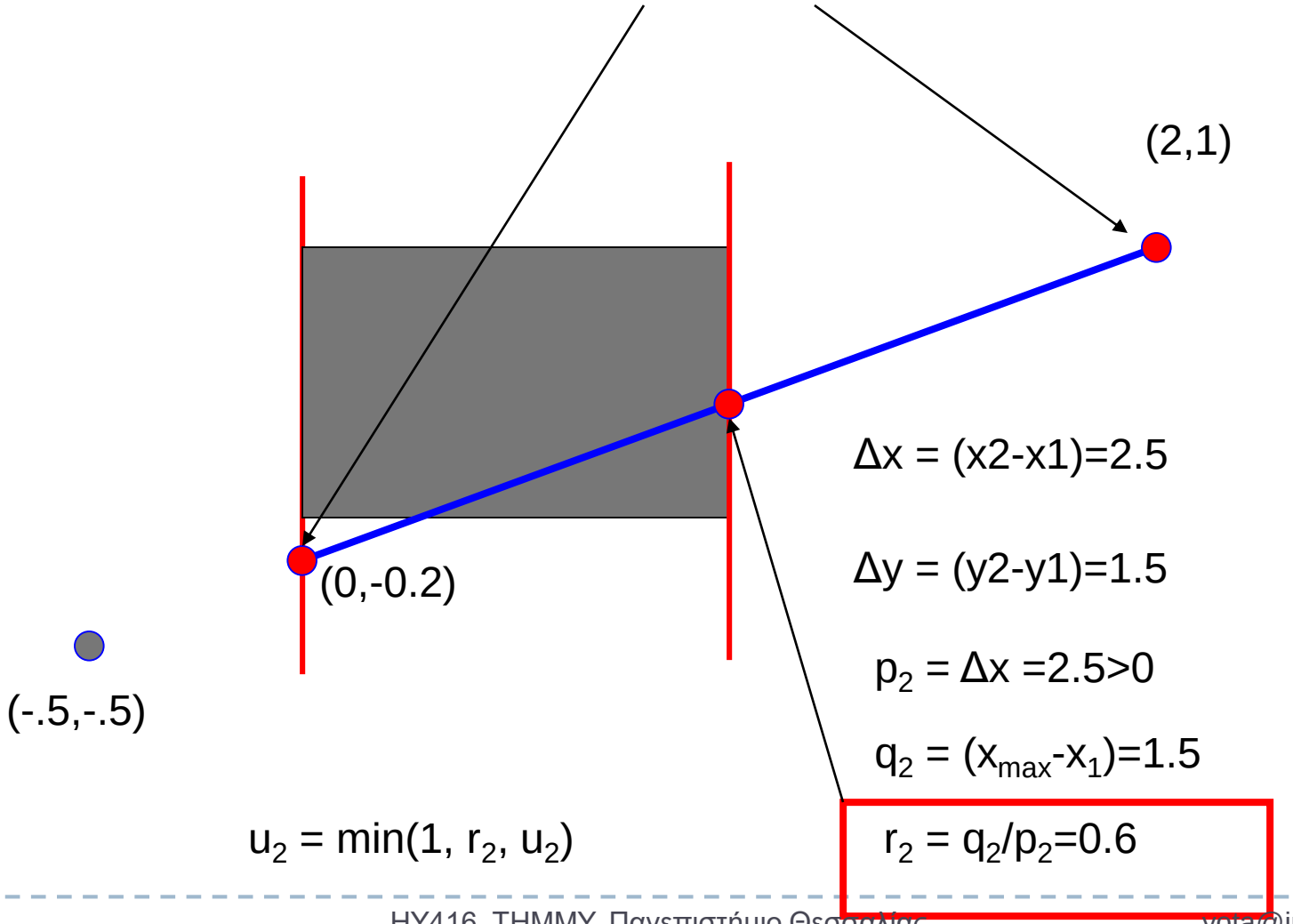
Liang-Barsky Algorithm

- Current clipped line (u_1=0.2,u_2=1)



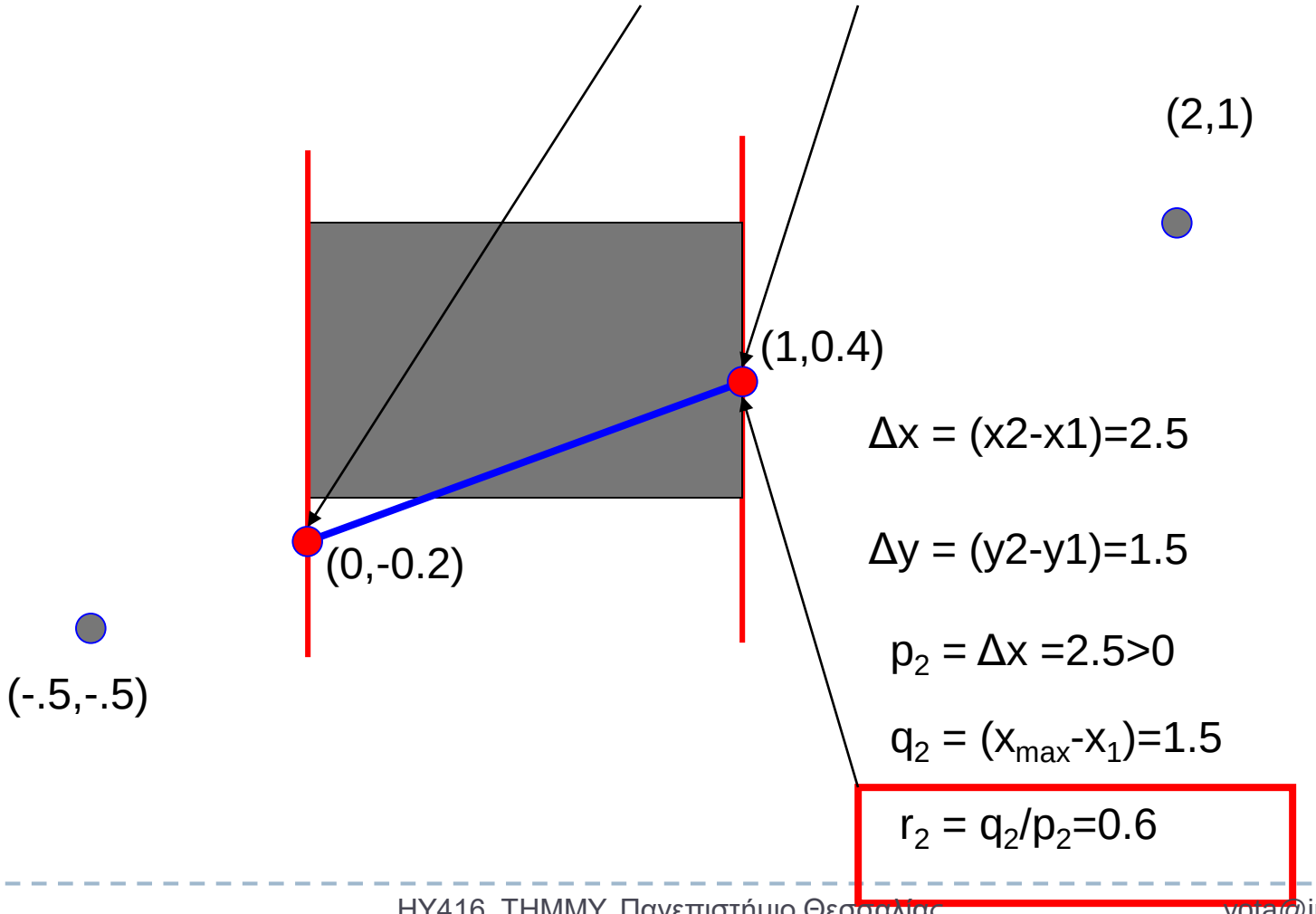
Liang-Barsky Algorithm

- Current clipped line (u_1=0.2,u_2=1)



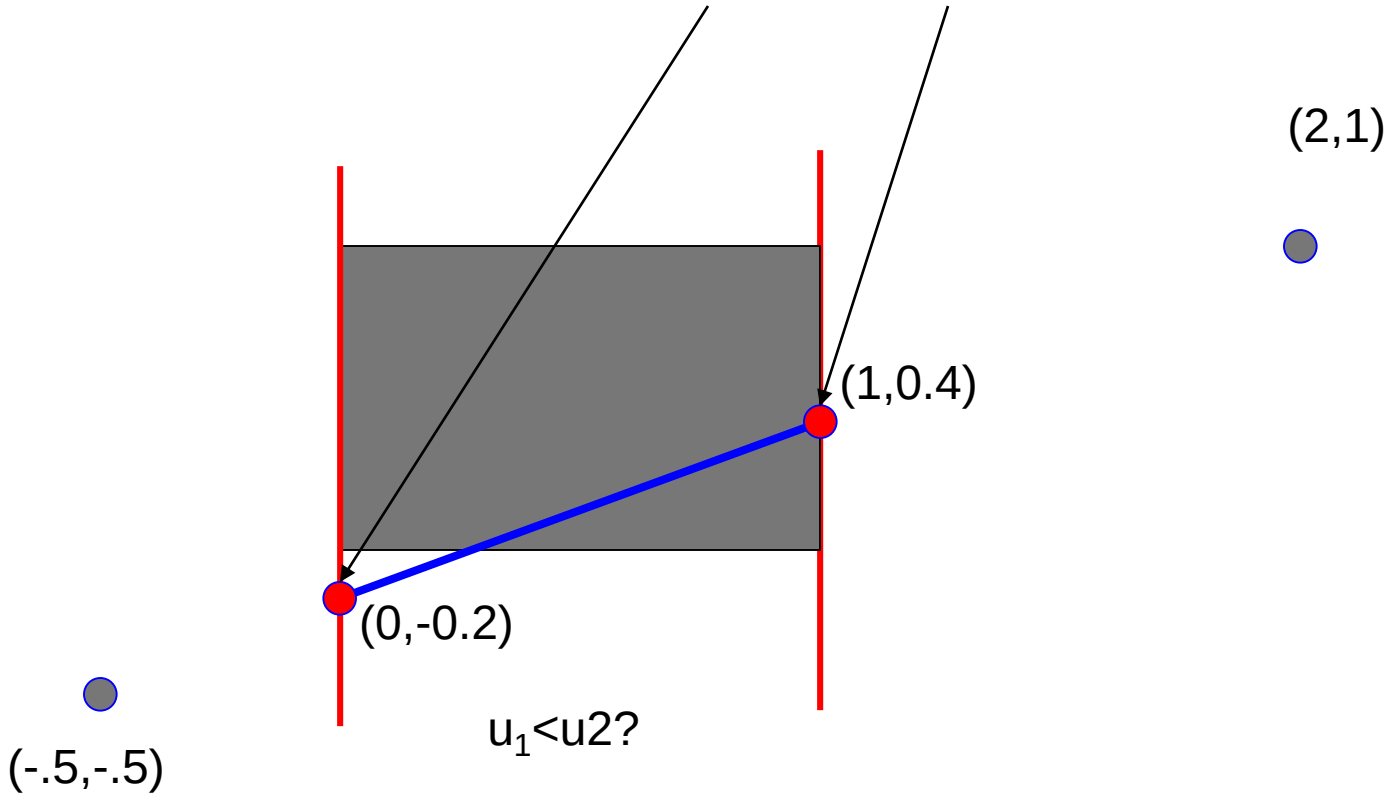
Liang-Barsky Algorithm

- Current clipped line (u_1=0.2,u_2=0.6)



Liang-Barsky Algorithm

- Current clipped line (u_1=0.2,u_2=0.6)



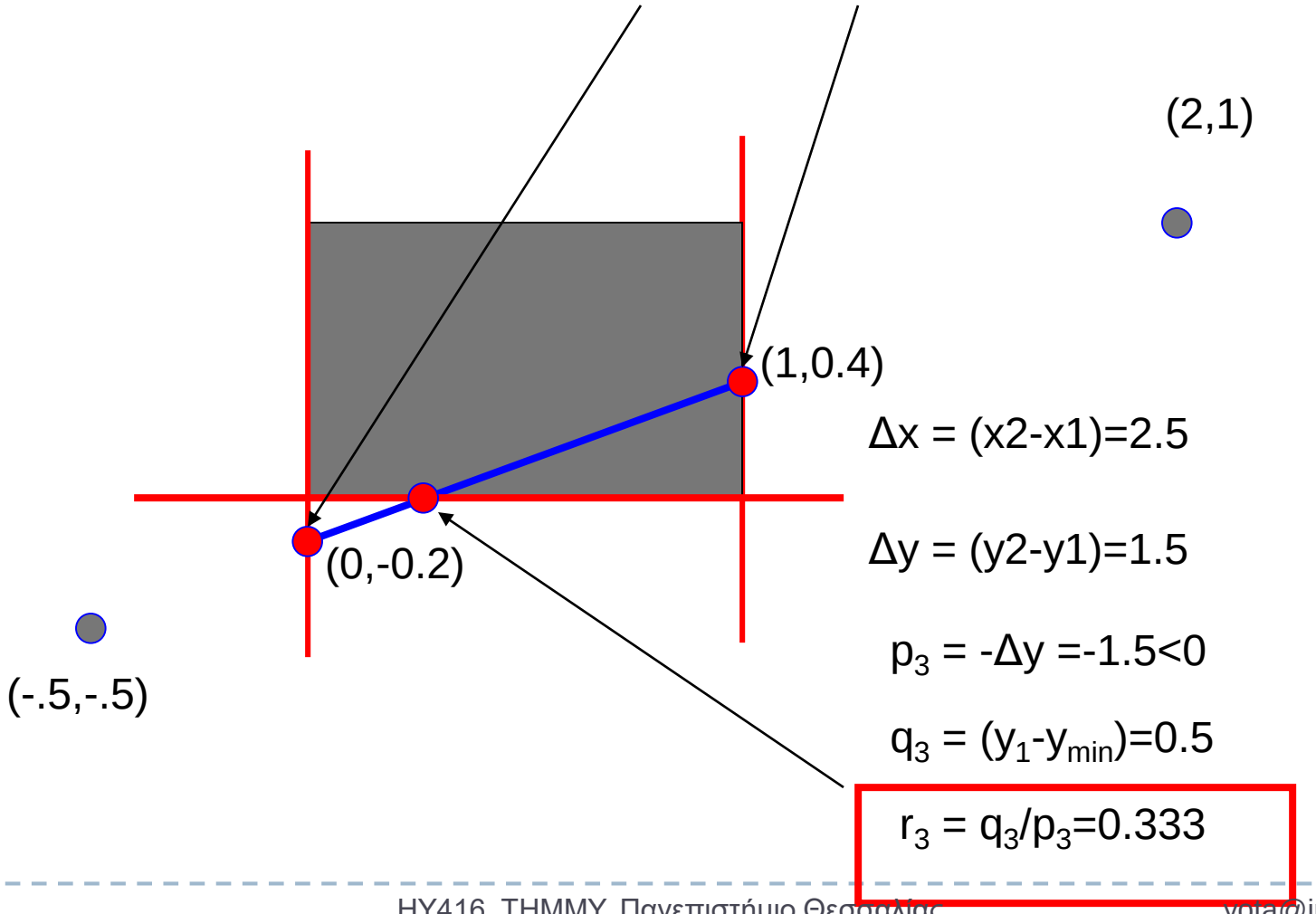
u_1<u2?

- yes: continue

- no: the line is completely outside window

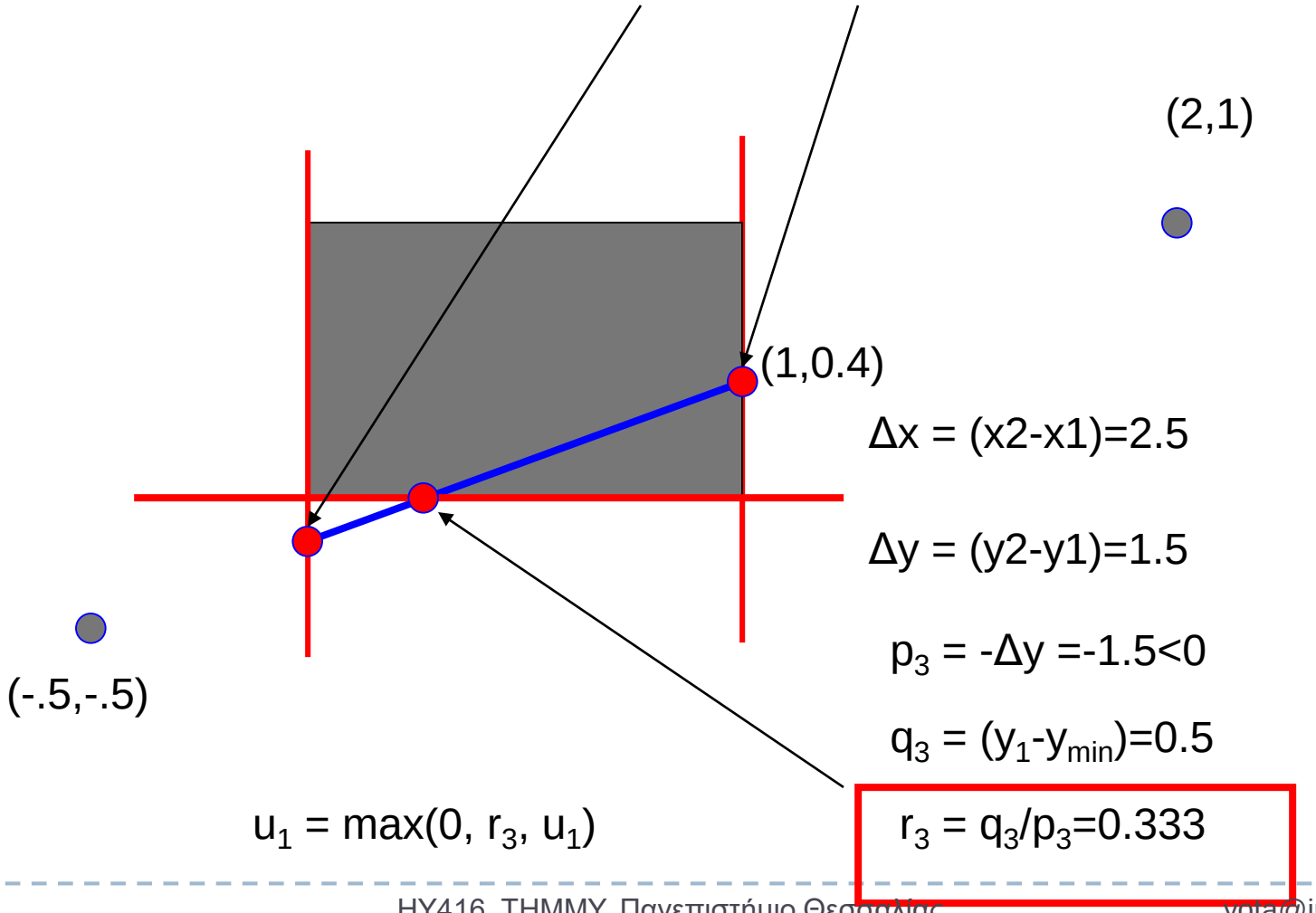
Liang-Barsky Algorithm

- Current clipped line (u_1=0.2,u_2=0.6)



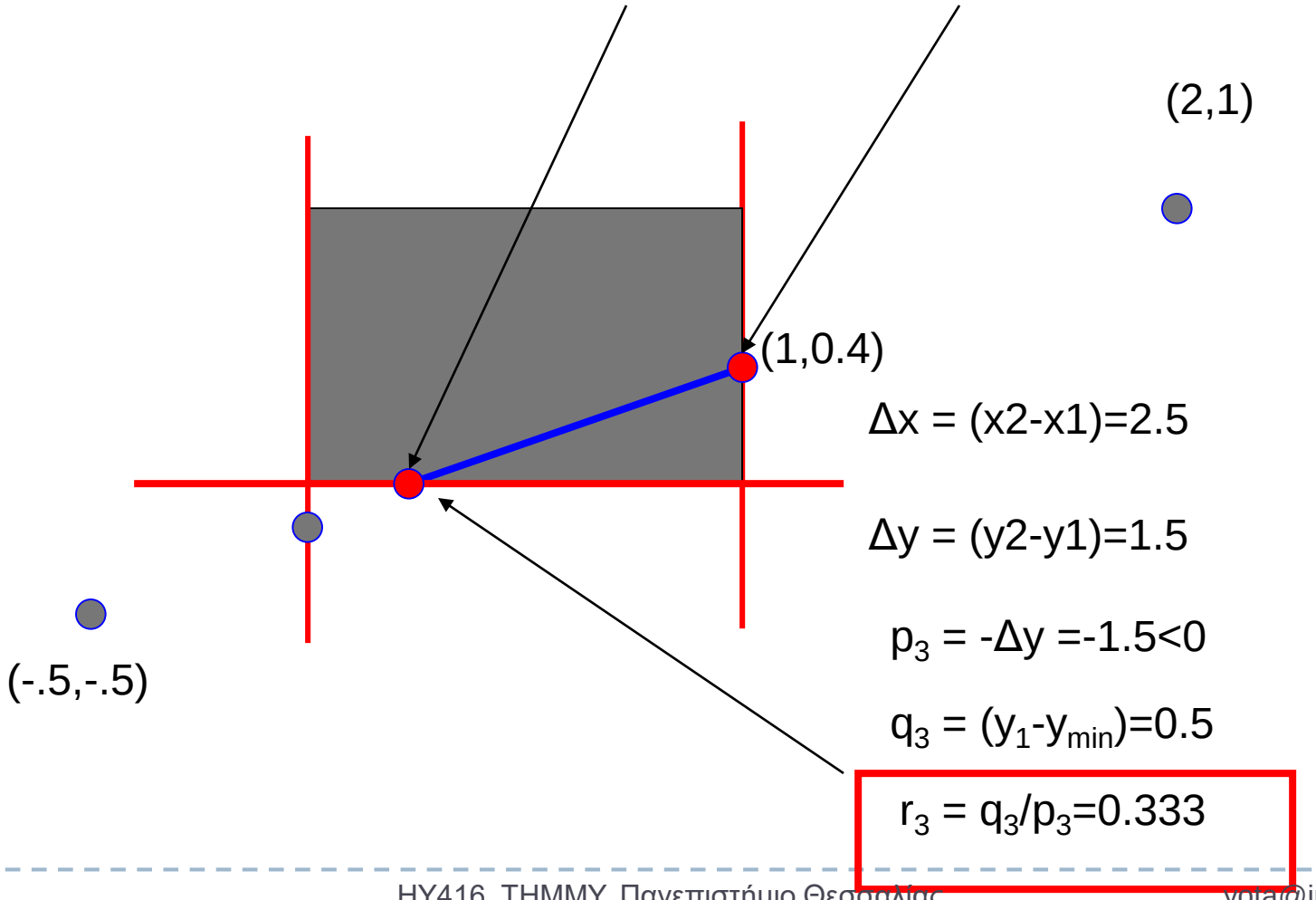
Liang-Barsky Algorithm

- Current clipped line (u_1=0.2,u_2=0.6)



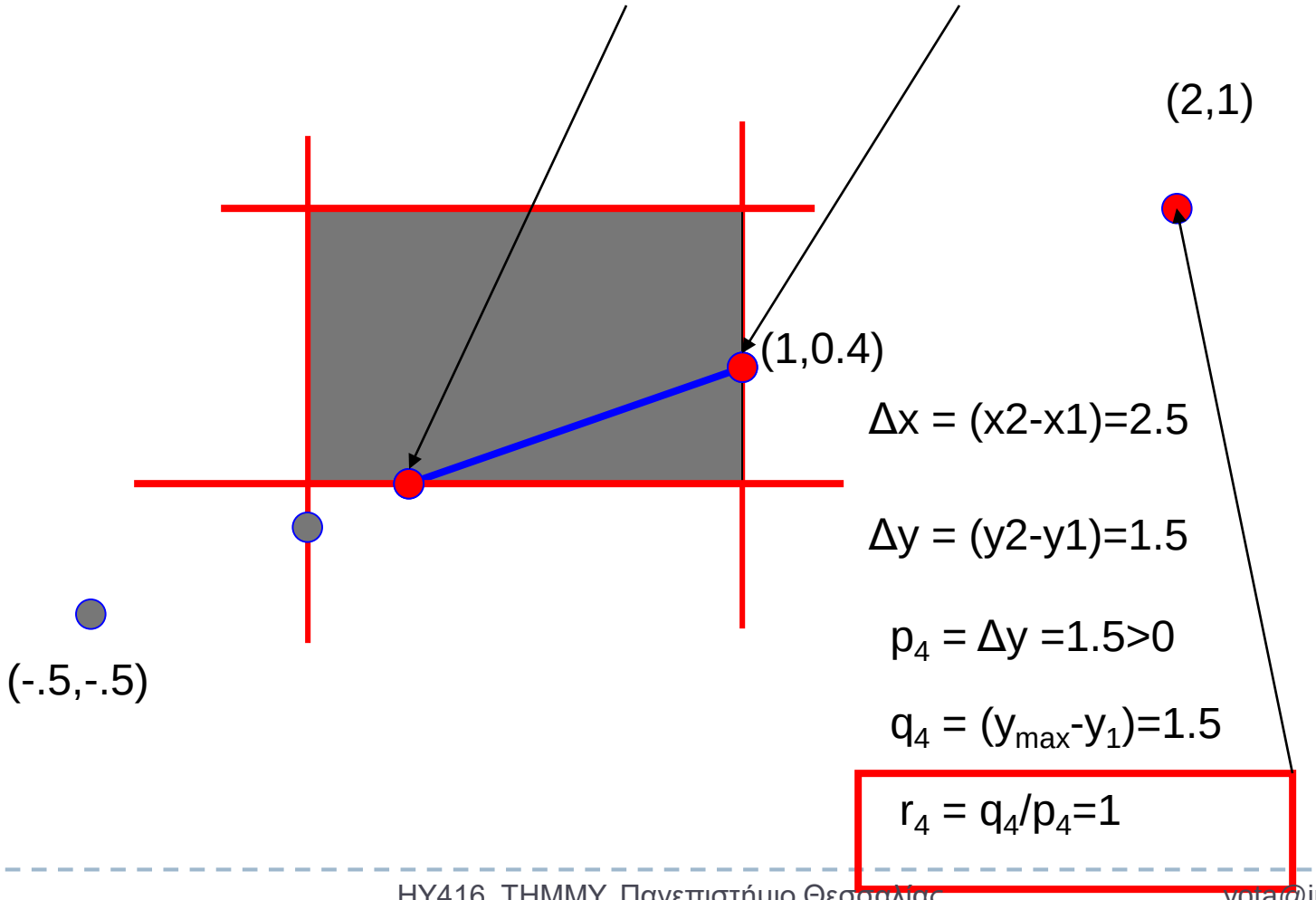
Liang-Barsky Algorithm

- Current clipped line (u_1=0.333,u_2=0.6)



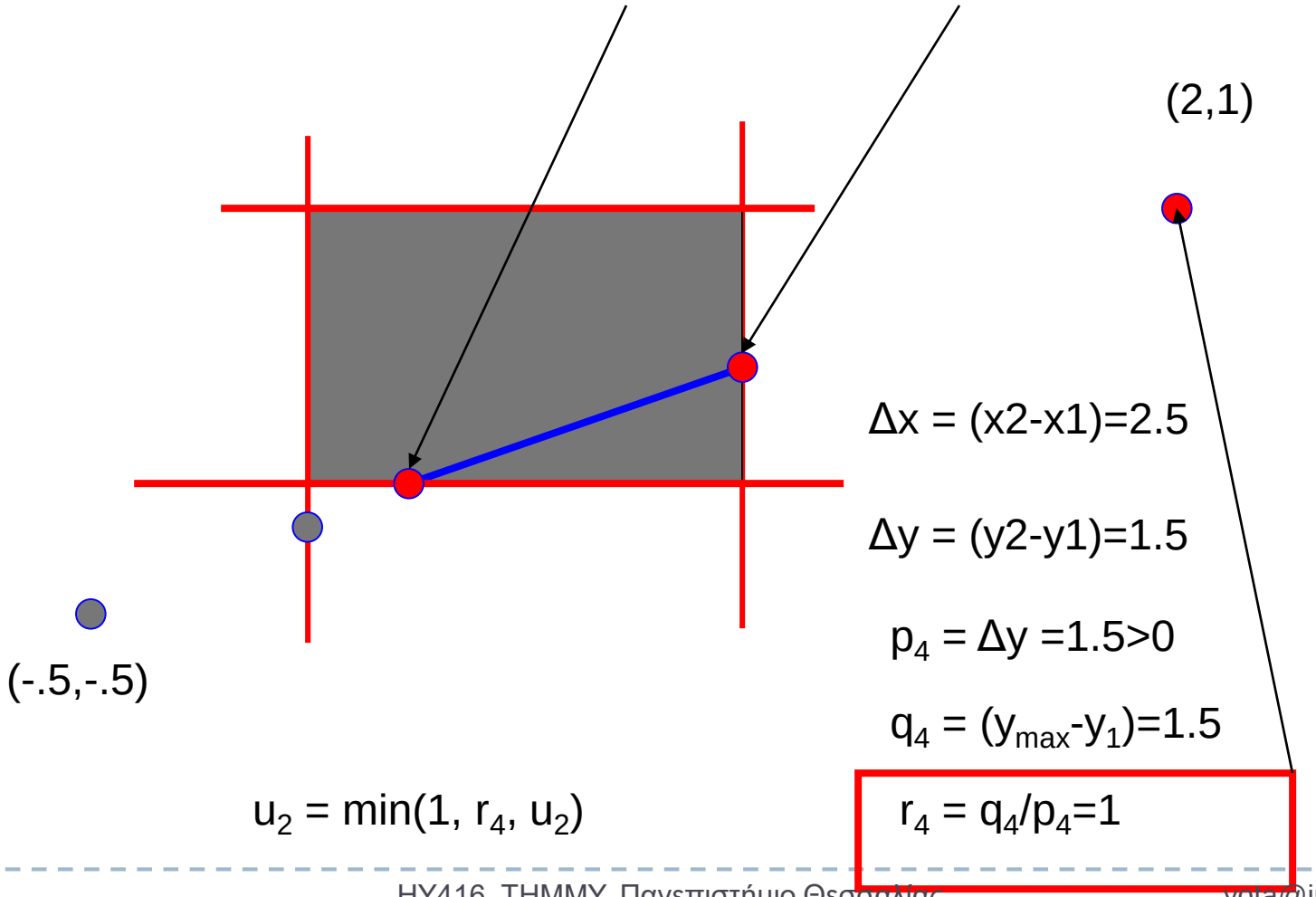
Liang-Barsky Algorithm

- Current clipped line (u_1=0.333,u_2=0.6)



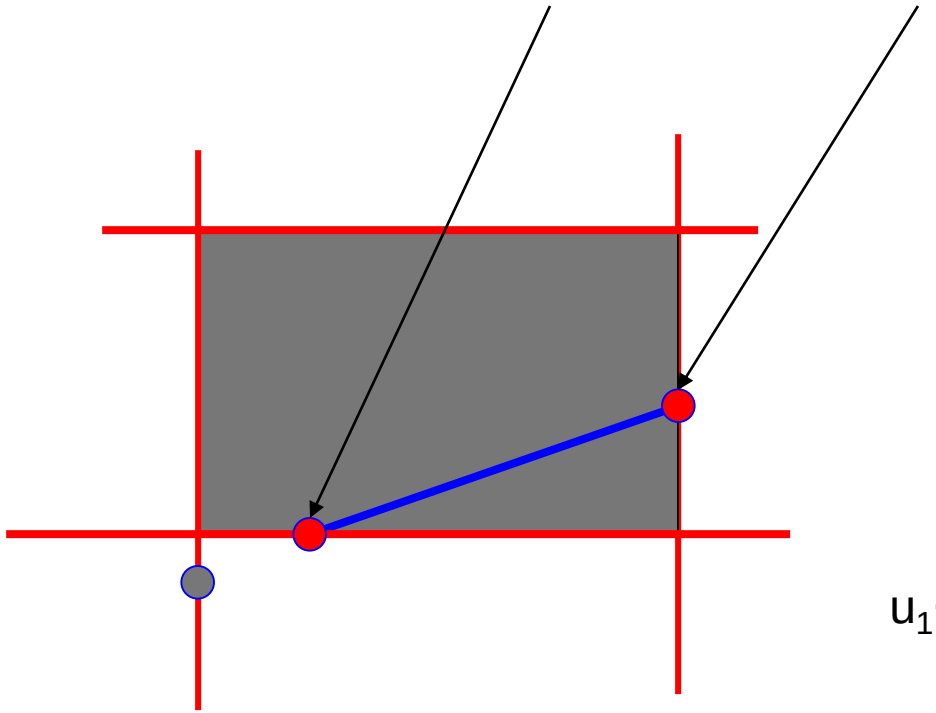
Liang-Barsky Algorithm

- Current clipped line (u_1=0.333,u_2=0.6)



Liang-Barsky Algorithm

- Check the left edge (u_1=0.333,u_2=0.6)



(2,1)

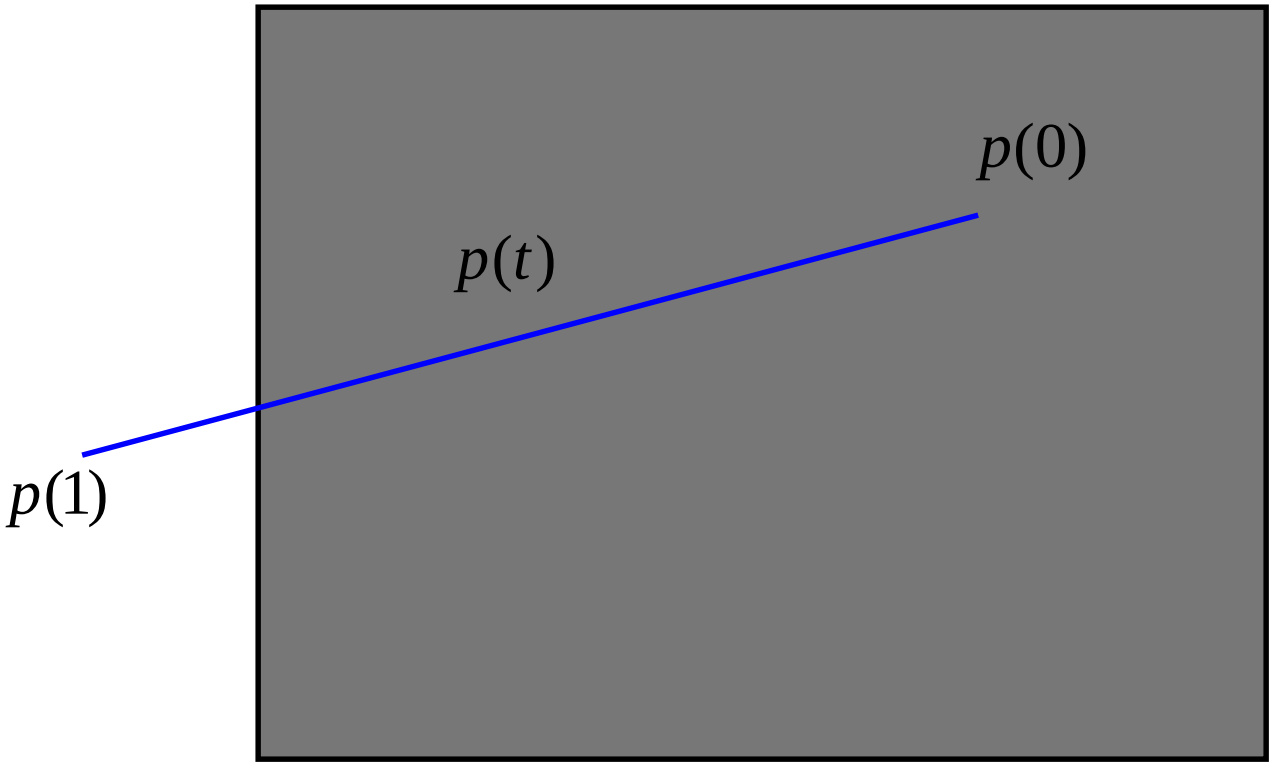
$u_1<u_2$

(-.5,-.5)

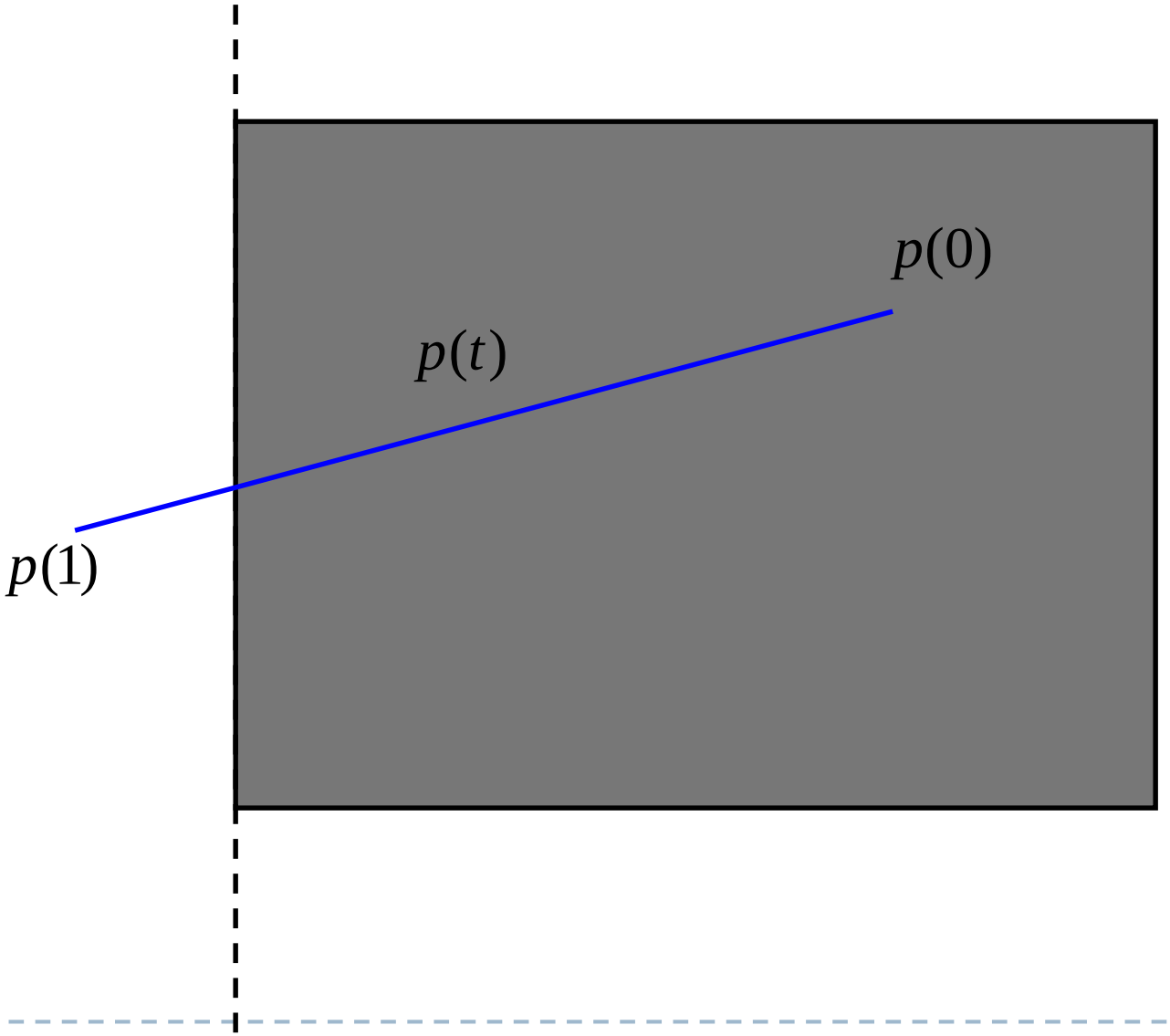
$x_{new1}=x_1+\Delta x*u_1$ $x_{new2}=x_1+\Delta x*u_2$

$y_{new1}=y_1+\Delta y*u_1$ $y_{new2}=y_1+\Delta y*u_2$

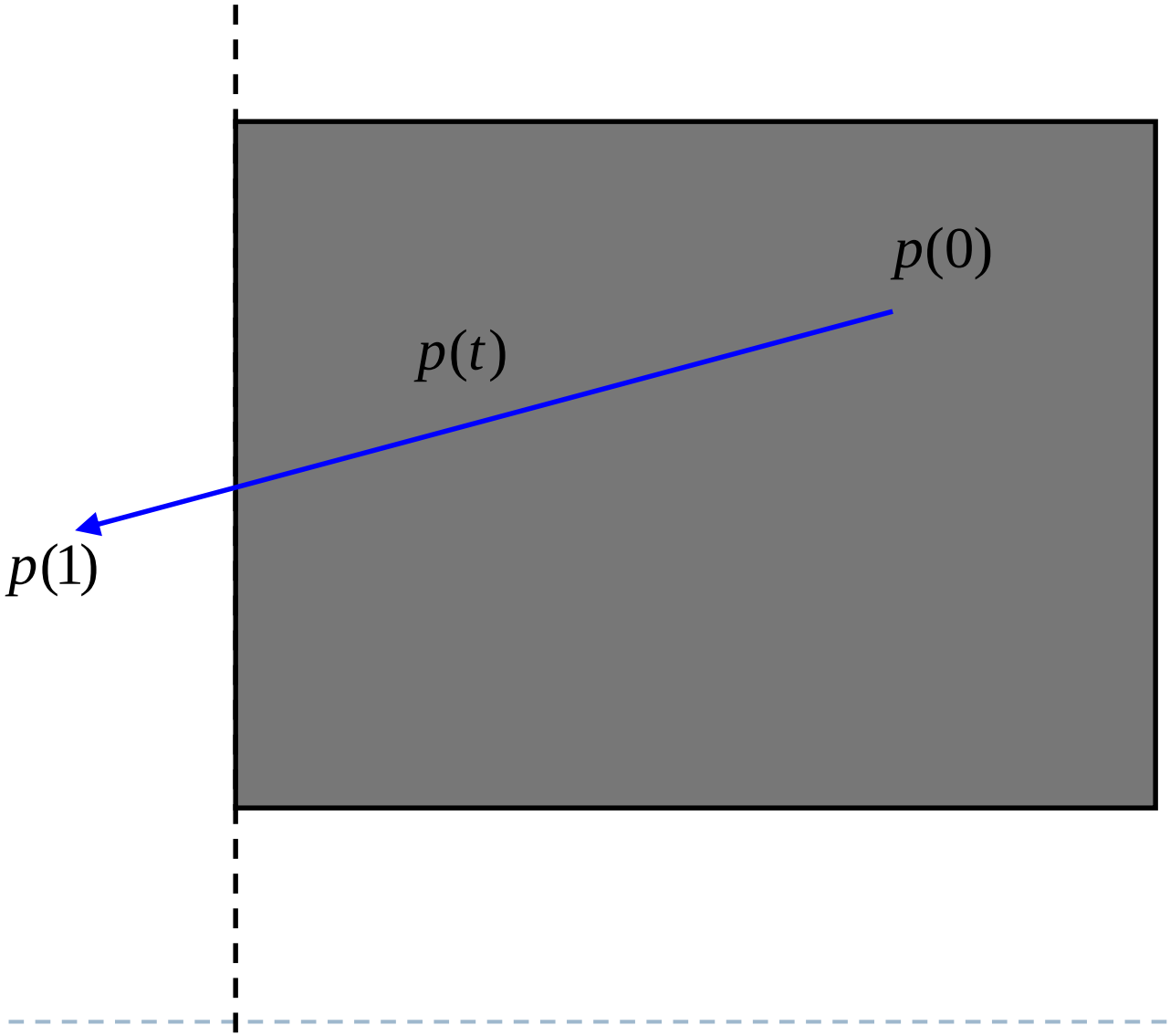
Liang-Barsky Algorithm



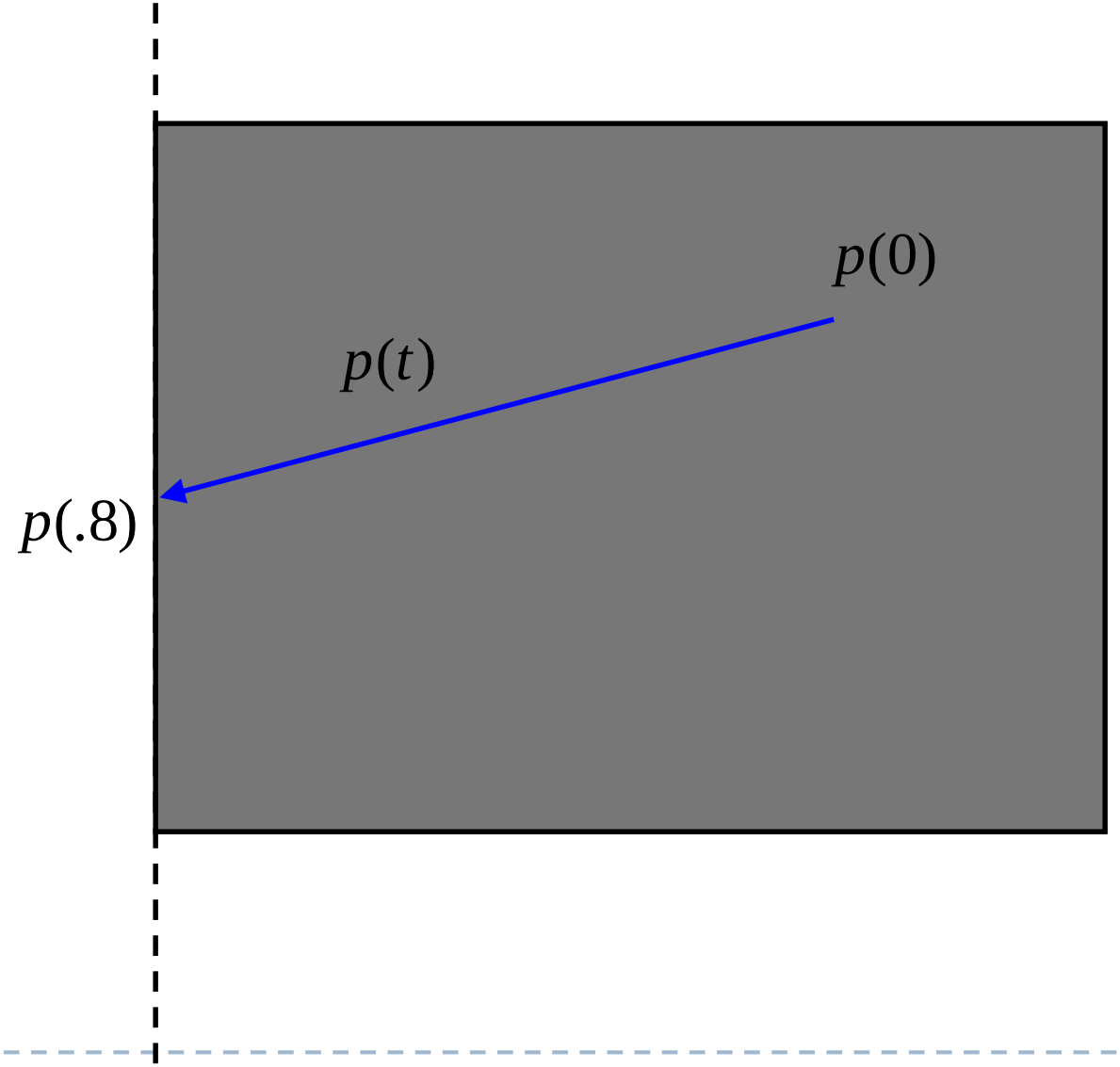
Liang-Barsky Algorithm



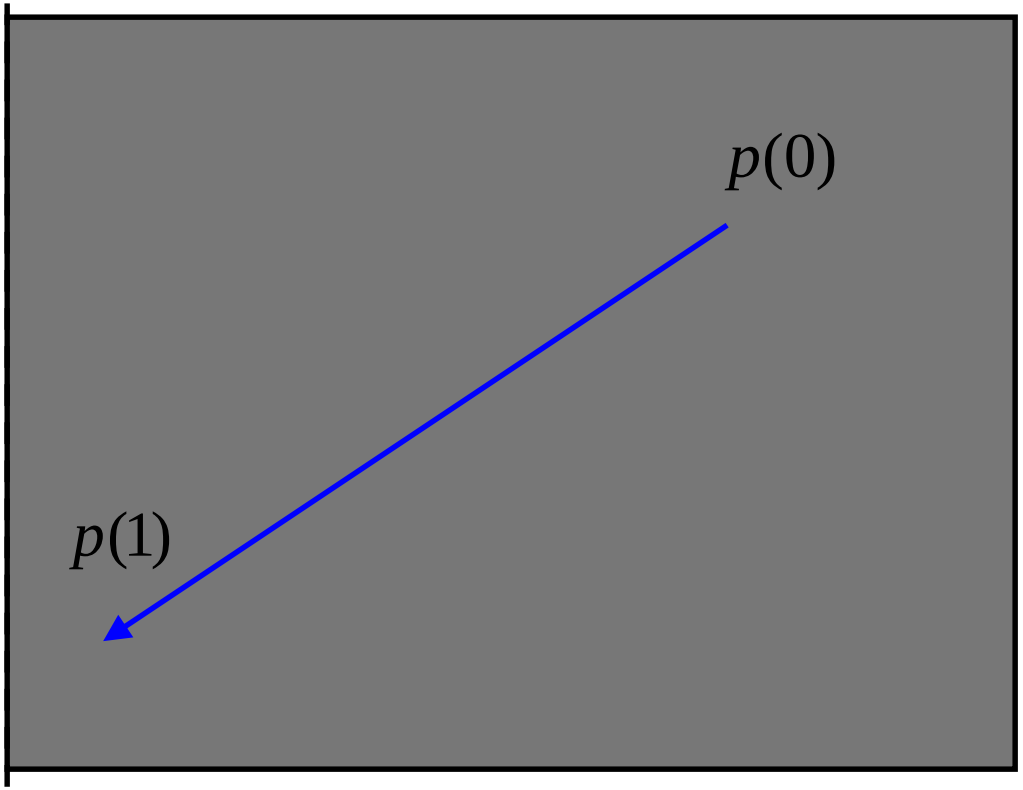
Liang-Barsky Algorithm



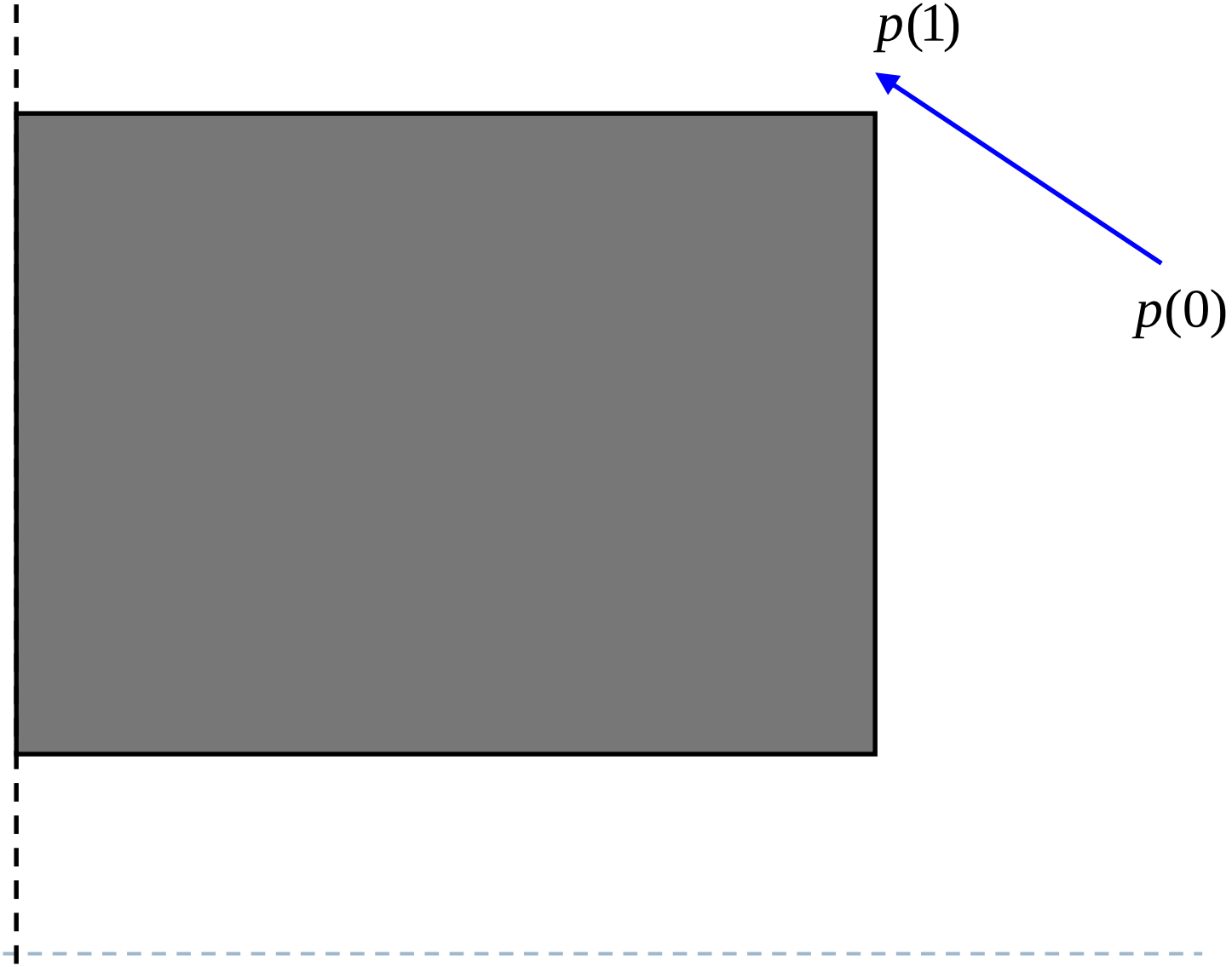
Liang-Barsky Algorithm



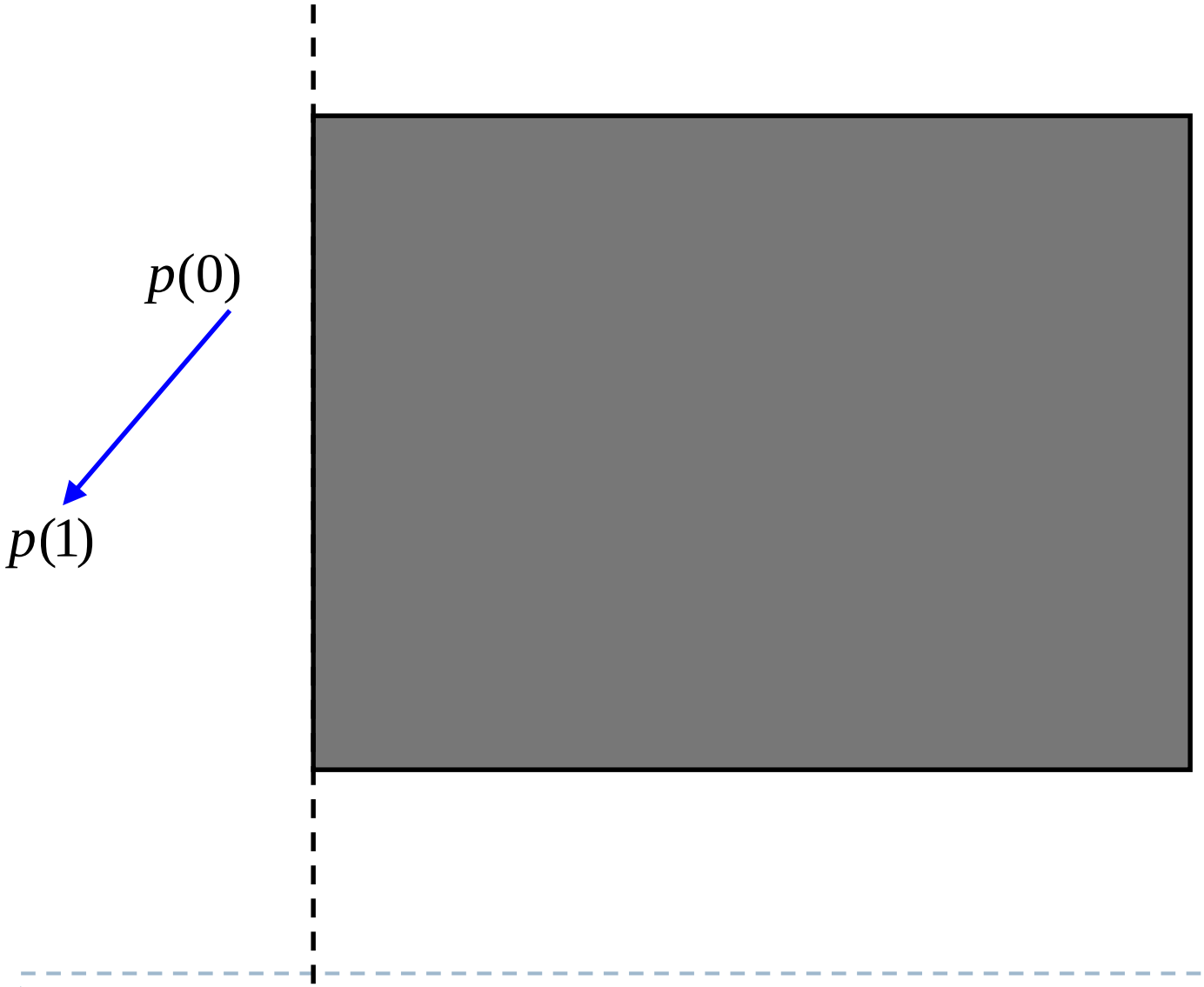
Liang-Barsky Algorithm



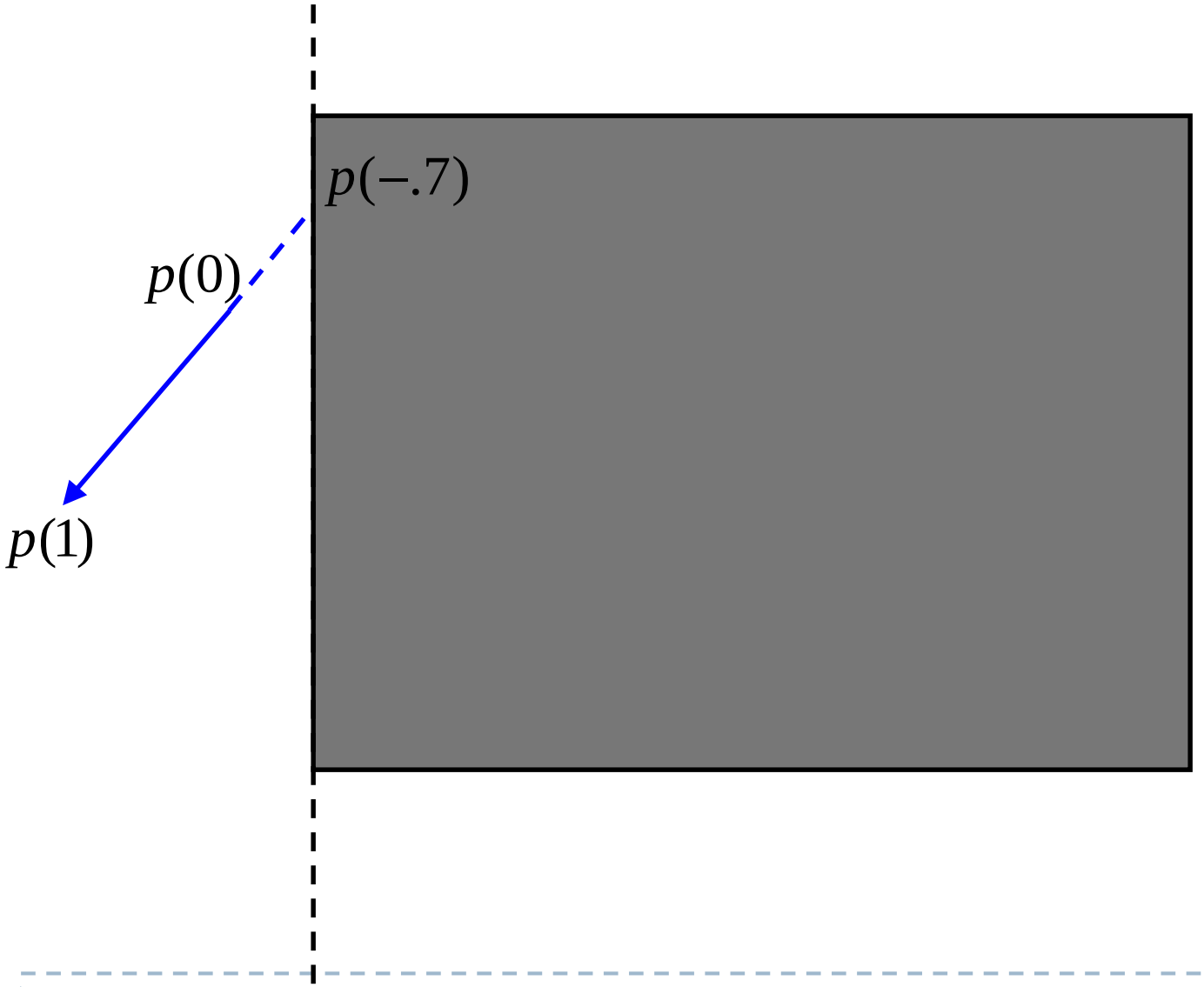
Liang-Barsky Algorithm



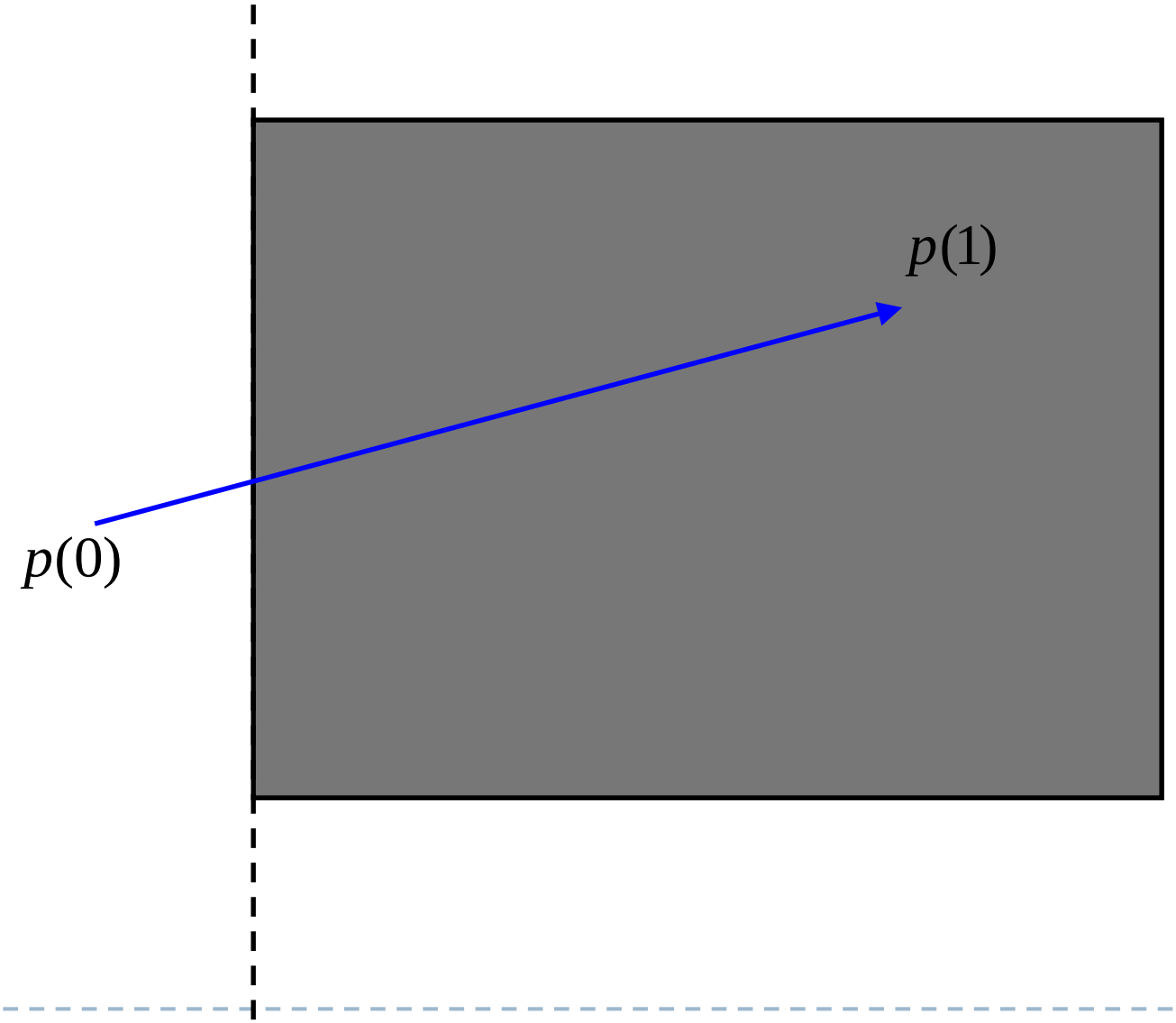
Liang-Barsky Algorithm



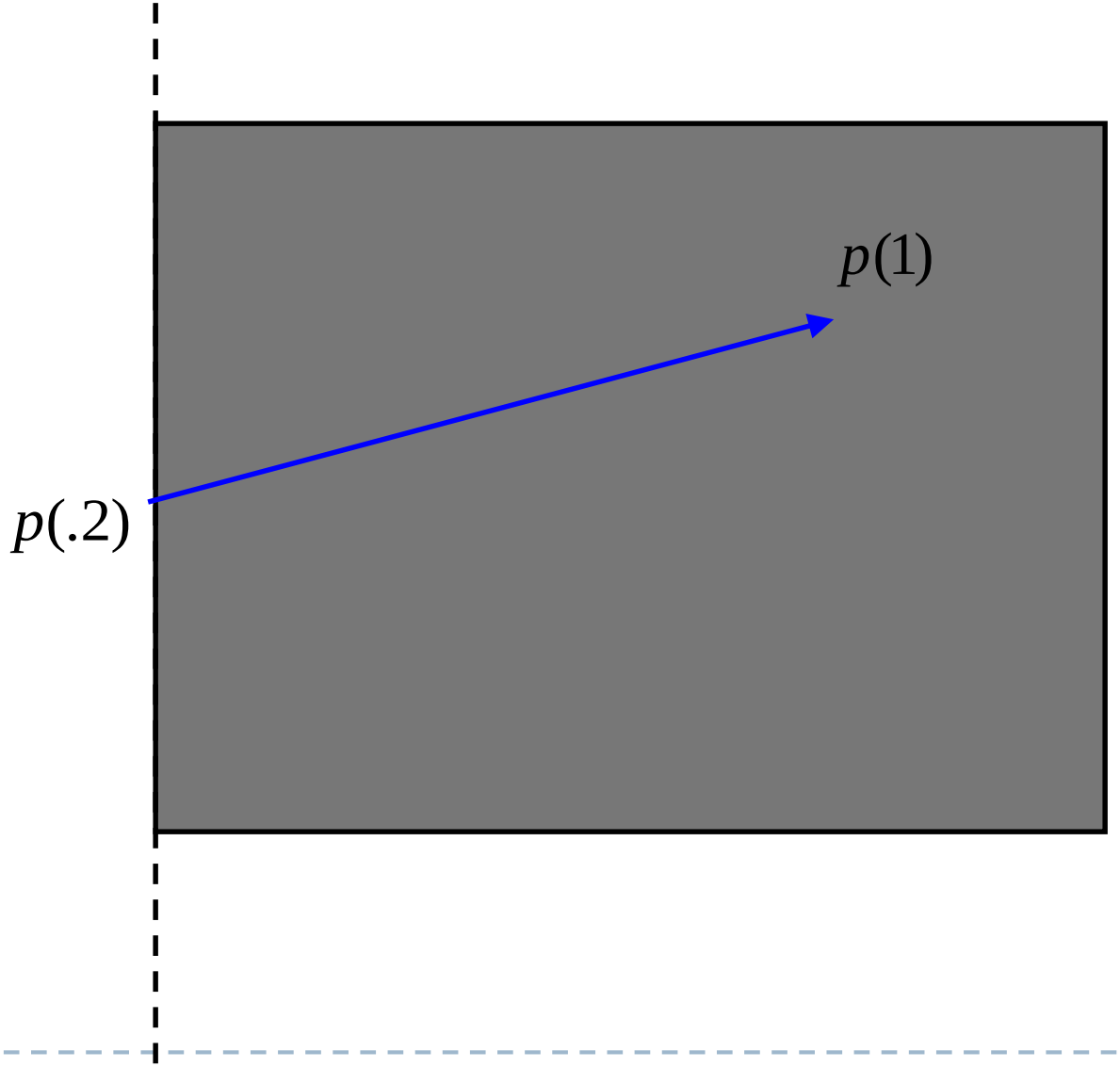
Liang-Barsky Algorithm



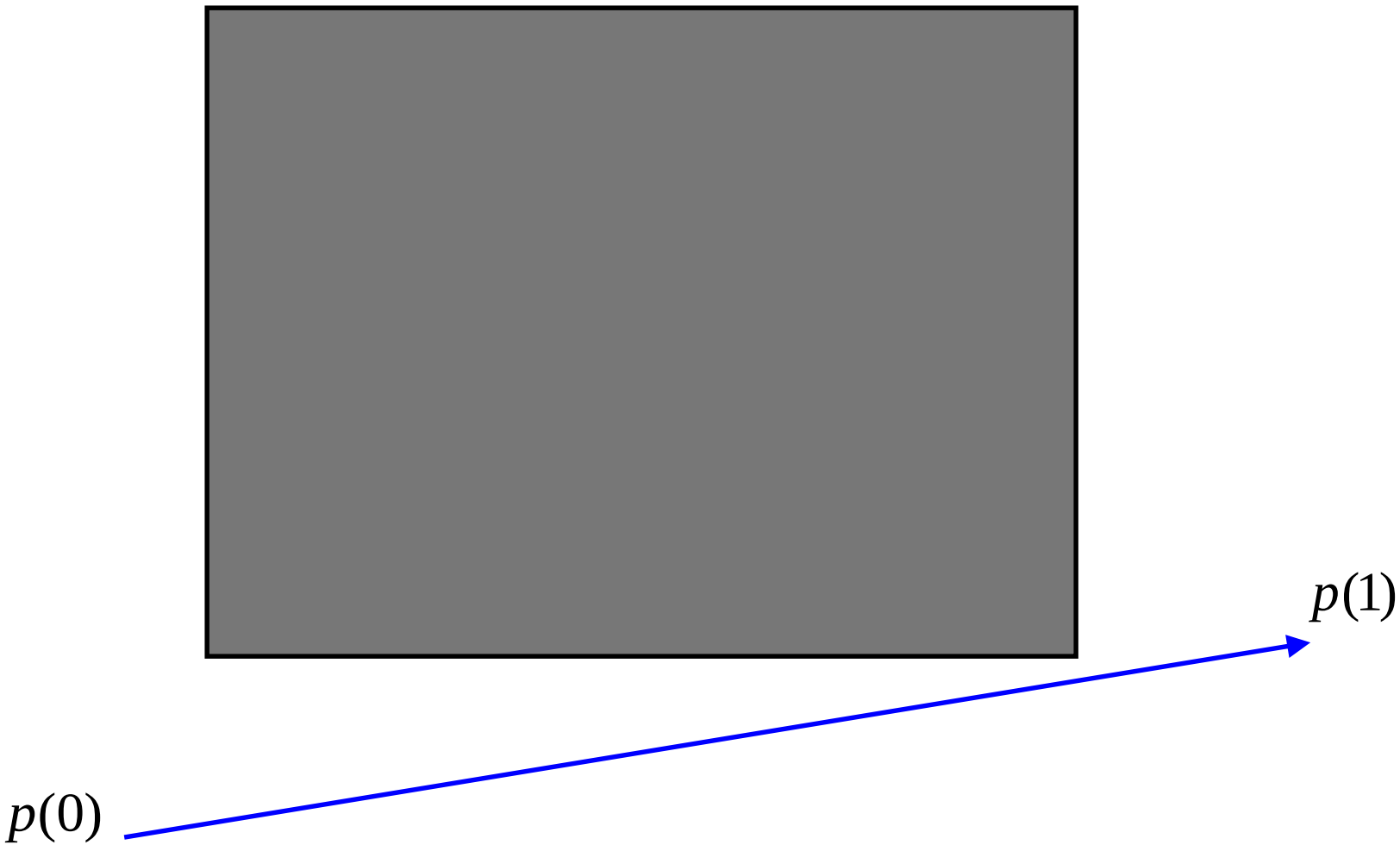
Liang-Barsky Algorithm



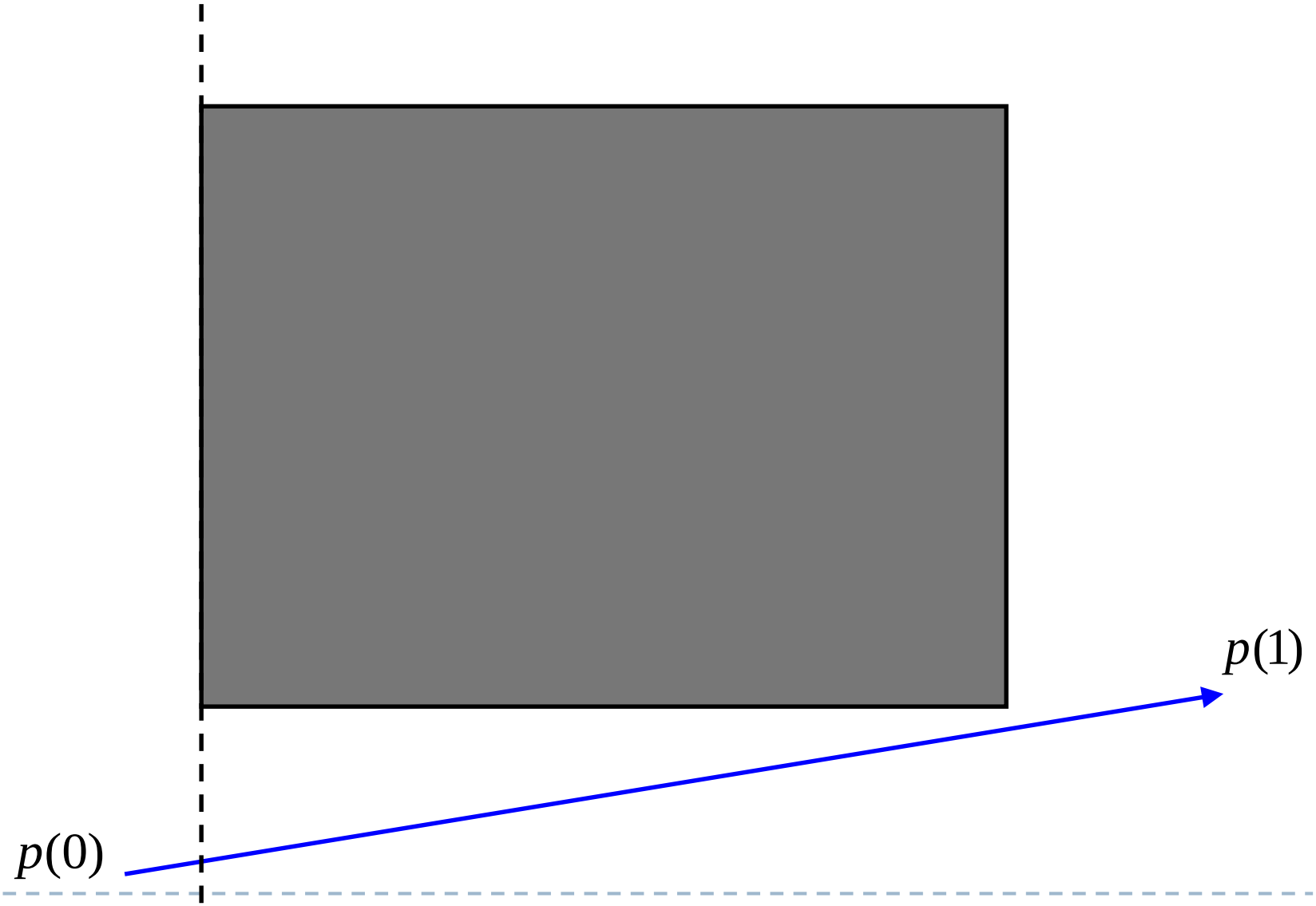
Liang-Barsky Algorithm



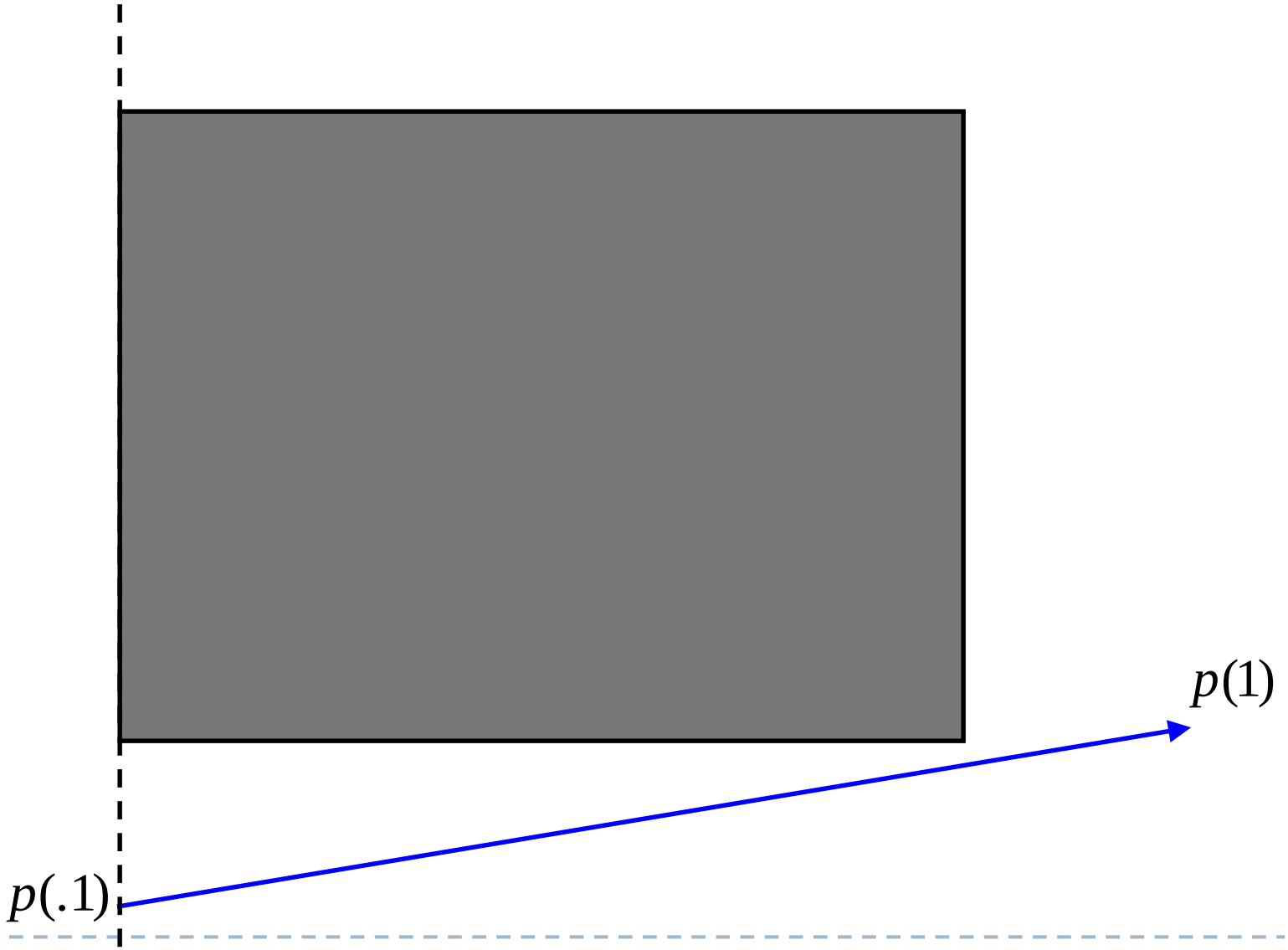
Liang-Barsky Algorithm



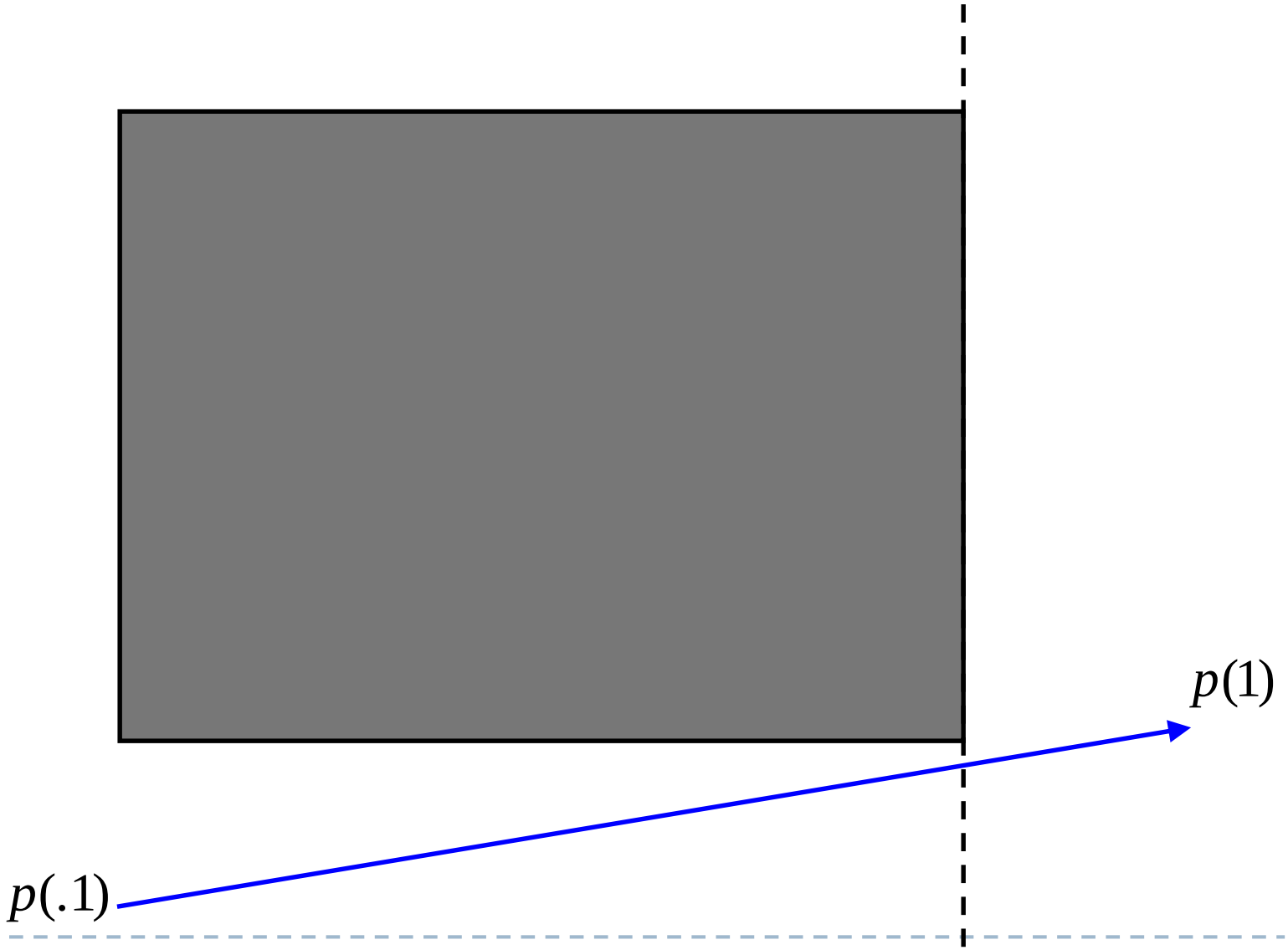
Liang-Barsky Algorithm



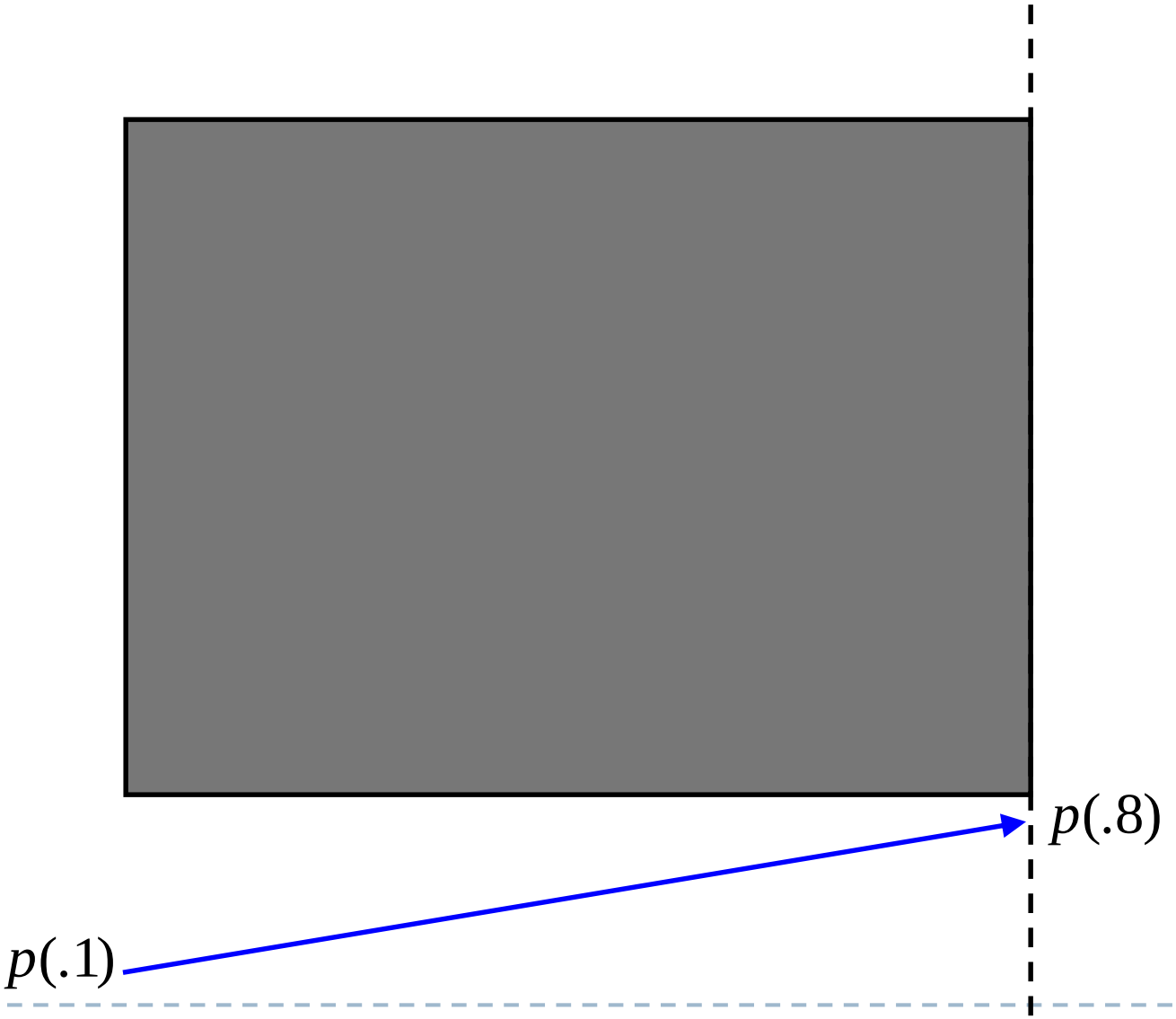
Liang-Barsky Algorithm



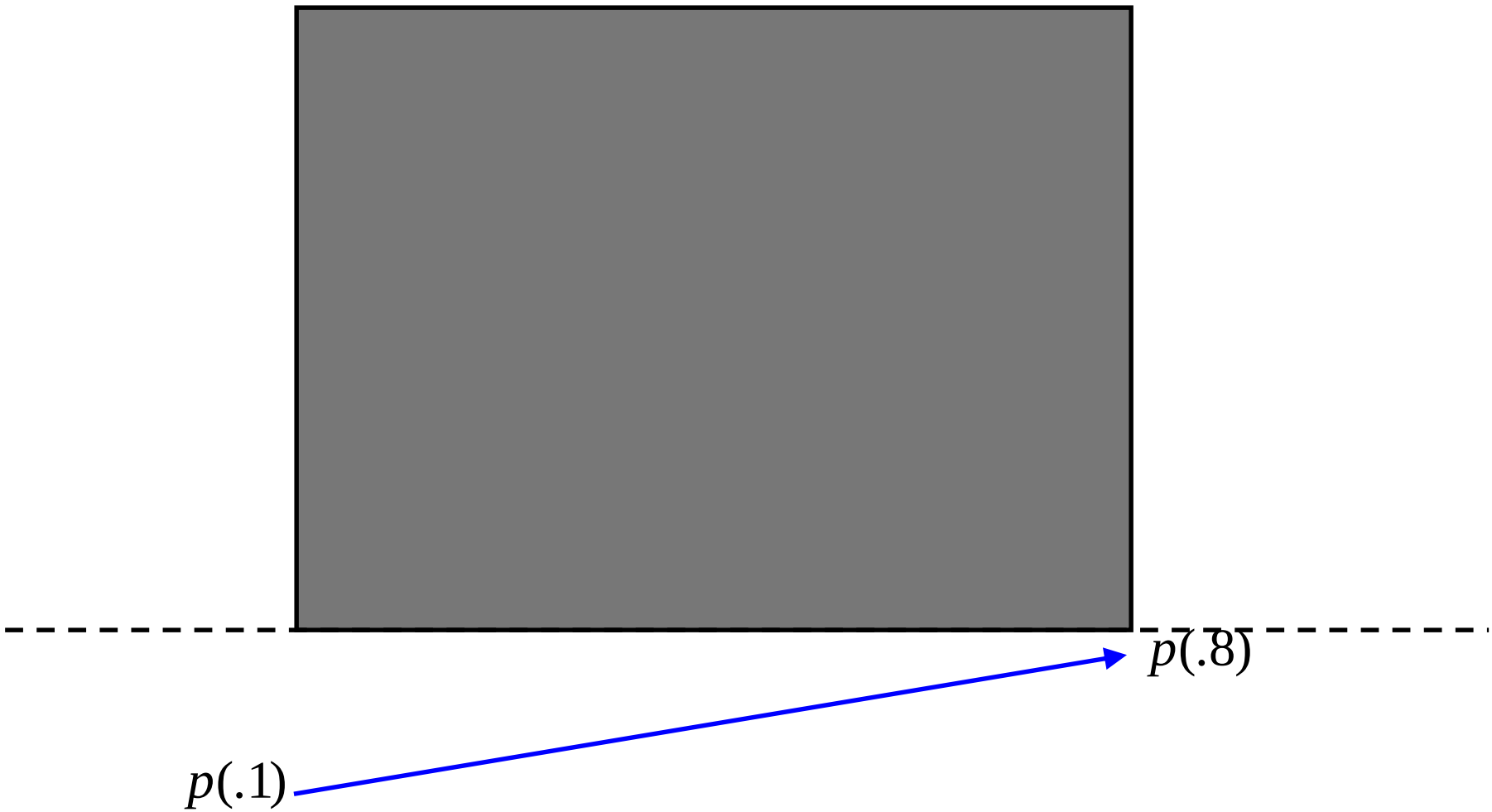
Liang-Barsky Algorithm



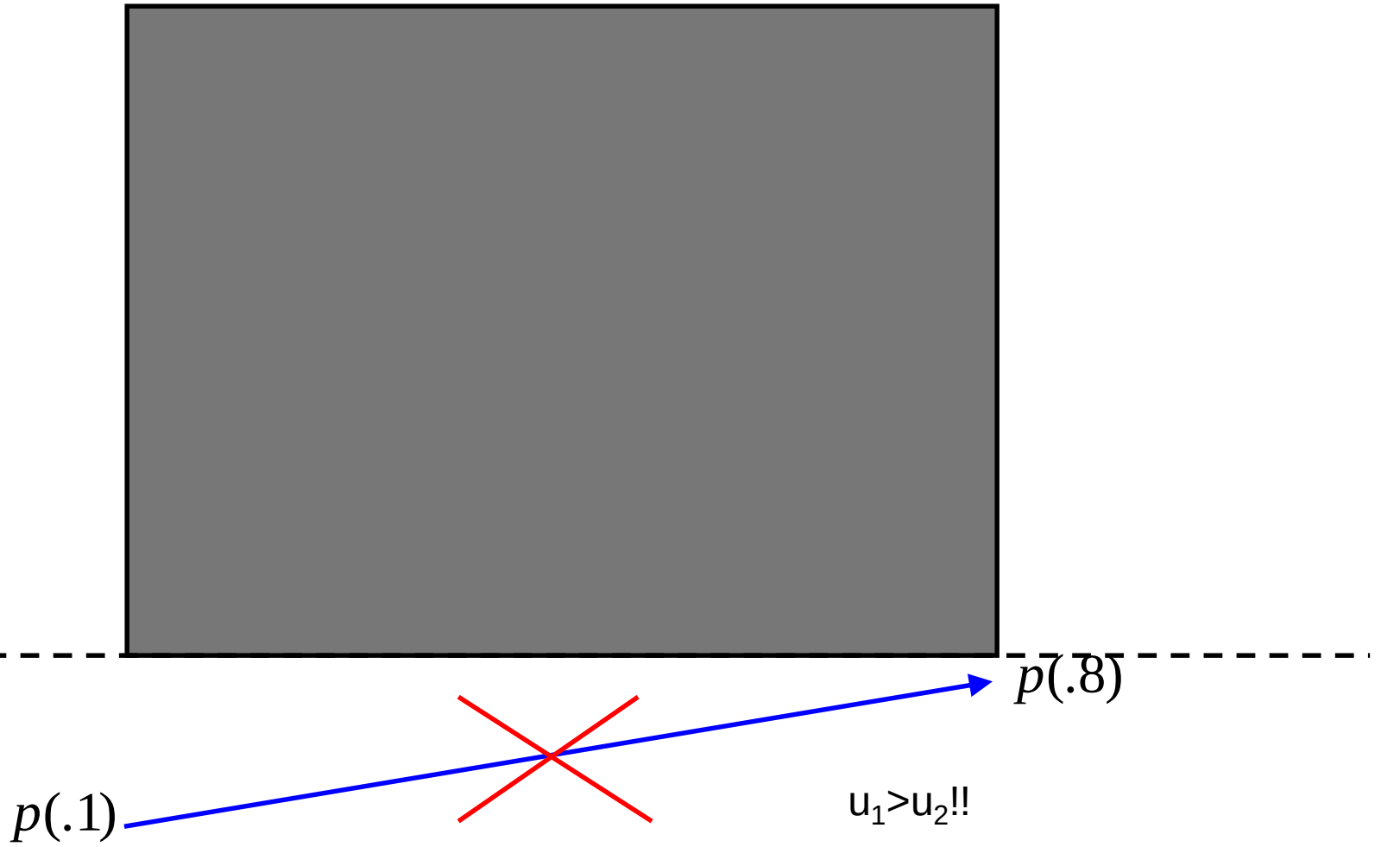
Liang-Barsky Algorithm



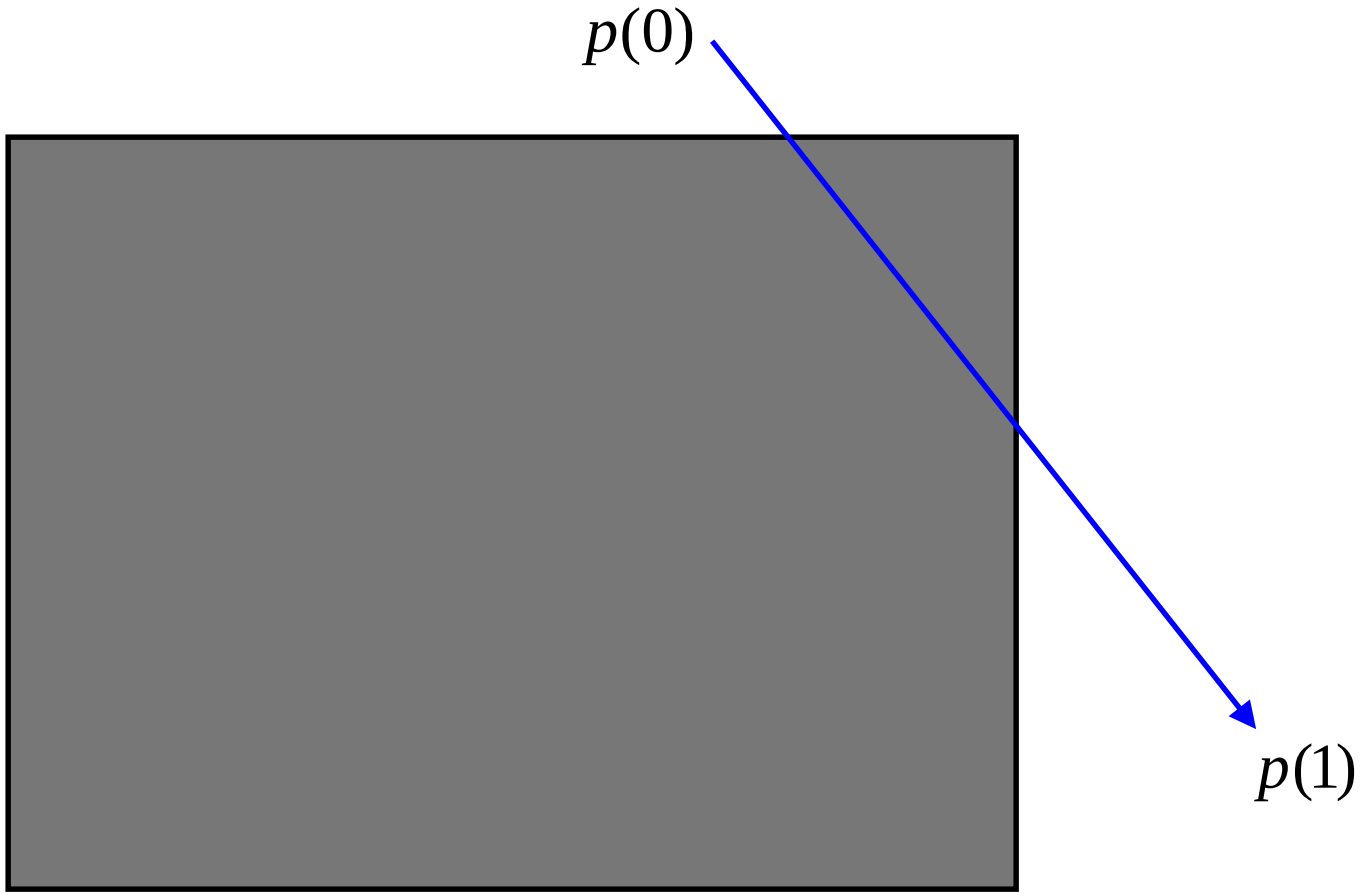
Liang-Barsky Algorithm



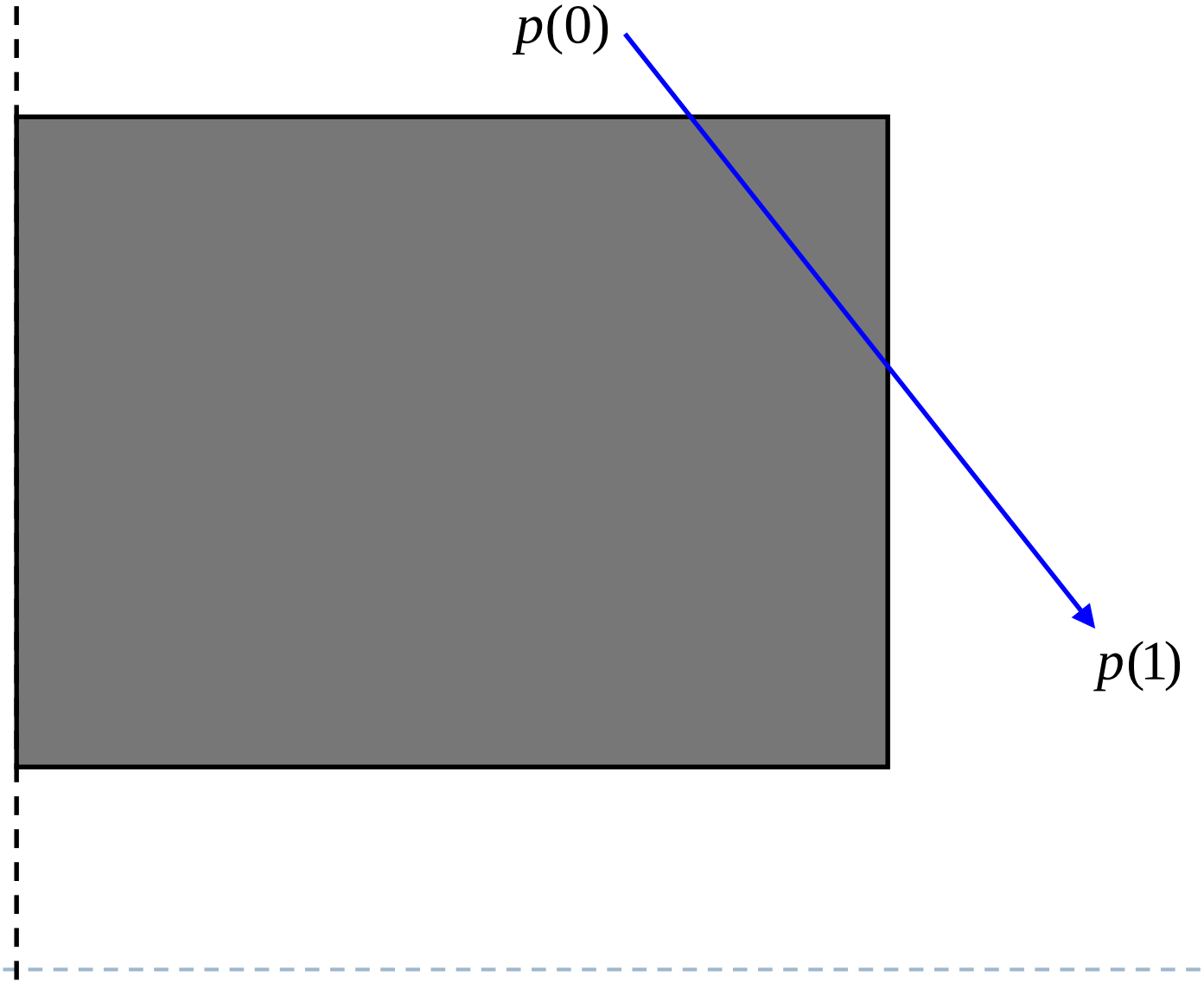
Liang-Barsky Algorithm



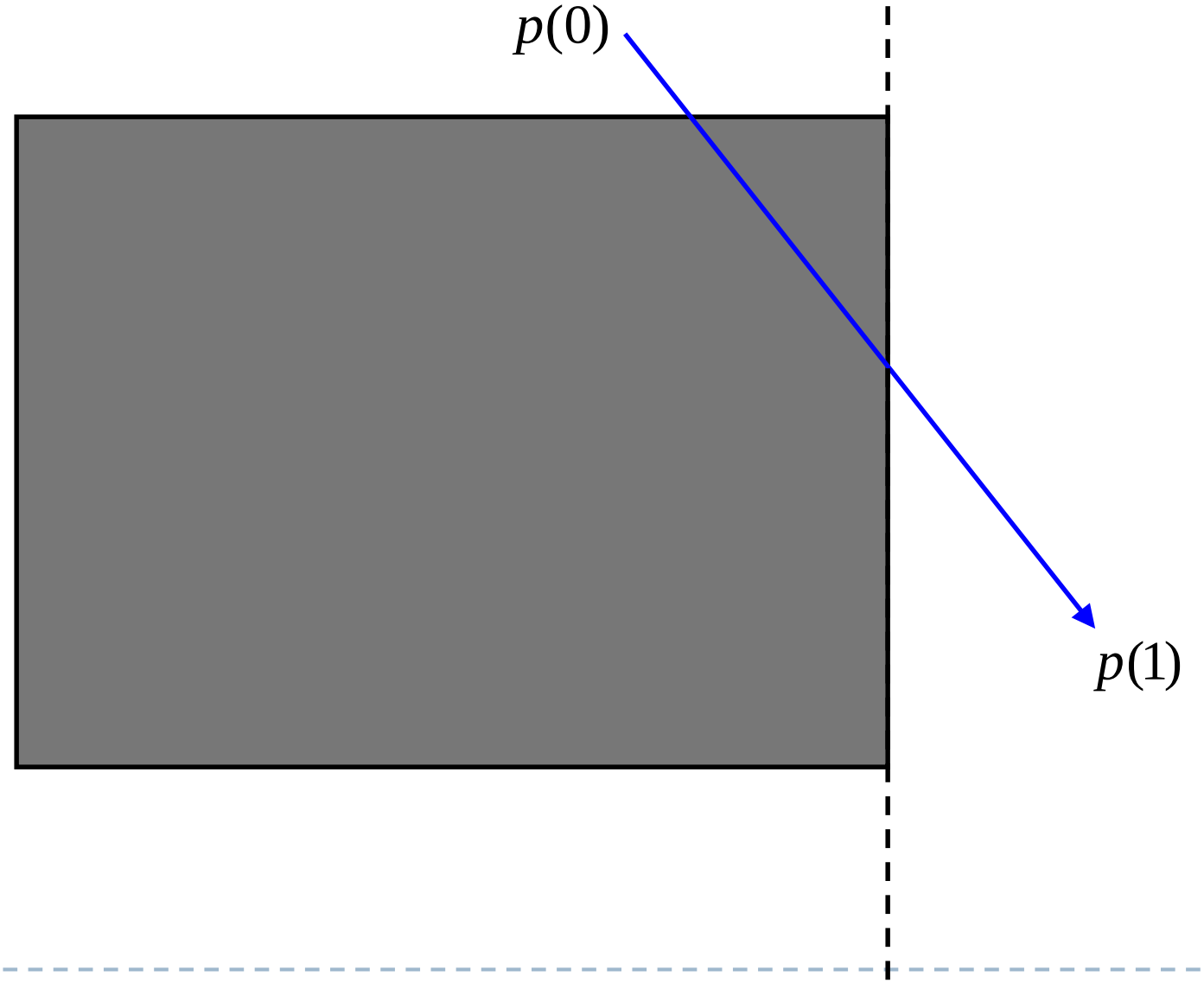
Liang-Barsky Algorithm



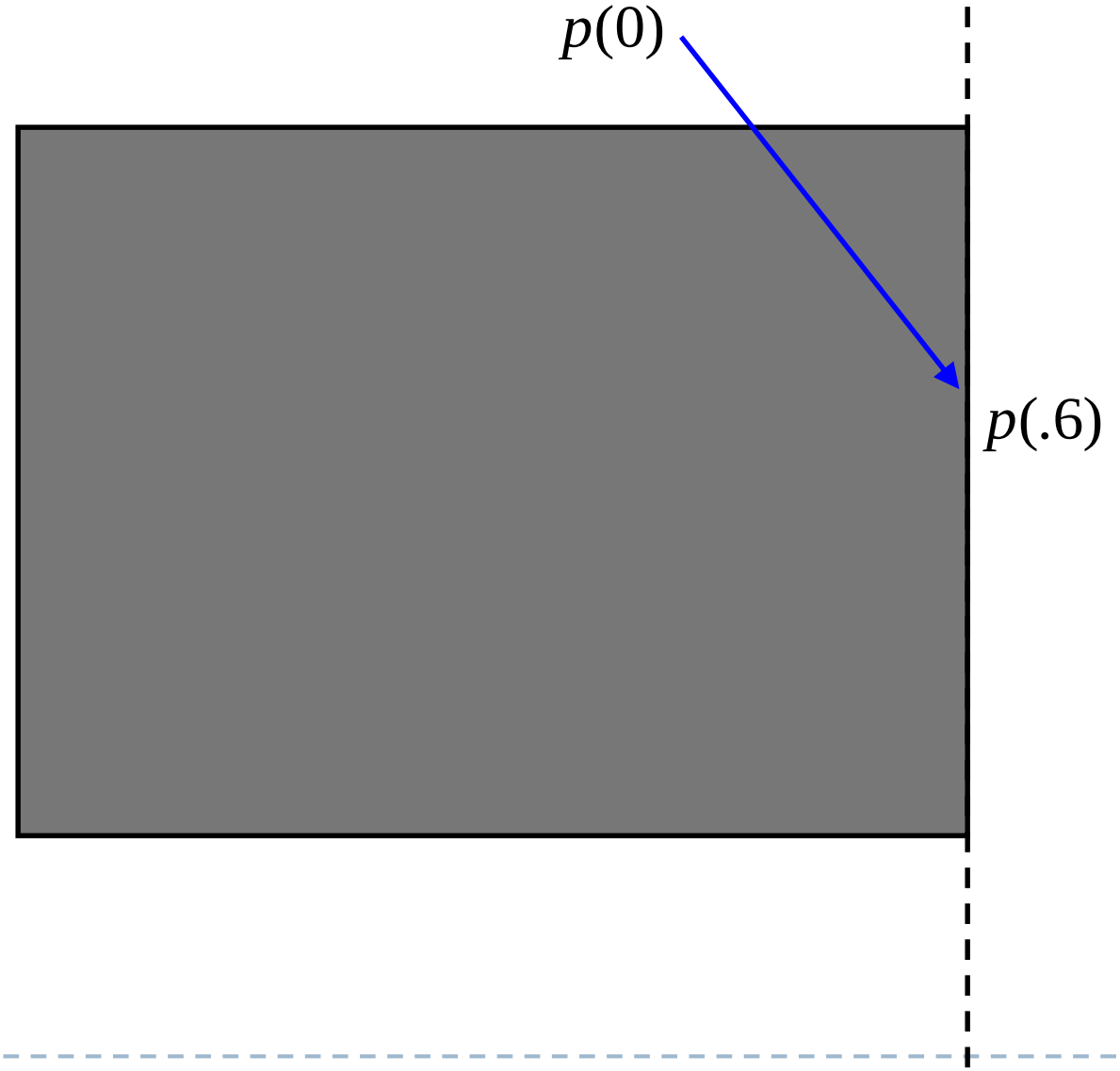
Liang-Barsky Algorithm



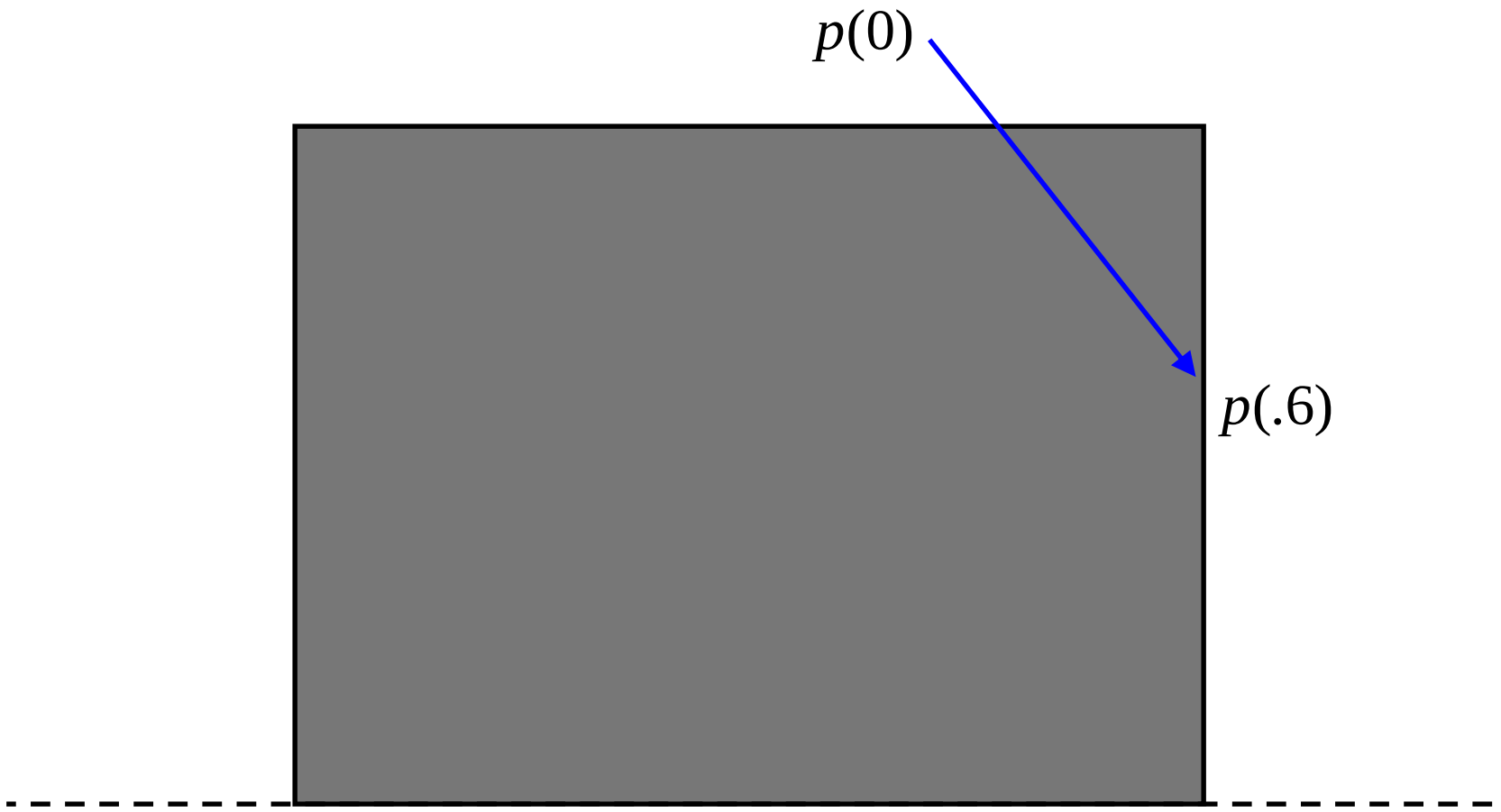
Liang-Barsky Algorithm



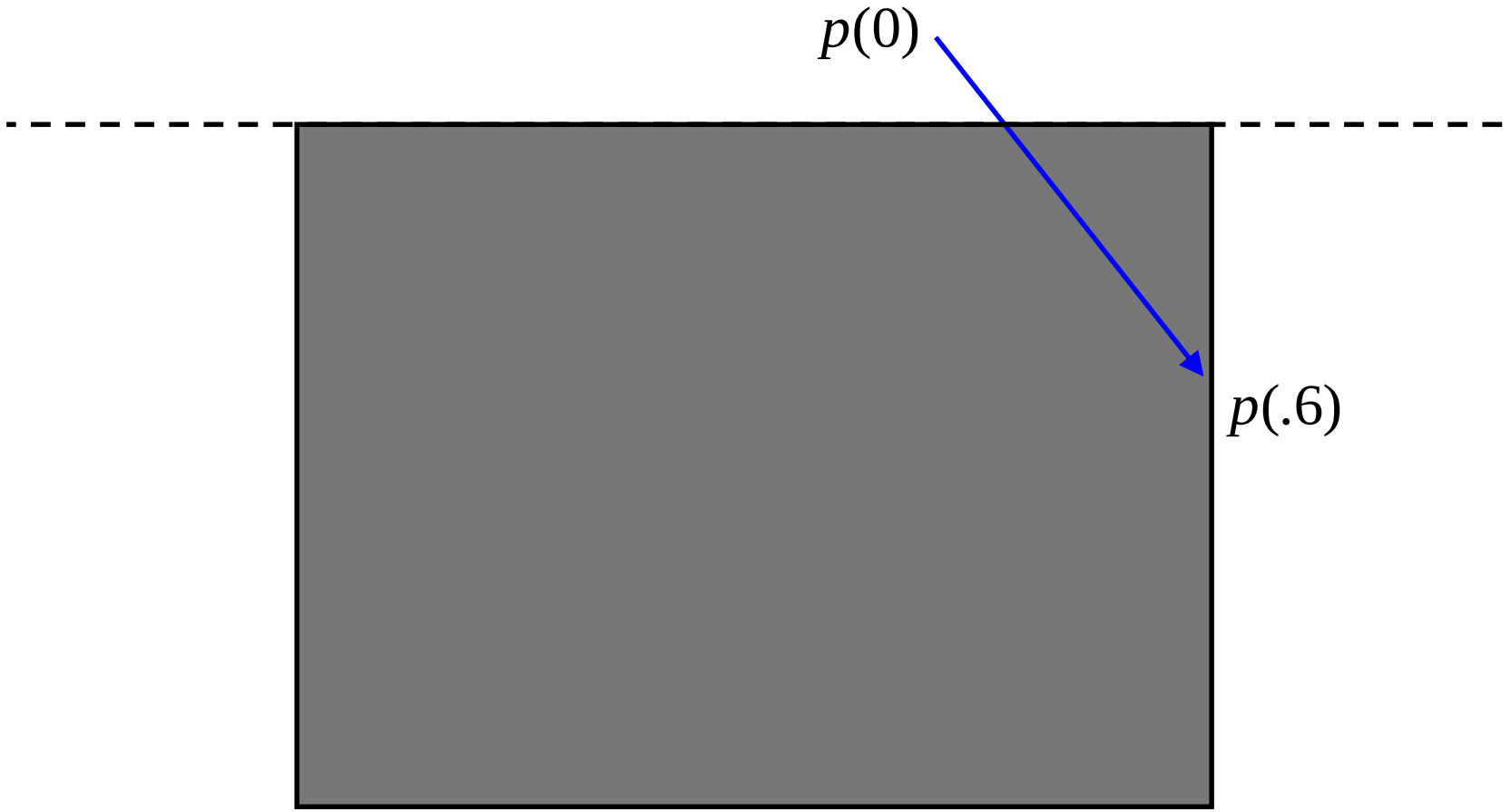
Liang-Barsky Algorithm



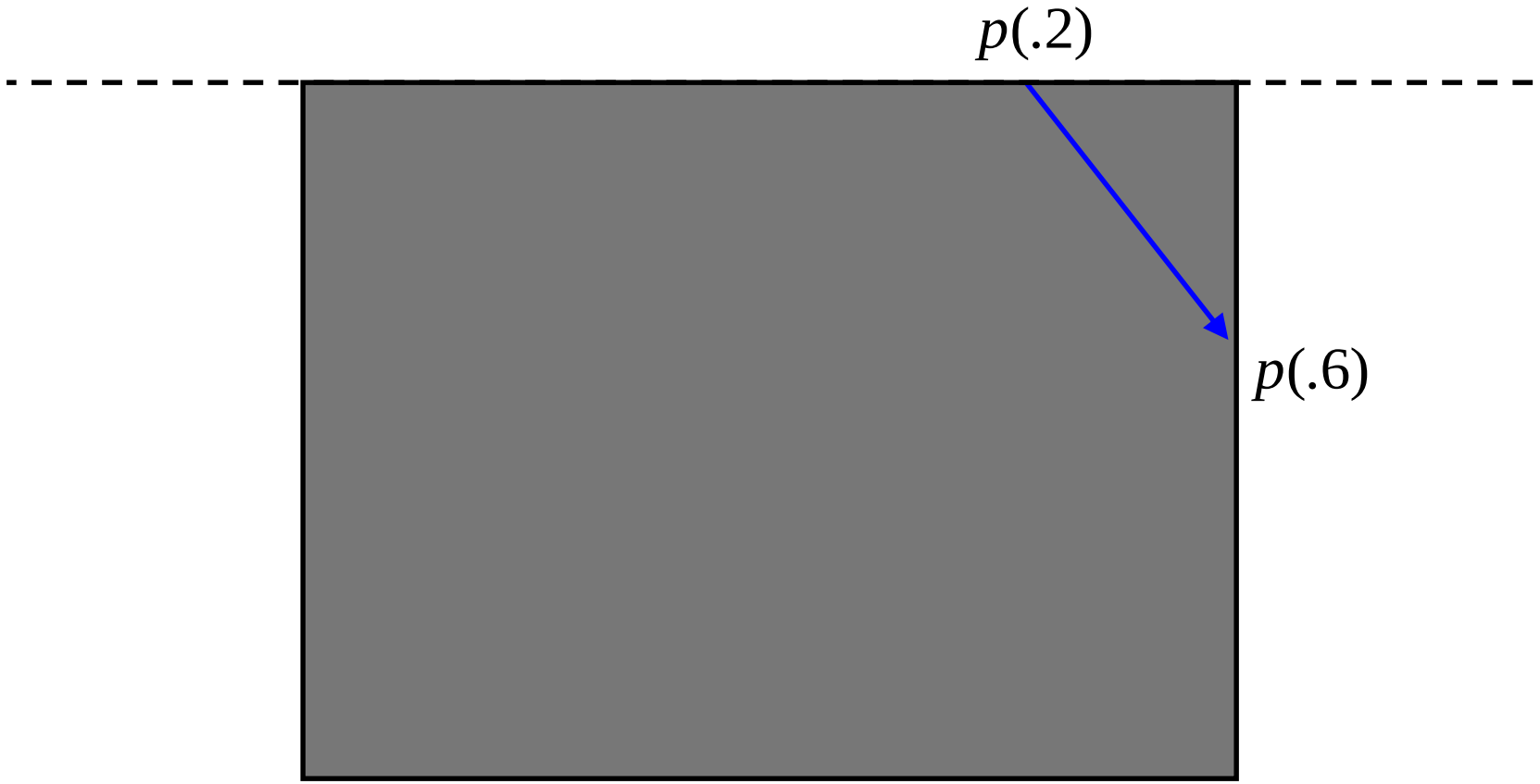
Liang-Barsky Algorithm



Liang-Barsky Algorithm



Liang-Barsky Algorithm



Liang-Barsky Algorithm

```
for each line segment to be clipped
  Pre-calculate pk and qk;
  if P1 = P0 then line is degenerate so clip as a point;
  else
    begin
      u1 = 0; u2 = 1;
      for each candidate intersection with a  clip edge
        if pk = 0 then {Ignore edges parallel to line}
          begin
            calculate u; {of line and clip edge intersection}
            use sign of pk to categorize as Pstart or Pend;
            if  Pstart then u1 = max(0,u1,u);
            if Pend then u2 = min(1,u2,u);
          end
        if u1 > u2 then return   nil
        else return P(u1) and P(u2) as true clip intersections
  end
```

Liang-Barsky Algorithm

- Faster than Cohen-Sutherland

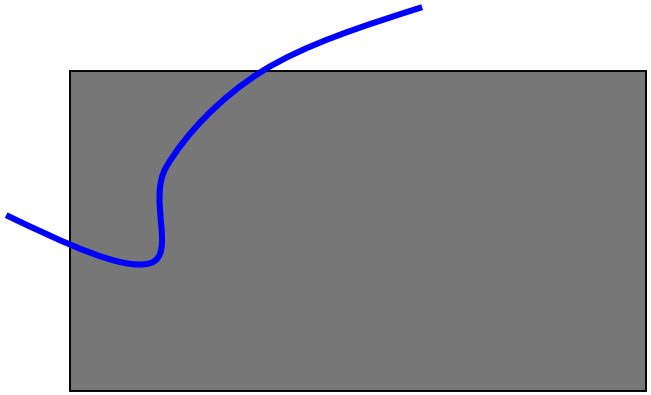
- Extension to 3D is easy
  - Parametric representation for 3D lines
  - Compute u1,u2 based on the intersection between line and plane

Comparison

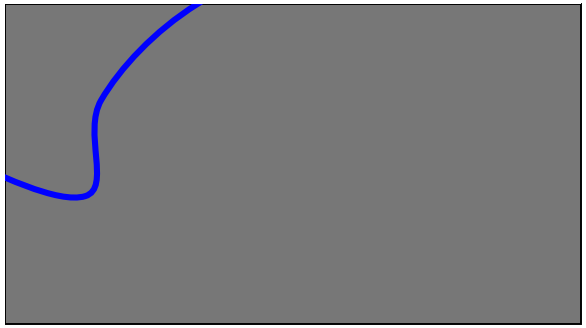
- Cohen-Sutherland
  - Repeated clipping is expensive
  - Best used when trivial acceptance and rejection is possible for most lines
- Liang-Barsky
  - Computation of t-intersections is cheap (only one division)
  - Computation of (x,y) clip points is only done once
  - Algorithm doesn't consider trivial accepts/rejects
  - Best when many lines must be clipped
- Nicholl et al.: Fastest, but doesn't do 3D

Αποκοπή καμπύλων

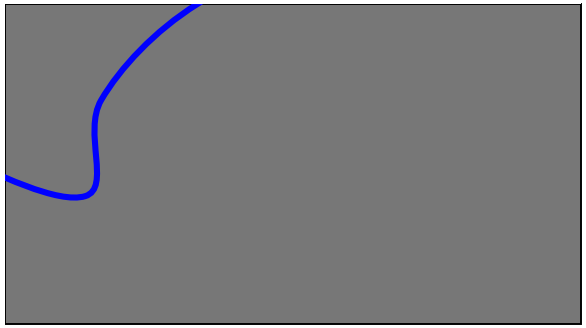
Curve Clipping



Curve Clipping



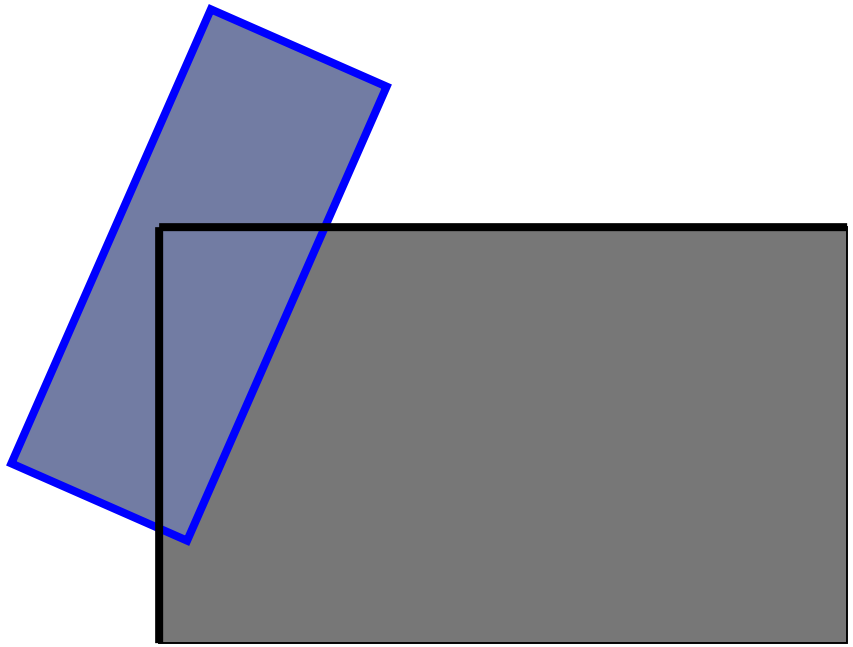
Curve Clipping



- Approximate a curve using a set of straight-line segments
- Apply line clipping for curve clipping

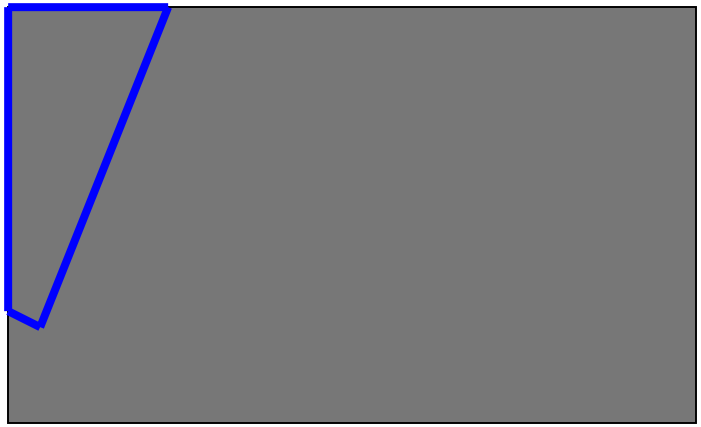
Next Lecture

- Polygon fill-area clipping



Next Lecture

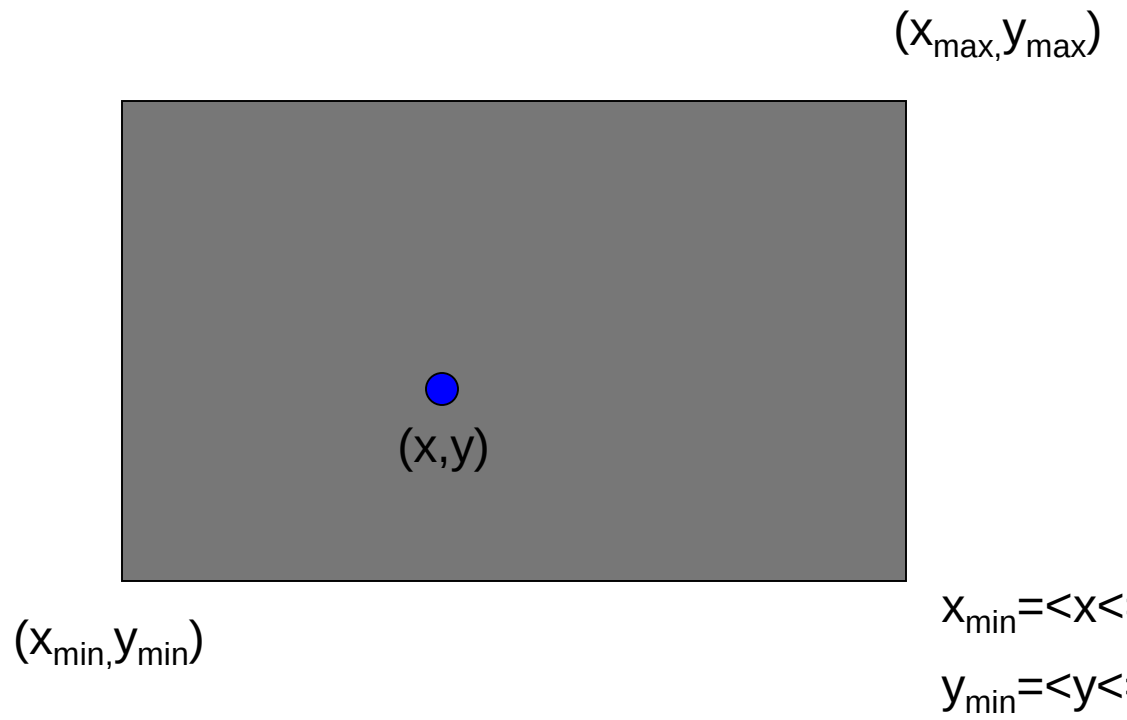
- Polygon fill-area clipping



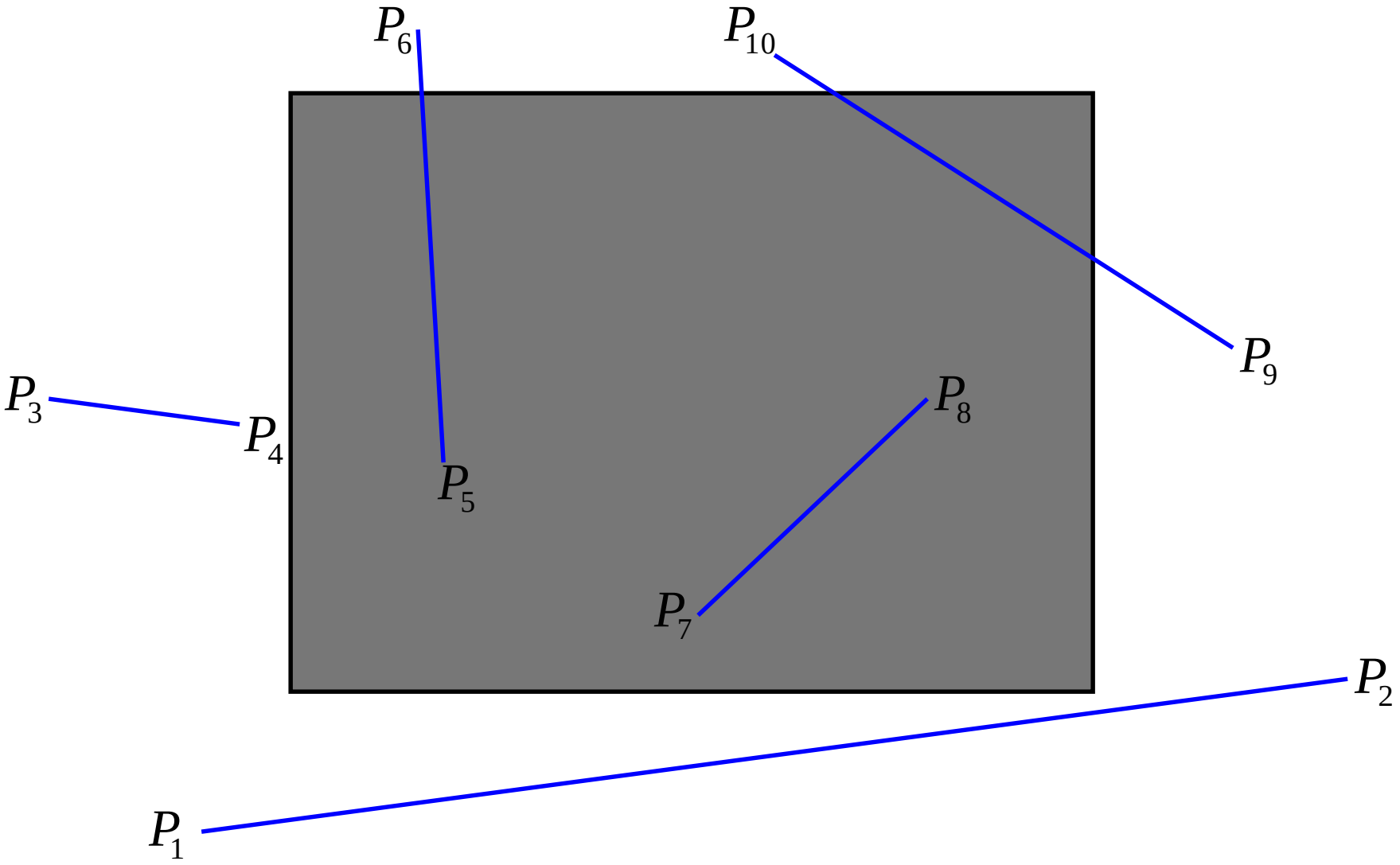
Αποκοπή πολυγώνων

Review: Clipping Points

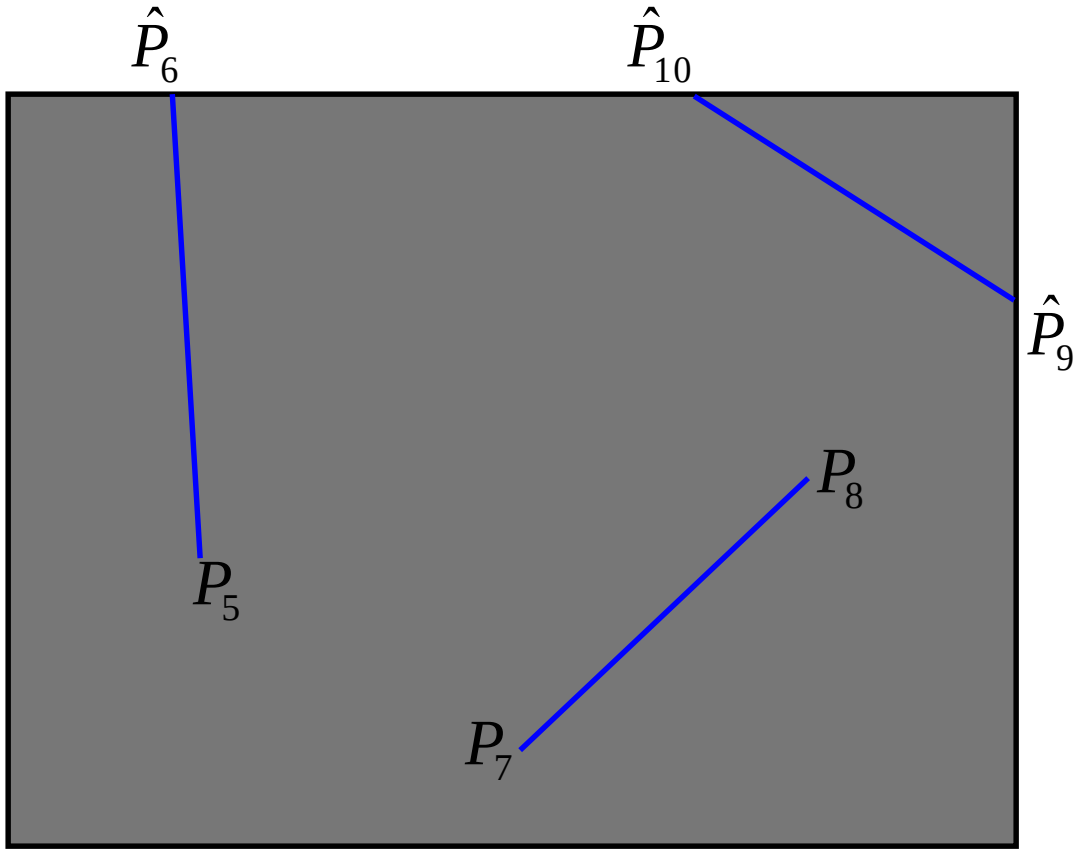
- Given a point (x, y) and clipping window (x_{min}, y_{min}), (x_{max}, y_{max}), determine if the point should be drawn



Review: Clipping Lines



Review: Clipping Lines

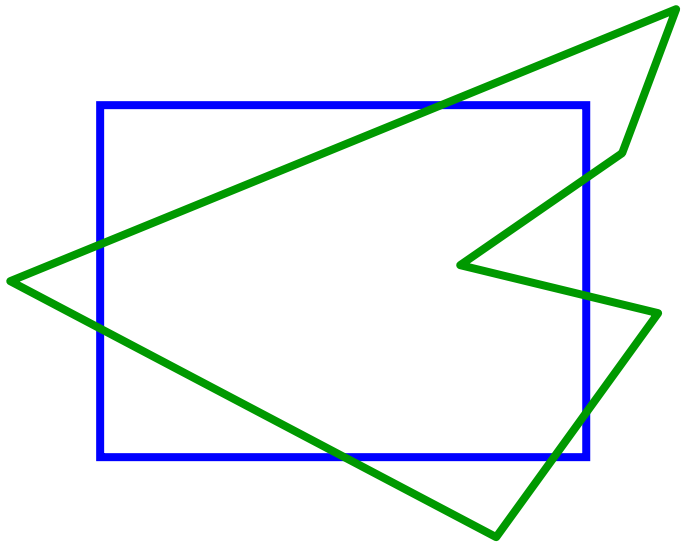


Review Clipping

- Simple line clipping algorithm
  - compute intersection points
  - clip the line segment between the outside point and intersection point
- Cohen-Sutherland
  - region coding (9 regions, 4bits)
  - best used when trivial acceptance and rejection is possible for most lines
- Liang-Barsky
  - computation of t-intersections is cheap (only one division)
  - computation of (x,y) clip points is only done once
  - algorithm doesn't consider trivial accepts/rejects
  - best when many lines must be clipped

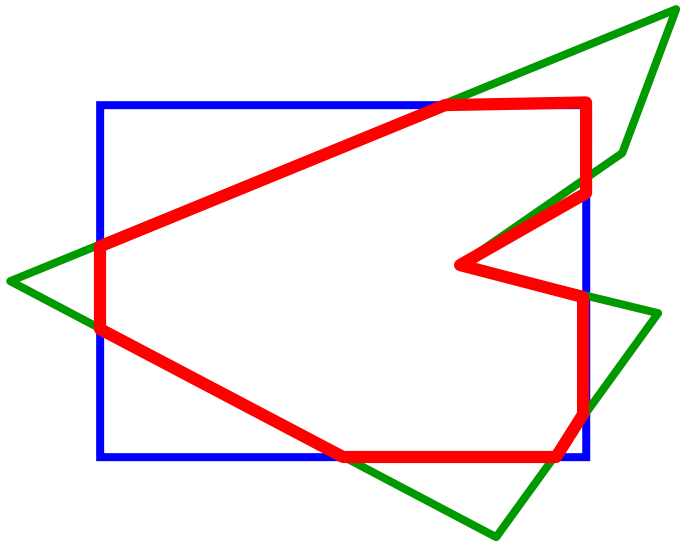
Clipping Polygons

- Clipping polygons is more complex than clipping the individual lines
  - - Input: polygon



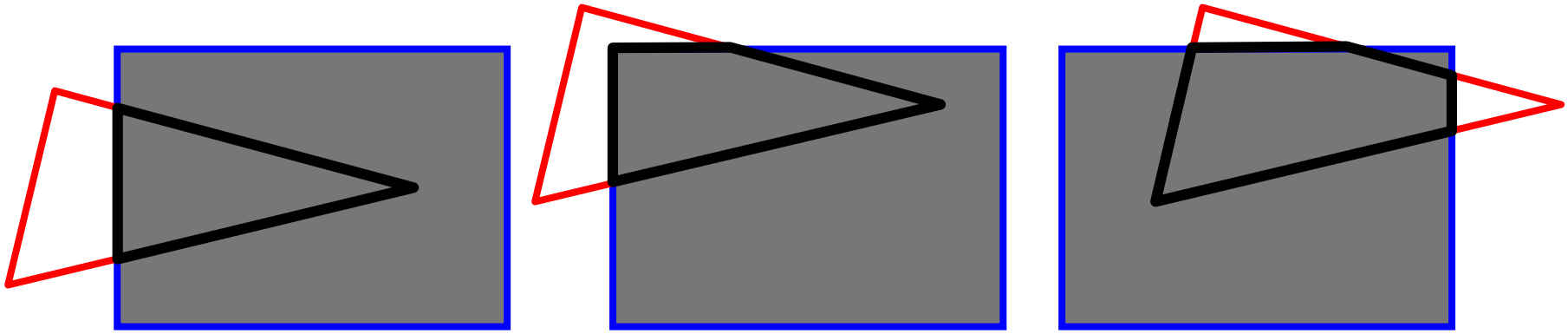
Clipping Polygons

- Clipping polygons is more complex than clipping the individual lines
  - - Input: polygon
  - - Output: original polygon, new polygon, or nothing



Why Is Polygon Clipping Hard?

- What happens to a triangle during clipping?
  - possible outcomes:



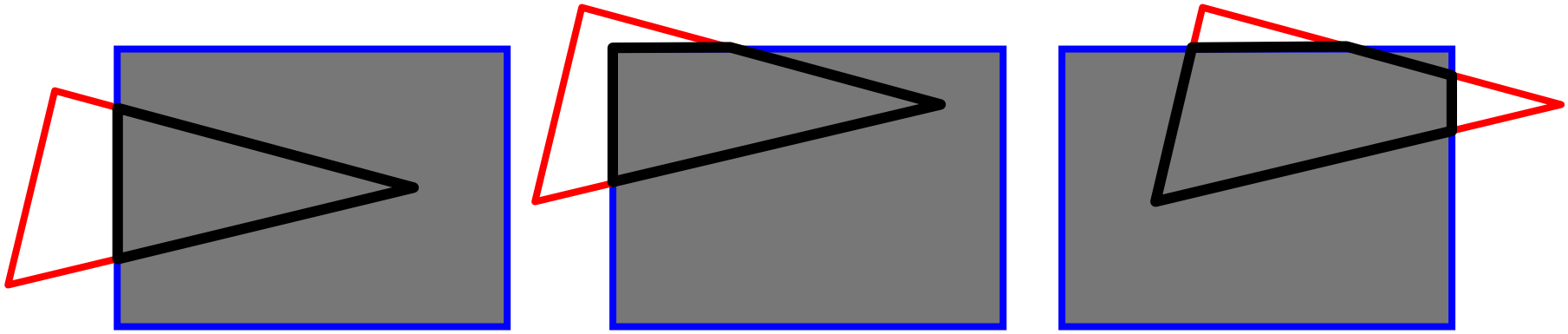
triangle->triangle triangle->quad triangle->5-gon

Why Is Polygon Clipping Hard?

- What happens to a triangle during clipping?
  - possible outcomes: triangle, quad, or else?

Why Is Polygon Clipping Hard?

- What happens to a triangle during clipping?
  - possible outcomes:

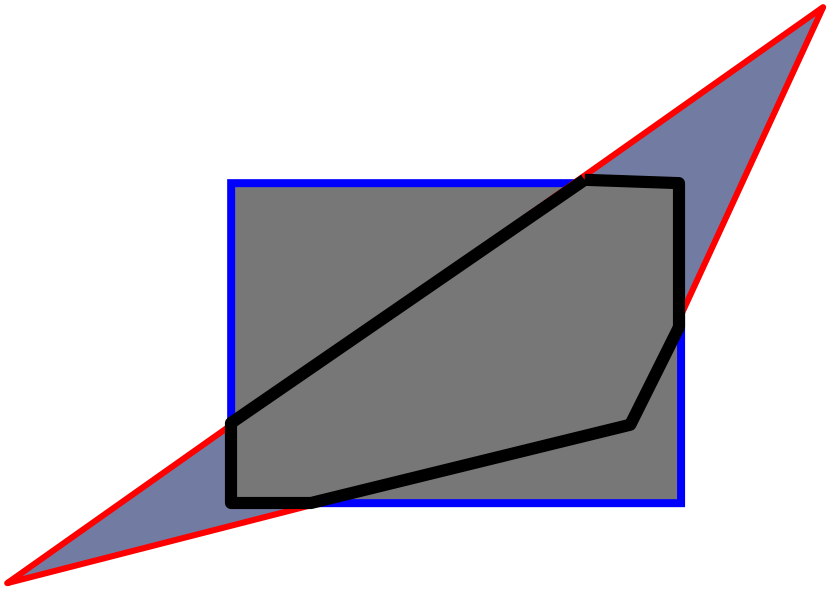


triangle->triangle triangle->quad triangle->5-gon

How many sides a clipped triangle might have?

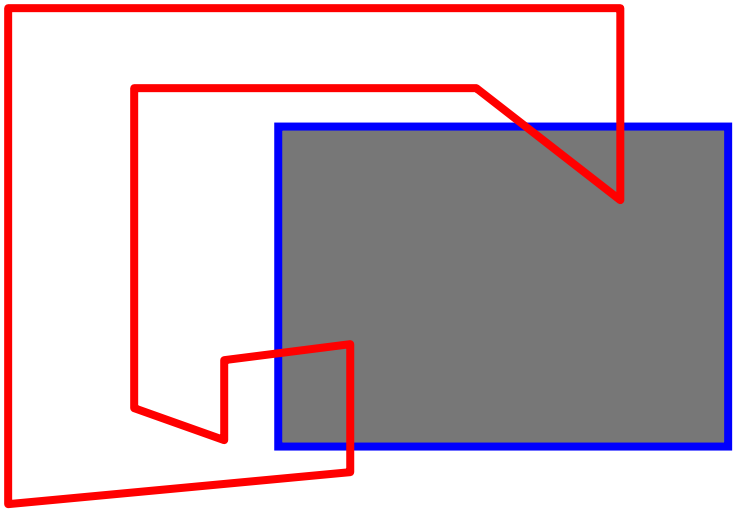
How Many Sides?

- Seven



Why Is Clipping Hard?

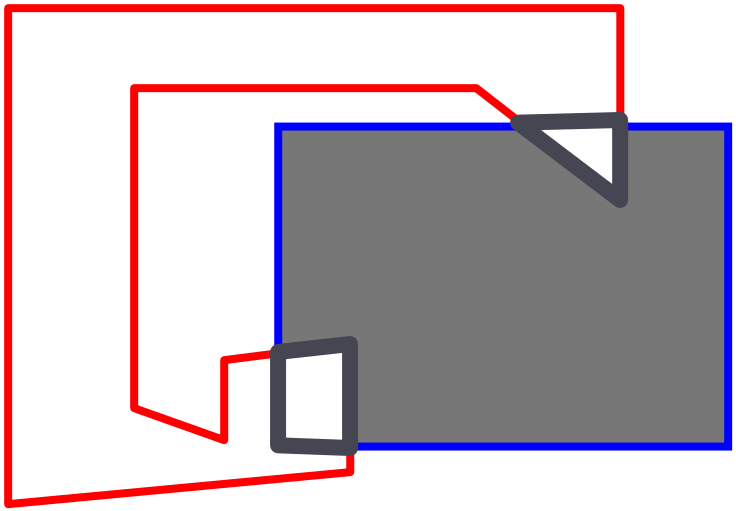
- A really tough case



Concave polygon -> Multiple polygons

Why Is Clipping Hard?

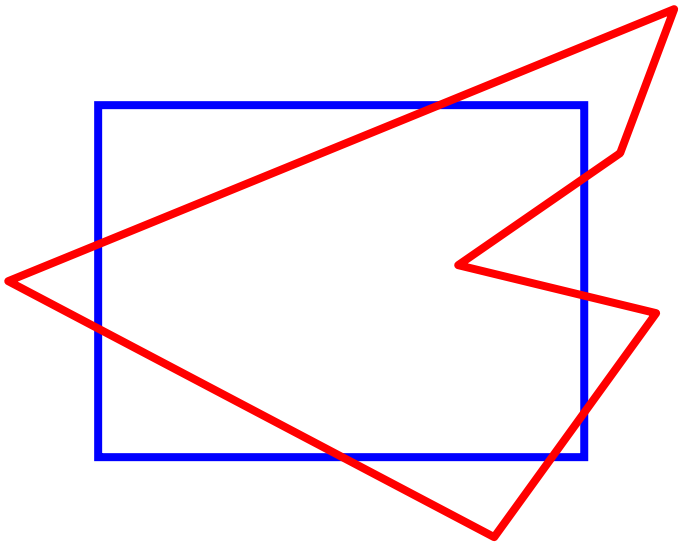
- A really tough case



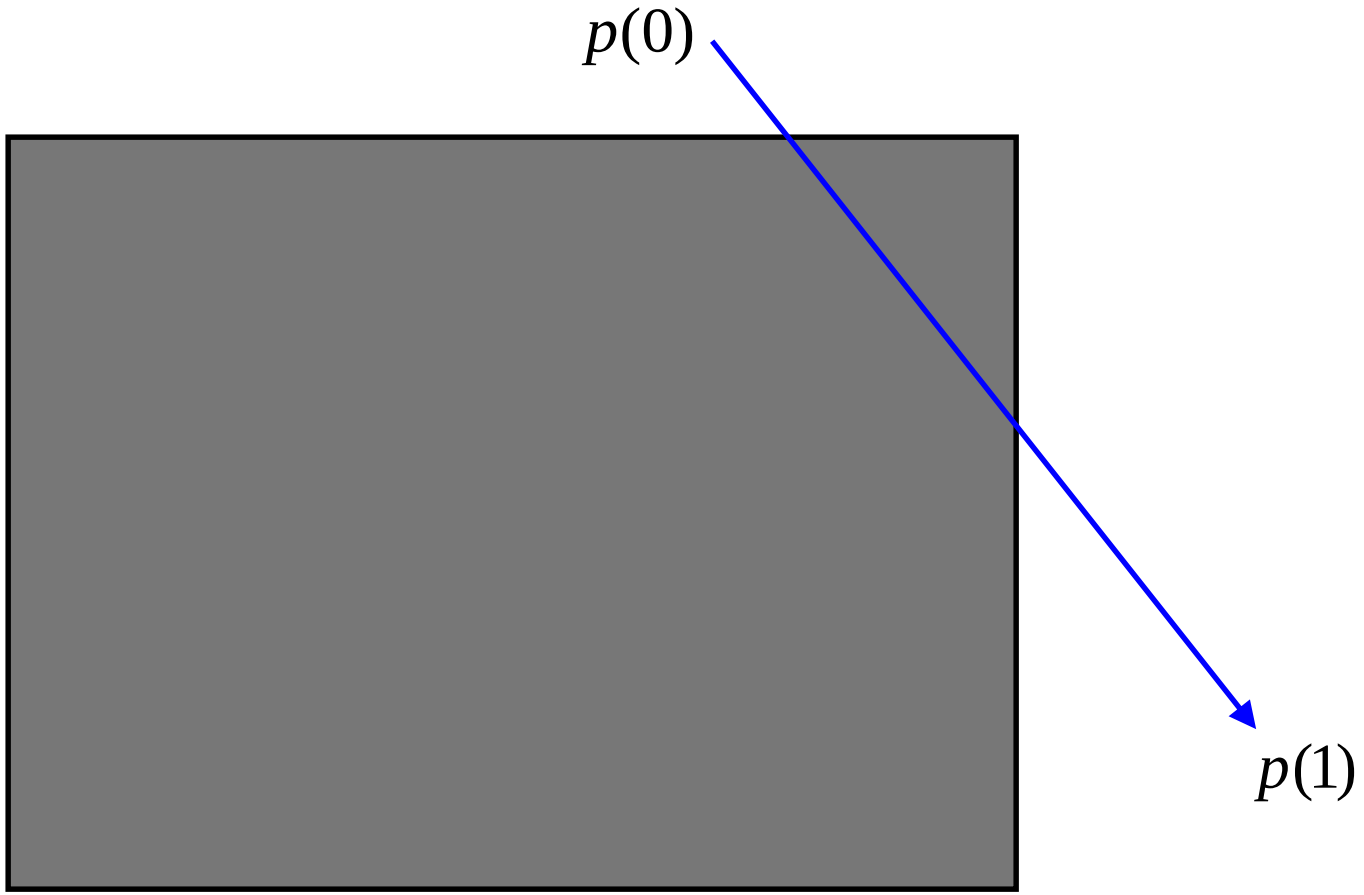
Outline

- Sutherland-Hodgman Clipping
  - convex polygons
- Weiler-Atherton Algorithm
  - general polygons

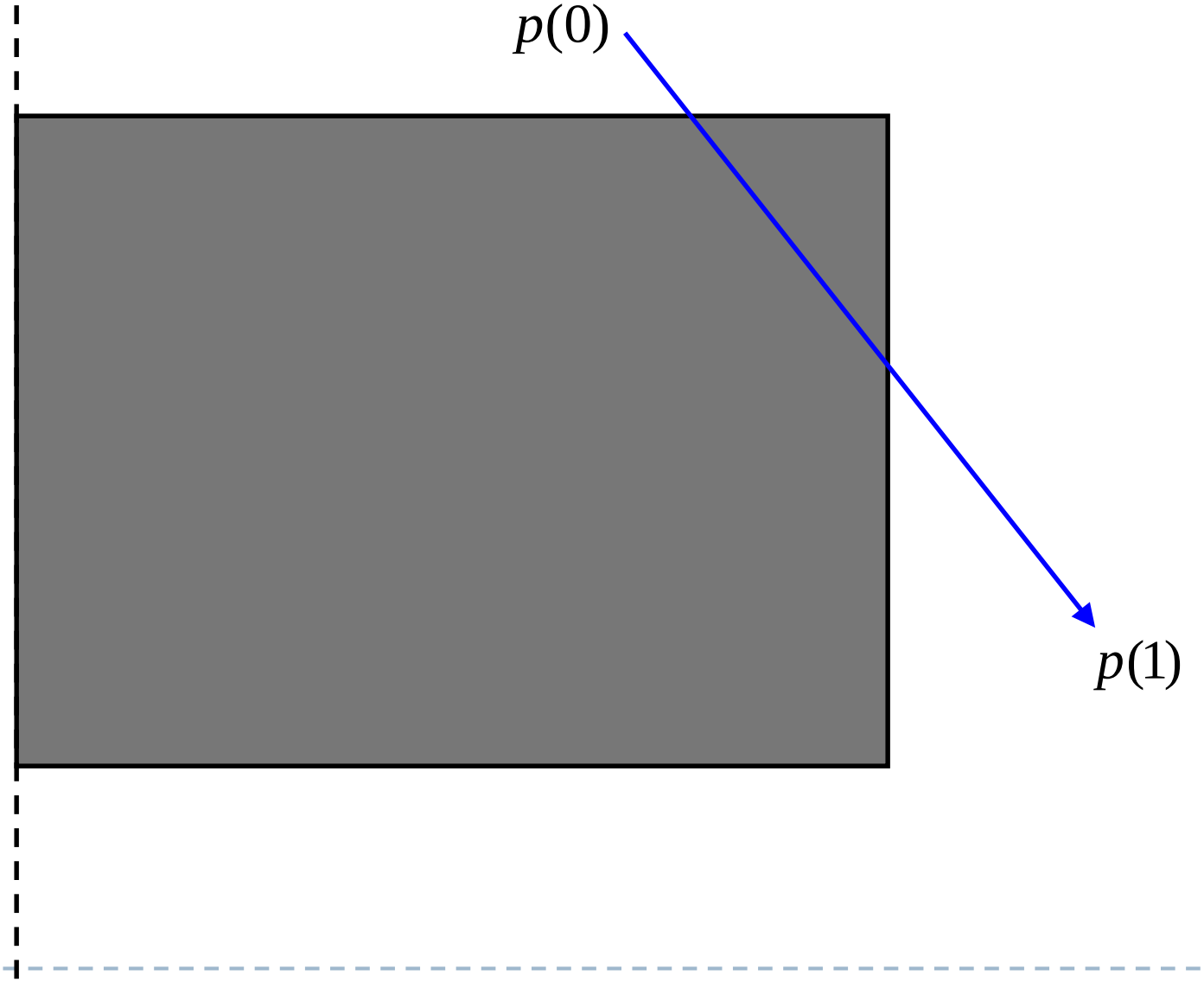
Any idea?



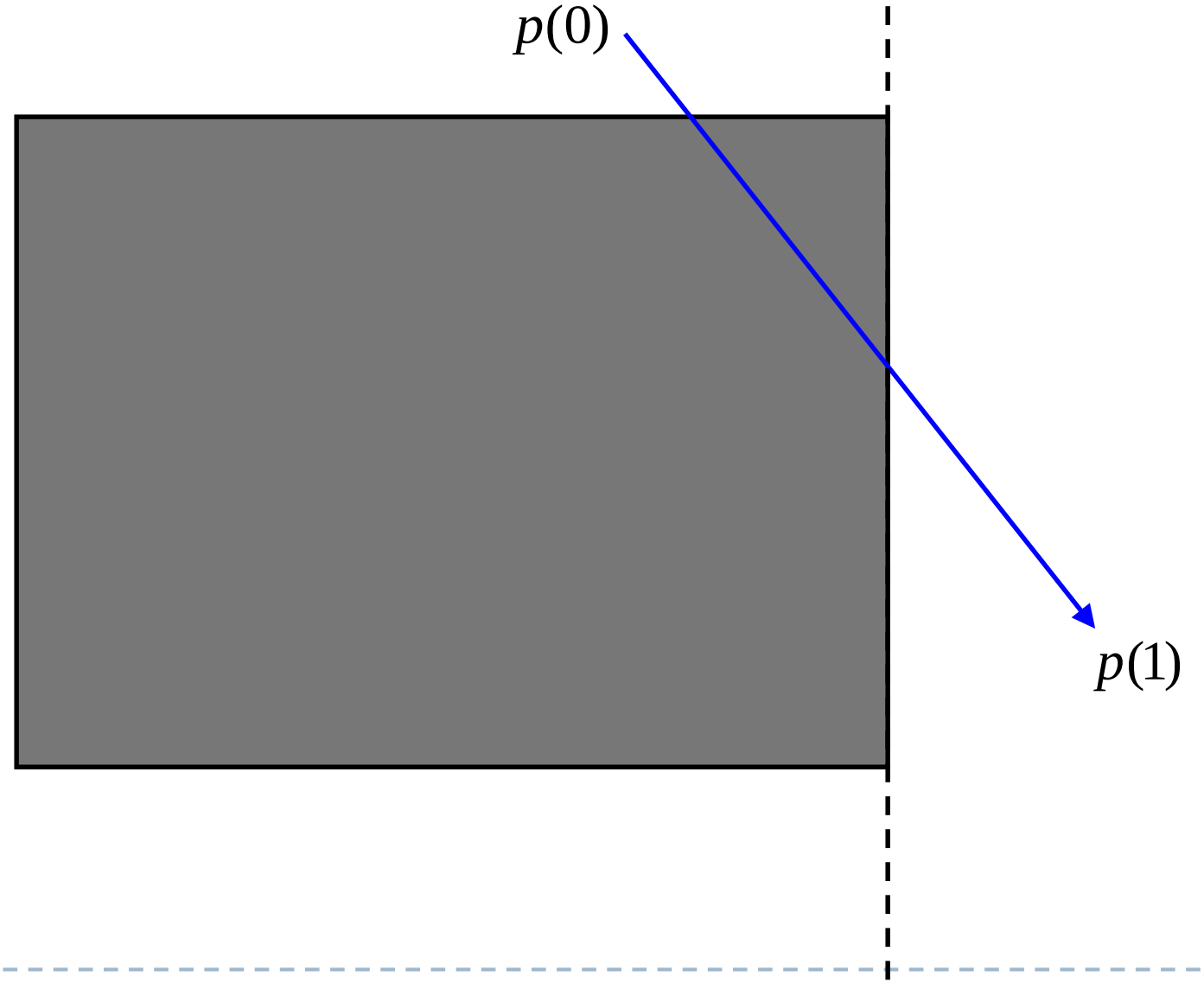
Review: Liang-Barsky Algorithm



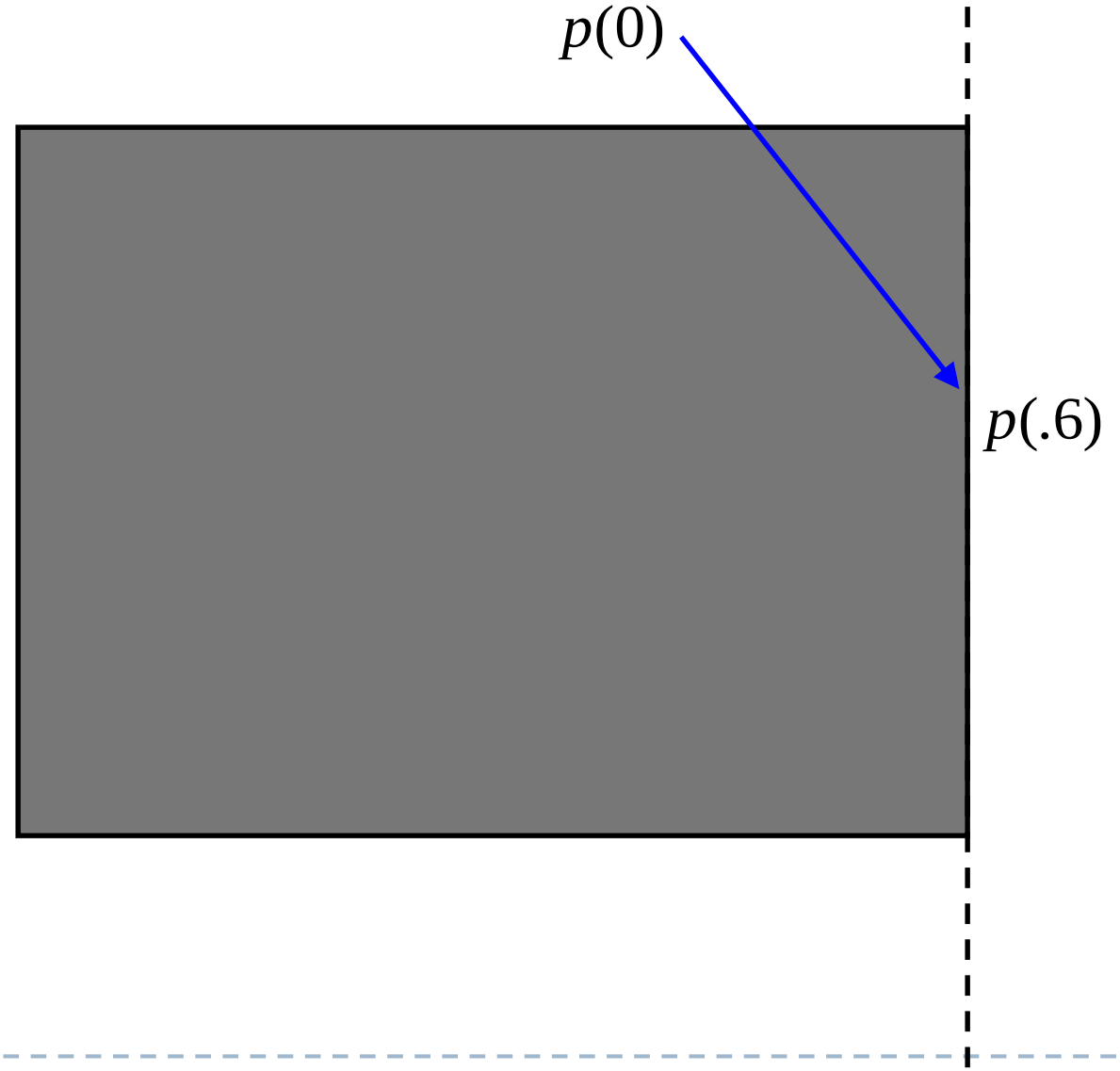
Review: Liang-Barsky Algorithm



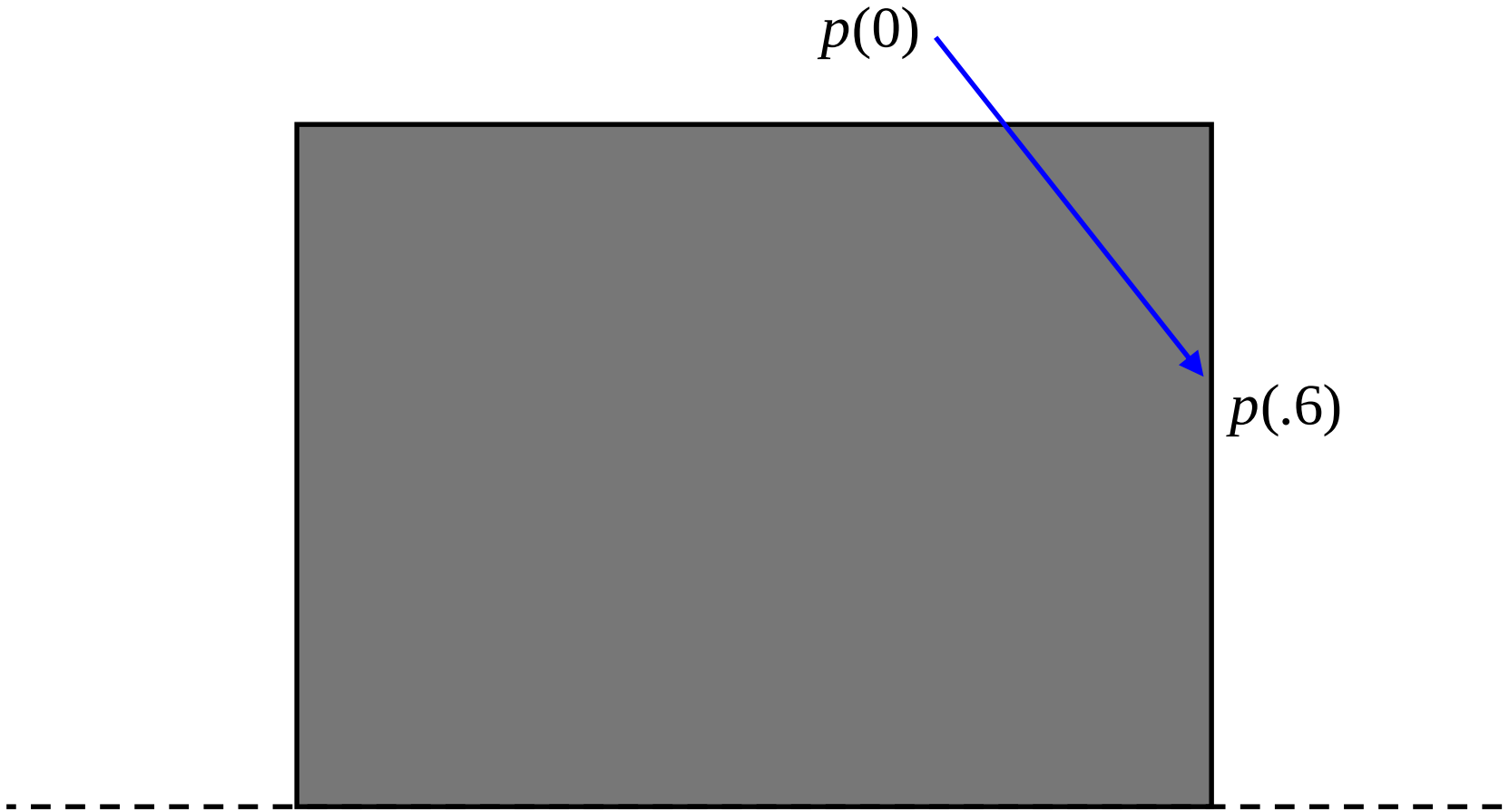
Review: Liang-Barsky Algorithm



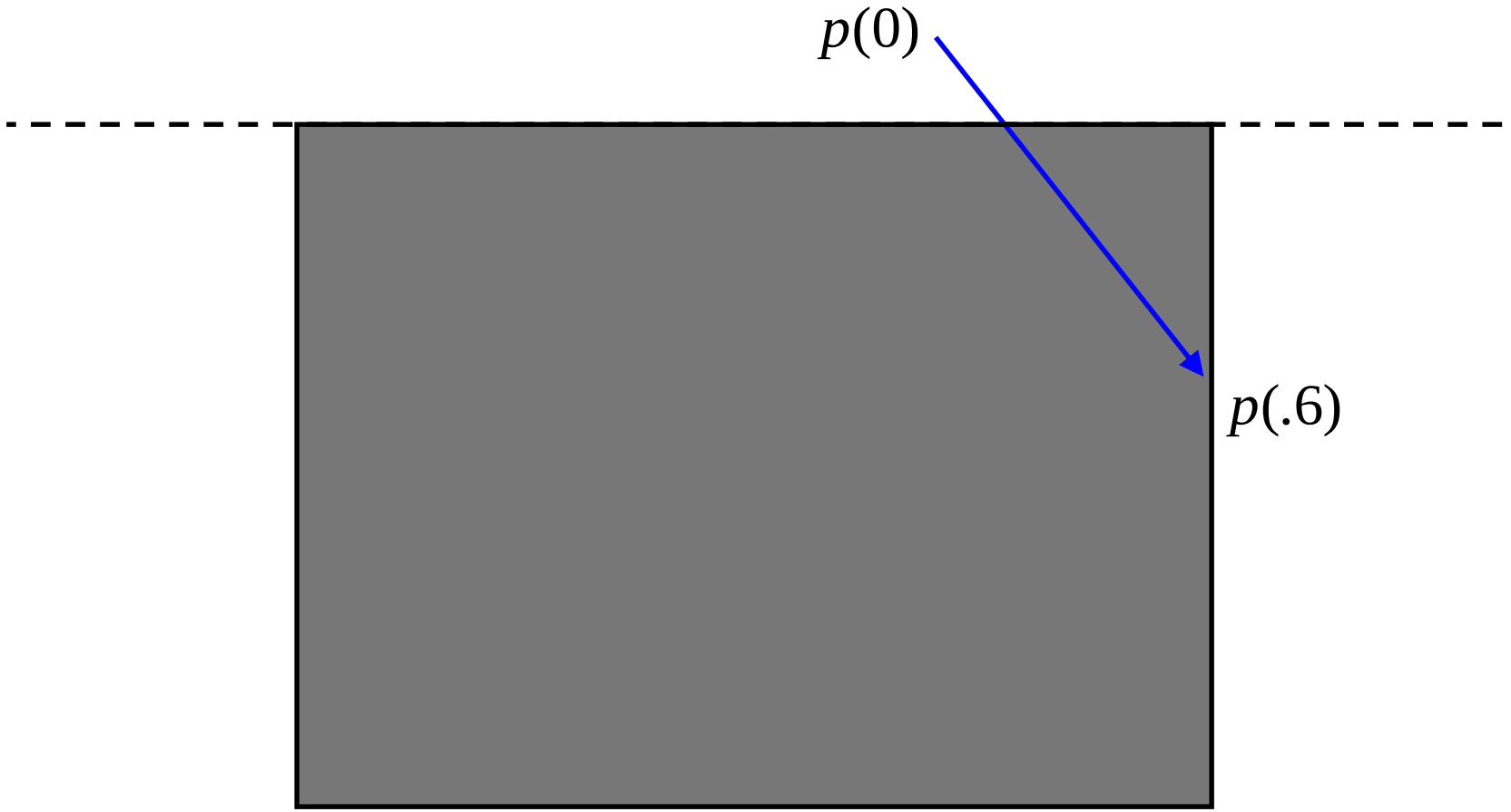
Liang-Barsky Algorithm



Review: Liang-Barsky Algorithm



Review: Liang-Barsky Algorithm



Review: Liang-Barsky Algorithm