

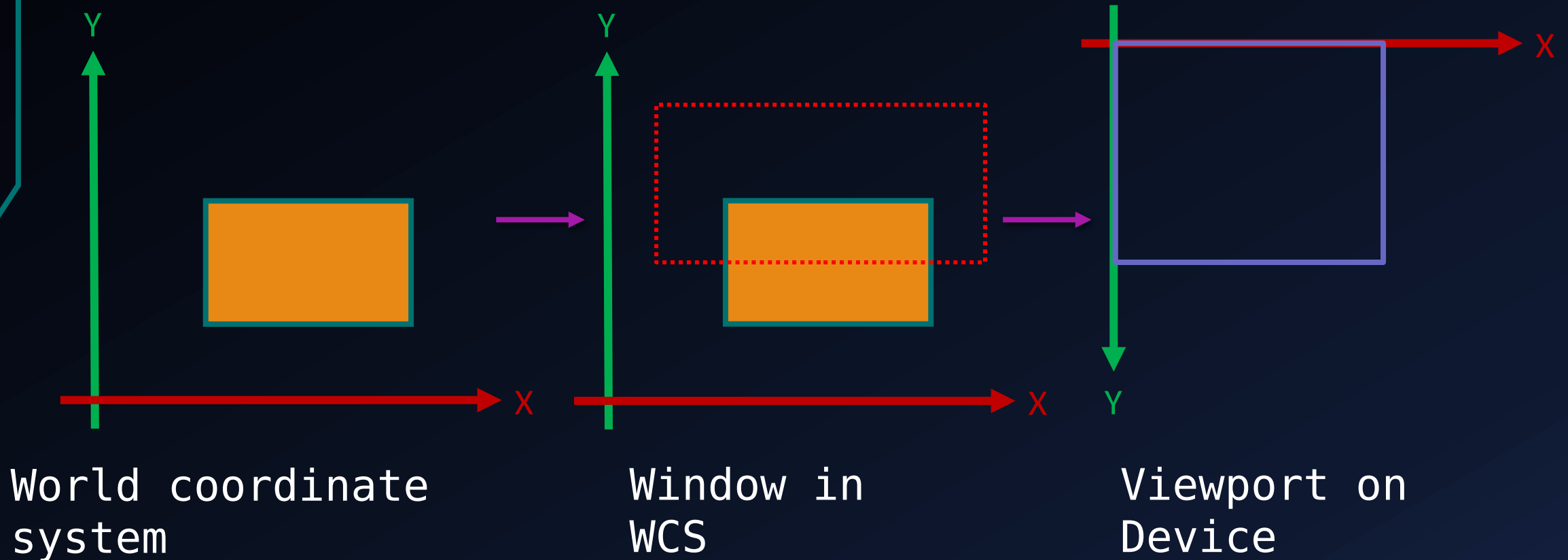
PLANAR CLIPPING

- Clipping lines and polygons
- Analytical clipping
- Cohen-Sutherland algorithm
- Liang-Barsky algorithm

Author: prof. Yevhenii Borodavka

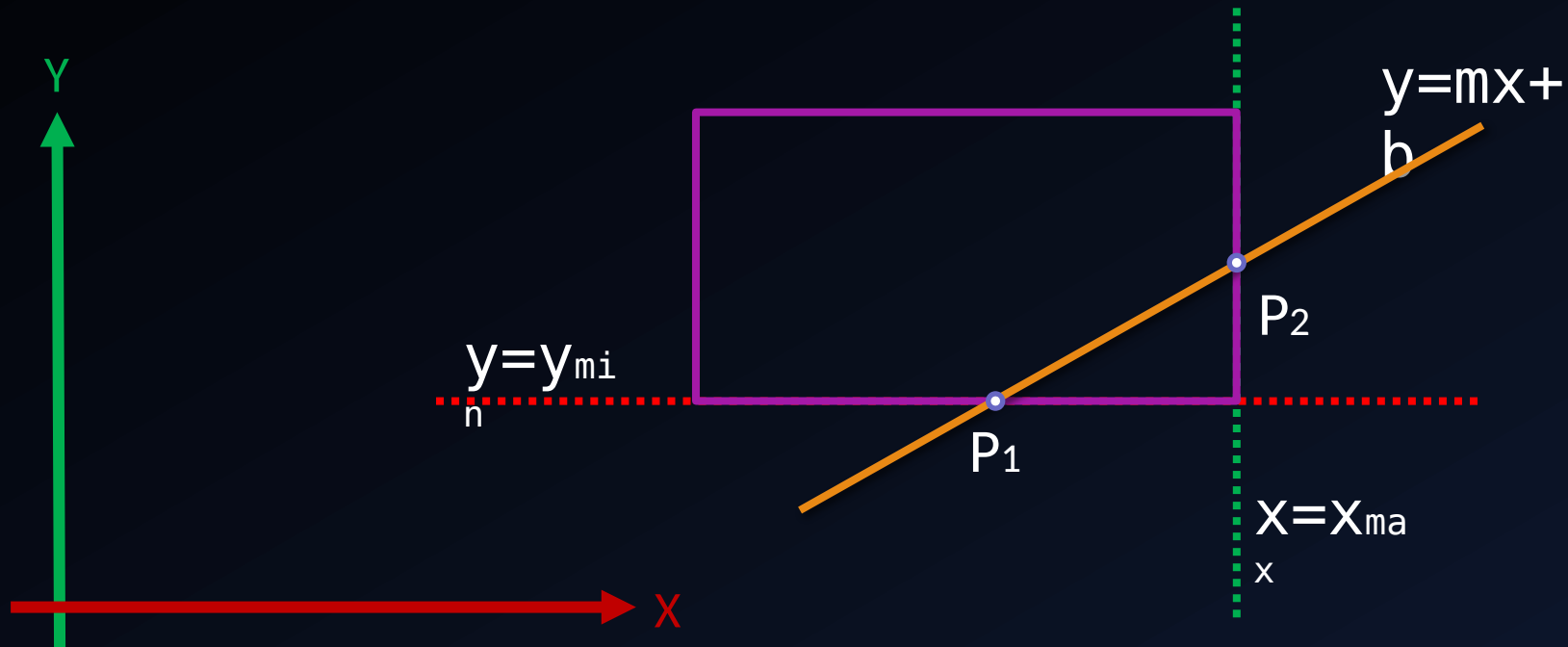
CLIPPING LINES AND POLYGONS

Goal: only render what will be visible in the drawing window; eliminate from the rendering pipeline all those primitives which will not appear in the final viewport.



ANALYTICAL CLIPPING

Brute force method: compute all intersections of line with clip rectangle



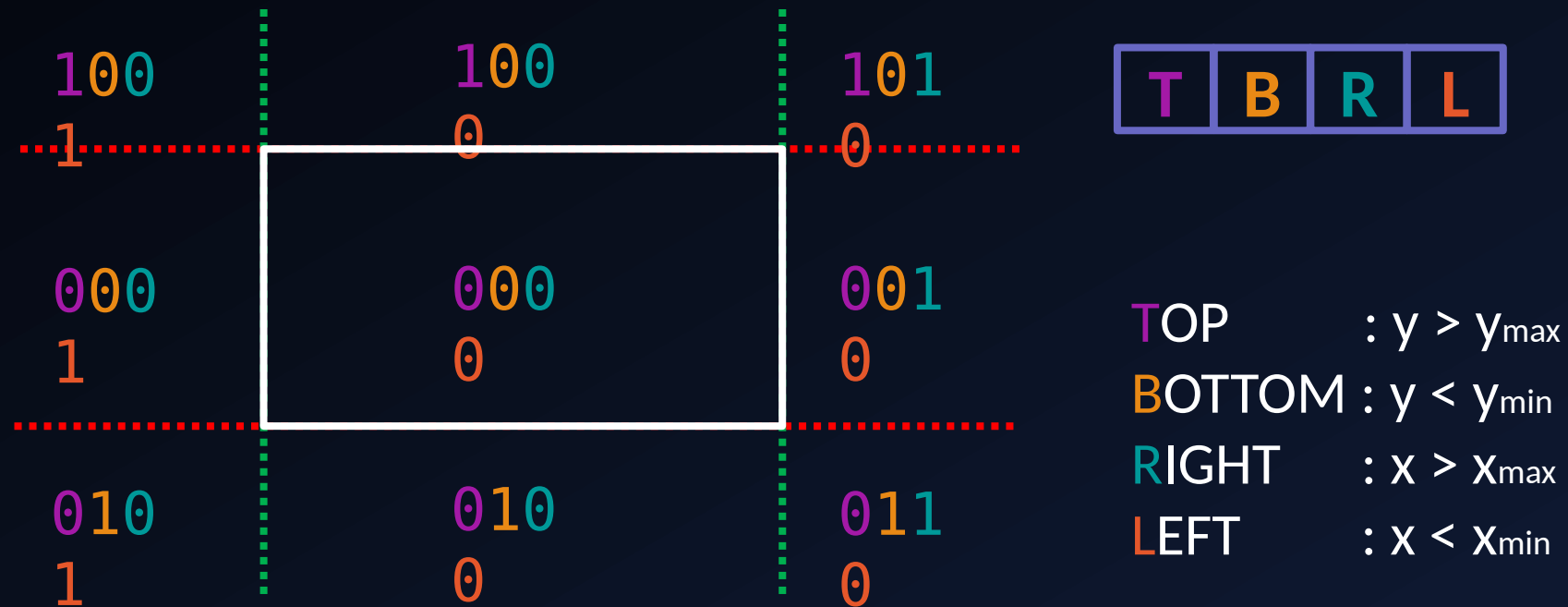
$$P_1 = \{y = y_{\min}\} \cap \{y = mx + b\}$$
$$P_2 = \{x = x_{\max}\} \cap \{y = mx + b\}$$

COHEN-SUTHERLAND ALGORITHM

Use regions to accept/reject lines.

Assume the clipping region is an axis-oriented rectangle.

Regions are labeled according to the half-planes defined by the clipping region.



COHEN-SUTHERLAND ALGORITHM

Let c_1 and c_2 be the region codes for the endpoints of the line to be clipped.

Trivially Accept: both codes are 0000 then segment is inside the clip region.

Trivially Reject: (c_1 & c_2) are not equal to zero then line is outside clip rectangle.

Repeat until the segment cannot be trivially accepted or rejected:

- Pick the endpoint which is outside the clip rectangle (one must be outside)
- Find the first non-zero bit: this corresponds to the clip edge which intersects the line
- Compute the intersection of the line with the edge
- Throw away the outside vertex up to the clip rectangle

LEFT

$$y = y_0 + (y_1 - y_0) * (x_{\min} - x_0) / (x_1 - x_0)$$

$$x = x_{\min}$$

TOP

$$x = x_0 + (x_1 - x_0) * (y_{\max} - y_0) / (y_1 - y_0)$$

$$y = y_{\max}$$

BOTTOM

$$x = x_0 + (x_1 - x_0) * (y_{\min} - y_0) / (y_1 - y_0)$$

$$y = y_{\min}$$

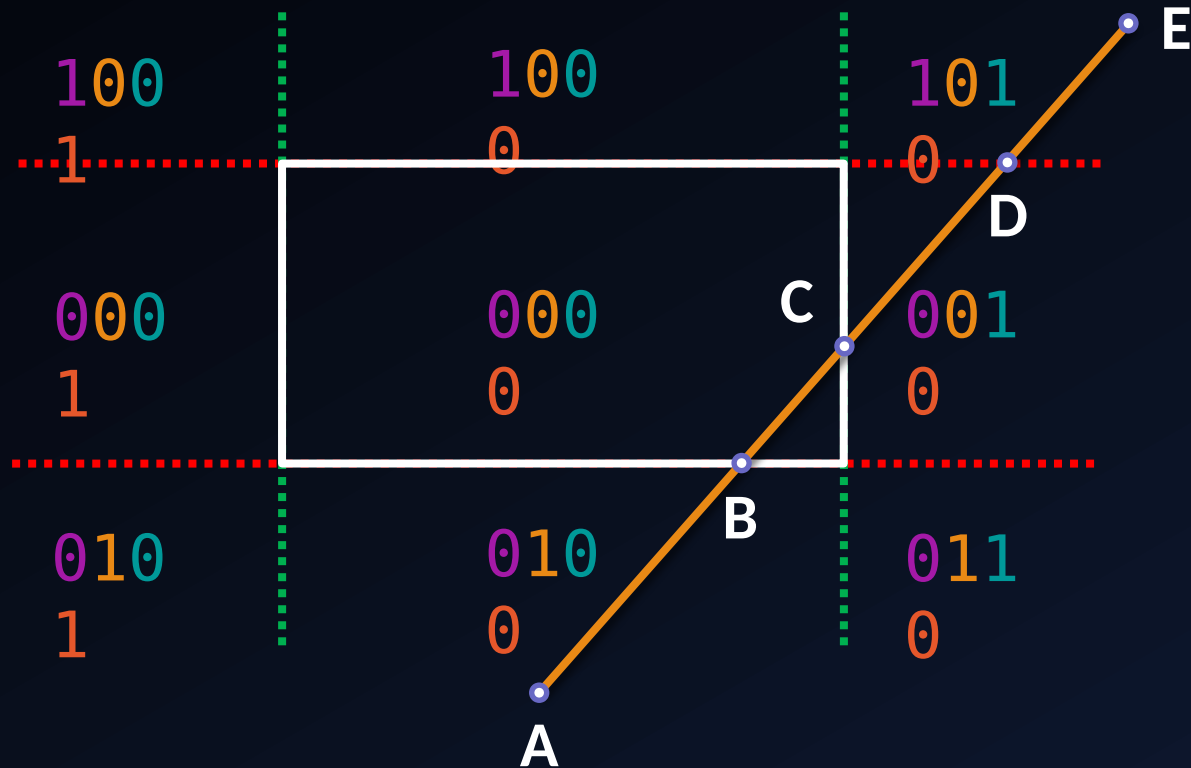
RIGHT

$$y = y_0 + (y_1 - y_0) * (x_{\max} - x_0) / (x_1 - x_0)$$

$$x = x_{\max}$$

COHEN-SUTHERLAND ALGORITHM

An example. STEP 1.



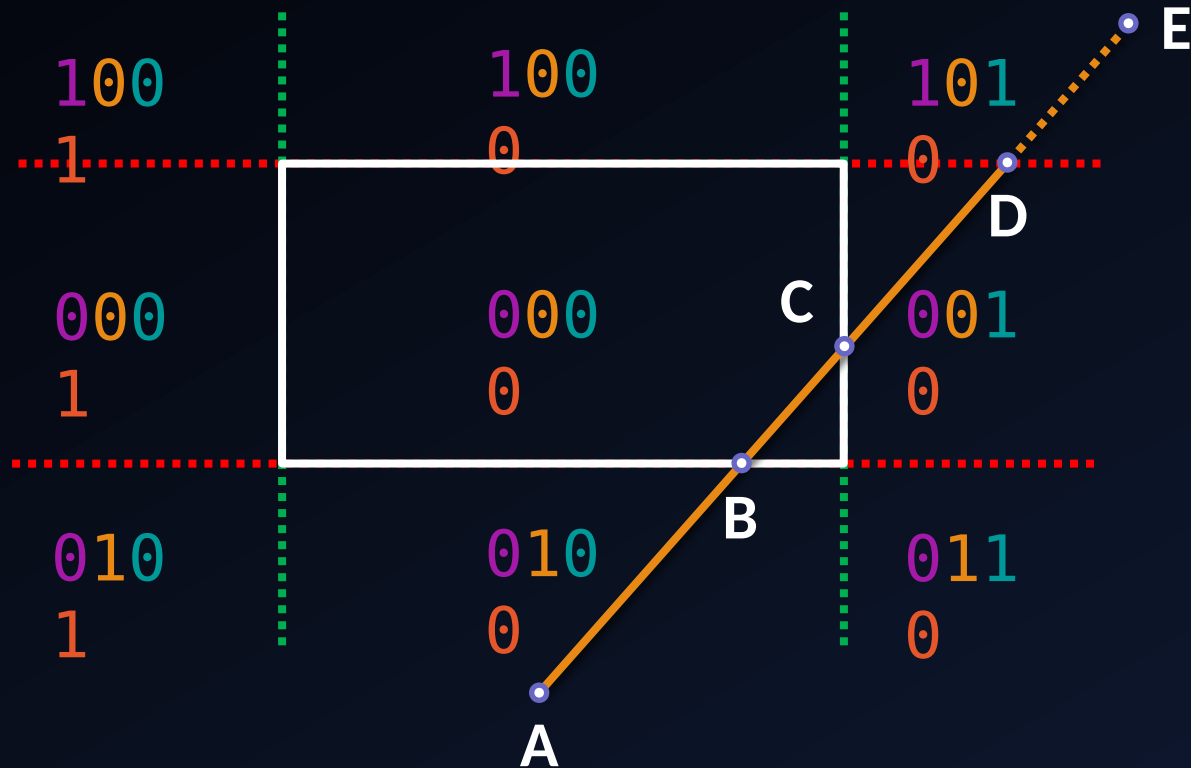
E : 1010

A : 0100 Subdivide using

& : 0000 either A or E

COHEN-SUTHERLAND ALGORITHM

An example. STEP 2.



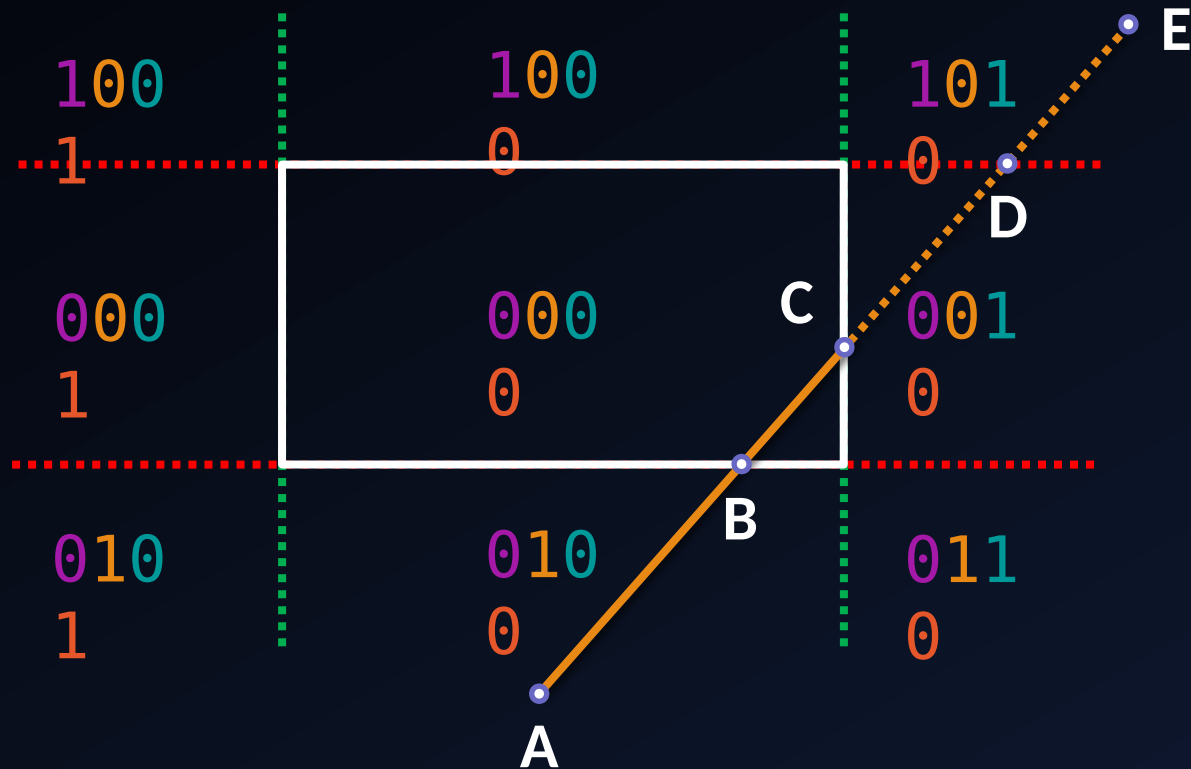
D : 0010

A : 0100 Subdivide: third bit of D indicates that the

& : 0000 line intersects the right clip line

COHEN-SUTHERLAND ALGORITHM

An example. STEP 3.



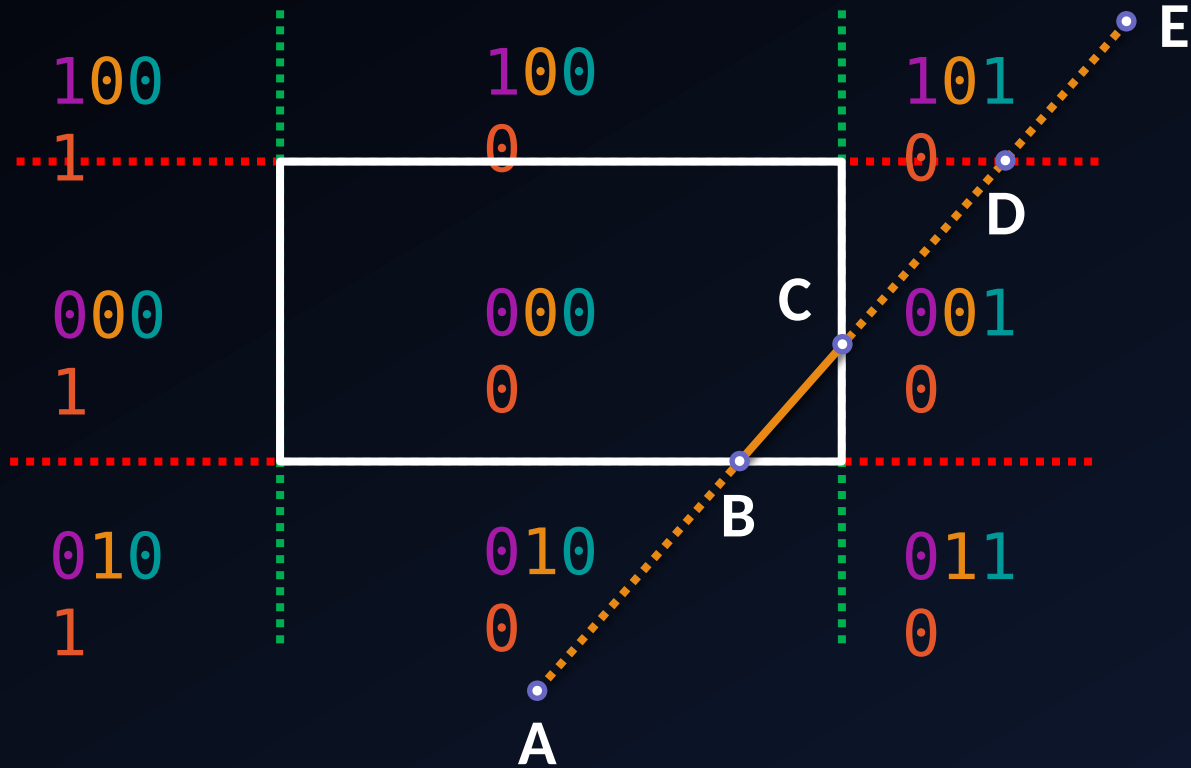
C : 0000

A : 0100 Subdivide: second bit of A indicates that the

& : 0000 line intersects the bottom clip line

COHEN-SUTHERLAND ALGORITHM

An example. STEP 4.



C : 0000

B : 0000 Trivially accept the

& : 0000 clipped segment BC

COHEN-SUTHERLAND ALGORITHM

Disadvantages:

- Fixed-order decision can do needless work
- Can improve using more regions (Nicholl-Lee-Nicholl Algorithm)
- Can generate more efficient rejection tests
- Clipping window must be rectangular

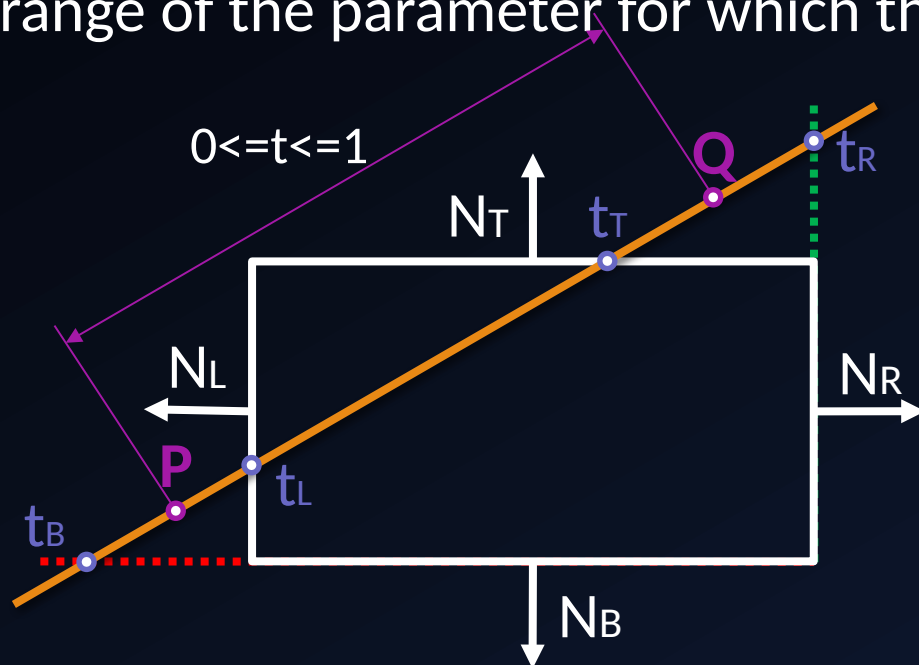
Advantages:

- Simple to implement
- Oriented for most simple window/viewport systems
- Extends to 3-D cubic volumes

LIANG-BARSKY ALGORITHM

The ideas for clipping line of Liang-Barsky and Cyrus-Beck are the same. The only difference is Liang-Barsky algorithm has been optimized for an upright rectangular clip window.

Liang and Barsky have created an algorithm that uses floating-point arithmetic but finds the appropriate end points with at most four computations. This algorithm uses the parametric equations for a line and solves four inequalities to find the range of the parameter for which the line is in the viewport.



Let $P(x_1, y_1)$, $Q(x_2, y_2)$ be the line which we want to study. The parametric equation of the line segment from gives x-values and y-values for every point in terms of a parameter that ranges from 0 to 1. The equations are:

$$x = x_1 + (x_2 - x_1) * t = x_1 + dx * t$$

$$y = y_1 + (y_2 - y_1) * t = y_1 + dy * t$$

LIANG-BARSKY ALGORITHM

The algorithm

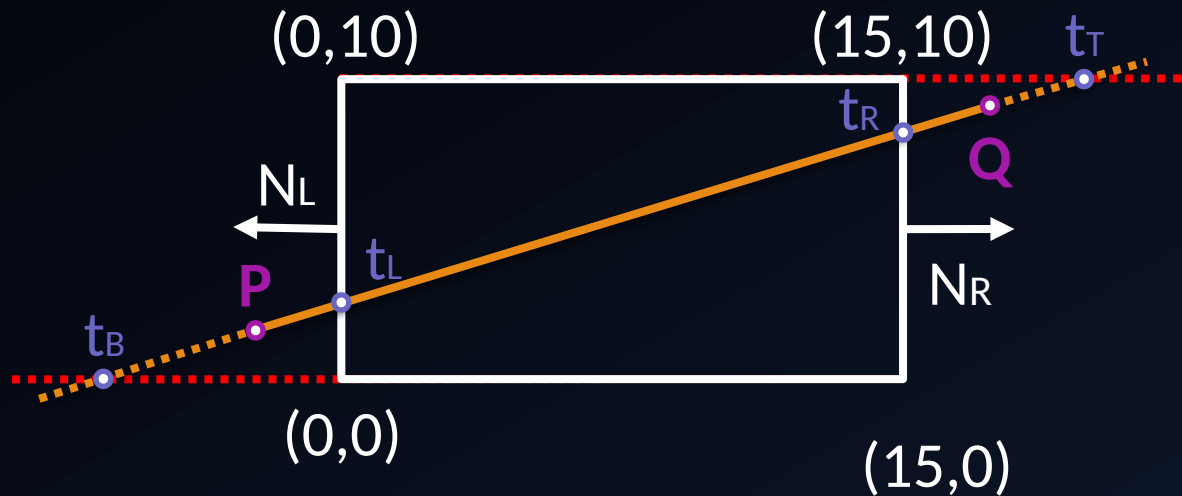
1. Set $t_{\min}=0$ and $t_{\max}=1$
2. Calculate the values of t_L , t_R , t_T , and t_B
 - if $t < t_{\min}$ or $t > t_{\max}$ ignore it and go to the next edge
 - otherwise classify the t value as entering or exiting (using inner product with edge normals to classify)
 - if t is entering value set $t_{\min}=t$; if t is exiting value set $t_{\max}=t$
3. If $t_{\min} < t_{\max}$ then draw a line from $(x_1+dx*t_{\min}, y_1+dy*t_{\min})$ to $(x_1+dx*t_{\max}, y_1+dy*t_{\max})$
4. If $t_{\min} > t_{\max}$ then segment out of the viewport and rejected

$$t_B = \frac{(B - y_1)}{(y_2 - y_1)} \quad t_L = \frac{(L - x_1)}{(x_2 - x_1)} \quad t_T = \frac{(T - y_1)}{(y_2 - y_1)} \quad t_R = \frac{(R - x_1)}{(x_2 - x_1)}$$

LIANG-BARSKY ALGORITHM

The example #1. STEP 1. The t values computation.

$B=0$, $T=10$, $L=0$, $R=15$, $P(-5,3)$, $Q(20,9)$.



$$t_B = \frac{(0 - 3)}{(9 - 3)} = -\frac{1}{2}$$

$$t_L = \frac{(0 - (-5))}{(20 - (-5))} = \frac{1}{5}$$

$$t_R = \frac{(15 - (-5))}{(20 - (-5))} = \frac{4}{5}$$

$$t_T = \frac{(10 - 3)}{(9 - 3)} = \frac{7}{6}$$

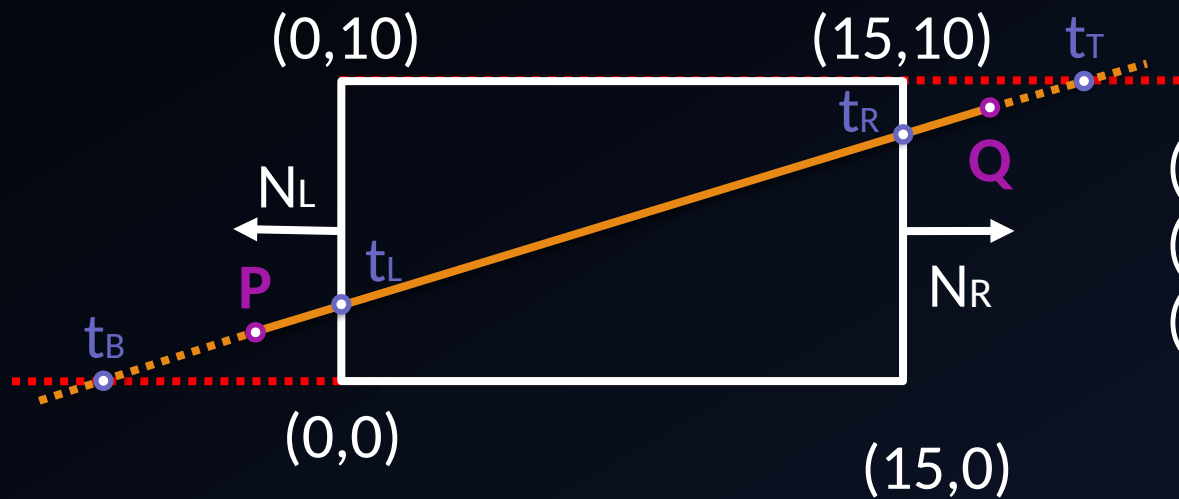
$t_B < 0$ and $t_T > 1$ are ignored

t_L and t_R are tested by inner product

LIANG-BARSKY ALGORITHM

The example #1. STEP 2. The testing with edges normal.

$B=0$, $T=10$, $L=0$, $R=15$, $P(-5,3)$, $Q(20,9)$.



$$(Q-P) = (20+5, 9-3) = (25, 6)$$

$$(Q-P) \cdot N_L = (25, 6) \cdot (-15, 0) = -25 < 0, \quad t_{\min} = t_L$$

$$(Q-P) \cdot N_R = (25, 6) \cdot (15, 0) = 25 > 0, \quad t_{\max} = t_R$$

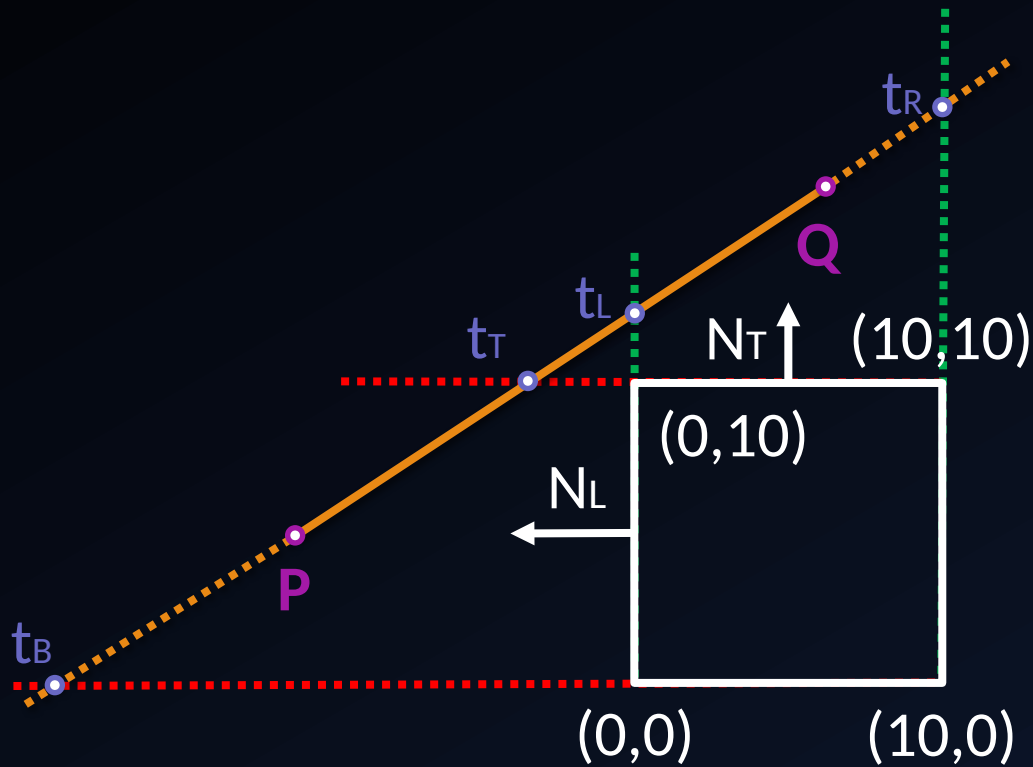
Because $t_{\min} < t_{\max}$ we draw a line

from $(-5+25 \cdot (1/5), 3+6 \cdot (1/5))$ to $(-5+25 \cdot (4/5), 3+6 \cdot (4/5))$

LIANG-BARSKY ALGORITHM

The example #2. STEP 1. The t values computation.

$B=0$, $T=10$, $L=0$, $R=10$, $P(-8,2)$, $Q(2,14)$.



$$t_B = \frac{(0 - 2)}{(14 - 2)} = -\frac{1}{6}$$

$$t_T = \frac{(10 - 2)}{(14 - 2)} = \frac{8}{12}$$

$$t_L = \frac{(0 - (-8))}{(2 - (-8))} = \frac{8}{10}$$

$$t_R = \frac{(10 - (-8))}{(2 - (-8))} = \frac{18}{10}$$

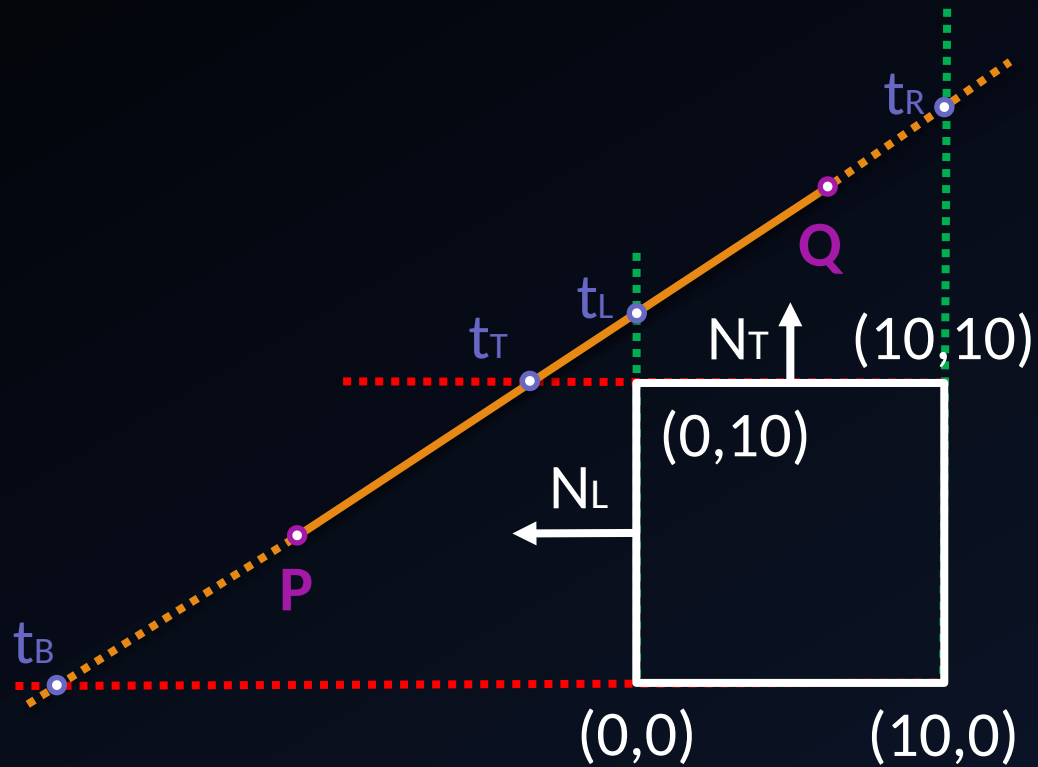
$t_B < 0$ and $t_R > 1$ are ignored

t_L and t_T are tested by inner product

LIANG-BARSKY ALGORITHM

The example #2. STEP 2. The testing with edges normal.

$B=0$, $T=10$, $L=0$, $R=10$, $P(-8,2)$, $Q(2,14)$.



$$(Q-P)=(2+8,14-2)=(10,12)$$

$$(Q-P) \cdot N_T=(10,12) \cdot (0,10)=120>0, \quad t_{\max}=t_T$$

$$(Q-P) \cdot N_L=(10,12) \cdot (-10,0)=-100<0, \quad t_{\min}=t_L$$

Because $t_{\min}>t_{\max}$ we don't have draw a line



Thank you!