CS230 – Game Implementation Techniques

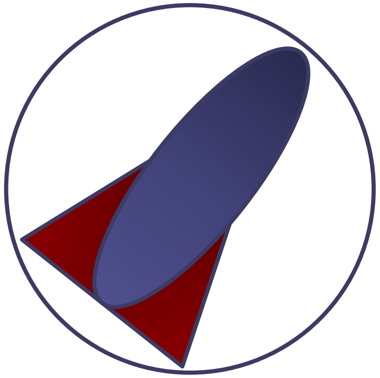
Lecture 18

Outline

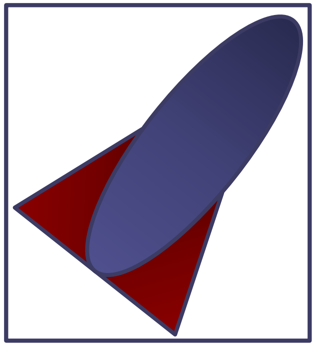
- Bounding Area
- Collision Detection
  - Rectangle - Rectangle

Bounding Area

- Definition: A bounding area (BA) is a single simple area encapsulating one or more objects of more complex nature.
  - Example:



Bounding Circle



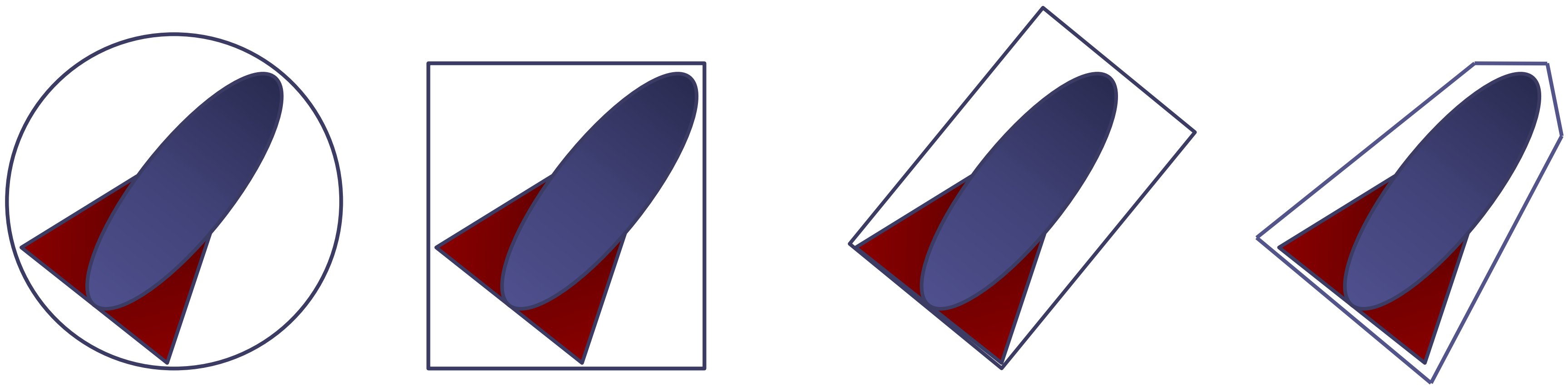
Bounding Rectangle

BA: Characteristics

- Inexpensive intersection tests
- Tight fitting
- Inexpensive to compute
- Easy to rotate and transform
- Use little memory

Better Bound, Better Culling →

← Faster Test, Less Memory



Bounding Circle

AABB (Axis Aligned Bounding Box)

OBB (Oriented Bounding Box)

Convex Hull

Outline

- Bounding Area
- Collision Detection
  - Rectangle - Rectangle

Bounding Rectangles (1/2)

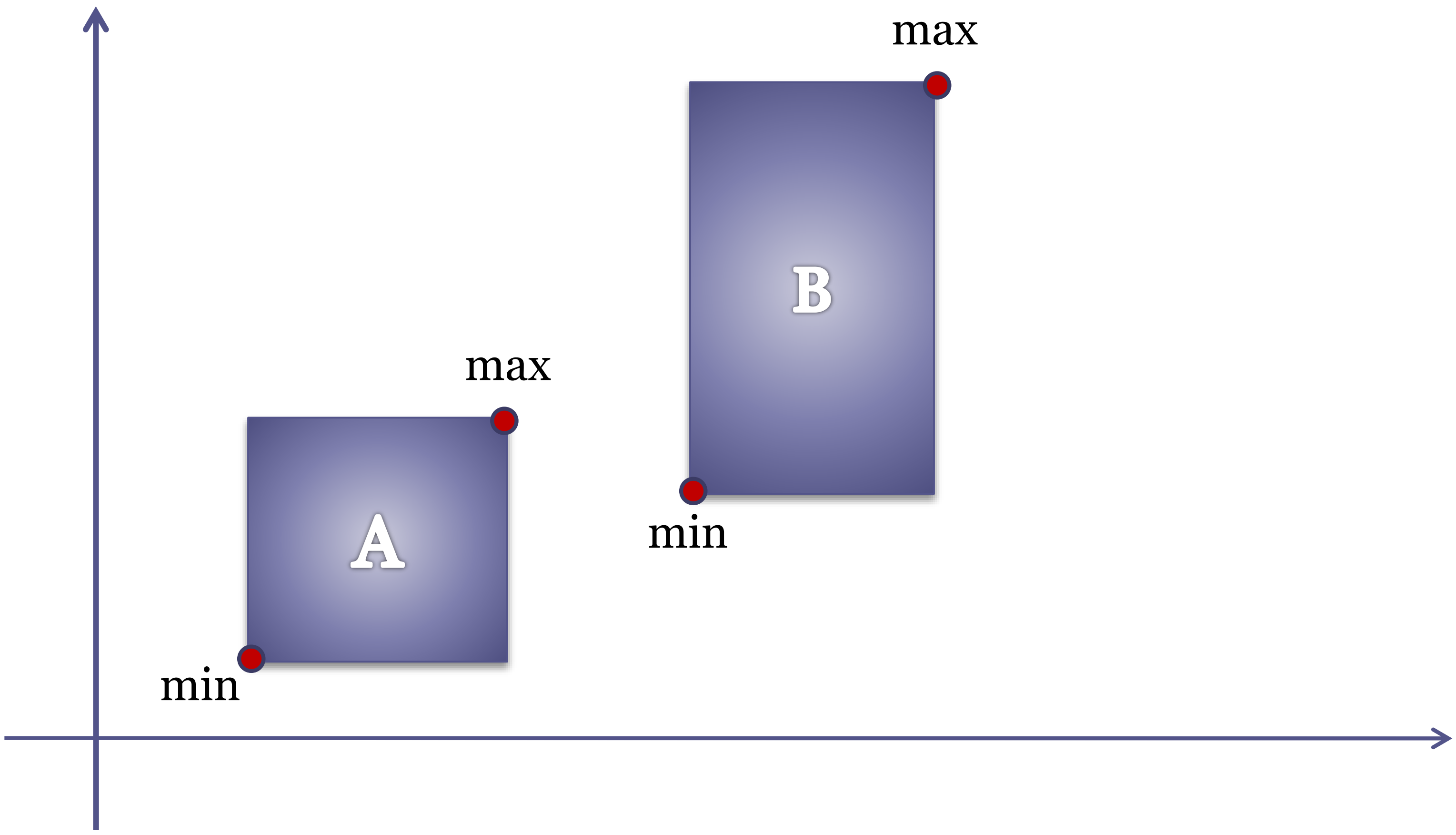
- Most 2D games use a bounding rectangle around the game object.
- This rectangle should be as small as possible, but it must still contain the actual game object.
- A bounding rectangle is defined using 4 values: *top*, *bottom*, *left* and *right*.

Bounding Rectangles (2/2)

- The structure of a bounding rectangle would look like this:

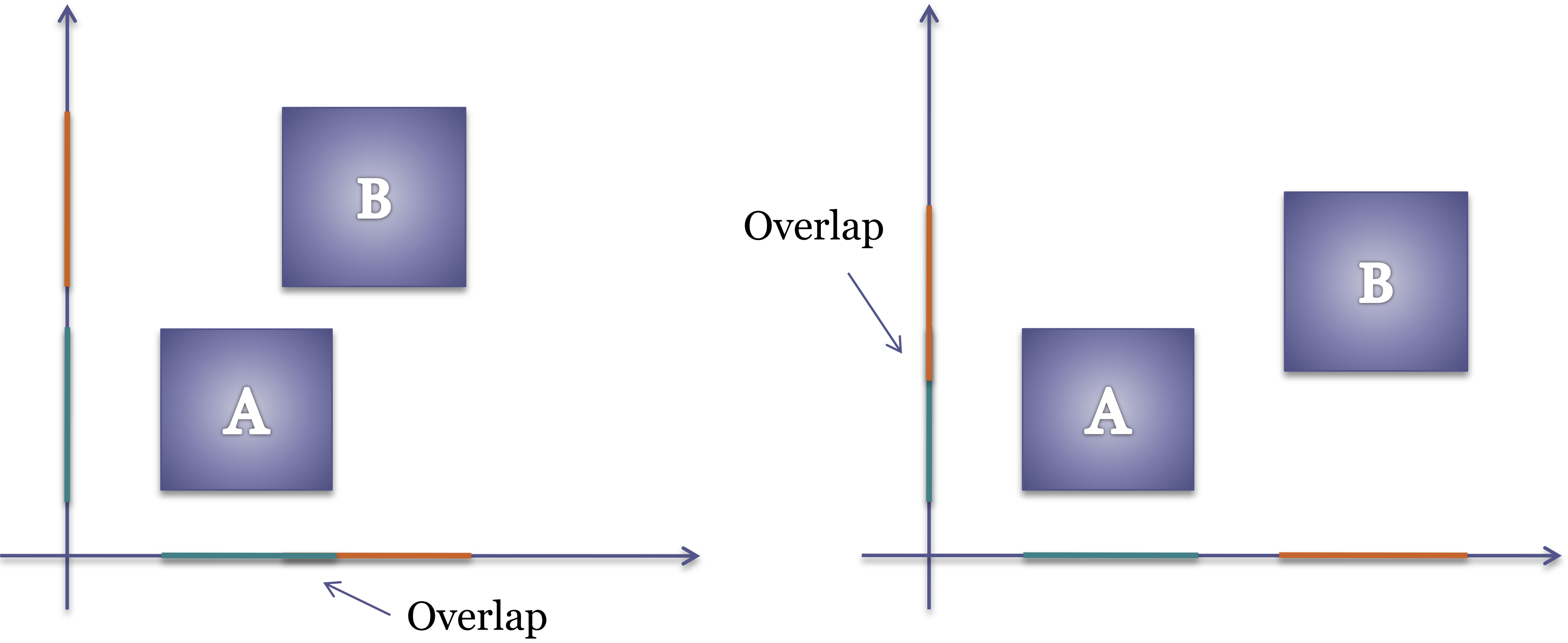
```
struct AABB
{
   Point2   min;
   Point2   max;
} ;
```

Rectangle – Rectangle Collision Detection



For two rectangles to intersect they must overlap on both axes (x-axis and y-axis)

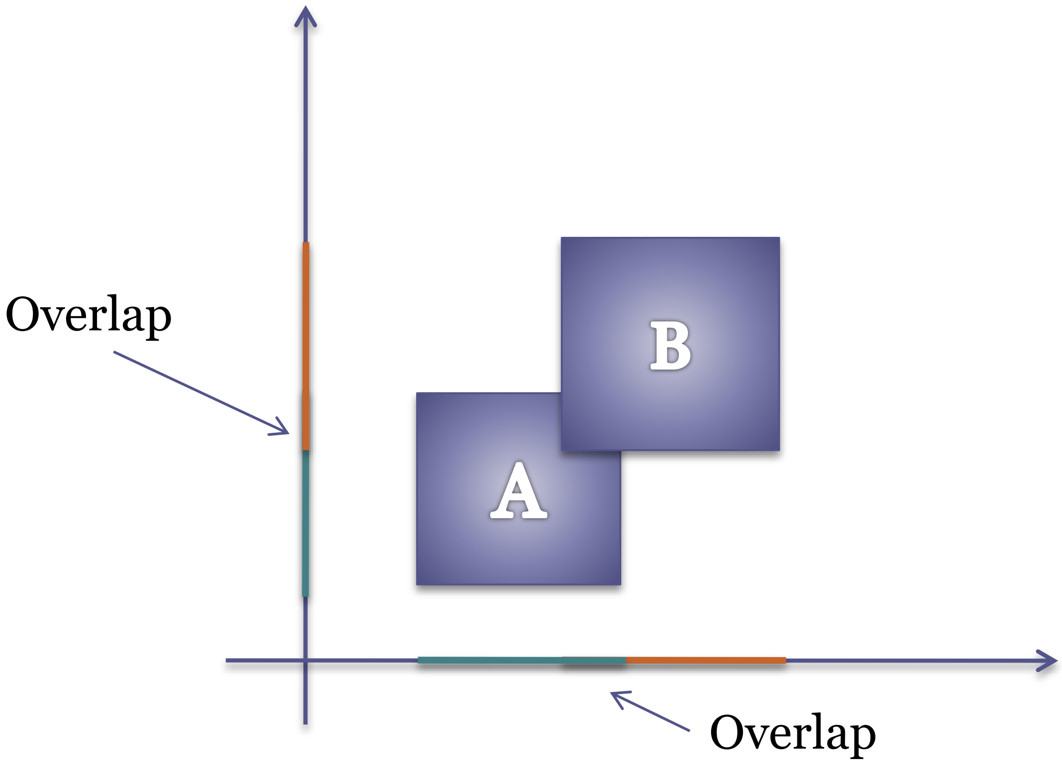
Separation-Axis Theorem (SAT) (1/2)



No Intersection

No Intersection

Separation-Axis Theorem (SAT) (2/2)



Intersection

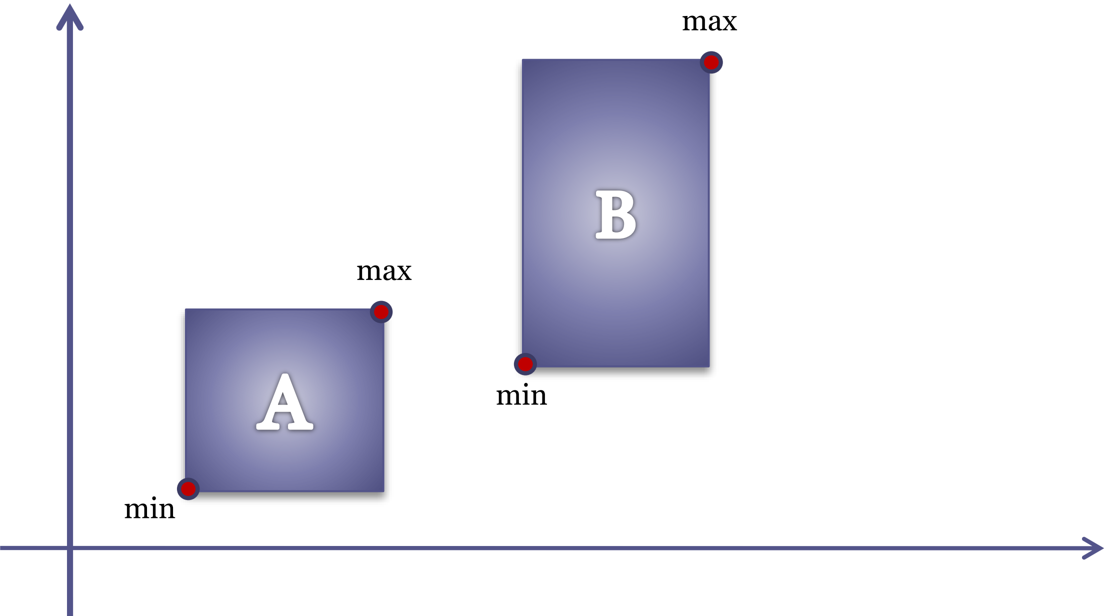
Testing for Intersection

if (A.max < B.min || A.min > B.max)

return 0; //no intersection

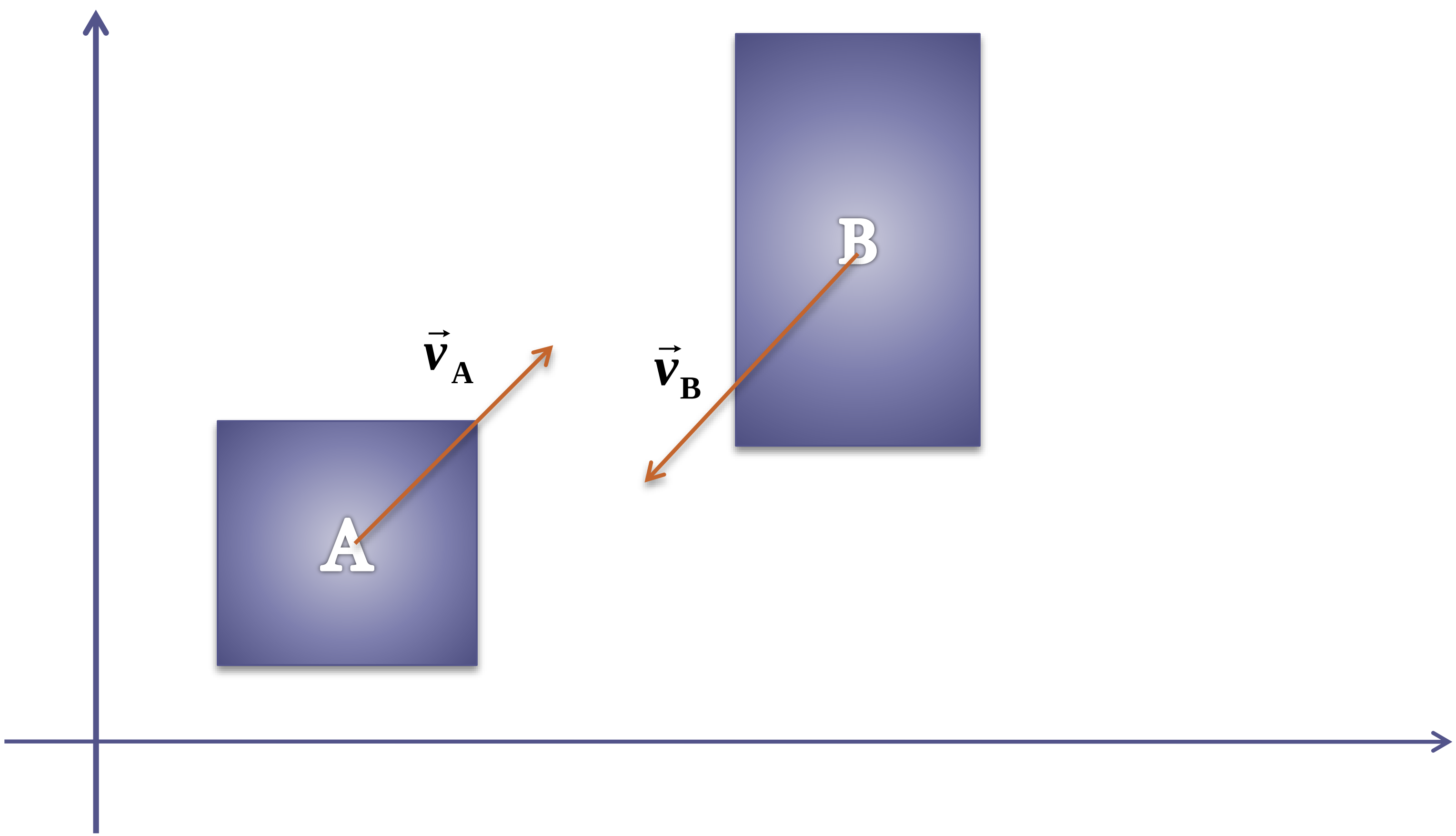
otherwise,

return 1; //overlapping rectangles



But what about moving rectangles?

Testing for Moving Rectangles (1/4)



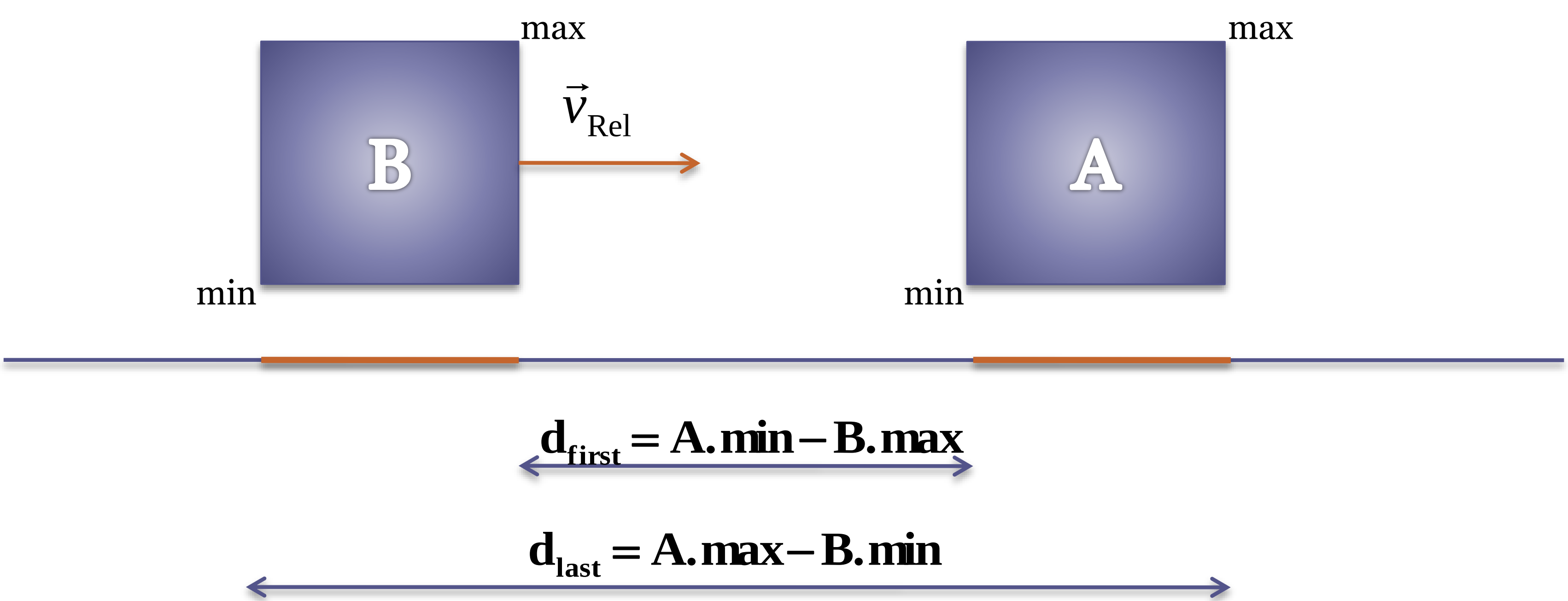
Testing for Moving Rectangles (2/4)

- Solving this problem would be by moving one of the rectangles which in our case is to move rectangle B and have rectangle A stationary.

$$\vec{v}_{\mathrm{B}} : \vec{v}_{\mathrm{B}} - \vec{v}_{\mathrm{A}} = \vec{v}_{\mathrm{Rel}}$$

$$\vec{v}_{\mathrm{A}} : \vec{v}_{\mathrm{A}} - \vec{v}_{\mathrm{A}} = 0$$

Testing for Moving Rectangles (3/4)



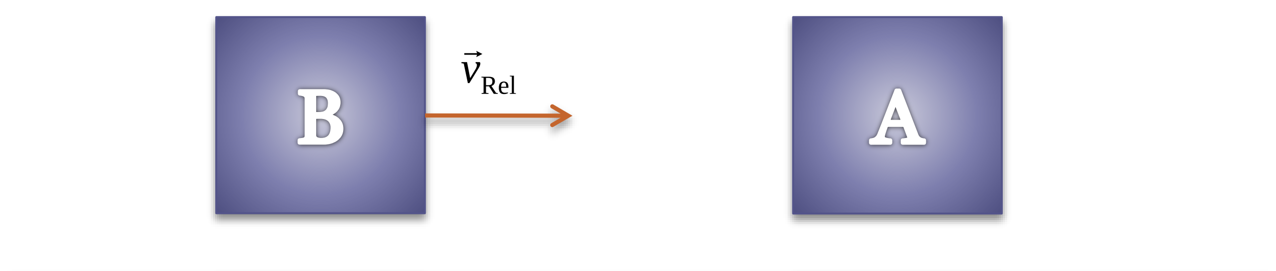
$$d_{first} = A.min - B.max$$

$$d_{last} = A.max - B.min$$

Testing for Moving Rectangles (4/4)

- We have 4 cases to check for:
  - Case 1:

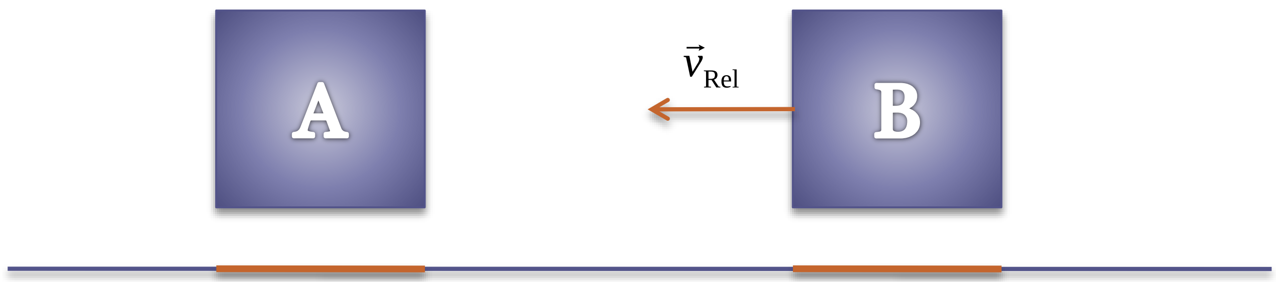
  - Case 2:



Testing for Moving Rectangles (4/4)

- Case 3:

- Case 4:



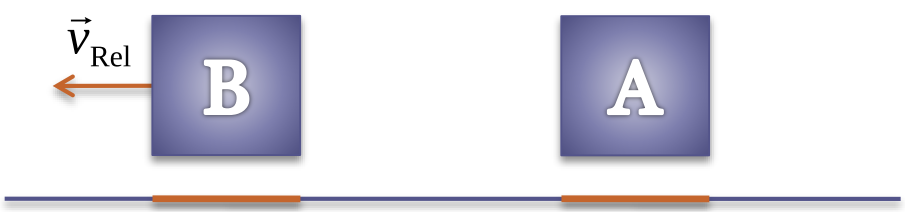
Case 1

- Dealing with one dimension (x-axis illustrated)

$$\text{if } (v_{\text{Rel}} < 0)$$

$$\quad \text{if } (A.\min > B.\max)$$

$$\quad\quad \text{return } 0; \text{ // No intersection and B is moving away}$$

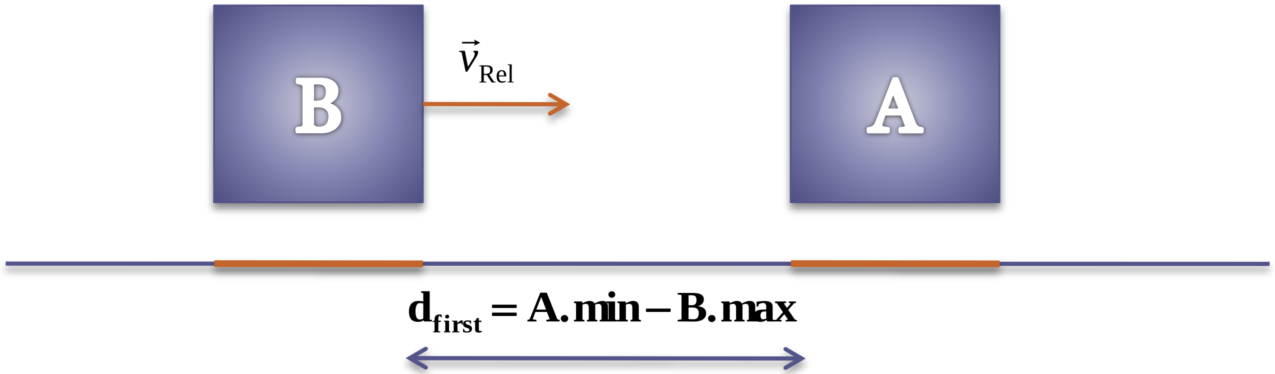


Case 2 (1/2)

$$\text{if } (v_{\text{Rel}} > 0)$$

$$\text{if } (\text{A.min} > \text{B.max})$$

$$\text{t}_{\text{First}} = \frac{\text{d}_{\text{First}}}{v_{\text{Rel}}} = \frac{\text{A.min} - \text{B.max}}{v_{\text{Rel}}}$$

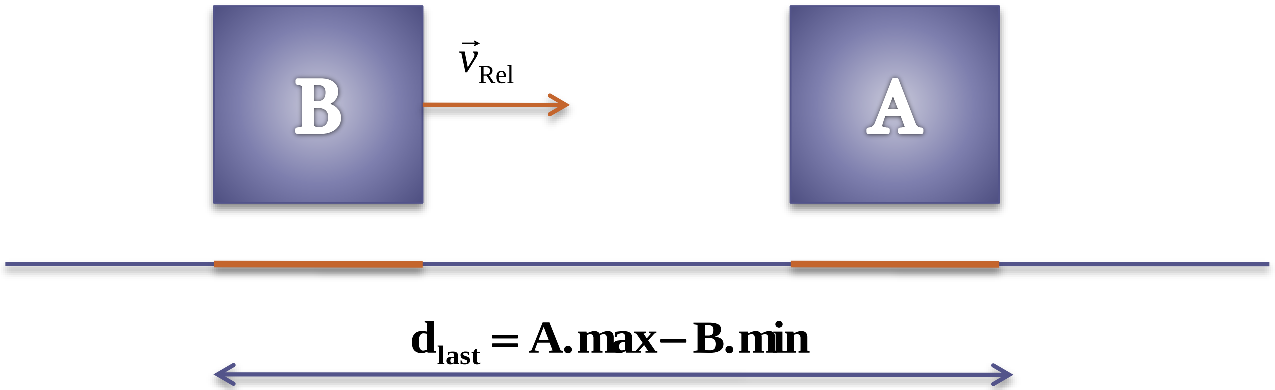


Case 2 (2/2)

$$\text{if } (v_{\text{Rel}} > 0)$$

$$\text{if } (\text{A.max} > \text{B.min})$$

$$\text{t}_{\text{Last}} = \frac{\text{d}_{\text{Last}}}{v_{\text{Rel}}} = \frac{\text{A.max} - \text{B.min}}{v_{\text{Rel}}}$$



Case 3

$$\text{if } (v_{\text{Rel}} > 0)$$

$$\quad \text{if } (\text{A.max} < \text{B.min})$$

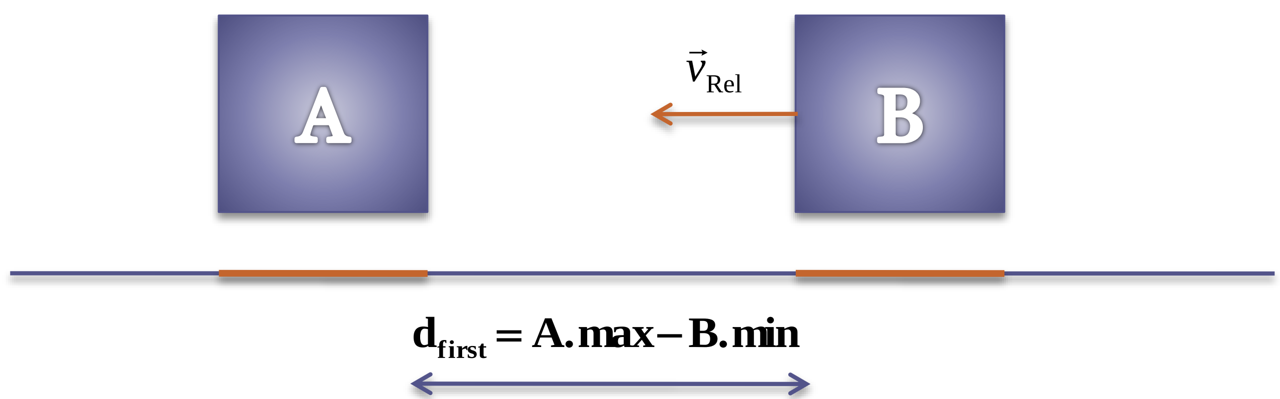
$$\qquad \text{return 0; // No intersection and B is moving away}$$

Case 4 (1/2)

$$\text{if } (v_{\text{Rel}} < 0)$$

$$\text{if } (\text{A.max} < \text{B.min})$$

$$t_{\text{First}} = \frac{d_{\text{First}}}{v_{\text{Rel}}} = \frac{\text{A.max} - \text{B.min}}{v_{\text{Rel}}}$$

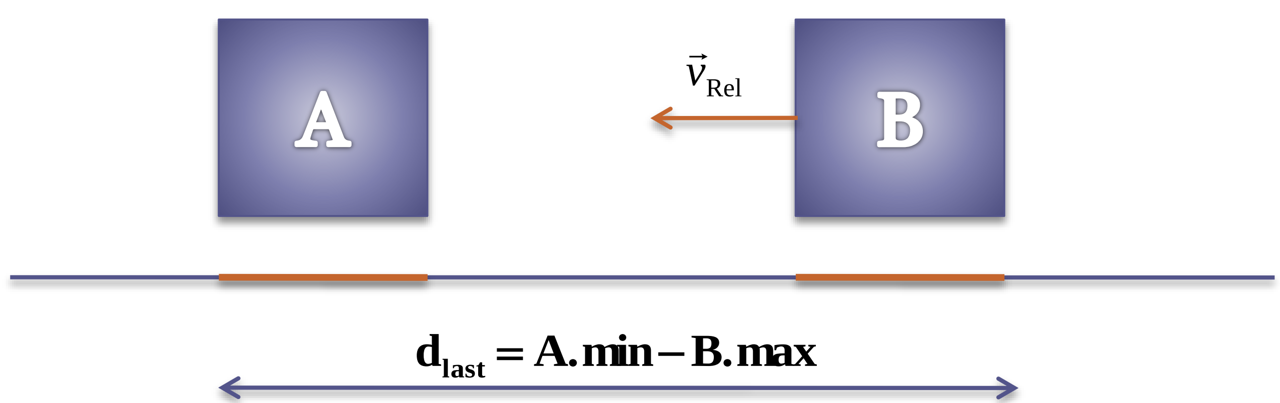


Case 4 (2/2)

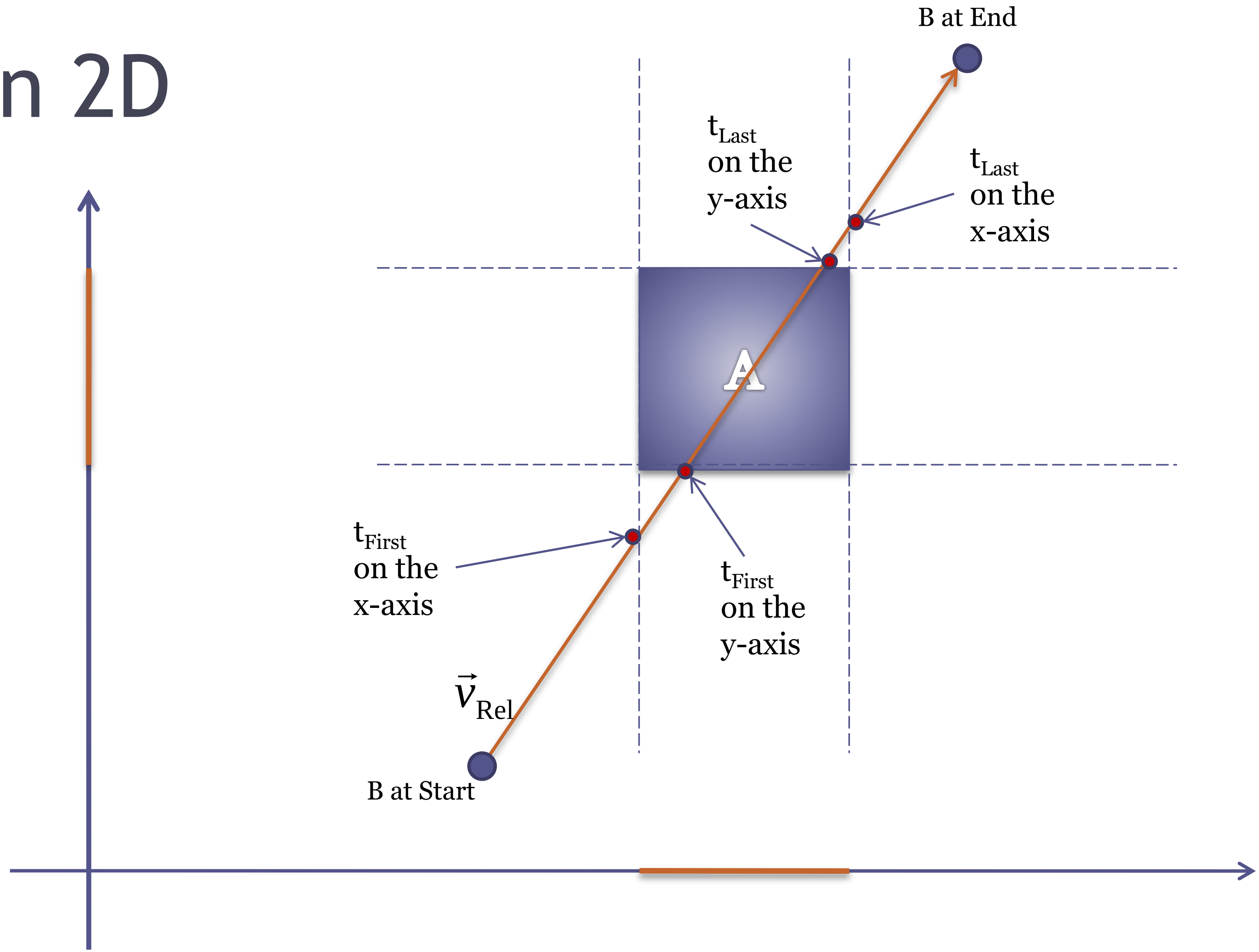
$$\text{if } (v_{\text{Rel}} < 0)$$

$$\text{if } (\text{A.min} < \text{B.max})$$

$$t_{\text{Last}} = \frac{d_{\text{Last}}}{v_{\text{Rel}}} = \frac{\text{A.min} - \text{B.max}}{v_{\text{Rel}}}$$



In 2D

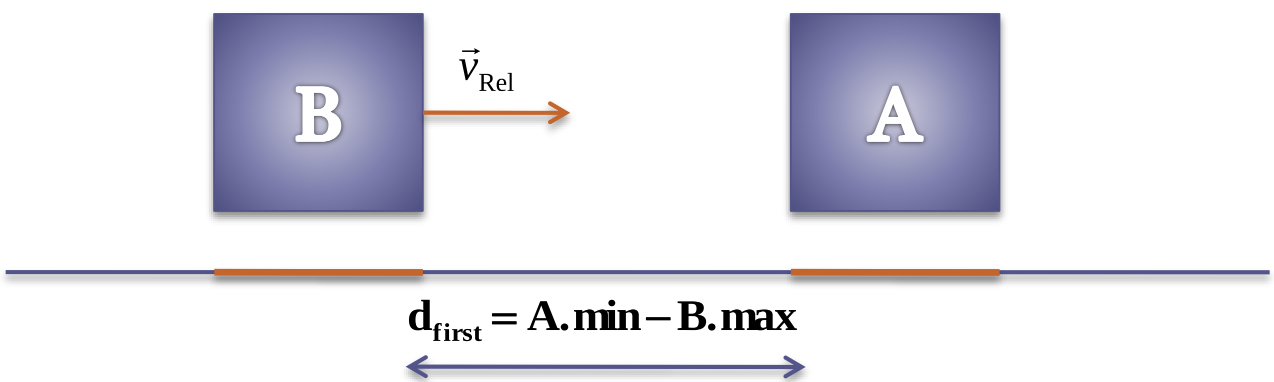

B at End
t_{Last} on the y-axis
t_{Last} on the x-axis
A
t_{First} on the x-axis
t_{First} on the y-axis
$\vec{V}_{Rel}$
B at Start

Case 2: Revisited (1/2)

$$\text{if } (v_{\text{Rel}} > 0)$$

$$\text{if } (\text{A.min} > \text{B.max})$$

$$\text{t}_{\text{First}} = \max\left(\frac{\text{d}_{\text{First}}}{v_{\text{Rel}}}, \text{t}_{\text{First}}\right) = \max\left(\frac{\text{A.min} - \text{B.max}}{v_{\text{Rel}}}, \text{t}_{\text{First}}\right)$$

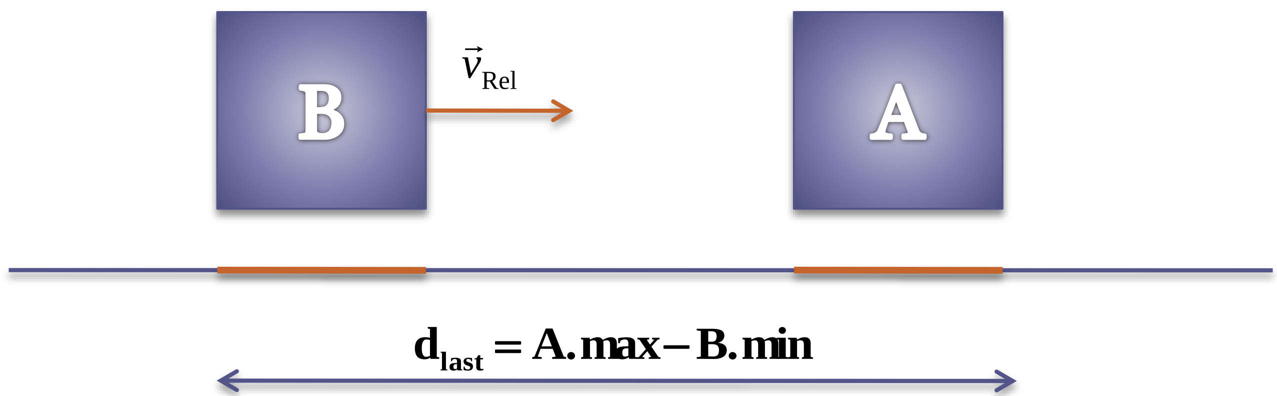


Case 2: Revisited (2/2)

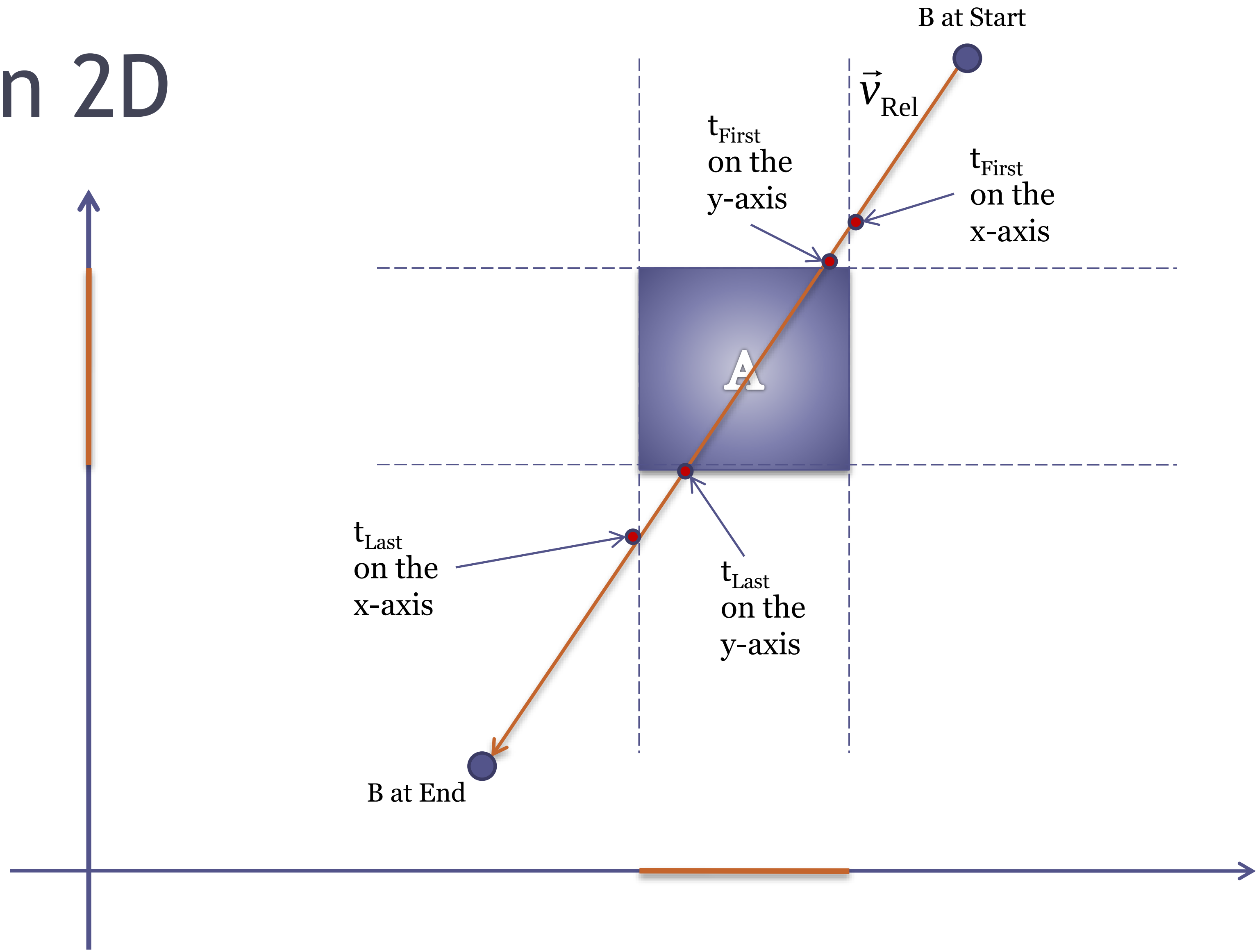
$$\text{if } (v_{\text{Rel}} > 0)$$

$$\text{if } (\text{A.max} > \text{B.min})$$

$$\text{t}_{\text{Last}} = \min\left(\frac{\text{d}_{\text{Last}}}{v_{\text{Rel}}}, \text{t}_{\text{Last}}\right) = \min\left(\frac{\text{A.max} - \text{B.min}}{v_{\text{Rel}}}, \text{t}_{\text{Last}}\right)$$



In 2D

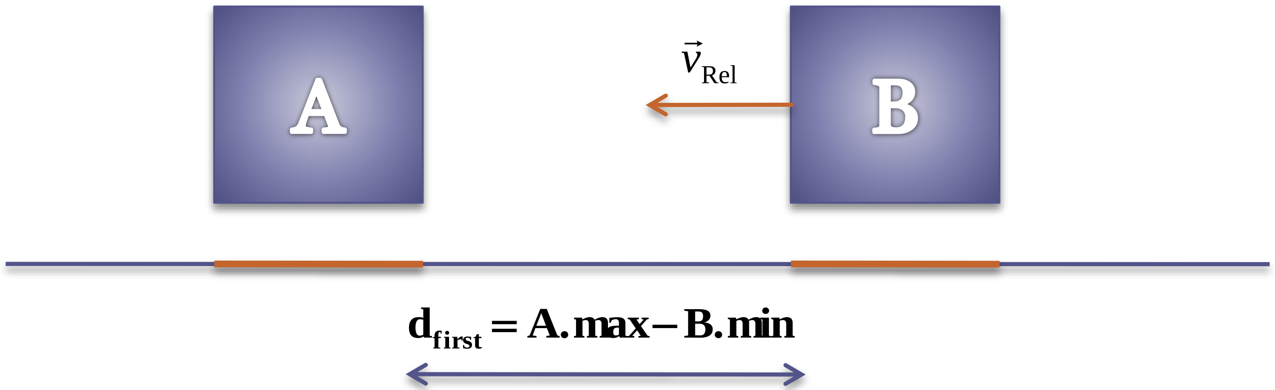

B at Start
$\vec{v}_{Rel}$
t_{First} on the y-axis
t_{First} on the x-axis
A
t_{Last} on the x-axis
t_{Last} on the y-axis
B at End

Case 4: Revisited (1/2)

$$\text{if } (v_{\text{Rel}} < 0)$$

$$\text{if } (\text{A.max} < \text{B.min})$$

$$\text{t}_{\text{First}} = \max\left(\frac{\text{d}_{\text{First}}}{v_{\text{Rel}}}, \text{t}_{\text{First}}\right) = \max\left(\frac{\text{A.max} - \text{B.min}}{v_{\text{Rel}}}, \text{t}_{\text{First}}\right)$$

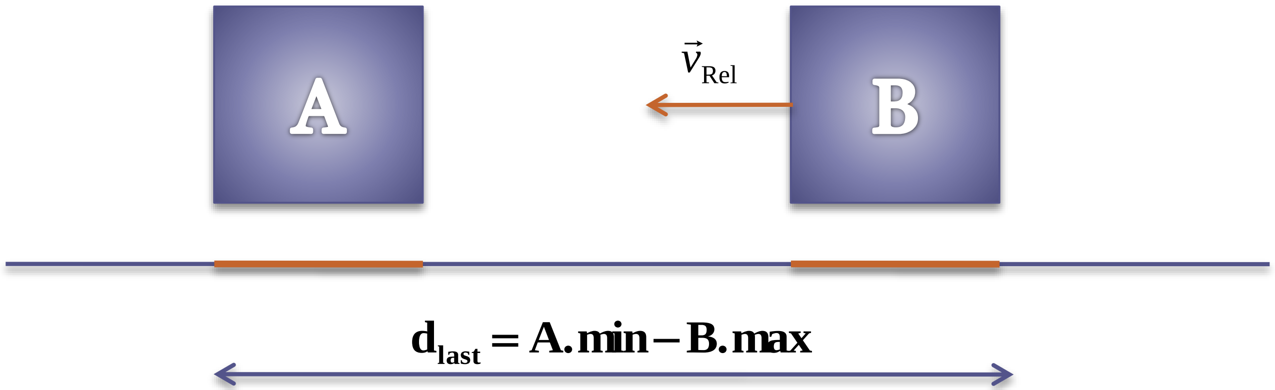


Case 4: Revisited (2/2)

$$\text{if } (v_{\text{Rel}} < 0)$$

$$\text{if } (\text{A.min} < \text{B.max})$$

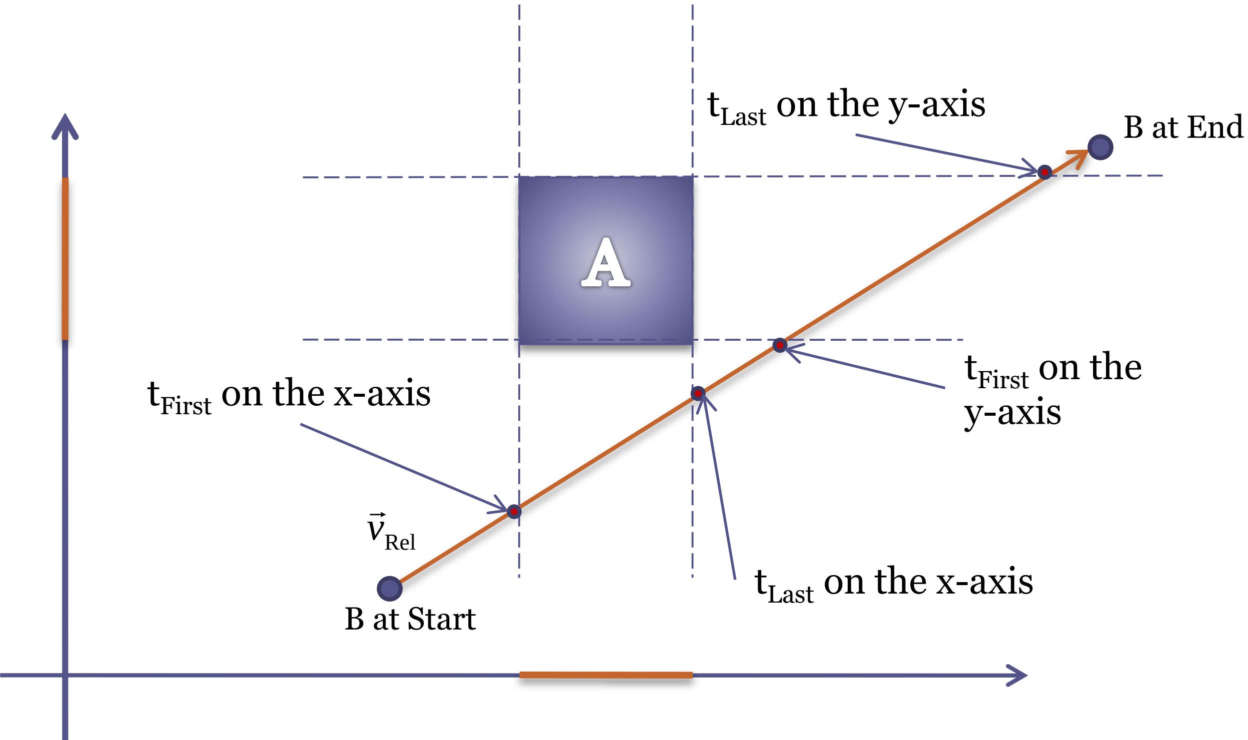
$$t_{\text{Last}} = \min\left(\frac{d_{\text{Last}}}{v_{\text{Rel}}}, t_{\text{Last}}\right) = \min\left(\frac{\text{A.min} - \text{B.max}}{v_{\text{Rel}}}, t_{\text{Last}}\right)$$



Case 5: No Intersection

if ($t_{First} > t_{Last}$)

return 0;



Pseudo-Code (1/2)

- Step 1: Check for collision detection between static rectangles. If check returns no overlap you continue with the following steps
- Step 2: Initialize and calculate the new velocity of v_{Rel}

  $$t_{First} = 0$$

  $$t_{\text{Last}} = dt$$

- Step 3: Working with one dimension (x-axis)

if $(v_{\text{Rel}} < 0)$

Case 1

Case 4 - Revisited

if $(v_{\text{Rel}} > 0)$

Case 2 - Revisited

Case 3

Case 5

Pseudo-Code (2/2)

- Step 4: Repeat step 3 on the y-axis
- Step 5: Otherwise the rectangles intersect

- Remark:
  - In Step1, we checked for static rectangles collision first. This means before checking for the collision with respective velocity movement, we must check the static intersection right before they move. Maybe they are already intersected, before movement!

References

- Real Time Collision Detection by Christer Ericson