Game Physics

Game Physics

Game Physics ≠ Physics

Game Physics is the illusion of Physics.

Topics in Game Physics

- Review of kinematics
- Review of linear algebra
- Game physics
- Programming a simple game physics engine
  - Rigid body motion
  - Collision
  - Collision detection

Topics in Game Physics

- Review of kinematics
- Review of linear algebra
- Game physics
- Programming a simple game physics engine
  - Rigid body motion
  - Collision
  - Collision detection

Kinematics

Study of the geometry of motion.

Newton's First Law of Motion

Objects at rest
stay at rest

and

objects in motion
stay in motion

unless they are acted
upon by some force.

Motion

Physical objects are called **bodies**.

Each body has **mass** that gives it weight.

Each body has a **position** in space.

Speed is distance over time (magnitude).

Velocity is distance in some direction over time (direction and magnitude).

Acceleration is the rate velocity changes.

Motion

Bodies have **position** and **mass**.

Velocity is change in position over time.

Acceleration is change in velocity over time.

Kinematics Example

You are in 1995 Ferrari GT, driving due East and pressing down on the gas petal as hard as you can to go from 0 to 60 mph in 5 seconds.

At first your speedometer reads 10 miles per hour. This is your **speed**.

Your **velocity** starts at 0 miles per hour East.

After 5 seconds, it is 60 miles per hour East.

Your **acceleration** is 60 miles per hour per 5 seconds, or 0.003 miles per second squared.

How does a body move over time?

We need to know it's position and velocity.

Remember that velocity is change in position over time.

At each moment of time, we add its velocity to its position to get its new position.

What causes an object to move? Force.

Newton's Second Law of Motion

$$F = ma$$

Force is mass times acceleration.

Earth's Gravity

The gravitational pull of earth is a force.

It accelerates objects at 9.8 meters per second per second.

(or 9.8 meters per second squared)

Friction

Friction is the force that causes surfaces to resist one another's movement.

Friction converts motion to heat.

More on this in a moment.

Newton's Third Law of Motion

For every action there is an equal but opposite reaction.

Collision

When moving objects collide, they change direction.

If there were no friction, they would continue moving around forever (**elastic collisions**).

When there is friction, some motion is converted to heat, meaning that objects eventually stop (**inelastic collisions**).

Restitution (or bounciness) determines how much motion is lost during a collision.

Topics in Game Physics

- Review of kinematics
- Review of linear algebra
- Game physics
- Programming a simple game physics engine
  - Rigid body motion
  - Collision
  - Collision detection

Review of Linear Algebra

The study of vector spaces in an arbitrary number of dimensions.

A vector is an ordered list of numbers, one per dimension of the space.

1D vector = (x)

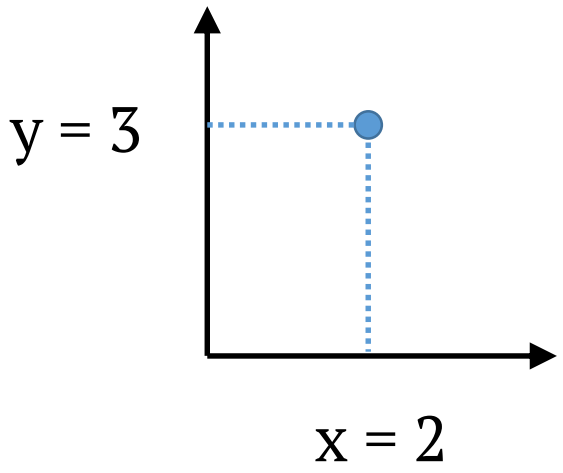
2D vector = (x,y)

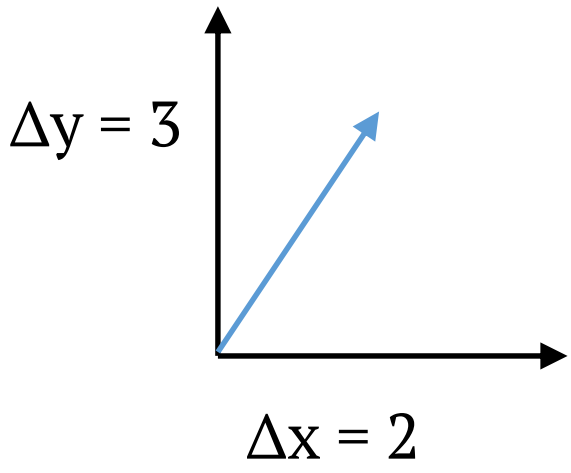
3D vector = (x,y,z)

Vector Representation

A vector is useful for representing:

- a point in space (i.e. position)
- the direction and magnitude of movement (i.e. velocity)

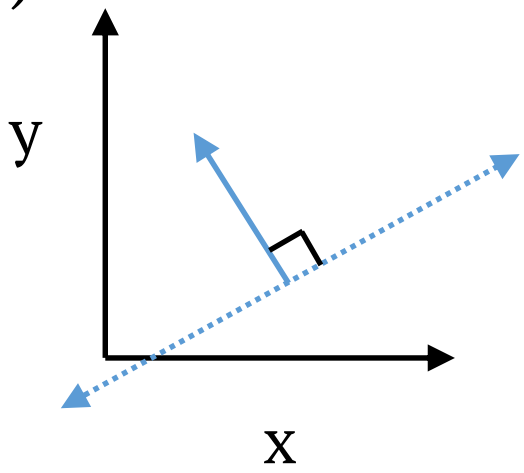




Vector Representation

A vector is useful for representing:

- a point in space (i.e. position)
- the direction and magnitude of movement (i.e. velocity)
- a plane (i.e. normal vector)



Basic 2D Vector Math

- Adding / subtracting 2 vectors

  $(2,4) + (3,5) = (5,9)$

- Multiplying / dividing vectors by a scalar

  $(2,4) \cdot 2 = (4,8)$

- Dot product of 2 vectors

  $(2,4) \cdot (3,5) = (2 \cdot 3 + 4 \cdot 5) = 26$

  Projection of one vector onto another

Topics in Game Physics

- Review of kinematics
- Review of linear algebra
- Game physics
- Programming a simple game physics engine
  - Rigid body motion
  - Collision
  - Collision detection

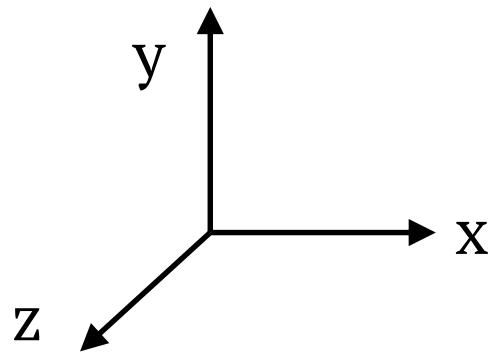
Bodies in Game Physics

- **Rigid Bodies** cannot be deformed (i.e. cannot change shape or be bent). Examples include boxes, walls, etc.
- **Soft Bodies** can be deformed. Examples include rubber balls and organic matter like muscles. They are usually simulated as:
  1. Groups of connected rigid bodies
  2. Particles connected by springs
- **Particle Systems** generate many small, fuzzy elements. Examples include smoke and fire.
- **Fluids** continually deform to fit a container. They are usually simulated as particles.

Degrees of Freedom

Degrees of freedom are the parameters needed to define the state of an object.

- A 3D engine has 6 degrees of freedom:
  - Position on the X, Y, and Z axes
  - Rotation around X, Y, and Z (or pitch, yaw, and roll)
- A 2D engine has 3 degrees of freedom:
  - Position on the X and Y axes
  - Rotation around the Z axis



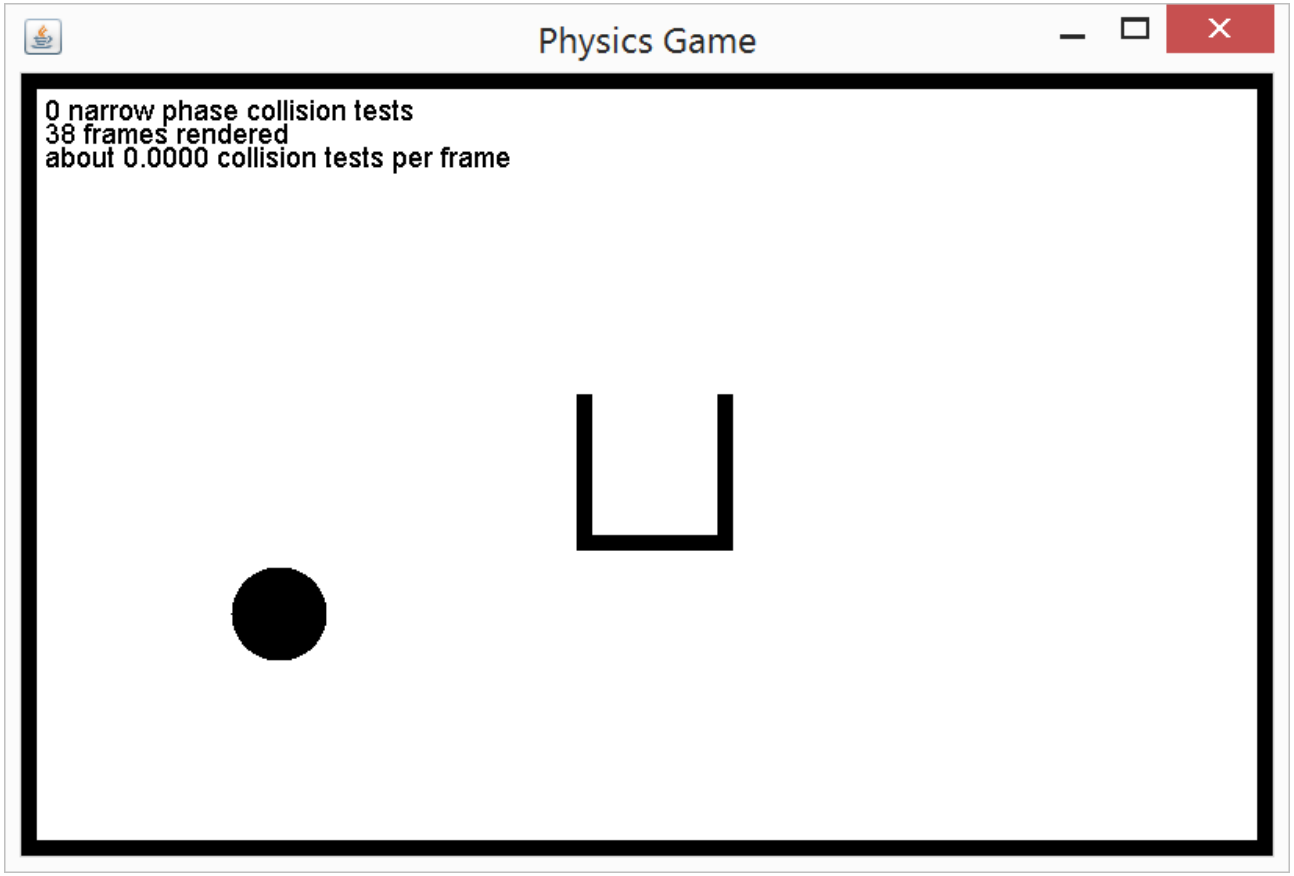
Time and Motion

- Time (and thus motion) are usually discretized.
- Most modern video games render between 30 and 60 frames per second, which means a delay of about 16 milliseconds between each frame.
- Discrete time steps make it is hard to predict the future and can lead to some strange bugs (e.g. tunneling).
- Games need to perform the same at different frame rates, so most engines can adapt and drop frames when CPU/GPU load spikes.

Topics in Game Physics

- Review of kinematics
- Review of linear algebra
- Game physics
- Programming a simple game physics engine
  - Rigid body motion
  - Collision
  - Collision detection

Simple Physics Game



Simple Physics Game

- 2 kinds of rigid bodies: circles and rectangles
- Runs at a constant 60 frames per second.
- 2 degrees of freedom (no rotation)
  (everything is “axis-aligned”)
- Sweep and Prune collision detection

Step 1: Simulating All Bodies

High level algorithm:

1. Adjust the position of each body.
2. Check for and process collisions.

Step 2: Rigid Body Motion

Explicit Euler Integration:

- Adjust each body's position by its velocity.
- Calculate the net force (sum of all forces) at work on the body. We consider only gravity.
- Calculate the body's new acceleration.
- Adjust the body's velocity by its acceleration.

Stationary Bodies

Bodies with infinite mass are not affected by collisions.

It is convenient to represent stationary bodies as bodies with a velocity of (0,0) and 0 mass. This will ensure that calculations affecting position and velocity of these bodies does not move them.

However, be careful of divide by zero errors!

Notes about Integration

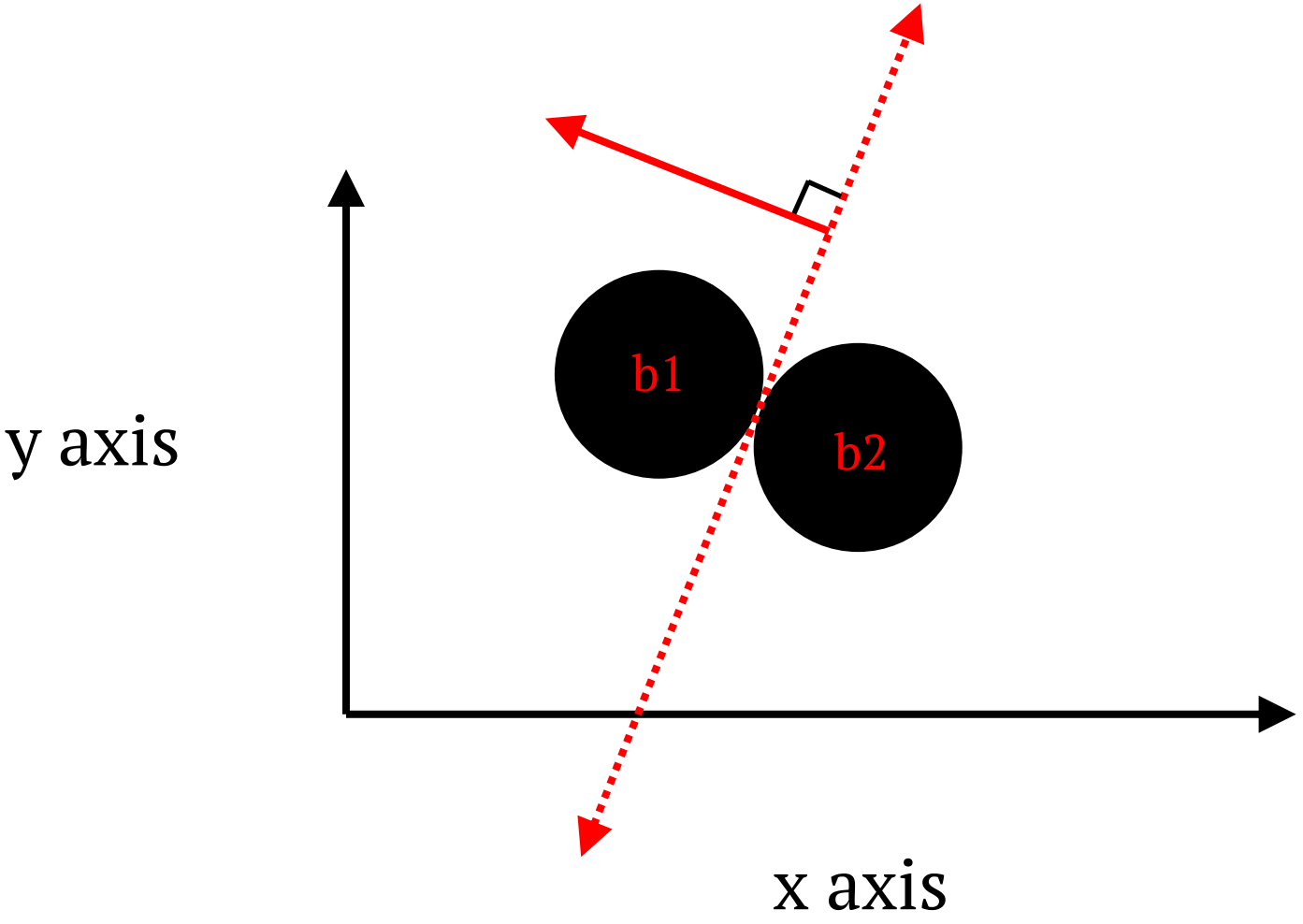
- This discretized calculation of a continuous curve is called **numerical integration**.
- Over time these incremental changes to acceleration will become more and more inaccurate.
- There are other, more accurate method of integration which are used on more advanced engines, including:
  - Implicit Euler
  - Verlet
  - RK4

Step 3: Prepare for Collision

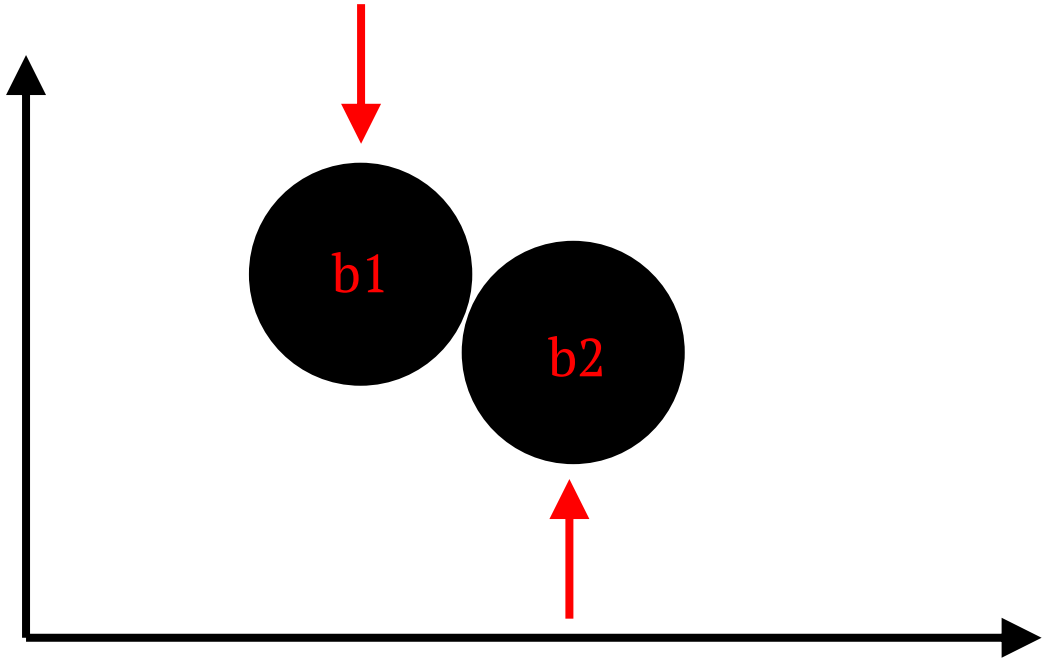
Given two bodies:

- Calculate the plane along which they collide (expressed as a normal vector)
- Calculate velocity along the collision normal

Collision Normal



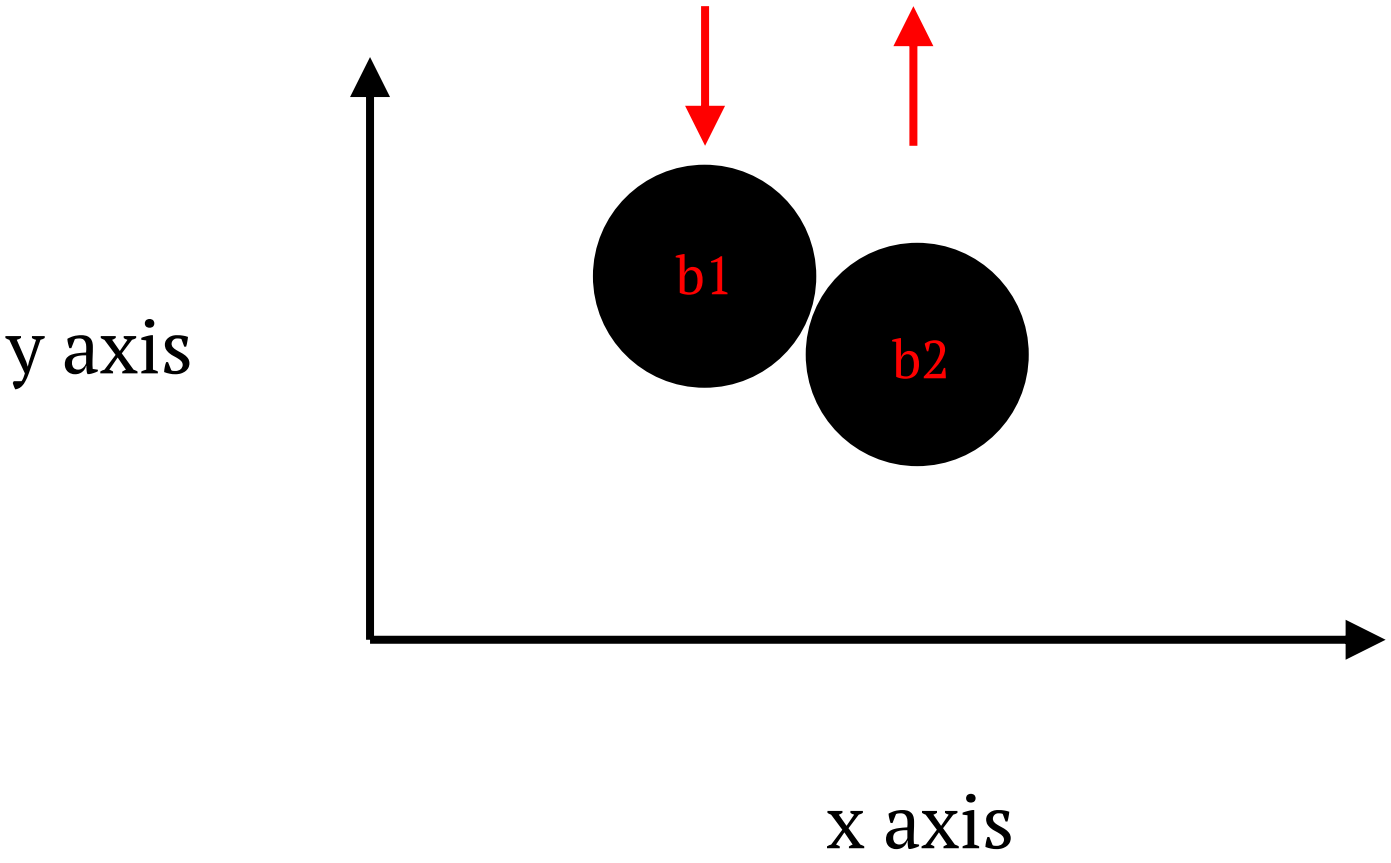
Relative Velocity



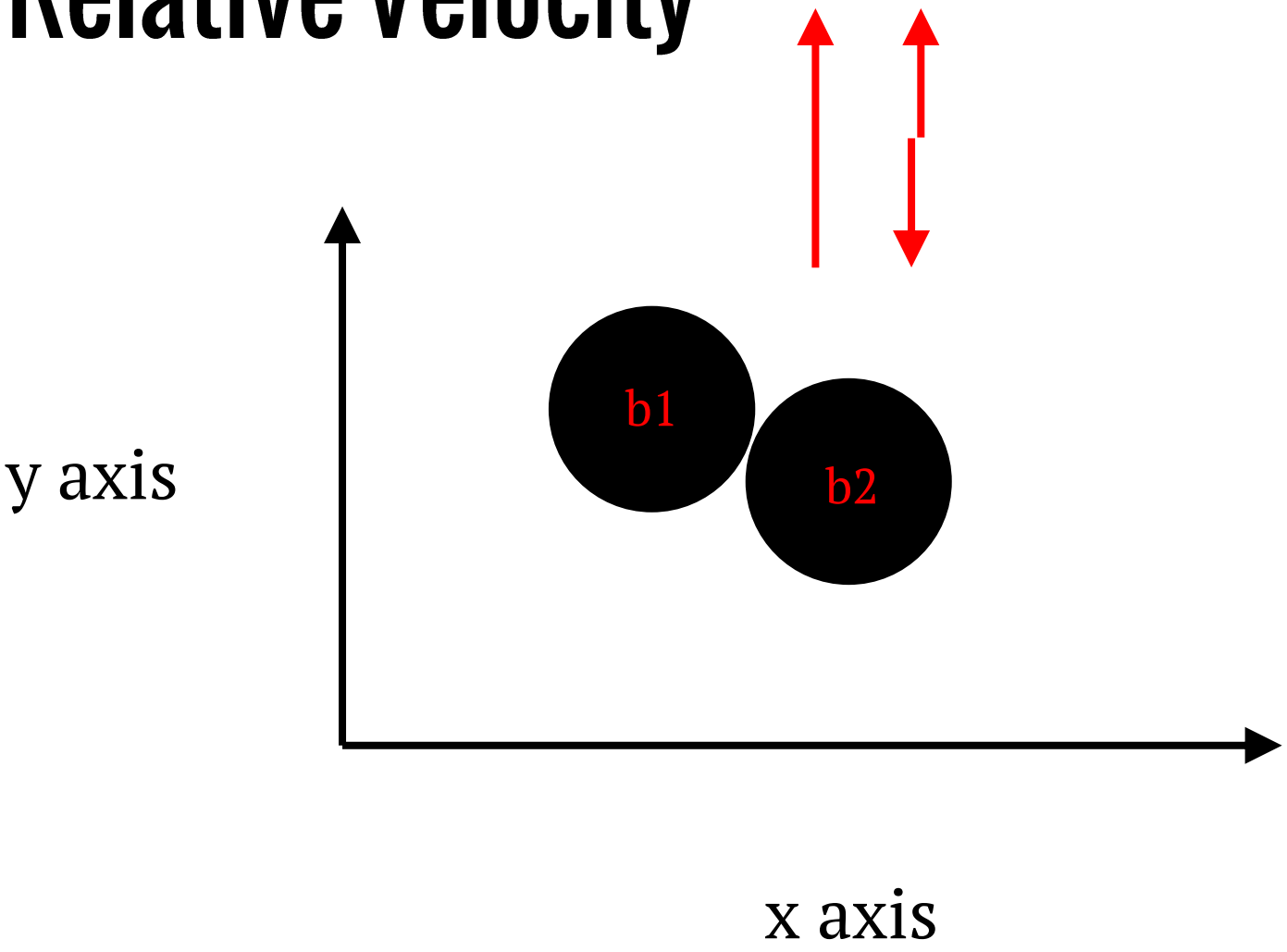
y axis

x axis

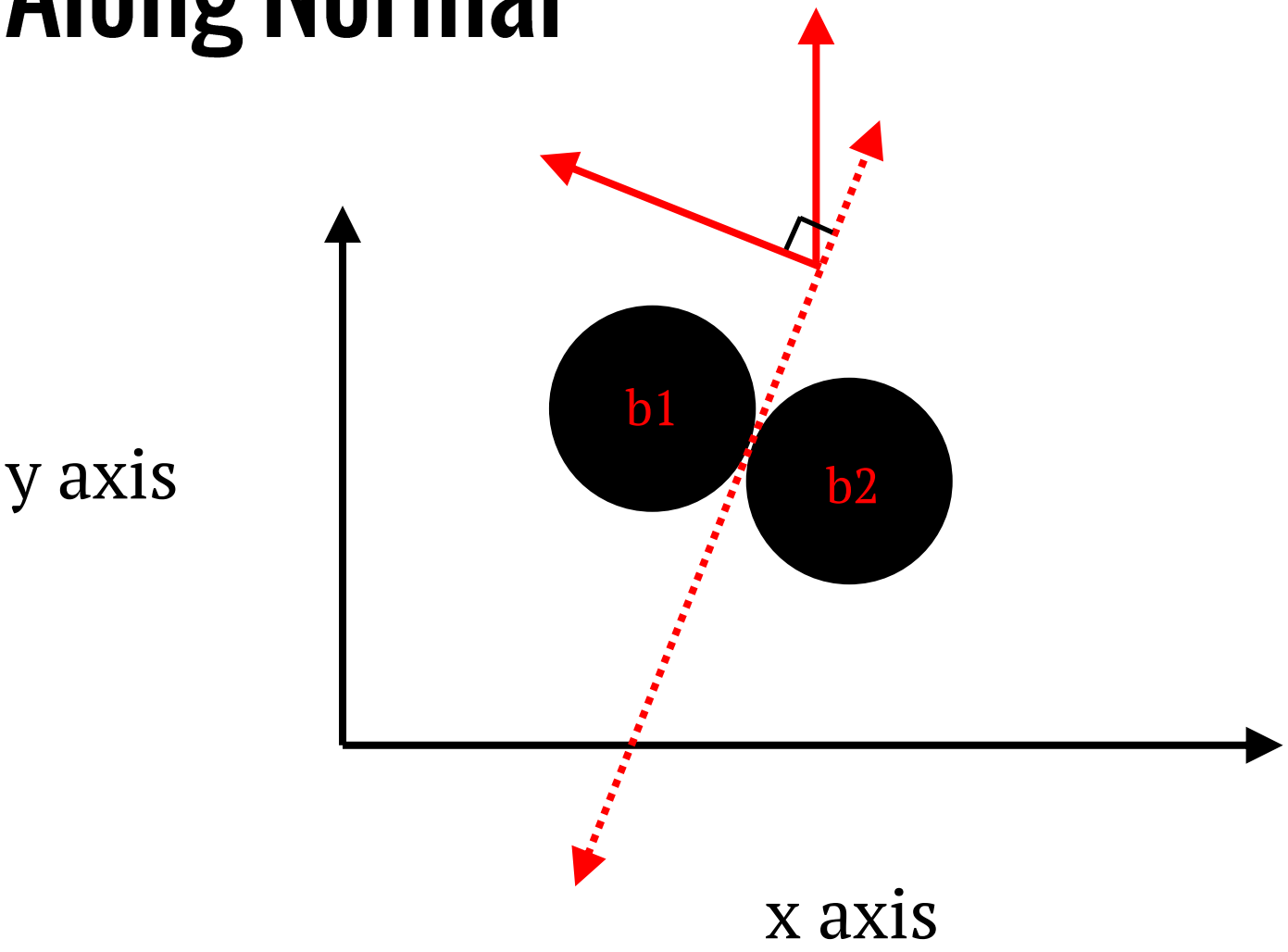
Relative Velocity



Relative Velocity



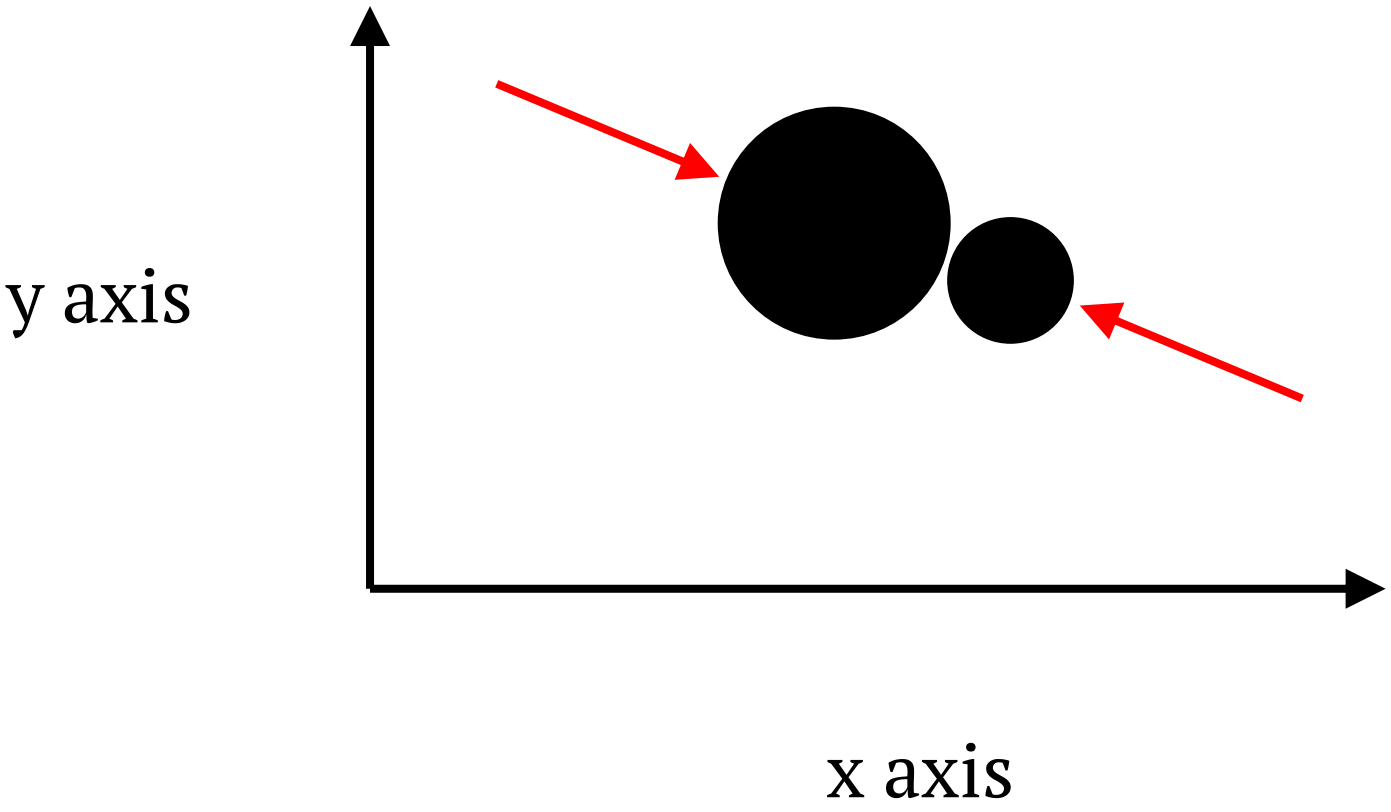
Along Normal



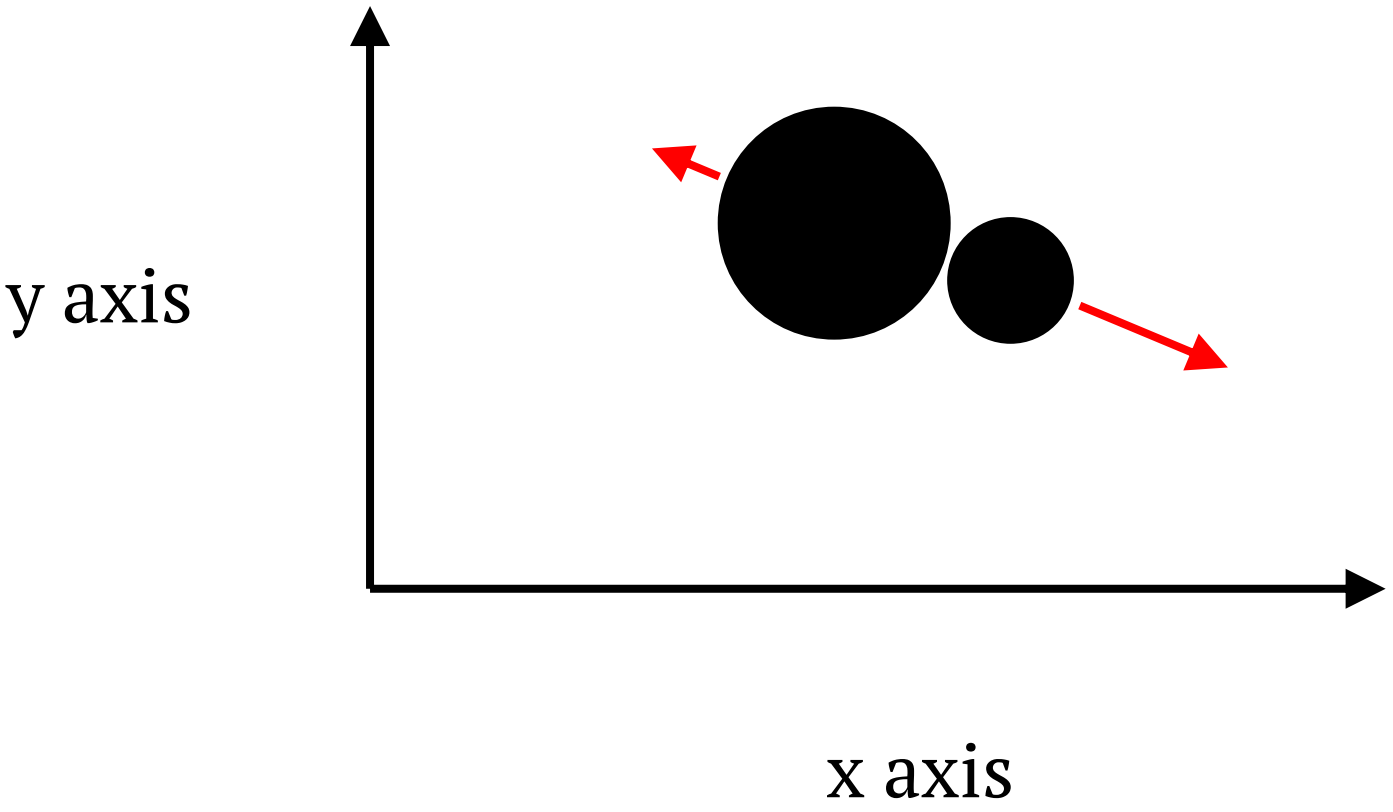
Step 4: Change Direction

- An **impulse** is an instantaneous change in velocity.
- Calculate impulse scalar.
- Calculate the impulse for each body (scaled to their respective masses).
- Apply impulses.

Relative Mass



Relative Mass

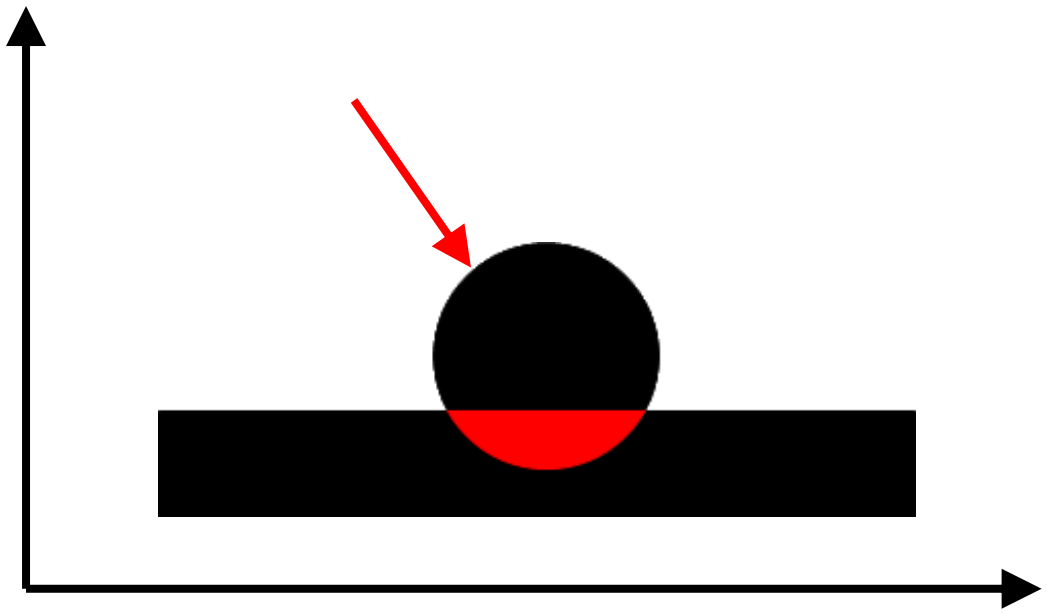


Step 5: Position Correction

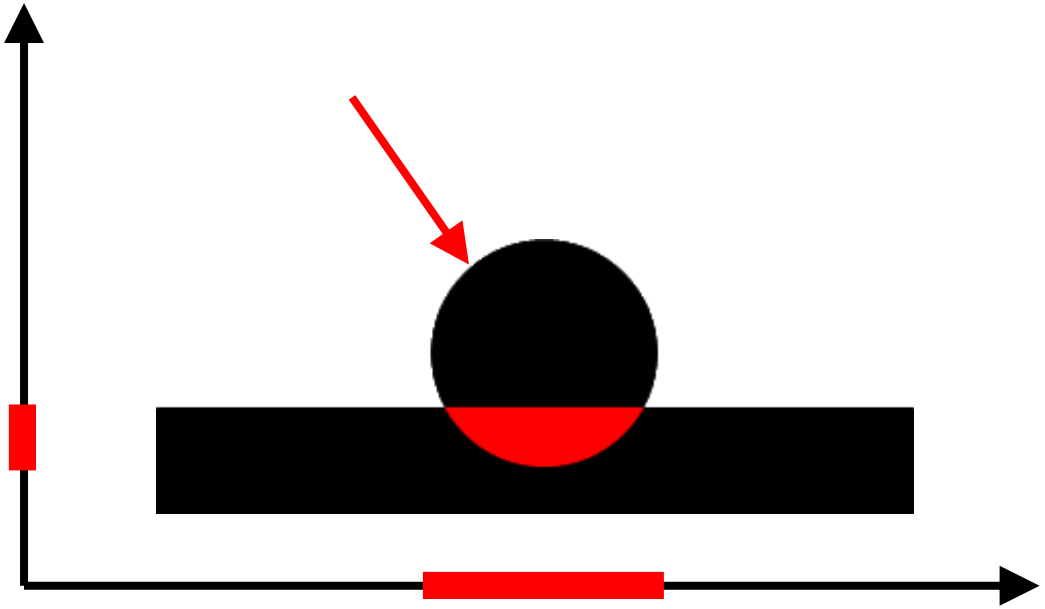
(Before updating velocity)

- Move bodies back the way they came until they no longer overlap.
- Bodies should move proportionately to their mass.

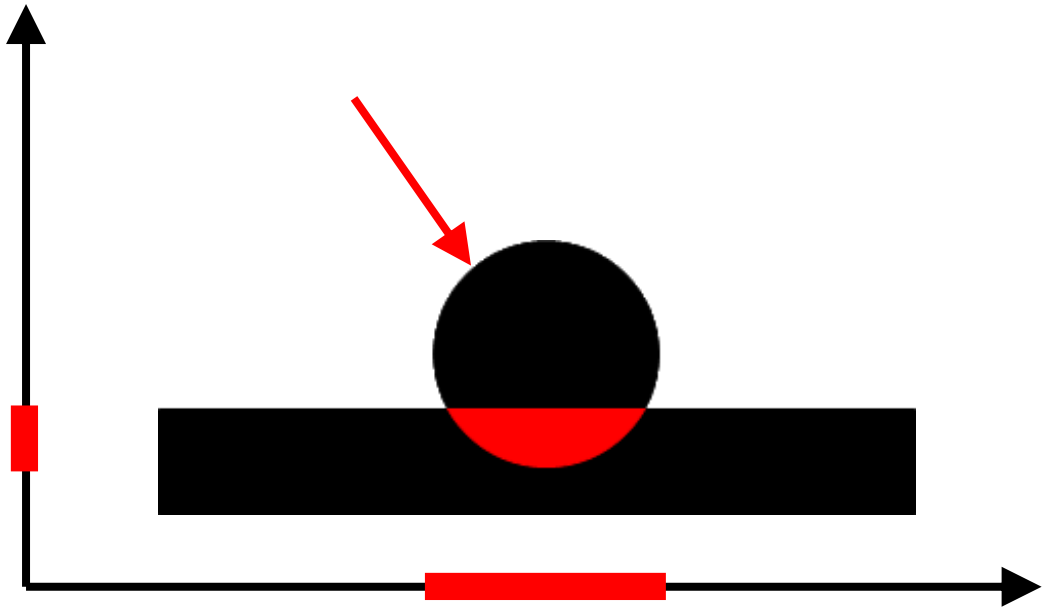
Penetration



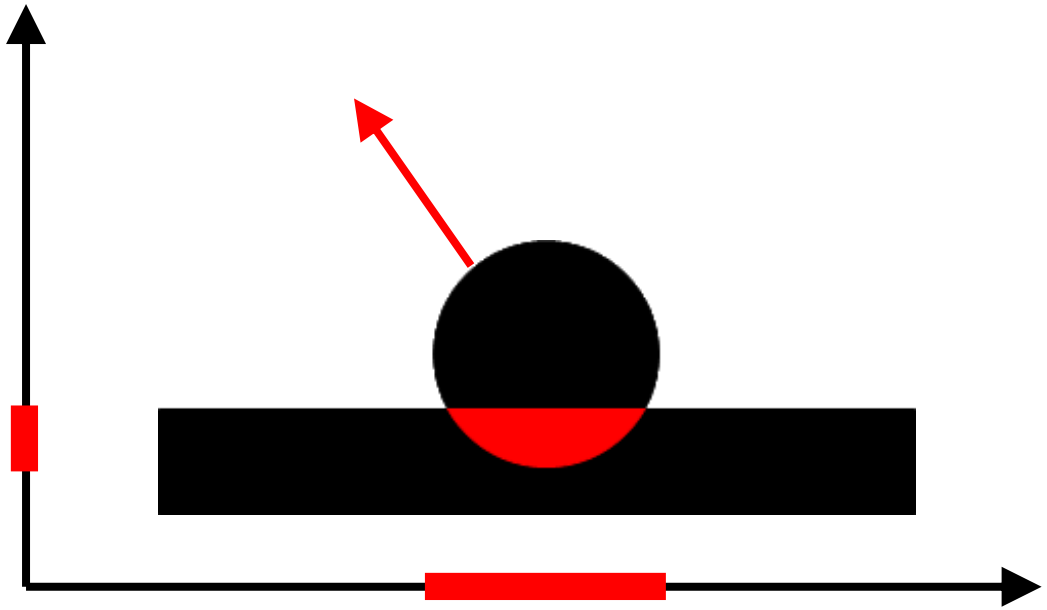
Overlap



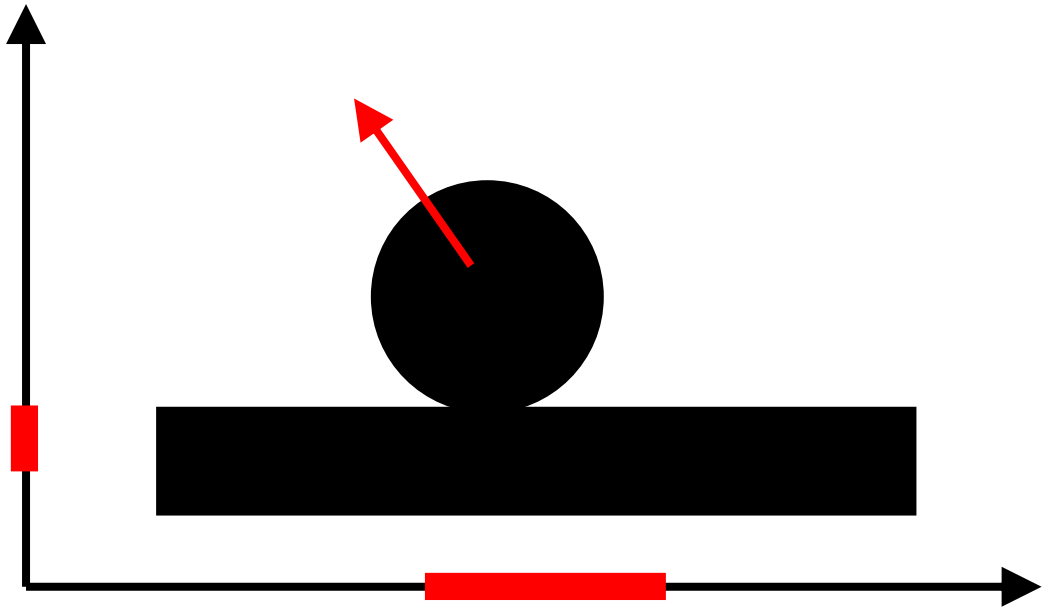
Position Correction



Position Correction



Position Correction



Step 6: User Input

- When the ball is clicked, add an impulse (instantaneous change in velocity) to the ball.
- The closer to the center of the ball the user clicks, the less powerful the impulse.

Collision Detection

The process of determining which bodies collide and when.

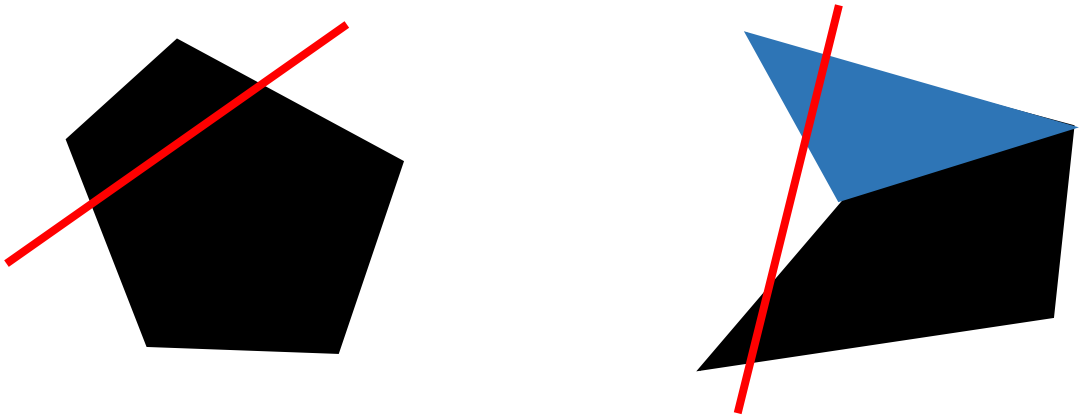
Separating Axis Theorem

Two convex shapes are separated (not penetrating) if and only if there exists an axis on which the projections of both objects does not overlap.

Collision detection in games: Given two shapes, find a separating axis. If no such axis exists, the shapes collide.

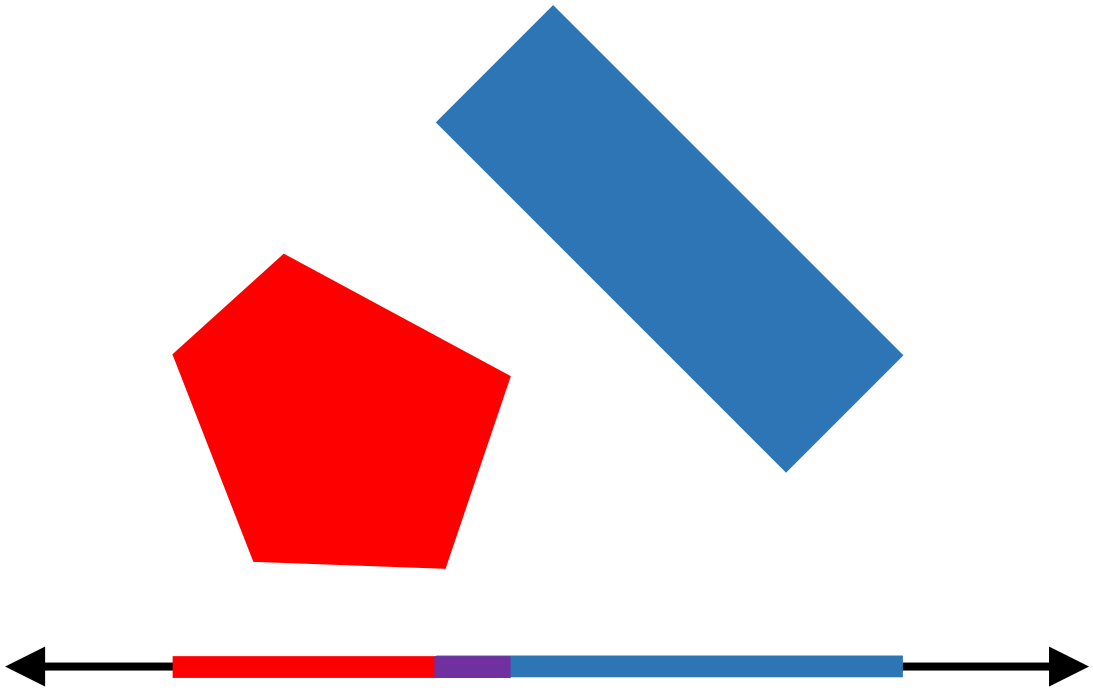
Convex vs. Concave Shapes

- A shape is **convex** if it is impossible to draw a line through the shape which intersects it in two places.
- Otherwise, a shape is **concave**.
- Concave shapes can be decomposed into groups of convex shapes.



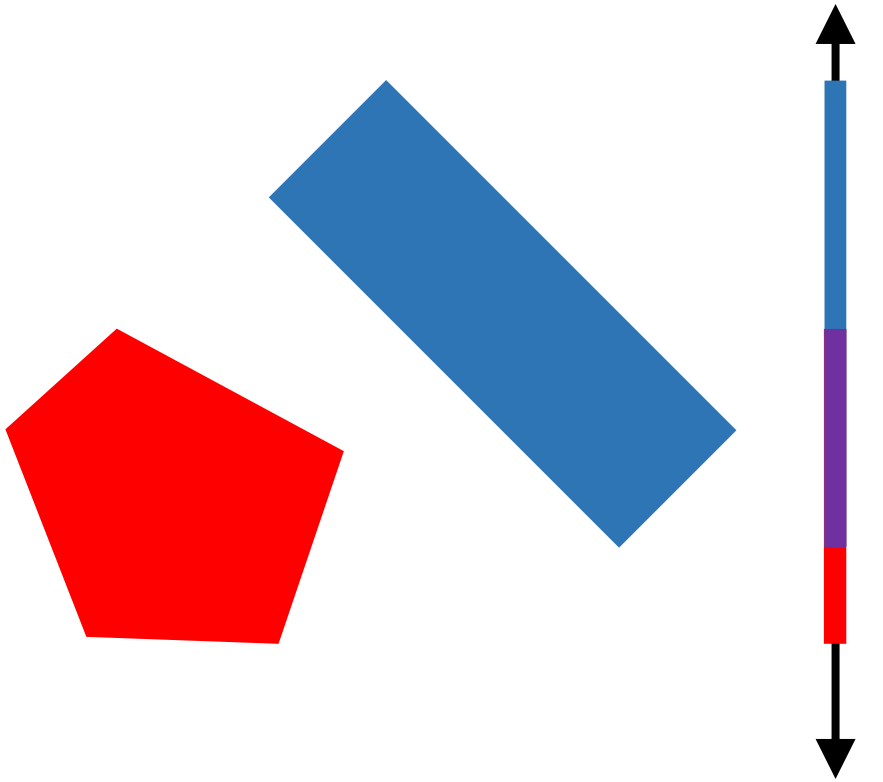
Projection on an Axis

The shadow cast by an object on a plane.



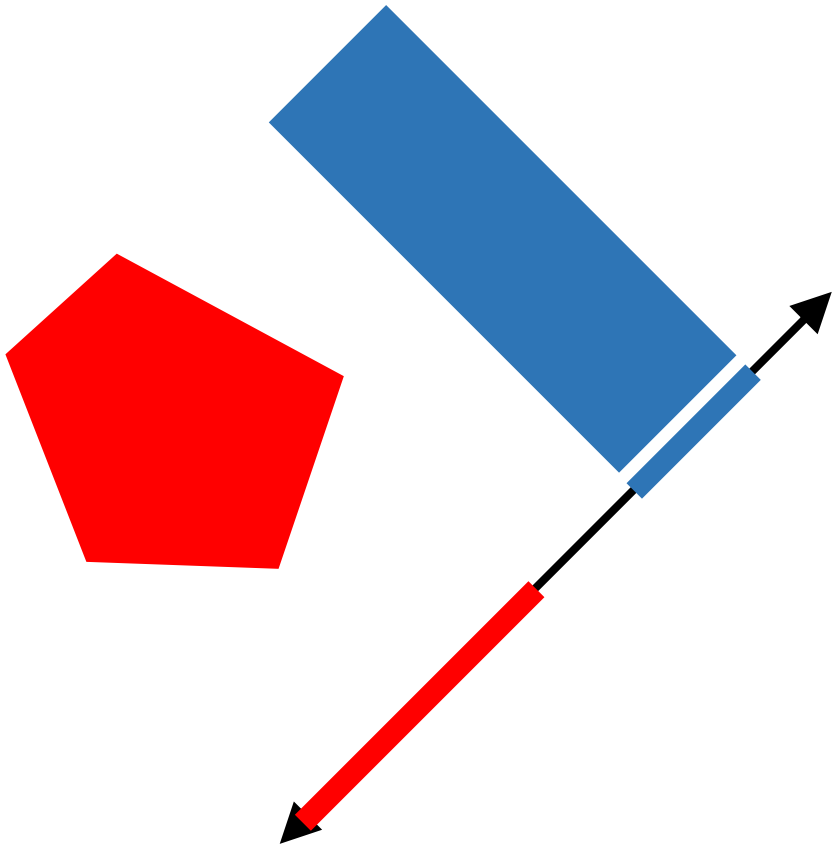
Projection on an Axis

The shadow cast by an object on a plane.



Projection on an Axis

The shadow cast by an object on a plane.



Collision Detection

Discrete (a posteriori): Detect collisions once two objects overlap.

Continuous (a priori): Calculate the moment of collision in advance.

Pro's and Con's

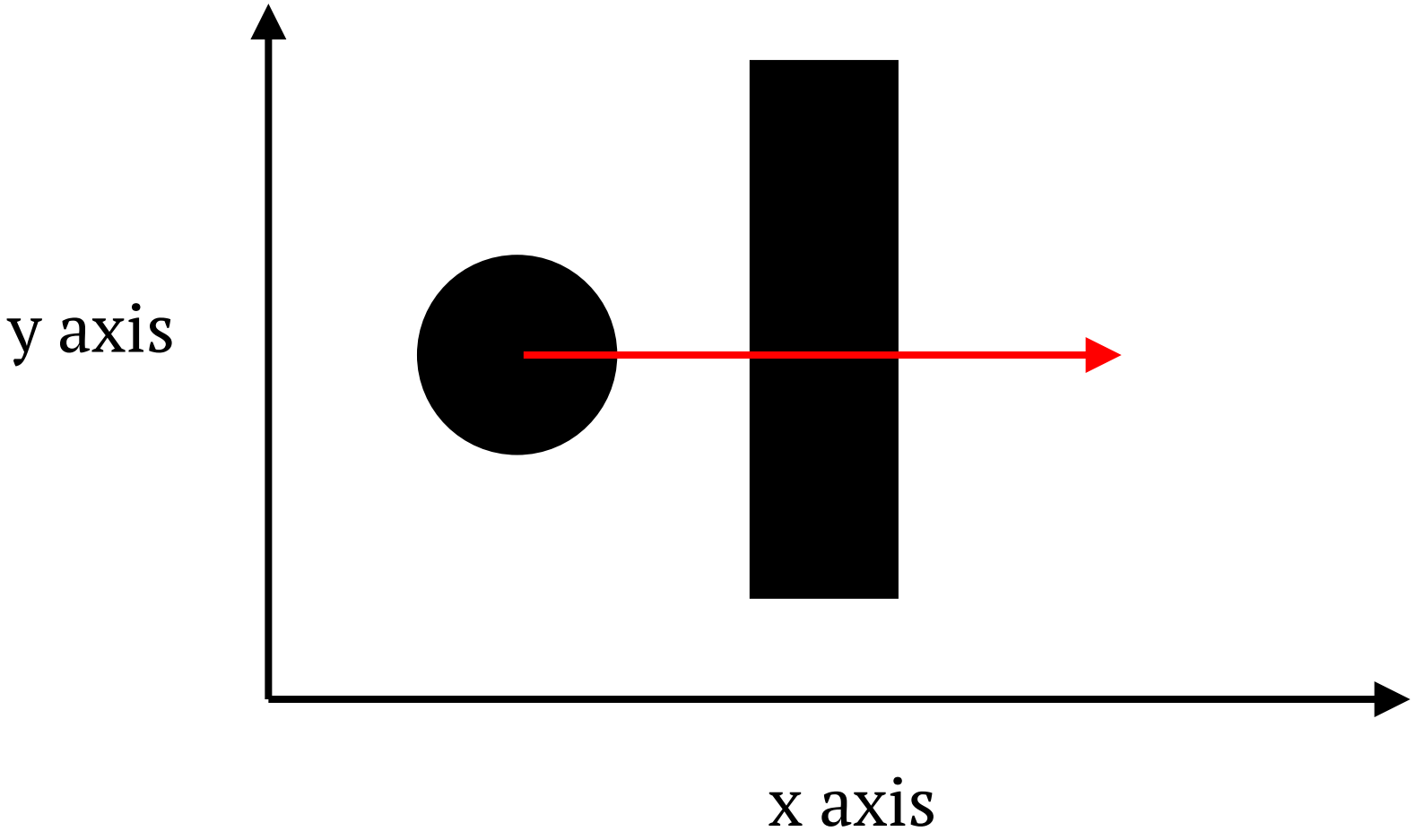
Discrete

- Computationally less expensive.
- May miss collisions (the tunneling problem).

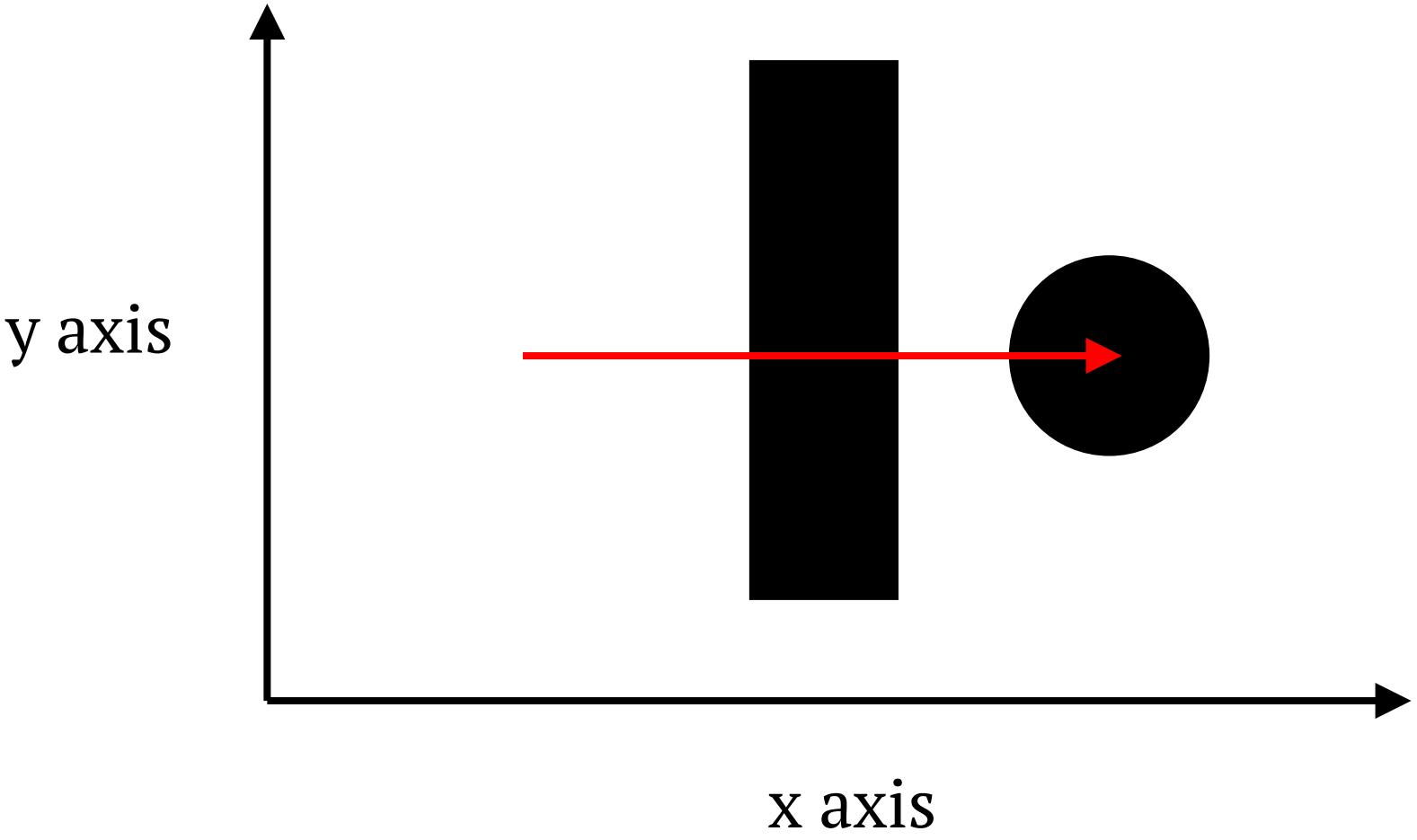
Continuous

- Computationally more expensive.
- Will not miss any collisions.

Tunneling Problem



Tunneling Problem



Tunneling Problem "Solutions"

- Don't allow fast-moving bodies.
- Use continuous collisions detection for fast-moving bodies only.
- Never move a body farther than its size. (That is, to move a body a distance 3 times its length, move it exactly its length 3 times, checking for collisions after each movement.)
- Check collisions for fast moving objects (e.g. bullets) with raycasting.

Naïve Collision Detection

Compare every pair of bodies every frame.

10 bodies = 45 comparisons
11 bodies = 55 comparisons
12 bodies = 66 comparisons
13 bodies = 78 comparisons
14 bodies = 91 comparisons
15 bodies = 105 comparisons

$$\frac{n(n-1)}{2} \in \Theta(n^2)$$

Temporal Coherence

From one moment to the next, the positions of most objects stay about the same.

Intuition: If objects did not overlap last frame, they probably don’t overlap this frame either.

2-Phase Collision Detection

Detecting whether or not two shapes overlap means checking every possible separating axis and can be computationally expensive.

2-phase collisions detection saves time:

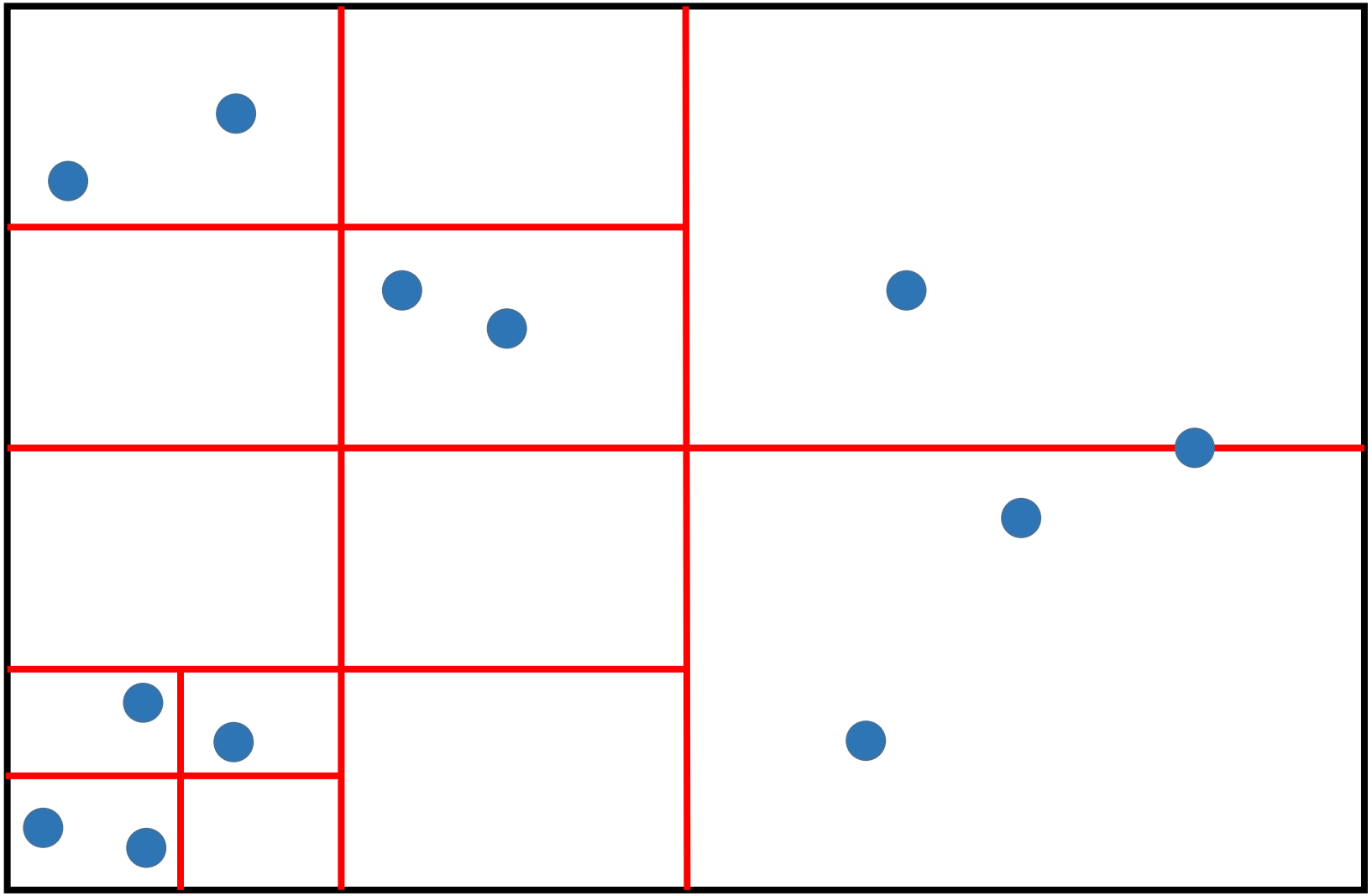
- The **broad phase** uses simple and efficient checks to rule out a large number of pairs.
- Those pairs which cannot be ruled out are then checked in the **narrow phase**.

Broad Phase

- Quad Trees / Octrees
- Sweep and Prune

Quad Tree (Broad Phase)

Objects are grouped into hierarchical sets of 4 buckets. When a bucket fills up, it is split into 4 smaller buckets.



Ideally, only 3 elements allowed per bucket.

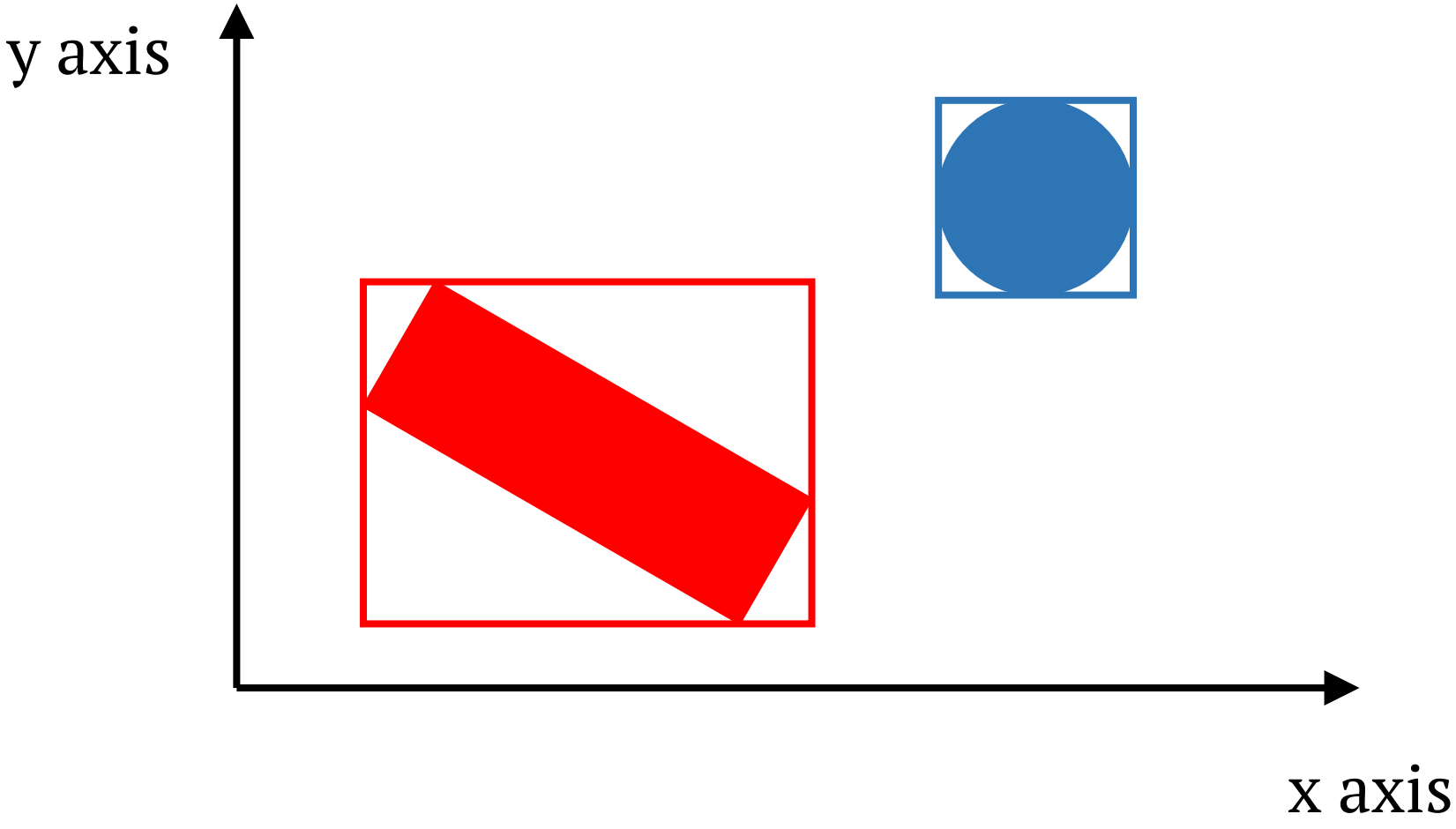
Quad Tree

- A bucket contains all the objects in it, plus all the objects in its sub-buckets.
- When it is time for collisions detection, an object only needs to be tested against other objects in its same bucket.
- A Quad Tree in 3 dimensions uses groups of 8 buckets and is called an Octree.

Sweep and Prune

Only do narrow phase collision detection if the axis-aligned bounding boxes of two bodies overlap.

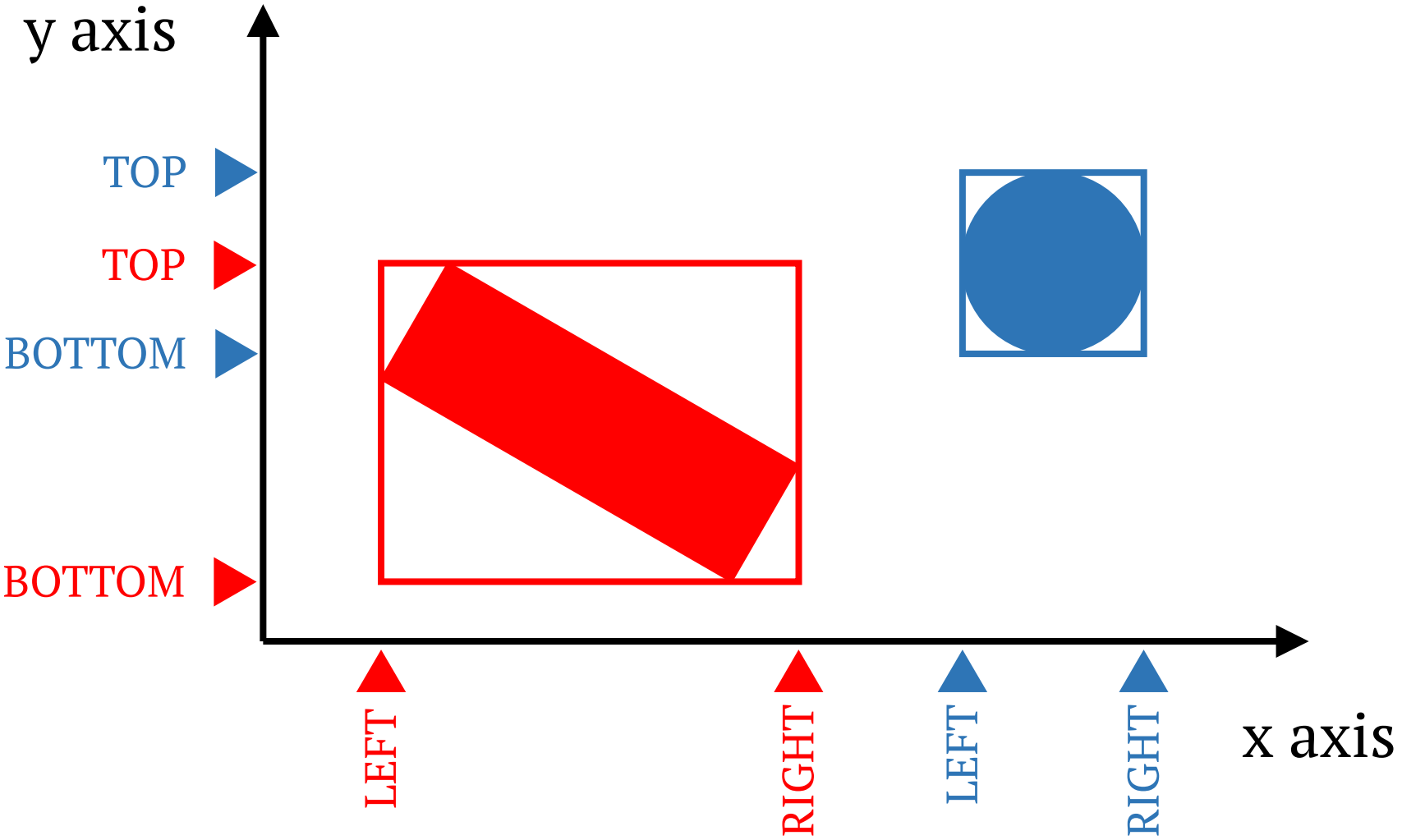
Axis-Aligned Bounding Box



Sweep and Prune

- Sort the first and second edges of each body's axis-aligned bounding box along each dimension.
- Check for overlap along each dimension.
- Only do narrow phase collision detection for objects whose AABB's overlap in all dimensions.

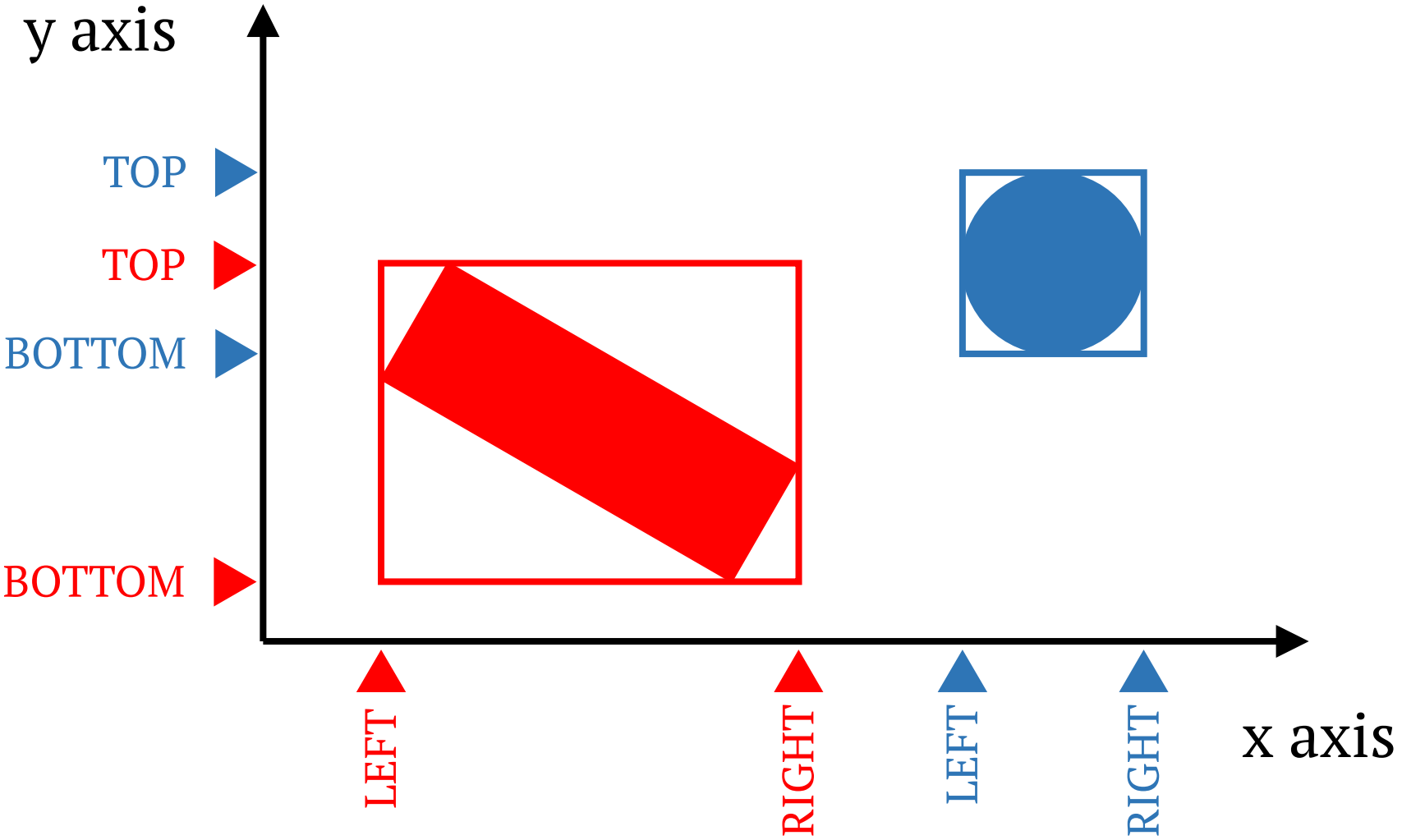
Edge Lists



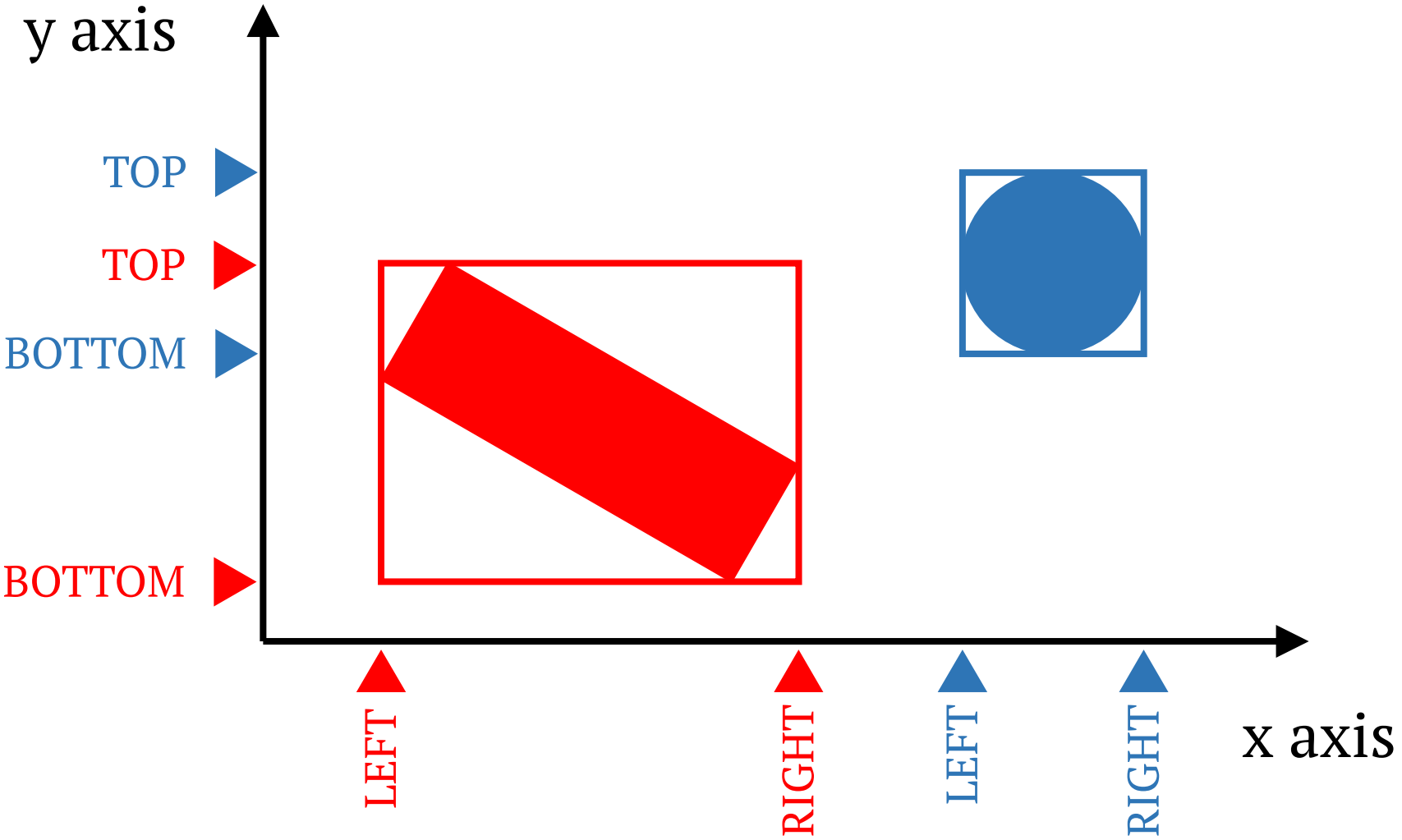
Sweep and Prune

1. Move the bodies.
2. Sort the edge lists for each dimension.
3. Check for overlaps on the first dimension.
4. If a pair overlaps on the first dimension, check if they also overlap on the second dimension. (etc. for third dimension)
5. If a pair overlaps on all dimensions, do narrow phase collision detection for that pair.

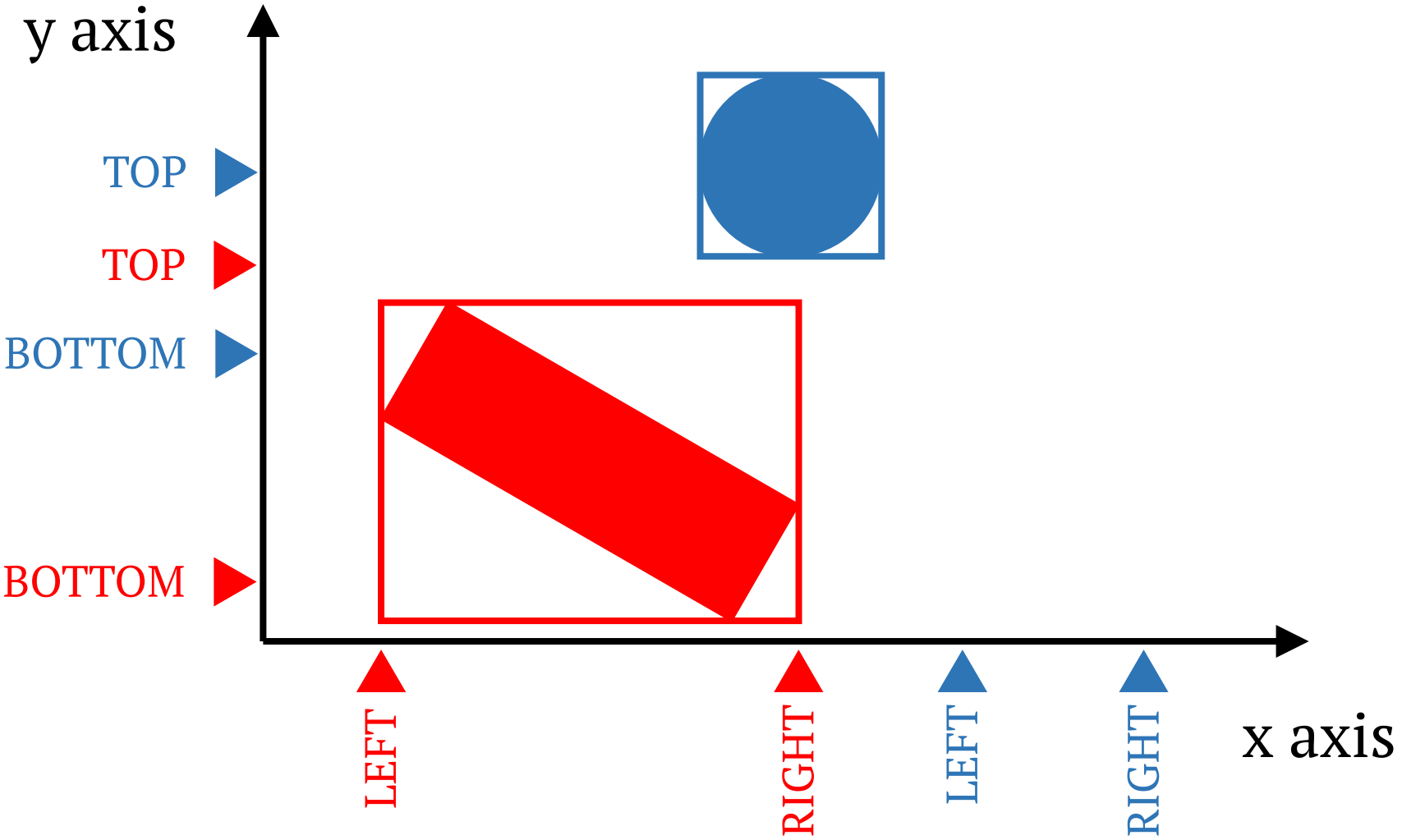
Check the X axis. The bodies do not overlap on the X axis, so they cannot overlap.



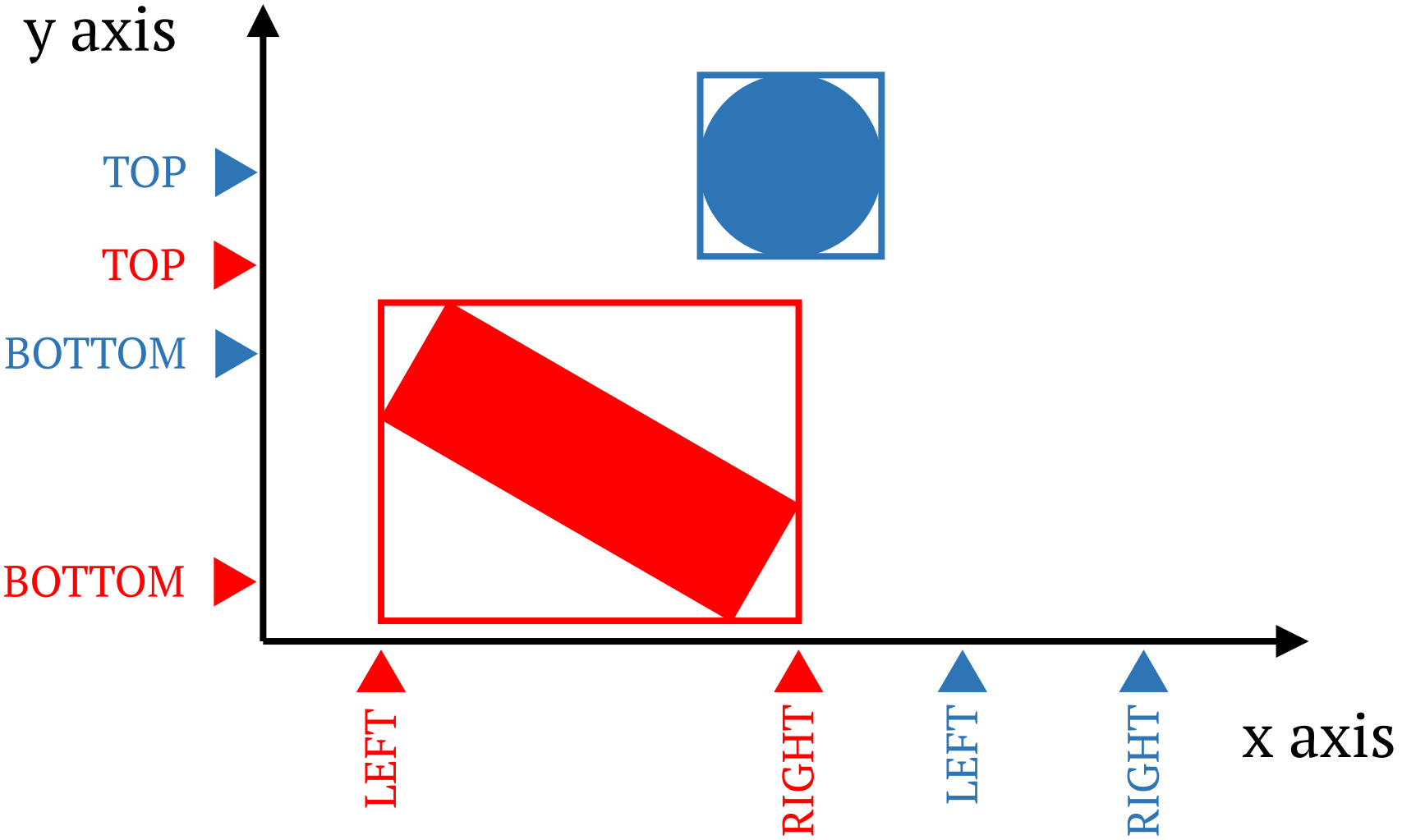
Move the bodies.



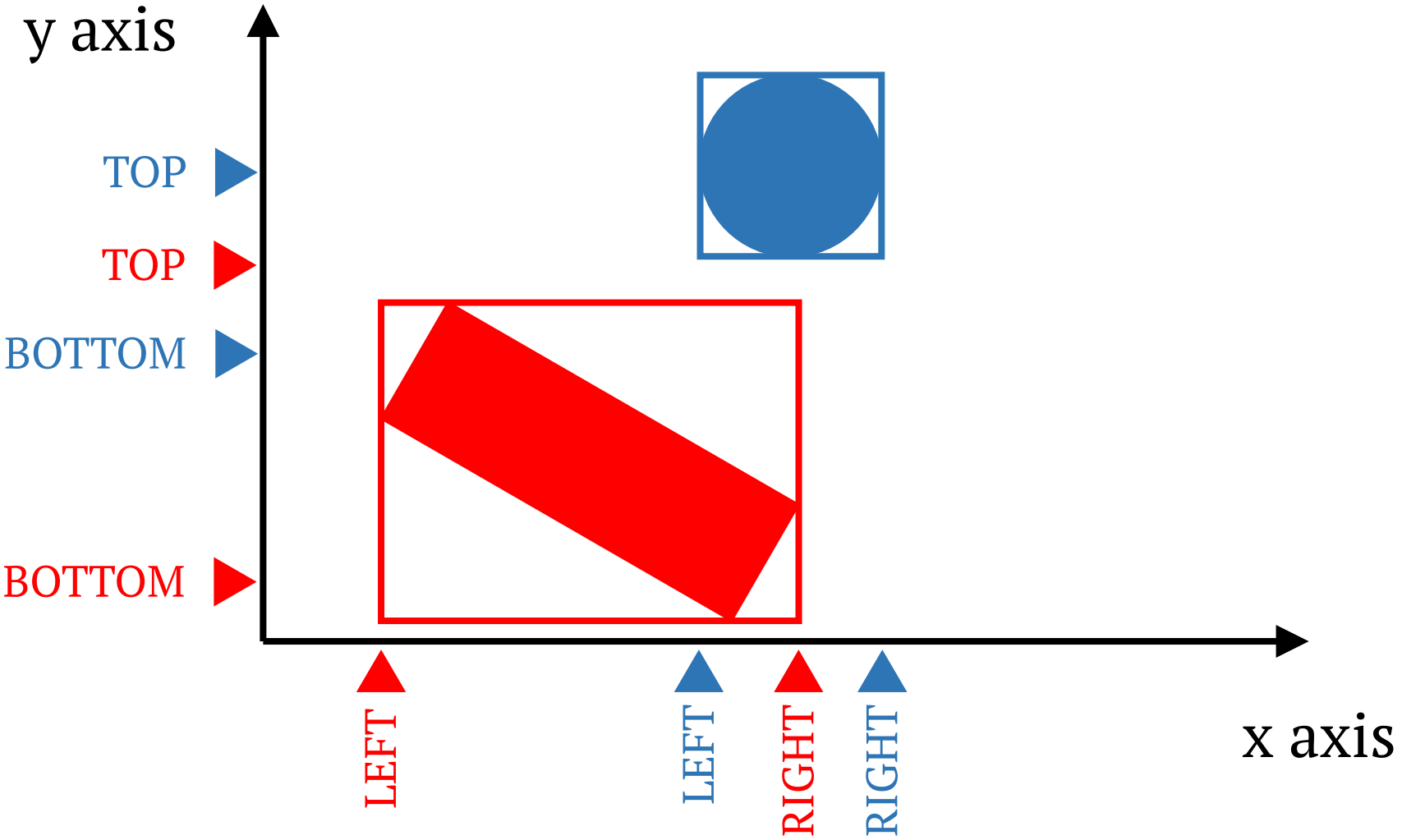
Move the bodies.



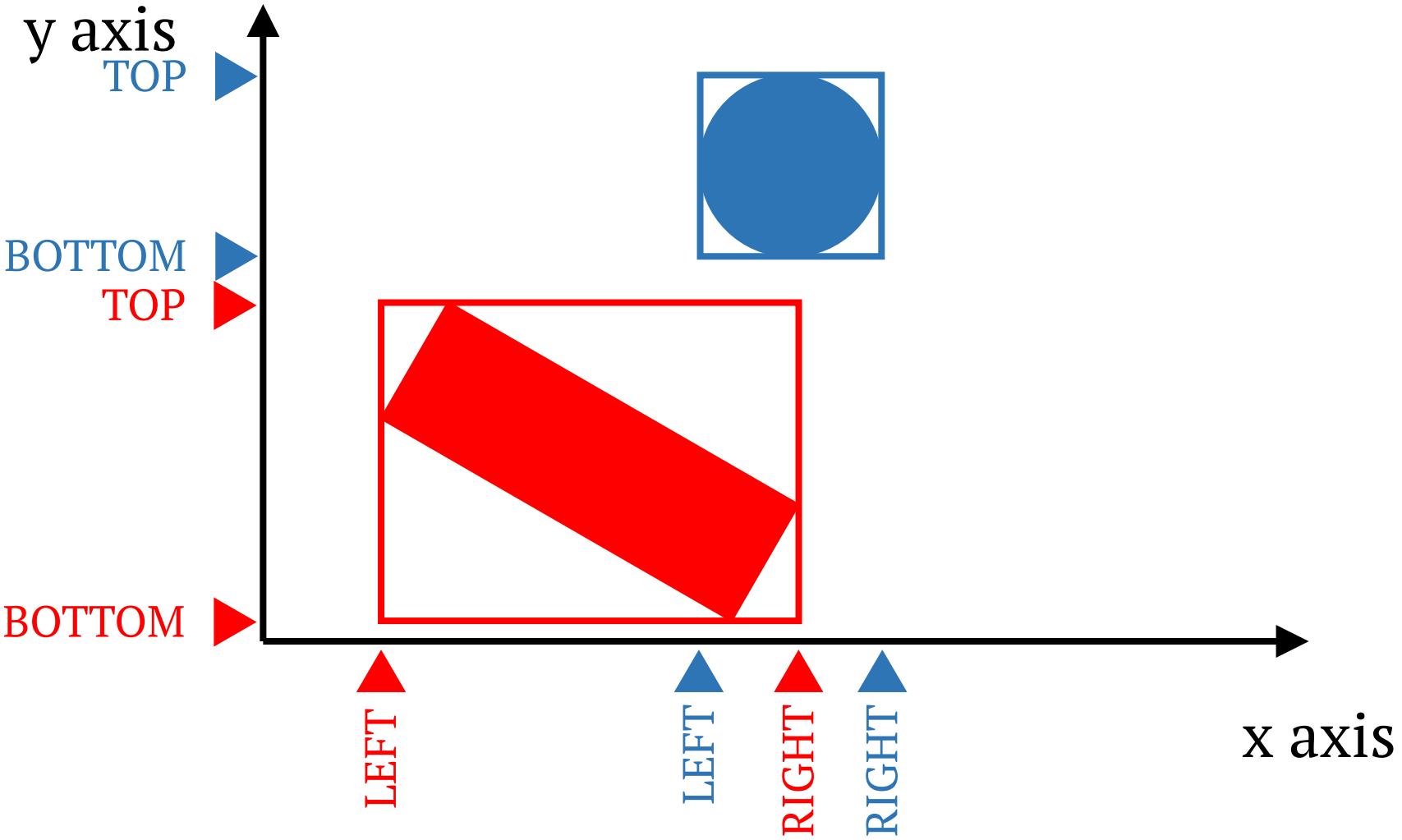
Sort the edge lists for each dimension.



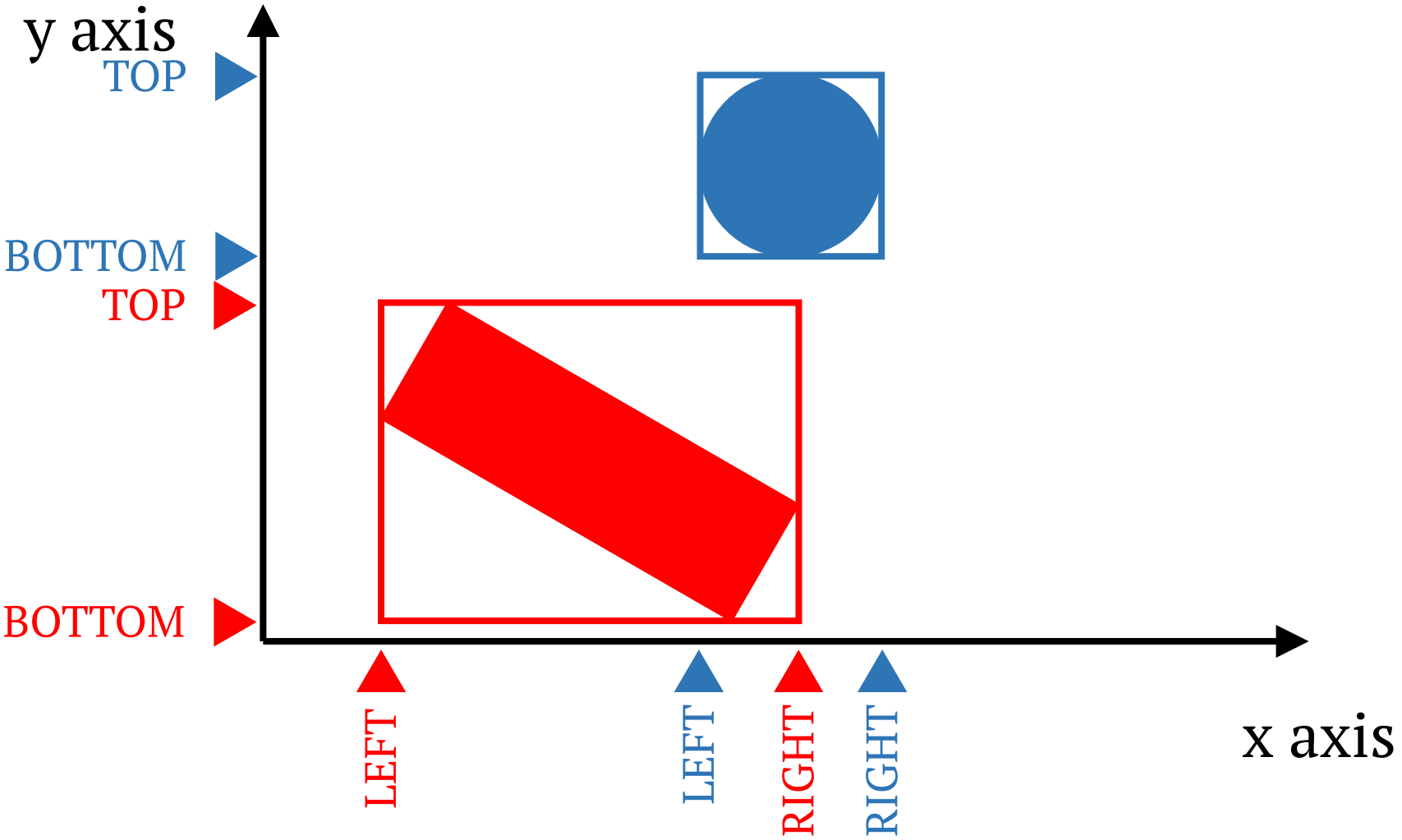
Sort the edge lists for each dimension.



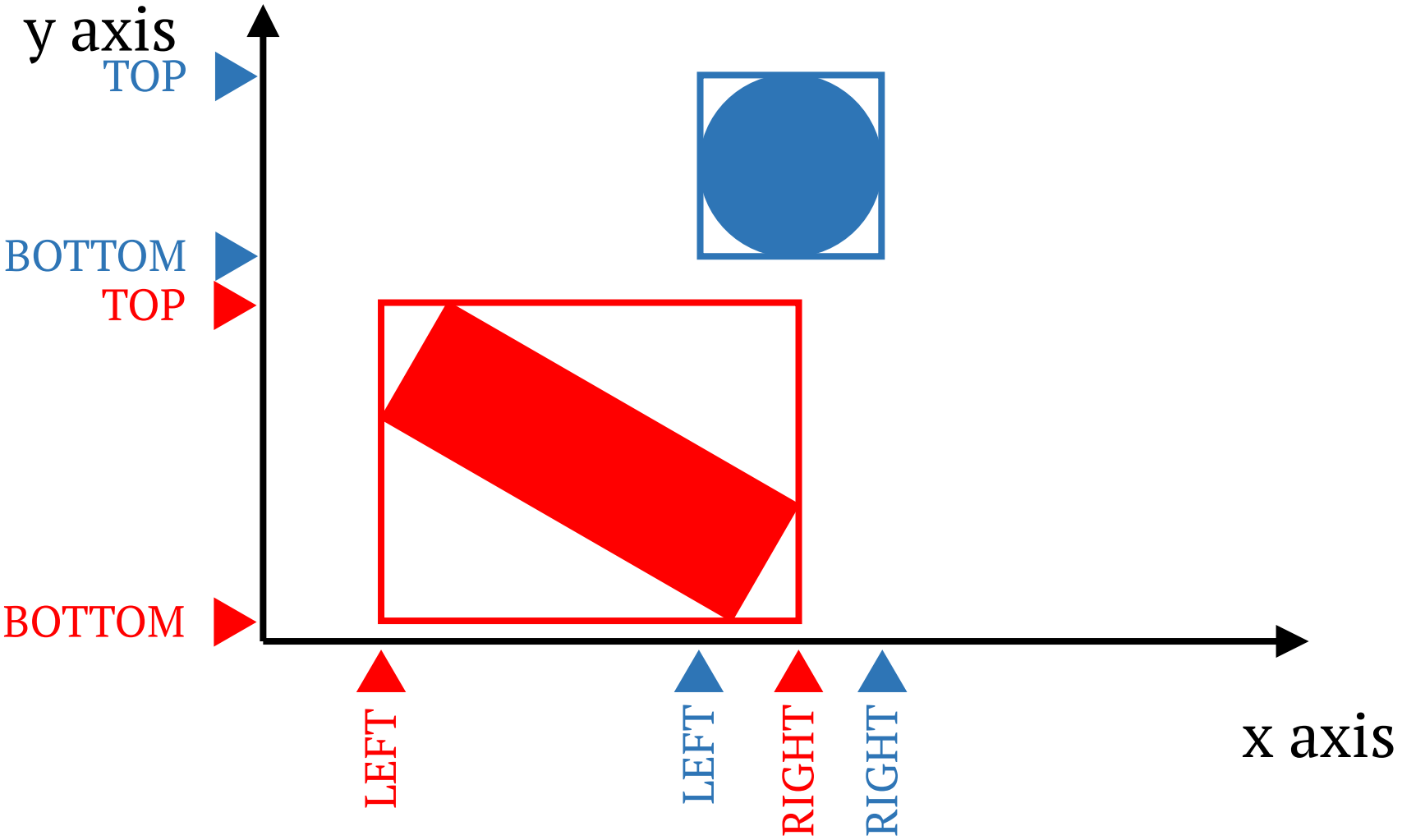
Sort the edge lists for each dimension.



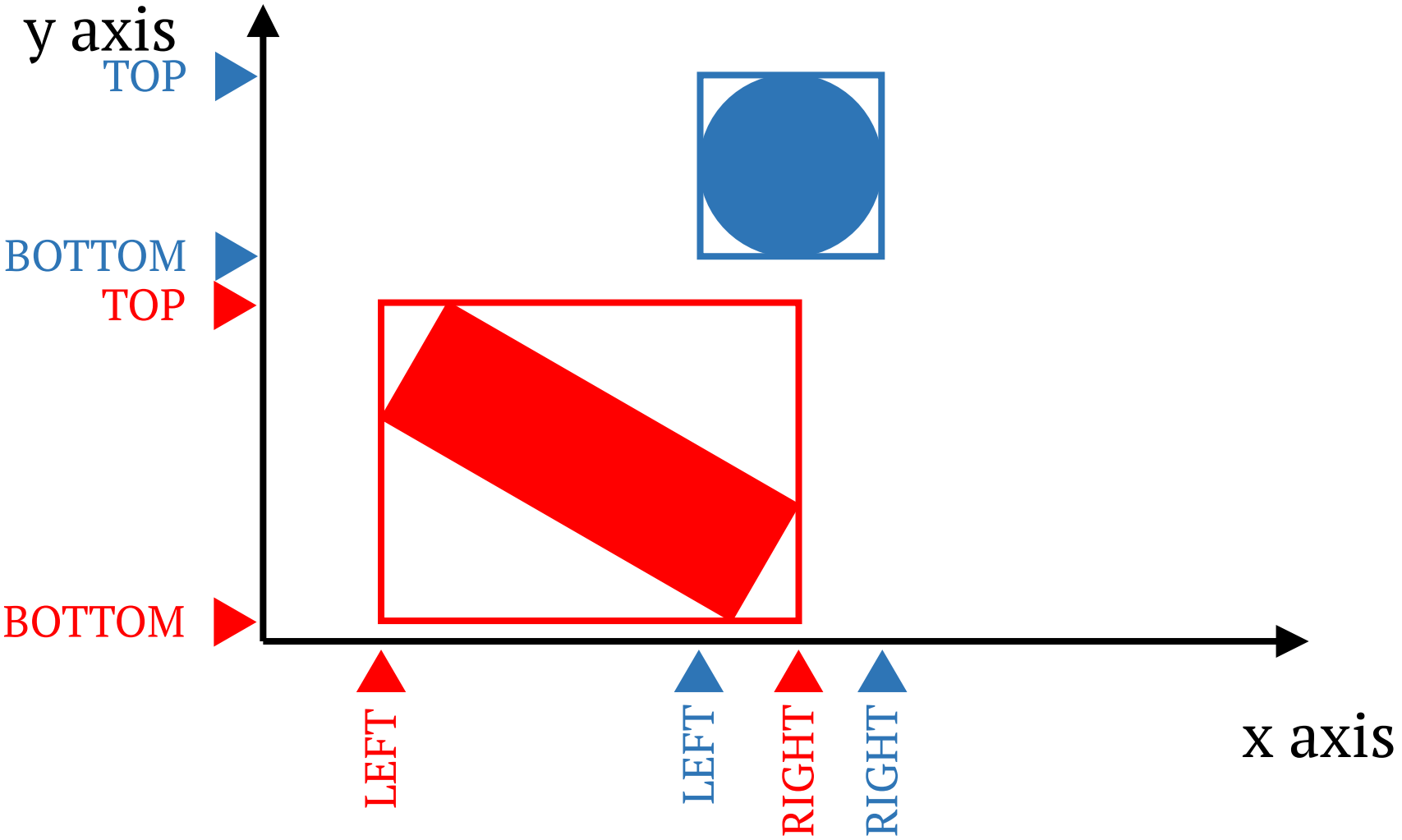
Check the X axis. The bodies overlap on the X axis, so check the Y axis.



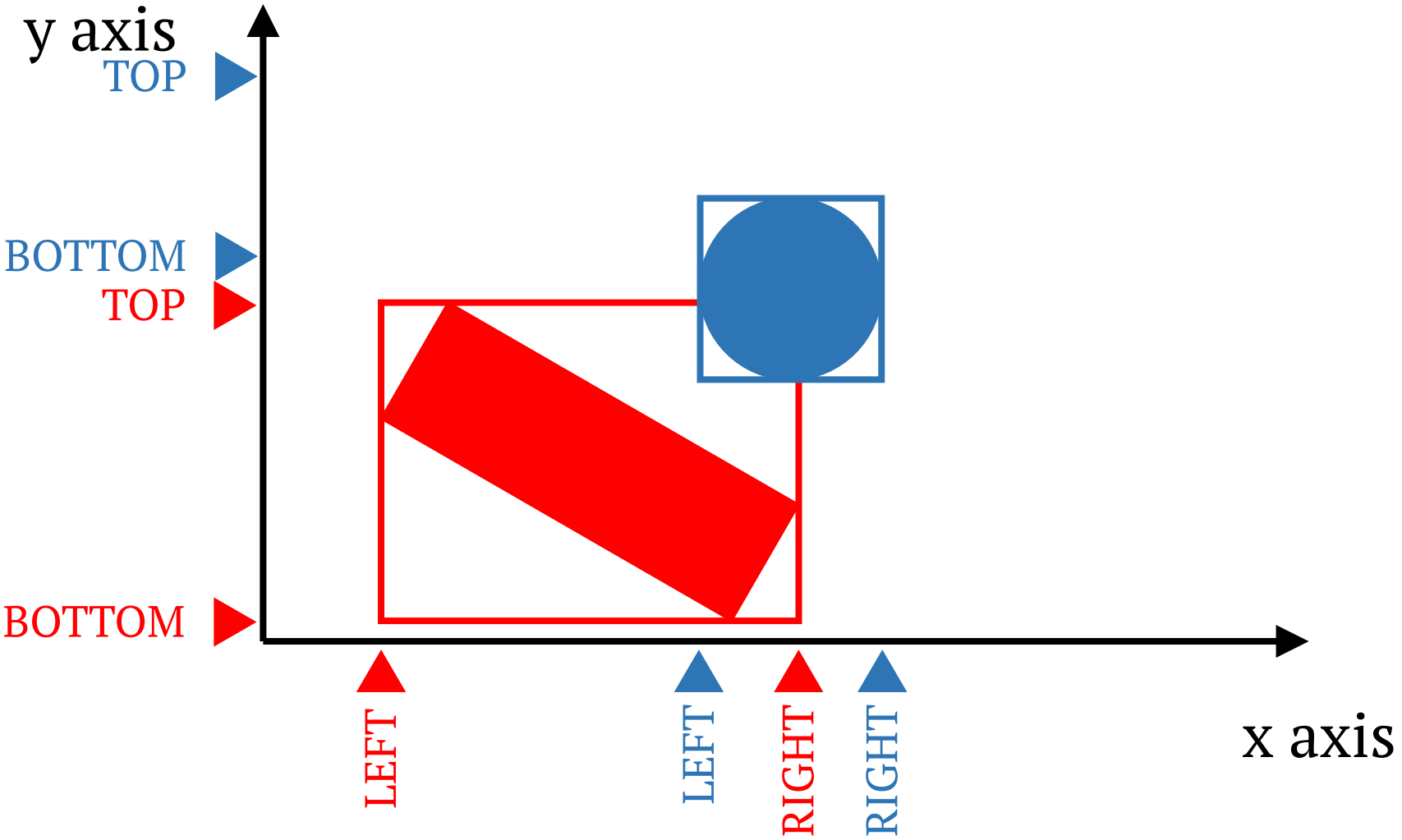
The bodies do not overlap on the Y axis, so they cannot overlap.



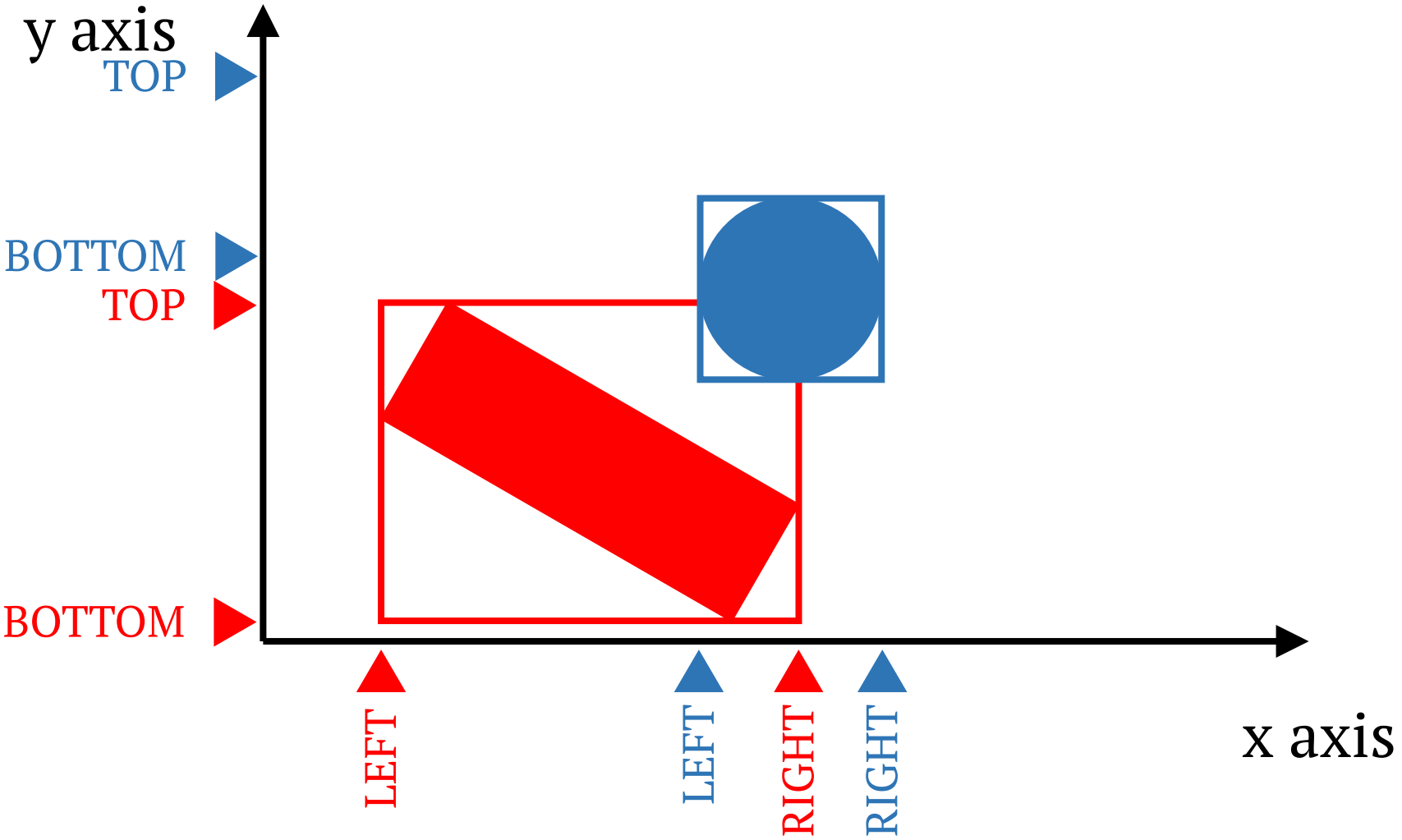
Move the bodies.



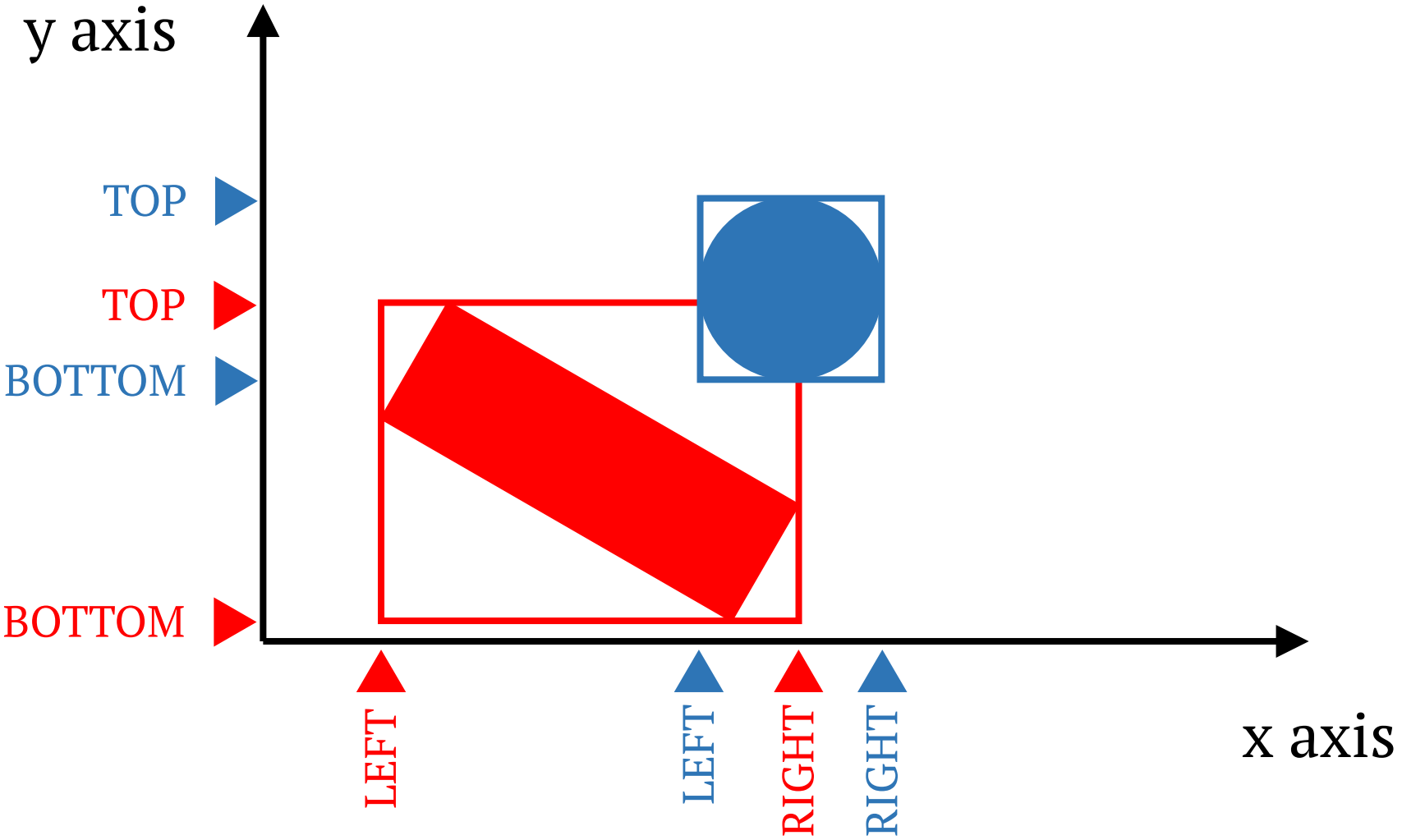
Move the bodies.



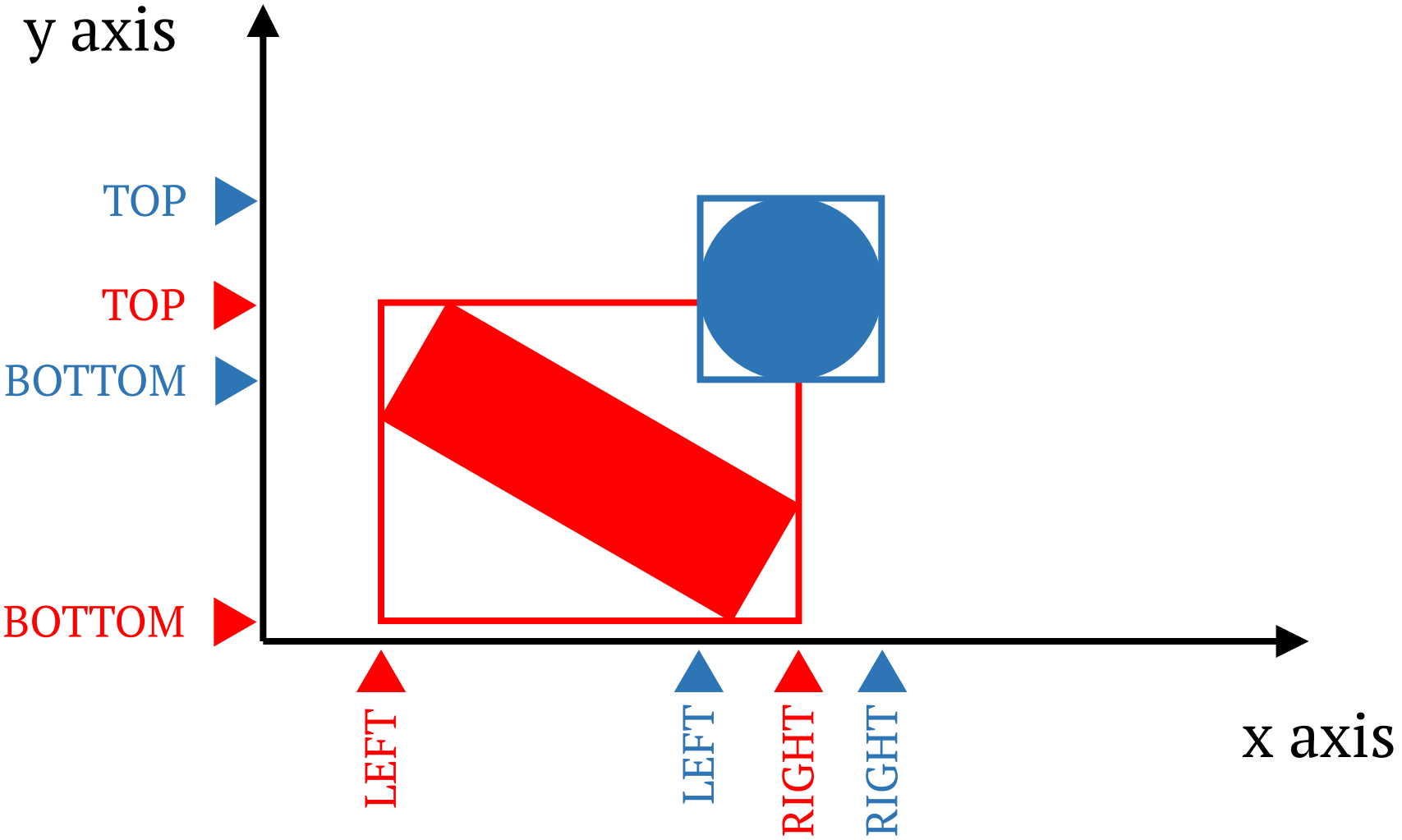
Sort the edge lists for each dimension.



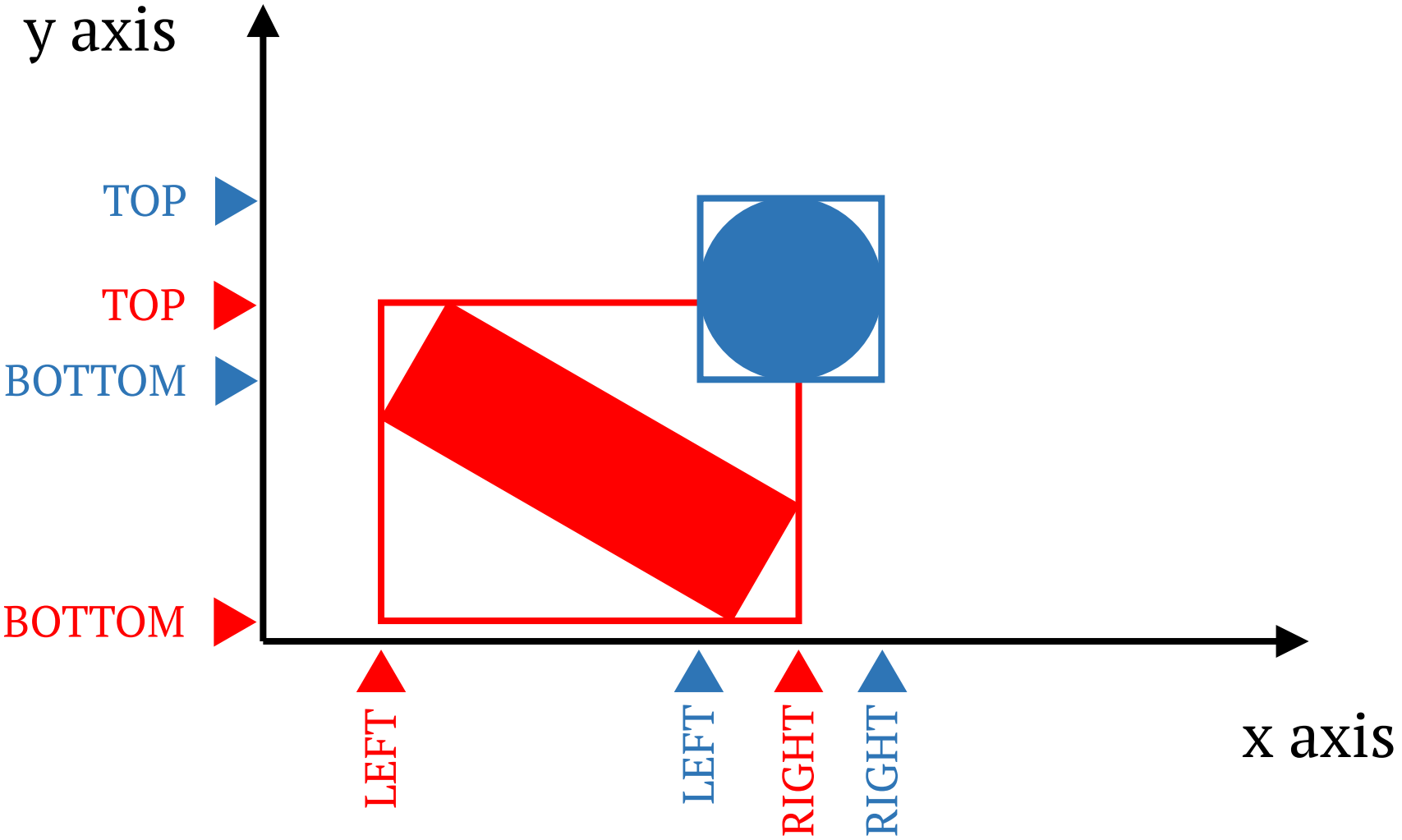
Sort the edge lists for each dimension.



Check the X axis. The bodies overlap on the X axis, so check the Y axis.



The bodies overlap on the Y axis, so do narrow phase collision detection.



Checking for Overlap

1. Begin at the first edge (e.g. LEFT).
2. Move through the list toward the second edge (e.g. RIGHT).
3. If along the way you find the first edge of another body, this pair of bodies overlaps on this dimension. Try the next dimension.