Game Physics

Game and Media Technology
Master Program - Utrecht University

Dr. Nicolas Pronost

Collision resolution

Problem

- We assume that we know a collision occurs (e.g. early termination of GJK algorithm)
- We want to solve it, *i.e.* move back the colliding objects apart from each other
- But in which direction and with what intensity?

Collision resolution

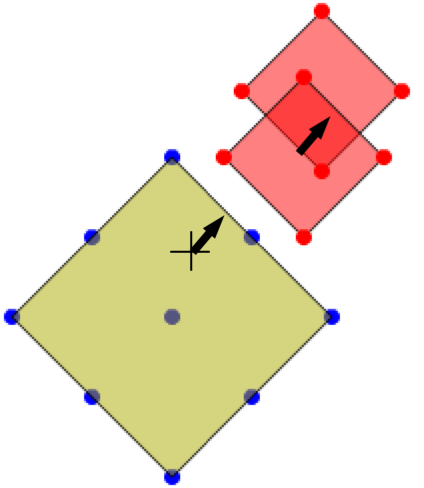
- To resolve the collision it is necessary to have
  - The time of collision, which is the first instant the collision is detected
  - The contact point, which is the point where the colliding objects first touch
  - A contact normal, which is a normal to the plane passing through the contact point and oriented in the direction separating the two objects

Time of collision

- Computing the exact time (somewhere between t and $t + \Delta t$) of collision is not always feasible
- We can approximate it by bisection
- By repeatedly bisecting the time interval and performing an intersection test, we can find an arbitrary short interval $[t_0, t_1]$ for which:
  - The objects do not collide at t_0
  - The objects collide at t_1
- Computationally expensive, so usually in games the frame rate Δt is small enough to not having to do bisection

Contact point and normal

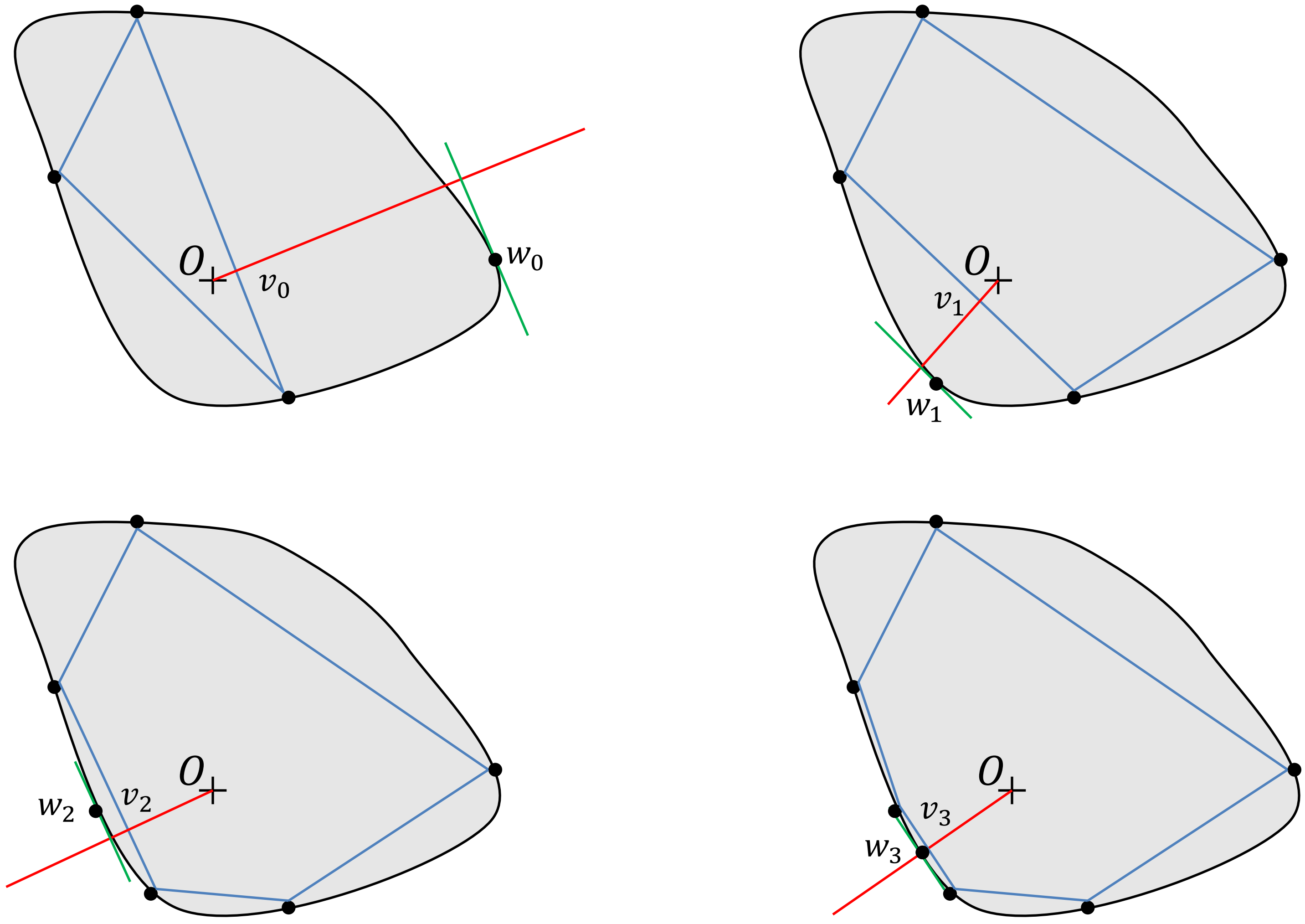
- If you use the GJK algorithm in the narrow phase of the collision detection, the final simplex can be used to determine the contact points
  - The vector from the origin to its closest point on the surface is the penetration vector
  - Expanding-polytope algorithm (EPA) is used to calculate that point and vector



Expanding-polytope algorithm

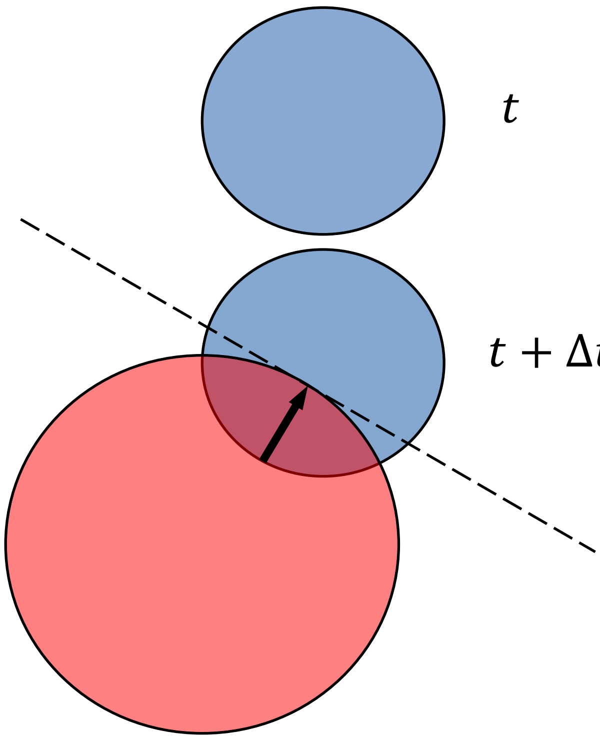
- Starting from the GJK termination simplex (which contains the origin), we 'inflate' the polygon by splitting edges
  - At each iteration k, we compute the point v_k on the affine hull of the edge the closest to the origin
  - Then that closest edge is cut into two edges by adding the support point w_k to the polytope
  - We repeat that until the distance between v_k and w_k is small enough ($< \varepsilon$)

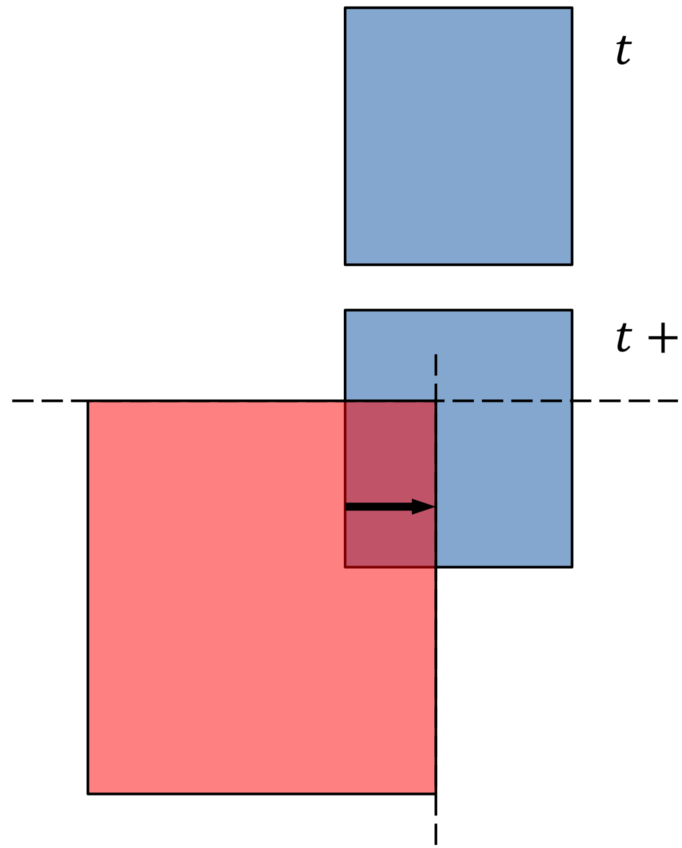
Expanding-polytope algorithm



Contact point and normal

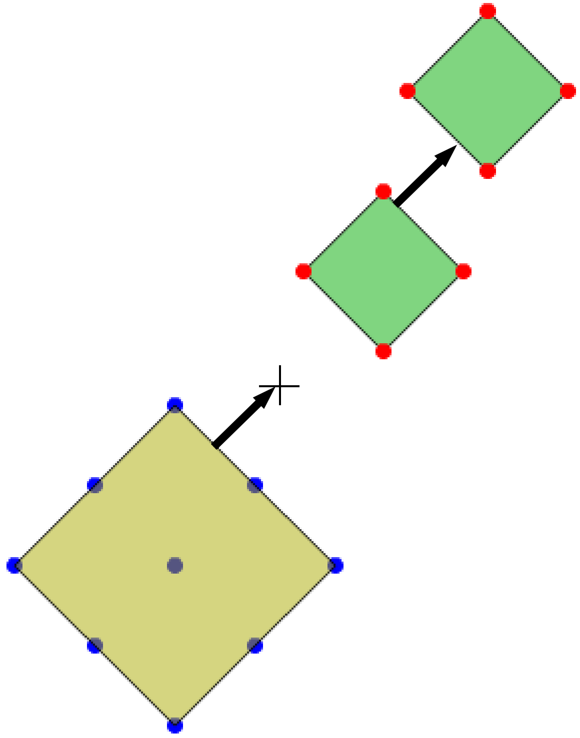
- It does not take the motion of the objects into account





Contact point and normal

- To estimate the direction of the collision we can use the position of the objects at the frame before the collision
  - Assume that the closest points at $t - \Delta t$ (GJK) are the points with largest depth penetration at t
  - Needs to store previous positions, but usually done for numerical integration anyway



Collision resolution

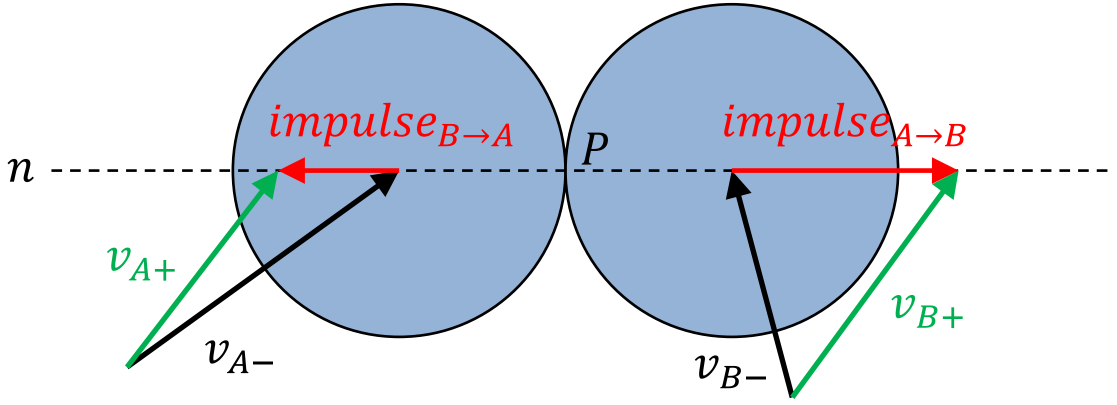
- We now have a mean to estimate time of collision, contact points and contact normal
- We still have to correct the position and orientation of the colliding objects

Linear velocity

- Let denote the two objects A and B, with respective mass m_A and m_B and velocity v_A and v_B, the unit collision normal n and the contact point P
- We can solve the collision by using an impulse-based technique
  - at collision time we apply an impulse on each object at P in the direction n ($-n$ for one of the object)
  - the impulse 'pushes' the two objects apart
  - as the two objects have different masses and incoming velocities, they are not pushed away with the same magnitude
  - we denote the scaling factor of that magnitude j

Linear velocity

- The scaled magnitude of impulse is then added to the current velocity of each object
  - v_- is the velocity at collision time and v_+ is the velocity after collision resolution



Types of collision

- Inelastic collisions
  - where energy is not preserved in the collision
    - *e.g.* objects stop in place
  - are easy to implement
    - *e.g.* backing out or stopping process
- Elastic collisions
  - where energy is fully preserved in the collision
    - *e.g.* billiard balls
  - are more difficult to calculate
    - *e.g.* magnitude of resulting velocities

Coefficient of restitution

- To model the elasticity of a collision we define the coefficient of restitution C_R of the collision
- It describes the ratio of speeds after and before collision along the collision normal

$$C_R = -\frac{(v_{A+} - v_{B+}) \cdot n}{(v_{A-} - v_{B-}) \cdot n}$$

- In practice we define a coefficient for each object with respect to collisions with a perfectly rigid and elastic object
  - if $C_R = 1$, we have an elastic collision (E_k is conserved)
  - if $C_R < 1$, we have an inelastic collision (lost of velocity)
  - if $C_R = 0$, the objects will stick together after collision

Reminder

- An impulse is the rate of change of the momentum, *i.e.* a force delivered in an instant

$$F\Delta t = \Delta p = m(v(t+\Delta t) - v(t))$$
$$\tau\Delta t = \Delta L = I(\omega(t+\Delta t) - \omega(t))$$

- The momentum of a system is always conserved

$$m_A v_A(t+\Delta t) + m_B v_B(t+\Delta t) = m_A v_A(t) + m_B v_B(t)$$

- The energy of a system is always conserved

$$E_{Kt}(t+\Delta t) + E_P(t+\Delta t) + E_{Kr}(t+\Delta t)$$
$$= E_{Kt}(t) + E_P(t) + E_{Kr}(t) + E_O$$

Linear velocity

- The total momentum of the system before and after collision is conserved, so

$$m_A v_{A-} + j_A * n = m_A v_{A+}$$
$$m_B v_{B-} - j_B * n = m_B v_{B+}$$

- Which can be written

$$v_{A+} = v_{A-} + \frac{j_A}{m_A} n$$
$$v_{B+} = v_{B-} - \frac{j_B}{m_B} n$$

Linear velocity

- We also know that the velocities before and after collision relate with the coefficient of restitution

$$C_R = -\frac{(v_{A+} - v_{B+}) \cdot n}{(v_{A-} - v_{B-}) \cdot n}$$

- So we have

$$j_A = \frac{-(1 + C_{R_A})(v_{A-} - v_{B-}) \cdot n}{\left(\frac{1}{m_A} + \frac{1}{m_B}\right)}$$

$$j_B = \frac{-(1 + C_{R_B})(v_{A-} - v_{B-}) \cdot n}{\left(\frac{1}{m_A} + \frac{1}{m_B}\right)}$$

Linear velocity

7.1

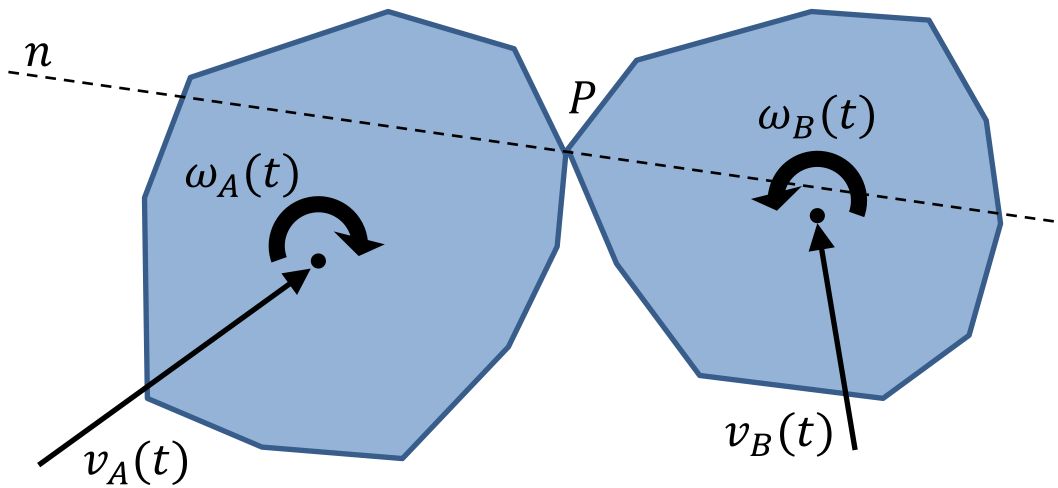
- We can finally calculate the outgoing velocities

$$v_{A+} = v_{A-} + \frac{j_A}{m_A} n$$
$$v_{B+} = v_{B-} - \frac{j_B}{m_B} n$$

  - the larger the mass of an object, the more resistant it is to velocity change
  - but (from *j*) less resistant when the relative velocities of the objects increase or when the combined masses increase

Angular velocity

- Of course a collision where the normal is off the center of rotation of the objects produce also a rotation of the two objects



Angular velocity

- The way to handle rotational collision is very similar to how we handled linear collision
- The impulse factor j needs to be adapted
- If one or both objects are rotating, linear velocity from the rotation is added to the velocity

$$\bar{v}_A = v_A + \omega_A \times r_A$$
$$\bar{v}_B = v_B + \omega_B \times r_B$$

where ω are the angular velocities and r the displacement from the center of rotation to the points of contact

Angular velocity

- We thus have the following updated coefficient of restitution

$$C_R = -\frac{(\bar{v}_{A+} - \bar{v}_{B+}) \cdot n}{(\bar{v}_{A-} - \bar{v}_{B-}) \cdot n}$$

- This coefficient is used for further calculation of the linear velocity through the updated j_A and j_B

Angular velocity

- The angular momentum before and after collision is also conserved

$$I_A \omega_{A-} + r_A \times (j * n) = I_A \omega_{A+}$$
$$I_B \omega_{B-} - r_B \times (j * n) = I_B \omega_{B+}$$

- Which can be written

$$\omega_{A+} = \omega_{A-} + I_A^{-1}(r_A \times (j * n))$$
$$\omega_{B+} = \omega_{B-} - I_B^{-1}(r_B \times (j * n))$$

Angular velocity

- As we did with linear velocity we can now calculate the updated factor j

$$j = \frac{-(1 + C_R)(v_{A-} - v_{B-}) \cdot n}{\left(\frac{1}{m_A} + \frac{1}{m_B}\right) + \left[\left(I_A^{-1}(r_A \times n)\right) \times r_A + \left(I_B^{-1}(r_B \times n)\right) \times r_B\right] \cdot n}$$

with $j = j_A$ when $C_R = C_{R_A}$, and $j = j_B$ when $C_R = C_{R_B}$

Angular velocity

- With this updated factor j, we calculate the outgoing angular velocities

$$\omega_{A+} = \omega_{A-} + I_A^{-1}(r_A \times (j_A * n))$$
$$\omega_{B+} = \omega_{B-} - I_B^{-1}(r_B \times (j_B * n))$$

- This factor is also used to calculate the outgoing linear velocities (same as linear resolution)

$$v_{A+} = v_{A-} + \frac{j_A}{m_A} n$$
$$v_{B+} = v_{B-} - \frac{j_B}{m_B} n$$

Collision resolution

7.2

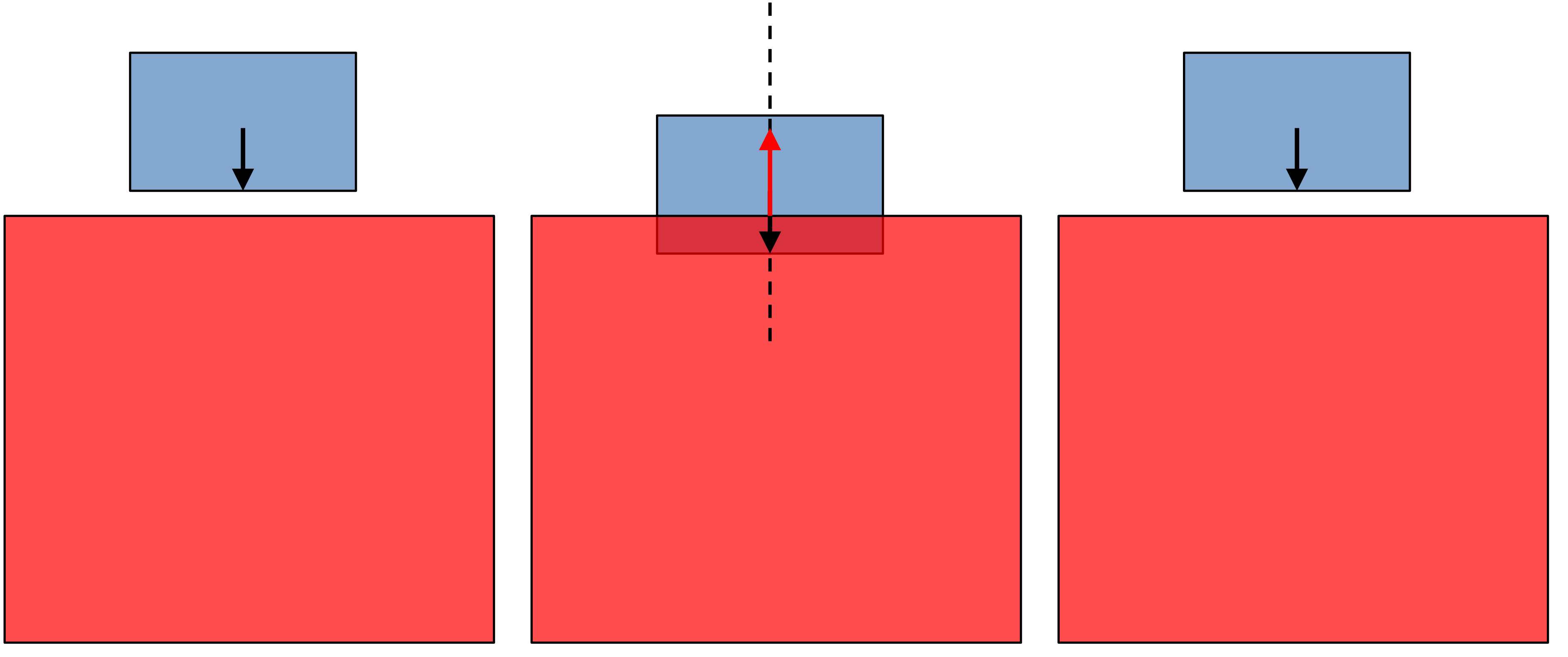
- The final algorithm can be summarized as follows
  - Run collision detection to find contact point and contact normal
  - Calculate linear and angular velocities at that contact points
  - Use coefficients of restitution and conservation of momentum to determine the impulses to apply
  - Solve for velocities using the impulses

Resting contact

- Our resolution system as it is will work fine, correcting position and orientation of colliding objects
- But some special cases can be handled more efficiently
- One of these cases occurs when we have resting contacts between objects
  - for example a box sitting on the floor
  - the floor theoretically moves down, but its mass is very large, and its outgoing velocity is very small and therefore neglected

Resting contact

- In our current framework, a box sitting on the floor will ‘oscillate’ around the surface



Resting contact

- A resting contact occurs when the relative velocity of the two objects along the normal is null (for us it means smaller than an ε)
- One solution is to ‘artificially’ reduce the coefficient of restitution when we are in that case
  - typically linearly dependent on the relative velocity or directly set to zero
  - after resolution the two objects will have a null relative velocity, so the box will stick on the floor which itself does not move

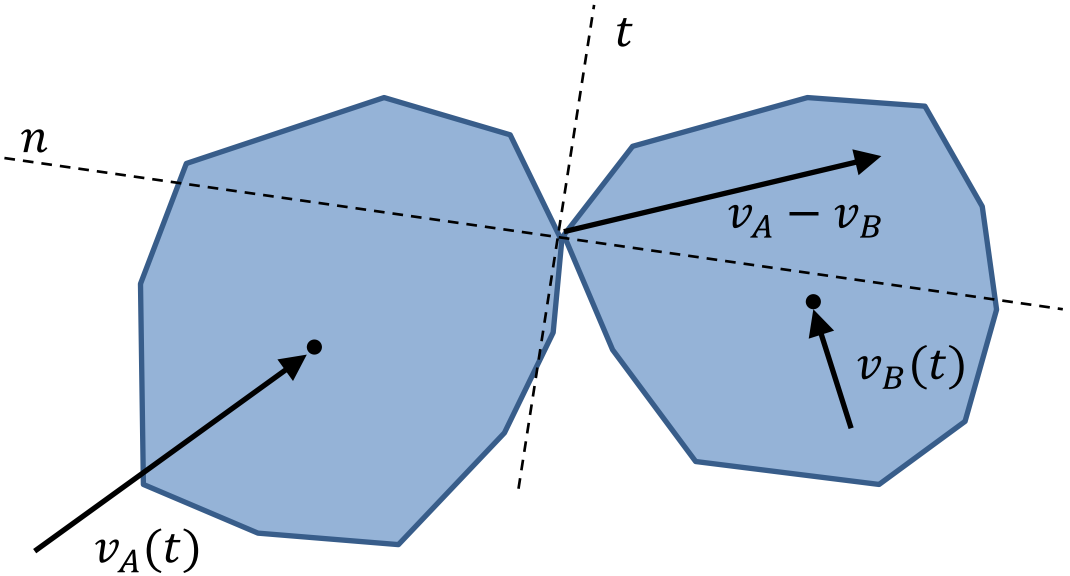
Friction

- Remember that there is friction between two objects when they are in contact
  - static friction when they do not move relatively to each other
  - kinetic friction when they move relatively to each other
  - rolling friction is usually ignored in game physics
- When they are in contact, we can add the friction force in our previous equations using impulses

Friction

- The friction acts in the tangential plane of the collision normal and resists the movement

$$t = \left(n \times (v_A - v_B)\right) \times n$$



where the vectors are normalized

Friction

- The velocity equations become

$$v_{A+} = v_{A-} + \frac{j_A(n + \mu_k t)}{m_A}$$

$$v_{B+} = v_{B-} - \frac{j_B(n + \mu_k t)}{m_B}$$

$$\omega_{A+} = \omega_{A-} + I_A^{-1}(r_A \times (j * (n + \mu_k t)))$$

$$\omega_{B+} = \omega_{B-} - I_B^{-1}(r_B \times (j * (n + \mu_k t)))$$

Friction

- We assumed here that the friction was a kinetic friction
- If the relative velocity is small enough, static friction should be used instead and the friction impulses need to be adjusted
- The tangential direction is sometimes undetermined (collision normal and relative velocity parallel), then alternative techniques should be used

End of
Collision resolution

Next

Soft body physics