COMP 476

Advanced Game Development

Session 10
Contact Physics

(Reading: Millington § 14-15, 16.2, 17.2)

Lecture Overview

- Collision Resolution
- Resting Contacts and Friction
- Optimizations

Collision Resolution

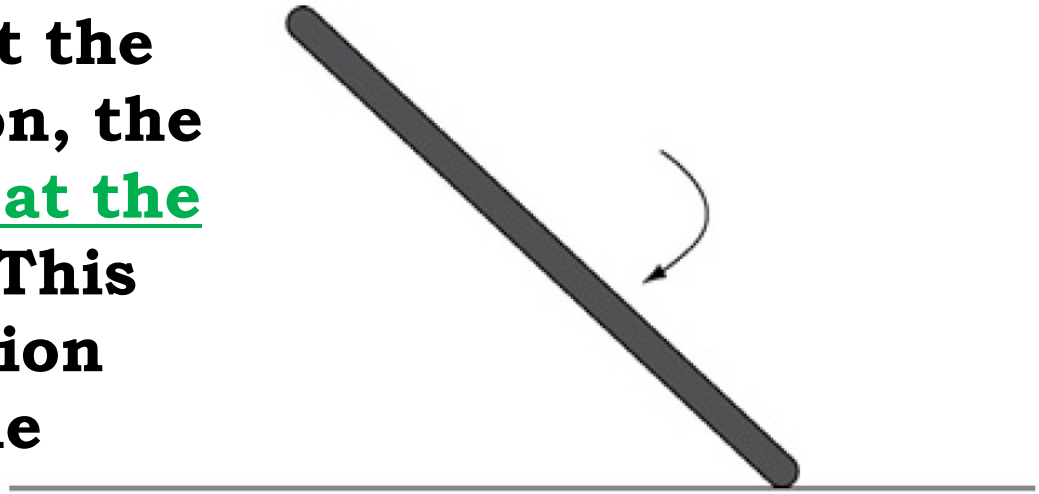
- **With *the generated set of contact data from the collision detector*, using *rigid-body equations of motion*, we can have rotating objects respond to contacts.**
- **First we will look in detail at the physics of contact, including basic collision handling.**

Impulses

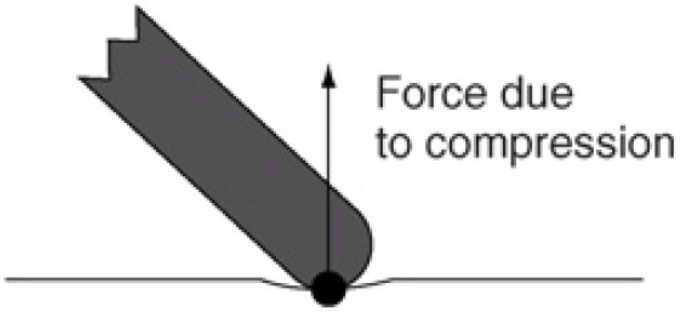
- ***To simulate the instant compression and decompression of objects during collision***, **we will use impulses. These impulses will change the velocity of each object in the collision.**
- **With *rotating rigid objects*, there are additional details to address. If you bounce an object that is spinning on the ground, you will notice that the object not only starts to move back upward, but *its angular velocity will normally change too*.**
- **We need to understand how the collision affects both linear and angular velocities.**

Impulsive Torque

- **In the real world at the moment of collision, the object will deform at the point of collision. This causes a compression force to push in the direction shown (of collision normal).**



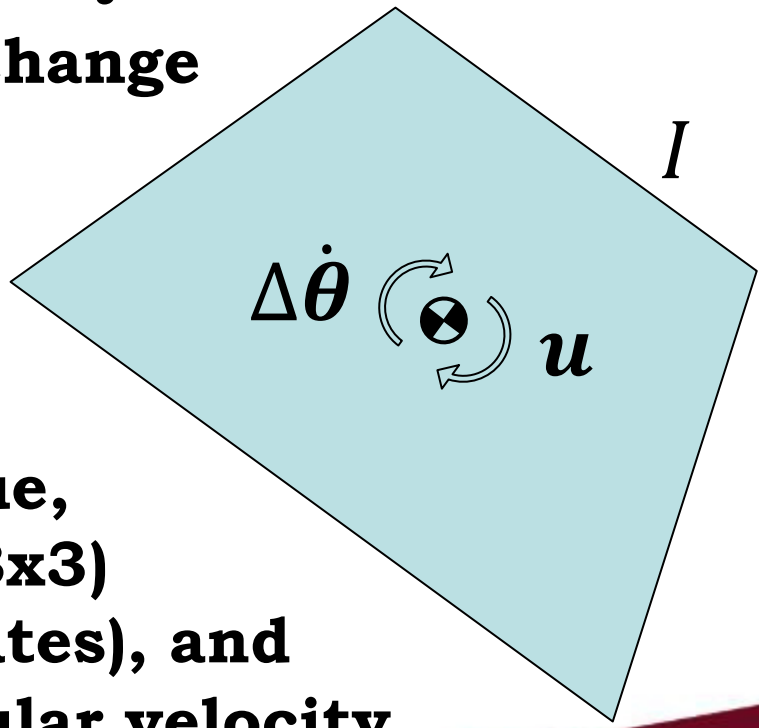
- **Physics: any force *f* acting on an object generates both linear and angular acceleration.**



Impulsive Torque

- A collision (and resulting impulse) will generate:
  - a linear change in velocity, and
  - an angular change in velocity.
- An instantaneous angular change in velocity is caused by *an impulsive torque*:

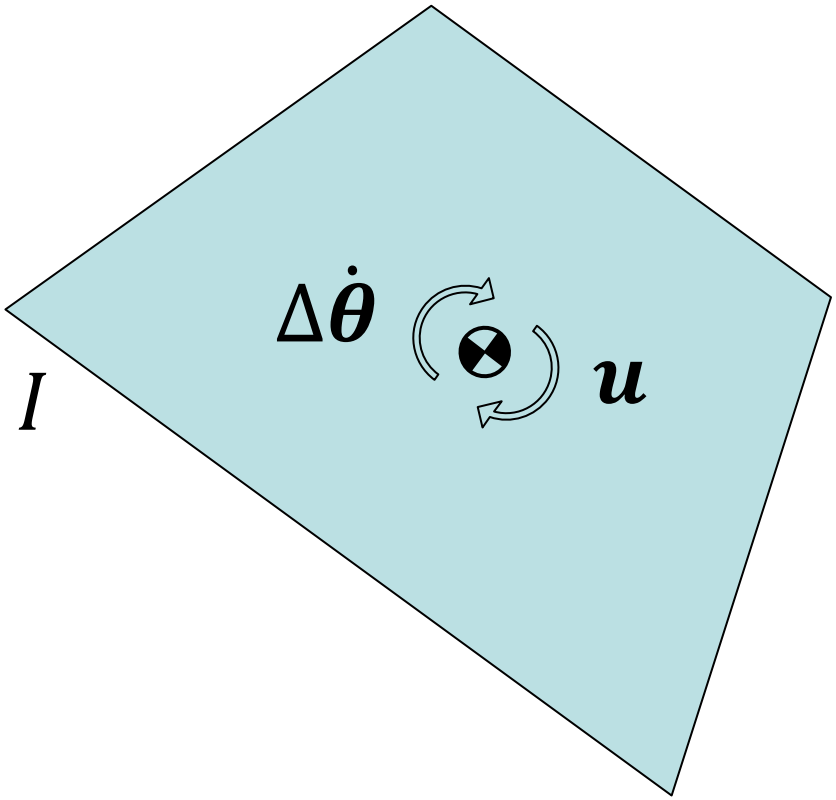
$$u = I(\Delta\dot{\theta})$$



where

u **is the impulsive torque,**
I **is the inertia tensor (3x3) (in world coordinates), and**
$\Delta\dot{\theta}$ **is the change in angular velocity.**

Impulsive Torque



Correspondingly *change in angular velocity* $\Delta\dot{\theta}$ is

$$\Delta\dot{\boldsymbol{\theta}} = I^{-1}\boldsymbol{u}$$

Impulsive Torque

- The *impulse torque generated by an impulse g* is given by

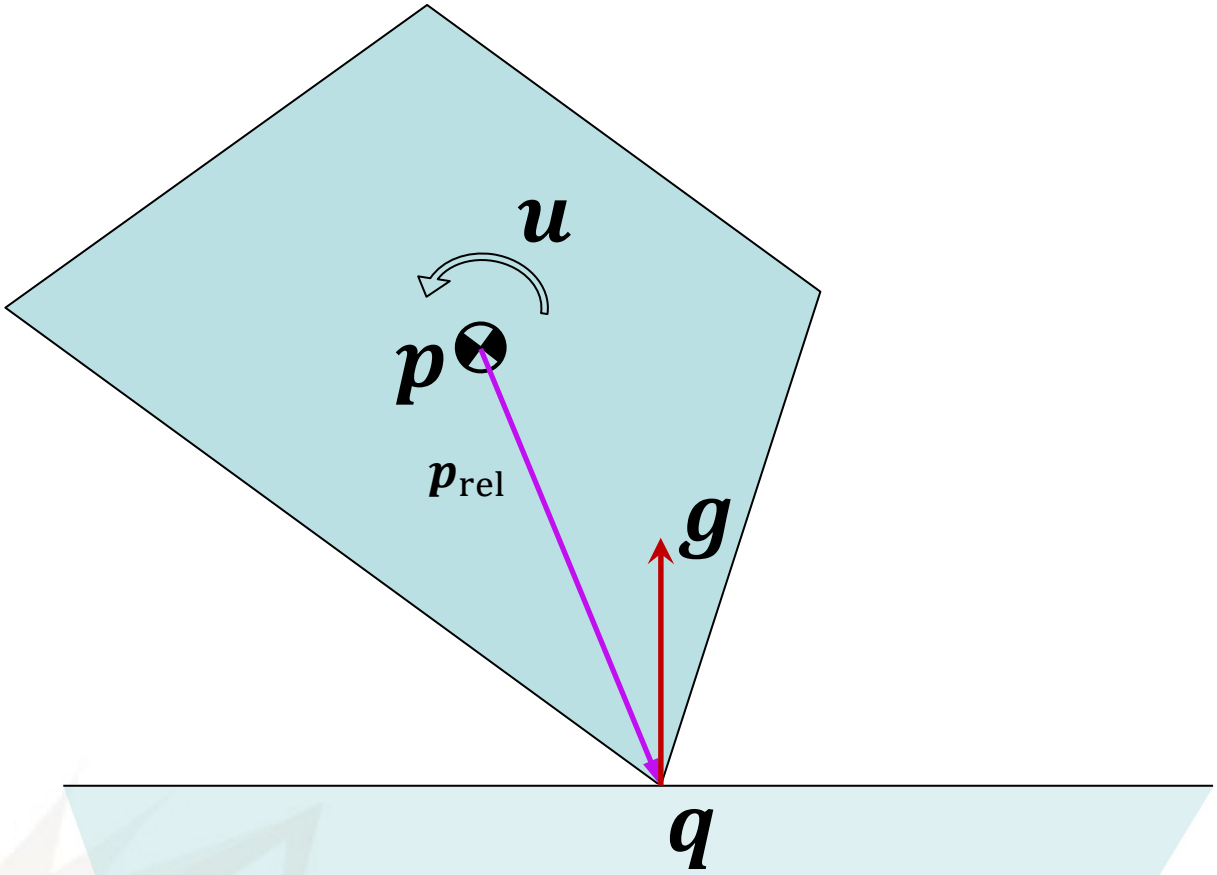
$$u = p_{\text{rel}} \times g$$

- In our case, for collisions the point of application (p_{rel}) is given by the contact point and the origin of the object:

$$p_{\text{rel}} = q - p$$

where q is the position of the contact (*i.e.*, contact point) in world coordinates and p is the *position of the origin of the object* in world coordinates.

Impulsive Torque



Rotating Collisions

- Recall that if we tracked the two collision points (one from each object) around the time of the collision, we'd see that their ***separating velocity*** is given by

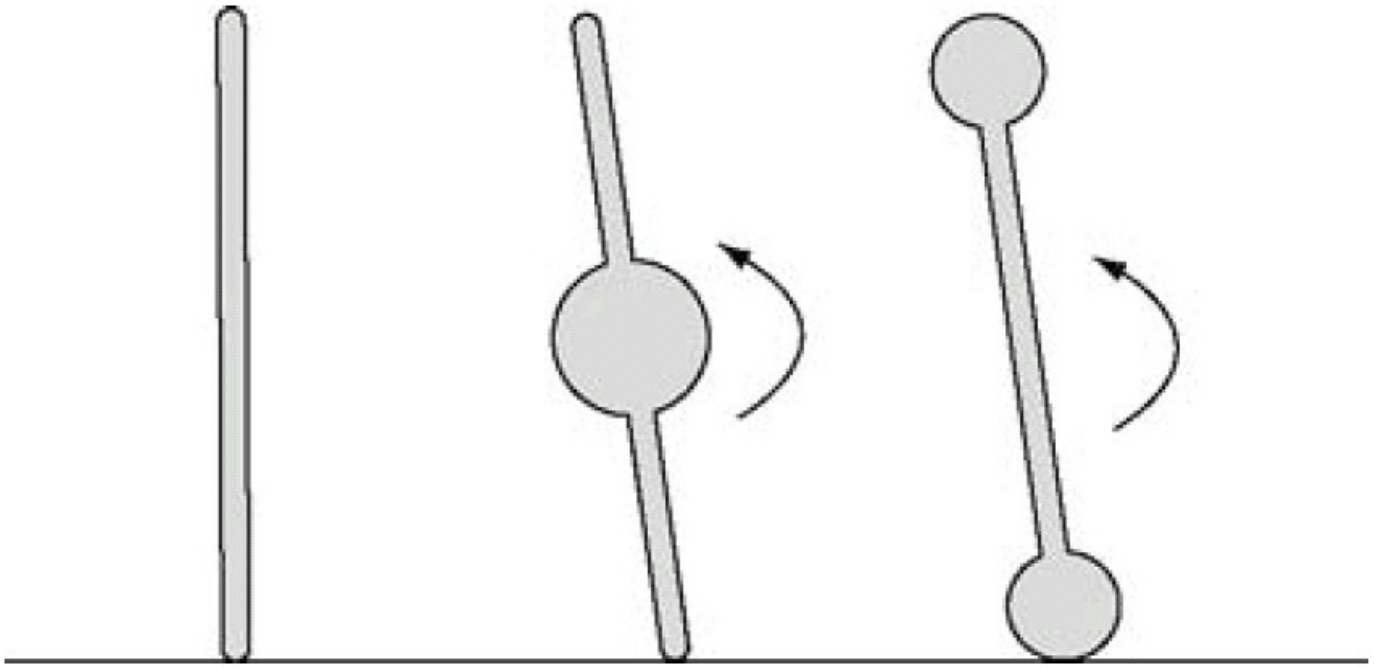
$$v_s' = -cv_s$$

where v_s is relative velocity of the objects immediately before the collision, v_s' is relative velocity after the collision, and c is coefficient of restitution.

- Depending on the ***characteristics of the objects involved***, and ***direction of the contact normal***, this separation velocity will be made up of a ***different degree of linear and rotational motion***.

Rotating Collisions

- **Below are *different objects engaged in the same collision* with an unmoving ground:**



negligible rotation ***strong rotation*** ***very small rotation***

Collision Impulse

- **To resolve the relative motion of the two objects *we need to calculate four components resulting from the impulse*:**

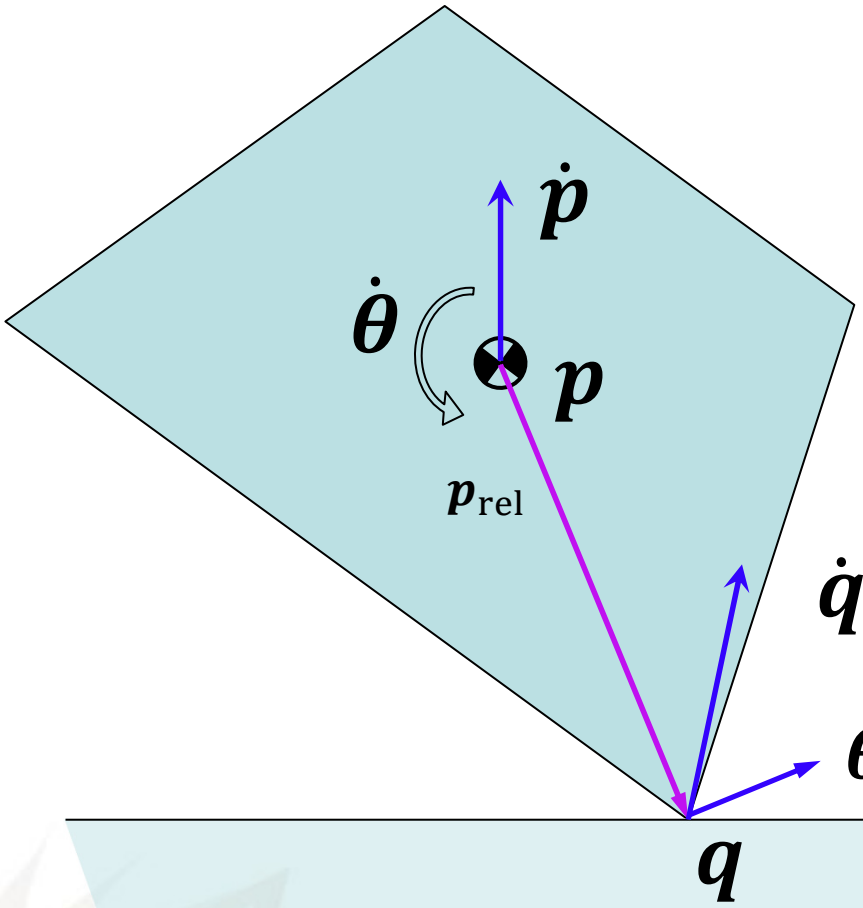
  ***linear and angular components for each object*.**

- **Physics: The velocity of a point q on an object with its center of gravity p is related to both its linear and angular velocity by**

$$\dot{q} = \dot{\theta} \times (q - p) + \dot{p} \qquad \textbf{(A)}$$

- **Because we are only interested in the *movement of the colliding points* at this stage, we can simplify the mathematics by doing calculations relative to the point of collision.**

Collision Impulse



$$\dot{q} = \dot{\theta} \times p_{\text{rel}} + \dot{p}$$

$$\dot{\theta} \times p_{\text{rel}}$$

Velocity Change

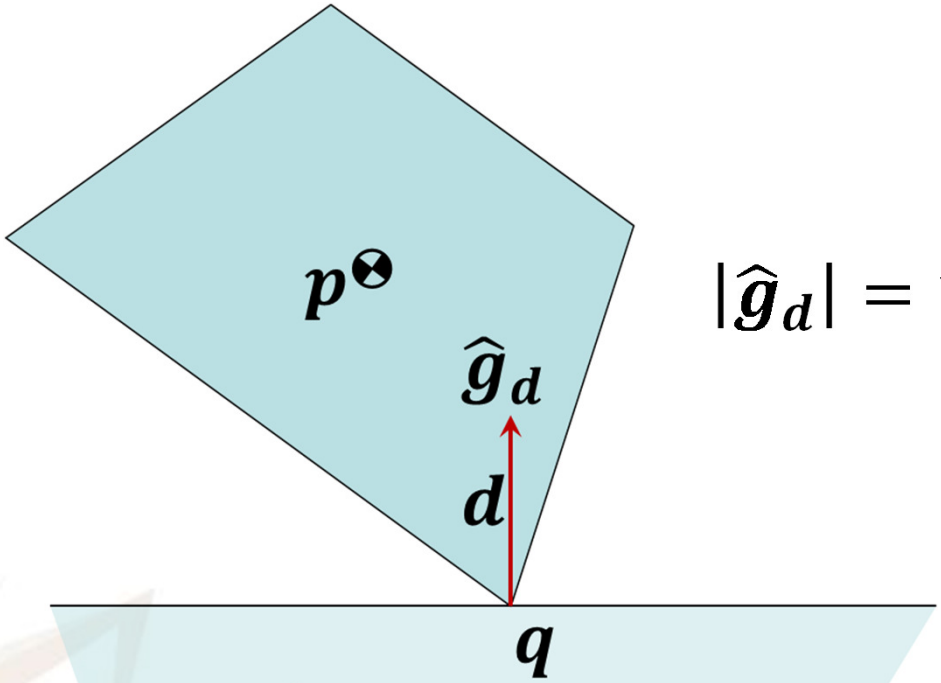
By Impulse

- **Impulses cause a change in velocity (with *both angular and linear components*).**
- **So, if our goal is to calculate the impulse at the collision, we need to understand *what effect an impulse will have on each object*.**
- **We can deal with the angular and linear components of the velocity change separately and combine them at the end.**
- **We will need to *find the linear and angular velocity change for each object* involved in the collision.**

Velocity Change

By Unit Impulse

- To start, we will consider the effect of a **unit size impulse**, $\hat{g}_d$, along the contact normal d …



$$|\hat{g}_d| = 1$$

Linear Component (Initial)

- **Linear component is very simple. *Linear change in velocity* for a unit impulse will be in direction of the contact normal *d*, with *a magnitude given by the inverse mass* (from $\hat{g}_d = m\Delta\dot{p}_d$):**

$$\Delta\dot{p}_d = m^{-1}$$

- **For *collisions involving two objects*, the linear component is *simply sum of two inverse masses* (along the contact normal between q_1 & q_2):**

$$\Delta\dot{p}_d = m_1^{-1} + m_2^{-1}$$

- **Remember that this equation holds only for the linear component of velocity—not the complete picture yet!**

Angular Component

(Initial)

- **The angular component is more complex. We'll need to bring together 3 equations:**

$$q_{\text{rel}} = q - p \quad \textbf{(position of relative contact)}$$
$$u = q_{\text{rel}} \times \hat{g}_d = q_{\text{rel}} \times \hat{d}$$
$$\Delta\dot{\theta} = I^{-1}u$$

where, as before, $\hat{g}_d$ *is the normalized impulse* in the *direction of the contact normal* $\hat{d}$.

- **The rotational component of the total velocity change as given in equation (A):** $\Delta\dot{q} = \Delta\dot{\theta} \times q_{\text{rel}}$
- **Result will be *velocity change* caused by rotation resulting from unit impulse along the contact normal.**

Angular Component (Initial)

- The result is a vector: it is a velocity in world space, in any direction. We are only interested in the velocity along the contact normal.
- There are two ways to compute this:
  1. *Transform the vector from world space coordinates to contact space coordinates* (*assuming x-coordinate is the direction of the contact normal*) and *keep just x component*.
  2. Compute the *dot product between vector and the contact normal* (in world coordinates):

$$\Delta\dot{q}_d = \Delta\dot{q} \cdot \hat{d}$$

Putting It Together

- For *contacts with two objects* involved we have four values: the velocity caused by linear motion and by angular motion for each object.
- For contacts where only *one rigid body* is involved, we have just two values for that body.
- In both cases we add resulting values together to get an overall change in velocity per unit impulse value, $\Delta\dot{q}_t = \Delta\dot{p}_d + \Delta\dot{q}_d$ (linear + angular)
- Then the size of the impulse needed to achieve a given velocity change is $g = \Delta v_s / \Delta\dot{q}_t$ where
  - Δv_s is the *desired change in velocity* and
  - g is the *impulse required along the contact normal*.

Desired Velocity Change

- This stage of the algorithm has two parts.
- First we need to *calculate the current closing velocity at the contact point*.
  - Add linear and angular velocity components of bodies together using equation (A). This gives a total closing velocity in world coordinates. *It is then transformed to contact coordinates*.
- Second we need to calculate the exact change in velocity (*in contact normal's direction*, to have a final velocity of v_s') we are looking for.

$$v_s' = -cv_s \Rightarrow \Delta v_s = -v_s - cv_s = -(1+c)v_s$$

Calculating the Impulse

- With the desired velocity change in hand, the impulse is given by $g = \Delta v_s / \Delta \dot{q}_t$
- Because we are not concerned with friction, we are only concerned with the *impulse in the direction of the contact normal*. In contact coordinates the contact normal is the x-axis, so the final impulse vector is

$$g_{\text{contact}} = \begin{bmatrix} g \\ 0 \\ 0 \end{bmatrix}$$

- This is then converted back into world coordinates:

$$g_{\text{world}} = M g_{\text{contact}}$$

Applying the Impulse

- **When applying the impulse, we *change both the linear and angular velocities*:**
  - **Change in linear velocity:** $\Delta\dot{\boldsymbol{p}} = \frac{\boldsymbol{g}}{m}$ *in world coordinates*
  - **Change in angular velocity:**

$$\Delta\dot{\boldsymbol{\theta}} = I^{-1}\boldsymbol{u} = I^{-1}(\boldsymbol{q}_{\text{rel}} \times \boldsymbol{g})$$

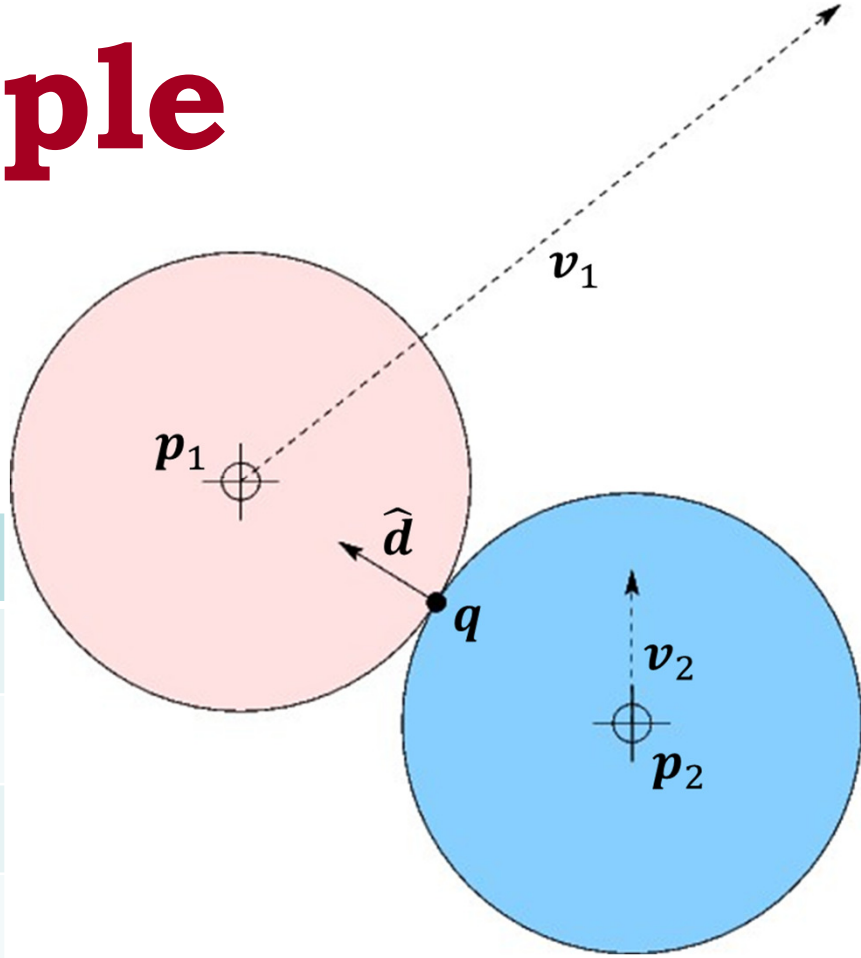
(in quaternions)

- **If two objects are involved, *both objects involved in a collision will receive the same sized impulse*, but in opposite directions. So, before applying the above calculations to the second body, *we change the sign of the impulse: g = –g***

Example

- ***Consider the collision shown with the following data:***

Data	Value
p_1	(21.7832, 26.3132, 0)
p_2	(32, 20, 0)
v_1	(3.9124, 3.1134, 0)
v_2	(0, 1, 0)
q	(26.8916, 23.1566, 0)
$\hat{d}$	(-0.8507, 0.5257, 0)
c	0.8



Example Continued…

- **First we need to *determine the required change in velocity* (in contact normal direction), Δv_s**
- **The velocities of two spheres along the contact normal direction are:**

$$v_1 \cdot \hat{d} = (3.9124, 3.1134, 0) \cdot (-0.8507, 0.5257, 0) \approx -1.6916$$

$$v_2 \cdot \hat{d} = (0, 1, 0) \cdot (-0.8507, 0.5257, 0) \approx 0.5257$$

- ***so that the collision velocity is***

$$v_c \approx 2.2173$$

or a separating velocity of $v_s \approx -2.2173$

- ***Then***

$$\Delta v_s = -(1+c)\, v_s = -(1+0.8)(-2.2173) \approx 3.9911$$

Example Continued…

- To continue, we also need *mass information for the spheres*:

	Sphere 1	*Sphere 2*
Type of Sphere	*Hollow, r = 9*	*Solid, r = 9*
Mass, m_i	10	20
3D moment of inertia tensor for the sphere, I_i	$\begin{bmatrix} 240 & 0 & 0 \\ 0 & 240 & 0 \\ 0 & 0 & 240 \end{bmatrix}$	$\begin{bmatrix} 288 & 0 & 0 \\ 0 & 288 & 0 \\ 0 & 0 & 288 \end{bmatrix}$
Inverse of moment of inertia tensor for the sphere, I_i^{-1}	$\begin{bmatrix} \frac{1}{240} & 0 & 0 \\ 0 & \frac{1}{240} & 0 \\ 0 & 0 & \frac{1}{240} \end{bmatrix}$	$\begin{bmatrix} \frac{1}{288} & 0 & 0 \\ 0 & \frac{1}{288} & 0 \\ 0 & 0 & \frac{1}{288} \end{bmatrix}$

Example Continued…

- **The *linear change in velocity* for *a unit impulse along the contact normal* is**

$$\Delta\dot{p}_d = m_1^{-1} + m_2^{-1} = \frac{1}{10} + \frac{1}{20} = \frac{3}{20} = 0.15$$

- **For the velocity change caused by rotation per unit impulse, we need:**
- **Sphere 1:**

$$\boldsymbol{q}_{\text{rel}} = \boldsymbol{q} - \boldsymbol{p} = \boldsymbol{q} - \boldsymbol{p}_1$$

$$= (26.8916, 23.1566, 0) - (21.7832, 26.3132, 0)$$

$$= (5.1084, -3.1566, 0)$$

Example Continued…

- **Sphere 1, continued:**

$$\boldsymbol{u} = \boldsymbol{q}_{\text{rel}} \times \hat{\boldsymbol{g}}_{\boldsymbol{d}} = \boldsymbol{q}_{\text{rel}} \times \hat{\boldsymbol{d}} = \begin{vmatrix} \hat{\imath} & \hat{\jmath} & \hat{k} \\ 5.1084 & -3.1566 & 0 \\ -0.8507 & 0.5257 & 0 \end{vmatrix}$$

$$= (0,0,5.1084 * 0.5257 - (3.1566) * (-0.8507))$$

$$= (0,0,0)$$

So, $\Delta\dot{\boldsymbol{\theta}} = I^{-1}\boldsymbol{u} = (0,0,0)$ **such that the rotational component of the total velocity change at** q **is:**

$$\Delta\dot{\boldsymbol{q}} = \Delta\dot{\boldsymbol{\theta}} \times \boldsymbol{q}_{\text{rel}} = (0,0,0)$$

Giving: $\Delta\dot{q}_{\boldsymbol{d}} = \Delta\dot{\boldsymbol{q}} \cdot \hat{\boldsymbol{d}} = 0$

- **Sphere 2: By the same computation:** $\Delta\dot{q}_{\boldsymbol{d}} = 0$

Example Continued…

- **Then** $\Delta\dot{q}_t = \Delta\dot{p}_d + \Delta\dot{q}_d = 0.15 + 0 = 0.15$ **such that**

$$g = \frac{\Delta v_s}{\Delta\dot{q}_t} = \frac{3.9911}{0.15} \approx 26.61$$

is the impulse along the contact normal. In *world coordinates*: $\boldsymbol{g} = g\hat{\boldsymbol{d}} = (-22.63, 13.99, 0)$

- **So the changes in linear velocities are:**
  - **Sphere 1**

$$(\Delta\dot{\boldsymbol{p}})_1 = \frac{\boldsymbol{g}}{m_1} = \frac{(-22.63, 13.99, 0)}{10} = \boldsymbol{(-2.263, 1.399, 0)}$$

  - **Sphere 2**

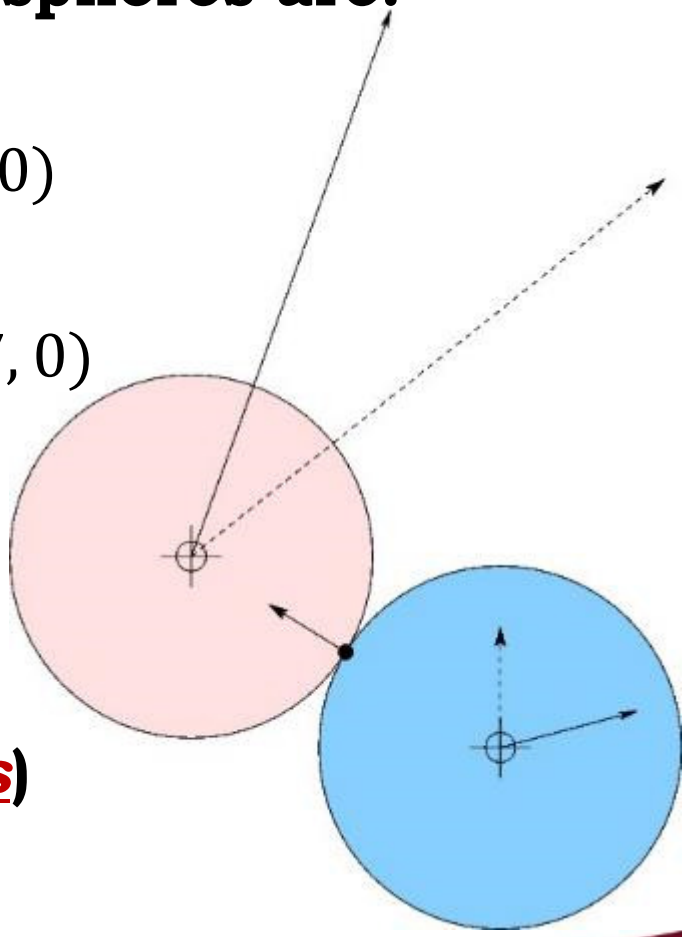
$$(\Delta\dot{\boldsymbol{p}})_2 = \frac{-\boldsymbol{g}}{m_2} = \frac{(22.63, -13.99, 0)}{20} = \boldsymbol{(1.317, -0.6993, 0)}$$

Example Continued…

- **So, the final velocities of the spheres are:**
  - **Sphere 1**

    $\boldsymbol{v}_1 = \boldsymbol{v}_1 + (\Delta\dot{\boldsymbol{p}})_1 = (1.649,\ 4.512, 0)$
  - **Sphere 2**

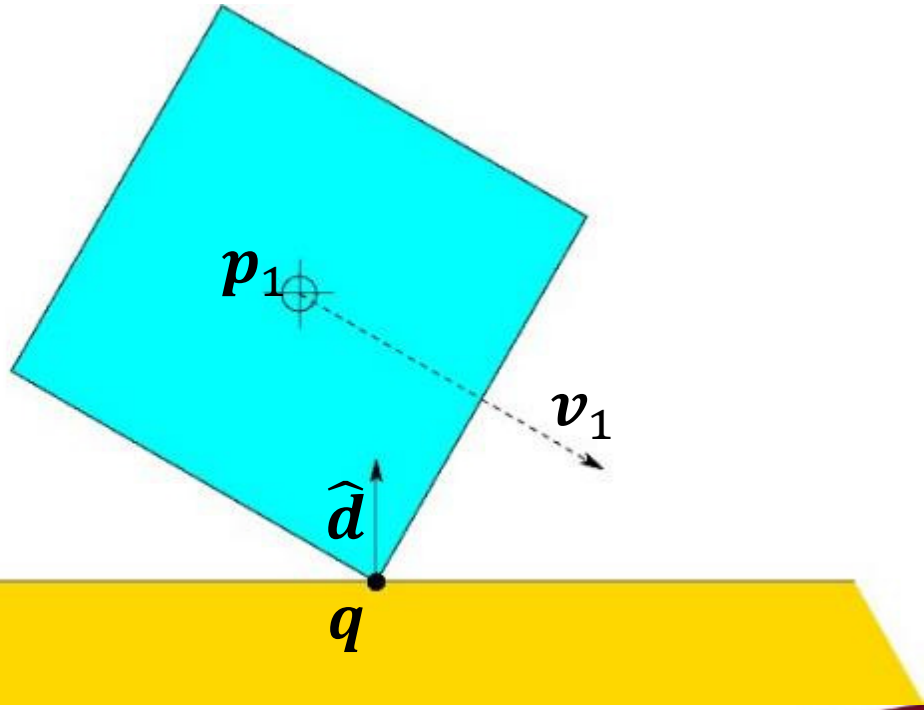
    $\boldsymbol{v}_2 = \boldsymbol{v}_2 + (\Delta\dot{\boldsymbol{p}})_2 = (1.132, 0.3007, 0)$
- ***Final velocities are shown as solid line*** **vectors in the figure (recall that there are *no changes to angular velocities*)**



Second Example

- Consider the collision shown with the following data:

Data	Value
p_1	(20.1966, 16.1966, 0)
v_1	(2.165, -1.25, 0)
q	(22.3932, 8, 0)
$\hat{d}$	(0, 1, 0)
c	0.8



- Object 2 is an immovable object, *e.g.* a wall or floor (consider its mass to be infinite)

Example Continued…

- **First we need to determine the *required change in velocity* (in contact normal direction),** Δv_s **.**
- **Velocities of the two objects along the contact normal direction are:**

$$v_1 \cdot \hat{d} = (2.165, -1.25, 0) \cdot (0,1,0) = -1.25$$

$$v_2 \cdot \hat{d} = (0,0,0) \cdot (0,1,0) = 0$$

so that the collision velocity is

$$v_c = 1.25$$

or a separating velocity of $v_s = -1.25$

- **Then**

$$\Delta v_s = -(1 + c)v_s = -(1 + 0.8)(-1.25) = 2.25$$

Example Continued…

- To continue, we also need ***mass information for the objects***:

	Cube	Floor
Type of Object	***Solid, sides = 12***	***Half-Space***
Mass, m_i	10	∞
3D moment of inertia tensor for the object, I_i	$\begin{bmatrix} 240 & 0 & 0 \\ 0 & 240 & 0 \\ 0 & 0 & 240 \end{bmatrix}$	$\begin{bmatrix} \infty & 0 & 0 \\ 0 & \infty & 0 \\ 0 & 0 & \infty \end{bmatrix}$
Inverse of moment of inertia tensor for the object, I_i^{-1}	$\begin{bmatrix} \frac{1}{240} & 0 & 0 \\ 0 & \frac{1}{240} & 0 \\ 0 & 0 & \frac{1}{240} \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$

Example Continued…

- **The linear change in velocity for a unit impulse *along the contact normal* is**

$$\Delta \dot{p}_d = m_1^{-1} + m_2^{-1} = \frac{1}{10} + \frac{1}{\infty} = \frac{1}{10} = 0.1$$

- **For the velocity change *caused by rotation per unit impulse*, we need:**
- **Cube:**

$$\begin{aligned} \boldsymbol{q}_{\text{rel}} &= \boldsymbol{q} - \boldsymbol{p} = \boldsymbol{q} - \boldsymbol{p}_1 \\ &= (22.3932, 8, 0) - (20.1966, 16.1966, 0) \\ &= (2.1966, -8.1966, 0) \end{aligned}$$

Example Continued…

- **Cube, continued:**

$$\boldsymbol{u} = \boldsymbol{q}_{\text{rel}} \times \hat{\boldsymbol{g}}_{\boldsymbol{d}} = \boldsymbol{q}_{\text{rel}} \times \hat{\boldsymbol{d}} = \begin{vmatrix} \hat{\imath} & \hat{\jmath} & \hat{k} \\ 2.1966 & -8.1966 & 0 \\ 0 & 1 & 0 \end{vmatrix}$$

$$= (0,0,2.1966 * 1 - (0) * (-8.1966))$$

$$= \mathbf{(0, 0, 2.1966)}$$

$$\textbf{So, } \Delta\dot{\boldsymbol{\theta}} = I^{-1}\boldsymbol{u} = \begin{bmatrix} \frac{1}{240} & 0 & 0 \\ 0 & \frac{1}{240} & 0 \\ 0 & 0 & \frac{1}{240} \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 2.1966 \end{bmatrix}$$

$$= \mathbf{(0, 0, 0.009153)}$$

Example Continued…

- **Cube, continued:**
- **Then the *rotational component of the total velocity change* for Cube *for a unit impulse* is:**

$$\Delta\dot{q} = \Delta\dot{\theta} \times q_{\text{rel}} \approx (0.07502, 0.0201, 0)$$

Giving, along contact normal: $\Delta\dot{q}_d = \Delta\dot{q} \cdot \hat{d} = 0.0201$

- **Then** $\Delta\dot{q}_t = \Delta\dot{p}_d + \Delta\dot{q}_d = 0.1 + 0.0201 = 0.1201$ **such that the impulse along the contact normal is**

$$g = \frac{\Delta v_s}{\Delta\dot{q}_t} = \frac{2.25}{0.1201} \approx 18.7337.$$

In world coordinates: $g = g\hat{d} = (0, 18.7337, 0)$

Example Continued…

- So the ***change in linear velocity*** for Cube are:

$$(\Delta\dot{\boldsymbol{p}})_1 = \frac{\boldsymbol{g}}{m_1} = \frac{(0,18.7337,0)}{10} = (\mathbf{0, 1.8734, 0})$$

The final velocity: $\boldsymbol{v}_1 = \boldsymbol{v}_1 + (\Delta\dot{\boldsymbol{p}})_1 = (\mathbf{2.165, 0.6234, 0})$

- And ***change in rotational velocity*** for Cube is:

$$\boldsymbol{u} = \boldsymbol{q}_{\text{rel}} \times \boldsymbol{g} = \begin{vmatrix} \hat{\imath} & \hat{\jmath} & \hat{k} \\ 2.1966 & -8.1966 & 0 \\ 0 & 18.7337 & 0 \end{vmatrix} = (\mathbf{0, 0, 41.1505})$$

$$\Delta\dot{\boldsymbol{\theta}} = I^{-1}\boldsymbol{u} = (\mathbf{0, 0, 0.1715})$$

which roughly translates to about 20 degrees/sec in a counter-clockwise direction about the z-axis.

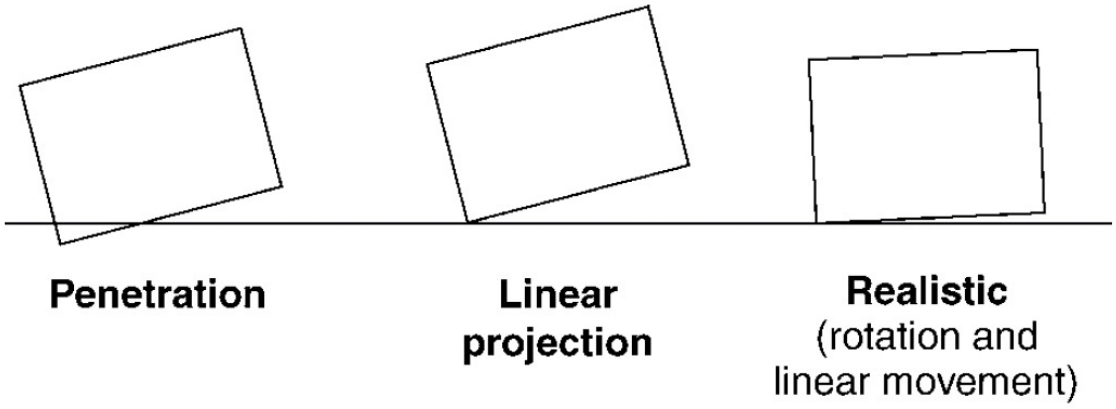
Resolving Interpenetration

- Unfortunately, since a simulation proceeds in time steps, the objects can pass into one another before we detect that a collision has occurred.
- We *need to resolve this interpenetration in some way*; otherwise *objects in the game will not appear solid*.
- If the bodies do not rotate, when two objects were interpenetrating, it was quite easy to move them apart. We *move each object back along the line of the contact normal* to the first point where they no longer intersected.

Choosing a Resolution Method

- For rotating rigid bodies the situation is a little more complex. There are several strategies we could employ to resolve interpenetration.

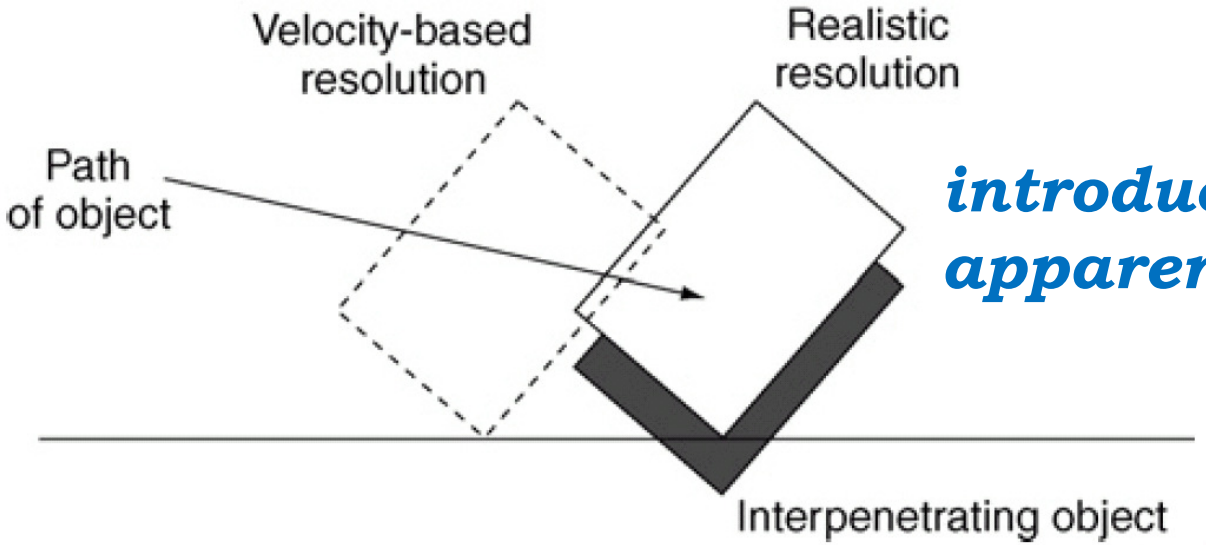
1. **Linear Projection**
   - As before: changing the position of each object so that it is moved apart in the direction of the contact normal. The ***amount of movement should be the smallest possible such that the objects no longer touch***.



Choosing a Resolution Method

2. Velocity-Based Resolution

- **At some point in their motion the two objects will have just touched. After that time they will continue interpenetrating to the end of the time step.**
- **To resolve the interpenetration *we could move them back to the point of first collision*.**



introduces apparent friction

Choosing a Resolution Method

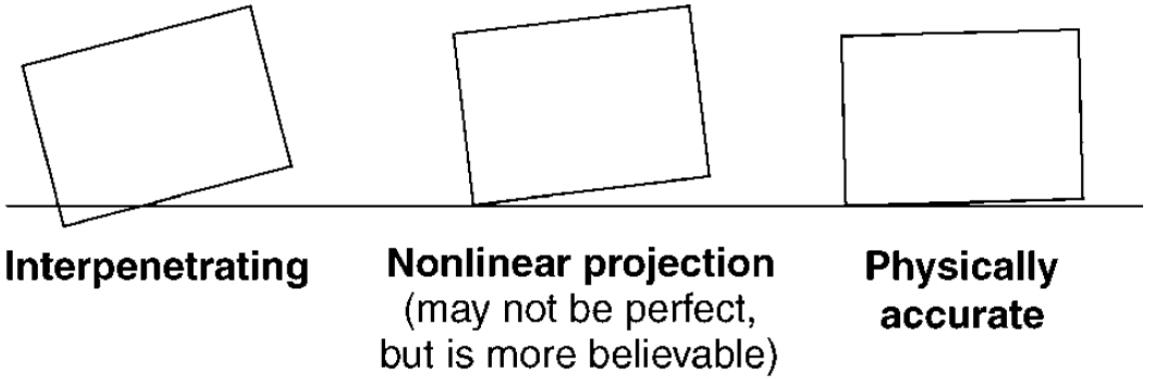
3. Nonlinear Projection

- Rather than just moving the objects back linearly, we *use a combination of linear and angular movement to resolve the penetration*.
- Again, we move both objects in the direction of the contact normal until they are no longer interpenetrating. *The movement, rather than being linear, can also have an angular component*.
- For each object in the collision we need to *calculate the amount of linear motion* and *the amount of angular motion* so the total effect is exactly enough to resolve the interpenetration.

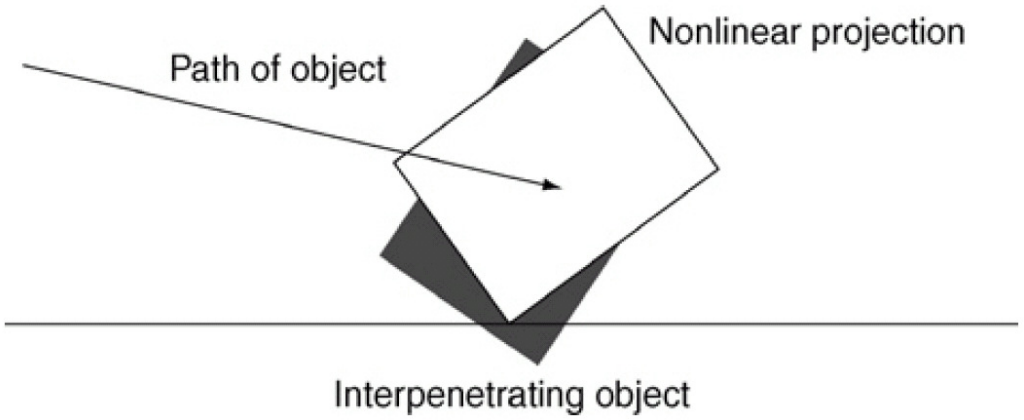
Choosing a Resolution Method

3. Nonlinear Projection

- is more believable



- does not add friction



Choosing a Resolution Method

4. Relaxation

- **Relaxation *resolves only a proportion of the interpenetration at one go*, and can be *used in combination with any other method* (most commonly nonlinear projection).**
- **Relaxation is useful when there are lots of contacts on one object. As each contact is resolved, it may move the object in such a way that other objects are now interpenetrated.**

https://www.freepik.com/premium-photo/background-texture-red-bricks-wall_5265436.htm

Nonlinear Projection

Implementation

- Start with the penetration depth of the contact, d_{pen}: this is the *total amount of movement we'll need to resolve the interpenetration*. **Goal**: *find the proportion of this movement contributed by the linear & angular motion for each object.*
- **Assumption**: we imagine the *objects* we *are* simulating were *pushed together so that they deformed*.
- Then the amount of interpenetration acts as the amount of deformation in both bodies. This causes a force that pushes the objects apart. This *force has both linear and angular effects*.

Nonlinear Projection

Implementation

- Recall that the *resistance of an object to being moved* is called its "inertia". So we want to find inertia of each object in the direction of contact normal $\hat{d}$ (against a unit distance change).
- This inertia has linear & rotational components.
- The linear component of inertia is, simply the inverse mass, $(I_l)_i = m_i^{-1}$, for object $i = 1, 2$
- The angular component is calculated using same sequence of operations that we used previously:

$$(I_a)_i = ((\Delta\boldsymbol{\theta})_i \times (\boldsymbol{q}_{\text{rel}})_i) \cdot \hat{\boldsymbol{d}}$$

where $(\Delta\boldsymbol{\theta})_i = I_i^{-1}\left((\boldsymbol{q}_{\text{rel}})_i \times \hat{\boldsymbol{d}}\right)$

Nonlinear Projection
Implementation

- Then the total inertia is

$$I_t = \sum_i [(I_l)_i + (I_a)_i]$$

- Thus the *amount of linear movement* is:

$$(\Delta_l)_i = (-1)^{i+1} d_{\text{pen}} \left(\frac{(I_l)_i}{I_t} \right)$$

This can be *applied directly to the centre(s) of mass* in the contact normal direction.

- And the amount of movement due to *angular motion* is:

$$(\Delta_a)_i = (-1)^{i+1} d_{\text{pen}} \left(\frac{(I_a)_i}{I_t} \right)$$

Nonlinear Projection

Implementation

- The angular motion is a little more difficult. We know the amount of linear movement we are looking for; we need to calculate the change in the orientation quaternion that will give it to us.
- We do this in three stages. First, we calculate the rotation needed to move the contact point by one unit.

$$\frac{(\Delta\boldsymbol{\theta})_i}{(\boldsymbol{I_a})_i}$$

- Second, we multiply this by the number of units needed (*i.e.*, the $(\Delta a)_i$ value) for the total rotation:

$$(\Delta\boldsymbol{\theta})_i \left(\frac{(\Delta_a)_i}{(I_a)_i}\right)$$

Nonlinear Projection

Implementation

- **Finally, we apply the rotation to the orientation quaternion (assuming object i has an initial orientation of θ_i, *e.g.*, θ_i = [1, 0, 0, 0]):**

$$\theta_i{}' = \theta_i + \left(\frac{1}{2}\right)\left[0, (\Delta\theta)_i \left(\frac{(\Delta_a)_i}{(I_a)_i}\right)\right]\theta_i$$

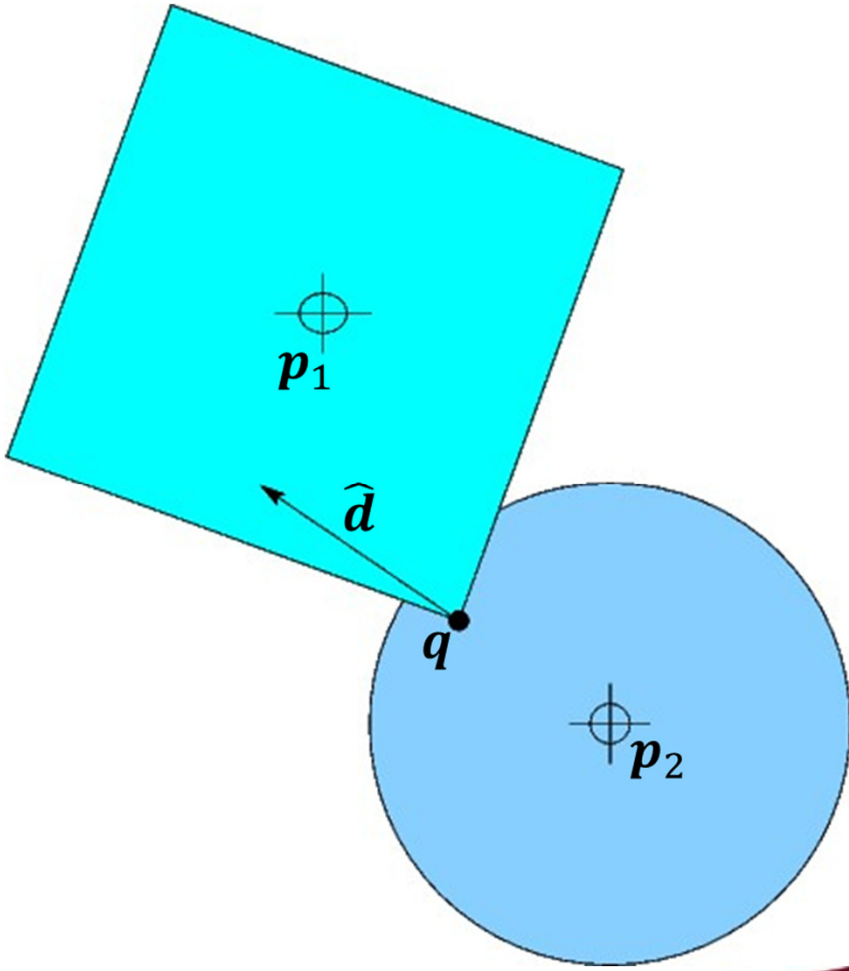
  using quaternion math (I use a shareware program called Calc 3D Pro to do the math).

- **You can then use a Java online applet for converting quaternions to Euler rotations (at bottom of page):**

  www.euclideanspace.com/maths/geometry/rotations/conversions/quaternionToEuler/

Example

- Consider the interpenetration shown with the following data:



Data	Value
p_1	(24.6392,30.2575, 0)
p_2	(32, 20, 0)
q	(28.2268,22.5668, 0)
$\hat{d}$	(-0.8268,0.5625, 0)
d_{pen}	1.4365

Example Continued…

- To continue, we also need mass information for the objects:

	Cube, $i = 1$	Sphere, $i = 2$
Type	Solid; sides=12	Solid; r = 6
Mass, m_i	10	100
3D moment of inertia tensor for the object, I_i	$\begin{bmatrix} 240 & 0 & 0 \\ 0 & 240 & 0 \\ 0 & 0 & 240 \end{bmatrix}$	$\begin{bmatrix} 1440 & 0 & 0 \\ 0 & 1440 & 0 \\ 0 & 0 & 1440 \end{bmatrix}$
Inverse of moment of inertia tensor for the object, I_i^{-1}	$\begin{bmatrix} \frac{1}{240} & 0 & 0 \\ 0 & \frac{1}{240} & 0 \\ 0 & 0 & \frac{1}{240} \end{bmatrix}$	$\begin{bmatrix} \frac{1}{1440} & 0 & 0 \\ 0 & \frac{1}{1440} & 0 \\ 0 & 0 & \frac{1}{1440} \end{bmatrix}$

Example Continued...

- **Linear components of the inertia:**

$$(I_l)_1 = m_1^{-1} = \frac{1}{10} = 0.1,$$

and

$$(I_l)_2 = m_2^{-1} = \frac{1}{100} = 0.01.$$

- **Angular components of the inertia:**
- **For Cube,**

$$(q_{\text{rel}})_1 = q - p_1 = (3.5876, -7.6907, 0)$$

such that

$$(q_{\text{rel}})_1 \times \hat{d} = (0, 0, -4.3409).$$

Example Continued…

- **For Cube, continued…**

such that

$$(\Delta\boldsymbol{\theta})_1 = \boldsymbol{I}_1^{-1}\left((\boldsymbol{q}_{rel})_1 \times \hat{\boldsymbol{d}}\right) = (0, 0, -0.01809),$$

and

$$(I_a)_1 = \left((\Delta\boldsymbol{\theta})_1 \times (\boldsymbol{q}_{rel})_1\right) \cdot \hat{\boldsymbol{d}}$$

$$= (-0.1391, -0.6489, 0) \cdot \hat{\boldsymbol{d}} = 0.07852.$$

- **For Sphere:**

$$(\boldsymbol{q}_{rel})_2 = \boldsymbol{q} - \boldsymbol{p}_2 = (-3.7732, 2.5668, 0)$$

such that $(\boldsymbol{q}_{rel})_2 \times \hat{\boldsymbol{d}} = (0, 0, 0)$. **Then** $(I_a)_2 = 0$. **So, there is no rotational component to Sphere's displacement.**

Example Continued…

- **The total inertia:**

$$I_t = \sum_i [(I_l)_i + (I_a)_i] = [0.1 + 0.01] + [0.07852 + 0]$$

$$= 0.1885$$

For Cube, the amount of linear movement is then

$$(\Delta_l)_1 = (-1)^{1+1} d_{\text{pen}} \left(\frac{(I_l)_1}{I_t} \right) = (1.4365) \left(\frac{0.1}{0.1885} \right)$$

$$= 0.762$$

This is along the contact normal, which in world coordinates is

$$0.762\hat{d} = 0.762(-0.8268, 0.5625, 0)$$

$$= (-0.63, 0.4268, 0).$$

Example Continued…

- For Sphere, the amount of linear movement is

$$(\Delta_l)_2 = (-1)^{2+1} d_{\text{pen}} \left(\frac{(I_l)_2}{I_t} \right) = -(1.4365) \left(\frac{0.01}{0.1885} \right)$$
$$= -0.0762$$

This is along the contact normal, which in world coordinates is

$$-0.0762\hat{d} = -0.0762(-0.8268, 0.5625, 0)$$
$$= (0.063, -0.04286, 0).$$

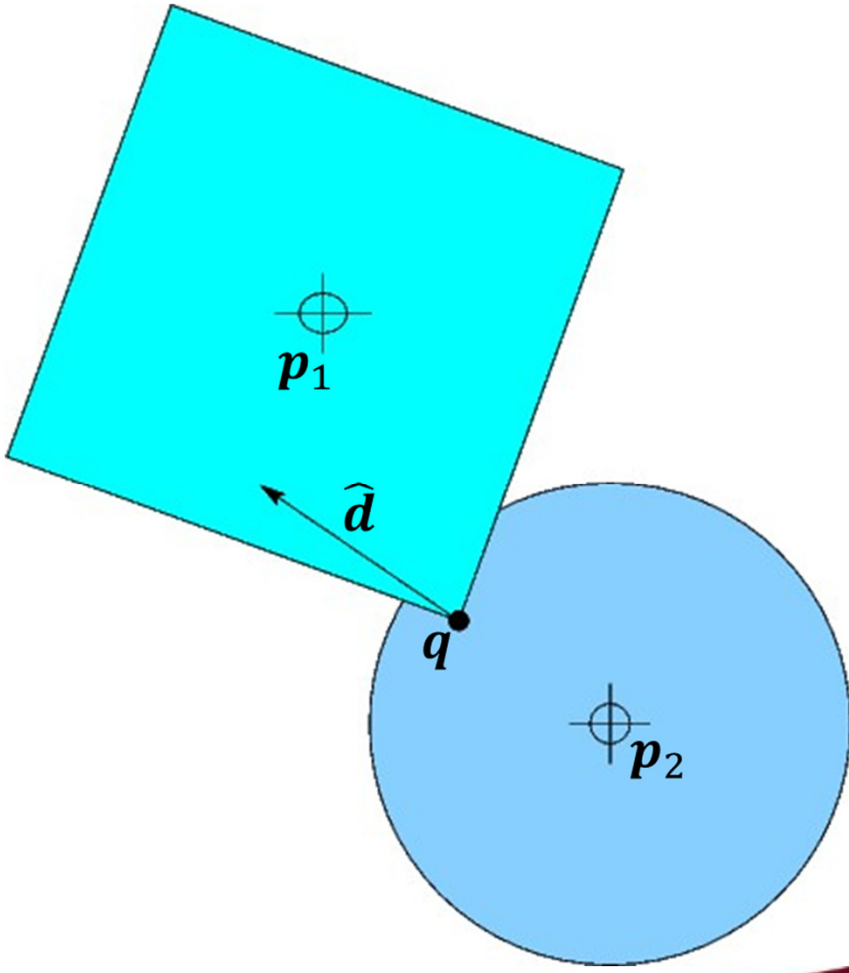
- Note that $(-1)^{2+1}$ results in a negative result which is due to an opposite direction motion being applied to the second object.

Example Continued…

- **Result from linear movement (only):**

$$\boldsymbol{p}_1 = \boldsymbol{p}_1 + (\Delta_l)_1 \hat{\boldsymbol{d}}$$
$$= (24.6392, 30.2575, 0) + (-0.63, 0.4268, 0)$$
$$= (22.6934, 31.5812, 0)$$

$$\boldsymbol{p}_2 = \boldsymbol{p}_2 + (\Delta_l)_2 \hat{\boldsymbol{d}}$$
$$= (32, 20, 0) + (0.063, -0.04286, 0)$$
$$= (32.063, 19.9571, 0)$$



Example Continued…

- **For Cube, the amount of linear movement due to angular motion is**

$$(\Delta_a)_1 = (-1)^{1+1} d_{\text{pen}} \left(\frac{(I_a)_1}{I_t} \right) = (1.4365) \left(\frac{0.07852}{0.1885} \right)$$

$$= 0.5983$$

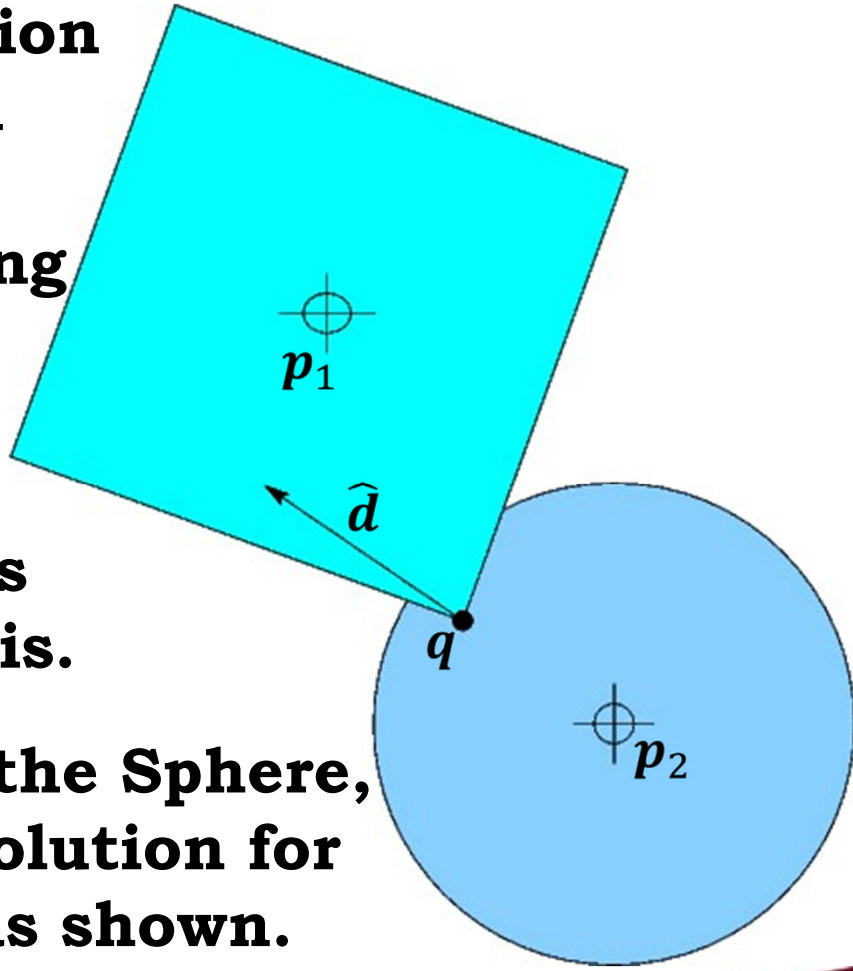
along the contact normal, which in world coordinates is a rotation of

$$(\Delta\theta)_1 \left(\frac{(\Delta_a)_1}{(I_a)_1} \right) = (0, 0, -0.01809) \left(\frac{0.5983}{0.07852} \right)$$

$$= (0, 0, -0.1378)$$

Example Continued…

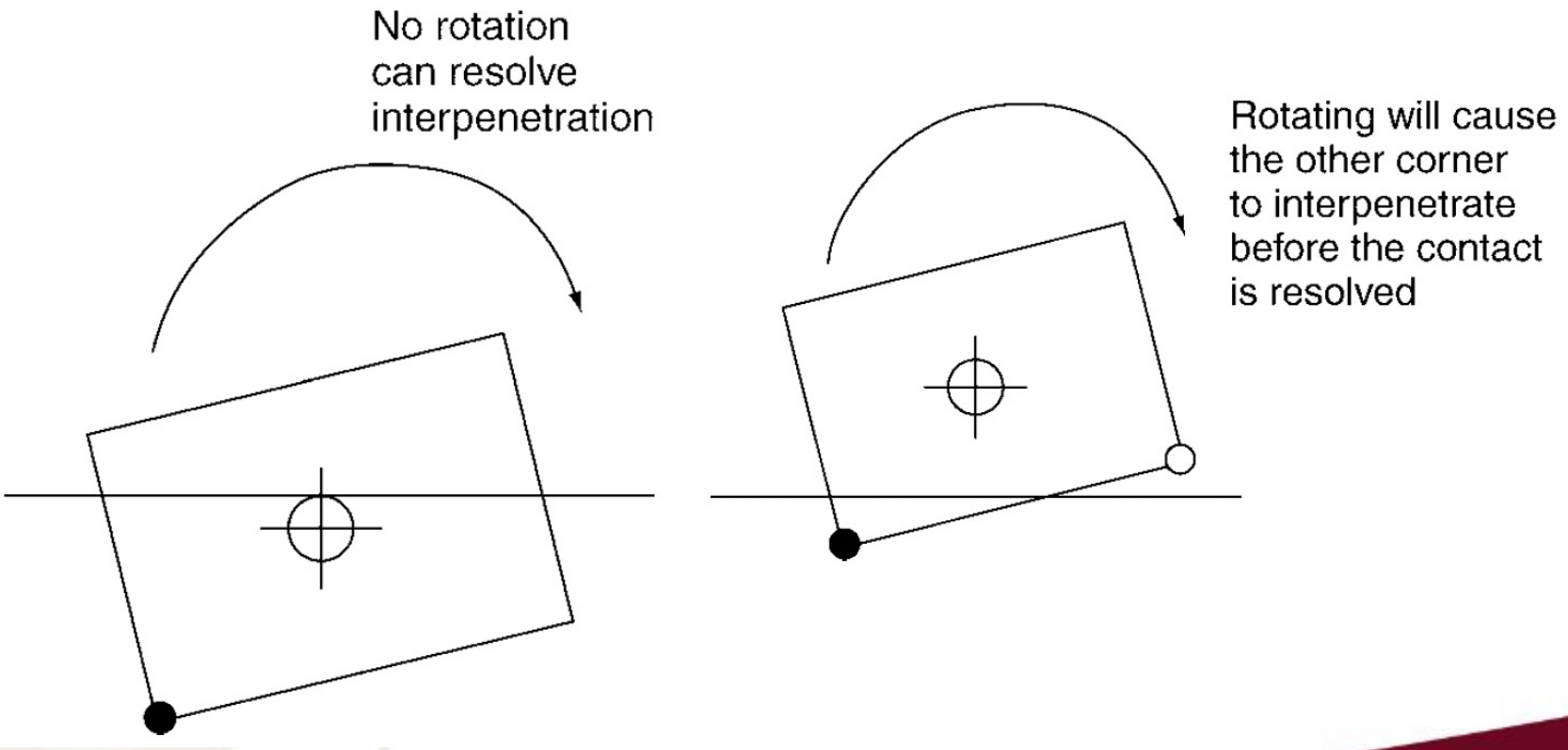
- After applying this rotation to the initial orientation quaternion for the Cube of [1,0,0,0] and converting the resulting quaternion ([1, 0, 0, −0.0689]) to Euler angles, this is a rotation of 7.922 degrees clockwise about the z-axis.



- There is no rotation for the Sphere, so the final collision resolution for the interpenetration is as shown.

Avoiding Excessive Rotation

- **Rotating an object too much as they are moved out of penetration can cause problems.**

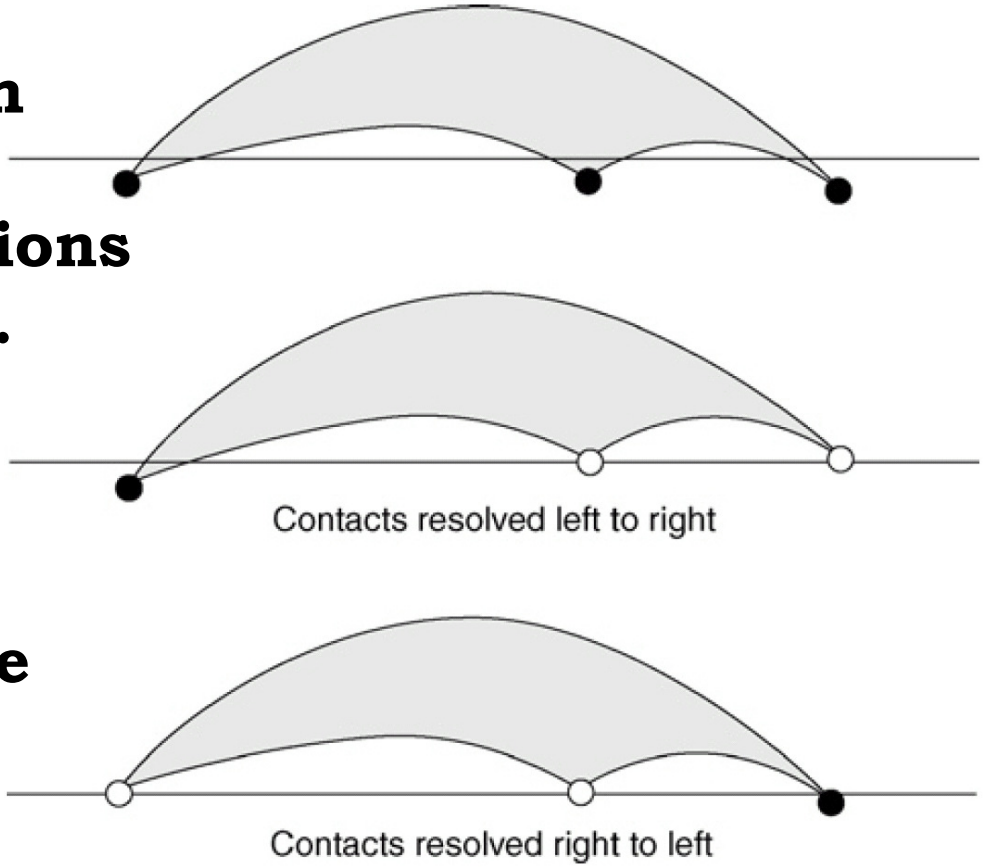


Avoiding Excessive Rotation

- **For both issues we need to limit the amount of rotation that can be part of our penetration resolution.**
- **We can simply check that values of $(\Delta_a)_i$ are not too great. If they are, we can transfer some of the burden from them onto the corresponding $(\Delta_a)_i$ component.**
- **What's "too great"? Could use a fixed amount, or a fraction of the revolution the object can make**
- **A simple alternative is to scale *the angular move by the size of the object* (approximated by the size of the relative contact position vector).**

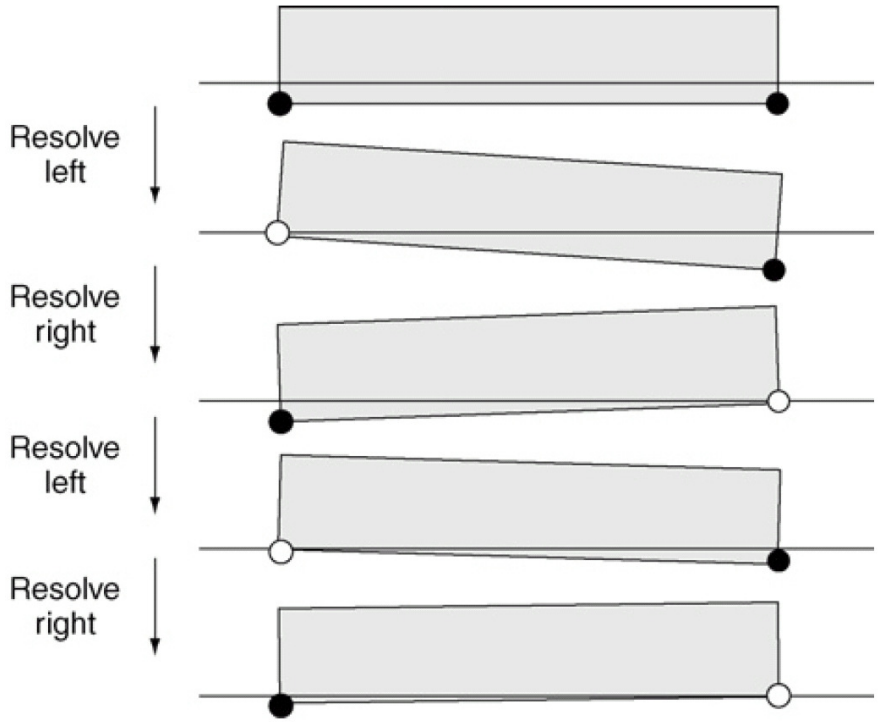
Resolving Penetration

- **We will take each interpenetration contact in turn and resolve it.**
  To avoid the problem shown in the figure, we resolve the collisions in penetration order.
- **At each iteration, we search through the contacts to find the collision with the *deepest penetration value*.**



Resolving Penetration

- The process is then repeated up to some maximum number of iterations (or until there are no more interpenetrations to resolve, whichever comes first).
- This algorithm can revisit the same contacts several times.
- To ensure that shallow contacts are handled: run through entire list once, then prioritize based on depth.



Updating Penetration Depths

- Moving an object out of penetration may cause another contact to disappear altogether, or bring new contacts that weren't expected before.
- Unfortunately collision detection is far too complex to be run for each iteration of the resolution algorithm. We need a faster way.
- There is an approximation we can use that gives good results: ***only these contacts that apply to the objects just resolved are updated*** based on the linear and angular movements of the interpenetration resolution.

OK, where are we at…?

- The resulting physics system at this point is quite usable. A simulation has no friction, so objects would slide across one another.
- For simple sets of objects it's likely to work fine. For more complex scenarios you may notice problems with vibrating objects/ slow performance.
- Next, we will address these three limitations. We'll look at the difference between collisions we have been dealing with so far and with ***resting contacts*** (this is part of the vibration problem). It also introduces ***friction***. Once friction is introduced, more ***stability*** problems become visible.

(§15)

Resting Contacts and Friction

Terminology

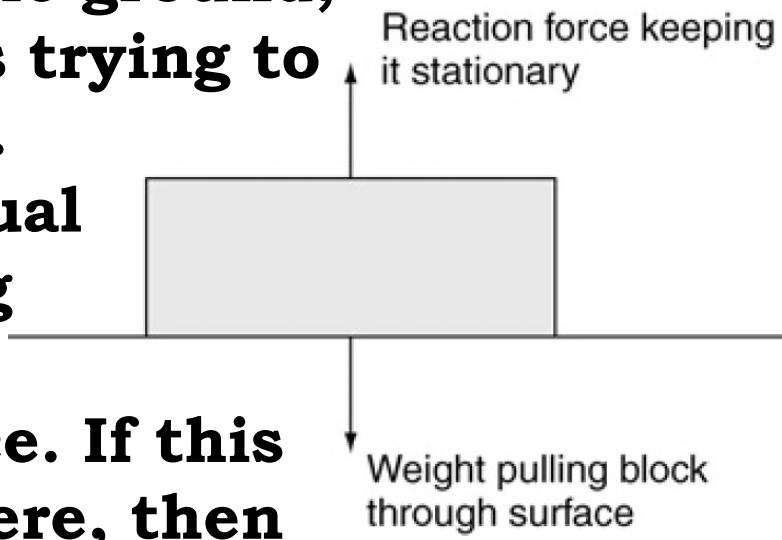
- Let us refine our terminology use a little:
  - A contact: any location in which objects are touching or interpenetrating;
  - A collision: a type of contact, one in which the objects are moving together at a speed (this is also called an "impact" in some physics systems);
  - A resting contact: When two objects are in contact for a longer than, say, a single physics update. The objects involved are moving neither apart nor together;
  - A separating contact: the objects involved are already moving apart.

Resting Forces

- **When an object is resting on another, Newton's third and final law of motion comes into play:**

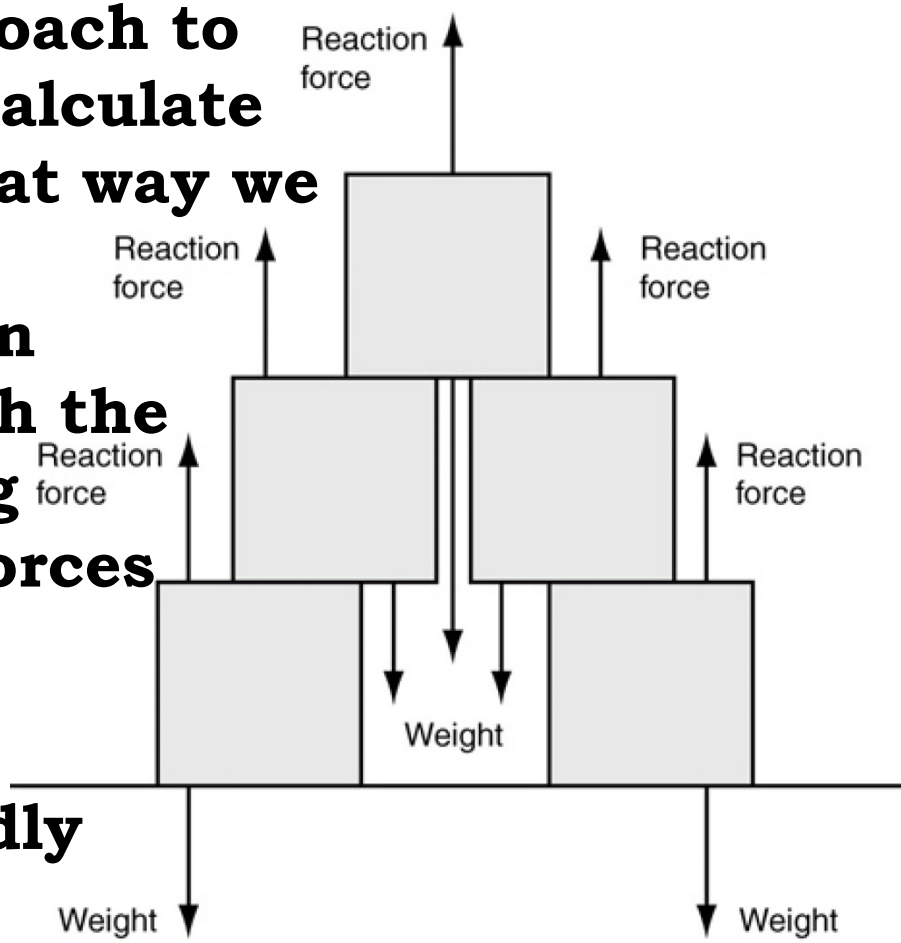
"For every action there is an equal and opposite reaction."

- **If an object is resting on the ground, then the force of gravity is trying to pull it through the ground. In addition, there is an equal and opposite force keeping the object on the ground. This is called reaction force. If this reaction force were not there, then the object would accelerate down through the ground.**



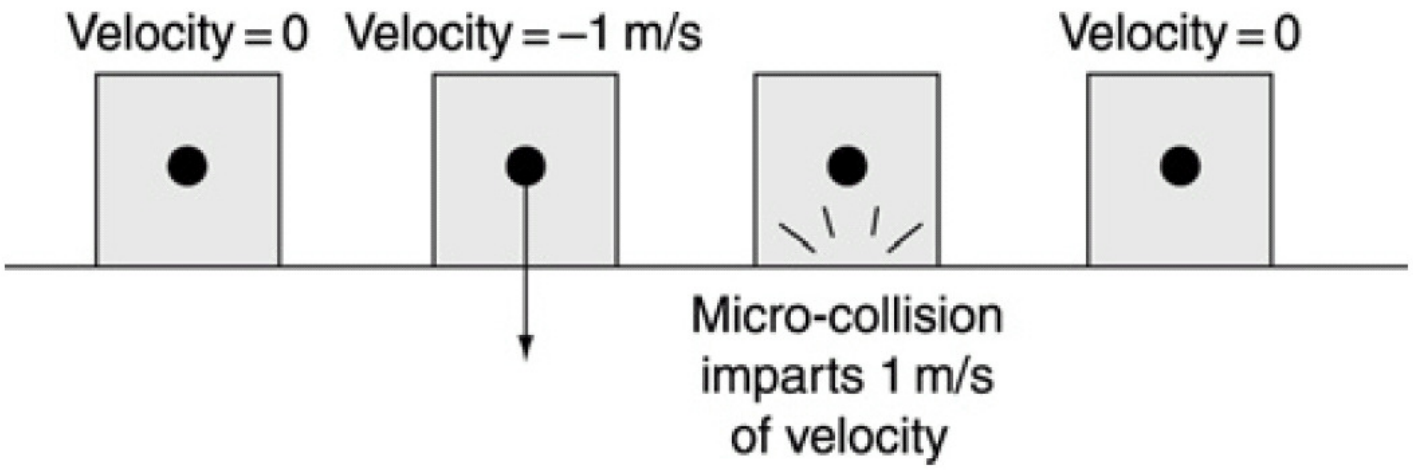
Force Calculations

- The most obvious approach to resting contacts is to calculate the reaction forces. That way we can add the forces into the equations of motion of our rigid bodies. With the reaction forces working alongside the regular forces we apply, the body will behave correctly.
- But this approach rapidly gets complex.



Micro-Collisions

- **Fortunately there is a much simpler (though slightly less accurate) solution. Rather than resolving all contacts using forces, we can treat resting contacts as if they were collisions.**
- **Micro-Collisions replace reaction forces by a series of impulses: one per update.**



Micro-Collisions

- There is a significant problem with treating resting contacts as collisions:
- It has to do with the way collisions bounce. Recall the final velocity is related to the separating velocity by the coefficient of restitution:

$$v_s' = -cv_s$$

- So whatever velocity built up over the course of the interval between updates will cause a little "bounce" to occur. This has the effect of making resting contacts appear to vibrate.

Micro-Collisions

- Setting a lower coefficient of restitution will help solve the vibration problem but limits the kinds of situations that can be modeled.
- A useful solution involves making two changes:
  - We remove any velocity that built up from acceleration in the previous rigid-body update.
  - We artificially decrease the coefficient of restitution for collisions involving very low speeds
- Independently each of these can solve the vibration problem for some simulations but will still show problems in others. Together they are about as good as we can get.

Remove Accelerated Velocity

- To remove the velocity due to the previous frame's acceleration, we need to ***keep track of the acceleration at each rigid-body update***.
  - Some developers choose to ignore any force except gravity when calculating the velocity added in the last frame.
- When we calculate the desired change in velocity for a contact, we subtract the acceleration-induced velocity, in the direction of the contact normal:

$$\Delta v = -v_{\text{acc}} - (1 + c)(v_s - v_{\text{acc}})$$

Lower the Restitution

- When the acceleration compensation alone doesn't work, we can manually lower the coefficient of restitution to discourage vibration. This can be done in a very simple way:

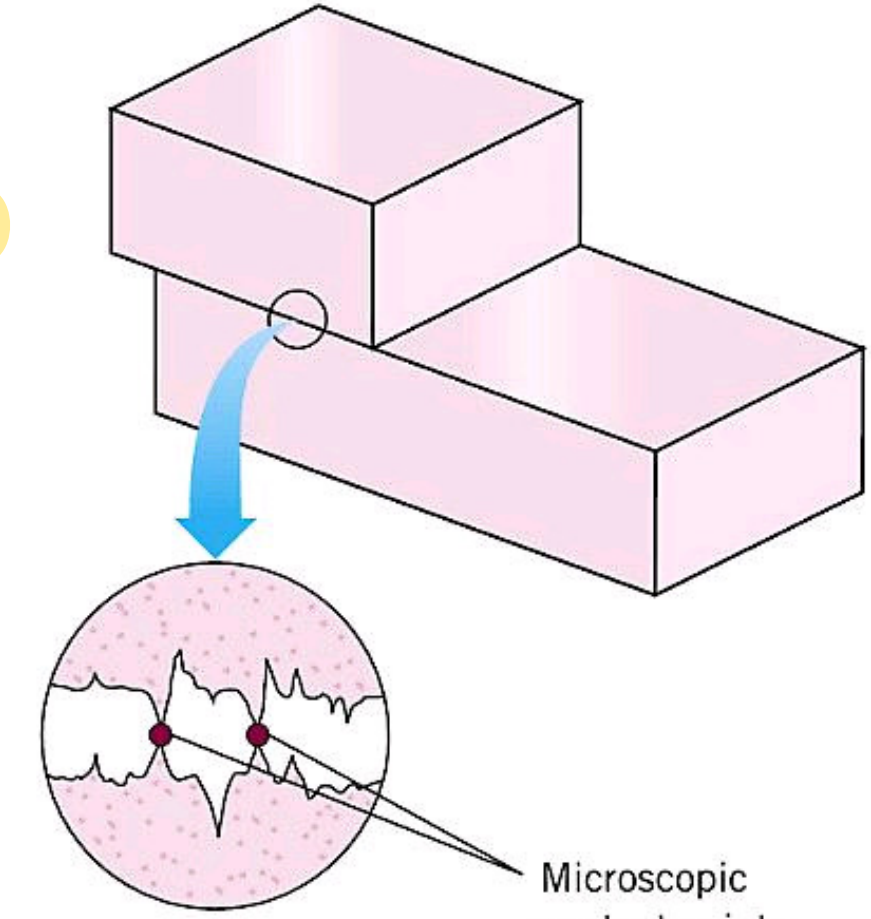
```
real appliedRestitution = restitution;
if (contactVelocity.magnitude() < velocityLimit)
{
    appliedRestitution = (real)0.0f;
}
```

- We could use a more sophisticated method, where the restitution is scaled so that it is smaller for smaller velocities, but the version here works quite well in practice.

Adding Friction

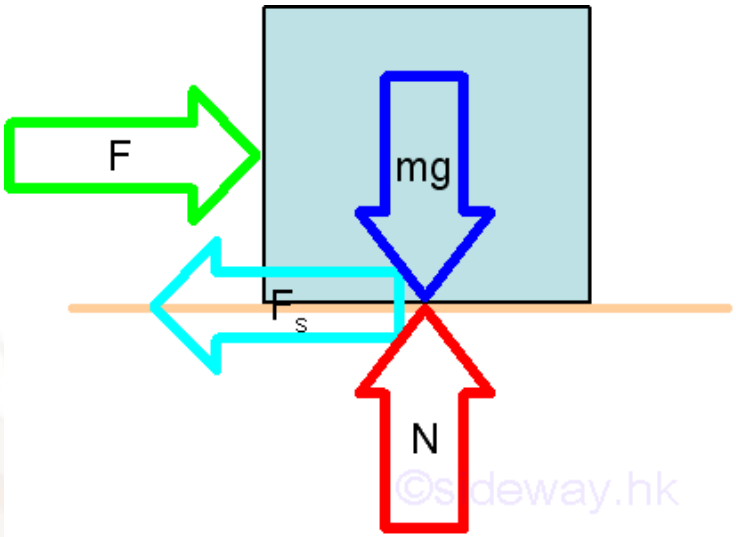
Types of Friction

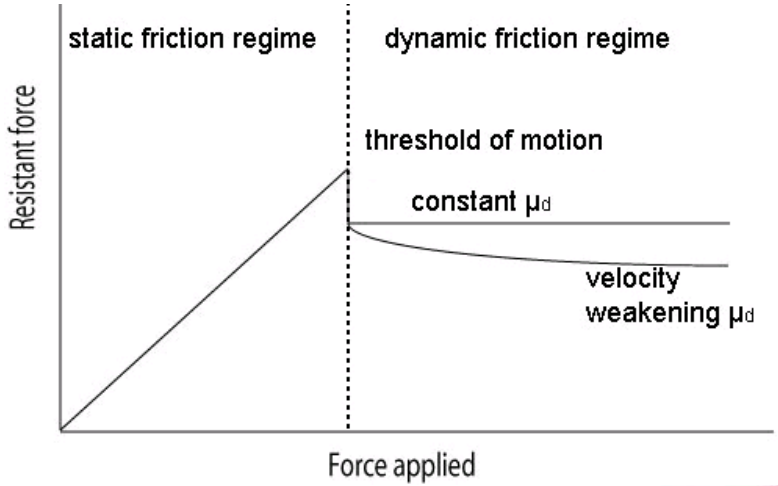
- **Friction**: the force generated when one object moves, or tries to move, in contact with another.
- There are two forms of friction:
- *static* and *dynamic*.
- They behave in slightly different ways.



Static Friction

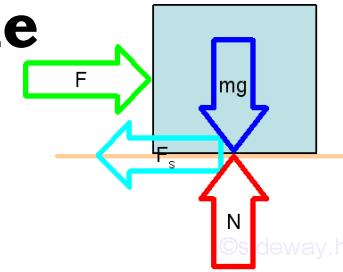
- **Static Friction: a force that stops an object from moving when it is stationary.**
- **This is a kind of reaction force: the more you push, the more friction pushes back. At some point, however, the pushing force is too much for friction, and the object begins to move.**





Static Friction

- The static friction force depends on the materials at the point of contact and the reaction force:



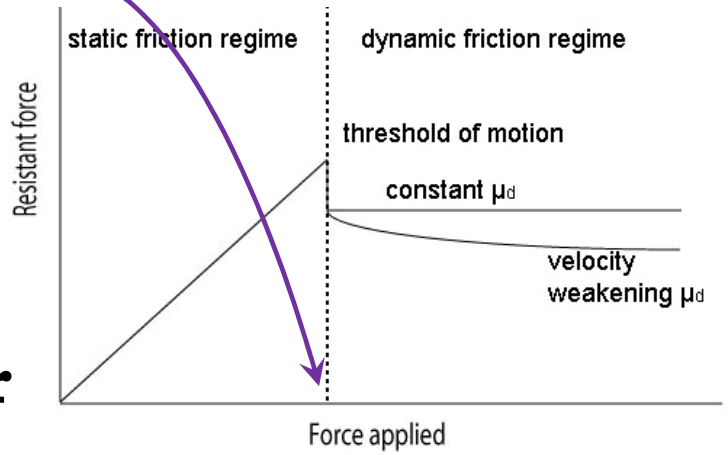
$$|f_{\text{static}}| \leq \mu_{\text{static}} |r|$$

  where r is the reaction force in the direction of the contact normal, f_{static} is the friction force, and μ_{static} is the coefficient of static friction.

- The coefficient of friction encapsulates all the material properties at the contact in a single number. The value depends on both objects; it cannot be generated simply by adding a coefficient for one object to one for another.

Static Friction

- Up to limit $\mu_{\text{static}}|r|$, magnitude of the friction force is exactly the same as force exerted on the object.
- So the overall expression for the static friction force is

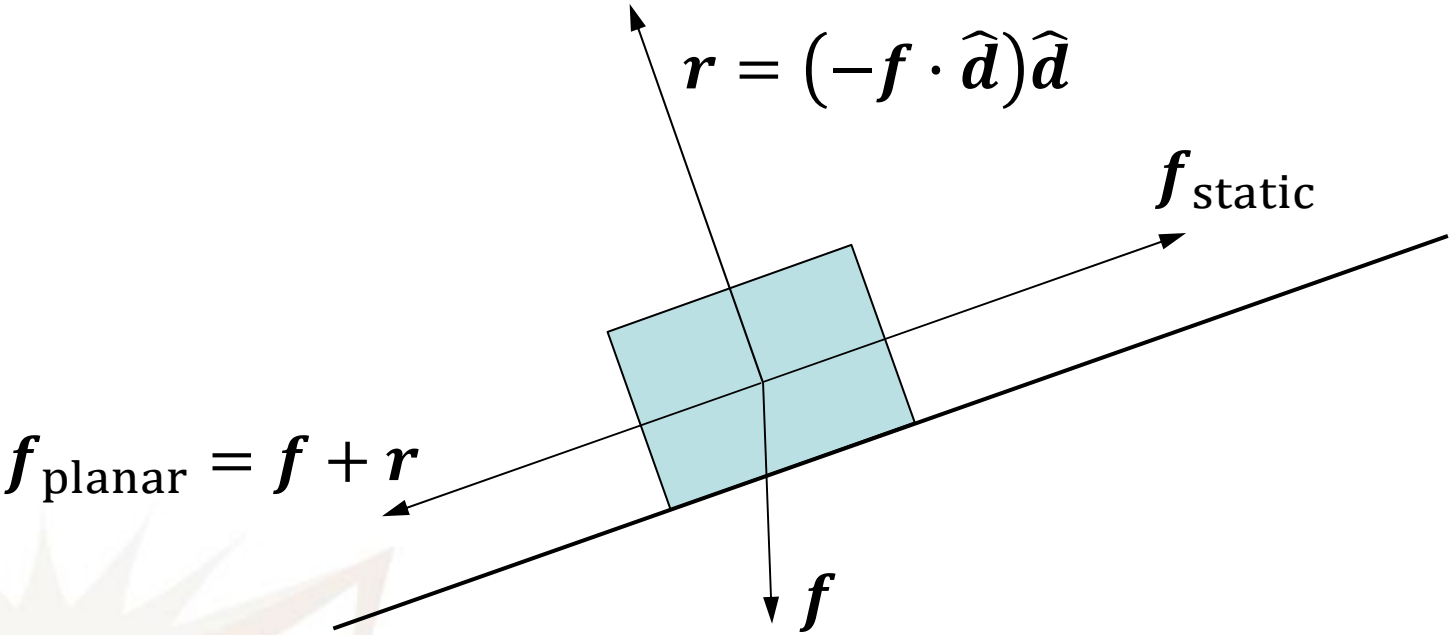


$$f_{\text{static}} = -f_{\text{planar}} \text{ if } |f_{\text{planar}}| \leq \mu_{\text{static}}|r|$$

where f_{planar} is the total force on the object in the plane of the contact only.

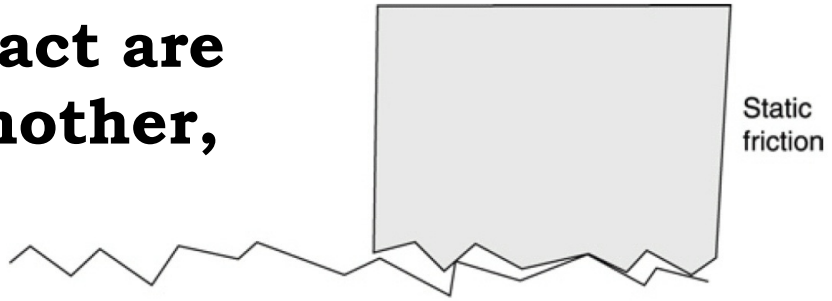
Static Friction

- If f is the total force exerted, then
- $r = (-f \cdot \hat{d})\hat{d}$ is the reaction force, and
- $f_{\text{planar}} = f + r$



Dynamic Friction

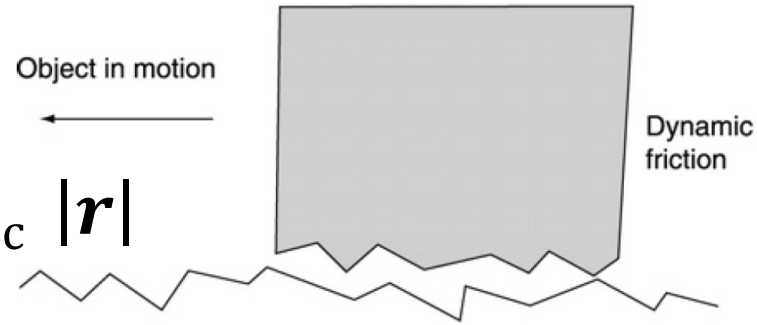
- **Dynamic friction, also called "kinetic friction", behaves in a similar way to static friction but has a different coefficient of friction.**
- **When objects at the contact are moving relative to one another, they are typically leaving contact at the microscopic level.**



- **Dynamic friction always obeys the equation**

$$f_{\text{dynamic}} = -\hat{v}_{\text{planar}}\,\mu_{\text{dynamic}}\,|r|$$

coefficient of dynamic friction



Friction as Impulses

- **Static friction stops a body from moving when a force is applied to it. It acts to keep the velocity of the object at zero in the contact plane.**
- **In our simulation we allow velocity to build up, and then we remove it with a micro-collision.**
- **We can simulate static friction by removing sliding velocity along the two contact directions (*i.e.*, the directions that are in the plane of the contact).**
- **This would give the effect of static friction.**

Friction as Impulses

- **This approach would remove all sliding. But static friction has a limit: it can only prevent objects from sliding up to a maximum force.**
- **When dealing with collision, we don't have any forces, only velocities. *How do we decide the max amount of velocity that can be removed?***
- **Recall that the change in velocity is related to impulse:**

$$\Delta \dot{\boldsymbol{p}} = m^{-1}\boldsymbol{g}$$

- **If we know the amount of velocity we need to remove, *we can calculate the impulse required to remove it (how? We'll get there...).***

Friction as Impulses

- In the same way, impulse is a force exerted over a short period of time:

$$g = ft$$

where f is force and t is time.

- The equation requires a ***normal reaction force***.

  ⇨ calculate from the amount of velocity removed in the direction of the contact normal.

- If the desired change in velocity in the contact normal direction is Δv, then the reaction force can be approximated as

$$r = \frac{(\Delta v)m}{t}$$

Friction as Impulses

- The velocity resolution algorithm we already have involves calculating the impulse needed to achieve the desired change in velocity. This impulse is initially found in contact coordinates.
- Since we will be working in impulses, we can combine the preceding equations with the friction equations, such that the maximum impulse g we can apply with static friction is:

$$g_{\text{normal}}\,\mu$$

where g_{normal} is the impulse in the direction of the contact normal.

Friction as Impulses

- **So the initial version of the procedure is the following:**

1. *Compute, in Contact Coords, the impulse g = (g.x, g.y, g.z) required to remove all (x,y,z) components of velocity. (how, again? next topic…)*
2. *Compute g_{normal} = (g.x, 0, 0) // reaction force*
3. *Compute $g_{planar} = g - g_{normal}$*
4. *If ($|g_{planar}| \leq |g_{normal}| * \mu$)*

   // Handle as Static Friction

   Else

   *g.y = (g.y/$|g_{planar}|$)*μ*g.x // Handle as Dynamic*

   *g.z = (g.z/$|g_{planar}|$)*μ*g.x // Friction*

Computing Impulse

- To determine the impulse needed, we need to handle all three directions at the same time, and we need to take into account the fact that an impulse in one direction can increase the velocity of the contact in a different direction.
- A full contact coordinate system must be used.
- We need to calculate the change in velocity given any combination of impulses in the three contact directions.
- What we need is a matrix : it will transform a vector (the impulse) into another vector (the resulting velocity).

Computing Impulse

- **Rather than use just the contact normal in the first stage, we need to use all three directions of the contact: the basis matrix.**
- **We can use the following property of cross products:**

If

$$\boldsymbol{v} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

then v $\times$ x is equivalent to

$$\begin{bmatrix} 0 & -c & b \\ c & 0 & -a \\ -b & a & 0 \end{bmatrix} \boldsymbol{x}$$

Computing Impulse

- **Then if q_{rel} = (a, b, c), then**

$$M_{\hat{g}\to\tau} = \begin{bmatrix} 0 & -c & b \\ c & 0 & -a \\ -b & a & 0 \end{bmatrix}$$

such that angular contribution to the change in velocity (in contact coordinates) due to a *unit impulse* at the contact point is

$$M_{\hat{g}\to\Delta\dot{q}} = M_{C\to W}\left(\left(-M_{\hat{g}\to\tau}\right)I^{-1}M_{\hat{g}\to\tau}\right)M_{C\to W}^{T}$$

- **As before, the linear contribution is simply**

$$M_{\hat{g}\to\Delta\dot{p}} = m^{-1}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Computing Impulse

- Then total change in velocity due to the unit impulse is

$$M_{\hat{g}\to(\Delta\dot{q})_t} = M_{\hat{g}\to\Delta\dot{q}} + M_{\hat{g}\to\Delta\dot{p}}$$

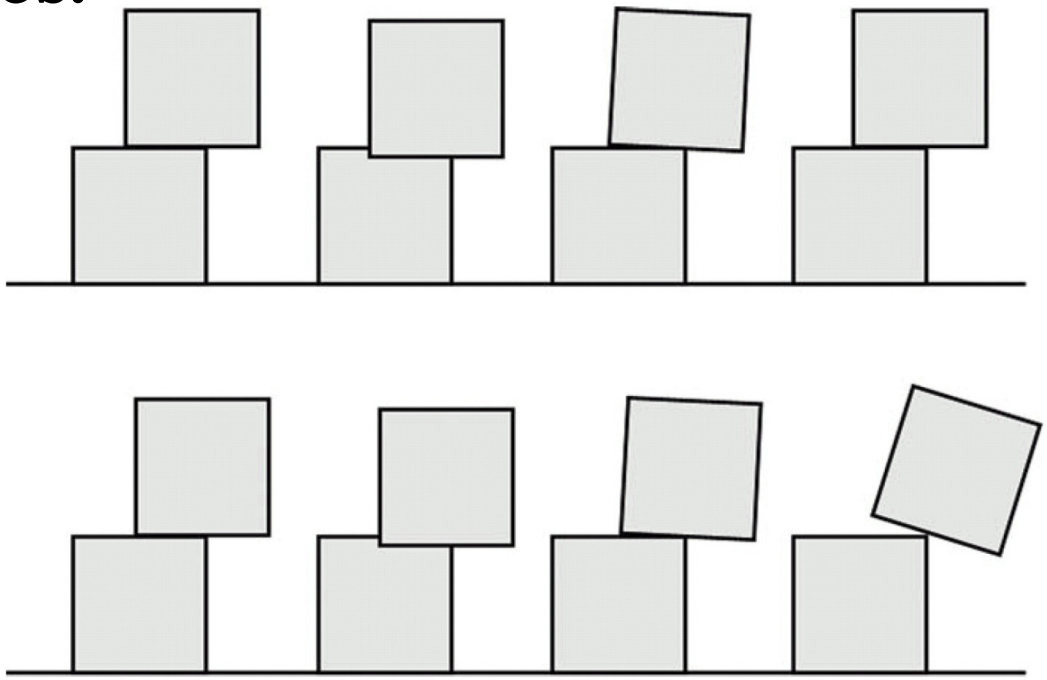
- Then the impulse to "kill" the target velocities is

$$\boldsymbol{g} = M^{-1}_{\hat{g}\to(\Delta\dot{q})_t} \begin{bmatrix} v_s' \\ -v_s.y \\ -v_s.z \end{bmatrix}$$

- With ***g*** we can apply check whether the friction is ***static*** or ***kinetic***.

Friction and Sequential Contact Resolution

- At this stage we can also see main unavoidable limitation of the micro-collision approach to physics.



§16.2

Optimizations

Sleep

- An optimization commonly used for physics engines is to avoid simulating objects that are stable and not moving. In fact this encompasses the majority of objects in a typical simulation.
- Stopping the simulation of objects at rest is called putting them to "sleep." A powerful sleep system can improve the performance of a physics simulation by many ***hundreds of times***, for an average game level.
- There are two components to the sleep system:
  - one algorithm to put objects to sleep, and
  - another to wake them up again.

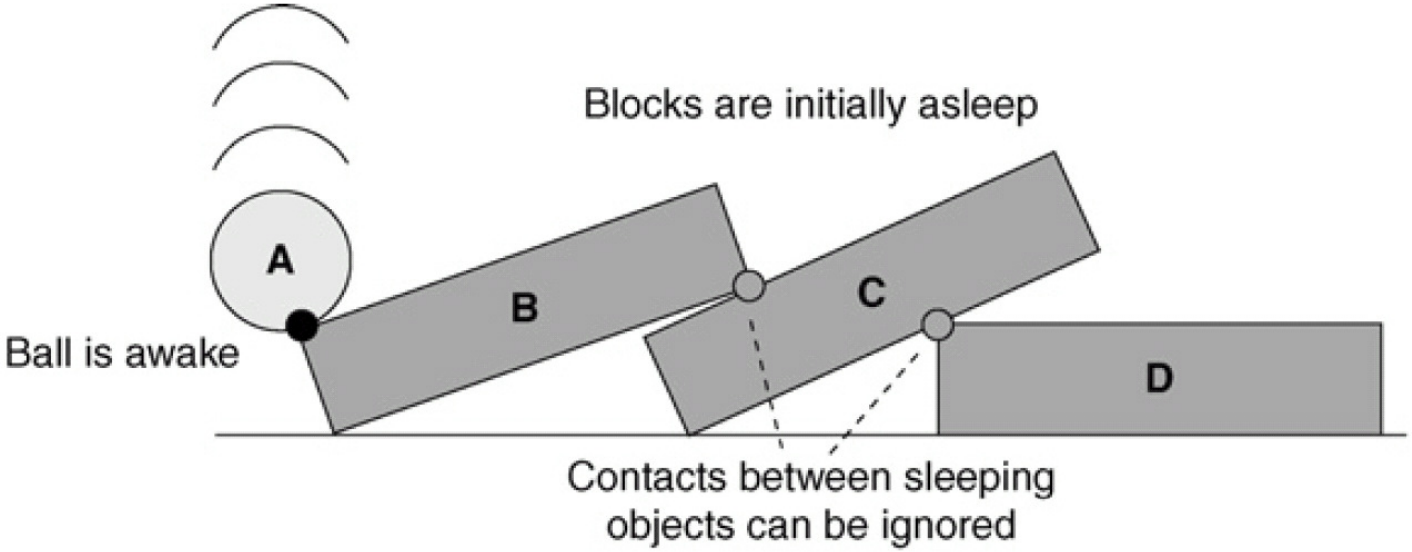
Sleep: Putting to Sleep

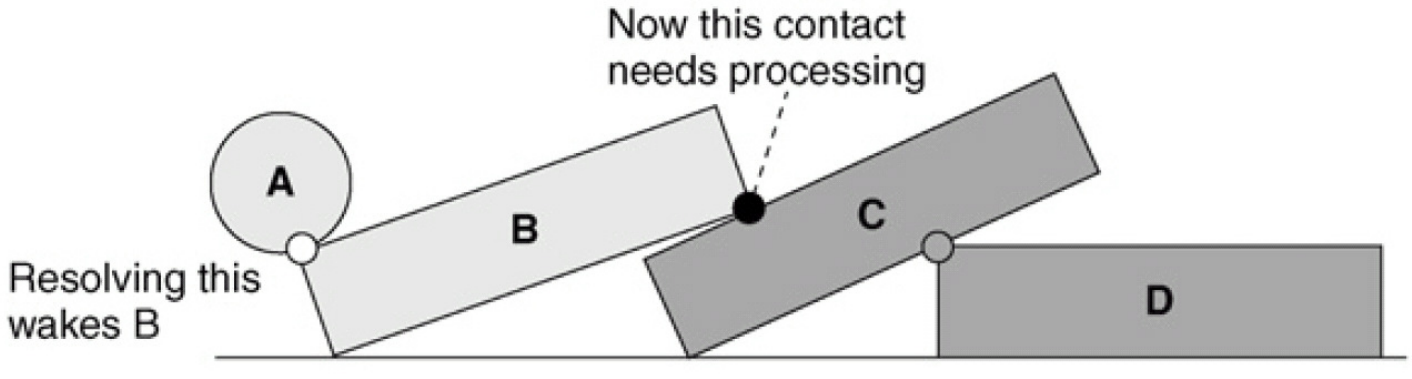
- A Boolean variable is used to tell us whether the body is currently asleep.
- When we perform a rigid-body update, we check the body and return without processing if it is asleep (this includes collision handling).
- The collision detector should still return contacts between objects that are asleep.
- Each body is also given a motion variable that will keep track of the current movement speed (both linear and angular) of the object
- An object is put to sleep when the value of this motion drops below a certain threshold.

Sleep: Waking Up

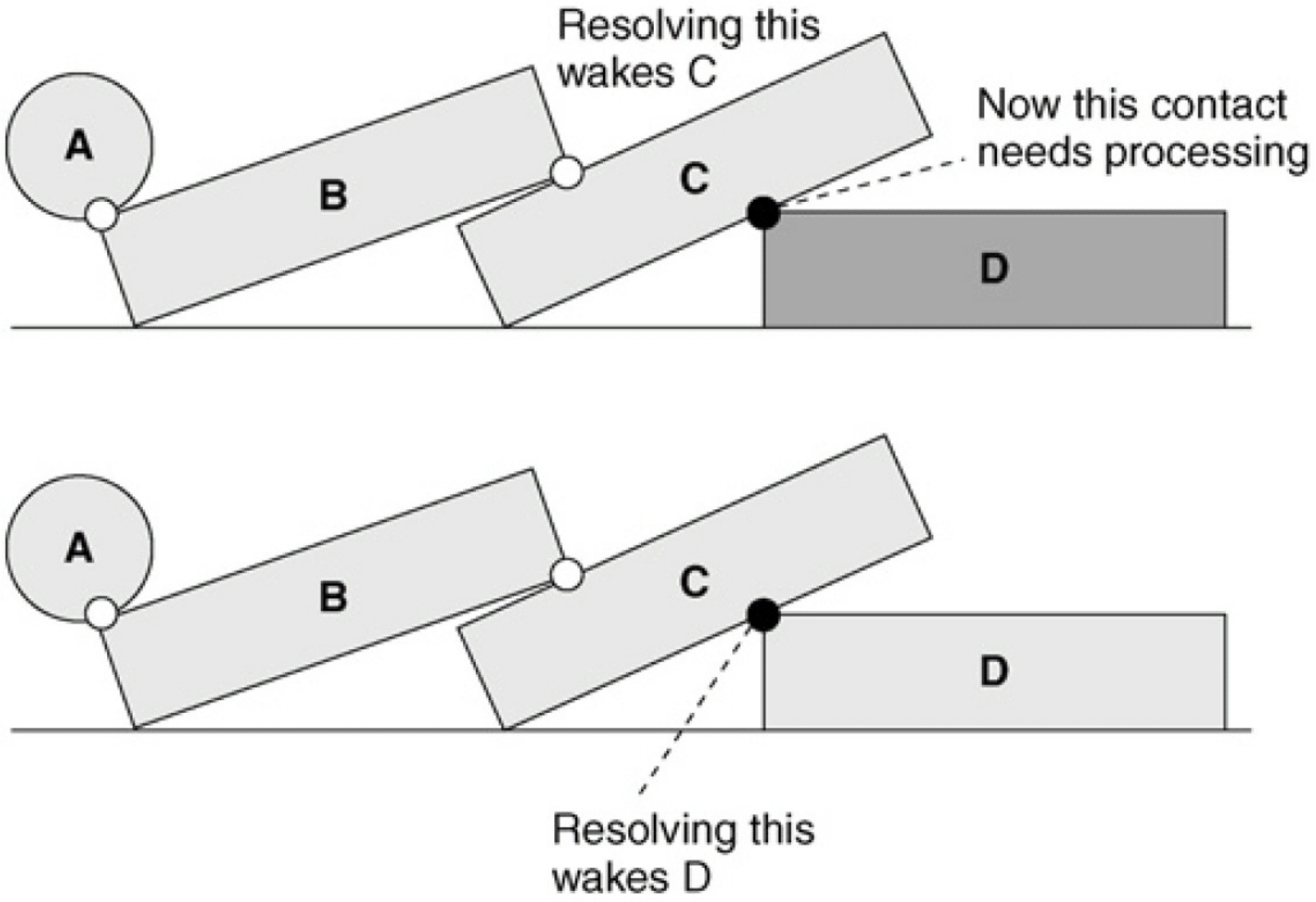
- Objects can be awakened manually. We also need to wake objects up when they must respond to new collisions.
- When a new object (the player, for example, or a projectile) comes along and collides with a sleeping object, we want all objects that could be affected by the collision to wake up.
- We also need to wake up a rigid body when a force is applied to it (ignoring constant forces such as gravity). This can be done automatically.

Sleep: Waking Up





Sleep: Waking Up



References/Resources

- **Game Physics Engine Development, 2nd Ed., by Ian Millington. [text]**