

Fast and Simple Physics using Sequential Impulses

Erin Catto
Crystal Dynamics

Physics Engine Checklist

- ⊕ Collision and contact
- ⊕ Friction: static and dynamic
- ⊕ Stacking
- ⊕ Joints
- ⊕ Fast, simple, and robust

Box2D Demo

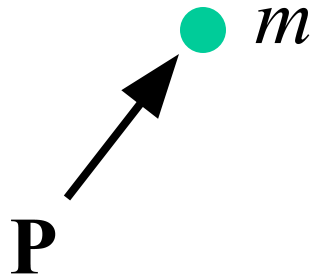
- ⊕ It's got collision
- ⊕ It's got friction
- ⊕ It's got stacking
- ⊕ It's got joints
- ⊕ Check the code, it's simple!

Fast and Simple Physics

- ⊕ Penalty method?
Nope
- ⊕ Linear complementarity (LCP)?
Nope
- ⊕ Joint coordinates (Featherstone)?
Nope
- ⊕ Particles (Jakobsen)?
Nope
- ⊕ Impulses?
Bingo!

Why Impulses?

- ⊕ Most people don't hate impulses
- ⊕ The math is almost understandable
- ⊕ Intuition often works
- ⊕ Impulses can be robust



$$\Delta \mathbf{v} = \frac{\mathbf{P}}{m}$$

Making Impulses not Suck

- ⊕ Impulses are good at making things bounce.
- ⊕ Many attempts to use impulses leads to bouncy simulations (aka jitter).
- ⊕ Forget static friction.
- ⊕ Forget stacking.

Impulses without the Bounce

- ⊕ Forget bounces for a moment.
- ⊕ Let's concentrate on keeping things still.
- ⊕ It's always easy to add back in the bounce.

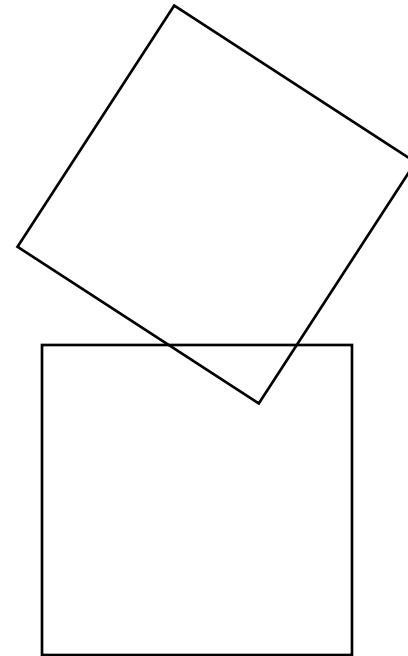
The 5 Step Program

(for taking the jitter out of impulses)

- ⊕ Accept penetration
- ⊕ Remember the past
- ⊕ Apply impulses early and often
- ⊕ Pursue the true impulse
- ⊕ Update position last

Penetration

- ⊕ Performance
- ⊕ Simplicity
- ⊕ Coherence
- ⊕ Game logic
- ⊕ Fewer cracks



Algorithm Overview

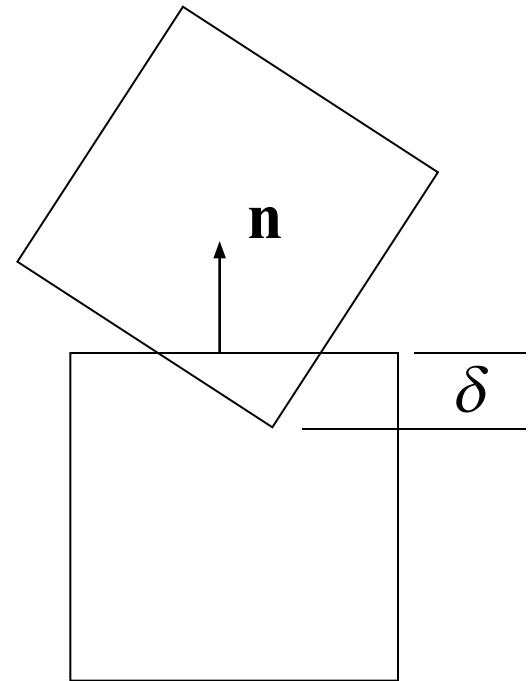
- ④ Compute contact points
- ④ Apply forces (gravity)
- ④ Apply impulses
- ④ Update position
- ④ Loop

Contact Points

- ④ Position, normal, and penetration
- ④ Box-box using the SAT
- ④ Find the axis of minimum penetration
- ④ Find the incident face on the other box
- ④ Clip

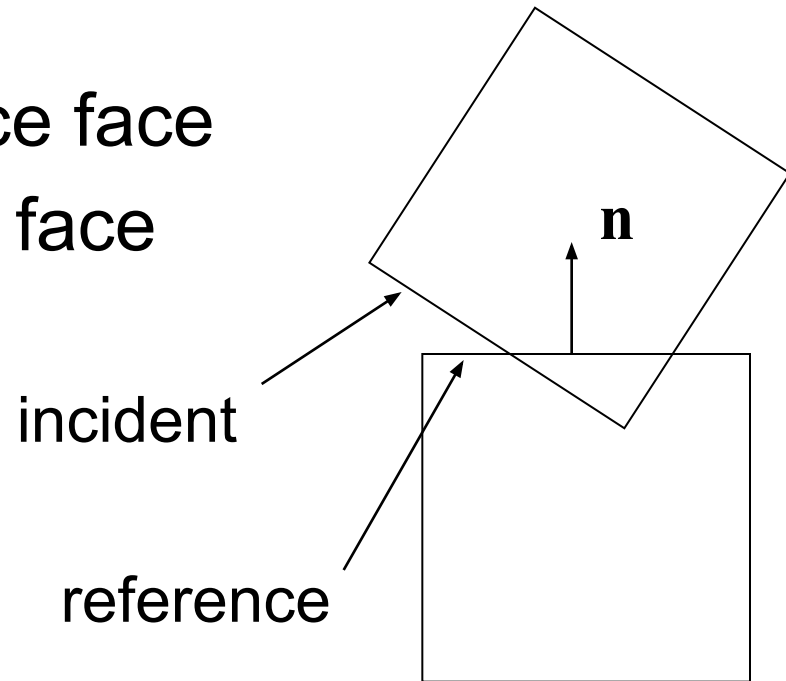
Box-Box SAT

- ⊕ First find the separating axis with the minimum penetration.
- ⊕ In 2D the separating axis is a face normal.



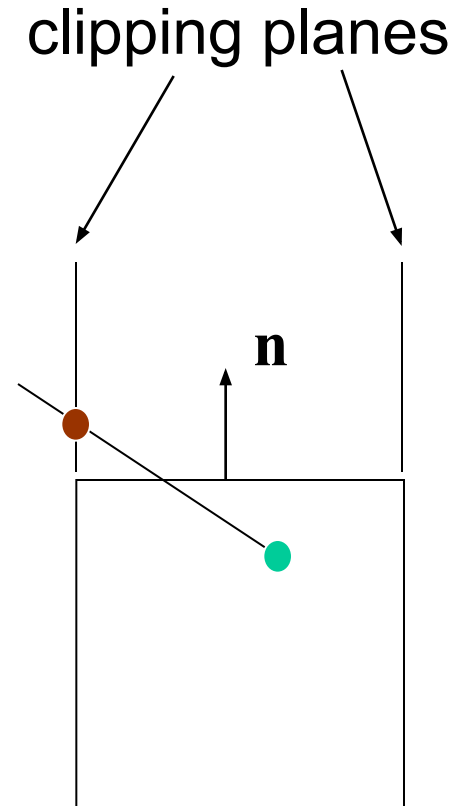
Box-Box Clipping Setup

- ④ Identify reference face
- ④ Identify incident face



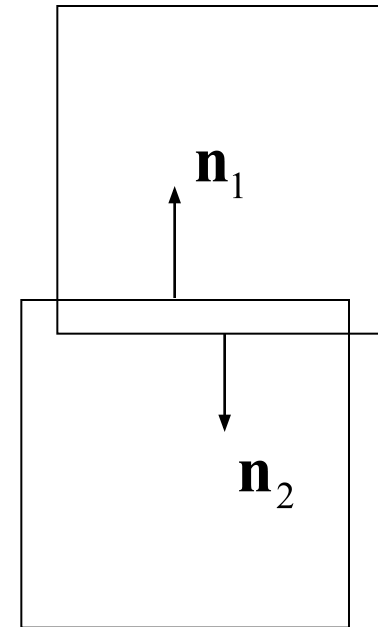
Box-Box Clipping

- ④ Clip incident face against reference face side planes (but not the reference face).
- ④ Consider clip points with positive penetration.



Feature Flip-Flop

- Which normal is the separating axis?
- Apply weightings to prefer one axis over another.
- Improved coherence.



Apply Forces

Newton's Law

$$\dot{m}\mathbf{v} = \mathbf{F}$$

$$\dot{I}\boldsymbol{\omega} + \boldsymbol{\omega} \times I\boldsymbol{\omega} = \mathbf{T}$$

Ignore gyroscopic term
for improved stability

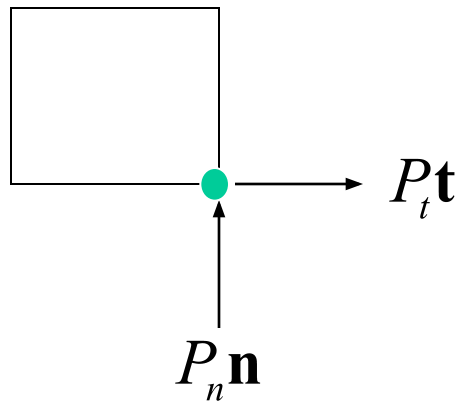
Use Euler's rule

$$\mathbf{v}_2 = \mathbf{v}_1 + \Delta t m^{-1} \mathbf{F}$$

$$\boldsymbol{\omega}_2 = \boldsymbol{\omega}_1 + \Delta t I^{-1} \mathbf{T}$$

Impulses

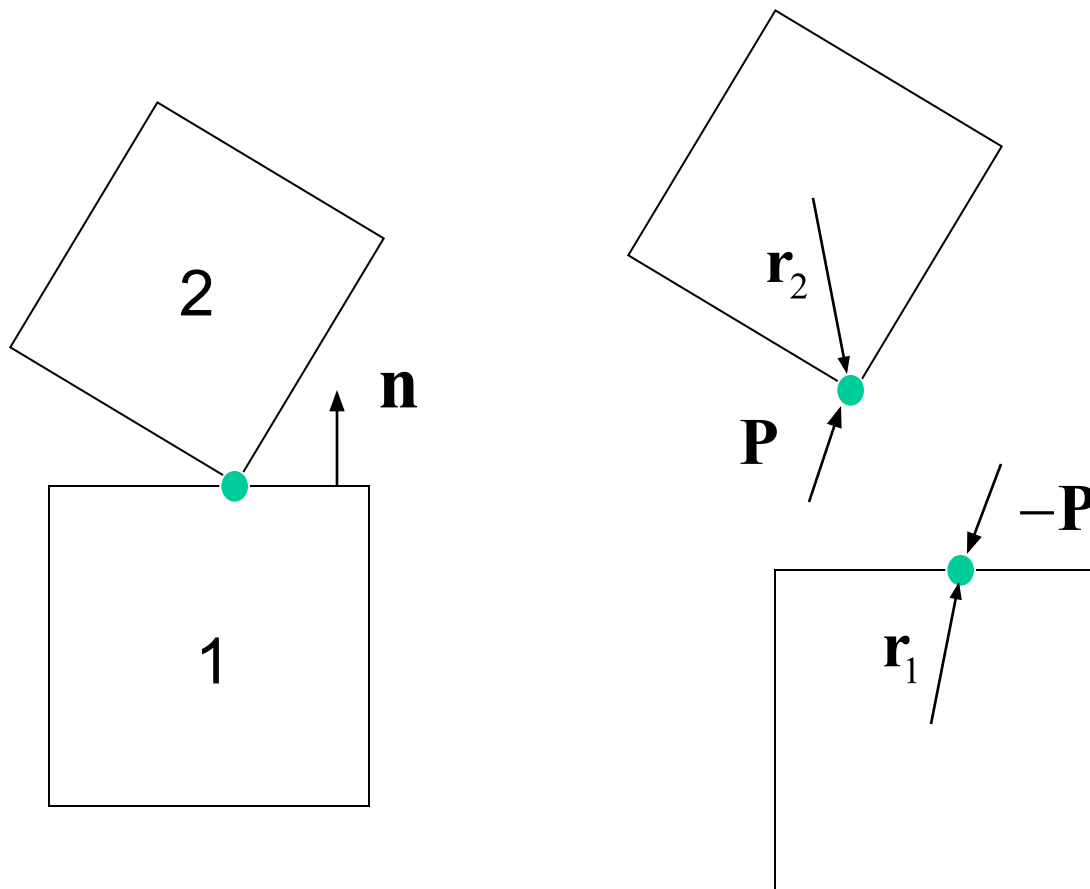
- ⊕ Impulses are applied at each contact point.
- ⊕ Normal impulses to prevent penetration.
- ⊕ Tangent impulses to impose friction.



$$P_n \geq 0$$

$$|P_t| \leq \mu P_n$$

Computing the Impulse



Linear Momentum

The normal impulse causes an instant change in velocity.

We know the direction of the normal impulse. We only need it's magnitude.

$$\mathbf{v}_1 = \bar{\mathbf{v}}_1 - \mathbf{P} / m_1$$

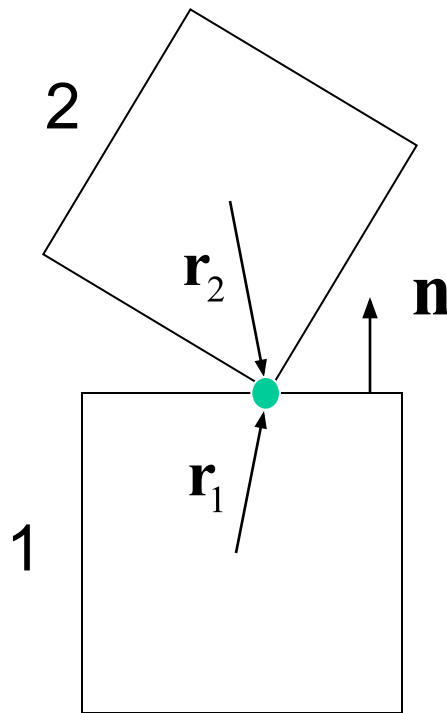
$$\boldsymbol{\omega}_1 = \bar{\boldsymbol{\omega}}_1 - I_1^{-1} \mathbf{r}_1 \times \mathbf{P}$$

$$\mathbf{v}_2 = \bar{\mathbf{v}}_2 + \mathbf{P} / m_2$$

$$\boldsymbol{\omega}_2 = \bar{\boldsymbol{\omega}}_2 + I_2^{-1} \mathbf{r}_2 \times \mathbf{P}$$

$$\mathbf{P} = P_n \mathbf{n}$$

Relative Velocity



$$\Delta \mathbf{v} = \mathbf{v}_2 + \mathbf{r}_2 \times \boldsymbol{\omega}_2 - \mathbf{r}_1 - \boldsymbol{\omega}_1 \times \mathbf{r}_1$$

Along Normal:

$$v_n = \Delta \mathbf{v} \cdot \mathbf{n}$$

The Normal Impulse

Want: $v_n = 0$ $P_n \geq 0$

Get: $P_n = \max\left(\frac{-\Delta\bar{\mathbf{v}} \cdot \mathbf{n}}{k_n}, 0\right)$

Fine Print:

$$\Delta\bar{\boldsymbol{\omega}} = \bar{\mathbf{v}}_2 + \bar{\mathbf{v}}_2 \times \boldsymbol{\omega} - \bar{\mathbf{r}}_1 - \bar{\mathbf{r}}_1 \times \boldsymbol{\omega}$$

$$k_n = \frac{1}{m_1} + \frac{1}{m_2} + \left[I_1^{-1} (\mathbf{r}_1 \times \mathbf{n}) \times \mathbf{r}_1 + I_2^{-1} (\mathbf{r}_2 \times \mathbf{n}) \times \mathbf{r}_2 \right] \cdot \mathbf{n}$$

Bias Impulse

- ④ Give the normal impulse some extra oomph.
- ④ Proportional to the penetration.
- ④ Allow some slop.
- ④ Be gentle.

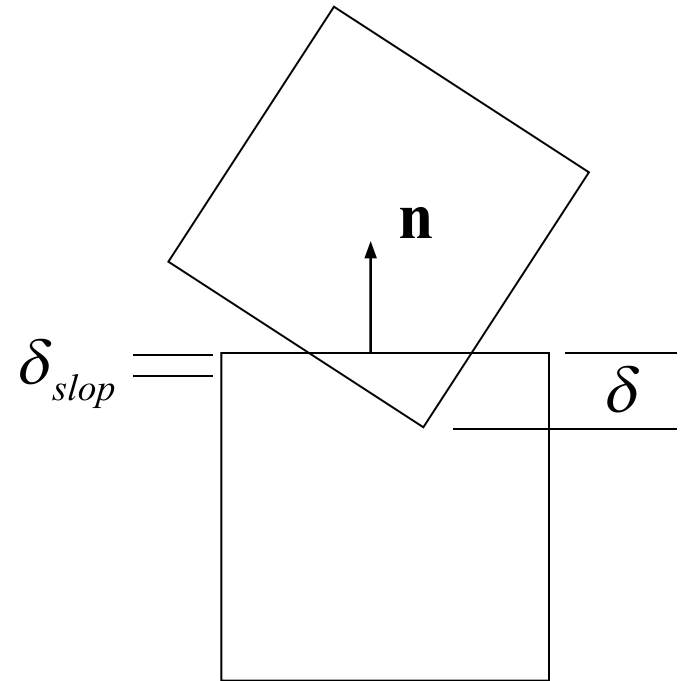
Bias Velocity

Slop: δ_{slop}

Bias Factor: $\beta \approx [0.1, 0.3]$

Bias velocity:

$$v_{bias} = \frac{\beta}{\Delta t} \max(0, \delta - \delta_{slop})$$



Bias Impulse

With bias velocity, this:

$$P_n = \max\left(\frac{-\Delta\bar{\mathbf{v}} \cdot \mathbf{n}}{k_n}, 0\right)$$

Becomes:

$$P_n = \max\left(\frac{-\Delta\bar{\mathbf{v}} \cdot \mathbf{n} + v_{bias}}{k_n}, 0\right)$$

Friction Impulse

Tangent Velocity: $v_t = \Delta \mathbf{v} \cdot \mathbf{t}$

Want: $v_t = 0 \quad -\mu P_n \leq P_t \leq \mu P_n$

Get: $P_t = \text{clamp}\left(\frac{-\Delta \bar{\mathbf{v}} \cdot \mathbf{t}}{k_t}, -\mu P_n, \mu P_n\right)$

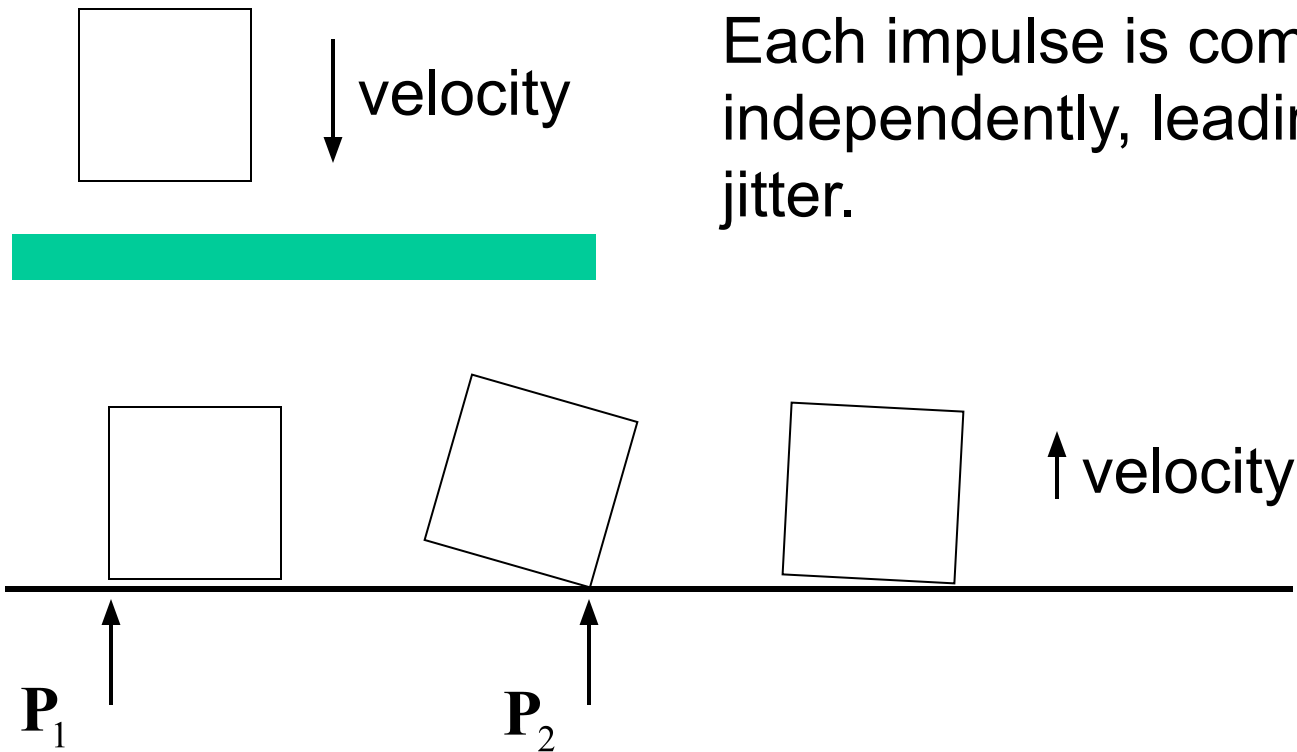
Fine Print:

$$k_t = \frac{1}{m_1} + \frac{1}{m_2} + \left[I_1^{-1} (\mathbf{r}_1 \times \mathbf{t}) \times \mathbf{r}_1 + I_2^{-1} (\mathbf{r}_2 \times \mathbf{t}) \times \mathbf{r}_2 \right] \cdot \mathbf{t}$$

Sequential Impulses

- ④ Apply an impulse at each contact point.
- ④ Continue applying impulses for several iterations.
- ④ Terminate after:
 - fixed number of iterations
 - impulses become small

Naïve Impulses

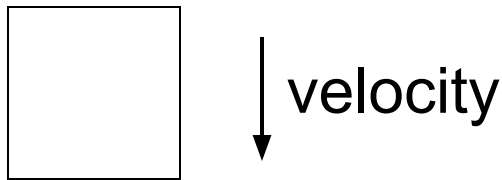


Each impulse is computed independently, leading to jitter.

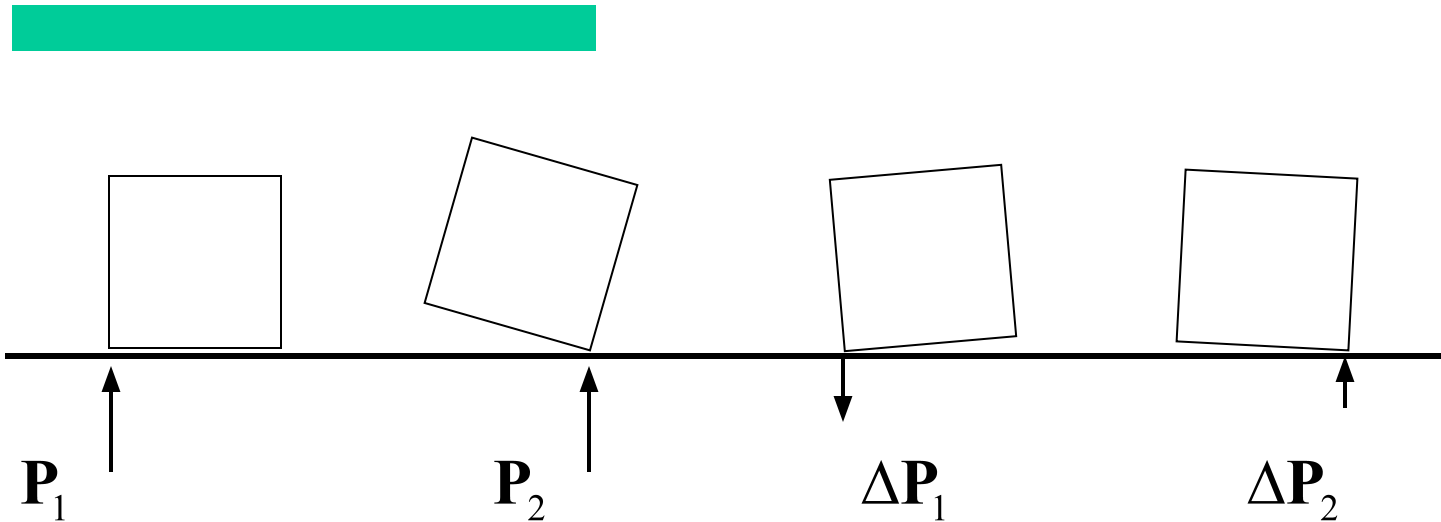
Where Did We Go Wrong?

- ⊕ Each contact point forgets its impulse history.
- ⊕ Each contact point requires that every impulse be positive.
- ⊕ There is no way to recover from a bad impulse.

Accumulated Impulses



Each impulse adds to the total. Increments can be negative.



The True Impulse

- ⊕ Each impulse adds to an accumulated impulse for each contact point.
- ⊕ The accumulated impulse approaches the true impulse (hopefully).
- ⊕ True impulse: an exact global solution.

Accumulated Impulse

- ⊕ Clamp the accumulated impulse, not the incremental impulses.

Accumulated impulses:

$$P_{\Sigma n}$$

$$P_{\Sigma t}$$

Correct Clamping

Normal Clamping:

$$temp = P_{\Sigma n}$$

$$P_{\Sigma n} = \max(P_{\Sigma n} + P_n, 0)$$

$$P_n = P_{\Sigma n} - temp$$

Friction Clamping:

$$temp = P_{\Sigma t}$$

$$P_{\Sigma t} = \text{clamp}(P_{\Sigma t} + P_t, -\mu P_{\Sigma n}, \mu P_{\Sigma n})$$

$$P_t = P_{\Sigma t} - temp$$

Position Update

- ④ Use the new velocities to integrate the positions.
- ④ The time step is complete.

Extras

- ⊕ Coherence
- ⊕ Feature-based contact points
- ⊕ Joints
- ⊕ Engine layout
- ⊕ Loose ends
- ⊕ 3D Issues

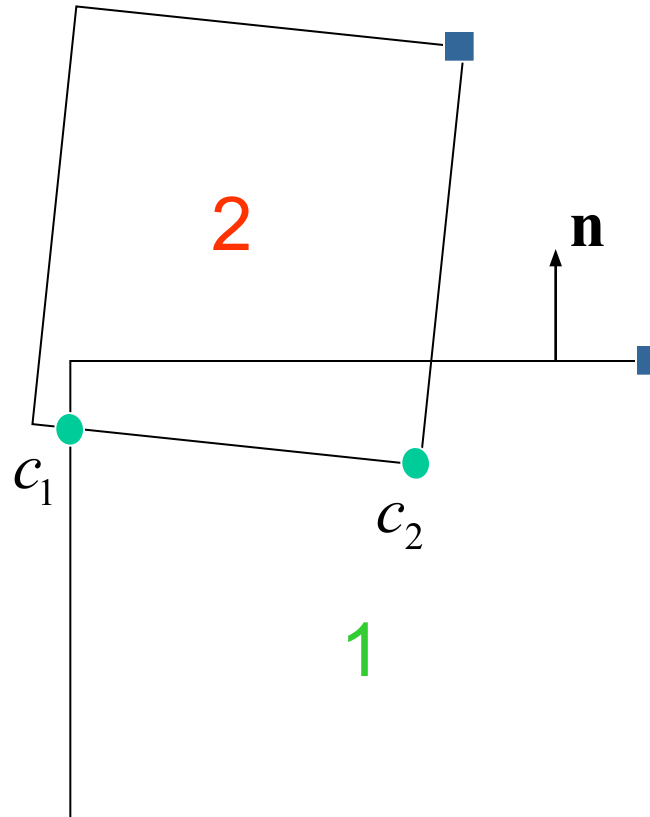
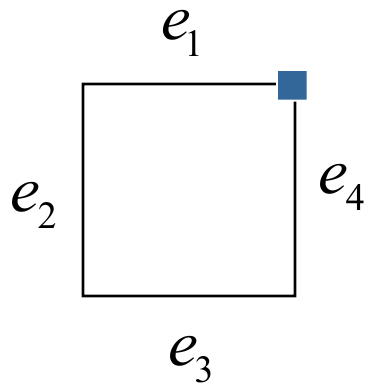
Coherence

- ⊕ Apply old accumulated impulses at the beginning of the step.
- ⊕ Less iterations and greater stability.
- ⊕ We need a way to match old and new contacts.

Feature-Based Contact Points

- ⊕ Each contact point is the result of clipping.
- ⊕ It is the junction of two different edges.
- ⊕ An edge may come from either box.
- ⊕ Store the two edge numbers with each contact point – this is the Contact ID.

Contact Point IDs



c_1
box 1 edge 2
box 2 edge 3

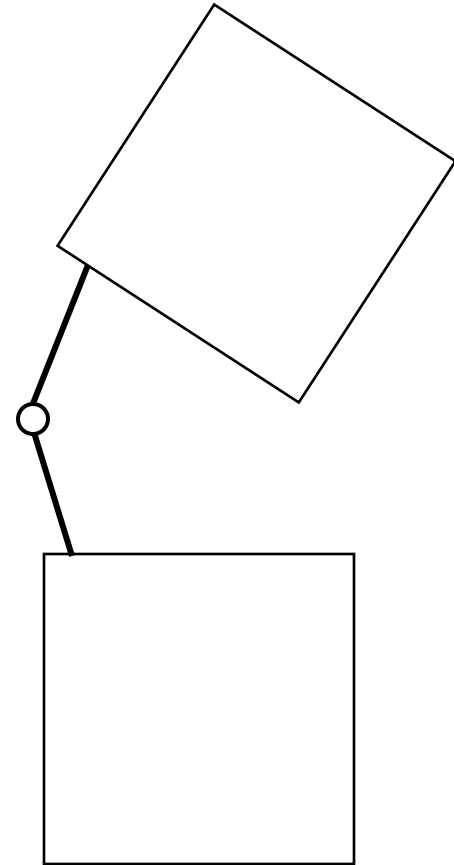
c_2
box 2 edge 3
box 2 edge 4

Joints

- ④ Specify (constrain) part of the motion.
- ④ Compute the impulse necessary to achieve the constraint.
- ④ Use an accumulator to pursue the true impulse.
- ④ Bias impulse to prevent separation.

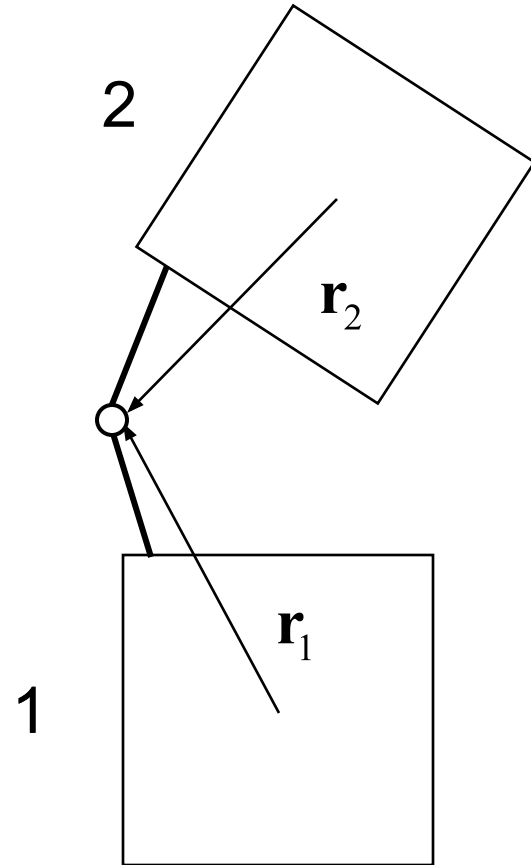
Revolute Joint

- 🌀 Two bodies share a common point.
- 🌀 They rotate freely about the point.



Revolute Joint

- ⊕ The joint knows the local anchor point for both bodies.



Relative Velocity

- ⊕ The relative velocity of the anchor points is zero.

$$\Delta \mathbf{v} = \mathbf{v}_2 + \mathbf{v}_2 \times \boldsymbol{\omega} - \mathbf{r}_1 - \mathbf{r}_1 \times \boldsymbol{\omega} = 0$$

- ⊕ An impulse is applied to the two bodies.

P

Linear Momentum

- Apply linear momentum to the relative velocity to get:

$$K\mathbf{P} = -\Delta\bar{\mathbf{v}}$$

- Fine Print:

$$K = \left(\frac{1}{m_1} + \frac{1}{m_2} \right) \mathbf{1} \tilde{\mathbf{r}}_1 I_1^{-1} \mathbf{r}_1 \tilde{\mathbf{r}}_2 I_2^{-1} \mathbf{r}_2$$

- Tilde (~) for the cross-product matrix.

K Matrix

- ⊕ 2-by-2 matrix in 2D, 3-by-3 in 3D.
- ⊕ Symmetric positive definite.
- ⊕ Think of K as the inverse mass matrix of the constraint.

$$M_c = K^{-1}$$

Bias Impulse

- ⊕ The error is the separation between the anchor points

$$\Delta \mathbf{p} = \mathbf{x}_2 + \mathbf{r}_2 - \mathbf{x}_1 - \mathbf{r}_1$$

- ⊕ Center of mass: \mathbf{x}
- ⊕ Bias velocity and impulse:

$$\mathbf{v}_{bias} = -\frac{\beta}{\Delta t} \Delta \mathbf{p}$$

$$K\mathbf{P} = -\Delta \bar{\mathbf{v}} + \mathbf{v}_{bias}$$

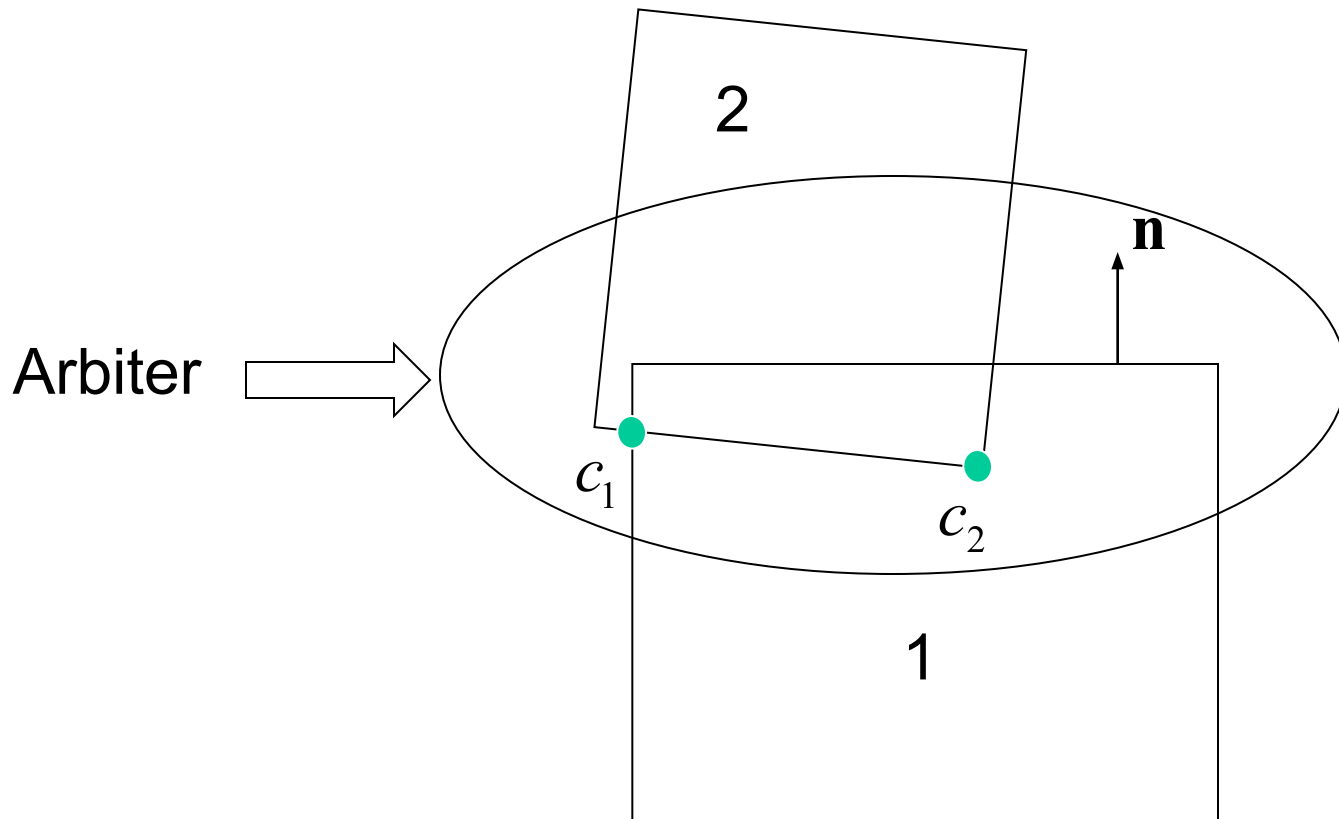
Engine Layout

- ⊕ The *World* class contains all bodies, contacts, and joints.
- ⊕ Contacts are maintained by the *Arbiter* class.

Arbiter

- ⊕ An arbiter exists for every touching pair of boxes.
- ⊕ Provides coherence.
- ⊕ Matches new and old contact points using the Contact ID.
- ⊕ Persistence of accumulated impulses.

Arbiters



Collision Coherence

- ④ Use the arbiter to store the separating axis.
- ④ Improve performance at the cost of memory.
- ④ Use with broad-phase.

More on Arbiters

- ⊕ Arbiters are stored in a set according to the ordered body pointers.
- ⊕ Use time-stamping to remove stale arbiters.
- ⊕ Joints are permanent arbiters.
- ⊕ Arbiters can be used for game logic.

Loose Ends

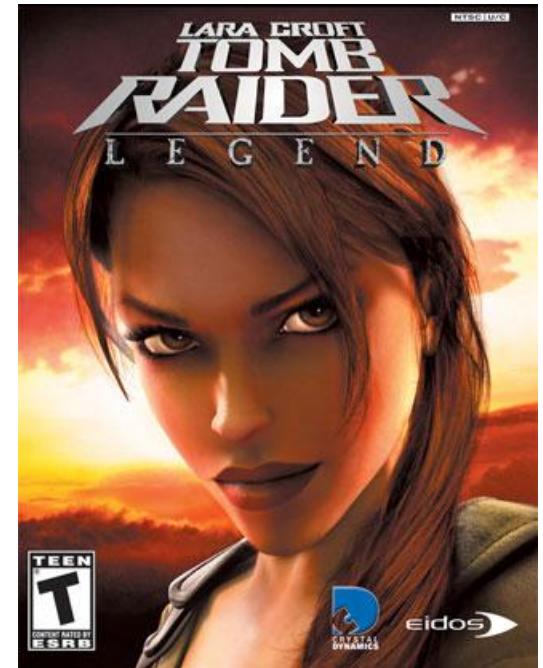
- ⊕ Ground is represented with bodies whose inverse mass is zero.
- ⊕ Contact mass can be computed as a pre-step.
- ⊕ Bias impulses shouldn't affect the velocity state (TODO).

3D Issues

- ⊗ Friction requires two axes.
- ⊗ Align the axes with velocity if it is non-zero.
- ⊗ Identify a *contact patch* (manifold) and apply friction at the center.
- ⊗ This requires a *twist friction*.
- ⊗ Big CPU savings.

Questions?

- ④ <http://www.gphysics.com>
- ④ erincatto at that domain
- ④ Download the code there.
- ④ Buy Tomb Raider Legend!



References

- ⊕ Physics-Based Animation by Kenny Erleben et al.
- ⊕ Real-Time Collision Detection by Christer Ericson.
- ⊕ Collision Detection in Interactive 3D Environments by Gino van den Bergen.
- ⊕ Fast Contact Reduction for Dynamics Simulation by Adam Moravanszky and Pierre Terdiman in Game Programming Gems 4.