
Game Developers Conference®
February 28 - March 4, 2011
Moscone Center, San Francisco
www.GDConf.com
GDC
UBM

Physics for Game Programmers

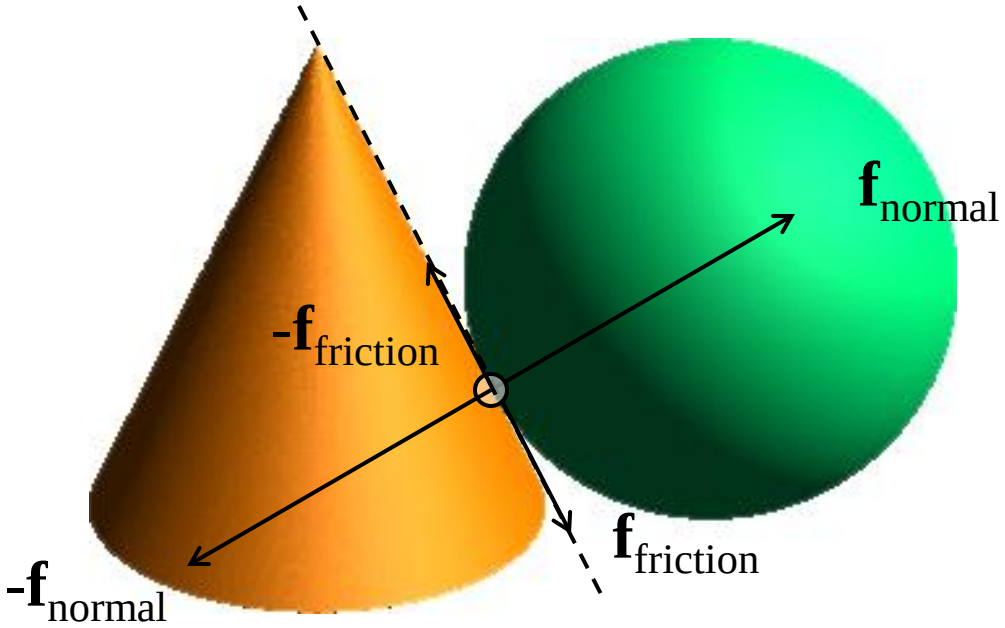
Contacts
or
"How (Not) To Make It Stick"

Gino van den Bergen
gino@dtecta.com

Contacts Are Tricky

- Work only in one direction (do not pull).
- Commonly exist for a short period only.
- Occur anywhere on an object's surface.
- Involve sliding and rolling.
- Multiple solutions due to overconstraining

Contact Plane



Contact Plane (Cont'd)

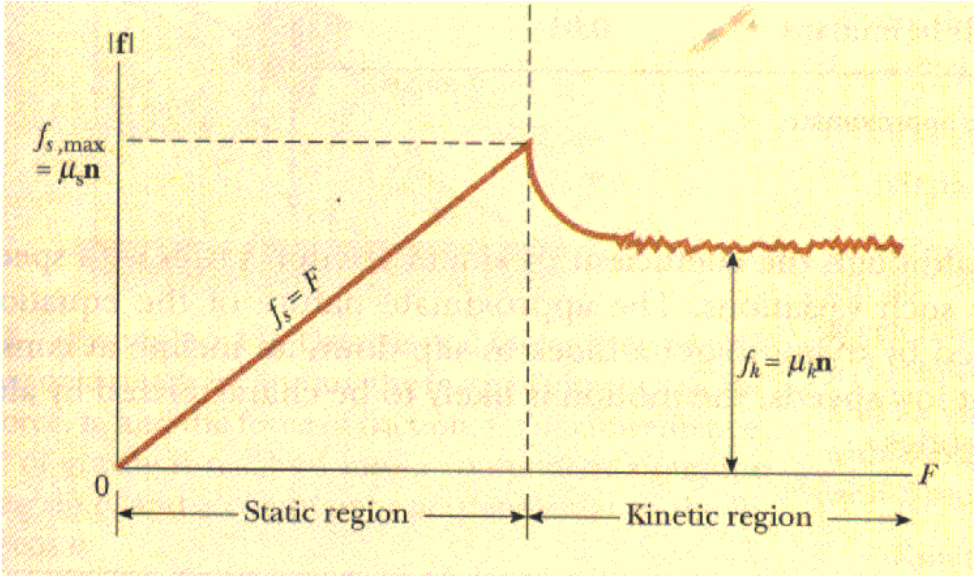
- Normal forces/impulses act along the plane normal.
- Friction forces/impulses act in the contact plane.
- Forces/impulses apply to the contact point.

Coulomb or "Dry" Friction

- The maximum tangential force to the contact plane exerted by friction is equal to the normal force times the coefficient of friction

$$\mathbf{f}_{\text{friction}} \leq \mu \mathbf{f}_{\text{normal}}$$

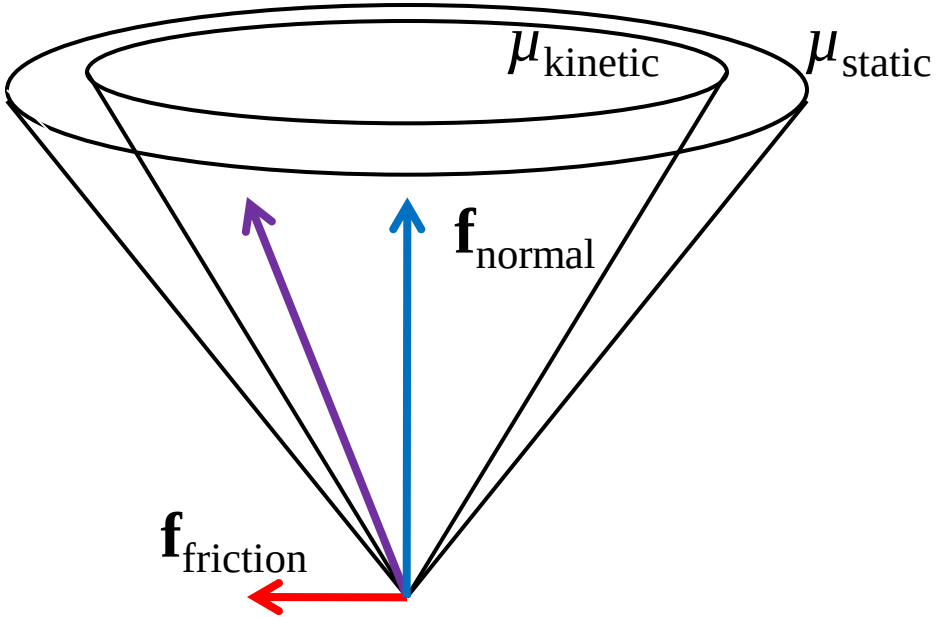
Static vs Kinetic Friction



Friction Cone

- The total force exerted by a contact is normal force plus friction.
- Coulomb’s law constrains the total force to a cone.
- Friction forces can reduce tangential velocity but cannot revert it.

Friction Cone (Cont'd)



Solving Contacts

- Determine contact impulses that filter out inadmissible relative velocities.
- Impulses do not increase momentum.
- Restitution of velocity (rebound) only at impact and only in normal direction.
- Restitution is zero for resting contacts.

Iterative Methods

- Are generally cheaper than direct methods: $O(n)$ as opposed to $O(n^3)$
- Always return a solution (even when there isn't one).
- Exploit frame coherence (warm starting).
- Converge somewhat unpredictably.

Gauss-Seidel Method

- Solve each variable in a linear system in isolation using the best approximation so far for the other variables.
- Converges for positive-definite systems. Any physics-based system (involving inertia) is bound to be positive definite.

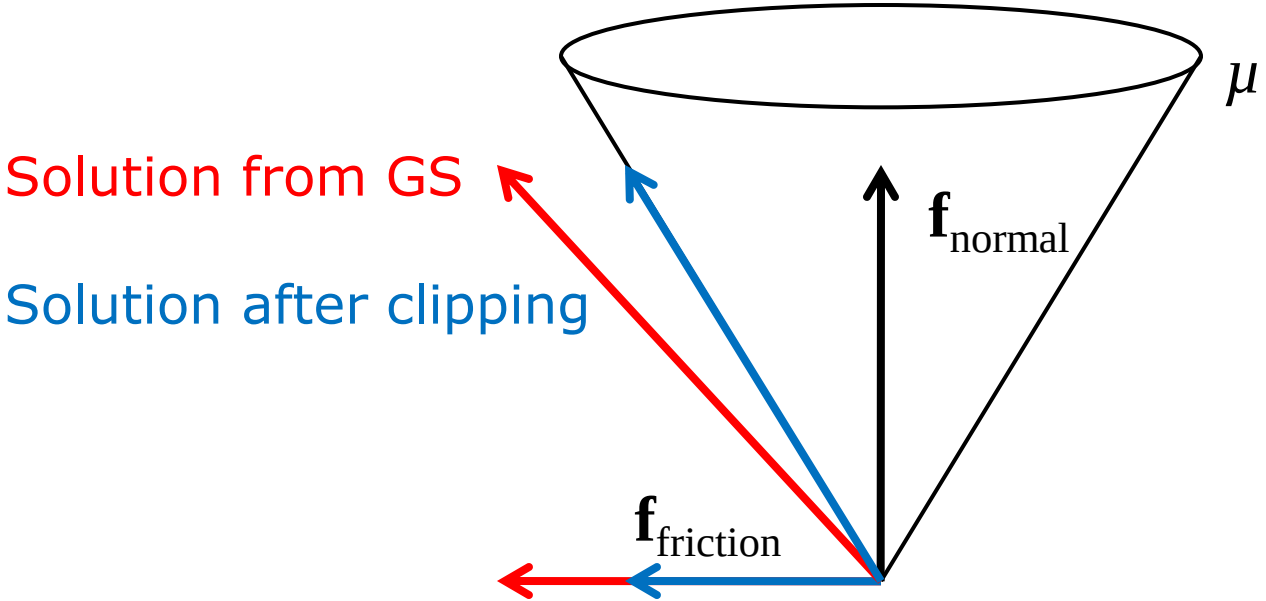
Solving Contacts: Step #1

- Solve contacts as ball joints, i.e. compute impulses that keep contacts glued.
- Solve simultaneous contacts using a Gauss-Seidel (block) solver.
- Contacts are solved one at a time.
- Momentum changes with each step.

Solving Contacts: Step #2

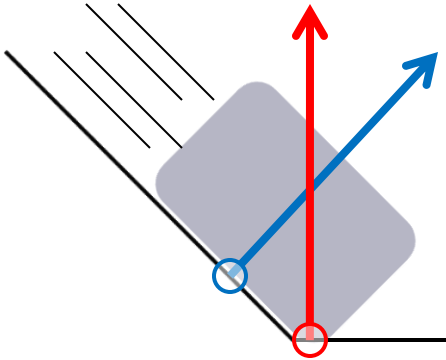
- The solution returned by the GS solver is “clamped” to the applicable friction cone:
- Negative (pulling) impulses are removed.
- Friction impulses that exceed the friction cone are clipped.

Clip Against The Friction Cone



Beware of Sliding Contacts!

- Prioritize contacts on relative velocities in the normal direction.



Red contact should be solved before blue one in GS solver. Prefer normal impulses over friction. The latter may get clipped.

Solving Penetration

- Solve penetration either by:
- Adding a tiny target velocity that moves the bodies out (Baumgarte stabilization).
- Solving penetrations independent of velocities (pseudo-impulses, post-stabilization).

Solving Penetration (Cont'd)

- Don't try to solve penetration completely!
- In-and-out contacts are a source of jitter.
- Often penetration is inevitable. Solving it completely may result in a blowup.
- Successive under-relaxation ($\omega<1$) helps.

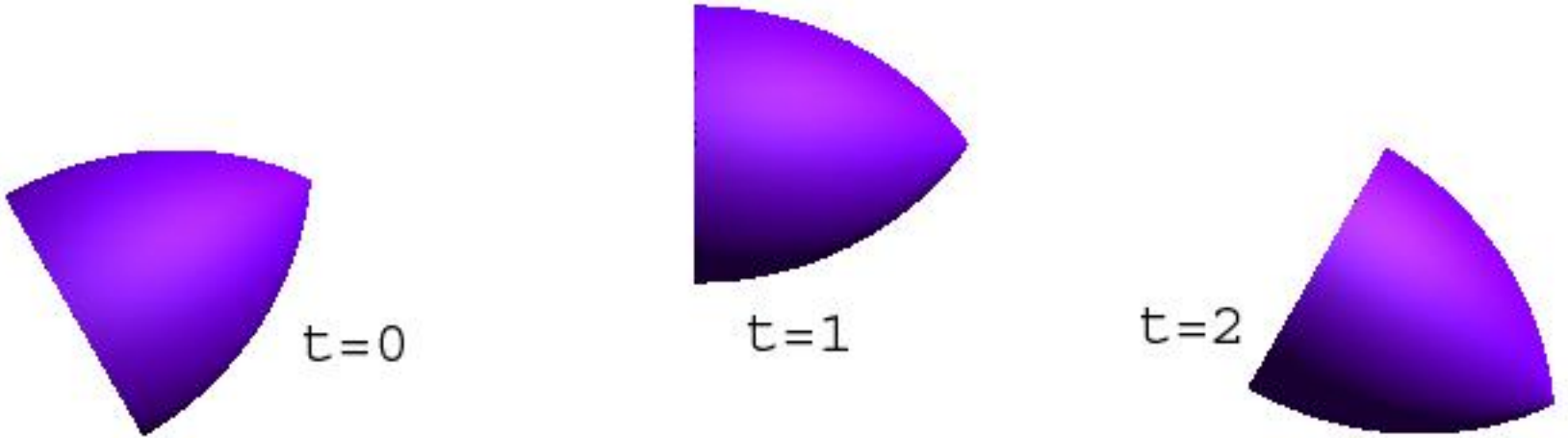
Collision Detection

- Find all pairs of objects that are colliding now, or will collide over the next frame.
- Compute data for response:
  - Contact normal
  - Contact point(s)
  - Penetration depth

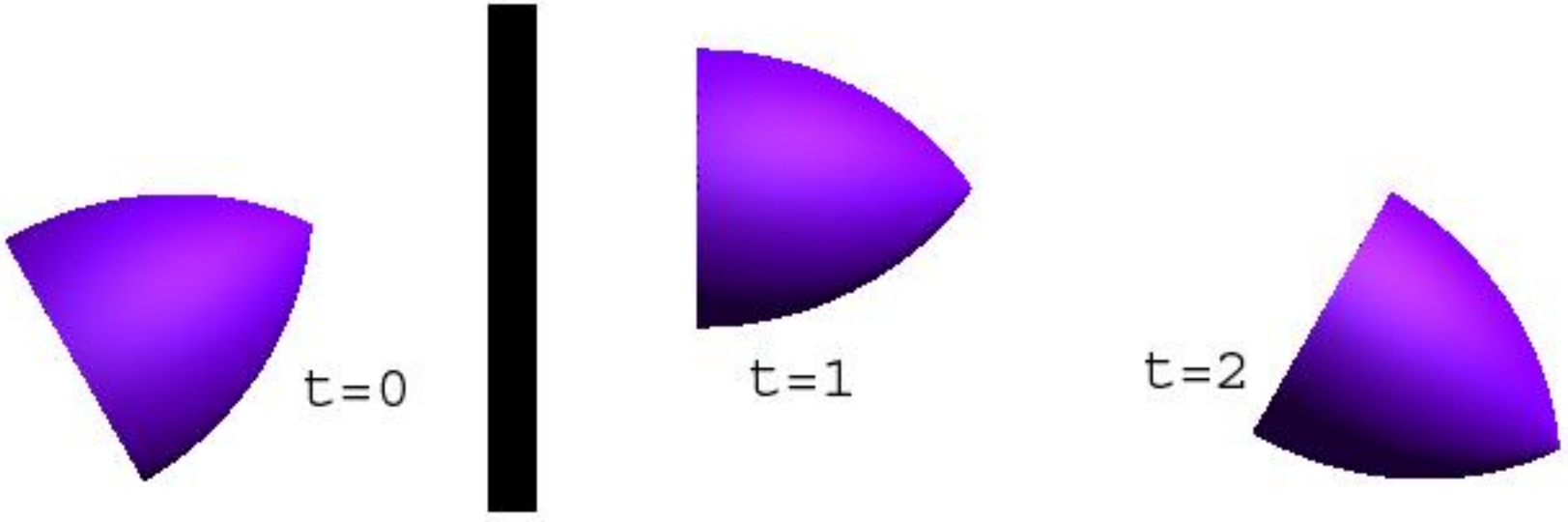
Collision Objects

- Static environment (buildings, terrain) is typically modeled using polygon meshes.
- Moving objects (player, NPCs, vehicles, projectiles) are typically convex shapes.
- We need to detect convex-convex and convex-mesh collisions.

Continuous Collision Detection



Continuous Collision Detection



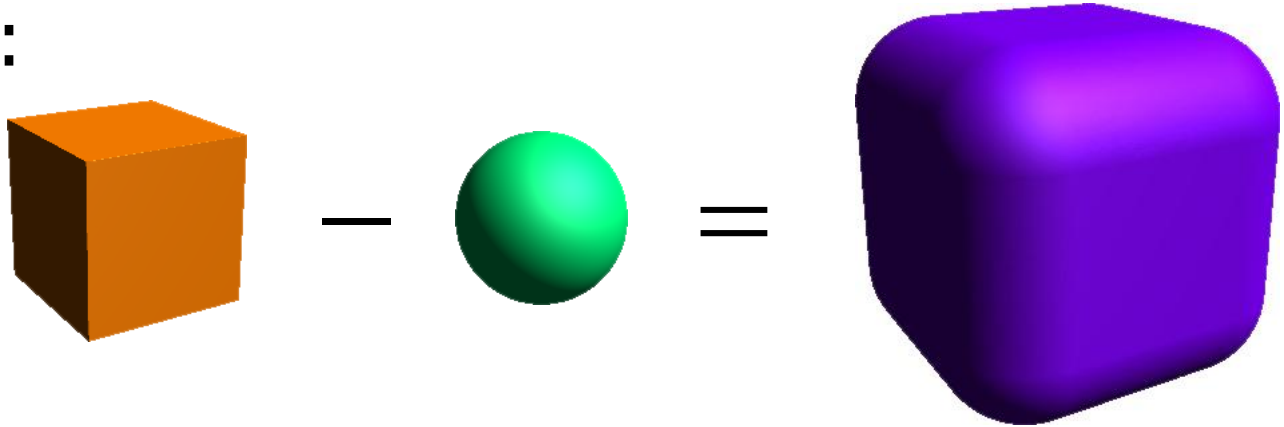
Configuration Space

- The *configuration space obstacle* of objects A and B is the set of all vectors from a point of B to a point of A.

$$A - B = \{\mathbf{a} - \mathbf{b} : \mathbf{a} \in A, \mathbf{b} \in B\}$$

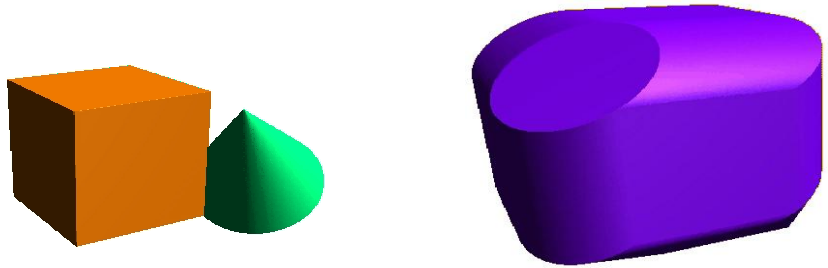
Configuration Space (Cont'd)

- CSO is basically one object dilated by the other:



Translation

- Translation of A and/or B results in a translation of $A - B$.



Rotation

- Rotation of A and/or B changes the shape of $A - B$.

Configuration Space?

- Collision queries on a pair of convexes are reduced to queries on the position of the origin with respect to the CSO.
- Point queries are easier than queries on pairs of shapes.

Queries

- The distance between two objects is the distance from the origin to the CSO.

$$d(A,B) = \min\left\{\|\mathbf{x}\| : \mathbf{x} \in A - B\right\}$$

- The objects intersect if the origin is contained by the CSO.

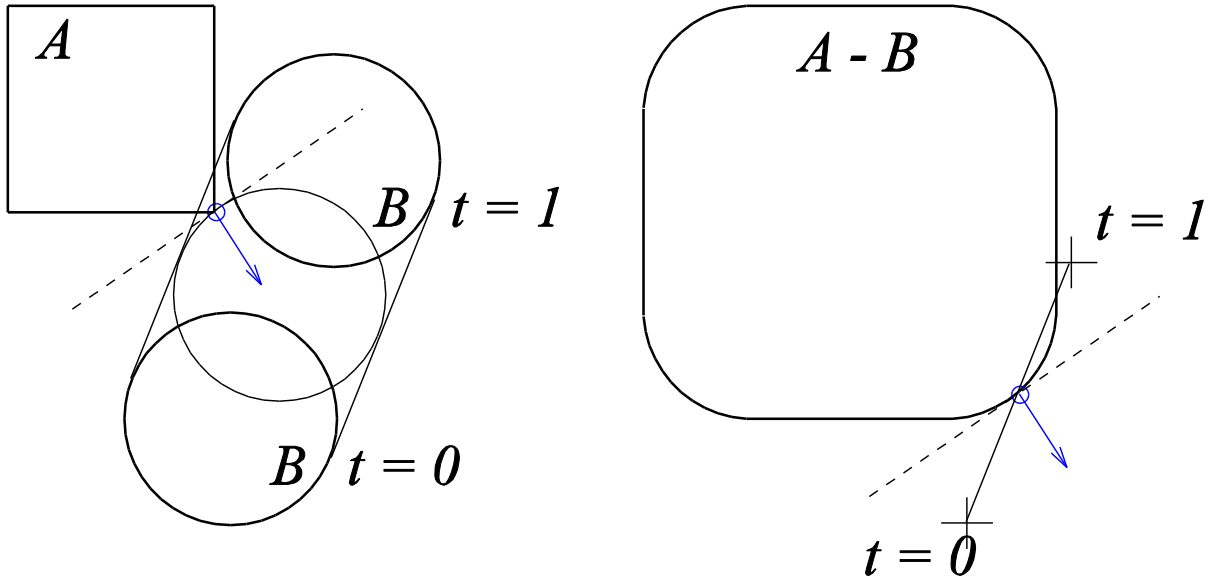
$$A \cap B \neq \varnothing \Leftrightarrow \mathbf{0} \in A - B$$

Queries (Cont'd)

- Whether an when two translated objects ever come in contact boils down to a ray query from the origin onto the CSO.
- For A translated over $\mathbf{s}$ and B over $\mathbf{t}$, the ray is cast from the origin along $\mathbf{r} = \mathbf{t} - \mathbf{s}$.

$$\min\{\lambda : \lambda\mathbf{r} \in A - B, 0 \le \lambda \le 1\}$$

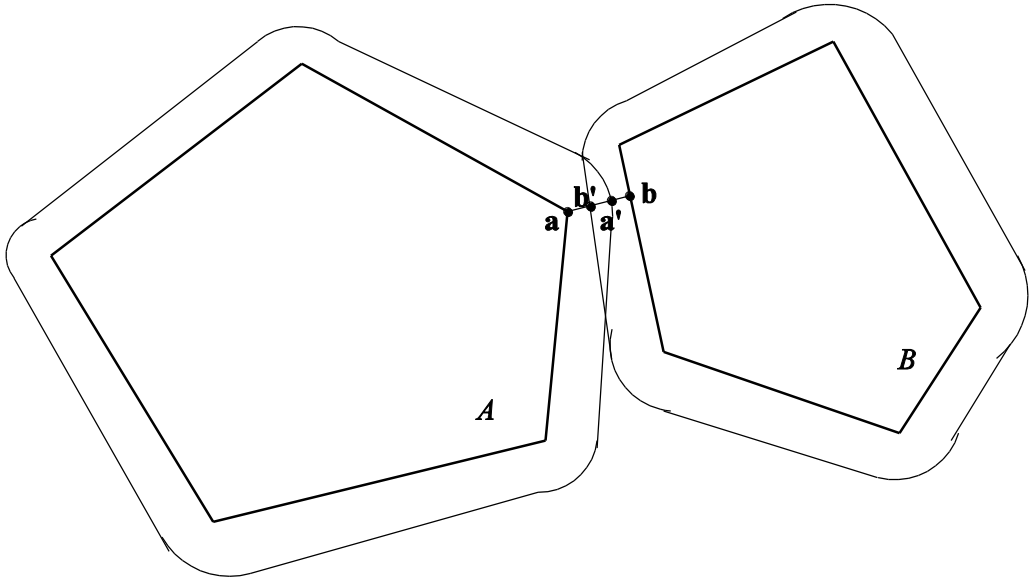
Ray Query on the CSO



Resting Contacts

- Contact data for resting contacts are obtained through a hybrid approach.
- Objects are dilated slightly to add a skin.
- For interpenetrations that are only skin-deep the closest points of the “bones” give us the contact data.

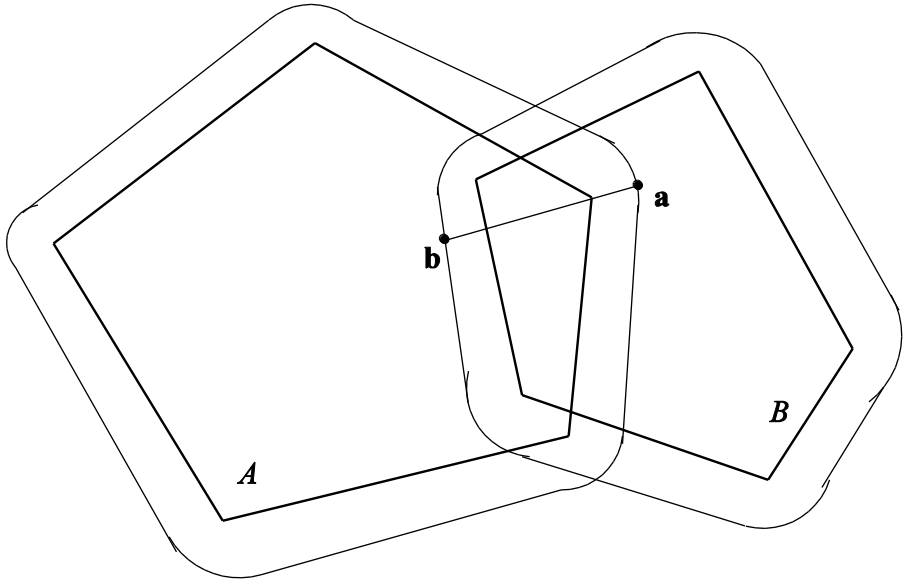
Shallow Interpenetrations



Resting Contacts

- For deeper interpenetrations contact data are obtained from the penetration-depth vector.
- This should only be necessary in emergencies.

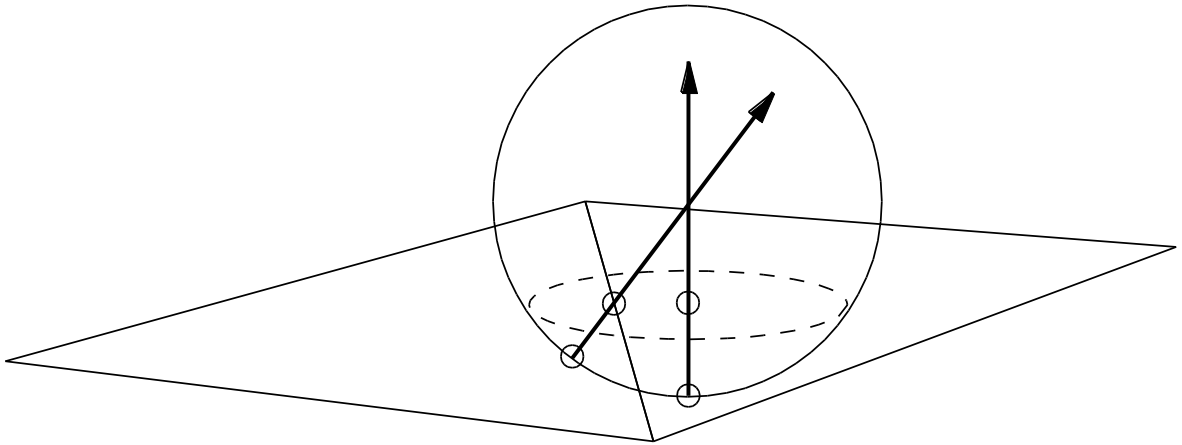
Deep Interpenetrations



GJK Saves The Day

- GJK is an iterative method that computes closest points.
- The GJK ray cast can perform continuous collision detection.
- The *expanding polytope algorithm* (EPA) returns the penetration depth vector.

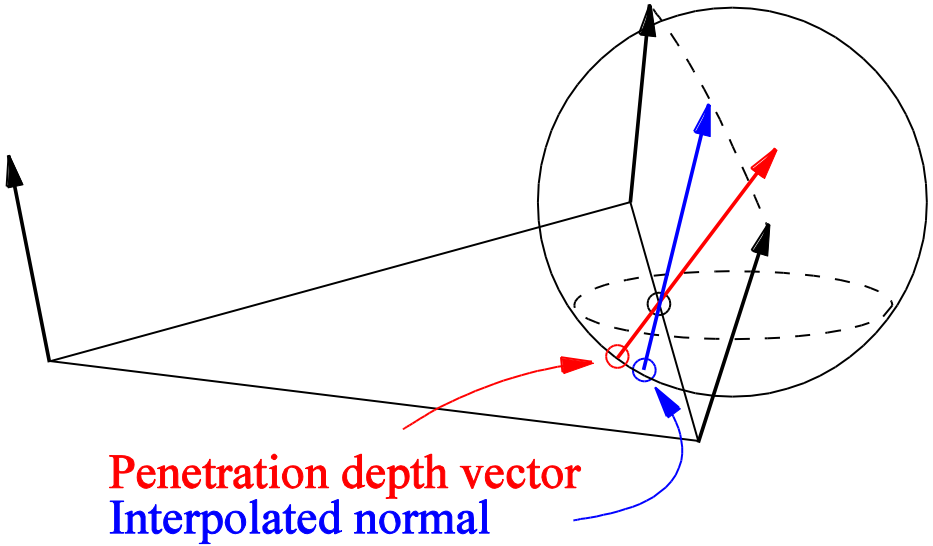
Meshes Have Bumpy Edges



Solving Bumpy Edges

- GJK returns barycentric coordinates of the closest point.
- Use these coordinates to interpolate the vertex normals.
- Similar to Phong shading: Use a normalized lerp.

Smooth Interpolated Normals



References

- Eran Guendelman, Robert Bridson, and Ronald Fedkiw. *Nonconvex rigid bodies with stacking*. In Proc. SIGGRAPH, 2003.
- Michael B. Cline and Dinesh K. Pai, *Post-Stabilization for Rigid Body Simulation with Contact and Constraints*. In Proc. ICRA, 2003

References (Cont'd)

- Chris Hecker. *How to Simulate a Ponytail.* Game Developer Magazine*, 2000.* `chrishecker.com`
- Gino van den Bergen. *Collision Detection in Interactive 3D Environments.* Morgan Kaufmann Publishers, 2004.
- Gino van den Bergen. *Smooth Mesh Contacts with GJK*. In *Game Physics Pearls,* A K Peters, 2010.

Thank You!

- For papers and other information, check:

  www.dtecta.com
  www.gamephysicspearls.com