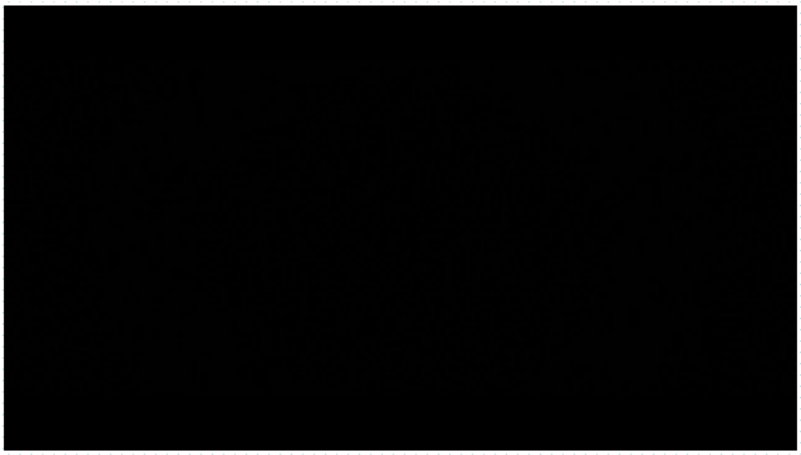
Physics Engines

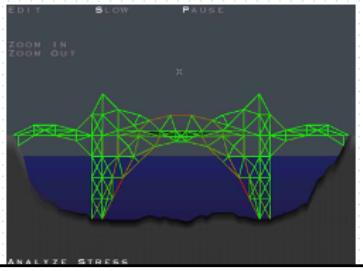


Jeff Wilson
jeff@imtc.gatech.edu

BeamNG

What is a Physics Engine?

- Provides physics simulation in a virtual environment
- High Precision vs. Real Time
- Real Time requires a lot of approximations
- Can be used in creative ways
  - Bridge Builder
  - Trials

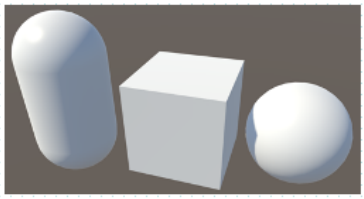


Real Time Physics Engines

- Havok (commercial), Newton (closed source), Open Dynamics Engine (ODE) (open source), Nvidia PhysX (previously Aegia with accelerator board, who was previously NovodeX)
- Unity
  - PhysX integration

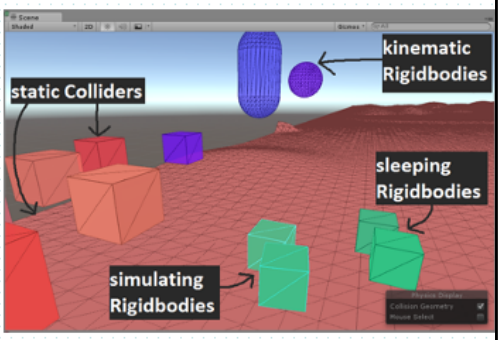
Components

- Bodies
  - Box, sphere, capsule, triangle mesh etc.
- Connectors
- Forces
- Constraints
- Collision Detection
  - Can be used by itself with no dynamics

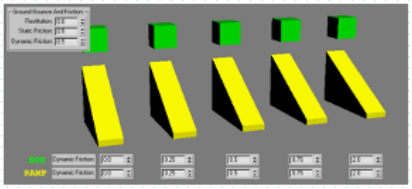


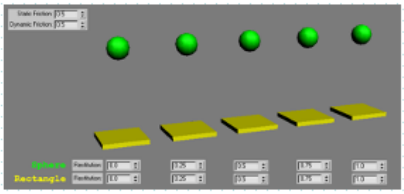
Bodies

- Rigid
- Movable (Dynamic)
  - Kinematic – programmatic or animation control
- Unmovable (Static)
  - Like infinite mass
- Enabled/Disabled
- Sleeping/Awake
- Layer/Group Membership



Bodies

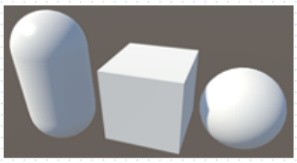


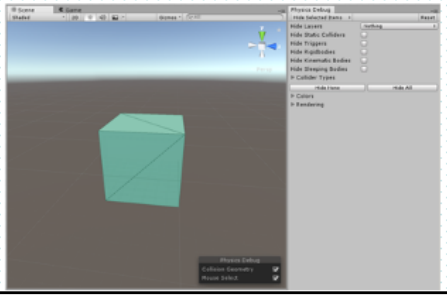


- Position
- Orientation
- Velocity
- Angular Velocity
- Properties
  - Mass, dynamic/static friction, restitution (bounciness), softness
    - Anisotropic friction (skateboard)
  - Mesh shapes with per triangle materials (terrain)

Collision Geometry

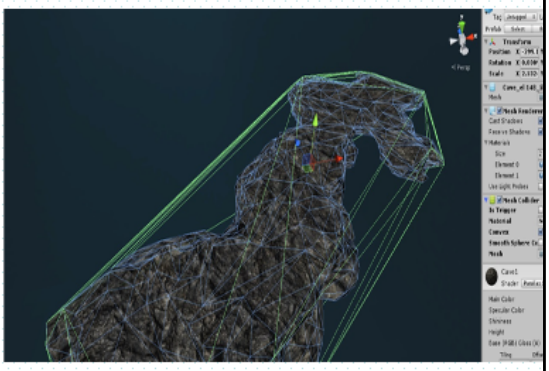
- Collision hulls reduce complexity of collision calculation
  - Made from the primitives mentioned before
    - Boxes, capsules etc.
- How your model is encapsulated determines accuracy, and computational requirements

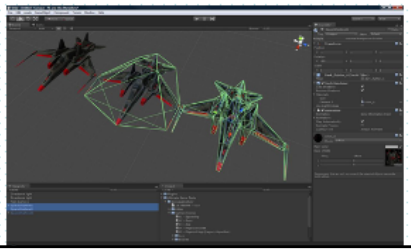




Mesh Collision Hulls

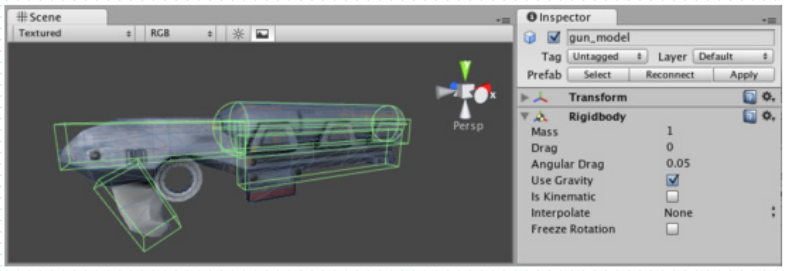
- Often required to be convex (esp. if non-static) for performance
- Minimum convex hull to convert concave source





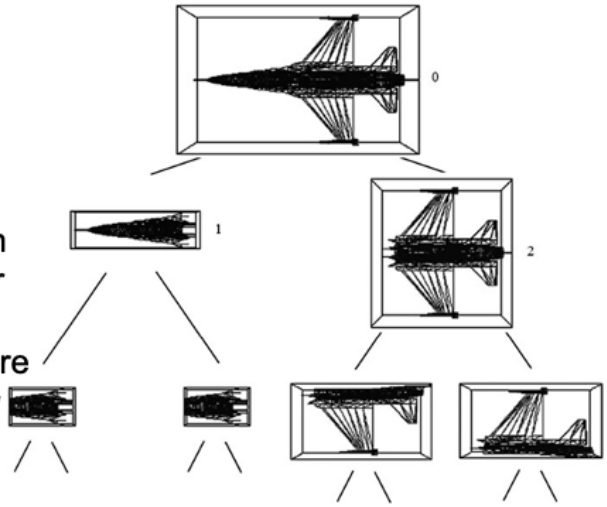
Compound Colliders

- Compound colliders (with fixed joints/connectors) can produce concave effect
- Easy to make in Unity!



Hierarchical Collision Hull

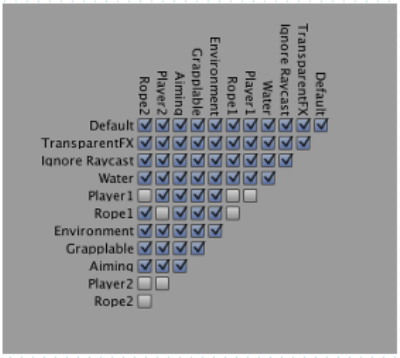
- Simplify complex geometry with hierarchy
- Large simple bounding collision hull tested first for intersection
- Work down to more specific geometry



Collision Geometry

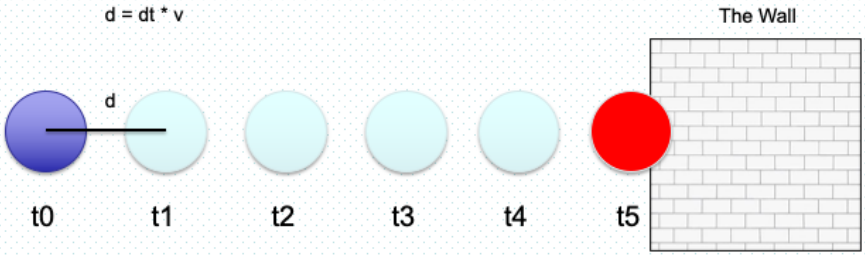
- Skin width – deal with jitter, relax collision constraints (PhysX: contactOffset+restOffset is similar)
- Collision layers/filtering/groups
  - Tweak simulation, game play, path planning, reduce complexity

Skin is dilation of rigid body



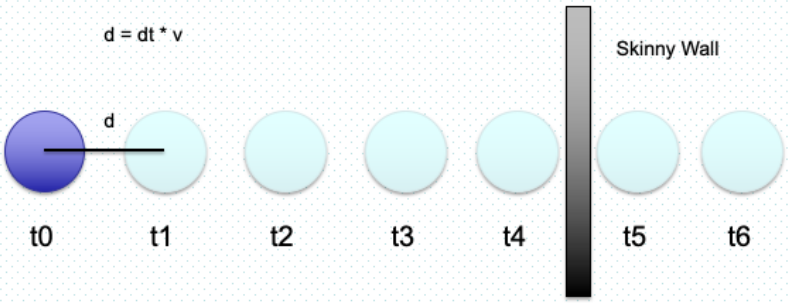
http://docs.nvidia.com/gameworks/content/gameworkslibrary/physx/guide/Manual/AdvancedCollisionDetection.html

Discrete Collision Detection

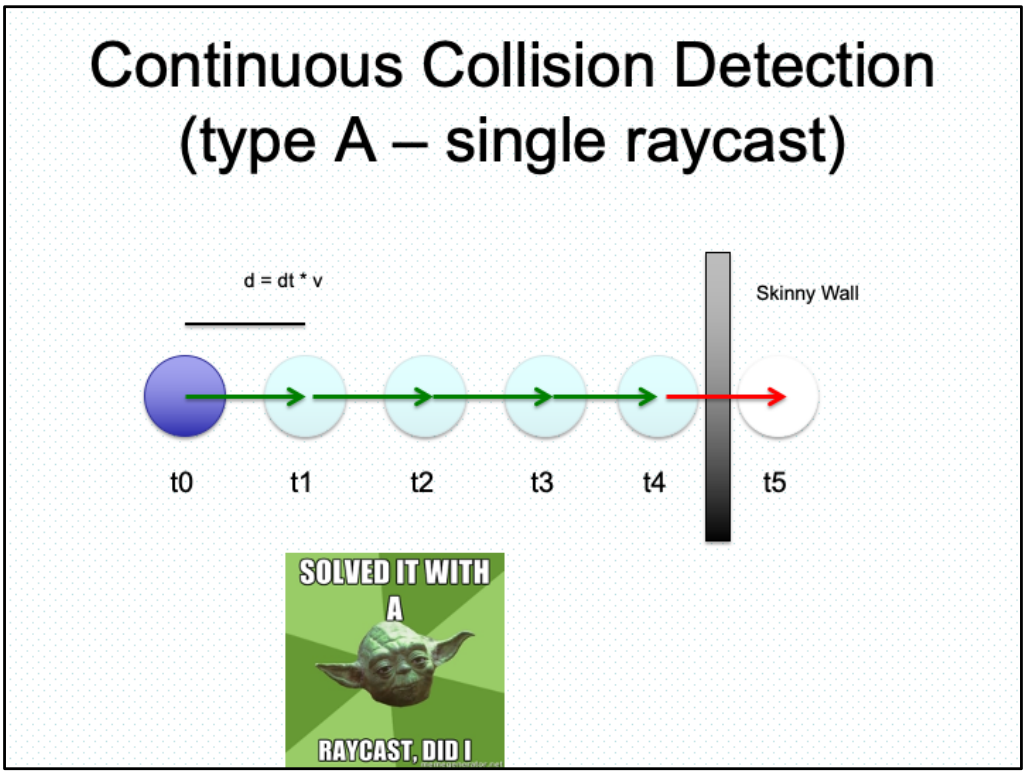


3D Geometric Intersection Tests

Discrete Collision Detection (tunneling)

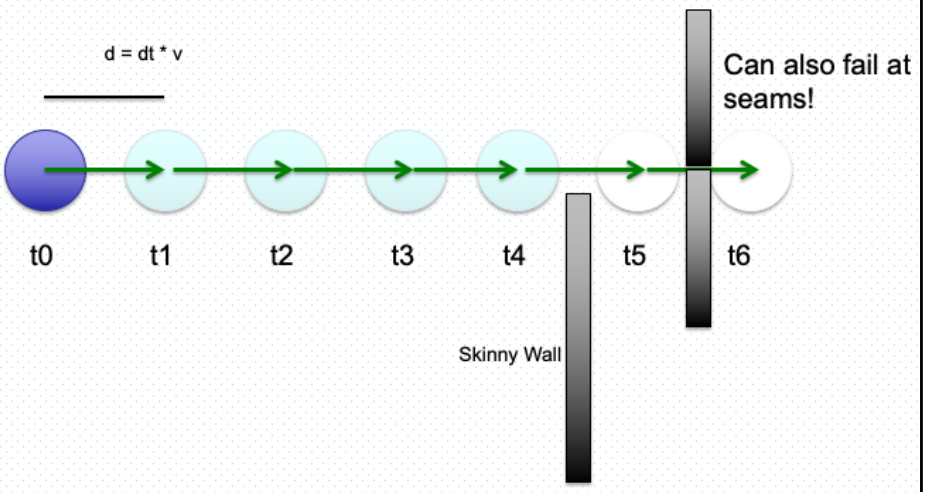


Intersection or it didn't happen

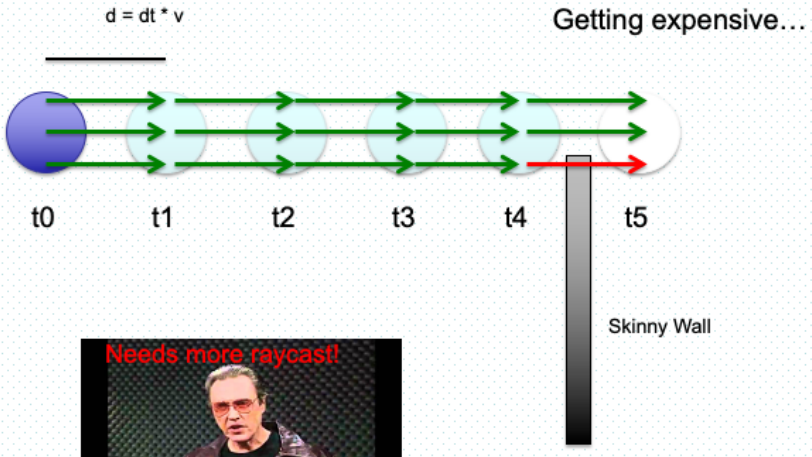


PhysX actually recommends this for those that want lower overhead than their regular continuous collision detection. I think they provide some code to do so

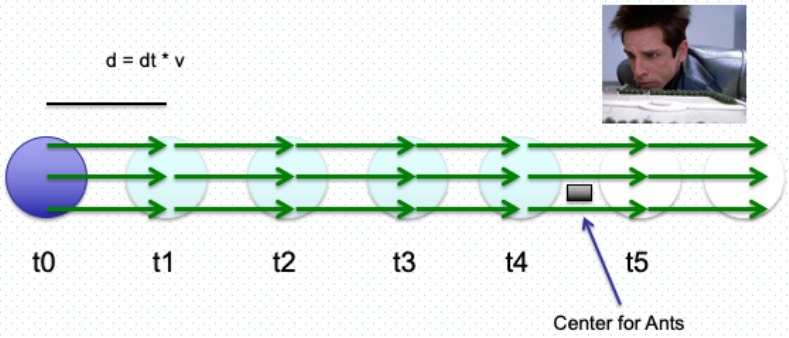
Continuous Collision Detection (type A – single raycast) Failure



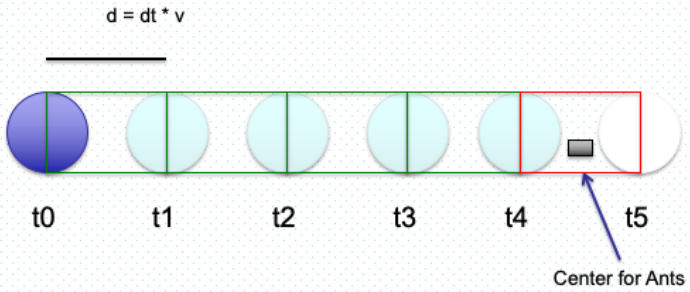
Continuous Collision Detection (type B – multi-raycast)



Continuous Collision Detection (type B – multi-raycast) Failure

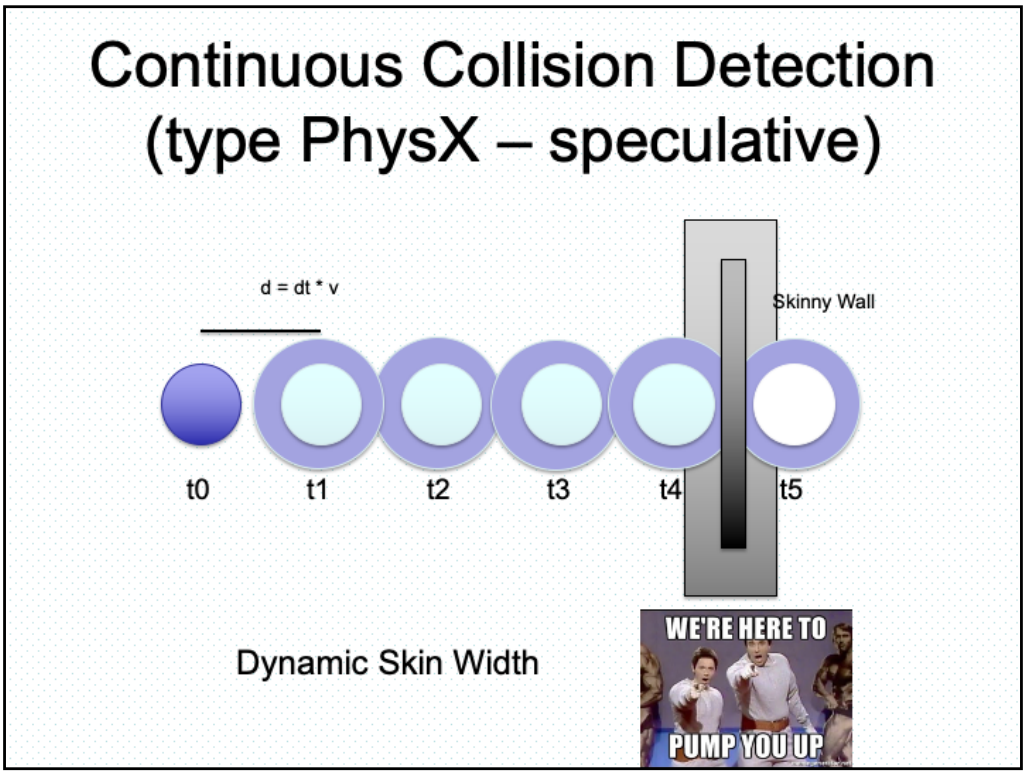


Continuous Collision Detection (type C – Silhouette Extrusion)



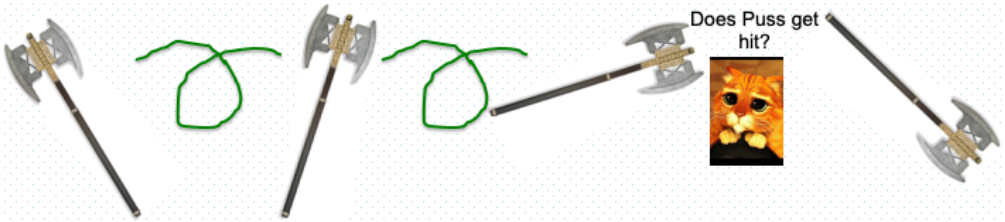
This approach is easy with spheres (think rotation), but other shapes?

Things are getting pretty expensive now

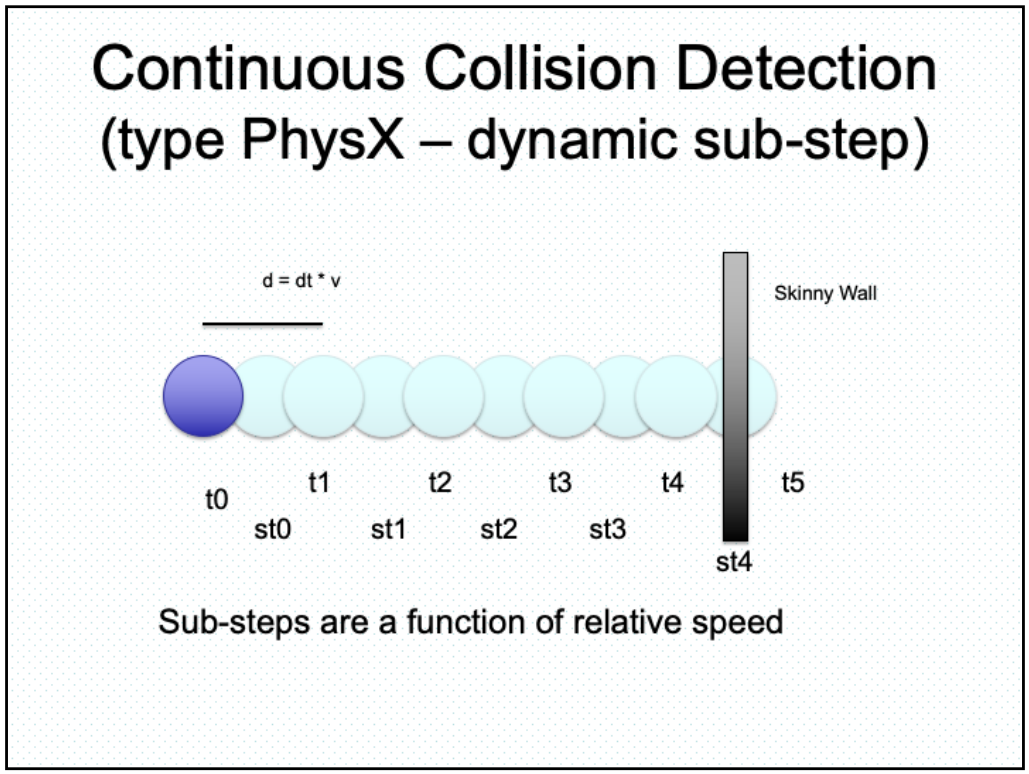


Skin width grows/shrinks. Can fail in PhysX under hard acceleration

And My Axe! (Rotating & Translating Colliders)



- Fast rotations make all of the previous techniques tricky!
- Speed limit on rotation as crutch
- Unity 5.x Rigidbody.maxAngularVelocity
  - WILL AFFECT CARS WITH TORQUE MOTORS



http://docs.nvidia.com/gameworks/content/gameworkslibrary/physx/guide/Manual/AdvancedCollisionDetection.html

Knowing the objects size, a speed threshold can be identified where tunneling might occur. When that speed is exceeded, start sub-stepping but otherwise use discrete collision detection

This approach works well with rotations too, provided we don't allow rotations that are too fast

Collision Dynamics

- How to resolve a collision?

Penalty Force Method

- Allow interpenetration, then correct with a force
- Only consider deepest penetration
- Can be unstable, never reaching a resting state due to oscillations
- Used by Trespasser

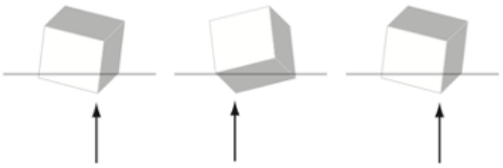


Fig. 1. The result of the application of restoring forces to the point of deepest penetration on successive simulation steps. The cube oscillates on the planar surface as a result.

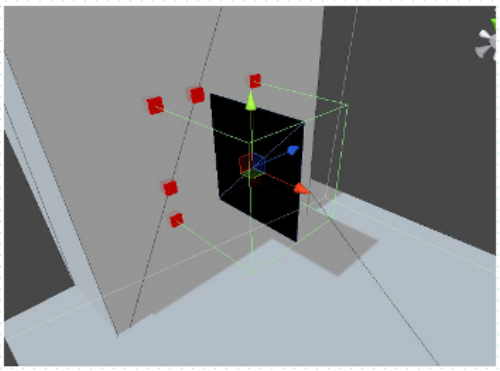
Improvement on Penalty Force

1. Relax contact criteria via skin width
2. Consider **all** collision contacts (not just deepest)
3. Find a solution that addresses all constraints introduced by the collision contacts

Describing PhysX at a high level here

All Contact Points

- Surface area contacts simplified to perimeter vertices
- Contacts as points:
  - Vertex to surface
  - Edge to edge
  - Edge to curved surface
  - ???



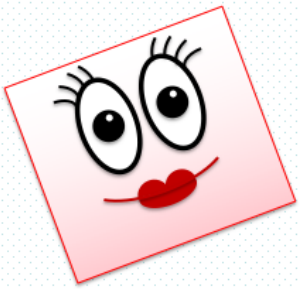
Spheres, cylinders, and capsule ends are unique cases as compared to triangles

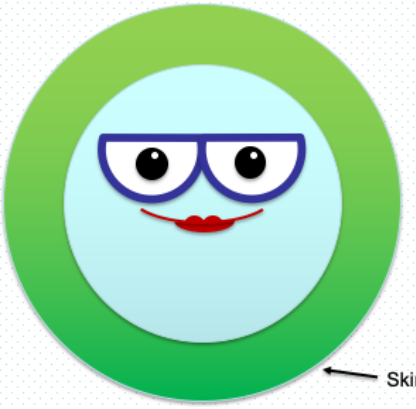
Skin Width

- Contact identified upon skin interpenetration
- Soft collision correction as skin interpenetrates, becomes stronger and stronger as further interpenetration
- Great for friction modeling and reducing jitter

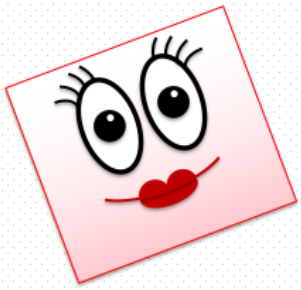
Friction modeling benefits from fact that skin interpenetration can persist across simulation steps and therefore contact point persist

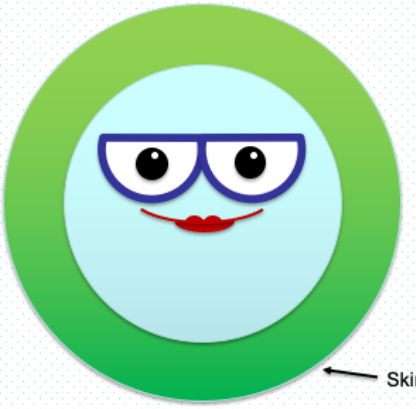
Edgry and Rolonda

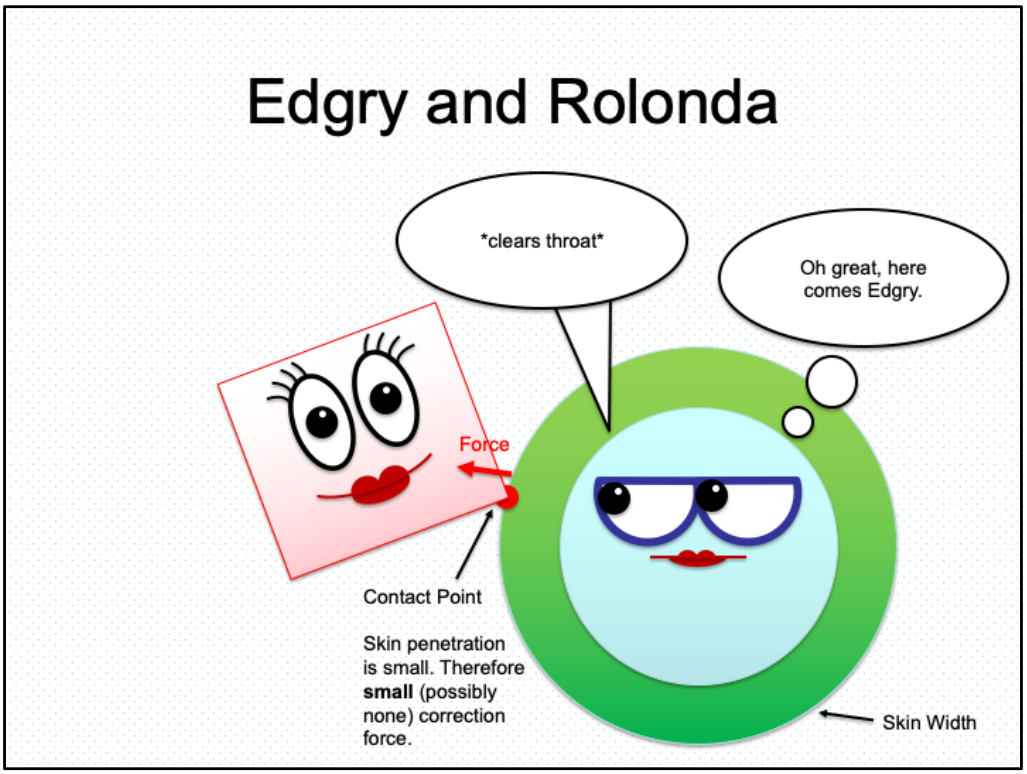




Edgry and Rolonda



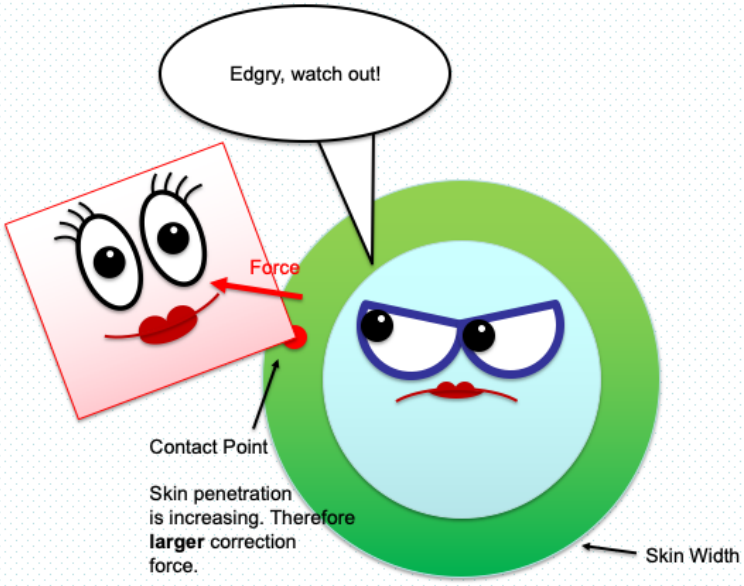




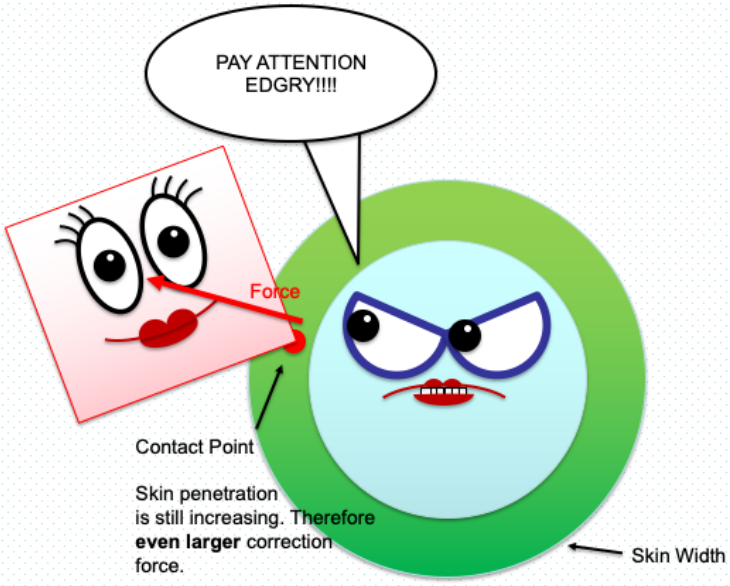
http://docs.nvidia.com/gameworks/content/gameworkslibrary/physx/guide/Manual/AdvancedCollisionDetection.html

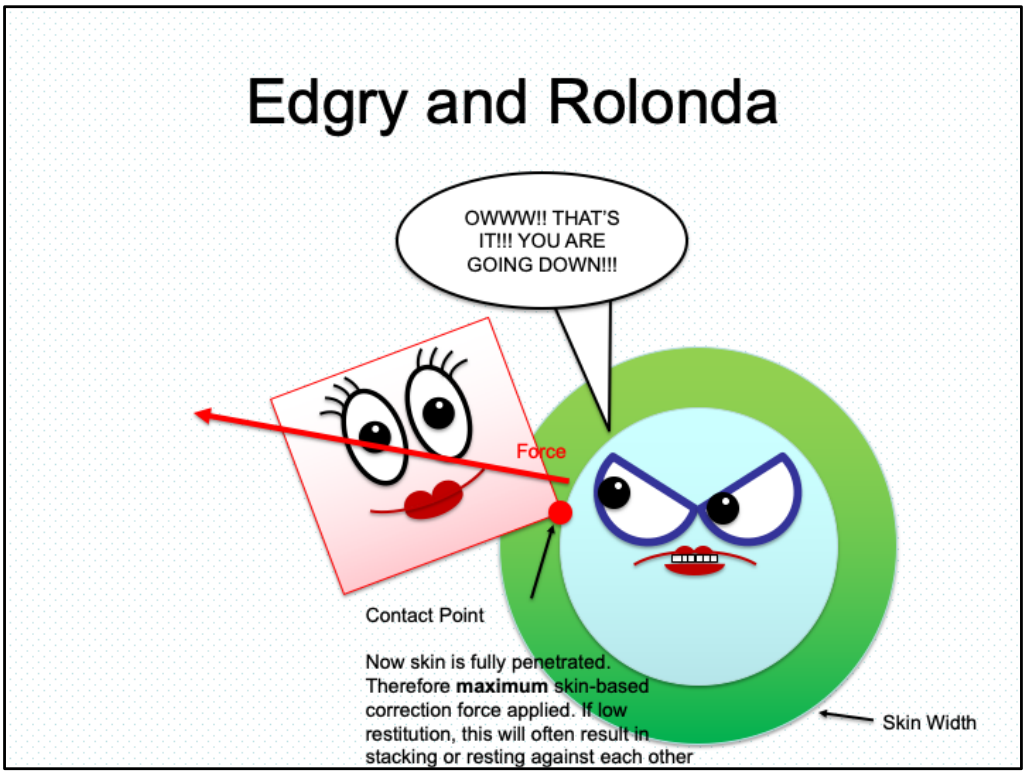
The generation of contact points does not however mean that a large impulse will immediately be applied at these locations to separate the shapes, or to even prevent further motion in the direction of penetration. This would make the simulation jitter unless the simulation time step is selected to be tiny, which is not desirable for real time performance. Instead, we want the force at the contact to smoothly increase as penetration increases until it reaches a value sufficiently high to stop any further penetrating motion.

Edgry and Rolonda



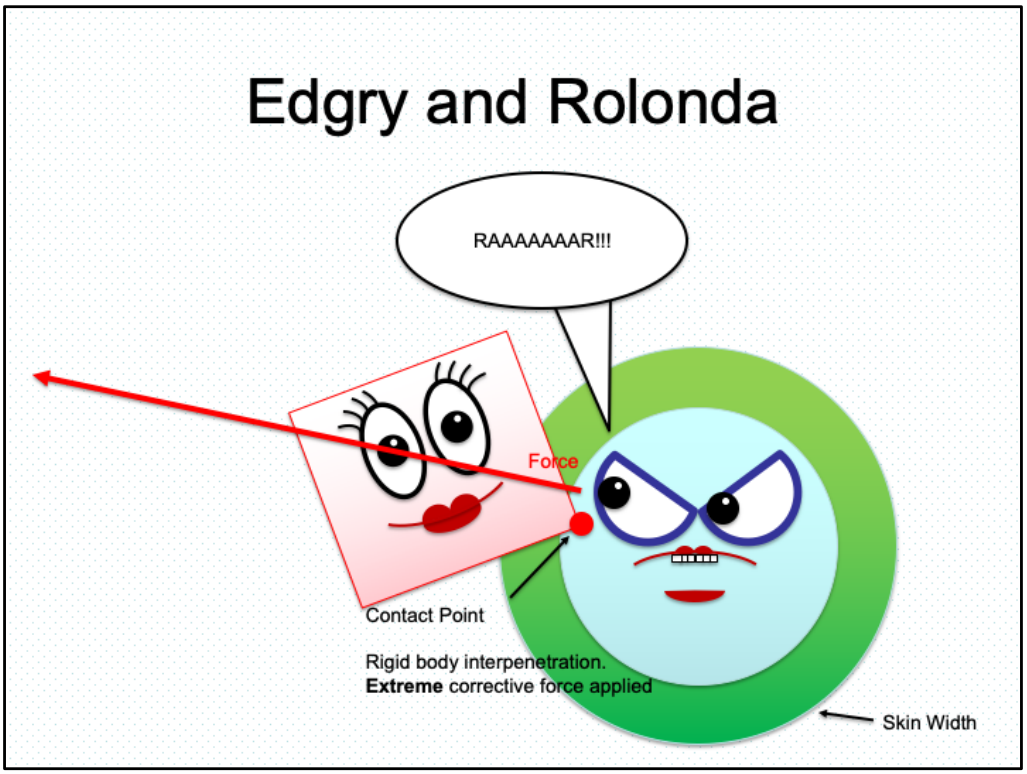
Edgry and Rolonda





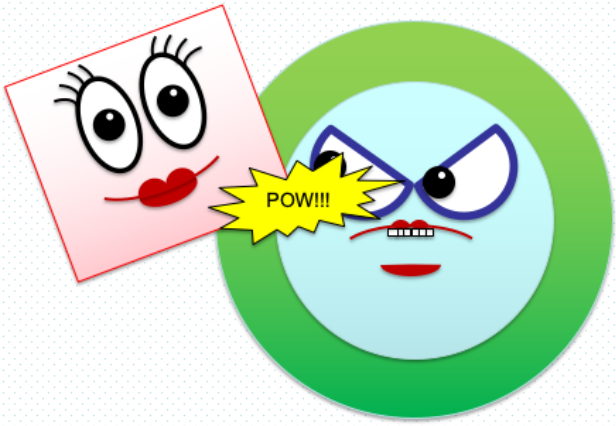
This should be at “rest distance”

The distance at which this maximum force is reached is the restDistance, because at this distance two shapes stacked on each other will reach static equilibrium and come to rest.



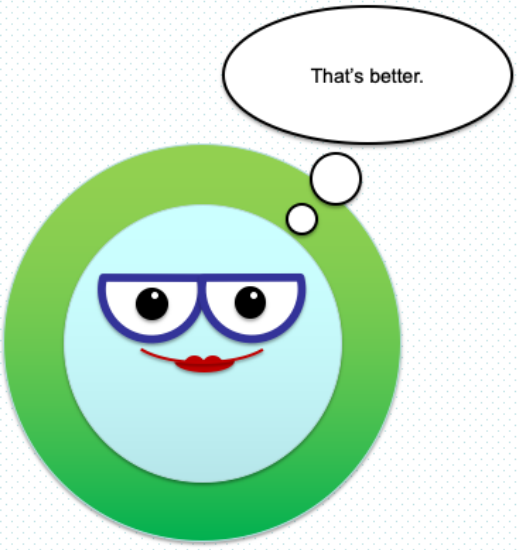
When the shapes are for some reason pushed together so much that they have a distance below restDistacnce, an even greater force is applied to push them apart until they are at restDistance again. The variation of force applied as the distance changes is not necessarily linear, but it is smooth and continuous which results in a pleasing simulation even at large time steps.

Edgry and Rolonda



Edgry and Rolonda

~~Edgry and~~ Rolonda



Common Bug

Sea of Thieves

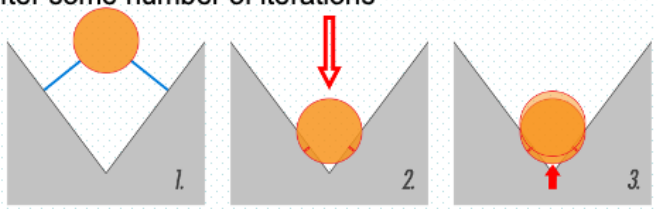
Likely scenario:

Kinematic sinking animation until the animation completes, then ship falls for a bit before being deleted.

However, with shallow water, ocean floor is intersected by the animation translation downwards. Turning physics back on, ship gets launched

Physics Constraint Solver (for collisions)

- RT solutions usually iterative and based on Linear-Complementary Problem (LCP) with Jacobian constraints
  - Apply corrective "sequential impulses" for each contact point independently
  - Determine error for each contact if all corrections were applied
  - Iterate and determine correction refinements
  - Stop after some number of iterations



Jacobian – like a pre-derived path of least resistance to enforce a constraint (e.g. a direction vector in constraint space)

A function of contact point center of mass and degrees of freedom (position and orientation)

http://www.nvidia.com/object/physx_knowledge_base.html

What kind of solver does Novodex use, specifically?

The rigid body constraint solver is an iterative solver.

This means that each constraint applies an impulse that satisfies the

constraint without regard for any other constraint in the system. After all the constraints have been 'solved' in this way, one iteration has taken place.Of course, it is inevitable that solving some constraints has counteracted the solving impulses of of other constraints to some degree, though usually the resulting 'error' is smaller than the original. By performing multiple iterations, these errors can be reduced to invisible amounts.

Increasing the number of constraints increases the computation by O(n)

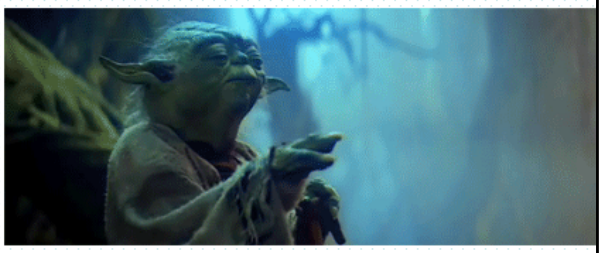
Jacobian matrix solvers tend to increase in complexity by O(n2) for each

additional constraint.

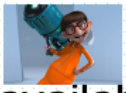
More here:
https://www.cc.gatech.edu/~karenliu/RTQL8_files/lcp.pdf

Forces

- Force
  - constant
- Impulse
  - Instantaneous



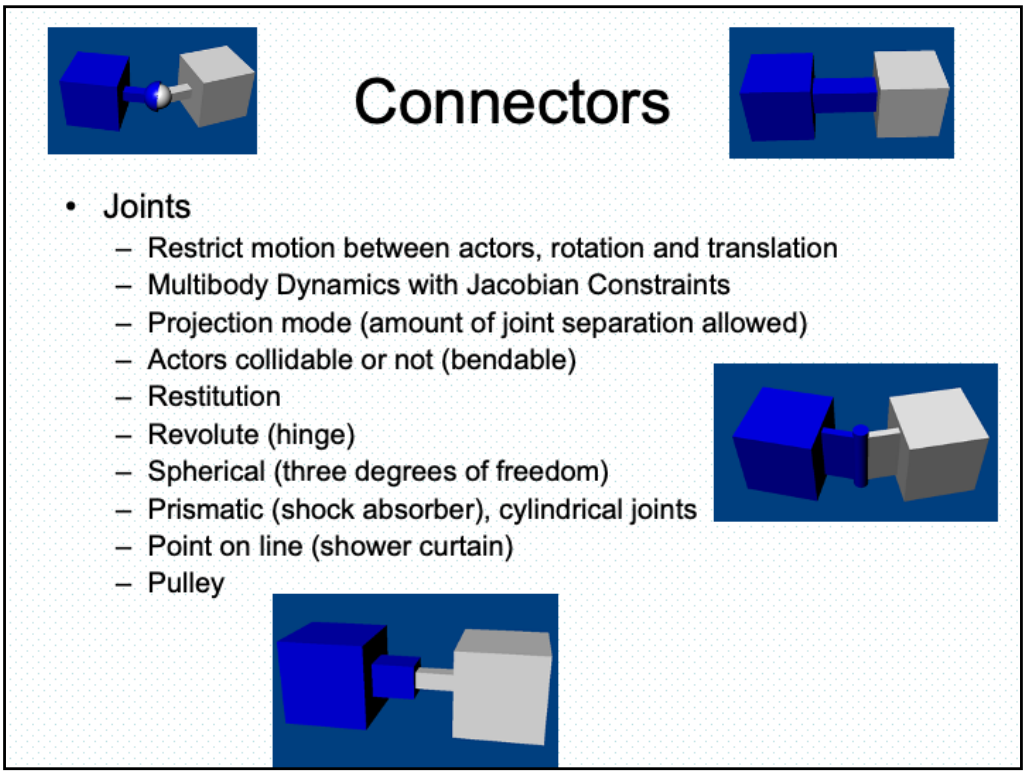
- Vector - direction **and** magnitude



- Acceleration and smooth options available
- Torque (spin)

Physicsaction_applyforce.dir

Physicsaction_applyimpulse.dir, modify impulse to do 100 in z



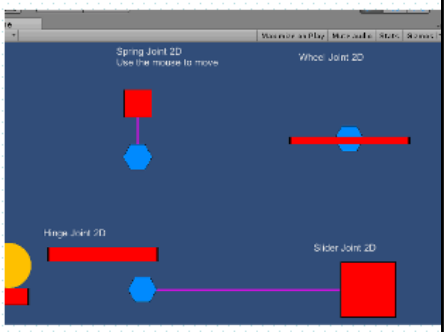
More info:

http://www.cs.cmu.edu/~baraff/sigcourse/

https://www.cc.gatech.edu/~karenliu/RTQL8_files/dynamics.pdf

Connectors

- Spring
  - Joint with natural resting angle
  - Force
  - Damping
- Joint Motor (apply relative torque)
- Breakable joints (max force, max torque)



Constraints

- Hard
  - Never violated
  - In reality will be violated by errors in simulation
  - Unity uses a notion of “projection” to force violated constraints back to an allowed pose
- Soft
  - Designed to be violated
- Joint constraints
  - Degrees of freedom, linear/angular amounts
- Freeze flags (position and rotation)
- Linear and angular damping
  - In absence of friction and collision (wind resistance)

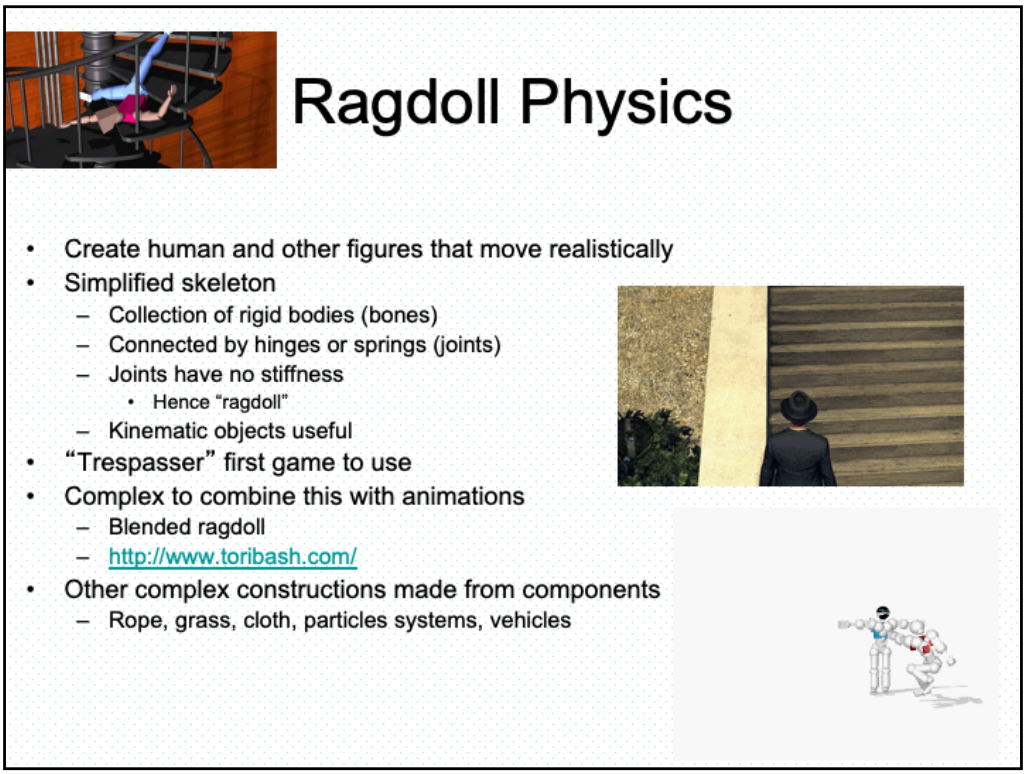
(Some constraints are clearly just a suggestion)

AngularDashpot

LinearDashpot

Deactivation

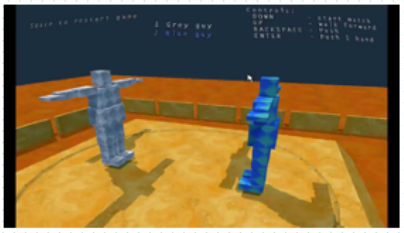
- Limit actors that are active in simulation
- Sleep linear and angular velocity
- Bounce threshold (stops vibration, ends bounces when no longer seen)
- Skin-width related settings
  - How reactive objects are to collision and how easily awakened



Toribash - https://www.youtube.com/channel/UCKAqFCwB1BUZ1dN2qEAfn1w

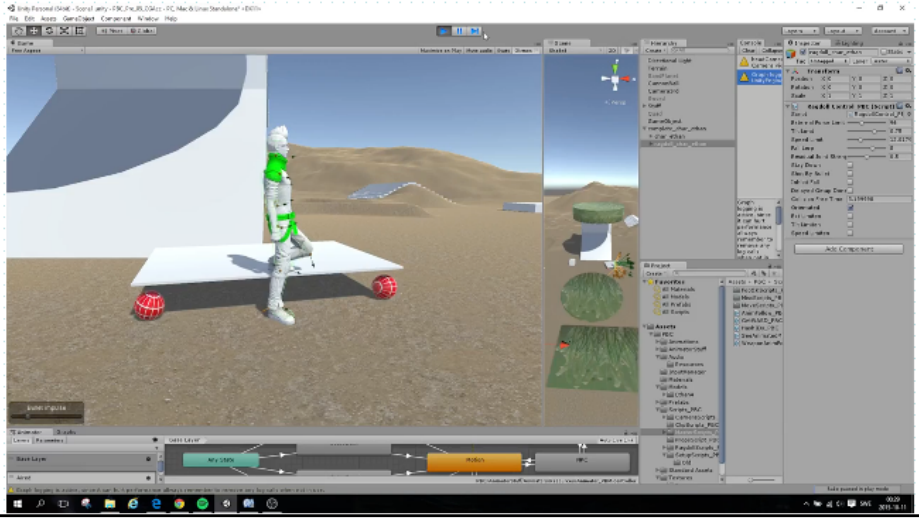
Why not completely physically simulated characters?

- AI problem
- Balance
- Recovery



Hybrid: Animation with Simulation

- Can blend approaches



Putting it all in motion

- Set gravity vector
- Make some kind of ground surface that doesn't fall
- Apply forces to bodies
- Adjust joint parameters as necessary
- Call collision detection
- Step the simulation based on time
  - Tradeoff of speed and accuracy
  - Substeps
- Keep the graphics object and physics object in synch
- Deactivate objects (manually or automatically)

SamplePhysics example

Why does my simulation look wrong?

- Scale of your objects and world
  - May look wrong
  - May cause anomalies in simulation
- Disconnect between graphics and physics world
- Slow downs due to number of active objects
- Properties and their interactions also may result in strange results
  - Mass, Friction, magnitude of forces etc.
  - Velocities too fast for timestep
  - Whips
- Collision detection may break down
  - interpenetration
- Must move objects using the functionality of the physics engine. No "hand of god" behavior