Stable, Robust, and Versatile Multibody Dynamics Animation

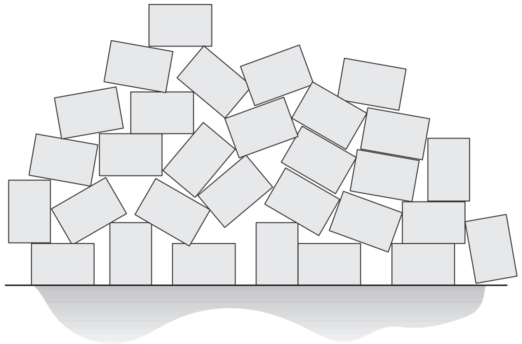
Kenny Erleben

Department of Computer Science

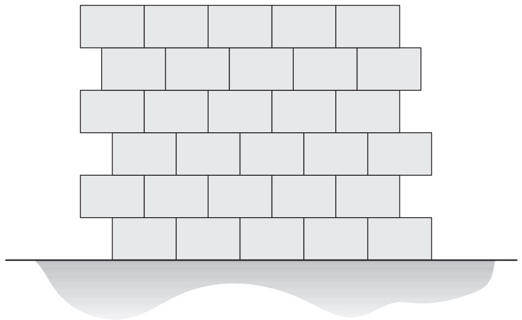
University of Copenhagen

The Problem

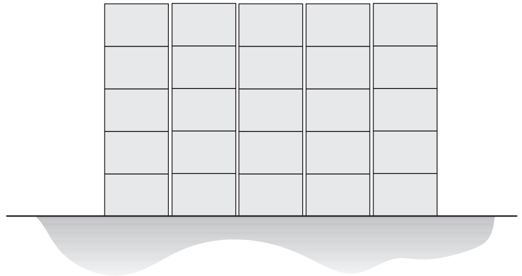
Large scale (today ≥ 1000 objects).



Dense random stacking



Dense structured stacking



Sparse Stacking

- Dense $\Rightarrow$ more difficult.
- Random looks OK if top-most objects OK.
- More sparse $\Rightarrow$ less dependent constraints $\equiv$ faster convergence.

Notice: Dense $\Rightarrow$ more dependent constraints.

- Stable $\Rightarrow$ Handling Physical Unstable and Ill-posed.
- Robust $\Rightarrow$ Handling Degenerate and Faulty Cases.
- The Attack of The Malicious User!

My Solution

Ingredients to achieve high-performance:

- Velocity-based complementarity formulations.
- Leap-frog like time-stepping scheme.
- Iterative LCP solver.
- Error-correction by projection.
- Shock-propagation, followed by a smoothing correction phase.

Complementarity Formulation

From physics

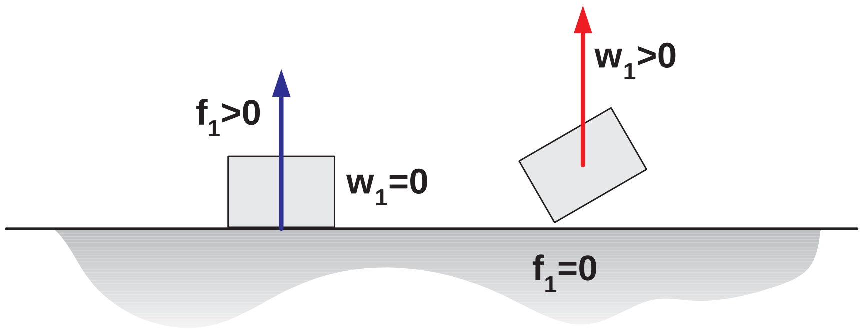
- Normal force repulsive ($\lambda_1 \geq 0$).
- Normal force zero at separation ($\lambda_1 = 0$).

If bodies are moving apart

$$w_1 = J_{\text{row}_1} \vec{u} > 0 \Rightarrow \lambda_1 = 0, \tag{1}$$

If bodies are resting

$$\lambda_1 > 0 \Rightarrow w_1 = J_{\text{row}_1} \vec{u} = 0. \tag{2}$$

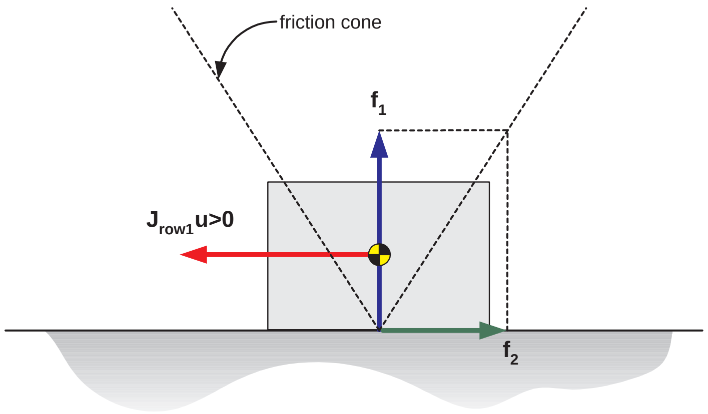


Either λ_1 is zero and $J_{\text{row}_1} \vec{u}$ is non-zero or vice-versa.

$$w_1 = J_{\text{row}_1} \vec{u} \geq 0 \quad \text{compl.} \quad \lambda_1 \geq 0 \tag{3}$$

Complementarity Formulation

Coulomb's Friction Model.

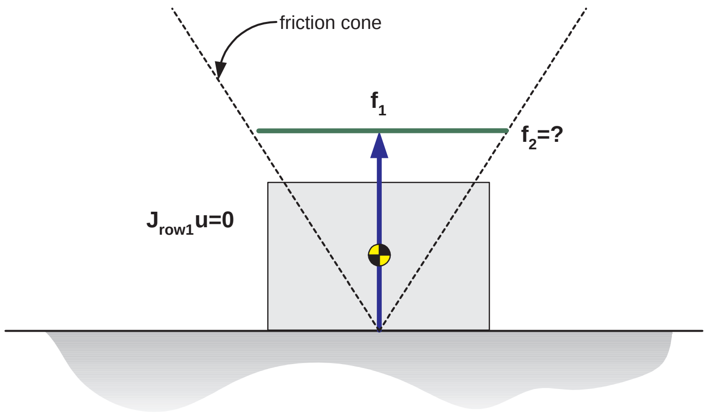


Dynamic friction:

$$w_2 = J_{\mathsf{row}_2}\vec{u} > 0 \Rightarrow \lambda_2 = -\mu\lambda_1, \tag{4}$$

$$w_2 = J_{\mathsf{row}_2}\vec{u} < 0 \Rightarrow \lambda_2 = \mu\lambda_1, \tag{5}$$

μ is the coefficient of friction.



Static friction:

$$w_2 = J_{\mathsf{row}_2}\vec{u} = 0 \Rightarrow \lambda_2 <\mid \mu\lambda_1 \mid, \tag{6}$$

Similar constraints for $w_3 = J_{\mathsf{row}_3}\vec{u}$ and λ_3.

Complementarity Formulation

Combine discritization of $\mathbf{M}\dot{\vec{u}} = \vec{f}_{\text{ext}} - J^T\vec{\lambda}$ with $\vec{w} = J\vec{u}$

$$\vec{w} = \underbrace{J\mathbf{M}^{-1}J^T}_{\mathbf{A}}\Delta t\vec{\lambda} + \underbrace{J\left(\vec{u}^{\,t} + \Delta t\mathbf{M}^{-1}\vec{f}_{\text{ext}}\right)}_{\vec{b}}$$

$$\vec{w} = \mathbf{A}\vec{\lambda} + \vec{b} \tag{7}$$

Notice

- Δt is moved into the Lagrange multiplier vector.
- $\vec{w}$ linear in $\vec{\lambda}$.

Further

$$\lambda_i = \vec{\lambda}_{\text{lo}_i} \Rightarrow \vec{w}_i \geq 0 \tag{8}$$

$$\lambda_i = \vec{\lambda}_{\text{hi}_i} \Rightarrow \vec{w}_i \leq 0 \tag{9}$$

$$\vec{\lambda}_{\text{lo}_i} < \lambda_i < \vec{\lambda}_{\text{hi}_i} \Rightarrow \vec{w}_i = 0 \tag{10}$$

A box linear complementarity problem (LCP) problem.

The Time-Stepping Method

Dynamics explicit time-step scheme,

$$\textbf{dynamics}\ (\Delta t) \tag{11}$$

given by

$$\vec{b} = J\left(\vec{u}^{\,t} + \Delta t \mathbf{M}^{-1}\vec{f}_{\text{ext}}\right) \tag{12a}$$

$$\mathbf{A} = J\mathbf{M}^{-1}J^T \tag{12b}$$

$$\vec{\lambda} = \textbf{lcp}\ \left(\mathbf{A}, \vec{b}\right), \tag{12c}$$

$$\vec{u}^{\,t+1} = \vec{u}^{\,t} + \mathbf{M}^{-1}J^T\vec{\lambda} + \Delta t\mathbf{M}^{-1}\vec{F}_{\text{ext}}, \tag{12d}$$

$$\vec{s}^{\,t+1} = \vec{s}^{\,t} + \Delta t\mathbf{S}\vec{u}^{\,t+1}. \tag{12e}$$

But how do we solve for the Lagrange multipliers?

Brute Force...?

Iterative LCP solver

The splitting

$$\mathbf{A} = \mathbf{L} + \mathbf{D} + \mathbf{U}$$

Iterative Gauss-Seidel

$$\mathbf{A}\vec{\lambda} = -\vec{b} \tag{13}$$

$$\left(\mathbf{L} + \mathbf{D} + \mathbf{U}\right)\vec{\lambda} = -\vec{b} \tag{14}$$

$$\vec{\lambda}^{\,k+1} = \mathbf{D}^{-1}\left(\mathbf{L}\vec{\lambda}^{\,k} - \mathbf{U}\vec{\lambda}^{\,k} - \vec{b}\right) \tag{15}$$

Loop over all variables $i \in [1..3K]$

$$\vec{\lambda}_i^{\,k+1} = \frac{\left(-\sum_{j=0}^{i-1}\mathbf{L}_{i,j}\vec{\lambda}_j^{\,k+1} - \sum_{j=i+1}^{n-1}\mathbf{U}_{i,j}\vec{\lambda}_j^{\,k} - \vec{b}_i\right)}{\mathbf{D}_{i,i}} \tag{16}$$

$$= \frac{-\mathbf{L}_{\text{row}_i}\vec{\lambda}^{\,k+1} - \mathbf{U}_{\text{row}_i}\vec{\lambda}^{\,k} - \vec{b}_i}{\mathbf{D}_{i,i}}. \tag{17}$$

Superscript = iteration number.

Iterative LCP solver

Upper and lower limits are updated if the i'th variable is a friction constraint,

$$(r = i \textbf{ mod } 3) \neq 0 \Rightarrow \vec{\lambda}_{\text{lo}_i} = -\vec{\lambda}_{\text{hi}_i} = \mu \vec{\lambda}_{i-r}, \tag{18}$$

projection step

$$\vec{\lambda}_i^{k+1} = \max\left(\min\left(\vec{\lambda}_{\text{lo}_i}, \vec{\lambda}_i^{k+1}\right), \vec{\lambda}_{\text{hi}_i}\right). \tag{19}$$

Now

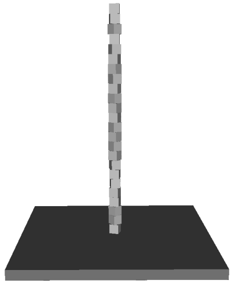
- Each row of the Jacobian have exactly 12 non-zero elements.
- $\mathbf{M}$ is 3-by-3 block diagonal matrix.

Using a sparse matrix representation for $\mathbf{M}$, $\mathbf{J}$, and $\mathbf{A}$,

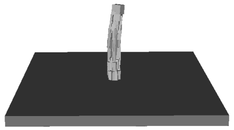
- Linear time to computing $\mathbf{A}$ and $\vec{b}$.

LCP solver Results

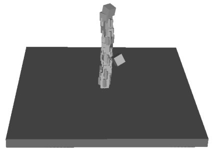
Convergence Results



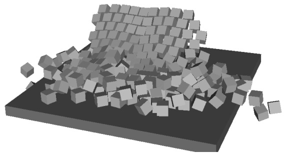
Initial position.



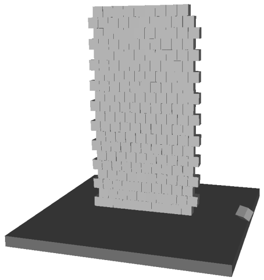
10 iterations after 4 secs.



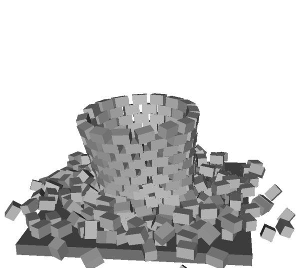
100 iterations after 4 secs.



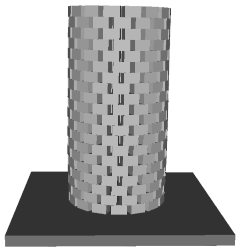
10 iterations after 4 secs.



100 iterations after 4 secs.



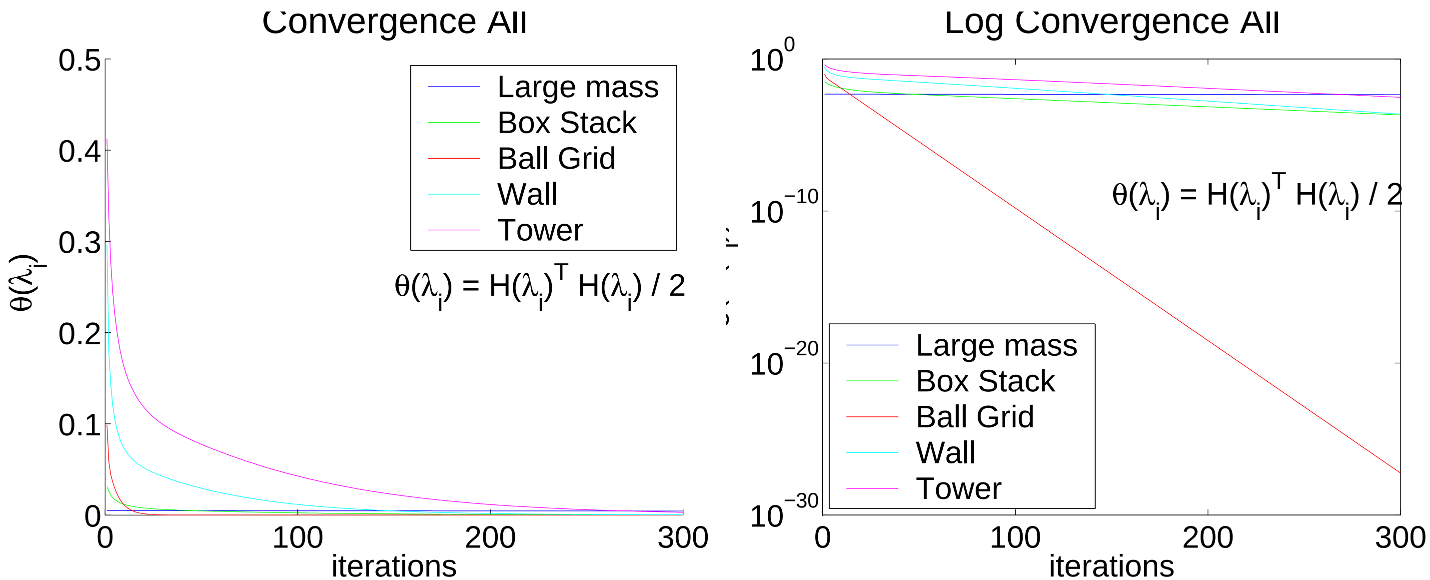
10 iterations after 4 secs.



100 iterations after 4 secs.

Convergence Rate

Convergence rate $\approx O(e^{-kn})$

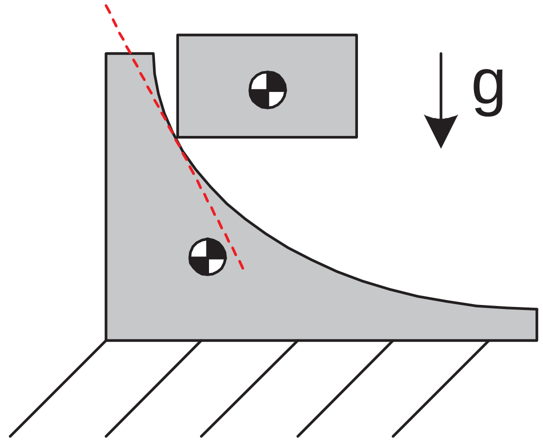


- More structure (in sense of all-pair dependency) means increasing k, less structure means decreasing k.
- Value of θ has nothing to do with accuracy, when θ is “flat” we got enough accuracy.

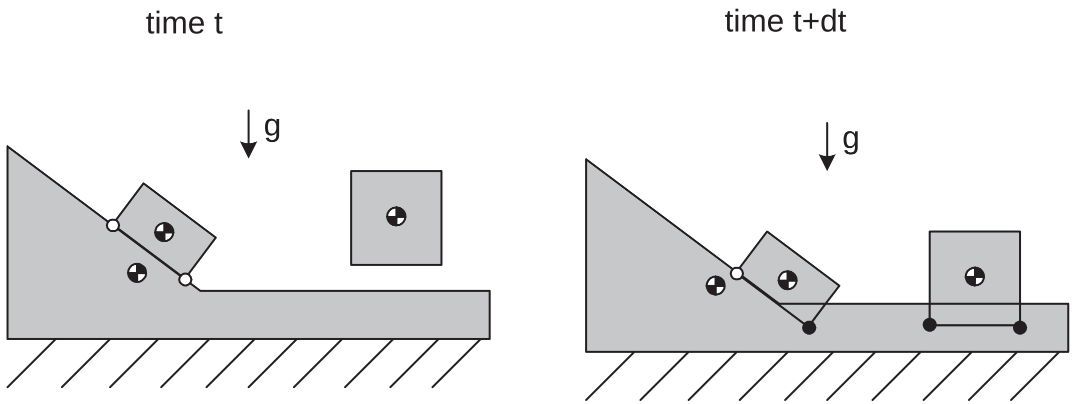
Error Correction

Penetrations are unavoidable

Curved surfaces.



Undetected contacts.



Error Correction

A first order world simulation: The equations of motion

$$\vec{F} = m\vec{a} \quad \text{and} \quad \vec{\tau} = \frac{d\vec{L}}{dt} \tag{20}$$

is replaced by

$$\vec{F} = m\vec{v} \quad \text{and} \quad \vec{\tau} = \mathbf{I}\vec{\omega}. \tag{21}$$

Used for error correction, yields the scheme,

$$\textbf{correction}\,() \tag{22}$$

given by

$$\mathbf{A} = J\mathbf{M}^{-1}J^T, \tag{23a}$$

$$\vec{\lambda} = \textbf{lcp}\left(\mathbf{A}, \vec{d}_{\text{penetration}}\right), \tag{23b}$$

$$\vec{s}^{\,t+1} = \vec{s}^{\,t} + \mathbf{S}\mathbf{M}^{-1}J^T\vec{\lambda}, \tag{23c}$$

Shock-Propagation

The general idea

```
Velocity-based-Shock-Propagation(f,dt)
  collision detection at time t
  dynamics(f*dt)
  shock-propagration(
     dynamics((1-f)*dt)
     correction()
  )
  collision detection at time t + dt
  correction()
  t = t + dt
```

Velocity Correction

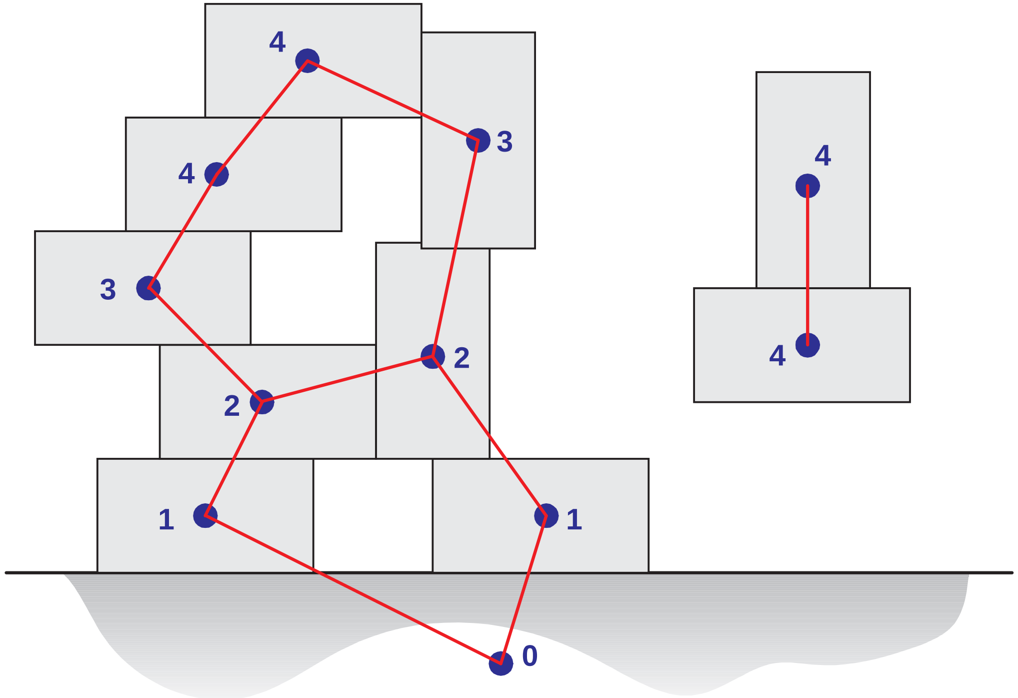
Position Correction

Smoothing

Shock-Propagation

Stack Height Definition.

- Bodies are assigned a stack-height number.
- The stack height: ♯bodies on the closest path to a fixed body.
- Free bodies are assigned the maximum stack-height.

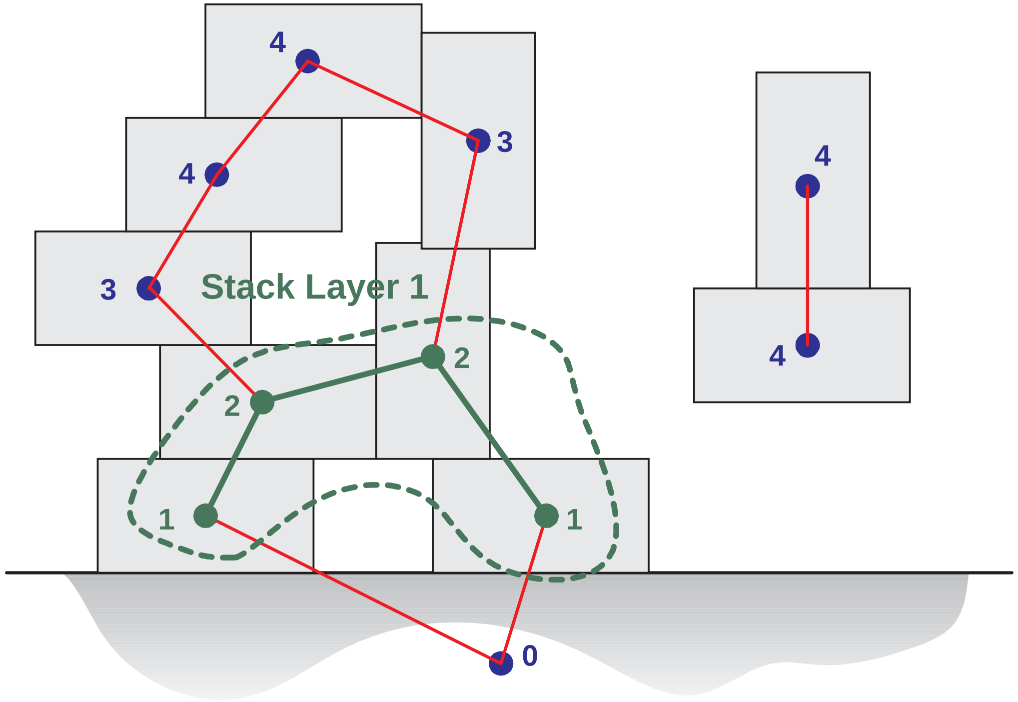


Notice: Stack-height is the path-cost of a breadth-first traversal starting at the fixed bodies.

Shock-Propagation

Stack-Layer Definition:

- Stack layer i: Bodies with stack height i and $i+1$, contact points between body-pairs
  - with stack height i and $i+1$,
  - with stack height $i+1$ and $i+1$.



Edges are given a stack layer number:

- If stack-heights are equal stack layer number is stack height minus one.
- Otherwise stack layer number is minimum.

Shock-Propagation

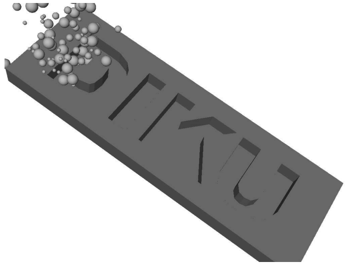
The Shock-Propagation algorithm:

- Apply an algorithm sequentially to all stack layers.
- Treat stack layers in bottom-to-top order.
- Set bottom-most bodies to fixed before applying, afterwards unfix.

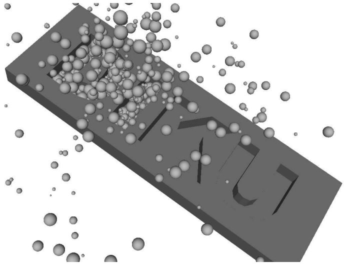
```
shock-propagation(algorithm A)
  compute contact graph
  for each stack layer in bottom up order
    fixate bottom-most objects of layer
    apply algorithm A to layer
    un-fixate bottom-most objects of layer
  next layer
```

Results

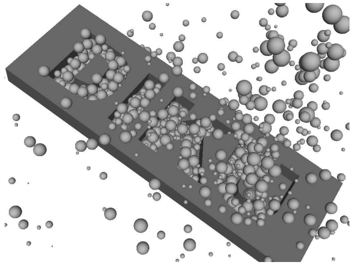
1000 balls.



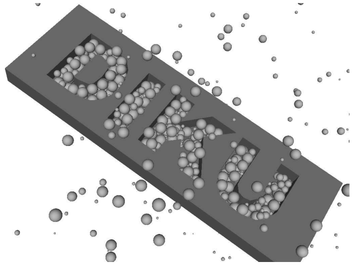
2.0 secs.



4.0 secs.

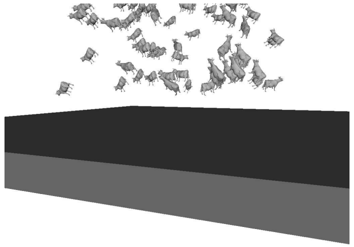


6.0 secs.

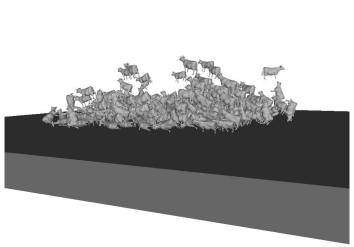


8.0 secs.

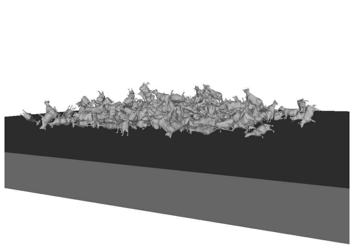
250 cows.



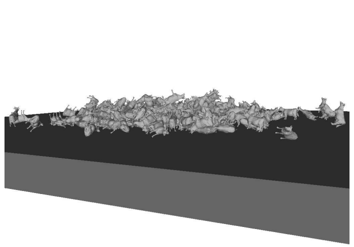
1.0 secs.



2.0 secs.

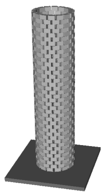


3.0 secs.

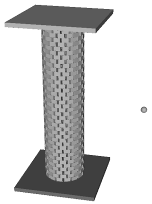


4.0 secs.

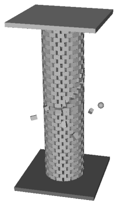
Roof and canon-ball.



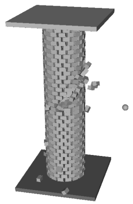
2.0 secs.



4.0 secs.

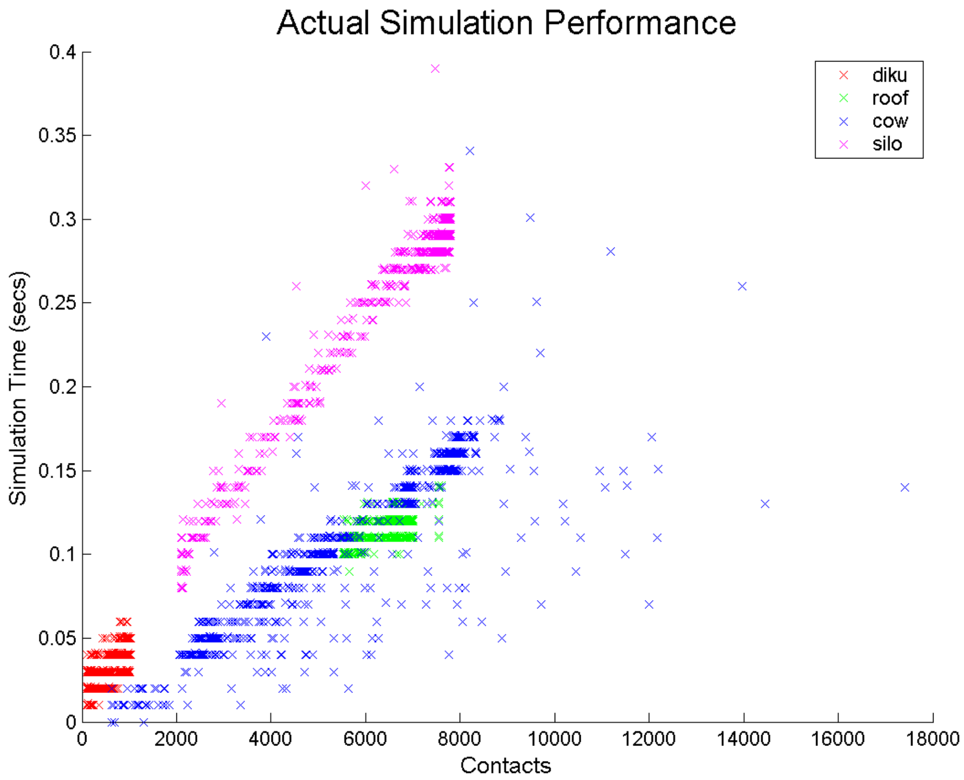


6.0 secs.



8.0 secs.

Results



Worst case frame-times:

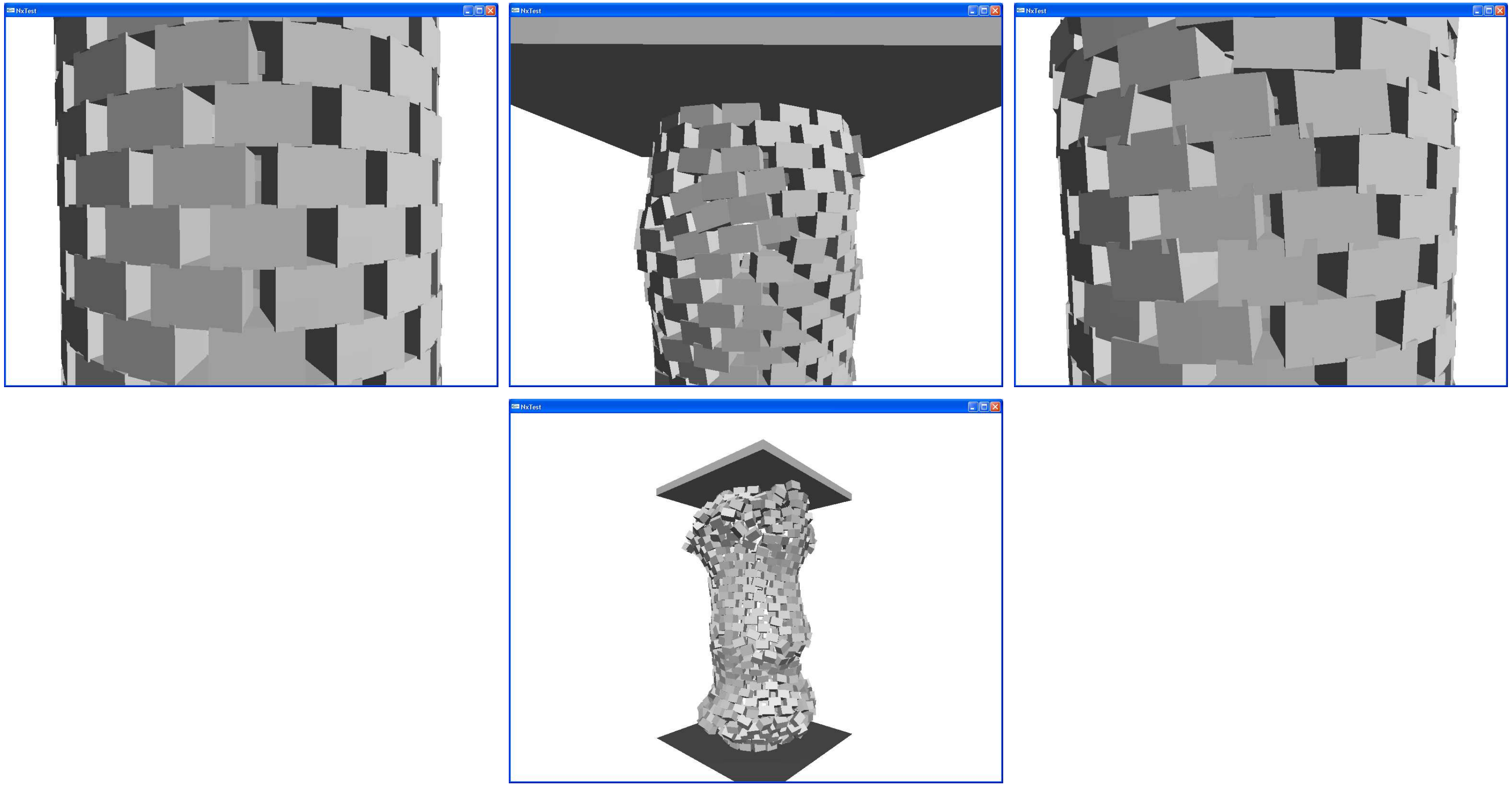
Engraving	0.18 secs
Cow pile	1.2 secs
Roof	0.25 secs
Silo	1.1 secs

Time-step = 0.01 secs.

- Pentium M, 1.7 GHz, 1GB RAM.
- Time-Stepping is linear in the number of contact points.
- Different slopes are a result of the stacking topology.

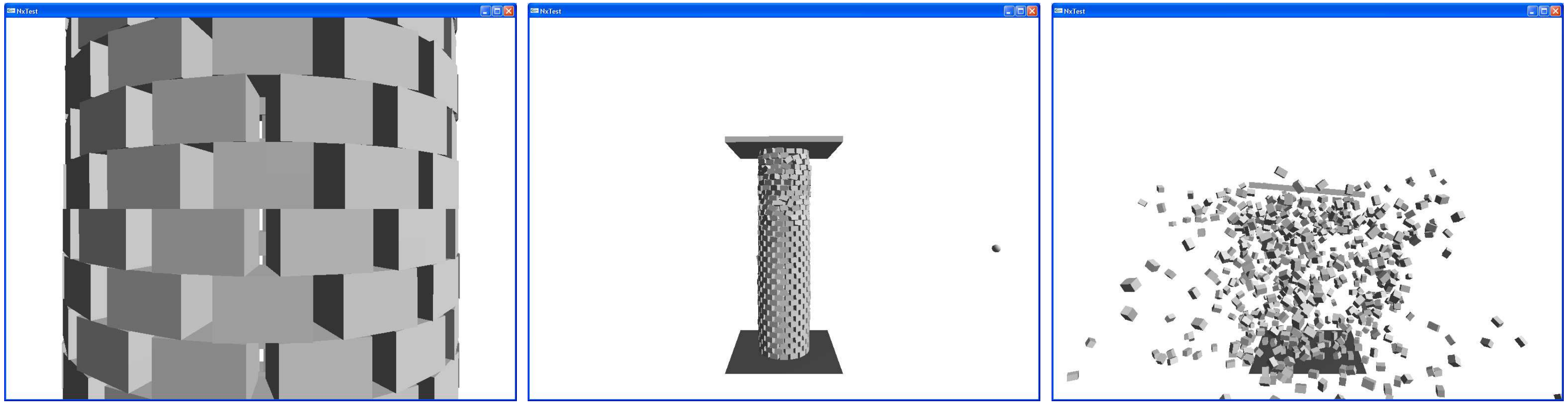
Comparisons

Novodex Default



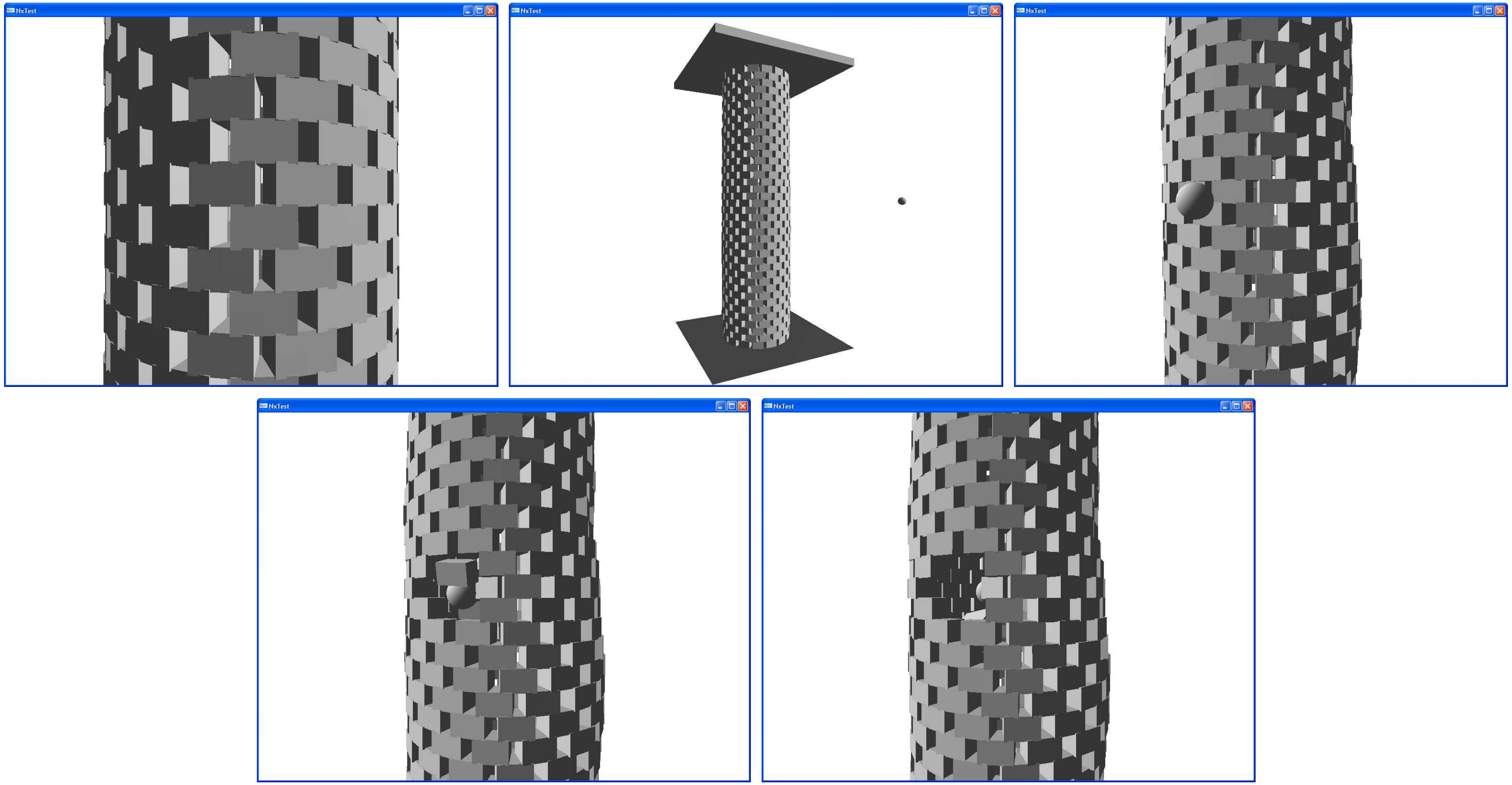
Comparisons

Novodex 300 Iterations low separation distance



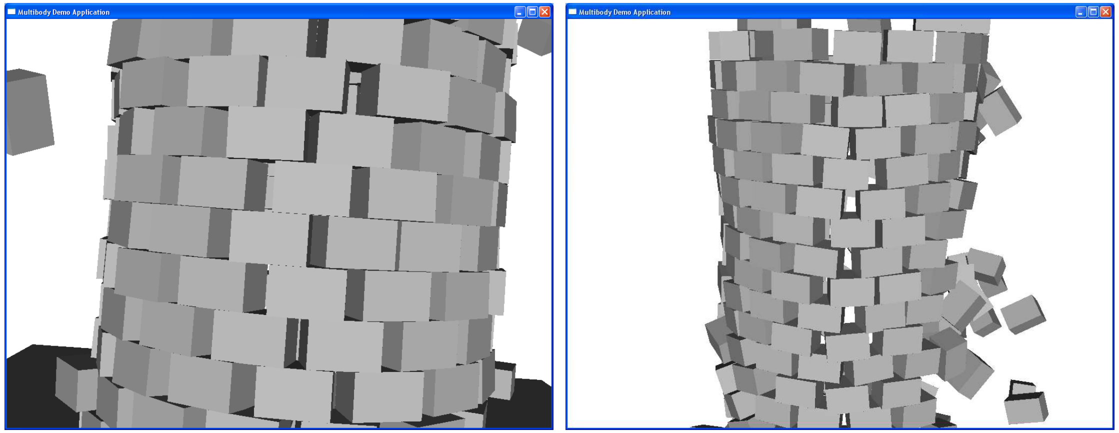
Comparisons

Novodex 30 Iterations, Tweaked Mass, gravity,...



Comparisons

Guendelman et. al. (Siggraph 2003)



Comparisons

Maya

Fatal error...

Comparisons