GPU TECHNOLOGY CONFERENCE

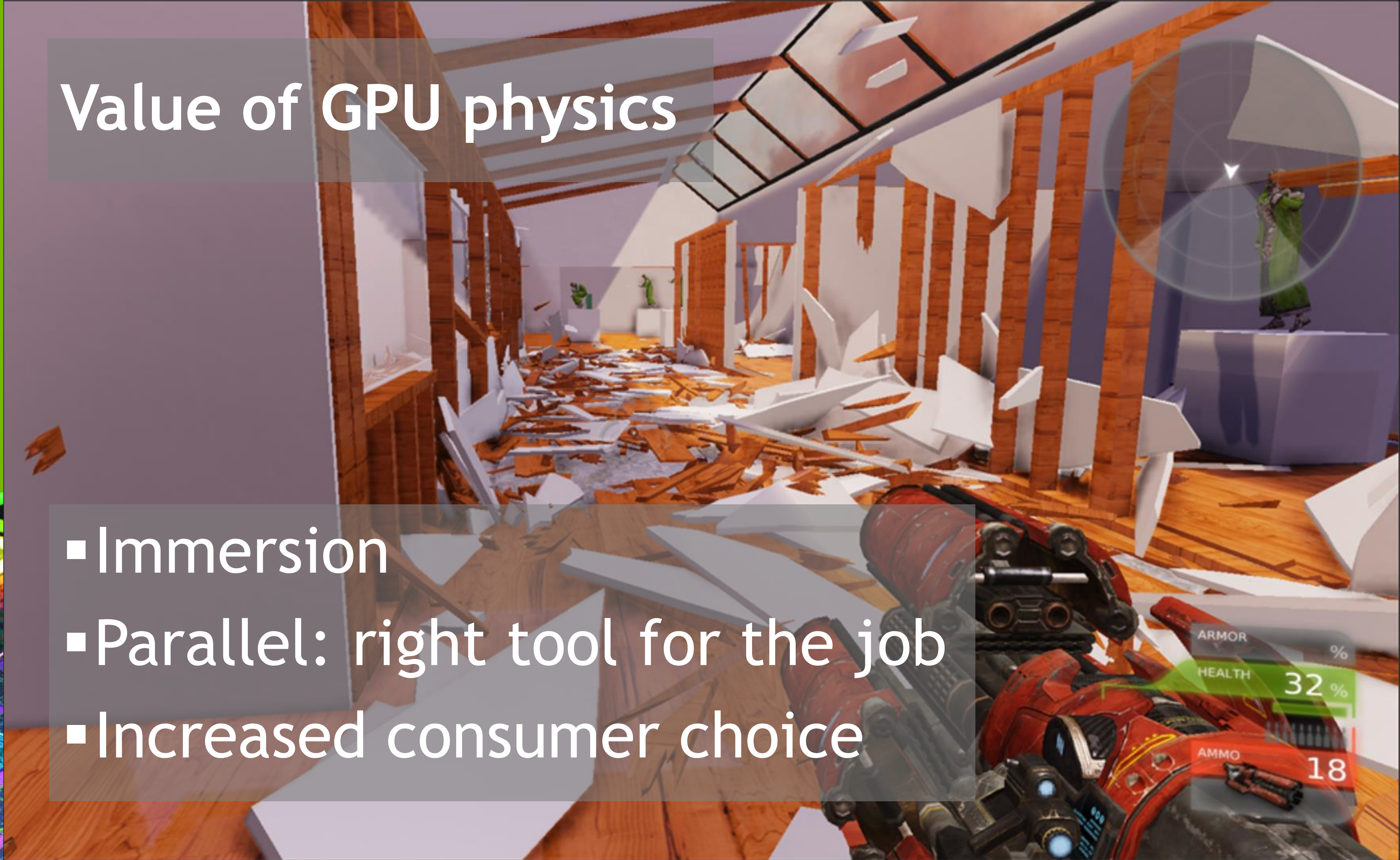
GPU Rigid Body Dynamics

Demo credits

- Matthias Müller-Fischer
- Nuttapong Chentanez
- Tae-Yong Kim
- Aron Zoellner
- Kevin Newkirk

Value of GPU physics

- Immersion
- Parallel: right tool for the job
- Increased consumer choice



GPU TECHNOLOGY CONFERENCE

SIGGRAPH paper

Mass Splitting for Jitter-Free Parallel Rigid Body Simulation

Richard Tonge, Feodor Benevolenski, Andrey Voroshilov

Mass Splitting for Jitter-Free Parallel Rigid Body Simulation

Richard Tonge, NVIDIA *
Feodor Benevolenski, NVIDIA
Andrey Voroshilov, NVIDIA

Figure 1: *Left: 5000 boxes with 40000 contacts coming to rest on a non-convex triangle mesh without jitter, simulated on an NVIDIA GTX580 at over 60 FPS. Middle: Mass splitting used in a real-time fracture simulation. The debris have irregular shapes and large mass ratios. Right: The method allows us to simulate large scale building destruction in real-time in a video game.*

Abstract

We present a parallel iterative rigid body solver that avoids common artifacts at low iteration counts. In large or real-time simulations, iteration is often terminated before convergence to maximize scene size. If the distribution of the resulting residual energy varies too much from frame to frame, then bodies close to rest can visibly jitter. Projected Gauss-Seidel (PGS) distributes the residual according to the order in which contacts are processed, and preserving the order in parallel implementations is very challenging. In contrast, Jacobi-based methods provide order independence, but have slower convergence. We accelerate projected Jacobi by dividing each body mass term in the effective mass by the number of contacts acting on the body, but use the full mass to apply impulses. We further accelerate the method by solving contacts in blocks, providing wallclock performance competitive with PGS while avoiding visible artifacts. We prove convergence to the solution of the underlying linear complementarity problem and present results for our GPU implementation, which can simulate a pile of 5000 objects with no visible jittering at over 60 FPS.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physically based modeling; I.6.8 [Simulation and Modeling]: Types of Simulation—Animation

Keywords: rigid bodies, non-smooth dynamics, contact, friction

Links: DL PDF

*e-mail: rtonge@nvidia.com

1 Introduction

Rigid body dynamics is widely used in applications ranging from movies to engineering to video games. Piles of objects are particularly common, because ultimately, gravity pulls all rigid bodies to the ground. Some of the most visually interesting simulations involve destruction, such as projectile impacts and explosions, and these can generate large piles of debris. In mechanical engineering some of the most computationally challenging problems involve simulating interaction with large resting systems of soil particles or rocks. Piles require stable simulation of static friction, dynamic friction and resting contact, which presents many challenges.

In large or real-time simulations, the computation budget can be small compared to the number of rigid body contacts. In these cases we must use iterative methods, terminating the iteration before convergence. By stopping early we introduce residual energy into the system, which can cause objects near rest to jitter. See the accompanying video for examples of these artifacts.

A commonly used iterative algorithm is projected Gauss-Seidel (PGS), which solves contacts in sequence. When we terminate the iteration, the last contact solved has no error and the other contacts have non-zero error, resulting in an uneven distribution of the residual energy. On single threaded implementations we can ensure that the distribution of this error is consistent from frame to frame by, for example, ensuring that collisions are detected in the same order each frame and that addition and deletion of bodies do not disrupt the order. Unfortunately, things are not as straightforward for parallel implementations.

Due to the widespread availability of multi-core CPUs and GPUs, parallel computing is increasingly being used to simulate rigid bodies. PGS has limited parallelism, as updates to bodies having multiple contacts must be serialized to ensure they are not lost and the connectivity between bodies can be complex, especially in piles. This serialization changes the order in which constraints are processed, often changing it dramatically from frame to frame. Operating systems and GPU schedulers can also introduce non-determinism into the order of operations. For these reasons it is very challenging to avoid jittering with parallel PGS, and due to the serialization its performance does not scale well as more threads are added.

In real-time applications such as games, the designer does not know in advance what the player is going to do and which objects are

Contents

- Past
  - The challenges of rigid body simulation
  - Model
  - Existing solvers
  - Value of fixing jitter
- Present
  - Jitter-free solver
  - Video
- Future

NVIDIA PhysX

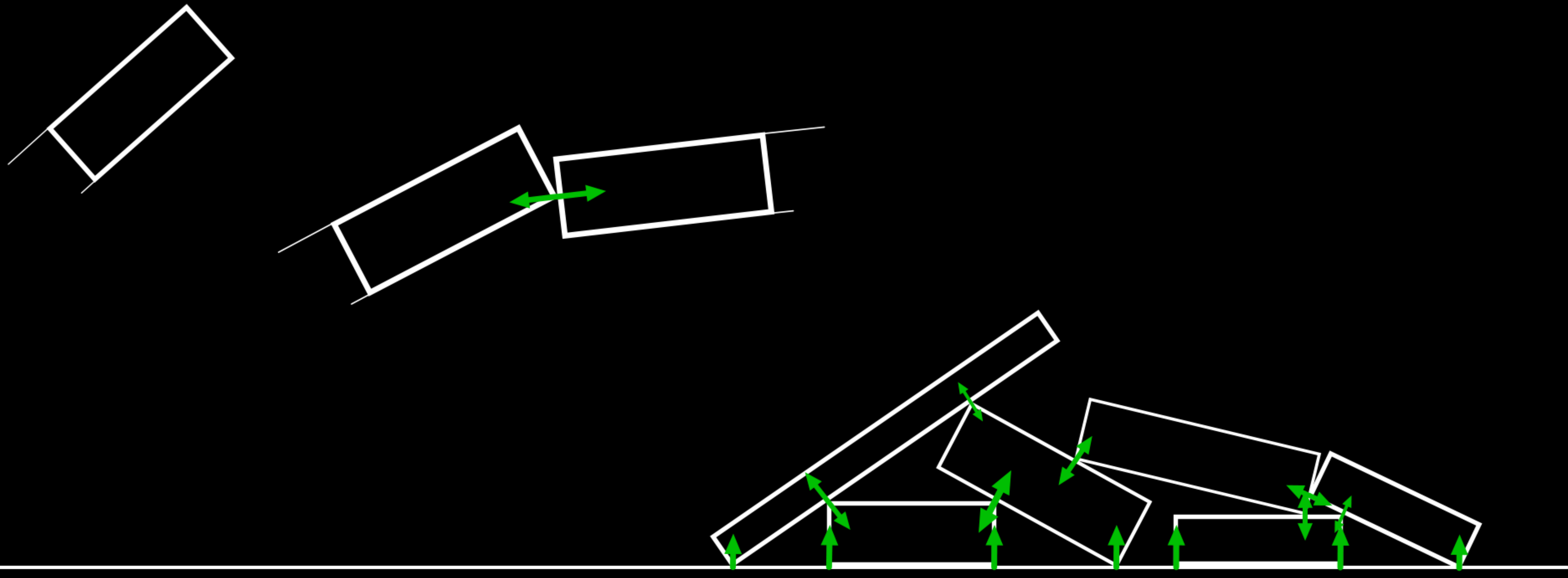
- Turbulence
- Clothing
- Particles and fluids
- Destruction

Past: The challenges of rigid body simulation

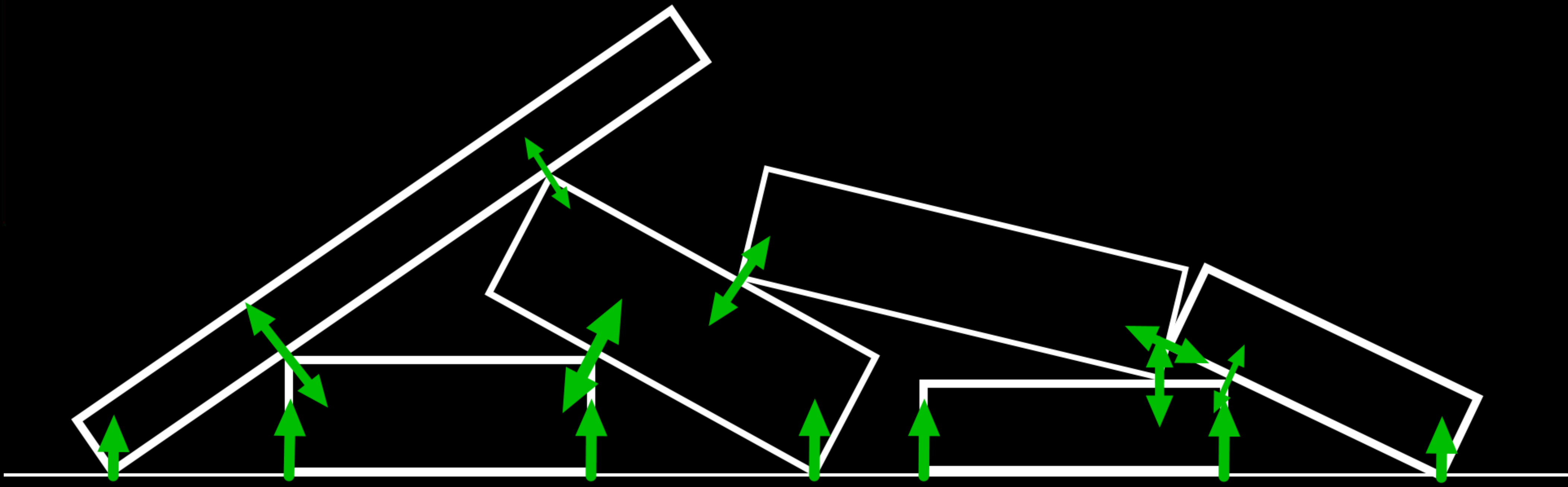
Why is rigid body simulation on GPU hard?

- Generating enough parallel work
- Irregular
- Real-time
- Jitter

Jitter video



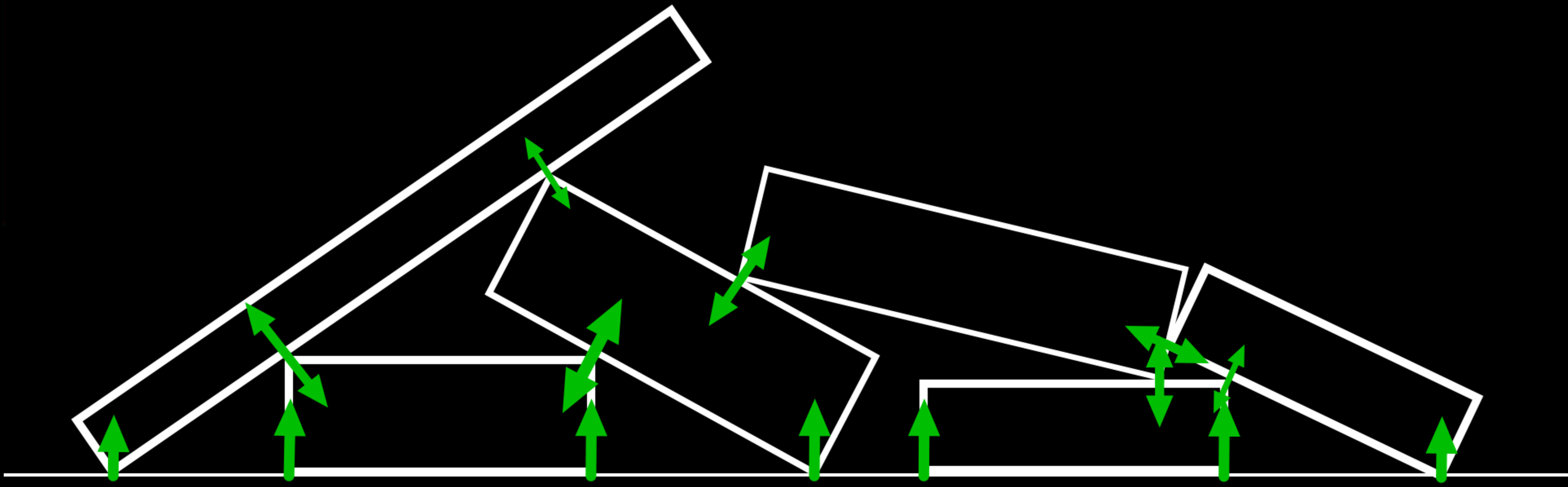
Stable piles = balanced forces



Past: Model & Discretization

Model

Non-penetration condition: $\Phi(\mathbf{x}) \geq 0$

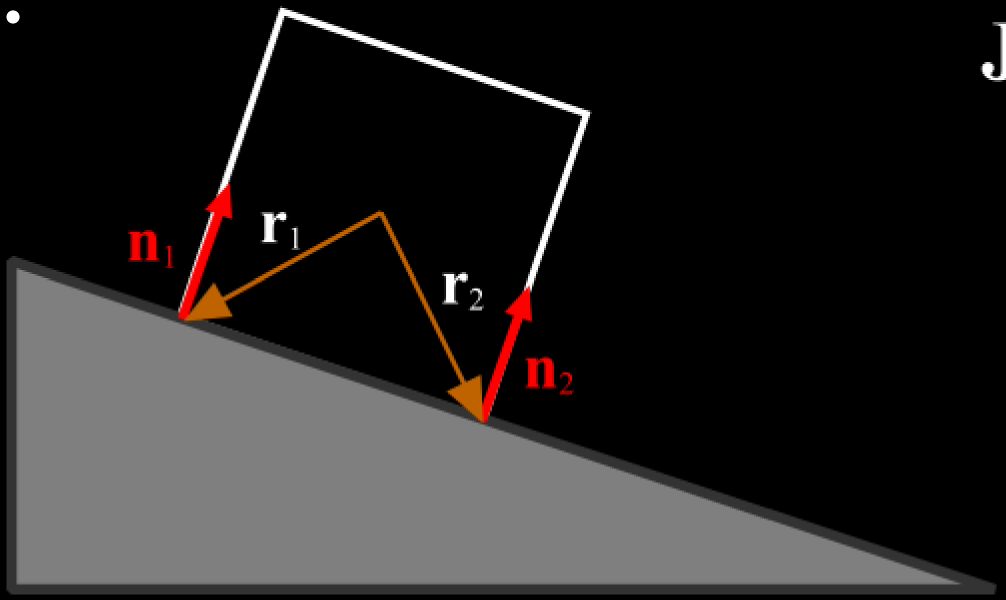


Jacobian

$$\Phi(\mathbf{x}) \geq 0$$

$\mathbf{J} = \frac{\partial \Phi}{\partial \mathbf{x}}$, maps body velocity to contact velocity

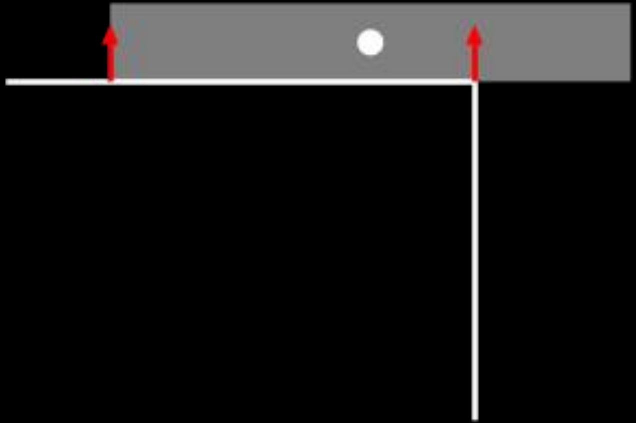
Example:

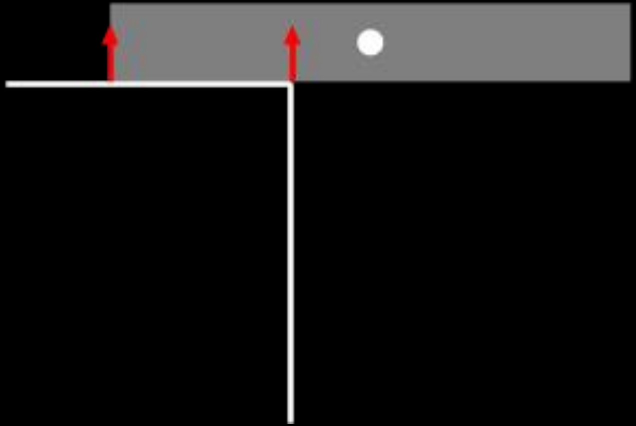


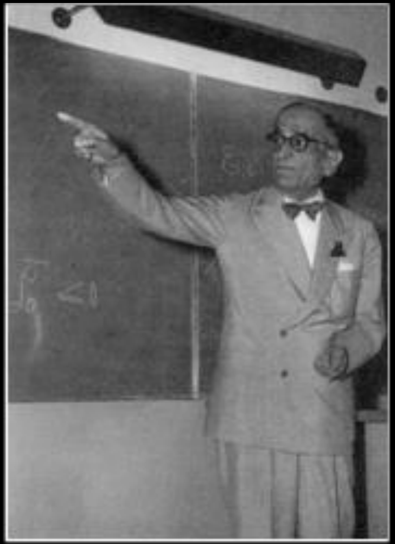
$$\mathbf{J} = \left[\begin{array}{c|c} \mathbf{n}_1 & \mathbf{r}_1 \times \mathbf{n}_1 \\ \hline \mathbf{n}_2 & \mathbf{r}_2 \times \mathbf{n}_2 \end{array}\right]$$

$$= \left[\begin{array}{ccc|ccc} \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \hline \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{array}\right]$$

When should contacts break?







Antonio Signorini

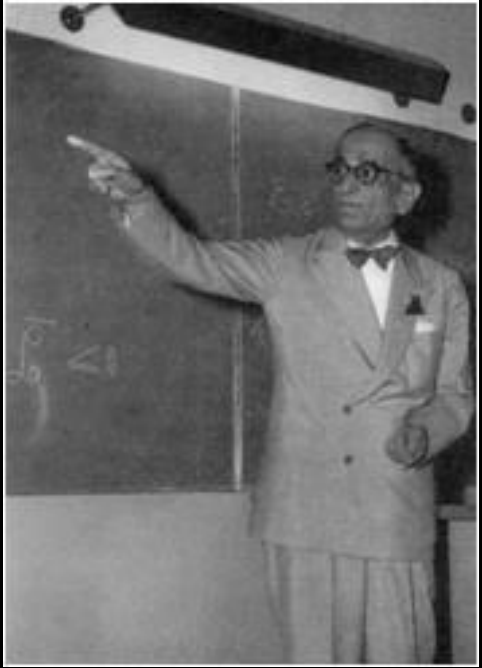
The Signorini Conditions:

$0 \leq \mathbf{v}_{\mathrm{rel}}$ All relative velocities must be zero or separating

$0 \leq \lambda$ All contact forces must be non attractive

$$(\mathbf{v}_{\mathrm{rel}})_{\mathrm{i}} = 0 \text{ or } \lambda_{\mathrm{i}} = 0$$

No force at separating contacts



Antonio Signorini

Model

Non-penetration condition: $\Phi(\mathbf{x}) \geq 0$

$$\mathbf{J} = \frac{\partial \Phi}{\partial \mathbf{x}}$$

$$\mathbf{M}\ddot{\mathbf{x}} = \mathbf{J}^{\mathrm{T}}\boldsymbol{\lambda} + \mathbf{f}_e$$

$$\dot{\mathbf{x}} = \mathbf{v}$$

$$\boldsymbol{\lambda} \geq \mathbf{0} \perp \mathbf{J}\mathbf{v} \geq \mathbf{0}$$

Notation

h	time step size
$\mathbf{x}$	positions and orientations
$\mathbf{v}$	linear and angular velocities
$\mathbf{f}_e$	external forces
$\mathbf{M}$	mass matrix
$\Phi(\mathbf{x})$	contact separation
$\mathbf{J}$	$\frac{\partial \Phi}{\partial \mathbf{x}}$, the Jacobian of Φ
$\mathbf{z}$	contact impulses

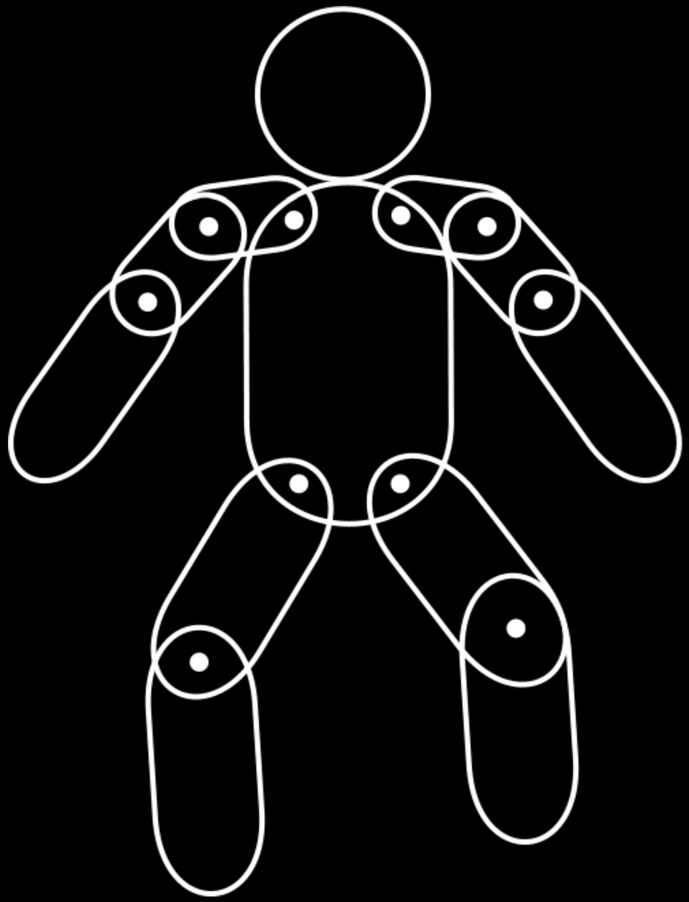
Discretization

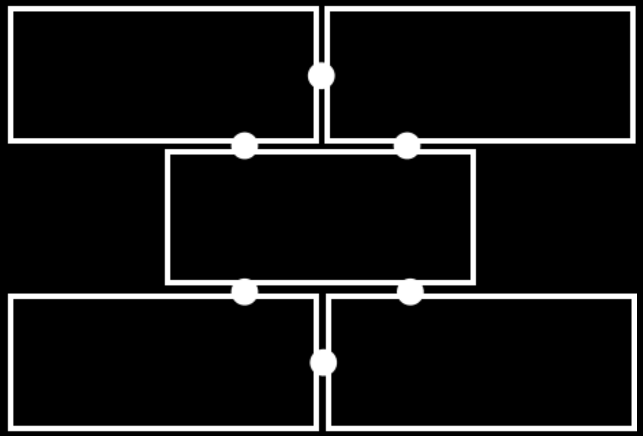
$$\mathbf{M}(\mathbf{v}_{\text{new}} - \mathbf{v}_{\text{old}}) = \mathbf{J}^{\mathbf{T}}\mathbf{z} + h\mathbf{f}_e$$
$$\mathbf{x}_{\text{new}} - \mathbf{x}_{\text{old}} = h\mathbf{v}_{\text{new}}$$
$$\mathbf{z} \geq \mathbf{0} \perp \mathbf{J}\mathbf{v}_{\text{new}} \geq \mathbf{0}$$

Solution

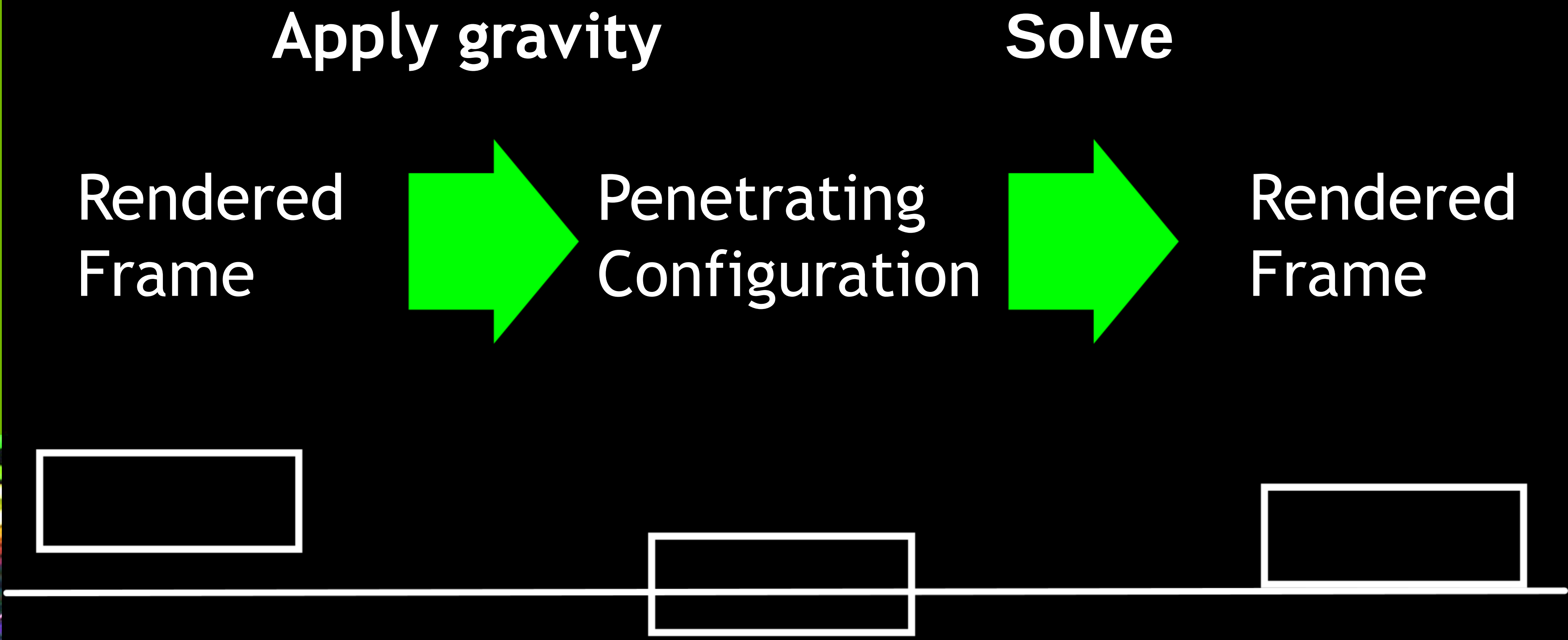
$$\mathbf{q} := \mathbf{J}(\mathbf{v}_{\text{old}} + h\mathbf{M}^{-1}\mathbf{f}_e)$$
$$\mathbf{N} := \mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T$$
$$\mathbf{z} := \text{LCP}(\mathbf{N}, \mathbf{q})$$
$$\mathbf{v}_{\text{new}} := \mathbf{v}_{\text{old}} + \text{h}\mathbf{M}^{-1}\mathbf{J}^T\mathbf{z}$$
$$\mathbf{x}_{\text{new}} := \mathbf{x}_{\text{old}} + \text{h}\mathbf{v}_{\text{new}}$$

Joints



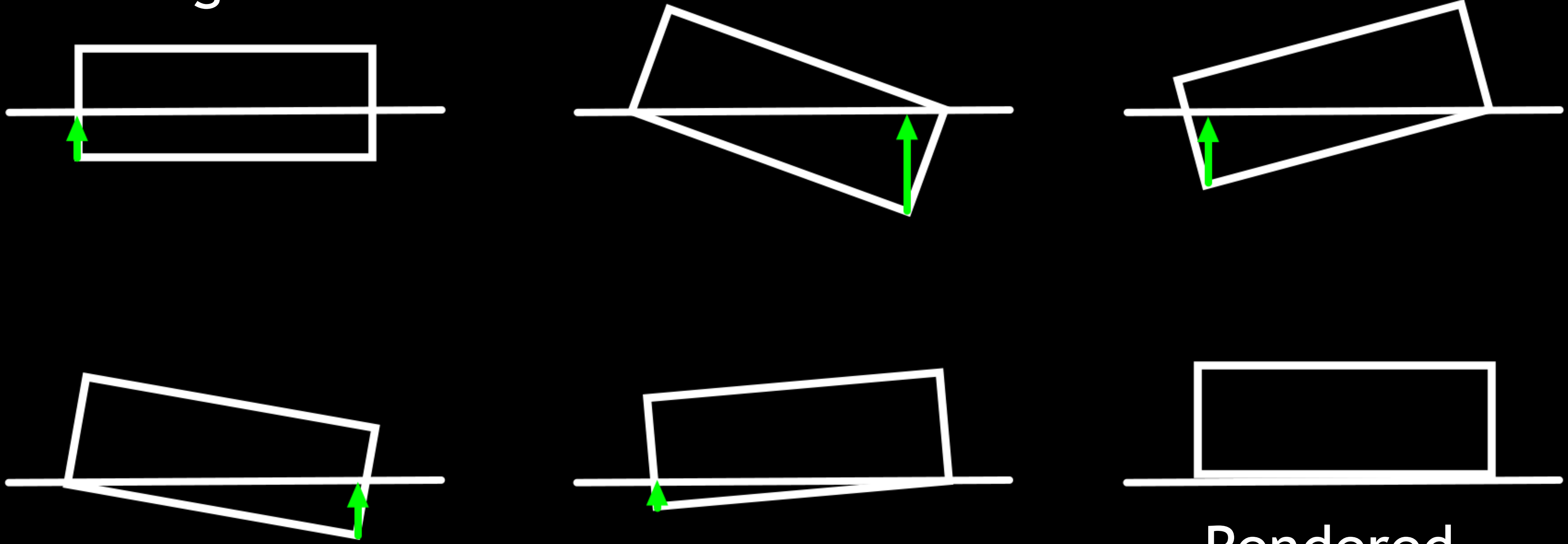


Past: Existing methods


Apply gravity
Solve
Rendered Frame
Penetrating Configuration
Rendered Frame

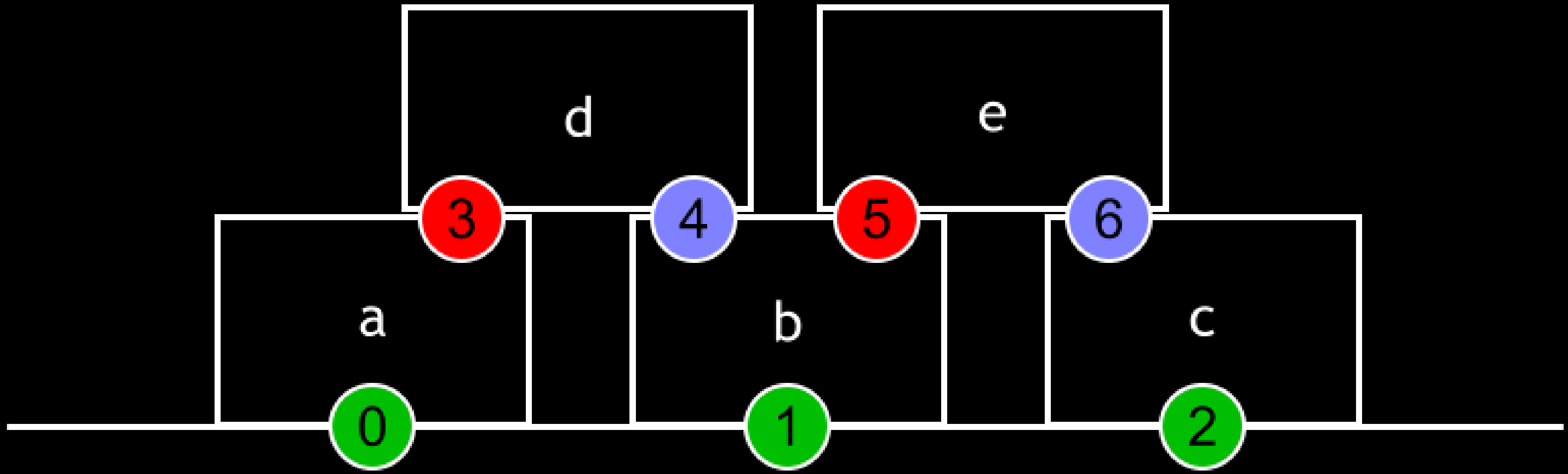
Existing solver method 1:

Penetrating configuration

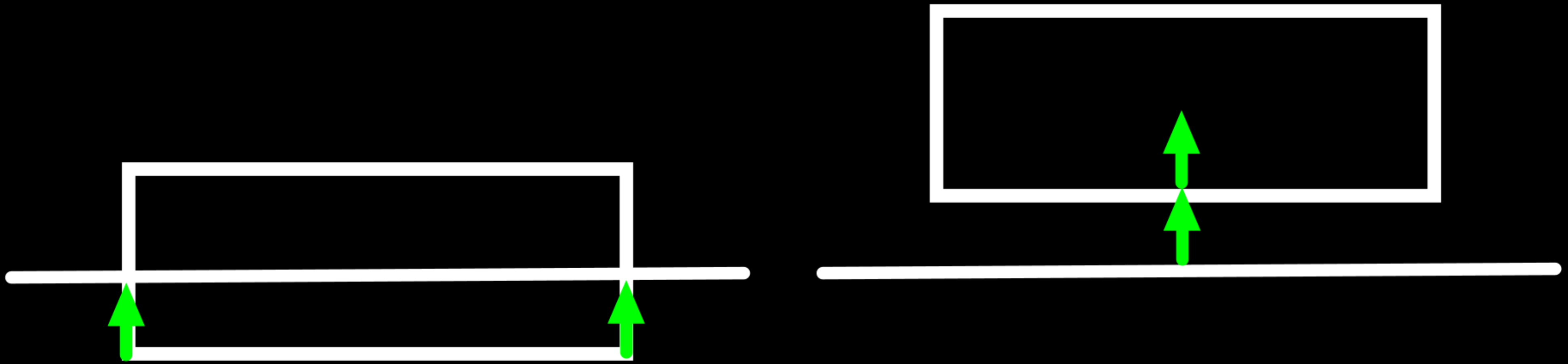


Rendered Frame

Parallel PGS - coloring



Existing solver method 2:



GPU TECHNOLOGY CONFERENCE

- Method 1 (Parallel Projected Gauss Seidel, PGS)
  - Provably convergent ✔
  - Limited parallelism ✘
  - Jitters ✘
  - Widely used

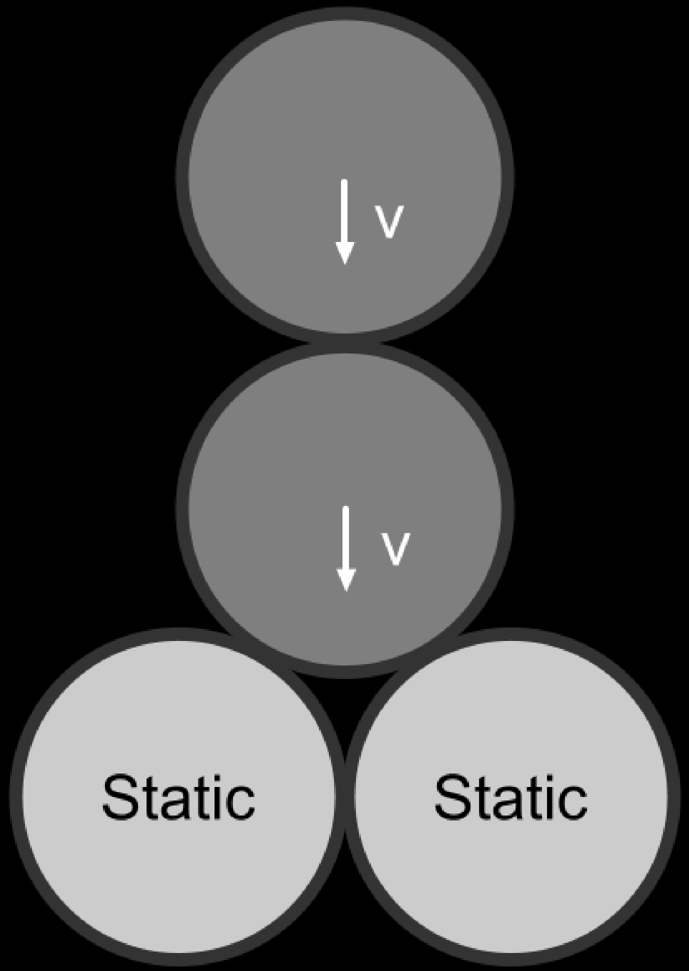
- Method 2 (Parallel Projected Jacobi)
  - Maximally parallel ✔
  - Jitter free ✔
  - Non convergent in many cases ✘
  - Converges slowly ✘
  - Unusable in games

Past: Value of fixing jitter

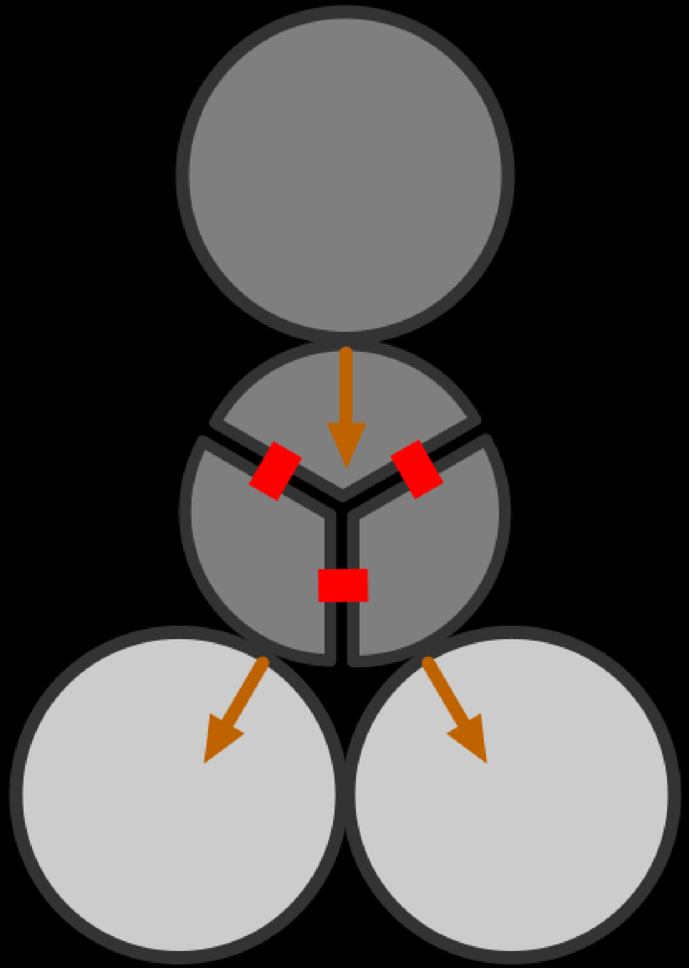
GPU TECHNOLOGY CONFERENCE

Present: Jitter-free GPU solver

Example



First idea: Spatial splitting



Extend PGS to solve joints exactly

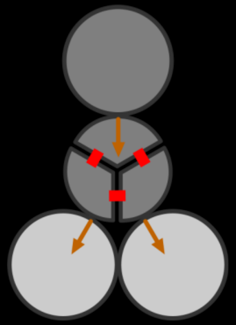
Split and join bodies to get parallelism

split solution
= unsplit solution

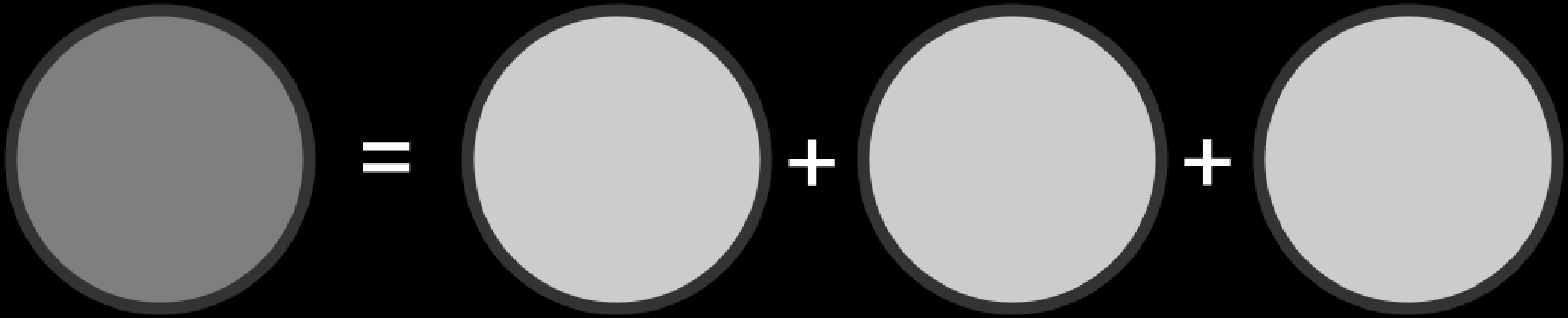
Split spatially

Splitting too expensive
Joints too expensive

Dead end

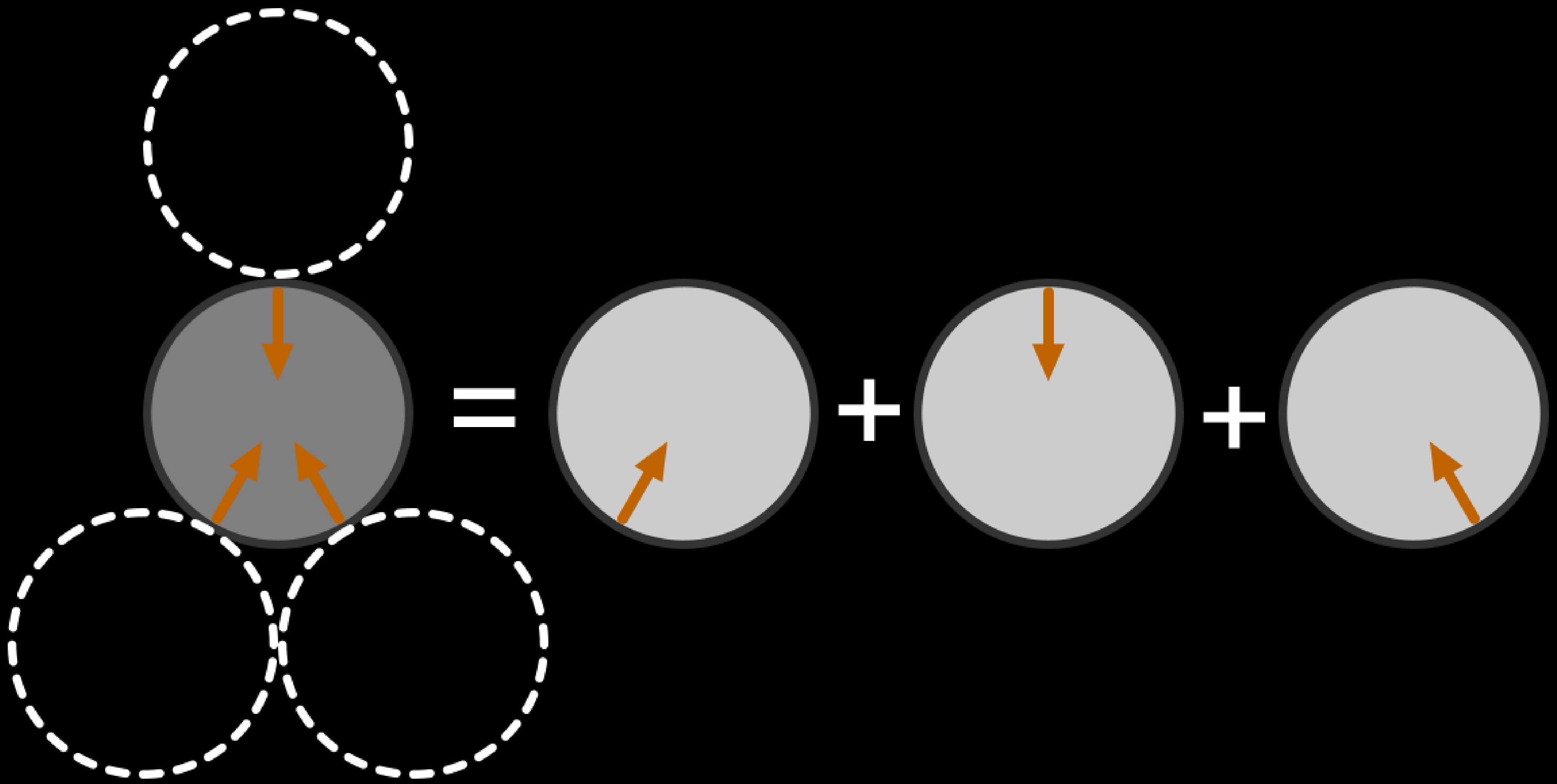


New idea: Mass splitting

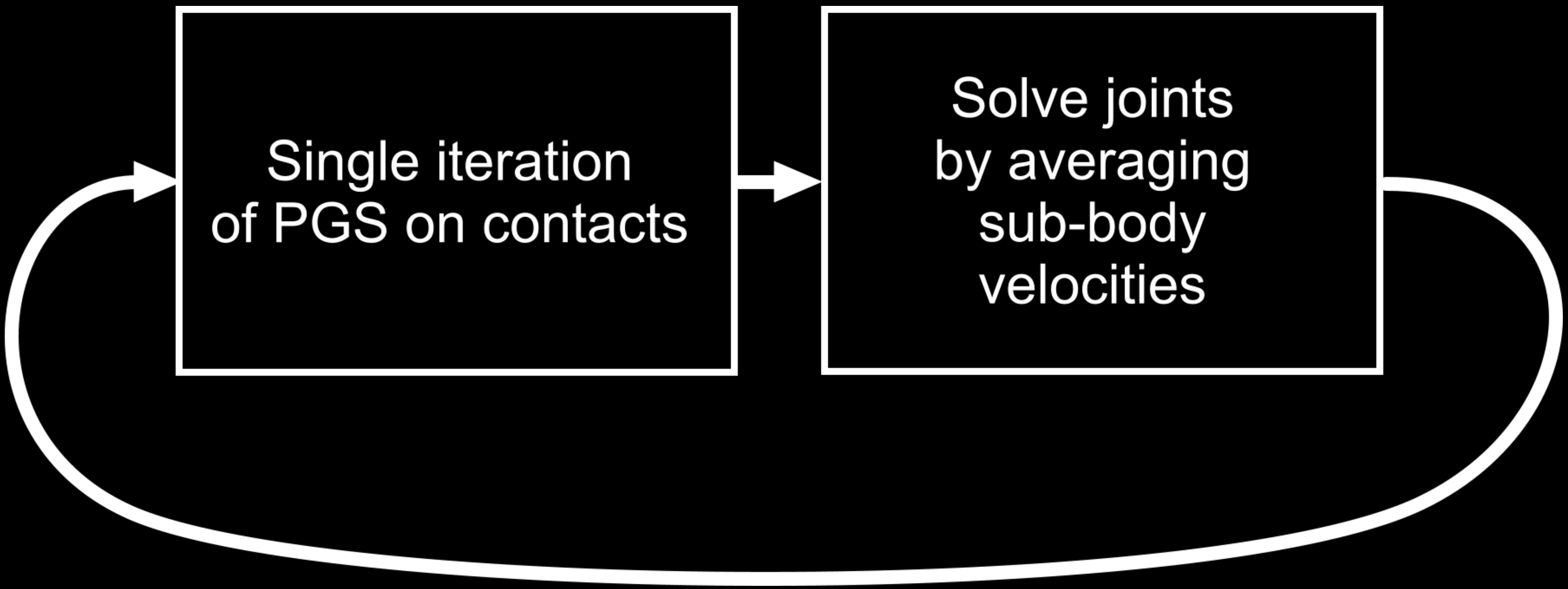


+ two fixed joints at C.O.M.

New idea: Mass splitting



PGS with exact joints



Extend PGS to solve joints exactly

Split and join bodies to get parallelism

split solution = unsplit solution

Split spatially

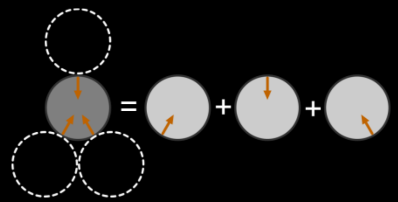
Splitting too expensive
Joints too expensive

Dead end ✗

Split mass non-spatially

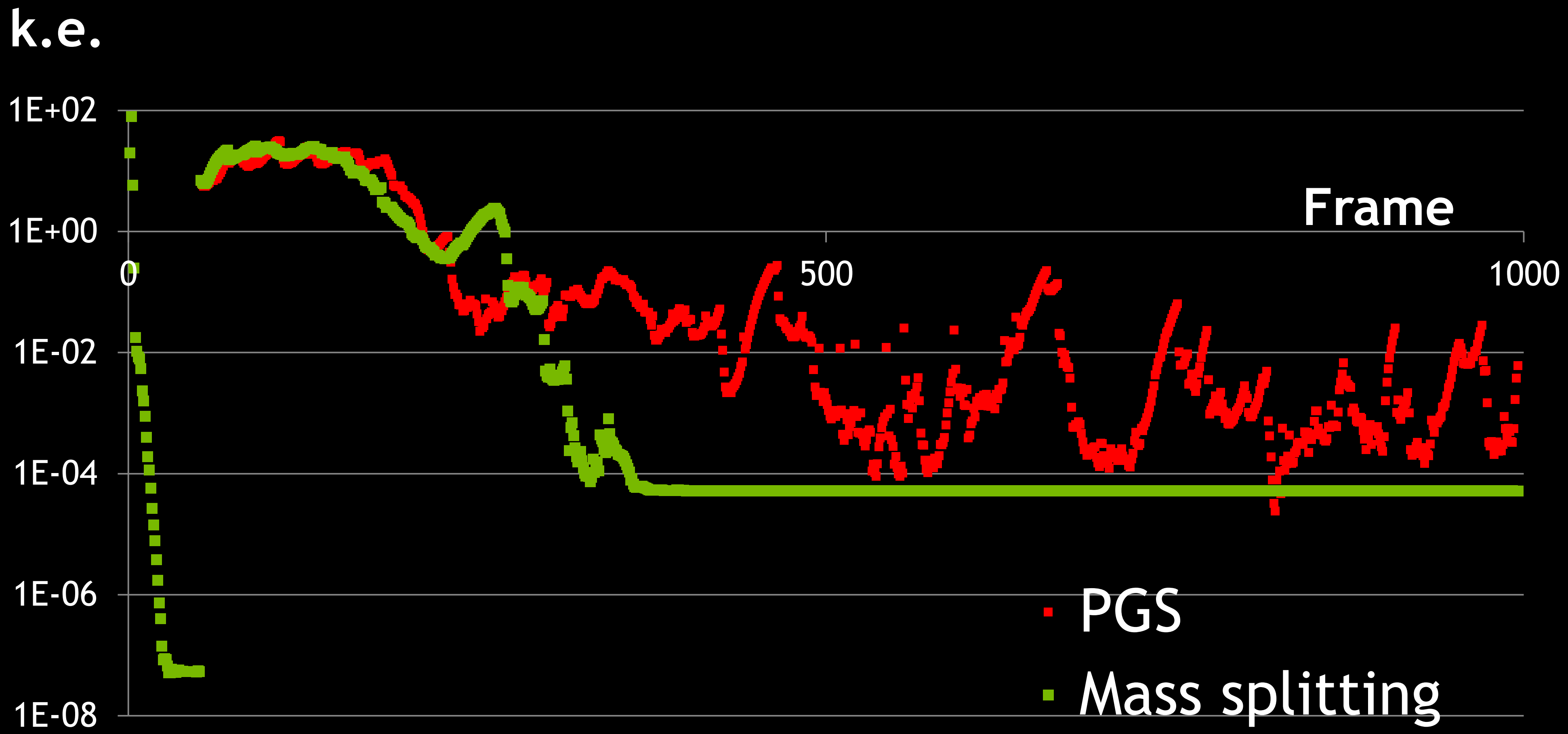
Closed form for joints

Inexpensive
Convergent
+ no jitter



win! ✓

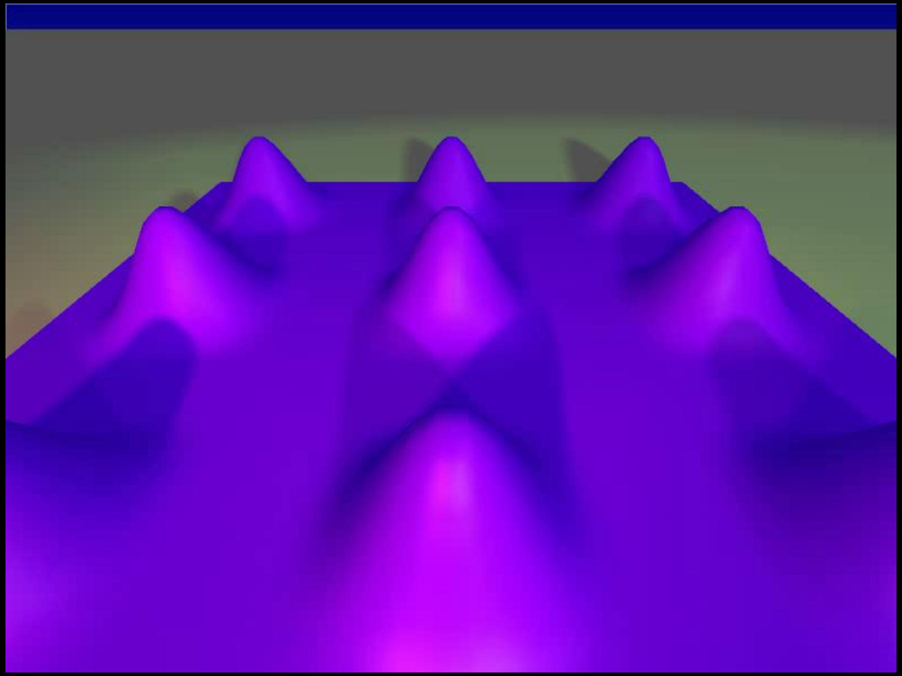
Results


k.e.
1E+02
1E+00
1E-02
1E-04
1E-06
1E-08
0
500
1000
Frame
PGS
Mass splitting

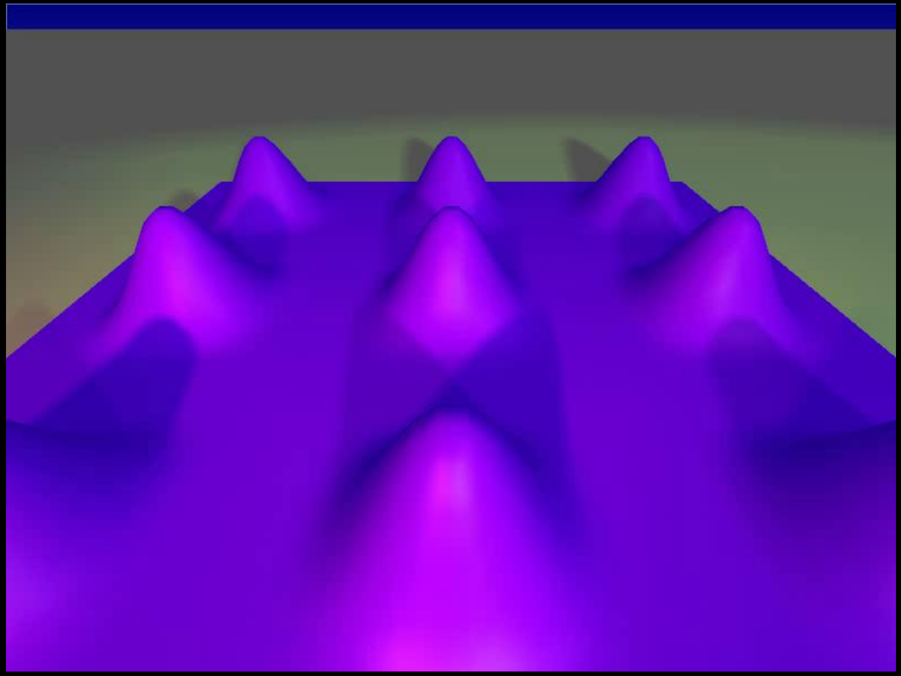
Results

Future

Rigidity

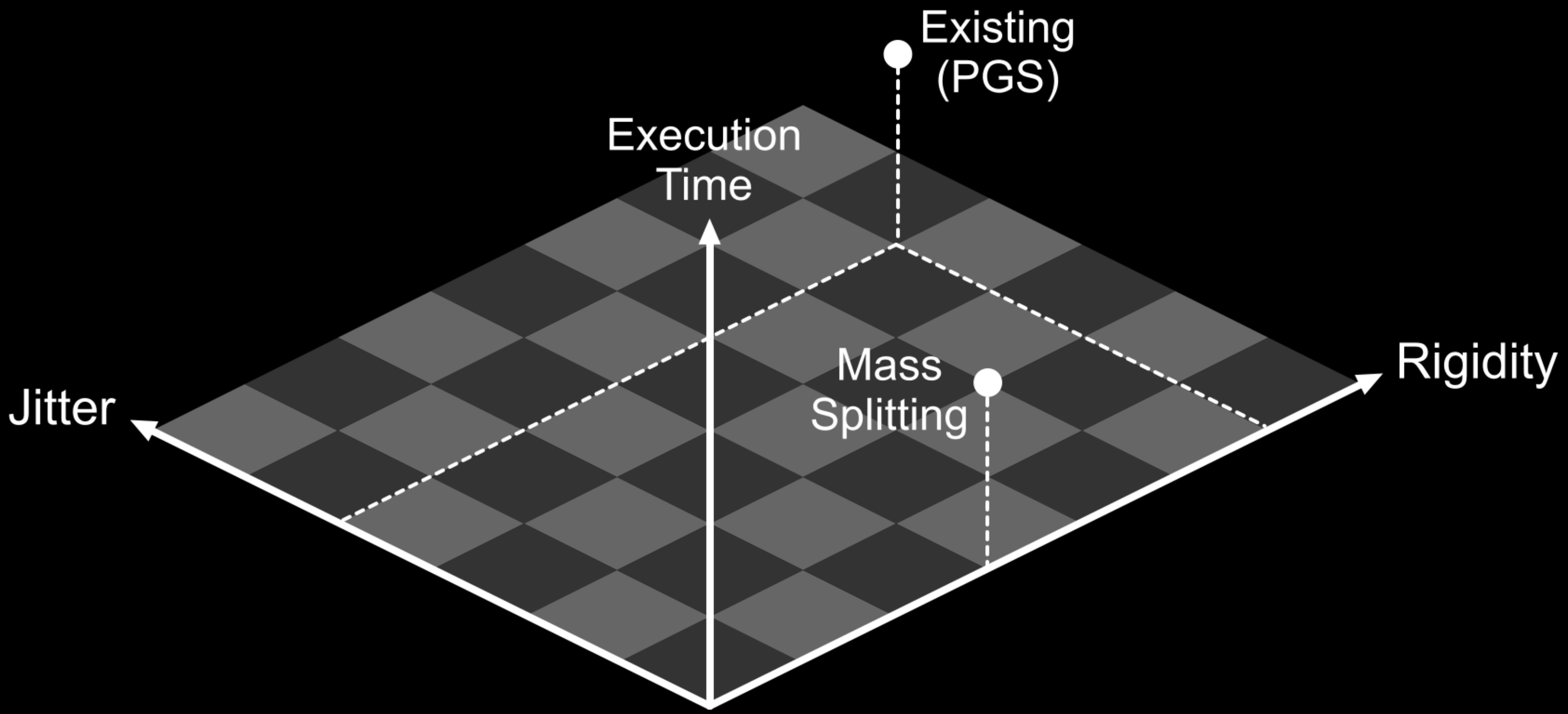


Not real-time
(500 iterations)



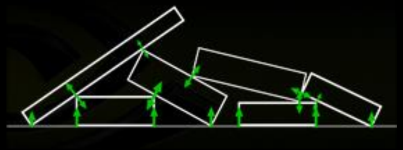
Real-time
(15 iterations)

Design space

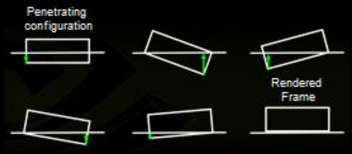


Summary

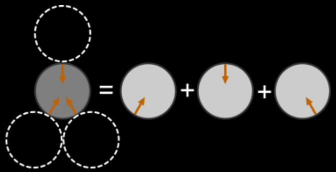
- What a solver does



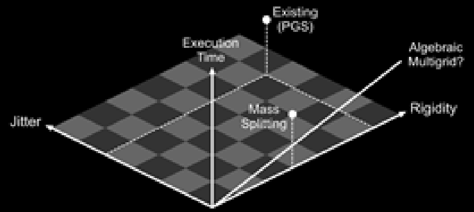
- Model $\lambda \geq 0 \perp Jv \geq 0$
- Previous standard: PGS



- Jitter free => solver can move to GPU



- Idea: Split bodies non-spatially
- Provably convergent – necessary for games
- Future



Acknowledgments

GPU rigid body technology

- Richard Tonge
- Feodor Benevolenski
- Andrey Voroshilov

Fracture technology and demo

- Matthias Müller-Fischer
- Nuttapong Chentanez
- Tae-Yong Kim
- Aron Zoellner

Thanks also to the PhysX SDK team