Physics in Games

Matthias Müller

www.MatthiasMueller.info
Outline

• Comparison
  – Physical simulations in engineering
  – Offline physics in graphics (mostly movies)
  – Interactive physics
  – Real time physics in games

• Position Based Dynamics
  – Algorithm
  – Examples: cloth, rigid bodies, fluids, unified solver

• Q & A
Simulations in Engineering

- Complement real experiments
- Extreme conditions, spatial scale, time scale
- **Accuracy** most important factor
- Low accuracy: Useless result!
- One central **gigantic computer**
Evolution of Compute Power

Zuse’s Z1 (1938)
0.2 ops

Titan (currently number 2)
using 18,000 nvidia GPUs
~27,000,000,000,000,000,000 flops!
Simulation of Hurricane Sandy

- National Center for Supercomputing Applications
- 9120 x 9216 x 48 cells (500 m)
- 13,680 nodes and 437,760 cores on Titan
- Sustained rate of 285 teraflops
Physics in Graphics
Re-inventing the Wheel?

• Since late 80’s [Terzopoulos et al. 87, 88]

• Rediscoveries
  – Semi-Lagrangian advection, co-rotational FEM,
    X introduced Y to graphics (SPH, MPM, FLIP, …)

• Goals of physics in graphics
  – Imitation of physical phenomena / effects
  – Plausible behavior (cheating possible)
  – Trade accuracy for speed, stability, simplicity
  – Control (by director / game developer)

• New goals require new methods!
Offline Methods

• Main application: Movies
• $\gg$ 1 sec of computation for 1 sec of simulation allows:
  – High resolution (fluid grid, FEM mesh, time steps)
  – Re-runs and adaptive time steps
  – Time consuming shading

[Emmerich, Movie 2012]
Interactive Physics

- Between offline and game physics
- Virtual surgery, virtual reality, demos
- All available compute power
- > 15 fps
- No adaptive time steps
- Robust
  - No re-runs
  - Unforeseeable situations
Water Demo (GTC 2012)

• First time real-time Eulerian water sim + ray tracing

2 x GTX 680
Multi-grid
[Chentanez et al., 2011]
OptiX
Dragon

- Eulerian fluid simulation + combustion model + volumetric rendering
Physics in Games
Game Requirements

• Cheap to compute
  – 30-60 fps of which physics only gets a small fraction

• Low memory consumption
  – Consoles, fit into graphics (local) memory

• Stable in extreme settings
  – 180 degree turns in one time step

• High level of control

• Challenge
  • Meet all these constraints
  • Get to offline results as close as possible
Speedup Tricks

• Reduce simulation resolution
  – Simple: Use same algorithms
  – Interesting details disappear

• Reduce dimension (e.g. 3d → 2d)

• Use different resolution for physics and appearance

• Simulate only in active regions (sleeping)

• Camera dependent level of detail (LOD)

• Invent new simulation methods!

• Use nvidia GPUs and CUDA! 🎉
Game Physics Methods
Animation

• Pros:
  – Can be and still is used for almost everything (3d movie playback)
  – Full control
  – What artists are used to do

• Cons
  – Time consuming manual work
  – Hard to handle complex phenomena
  – Repeating behavior
Particle Physics

- **Simplest and very popular** form of physics effect
  - droplets, smoke, fire, debris [Reeves, 1983]

- **Effects physics vs. game play physics**
  - does not influence game play, no path blocking

- **Most expensive part:**
  - collision detection with large environments
  - particle-particle interaction (often not needed)
  - Advection by incompressible velocity field (fluid solver)
Rigid Bodies

• Game physics engines = rigid body engines
• Challenges
  • Stability (stacking)
  • Speed (solver and collision detection)
  • Continuous collision detection (fast moving objects)
• Rarely in-house
• Middleware popular (*PhysX*, *havok*, *bullet*)
Destruction

• Traditional: static fracture
• Artists **pre-fracture** models
• Models are replaced by parts when collision forces exceed a threshold

• **Pro:**
  • High level of control

• **Cons:**
  • Tedious manual work
  • **Independent of impact location**
PhysX Destruction Tool
Pattern Based Fracture

- Pre-designed fracture pattern

[Müller et al., 2013]
Pattern Based Fracture

- Pre-designed fracture pattern
- Align pattern with impact location at runtime
- Use pattern as stencil

[Müller et al., 2013]
Arena Destruction  
(SG 2013 real time live)

- 500k faces at start
- GPU1: rigid body simulation
- GPU2: smoke, rendering
- CPU: dynamic fracturing
Deformable Objects

• 1d: Ropes, hair

• 2d: Cloth, clothing

• 3d: Fat guys, tires
Existing Methods

• Force based
• Mass-Spring Systems / FEM
• Explicit integration unstable
• Implicit integration
  – Expensive
  – Large time steps for real time simulation needed
  – Numerical damping
Position Based Dynamics

[Müller et al., 2006]
Position Based Dynamics

[Müller et al., 2006]

[google scholar]
Force Based Update

- Reaction lag
- Small spring stiffness → squashy system
- Large spring stiffness → stiff system, overshooting
Position Based Update

- Controlled position change
- Only as much as needed → no overshooting
- Velocity update needed to get 2\textsuperscript{nd} order system!
Position Based Integration

\[ \text{init } x_0, v_0 \]

\[ \text{loop} \]
\[ \quad p \leftarrow x_n + \Delta t \cdot v_n \quad \text{prediction} \]
\[ \quad x_{n+1} \leftarrow \text{modify } p \quad \text{position correction} \]
\[ \quad u \leftarrow (x_{n+1} - x_n) / \Delta t \quad \text{velocity update} \]
\[ \quad v_{n+1} \leftarrow \text{modify } u \quad \text{velocity correction} \]

end loop
Position Correction

- Example: Particle on circle
Velocity Correction

- External forces: $v_{n+1} = u + \Delta t \frac{g}{m}$
- Internal damping
- Friction
- Restitution

![Diagram showing velocity correction process]
Distance Constraint

\[
\Delta x_1 = - \frac{w_1}{w_1 + w_2} (|x_1 - x_2| - l_0) \frac{x_1 - x_2}{|x_1 - x_2|}
\]

\[
\Delta x_2 = + \frac{w_2}{w_1 + w_2} (|x_1 - x_2| - l_0) \frac{x_1 - x_2}{|x_1 - x_2|}
\]

- Conservation of momentum
- Stiffness: scale corrections by \( k \in [0,1] \)
  - Easy to tune
  - Effect dependent on time step size and iteration count
  - Often constant in games
General Internal Constraint

• Define constraint via scalar function:

\[ C_{dist}(x_1, x_2) = |x_1 - x_2| - l_0 \]

\[ C_{volume}(x_1, x_2, x_3, x_4) = [(x_2 - x_1) \times (x_3 - x_1)] \cdot (x_4 - x_1) - 6v_0 \]

• Find configuration for which \( C = 0 \)

• Search along \( \nabla C \)

![Diagram showing a contour plot with \( C = 0 \) and \( \nabla C \) directions, and a note on rigid body modes.]
Constraint Projection

\[ C(x + \Delta x) = 0 \]

• Linearization (equal for distance constraint)

\[ C(x + \Delta x) \approx C(x) + \nabla C(x)^T \Delta x = 0 \]

• Correction vectors

\[ \Delta x = \lambda \nabla C(x) \]

\[ \lambda = -\frac{C(x)}{\nabla C(x)^T \nabla C(x)} \]

\[ \Delta x = \lambda M^{-1} \nabla C(x) \]

\[ \lambda = -\frac{C(x)}{\nabla C(x)^T M^{-1} \nabla C(x)} \]

\[ M = \text{diag}(m_1, m_2, \ldots, m_n) \]
Constraint Solver

• Gauss-Seidel
  – Iterate through all constraints and apply projection
  – Perform multiple iterations
  – Simple to implement
  – Atomic operations required for parallelization

• Modified Jacobi
  – Process all constraints in parallel
  – Accumulate corrections
  – After each iteration, average corrections [Bridson et al., 2002]

• Both known for slow convergence
Global Solver

• Constraint vector

\[
C(x) = \begin{bmatrix} C_1(x) \\ \vdots \\ C_M(x) \end{bmatrix} \quad \nabla C(x) = \begin{bmatrix} \nabla C_1(x)^T \\ \vdots \\ \nabla C_M(x)^T \end{bmatrix} \quad \lambda = \begin{bmatrix} \lambda_1 \\ \vdots \\ \lambda_M \end{bmatrix}
\]

\[
\Delta x = M^{-1} \nabla C(x) \lambda \\
\lambda = - \frac{C(x)}{\nabla C(x)^T M^{-1} \nabla C(x)}
\]

\[
\Delta x = M^{-1} \nabla C(x)^T \lambda \\
[\nabla C(x) M^{-1} \nabla C(x)^T] \lambda = -C(x)
\]

[Goldenthal et al., 2007]
Global vs. Gauss-Seidel

- Gradients fixed
- Linear solution ≠ true solution
- Multiple Newton steps necessary
- Current gradients at each constraint projection
- Solver converges to the true solution
Other Speedup Tricks

• Use as smoother in a multi-grid method
• Long range distance constraints (LRA)
• Shape matching
• Hierarchy of meshes
Amazing Gauss-Seidel!

- Can handle unilateral (inequality) constraints (LCPs, QPs)!
  - Fluids: separating boundary conditions [Chentanez et al., 2012]
  - Rigid bodies: LCP solver [Tonge et al., 2012]
  - Deformable objects: Long range attachments [Kim et al., 2012]
- Works on non-linear problem directly
- Handles under and over-constrained problems
- GS + PBD: garbage in, simulation out (almost 😊)
- Fine grained interleaved solver trivial
- Easy to implement and parallelize
Analysis of PBD
Correction = Acceleration

• Predicted position
  \[ p = x_n + \Delta t v_n = x_n + \Delta t \frac{(x_n - x_{n-1})}{\Delta t} = 2x_n - x_{n-1} \]

• Projection
  \[ x_{n+1} = p + \Delta x \]

\[ \Delta x = x_{n+1} - 2x_n + x_{n-1} \]
Implicit Euler

\[
M \frac{x_{n+1} - 2x_n + x_{n-1}}{\Delta t^2} = f(x_{n+1})
\]

\[
M \Delta x = \Delta t^2 f(x_{n+1})
\]

Formulation as an optimization problem for \( \Delta x \):

\[
\min \left( \frac{1}{2} \Delta x^T M \Delta x + \Delta t^2 E(x_{n+1}) \right)
\]

inertia term \hspace{1cm} energy term
Stiffness → Infinity

\[ \min \left( \frac{1}{2} \Delta x^T M \Delta x + \Delta t^2 \frac{1}{2} kC^2(x_{n+1}) \right) \]

\[ E(x) = \frac{1}{2} kC^2(x) \]

Now let \( k \to \infty \)

\[ \min \left( \frac{1}{2} \Delta x^T M \Delta x \right) \text{ subject to } C(x_{n+1}) = 0 \]

- \( C(x_{n+1}) = 0 \)
- \( M \Delta x = \lambda \nabla C(x_{n+1}) \)

\[ \Delta x = \lambda M^{-1} \nabla C(x_{n+1}) \]

PBD
Two Interpretations

\[ \nabla C = \lambda \nabla (\Delta x^2) \]

rigid body modes

\[ C = 0 \]
Constraint Solver

- PBD solves a non-linear optimization problem

\[ \min \left( \frac{1}{2} \Delta x^T M \Delta x \right) \text{ subject to } C_i(x_{n+1}) = 0, \ i \in [1, \ldots, m] \]

by solving a sequence of QPs:

\[ \min \left( \frac{1}{2} \Delta x^T M \Delta x \right) \text{ subject to } C_i(x_{n+1}) = 0 \]
Clothing Demo

Nurien
Cloth

• Slow error propagation $\rightarrow$ stretchy cloth
• Low resolution: no detailed wrinkles

• Solutions
  – Use hierarchy of meshes (complicated)
  – Has been an open problem for us
  – Found an embarrassingly simple solution
Long Range Attachments (LRA)

- **Upper distance constraint** to closest attachment point
- **Unilateral**: project only if distance too big

[Kim et al., 2012], 90k particles
Challenge

• Similar idea for compression?
• Long range distance constraint to the ground?
Rigid Objects

• Optimally match un-deformed with deformed shape
• Only allow translation and rotation
• Global correction, no propagation needed
• No mesh needed!
Position Based Fluids

• Particle based

• Pair-wise lower distance constraints → granular behavior

• Move particles in local neighborhood such that density = rest density

• Density constraint

\[ C(x_1, \ldots, x_n) = \rho_{SPH}(x_1, \ldots, x_n) - \rho_0 \]
Mesh Independent Deformations

For each triangle:

\[ C(x_1, \ldots, x_3) = G_{ij}(x_1, \ldots, x_3) \]

\[ G = F^T F - I \]

[Müller et al, 2014]
FEM

[Bender et al, 2014]

• For each tetrahedron:

\[ C(x_1, \ldots, x_4) = E_{FEM}(x_1, \ldots, x_4) \]
Unified Solver

[MacKlin et al., 2014]

- Putting it all together
- Plus
  - Static friction
  - Stiff stacks via mass modifications
  - Two-way fluid – solid coupling
Acknowledgements

• PhysX Research Group

• PhysX Group

Nuttapong Chentanez
Tae-Yong Kim
Miles Macklin