Lecture IX: Position-Based Dynamics
Previously: Constraints Calculus

• Constraint: $C(p, p', p'') = 0$.
• Velocity always tangent: $J \ddot{v} = 0$
• Constraint force always normal (virtual work):

$$
\vec{F}_c = \sum_i \lambda_i J_i \cdot \vec{n} = J^T \lambda
$$
Previously: Sequential Impulses

- **while (!done) {**
  - **for** all constraints c **do** solve c;
- **}**

- Computing the impulse:
  - For old velocity: $J_i v \neq 0$
  - For new velocity: $J_i (v + \Delta v) = 0$.
  - Apply impulse: $P = M \Delta v = J_i^T \lambda_i$, $M$ is mass matrix.

- To compute $\lambda_i$
  \[
  J_i (v + M^{-1} J_i^T \lambda_i) = 0
  \]
  \[
  \lambda_i = \frac{-J_i v}{J_i M^{-1} J_i^T}
  \]
Previously: Semi-implicit Integration

• Also known as *symplectic*.

• Velocity integrated forward:

\[ v(t + \Delta t) = v(t) + \frac{F}{m} \Delta t \]

• Position integrated backwards:

\[ x(t + \Delta t) = x(t) + \Delta t v(t + \Delta t) \]
Position-Based Dynamics Principles

• Velocity always set to match position change at each iteration:
  \[ v = \frac{\Delta p}{\Delta t} \]

• Constraints expressed by positions alone
  \[ C(p, p', p'') = C(p) = 0 \]

• “Classic” version: works on particles.
  • Connection (like rigidity) modeled with constraints.
  • Does not explicitly handle rotations
The Position-Based Game-Engine Loop

Forces $\vec{F}(t)$

Integrate velocities and positions

Previous state: $\begin{pmatrix} \vec{v}(t) \\ \vec{x}(t) \end{pmatrix}$

Per-body change
$\begin{pmatrix} \vec{v}(t + \Delta t) \\ \vec{x}(t + \Delta t) \end{pmatrix}$

Resolve Constraints Sequentially

Position correction
$\begin{pmatrix} \vec{v}(t + \Delta t) \\ \vec{x}(t + \Delta t) \end{pmatrix}$

Velocity from positions

Position correction
$\vec{v}(t + \Delta t) = \vec{x}(t + \Delta t) - \vec{x}(t) / \Delta t$

Resolve Collisions

Position correction
$\begin{pmatrix} \vec{v}(t + \Delta t) \\ \vec{x}(t + \Delta t) \end{pmatrix}$
Position-Based Dynamics

• Advantages
  • Unconditional stability
  • Modularity and uniformity

• Disadvantages
  • Not physically accurate (but visually OK)
  • Resolves rotation by constraint solving
    • But then uniformly handles non-rigidity.

https://www.youtube.com/watch?v=j5igW5-h4ZM
Constraints Solving

• For body positions $X = x_1, x_2, \ldots, x_m$.
• Set of equality constraints $C_i(X) = 0$.
  • lengths, connectivity.
• Set of inequality constraints $C_i(X) \geq 0$.
  • Collision, deformation limits.
• Sequential solution:
  • For each violating $C_i(X)$ in turn, compute $\Delta X$ s.t. $C_i(X + \Delta X)$ is valid.
Conservation

• Conserving linear momentum:
  \[ \sum_{i=1}^{m} m_i \Delta x_i = 0 \]

• In matrix writing: \( M\Delta X = 0 \).
  • \( M = \text{diag}(m_1, m_1, m_1, \ldots, m_n, m_n, m_n) \).

• Conserving angular momentum (\( r_i \) to a fixed origin)
  \[ \sum_{i=1}^{m} r_i \times m_i \Delta x_i = 0 \]
Resolving Constraints

• Linear approximation:
  \[ C_i(p + \Delta p) \approx C_i(p) + \nabla C_i \cdot \Delta p \]

• Closest point: move in the gradient direction.
  \[ \Delta p = \lambda \nabla C_i \]

• To conserve momenta: Alternative weighting
  \[ \Delta p = \lambda_i M^{-1} \nabla C_i \]
    • Intuitive explanation: exchanging impulses and not velocities.

• For equality constraints:
  \[ C_i(p) + \lambda_i \nabla C_i^T M^{-1} \nabla C_i = 0 \]
    \[ \lambda_i = \frac{-C_i(p)}{\nabla C_i^T M^{-1} \nabla C_i} \]
Inequality Constraints

• Essentially resolved the same
  \[ C_i(p + \Delta p) \approx C_i(p) + \nabla C_i \cdot \Delta p \geq 0 \]

• Main difference: projection only performed if constraint is violated
  • Example: if collision happened.

Not Valid

Projection into valid state
Constraint Stiffness

• Scaling each constraint by a scalar \( k_i \in [0,1] \).
• Then we use: \( \Delta p = \lambda_i k_i M^{-1} \nabla C_i \)
• Makes the constraint “less” or “more” important.
  • The solution space is biased towards a solution that is closest to the components of the previous solution that satisfy the important constraints.
Stretch Constraint

• Between two points: \( C(x_1, x_2) = |x_1 - x_2| - d_{12} \).

• Gradient components: \( \nabla_1 C = \hat{n}, \nabla_2 C = -\hat{n}, \hat{n} = \frac{x_1 - x_2}{|x_1 - x_2|} \).

• Resulting movement (\( w = m^{-1} \)):

\[
\Delta x_1 = \frac{-w_1}{w_1 + w_2} C(x_1, x_2)\hat{n}
\]

\[
\Delta x_2 = \frac{w_2}{w_1 + w_2} C(x_1, x_2)\hat{n}
\]
Connectors (Attachment Constraint)

• If two (rigid) objects need to connect at two points $x_1, x_2$.

• Add connector constraint $C_i(x_1, x_2) = x_1 - x_2$.
  • Usually with the highest stiffness $k_i = 1$.

• Extensions: 1D translational constraint along $\hat{n}$:

  $$C_i(x_1, x_2) = (x_1 - x_2)\hat{n} = 0$$
Example (rigid collision)

STEP 0

STEP 1 before constraints

STEP 1 after 1st constraint
Example

Rotation is induced by rigidity!

STEP 1
after all constraints multiple times

STEP 1
(implicit) velocities
Collision Constraint

• In previous iteration: $x_i$ is out of the object.
• After position integration: $x_{i+1}$ is penetrating.
• Penetration normal: $\hat{n}$ (pointing outwards)
• Must make sure: the projected point $x_{i+1}$ is non-penetrating.
• Constraint: $C(x_{i+1}) = (x_{i+1} - x_i)\hat{n} \geq 0$.
  • with the highest stiffness $k_i = 1$
• Disadvantage: not re-colliding objects.
  • Artefacts usually negligible.
Specific Collision Constraint: Point through Triangle

- A point orientation vs. triangle normal:
  \[ C_{ptt}(q, x_1, x_2, x_3) = (q - x_1) \cdot \frac{(x_2 - x_1) \times (x_3 - x_1)}{|(x_2 - x_1) \times (x_3 - x_1)|} - h \]
- \( h \) - thickness of triangle
  - Used for cloth simulation.
Bending Constraint

• Trying to preserve dihedral angle $\vartheta_{fg}$ between triangle $f$, $g$ as much as possible.

$$C_{bend}(x_1, x_2, x_3, x_4) = \arccos(\hat{n}_f \cdot \hat{n}_g) - \vartheta_{fg}$$

• $\hat{n}_f$: normal to triangle $f$ (resp. $g$)

$$\hat{n}_f = \frac{(x_2 - x_1) \times (x_3 - x_1)}{|(x_2 - x_1) \times (x_3 - x_1)|}$$
Velocity Damping

• For integrating velocities, before integrating positions.
  \[ v_i \leftarrow v_i + k_{damp} \Delta v_i \]
  
• Many schemes exist.
Position-Based Particles

• **Idea**: represent entire mesh as set of particles and their inter-relations
• **Advantage**: much easier and more uniform constraints
• **Disadvantage**: hard to truly approximate surfaces

Unified Particle Physics for Real-Time Applications [Macklin et al. 2014]
Solid Representation

• **Algorithm:** the usual position-based approach.
  - Position $\vec{x}$
  - velocity $\vec{v}$
  - inverse mass $w$
  - radius $r$
  - affiliation (which mesh does particle belong to).

• **Approximation:** usually vertex-based.
Collision Detection

• Just sphere to sphere.
  • Or sphere to plane (floor)
• Naïve collision constraint:
  \[ c(\vec{x}_i, \vec{x}_j) = |\vec{x}_i - \vec{x}_j| - (r_i + r_j) > 0 \]
• Problem: interlocking.
• Solution: Directional rejection.
Directional Rejection

• Penetration normal $\vec{n}_{ij}$
  • From the geometry of the mesh
• Center-to-center vector $\vec{x}_{ij}$
• Penetration depth $d = (r_i + r_j) - |\vec{x}_{ij}|$.
• Using alternative normal:
  • $n^*_i = \begin{cases} 
\vec{x}_{ij} - 2\langle \vec{x}_{ij}, \vec{n}_{ij} \rangle \vec{n}_{ij} & \langle \vec{x}_{ij}, \vec{n}_{ij} \rangle < 0 \\
\vec{n}_{ij} & \text{else}
\end{cases}$
Rigid-Body Handling

• Can be done the usual way with rigidity constraints.
• **Alternative**: estimate COM $\vec{c}$ and average orientation $R$ for all particles of same affiliation
• Update all particles of affiliation rigidly.
• $\vec{c}$: the usual weighted average.
• How to find $R$?
Mass-Spring System

• Object consists of **point masses** $m_i, i = 1 \cdots n$
• Connected by a network of **massless springs**.
• System state: **positions** $x_i$ and **velocities** $v_i$.
• Point force sum $f_i$:
  • External forces (e.g. gravity, friction).
  • Spring connections with neighbors.

Mass-Spring System

• Mass points are initially regularly spaced in a 3D lattice.
• The edges are connected by **structural springs**.
  • resist longitudinal deformations
• Opposite corner mass points are connected by **shear springs**.
  • resist shear deformations.
• The rest lengths define the rest shape of the object.
Mass-Spring System

• The force acting on mass point $i$ generated by the spring connecting $i$ and $j$ is

\[ f_i = K_i \left( |\ddot{x}_{ij}| - l_{ij} \right) \frac{\ddot{x}_{ij}}{|\ddot{x}_{ij}|} \]

• $\ddot{x}_{ij} = x_j - x_i$.
• $K_i$: stiffness of the spring.
• $l_{ij}$: rest length.

• To simulate dissipation of energy, a damping force is added:

\[ f_i = K_i \langle \dot{v}_{ij}, \ddot{x}_{ij} \rangle \frac{\ddot{x}_{ij}}{|\ddot{x}_{ij}|^2} \]
Mass-Spring System

- **Pro**: intuitive and simple to implement.
- **Con**: Not accurate and does not necessarily converges to correct solution.
  - depends on the mesh resolution and topology
  - ...and the choice of spring constants.
- Can be good enough for games, especially cloth animation
  - For possible strong *stretching* resistance and weak *bending* resistance.