Position Based Dynamics

Tags	Papers
Address	https://matthias-research.github.io/pages/publications/posBasedDyn.pdf
Author	Bruno Heidelberger, John Ratcliff, Marcus Hennix, Matthias Müller
Preference	High
Status	Done
Year	2006

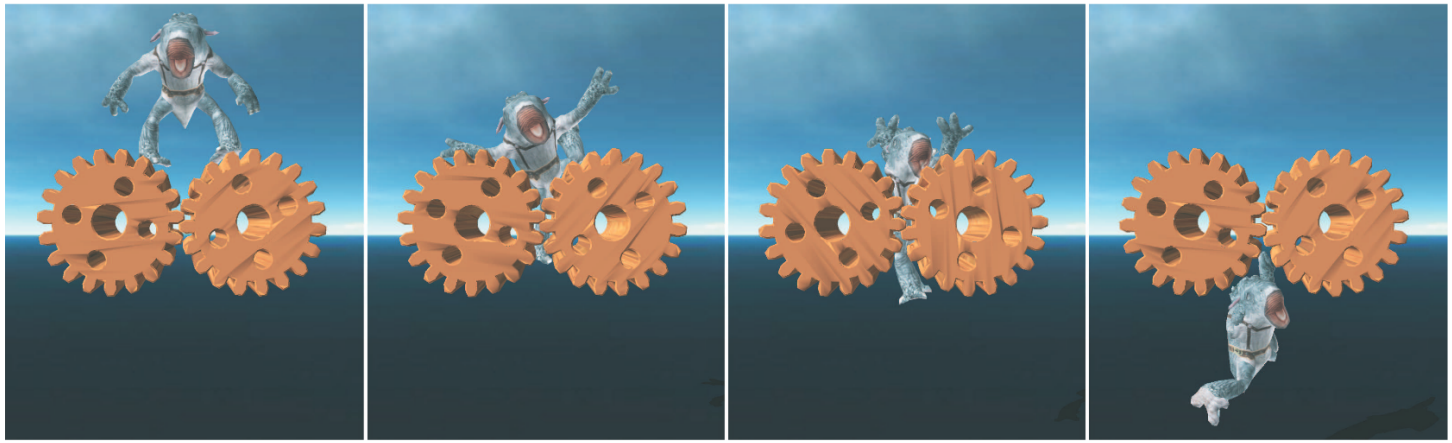

Figure 1: A known deformation benchmark test, applied here to a cloth character under pressure.

Abstract

- Force-based method:
  - Internal and external forces are accumulated from which accelerations are computed based on **Newton's second law of motion**.
  - A time integration method is then used to update the velocities and finally the positions of the object.
  - A few simulation methods (most rigid body simulators) use **impulse-based dynamics** and **directly manipulate velocities.**
- Position-based method:
  - **omit the velocity layer as well and immediately work on the positions**
  - The main advantage of a position-based method is **controllability**
  - Avoid the overshooting problems of explicit integration schemes in force-based systems
  - **collision constraints can be handled easily**
  - By projecting points to valid locations, the penetrations can be resolved completely
  - We have used the approach to build a real-time **cloth simulator**, which is part of a physics software library for games.
  - This application demonstrates the strengths and benefits of the method.

3. Position-Based Simulation

3.1 Algorithm Overview

- Dynamic object representation:
  - a set of N vertices and M constraints
  - vertex $i \in [1, \dots, N]$
    - a mass m_i
    - a position $\mathbf{x}_i$
- Given this data and a time step Δt

Algorithm 1 Position-based dynamics

1: **for all** vertices i **do**

2: initialise $\mathbf{x}_i = \mathbf{x}_i^0, \mathbf{v}_i = \mathbf{v}_i^0, w_i = 1/m_i$

3: **end for**

        - a velocity $\mathbf{v}_i$
- Constraint $j \in [1, \ldots, M]$
    - a cardinality n_j
    - a function $C_j : \mathbb{R}^{3n_j} \rightarrow \mathbb{R}$,
    - a set of indices $i_1, ...i_{n_j}, i_k \in [1, ...N]$
    - a stiffness parameter $k_j \in [0, ...1]$ (defines the strength of the constraint)
    - a type of either ***equality*** or ***inequality***
        - Equality: $C_j(\mathbf{x}_{i_1}, ...\mathbf{x}_{i_{n_j}}) = 0$
        - Inequality: $C_j(\mathbf{x}_{i_1}, ...\mathbf{x}_{i_{n_j}}) \geq 0$.

4: **loop**

5: **for all** vertices i **do** $\mathbf{v}_i \leftarrow \mathbf{v}_i + \Delta t w_i \mathbf{f}_{ext}(\mathbf{x}_i)$

6: $dampVelocities(\mathbf{v}_1, \ldots, \mathbf{v}_N)$

7: **for all** vertices i **do** $\mathbf{p}_i \leftarrow \mathbf{x}_i + \Delta t \mathbf{v}_i$

8: **for all** vertices i **do** $genCollConstraints(\mathbf{x}_i \rightarrow \mathbf{p}_i)$

9: **loop** solverIteration **times**

10: $projectConstraints(C_1, \ldots, C_{M+M_{coll}}, \mathbf{p}_1, \ldots, \mathbf{p}_N)$

11: **end loop**

12: **for all** vertices i **do**

13: $\mathbf{v}_i \leftarrow (\mathbf{p}_i - \mathbf{x}_i)/\Delta t$

14: $\mathbf{x}_i \leftarrow \mathbf{p}_i$

15: **end for**

16: $velocityUpdate(\mathbf{v}_1, \ldots, \mathbf{v}_N)$

17: **end loop**

3.2 The Solver

- $projectConstraints(C_1, \ldots, C_{M+M_{coll}}, \mathbf{p}_1, \ldots, \mathbf{p}_N)$
    - inputs:
        - the $M + M_{coll}$ constraints
        - the estimates $\mathbf{p}_1, \ldots, \mathbf{p}_N$ for the new locations of the points
- Modify the estimates to satisfy all the constraints

- Gauss-Seidel: solving each constraint independently one after the other
- Dependent on the order in which constraints are solved
- In over-constrained situations, the process can lead to **oscillations** if the order is not kept constant.

- Non-linear && Inequality

3.3 Constraint Projection

- Projection: move the points to satisfy the constraints
- Issue: the conservation of linear and angular momentum
- $\Delta \mathbf{p}_i$: the displacement of vertex i by the projection
- Linear momentum is conserved if

$$\sum_i m_i \Delta \mathbf{p}_i = \mathbf{0} \tag{1}$$

  which amounts to conserving the center of mass.

- Angular momentum is conserved if

$$\sum_i \mathbf{r}_i \times m_i \Delta \mathbf{p}_i = 0 \tag{2}$$

  where the $\mathbf{r}_i$ are the distances of the $\mathbf{p}_i$ to an arbitrary common rotation center.

- If a projection violates one of these constraints, it introduces so called **ghost forces** which act like external forces dragging and rotation the object.
- Only **internal constraints** need to conserve the momenta. Collision or attachment constraints are allowed to have global effects on the object.
- Point-based vs Force-based
  - In PBD, use the constraint function directly
- Look at a constraint with cardinality n on the points $\mathbf{p}_1, \dots \mathbf{p}_n$ with constraint function $\mathbf{C}$ and stiffness $\mathbf{k}$.
- $\mathbf{p}$: $[\mathbf{p}_1^T, ..., \mathbf{p}_n^T,]^T$.
- For **internal constraints**, **C is independent of rigid body modes**, i.e. translation and rotation.
- This means that **rotating or translating the points does not change the value of the constraint function**.
- Therefore, the gradient $\nabla_{\mathbf{p}} C$ is **perpendicular to rigid body modes** because **it is the direction of maximal change**.
- If the correction $\Delta \mathbf{p}$ is chosen to be along $\nabla C_{\mathbf{p}}$ both momenta are automatically conserved if **all masses are equal** (we handle different masses later).
- Given **p** we want to find a correction **Δp** such that **C(p + Δp) = 0**. This equation can be approximated by

$$C(\mathbf{p} + \Delta \mathbf{p}) \approx C(\mathbf{p}) + \nabla_{\mathbf{p}} C(\mathbf{p}) \cdot \Delta \mathbf{p} = 0 \tag{3}$$

- Restricting $\Delta \mathbf{p}$ to be in the direction of $\nabla_{\mathbf{p}} C$ means choosing a scalar λ such that

$$\Delta \mathbf{p} = \lambda \nabla_{\mathbf{p}} C(\mathbf{p}) \tag{4}$$

- Substituting **Eq. (4)** into **Eq. (3**), solving for λ and substituting it back into **Eq. (4)** yields the final formula for $\Delta \mathbf{p}$

- In force-based methods: derive forces via an energy term

$$\Delta \mathbf{p} = -\frac{C(\mathbf{p})}{|\nabla_{\mathbf{p}} C(\mathbf{p})|^2} \nabla_{\mathbf{p}} C(\mathbf{p}) \tag{5}$$

- which is a regular **Newton-Raphson** step for the iterative solution of the non-linear equation given by a single constraint.

- For the correction of an individual point $\mathbf{p}_i$ we have

$$\Delta \mathbf{p}_i = -s \nabla_{\mathbf{p}_i} C(\mathbf{p}_1, ..., \mathbf{p}_n) \tag{6}$$

  where the scaling factor

$$s = \frac{C(\mathbf{p}_1, ..., \mathbf{p}_n)}{\sum_j |\nabla_{\mathbf{p}_i} C(\mathbf{p}_1, ..., \mathbf{p}_n)|^2} \tag{7}$$

  is **the same for all points**.

- If the points have individual masses, we weight the corrections $\Delta \mathbf{p}_i$ by the inverse masses $w_i = 1/m_i$.
- In this case **a point with infinite mass**, i.e. wi = 0, does not move for example as expected.
- Now **Eq. (4)** is replaced by $\Delta \mathbf{p}_i = \lambda w_i \nabla_{\mathbf{p}_i} C(\mathbf{p})$, yielding

$$s = \frac{C(\mathbf{p}_1, ..., \mathbf{p}_n)}{\sum_j w_j |\nabla_{\mathbf{p}_i} C(\mathbf{p}_1, ..., \mathbf{p}_n)|^2} \tag{8}$$

  for the scaling factor and for the final correction

$$\Delta \mathbf{p}_i = -s w_i \nabla_{\mathbf{p}_i} C(\mathbf{p}_1, ..., \mathbf{p}_n) \tag{9}$$

- Example: consider the distance constraint function $C(\mathbf{p}_1, \mathbf{p}_2) = |\mathbf{p}_1 - \mathbf{p}_2| - d$.
- The derivative with respect to the points are:

$$\begin{gathered} \nabla_{\mathbf{p}_1} C(\mathbf{p}_1, \mathbf{p}_2) = \mathbf{n} \\ \nabla_{\mathbf{p}_2} C(\mathbf{p}_1, \mathbf{p}_2) = -\mathbf{n} \\ \mathbf{n} = \frac{\mathbf{p}_1 - \mathbf{p}_2}{|\mathbf{p}_1 - \mathbf{p}_2|} \end{gathered}$$

- The scaling factor s is, thus, $s = \frac{|\mathbf{p}_1 - \mathbf{p}_2| - d}{w_1 + w_2}$ and the final corrections

$$\Delta \mathbf{p}_1 = -\frac{w_1}{w_1 + w_2} (|\mathbf{p}_1 - \mathbf{p}_2| - d) \frac{\mathbf{p}_1 - \mathbf{p}_2}{|\mathbf{p}_1 - \mathbf{p}_2|} \tag{10}$$

$$\Delta \mathbf{p}_2 = +\frac{w_2}{w_1 + w_2} (|\mathbf{p}_1 - \mathbf{p}_2| - d) \frac{\mathbf{p}_1 - \mathbf{p}_2}{|\mathbf{p}_1 - \mathbf{p}_2|} \tag{11}$$

  which are the formulas proposed in [Jak01] for the projection of distance constraints (see **Figure 2**). They pop up as a special case of the general constraint projection method.

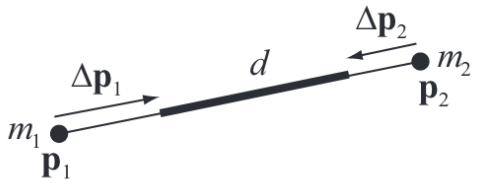

Figure 2: Projection of the constraint C(p1, p2) = |p1 − p2| − d. The corrections Δpi are weighted according to the inverse masses wi = 1/mi.

- If the constraint type is equality: always perform a projection
- If the constraint type is inequality: projection performed if $C(\mathbf{p}_1, \ldots, \mathbf{p}_n) < 0$

- replace the stiffness $k \rightarrow k' = 1 - (1 - k)^{1/n_s}$
- the error $\Delta\mathbf{p}(1 - k')^{n_s} = \Delta\mathbf{p}(1 - k)$, linearly dependent on k and independent of n_s

3.4 Collision Detection and Response

- Generate the collision constraints M_{coll}: $generateCollisionConstrains(\mathbf{x}_i \rightarrow \mathbf{p}_i)$
- For continuous collision handling:
  - test each vertex i the ray $\mathbf{x}_i \rightarrow \mathbf{p}_i$
  - if this ray enters an object,
    - compute the entry point $\mathbf{q}_c$
    - compute the surface normal $\mathbf{n}_c$ at $\mathbf{q}_c$
    - inequality constraint: $C(\mathbf{p}) = (\mathbf{p} - \mathbf{q}_c) \cdot \mathbf{n}_c$
    - stiffness $k = 1$
  - if the ray lies completely inside an object
    - compute the surface point $\mathbf{q}_s$ which is closest to $\mathbf{p}_i$

- Friction and restitution can be handled by manipulating the velocities of colliding vertices in step (16) $velocityUpdate(\mathbf{v}_1, ..., \mathbf{v}_N)$ of the algorithm.
- The velocity of each vertex generate collision constraint:
  - dampened perpendicular to the collision normal
  - reflected in the direction of the collision normal.
- Correct response for two dynamic colliding objects, e.g.
  - if a point $\mathbf{q}$ of one objects moves through a triangle $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3$ of another object
  - insert an **inequality constraint** with constraint function $C(\mathbf{q}, \mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3) = \pm(\mathbf{q} - \mathbf{p}_1) \cdot [(\mathbf{p}_2 - \mathbf{p}_1) \times (\mathbf{p}_3 - \mathbf{p}_1)]$

- compute the surface normal $\mathbf{n}_s$ at $\mathbf{q}_s$
- inequality constraint: $C(\mathbf{p}) = (\mathbf{p} - \mathbf{q}_s) \cdot \mathbf{n}_s$
- stiffness $k = 1$
    - which keeps the point q on the correct side of the triangle.

3.5 Damping

- the system properties

  (1) $\mathbf{x}_{cm} = (\sum_i \mathbf{x}_i m_i / (\sum_i \mathbf{m}_i)$

  (2) $\mathbf{v}_{cm} = (\sum_i \mathbf{v}_i m_i / (\sum_i \mathbf{m}_i)$

  (3) $\mathbf{L} = \sum_i \mathbf{r}_i \times (m_i \mathbf{v}_i)$

  (4) $\mathbf{I} = \sum_i \tilde{\mathbf{r}}_i \tilde{\mathbf{r}}_i^T m_i$

  (5) $\omega = \mathbf{I}^{-1} \mathbf{L}$

  (6) $\mathbf{foreach}\ vertices\ i$

  (7) $\quad \Delta \mathbf{v}_i = \mathbf{v}_{cm} + \omega \times \mathbf{r}_i - \mathbf{v}_i$

  (8) $\quad \mathbf{v}_i \leftarrow \mathbf{v}_i + k_{damping} \Delta \mathbf{v}_i$

  (9) $\mathbf{endfor}$

- where:
    - $\mathbf{r}_i = \mathbf{x}_i - \mathbf{x}_{cm}$
    - $\tilde{\mathbf{r}}_i$ is the 3 by 3 matrix with the property $\tilde{\mathbf{r}}_i \mathbf{v} = \mathbf{r}_i \times \mathbf{v}$, and
    - $k_{damping} \in [0 ... 1]$ is the damping coefficient.
- Lines (1)-(5): compute the global linear velocity $\mathbf{x}_{cm}$ and angular velocity ω of the system.
- Lines (6)-(9) then only damp the individual deviations $\Delta \mathbf{v}_i$ of the velocities $\mathbf{v}_i$ from the global motion $\mathbf{v}_{cm} + \omega \times \mathbf{r}_i$.
- Thus, in the extreme case $k_{damping} = 1$, only the global motion survives and the set of vertices behaves like a rigid body.
- For arbitrary values of $k_{damping}$, the velocities are globally dampened but without influencing the global motion of the vertices.

3.6 Attachments

- For vertex attach in a static object:
    - set the position as the same as the static target
- For vertex attach in a kinematic object:
    - update the position at every time step to coincide with the position of the kinematic object

- To make sure other constraints containing this vertex do not move it, its inverse mass w_i is set to zero.

4 Cloth Simulation

4.1 Representation of Cloth

- cloth representation:
  - arbitrary **triangle** meshes
    - represent a manifold, i.e. each edge is shared by at most two triangles
    - each node becomes a simulated vertex
  - density $\rho[kh/m^2]$
- For each edge, generate a stretching constraint with constraint function:

$$C_{stretch}(\mathbf{p}_1, \mathbf{p}_2) = |\mathbf{p}_1 - \mathbf{p}_2| - l_0$$

stiffness $k_{stretch}$ and type **equality**.

l_0 : the initial length of the edge

$k_{stretch}$: a global parameter provided by the user. It defines the stretching stiffness of the cloth.

- For each pair of adjacent triangles $(\mathbf{p}_1, \mathbf{p}_3, \mathbf{p}_2)$ and (p1, p2, p4) we generate a **bending constraint** with constraint function

$$C_{bend}(\mathbf{p}_1, \mathbf{p}_1, \mathbf{p}_3, \mathbf{p}_4) = acos(\frac{(\mathbf{p}_2 - \mathbf{p}_1) \times (\mathbf{p}_3 - \mathbf{p}_1)}{|(\mathbf{p}_2 - \mathbf{p}_1) \times (\mathbf{p}_3 - \mathbf{p}_1)|} \cdot \frac{(\mathbf{p}_2 - \mathbf{p}_1) \times (\mathbf{p}_4 -}{|(\mathbf{p}_2 - \mathbf{p}_1) \times (\mathbf{p}_4 -}$$

stiffness k_{bend} and type equality.

φ_0 : the **initial dihedral angle between the two triangles**

k_{bend} : **a global user parameter defining the bending stiffness of the cloth** (see **Figure 4**).

- This bending term is independent of stretching, i.e. independent of edge lengths. This way, the user can specify cloth with **low stretching stiffness but high bending resistance for instance** (see **Figure 3**).

In the appendix A we derive **the formulas to project the bending constraints**.

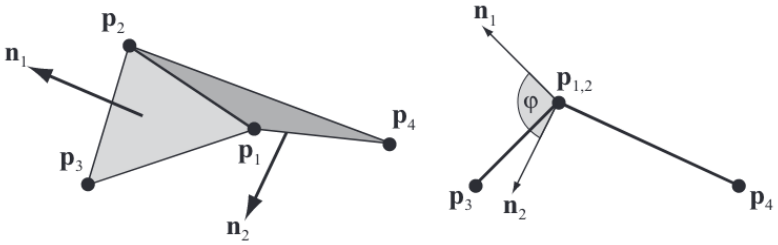

Figure 4: For bending resistance, the constraint function C(p1, p2, p3, p4) = arccos(n1 · n2) − φ0 is used. The actual dihedral angle φ is measure as the angle between the normals of the two triangles.

4.2 Collision with Rigid Bodies

- Section 3.4
- To get two-way interactions, we apply an impulse $m_i \Delta \mathbf{p}_i / \Delta t$ to the rigid body at the contact point, each time vertex i is projected due to collision with that body.
- Testing only cloth vertices for collisions is not enough because small rigid bodies can fall through large cloth triangles.
- Therefore, **collisions of the convex corners of rigid bodies against the cloth triangles are also tested.**

4.3 Self Collision

- Assumption: the triangles all have about the same size
- Use **spatial hashing** to find vertex triangle collisions.
- If a vertex $\mathbf{q}$ moves through a triangle $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3$ we use the constraint function:

$$C(\mathbf{q}, \mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3) = (\mathbf{q} - \mathbf{p}_1) \cdot \frac{(\mathbf{p}_2 - \mathbf{p}_1) \times (\mathbf{p}_3 - \mathbf{p}_1)}{|(\mathbf{p}_2 - \mathbf{p}_1) \times (\mathbf{p}_3 - \mathbf{p}_1)|} - h \quad (12)$$

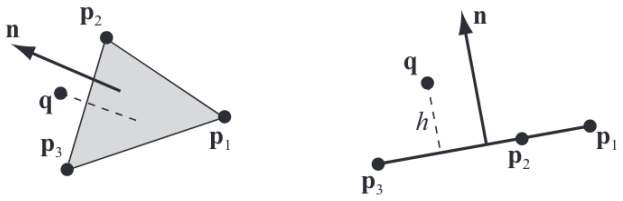

Figure 5: Constraint function C(q, p1, p2, p3) = (q − p1) · n − h makes sure that q stays above the triangle p1, p2, p3 by the the cloth thickness h.

where h is the cloth thickness (see **Figure 5**).

- If the vertex enters from below with respect to **the triangle normal**, the constraint function has to be

$$C(\mathbf{q}, \mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3) = (\mathbf{q} - \mathbf{p}_1) \cdot \frac{(\mathbf{p}_3 - \mathbf{p}_1) \times (\mathbf{p}_2 - \mathbf{p}_1)}{|(\mathbf{p}_3 - \mathbf{p}_1) \times (\mathbf{p}_2 - \mathbf{p}_1)|} - h \quad (13)$$

  to **keep the vertex on the original side**.

- Note: **Testing continuous collisions is insufficient if cloth gets into a tangled state.**

- **Projecting these constraints conserves linear and angular momentum** which is essential for cloth self collision since it is an internal process.
- **Figure 6** shows a rest state of a piece of cloth with self collisions.

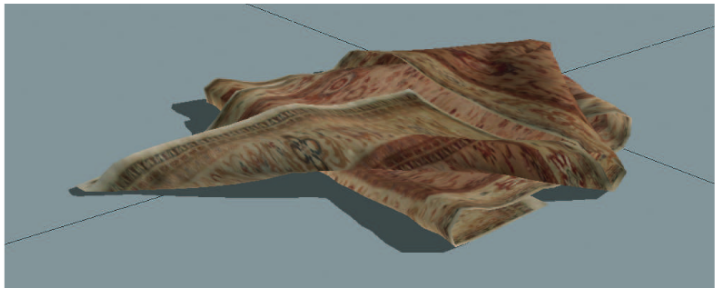

Figure 6: This folded configuration demonstrates stable self collision and response.

4.4 Cloth Balloons

- For closed triangle meshes, overpressure inside the mesh can easily be modeled (see **Figure 7**).
- Add an equality constraint concerning all N vertices of the mesh with constraint function

$$C(\mathbf{p}_1, ..., \mathbf{p}_N) = \left(\sum_{i=1}^{n_{triangles}} (\mathbf{p}_{t_1^i} \times \mathbf{p}_{t_2^i}) \cdot \mathbf{p}_{t_3^i} \right) - k_{pressure} V_0 \quad (14)$$

- This constraint function yields the gradients

- stiffness **k = 1** to the set of constraints.
- $t_1^i, t_2^i \; and \; t_3^i$ are the three indices of the vertices belonging to triangle i.
- The sum computes **the actual volume** of the closed mesh.
- It is compared against **the original volume V0** times the **overpressure factor** `kpressure`.

$$\Delta_{\mathbf{p}_i} C = \sum_{j:t_1^j=i} (\mathbf{p}_{t_2^j} \times \mathbf{p}_{t_3^j}) + \sum_{j:t_2^j=i} (\mathbf{p}_{t_3^j} \times \mathbf{p}_{t_1^j}) + \sum_{j:t_3^j=i} (\mathbf{p}_{t_1^j} \times \mathbf{p}_{t_2^j}) \quad (15)$$

- These gradients have to be scaled by **the scaling factor** given in **Eq. (7)** and weighted by **the masses** according to **Eq. (9)** to get the final projection offsets $\Delta \mathbf{p}_i$.

Figure 7: Simulation of overpressure inside a character.

5. Results

- Independent Bending and Stretching
  - the bending term only depends on the dihedral angle of adjacent triangles, not on edge lengths, so bending and stretching resistances can be chosen independently.
  - Figure 3 shows a cloth bag with various stretching stiffnesses, first with bending resistance enabled and then disabled. As the top row shows, bending does not influence stretching resistance.
- Attachments with Two Way Interaction.
  - The cloth stripes in **Figure 8** are attached via one way constraints to the static rigid bodies at the top. In addition, two way interaction is enabled between the stripes and the bottom rigid bodies. This configuration results in realistically looking swing and twist motions of the stripes. The scene features 6 rigid bodies and 3 pieces of cloth which are simulated and rendered with more than 380 fps.

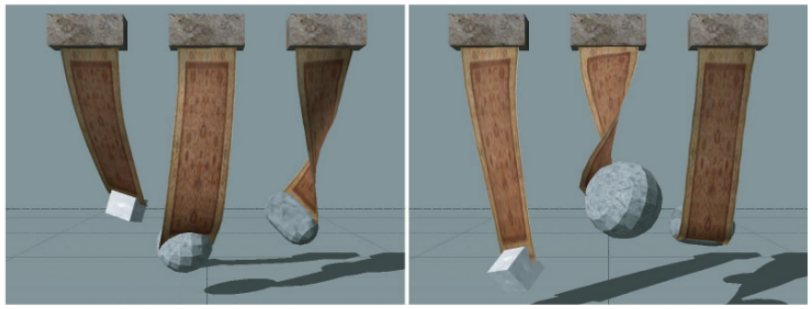

Figure 8: Cloth stripes are attached via one way interaction to static rigid bodies at the top and via two way constraints to rigid bodies at the bottom.

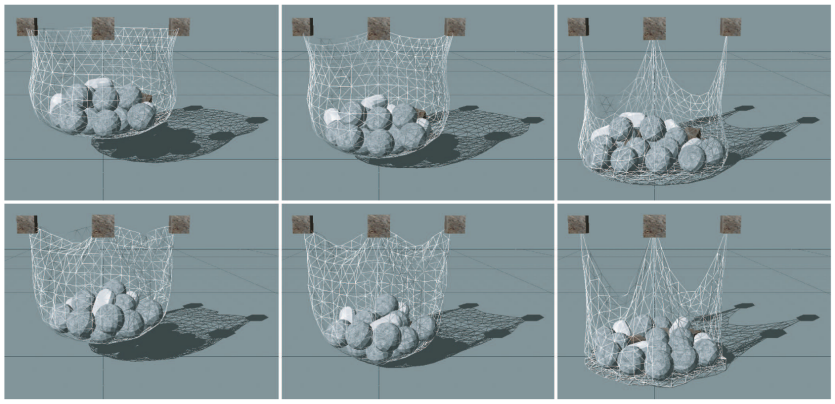

Figure 3: With the bending term we propose, bending and stretching are independent parameters. The top row shows ($k_{stretching}, k_{bending}$) = (1, 1), (1/2 , 1) and (1/100 , 1). The bottom row shows ($k_{stretching}, k_{bending}$) = (1, 0), (1/2 , 0) and (1/100 , 0).

- Real Time Self Collision.
  - The piece of cloth shown in **Figure 6** is composed of 1364 vertices and 2562 triangles. The simulation runs at 30 fps on average including self collision detection, collision handling and rendering.

- Tearing and stability.
  - **Figure 10** shows a piece of cloth consisting of 4264 vertices and 8262 triangles that is torn open by an attached cube and finally ripped apart by a thrown ball.
  - This scene is simulated and rendered with 47 fps on average. Tearing is simulated by a simple process: Whenever the stretching of an edge exceeds a specified threshold value, we select one of the edge's adjacent vertices. We then put a split plane through that vertex perpendicular to the edge direction and split the vertex. All triangles above the split plane are assigned to the original vertex while all triangles below are assigned to the duplicate.

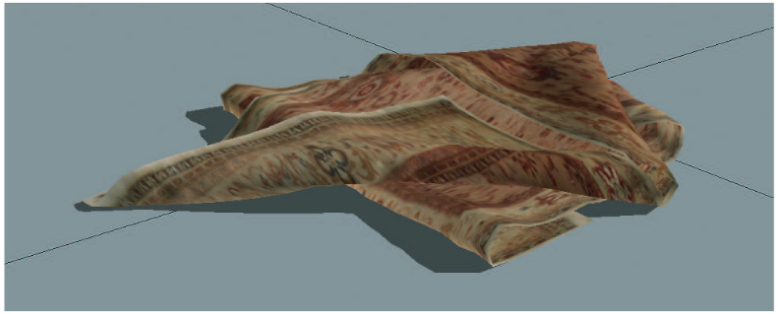

Figure 6: This folded configuration demonstrates stable self collision and response.

- The effect of friction is shown in **Figure 9** where the same piece of cloth is tumbling in a rotating barrel.

Figure 9: Influenced by collision, self collision and friction, a piece of cloth tumbles in a rotating barrel.

- Our method remains stable even in extreme situations as shown in **Figure 1**, a scene inspired by [ITF04]. An inflated character model is squeezed through rotating gears resulting in multiple constraints, collisions and self collisions acting on single cloth vertices.

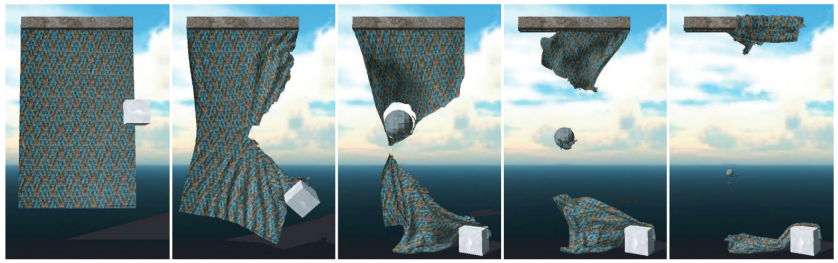

Figure 10: A piece of cloth is torn open by an attached cube and ripped apart by a thrown ball.

- **Complex Simulation Scenarios**.
  - The presented method is especially suited for complex simulation environments (see **Figure 12**).
  - Despite the extensive interaction with animated characters and geometrically complex game levels, simulation and rendering of multiple pieces of cloth can still be done at interactive speed.

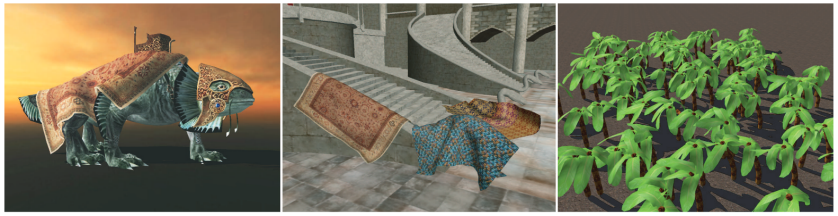

Figure 12: Extensive interaction between pieces of cloth and an animated game character (left), a geometrically complex game level (middle) and hundreds of simulated plant leaves (right).

6. Conclusions

- We have presented a **position based dynamics framework** that can handle **general constraints formulated via constraint functions**.
- With the position based approach it is possible to **manipulate objects directly** during the simulation. This significantly **simplifies the handling of collisions**, **attachment constraints** and **explicit integration** and it makes direct and immediate control of the animated scene possible.
- We have implemented **a robust cloth simulator** on top of this framework which provides features like two way **interaction of cloth with rigid bodies**, **cloth self collision** and **response and attachments of pieces of cloth to dynamic rigid bodies**.

Appendix A:

- Gradient of the Normalized Cross Product
  - Given the normalized cross product $\mathbf{n} = \frac{\mathbf{p}_1 \times \mathbf{p}_2}{|\mathbf{p}_1 \times \mathbf{p}_2|}$, the derivative with respect to the first vector is

- Bending Constraint Projection
  - beding constraint function: $C = arccos(d) - \varphi_0$, where $d = \mathbf{n}_1 \cdot \mathbf{n}_2 = \mathbf{n}_1^T \mathbf{n}_2$.

$$\frac{\partial \mathbf{n}}{\partial \mathbf{p}_1} = \begin{bmatrix} \frac{\partial n_x}{\partial p_{1x}} & \frac{\partial n_x}{\partial p_{1y}} & \frac{\partial n_x}{\partial p_{1z}} \\ \frac{\partial n_y}{\partial p_{1x}} & \frac{\partial n_y}{\partial p_{1y}} & \frac{\partial n_y}{\partial p_{1z}} \\ \frac{\partial n_z}{\partial p_{1x}} & \frac{\partial n_z}{\partial p_{1y}} & \frac{\partial n_z}{\partial p_{1z}} \end{bmatrix} \tag{16}$$

$$= \frac{1}{|\mathbf{p}_1 \times \mathbf{p}_2|} \left(\begin{bmatrix} 0 & p_{2z} & -p_{2y} \\ -p_{2z} & 0 & p_{2x} \\ p_{2y} & -p_{2x} & 0 \end{bmatrix} + \mathbf{n}(\mathbf{n} \times \mathbf{p}_2)^T \right) \tag{17}$$

Shorter and for both arguments we have

$$\frac{\partial \mathbf{n}}{\partial \mathbf{p}_1} = \frac{1}{|\mathbf{p}_1 \times \mathbf{p}_2|}(-\widetilde{\mathbf{p}}_2 + \mathbf{n}(\mathbf{n} \times \mathbf{p}_2)^T) \tag{18}$$

$$\frac{\partial \mathbf{n}}{\partial \mathbf{p}_2} = -\frac{1}{|\mathbf{p}_1 \times \mathbf{p}_2|}(-\widetilde{\mathbf{p}}_1 + \mathbf{n}(\mathbf{n} \times \mathbf{p}_1)^T) \tag{19}$$

where $\widetilde{\mathbf{p}}$ is the matrix with the property $\widetilde{\mathbf{p}}\mathbf{x} = \mathbf{p} \times \mathbf{x}$.

$$C_{bend}(\mathbf{p}_1, \mathbf{p}_1, \mathbf{p}_3, \mathbf{p}_4) = acos(\mathbf{n}_1 \cdot \mathbf{n}_2) - \varphi_0$$

$$\mathbf{n}_1 = \frac{(\mathbf{p}_2 - \mathbf{p}_1) \times (\mathbf{p}_3 - \mathbf{p}_1)}{|(\mathbf{p}_2 - \mathbf{p}_1) \times (\mathbf{p}_3 - \mathbf{p}_1)|}$$

$$\mathbf{n}_2 = \frac{(\mathbf{p}_2 - \mathbf{p}_1) \times (\mathbf{p}_4 - \mathbf{p}_1)}{|(\mathbf{p}_2 - \mathbf{p}_1) \times (\mathbf{p}_4 - \mathbf{p}_1)|}$$

- Set $\mathbf{p}_1 = 0$, get

$$\mathbf{n}_1 = \frac{\mathbf{p}_2 \times \mathbf{p}_3}{|\mathbf{p}_2 \times \mathbf{p}_3|}$$

$$\mathbf{n}_2 = \frac{\mathbf{p}_2 \times \mathbf{p}_4}{|\mathbf{p}_2 \times \mathbf{p}_4|}$$

- With $\frac{d}{dx} arccos(x) = -\frac{1}{\sqrt{1-x^2}}$ we get the following gradients:

$$\Delta_{\mathbf{p}_3} C = -\frac{1}{\sqrt{1-d^2}}(\left(\frac{\partial \mathbf{n}_1}{\partial \mathbf{p}_3}\right)^T \mathbf{n}_2) \tag{20}$$

$$\Delta_{\mathbf{p}_4} C = -\frac{1}{\sqrt{1-d^2}}(\left(\frac{\partial \mathbf{n}_2}{\partial \mathbf{p}_4}\right)^T \mathbf{n}_1) \tag{21}$$

$$\Delta_{\mathbf{p}_2} C = -\frac{1}{\sqrt{1-d^2}}(\left(\frac{\partial \mathbf{n}_1}{\partial \mathbf{p}_2}\right)^T \mathbf{n}_2 + \left(\frac{\partial \mathbf{n}_2}{\partial \mathbf{p}_2}\right)^T \mathbf{n}_1) \tag{22}$$

$$\Delta_{\mathbf{p}_1} C = -\Delta_{\mathbf{p}_2} C - \Delta_{\mathbf{p}_3} C - \Delta_{\mathbf{p}_4} C \tag{23}$$

Using the gradients of normalized cross products, first compute :

$$\mathbf{q}_3 = \frac{\mathbf{p}_2 \times \mathbf{n}_2 + (\mathbf{n}_1 \times \mathbf{p}_2)d}{|\mathbf{p}_2 \times \mathbf{p}_3|} \tag{24}$$

$$\mathbf{q}_4 = \frac{\mathbf{p}_2 \times \mathbf{n}_1 + (\mathbf{n}_2 \times \mathbf{p}_2)d}{|\mathbf{p}_2 \times \mathbf{p}_4|} \tag{25}$$

$$\mathbf{q}_2 = -\frac{\mathbf{p}_3 \times \mathbf{n}_2 + (\mathbf{n}_1 \times \mathbf{p}_3)d}{|\mathbf{p}_2 \times \mathbf{p}_3|} - \frac{\mathbf{p}_4 \times \mathbf{n}_1 + (\mathbf{n}_2 \times \mathbf{p}_4)d}{|\mathbf{p}_2 \times \mathbf{p}_4|} \tag{26}$$

$$\mathbf{q}_1 = -\mathbf{q}_2 - \mathbf{q}_3 - \mathbf{q}_4 \tag{27}$$

Then the final correction is

$$\Delta\mathbf{p}_i = -\frac{w_i\sqrt{1-d^2}(arccos(d) - \varphi_0)}{\sum_j w_j|\mathbf{q}_i|^2}\mathbf{q}_i \tag{28}$$