Workshop Series: Position-Based Dynamics (e.g., Verlet System)

Benjamin Kenwright[1]*

Abstract
This practical focuses on position-based dynamics using the Verlet integration scheme. The student needs to implement a real-time interactive soft-body effect using Verlet mechanics (e.g., cloth with collision detection, forces, and movement).

Keywords
Verlet Integration, Distance Constraints, Position-Based Dynamics, Newtonian Mechanics, Hooke's Law, Springs, Particles, Constraints, Soft-Bodies, Classical Mechanics

[1] *Workshop Series (www.xbdev.net) - Benjamin Kenwright*

Contents

Introduction

Position-Based Dynamics The topic of this practical is the implementation of a real-time particle constraint simulation (i.e., an interconnected set of particle bodies). The user should be able to interact with the simulation while it is running (e.g., through the mouse or keyboard) to control the cloth simulation- such as push or pull the object around. The cloth should be implemented using position-based mechanic principles (e.g., Verlet velocity approximation integration, with distance 'snapping' constraints, and environmental collision detection, such as, sphere-sphere and sphere-plane for the ground).

Tasks

1. Visually display interconnected position-based simulation system using the Verlet method (e.g, a cloth)
2. User input (e.g., mouse or keyboard) to control and move the Cloth around
3. Cloth effect should be under the influence of gravity and come to rest on the ground (i.e., sphere-plane collision detection)

1. Overview

Principles and Concepts Position-based dynamics allow us to easily resolve collision constraints while resolving penetration violations completely by projecting intersecting points to valid locations. We use this approach in this practical to build a real-time soft-body simulator which is part of a physics library of interactive environments, such as games. This approach demonstrates the strengths and benefits of the position-based method (e.g., Verlet integration).

Remember that, at this point, a particle physics system is still dealing with forces and masses - the internal workings of the simulation system works with a velocity approximation (i.e,. the velocity is calculated based on the current and previous position - 'velocity-less' simulation).

Position-Based Dynamics Features The main features and advantages of position based dynamics are

- The formulation we propose allows the handling of general constraints in the position based setting
- Position based simulation gives control over explicit integration and removes the typical instability problems
- Positions of vertices and parts of objects can directly be manipulated during the simulation
- The explicit position based solver is easy to understand and implement

Cloth Generating realistic real-time cloth effects on-the-fly for interactive environments, such as games, is challenging and interesting. This practical gives an introductory explanation for students to enable them to integrate cloth effects into their demos. We explain the principles, computational over-

heads, and numerical approximations, necessary for achieving an aesthetically pleasing realistic interactive soft body effect.

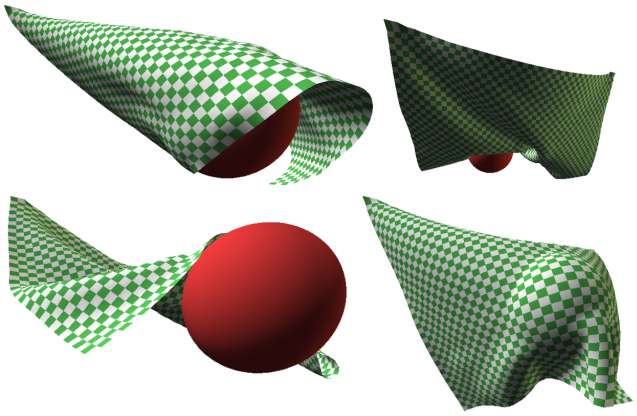

Figure 1. Animated Cloth Simulation - Real-Time Interactive Cloth Simulation Screen Capture.

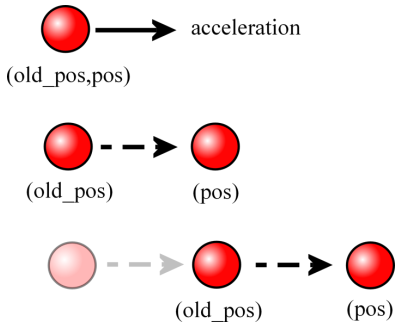

Figure 2. Verlet Integration (Velocity and Position) - 'Velocity'-less integration system (i.e., know as Verlet Integration), which uses the current and previous position.

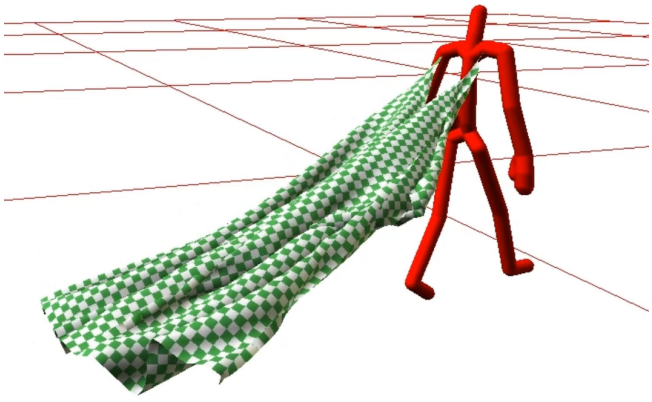

Figure 3. Screen Capture - Attaching the cloth to an animated character.

Figure 4. Screen Capture - Wrapping cloth around rigid body objects.

2. Particles

To achieve real-time frame-rates, we focus on a particle-based approach. The simulated movement of each particle uses Newtonian mechanics (i.e., $f = ma$). Hence, each particle needs to have a:

- position
- acceleration
- mass

However, since we are constantly calculating 1/mass, it's more convenient and computationally faster to store the inverse mass (i.e., *invmass*).

3. Integration

Differentiation is used to represent the 'rate-of-change' of something (e.g., change in position with time - velocity) - while integration is used to find the opposite (e.g., how the velocity causes a change in position). We apply forces to our particles, either from wind or from neighbouring constraints. From Newton's second law (i.e., $f = ma$), we can derive the acceleration. Integrating the acceleration with respect to time gives us the velocity. Integrating the velocity with respect to time gives us the position. This provides a relationship that we use to predict how the velocity and position change with respect to time in relation to the applied forces.

Euler Integration A first order approximation method that is very popular is known as Euler's integration, shown below in Equation 1:

$$\begin{aligned} v(t+1) &= v(t) + a\Delta t \\ p(t+1) &= p(t) + v(t+1)\Delta t \end{aligned} \tag{1}$$

However, for the simulation to remain stable and realistic, the time-step must be extremely small and the forces (i.e., accelerations) must remain within reasonable limits. Since we're interested in real-time interactive environments (e.g., games), stability is important. Therefore, we modify the first order

Euler equation to formulate the Verlet integration method (i.e., a velocity-less) to create a more stable and efficient technique.

Verlet Integration The Verlet integration method (i.e., velocity approximation approach) works by using the current and previous position to create the velocity (i.e., instead of using the exact velocity) as done with traditional methods in classical mechanics (see Figure 2). The velocity approximation is given below in Equation 2:

$$v(t) \approx p(t+1) - p(t) \tag{2}$$

The velocity approximation in Equation 2 is substituted into the first order Euler Equation 1 and the popular Verlet Equation 3 is produce below:

$$p(t+1) = 2p(t) - p(t-1) + a\Delta t^2 \tag{3}$$

The Verlet integration scheme is algorithmically straightforward and computationally fast to implement as shown in Listing 1 below:

Listing 1. Verlet Integration (without Damping).

```
1  Vector3 temp = pos;
2  pos = 2 * pos - old_pos  + acceleration*dt*dt
3  old_pos = temp;
```

Drag/Damping We introduce drag/damping to reduction the amplitude of oscillation and ensure the system of interconnected particles converges on a stable still result. Damping effectively causes energy to be lost (i.e., drained) from the system to overcome numerical inaccuracies and approximations to help ensure our simulation remains stable. We modify the uncomplicated Verlet integration Equation 3 to include damping as shown below in Equation 4:

$$\begin{aligned} p(t+1) &= p(t)(2-k) - (1-k)p(t-1) + a\Delta t^2 \\ &= p(t) + [p(t) - p(t-1)](1-k) + a\Delta t^2 \end{aligned} \tag{4}$$

where k is the damping constant between 0.0 and 1.0.

Verlet integration 'with' damping is a bit more complex and expensive compared to the basic Verlet scheme shown in Listing 1; however, it's a necessary evil if we want our system to remain stable and converge. The Verlet damped integration implementation is shown below in Listing 2 with the DAMPING constant set to a value between 0 and 1 (e.g., 0.9):

Listing 2. Verlet Integration (with Damping).

```
1  Vector3 temp = pos;
2  pos += (pos-old_pos)*(1.0f-DAMPING) +  acceleration*↩
       dt*dt;
3  old_pos = temp;
```

4. Cloth Interconnection Structure

Constraint-Pairs The cloth structure is formed by creating an array of constraints for each particle. Each constraint points to 'two' particles (i.e., a distance constraint). We can use as many distant constraints as are necessary to create the necessary stiff cloth effect. We iteratively update each constraint individually. As you can imagine, as we update and correct one constraint, it affects the adjacently connected particles and their constraints. However, we find that, after a finite number of iterations, the system of interconnected particles will eventually 'converge' on a result with all the constraints in a valid state. We can limit the number of iterations to ensure we maintain a real-time frame-rate.

Hooke's Law For traditional interconnected set of springs we would use the popular Hooke's law (i.e., as demonstrated in the previous practical) - where we calculate the force for each constraint and apply it to each particle. We store the rest length for the distance between each particle at the start. The error between the current and stored rest length is used to calculate the correcting force. However, for our velocity Verlet integration scheme, we can simply snap the positions into place. (i.e., see Listing 3)

Snap-To-Constraints Verlet constraints are simple. We adjust the 'current' position each frame. We work out the distance error and update the constraint so that each particle is pushed back into place. As we update each constraint, it will invalidate adjacently connected particle constraints. However, if we iteratively keep updating the constraints, after a finite time the system as a whole will converge on a result.

Listing 3. As shown in Figure 5, we need to push each constraint back into place so that the constraint is valid.

```
1  err_length = cur_length - rest_length
2  err_direction = Length(p1-p0)
3  p0 -= err_direction*err_length*0.5
4  p1 += err_direction*err_length*0.5
```

where $p0$ and $p1$ are the particle positions for the constraint, the rest length is the distance between the two particles initially (i.e., rest_length=Length(p1-p0).

Topology - Structure, Shear, and Bending We need to add additional interconnected constraints to create a cloth effect that looks like cloth (i.e., not like a set of rigid hinges). Hence, as well as the triangle edge constraints, we include shearing and bending constraints, see Figure 6.

5. Forces

We need to inject various interactive forces into our simulation, such as gravity and wind.

Gravity Gravity is added by iterating over every particle and adding a downward force. This causes the cloth to swoosh and drop.

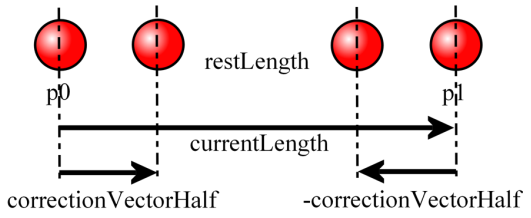

Figure 5. Fixed-Distance Constraints - Enforcing the distance between each particle remains constant (i.e., a fixed-length).

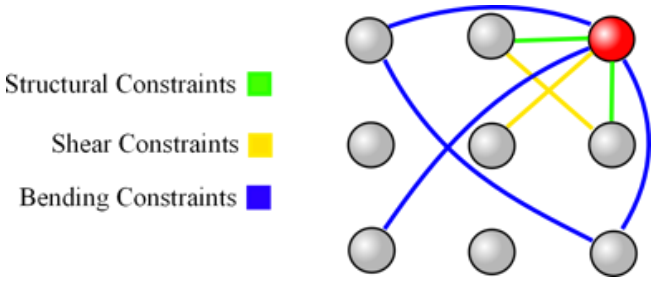

Figure 6. Interconnections - It isn't enough for us to merely connect each particle with a fixed-distance constraint. We need to add additional constraints to produce a more life-like cloth effect.

Wind Without wind the cloth effect is rather simple and doesn't really catch the viewers eyes. We can add in an additional wavy flag type effect by adding a force based on the triangles normal.

6. Contacts and Collisions

While the interconnected distance constraints that make up the cloth effect are constant and enforce the particle stay together, we can also add temporary contact constraints, that push the particles out of contact with other rigid body surfaces (e.g., spheres and planes).

The Vertlet approach uses 'position-based' physics to ensure constraint satisfaction. For example, for each particle, find the closest position that satisfies the constraint and move it there.

An uncomplicated example - if we have a ground plane (i.e., $y == 0$). For each iteration, we check if $pos.y < 0$, and if it is, we set $pos.y = 0$. Same as for the distance constraints in Section 4. The distance constraints acts like springs, i.e., solve for the equilibrium position, then force the positions of the nodes into place.

Collision Handling We compute penetration depth (i.e., between particles and objects). We move particle object along the contact normal by the penetration amount (i.e., out of penetration to be just in contact).

Sphere When a particle penetrates a sphere, we push the particle out. The direction is from the centre of the sphere to the particle, and the distance is the penetration amount.

Plane For a plane, we can use the plane equation. The principle is the same for the sphere and distance constraint. When the particle violates the constraint, that is, it penetrates the plane surface, we move the particle position along the plane normal by the penetration amount.

7. Summary

Creating a realistic real-time cloth effect has never been easier. The principle is computationally simple and relatively robust. This practical enables the student to implementation a real-time cloth effect that is stable and interactive. The practical enables students to quickly, dissect, experiment, and expand upon the basic cloth principle (e.g., working through the exercises). The initial uncomplicated implementation should produce an output similar to Figure 1.

This practical introduced position-based dynamics for soft body effects such as cloth. The technique uses a velocity approximation method and enforces constraints, such as, distance and collisions. 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

8. Exercises

This piratical only gives a brief taste of the potential of position-based dynamics. As an exercise for the student to help enhance their understanding:

Intermediate

- Add tearing to the cloth effect
- Modify maximum constraint iterations (i.e., for more softer/rigid constraints)
- Cloth fall hit floor (i.e. use plane equation for ground)
- Multiple objects (e.g., multiple cubes and spheres)
- Animate cloth - i.e., attach it to a character and move it around (cape)
- Investigate inter-cloth collisions (e.g., twisting cloth)
- Performance profile the simulation and identify bottlenecks
- Modify code for multi-threading (e.g.,)GPU/CUDA version of the code)
- Put texture on the cloth (i.e., texture map)
- Port to Sony's PS3/PSP or Microsoft's XBOX/XNA
- Disable damping and see what happens
- Add space partitioning or collision detection (e.g., octree)
- Different masses for each particle
- Update different parts of the cloths with a different number of iterations (i.e., stiffer/less stiffer constraints)
- Draw the particles with different colours to emphasis stress and maximum distance correction error

Advanced

- Implement a simple articulated ragdoll skeleton (i.e., feet, hands, head) using Verlet particle system
- Mix particle gas/fluid effects with the cloth (e.g., smoke)
- Create hair effects based on the Verlet cloth effect
- Compare the integration with an implicit integrator approach
- Create a scene with large number of cloth effects (e.g., flags, curtains, hair, character clothes) and have them move and interact based on wind and contacts

Acknowledgements

We would like to thank all the students for taking time out of their busy schedules to provide valuable and constructive feedback to make this practical more concise, informative, and correct. However, we would be pleased to hear your views on the following:

- Is the practical clear to follow?
- Are the examples and tasks achievable?
- Do you understand the objects?
- Did we missed anything?
- Any surprises?

The practicals provide a basic introduction for getting started with physics-based animation effects. So if you can provide any advice, tips, or hints during from your own exploration of simulation development, that you think would be indispensable for a student's learning and understanding, please don't hesitate to contact us so that we can make amendments and incorporate them into future practicals.

Recommended Reading

Physics for Game Developers, David M Bourg, Publisher: O'Reilly Media, ISBN: 978-1449392512

Computer Animation: Algorithms & Techniques, Rick Parent, Publisher: Morgan Kaufmann, ISBN: 978-0124158429

Code Complete: A Practical Handbook of Software Construction, Steve McConnell, ISBN: 978-0735619678

Clean Code: A Handbook of Agile Software Craftsmanship, Robert C. Martin, ISBN: 978-0132350884

Game Inverse Kinematics: A Practical Introduction (2nd Edition) Kenwright. ISBN: 979-8670628204

Kinematics and Dynamics Paperback. Kenwright. ISBN: 978-1539595496

Game Collision Detection: A Practical Introduction (Paperback). Kenwright. ISBN: 978-1511964104

Game C++ Programming: A Practical Introduction (Paperback). Kenwright. ISBN: 978-1516838165

Computational Game Dynamics: Principles and Practice (Paperback). Kenwright. ISBN: 978-1501018398

Game Physics: A Practical Introduction (Paperback). Kenwright. ISBN: 978-1471033971

Game Animation Techniques: A Practical Introduction (Paperback). Kenwright. ISBN: 978-1523210688