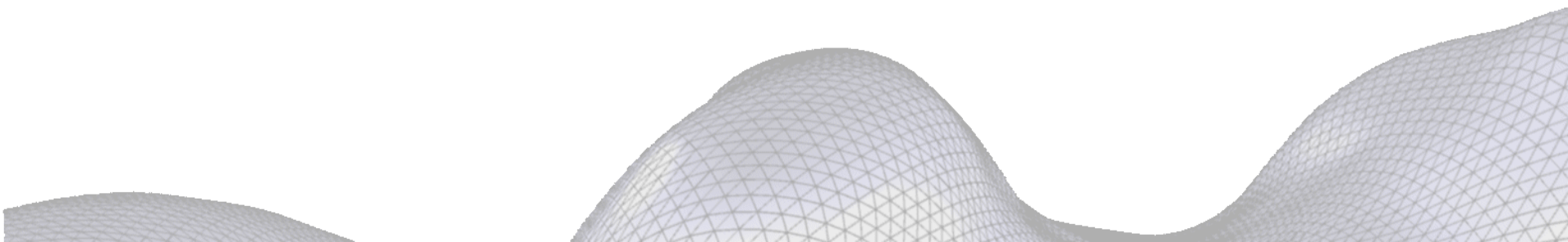
A Physics Engine Suitable for Videogames

Marco Fratarcangeli

http://www.fratarcangeli.net/

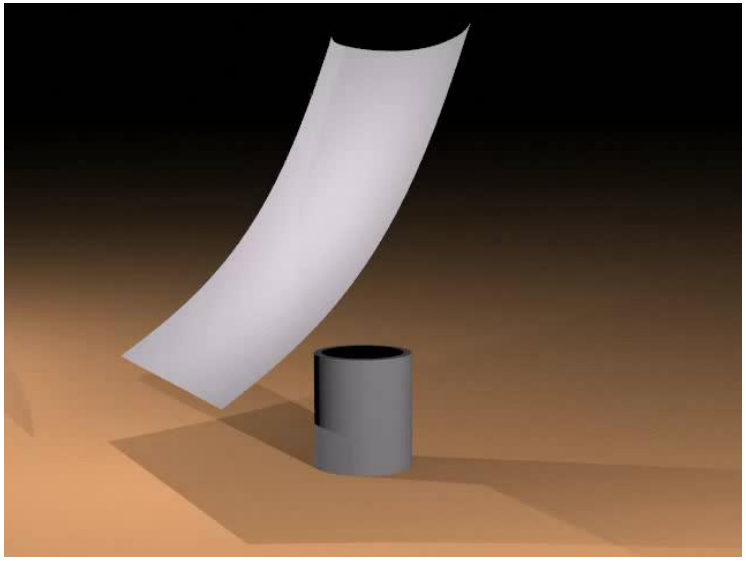
Sapienza University of Rome

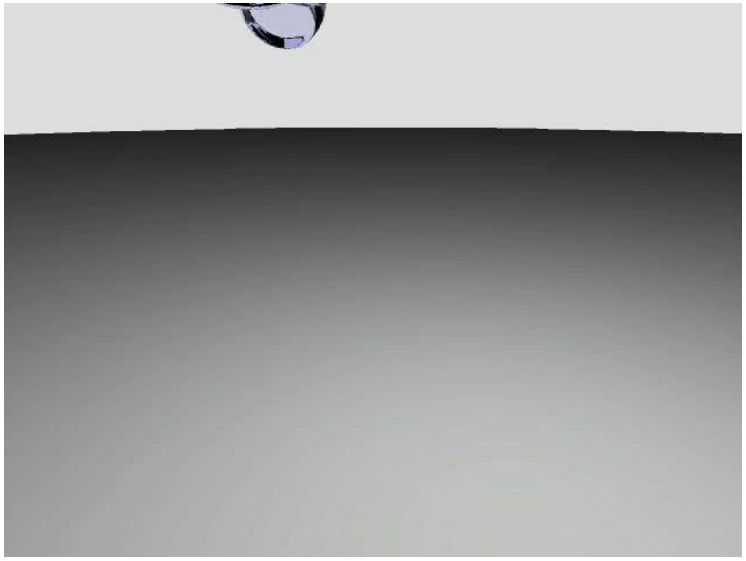
Chalmers University of Technology

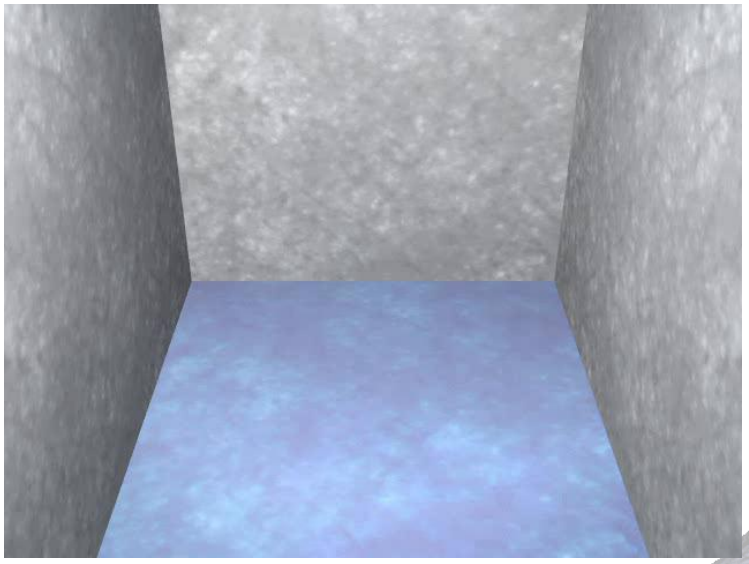


Physic Simulation

- Used to obtain realistic effects in
  - Computer Graphics (2D / 3D)
    - Videogames
    - Movies
  - Computer-aided Surgery (CaS)
    - Diagnosis
    - Prototyping of the operation
  - Structural analysis of the materials
  - ...







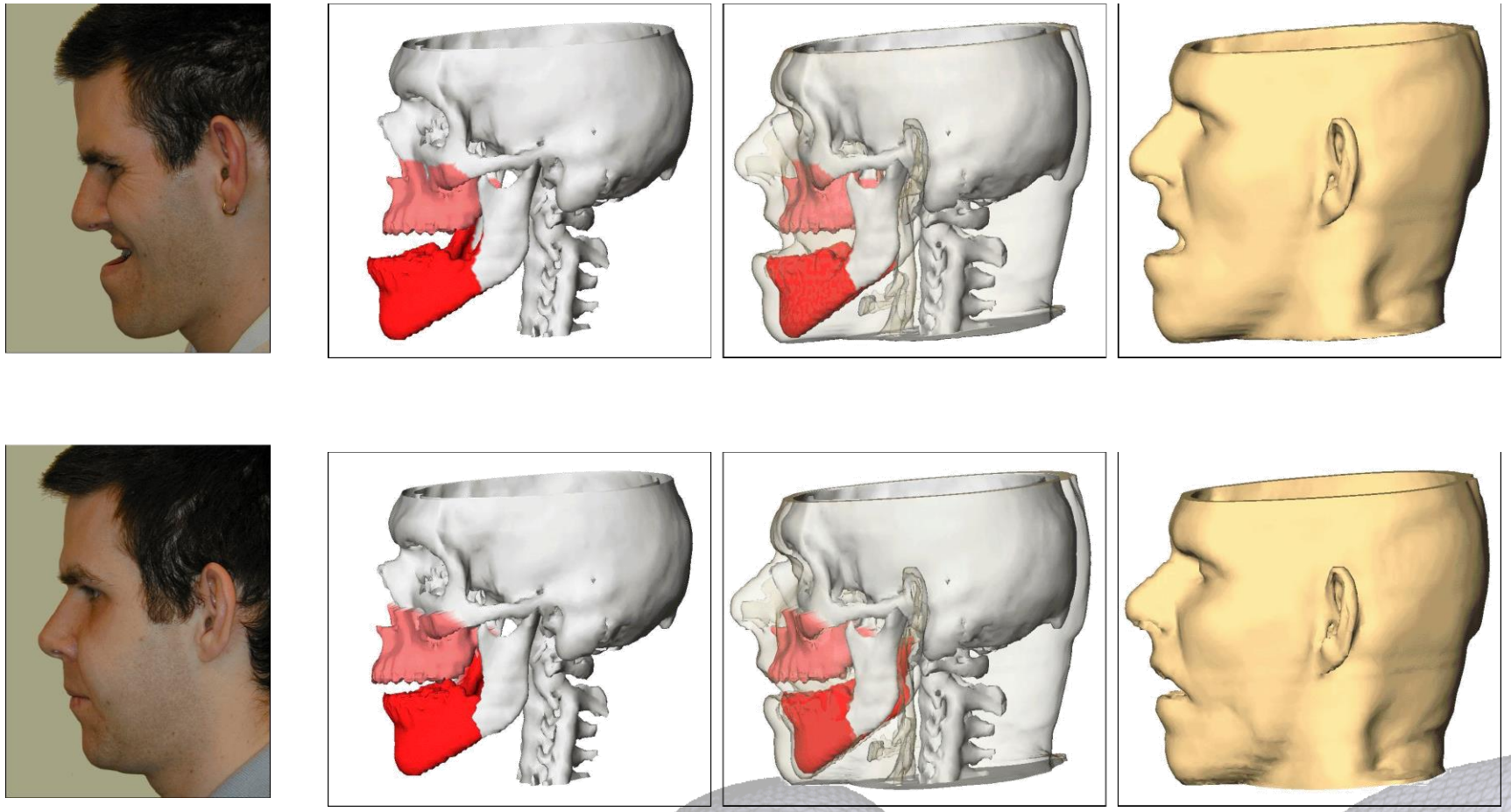
Rigid Fluid: Animating the Interplay Between Rigid Bodies and Fluid

Mark Carlson
Peter J. Mucha
Greg Turk

Georgia Institute of Technology

Sound FX by Andrew Lackey, M.P.S.E.

Medical Applications



How to Simulate a Physics System

- Basic Steps:
  - Description through the math language
    - Differential equation in continue time domain
    - Initial state of the system (a.k.a. boundary conditions)
  - Numerical Resolution in discrete time domain
- This is called a **model** of the system

$$\mathbf{v}(t) = \mathbf{v}_0 + \frac{\mathbf{f}(t)}{m} dt$$

$$\mathbf{x}(t) = \mathbf{x}_0 + \mathbf{v}(t) dt$$

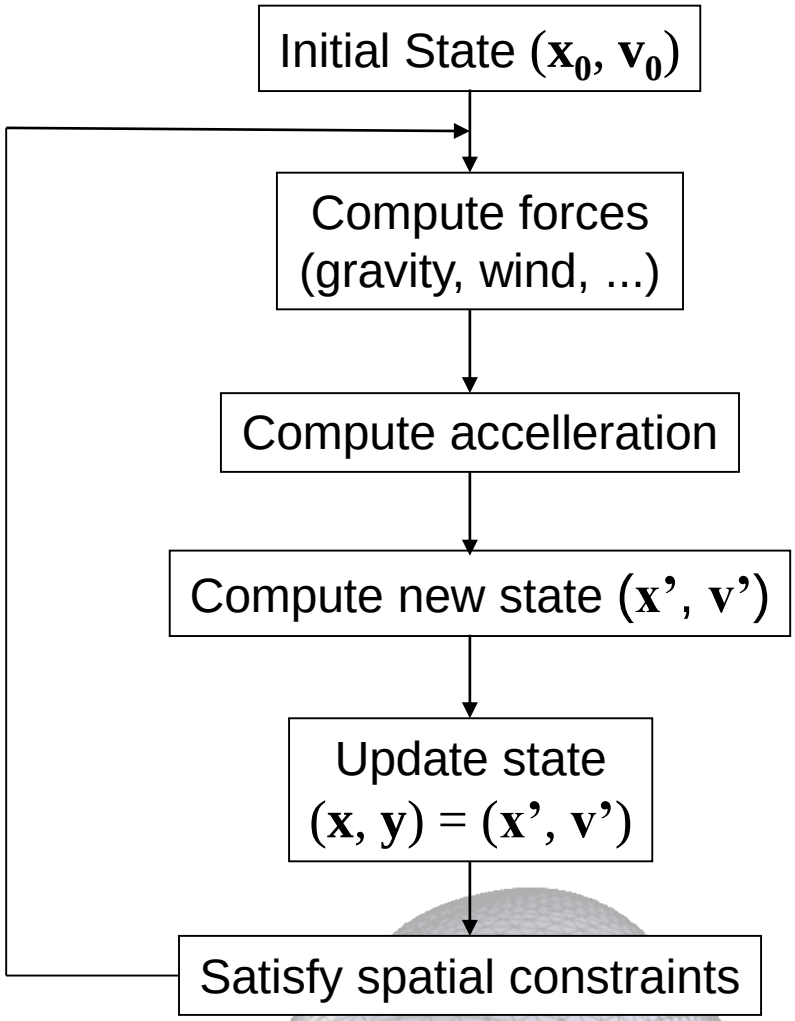
Initial state

(Euler's method)

$$\mathbf{v}_n = \mathbf{v}_{n-1} + \frac{\mathbf{f}_n}{m} \Delta t$$

$$\mathbf{x}_n = \mathbf{x}_{n-1} + \mathbf{v}_n \Delta t$$

Generic Integration Cycle



Features of the simulation

- There exist a lot of models, each one with its own advantages and drawbacks;
- The model must satisfy different requirements according to the particular application
- **Generality**
  a model is valid for different physical phenomena and different conditions;
- **Accuracy**
  results obtained from the virtual environment are nearly equal to the real ones;
- **Efficiency**
  Computation time scaled to time requirements (hard real-time, interactive, off-line.

Physics Simulation in Videogames

- **Efficiency**
  - Interactive applications require at least a scene update approx. every 33 ms (30 fps)
  - Physics simulation is allowed to use just a small amount of this time (5% - 30%)
- **Plausability**
  - Results must ***seem*** real;
  - Simulation must be stable even though the approximation errors

Case of Study

- Hitman: Codename 47 *[IO Interactive]*
- Framework for the simulation of
  - Cloth
  - Ragdoll
- Easy to implement
- High efficiency
- Scalable error/efficiency ratio
- Extremely stable

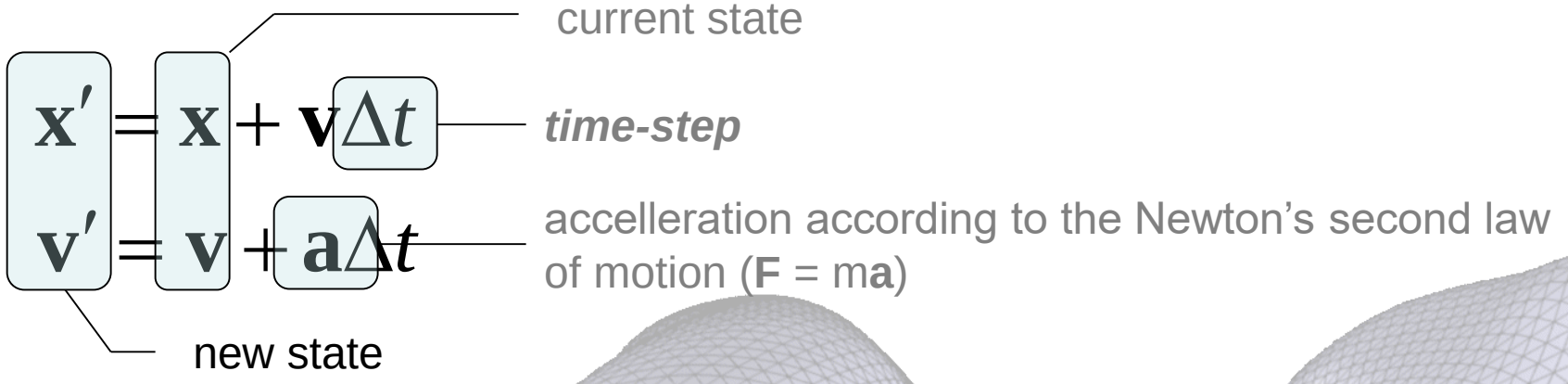
Outline

- Numerical Integration with Euler method

- Numerical Integration with Verlet method
- Collision handling and response through projection
- Spatial contraints solved by relaxation

Euler Integration

- Particle system
- Each particle has a state composed by:
  - $\mathbf{X}$: position
  - $\mathbf{V}$: velocity
- The state is updated by computing



Euler pseudo code

```
float t = 0;
float dt = 1;
float velocity = 0;
float position = 0;
float force = 10;
float mass = 1;

while ( t < = 10 )
{
      position = position + velocity * dt;
      velocity = velocity + ( force / mass ) * dt;
      t = t + dt;
}
```

Euler's problems: example

- In the real world
  $x(t) = v * t + 0.5 * a * t^2$
- Let's set v = 0 and a = 10,
  x(10) = 0.5 * 10 * 100 = 500
- With Euler's integration (time step = 1):

$$\mathbf{x}' = \mathbf{x} + \mathbf{v}\Delta t$$

$$\mathbf{v}' = \mathbf{v} + \mathbf{a}\Delta t$$

t=1: position = 0, velocity = 10

t=2: position = 10, velocity = 20

t=3: position = 30, velocity = 30

t=4: position = 70, velocity = 40

t=5: position = 110, velocity = 50

t=6: position = 160, velocity = 60

t=7: position = 220, velocity = 70

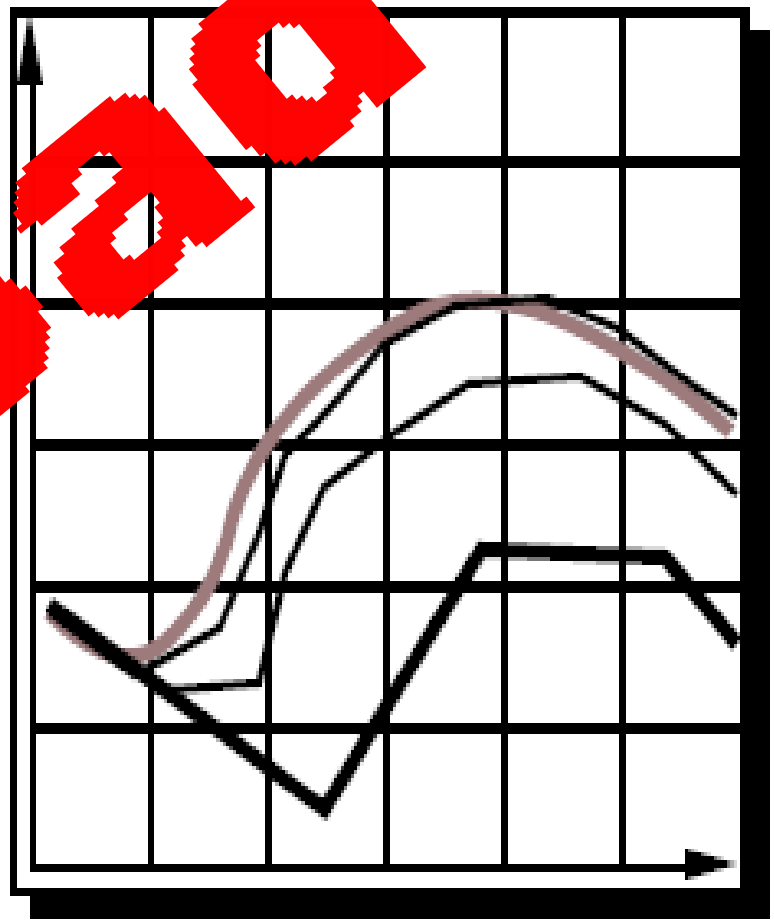
t=8: position = 290, velocity = 80

t=9: position = 380, velocity = 90

t=10: position = **470**, velocity = 100

Why Euler is never good enough?

- It is 100% accurate only if the rate of change is constant over the timestep
- if we shrink the time-step the error is smaller but it always grows ....
- ... and more time-steps means **increasing computational cost**



Too Bad!

Outline

- Numerical Integration with Euler method

- **Numerical Integration with Verlet method**
- Collision handling and response through projection
- Spatial contraints solved by relaxation

Verlet Integration

Taylor expansion of $\mathbf{x}(t)$ for $(t + \Delta t)$ and $(t - \Delta t)$, then sum:

$$\mathbf{x}(t+\Delta t) = \mathbf{x}(t) + \dot{\mathbf{x}}(t)\Delta t + \frac{1}{2}\ddot{\mathbf{x}}(t)\Delta t^2 + \frac{1}{6}\dddot{\mathbf{x}}(t)\Delta t^3 + O(\Delta t^4)$$

+

$$\mathbf{x}(t-\Delta t) = \mathbf{x}(t) - \dot{\mathbf{x}}(t)\Delta t + \frac{1}{2}\ddot{\mathbf{x}}(t)\Delta t^2 - \frac{1}{6}\dddot{\mathbf{x}}(t)\Delta t^3 + O(\Delta t^4)$$

=

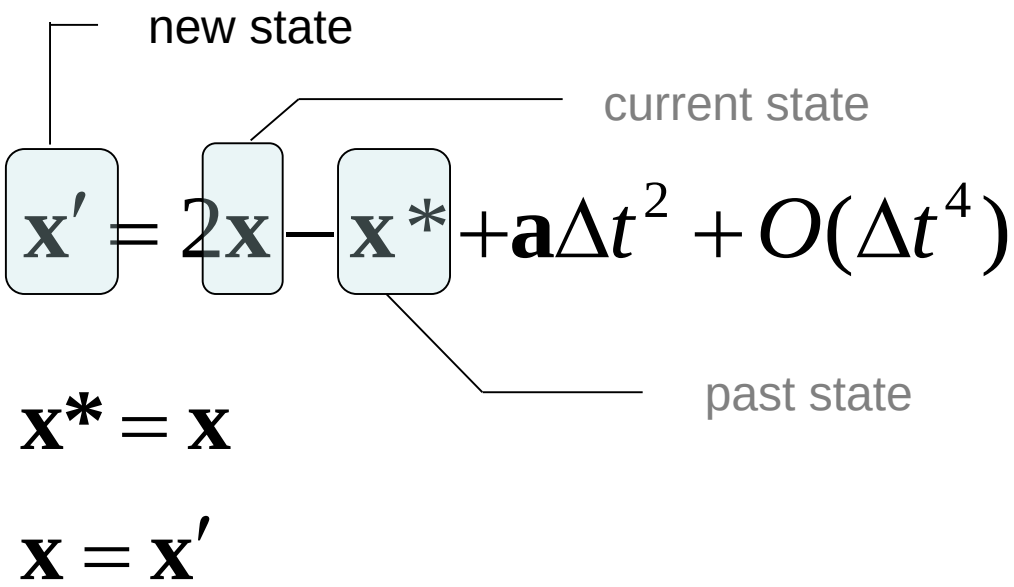
$$\mathbf{x}(t+\Delta t) + \mathbf{x}(t-\Delta t) = 2\mathbf{x}(t) + \ddot{\mathbf{x}}(t)\Delta t^2 + O(\Delta t^4)$$

rearranging...

$$\mathbf{x}(t+\Delta t) = 2\mathbf{x}(t) - \mathbf{x}(t-\Delta t) + \ddot{\mathbf{x}}(t)\Delta t^2 + O(\Delta t^4)$$

Verlet Integration

- With the same notation as before:



Verlet Integration: some comments

- Velocity term disappeared: in a single shot, the new position is computed;
- It is reversible in time: if a negative time-step is used, the system rolls back exactly to the start point;
- This means that *the energy is conserved* and thus *the [symplectic] method is very stable*

Verlet pseudo code

```
float t = 0;
float dt = 1;
float acc = 10;
float velocity = 0;
float position = 0;

float old_pos = position;
position = velocity * dt + acc * dt * dt;

while ( t < = 10 )
{
  float temp = position;
  position += position - old_pos + acc * dt * dt;
  old_pos = temp;
  t += dt;
}
```

Outline

- Numerical Integration with Euler method

- Numerical Integration with Verlet method
- **Collision handling and response through projection**
- Spatial contraints solved by relaxation

Collision handling

This is valid only because the particular shape of the Verlet integration formula:

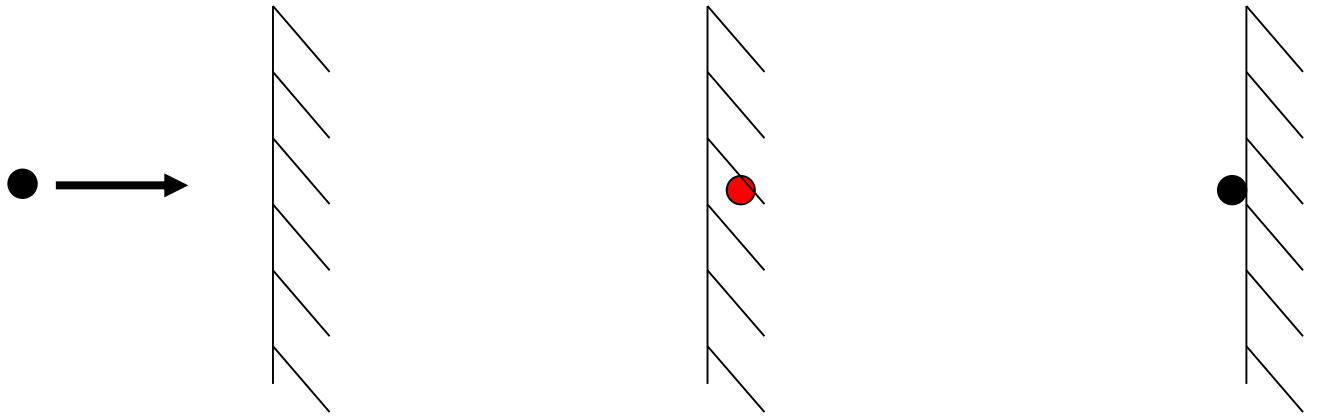
$$\mathbf{x}' = 2\mathbf{x} - \mathbf{x}^* + \mathbf{a}\Delta t^2 + O(\Delta t^4)$$

When $\mathbf{x}$ is projected in a valid position, it is like if the particle has a velocity such that it goes exactly in that valid position !

[Note that the velocity can be approximated as $\mathbf{v} = \frac{\mathbf{x} - \mathbf{x}^*}{\Delta t}$
So, changing $\mathbf{x}$ do means changing the velocity]

Collision Handling

- When a particle is in a not-valid position, the position of the particles *is projected* in the nearest valid position.



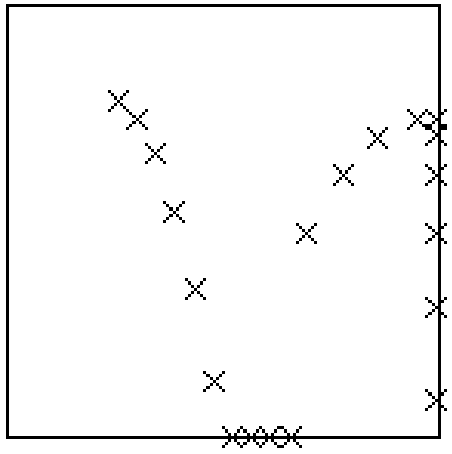
not valid

projected in a valid position

Collision handling: example

Particle constrained in the square (0, 0) - (1000, 1000)

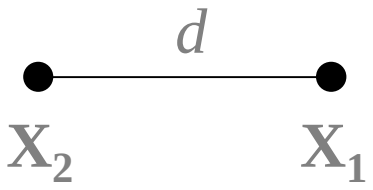
```
// for each particle
for(int i=0; i<NUM_PARTICLES; i++)
{
  vx = ( max(vx.x,    0), max(vx.y,    0) );
  vx = ( min(vx.x, 1000), min(vx.y, 1000) );
}
```



Equidistance constraint

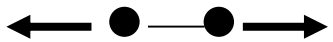
Let's define a rigid link among two particles, that is, the distance is constant:

$$\left|\mathbf{x}_2 - \mathbf{x}_1\right| = d$$

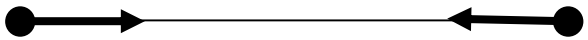


During the simulation, we can have two not-valid cases:

- particles too close each other:



- particles too far from each other:



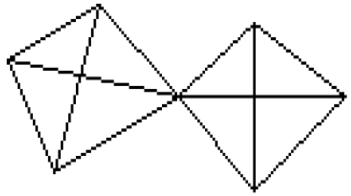
To satisfy the constraint, we simply project the particle in the valid position

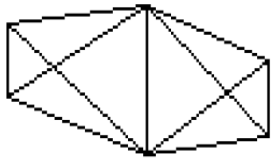
Equidistance pseudo code

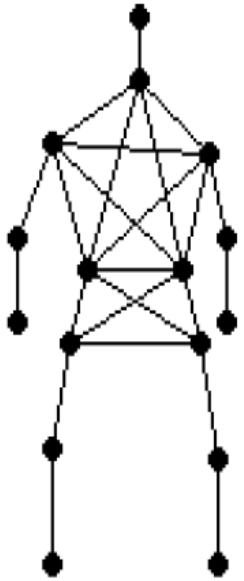
```
delta = x2 - x1;

deltalength = sqrt(delta * delta);

diff = (deltalength - restlength) / deltalength;

x1 += delta * 0.5 * diff;
x2 -= delta * 0.5 * diff;
```

Joints

- Combining together equidistance constraints, we obtain:
  - Joints
  - Rigid structures
  - Cloths, ragdolls, ...







Constraint' Satisfaction by Relaxation

- In general, there are many concurrent constraints;
- Solving a constraint can break another one
- Solving all the constraints at once involves the resolution of a (non-linear) equation system... Inefficient
- Solution:

Repeat N_ITERS times until all the constraints are satisfied (or the error is acceptable)

Constraint Satisfaction by Relaxation

- Relaxation method (or Jacobi or Gauss-Seidel)
- Precondition: a constraint does not contradict another constraint;
- The method converges towards a solution;
- Usually N_ITERS is between 1 and 10: extremely efficient.

Demo

References

- Witkin, A. and Baraff, D.
  Physically Based Modeling: Principles and Practice
  Siggraph ’97 course notes, 1997
  http://www.cs.cmu.edu/~baraff/sigcourse/index.html

- Jakobsen T.
  Advanced Character Physics. Proceedings
  Game Developer's Conference 2001, San Jose, 2001.

- Verlet, L.
  Computer experiments on classical fluids. I. Thermodynamical properties of Lennard-Jones molecules
  Phys. Rev., **159**, 98-103 (1967)