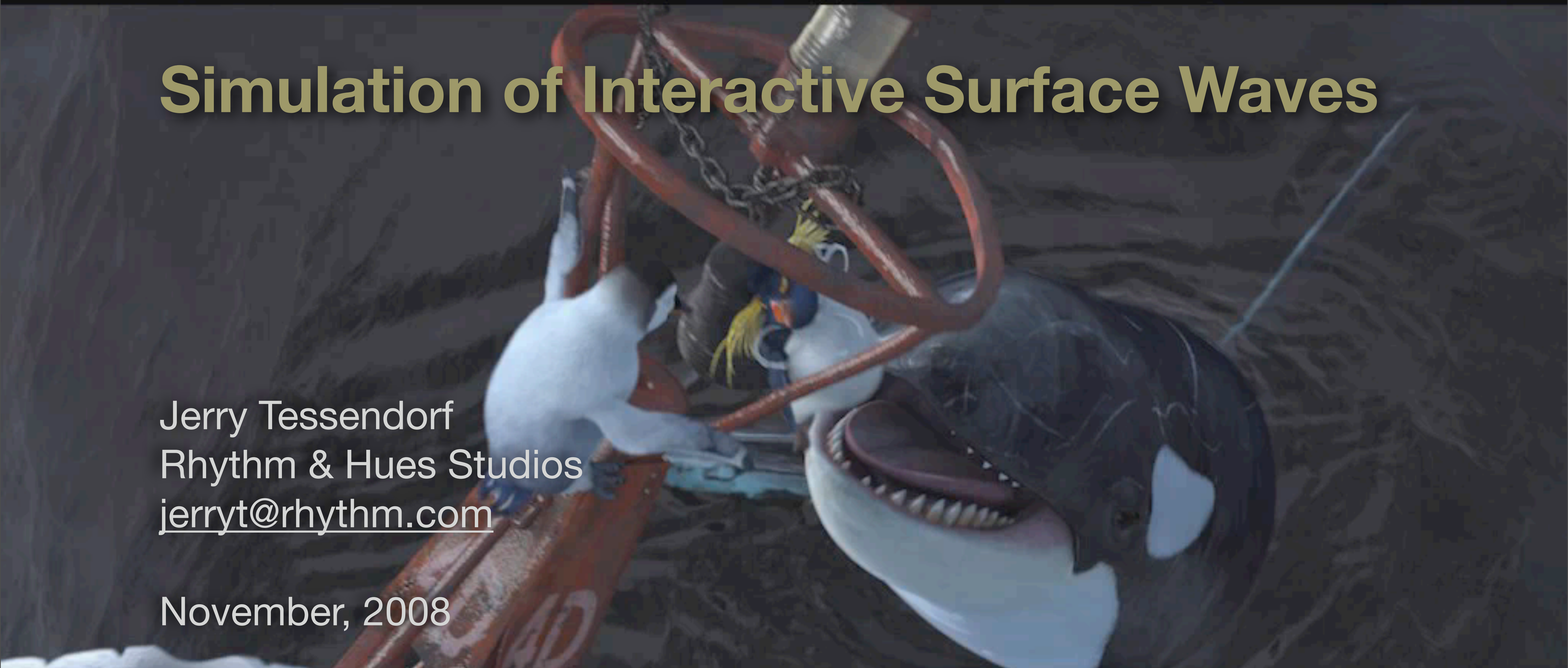
Simulation of Interactive Surface Waves



Jerry Tessendorf
Rhythm & Hues Studios
jerryt@rhythm.com

November, 2008

In Production

Outline for this week

Outline for this week

- Surface wave examples in real life and film - context
- Equations of Motion - the full theory
- Numerically Solving Dynamics Problems - the digestible way

Day 1

Outline for this week

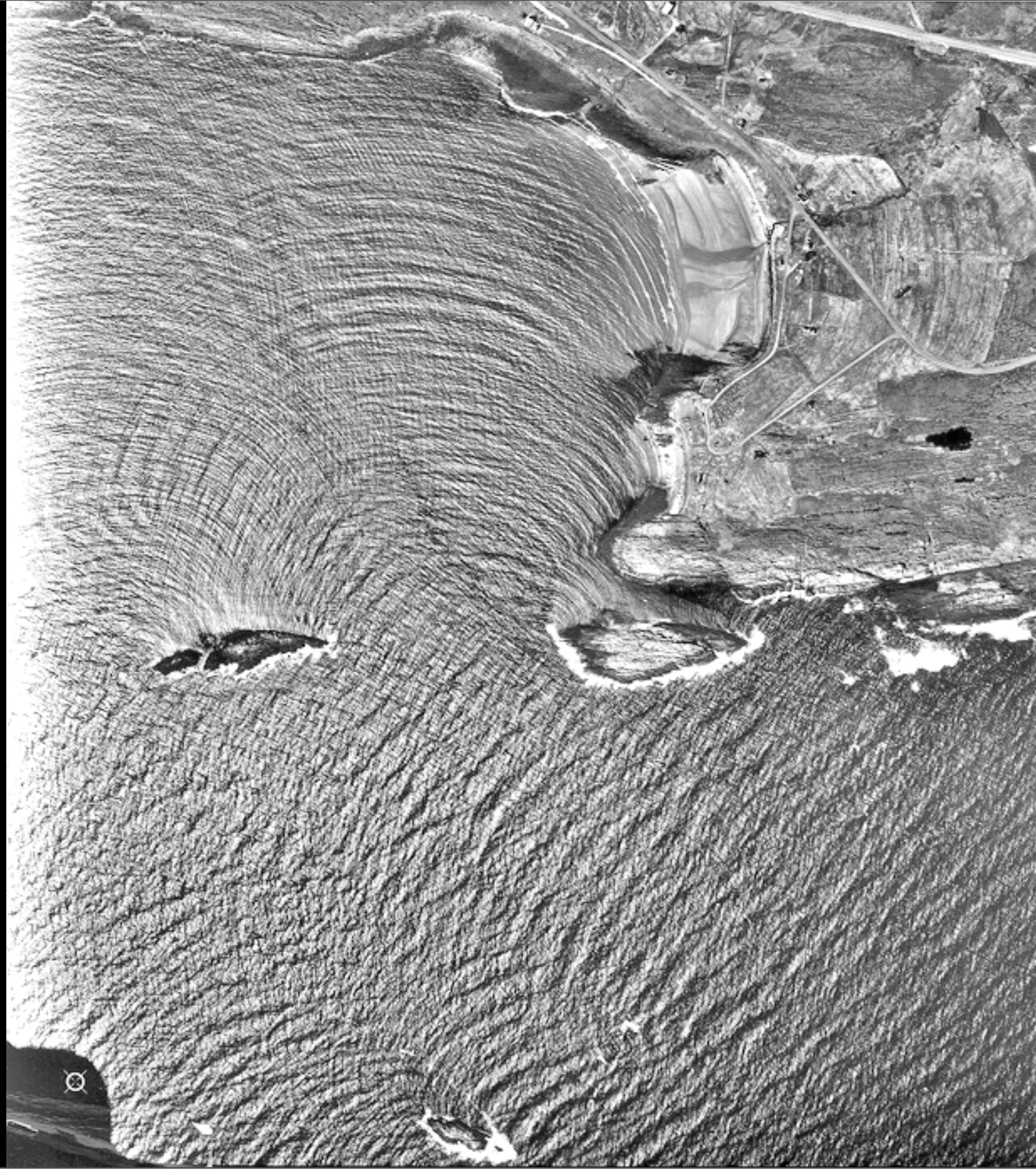
- Surface wave examples in real life and film - context
- Equations of Motion - the full theory

Day 1

- Numerically Solving Dynamics Problems - the digestible way

- Vertical Gradient of Velocity Potential from 2D Convolution
- Basic Obstacles

Day 2



Airborne
Remote
Optical
Sensing
System

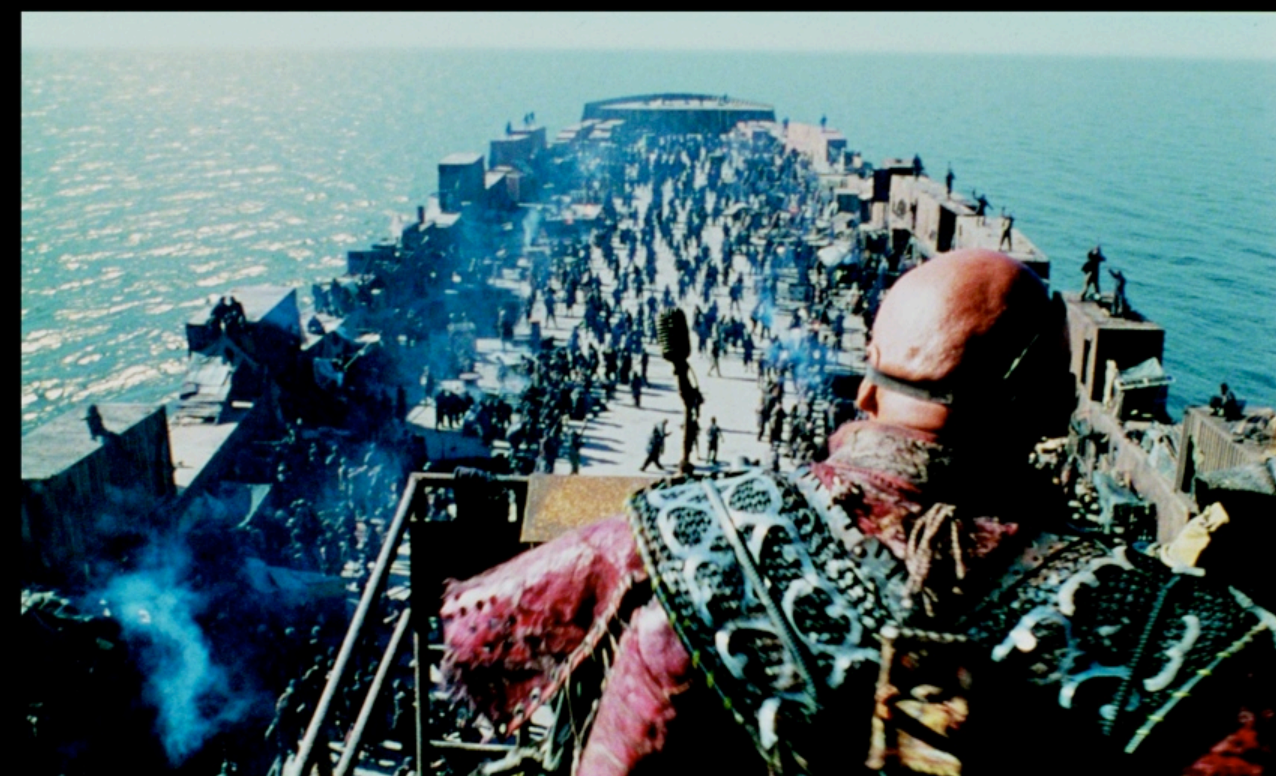
Renderworld (1994)

Sink The Bismark (1960)

Water waves do not scale well because of range of motions

CG Surface Waves in Film Visual Effects

Waterworld (1995)



Titanic (1997)

Titanic (1997)

Playstation 3 SDK realtime waves in 720p HD
Cuspy FFT Method on 3 Cell SPEs. 460 fps.

courtesy Caspar Sawyer, Sony

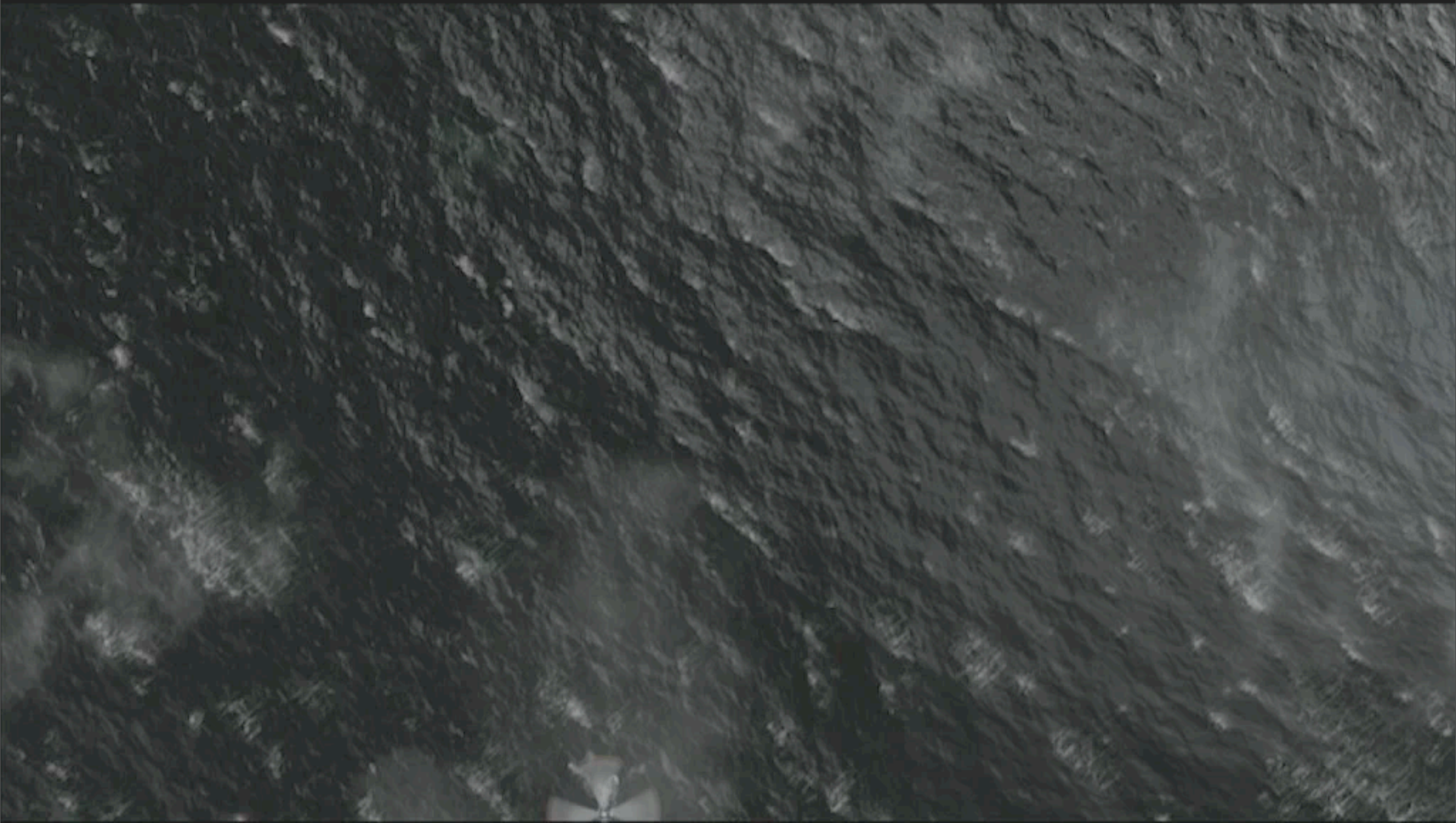
this movie has been converted from 720p HD for presentation purposes

Orange County (2002)

0111

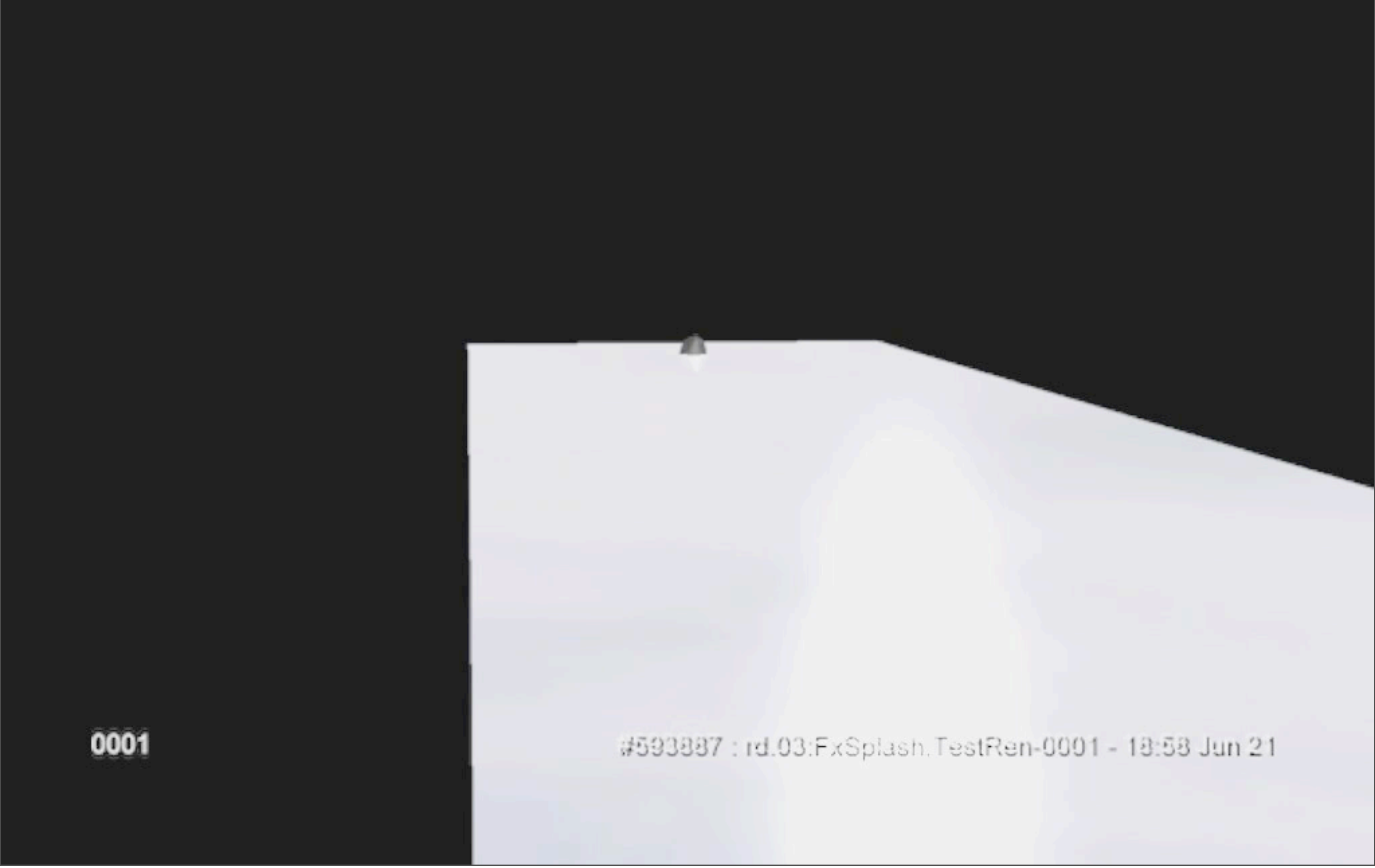

12:22:30:00
ORANGE COUNTY
489
VFX4C
4
22 fps
J. KASDAN
K. PETERMAN
5/4/01
11:13:13:13.2
004

Superman Returns (2006)

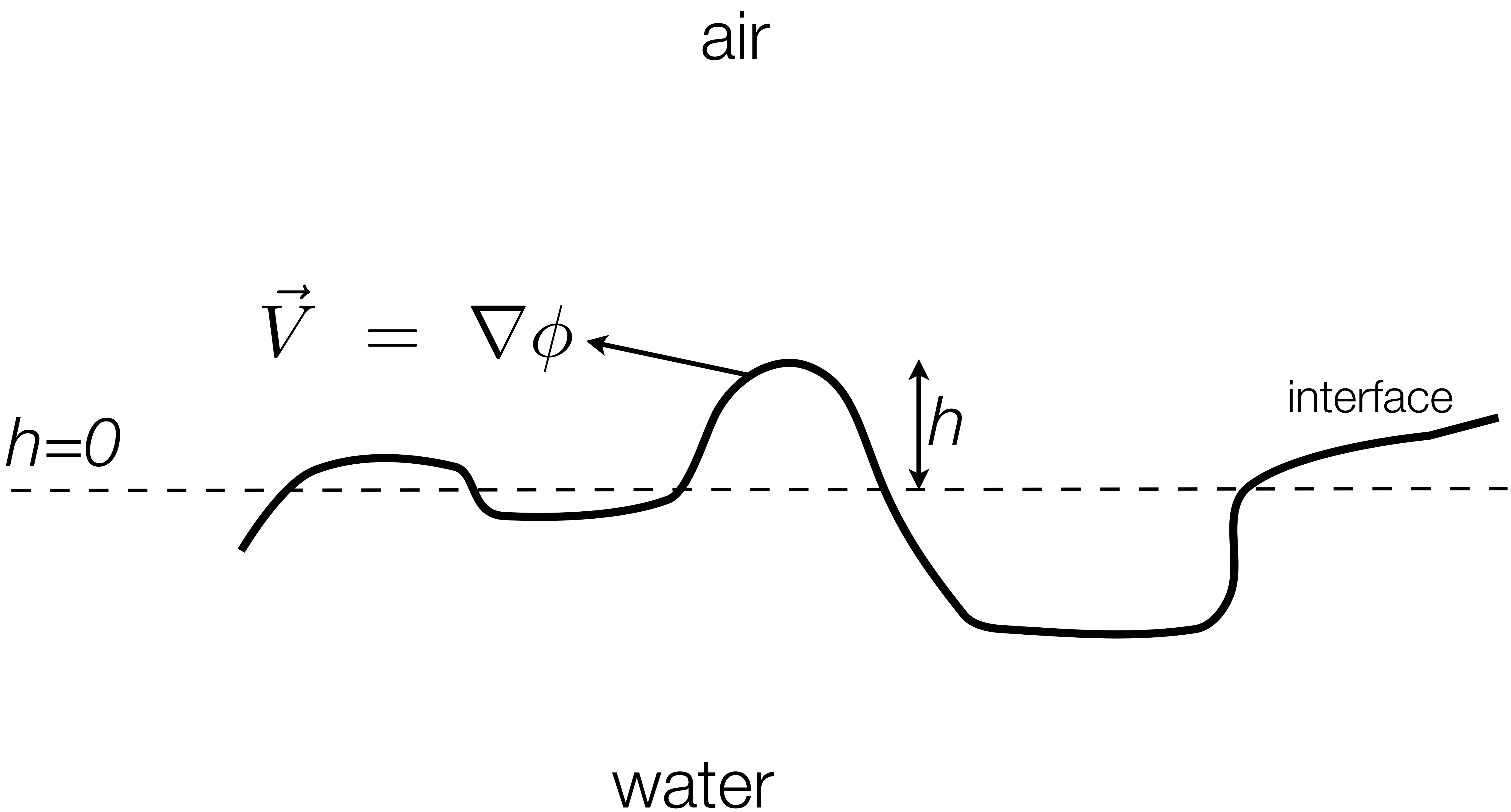


Surf's Up (2007)

Happy Feet production example



Happy Feet (2006)


air
$\vec{V} = \nabla\phi$
h=0
h
interface
water

Air-Water Interface

- Waves move up/down and propagate because of two factors
  - ★gravity
  - ★mass conservation
- Two quantities describe motion
  - ★height *h* of displacement from still surface
  - ★velocity potential ϕ, which acts like momentum

Strategy

- Fluid equations describe motion in a 3D volume
- We reduce the problem to motion of the surface using 2D
- We reduce the complexity to make the code fast, simple, but still useful
- Add some (non-physical) hacks to allow interaction with surface objects

Bernoulli's Equations

with free surface vertical displacement $h(\mathbf{x},t)$

Bernoulli's Equations

with free surface vertical displacement $h(\mathbf{x},t)$

$$\frac{\partial}{\partial t}\phi(\mathbf{x},t) + \frac{1}{2}\left|\nabla\phi\right|^2 = -gh(\mathbf{x},t)$$

Fluid Motion

Bernoulli's Equations

with free surface vertical displacement $h(\mathbf{x},t)$

$$\frac{\partial}{\partial t}\phi(\mathbf{x},t)+\frac{1}{2}\left|\nabla\phi\right|^2=-gh(\mathbf{x},t)$$

Fluid Motion

$$\left(\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial y^2}+\frac{\partial^2}{\partial z^2}\right)\phi=0$$

Mass Conservation

Bernoulli's Equations

with free surface vertical displacement $h(\mathbf{x},t)$

$$\frac{\partial}{\partial t}\phi(\mathbf{x},t) + \frac{1}{2}\left|\nabla\phi\right|^2 = -gh(\mathbf{x},t)$$

Fluid Motion

$$\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}\right)\phi = 0$$

Mass Conservation

$$\frac{\partial h}{\partial t} + \nabla\phi \cdot \nabla h = \frac{\partial \phi}{\partial y}$$

Free Surface Motion

Bernoulli's Equations

with free surface vertical displacement $h(\mathbf{x},t)$

$$\frac{\partial}{\partial t}\phi(\mathbf{x},t) \qquad = -gh(\mathbf{x},t)$$

Fluid Motion

$$\left(\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial y^2}+\frac{\partial^2}{\partial z^2}\right)\phi = 0$$

Mass Conservation

$$\frac{\partial h}{\partial t} \qquad = \frac{\partial \phi}{\partial y}$$

Free Surface Motion

Bernoulli's Equations

with free surface vertical displacement $h(\mathbf{x},t)$

$$\frac{\partial}{\partial t}\phi(\mathbf{x},t) = -gh(\mathbf{x},t)$$

Fluid Motion

L I N E A R

$$\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}\right)\phi = 0$$

Mass Conservation

$$\frac{\partial h}{\partial t} = \frac{\partial \phi}{\partial y}$$

Free Surface Motion

Linearized Bernoulli Equation
(Again)

$$\frac{\partial}{\partial t}\phi(\mathbf{x},t) = -gh(\mathbf{x},t)$$ Fluid Motion

$$\left(\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial y^2}+\frac{\partial^2}{\partial z^2}\right)\phi = 0$$ Mass Conservation

$$\frac{\partial h}{\partial t} = \frac{\partial \phi}{\partial y}$$ Free Surface Motion

$$\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}\right)\phi = 0$$ Mass Conservation

Rearrange terms

$$\frac{\partial^2}{\partial y^2}\phi = -\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial z^2}\right)\phi$$

Take square root

$$\frac{\partial}{\partial y}\phi = \sqrt{-\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial z^2}\right)\phi}$$

Take square root

$$\frac{\partial}{\partial y}\phi = \sqrt{-\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial z^2}\right)\phi}$$

- Evaluate as a 2D spatial convolution

Take square root

$$\frac{\partial}{\partial y}\phi = \sqrt{-\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial z^2}\right)\phi}$$

- Evaluate as a 2D spatial convolution
- 13x13 moving window is the smallest with reasonable quality

Mass Conserving Surface Equations

$$\frac{\partial}{\partial t}\phi = -gh$$

$$\frac{\partial h}{\partial t} = \left(\sqrt{-\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial z^2}\right)}\right) \quad \phi$$

Mass Conserving Surface Equations

$$\frac{\partial}{\partial t}\phi = -gh$$

$$\frac{\partial h}{\partial t} = \left(\sqrt{-\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial z^2}\right)}\right) \phi$$

- Automatically conserve mass

Mass Conserving Surface Equations

$$\frac{\partial}{\partial t}\phi = -gh$$

$$\frac{\partial h}{\partial t} = \left(\sqrt{-\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial z^2}\right)}\right) \phi$$

- Automatically conserve mass
- Dynamics confined to fast surface 2D calculations

Mass Conserving Surface Equations

$$\frac{\partial}{\partial t}\phi = -gh$$

$$\frac{\partial h}{\partial t} = \left(\sqrt{-\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial z^2}\right)}\right) \phi$$

- Automatically conserve mass
- Dynamics confined to fast surface 2D calculations
- Moving window or FFT convolution can be used

Mass Conserving Surface Equations

$$\frac{\partial}{\partial t}\phi = -gh$$

$$\frac{\partial h}{\partial t} = \left(\sqrt{-\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial z^2}\right)}\right) \phi$$

- Automatically conserve mass
- Dynamics confined to fast surface 2D calculations
- Moving window or FFT convolution can be used
- Classic leapfrog & verlet solvers are stable

OpenGL Interactive Wave (IWave) Example

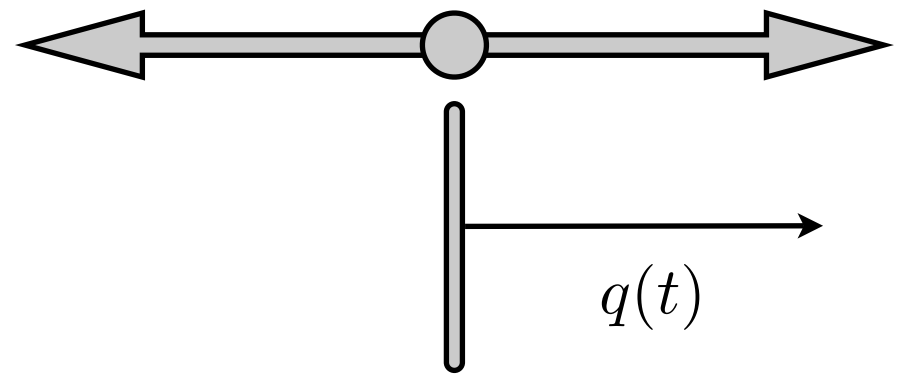
Mass Conserving Surface Equations

$$\frac{\partial}{\partial t}\phi = -gh$$

$$\frac{\partial h}{\partial t} = \left(\sqrt{-\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial z^2}\right)}\right) \phi$$

- Automatically conserve mass
- Dynamics confined to fast surface 2D calculations
- Moving window or FFT convolution can be used
- Classic leapfrog & verlet solvers are stable

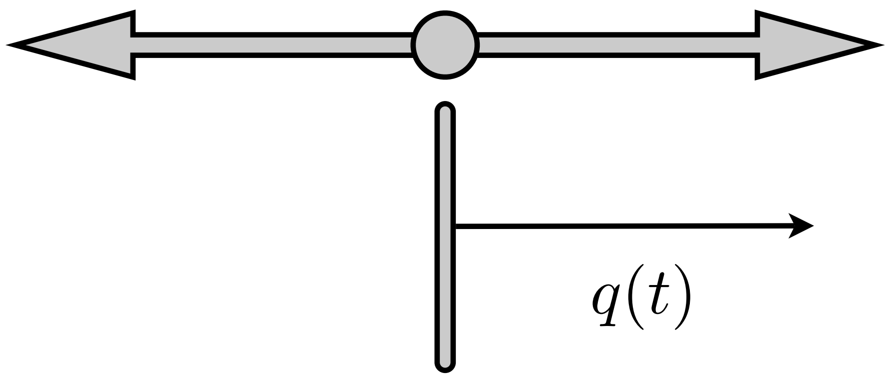
Harmonic Oscillator



Harmonic Oscillator

- Force opposes displacement $q(t)$

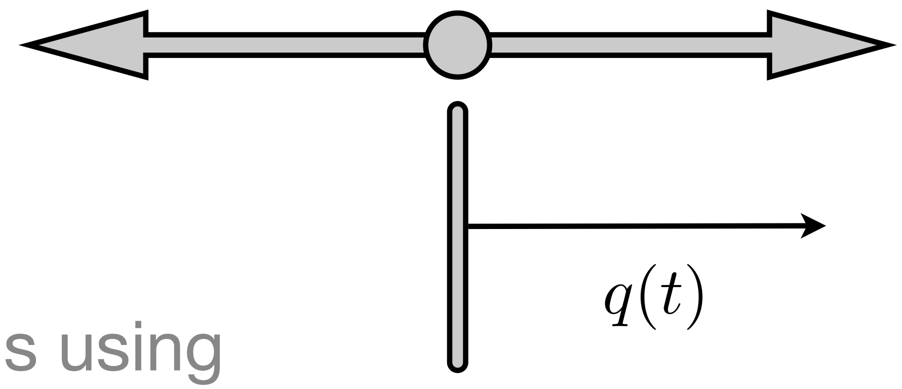
$$m \frac{d^2q(t)}{dt^2} = -k\ q(t)$$



Harmonic Oscillator

- Force opposes displacement $q(t)$

$$m \; \frac{d^2 q(t)}{dt^2} = -k \; q(t)$$



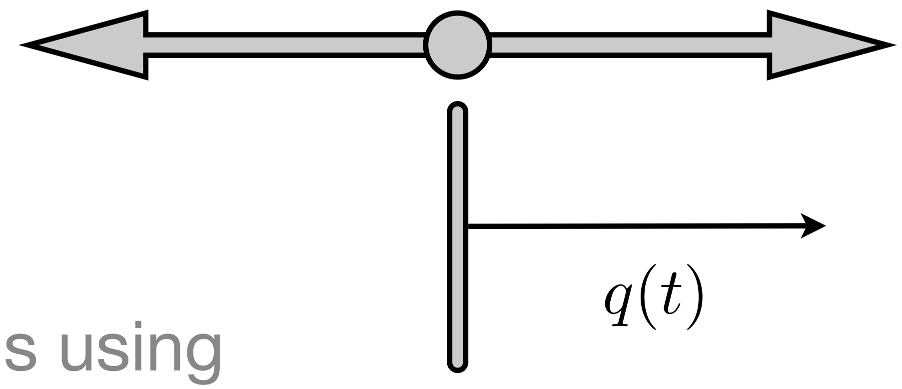
- Split into two first order differential equations using momentum

$$\begin{aligned} \dot{q}(t) &= p(t) \\ m \; \dot{p}(t) &= -k \; q(t) \end{aligned}$$

Harmonic Oscillator

- Force opposes displacement $q(t)$

$$m\ \frac{d^2q(t)}{dt^2} = -k\ q(t)$$



- Split into two first order differential equations using momentum

$$\begin{aligned}\dot{q}(t) &= p(t)\\ m\ \dot{p}(t) &= -k\ q(t)\end{aligned}$$

- This has an exact solution

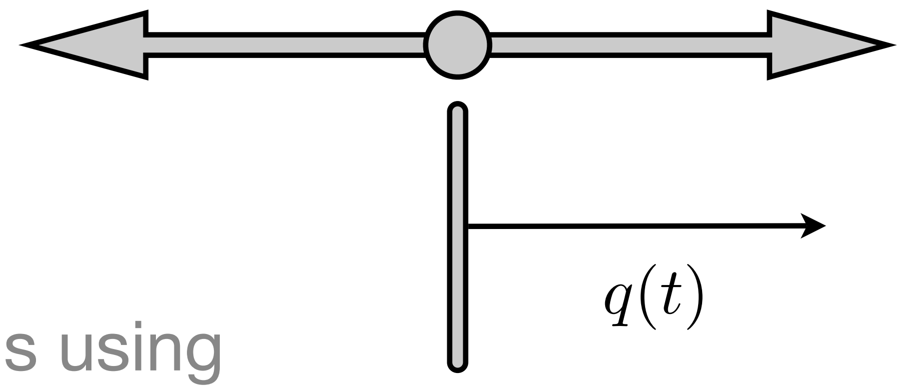
$$\begin{aligned}q(t) &= q(0)\ \cos(\omega t)\ +\ \frac{p(0)}{\omega}\ \sin(\omega t)\\ p(t) &= p(0)\ \cos(\omega t)\ -\ \omega q(0)\ \sin(\omega t)\end{aligned}$$

$$\omega = \sqrt{\frac{k}{m}}$$

Harmonic Oscillator

- Force opposes displacement $q(t)$

$$m\ \frac{d^2q(t)}{dt^2} = -k\ q(t)$$



- Split into two first order differential equations using momentum

$$\begin{aligned} \dot{q}(t) &= p(t) \\ m\ \dot{p}(t) &= -k\ q(t) \end{aligned}$$

- This has an exact solution

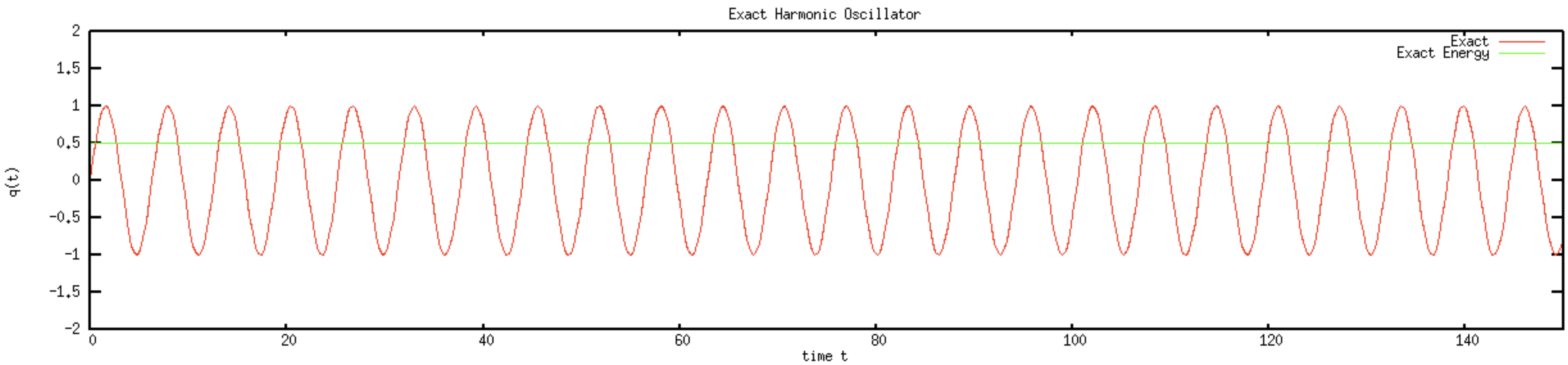
$$\begin{aligned} q(t) &= q(0)\ \cos(\omega t)\ +\ \frac{p(0)}{\omega}\ \sin(\omega t) \\ p(t) &= p(0)\ \cos(\omega t)\ -\ \omega q(0)\ \sin(\omega t) \end{aligned}$$

$$\omega = \sqrt{\frac{k}{m}}$$

- Energy is constant over time

$$E = \frac{p^2(t)}{2m} + \frac{k}{2}q^2(t) = \frac{p^2(0)}{2m} + \frac{k}{2}q^2(0)$$

Harmonic Oscillator Visual



How Do We Numerically Solve HO?

- Have to make the problem discrete in time.
- There are many ways to do that
- Different methods have dramatically different properties
- Here we illustrate four methods and their consequences
  - ★Forward Euler
  - ★Leapfrog
  - ★Implicit
  - ★Exponential
- We will actually use only Leapfrog in this course

Forward Euler Solution

- Discrete time steps Δt
- Derivative becomes finite difference

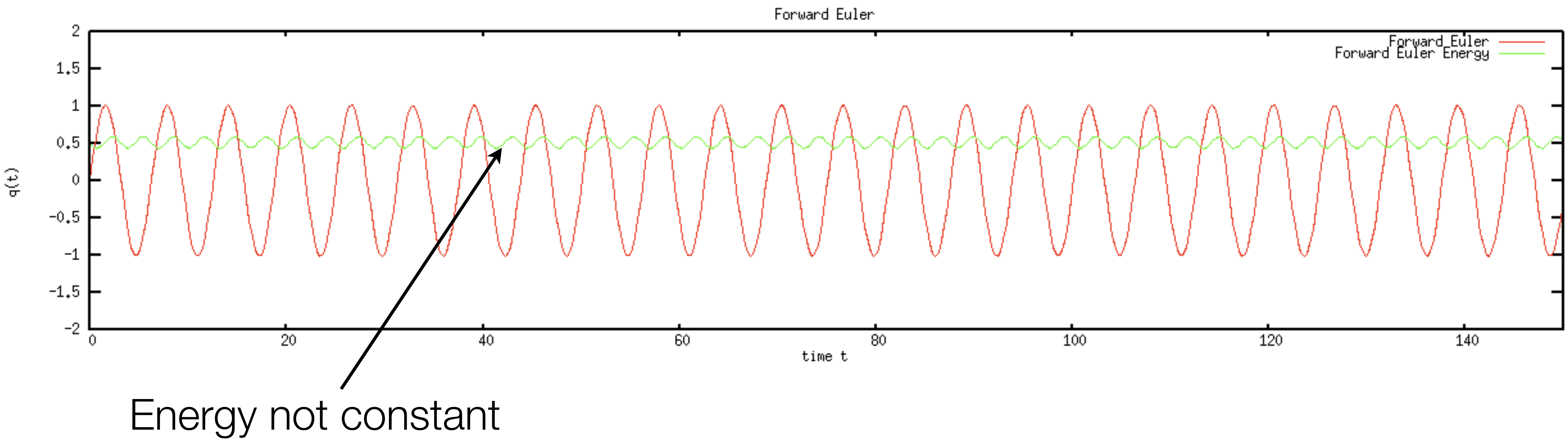
$$\dot{q}(t) = \frac{q(t+\Delta t) \; - \; q(t)}{\Delta t} \qquad \dot{p}(t) = \frac{p(t+\Delta t) \; - \; p(t)}{\Delta t}$$

- Evaluate force at the latest time available

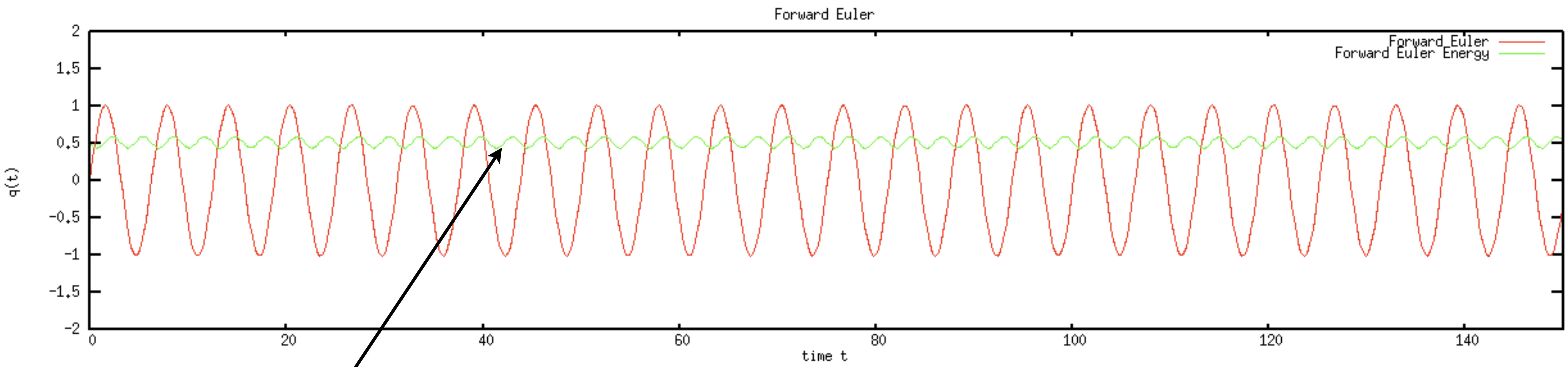
$$\begin{aligned} q(t+\Delta t) &= q(t) + \Delta t \; p(t) \\ p(t+\Delta t) &= p(t) - \omega^2 \Delta t \; q(t+\Delta t) \end{aligned}$$

- Conveniently allows update of q,p in-place.

Forward Euler Visualization

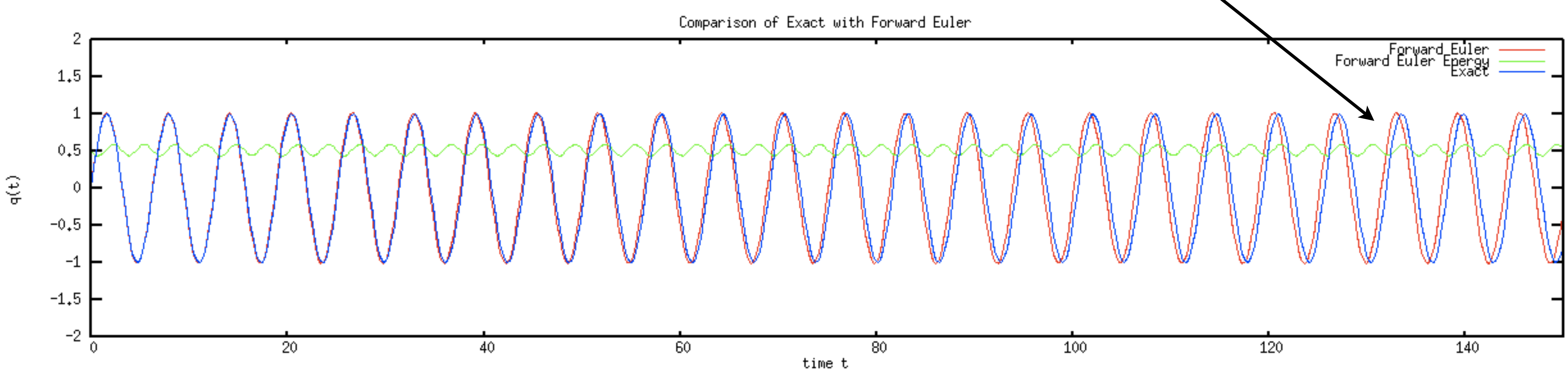


Forward Euler Visualization



Energy not constant

Phase shifting over time

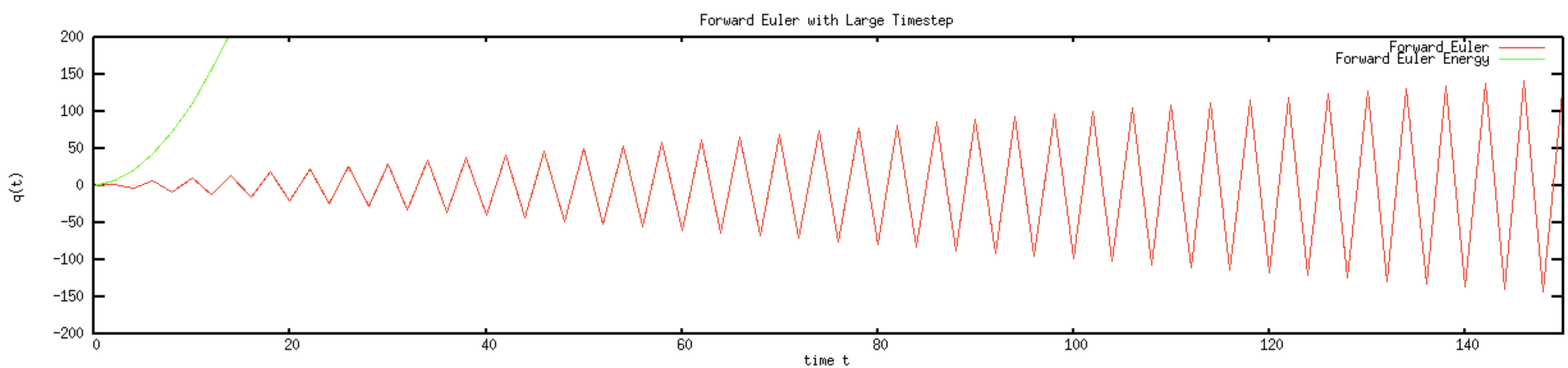


Errors in Forward Euler

- Actual frequency is related to physical frequency and timestep

$$\omega_{actual} = \frac{1}{\Delta t}\tan^{-1}\left\{\omega\sqrt{\frac{\Delta t^2}{1-\omega^2\Delta t^2/2}}\right\}$$

- For large timestep $\omega\Delta t > \sqrt{2}$ frequency becomes complex and solution explodes



Leapfrog Solution

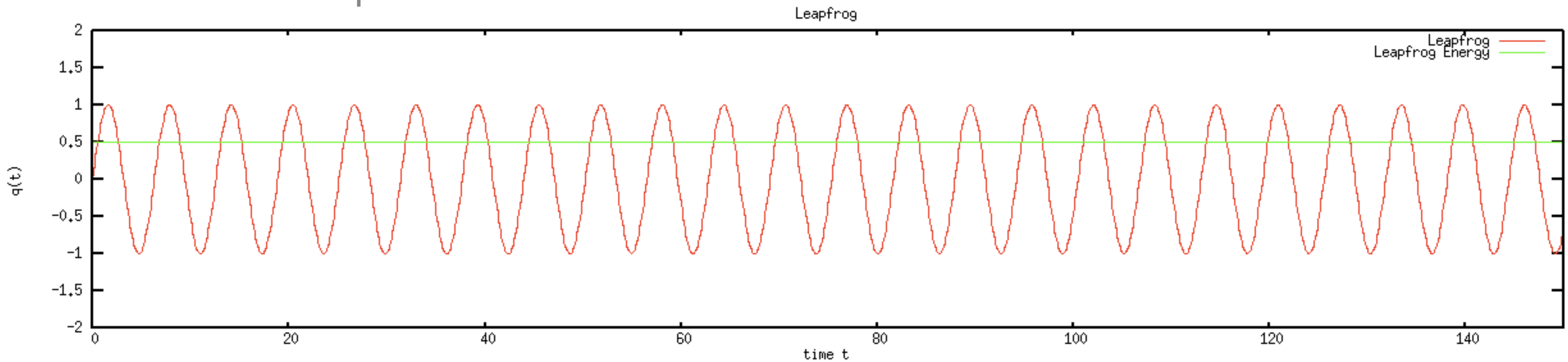
- Evaluate force at midpoint of time step

$$
\begin{aligned}
q_* &= q(t) + \frac{\Delta t}{2}\, p(t) \\
p(t+\Delta t) &= p(t) - \omega^2 \Delta t\; q_* \\
q(t+\Delta t) &= q_* + \frac{\Delta t}{2}\, p(t+\Delta t)
\end{aligned}
$$

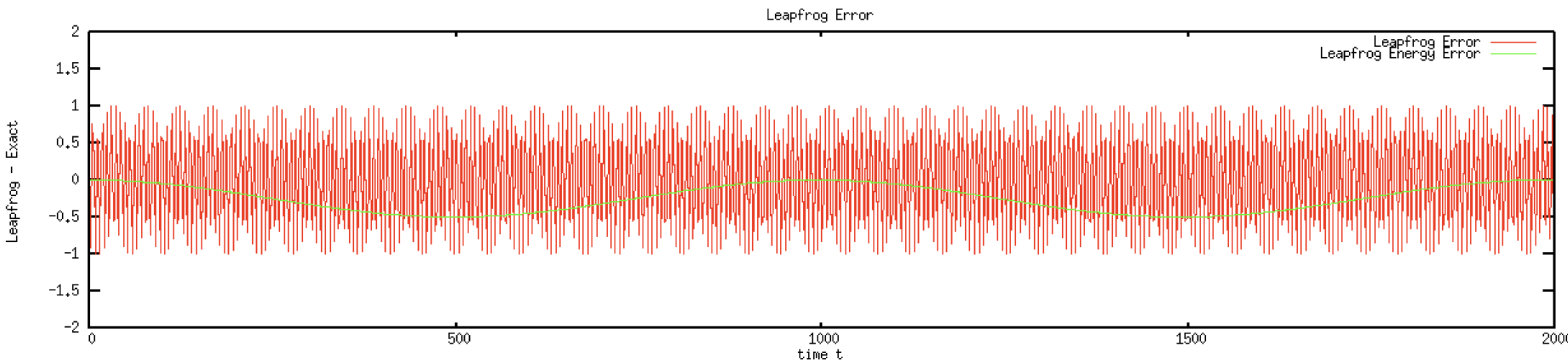
- This small change is much more stable
- Simplest member of a class of solutions called Geometric Integration
  - ★Error in solution fluctuates, but does not grow unbounded.

Leapfrog Solution Visualization

- No substantial phase shift



- Errors are bounded even for large time steps



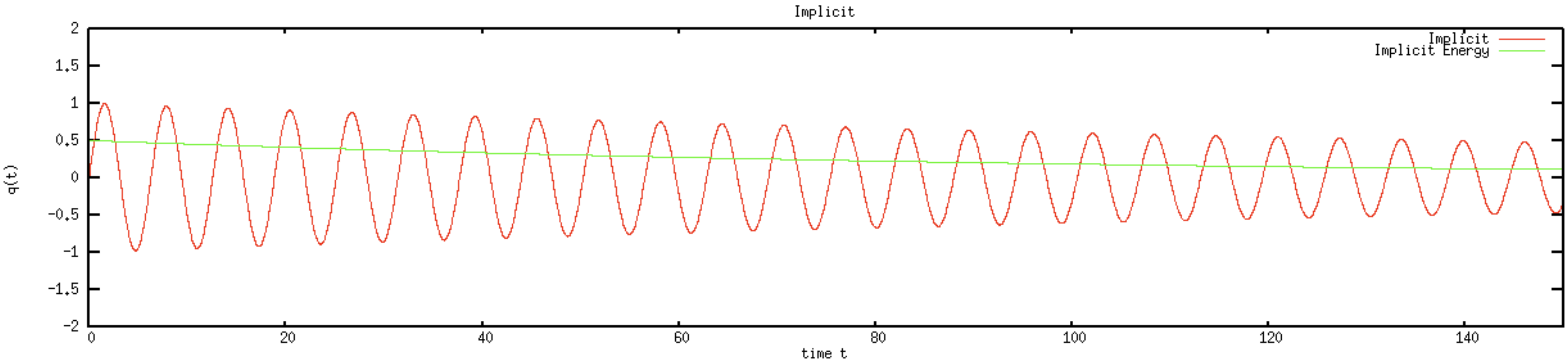
Implicit Solution

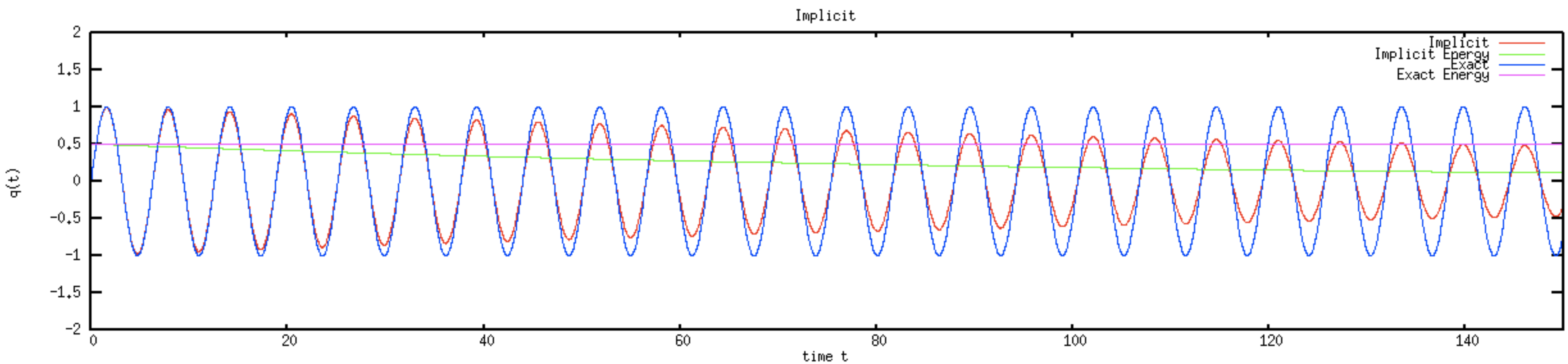
- Evaluate Force and momentum at end of time step

$$
\begin{aligned}
q(t+\Delta t) &= q(t) + \Delta t \; p(t+\Delta t) \\
p(t+\Delta t) &= p(t) - \omega^2 \Delta t \; q(t+\Delta t)
\end{aligned}
$$

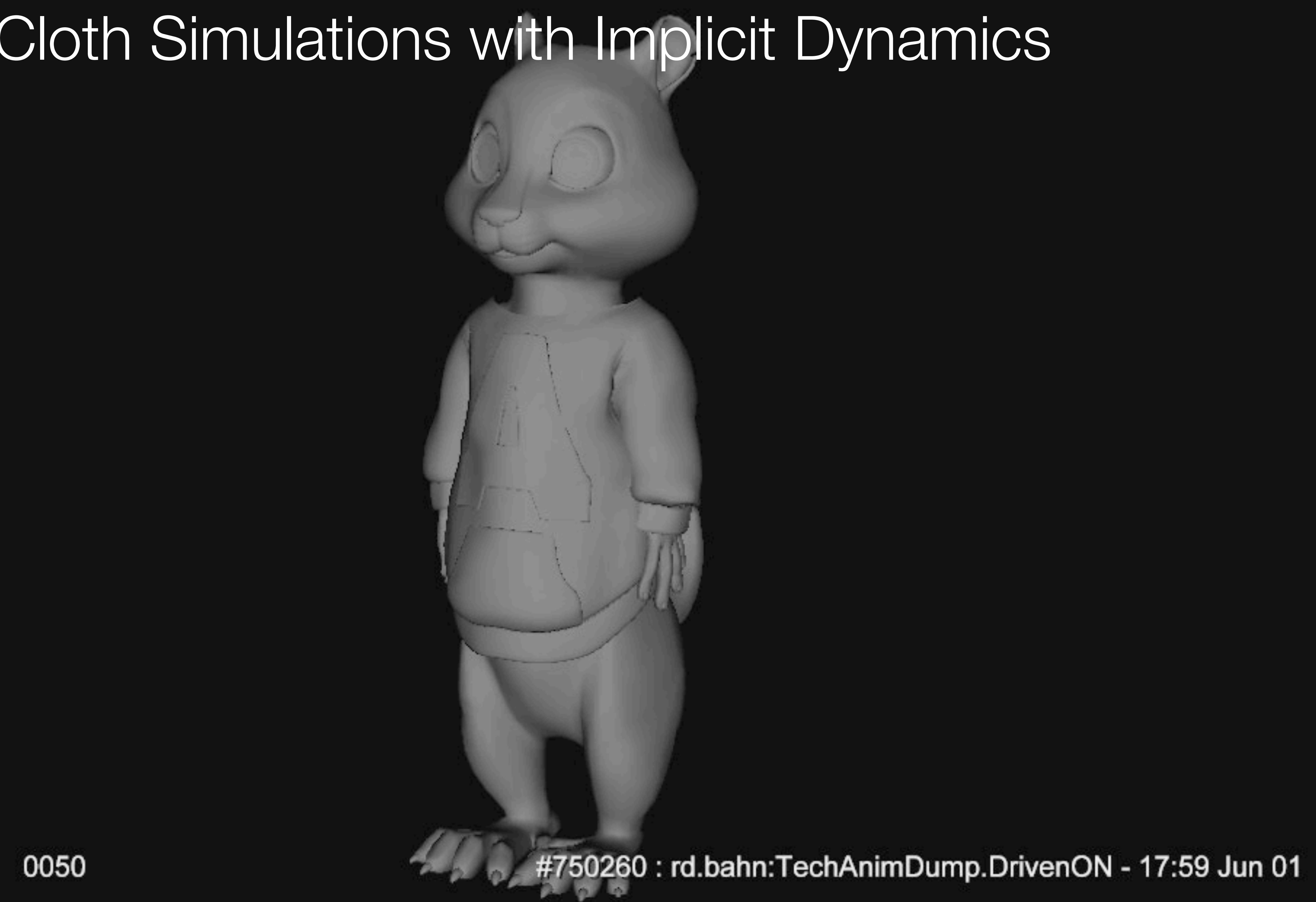
- Generally stable because it dampens the solution
- Dampening reduces accuracy
- In productions, the dampening is sometimes useful because explosions are never desirable.

Implicit Solution Visualization

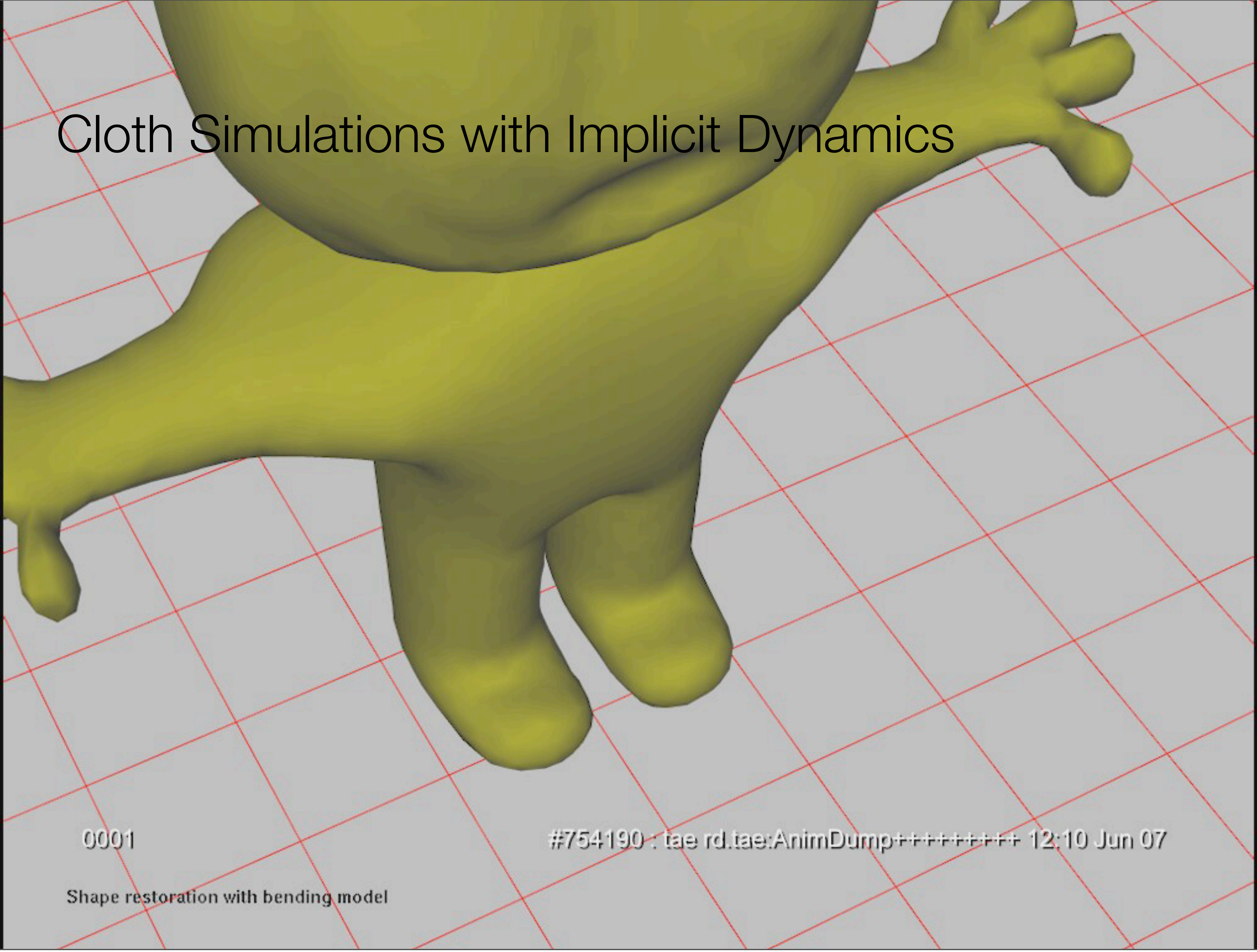




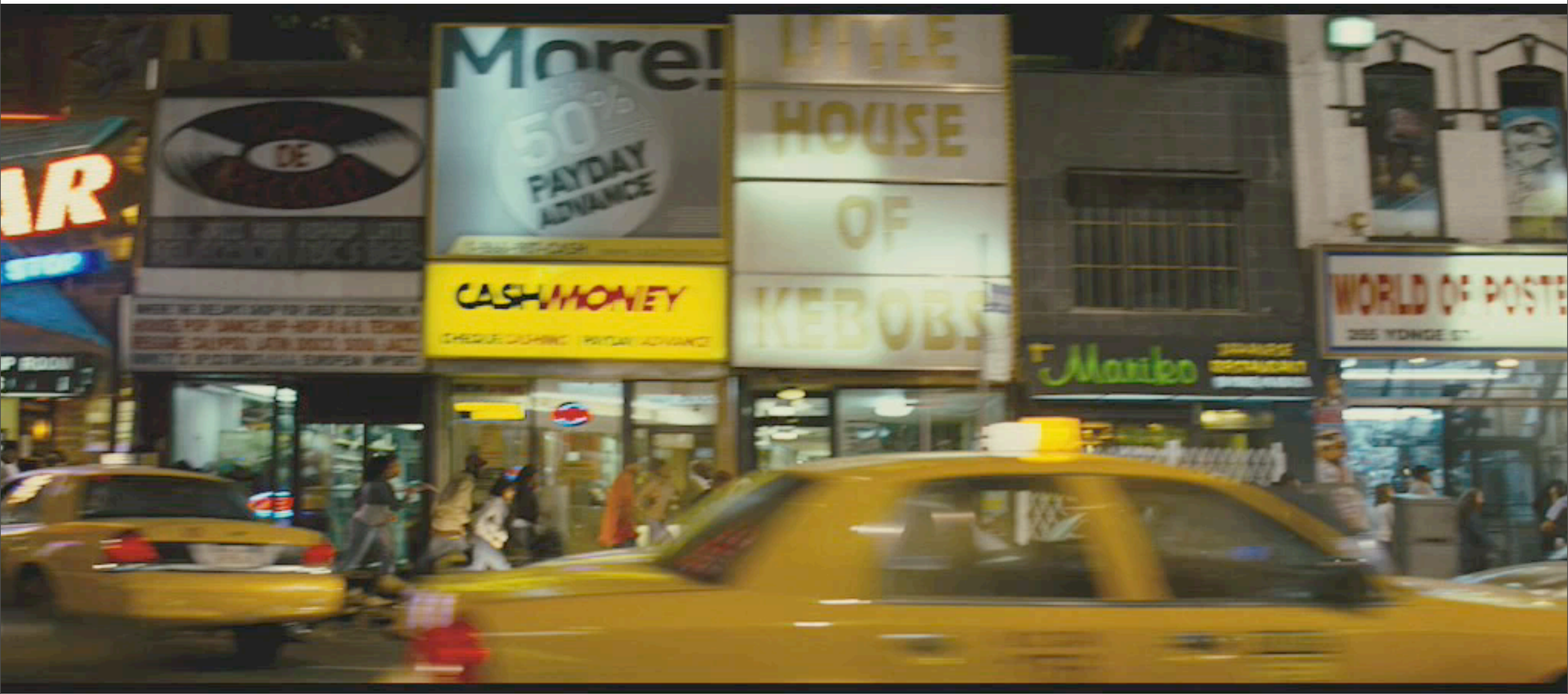
Cloth Simulations with Implicit Dynamics



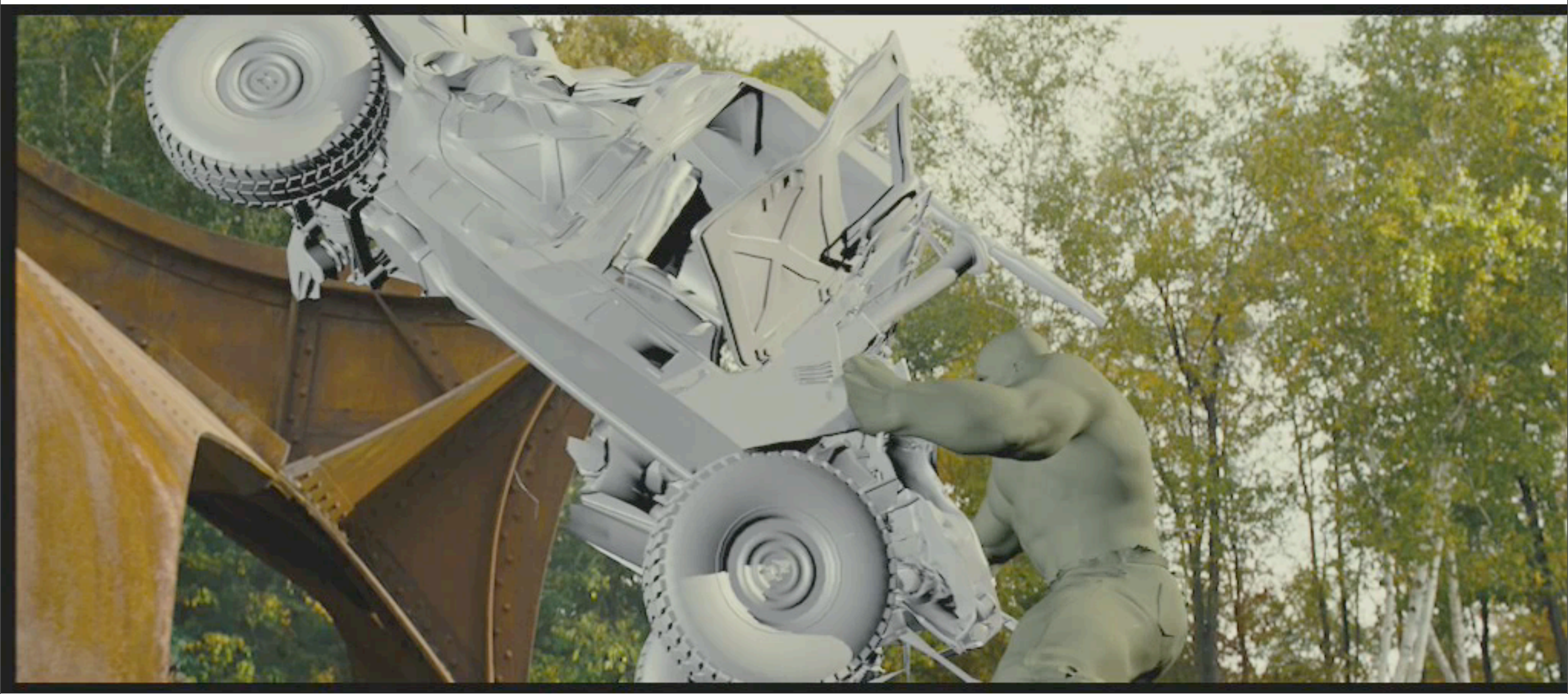
Cloth Simulations with Implicit Dynamics



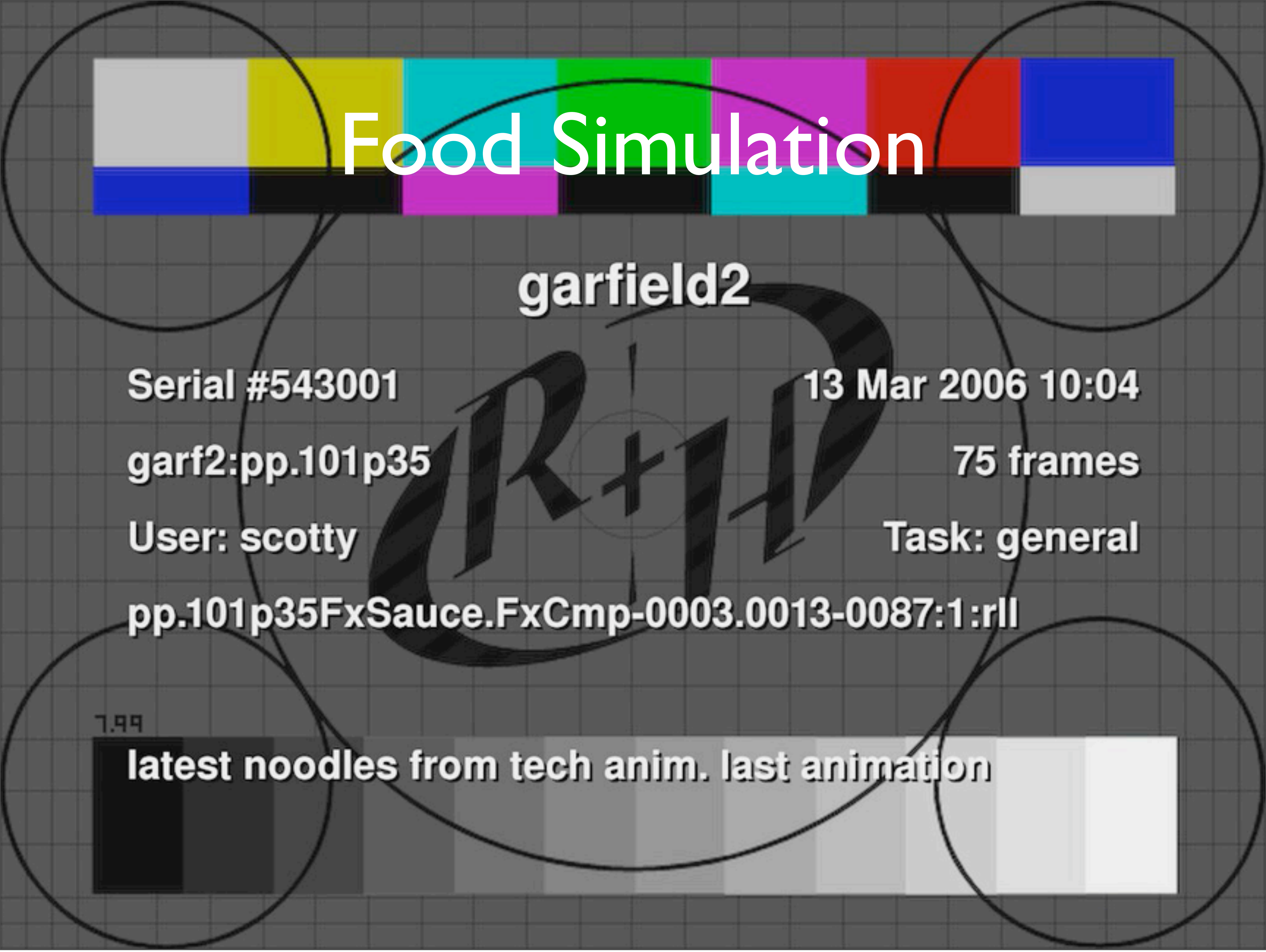
Cloth Simulations with Implicit Dynamics



Cloth Simulations with Implicit Dynamics
Rigid Body Sims with Forward Euler Dynamics



Food Simulation



Exponential Solution

- Frame dynamical equation as a linear equation over short lengths of time.
- State vector $Y(t) = \begin{pmatrix} q(t) \\ p(t) \end{pmatrix}$
- Equation of motion for Harmonic Oscillator state vector

$$\frac{dY(t)}{dt} = \begin{pmatrix} 0 & 1 \\ -\omega^2 & 0 \end{pmatrix} Y(t)$$

- Solution is in terms of exponentiated matrix

$$Y(t) = e^{Mt}\, Y(0)$$

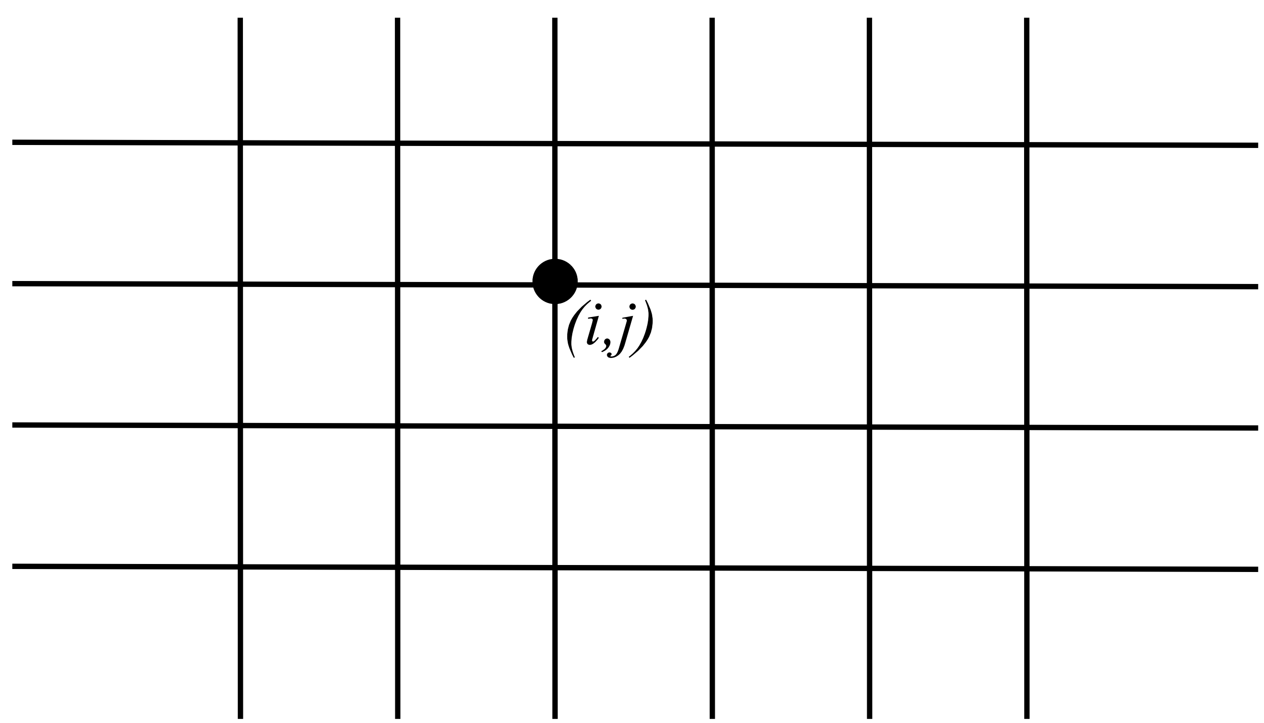
$$M = \begin{pmatrix} 0 & 1 \\ -\omega^2 & 0 \end{pmatrix}$$

Relative Merits of the Numeric Solution Schemes

	Pros	Cons
Forward Euler	*very simple implementation	*errors grow unbounded *unstable at large steps
Leapfrog	*simple implementation *errors do not grow *part of a larger system	*unstable at very large steps
Implicit	*almost always stable	*complex implementation *numerical dampening
Exponential	*Exact for linear systems *Simpler than implicit *No dampening	*complex implementation

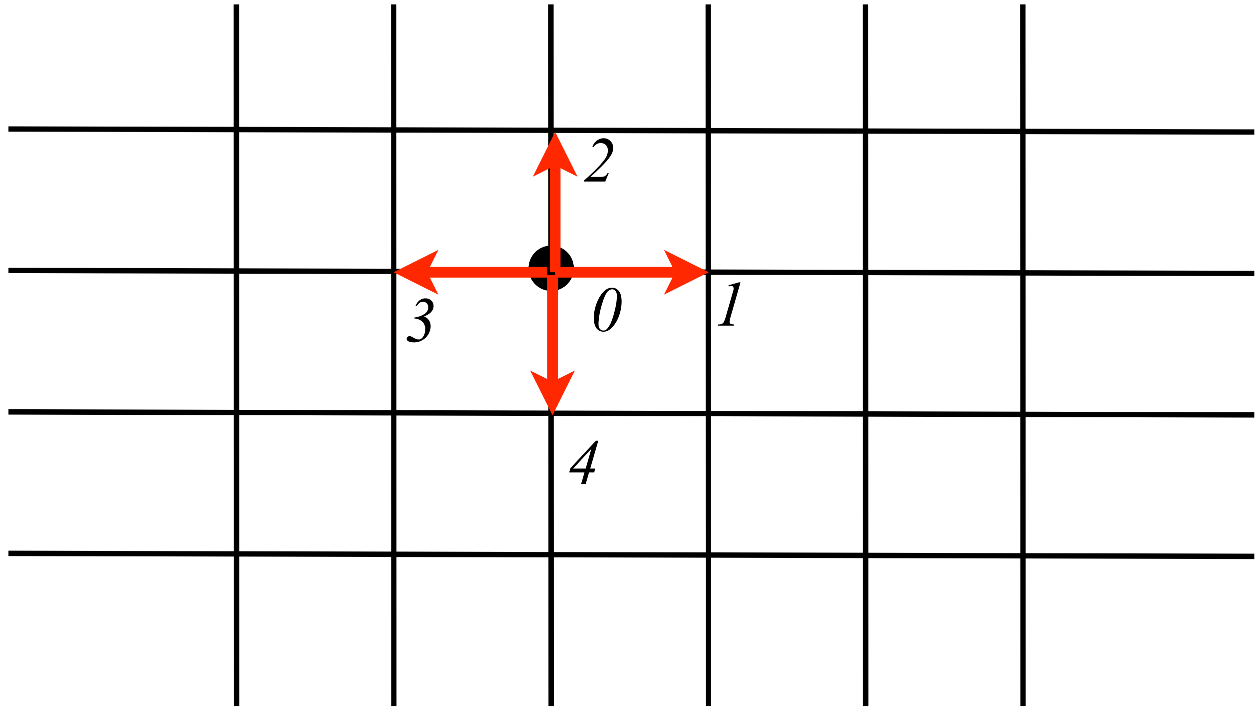
Grid of Harmonic Oscillators

- 2D grid of points.
- Vertices only move up/down (out of page) with displacement $h_{ij}(t)$
- Springs can be connected between vertices



Nearest Neighbor Springs

- Only connect to immediate neighbors on grid
- Each grid point is assigned a weight w_{ij}
- Force on vertex 0:



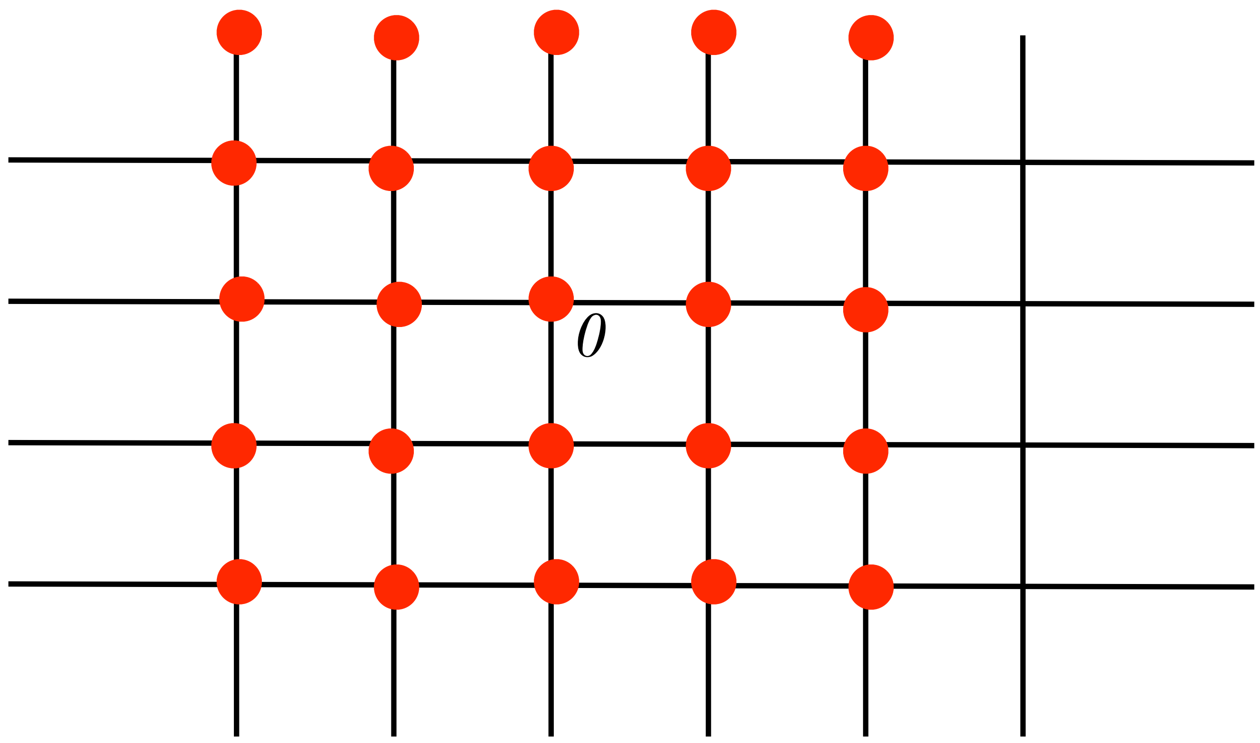
$$F_0 = w_0\, h_0(t) + w_1\, h_1(t) + w_2\, h_2(t) + w_3\, h_3(t) + w_4\, h_4(t)$$

- Example: Laplace weights

$$w_0 = -4$$
$$w_1 = w_2 = w_3 = w_4 = 1$$

Bigger Neighborhoods

- (2n+1)X(2n+1) windows centered on each point
- The force at the center is the weighted sum of the displacements of each point in the window
- Weights are chosen from the physical problem



5X5

Surfaces Waves as a Grid of Harmonic Oscillators

- Grid of points represents water surface
- Displacement $h_{ij}(t)$ is the vertical displacement of the surface.
- The momentum is called the velocity potential $\phi_{ij}(t)$
- Equations of motion are a little different from harmonic oscillators
- Equations of motion are simplified version of Bernoulli's equation

Mass Conserving Surface Equations

$$\frac{\partial}{\partial t}\phi = -gh$$

$$\frac{\partial h}{\partial t} = \left(\sqrt{-\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial z^2}\right)}\right) \phi$$

- Automatically conserve mass
- Dynamics confined to fast surface 2D calculations
- Moving window or FFT convolution can be used
- Classic leapfrog & verlet solvers are stable

Momentum Equation for Surface Waves

- Momentum changes as gravity (g) pushes waves up and down.

$$\frac{d\phi_{ij}(t)}{dt} = -g\ h_{ij}(t) \quad + S_{ij}(t)$$

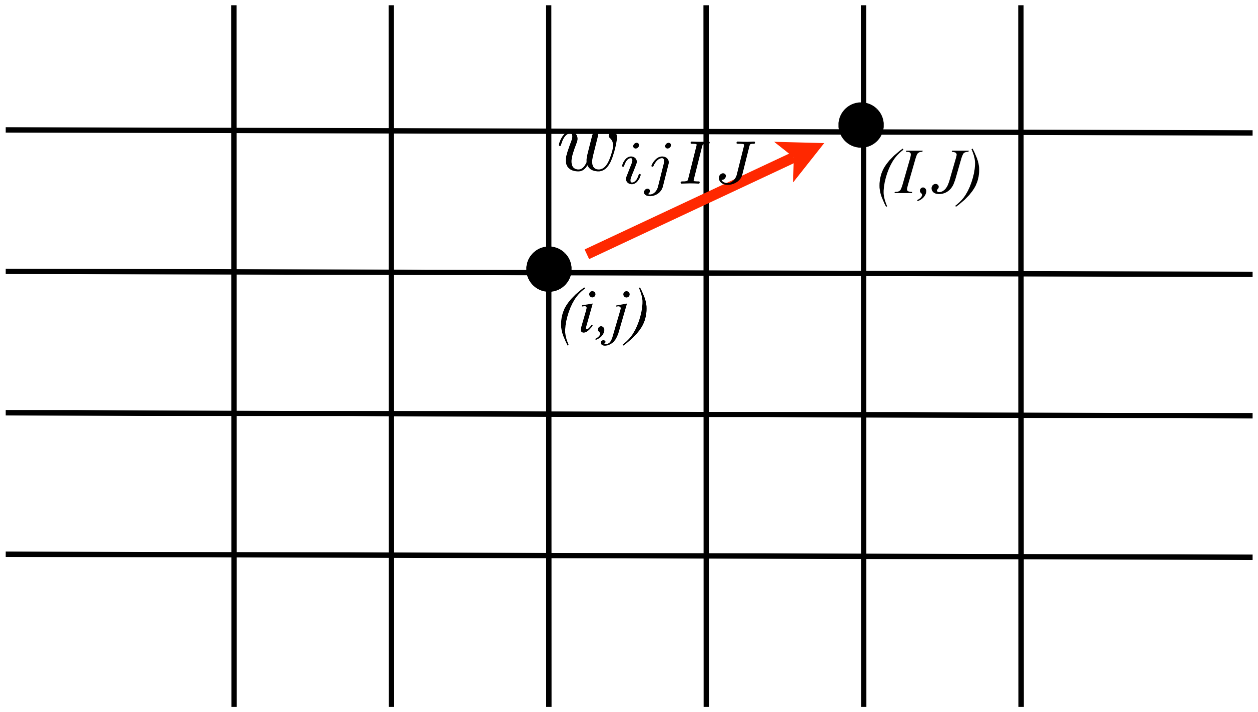
- Add a source function $S(t)$ to drive the motion

Height Displacement Equation

- This equation does two things:
  1. Pushes waves up/down due to momentum
  2. Enforces mass conservation by the choice of weights w_{ijIJ} that connect vertex ij to vertex IJ

$$\frac{dh_{ij}(t)}{dt} = \sum_{IJ} w_{ijIJ}\ \phi_{IJ}(t) \ + T_{ij}(t)$$

- Add source *T(t)* to drive the motion



Finite Time Steps

- Two linear equations

$$\frac{dh_{ij}(t)}{dt} = \sum_{IJ} w_{ijIJ}\ \phi_{IJ}(t) \quad + T_{ij}(t)$$

$$\frac{d\phi_{ij}(t)}{dt} = -g\ h_{ij}(t) \quad + S_{ij}(t)$$

- Time step Δt

$$\frac{dh_{ij}(t)}{dt} = \frac{h_{ij}(t+\Delta t) - h_{ij}(t)}{\Delta t}$$

$$\frac{d\phi_{ij}(t)}{dt} = \frac{\phi_{ij}(t+\Delta t) - \phi_{ij}(t)}{\Delta t}$$

Leapfrog Equations

- Update momentum at half time step:

$$\phi^*_{ij} \quad = \quad \phi_{ij}(t) \; - g \; \frac{\Delta t}{2} \; h_{ij}(t) \; + \; \frac{\Delta t}{2} S(t)$$

- Update height a full time step

$$h_{ij}(t+\Delta t) \quad = \quad h_{ij}(t) + \Delta t \; \sum_{IJ} \; w_{ijIJ} \; \phi^*_{IJ} \quad + \; \Delta t \; T_{ij}(t+\Delta t)$$

- Update momentum remaining half time step

$$\phi_{ij}(t+\Delta t) \quad = \quad \phi^*_{ij} \; - \; g \; \frac{\Delta t}{2} \; h_{ij}(t+\Delta t) \; + \; \frac{\Delta t}{2} S(t+\Delta t)$$

Leapfrog Equations

- Update momentum at half time step:

$$\phi^*_{ij} \quad = \quad \phi_{ij}(t) \; - g \; \frac{\Delta t}{2} \; h_{ij}(t) \; + \; \frac{\Delta t}{2} S(t)$$

- Update height a full time step

DPhi

$$h_{ij}(t + \Delta t) \quad = \quad h_{ij}(t) + \Delta t \boxed{\sum_{IJ} \; w_{ijIJ} \; \phi^*_{IJ}} \; + \; \Delta t \; T_{ij}(t + \Delta t)$$

- Update momentum remaining half time step

$$\phi_{ij}(t + \Delta t) \quad = \quad \phi^*_{ij} \; - \; g \; \frac{\Delta t}{2} \; h_{ij}(t + \Delta t) \; + \; \frac{\Delta t}{2} S(t + \Delta t)$$

Pseudo-code

```
// Update momentum half time step
for( all gridpoints i )
{
   phi[i] -= 0.5*dt * g * h[i];
   phi[i] += 0.5*dt * source[i];
}


// Compute DPhi and put it in array Dphi
ComputeDPhi( phi, Dphi );


// Update height full time step,
// then momentum remaining half time step
for( all gridpoints i )
{
   h[i]   +=     dt * Dphi[i];
   phi[i] -= 0.5*dt * g * h[i];
   phi[i] += 0.5*dt * source[i];
}
```

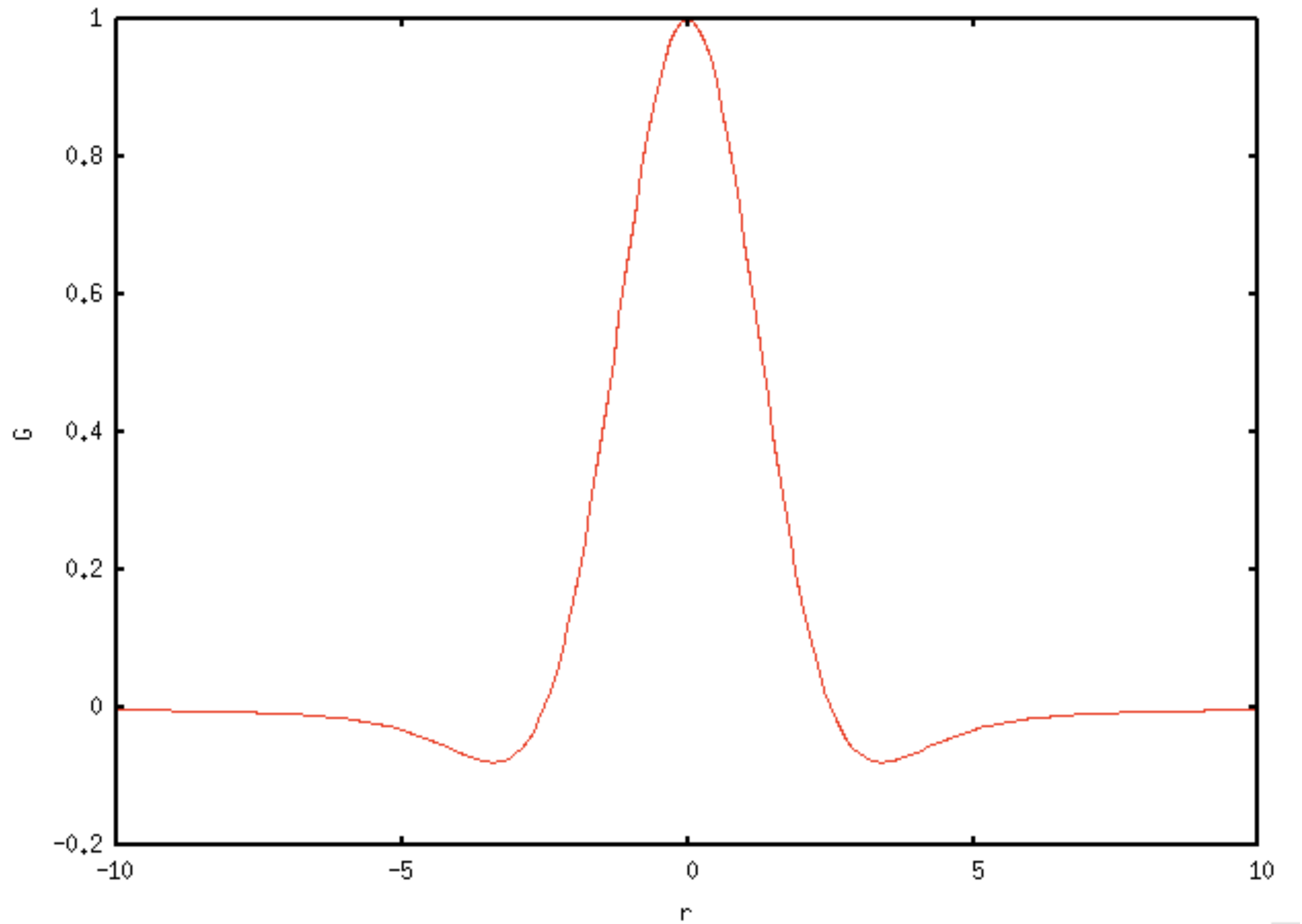
Weights from the Vertical Gradient

- Math details in notes

Weights pseudo-code

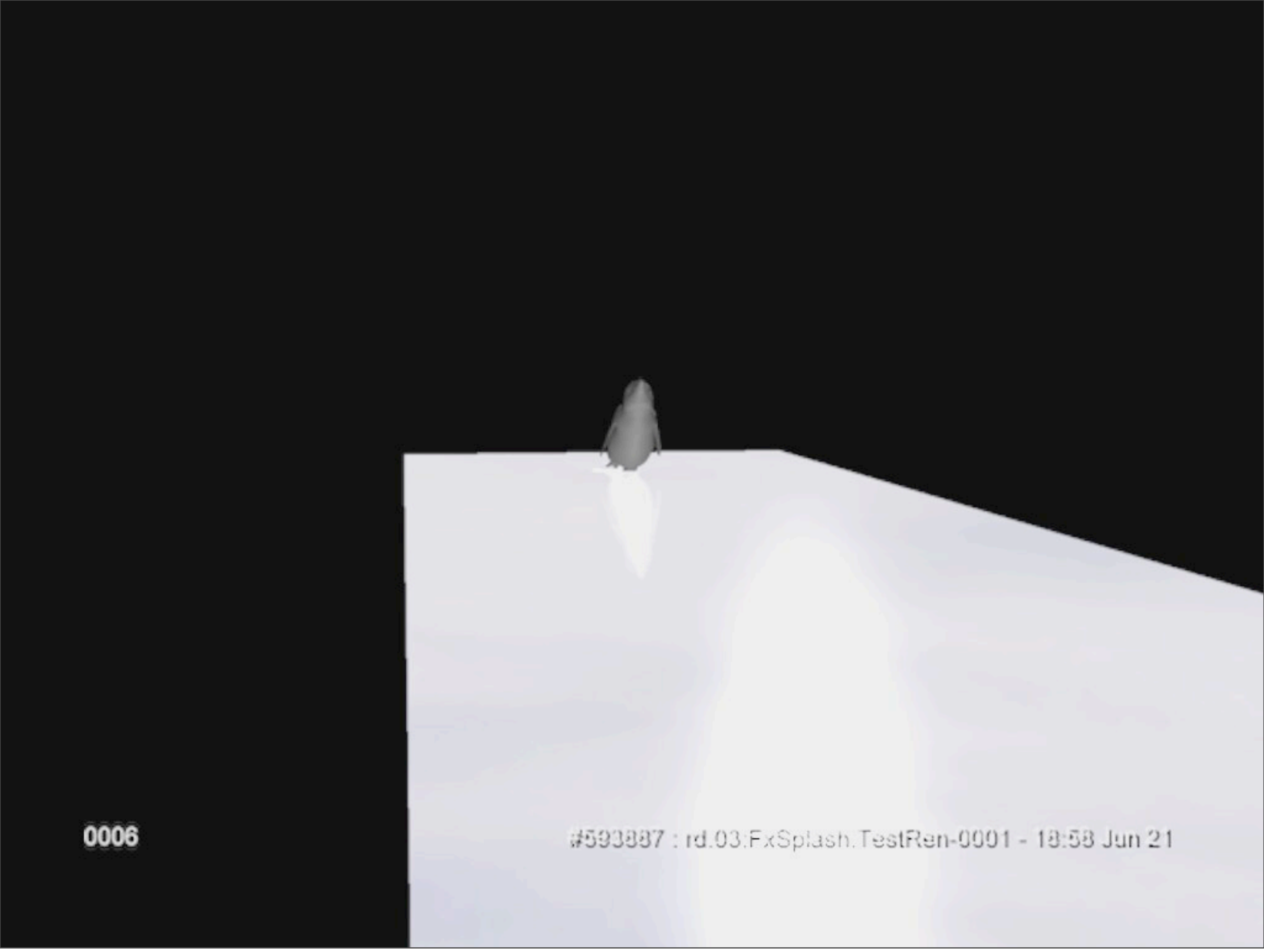
```
const float W( const double x, const double sigma )
{
    double dk = 0.01;
    double norm = 0;

    for(double k=0;k<10;k+=dk)
    {
        norm += k*k*exp(-sigma*k*k);
    }

    const double r = fabs(x);
    double kern = 0;
    for( double k=0;k<10;k+=dk)
    {
       kern += k*k*exp(-sigma*k*k)*j0(r*k);
    }
    return (kern/norm);
}
```

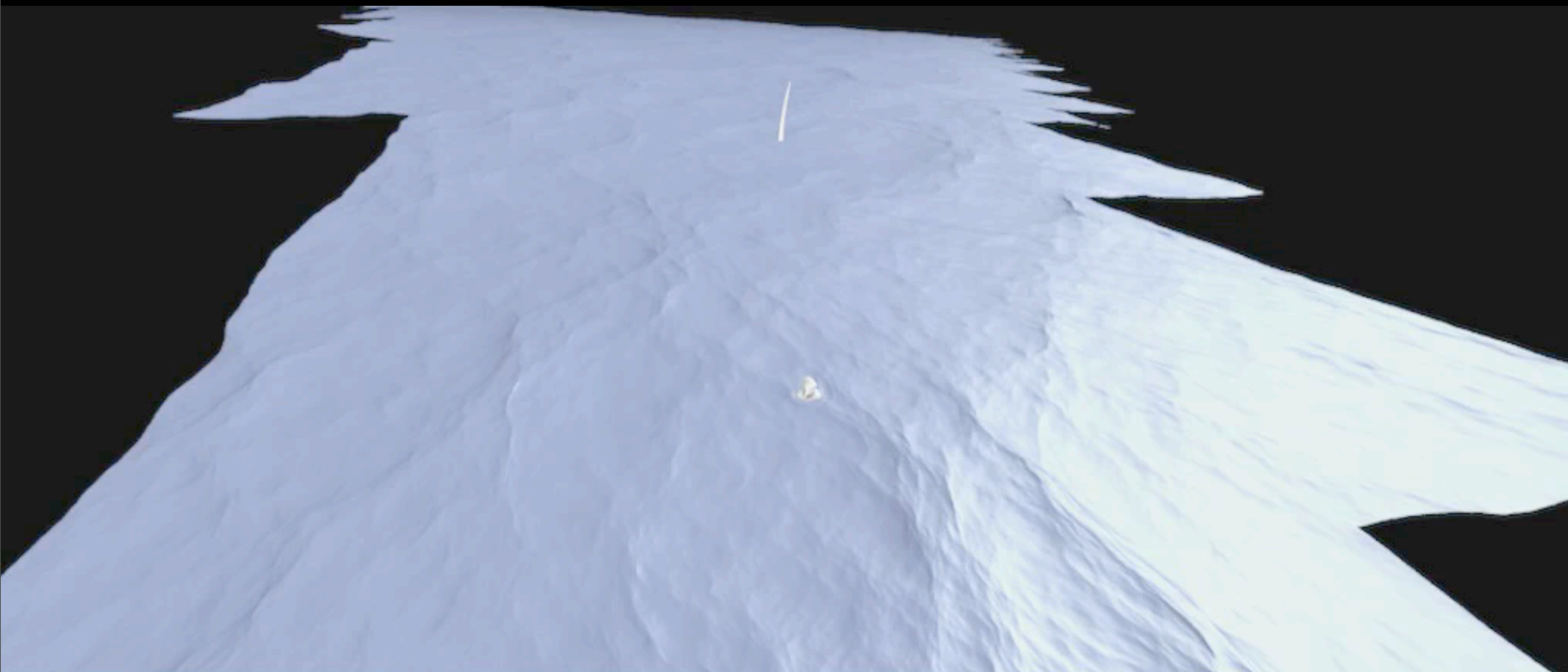
Convolution Kernel vs Separation



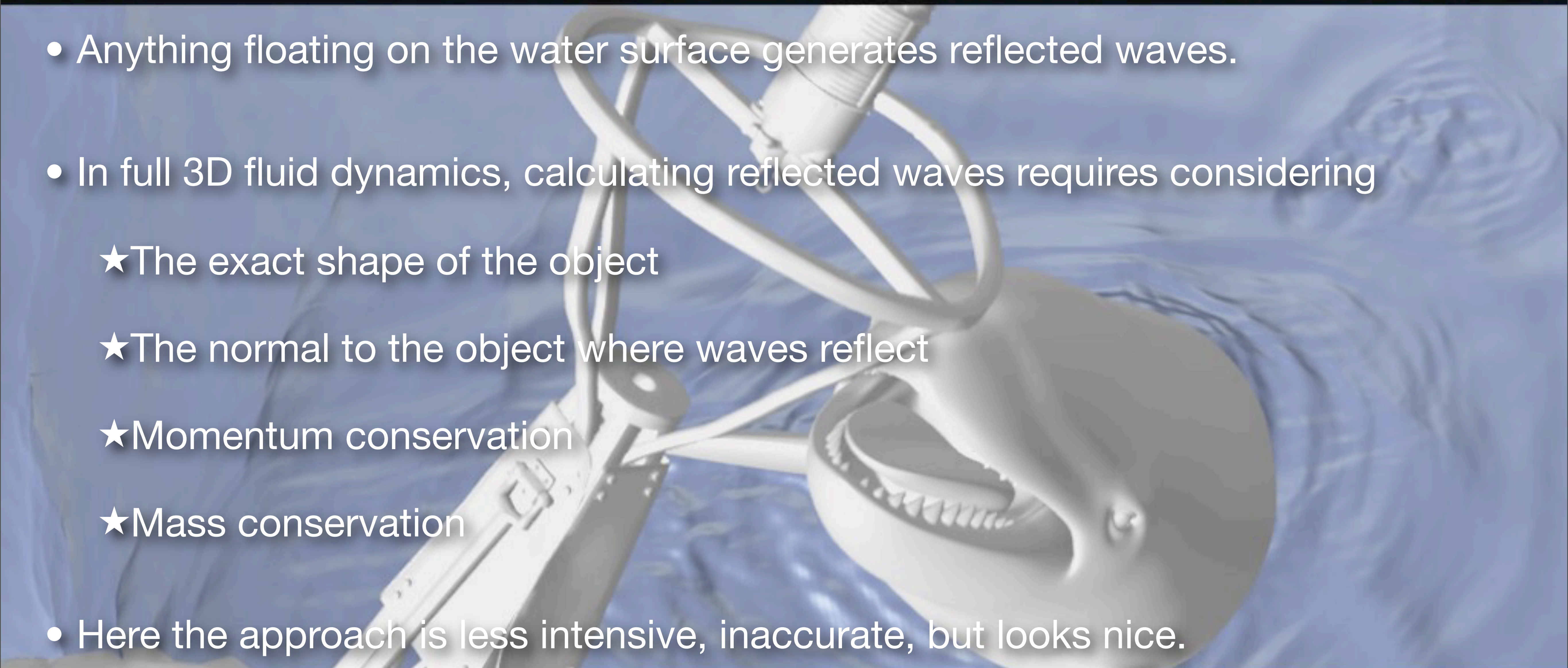
DPhi pseudo-code

```
// Weights have been precomputed into a table W[][]
void ComputeDPhi(float * phi, float * dphi)
{
    // just the interior in this example
    for(int ix=KERNEL_SIZE;ix<iwidth-KERNEL_SIZE;ix++)
    {
        for(int iy=KERNEL_SIZE;iy<iheight-KERNEL_SIZE;iy++)
        {
            int index = ix + iwidth*iy;
            float vd = 0;
            for(int iix=-KERNEL_SIZE;iix<=KERNEL_SIZE;iix++)
            {
                for(int iiy=-KERNEL_SIZE;iiy<=KERNEL_SIZE;iiy++)
                {
                    int iindex = ix+iix + iwidth*(iy+iiy);
                    vd += W[iix+KERNEL_SIZE][iiy+KERNEL_SIZE] * phi[iindex];
                }
            }
            dphi[index] = vd;
        }
    }
}
```


0006
#593887 : rd.03:FxSplash.TestRen-0001 - 18:58 Jun 21



Obstacles in a Scene



- Anything floating on the water surface generates reflected waves.
- In full 3D fluid dynamics, calculating reflected waves requires considering
  - ★The exact shape of the object
  - ★The normal to the object where waves reflect
  - ★Momentum conservation
  - ★Mass conservation
- Here the approach is less intensive, inaccurate, but looks nice.

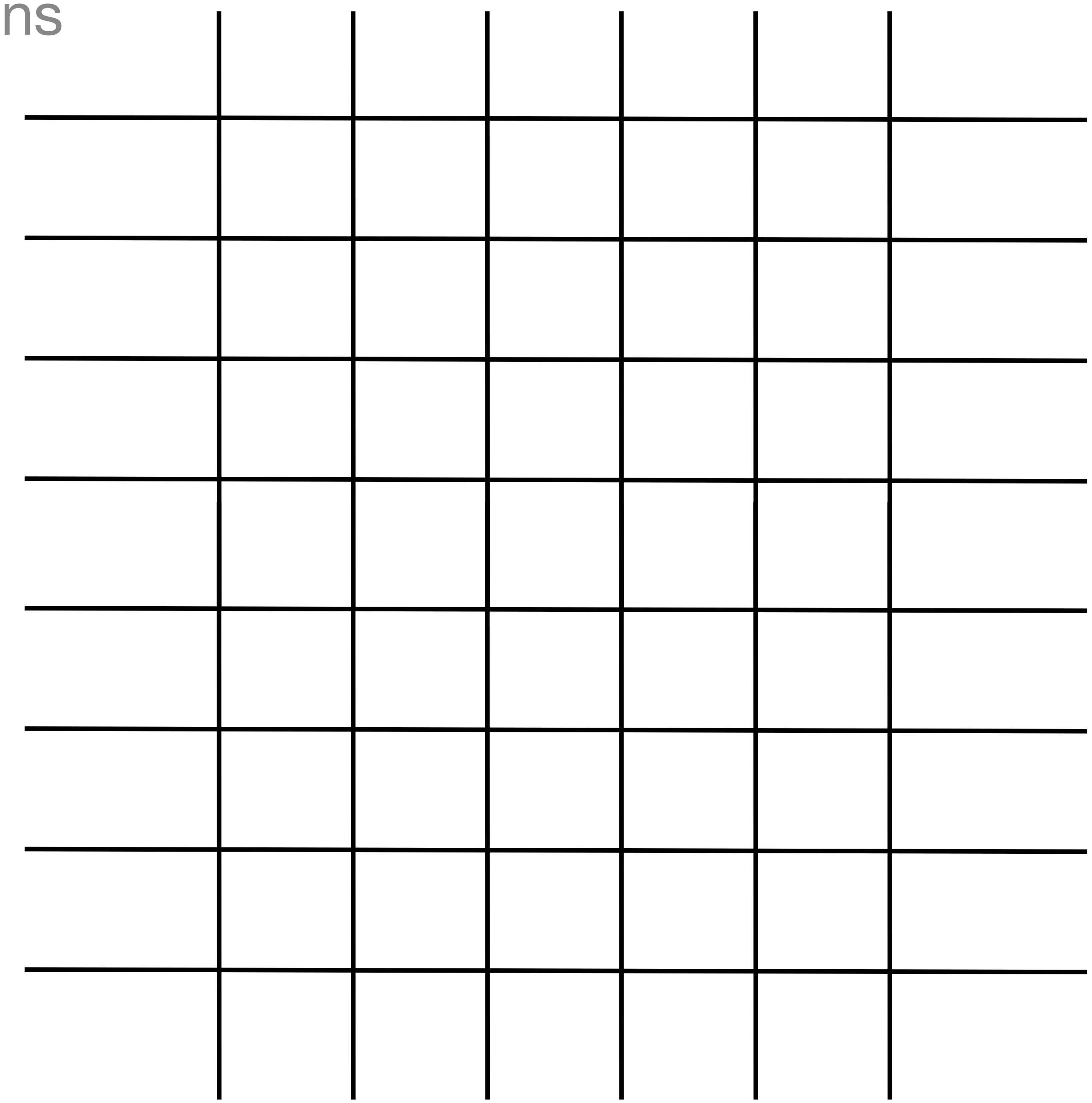
Obstacle Matte

- Create array that is the same dimensions as the simulation grid.
- Set all of the gridpoints to have value 1.
- For gridpoints inside the obstacle, set the value to 0.
- For gridpoints on the edge of the obstacle, set the value between 0 and 1.

Assigning Obstacle Matte Values

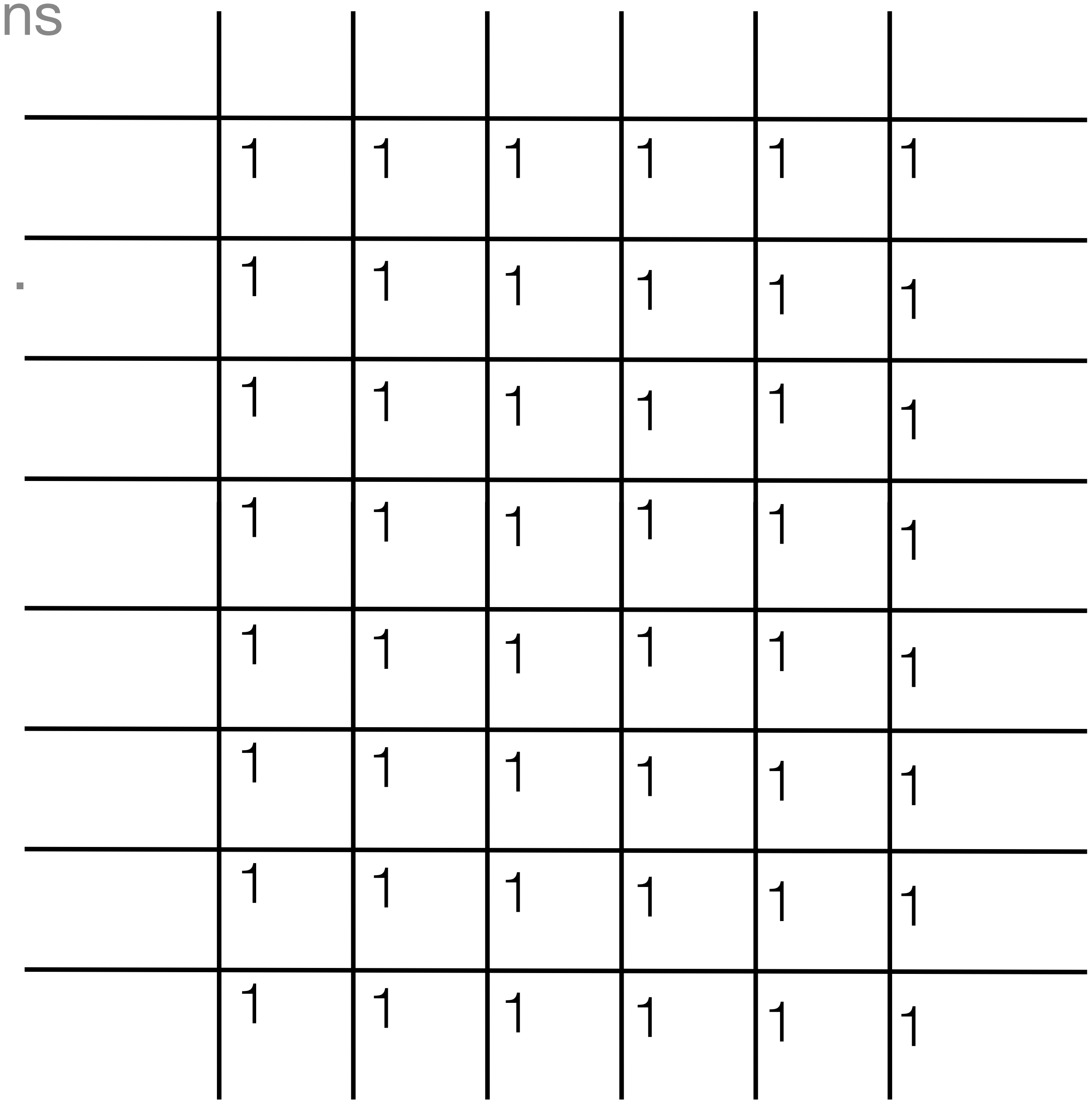
Assigning Obstacle Matte Values

- Create array that is the same dimensions as the simulation grid.



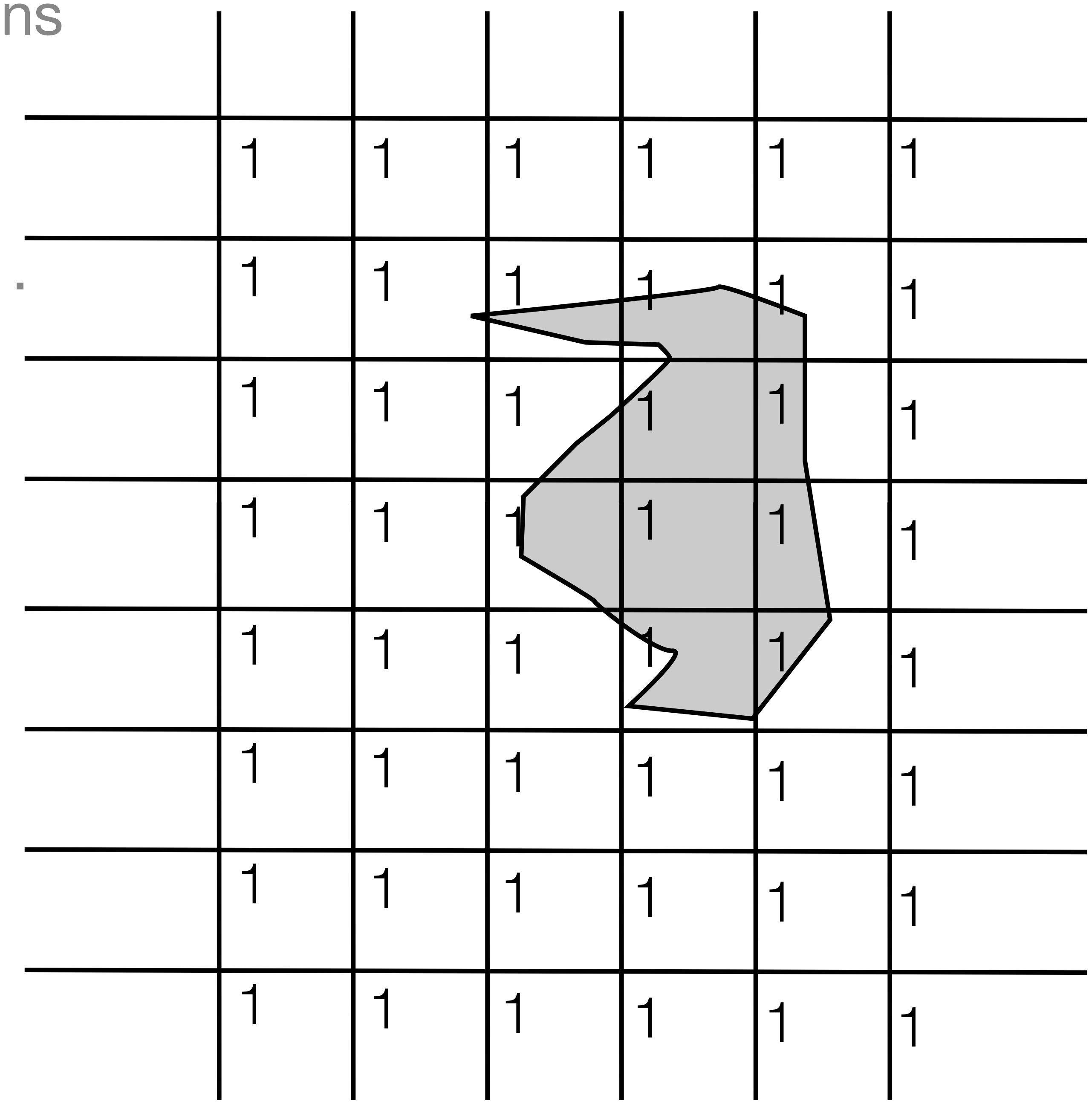
Assigning Obstacle Matte Values

- Create array that is the same dimensions as the simulation grid.
- Set all of the gridpoints to have value 1.



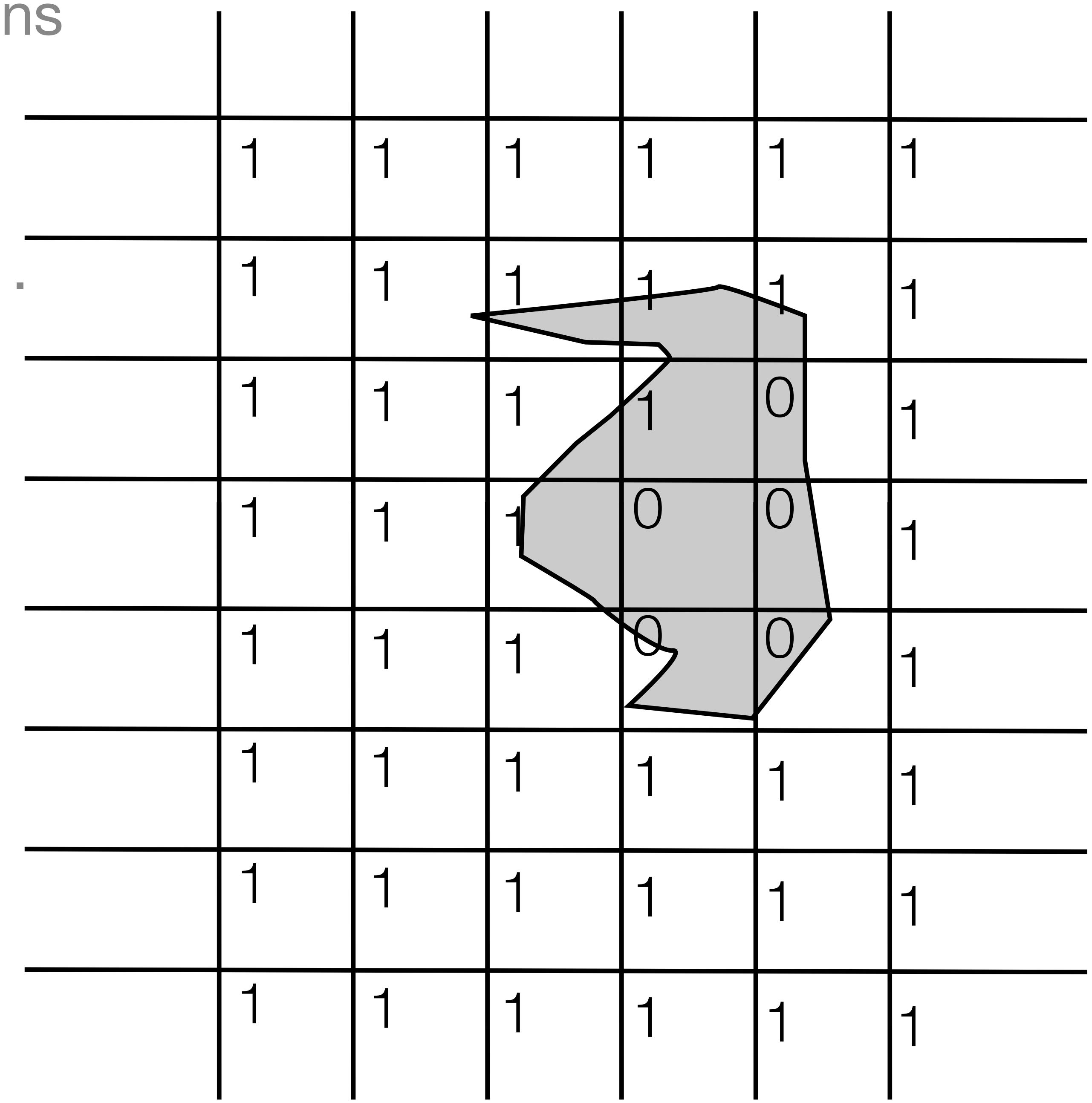
Assigning Obstacle Matte Values

- Create array that is the same dimensions as the simulation grid.
- Set all of the gridpoints to have value 1.
- Shape overlay on grid



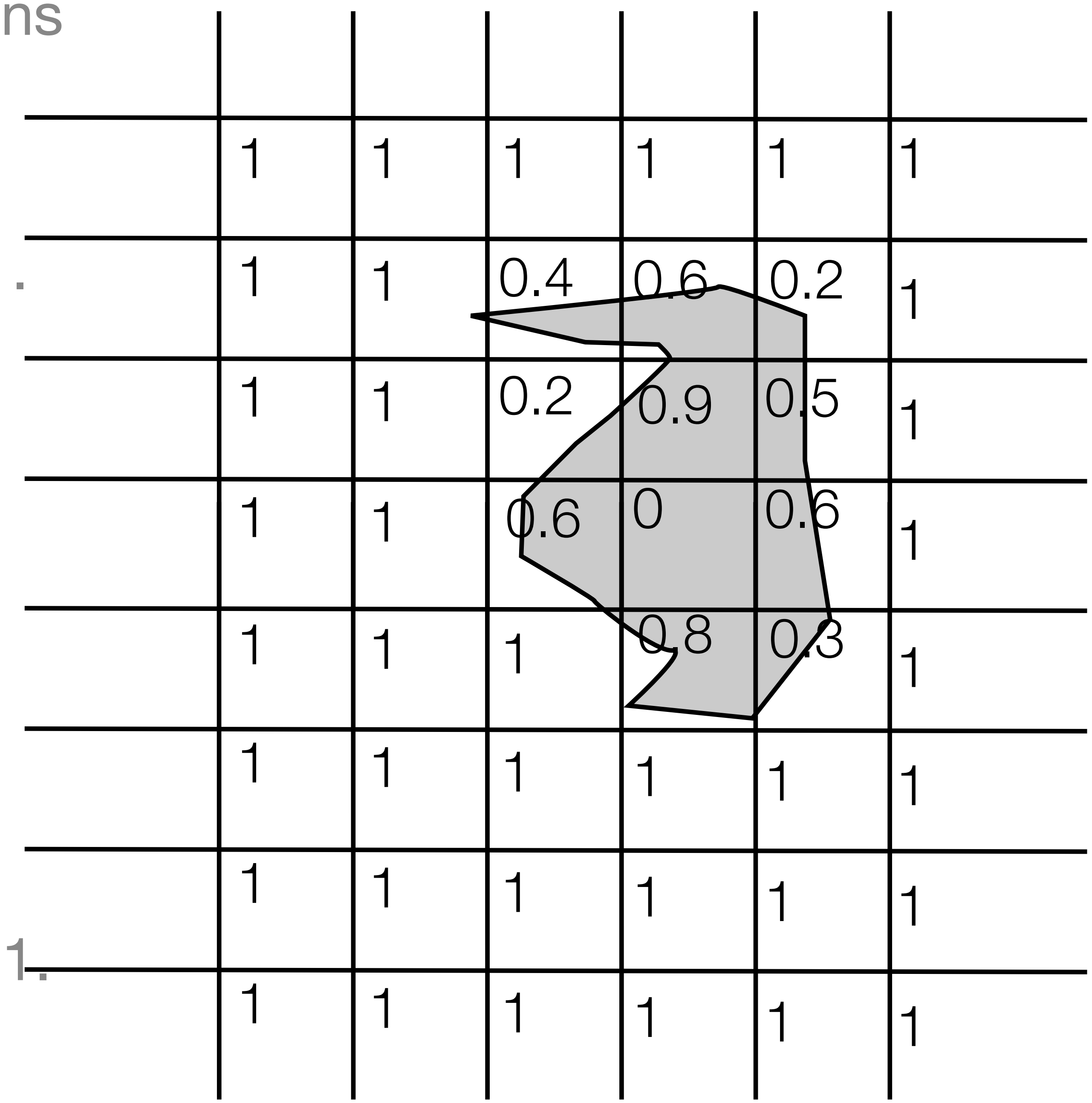
Assigning Obstacle Matte Values

- Create array that is the same dimensions as the simulation grid.
- Set all of the gridpoints to have value 1.
- Shape overlay on grid
- For gridpoints inside the obstacle, set the value to 0.



Assigning Obstacle Matte Values

- Create array that is the same dimensions as the simulation grid.
- Set all of the gridpoints to have value 1.
- Shape overlay on grid
- For gridpoints inside the obstacle, set the value to 0.
- For gridpoints on the edge of the obstacle, set the value between 0 and 1.

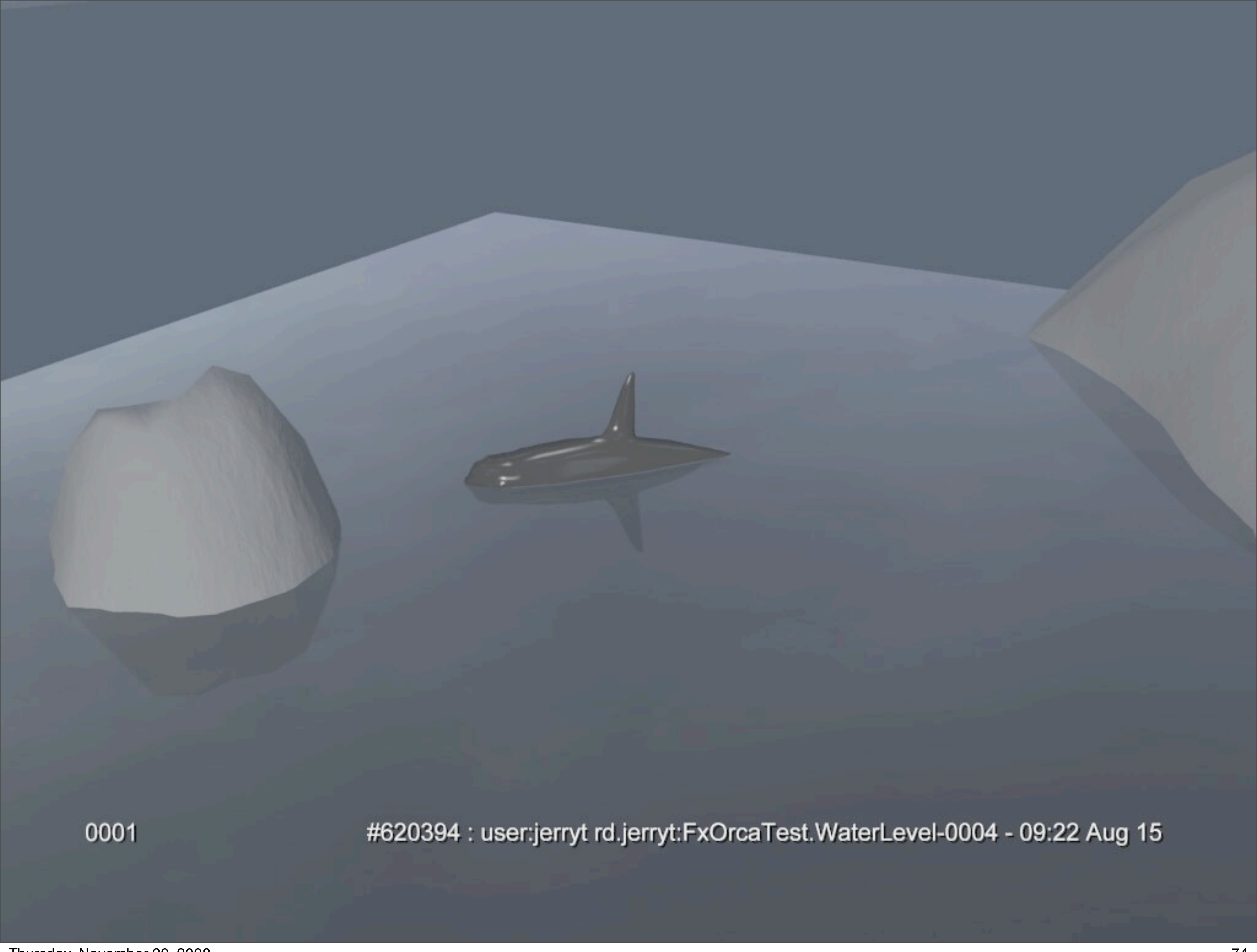


Using Obstacle Matte

- Multiply height and momentum grids by obstacle matte, suppressing height and momentum inside the obstacle.
- The propagation and DPhi filtering automatically generate reflected waves.

Pseudo-code

```
// Update momentum half time step
for( all gridpoints i )
{
    phi[i] -= 0.5*dt * g * h[i];
   phi[i] += 0.5*dt * source[i];
}


// Compute DPhi and put it in array Dphi
ComputeDPhi( phi, Dphi );


// Update height full time step,
// then momentum remaining half time step
for( all gridpoints i )
{
    h[i]   +=     dt * Dphi[i];
    phi[i] -= 0.5*dt * g * h[i];
   phi[i] += 0.5*dt * source[i];
   h[i] *= obstacle_matte[i];
   phi[i] *= obstacle_matte[i];
}
```


0001
#620394 : user:jerryt rd.jerryt:FxOrcaTest.WaterLevel-0004 - 09:22 Aug 15

iwave_paint demo

- Source code in an appendix to 2004 version of Siggraph Course Notes
- http://tessendorf.org/reports.html