Game Developers Conference 08
February 18-22
San Francisco
Moscone Center

NVIDIA
ageia

Fast Water Simulation for Games Using Height Fields

Matthias Müller-Fischer

Intro

PhysX NVIDIA

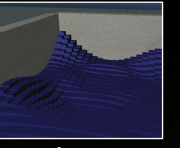
- 1999 Ph. D. ETH Zürich: Polymer simulation
- 1999-2001 Post doc MIT: Simulation in CG
- 2002 Co-founder NovodeX (physics middleware)
- 2004-2008 Head of research Ageia (SDK features)
- 2008 Research lead PhysX SDK at Nvidia

- Zürich office: R&D PhysX SDK
- currently Height Field Fluids
  - New Ideas
  - Lessons learned so far

Hello World!

PhysX NVIDIA

- Use two arrays float `u[N,M], v[N,M]`
- Initialize `u[i,j]` with interesting function
- Initialze `v[i,j]=0`



demo

```
loop
   v[i,j] +=(u[i-1,j] + u[i+1,j] + u[i,j-1] + u[i,j+1])/4 – u[i,j]
   v[i,j] *= 0.99
   u[i,j] += v[i,j]
   visualize(u[])
endloop
```

- Clamp at boundary e.g. def. `u[-1,j] = u[0,j]`

Motivation

PhysX NVIDIA

That's it: Thank you for your attention!

- The magic (physics) behind it
- Object interaction
- Boundary conditions
- More complex domains
- Barking waves

- Implement it yourself (use CUDA!)
- Soon:
  - Use the PhysX SDK: `scene->createHeightfieldWater(..)`
  - Drop HF water into your game

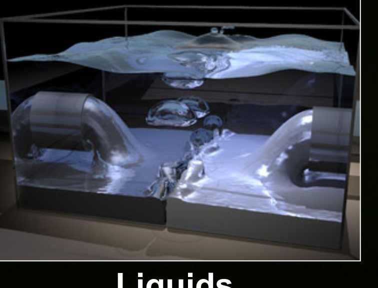
Outline

- Introduction
  - Fluids are cool!
  - From offline to real-time
- Physics is not hard
  - f = ma
  - Simulation
- Height field water
  - Simulation of columns
  - Boundaries
  - Object interaction
  - Horizontal motion

Off-line Fluid Simulation

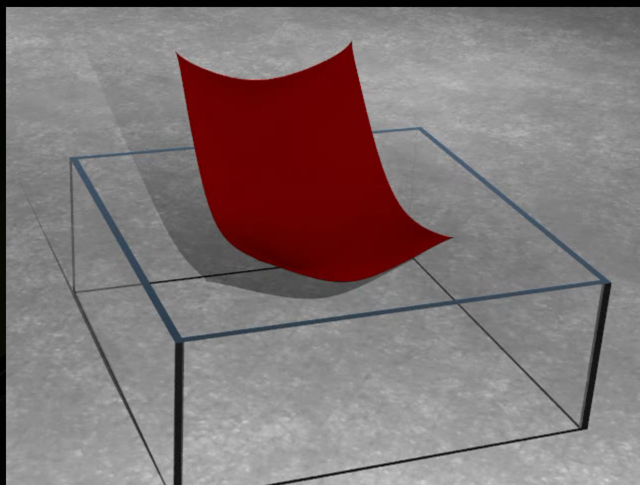
- State of the art is impressive!
- Google "Robert Bridson", "Ron Fedkiw", "James O'Brien", …

Gas



Liquids

Example



Guendelman et al. (SIGGRAPH 2005)

Off-line Simulation Times

- Typical setup
  - Grid size 512^3 cells
  - Linear system with >100 million unknowns!
  - Level sets on even finer grids
  - Raytracing (reflection / refraction / caustics)
- Photorealistic results
- 10 seconds – 50 minutes per frame!
- Recently real-time on a GPU (Crane et al.)
  - Coarser grid, simplified physics and rendering

Game Requirements

- CHEAP TO COMPUTE!
  - Small fraction of the 15 ms of a frame
- Stable even in non-physical settings
  - Kinematic, fast moving objects / characters
- Low memory consumption
- Challenge:
  - Get as close as possible to off-line results
  - Meet all these constraints!

Solutions

PhysX NVIDIA

- Resolution reduction
  - Blobby and coarse look
  - Details disappear
- Use real-time tricks!
  - Reduction dimension 3d → 2d
  - Physics low-res, appearance hi-res (shader effects)
  - Simulate active and visible regions only (sleeping)
  - Level of detail (LOD)

GameDevelopers Conference 08

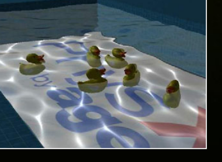
Approaches

PhysX NVIDIA

- Procedural Water
  - Unbounded surfaces, oceans

- Particle Systems
  - Splashing, spray, puddles, smoke

- Height field Fluids
  - Ponds, lakes, rivers



GameDevelopers Conference 08

Procedural Animation

PhysX NVIDIA

- Simulate the effect, not the cause
  - [Fournier86], [Hinsinger02], [Bridson07], [Yuksel07]
- Pros
  - High level of control
  - No limits to creativity (e.g. superimpose sine waves)
- Cons
  - Modeling fluid-scene interaction more difficult
  - Less realistic

OGRE demo

GameDevelopers Conference 08

Particle Based Fluids

PhysX NVIDIA

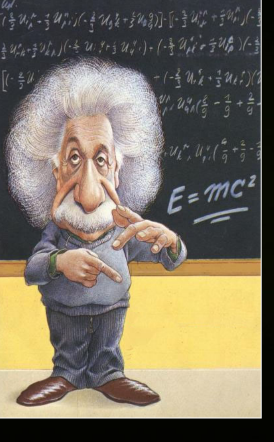
- Fluid is represented as a set of particles
  - [Monaghan92], [Premoze03], [Müller03]
- Pros
  - Particle simulation is fast and quite simple
  - Spray, splashing, small puddles, blood, runnels, debris
- Cons
  - Surface tracking difficult
    - Screen space meshes [Müller07]
  - Not suited for lakes, rivers, oceans

demo

GameDevelopers Conference 08

PhysX NVIDIA

Physics for simulations is not that hard



GameDevelopers Conference 08

Useful Equations (PDEs)

PhysX NVIDIA

- **Fluid (gas / liquid):**

$$\underbrace{\rho}_{m}\underbrace{(\mathbf{v}_t + \mathbf{v}\cdot\nabla\mathbf{v})}_{\mathbf{a}} = \underbrace{-\nabla p+\mathbf{f}}_{\mathbf{f}}$$

- **Elastic solid:**

$$\underbrace{\rho}_{m}\ \underbrace{\mathbf{u}_{tt}}_{\mathbf{a}} = \underbrace{\nabla\cdot\sigma_s(\mathbf{u})+\mathbf{f}}_{\mathbf{f}}$$

- **Membrane:**

$$\underbrace{u_{tt}}_{\mathbf{a}} = \underbrace{c^2\,\nabla^2 u}_{\mathbf{f}/m}$$

Game Developers Conference 08

Newton's Second Law

PhysX NVIDIA

$$\mathbf{f} = m\mathbf{a}$$

$$\mathbf{a} = \mathbf{f}\ /\ \mathrm{m}$$

change of velocity per unit time = force divided by mass

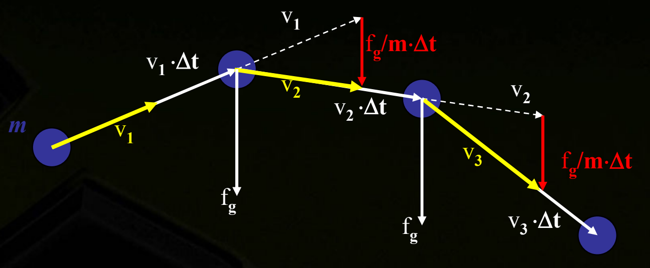
Simulation Loop (explicit Euler integration):

- compute force (based on actual positions and velocities)
- velocity += force/mass · time step
- position += velocity · time step

Game Developers Conference 08

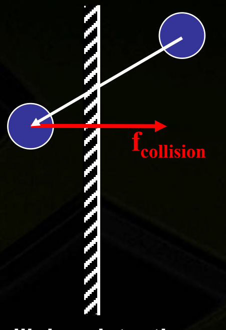
Example: Particle

- Need to store position and velocity!



Other Particle Forces

Smoke, debris



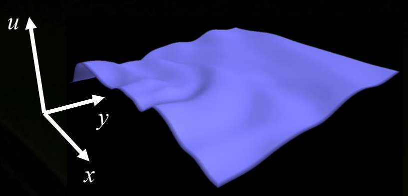
Collision detection
particle ↔ game level!

Liquid (e.g. SPH)

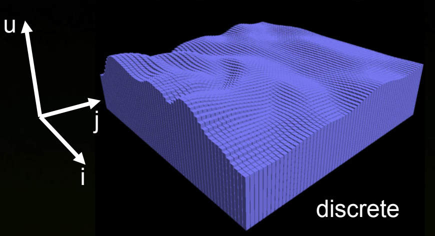
Neighbor search! [Teschner03]

Height Field Fluids

Height Field Water



continuous

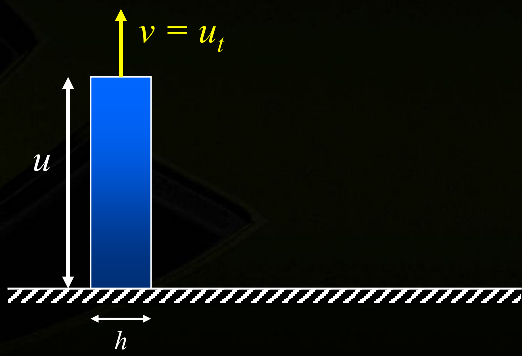


discrete

- Represent fluid surface as a
  - 2d function $u(x,y)$
  - 2d array `u[i,j]`
- Pro: Reduction from 3d to 2d
- Cons: One value per $(x,y) \rightarrow$ no breaking waves

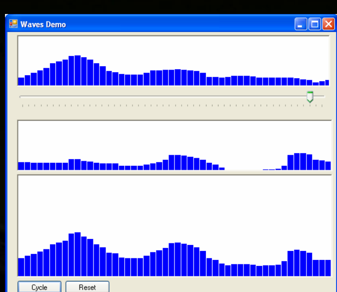
Position and Velocity

- Position = column height
- Velocity = change of column height per time unit
- (h = column width)



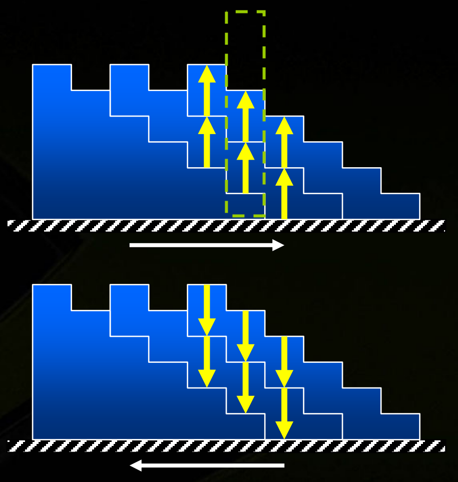
Force

- Force causes waves to travel at speed c in all directions



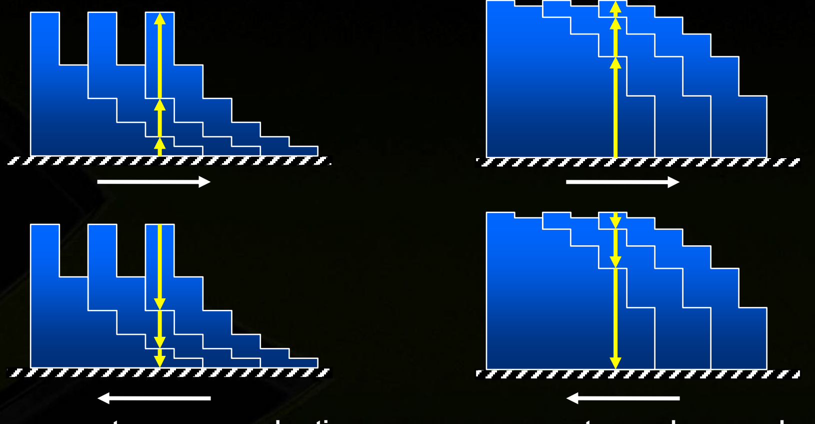
demo

Constant Slope



- Constant slope → no velocity change → no force

Curved Wave Front



pos curvature → up acceleration

neg curvature → down acceleration

→ force proportional to curvature!

Wave Equation

- **Newton's Law (continuous):**

$$u_{tt} = f/m = k \cdot u_{xx}/m$$

- **Assume m constant : Can replace k/m by c^2**

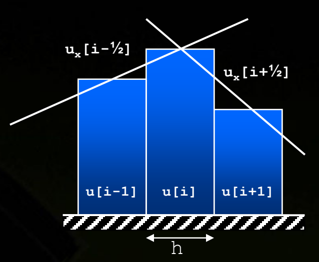
$$\boxed{u_{tt} = c^2 \cdot u_{xx}}$$ **1d wave equation [Jeffrey02]**

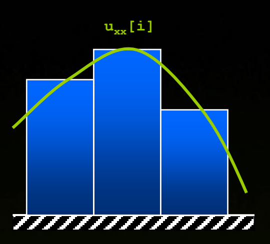
- **Solution:**

$$u(x,t) = f(x + ct) + g(x - ct)$$ **for any f,g**

- **Constant c is the speed at which waves travel**

Curvature






```
ux[i-½]=(u[i]-u[i-1])/h
ux[i+½]=(u[i+1]-u[i])/h
```

```
uxx[i]=(ux[i+½]-ux[i-½])/h
      =(u[i+1]-2u[i]+u[i-1])/h²
```

- **1d curvature:**

```
(u[i+1]+u[i-1]-2u[i])/h²
```

- **2d curvature:**

```
(u[i+1,j]+u[i-1,j]+u[i,j+1]+u[i,j-1]-4u[i,j])/h²
```

Column Simulation Step

- 1d simulation

```
forall i
   f      = c²*(u[i-1] + u[i+1] – 2u[i])/h²
   v[i]   = v[i] + f*Δt
   u_new[i] = u[i] + v[i]*Δt
endfor
forall i: u[i] = u_new[i]
```

- 2d simulation

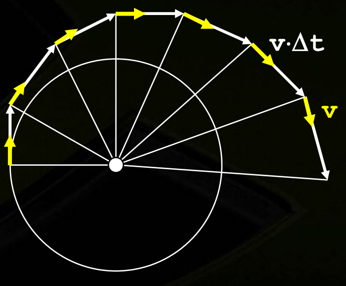
```
forall i,j
   f        = c²*(u[i+1,j]+u[i-1,j]+u[i,j+1]+u[i,j-1]
                – 4u[i,j])/h²
   v[i,j]   = v[i,j] + f*Δt
   u_new[i,j] = u[i,j] + v[i]*Δt
endfor
forall i,j: u[i,j] = u_new[i,j]
```


Some Remarks

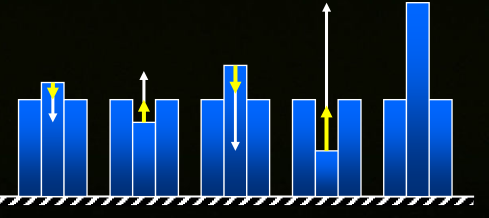
- Information can only propagate 1 cell per time step
- Upper limit for choice of wave speed
  - $c < h / \Delta t$
- or upper limit for choice of time step
  - $\Delta t < h/c$ (Courant–Friedrichs–Lewy (CFL) condition)
- Boundary conditions needed
  - Clamp yields wave reflection
  - Wrap yields periodic propagation

Numerical Explosions

- Explicit Euler step `x += v·Δt`
- Assumes velocity constant during a time step
- Particle sample:



- Column sample (overshooting):



Damping vs. Clamping

- **Damping force** (physically "correct")
  - `f = -k·v`
  - But how to choose `k`? No stability guarantees
- **Scaling** (unphysical, more direct control)

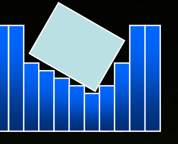
```
v = s·v   // s < 1, smaller time step -> stronger effect
```

- **Clamping** (unphysical, direct control)

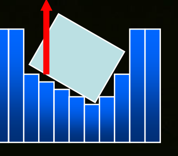
```
offset = (u[i+1,j]+u[i-1,j]+u[i,j+1]+u[i,j-1])/4 – u[i,j]
maxOffset = maxSlope * h  // independence of resolution h:

if (offset > maxOffset)  u[i,j] += offset – maxOffset
if (offset < -maxOffset) u[i,j] += offset + maxOffset
```

Object Interaction

- Object → water
  - Object pushes columns beneath it down
  - Add the removed water in the vicinity!

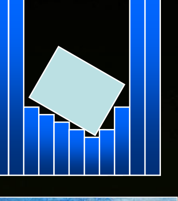


- Water → object
  - Archimedes' principle
  - Each column below the object applies force $f = -\Delta u \cdot h^2 \cdot \rho \cdot g$ to body at its location
  - Δu is the height replaced by the body, ρ water density, g gravity

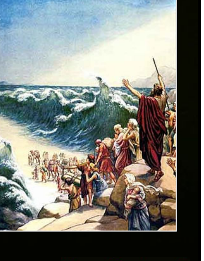


Fully Immersed Bodies

- Body below water surface
- Hole appears above the body
- Non-physical



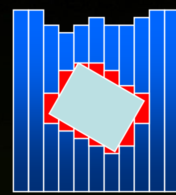
- See story of divided sea



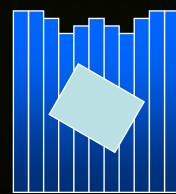
GameDevelopers Conference 08

Solution

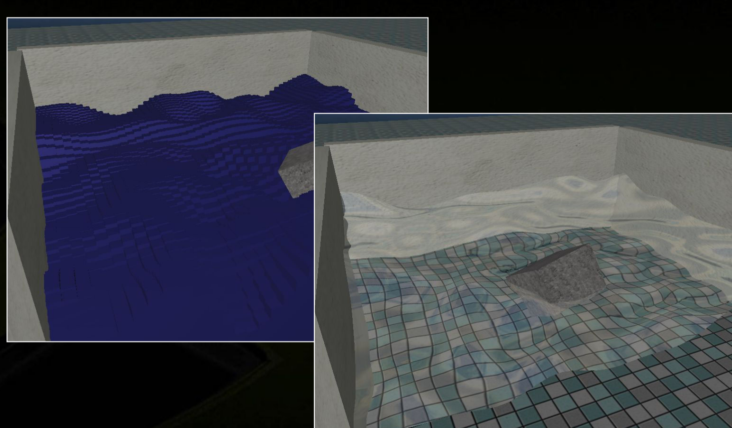
- New state variable r[i,j]:
  - Each column stores the part r[i,j] of u[i,j] currently replaced by solids



- At each time step:
  - u[i,j] is not modified directly
  - Δ r[i,j] = r^{t}[i,j]-r^{t-1}[i,j] is distributed as water u to the neighboring columns
  - In case of a negative difference water is removed

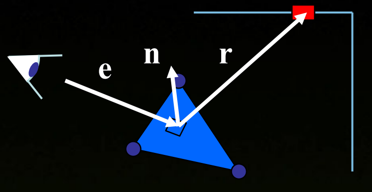


From Bars to Water



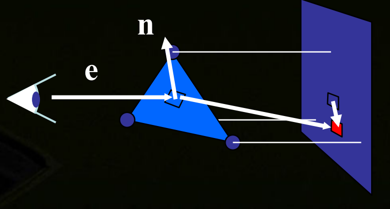
Water Rendering

- Reflection



cube map

- Refraction



Scene
below surface
rendered to
texture

- Caustics: Cheating - Animated texture

GameDevelopers Conference 08

Unbounded domain

- Simulations in tank not practical for games
- Solution: Hybrid Simulation

3d Euler
Height field

[Thuerey06]

Height field
Procedural

- No reflections or wrap on boundary!

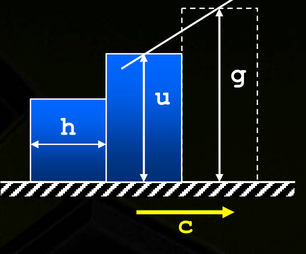
GameDevelopers Conference 08

3d-2d Simulation

- N. Thuerey et al., Animation of Open Water Phenomena with coupled Shallow Water and Free Surface Simulations, SCA 06

Open Boundaries

- Let waves with speed c pass freely
- Use ghost column on border with height memory g



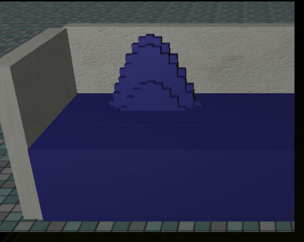
- Temporal change of g proportional to border slope

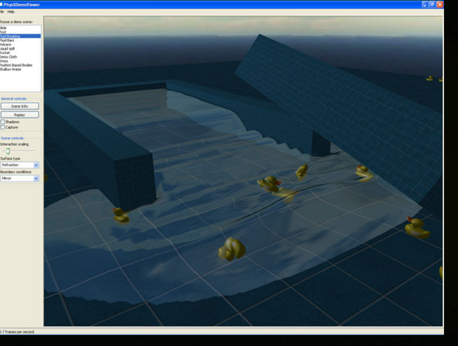
  $(g_{new} - g)/\Delta t = -c \cdot (g_{new} - u)/h$

- Update rule:

  $g_{new} = (c \cdot \Delta t \cdot u + g \cdot h) \ / \ (h + c \cdot \Delta t)$

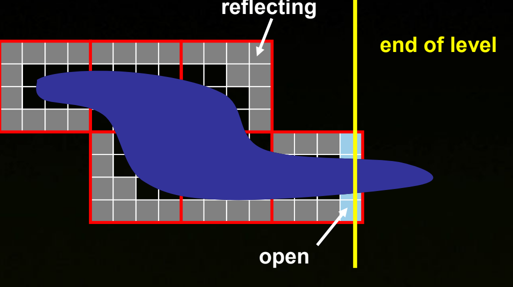
Demos



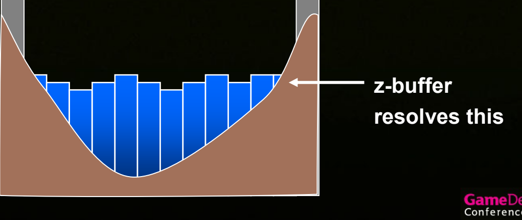


Irregular Domain

- Sparse tiling (top view)



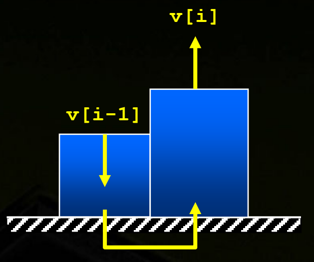
- General terrain (side view)



Horizontal Motion

PhysX NVIDIA

- We only work with vertical velocities
- Horizontal velocity for dragging floating objects



$v_h[i] = (v[i]-v[i-1])\cdot h + (v[i+1]-v[i])\cdot h$

- Better model: Shallow Water Equations

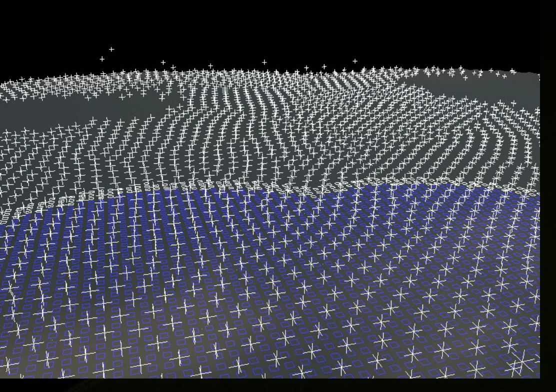
Shallow Water Equations

PhysX NVIDIA

$$Du/Dt = -u\nabla\cdot\mathbf{v}$$
$$D\mathbf{v}/Dt = -g\,\nabla(u+b) + \mathbf{a}_{ext}$$

- Oversimplified:
  - Compressible Eulerian fluid simulation on a 2d staggered horizontal grid
  - Density interpreted as height
- No further details at this time ☺

Demo

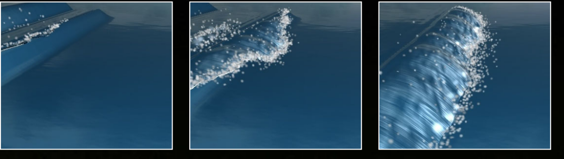


Water Loss

- Big problem in Euler / level set simulations
- Less problematic in height field fluids

- Simple, bullet proof solution:
  - $V = \Sigma_{i,j}\ u[i,j]$
  - $V_{new} = \Sigma_{i,j}\ u_{new}[i,j]$
  - Distribute $V\text{-}V_{new}$ evenly
  - Within connected regions!

Breaking Waves [Thuerey07]



- Height fields cannot capture breaking waves
- Identify steep wave fronts
- Construct and track line along front
- Emit patch of connected particles
- Generate mesh with given thickness for rendering (plus particles for foam)

Breaking Waves

References

[Bridson07] R. Bridson et al., Curl noise for procedural fluid flow, Siggraph 07

[Fournier86] A. Fournier and W. T. Reeves. A simple model of ocean waves, SIGGRAPH 86, pages 75–84

[Hinsinger02] D. Hinsinger et al., Interactive Animation of Ocean Waves, In *Proceedings of* SCA 02

[Jeffrey02] A. Jeffrey, Applied Partial Differential Equations, Academic Press, ISBN 0-12-382252-1

[Monaghan92] J. J. Monaghan, Smoothed particle hydrodynamics. Annual Review of Astronomy and Astrophysics, 30:543–574, 1992.

[Müller07] M. Müller et al., Screen Space Meshes, SCA 07.

[Müller03] M. Müller et al., Particle-Based Fluid Simulation for Interactive Applications, SCA 03, pages 154-159.

[O'Brien95] J. O'Brien and J. Hodgins, Dynamic simulation of splashing fluids, In Computer Animation 95, pages 198–205

[Premoze03] S. Premoze et al., Particle based simulation of fluids, Eurographics 03, pages 401-410

[Teschner03] M. Teschner et al., Optimized Spatial Hashing for Collision Detection of Deformable Objects, VMV 03

[Thuerey07] N. Thuerey et al., Real-time Breaking Waves for Shallow Water Simulations Pacfific Graphics 07

[Thuerey06] N. Thuerey et al., Animation of Open Water Phenomena with coupled Shallow Water and Free Surface Simulations, SCA 06.

[Yuksel07] Cem Yuksel et al., Wave Particles, Siggraph 07

Questions?

Slides available soon at www.MatthiasMueller.info