

Smoothed Particle Hydrodynamics

Application Example

Alan Heirich | November 29, 2010



Fluids

Navier-Stokes equations

Smoothed Particle Hydrodynamics

OpenCL simulation



Fluids

- Liquids, e.g. water
- Gasses, e.g. air
- Plasmas



Fluids

- Described by (incompressible) Navier-Stokes equations

$$\rho \left[\frac{\partial v}{\partial t} + v \bullet \nabla v \right] = \rho g - \nabla p + \mu \nabla^2 v \quad \rho(\nabla \bullet v) = 0$$

- Driven by gravity g , pressure ∇p and velocity $\mu \nabla^2 v$
 - Fluid flows from high pressure to low pressure
 - Viscosity μ determines fluid stickiness
 - Low viscosity: air, water
 - High viscosity: honey, mud



Fluids

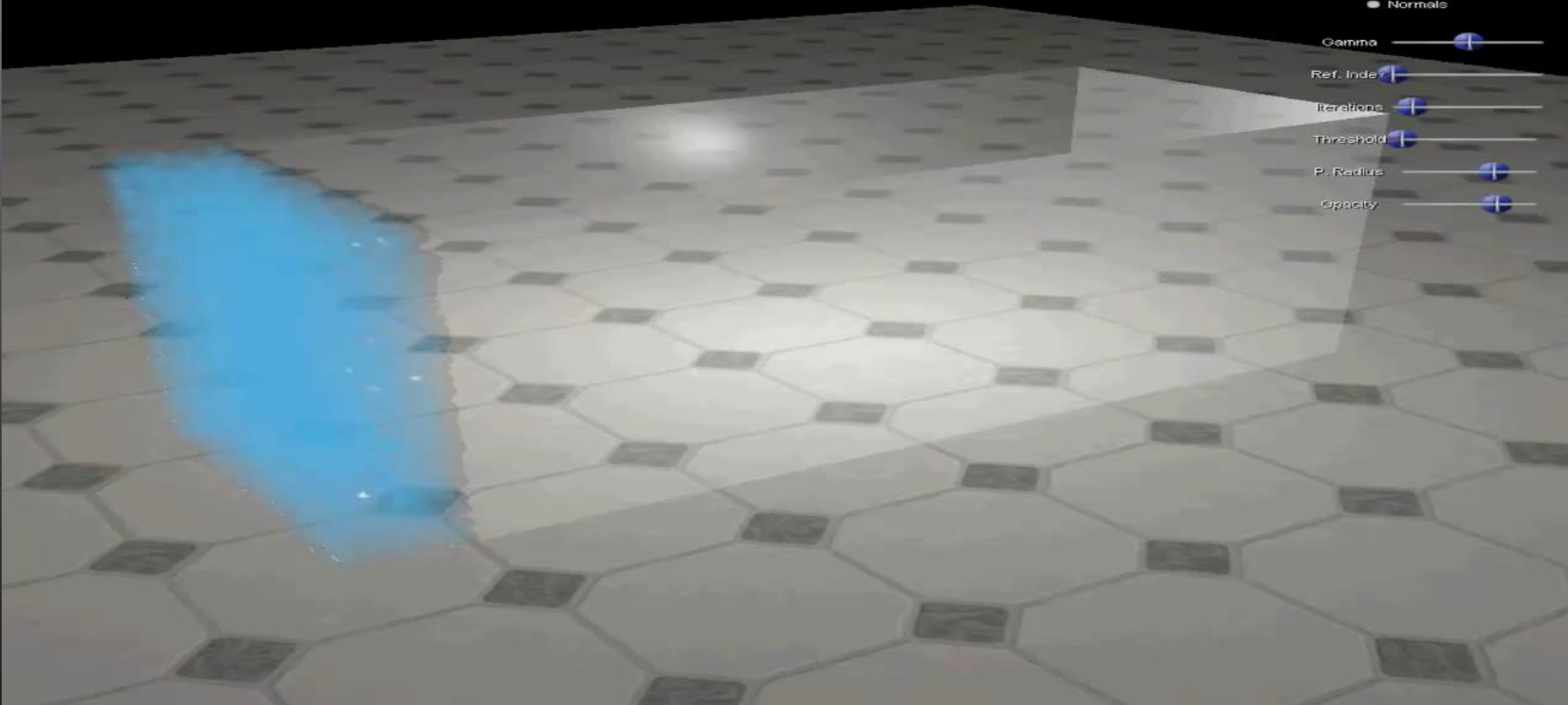
www.fraps.com

KEYBOARD
Press SPACE bar to start and stop the simulation
Press ESC to exit

- Toggle full screen
- Change device (F2)
- Toggle REF (F3)
- Toggle WARP (F4)

- Single Sphere
- Points
- Fluid
- Normals

- Gamma
- Ref. Index
- Iterations
- Threshold
- P. Radius
- Opacity



Fluids

Navier-Stokes equations

Smoothed Particle Hydrodynamics

OpenCL simulation



Incompressible Navier-Stokes equations

$$\rho \left[\frac{\partial \mathbf{v}}{\partial t} + \mathbf{v} \cdot \nabla \mathbf{v} \right] = \rho \mathbf{g} - \nabla p + \mu \nabla^2 \mathbf{v}$$

$$\rho (\nabla \cdot \mathbf{v}) = 0 \quad (\text{mass continuity})$$

- ρ - density, p - pressure (scalars)
- \mathbf{g} - gravity, \mathbf{v} - velocity (vectors)

- $\mathbf{v} \cdot \nabla \mathbf{v} \equiv \left[v_x \frac{\partial v_x}{\partial x}, v_y \frac{\partial v_y}{\partial y}, v_z \frac{\partial v_z}{\partial z} \right]$ convective acceleration

- $\nabla p \equiv \left[\frac{\partial p}{\partial x}, \frac{\partial p}{\partial y}, \frac{\partial p}{\partial z} \right]$ $p = k(\rho - \rho_0)$ resting density ρ_0

$$\nabla^2 \mathbf{v} \equiv \left[\nabla^2 v_x, \nabla^2 v_y, \nabla^2 v_z \right], \quad \nabla^2 v_x \equiv \frac{\partial^2 v_x}{\partial x^2} + \frac{\partial^2 v_x}{\partial y^2} + \frac{\partial^2 v_x}{\partial z^2}$$



Incompressible Navier-Stokes equations

$$\rho \left[\frac{\partial \mathbf{v}}{\partial t} + \mathbf{v} \cdot \nabla \mathbf{v} \right] = \rho \mathbf{g} - \nabla p + \mu \nabla^2 \mathbf{v}$$

– Mass continuity equation: $\rho(\nabla \cdot \mathbf{v}) = 0$

$$\nabla \cdot \mathbf{v} = \left(\frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y} + \frac{\partial v_z}{\partial z} \right) = 0$$

– Mass continuity will be satisfied trivially by using a particle formulation, since each particle has constant mass and particles are neither created nor destroyed



Incompressible Navier-Stokes equations

$$\rho \left[\frac{\partial \mathbf{v}}{\partial t} + \mathbf{v} \cdot \nabla \mathbf{v} \right] = \rho \mathbf{g} - \nabla p + \mu \nabla^2 \mathbf{v}$$

- The material derivative is the derivative along a path with velocity \mathbf{v} . To simulate with particles take the material derivative

$$\rho \frac{D\mathbf{v}}{Dt} = \rho \mathbf{g} - \nabla p + \mu \nabla^2 \mathbf{v}$$

- For a single particle i

$$\frac{dv_i}{dt} = \mathbf{g} - \frac{1}{\rho_i} \nabla p + \frac{\mu}{\rho_i} \nabla^2 \mathbf{v}$$



Incompressible Navier-Stokes equations

- The Navier-Stokes equations are sensitive to scale, so we simulate them at 0.004x scale relative to the physical environment.



Fluids
Navier-Stokes equations
Smoothed Particle Hydrodynamics
OpenCL simulation



Smoothed Particle Hydrodynamics

- [Monaghan 1992] introduced smoothing kernels W

$$A_i(r) = \int A(r')W(r-r',h)dr' \approx \sum_b A(r_b)W(r-r_b,h)$$

- And approximations to terms of the N-S equations

- $\rho_i \approx \sum_j m_j W(r-r_j,h)$ m – mass, r – position, h - radius

$$\frac{\nabla p_i}{\rho_i} \approx \sum_j m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2} \right) \nabla W(r-r_j,h)$$

$$\frac{\mu}{\rho_i} \nabla^2 v_i \approx \frac{\mu}{\rho_i} \sum_j m_j \left(\frac{v_j - v_i}{\rho_j} \right) \nabla^2 W(r-r_j,h)$$



Smoothed Particle Hydrodynamics

- Over time the literature has converged on these W :
 - $w = 0$ at distance h
 - w sums to 1 over sphere of radius h

$$W(r - r_b, h) \equiv \frac{315}{64\pi h^9} (h^2 - \|r - r_b\|^2)^3$$

$$\nabla W(r - r_b, h) \equiv \frac{-45}{\pi h^6} (h - \|r - r_b\|)^2 \frac{r - r_b}{\|r - r_b\|}$$

$$\nabla^2 W(r - r_b, h) \equiv \frac{45}{\pi h^6} (h - \|r - r_b\|)$$



Smoothed Particle Hydrodynamics

- From Navier-Stokes to SPH:

$$\frac{dv_i}{dt} = g - \frac{1}{\rho_i} \nabla p + \frac{\mu}{\rho_i} \nabla^2 v \quad (1)$$

$$\rho_i \approx \sum_j m_j \frac{315}{64\pi h^9} (h^2 - \|r - r_b\|^2)^3 \quad (2)$$

$$\frac{\nabla p_i}{\rho_i} \approx \sum_j m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2} \right) \frac{-45}{\pi h^6} (h - \|r - r_b\|)^2 \frac{r - r_b}{\|r - r_b\|} \quad (3)$$

$$\frac{\mu}{\rho_i} \nabla^2 v_i \approx \frac{\mu}{\rho_i} \sum_j m_j \left(\frac{v_j - v_i}{\rho_j} \right) \frac{45}{\pi h^6} (h - \|r - r_b\|) \quad (4)$$



Fluids
Navier-Stokes equations
Smoothed Particle Hydrodynamics
OpenCL simulation



OpenCL simulation

- Numerical algorithm:
 - density $\rho =$ equation (2).
 - pressure $p = k(\rho - \rho_0)$
 - pressure gradient $\frac{\nabla p_i}{\rho_i} =$ equation (3).
 - viscous term $\frac{\mu}{\rho_i} \nabla^2 v_i =$ equation (4).
 - acceleration = equation (1).
 - numerically integrate velocity, position.



OpenCL simulation

- A naïve algorithm computes interactions among all particles
 - Gives correct result because $W = 0$ for particles beyond the interaction radius
 - But this has complexity $O(n^2)$
 - Need an algorithm that only computes interactions among particles that are within the interaction radius



OpenCL simulation

- A better algorithm partitions space into local regions
 - Divide into voxels of size $2h$ on a side
 - Each particle can only interact with particles in the same voxel, and in immediately adjacent voxels
 - Total search volume = $2 \times 2 \times 2$ voxels
 - Further refinement: compute interactions with a limited number m of particles
 - $m = 32$ works well



OpenCL simulation

- The final algorithm:
 - Organize particles into voxels
 - Compute spatial index from voxel to particles
 - For every particle
 - Examine local region of 2x2x2 voxels
 - Compute interactions with 32 particles



OpenCL simulation

- Interop allows a buffer to be shared between OpenCL and a graphics subsystem.
 - This avoids an expensive round trip to host memory
 - This is crucial for high performance applications
- Due to limits of time we did not implement interop in the graphics code, however we will show you the OpenCL initialization for interop for reference.



OpenCL simulation

- To interop with dx10 include "cl_d3d10.h" and define USE_DX_INTEROP

```
#define USE_DX_INTEROP
#if defined(__APPLE__) || defined(__MACOSX)
#include <OpenCL/cl.hpp>
#include <OpenCL/cl_d3d10.h>
#else
#include <CL/cl.hpp>
#include <CL/cl_d3d10.h>
#endif
```



OpenCL simulation

- To interop with dx10 initialize the OpenCL context:

```
cl_context_properties *cprops;  
cprops = new cl_context_properties[ 6 ];  
cprops[ 0 ] = CL_CONTEXT_D3D10_DEVICE_KHR;  
cprops[ 1 ] = (intptr_t) DXUTGetD3D10Device();  
cprops[ 2 ] = CL_CONTEXT_PLATFORM;  
cprops[ 3 ] = (cl_context_properties)(platformList[0])();  
cprops[ 4 ] = cprops[ 5 ] = 0;  
context = cl::Context( CL_DEVICE_TYPE_GPU, cprops, NULL, NULL,  
&err);
```



OpenCL simulation

- Buffers
 - position, velocity, acceleration – float4
 - particleIndex – uint2
 - sortedPosition, sortedVelocity – float4
 - gridCellIndex, gridCellIndexFixedUp - uint
- Kernels
 - hashParticles, sort, sortPostPass
 - indexx, indexPostPass
 - findNeighbors
 - computeDensityPressure, computeAcceleration, integrate



OpenCL simulation

- The final algorithm:
 - Organize particles into voxels
 - hashParticles, sort, sortPostPass
 - Compute spatial index from voxel to particles
 - indexx, indexPostPass
 - For every particle
 - Examine local region of 2x2x2 voxels
 - findNeighbors
 - Compute interactions with 32 particles
 - computeDensityPressure, computeAcceleration, integrate



OpenCL simulation

- Organize particles into voxels: `global_id(0)=particle id`
 - `hashParticles`:
 - computes a scalar voxel id from position
 - Voxel size $2h \times 2h$
 - stores voxel id in `position.w`;
 - writes `{voxel id,global_id(0)}` to `particleIndex`
 - `sort`:
 - sorts `particleIndex` by voxel id
 - `radixSort` works only on GPU, use `qsort` on CPU
 - `sortPostPass`:
 - rewrite `position,velocity` into `sortedPosition,sortedVelocity` according to order of `particleIndex`



OpenCL simulation

- Compute spatial index from voxel to particles:
global_id(0) = voxel id
 - indexx:
 - computes gridCellIndex(i), index into sortedPosition of first particle in voxel i
 - Binary search in sortedPosition for lowest particle id
 - Leave -1 for empty voxels
 - indexPostPass:
 - Fills in index for empty voxels
 - gridCellIndex(i) = gridCellIndex(i+1) for i empty, i+1 nonempty



OpenCL simulation

- Examine local region of 2x2x2 voxels:
 - findNeighbors:
 - Locates particle in one corner of 2x2x2 voxel set
 - Searches up to 8 voxels until 32 neighbors are found
 - Retains only neighbors within interaction radius
 - Within each voxel search is randomized
 - Necessary to eliminate biasing artifacts
 - Specifically, compute random offset within voxel, then proceed sequentially
 - Alternate sequential directions according to odd/evenness of particle



OpenCL simulation

- Compute interactions with 32 particles:
 - computeDensityPressure:
 - Equation (2) followed by $p = k(\rho - \rho_0)$
 - computeAcceleration:
 - Equations (3), (4), (1)
 - integrate:
 - Semi-implicit Euler integration
 - $v = v + dt a$, position = position + dt v
 - Boundary conditions prevent particle escape



Summary

- Fluids
 - Governed by pressure, velocity
- Navier-Stokes equations
 - Incompressible equations, material derivative
- Smoothed Particle Hydrodynamics
 - Smoothing kernel approximations
 - Approximate ρ , ∇p , $\nabla^2 v$
- OpenCL simulation
 - Organize into voxels, create voxel index, compute equations (2), (3), (4), (1), integrate



Questions and Answers

Visit the OpenCL Zone on developer.amd.com

<http://developer.amd.com/zones/OpenCLZone/>

- Tutorials, developer guides, and more
- OpenCL Programming Webinars page includes:
 - Schedule of upcoming webinars
 - On-demand versions of this and past webinars
 - Slide decks of this and past webinars
 - Source code for Smoothed Particle Hydrodynamics webinar



Trademark Attribution

AMD, the AMD Arrow logo and combinations thereof are trademarks of Advanced Micro Devices, Inc. in the United States and/or other jurisdictions. Other names used in this presentation are for identification purposes only and may be trademarks of their respective owners.

©2009 Advanced Micro Devices, Inc. All rights reserved.

