Realtime GPU-Accelerated Smoothed Particle Hydrodynamics with Bidirectional Rigid Body Coupling

Shaun Silson

University of Cape Town

27 April 2013

Wait... WHAT!

Shaun Silson

Wait... WHAT!

Watch This

You said something about a GPU?

Shaun Silson

You said something about a GPU?

Graphics Processing Unit

And the rest?

Shaun Silson

And the rest?

Don't worry we'll get to that...

Outline

Outline

1 Background

2 Theory

Outline

Shaun Silson

Why?

- Special effects are popular in games and movies.

Why?

- Special effects are popular in games and movies.
- There's always demand for better performance.

Why?

Shaun Silson

- Special effects are popular in games and movies.
- There's always demand for better performance.
- GPUs are widely available and fairly cheap.

Why?

- Special effects are popular in games and movies.
- There's always demand for better performance.
- GPUs are widely available and fairly cheap.
- It's FUN!

Fluids on GPU

Shaun Silson

Interactive SPH Simulation and Rendering on the GPU. *Prashant Goswami*(University of Zurich), 2010.

Interactive SPH Simulation and Rendering on the GPU.
Prashant Goswami(University of Zurich), 2010.

Fluids on GPU

Interactive SPH Simulation and Rendering on the GPU.
Prashant Goswami(University of Zurich), 2010.

- GPU Based

Fluids on GPU

Interactive SPH Simulation and Rendering on the GPU.
Prashant Goswami (University of Zurich), 2010.

- GPU Based
- Interactive Framerates

Fluids on GPU

Interactive SPH Simulation and Rendering on the GPU.
Prashant Goswami(University of Zurich), 2010.

- GPU Based
- Interactive Framerates
- Realistic Rendering

Fluids on GPU

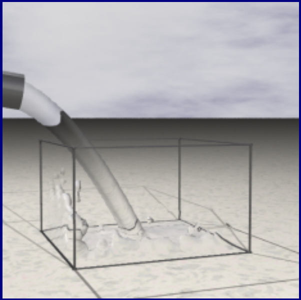
Interactive SPH Simulation and Rendering on the GPU.
Prashant Goswami (University of Zurich), 2010.

- GPU Based
- Interactive Framerates
- Realistic Rendering
- No Rigid Bodies!

Fluids on GPU

Interactive SPH Simulation and Rendering on the GPU.
Prashant Goswami (University of Zurich), 2010.

- GPU Based
- Interactive Framerates
- Realistic Rendering
- No Rigid Bodies!



Fluids with Rigid Bodies

Shaun Silson

Fluids with Rigid Bodies

Versatile Rigid-Fluid Coupling for Incompressible SPH.
Nadir Akinci(University of Freiburg), 2012.

Fluids with Rigid Bodies

Versatile Rigid-Fluid Coupling for Incompressible SPH.
Nadir Akinci (University of Freiburg), 2012.

Fluids with Rigid Bodies

Versatile Rigid-Fluid Coupling for Incompressible SPH.
Nadir Akinci (University of Freiburg), 2012.

- Rigid Bodies

Fluids with Rigid Bodies

Versatile Rigid-Fluid Coupling for Incompressible SPH.
Nadir Akinci(University of Freiburg), 2012.

- Rigid Bodies
- Not GPU!

Fluids with Rigid Bodies

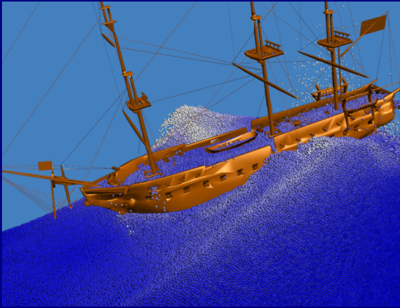
Versatile Rigid-Fluid Coupling for Incompressible SPH.
Nadir Akinci(University of Freiburg), 2012.

- Rigid Bodies
- Not GPU!
- Not Realtime!

Fluids with Rigid Bodies

Versatile Rigid-Fluid Coupling for Incompressible SPH.
Nadir Akinci (University of Freiburg), 2012.

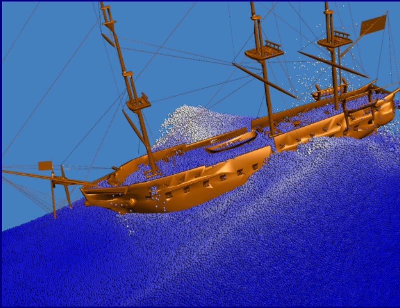
- Rigid Bodies
- Not GPU!
- Not Realtime!



Fluids with Rigid Bodies

Versatile Rigid-Fluid Coupling for Incompressible SPH.
Nadir Akinci(University of Freiburg), 2012.

- Rigid Bodies
- Not GPU!
- Not Realtime!



Wheeeeeeeeeee!!!

Time for Another Video...

Time for Another Video...

YAY!

So... now what?

We have

So... now what?

We have GPU with no rigids

So... now what?

We have GPU with no rigids and Non-GPU with rigids...

So... now what?

We have GPU with no rigids and Non-GPU with rigids...

Combine Them!

Fluids v.3

Fluids v.3

A Large Scale, Open Source Fluid Simulator Using the Smooth Particle Hydrodynamics Method.
Rama C. Hoetzlein, 2012.

Fluids v.3

A Large Scale, Open Source Fluid Simulator Using the Smooth Particle Hydrodynamics Method.
Rama C. Hoetzlein, 2012.

A Large Scale, Open Source Fluid Simulator Using the Smooth Particle Hydrodynamics Method.
Rama C. Hoetzlein, 2012.

Let's See it First

Fluids v.3

A Large Scale, Open Source Fluid Simulator Using the Smooth Particle Hydrodynamics Method.
Rama C. Hoetzlein, 2012.

Let's See it First

- GPU Based

Fluids v.3

A Large Scale, Open Source Fluid Simulator Using the Smooth Particle Hydrodynamics Method.
Rama C. Hoetzlein, 2012.

Let's See it First

- GPU Based ✓

Fluids v.3

A Large Scale, Open Source Fluid Simulator Using the Smooth Particle Hydrodynamics Method.
Rama C. Hoetzlein, 2012.

Let's See it First

- GPU Based ✓
- Realtime

Fluids v.3

A Large Scale, Open Source Fluid Simulator Using the Smooth Particle Hydrodynamics Method.
Rama C. Hoetzlein, 2012.

Let's See it First

- GPU Based ✓
- Realtime ✓

Fluids v.3

A Large Scale, Open Source Fluid Simulator Using the Smooth Particle Hydrodynamics Method.
Rama C. Hoetzlein, 2012.

Let's See it First

- GPU Based ✓
- Realtime ✓
- Open Source

Fluids v.3

A Large Scale, Open Source Fluid Simulator Using the Smooth Particle Hydrodynamics Method.
Rama C. Hoetzlein, 2012.

Let's See it First

- GPU Based ✓
- Realtime ✓
- Open Source ✓

Fluids v.3

A Large Scale, Open Source Fluid Simulator Using the Smooth Particle Hydrodynamics Method.
Rama C. Hoetzlein, 2012.

Let's See it First

- GPU Based ✓
- Realtime ✓
- Open Source ✓
- No Rigid Bodies

Fluids v.3

A Large Scale, Open Source Fluid Simulator Using the Smooth Particle Hydrodynamics Method.
Rama C. Hoetzlein, 2012.

Let's See it First

- GPU Based ✓
- Realtime ✓
- Open Source ✓
- No Rigid Bodies (yet...)

Fluids v.3

A Large Scale, Open Source Fluid Simulator Using the Smooth Particle Hydrodynamics Method.
Rama C. Hoetzlein, 2012.

Let's See it First

- GPU Based ✓
- Realtime ✓
- Open Source ✓
- No Rigid Bodies (yet...)

I plan to incorporate the method described by Akinci to implement rigid body interaction with fluids *on the GPU.*

But How Does it *Actually* Work?

Smoothed Particle Hydrodynamics

$$\rho_i = \sum_j m_j W(r_i - r_j, h)$$

Smoothed Particle Hydrodynamics

$$\rho_i = \sum_j m_j W(r_i - r_j, h)$$

Eeeeeeek!

Smoothed Particle Hydrodynamics

$$\rho_i = \sum_j m_j W(r_i - r_j, h)$$

Let's break it down:

Smoothed Particle Hydrodynamics

$$\rho_i = \sum_j m_j W(r_i - r_j, h)$$

Let's break it down:

ρ_i density of particle i

Smoothed Particle Hydrodynamics

$$\rho_i = \sum_j m_j W(r_i - r_j, h)$$

Let's break it down:

ρ_i density of particle i

m_j mass of particle j

Smoothed Particle Hydrodynamics

$$\rho_i = \sum_j m_j W(r_i - r_j, h)$$

Let's break it down:

ρ_i density of particle i

m_j mass of particle j

W **smoothing kernel**

Smoothed Particle Hydrodynamics

$$\rho_i = \sum_j m_j W(r_i - r_j, h)$$

Let's break it down:

ρ_i	density of particle i
m_j	mass of particle j
W	**smoothing kernel**
$r_i - r_j$	distance between particles i and j

Smoothed Particle Hydrodynamics

$$\rho_i = \sum_j m_j W(r_i - r_j, h)$$

Let's break it down:

ρ_i	density of particle i
m_j	mass of particle j
W	**smoothing kernel**
$r_i - r_j$	distance between particles i and j
h	smoothing cutoff radius

The Smoothing Kernel:

Must satisfy the following properties:

The Smoothing Kernel:

Must satisfy the following properties:

$\int_\Omega W(r,h)dr = 1$ Normalized

The Smoothing Kernel:

Must satisfy the following properties:

$$\int_\Omega W(r,h)dr = 1 \quad \text{Normalized}$$

$$W(r,h) \geq 0 \quad \text{Positive}$$

The Smoothing Kernel:

Must satisfy the following properties:

$$\int_\Omega W(r,h)dr = 1 \quad \text{Normalized}$$

$$W(r,h) \geq 0 \quad \text{Positive}$$

$$W(r,h) = W(-r,h) \quad \text{Symmetric}$$

Shaun Silson

The Smoothing Kernel:

Must satisfy the following properties:

$$\int_\Omega W(r,h)dr = 1 \quad \text{Normalized}$$

$$W(r,h) \geq 0 \quad \text{Positive}$$

$$W(r,h) = W(-r,h) \quad \text{Symmetric}$$

The Smoothing Kernel:

Must satisfy the following properties:

$$\int_\Omega W(r,h)dr = 1 \quad \text{Normalized}$$
$$W(r,h) \geq 0 \quad \text{Positive}$$
$$W(r,h) = W(-r,h) \quad \text{Symmetric}$$

(r is inter-particle distance and h is cutoff radius)

Shaun Silson

The Smoothing Kernel:

Just think of it as a Gaussian curve...

Shaun Silson

The Smoothing Kernel:

Just think of it as a Gaussian curve...

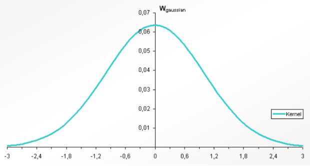


Diagram from Kelager[2006]*

Smoothed Particle Hydrodynamics

Shaun Silson

The Smoothing Kernel:

Just think of it as a Gaussian curve...

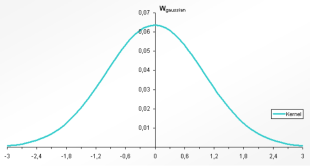


Diagram from Kelager[2006]*

The main point is further particles have less influence...

The Smoothing Kernel:

Just think of it as a Gaussian curve...

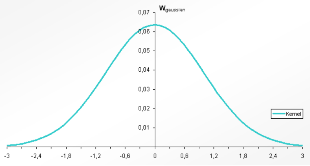


Diagram from Kelager[2006]*

The main point is further particles have less influence...
and no influence outside the cutoff radius.

The Smoothing Kernel:

Just think of it as a Gaussian curve...

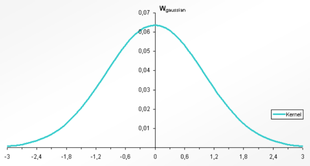


Diagram from Kelager[2006]*

The main point is further particles have less influence...
and no influence outside the cutoff radius.
The cutoff radius h is very important!

The Smoothing Kernel:

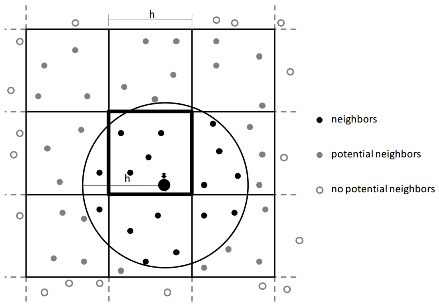


Diagram from: Realtime particle-based fluid simulation, *Stefan Auer*

The cutoff radius h is very important!
(we will come back to it in the performance section)

The Algorithm:

- To calculate the density of each particle we sum the masses of it's neighbours

The Algorithm:

- To calculate the density of each particle we sum the masses of it's neighbours (weighted by the smoothing kernel W).

The Algorithm:

- To calculate the density of each particle we sum the masses of it's neighbours (weighted by the smoothing kernel W).
- More particles nearby means more density

Shaun Silson

The Algorithm:

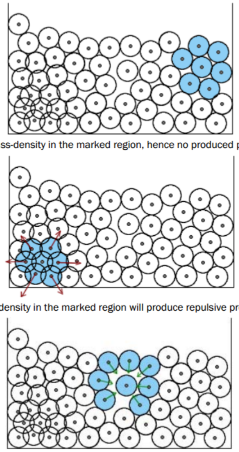
- To calculate the density of each particle we sum the masses of it's neighbours (weighted by the smoothing kernel W).
- More particles nearby means more density (duh!)

Smoothed Particle Hydrodynamics

The Algorithm:

- To calculate the density of each particle we sum the masses of it's neighbours (weighted by the smoothing kernel W).
- More particles nearby means more density (duh!)

* Diagram from: Lagrangian Fluid Dynamics Using Smoothed Particle Hydrodynamics, *Micky Kelager* (2006)



a) Balanced mass-density in the marked region, hence no produced pressure forces.

b) High mass-density in the marked region will produce repulsive pressure forces.

c) Low mass-density in the marked region will produce attractive pressure forces.

The Algorithm:

- Once we have the *density* (ρ) surrounding a particle we can calculate the *pressure* (p) exerted on it.

The Algorithm:

- Once we have the *density* (ρ) surrounding a particle we can calculate the *pressure* (p) exerted on it.
- Using the pressure we can work out the *force* (f).

The Algorithm:

- Once we have the *density* (ρ) surrounding a particle we can calculate the *pressure* (p) exerted on it.
- Using the pressure we can work out the *force* (f).
- From the force we calculate it's *motion*.

The Algorithm:

- Once we have the *density* (ρ) surrounding a particle we can calculate the *pressure* (p) exerted on it.
- Using the pressure we can work out the *force* (f).
- From the force we calculate it's *motion*.

Do this for every particle...

Shaun Silson

The Algorithm:

- Once we have the *density* (ρ) surrounding a particle we can calculate the *pressure* (p) exerted on it.
- Using the pressure we can work out the *force* (f).
- From the force we calculate it's *motion*.

Do this for every particle... every frame...

The Algorithm:

- Once we have the *density* (ρ) surrounding a particle we can calculate the *pressure* (p) exerted on it.
- Using the pressure we can work out the *force* (f).
- From the force we calculate it's *motion*.

Do this for every particle... every frame...
And you have a simulation!

BUT!

BUT!

This gets more expensive the more particles you have...

BUT!

This gets more expensive the more particles you have...

And we want **MILLIONS** of particles!

BUT!

This gets more expensive the more particles you have...

And we want **MILLIONS** of particles!

We'll deal with that in the final section on perfomance.

BUT!

This gets more expensive the more particles you have...

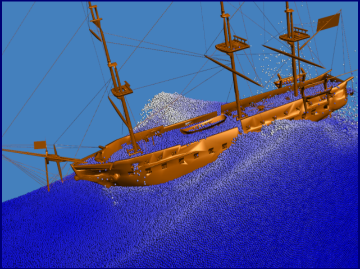
And we want **MILLIONS** of particles!

We'll deal with that in the final section on perfomance.
For now though...

What About Those Rigid Bodies?

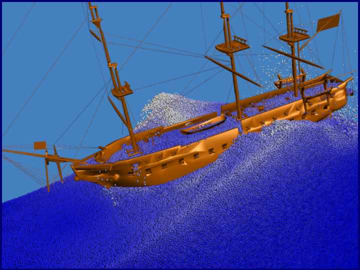
What About Those Rigid Bodies?

Remember this image:

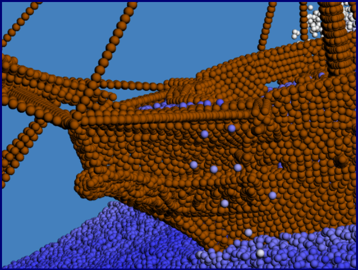


What About Those Rigid Bodies?

Remember this image:

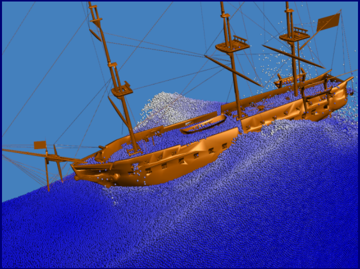


Let's take a closer look:

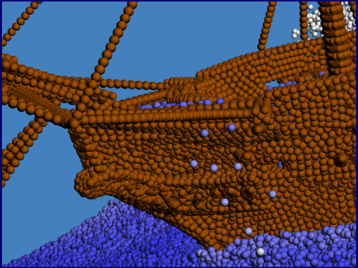


What About Those Rigid Bodies?

Remember this image:



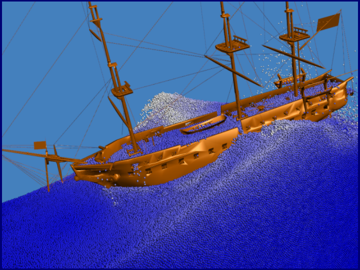
Let's take a closer look:



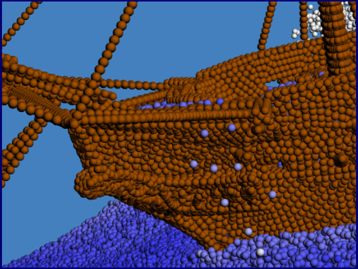
See how the boat is made up of particles?

What About Those Rigid Bodies?

Remember this image:



Let's take a closer look:



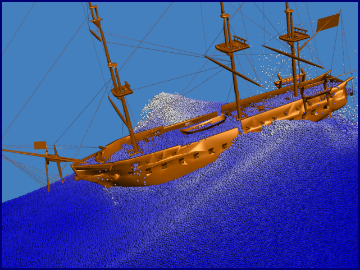
See how the boat is made up of particles?

Those are fluid particles...

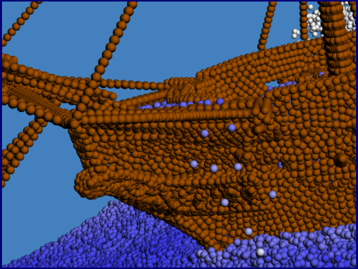
What About Those Rigid Bodies?

Shaun Silson

Remember this image:



Let's take a closer look:



See how the boat is made up of particles?

Those are fluid particles... attached to a rigid body!

The Boat is Made of Fluid Particles... So?

The Boat is Made of Fluid Particles... So?

Oh yea of little faith :P

- We can use the same algorithm described earlier to calculate the force applied to the boat.

The Boat is Made of Fluid Particles... So?

Oh yea of little faith :P

- We can use the same algorithm described earlier to calculate the force applied to the boat.
- We sum the forces from the surrounding fluid particles and apply it to the rigid body.

The Boat is Made of Fluid Particles... So?

Oh yea of little faith :P

- We can use the same algorithm described earlier to calculate the force applied to the boat.
- We sum the forces from the surrounding fluid particles and apply it to the rigid body.
- In between simulation steps: rather than letting the attached particles flow, we move the rigid body and update there relative positions.

The Boat is Made of Fluid Particles... So?

Oh yea of little faith :P

- We can use the same algorithm described earlier to calculate the force applied to the boat.
- We sum the forces from the surrounding fluid particles and apply it to the rigid body.
- In between simulation steps: rather than letting the attached particles flow, we move the rigid body and update there relative positions.
- The body is run in Bullet Physics.

The Boat is Made of Fluid Particles... So?

Oh yea of little faith :P

- We can use the same algorithm described earlier to calculate the force applied to the boat.
- We sum the forces from the surrounding fluid particles and apply it to the rigid body.
- In between simulation steps: rather than letting the attached particles flow, we move the rigid body and update there relative positions.
- The body is run in Bullet Physics.
- That's it!

Really?

Really?

No, not really...

Really?

No, not really... We haven't mentioned viscosity!

Really?

No, not really... We haven't mentioned viscosity!
But the principle is similar:

Really?

No, not really... We haven't mentioned viscosity!
But the principle is similar:

1. For each particle
2. Accumulate force contributions from neighbours
3. Weight with a smoothing kernel
4. Update

See bonus material at the end for more detail.

Shaun Silson

CPU vs GPU

CPU vs GPU

A CPU has 4 to 8 powerful cores...

CPU vs GPU

A CPU has 4 to 8 powerful cores... A GPU has *hundreds* of simpler ones.

CPU vs GPU

A CPU has 4 to 8 powerful cores... A GPU has *hundreds* of simpler ones.

Remember how we need to process *millions* of particles...

CPU vs GPU

Shaun Silson

Remember we need to process *millions* of particles...

- A CPU has to go through them one by one.

CPU vs GPU

Shaun Silson

Remember we need to process *millions* of particles...

- A CPU has to go through them one by one.
- But a GPU can split up the work!

CPU vs GPU

Remember we need to process *millions* of particles...

- A CPU has to go through them one by one.
- But a GPU can split up the work!
- Only because *the same thing* is done to each.

CPU vs GPU

Remember we need to process *millions* of particles...

- A CPU has to go through them one by one.
- But a GPU can split up the work!
- Only because *the same thing* is done to each.
- (It doesn't work this way for all types of problem!)

Parallel is great...

Parallel is great...

But to get millions of particles...

Parallel is great...

But to get millions of particles... *in realtime*

We still have to be clever!

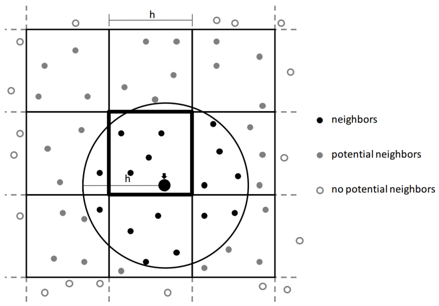
Let's go back to that cutoff radius...

Let's go back to that cutoff radius...

I did say it was important

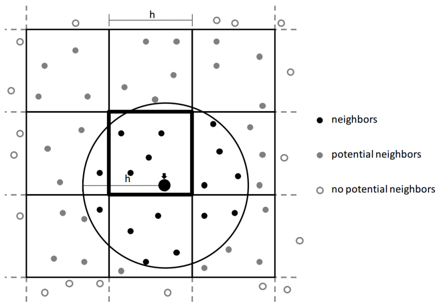
Let's go back to that cutoff radius...

I did say it was important, so here it is again:



Let's go back to that cutoff radius...

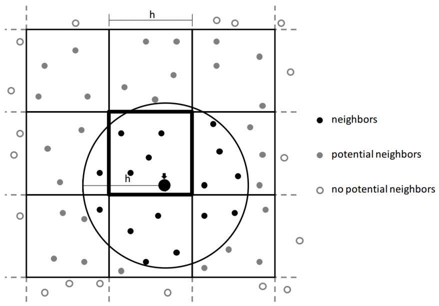
I did say it was important, so here it is again:



We only want to get force from neighbouring particles.

Let's go back to that cutoff radius...

I did say it was important, so here it is again:



We only want to get force from neighbouring particles.

How do we do that?

Neighbour Search

Neighbour Search

Most expensive part of the simulation!

Dumb Way:

- For each particle.

Neighbour Search

Most expensive part of the simulation!

Dumb Way:

- For each particle.
- Test every other particle.

Neighbour Search

Most expensive part of the simulation!

Dumb Way:

- For each particle.
- Test every other particle.
- Ignore if too far away.

Neighbour Search

Shaun Silson

Most expensive part of the simulation!

Dumb Way:

- For each particle.
- Test every other particle.
- Ignore if too far away.
- $O(n^2)$

Neighbour Search

Most expensive part of the simulation!

Dumb Way:

- For each particle.
- Test every other particle.
- Ignore if too far away.
- $O(n^2)$

Neighbour Search

Most expensive part of the simulation!

Dumb Way:

- For each particle.
- Test every other particle.
- Ignore if too far away.
- $O(n^2)$

Clever Way:

- Z-Indexing

Neighbour Search

Most expensive part of the simulation!

Dumb Way:

- For each particle.
- Test every other particle.
- Ignore if too far away.
- $O(n^2)$

Clever Way:

- Z-Indexing
- Groups potential neighbours each timestep.

Neighbour Search

Most expensive part of the simulation!

Dumb Way:

- For each particle.
- Test every other particle.
- Ignore if too far away.
- $O(n^2)$

Clever Way:

- Z-Indexing
- Groups potential neighbours each timestep.
- Much fewer tests.

Neighbour Search

Most expensive part of the simulation!

Dumb Way:

- For each particle.
- Test every other particle.
- Ignore if too far away.
- $O(n^2)$

Clever Way:

- Z-Indexing
- Groups potential neighbours each timestep.
- Much fewer tests.
- Also parallelizable!

Neighbour Search

Most expensive part of the simulation!

Dumb Way:

- For each particle.
- Test every other particle.
- Ignore if too far away.
- $O(n^2)$

Clever Way:

- Z-Indexing
- Groups potential neighbours each timestep.
- Much fewer tests.
- Also parallelizable!

Neighbour Search

Most expensive part of the simulation!

Dumb Way:

- For each particle.
- Test every other particle.
- Ignore if too far away.
- $O(n^2)$

Clever Way:

- Z-Indexing
- Groups potential neighbours each timestep.
- Much fewer tests.
- Also parallelizable!

(for more details see [Goswami2010] from earlier)

Timestep Size

Shaun Silson

Timestep Size

Biggest performance tradeoff!

Timestep Size

Shaun Silson

Biggest performance tradeoff!

Smaller:

- More Stable

Timestep Size

Biggest performance tradeoff!

Smaller:

- More Stable
- Slower

Timestep Size

Biggest performance tradeoff!

Smaller:

- More Stable
- Slower

Timestep Size

Biggest performance tradeoff!

Smaller:

- More Stable
- Slower

Larger:

- Much Faster

What can we do?

Timestep Size

Biggest performance tradeoff!

Smaller:

- More Stable
- Slower

Larger:

- Much Faster
- Crazy Artifacts

What can we do?

PCISPH

Shaun Silson

Shaun Silson

Predictive-Corrective

PCISPH

Predictive-Corrective Incompressible

PCISPH

Predictive-Corrective Incompressible SPH.

How does it work?

Water is uniformly dense so no fluctuations expected.

PCISPH

Predictive-Corrective Incompressible SPH.

How does it work?

Water is uniformly dense so no fluctuations expected.
But with bigger timesteps they are more likely.

PCISPH

Predictive-Corrective Incompressible SPH.

How does it work?

Water is uniformly dense so no fluctuations expected.
But with bigger timesteps they are more likely.
So in each step:

1. Predict particle densities before applying velocity.

For more details see Predictive-Corrective Incompressible SPH, *Barbara Solenthaler*, 2009

PCISPH

Predictive-Corrective Incompressible SPH.

How does it work?

Water is uniformly dense so no fluctuations expected.
But with bigger timesteps they are more likely.
So in each step:

1. Predict particle densities before applying velocity.
2. If artifacts occur, correct using a solver.

For more details see Predictive-Corrective Incompressible SPH, *Barbara Solenthaler*, 2009

PCISPH

Predictive-Corrective Incompressible SPH.

How does it work?

Water is uniformly dense so no fluctuations expected.
But with bigger timesteps they are more likely.
So in each step:

1. Predict particle densities before applying velocity.
2. If artifacts occur, correct using a solver.
3. Apply velocity updates.

For more details see Predictive-Corrective Incompressible SPH, *Barbara Solenthaler*, 2009

Summary

Summary

- Incorporate rigid bodies into Hoetzlein's Fluids v.3

And that's all!

Summary

- Incorporate rigid bodies into Hoetzlein's Fluids v.3
- This will be based on the method from Akinci, 2012

And that's all!

Summary

- Incorporate rigid bodies into Hoetzlein's Fluids v.3
- This will be based on the method from Akinci, 2012
- Speed it up using GPU parallelization

And that's all!

Summary

- Incorporate rigid bodies into Hoetzlein's Fluids v.3
- This will be based on the method from Akinci, 2012
- Speed it up using GPU parallelization
- Speed up more with fast neighbour-search and PCISPH

And that's all!