Universität Hamburg
Praktikum „Parallele Programmierung”

FluidSim

Parallel fluid particle simulation and visualization

by

Paul Bienkowski

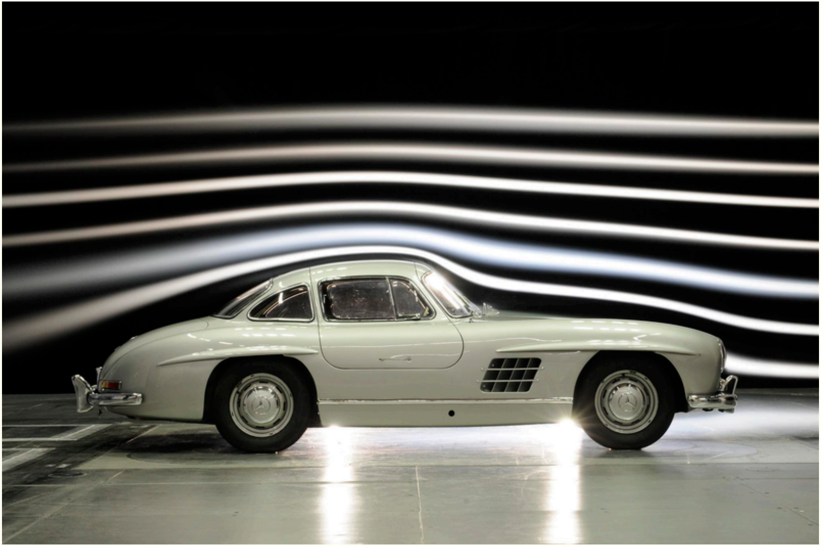
2bienkow@informatik.uni-hamburg.de

September 24, 2014

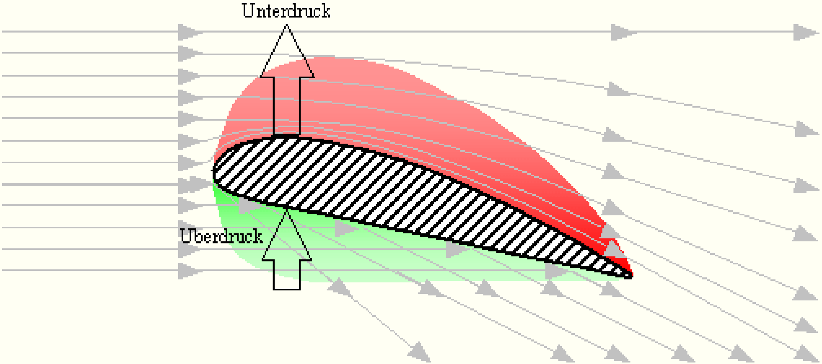
Contents

Introduction

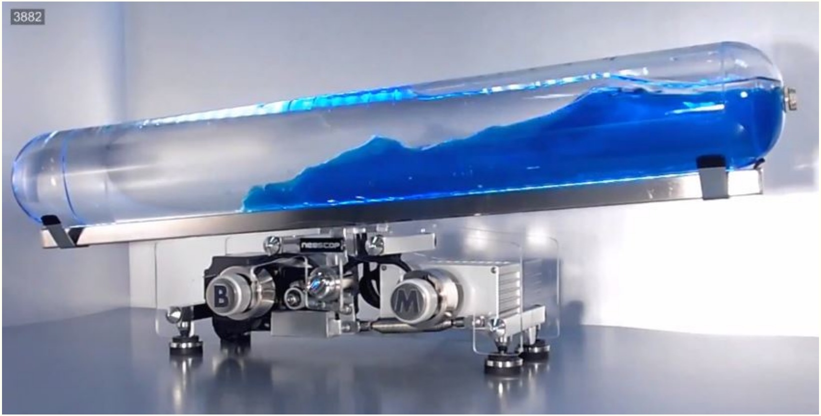
Motivation



Motivation



Motivation



Motivation

Idea

Simulate 2D-particles that repel each other

Motivation

Idea

Simulate 2D-particles that repel each other

Goal

Simulate an airplane wing and measure lift

Motivation

Idea

Simulate 2D-particles that repel each other

Goal

Simulate an airplane wing and measure lift

Technology

C++11, MPI, OpenMP, SFML

Particle Model

Each particle has 3 basic properties:

1. position
2. velocity
3. force

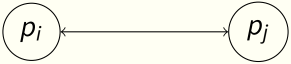
Particle Model

Each particle has 3 basic properties:

1. **position**
2. **velocity**
3. force

Only position and velocity need to be stored across iterations, force is recomputed every iteration.

Particle Model



Any two particles repel each other:

$$force_i := \sum_j \mathbf{force}(\left|p_i - p_j\right|) \cdot norm(p_i - p_j)$$

The force on a particle affects its velocity:

$$velocity_i := velocity_i + force_i \cdot dt$$

The velocity of a particle affects its position:

$$position_i := position_i + velocity_i \cdot dt$$

Force Model

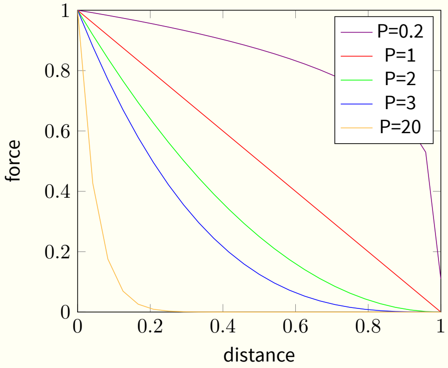
$$\textbf{force}(x) = \begin{cases} F \cdot \left(1 - \frac{x-T}{D}\right)^P & \text{for } 0 \leq x \leq D + T \\ 0 & \text{otherwise} \end{cases}$$

where

- x is the distance between the particles.
- F is the force strength factor
- D is the influence distance
- T is the distance threshold (particle radius)
- P is the force power

Force Model

$$\textbf{force}(x) = \begin{cases} F \cdot \left(1 - \frac{x-T}{D}\right)^P & \text{for } 0 \leq x \leq D + T \\ 0 & \text{otherwise} \end{cases}$$



Changing the Force Power (P)

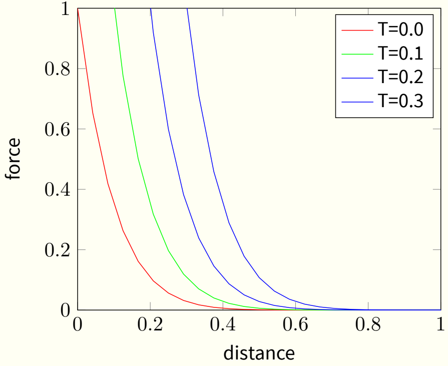
$F = 1$

$D = 1$

$T = 0$

Force Model

$$\textbf{force}(x) = \begin{cases} F \cdot \left(1 - \frac{x-T}{D}\right)^P & \text{for } 0 \leq x \leq D + T \\ 0 & \text{otherwise} \end{cases}$$



Changing the Distance Threshold (T)

$F = 1$

$D = 1$

$P = 10$

Force Model

$$\textbf{force}(x) = \begin{cases} F \cdot \left(1 - \frac{x-T}{D}\right)^P & \text{for } 0 \leq x \leq D + T \\ 0 & \text{otherwise} \end{cases}$$

I found these values work for the wing simulation:

$$\begin{aligned} D &= 0.001 \\ T &= 0.06 \\ P &= 1 \\ F &= 20 \end{aligned}$$

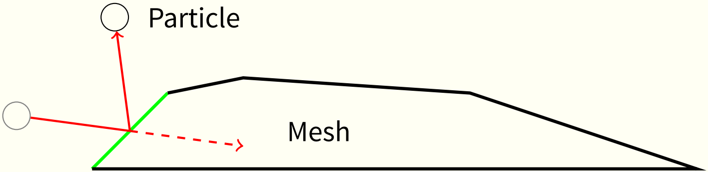
Meshes

A mesh is a simple polygon, each segment (line) is checked for collision with every particle.

Meshes

A mesh is a simple polygon, each segment (line) is checked for collision with every particle.

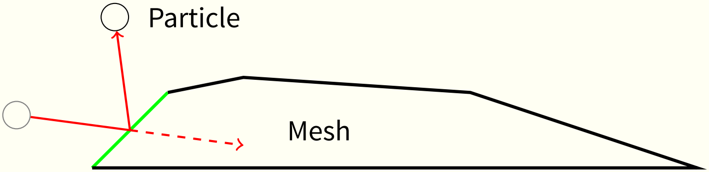
Simple linear algebra calculations are made for reflecting particles off mesh segments.



Meshes

A mesh is a simple polygon, each segment (line) is checked for collision with every particle.

Simple linear algebra calculations are made for reflecting particles off mesh segments.



This creates a force on the mesh.

The Simulator

Parallelization

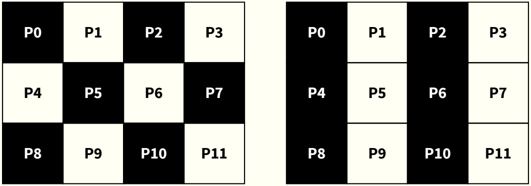
- each particle has to be updated (simple `for` loop)
  $\rightarrow$ threading trivial with OpenMP
- particles can be distributed across multiple processes
  $\rightarrow$ Domain Grid

Parallelization

- each particle has to be updated (simple `for` loop)
  $\rightarrow$ threading trivial with OpenMP
- particles can be distributed across multiple processes
  $\rightarrow$ Domain Grid

P0	P1	P2	P3
P4	P5	P6	P7
P8	P9	P10	P11

Synchronization



(a) Checkerboard (b) Stripes

Modes of Domain coloring

	Mode	Sending	Receiving	Directions
1.	Checkerboard	black	white	N, E, S, W
2.	Checkerboard	white	black	N, E, S, W
3.	Stripes	black	white	NE, SE, SW, NW
4.	Stripes	white	black	NE, SE, SW, NW

Data output

```
0000000  0020 0000 0001 0000 0004 0000 2710 0000
0000010  1387 0000 0000 0000 1389 0000 0000 0000
0000020  8a72 d187 c4bb 3fa1 73d1 4e75 d95f 3fbe
0000030  d3f4 2c9a cf42 3fbc b890 9e75 5a61 3fc1
...
00013a7  9aa6 af8e c353 3fc7 1df2 1539 3ec1 3fc3
00013b7  c065 7825 f853 3fb3 717a c31f 1f55 3fc3
...
```

- Header length: 32 bytes
- Iteration number: 1
- Number of processes: 4
- Particle count: 10000
- Particle count by process: 4999 / 0 / 5001 / 0
- Particle positions of P1 $(x_1, y_1, x_2, y_2, \ldots)$
- Particle velocities of P1

The Visualizer

Basic Technology

- SFML for window, input, rendering (OpenGL inside)
- Load Iterations into memory
- Play/Pause/*Live*
- Different display modes (coloring of particles)

Data Transfer

How is data transfered from Simulator to Visualizer?

Data Transfer

How is data transfered from Simulator to Visualizer?

- Socket/Network/MPI?

Data Transfer

How is data transfered from Simulator to Visualizer?

- Socket/Network/MPI? $\rightarrow$ too complicated

Data Transfer

How is data transfered from Simulator to Visualizer?

- Socket/Network/MPI? $\rightarrow$ too complicated
- SSHFS ☺

Data Transfer

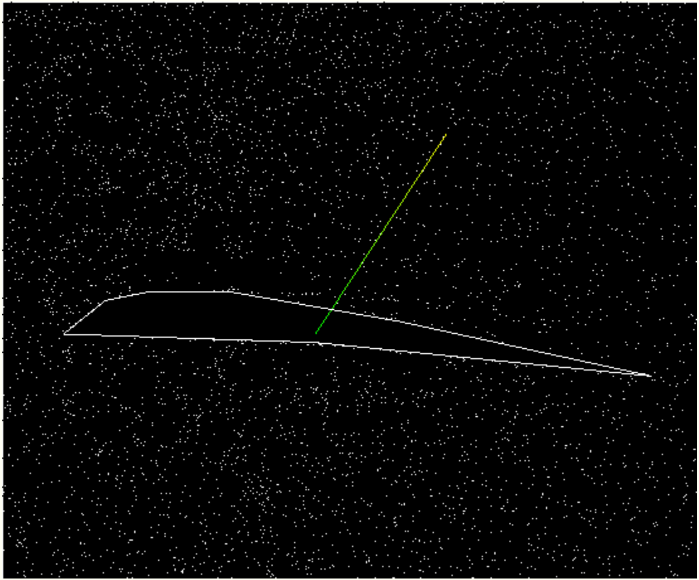
How is data transfered from Simulator to Visualizer?

- Socket/Network/MPI? $\rightarrow$ too complicated
- SSHFS ☺
- `status` file contains metadata
  - number of iterations
  - grid size
- visualizer reads status in regular intervals

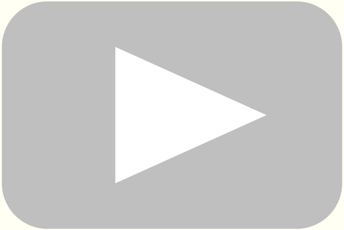
Results

Did it work?

Did it work?



Live Demonstration

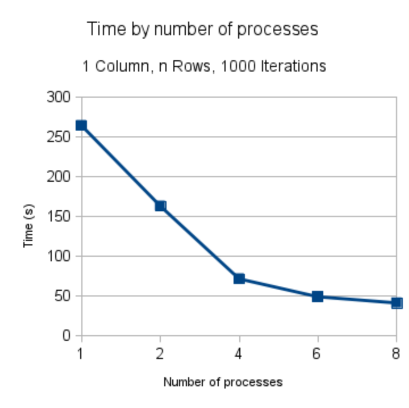


Performance

All measurements were made with IO disabled, no particle data was recorded.

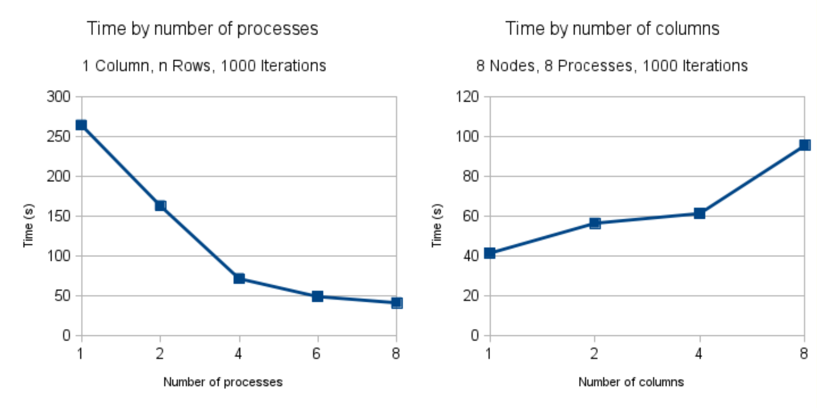
Performance

All measurements were made with IO disabled, no particle data was recorded.



Performance

All measurements were made with IO disabled, no particle data was recorded.



Difficulties

- MPI file input/output was hard to get working, miscalculated seek offsets etc.

Difficulties

- MPI file input/output was hard to get working, miscalculated seek offsets etc.
- lots of segmentation faults and uninitialized values ☺

```
Cannot insert: Count 0xdeafbeed >= Size 0xdeafbeed.
fluidsim: Quickset.hpp:14: Assertion ‘count < size’ failed.
Aborted.

Segmentation fault.
(gdb) frame 3
(gdb) print buf
$1 = 0xdeafbeeddeafbeed;
```

Difficulties

- MPI file input/output was hard to get working, miscalculated seek offsets etc.
- lots of segmentation faults and uninitialized values ☺

```
Cannot insert: Count 0xdeafbeed >= Size 0xdeafbeed.
fluidsim: Quickset.hpp:14: Assertion ‘count < size’ failed.
Aborted.

Segmentation fault.
(gdb) frame 3
(gdb) print buf
$1 = 0xdeafbeeddeafbeed;
```

- 2D collisions are not *that* trivial

Difficulties

- MPI file input/output was hard to get working, miscalculated seek offsets etc.
- lots of segmentation faults and uninitialized values ☺

```
Cannot insert: Count 0xdeafbeed >= Size 0xdeafbeed.
fluidsim: Quickset.hpp:14: Assertion ‘count < size’ failed.
Aborted.

Segmentation fault.
(gdb) frame 3
(gdb) print buf
$1 = 0xdeafbeeddeafbeed;
```

- 2D collisions are not *that* trivial
- the model (uplift) did not work out until I implemented surface damping

Problems

- software not optimized for RAM usage
  $\rightarrow$ can't run more than 6 processes locally ☹

Problems

- software not optimized for RAM usage
  $\rightarrow$ can't run more than 6 processes locally ☹
- `memcpy` is slow

Problems

- software not optimized for RAM usage
  $\rightarrow$ can't run more than 6 processes locally ☹
- `memcpy` is slow
- 16 send/receive operations on a 12-node cluster, probably room for improvement,

Problems

- software not optimized for RAM usage
  $\rightarrow$ can't run more than 6 processes locally ☹
- `memcpy` is slow
- 16 send/receive operations on a 12-node cluster, probably room for improvement,

Problems

- software not optimized for RAM usage
  $\rightarrow$ can't run more than 6 processes locally ☹
- `memcpy` is slow
- 16 send/receive operations on a 12-node cluster, probably room for improvement, *however* this method scales to every cluster size

Conclusion

Conclusion

- Yes, it works!
- Even in real-time!
- Even though $\mathcal{O}(n^2 + nm)$ with $n \in \mathcal{O}(10000)$
- It looks kind of fancy...
- I learned a lot.

Further works

- Load Balancer, based on number of particles in rows/columns

Further works

- Load Balancer, based on number of particles in rows/columns
- SIMD

Further works

- Load Balancer, based on number of particles in rows/columns
- SIMD
- “Ball” particle model (elastic collision of circle shapes)

Further works

- Load Balancer, based on number of particles in rows/columns
- SIMD
- “Ball” particle model (elastic collision of circle shapes)
- different World Scenarios / presets (gravity, water, ...)

Further works

- Load Balancer, based on number of particles in rows/columns
- SIMD
- “Ball” particle model (elastic collision of circle shapes)
- different World Scenarios / presets (gravity, water, ...)
- animation export

Thank you for your attention!

Questions?