Particle-Based Fluid Simulation
– SPH(Smoothed Particle Hydrodynamics)

CS231 Final presentation

Presented by Lingli Wang

Mar. 20th, 2013

Outline:

- Introduction
- Method and Technique
- Results
- Conclusions & Future work

Introduction: Motivation and scope

Motivation:



- Fluids like liquids and gases play an important role in the environment we live in. These phenomena seem simple and ordinary; however, it is complex and unfortunately difficult to simulate them.

- Demand for fluids in interactive application. Possible Applications: medical simulators, computer games or any type of virtual environment.

Scope:

Main goal: to achieve the fluid simulation numerically implemented in c++/OpenGL based on the method described in the paper[1]

[1]. Müller, Charypar and Gross, "Particle-Based Fluid Simulation for Interactive Applications", Eurographics, 2003

Methods

Basic simulation steps:

Initialization of the particles for the simulation

while simulation is running

// update particles' neighbors

for each particle

applying Grid-cell structure to get its neighbors

(including the index and the corresponding distance between them)

end

// update particles' density and pressure

for each particle *i*

for each neighbor *j* of this particle

add mass*W_poly6(rij, h) to density of particle *i* // $\rho(\boldsymbol{r}_i) = \sum_j m_j W(\boldsymbol{r}_i - \boldsymbol{r}_j, h)$

end

pressure of particle *i* <- k*(density of particle *i* – rest_density) // $p = k(\rho - \rho_0)$

end

//update particles' acceleration

for each particle *i*

set acceleration of particle *i* to be 0

for each neighbor j of particle *i*

add acceleration from pressure // $\boldsymbol{a}_i^{pressure} = -\sum_j m_j \frac{p_i + p_j}{2\rho_j \rho_i} \nabla W(\boldsymbol{r}_i - \boldsymbol{r}_j, h)$

add acceleration from viscosity // $\boldsymbol{a}_i^{viscosity} = \mu \sum_j m_j \frac{\boldsymbol{v}_j - \boldsymbol{v}_i}{\rho_j \rho_i} \nabla^2 W(\boldsymbol{r}_i - \boldsymbol{r}_j, h)$

add acceleration from the gravity

end

end

// update velocity and position

new velocity = old velocity +acceleration*elapsed-time

new position = old position+0.5*(old velocity + new velocity)*elapsed-time

enforcing the boundary constraints // simply push them out of the object and reflect the velocity component

end_while

Techniques: neighbor search

- naive method
  go over all of the particles to check the distance between two particle
  which would result in a computation complexity $O(n^2)$.

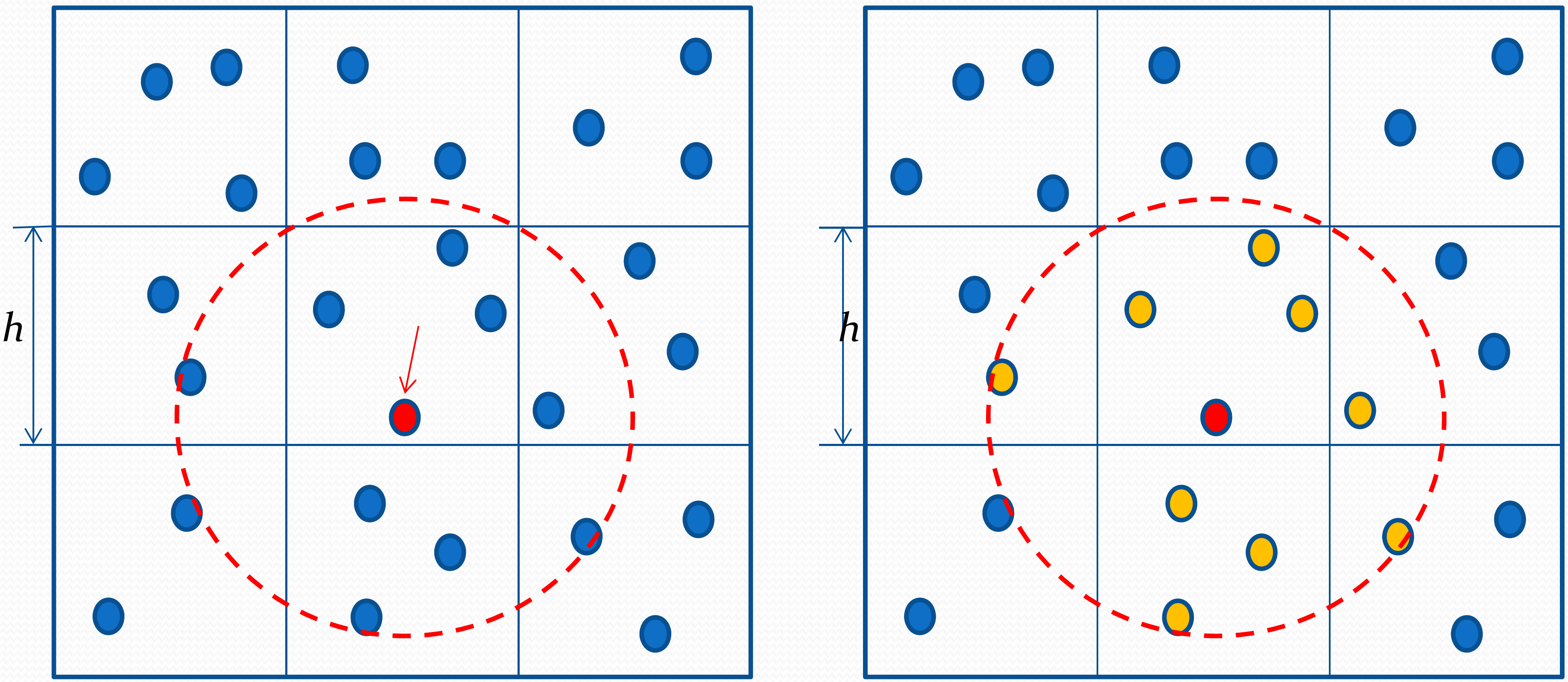
- Efficient acceleration structure: **grid-cell structure**
  -- the grid consists of cubic cells with a side length h,
  where h is the finite support of SPH kernel.

  -- each cell contains a reference to a list of all particles that map to
  the space partition associated with the cell.
  (we need update the content of the grid cell for each simulation step.)

  -- with grid-cell structure, it is much easier to find the neighbors of the particles.
  (all neighboring particles must be contained in the current or the 26 adjacent cells.)

  -- it reduces the time complexity from $O(n^2)$ to $O(nm)$,
  where m being the average number of particles per grid cell.

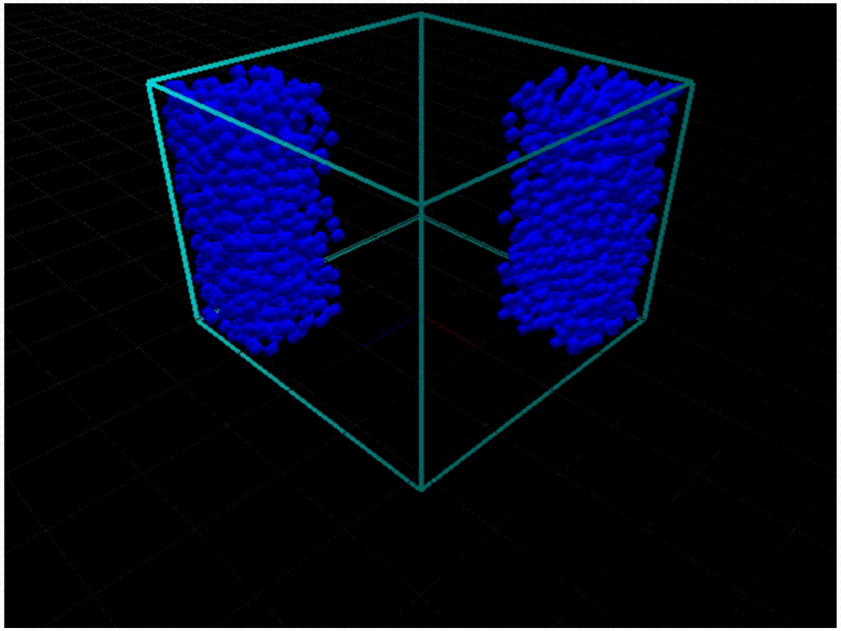
neighbor search with grid-cell structure



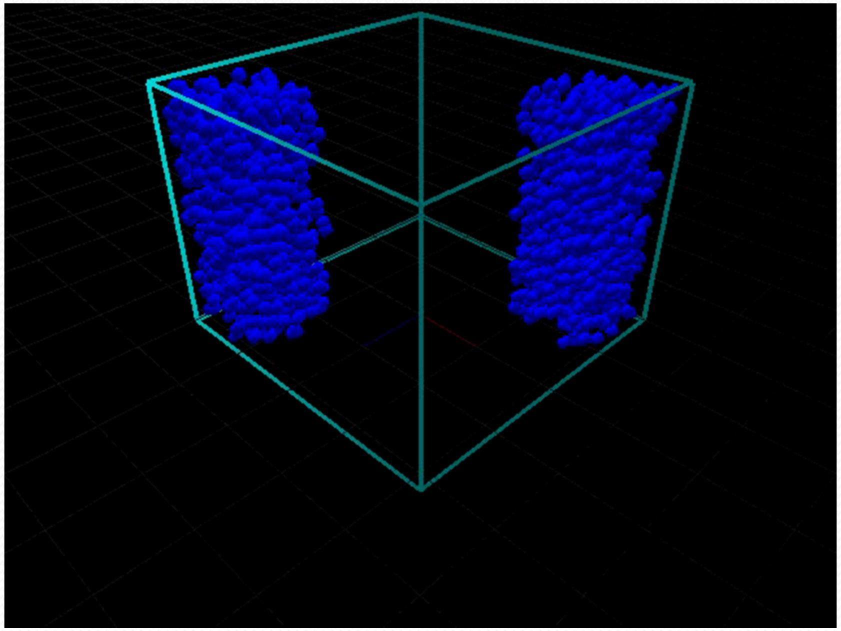
Results:

N = 2000, h =0.6, radius = 0.1, k = 8.0, rho0=20.0, viscosity =0.5, damp=0.8

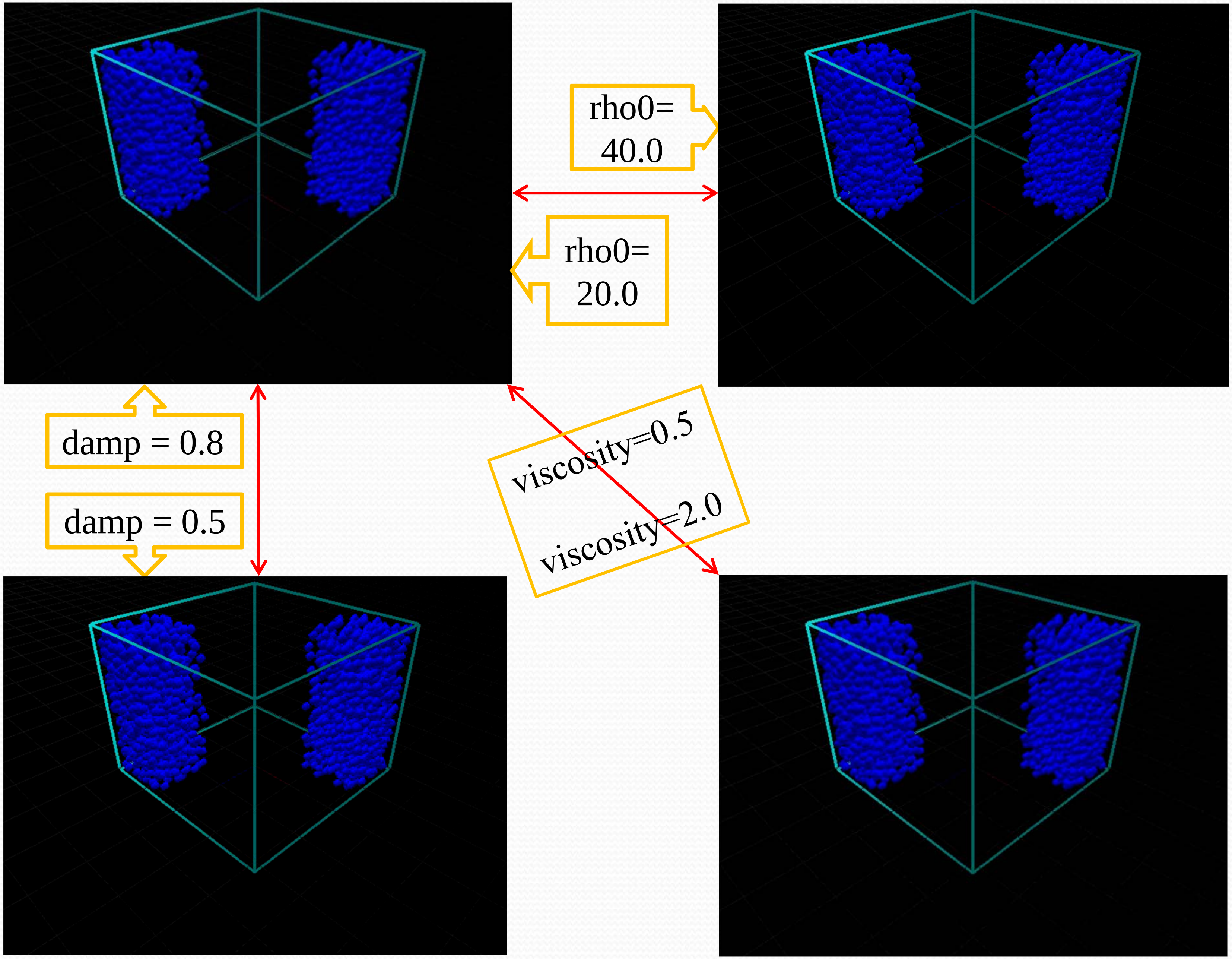
$\delta t = 0.05s$



$\delta t = 0.015s$



The computer simulation time for each frame is about 0.025s (interactive rates)


rho0=
40.0
rho0=
20.0
damp = 0.8
damp = 0.5
viscosity=0.5
viscosity=2.0

Conclusions & Future work

Conclusions:

- The model is based on Smoothed Particle Hydrodynamics and uses special purpose kernels to increase stability and speed.
- Grid-cell structure was used to reduce the computational complexity.
- Realize the fluid simulation *preliminary*.

Future work:

- My simulation result is still in **particles**. In the future, I'll spend more efforts on the visualization including the point splatting and marching cubes to let the simulation be more visually realistic.
- Moreover, add user-interactive feature.

Thank you !