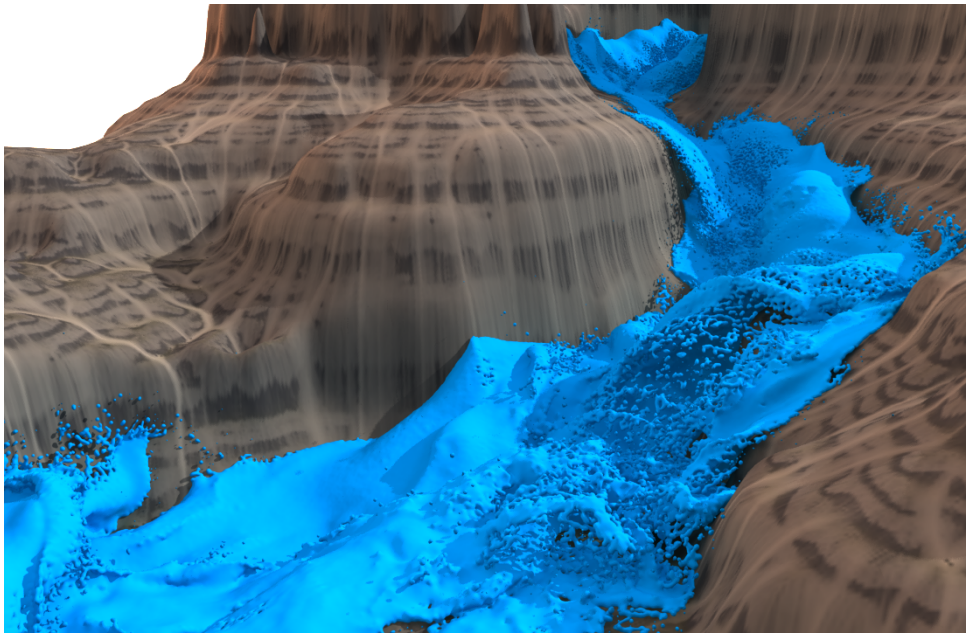
IGR 202 : PCISPH



An advanced fluid simulation method :
Predictive-Corrective Incompressible Smoothed Particle Hydrodynamics

Celio Boulay

Telecom Paris – supervised by Kiwon Um and Amal Dev Parakkat

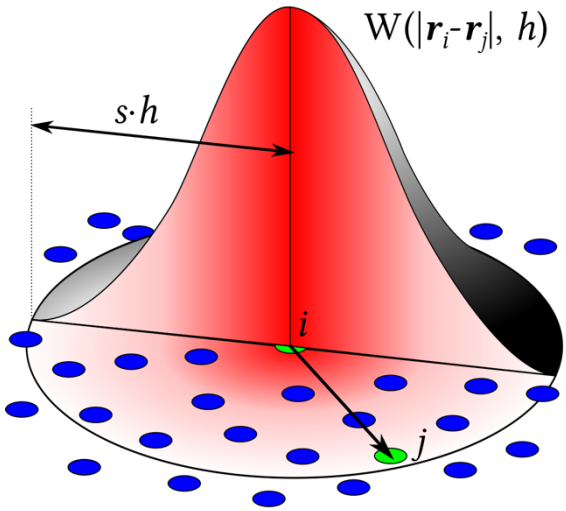
TELECOM Paris

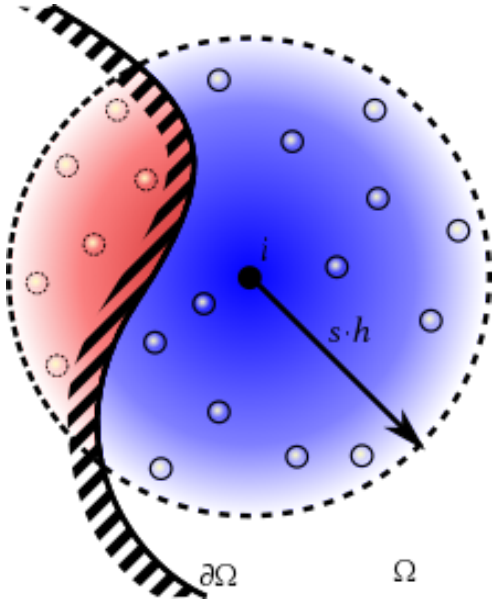
INSTITUT POLYTECHNIQUE DE PARIS

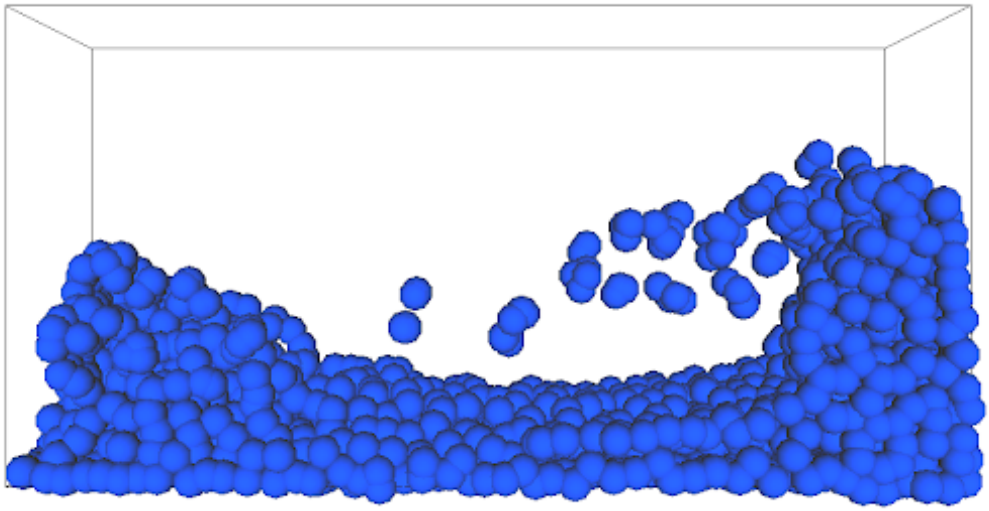
Table of contents

1) Introduction :

Fluid simulation is involved in many different topics, from animation to videogame design. The ability to precisely simulate the behavior of liquids (or gases) provides powerful tools for prediction, analysis, and visualization. Among the numerical methods developed for fluid simulation, the smoothed hydrodynamic particle (SPH) method stands out for its flexibility and adaptability, making it possible to model fluids with high fidelity compared to their real physical behaviors.



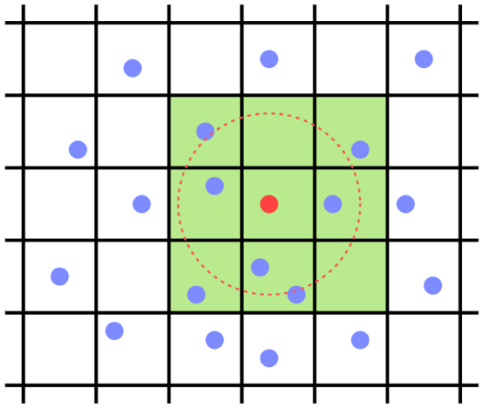




2) Current methods :

Standard SPH Method:

1. Density calculation and finding neighbors



$$\rho_i = \sum_j m_j W_{ij}$$

```
buildNeighbor();
computeDensity();
```

```
std::vector<tIndex> getNeighbors(const Vec2f& pos) const {
```

2. Forces

We consider body forces, viscosity, and pressure.

$$\mathbf{f}_i^{\text{body}} = m_i \mathbf{g} \qquad \mathbf{f}_i^{\text{pressure}} = -\frac{m_i}{\rho_i}\nabla p_i = -m_i \sum_j m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2}\right)\nabla W_{ij} \qquad \mathbf{f}_i^{\text{viscosity}} = 2\nu m_i \sum_j \frac{m_j}{\rho_j}\mathbf{u}_{ij}\frac{\mathbf{x}_{ij}\cdot\nabla W_{ij}}{\mathbf{x}_{ij}\cdot\mathbf{x}_{ij} + 0.01h^2}$$

3. Then we need to get the particles moving

```
void updateVelocity()
{
  for (tIndex k = 0; k < _pos.size(); k++)
    _vel[k] = _vel[k] + _dt*_acc[k];
}
```

```
void updatePosition()
{
  for (tIndex k = 0; k < _pos.size(); k++)
    _pos[k] = _pos[k] + _dt*_vel[k];
}
```

Incompressible SPH and Weakly Compressible SPH :

Based on an equation of state for pressure calculation.

$$p_i = \frac{k\rho_0}{\gamma}\left(\left(\frac{\rho_i}{\rho_0}\right)^{\gamma} - 1\right)$$

That involved long times to calculate the next time steps.

3) Predictive Corrective Incompressible SPH

This idea in the Predictive Corrective method is to replace the time-consuming equation of state (EOS) by a predictive-corrective iterative scheme.

```
1  while animating do
2    for all i do
3      find neighborhoods N_i(t)
4    for all i do
5      compute forces F^{v,g,ext}(t)
6      initialize pressure p(t) = 0.0
7      initialize pressure force F^p(t) = 0.0
8    while (ρ*_err(t+1) > η) || (iter < minIterations) do
9      for all i do
10       predict velocity v*_i(t+1)
11       predict position x*_i(t+1)
12     for all i do
13       predict density ρ*_i(t+1)
14       predict density variation ρ*_err(t+1)
15       update pressure p_i(t) += f(ρ*_err(t+1))
16     for all i do
17       compute pressure force F^p(t)
18   for all i do
19     compute new velocity v_i(t+1)
20     compute new position x_i(t+1)
```

Instead of calculating the updated pressure with the EOS, we predict the velocity and positions of particles, and then we predict the density they would have. But for this step, the pressure forces are initialized to 0. That's means the density calculated is not correct : it should be equal to the rest density of the fluid (here d0).

So we introduce the density variation, simply defined as the difference between the predicted density and the rest density of the fluid. Our goal is now to get that density variation to get as close as possible to 0.

To do that, we adjust the pressure of particles at each step, using specific formulas, determined by B. Solenthaler on her paper about PCISPH, at the University of Zurich.

$$\delta = \frac{-1}{\beta(-\sum_j \nabla W_{ij} \cdot \sum_j \nabla W_{ij} - \sum_j (\nabla W_{ij} \cdot \nabla W_{ij}))}$$

and

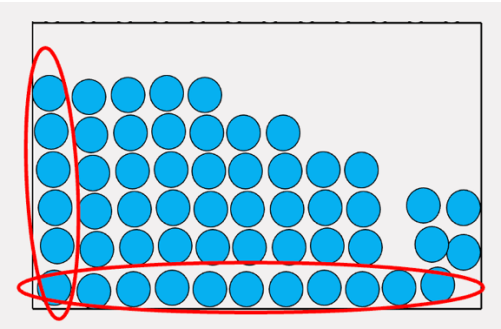
$$\tilde{p}_i = \delta \rho^*_{err_i}.$$

To reach shorter calculation times, we precompute the delta value at the beginning of the simulation and then use it for the entire simulation. For that purpose we choose a particle with a filled neighborhood, "in the middle of the borders".

4) Ideas for future improvements

Two main ideas could be explored to improve that study.

A better neighbor approximation: Recalculate the neighbourhood in specific cases instead of relying on virtual boundary particles.



3D simulation using external software: Adapting the SPH code to PCISPH in a 3D simulation would be too difficult. But we could use the PCISPH algorithm over 3D data, and then use the data on a different software, where we would just have to input the data timestep by timestep, instead of coding a 3D environment with OpenGL.