CSCI580 Final Project Report

Yumeng He
heyumeng@usc.edu

Hsin Li
hsinli@usc.edu

Yuchen Chen
ychen581@usc.edu

Xu Chen
xchen659@usc.edu

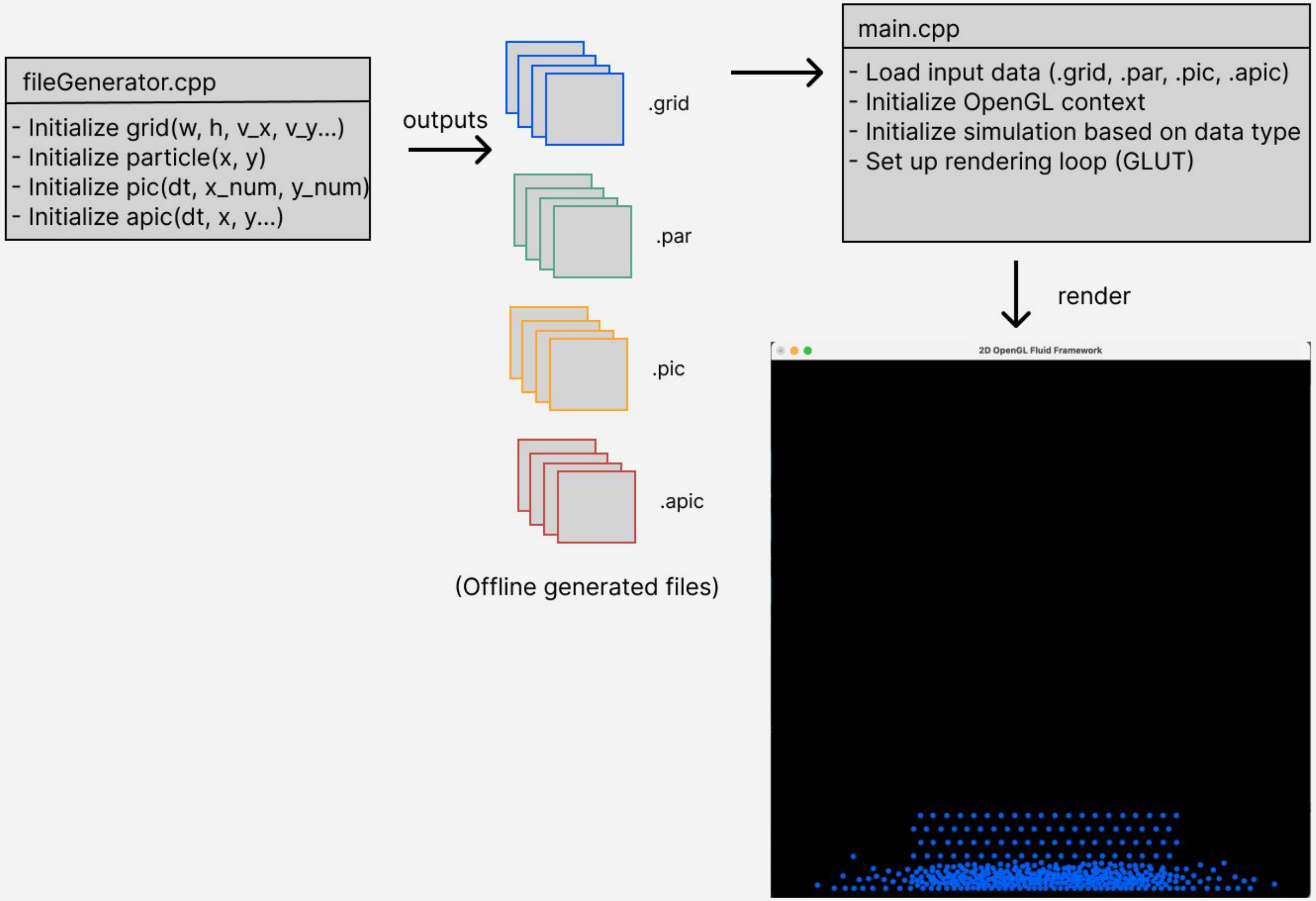
Introduction

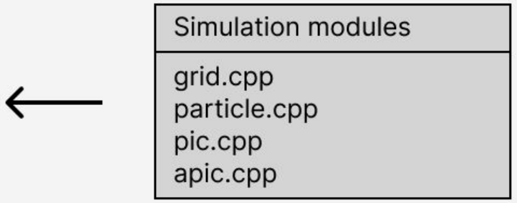
We implemented a **2D incompressible fluid simulation** using **C++** and **OpenGL**.
Our goal is to compare different simulation methods:

- Grid (Stable Fluids Method)
- Particle (Smoothed Particle Hydrodynamics)
- Particle-In-Cell (PIC)
- Affine Particle-In-Cell (APIC).

Visualization will leverage OpenGL with GLUT.

Render



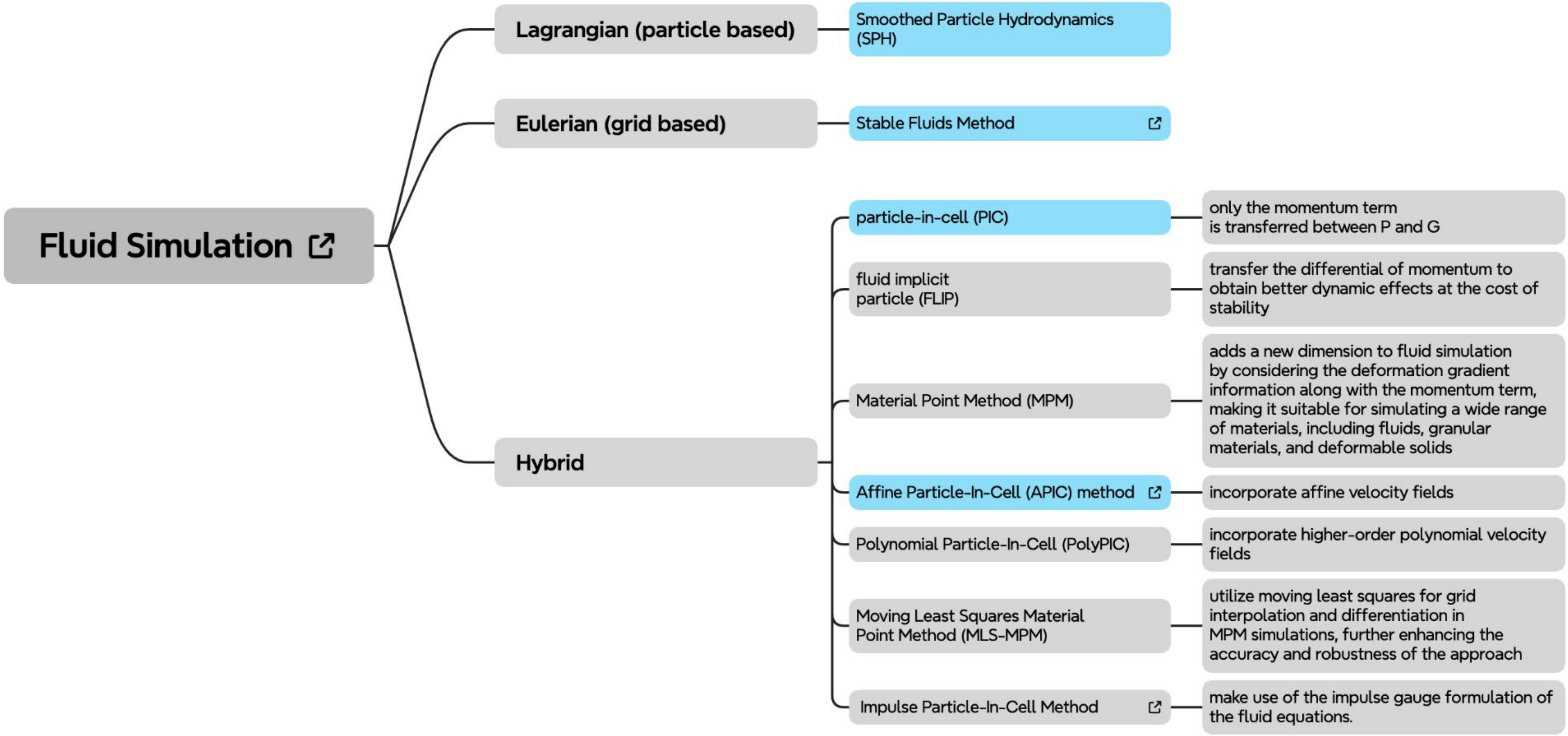


GL_QUADS: grayscale rectangles based on density

GL_POINTS: smooth points

glutSwapBuffers, idle callback: real-time simulation and rendering

Fluid Methods



Equations

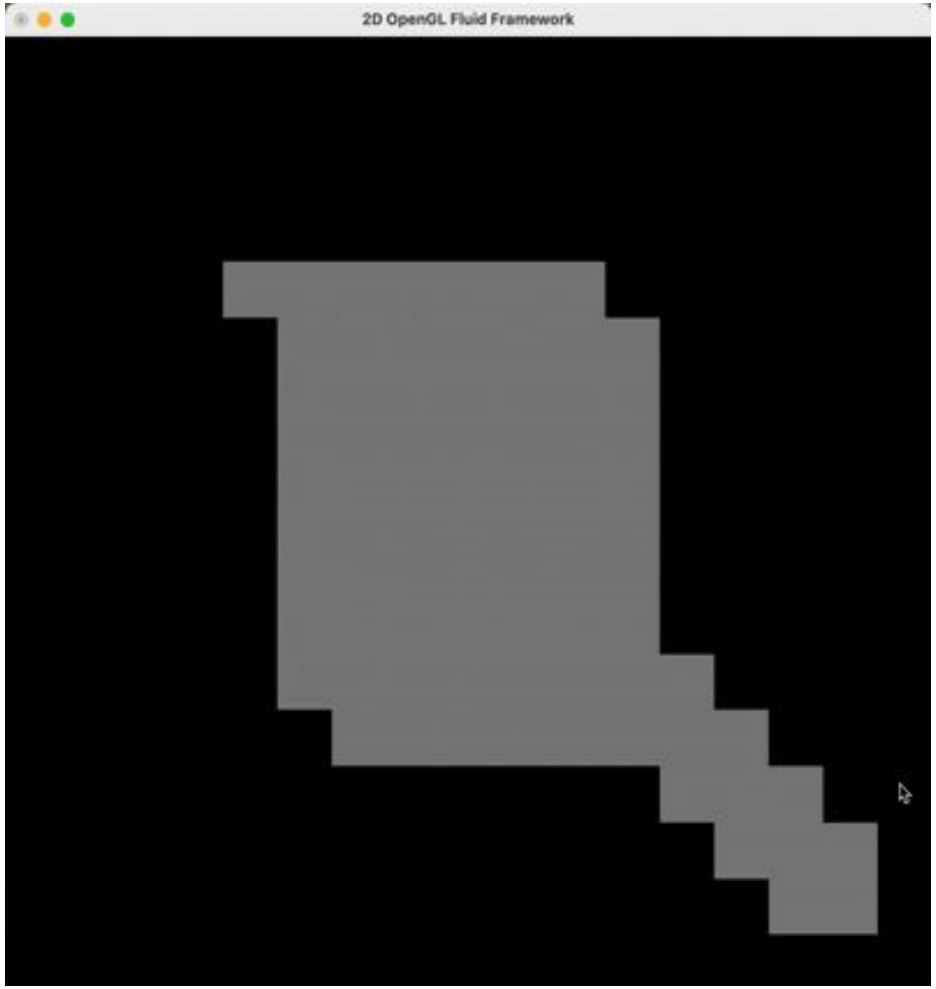
Navier-Stokes Equations for the velocity:

$$\frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u} \cdot \nabla)\mathbf{u} + \nu \nabla^2 \mathbf{u} + \mathbf{f}$$

The equation for a density moving through the velocity field:

$$\frac{\partial \rho}{\partial t} = -(\mathbf{u} \cdot \nabla)\rho + \kappa \nabla^2 \rho + S$$

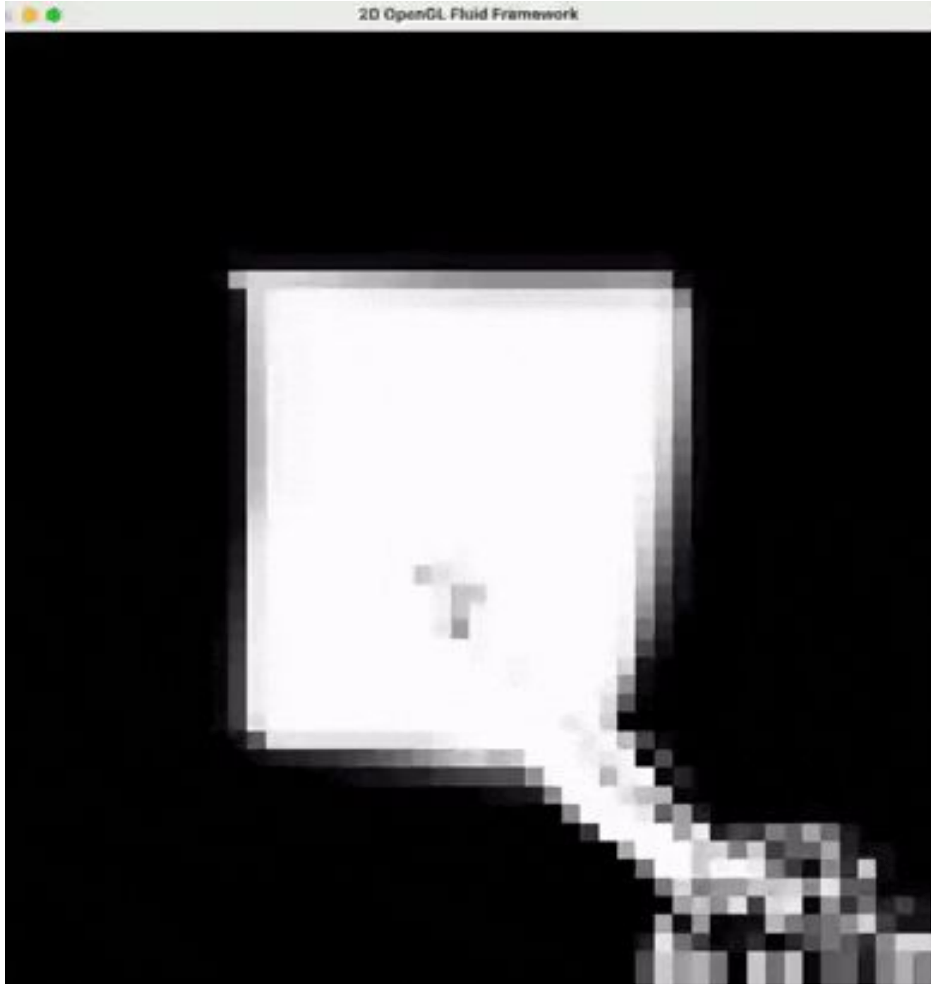
Grid - Stable Fluids Method



[Add Source] → [Diffuse] → [Project] → [Advect]

Step	Purpose
Add Source	Injects new momentum or density
Diffuse	Models viscosity and spreading
Project	Enforces incompressibility
Advect	Moves material/velocity with flow

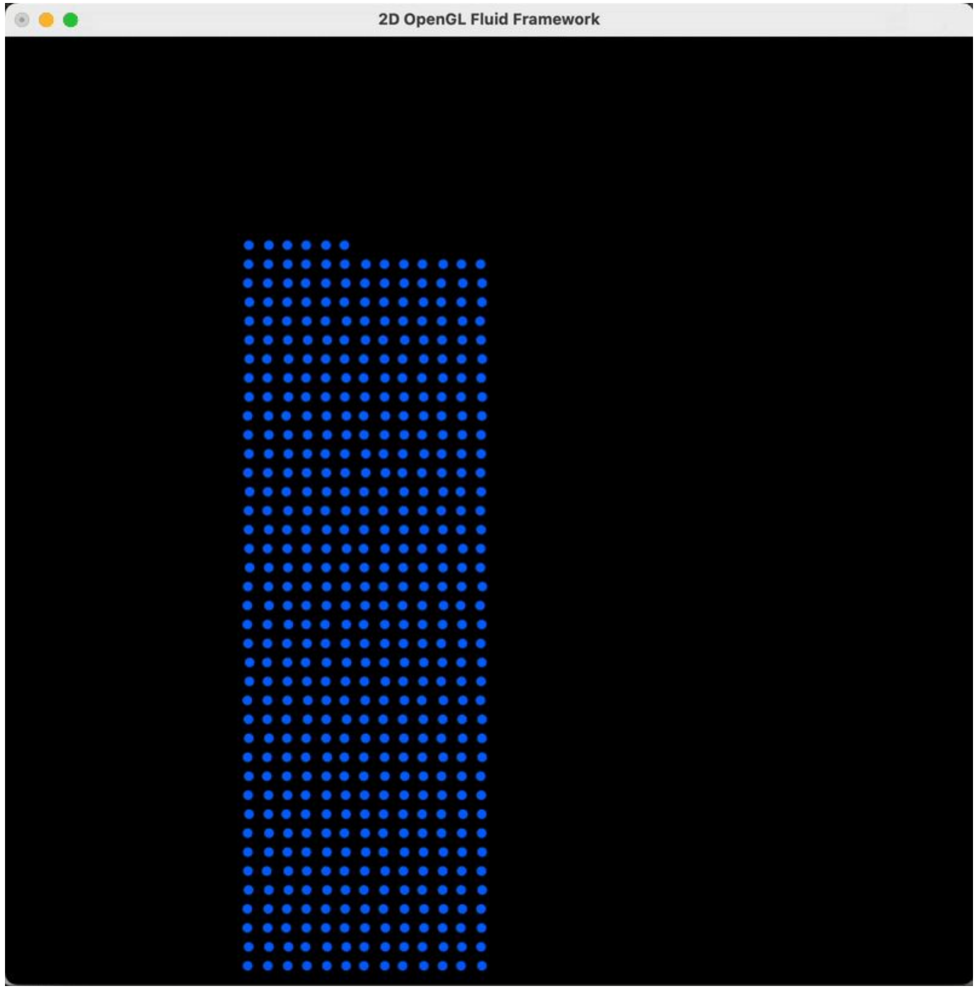
Grid - Stable Fluids Method



[Add Source] → [Diffuse] → [Project] → [Advect]

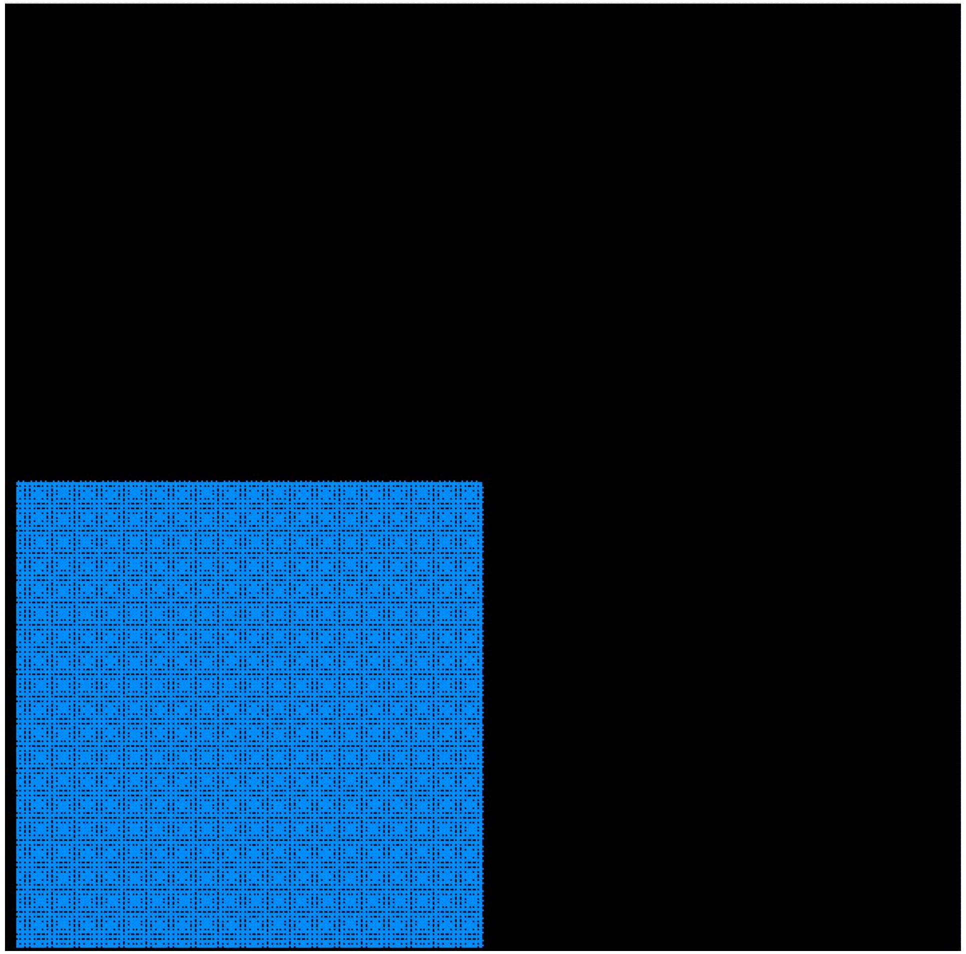
Step	Purpose
Add Source	Injects new momentum or density
Diffuse	Models viscosity and spreading
Project	Enforces incompressibility
Advect	Moves material/velocity with flow

Particle - Smoothed Particle Hydrodynamics



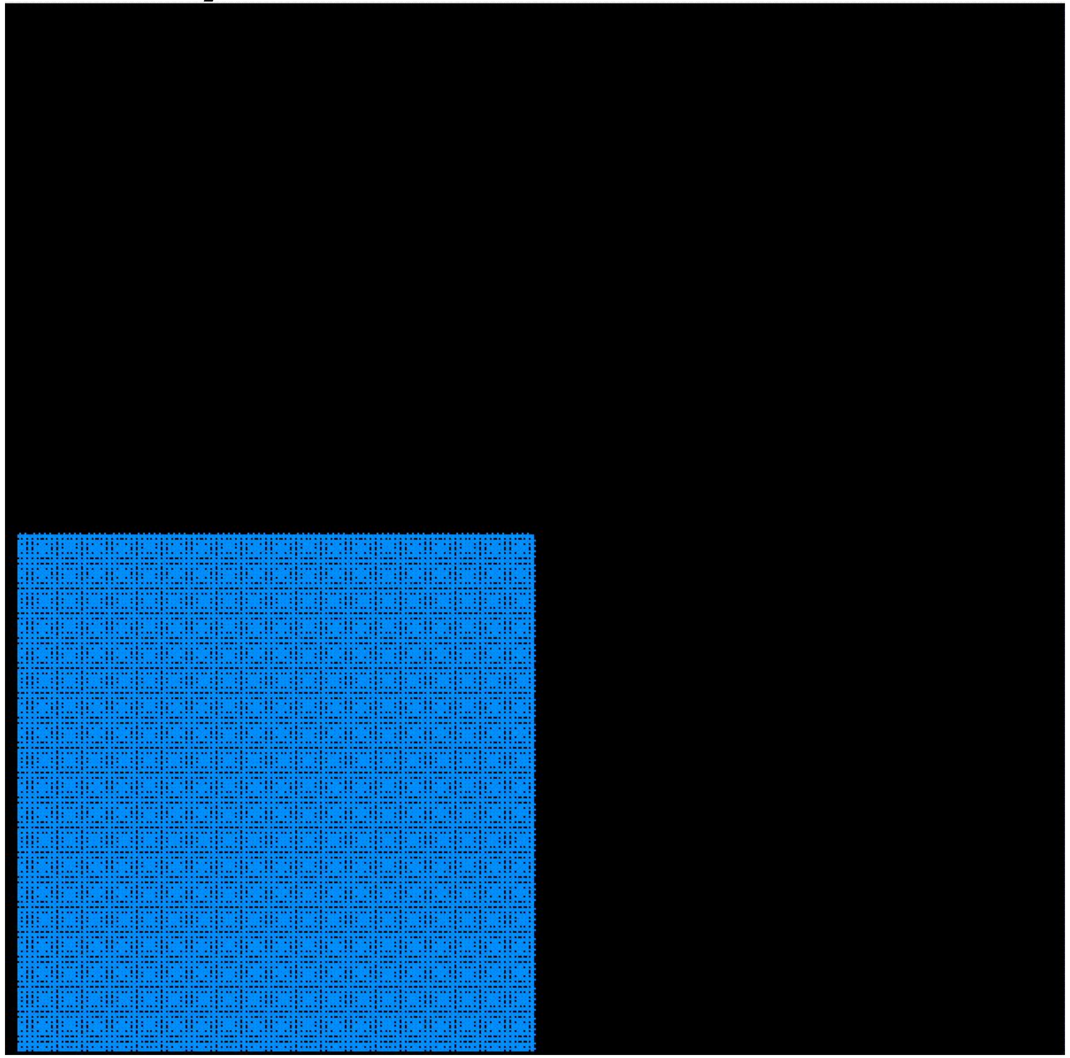
1. **ComputeDensityPressure()** accumulate density of neighboring particles and calculate pressure
2. **ComputeForces():**
   a. pressure: direction along the vector connecting particles
   b. viscosity: direction relative to velocity
   c. gravity
3. **Integrate():** for each particle update velocity and position

Particle in cell (PIC)



1. **particle_to_grid():** Transfer velocity/mass to grid (B-Spline weights)
2. **apply_grid_forces():** Apply gravity
3. **project_pressure():** Solve Poisson equation
4. **grid_to_particle():** Interpolate grid velocity back to particles
5. **advect_particles():** Move particles and enforce boundaries

PIC/FLIP

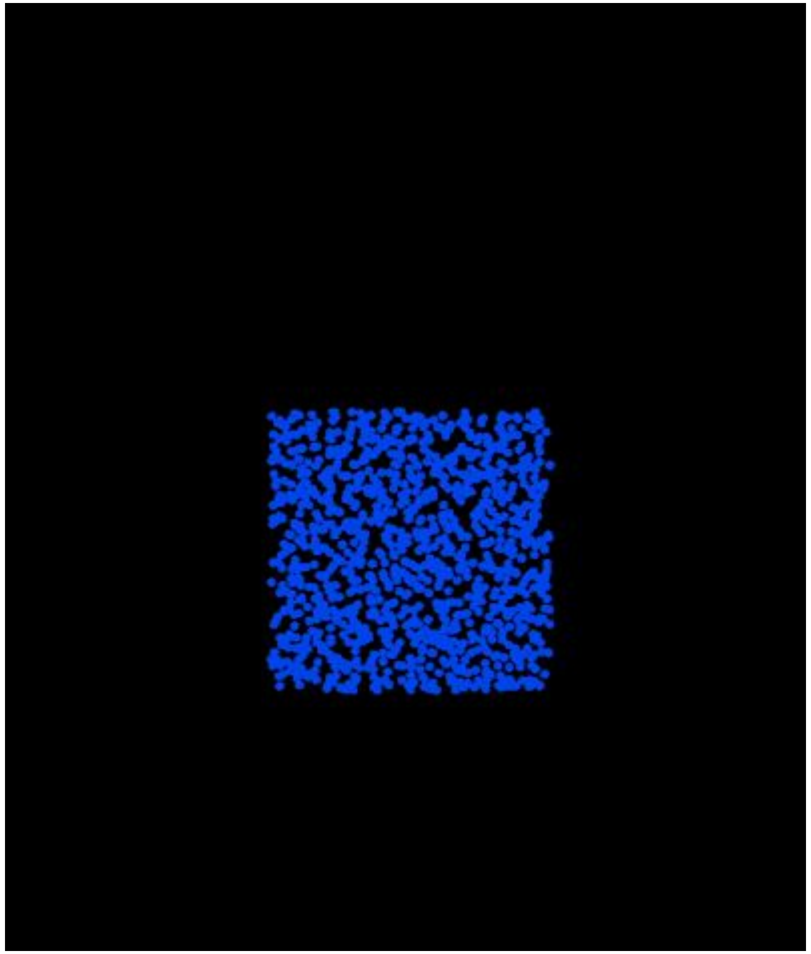


```
Vec2 flip_delta = new_vel - old_vel; // change in grid velocity (FLIP update term)
p.vel = p.vel + flip_delta * FLIP_RATIO + new_vel * (1.0f - FLIP_RATIO);
```

```
void step() {
    particle_to_grid();
    grid.velocity_old = grid.velocity; // here ! !
    apply_grid_forces();
    project_pressure();
    grid_to_particle();
    advect_particles();
}
```

- Implemented FLIP and PIC blending to improve particle behavior
- Set FLIP_RATIO = 0.96

Affine Particle in cell (APIC)



- **particle_to_grid_apic()** : Transfer affine velocity v = v□ + B × (xg - xp)
- **apply_grid_forces()**
- **project_pressure()**
- **grid_to_particle_apic()** : Interpolate velocity and update B
- **advect_particles()**

Comparison

	Pro	Con
Grid (Stable Fluids)	- Fast (real time) - Unconditionally stable	- Loss of detail - Not physically accurate
Particle (Smoothed Particle Hydrodynamics)	- Fast	- Limitation of input particle position - Particle collapse
Particle in cell (PIC)	- stable	- High numerical dissipation - Particles lose energy quickly
PIC/FLIP	- More realistic and dynamic motion	- Less stable - needs tuning
APIC	- Preserves rotation	- More complex - slower

Thank you

References

Stam, J. & Alias wavefront. (1999b). Stable fluids. In Alias Wavefront (p. 121) [Journal-article]. Alias wavefront. https://pages.cs.wisc.edu/~chaol/data/cs777/stam-stable_fluids.pdf

Stam, J. (2003, March). Real-time fluid dynamics for games. In Proceedings of the game developer conference (Vol. 18, No. 11). https://graphics.cs.cmu.edu/nsp/course/15-464/Fall09/papers/StamFluidforGames.pdf

Müller, M., Dorsey, J., McMillan, L., Jagnow, R., & Cutler, B. (2003). Stable Real-Time Deformations. In Eurographics/ACM SIGGRAPH Symposium on Computer Animation (p. 49–54) [Conference paper]. Eurographics Association. https://matthias-research.github.io/pages/publications/sca03.pdf

Robert Bridson. (2015). Fluid Simulation for Computer Graphics (2nd ed., Chapter 7.6). CRC Press. [Book].

Jiang, C., Schroeder, C., Selle, A., Teran, J., & Stomakhin, A. (2015). The affine particle-in-cell method. ACM Transactions on Graphics, 34(4), Article 51. https://doi.org/10.1145/2766996

Appendix I: APIC formula

$$m_i^n \mathbf{v}_i^n = \sum_p w_{ip}^n m_p (\mathbf{v}_p^n + \mathbf{B}_p^n (\mathbf{D}_p^n)^{-1} (\mathbf{x}_i - \mathbf{x}_p^n)), \tag{8}$$

where $\mathbf{C}_p^n = \mathbf{B}_p^n (\mathbf{D}_p^n)^{-1}$ and $\mathbf{D}_p^n$ is analogous to an inertia tensor. $\mathbf{D}_p^n$ is given by

$$\mathbf{D}_p^n = \sum_i w_{ip}^n (\mathbf{x}_i - \mathbf{x}_p^n)(\mathbf{x}_i - \mathbf{x}_p^n)^T \tag{9}$$

and is derived by preserving affine motion during the transfers. The corresponding transfer from the grid back to particles is

$$\mathbf{B}_p^{n+1} = \sum_i w_{ip}^n \tilde{\mathbf{v}}_i^{n+1} (\mathbf{x}_i - \mathbf{x}_p^n)^T. \tag{10}$$