🌊 Blub 💦

Quick tour through a GPU fluid solver

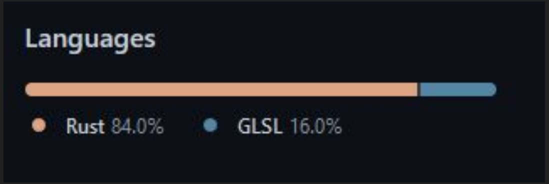
Introduction - Speaker

- Andreas Reich / @wumpf
- Working in Game Engines (currently @ Unity)
  - **Not** on rendering or physics!

- https://github.com/**Wumpf**
- **Rust & rendering/simulation as hobby**
- Occasional wgpu contributor

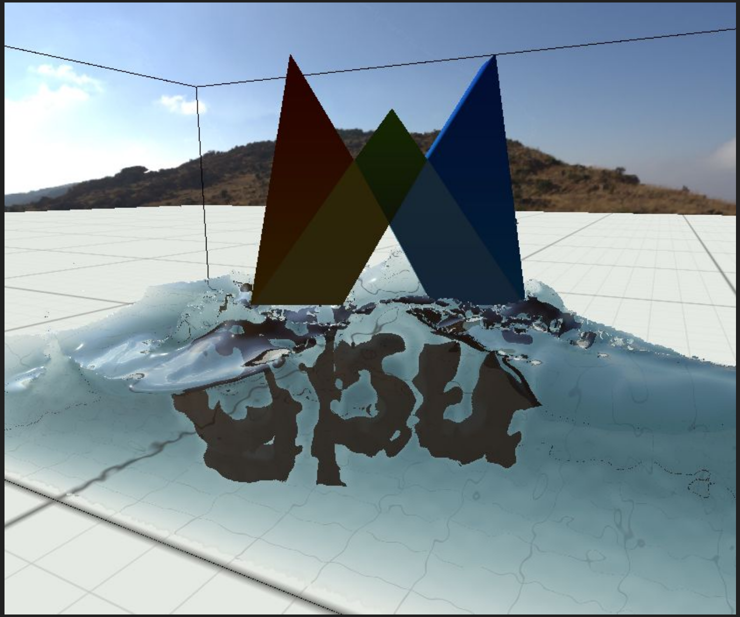
Introduction - Blub

- Realtime(*ish*) GPU fluid simulation & rendering
  - Powered by wgpu!
- Side project, no overarching goal in mind
  - Learn & play
  - Developed on and off for about a year
- Affine Particle in Cell (APIC)
  - More on that later!
- Supports only solid ➡ fluid interaction
  - "solid moves water"
  - ~~"water moves solid"~~



Languages

● Rust 84.0% ● GLSL 16.0%



Outline

- Brief Demo
- Fluid Sim Basics
- Blub implementation details
    - Pipeline
    - Voxelization
    - Particle Transfer
    - Pressure Solve

Not talking about rendering today!

Demo!

Fluid Simulation Basics

Navier-Stokes Equation (for incompressible fluids)

$$\frac{\partial \mathbf{u}}{\partial t} = -\mathbf{u} \cdot \nabla \mathbf{u} + \mathbf{g} + \nu \Delta \mathbf{u} - \frac{1}{\rho} \nabla p$$

$$\nabla \cdot \mathbf{u} = 0$$

Navier-Stokes Equation (for incompressible fluids)

🏃‍♀️ 💨 🍎 🍯 😣

Acceleration = Advection + External + Viscosity - Pressure

✗

Divergence = 0

= don’t add or remove fluid

Navier-Stokes Equation (for incompressible fluids)

🏃‍♀️ 💨 🍎 😣

Acceleration = Advection + Gravity - Pressure

Divergence = 0

= don’t add or remove fluid!

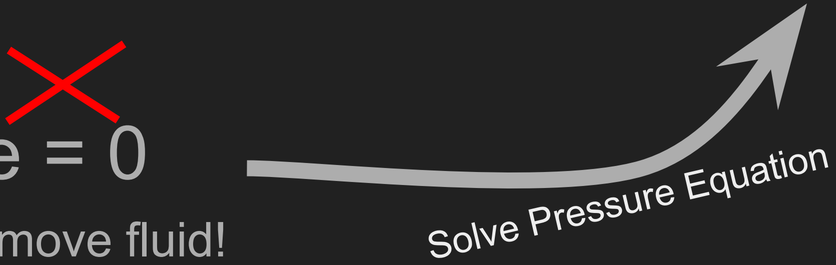
Navier-Stokes Equation (for incompressible fluids)

🏃‍♀️ 💨 🍎 😣

Acceleration = Advection + Gravity - Pressure

Divergence = 0

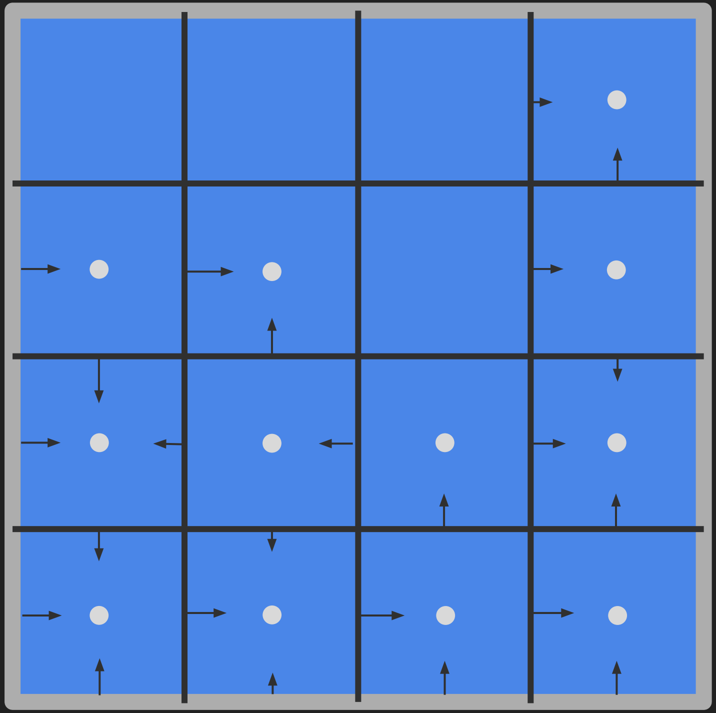
= don’t add or remove fluid!



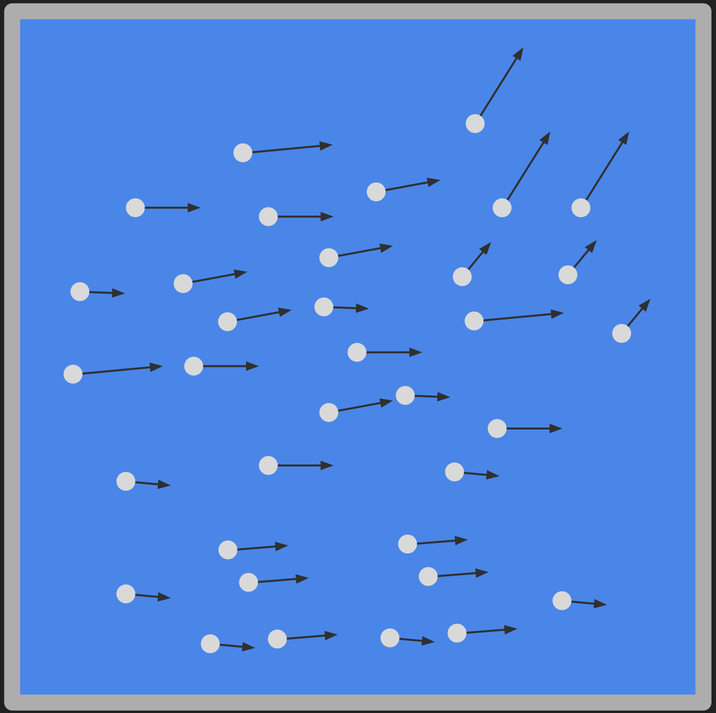
Solve Pressure Equation

Types of Fluid Simulation

Eulerian (Grid)



Lagrangian (Particles)



Types of Fluid Simulation

Eulerian (Grid)

- Easy pressure solve!
- Constant sampling rate
- Can be unconditionally stable

Lagrangian (Particles)

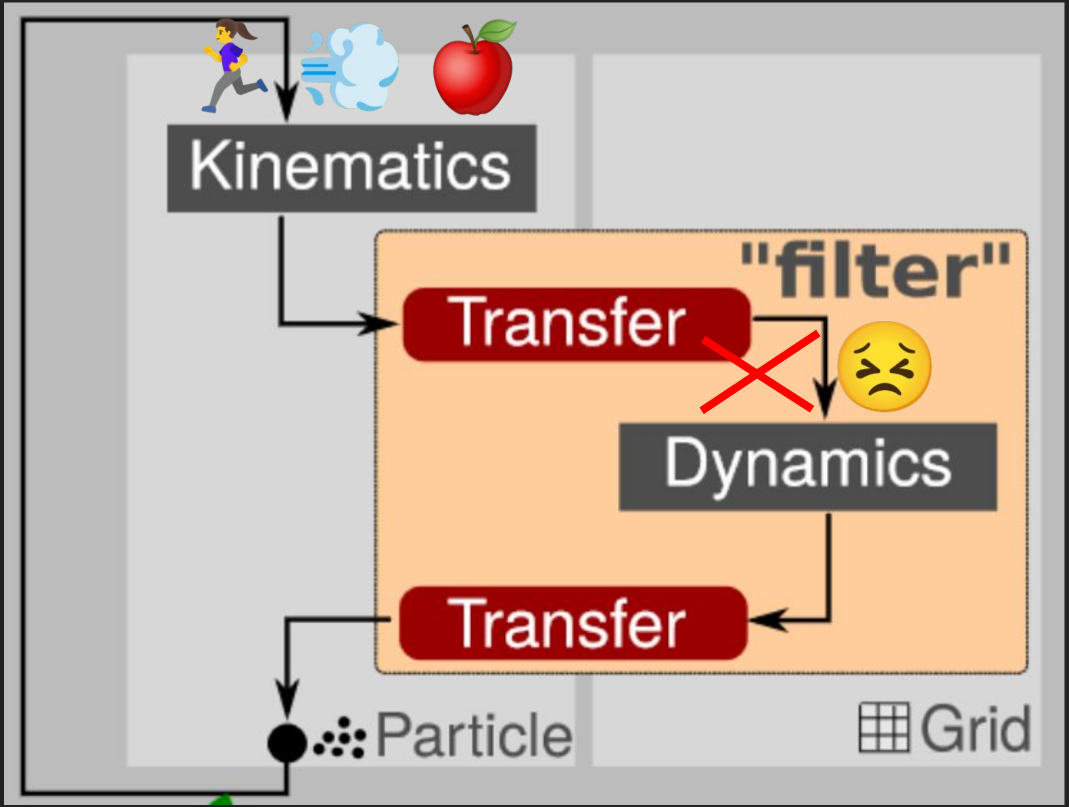
- Easy advection!
- Varying sampling rate
- Needs very small timesteps

Particles OR Grid?

- Advection 💨
  - Easy with particles!
- Pressure solve
  - “Easy” with grid! ❌ 😣

🤔

Hybrid / APIC



https://www.math.ucla.edu/~jteran/papers/JSSTS15.pdf The Affine Particle-In-Cell Method, Jiang et a.

[some interesting tidbits of]

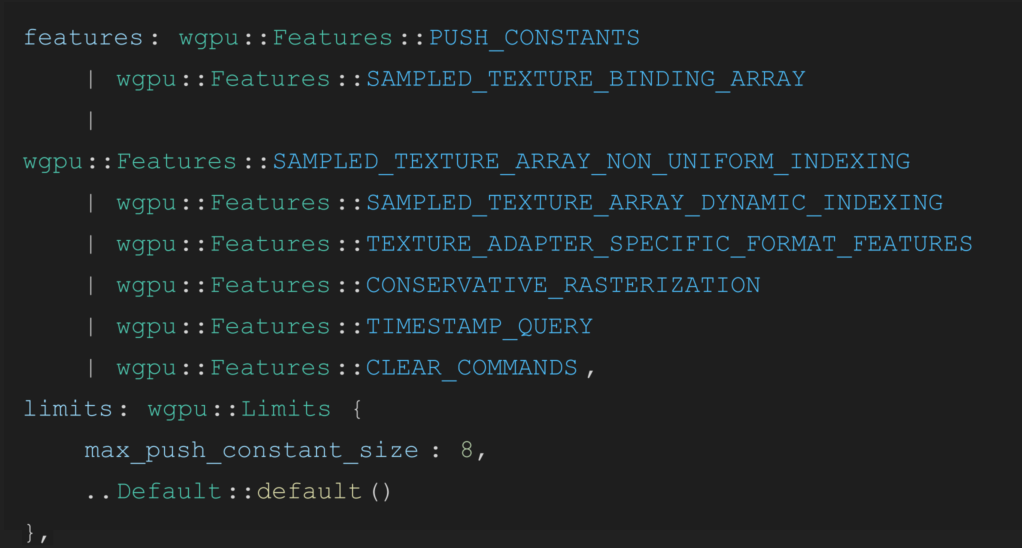
Blub Implementation

Compute Shader everywhere!

- 🙇‍♀️ Runtime Shader reloading 🙇‍♀️

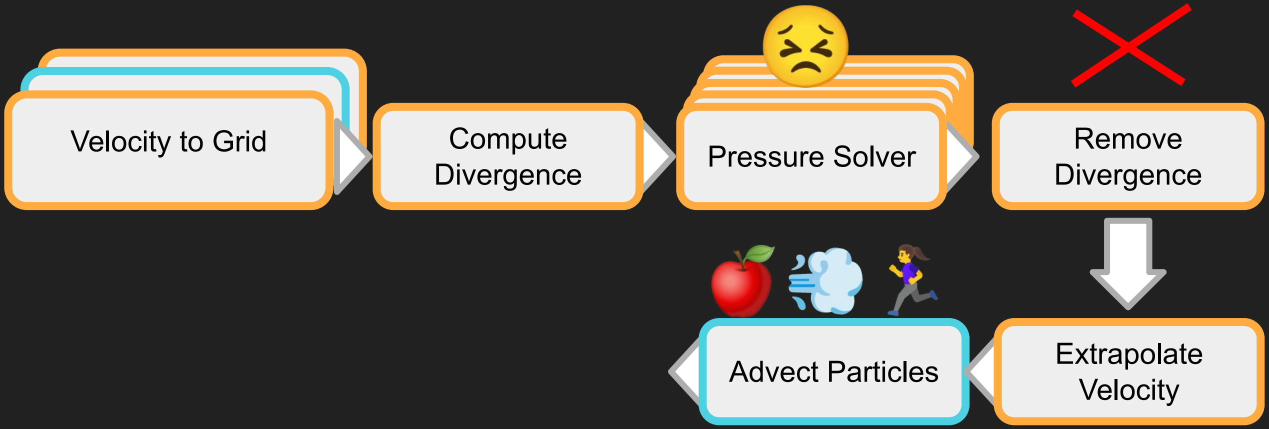
- Written in GLSL
- Grids are `image3d`
- Bunch of native extensions

- Not Metal/Mac compatible 🙁



```
features: wgpu::Features::PUSH_CONSTANTS
    | wgpu::Features::SAMPLED_TEXTURE_BINDING_ARRAY
    |
wgpu::Features::SAMPLED_TEXTURE_ARRAY_NON_UNIFORM_INDEXING
    | wgpu::Features::SAMPLED_TEXTURE_ARRAY_DYNAMIC_INDEXING
    | wgpu::Features::TEXTURE_ADAPTER_SPECIFIC_FORMAT_FEATURES
    | wgpu::Features::CONSERVATIVE_RASTERIZATION
    | wgpu::Features::TIMESTAMP_QUERY
    | wgpu::Features::CLEAR_COMMANDS,
limits: wgpu::Limits {
    max_push_constant_size: 8,
    ..Default::default()
},
```

Blub Simulation Step



Velocity to Grid

Compute Divergence

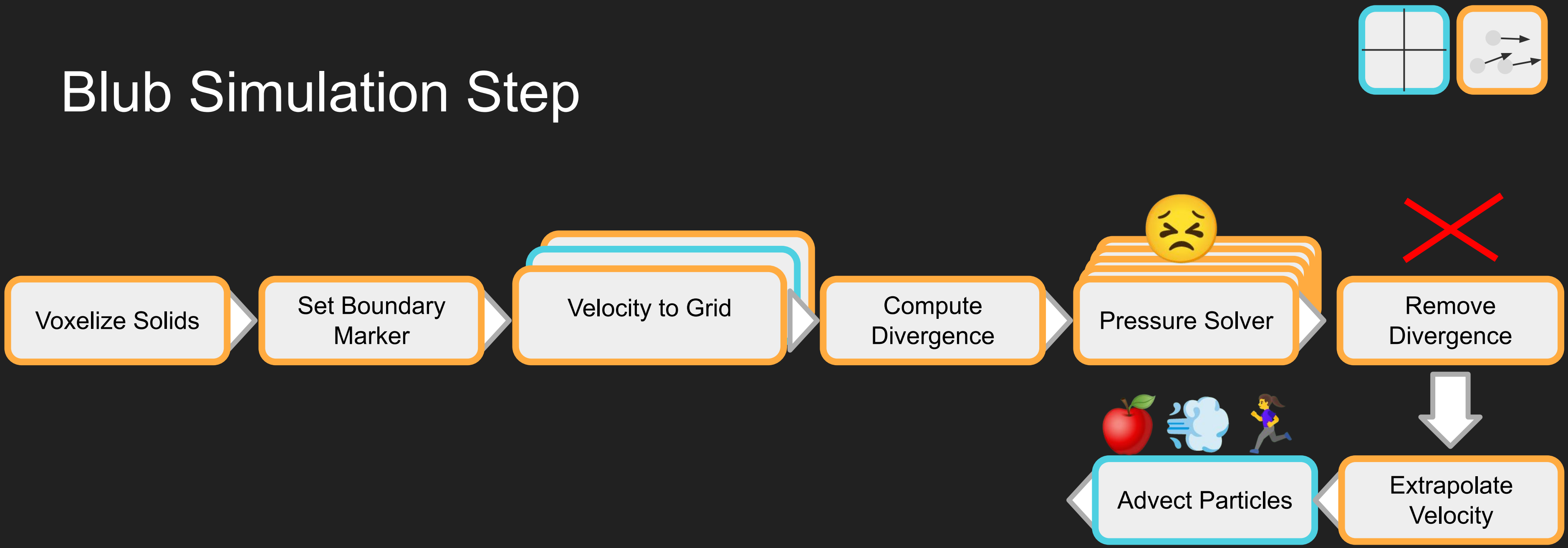
Pressure Solver

Remove Divergence

Extrapolate Velocity

Advect Particles

Blub Simulation Step



Voxelize Solids

Set Boundary Marker

Velocity to Grid

Compute Divergence

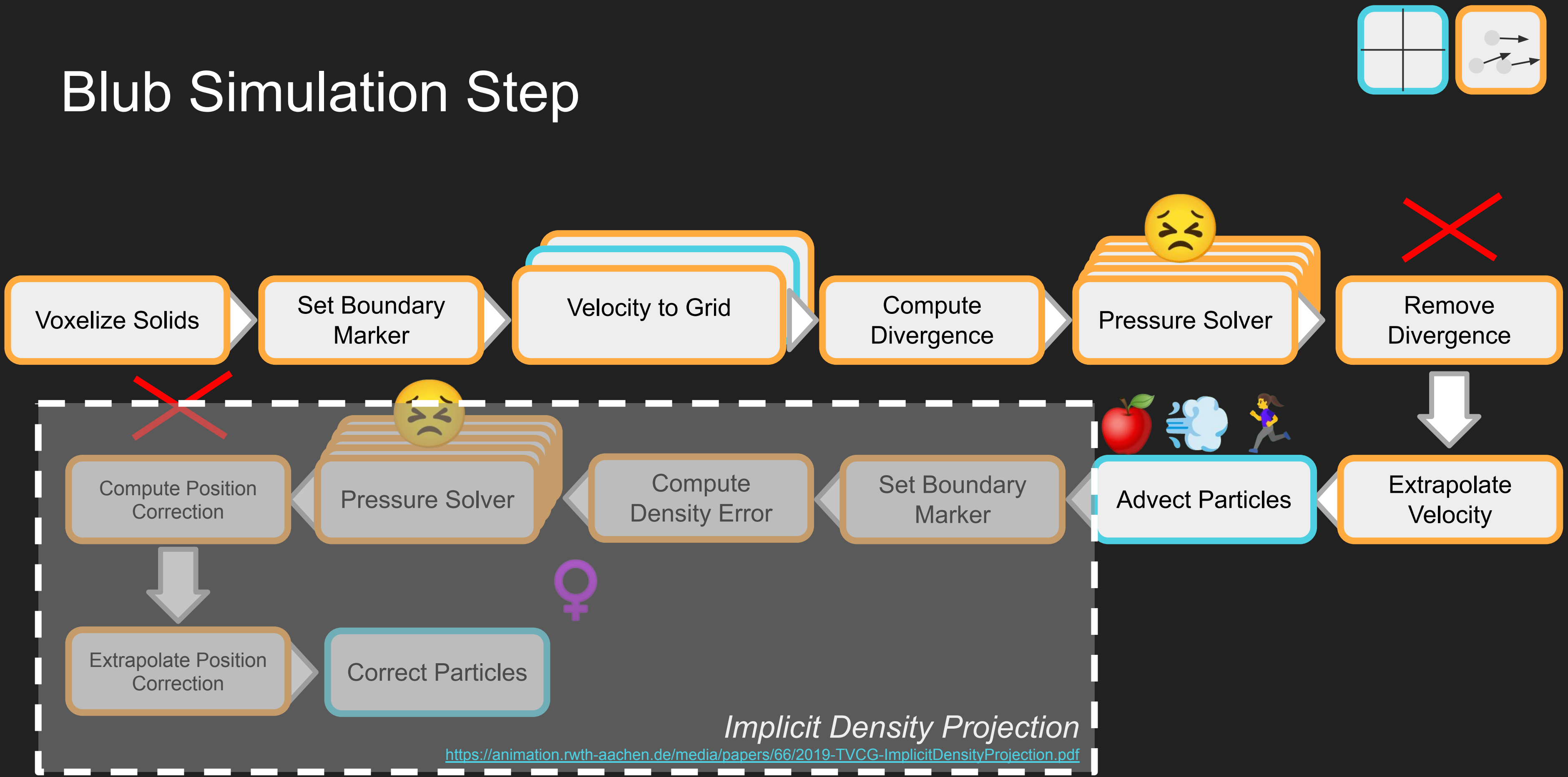
Pressure Solver

Remove Divergence

Extrapolate Velocity

Advect Particles

Blub Simulation Step



Voxelize Solids

Set Boundary Marker

Velocity to Grid

Compute Divergence

Pressure Solver

Remove Divergence

Extrapolate Velocity

Advect Particles

Set Boundary Marker

Compute Density Error

Pressure Solver

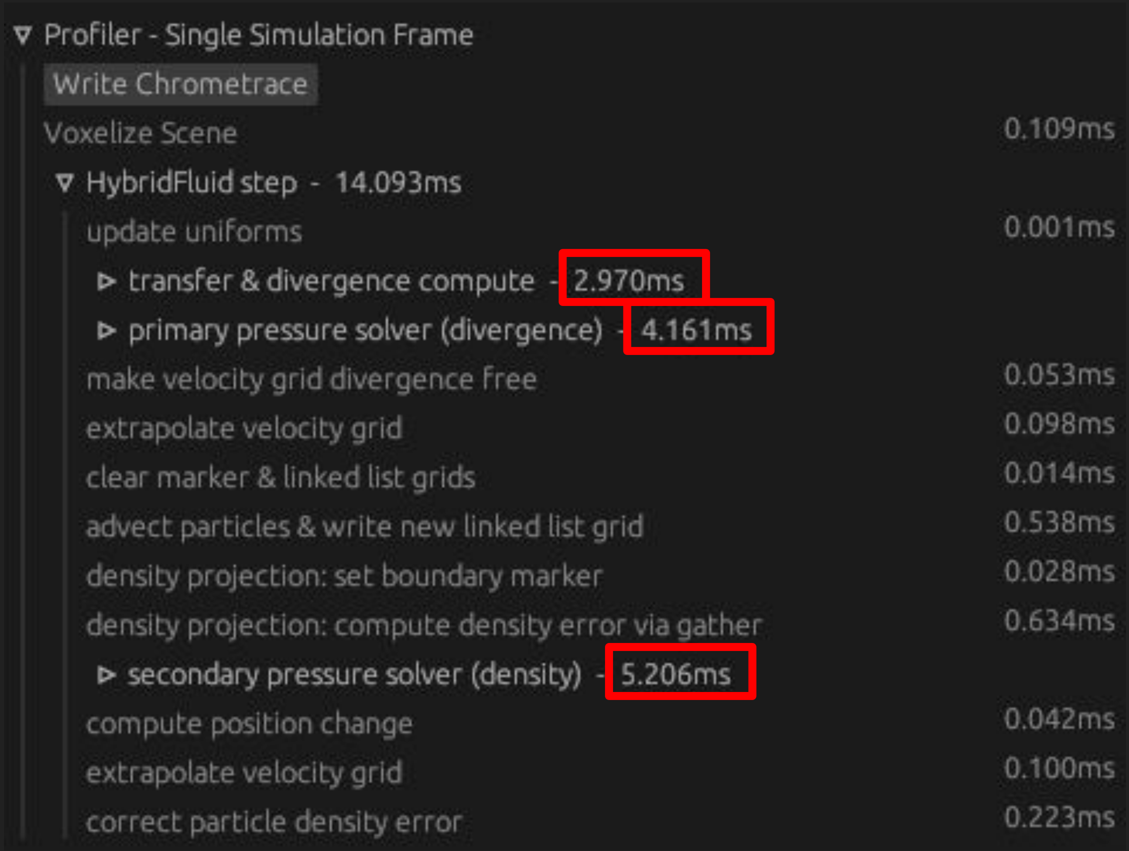
Compute Position Correction

Extrapolate Position Correction

Correct Particles

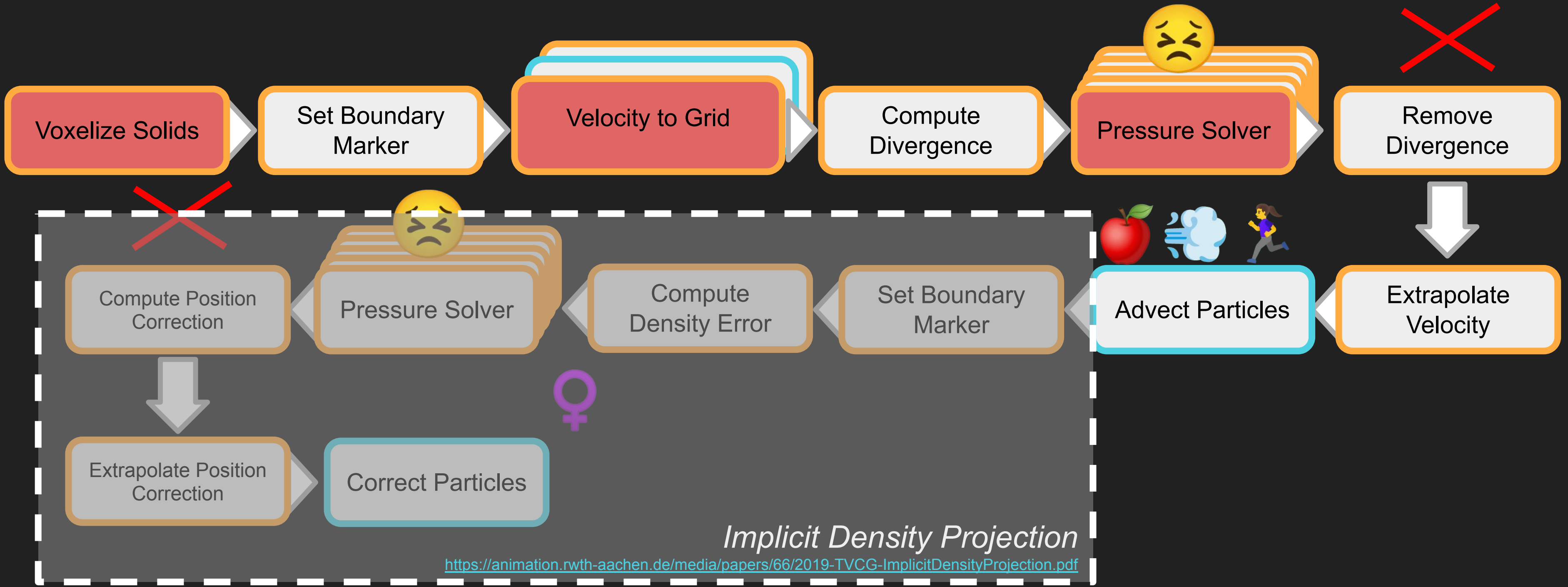
Implicit Density Projection

https://animation.rwth-aachen.de/media/papers/66/2019-TVCG-ImplicitDensityProjection.pdf



```
▽ Profiler - Single Simulation Frame
  Write Chrometrace
  Voxelize Scene                                                     0.109ms
  ▽ HybridFluid step - 14.093ms
    update uniforms                                                  0.001ms
    ▷ transfer & divergence compute - 2.970ms
    ▷ primary pressure solver (divergence) - 4.161ms
    make velocity grid divergence free                               0.053ms
    extrapolate velocity grid                                        0.098ms
    clear marker & linked list grids                                 0.014ms
    advect particles & write new linked list grid                    0.538ms
    density projection: set boundary marker                          0.028ms
    density projection: compute density error via gather            0.634ms
    ▷ secondary pressure solver (density) - 5.206ms
    compute position change                                          0.042ms
    extrapolate velocity grid                                        0.100ms
    correct particle density error                                   0.223ms
```

Blub Simulation Step



Voxelize Solids

Set Boundary Marker

Velocity to Grid

Compute Divergence

Pressure Solver

Remove Divergence

Extrapolate Velocity

Advect Particles

Set Boundary Marker

Compute Density Error

Pressure Solver

Compute Position Correction

Extrapolate Position Correction

Correct Particles

Implicit Density Projection

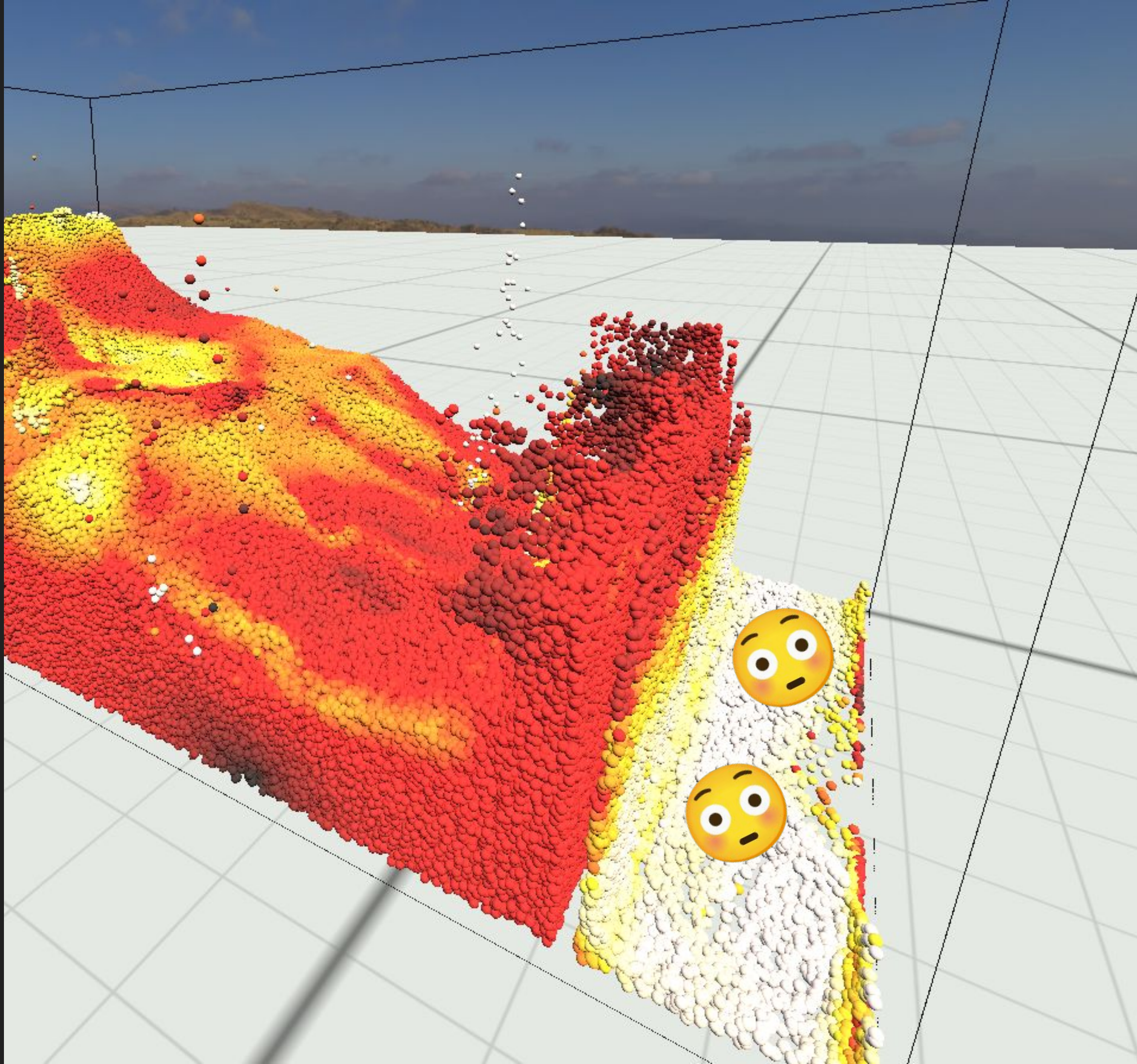
https://animation.rwth-aachen.de/media/papers/66/2019-TVCG-ImplicitDensityProjection.pdf

Voxelization

- To keep particles out of solids

- **It’s a hack!**
- Textbook impl needs signed distance field

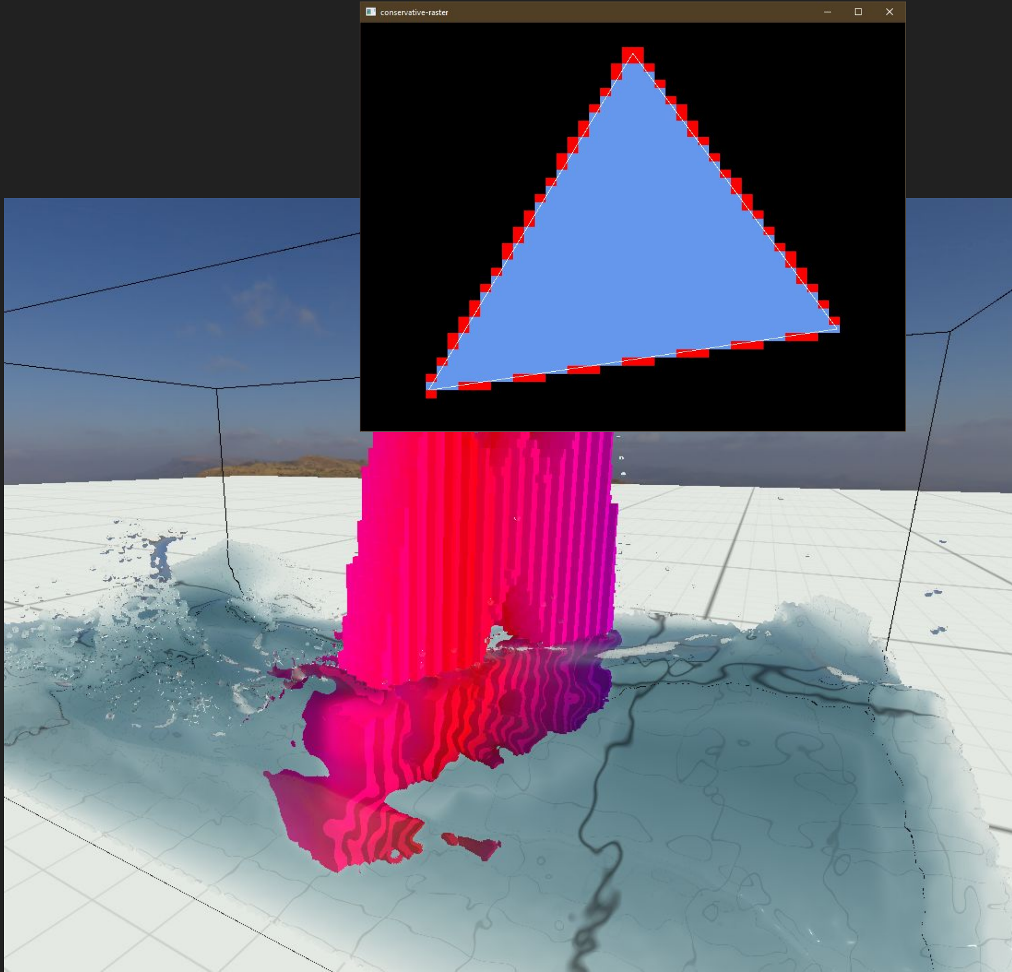
- Instead particles *try* to predict & avoid



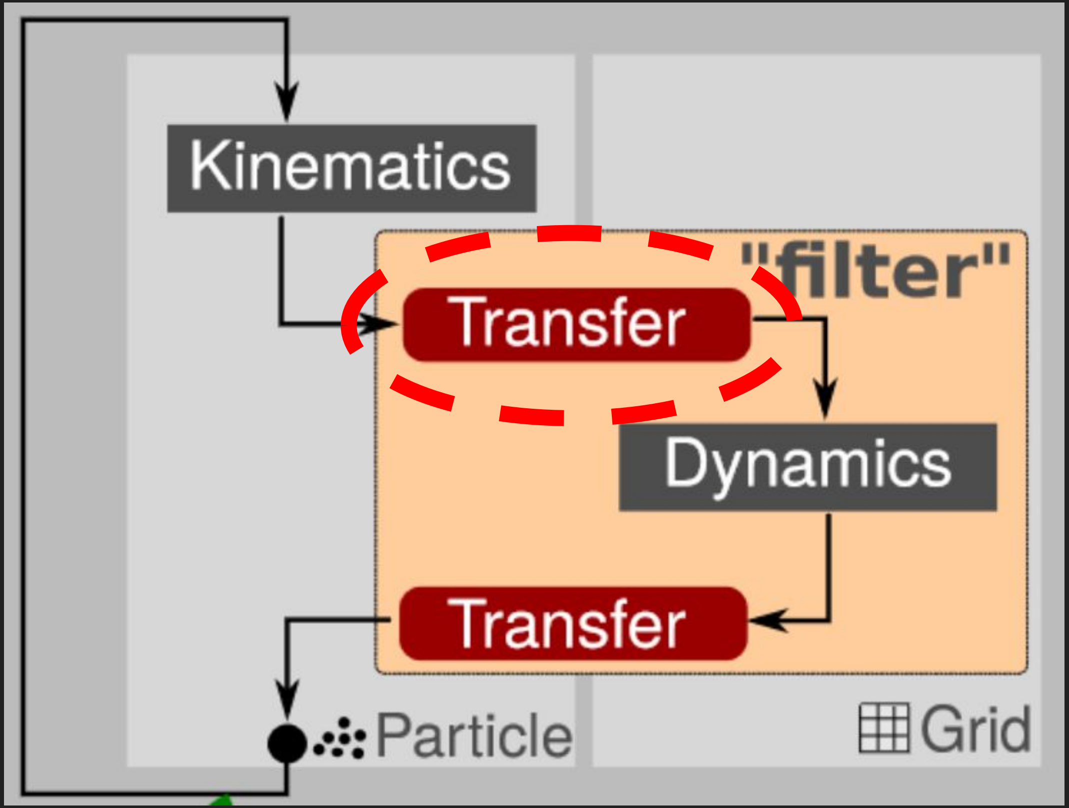
Voxelization

- Single Render Pass!
- Uses Conservative Rasterization

- Vertex Shader
  - Decide Dominant Axis
- Fragment shader
  - Write to all touched Voxel
    - Write Voxel’s velocity (packed)
  - don’t write to render target 🐱👤



Velocity to Grid

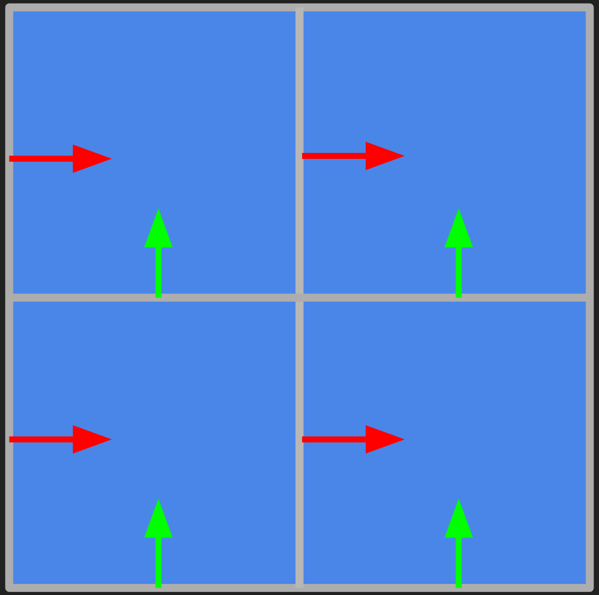


https://www.math.ucla.edu/~jteran/papers/JSSTS15.pdf The Affine Particle-In-Cell Method, Jiang et a.

Velocity to Grid - Staggered Grid

- Velocities are not at center, but at walls

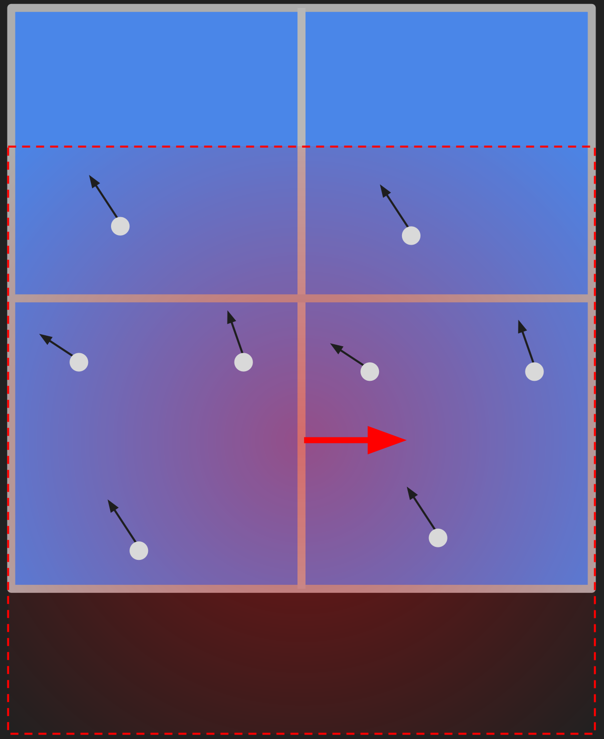
- X is not located where Y is
- Separate transfer passed per axis 😔
  - Otherwise affected area becomes too large



Velocity to Grid - Staggered Grid

- Velocities are not at center, but at walls

- X is not located where Y is
- Separate transfer passed per axis 😔
  - Otherwise affected area becomes too large



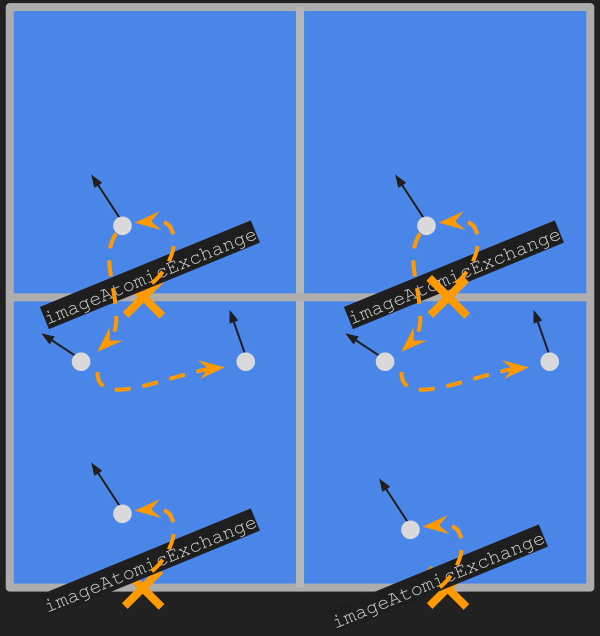
Velocity to Grid - Possibilities

- **Scatter:** Particles write to cell
    - No float atomics available!

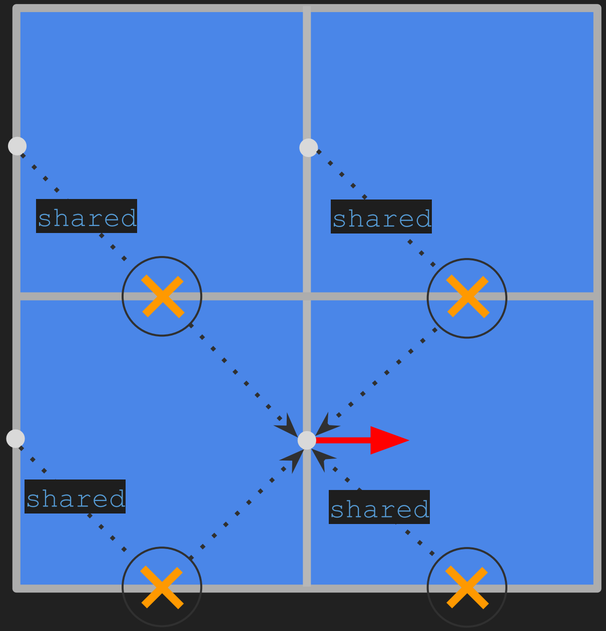
- **Gather:** Cells go through particles
    - Need to find the right particles

Velocity to Grid - Blub’s **Gather**

Create Linked List Volume



Average Particles



Pressure Solver

✗ 😣

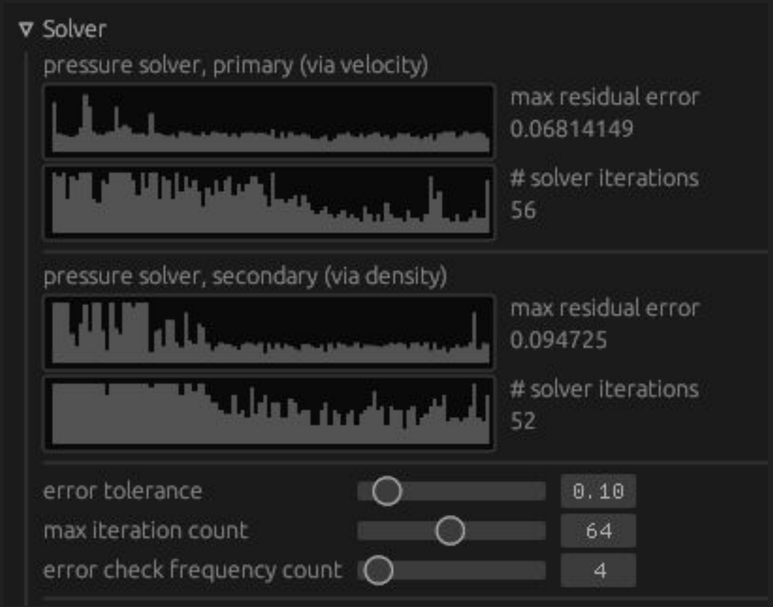
- Solve `Ax = b` with sparse `A`

- Blub uses `Preconditioned Conjugate Gradient Solver`
  - Use GPU friendly preconditioner (many aren't)
  - Excellent description here https://github.com/austinEng/WebGL-PIC-FLIP-Fluid#pressure-solve

Pressure Solver

❌ 😣

- What’s needed in Compute Shader Terms
  - Neighboring sampling in grid
  - Prefix sums
  - Tons of iterations
    - **How many??**
- Tried reading error back and decide on CPU
  - Delay too big!
- Instead
  - everything `dispatch_indirect`
  - Check error every n iterations



▼ Solver

pressure solver, primary (via velocity)

max residual error 0.06814149

solver iterations 56

pressure solver, secondary (via density)

max residual error 0.094725

solver iterations 52

error tolerance 0.10

max iteration count 64

error check frequency count 4

Want to learn more?

- Blub Readme
  https://github.com/Wumpf/blub#readme

- Various links to Fluid Sim resources
  https://gist.github.com/Wumpf/b3e953984de8b0efdf2c65e827a1ccc3

Q&A