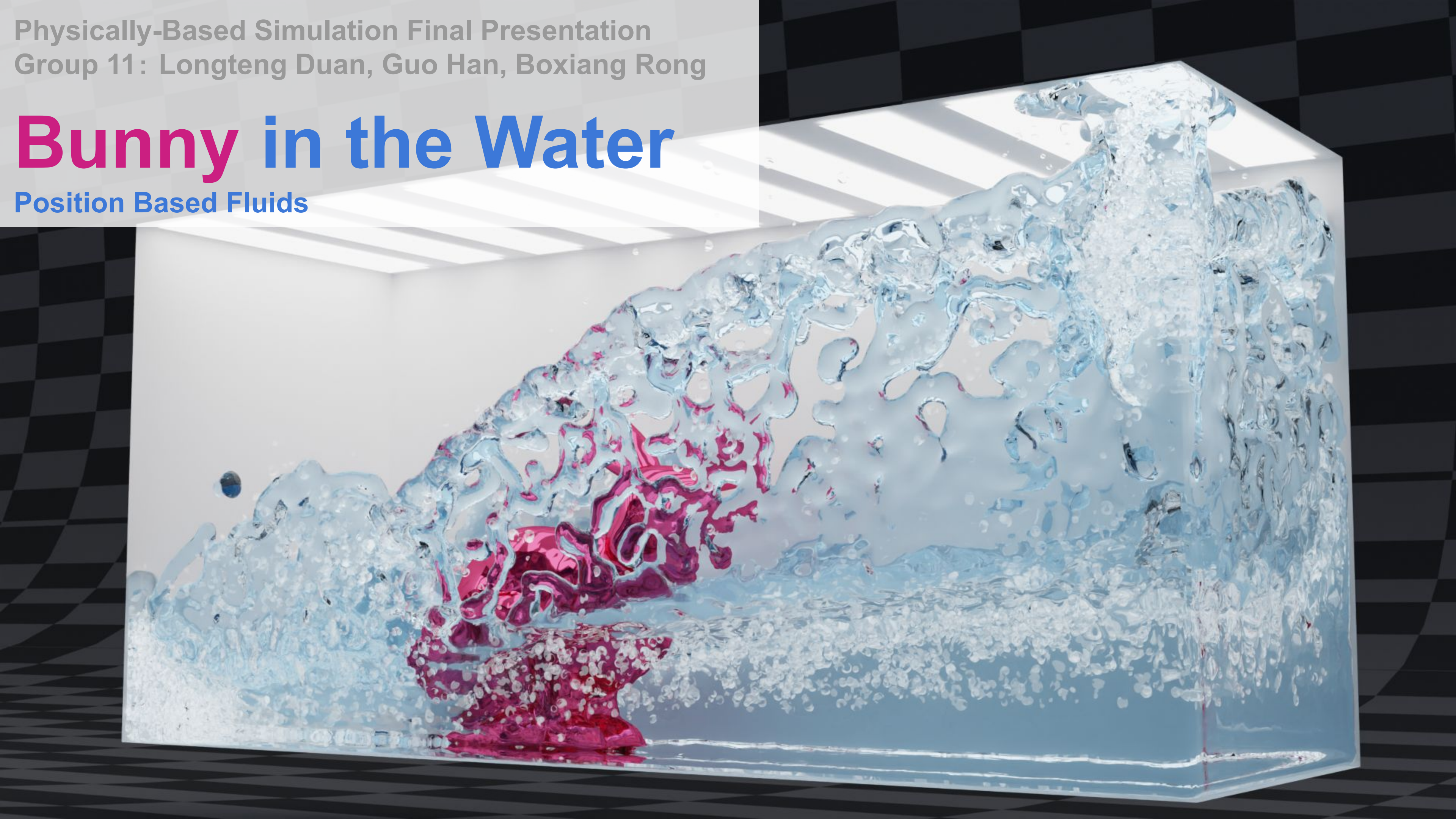
Physically-Based Simulation Final Presentation

Group 11：Longteng Duan, Guo Han, Boxiang Rong

Bunny in the Water

Position Based Fluids



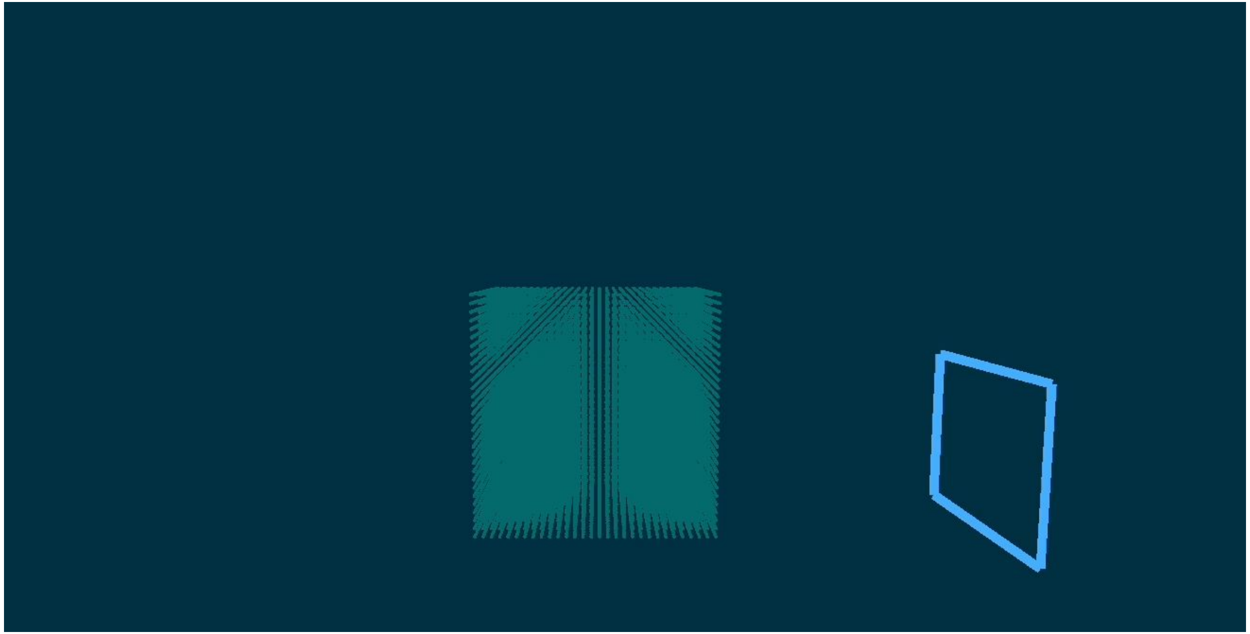
Feature Overview

- 3D Position-based fluids simulation
- Static rigid body collision detection and response within fluid
- Diffuse materials incorporating spray, foam, and bubbles
- CPU/GPU parallelization empowered by Taichi
- Real-time particle visualization via Taichi GGUI
- Offline rendering using Blender

Successfully accomplish all the targets outlined in the proposal!

Progress at Milestone

- 3D Position-based fluids simulation



Progress Since Then

- **Static rigid body** collision detection and response within fluid
- **Diffuse materials** incorporating spray, foam, and bubbles
- **CPU/GPU parallelization** empowered by Taichi
- **Offline rendering** using Blender

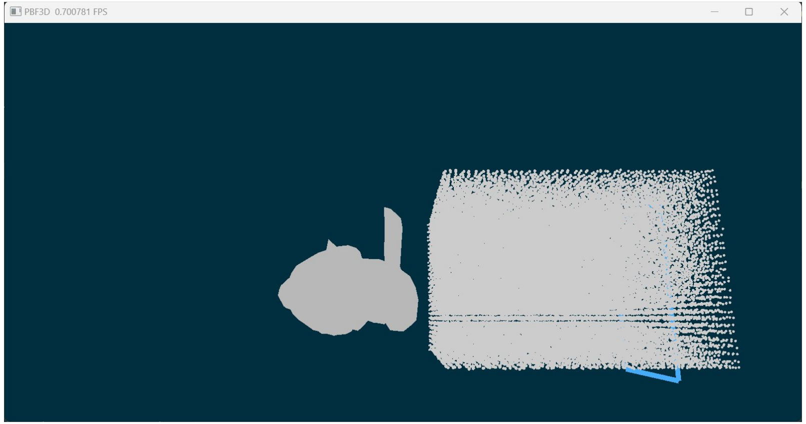
Static Rigid Body

Static Rigid Body

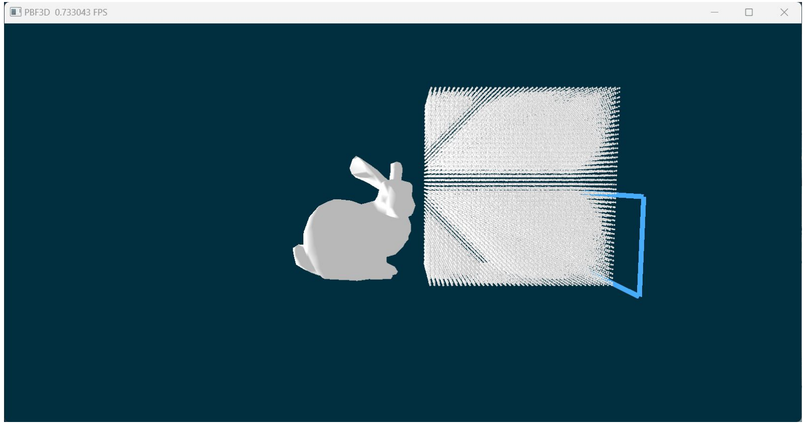
- Consideration limited to **a static watertight mesh**

Static Rigid Body

- Consideration limited to **a static watertight mesh**
- **Collision Detection**
  - Broad phase: axis-aligned bounding box (AABB)



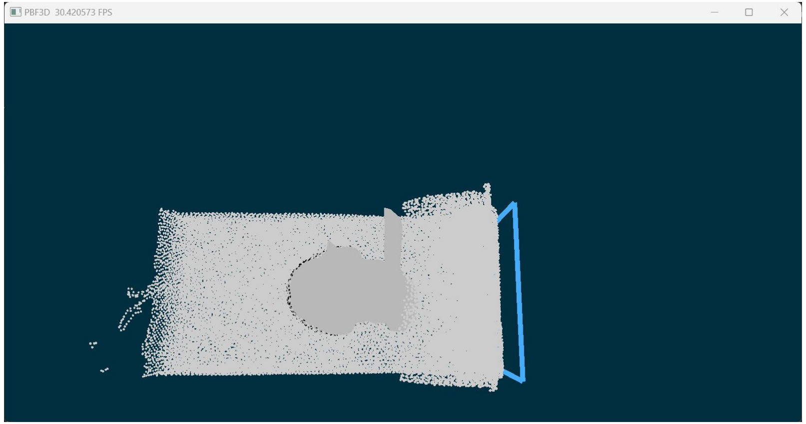
AABB rough detection top-down view



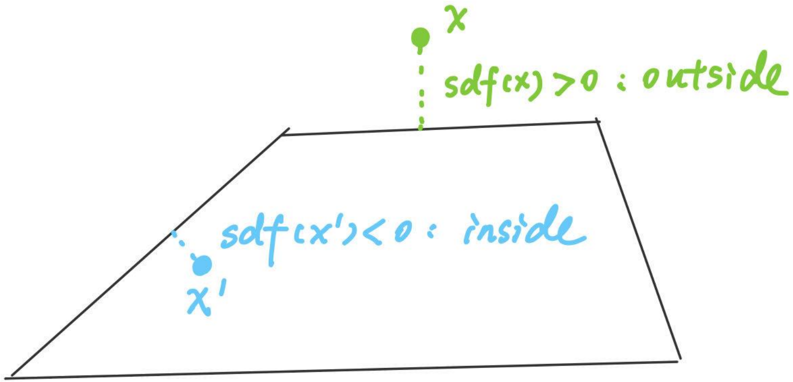
AABB rough detection side view

Static Rigid Body

- Consideration limited to **a static watertight mesh**
- **Collision Detection**
  - Broad phase: axis-aligned bounding box (AABB)
  - Narrow phase: signed distance function



SDF narrow detection top-down view

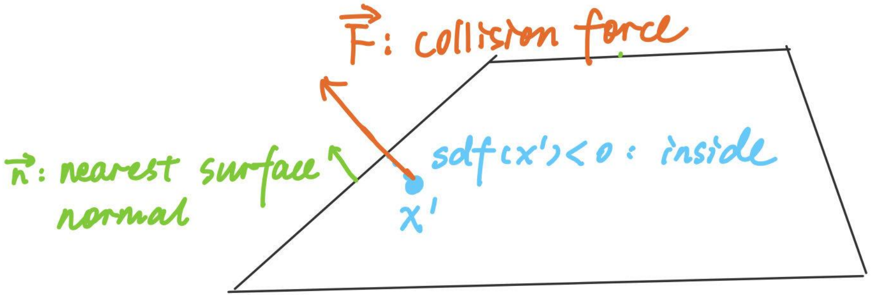


Static Rigid Body

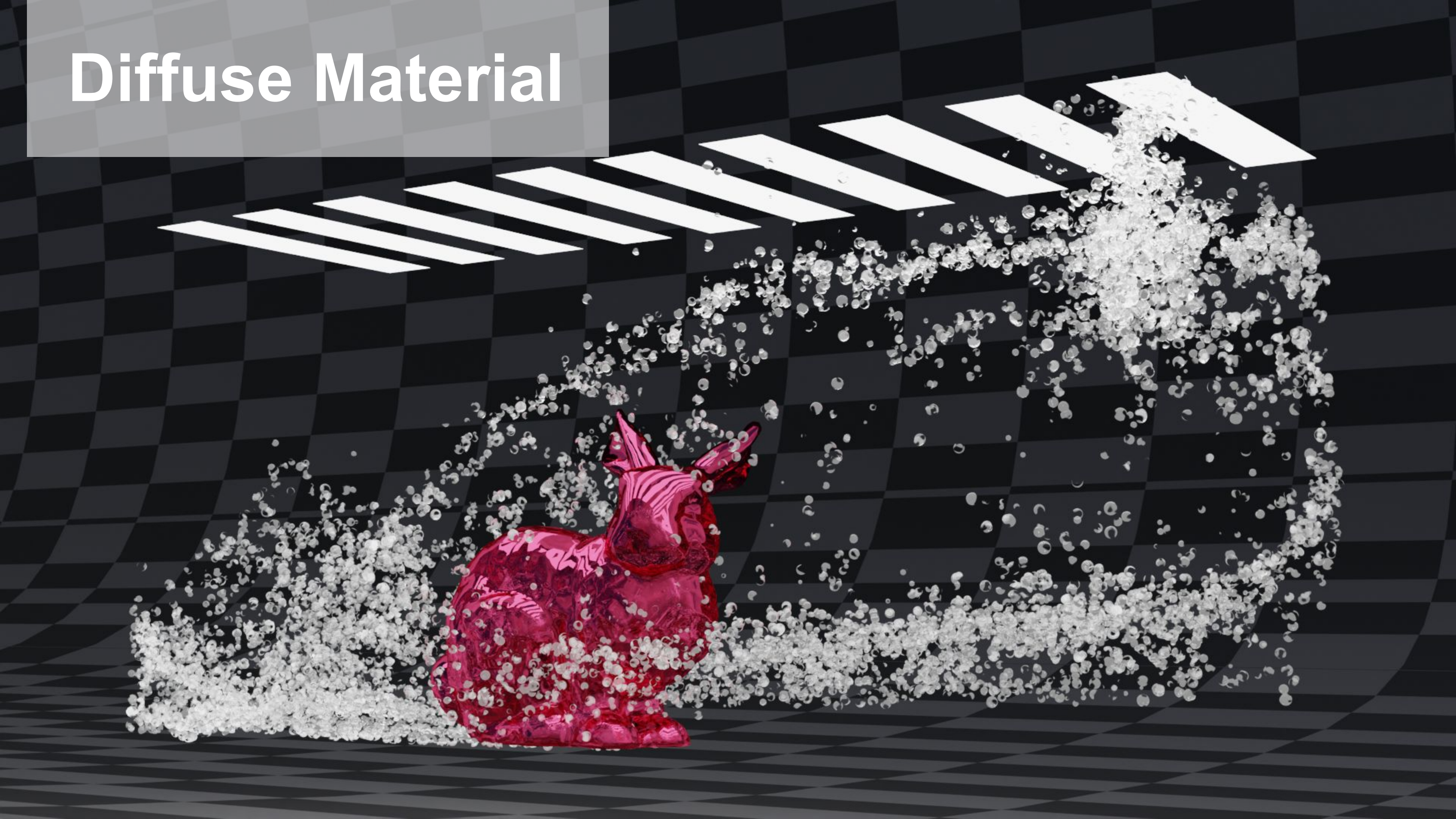
- Consideration limited to **a static watertight mesh**
- **Collision Response**
  - Apply a collision force to collided fluid particles

$$\vec{\mathbf{F}} = \text{stiffness} * \vec{\mathbf{n}} * (-\text{sdf}(\mathbf{x}))$$

Collision Response



Diffuse Material



Diffuse Material

- **Water-air mixtures: spray, foam and air bubbles**
- **Goal:** measure the potential of each fluid particle to mix with air

[1] Ihmsen, Markus & Akinci, Nadir & Akinci, Gizem & Teschner, Matthias. (2012). Unified spray, foam and air bubbles for particle-based fluids. The Visual Computer. 28. 669-677. 10.1007/s00371-012-0697-9.

Diffuse Material

- **Water-air mixtures: spray, foam and air bubbles**
- **Goal:** measure the potential of each fluid particle to mix with air
- **4 Potentials[1]:**
  - The potential to trap air I_{ta}
    - e.g. lip of wave falls down

[1] Ihmsen, Markus & Akinci, Nadir & Akinci, Gizem & Teschner, Matthias. (2012). Unified spray, foam and air bubbles for particle-based fluids. The Visual Computer. 28. 669-677. 10.1007/s00371-012-0697-9.

Diffuse Material

- **Water-air mixtures: spray, foam and air bubbles**
- **Goal:** measure the potential of each fluid particle to mix with air
- **4 Potentials[1]:**
  - The potential to trap air I_{ta}
  - The likelihood to be at the crest of a wave I_{wc}
    - wave crest breaks in case of strong wind
    - wave crest starts to fall and break when base is unstable

[1] Ihmsen, Markus & Akinci, Nadir & Akinci, Gizem & Teschner, Matthias. (2012). Unified spray, foam and air bubbles for particle-based fluids. The Visual Computer. 28. 669-677. 10.1007/s00371-012-0697-9.

Diffuse Material

- **Water-air mixtures: spray, foam and air bubbles**
- **Goal:** measure the potential of each fluid particle to mix with air
- **4 Potentials[1]:**
  - The potential to trap air I_{ta}
  - The likelihood to be at the crest of a wave I_{wc}
  - The vorticity difference I_{vo}
  - The kinetic energy I_{ke}

[1] Ihmsen, Markus & Akinci, Nadir & Akinci, Gizem & Teschner, Matthias. (2012). Unified spray, foam and air bubbles for particle-based fluids. The Visual Computer. 28. 669-677. 10.1007/s00371-012-0697-9.

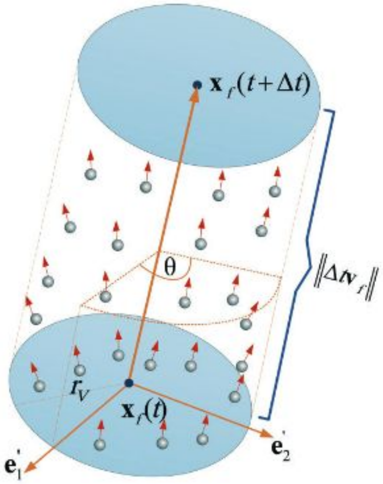
Diffuse Material

- **Water-air mixtures: spray, foam and air bubbles**
- **Goal:** measure the potential of each fluid particle to mix with air
- **4 Potentials[1]:**
  - The potential to trap air I_{ta}
  - The likelihood to be at the crest of a wave I_{wc}
  - The vorticity difference I_{vo}
  - The kinetic energy I_{ke}
- **Number of new diffuse particles =** $F(I_{ta}, I_{wc}, I_{vo}, I_{ke})$

[1] Ihmsen, Markus & Akinci, Nadir & Akinci, Gizem & Teschner, Matthias. (2012). Unified spray, foam and air bubbles for particle-based fluids. The Visual Computer. 28. 669-677. 10.1007/s00371-012-0697-9.

Diffuse Material

- **Generate foam** for each fluid particle $\boldsymbol{i}$



- Build a unit cylinder
- Uniformly sample position and velocity
  - $X_r,\ X_\theta,\ X_h \in [0,1]$
  - $r = r_V\sqrt{X_r}$
    $\theta = X_\theta 2\pi$
    $h = X_h \cdot ||\Delta t V_f||$
  - $x_d = x_f + r\cos\theta e_1' + r\sin\theta e_2' + h\hat{V}_f$
    $v_d = r\cos\theta e_1' + r\sin\theta e_2' + V_f$

[1] Ihmsen, Markus & Akinci, Nadir & Akinci, Gizem & Teschner, Matthias. (2012). Unified spray, foam and air bubbles for particle-based fluids. The Visual Computer. 28. 669-677. 10.1007/s00371-012-0697-9.

Diffuse Material

- **Pseudo Code:**

(simulation loop)

for all particles j **in** AllDiffuseParticles **do**
    removeParticles(AllDiffuseParticels)
    advectParticles(AllDiffuseParticels)

for all particles i **in** FluidParticles **do**
    $I_{ta}\ I_{wc}\ I_{ke}\ I_{vo}$ = computePotentials(x_i, v_i)
    AllDiffuseParticels $\leftarrow$ generateDiffuseParticles($I_{ta}\ I_{wc}\ I_{ke}\ I_{vo}$)

Diffuse Material

- **Pseudo Code:**

(simulation loop)

for all particles j **in** AllDiffuseParticles **do**
    removeParticles(AllDiffuseParticels) ← Remove diffuse particles that used up its lifetime
    advectParticles(AllDiffuseParticels)

for all particles i **in** FluidParticles **do**
    $I_{ta}\ I_{wc}\ I_{ke}\ I_{vo}$ = computePotentials(x_i, v_i)
    AllDiffuseParticels $\leftarrow$ generateDiffuseParticles($I_{ta}\ I_{wc}\ I_{ke}\ I_{vo}$)

Diffuse Material

- **Pseudo Code:**

(simulation loop)

for all particles j **in** AllDiffuseParticles **do**
    removeParticles(AllDiffuseParticels)
    advectParticles(AllDiffuseParticels) ← Change x_i, v_i of diffuse particles

for all particles i **in** FluidParticles **do**
    $I_{ta}\ I_{wc}\ I_{ke}\ I_{vo}$ = computePotentials(x_i, v_i)
    AllDiffuseParticels ← generateDiffuseParticles($I_{ta}\ I_{wc}\ I_{ke}\ I_{vo}$)

Diffuse Material

- **Pseudo Code:**

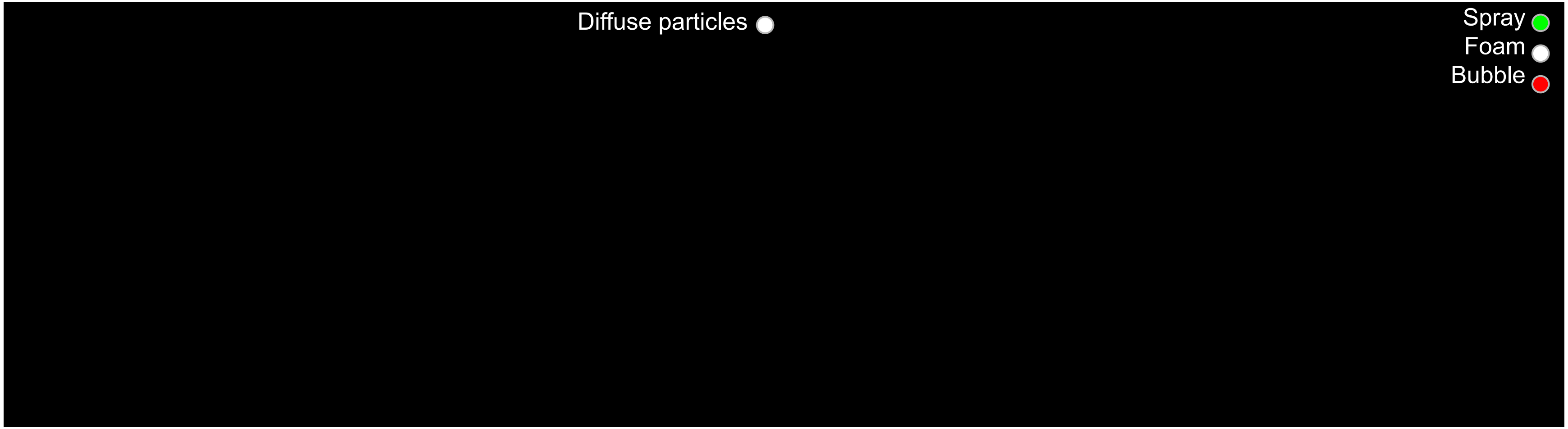
(simulation loop)

for all particles j **in** AllDiffuseParticles **do**
removeParticles(AllDiffuseParticels)
advectParticles(AllDiffuseParticels)

Generate new diffuse particles

for all particles i **in** FluidParticles **do**
$I_{ta}\ I_{wc}\ I_{ke}\ I_{vo}$ = computePotentials(x_i, v_i)
AllDiffuseParticels $\leftarrow$ generateDiffuseParticles($I_{ta}\ I_{wc}\ I_{ke}\ I_{vo}$)

Visualize: spray / foam / bubbles

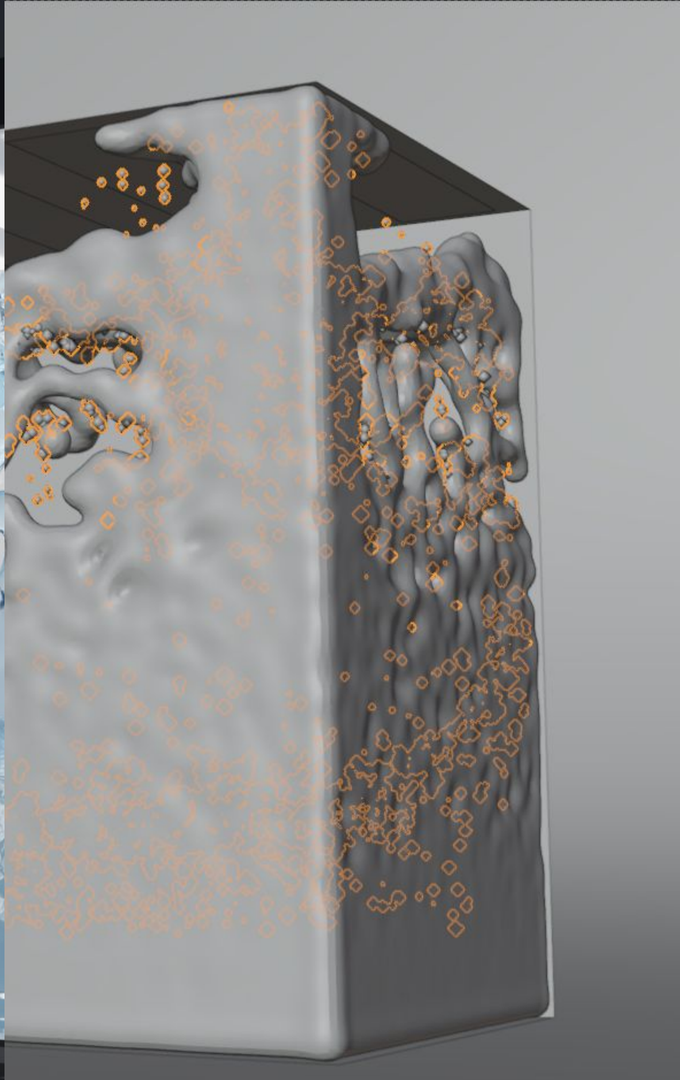


Classifying diffuse particles:

- Based on number of neighboring fluid particles
- Render with different materials

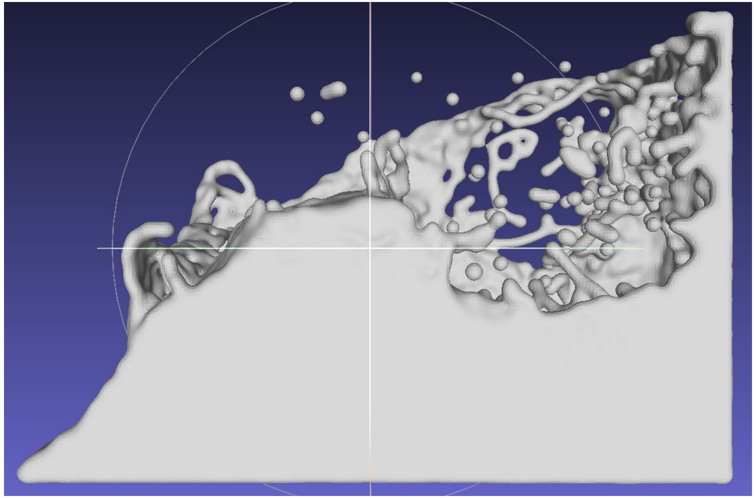
What We Have Now

Offline Rendering

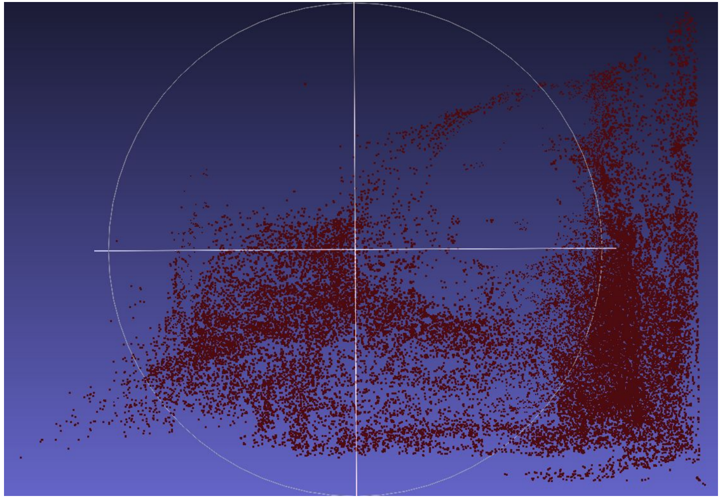


Offline Rendering

- **Bake mesh** with splashsurf and Open3D
  - Reconstruct fluid mesh with splashsurf
  - Export foam, bubble and spray as point clouds using Open3D



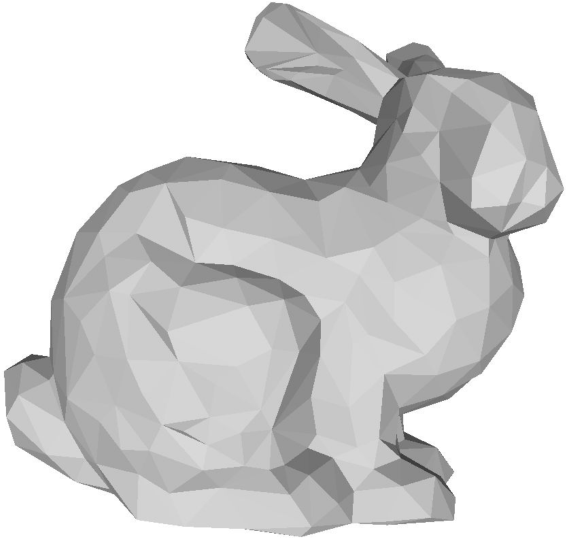
Fluid mesh



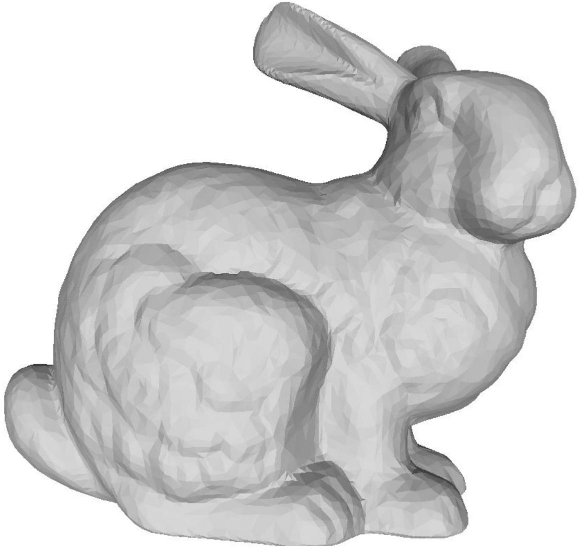
Diffuse material point cloud

Offline Rendering

- Use Blender Python API for rendering - **Rigid Body**



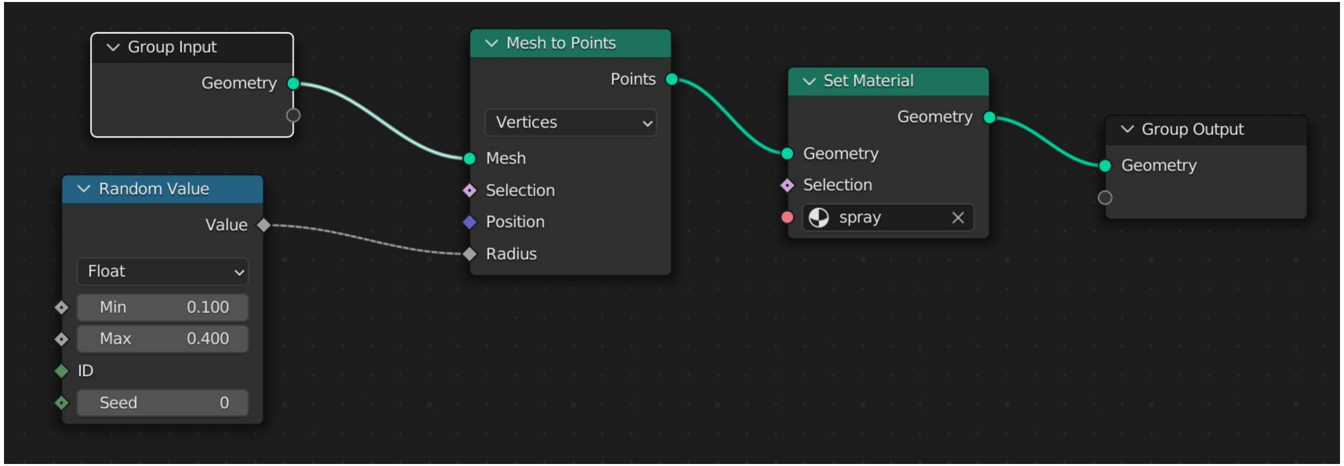
Low-poly watertight bunny mesh for simulation



High-poly bunny mesh for rendering

Offline Rendering

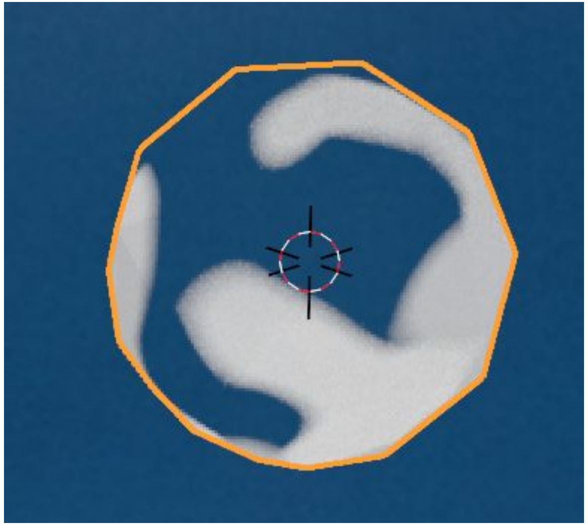
- Use Blender Python API for rendering - **Diffuse Material**
  - Apply *‘Mesh to Points’* node to **create spherical mesh** at diffuse particle positions with radius randomly assigned



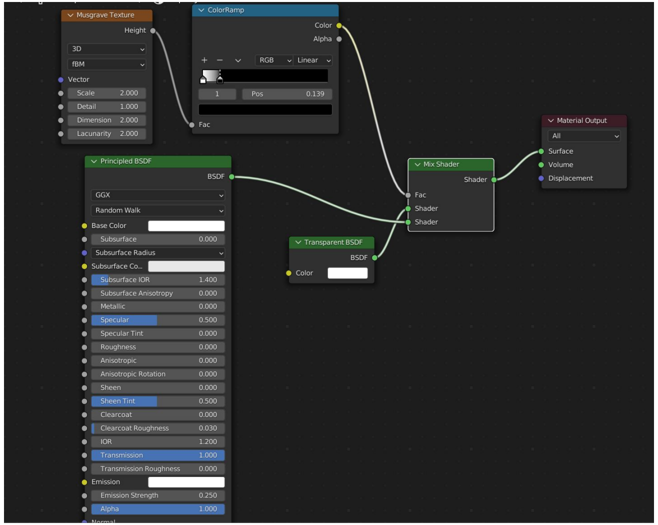
Blender setup to generate spherical mesh at diffuse particle positions

Offline Rendering

- Use Blender Python API for rendering - **Diffuse Material**
  - Utilize *'Musgrave texture', 'Transparent BSDF', and emission* settings to make diffuse materials both **partial emissive and partial transparent.**



Diffuse particle zoom in



Blender material setup for diffuse particles

ETH zürich

Final Demo

960px * 540px | 64 Samples | 61k Particles

References

[1] Macklin, Miles, and Matthias Müller. "Position based fluids." *ACM Transactions on Graphics (TOG)* 32.4 (2013): 1-12.

[2] Ihmsen, Markus, et al. "Unified spray, foam and air bubbles for particle-based fluids." *The Visual Computer* 28 (2012): 669-677.

[3] Taichi Blog for Collision Handling: https://docs.taichi-lang.org/blog/acclerate-collision-detection-with-taichi

[4] Taichi PBF 2D Example by Ye Kuang: https://github.com/taichi-dev/taichi/blob/master/python/taichi/examples/simulation/pbf2d.py

[5] SPlisHSPlasH Library for Diffuse Particles Synthesis: https://github.com/InteractiveComputerGraphics/SPlisHSPlasH

CPU/GPU Parallelization

- Taichi kernels automatically parallelize for-loops at the outermost level.

- The initial code for generating and removing foam relies on a **global counter** named *foam_counter*.
- **Serialization** is crucial to guarantee correct foam statuses update

```python
@ti.kernel
def generateFoam(self,):
    for idx in self.positions:
...
            for i in range(15):
                if int(self.foam_counter[None]) >= self.max_num_white_particles: continue
...
                self.foam_positions[self.foam_counter[None]] = xd
                self.foam_velocities[self.foam_counter[None]] = vd
                self.foam_lifetime[self.foam_counter[None]] = life
                # self.foam_type.append(0)
                ti.atomic_add(self.foam_counter[None], 1)
```

Taichi's parallelization optimization cannot be leveraged in this implementation.

Appendix / CPU/GPU Parallelization

- Taichi kernels automatically parallelize for-loops at the outermost level.

- **Global counter** to **local counter**
- Borrow the **grid design** from fluid simulation when generating foam
  - Each fluid particle has at most *x* foam particle
  - Foam generation is independent for each particle

```
@ti.kernel
def generateFoam(self,):
    for idx in self.positions:
...
        for i in range(num):
            if self.particle_to_foam_counter[idx] >= self.max_foam_per_particle:
                break
...
            self.particle_to_foam_grid[idx, self.particle_to_foam_counter[idx]]
            ti.atomic_add(self.particle_to_foam_counter[idx], 1)
```

Taichi's parallelization optimization can be used now!!!