Technische Universität München

# Large-Scale Liquid Simulation on Adaptive Hexahedral Grids

Florian Ferstl, Rüdiger Westermann and Christian Dick

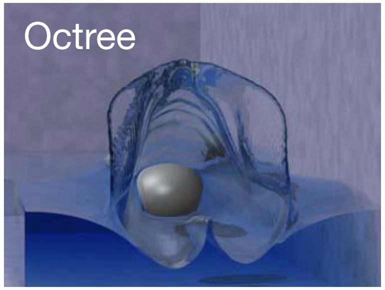
July 22, 2014

# Introduction & Related Work

- Goal: Large-scale liquid simulation
  - Adaptive octree grid
  - Effective resolutions ≥ $1024^3$

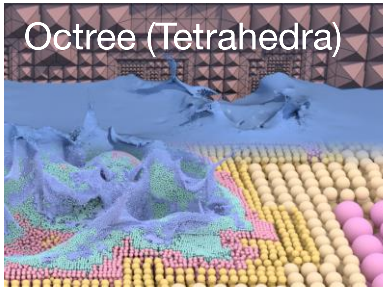
- Challenges
  - Memory consumption
  - Consistent, adaptive discretization
  - Performance (bottleneck: pressure solve)

- Our method combines
  - Adaptive octree grid
  - Multigrid solver (for irregular, adaptive grid)
  - FEM discretization (hexahedral elements):
    **Element matrix based formulation**



Losasso et al., 2004

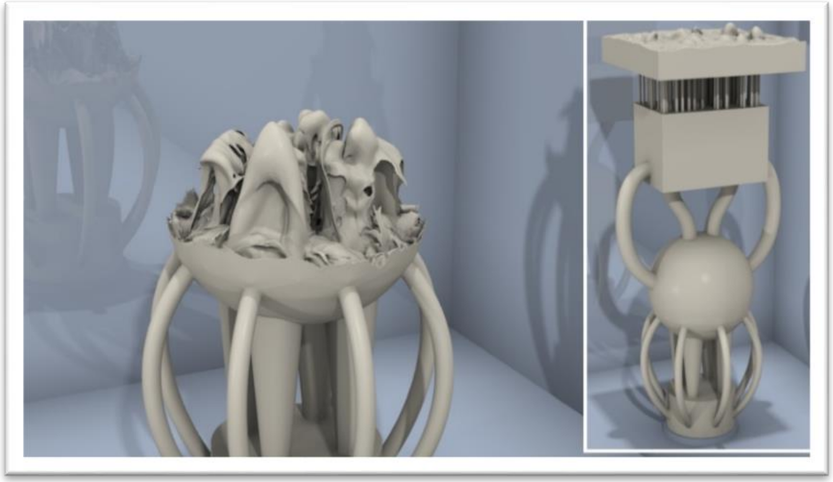
McAdams et al., 2010

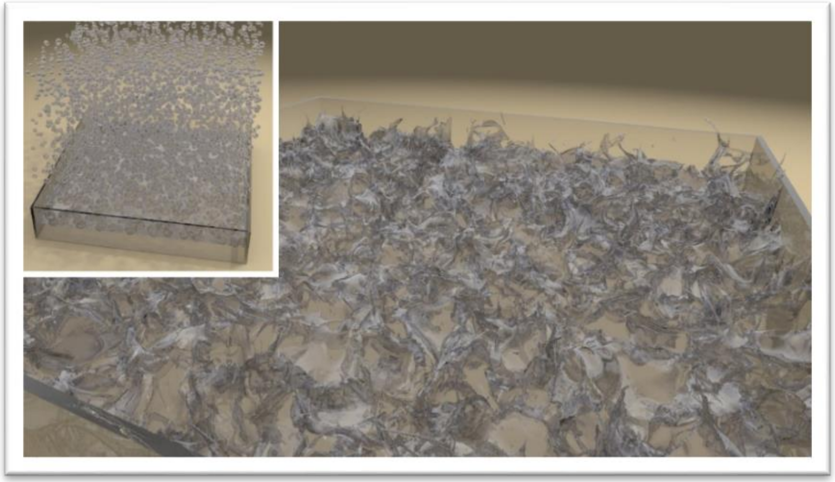


Ando et al., 2013

# Outline

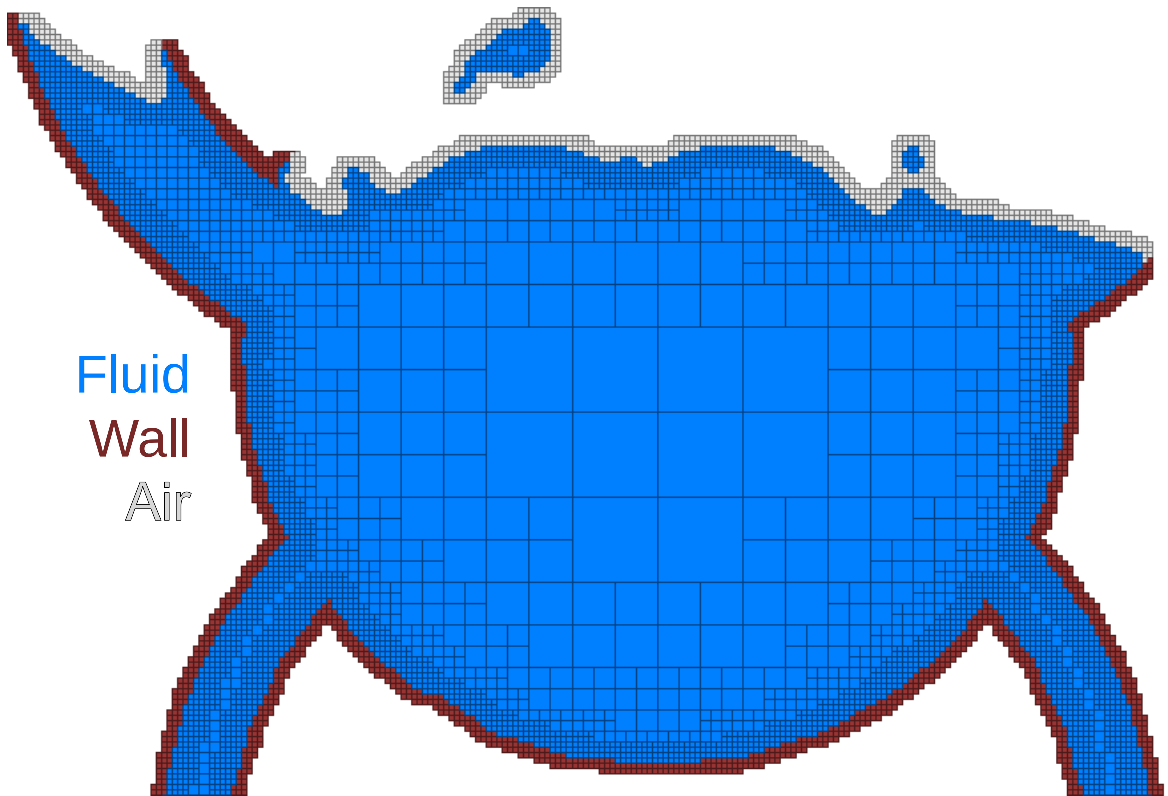
- Adaptive Octree Grid
- FEM Discretization & element matrices
- Hanging Vertices
- Second Order Boundary Conditions
- Multigrid Solver
- Results





# Adaptive Octree Grid

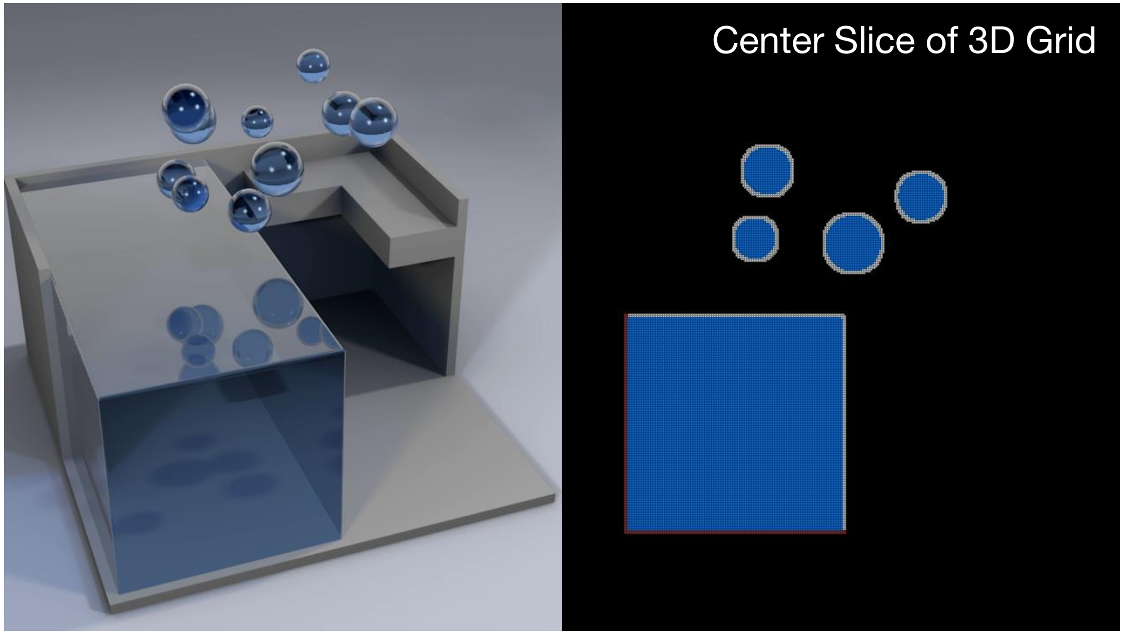
- Refinement strategy
  - Symmetric refinement band around surface (~5 cells wide)
  - Interior is held as coarse as possible
  - Octree is **restricted** to make resolution degrade smoothly towards interior
- Grid adapted after every surface advection step



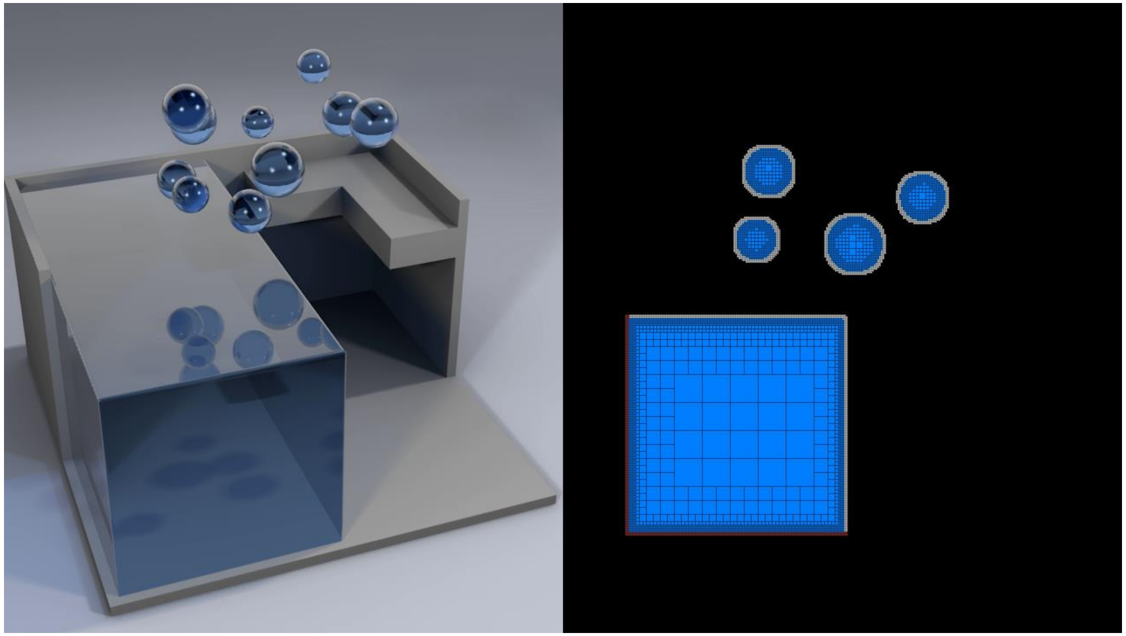

Slice of 3D Grid

# Grid Example

- Uncoarsened octree (= uniform grid), Resolution: $256^3$

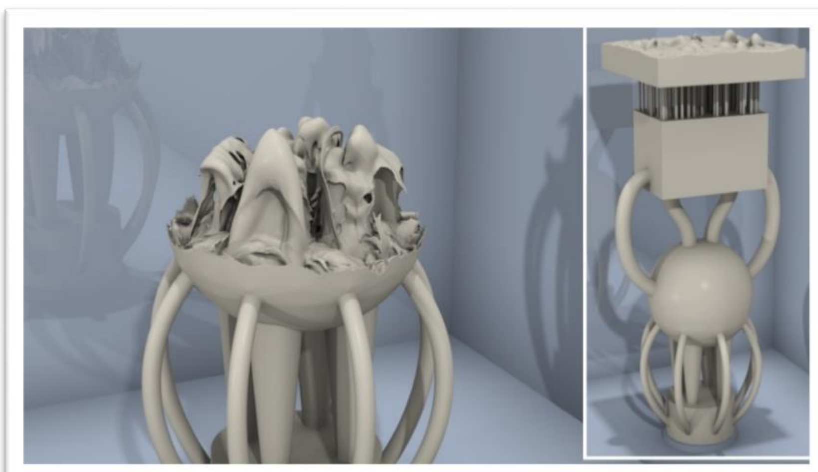


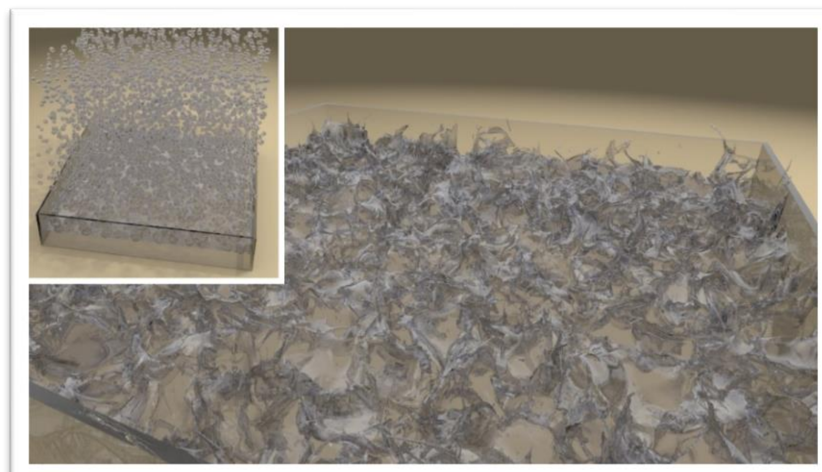

- Octree, Effective resolution: $256^3$



# Outline

- Adaptive Octree Grid
- FEM Discretization & element matrices
- Hanging Vertices
- Second Order Boundary Conditions
- Multigrid Solver
- Results





# Discretization

- Navier-Stokes Equations

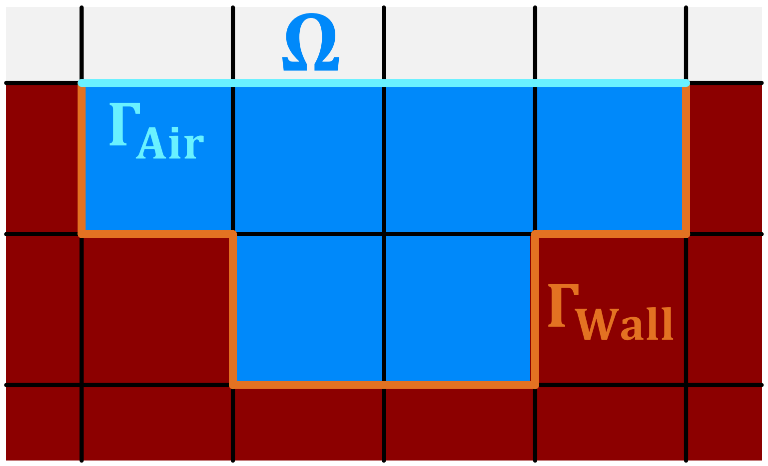
$$\dot{\boldsymbol{u}} = -\boldsymbol{u} \cdot \nabla \boldsymbol{u} + \frac{\mu}{\rho} \Delta \boldsymbol{u} - \frac{1}{\rho} \nabla p + \boldsymbol{g}$$

- Pressure Poisson Equation

$$\frac{1}{\rho} \Delta p = \frac{1}{\Delta t} \nabla \cdot \boldsymbol{u}^* \qquad \text{on } \Omega$$

$p$ prescribed on $\Gamma_{\text{Air}}$

$\boldsymbol{n} \cdot \boldsymbol{u}_\Gamma$ prescribed on $\Gamma_{Wall}$




- Time splitting:
  - Advection: Semi-Lagrange
  - Diffusion (Optional): FEM
  - External forces: FEM
  - Projection: FEM
- Liquid surface tracked with (particle-) level-set

# Finite Element Discretization

- Tri-linear ansatz functions $\phi_i$ for $\boldsymbol{u}$ and $p$
  → Co-located grid with all DOFs at cell vertices
- $\frac{1}{\rho}\Delta p = \frac{1}{\Delta t}\nabla \cdot \boldsymbol{u}^*$

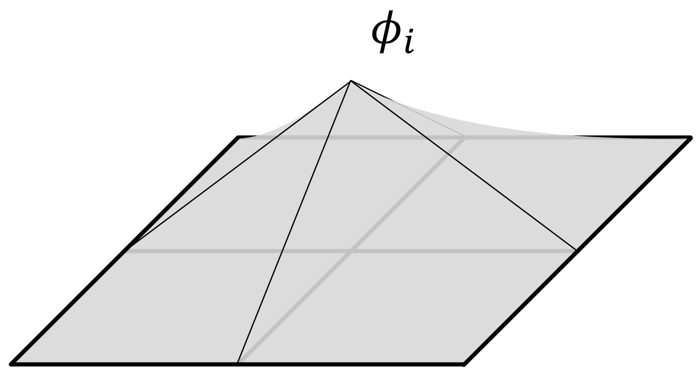
  $\downarrow$

  $Lp = D\boldsymbol{u}^* + B(\boldsymbol{u}^* - \boldsymbol{u}_\Gamma)$
- Entries of linear operators FEM are given by integration over ansatz functions, e.g. for $L$:

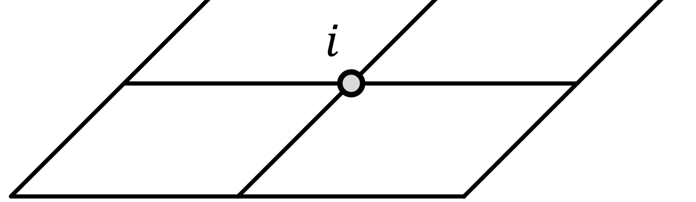
$$(L)_{i,j} = \int_\Omega \nabla\phi_i \cdot \nabla\phi_j$$

$$= \sum_{e\in\Omega} \boxed{\int_e \nabla\phi_i \cdot \nabla\phi_j}$$

Contribution of $e$ to $L$




"Hat function" (in 2D)



# FEM Element Matrices

- Every FEM operator can be expressed as a sum of *element matrices:*

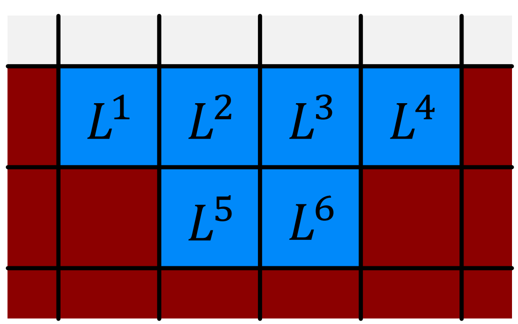
$$L = \sum_e \hat{L}^e \boxed{= \widehat{\sum_e} L^e}$$

sum of symmetric 8x8 matrices (in 3D)

- $L$ is sparse
- The $\hat{L}^e$ are **sparse** and **equal** (up to a cell-size scaling factor and row/column permutation)
- Element matrices $L^e \in \mathbb{R}^{8\times 8}$

$\rightarrow$ A single representative can be analytically pre-computed

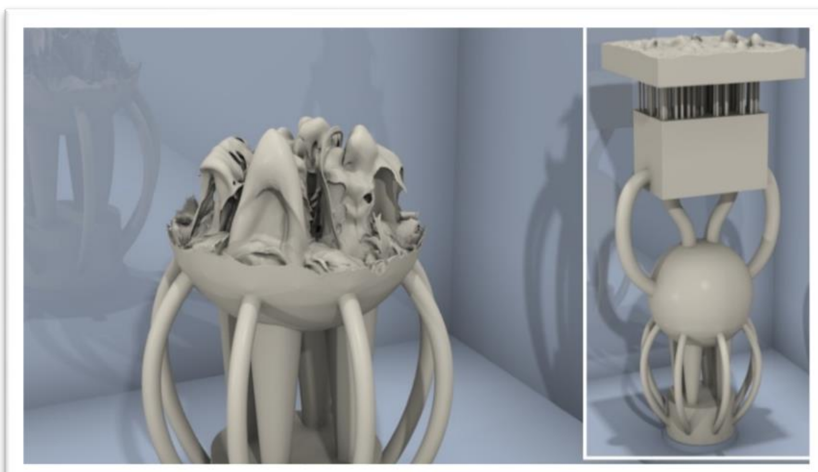
$\rightarrow$ Entries of L can be assembled on the fly from this representative

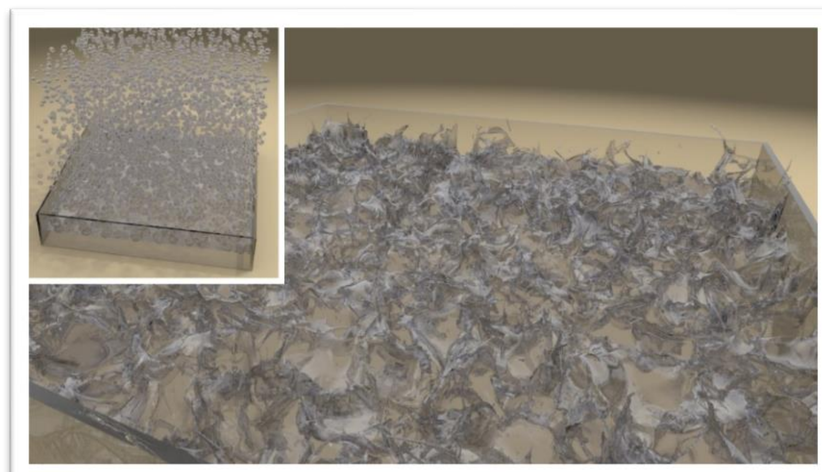



$$L = L^1 \hat{+} \ldots \hat{+} L^6$$

# Outline

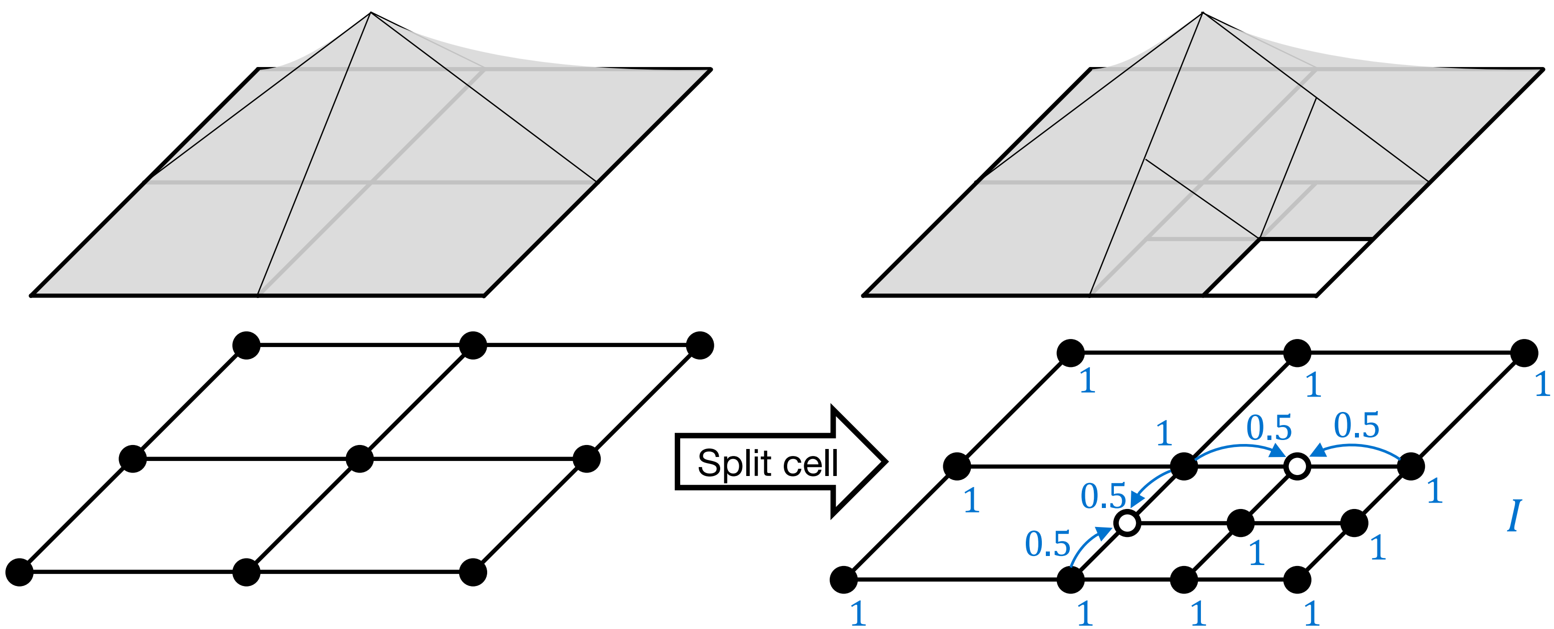
- Adaptive Octree Grid
- FEM Discretization & element matrices
- Hanging Vertices
- Second Order Boundary Conditions
- Multigrid Solver
- Results





# Hanging Vertices

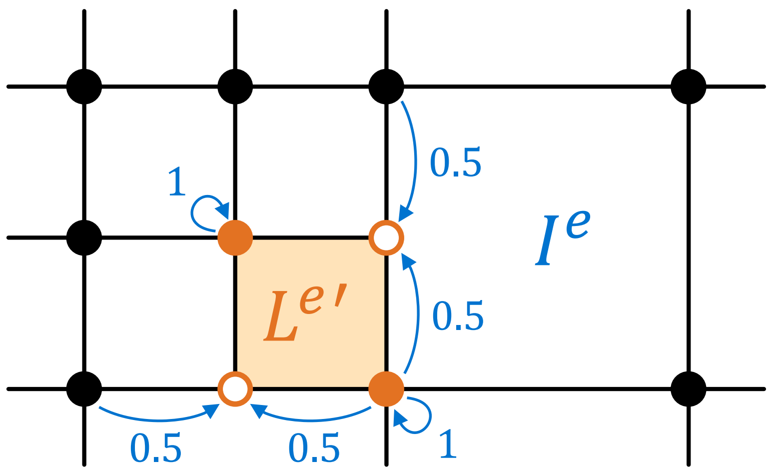
- Hanging vertices at grid level transitions are constrained to tri-linearly interpolated values
  - New basis functions are linear combinations of original basis functions
  - Formally:
    - $I$: Interpolation operator (non-hanging vertices to all vertices)
    - $L'$: “unconstrained FEM operator” (treating hanging vertices as DOFs)
    - Hanging vertex elimination: $L = I^T L' I$

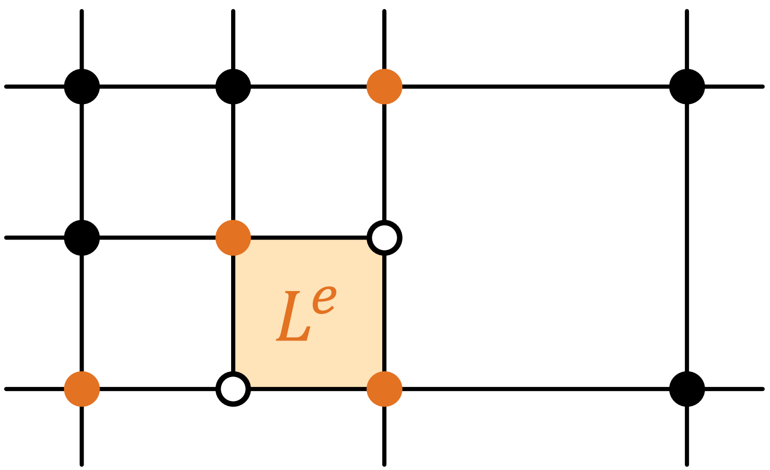


# Hanging Vertices

- Hanging vertex elimination: $L = I^T L' I$
- This can be done on element matrix level!
  - $L^{e\prime}$: relates the geometrically adjacent vertices of its cell
  - $L^e = (I^e)^T \, L^{e\prime} \, I^e$ (all $8x8$ matrices in 3D)
  - $L^e$ : relates the logically adjacent vertices of its cell
- Different hanging vertex configurations possible

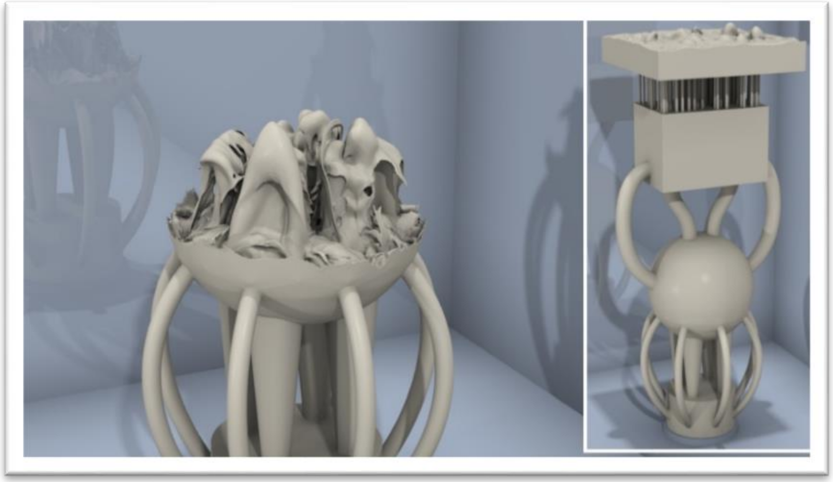
→ Instead of one precomputed $L^e$, lookup table with 512 precomputed $L^e$s

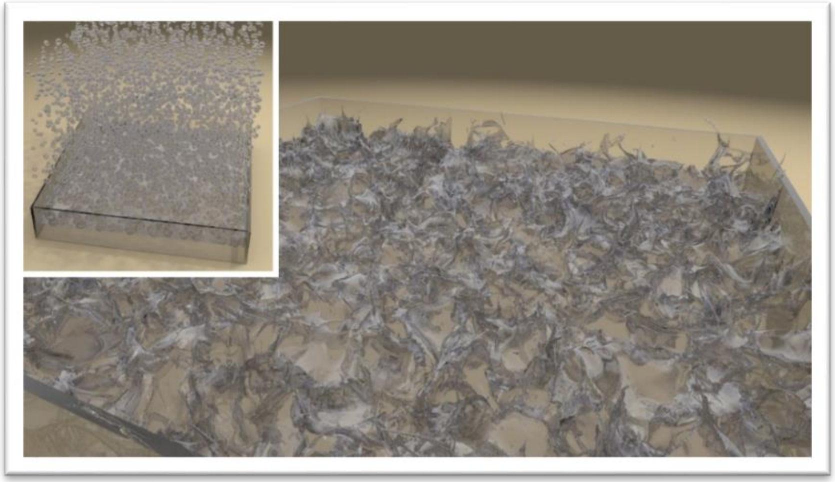


# Outline

- Adaptive Octree Grid
- FEM Discretization & element matrices
- Hanging Vertices
- Second Order Boundary Conditions
- Multigrid Solver
- Results

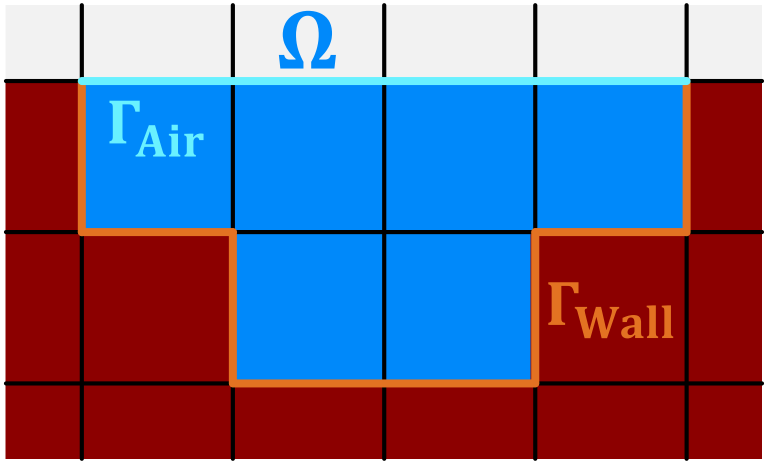




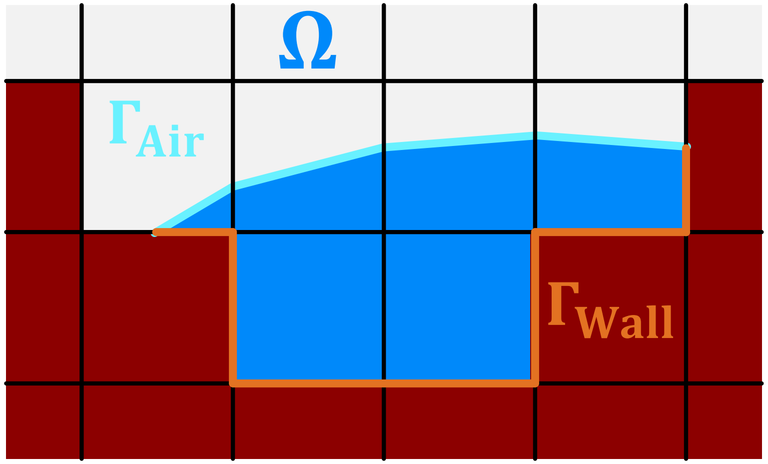
# Second Order Boundary Conditions

- Second order accurate BCs are essential for animation
- Ghost fluid method not applicable to the FEM discretization
- Our solution: An embedded interface method
  - Second order accurate
  - Currently only $\Gamma_{Air}$ treated with second order accuracy

First order



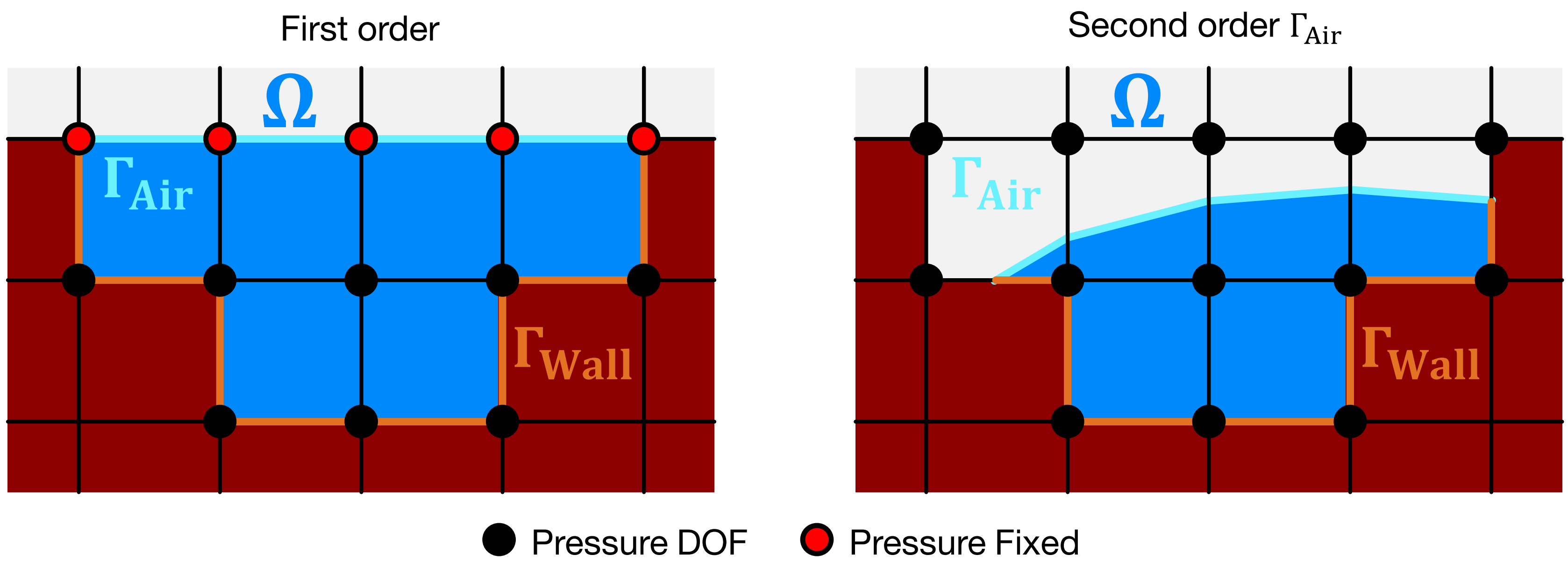

Second order $\Gamma_{\mathrm{Air}}$



# Boundaries: Formulation

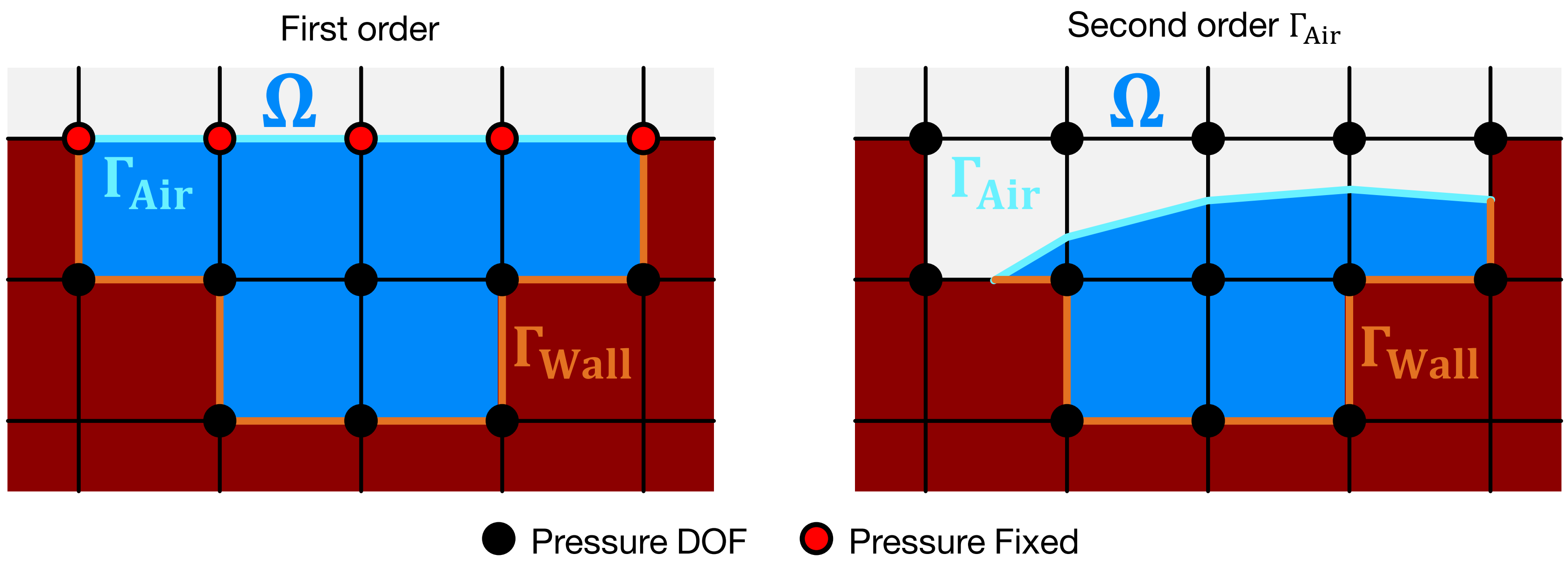
- Do not fix pressure at any vertices
- Add penalty term to enforce $p = 0$ at $\Gamma_{\text{Air}}$ in a variational sense
- Restrict integration to fluid filled portion of cells and the corresponding boundary:

$$(L)_{i,j} = \int_{\Omega} \nabla\phi_i \cdot \nabla\phi_j + \int_{\Gamma_{\text{Air}}} [\dots], \qquad (B)_{i,j} = \int_{\Gamma_{\text{Wall}}} [\dots]$$

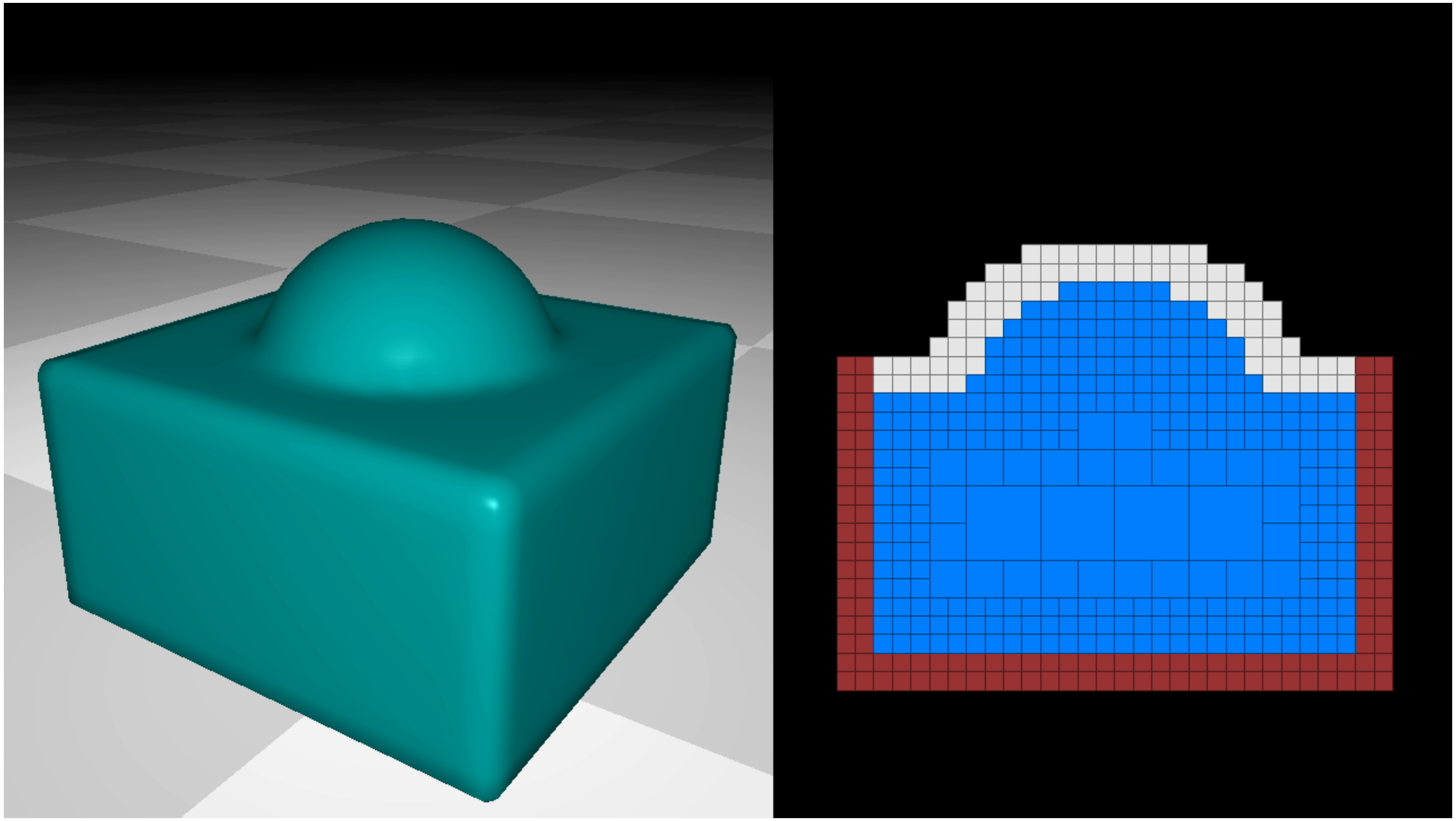


# Boundaries: Computation

- Changes only affect element matrices in boundary cells
- Compute those at runtime
  - Triangulate / tetrahedralize boundary cells using an extended, marching cubes style lookup table
  - Numerically integrate terms over the resulting tets / triangles
- Store element matrices explicitly (only) for boundary cells

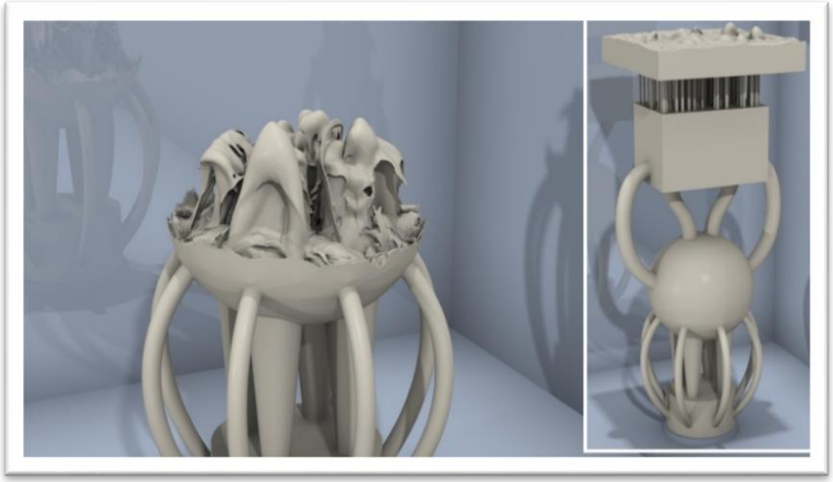


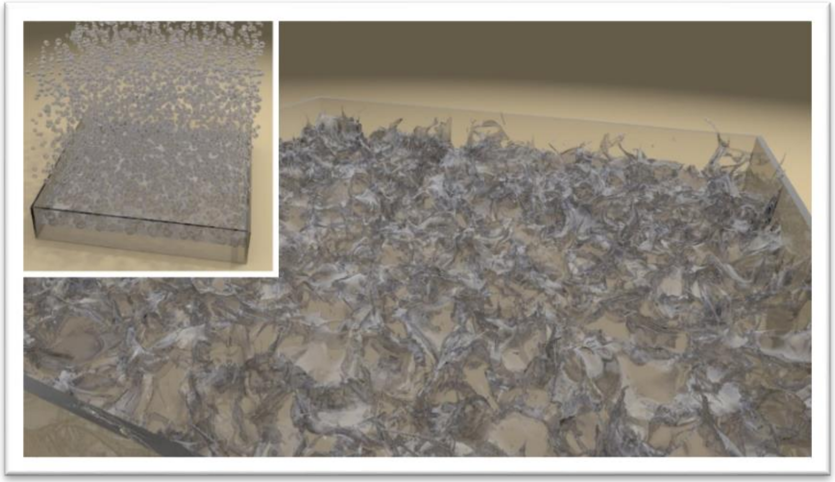
# Second Order Boundaries



# Outline

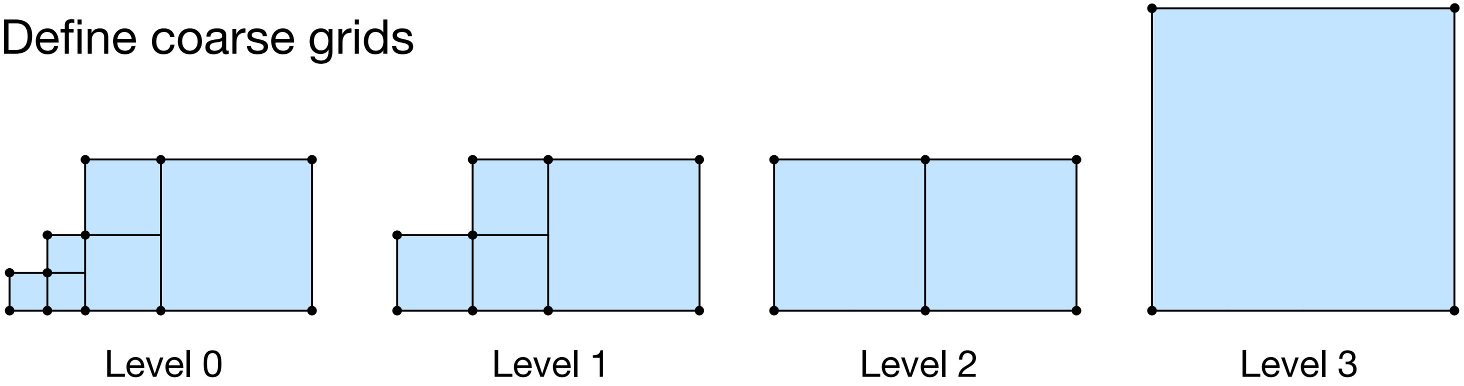
- Adaptive Octree Grid
- FEM Discretization & element matrices
- Hanging Vertices
- Second Order Boundary Conditions
- Multigrid Solver
- Results





# Multigrid Solver

- Galerkin-based coarsening
  - Define coarse grids




  - Define interpolation operators $I_{2h}^{h}$ (tri-linear interpolation)
  - Compute coarse grid operators according to

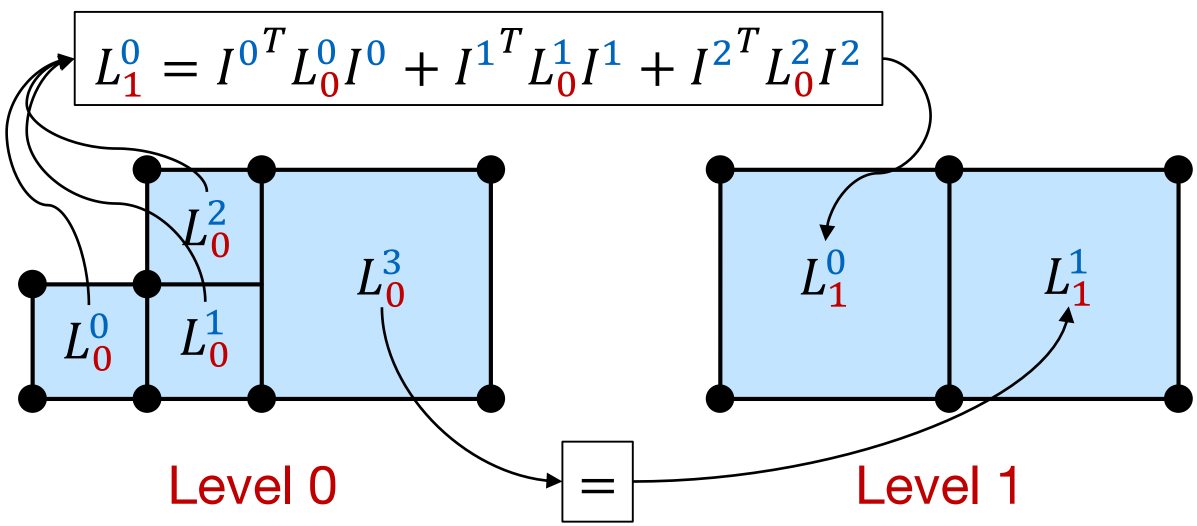
$$L_{2h} = \left(I_{2h}^{h}\right)^{T} L_{h} \, I_{2h}^{h}$$

- Solver performs standard V-Cycles
  - Gauss-Seidel smoother, PCG on coarsest level
  - MG efficient and stable if used as standalone solver, but even faster if used as preconditioner for CG

# Element Matrix Coarsening

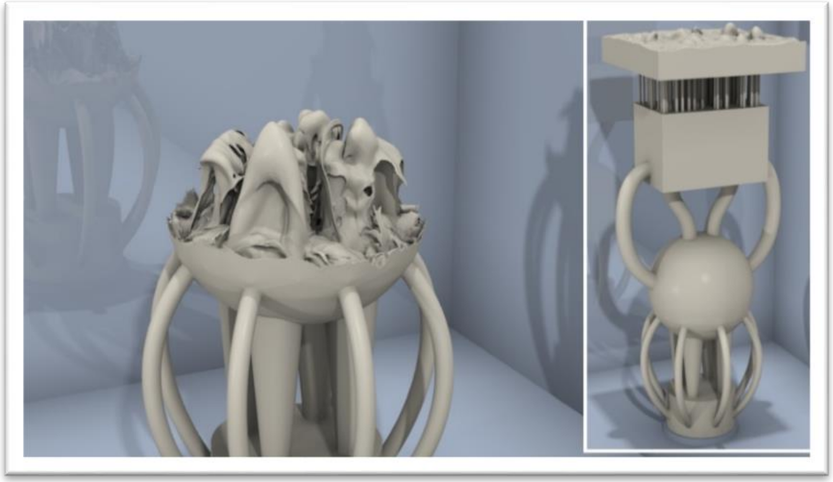
- Galerkin-based coarsening can be done on element matrix level!
  - All coarse grid operators can be written as sum of element matrices $L_l^e$
  - On coarser levels, these are purely mathematic constructs

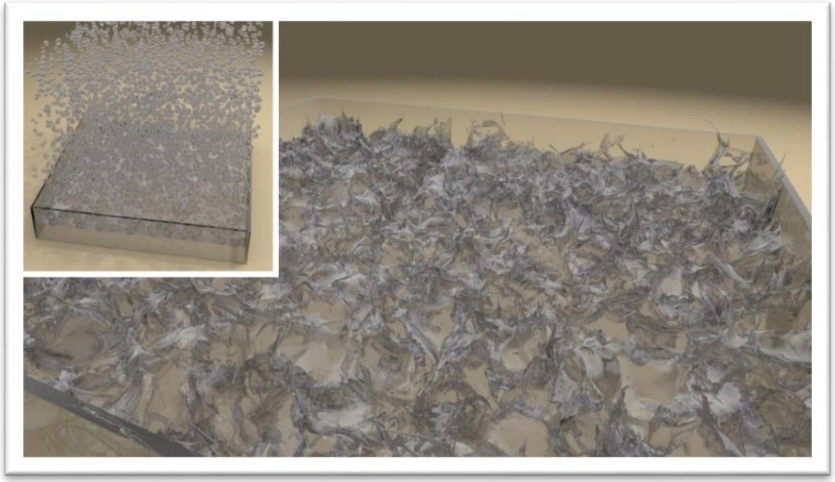
- Each coarse grid element matrix can be computed from its children fine grid element matrices



# A few more things…

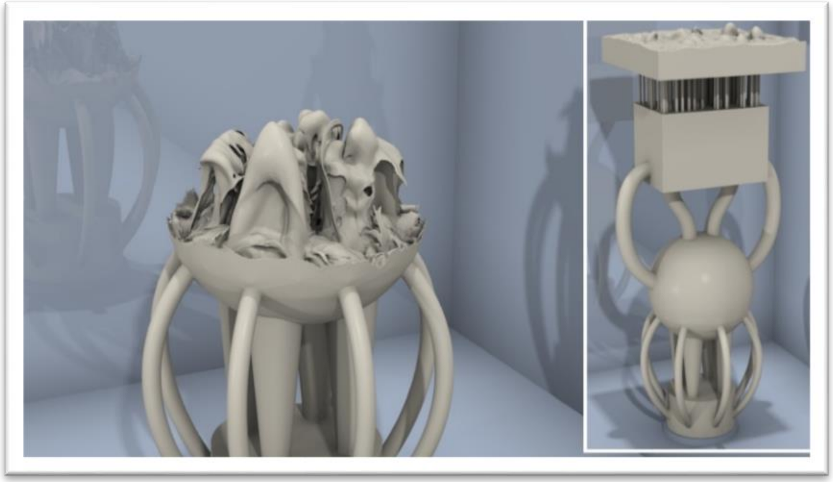
- Optional MG extension: Cell duplication
  - Complex, branching domains are often represented poorly on coarse grids (→ bad solver convergence)
  - Duplicate cells (and vertices) where necessary using a connectivity graph to track topology
- Parallelization: 90% parallelized (in terms of run-time on a single-core)

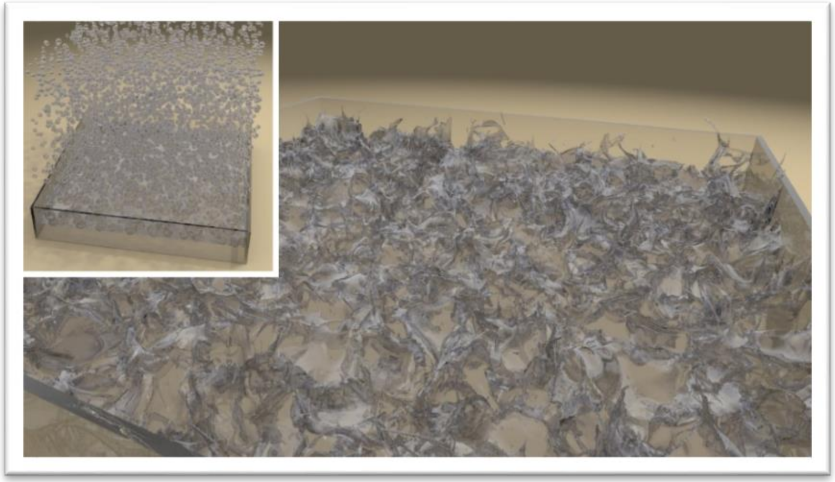




# Outline

- Adaptive Octree Grid
- FEM Discretization & element matrices
- Hanging Vertices
- Second Order Boundary Conditions
- Multigrid Solver
- Results

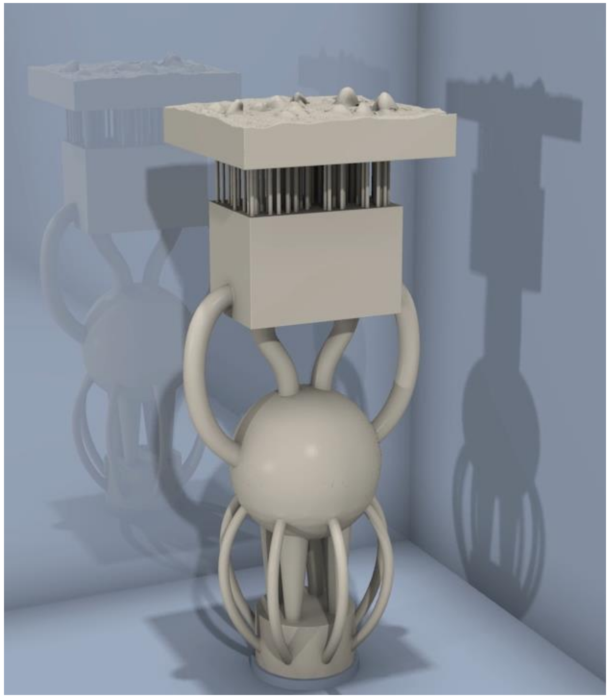


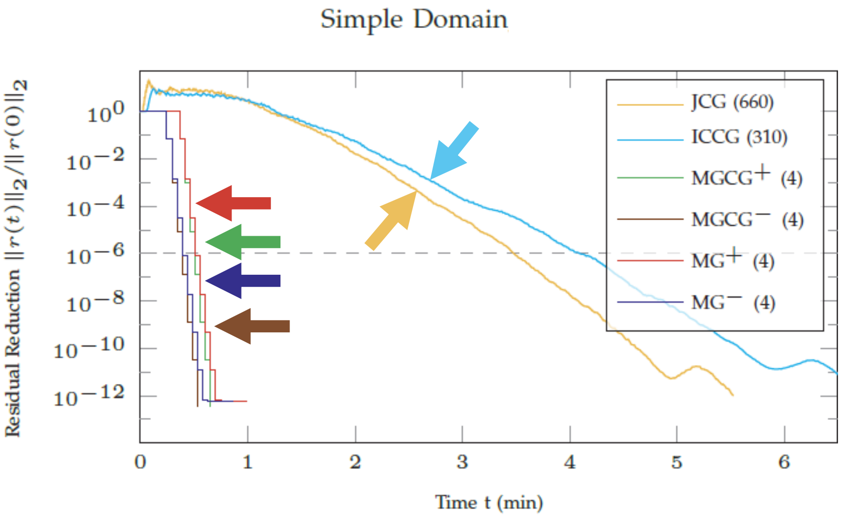


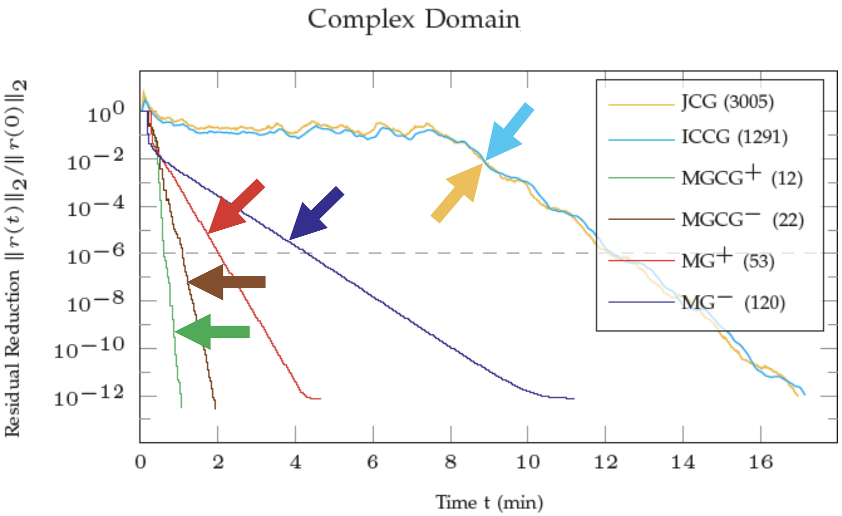
# Solver Convergence

- Simple Domain: Standing water in a box

- Complex Domain:

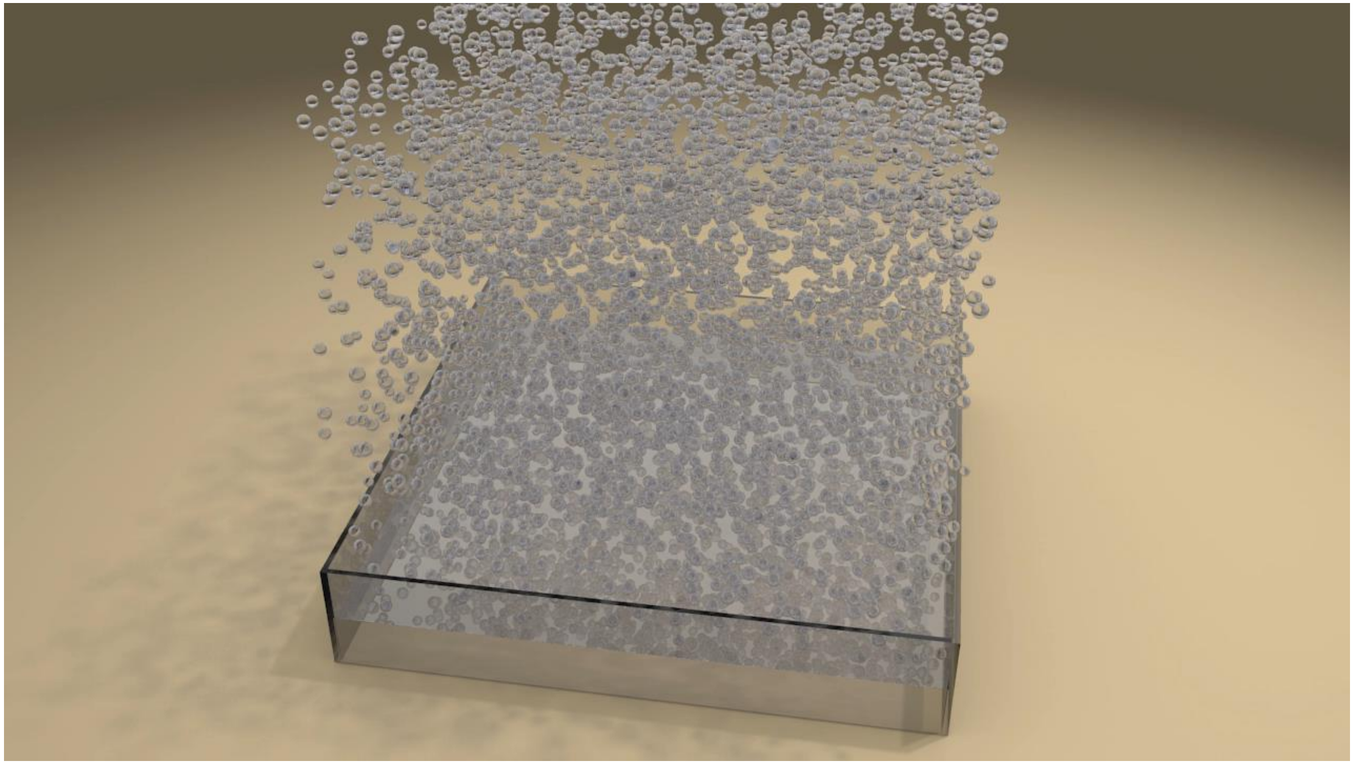




# More Results

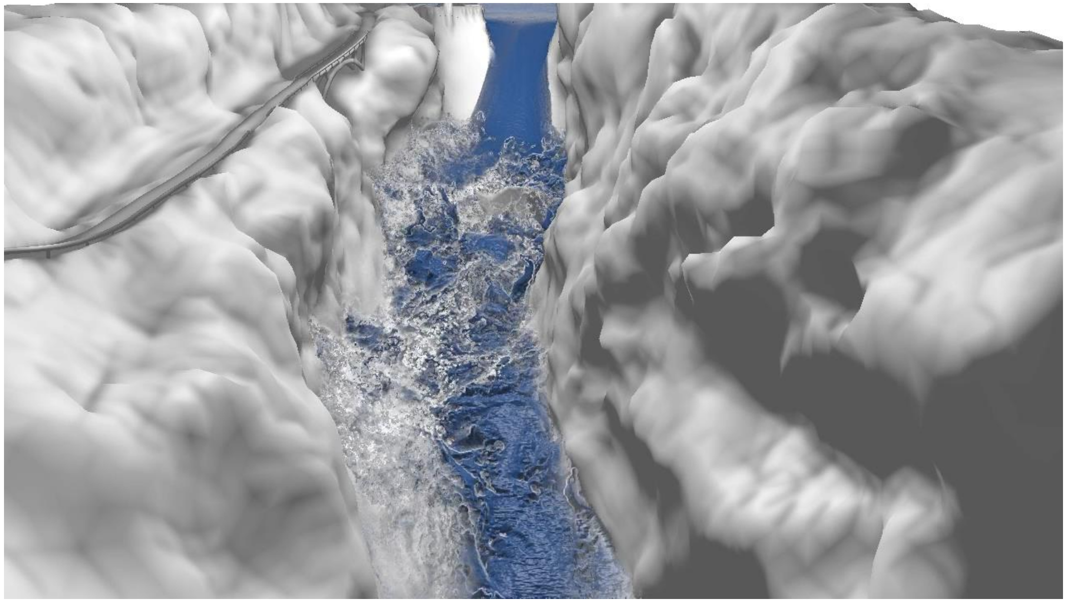
- **Fountain:** up to 34M adaptive cells, 25GB memory, effective resolution $1024^2$ x 3072, < 5 min per time-step
- **4000 Drops:** up to 33M adaptive cells, 22GB memory, effective resolution $1024^3$, < 5 min per time step

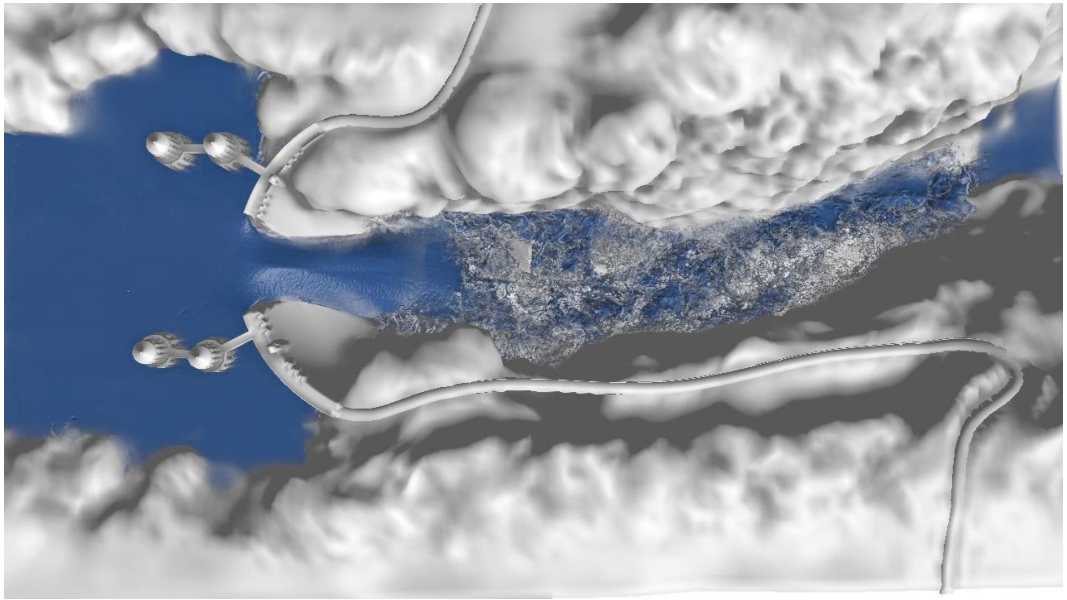


# Conclusion

- Liquid simulation:
  - Adaptive octree grid
  - Multigrid solver
  - FEM discretization

- Limitations & future work
  - Accurate wall boundaries
  - Grid refinement strategy
  - …





Additional example: FLIP simulation (over 100M particles) with octree background grid

# Thats all, thanks!