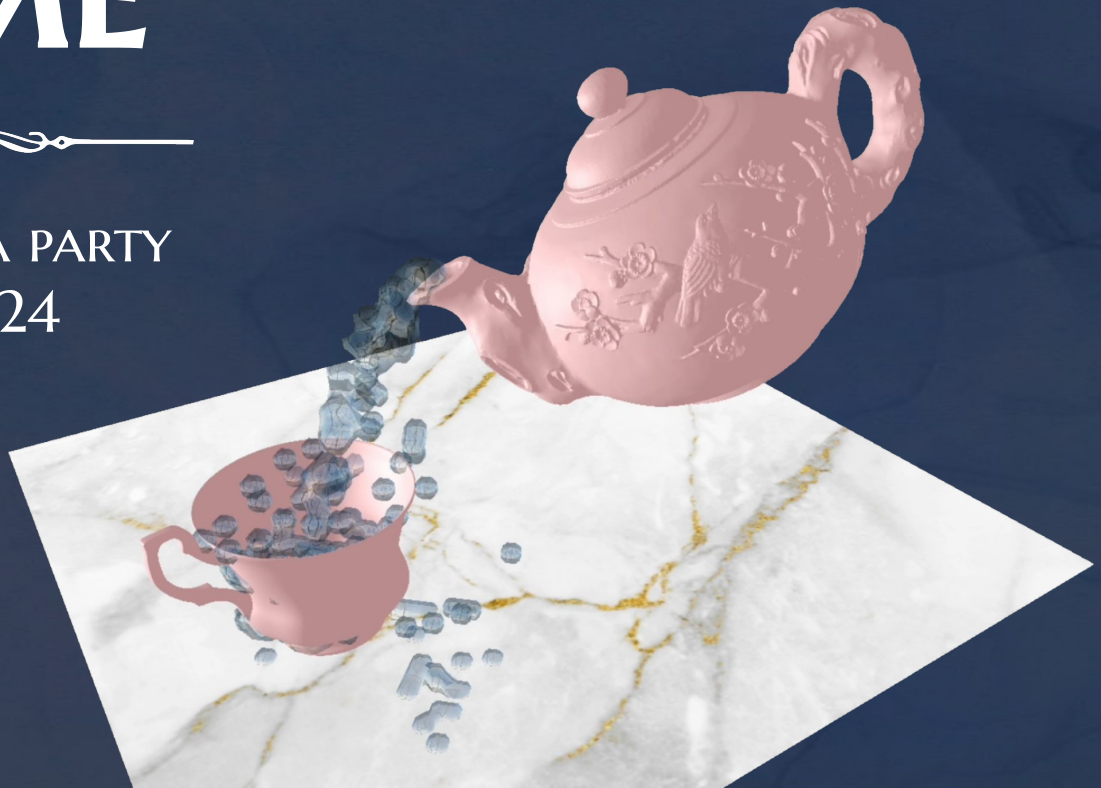
# Tea Time

A 3D polygonized tea party
CS 184 Spring 2024



**Tea Party (tea-m 38):**

Elana Ho, Josiah Adrineda,

N.O., Austin George

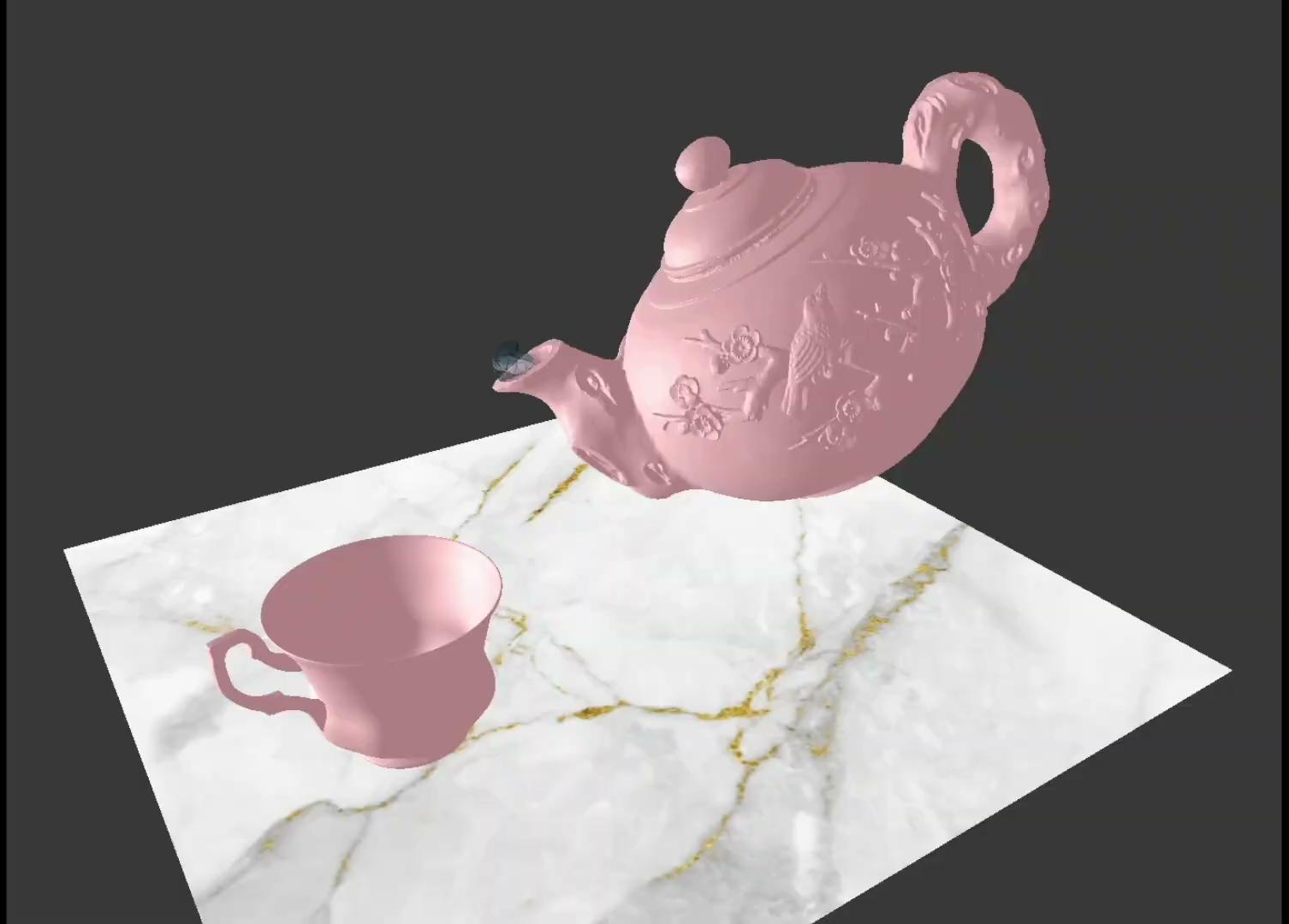
# Background

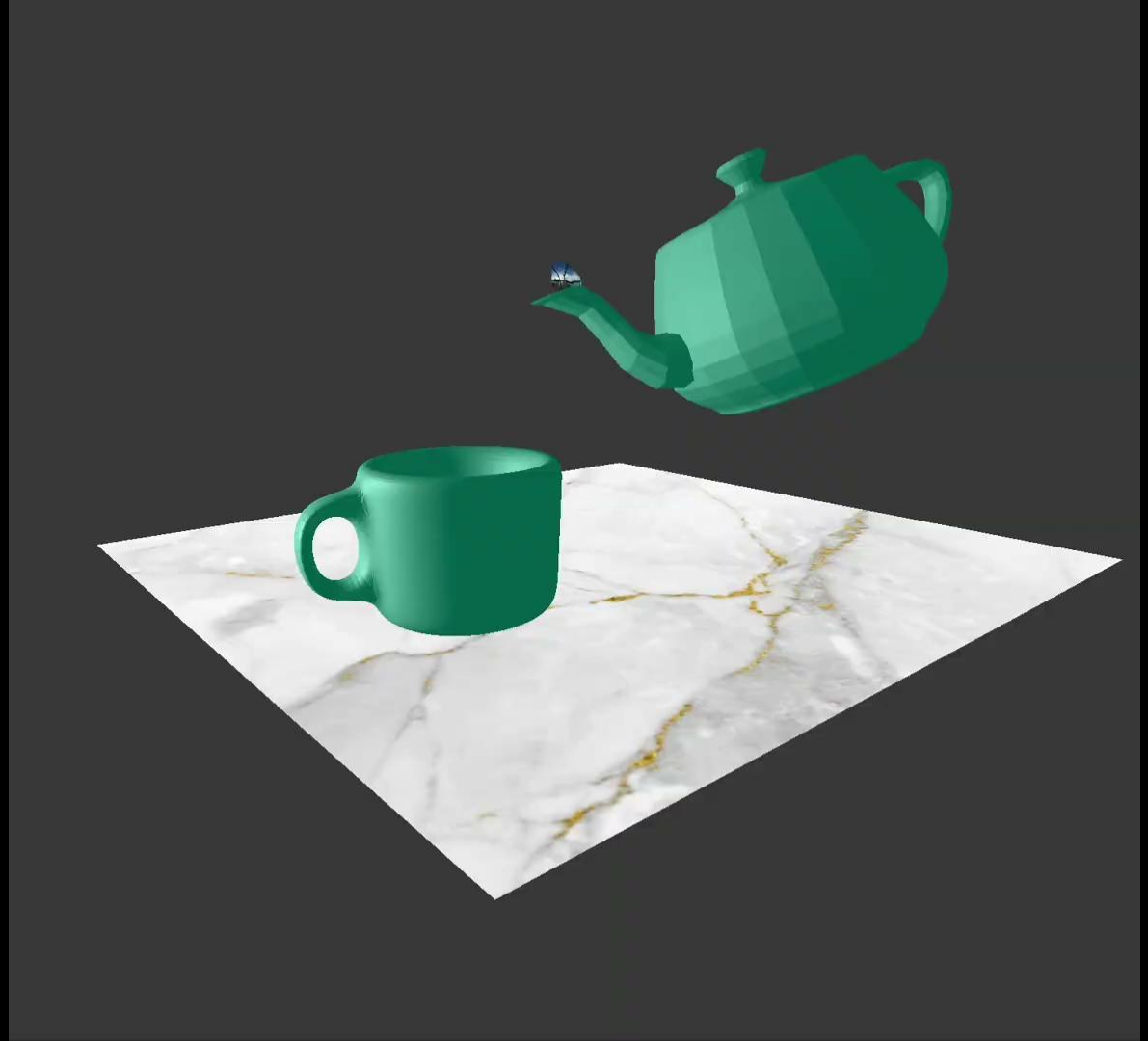
Motivation: fluid simulation with complex geometries

- Shape of the fluid
- Triangle mesh collision objects

Starting point: HW4 cloth simulator

Inspiration





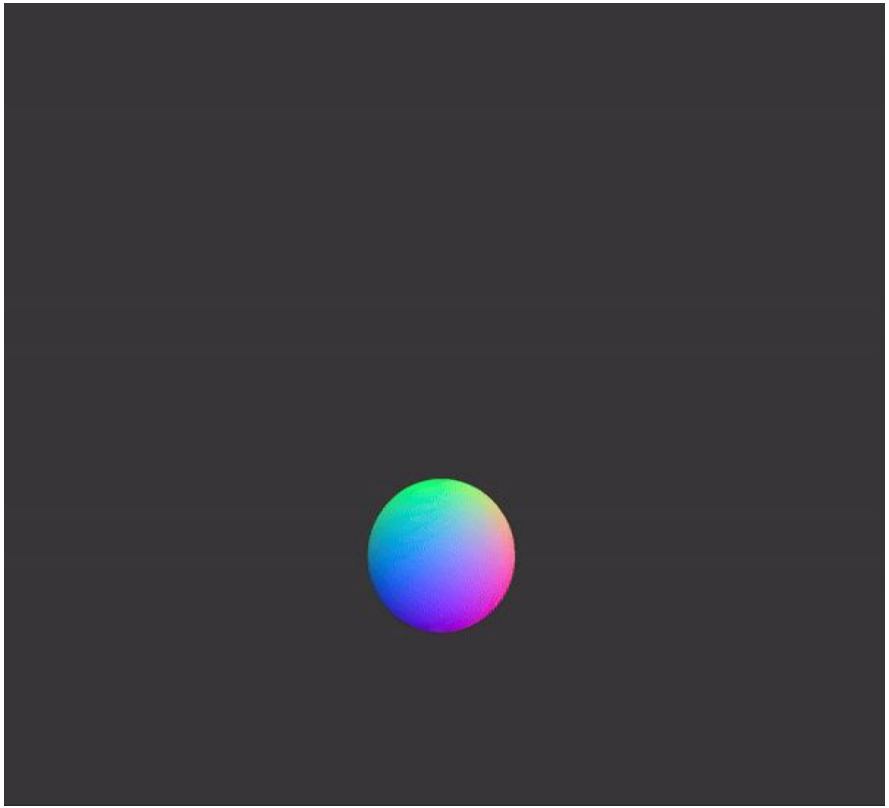
# Technical Approach

# Fluid Simulation

Particle-based fluid simulation

controlled by a 3D smoothed particle

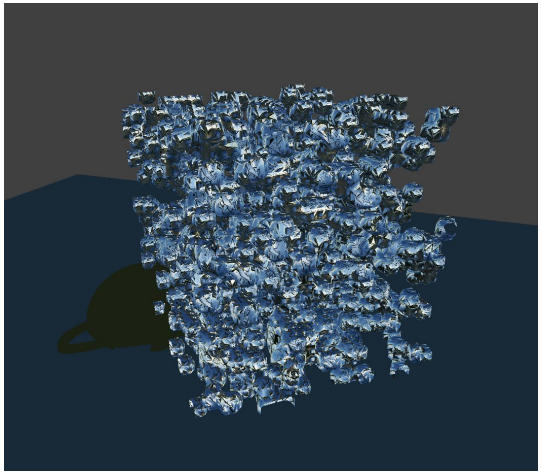
hydrodynamics (SPH) solver

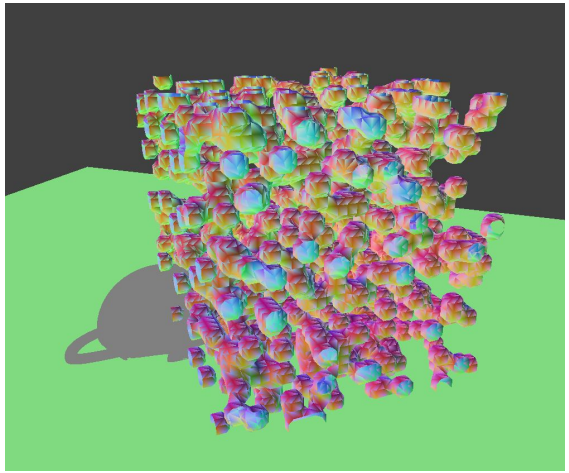


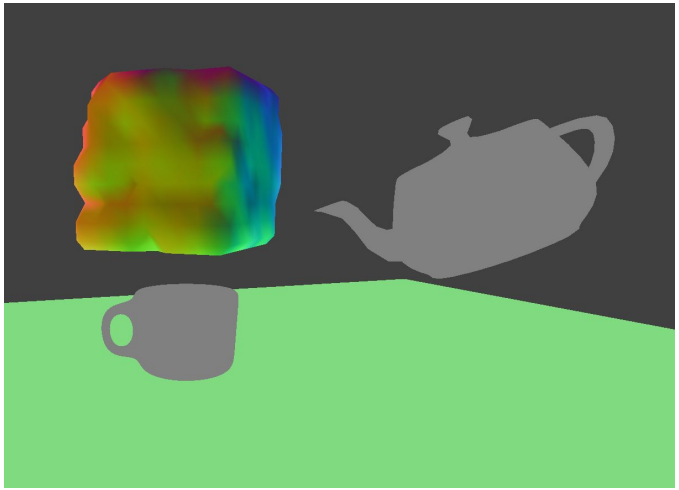
100 particles colliding with a sphere

# Marching Cubes

Particles → Triangle mesh

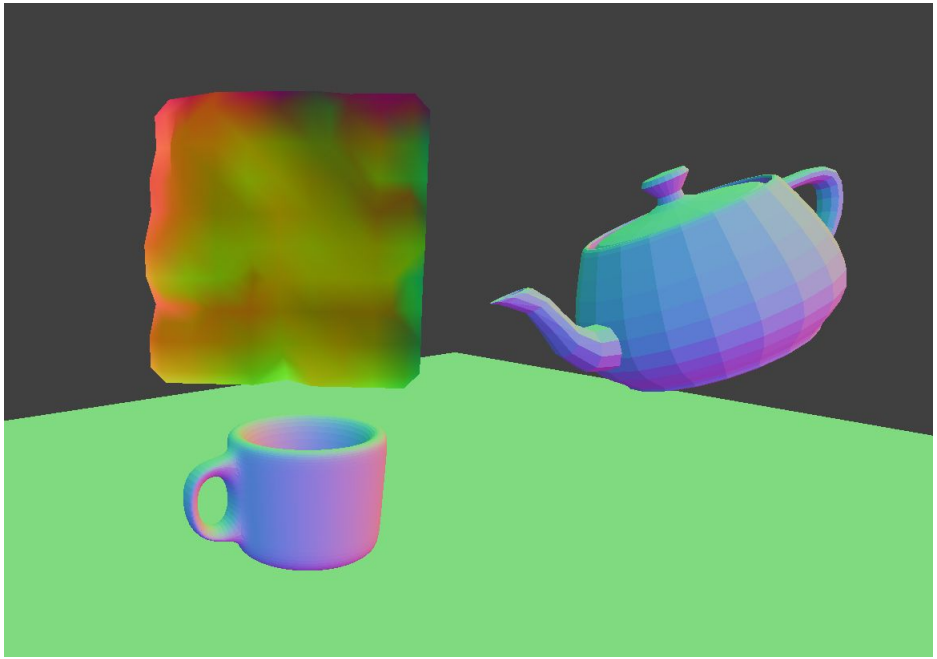






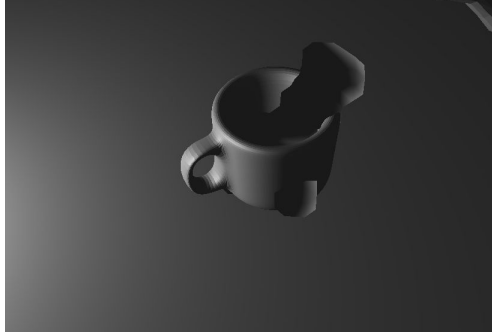
# STL Renderer

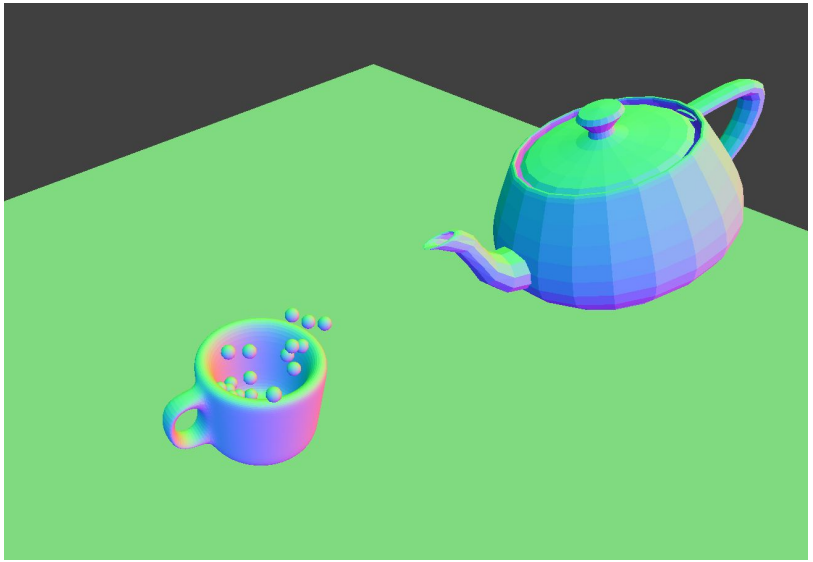
Tea set created in Blender → exported as STL → rendered by STL renderer

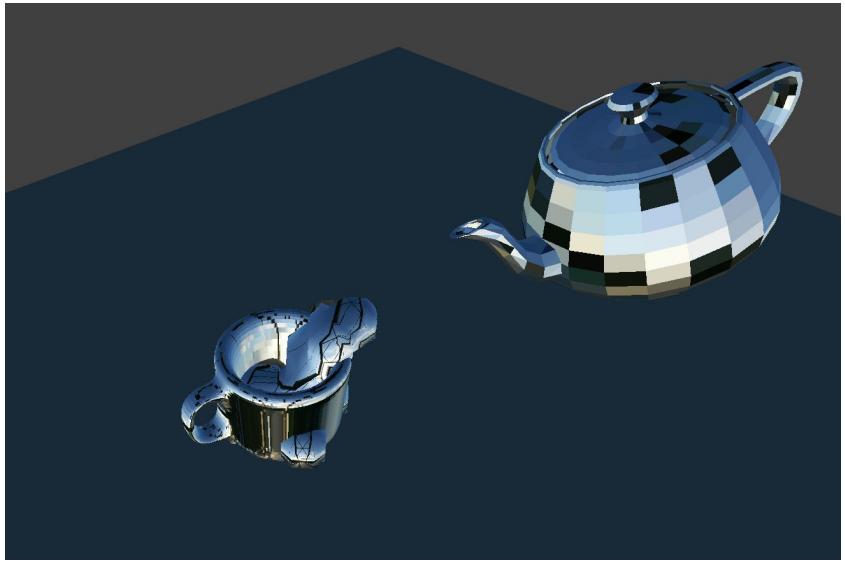



# Triangle Mesh Collisions

Support collisions by handling particle-triangle intersections







# Shaders

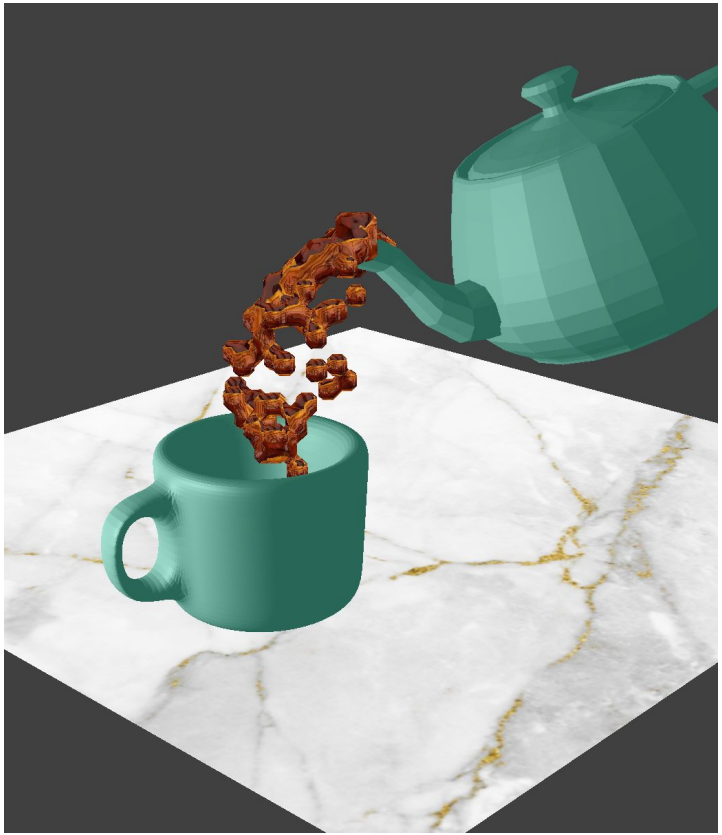
Different shaders

for different

collision objects

and liquid

# Liquid Material Rendering



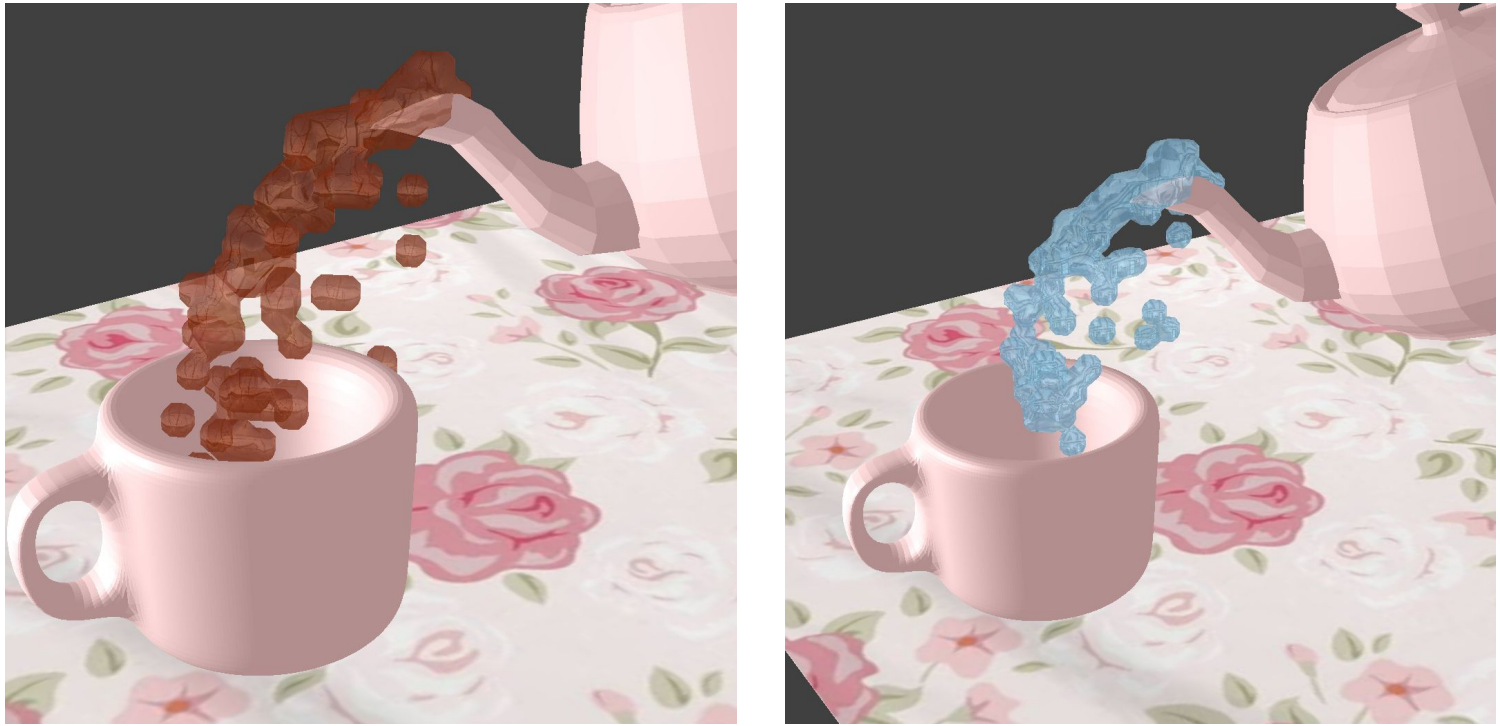
# Liquid Material Rendering

Mirror shading with dining room cube map

# Liquid Material Rendering

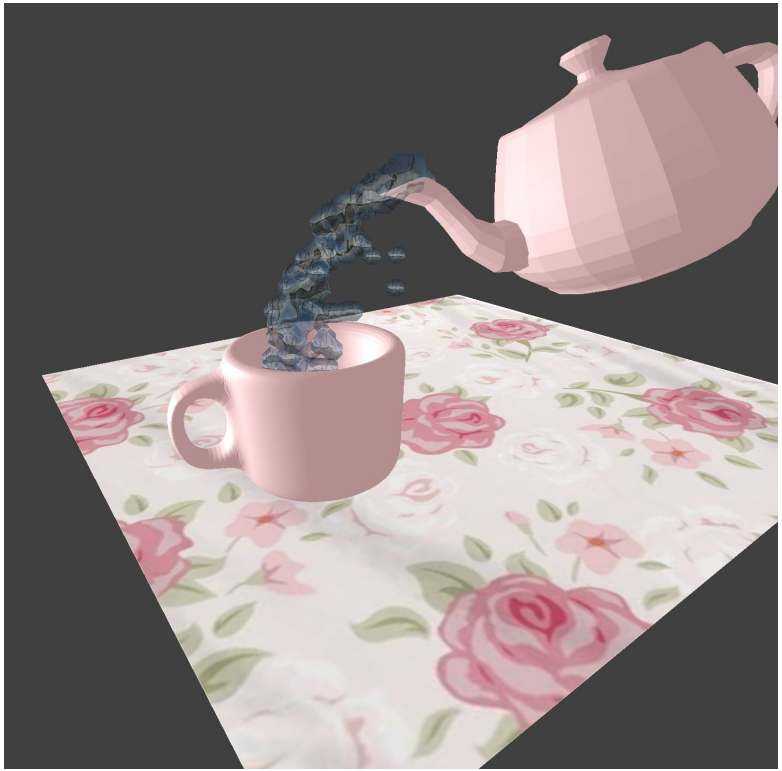
Honey-like tea with sky cube map

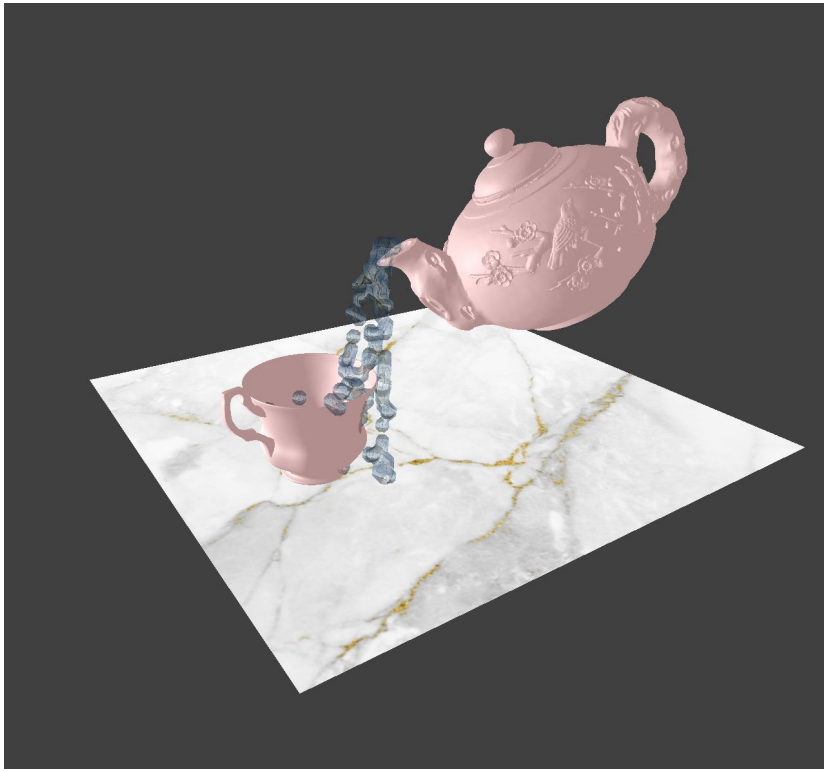
# Liquid Material Rendering



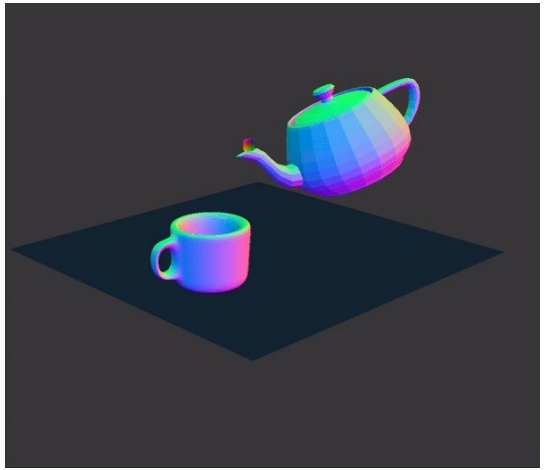
Translucent liquid

# Liquid Material Rendering

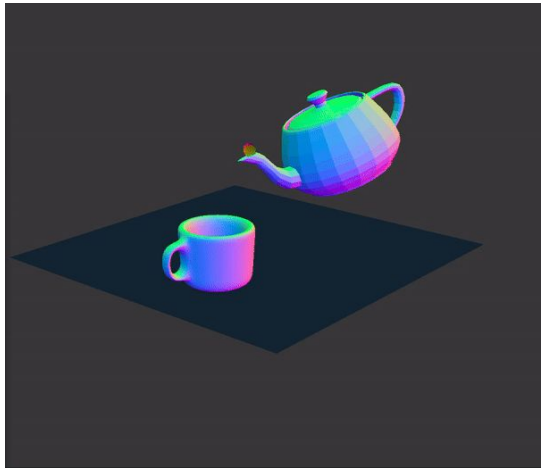




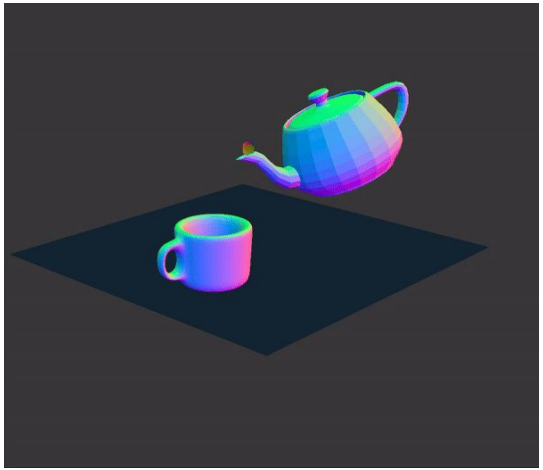
# Particle Count



num_particles = 10



num_particles = 100

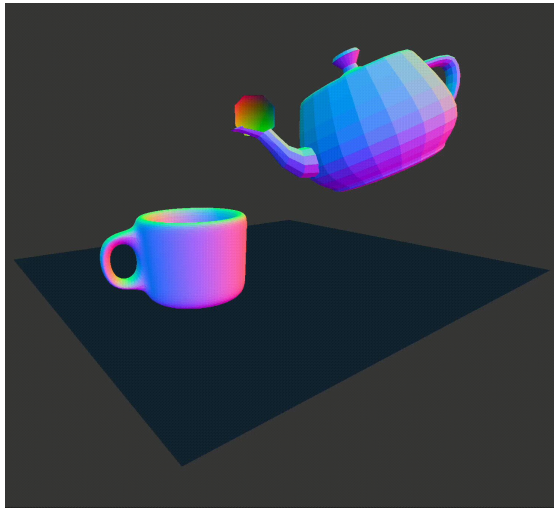


num_particles = 200

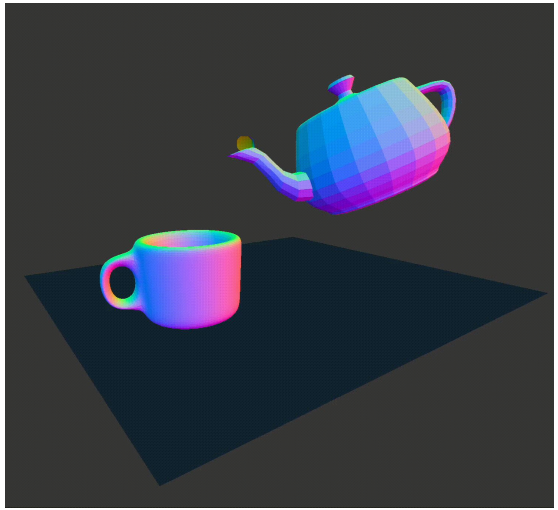
Thinner, faster rendering

Thicker, slower rendering, hits bounding box

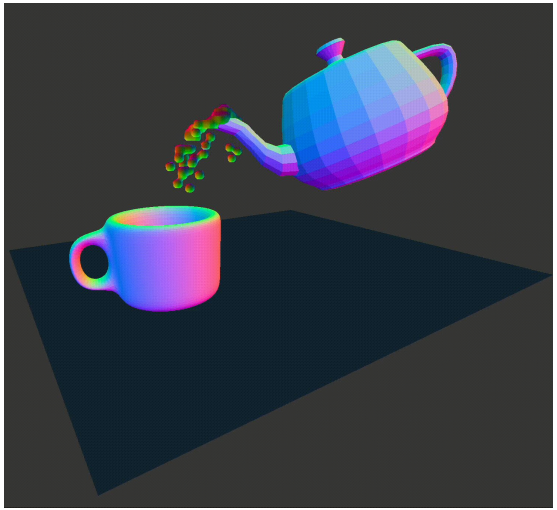
# Marching Cube Count



1,000 Cubes



125,000 Cubes

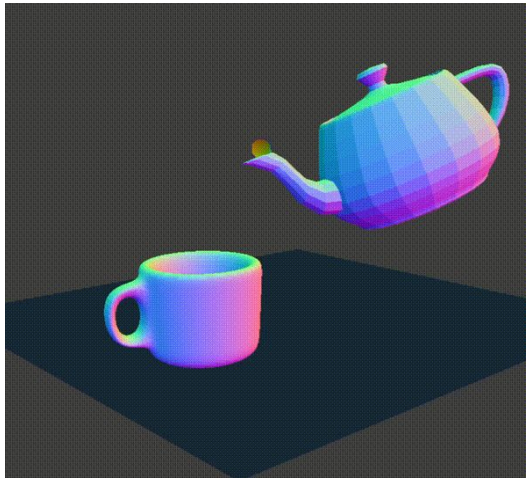


1,000,000 Cubes

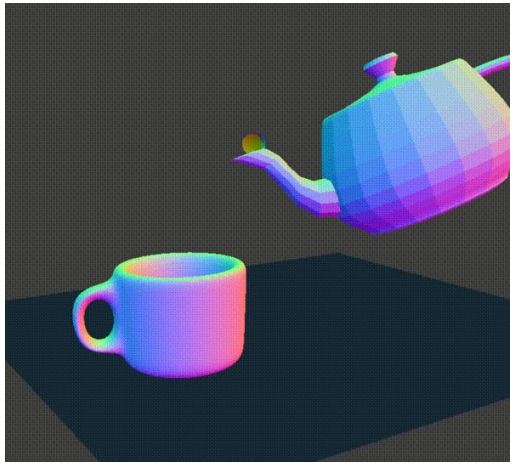
Thicker

More granular

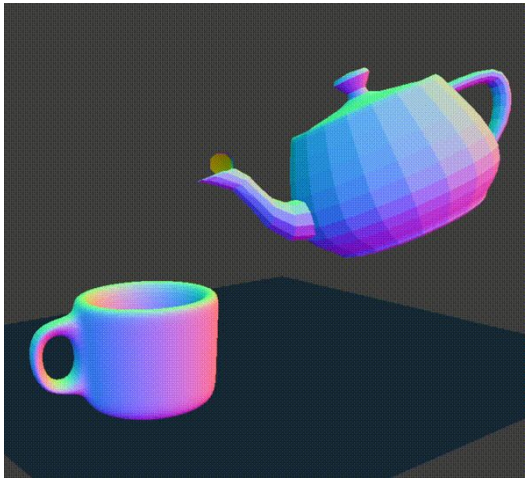
# Spawn Frequency



Every 10 Timesteps



Every 100 Timesteps

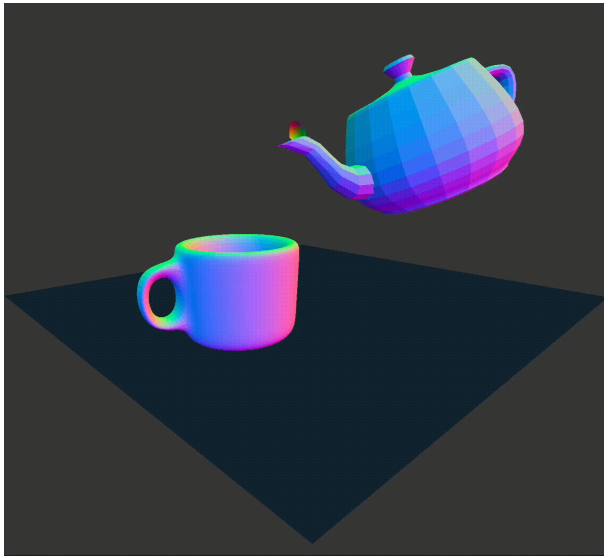


Every 1,000 Timesteps

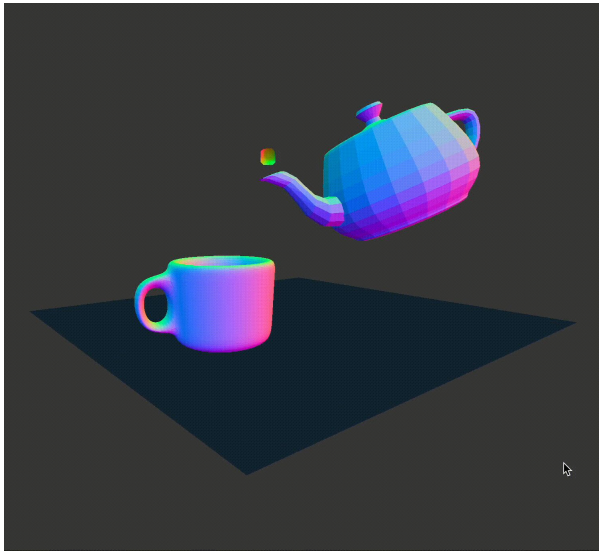
Overshooting

Falls down immediately

# Initial Velocity



Low Velocity{-0.13, 0.5, 0.7}



High Velocity {-2, 3, 0.7}

Pulled down by gravity

Resists gravity

# Conclusion

- Biggest challenge: integrating the fluid simulation and STL rendering
  - Solution: triangle-particle collisions → can generalize to any triangle mesh
- Learned how to leverage acceleration structures (Nearest Neighbors, BVH)
- Potential future enhancements
  - More data-level or thread-level parallelism
  - Refine movement of particles and realism

# Technical Approach

- Particle-based fluid simulation controlled by a 3D smoothed particle hydrodynamics (SPH) solver
- Surface reconstruction using the Marching Cubes algorithm
- Tea Set created in Blender → exported as STL → rendered by STL renderer
- Support collisions with the polygon mesh by handling particle-triangle intersections
- Different shaders for different collision objects and liquid

# Thank You!