#paanipuri

Based on - Miles Macklin, Matthias Müller: Unified Particle Physics for Real-Time Applications - SIGGRAPH 2014

CIS 563 Project -
debanshu singh, sanchit garg & bradley crusco

==========
Work Breakdown
==========

- [x] GLUT (660)-> GLFW OpenGL visualization
- [x] Create constraint classes - density & shape
- [x] Multiple Density (paper 1 - Unified Particle Physics for Real-Time Applications [section 7 & 7.1, before 7.2])
- [x] Density Constraint (paper 2 - Position Based Fluids, [section 3])
- [x] Shape Constraint (paper 1 - Unified Particle Physics for Real-Time Applications [section 5, before 5.1])
- [x] Shape Matching (paper 2 - Meshless deformations based on shape matching [section 3.3])
- [x] TBB Parallelization
- [x] Collisions - Particle Particle (method - spatial hash grid to find nearest neighbours)
- [x] Collisions - Particle Mesh (method - compare signed distance of predicted & current position to determine collision)
- [x] Fluid-Solid Coupling - Buoyancy & Sinking (method - solved both density & shape matching constraints for solids)
- [x] User Input - Fluid & Solids (see Usage for key commands)
- [x] Scene Setup - Fluid-Solid coupling & Cereal shot

More Explanation is at the end of this document

==========
External Code
==========
- [x] paani - 660 project position based fluids (single density) (by sanchit & debanshu)
- [x] learnopengl.com - opengl blinn phong shader & compilation

==========
External Dependencies
==========
- [x] assimp - Open asset import library
- [x] SOIL - simple opengl image library
- [x] TBB
- [x] GLFW

=====
Usage

=====

OSX - paanipuri.xcodeproj
Windows - Paanipuri_Windows.sln

Keyboard Controls -
q - quit
1 - Fluid (high density)
2 - Fluid (low density)
3 - Sphere
4 - Cube
5 - Fluid Pipe (high density)
6 - Fluid Pipe (low density)
WASD - camera movement
mouse movement - yaw & pitch

======
Method
======

- [x] Create constraint classes - density & shape
We changed our pbf code from particle-centric to constraint-centric. Our constraints have a Solve() method to change the deltaPi depending on constraint type (density/shape)

- [x] Multiple Density (paper 1 - Unified Particle Physics for Real-Time Applications [section 7 & 7.1, before 7.2])
We changed the density constraint formulation from our earlier position based fluids work to a mass-weighted version. This enabled simulation of fluids with different densities.

- [x] Shape Matching (paper 2 - Mesh-less deformations based on shape matching [section 3.3])
We implemented rigid shape-matching constraints to maintain rest-pose configuration of rigid bodies. During simulation particles are displaced by all constraints acting on them. The shape-matching constraint allows the rigid-body to then maintain its rest-pose shape. Shape matching acts like a rigid body simulator by respecting the inertial properties of the solid.

- [x] TBB Parallelization
We have parallelized all the steps of the simulation that were independent

- [x] Fluid-Solid Coupling - Buoyancy & Sinking (method - solved both density & shape matching constraints for solids)
We implemented fluid-solid coupling by allowing rigid bodies to solve both density and shape-matching constraints. Solving density constraints forced the particles of the rigid body to behave like fluids of different density allowing them to rise up (as their mass is less), while the shape-matching constraint forced them to maintain the rigid body rest pose.

- [x] Collisions - Particle Particle (method - spatial hash grid to find nearest

  neighbours of different phase)
We extend our spatial hash grid to find neighboring particles of different phase Example - Fluid-Solid will collide. Solid-Solid will collide. Fluid-Fluid will not collide (this is because fluid separation is taken care of automatically by density constraints)

- [x] Collisions - Particle Mesh (method - compare signed distance of predicted & current position to determine collision)
We use Assimp to load an obj mesh (glass bowl). We embed the object into a spatial hash grid to find the cells where mesh boundary is present. If a particle is in that cell, we check for collision with triangles in that cell using signed distance checking.