paani

पानी

A Position Based Fluids Solver for Houdini

Debanshu Singh
Sanchit Garg

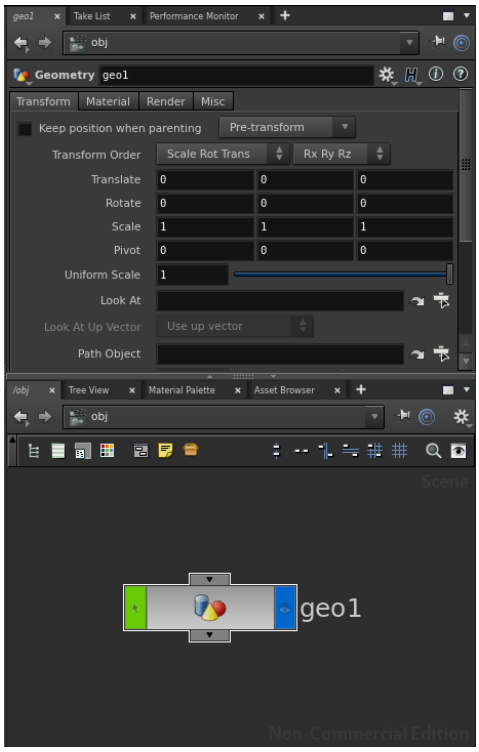
Based on:
Position Based Fluids
Miles Macklin, Matthias Muller (2013)

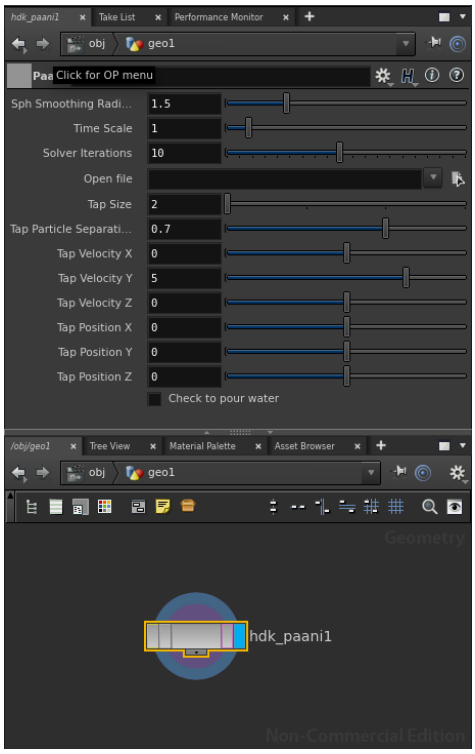
5. RESULTS

5.1 User Guide

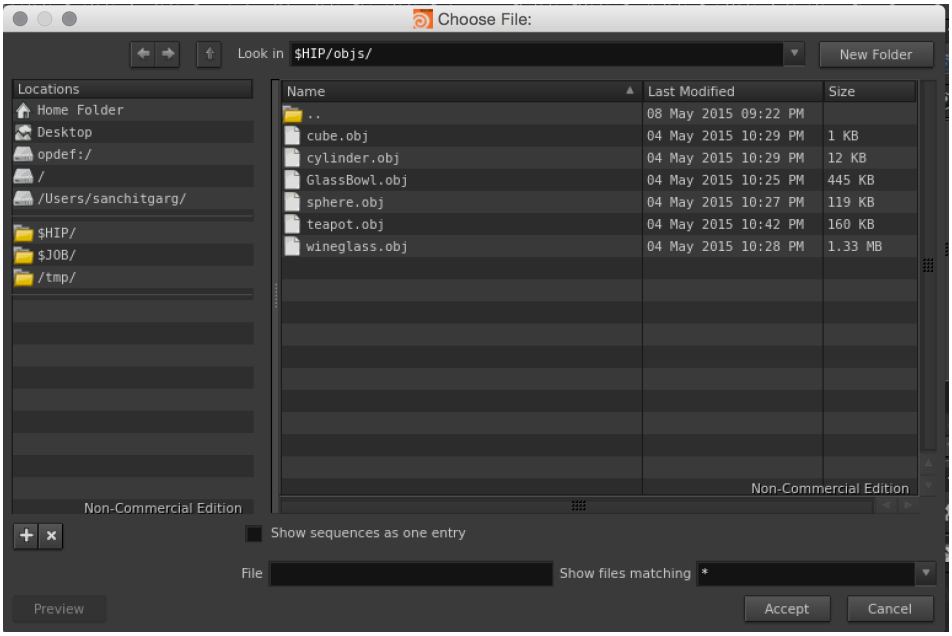
Step 1 : Create geometry object
Press <tab> key and type “geometry”. Press <enter> to create a geometry object

Step 2: Create paani node inside Geometry object.
<tab> Type paani to create a hdk_paani sop node



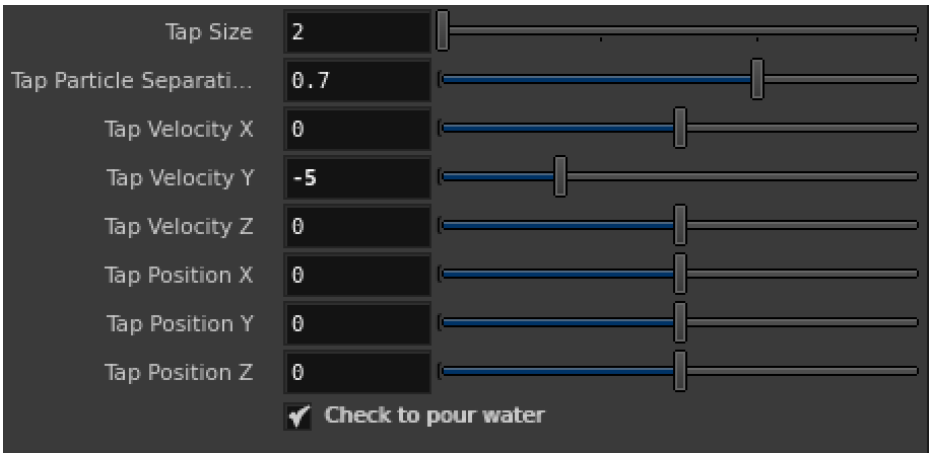


Step 3: <Optional> To load an arbitrary container for the fluid, load an obj file through the Open file button. This allows you to simulate the fluid in a container of your choice.



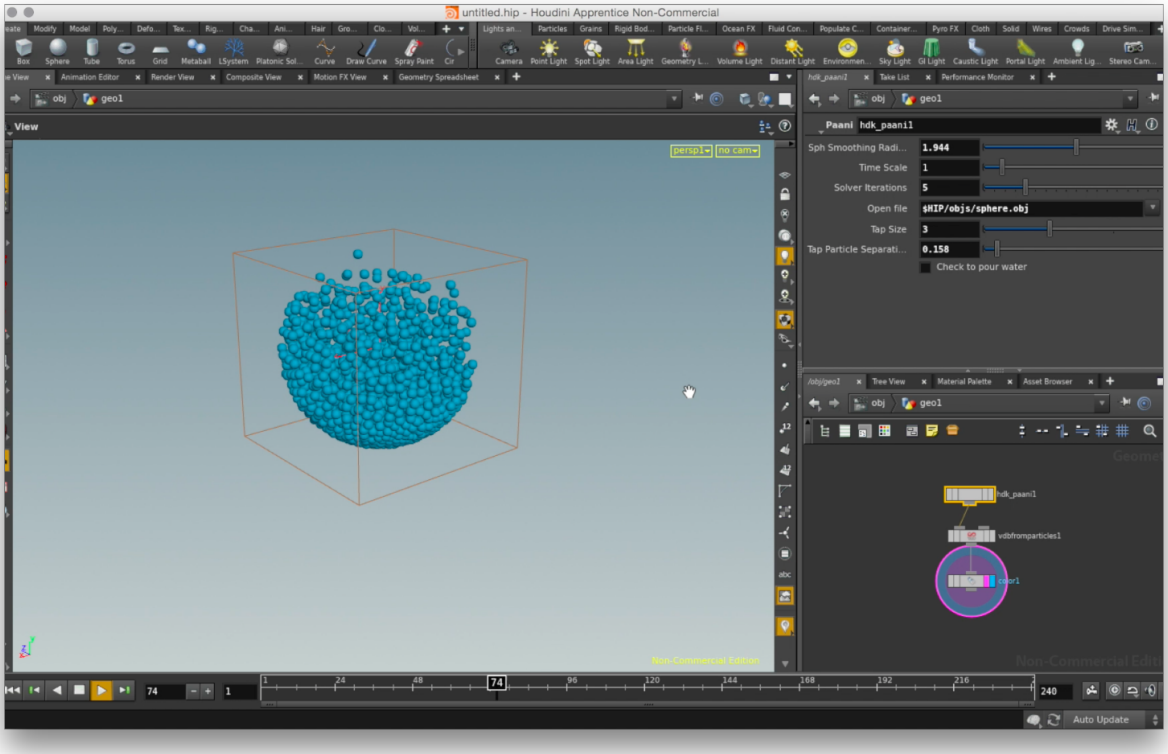
Step 4: <Optional> **Reducing** the Solver Iterations makes the simulation faster but less accurate. A good value is between 3-5. The higher this number the more the simulation converges, however the returns are diminishing after 10 iterations.

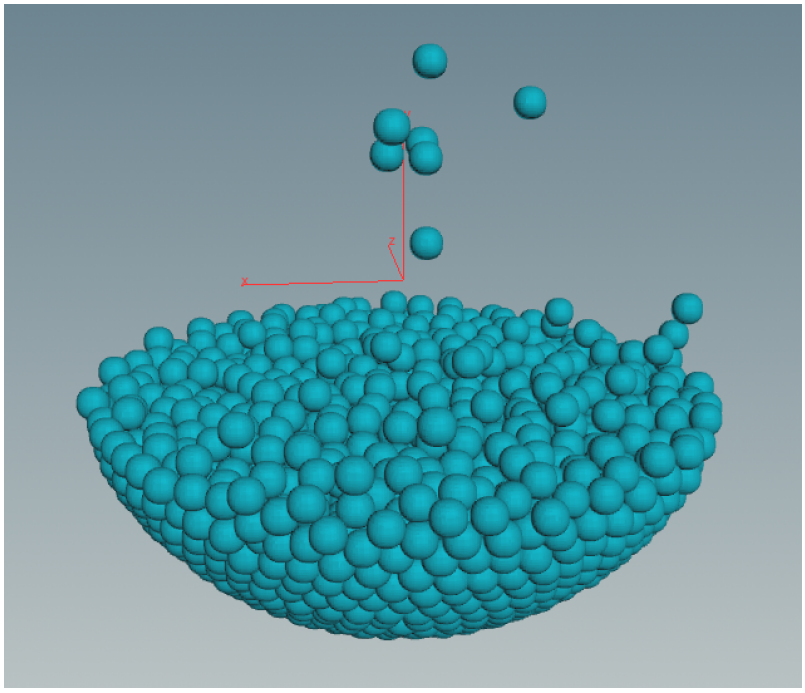
Step 5: Enable the “Check to Pour Water” button and click play to simulate the particles. We also provide the flexibility to change the source position, size and velocity.



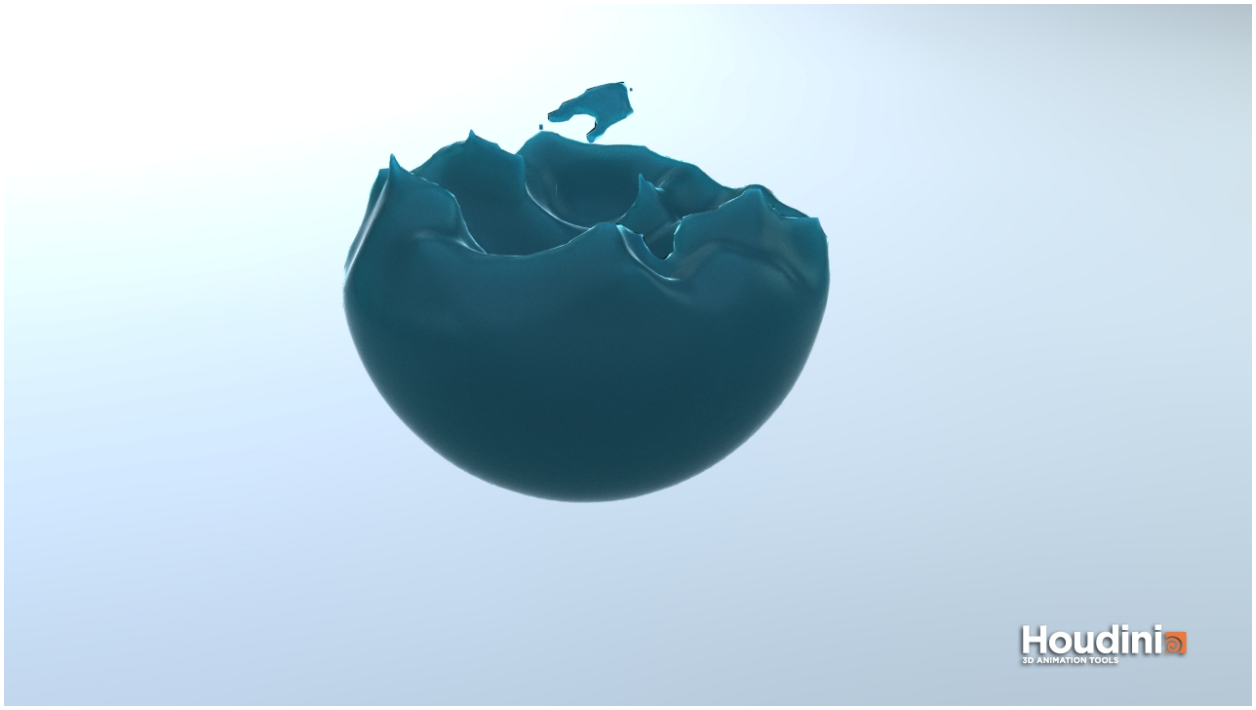
Step 6: Tune the particle separation parameter between the provided range. The initial position of the particles is factored by this value to give a more streamlined (higher value) to a dispersed (lower value) flow of water.

5.2 Content Creation






Houdini
3D ANIMATION TOOLS


Houdini
3D ANIMATION TOOLS

6. DISCUSSION

6.1 Accomplishments

Our implementation of Paani, met all the goals initially set for our authoring tool. Additionally, we added some critical features to Paani that make it more convenient for users. We met three key objectives for Paani - it is fast, stable and user-friendly.

The software design of Paani improves upon the initial spec and is more modular than was initially proposed. Paani implements the 3 main features in the Position Based Fluids paper - Position Based Dynamics solver, SPH density formulation and Artificial pressure. Paani is highly modular to allow for easy future development. For example, this particularly helped us as we switched our neighbour search algorithm from a $O(n^2)$ naive search to a $O(n)$ spatial hash grid.

Since one of the key goals is to be fast, we also realized the need to implement parallelization. While this was not in our initial spec, speed was critical to enabling user interactivity and parameter tuning. Therefore, we decided to refactor our code to enable TBB parallelization.

Paani supports arbitrary containers allowing artists to simulate fluids in a container of their choice. While this was also not in our initial spec, this feature enables our tool to be actually useful in production situations where collisions are with arbitrary containers.

Finally, we present Paani as a Surface Operator node in Houdini 14. Houdini is the leading choice for FX simulation among artists. Houdini's procedural node-based workflow enables artists to easily use the fluid particles output by Paani and combine them with other operators. We provide an example scene file along with Paani that shows how OpenVDB's Level Set operators are used to mesh the fluid particles created by the Paani SOP.

6.2 Design Changes

Our implementation followed the algorithm mentioned in the paper Position Based Fluids. As mentioned in 6.1, we were able to complete all the features that we had planned and added additional features to significantly improve user experience.

We do not implement Vorticity Confinement (section 2.1.3). We focused our efforts on making Paani more artist friendly. Therefore, we pushed vorticity confinement to future development.

6.3 Total Man-Hours worked

Debanshu : 102 hours
Sanchit : 102 hours

6.4 Future Work

We plan to extend Paani's features based on the "Unified Particle Physics for Real-Time Applications" paper presented in SIGGRAPH 2014. This will allow Paani to have fluid interaction of multiple densities, fluid-rigid body coupling, cloth and smoke. By implementing these features, an artist would be able to simulate a wide variety of shots like pouring milk to a cereal bowl, dropping objects in a water pool, etc.

In the future, we plan to refactor Paani to a GPU-based implementation using CUDA. This will help increase the speed of the simulation as well as support more particles. This will be key to enabling real-time feedback for a large number of fluid particles (>100,000) which is crucial for production quality fluid simulation.

6.5 Third Party Software

We use the following third party tools/ libraries for Paani -

01. Houdini 14
02. Open Asset Import Library - for loading arbitrary meshes.
03. Intel TBB - for multithreading
04. GLM, GLFW, GLEW

We used the following external resources during Paani's development -

01. http://ogldev.atspace.co.uk/www/tutorial22/tutorial22.html - obj importer code in Mesh class
02. http://www.tomdalling.com/blog/modern-opengl/01-getting-started-in-xcode-and-visual-cpp/ - GLFW window setup code
03. http://insanecoding.blogspot.com/2011/11/how-to-read-in-file-in-c.html
04. http://lists.apple.com/archives/mac-opengl/2012/Jul/msg00038.html
05. Utility Core - print functions and math constants (Yining Karl Li)
06. www.sidefx.com/docs/hdk14.0/_h_d_k__intro__compiling.html
07. http://www.deborahrfowler.com/C++Resources/HDK/HDK12_Intro.pdf

7. LESSONS LEARNED

We learned three important lessons in creating Paani.

First, parameter tuning. In fluid simulations the most time consuming process is to find the right parameters. Even if the simulation code is correct, it might take a lot of time to get the right set of values for a visually realistic fluid sim. We think a decent amount of time should be kept for tuning parameters.

Second, teamwork. Our team had a good understanding and that enabled us to achieve a lot by coding together. We tried different approaches, and the tactic that worked for us was to first micro-manage our software design steps through Asana. This gave us a roadmap to work on before diving deep into the code. By making our code highly modular, we could work independently and *in parallel.*

Third, the support for Houdini C++ API development is not mature yet. While there are significant pros to creating a HDK plugin, development support was really lacking. We did not find any resources, neither online resources nor people, who could guide us through Houdini plugin development. We could have benefited from more support or guidance.

Brownie lesson, Harmony helped us a lot in understanding fluid sims, critically thinking about our software design and finally understanding that parameter tuning is king.

We dedicate *Paani* to Harmony.