Implementation of Thin Sheet Preservation Techniques in Houdini

Nithish Kumar Etikala (s5431803)

Masters Project

N.C.C.A Bournemouth University

MSc Computer Animation and Visual Effects 2021-22

August 22, 2022

Bournemouth University

Abstract

This paper presents the implementation of a simulation technique using particles to keep fluid sheets preserved based on the papers (Ando, et al., 2012) and (Ando & Tsuruno, 2011). Liquid simulations in computer graphics is an important part of Computational Fluid Dynamics (CFD) to create realistic presentation of different forms of fluids such as water, smoke etc which are mostly used in visual effects for the movie industry. This project is an implementation of thin sheet preservation techniques with a fluid sheets of various animated liquids can be preserved using an adaptable sampled Fluid Implicit Particle (FLIP) model that is particle-based and meshless. maintaining the fluid sheets by collapsing the broken sheets in the deep water and filling the cracking sheets using particle splitting in thin places (Ando, et al., 2012). In comparison to the existing techniques for tracking the surface and the most thin areas, the anisotropy of the neighbor particles are calculated and that data is used as a resampling reference in the process of regeneration of the thin liquid surfaces. The final outputs generated inside houdini demonstrate that this tool can create realistic, aesthetically sophisticated liquid animations with thin sheet preservation.

Acknowledgements

I would like to thank my supervisor, Jon Macey for his support throughout the duration of the program, helpful suggestions and feedback with all my work. I would also like to thank my lecturers Jian Chang, Jian Zhang, Lihua You for providing the theoretical and technical knowledge in evaluating mathematical formulas and implemented them in our work, also would like to thank Phil Spicer for teaching us the technical skills required to work with Houdini and special thanks to Constantinos Glynos for his support and guidance with all the technical issues. Finally, I'm very grateful to all my classmates of MSc Computer Animation and Visual Effects 2021-22, Bournemouth University.

Contents

1. Introduction

Physically based simulation of fluids have accumulated a lot of interest in the visual effects , gaming and advertising industry over the past few decades. Modern liquid animations are being used to generate stunning visuals following various scientific research and creating realistic diverse fluid phenomena, like smoke, fire, and water in the post production of entertainment industry.

Simulating thin liquid surfaces has drawn more attention in recent years as it helps in creating content which are visually pleasing and with real life accuracy. With conventional techniques, it is quite a challenging to be able to follow and generate thin fluid features correctly. The numerical diffusion filters the features in an Eulerian representation, such as the level set approach. The reinitialization process of the particle level set approach separates the thin sheets. Today's researches advocate the front tracking method which is also known as mesh-based surface tracking method, when used with an Eulerian fluid flow. The common way often used to represent surfaces explicitly is by the surface tracking method, with their vertices forming a triangle and being translated by the underlying liquid movement. These methods can capture data from a simulation smaller than the size of a grid cell and do not cause any numerical diffusion, in contrast to implicit methods. Mesh based surface tracking techniques have the disadvantage that it is difficult to handle topology shifting frequently, anytime topology change events are encountered, to prevent tangling, the surface meshes must be thoroughly resampled.. Re-meshing can be accomplished by sewing together nearby vertices in front of a collision or by replacing overlapping surfaces that march in a cube mesh. Because of the complex nature of mesh connectivity, the sewing technique can be challenging to use and result in the loss of surface detail.

By having a thorough understanding of a FLIP solver and various algorithms like Eularian, Lagrangian, SPH theory, and Navier Stokes, this project aims to implement a particle-based framework that preserves thin liquid sheets based on research by Ryoichi Ando, Nils Thurey, and Reiji Tsuruno (Ando, et al., 2012). Houdini is used to accomplish this idea, and an HDA is developed utilising a variety of microsolvers, VEX wrangles, and VOP networks. Additional features, like as viscosity and surface tension, are generated externally with greater control and without using the FLIP solvers default features. Using this tool, users can quickly create a variety of intricate fluid animations with greater control over the details and the ability to add extra features.

2. Previous work

Due to their simplicity of use and applicability for interactive applications, particle-based approaches have gained popularity since (Müller, et al., 2003) introduced the SPH method to the field of computer graphics. Although (Lucy, 1977) (Gingold & Monaghan, 1977) created Smoothed Particle Hydrodynamics (SPH) to simulate astrophysical issues, the technique is sufficiently universal to be used in any fluid simulations.

Recent efforts on explicit surface tracking (Thurey, et al., 2010) and algorithms that leverage them (Yu & Turk, 2013) have made it increasingly evident how important surface-only simulations are. (Wojtan, et al., 2011)provide an overview of these recent changes. We suggest a post-process that can be applied immediately to any coarse particle simulation and is completely divorced from the simulator that produced the data, as opposed to explicitly modelling the fluid surface and carefully including it into the core simulation. The upkeep of a simulation mesh is laborious and frequently necessitates the use of external geometry processing tools. Because our meshless technique is self-contained, implementation is made easier. Mesh artefacts are produced while a mesh is being created for rendering, although this does not affect how stable the simulation is.

Splash animations can be produced using these particles based techniques, though the simulation may experience oscillations due to compressibility. Due to the ease with which such oscillations can easily separate thin structures, these approaches are not ideal for tracing very thin fluid features. Researchers also suggested different Eulerian-Lagrangian hybrid techniques, in addition to SPH. These techniques are made to take full advantage of grids and particles benefits. However, because of the nature of the Lagrangian description these can easily rupture because they still rely on particles to describe thin sheets. This work heavily rely on Yu and Turk's (Yu & Turk, 2013) proposed anisotropic kernels in our study. To determine a particle's stretch and direction, this approach computes a weighted PCA (Principle Component Analysis) for the nearby particles. Reconstructing smooth and detailed surfaces is done using this data. Additionally, To evaluate whether the particles are a thin sheet or part of the bulk volume, we use it as a criterion. We preferred the use of resampling for particles in thin sheets so that, while minimising changes to the simulations dynamics, maintains the thin sheets surface and assures regular particle spacing.

There are two types of surface tracking for liquids: implicit methods and explicit methods. The level set approach is one of the most extensively utilised implicit strategies established by Osher and Sethian (Osher & Sethia, 1988). The nearest surface position is given a distance function from each grid node in the level set procedure, and the surface is implicitly located when the function is equal to zero. The approach has undergone numerous revisions and refinements. Lagrangian particles were positioned by (Enright, et al., 2002) & (Wang, et al., 2009) to lessen numerical dissipation. A high resolution grid can be used to add more detail overall by using these implicit techniques. To the highest of our knowledge, these methods can't completely prevent thin surfaces from rupturing. Additionally, other researchers (Foster & Fedkiw, 2001) & (Kim, et al., 2006) employed seeded particles to make fluid sheets or splashes.

Volume of fluid vof technique (Hirt & Nichols, 1981), which utilises a portion of the interface for the entire cell, is one of the explicit approaches for the dynamics of free borders. Due to the difficulty of managing a discontinuous interface, this strategy is rarely used. The Lagrangian approaches are frequently preferred in explicit procedures. Several mesh-based surface tracking techniques have been put out during the last two decades in a variety of research areas, including fluid dynamics and the processing of medical images (Muller, 2009) (Wojtan, et al., 2010). Typically, a mesh-based surface tracker advects explicit surface components from the underlying motion; however, this method may be hindered by complex topology changes or self-intersection. Resampling meshes only where collisions occur may help to fix these problems. The algorithm tends to become more sophisticated as a result of this tactic because it must be able to handle many complex circumstances. We want to be clear that our approach has nothing to do with this group of surface tracking techniques. On the other hand, there is no connectivity information in our particle-based method. As a result, our approach avoids such complications.

SPH, Moving Particle Semi-implicit, and Meshless Physical Simulations are three examples of particle methods that are frequently used to reconstruct surfaces in addition to simulating complete physical motion. An original surface reconstruction algorithm from a point cloud was proposed by Zhu and Bridson (Zhu & Bridson, 2005), Adams (Adams, et al., 2007), Yu and Turk (Yu & Turk, 2013). Surfaces are implicitly specified in their approach in relation to particle proximity. The ruptures produced by these techniques, however, are blobby and unsatisfactory in areas with sparse particles. We use the Yu and Turk (Yu & Turk, 2013) method in our particle-based approach to recreate surfaces after securing ruptures with additional particles to protect the sheets. We employ a modified version of the hybrid grid and particle based method suggested in as the underpinning fluid solver.

According to academic papers by (Adams, et al., 2007) & (Hong, et al., 2008) , the adaptive resampling aspect of our method for ocean depths volumes is viewed as a version of adapting particle-based approaches, in which the particles typically resampled utilising a range of sizes to reduce the processing cost while keeping visual detail. On the other hand, in our concept for thin sheet preservation, we mainly concentrate on sustaining the flowing fluid sheets rather than increasing efficiency.

3. Technical Background

3.1 Smoothed Particle Hydrodynamics

SPH is a technique for particle system interpolation. SPH allows for the evaluation of field quantities that are solely defined at discrete particle positions. The momentum and energy equations become ordinary differential equations using this approach. As an example, the force between particles will represent the pressure gradient. Also employed and referred to as "the heart of SPH" is an interpolation technique that enables any function to be stated in terms of values at a collection of disordered locations. The following formulas define the integral interpolant of any function:

$$A_I(\mathrm{r}) = \int A(r')W(r-r',h)\,dr' \qquad (1)$$

W is an interpolant kernel that is provided with two specific properties: the first and second conditions of existence of the kernel. The interpolation is over the entire space.

$$\int W(r-r',h)\,dr' = 1 \qquad (2)$$

$$\lim_{h \to 0} W\,(r - r', h) = \delta(r - r') \qquad (3)$$

SPH uses radial symmetrical smoothing kernels to distribute values in the immediate vicinity of each particle for this reason. SPH states that a scalar quantity **A** is interpolated at location **r** by the weighted total of all particle contributions.

$$A_s(\text{r}) = \sum_j m_j \frac{A_j}{\rho_j} w\left(r - r_j, h\right) \qquad (4)$$

A set of particles with certain masses is used in SPH to represent the fluid volume. Physical characteristics, such as pressure and density, are represented by values that are linked to each particle in a specific simulation phase. A weighted contribution of the adjacent particle masses m_j is added to the density ρ_i for particle *i* at location x_i to interpolate the density.

$$\rho_i = \sum_j m_j W\left(x_j - x_i, h_j\right) \qquad (5)$$

The smoothing radius connected to particle *j* is known as *hj*, and *W* is the smoothing kernel. The Tait equation (Monaghan, 1994), which is a common formula for describing the pressure *pi* of a particle *i* as a function of fluid density, is

$$p_i = k\rho_0 \left(\left(\frac{\rho_i}{\rho_0}\right)^{\gamma} - 1\right) \qquad (6)$$

where the stiffness parameters k and $\gamma$are present, and ρ_0 represents the fluid's rest density. The Navier Stokes equation becomes an ordinary differential equation (ODE) of the form when it is discretized on particle locations in the SPH framework.

$$\rho \frac{\partial v_i}{i\partial t} = -(\nabla p)(x)_i + \ \mu(\Delta v)(x_i) + f_i^{ext} \qquad (7)$$

3.2 Navier-Stokes equations

The Navier-Stokes equations serve as the foundation for the basic theory of fluid motion. The physical equations that explain the motion of fluids are known as Navier-Stokes equations. Because of this, they frequently serve as the basis for many solutions in current software programmes like Houdini. The Impermeable As the foundation for many of the existing solutions, a group of partially differential equations are known as the Navier-Stokes equations. that are used as the main model to simulate fluids (liquids and gases). These equations can be solved using a variety of techniques to produce fluid simulations. Some of them are the Smooth Particle Hydrodynamics (SPH), Lattice Boltzmann methods, Eulerian or Lagrangian approaches, and others. Depending on the particular requirements, several methods are chosen.

A pressure field and a velocity vector field are used to model the condition of a liquid at a specific instant in time. These two fields are connected, and the Navier-Stokes equations explain how they evolve over time. The equations offer a mathematical representation of the intricate mathematics of fluid motion in accordance with the rules of physics. By resolving them, we obtain values for the fluid's unknown, specifically the new acceleration that enables the determination of its new position. Although the equations can be stated in a variety of ways, we'll write them as

$$\frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot \nabla \vec{u} + \frac{1}{\rho} \nabla p = g + \nu \nabla \cdot \nabla \vec{u} \tag{8}$$

$$\nabla \vec{u} = 0 \tag{9}$$

Water naturally has limited viability. As a result, the viscosity-related element in the Navier-Stokes equations can be disregarded because it is not as significant as the other terms. With Equation 8's viscosity term removed, we obtain

$$\frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot \nabla \vec{u} + \frac{1}{\rho} \nabla p = g \tag{10}$$

3.2.1 Eulerian fluid simulation

The Navier-Stokes equation is typically resolved over a grid in eulerian fluid simulations. The Navier-Stokes equations for incompressible flow were used by Foster and Metaxas to create the first fully 3D liquid simulation in the computer graphics community. Based on the pioneering work by Harlow and Welch [11] that created the Marker and Cell method, their simulation approach was Eulerian. They computed partial spatial derivatives on a fixed finite-difference grid, explicitly integrated advection and diffusion, relaxed incompressibility, and represented surfaces with massless marker particles. Using a semi-Lagrangian methodology to manage the convection component and implicit integration to handle the diffusion term, Stam significantly improved the stability of this method of fluid modelling, enabling the use of considerably bigger time steps. The fluid moves through the rectangular cells that make up the grid space. At each grid point, the fluid quantities are specified. We record the cells of a grid instead of each individual particle and its properties because this allows us to track how the values of the cells change over time (Bridson, 2008).

3.2.2 Lagrangian fluid simulation

In Lagrangian simulations like SPH (Müller, et al., 2003) and Moving Particle Semi-Implicit (MPS) (Premože, et al., 2003), the fundamental momentum-carrying fluid mass component is the particle. The smoothed particle hydrodynamics (SPH) model that Lucy (Lucy, 1977) created was intended to mimic astronomical occurrences. A smoothing kernel can compute the inter-particle forces between the fluid's particles, including pressure and viscosity, at each particle's position. SPH was first applied to computer graphics by Desbrun and Cani-Gasculel to represent highly deformable materials like lava flow. The field quantity could be sampled adaptively during particle refining. In less deformed locations, particles were combined into a single giant particle, whereas in more damaged parts, they were divided into multiple smaller ones. The fundamental principle of the adaptive sampling technique for Lagrangian fluid is similar to Geoffrey's theory (Irving, et al., 2006), which used fluid cell coarsening away from the interface for efficiency in Eulerian fluid and refinement near the interface for detailed representation. Muller proposed a new method for fluid behaviour that takes into account complex interactions between fluids, including those involving boiling water, trapped air, and highly deformable objects, by extending Desbrun's SPH based technique to simulate highly deformable objects for interactive fluid simulation. The Lagrangian perspective defines fluid flow as a flow of particles, each of which has its own characteristics, such as mass, velocity, density, etc. Each fluid particle is directly subject to the principles of conservation of mass and Newton. The Lagrangian approach is simple to use because all calculations are done on the particles, but because they rely so largely on density, they may be highly inaccurate in places with low density (Strantzi, 2016).

3.3 Hybrid Particle Methods

Lagrangian particles or Eulerian grids are the two basic methods for simulating fluid effects in computer graphics. The two methods each have advantages and disadvantages. Lagrangian particles are excellent at advection but struggle with the pressure and incompressibility limitation. Fortunately, the Eulerian grid approach is excellent at handling the pressure and incompressibility constraint, but the method has issues with the advection component because of interpolation errors from the semi-Lagrangian advection. It is clear that where the Lagrangian approach has problems, the Eulerian approach performs very well. By combining these two approaches, Lagrangian particles can handle the advection portion while the Eulerian grid can handle the pressure and incompressibility constraint, resulting in a more accurate simulation of water effects (Englesson, et al., 2011).

Particle level sets, The Particle in Cell (PIC), and the Fluid Implicit Particle (FLIP) method are just a few examples of the many various hybrid strategies that exist. Foster and Fedkiw (Foster & Fedkiw, 2001) established the particle level set approach in while Harlow (Evans & Harlow, 1957) launched the PIC method in 1963. Brackbill and Ruppel (Brackbill & Ruppel, 1986) later refined it with the FLIP method in 1986. In 2005, Zhu and Bridson (Zhu & Bridson, 2005) introduced the FLIP method to the field of incompressible flow, and it is now the most advanced approach for simulating fluids.

3.3.1 PIC / FLIP method

Despite the fact that the PIC and FLIP procedures are nearly identical save for a small variation in the velocity update, the fluid properties vary considerably. Due to numerical dissipation caused by the double interpolation of the velocity, a fluid that simply utilises PIC is more viscous than a FLIP fluid; however, more on that in a moment. However, the FLIP approach has little viscosity and is therefore excellent for water effects, although there is undesirable surface noise present. One can create a fluid with low viscosity and no surface noise by linearly combining the PIC and FLIP approaches.

Interpolation is the root cause of the PIC method's significant numerical dissipation. The Navier-Stokes equations are used to calculate the new particle velocities, and the smoothed velocity values are then interpolated back to the particles to replace the old velocities. The particle velocities are transferred to the grid by interpolation, which introduces some smoothing. As previously stated, a PIC fluid will seem viscouster than a FLIP fluid because of this excessive double interpolation. Brackbill and Ruppel found a solution by interpolating the change in velocity and adding it to the already calculated particle velocities rather than interpolating the newly calculated velocities to the particles from the grid and replacing the velocities. By doing this, only one interpolation's worth of smoothing is carried out, as opposed to the PIC technique's accumulative smoothing, which virtually eliminates numerical dissipation in the FLIP approach.

4. Implementation

4.1 Interpolation

The Eulerian-Lagrangian hybrid Particle in Cell or the Fluid Implicit Particle (PIC or FLIP) technique, which performing similar activities in different a liquid domain with a collection of particles, is the basis of our solution. We extrapolate qualities that are based on particles, like velocities, using SPH-like interpolation kernels as opposed to the conventional FLIP technique. This interpolation is utilised later on in our thin sheet preservation approach as well as for the underlying simulation. The fluid's velocity (u) and the position's particle density (x) are determined as follows given a particle distribution:

$$u(x) = \frac{\Sigma_i m_i u_i W_{Sha} \quad (P_i - x, \alpha_u d_i)}{\Sigma_i m_i W_{sharp}(P_i - x, \alpha_u d_i)} \tag{11}$$

$$\rho(x) = \sum_i m_i W_{smooth}\left(P_i - x, \alpha_\rho d_i\right) \tag{12}$$

where, in order, d_i, P_i, u_i, and m_i stand for the particle's radius, location, velocity, and mass i, respectively. The scaling factors for radius d_i are u and. Both α_u = 1.0 and α_ρ = 4.0 were applied. We utilise two basic kernels for weighting:

$$W_{smooth}(r, h) = \begin{cases} \frac{h^2}{||r||^2} - 1, & 0 \leq ||r|| \leq h \\ 0 & otherwise \end{cases} \tag{13}$$

$$W_{smooth}(r, h) = \begin{cases} 1 - \frac{||r||^2}{h^2}, & 0 \leq ||r|| \leq h \\ 0 & otherwise \end{cases} \tag{14}$$

While $W_{smooth}(r, h)$ is used to calculate the average of a quantity from neighbouring particles, our sharp kernel $W_{smo} \quad (r, h)$ is designed to resample a quantity of the particle such that $u\ (P_i)$ = u_i. Any kernels with comparable qualities, such as those frequently used for SPH simulations, would be relevant in this situation. In order to expedite the neighbourhood lookup, we group the particles into the grid prior to processing each step of the simulation. If Equation 11's denominator is close to zero, we use the velocity of something like the closest particle to prevent roundoff errors. In order to determine the location of the free surface, we also compute a level set function for each grid cell:

$$\emptyset_L(x) = \alpha_L N_0 \rho_0 - \sum_i \rho(P_i) \tag{15}$$

where i and ρ_0 stand for the initial maximum density at the start of the simulation and the particle index within a cell, respectively. N_0 is the number of particles that are initially supplied to a cell, whereas α_L is a quantity that is utilised to discriminate between splashing particles and incompressible liquid volumes. All cells with $\emptyset_L(x) \leq 0$ belong to the liquid domain. We continuously run our three-dimensional simulations with α_L = 0.2 and N_0 = 8. In locations where particles are in low quantity, such as liquid sheets or holes in the liquid domain, cells should not be mistakenly classified as liquid. This is why Equation 15's implicit functions are used. As a result, undesirable volume growth is avoided and liquid particles are kept from drifting on the free surfaces.

4.2 Grid-based Solving

The particles speed as they are being advected is first mapped onto a Marker and Cell (MAC) grid in our simulation. The primary Poisson equation is solved on the grid, making the velocity divergence-free. An incompressible particle flow is produced in PIC (Harlow, 1962) by mapping the anticipated grid velocity to the individual particles. In line with (Hong, et al., 2008), and (Hong, et al., 2009), we employ Equation 11 in our implementation to map the particle-to-grid velocity. Although grid based interpolation, for example, as in (Zhu & Bridson, 2005), might be utilised in this situation, we opt to employ a similar SPH interpolation since we can reliably use it for both our position based corrective strategy from Anistropic Position Correction and for interpolating arbitrary physical quantities. To convert particle flow velocity from the grid to the particles, we employ a tri-linear interpolation.

Instead of completing an advection step based on a grid, the momentum is naturally transferred by PIC utilising the particles. However, PIC is known to result in numerical diffusion due to the repetitive back-and-forth interpolation of velocities. For this purpose, (Brackbill & Ruppel, 1986) introduced FLIP. In FLIP, only the transition from one time step to the next in the grid velocity is projected back to the particles. The updated particle speed for the next time step is obtained by adding this delta to the particle velocities. The velocity field on the grid is then used to move the particles. FLIP does not experience numerical dissipation, in contrast to PIC. However, because there is no viscosity at all in the simulation, FLIP can exhibit noisy behaviour. With the use of a scalar parameter called v, the PIC and FLIP velocities are linearly blended as in (Zhu & Bridson, 2005), we solve this issue as follows:

$$u = PIC\ /\ FLIP(u^*) = \alpha_v FLIP(u^*) + (1 - \alpha_v) PIC(u^*) \tag{16}$$

4.3 Preserving Thin Sheets

The key component of our approach is an algorithm that prevents thin sheets from rupturing by adding additional particles to vulnerable areas. A general overview of our process is shown in Figure 4. We start by removing thin fluid regions. To prevent collisions, fresh particles are cautiously injected into such locations at newly computed candidate places. In the subsections that follow, we go into greater depth about these steps.

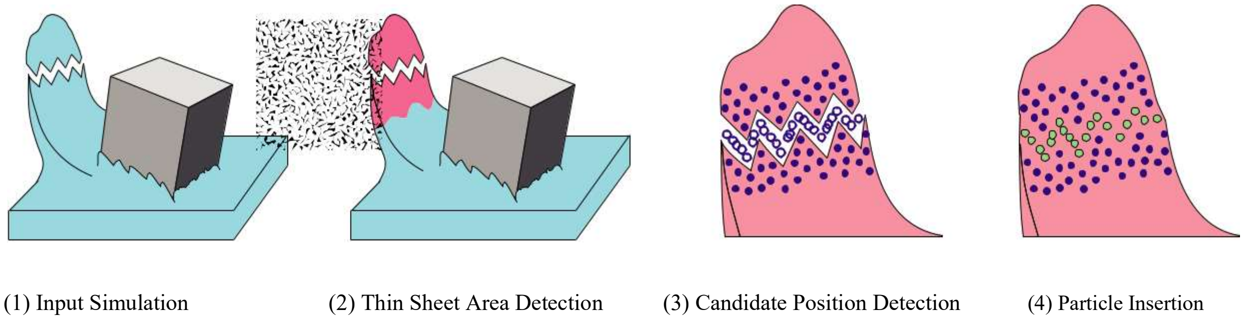
(1) Input Simulation
(2) Thin Sheet Area Detection
(3) Candidate Position Detection
(4) Particle Insertion

Fig. 1. (1) Given a collection of particles as input, (2) Particles in thin regions are computed. Filling out breaking sheets, (3) We calculate the potential insertion places, and (4) We place the particles at the appropriate distance.

4.3.1 Thin Particle Extraction

We can determine the so-called thin particles, which produce thin patches, by evaluating the flexibility of the distributions of surrounding particles that may eventually fracture. We use (Yu & Turk, 2013)'s anisotropic kernel approach to do this. For each particle, we determine the weighted mean covariance of the particles C as follows:

$$C_i = \frac{\Sigma_j(P_j - \bar{P}_i)\,(P_j - \bar{P}_i)^T W_{smoot}\ \ (P_j - P_i, \alpha_\rho d_0)}{\Sigma_j W_{smoot}\ \ (P_j - P_i, \alpha_\rho d_0)} \quad (17)$$

Where

$$\bar{P}_i = \frac{\Sigma_j P_j\, W_{smo}\quad (P_j - P_i, \alpha_\rho d_0)}{\Sigma_j W_{smoot}\ \ (P_j - P_i, \alpha_\rho d_0)} \quad (18)$$

The accompanying C_i Singular Value Decomposition (SVD) provides the following eigenvectors and eigenvalues for the direction and stretch of surrounding particle positions:

$$C_i = R \begin{bmatrix} \sigma_1 & & \\ & \sigma_2 & \\ & & \sigma_3 \end{bmatrix} R^T \quad (19)$$

where $(\sigma_1 > \sigma_2 > \sigma_3)$ denotes the eigenvalues in ascending order of magnitude and R and σ_n the eigenvector matrix, respectively. The degree of stretch is indicated by the eigenvalues. Keep in mind that the SVD is only used when the particle density is below a certain threshold, $\rho(p_i) < \alpha_{\partial\Omega}\rho_0$; otherwise, diag($\sigma_1$, σ_2, σ_3) = I is used in its place. This may be carried out relatively quickly because the SVD is only computed close to the liquid surface. Thin particles can be distinguished by:

$$\sigma_3 \leq \beta_{thin}\sigma_1 \quad (20)$$

where β_{thin} refers to a cutoff point that establishes the level of thinness. We applied the parameters β_{thin} = 0.2 and $\alpha_{\partial\Omega} = 0.7$ in our implementation. In the subsequent processes, the location of new particles is determined using these thin particles. Algorithm 1 displays the pseudo code for our thin sheet preservation.

4.3.2 Particle Insertion and Removal

The available openings are typically highly thick, therefore they cannot be employed carelessly. We outline a straightforward technique to introduce candidates with the right level of sparsity in order to avoid the formation of excessively huge numbers of particles. Let *I* be the last applicant on the list. Given by the lowest density, entry I_1 is the least dense contender within S.

$$I_1 = \arg\min_{\mathrm{j}\in\mathrm{S}} \rho(p_j) \quad (21)$$

$$I_2 = \arg\min_{\mathrm{j}\in N_{I_1}} ||p_j - p_{I_1}|| \quad (22)$$

We look for pairs (p_i, p_j) that span breaking holes in the fluid volume inside the collected thin particles. A candidate place for insertion is noted as the pair's midway, (p_i+p_j)/2. In our method, we select pairings (p_i, p_j) that meet each of the requirements below:

$$\begin{cases} \beta_{min}d_0 \ \leq \ \left\|p_j - p_{I_1}\right\| \ \leq \ \beta_{min}d_0 \\ \sum_k W_{smooth}\left(\frac{(p_i+p_j)}{2} - p_k, \beta_{min}d_0\right) = 0 \\ (p_j - p_i).(u_j - u_i) > 0 \end{cases} \tag{23}$$

In comparison, if an added particle pi meets at the very least one of the two criteria, it is withdrawn once again:

$$\text{Any}\begin{cases} \rho(p_i) > \ \beta_{max\rho}\rho_0 \ and \ \sigma_3 \ \geq \ \beta_{thin}\sigma_1 \\ for\ any\ particle\ j\ \left\|p_i - p_j\right\| \ \leq \ \beta_{dist}d_0 \end{cases} \tag{24}$$

Algorithm 1 displays the pseudo code for our thin sheet preservation.

Algorithm 1 PRESERVE THIN FLUID SHEETS (Ando, et al., 2012)

```
 1: C ← Compute Covariance Around P by Eq. 8
 2: $RΣR^T$ ← SVD(C)
 3: P ← Thin Sheet Particles for Extraction (Σ) by Eq. 20
 4: for all (i, j) ∈ P do
 5:     if a pair ($P_i$, $P_j$) satisfies Eq. 23 then
 6:         S ← Add a fresh candidate ($P_i$ + $P_j$)/2
 7:     end if
 8: end for
 9: $I_1$ ← by Eq. 21 in S
10:     for n = 2, 3, 4, ⋯ do
11:         $I_n$ ← search($I_{n-1}$) in S as Eq. 22
12:     end for
13:     Insert New Particles I
14:     Collapse Particles that Satisfy Eq. 24
```

We go over three extensions to the simulation algorithm we've been talking about so far in the subsections that follow. These include adaptivity, which lowers the overall number of particles included in the simulation, and an anisotropic position adjustment for uniform particle distribution.

4.4. Adaptively Sampled Particles

The quality of the surfaces produced by the FLIP process is directly influenced by the quantity of liquid particles. Typically, one simulation grid cell starts out with eight particles. Because the FLIP technique cannot record fluid movement smaller than a single grid cell, we can save a considerable amount of computation resources through iteratively resampling the huge volume of the fluid velocity with fewer particles while keeping a high particle sampling along the visible surfaces. By doing so, we can reduce the number of particles utilised to sample the bulk volume while concentrating calculations on the surface detail that is apparent and retaining the densely packed particles there. (Adams, et al., 2007), (Hong, et al., 2009) and (Hong, et al., 2008) are the sources of inspiration for our adaptive sampling approach, which we have modified to fit included into the FLIP simulation framework. The prior techniques have undergone two adjustments. First, unlike (Hong, et al., 2009) and (Hong, et al., 2008), we merely examine if particles are present next to the free surface for particle fusion and separation instead of placing numerous layers and assessing Particle separation via Reynolds number analysis. Second, instead of looking for nearby free space like (Adams, et al., 2007) did due to the fact that the FLIP simulation is steady independent of the particle distance, we just randomly position the particles. In addition, we outline a brand-new, straightforward, parallel method for merging particles.

Large particles must be removed from surface cells in order to stop the merger. We divide the large particle into the appropriate number of smaller particles, which are scattered randomly over a radius of d_n surrounding the source particle. The freshly formed little particles' radii are initialised in accordance with the giant particle's mass, which is equally distributed among them. The other physical quantities have all been duplicated.

In a typical SPH simulation, one such particle split could result in an abruptly huge spring force. However, this is not a problem because the standard FLIP algorithm doesn't really apply forces based on particle density. To ensure that the particles are spaced evenly, we explain a progressive particle displacement technique in the following section. Although a particle split affects the force produced, we did not see any instability as a result of the particle split because of its tiny magnitude.

The combining and splitting operations can be carried out step-by-step in tandem. Since no neighbourhood information is needed, splitting operations can be carried out relatively quickly. In our implementation, we introduce a layer of cells between the surface and the deep fluid layers where no conversions between small and large particles are made. As a result, excessive combining and splitting procedures are avoided at the junction of deep and surface cells. We discovered that adaptively merging particles considerably enhances performance while also reducing the total number of particles in the simulation. This kind of particle merging can affect the velocities that are estimated for the deep cells during the FLIP simulation. We have discovered that this influence is minimal and hardly moves the surface that is visible. Algorithm 2 displays the pseudo code for our adaptive sampling algorithm.

Algorithm 2 ADAPTIVELY SAMPLE PARTICLES (Ando, et al., 2012)

```
 1: DeepCells ← FluidCells − dilate(EmptyCells,twice)
 2: SurfaceCells ← FluidCells − dilate(DeepCells,once)
 3: repeat
 4:     for all particles ($P_i$, $P_j$) ∈ DeepCells do
 5:         skip if $||P_i - P_j|| > (d_i + d_j)/2$
 6:         skip if $\arg\min_k ||P_i - P_k|| = j$
 7:         skip if $\arg\min_k ||P_j - P_k|| = i$
 8:         n ← Sum of small particles.
 9:         skip if $n > m$
10:         $P_k$ ← Generate a new particle.
11:         $d_k \leftarrow n^{\frac{1}{D}} d_0$
12:         Remove the particles $P_i$, $P_j$
13:     end for
14:     until no particles merge
15:     for all merged particle $P_k$ ∈ SurfaceCells do
```

```
16:     for  i = 1 → number of particles in $P_k$  do
17:         $P_i$ ← Generate a new particles at random position around $P_k$ within radius $d_k$ .
18:         $d_i$ ← $d_0$
19:     end  for
20:     Remove particle $P_k$
21:   end  for
```

4.5. Mean Curvature Flow

We will first go through how to compute the original fluid surface's smoothed equivalent, which is represented as a triangle mesh. Sussman demonstrated that the distance between the liquid surface and a different version of the surface projected by a volume-preserving mean curvature flow can be related to the surface tension forces integrated over the span of a time step (VCF). In the limit, the VCF solution will condense to one or more spheres with a volume equal to that of the fluid component, which is the precise outcome that an isolated surface tension fluid simulation

with viscosity would produce. Keep in mind that the surface tension may cause a single component to break up into numerous spheres. We use a VCF to calculate the smoothed surfaces S and T, but we solve the VCF directly on the surface mesh rather than the grid. Additionally, we apply this technique to both layers of our simulation: the grid's surface tension forces and the fluid's surface sub-grid scales. S therefore creates a division between tiny spatial scales, which will be handled by our surface dynamics model, and larger scales, which will be handled by the Eulerian fluid simulation in our case. We must first find a VCF for the initial, perhaps extremely detailed surface of the liquid. We may exploit the wealth of knowledge on curvature fluxes for meshes in this situation. The collection of vertex coordinates X of a triangle mesh's general mean curvature flow can be expressed as

$$X_t = \gamma \nabla^2 X \tag{25}$$

4.6. Surface Tension

One of the most prevalent and significant physical properties that reveal microscopic effects of fluids is surface tension. The simulation of surface tension is become more and more alluring due to the rising need for specifics and realism in fluid simulation. The cohesion of the fluid molecules causes surface tension. Because the molecules inside a fluid are subject to forces coming from all angles, they eventually establish mechanical equilibrium. Surface tension is produced even if the resultant force of fluid molecules on the surface, which is not zero, indicates the fluid's interior. The Laplacian law also states that surface tension has the ability to reduce surface area. For instance, when no external force is applied, the droplets condense into a sphere. (Wang, et al., 2017)

Based on (Morris, 2000) theories, we explicitly model surface tension forces. A fluid's molecules are subject to the attraction forces of its nearby molecules. These intermolecular forces balance one another and are equal in all directions inside the fluid. On the other hand, the forces on molecules near the free surface are imbalanced. The surface normal to the fluid is the direction in which the net forces, or surface tension forces, act. They also have a tendency to reduce the surface's curvature. The force increases with curvature.

Every particle in a particle-based fluid simulation is given a colour field value i, or c_i, and all of the fluid's particles share the same colour of the field value. This is known as the surface tension model. The colour field interpolation formula can be obtained by adding c_i to the SPH interpolation equation as follows:

$$c_i = \sum_j \frac{m_j c_j}{\rho_j} w_{ij} \tag{26}$$

It is possible for the normal vector of the surface to point toward the fluid's interior when the normal vector of the surface is calculated using $n = \nabla c$. And the divergence of the normal vector can be used to calculate surface curvature (Wang, et al., 2017). Its shape is displayed as follows:

$$k = \frac{-\nabla^2 c}{|n|} \tag{27}$$

The normal vector and surface curvature can be used to structure the surface tension of a fluid is shown as follows:

$$F^s = \sigma k n = -\sigma \nabla^2 c \frac{n}{|n|} \tag{28}$$

4.7. Viscosity

The term "viscosity," which is frequently used in fluid simulation, simply refers to a fluid's propensity to restrict flow and manifests itself as thickness. For instance, water has low viscosity and flows freely. Because honey is thick and opposes flow, it has a high viscosity. The scalar v has direct control over the level of viscosity. It should be noted that when mimicking water, this phrase is frequently ignored. This factor, which would otherwise be resolved by a Poisson equation, is frequently well replaced by the level of numerical degradation introduced by interpolation mistakes.

$$\frac{\partial V}{\partial t} = F + v\nabla^2 V - V.\nabla V - \frac{\nabla \mathrm{p}}{\rho} \tag{29}$$

$$\nabla.V = 0 \tag{30}$$

When the SPH rule is applied to the viscosity term $\mu\nabla^2 v$, asymmetric forces are once more produced.

$$f_i^{viscosity} = \mu \nabla^2 v(r_a) = \mu \sum_j m_j \frac{v_j}{\rho_j} \nabla^2 w(r_i - r_j, h) \tag{31}$$

due to the fact that each particle's velocity field is unique. There is a natural way to symmetrize the viscosity forces by employing velocity differences because they are solely reliant on velocity differences and not on absolute velocities:

$$f_i^{viscosity} = \mu \sum_j m_j \frac{v_j - v_i}{\rho_j} \nabla^2 w(r_i - r_j, h) \tag{32}$$

4.8. Houdini

Since most mathematical equations are already incorporated in the form of microsolvers and the main objective of the project is the implementation of thin sheet preservation techniques, the major difficulty is to study the huge proportion of microsolvers and decide which ones to use and modify them in accordance with the requirements. After analysing the Navier- Stokes equations and understanding the concepts that regulate fluid motion, it is relatively easy to implement them in Houdini. The programme must enable users to quickly and easily construct any type of FLIP animation they desire, complete with accurate fluid particle structure and movement. The HDA tool has a series of steps that start with creating the particles needed for an emitter source, followed by setting up the boundaries for the FLIP sim environment, using a custom procedure to direct the fluid simulation to follow a specific pattern using user-defined velocity directions as normals, and finally controlling the vdb setup for the collision geometry that is configured to interact with the fluid simulation. These are the initial setup parameters for the tool that the user can change and adjust to determine the desired bounds for the simulation.

The implementation of thin sheet preservation approaches is done in the subsequent step using a variety of microsolvers, VEX code, and VOP networks to create a particle system that accurately represents the fluid properties and complies with the preservation methodology. Other fundamental principles of fluid simulation, such as surface tension, viscosity, and curvature forces, are also implemented together with thin sheet techniques, allowing the user additional control over the simulation using present properties. The default solver solver properties are disregarded and not used in the project's key principles are implemented using a variety of microsolvers, attribute wrangles, pop wrangles, and VOP networks, so the tool can be true to its implementation. The surface field's curvature and gradient are computed, and we use the two particular fields to push and pull in convex and concave zones such that the particle field tends to form segments and shape in attractive ways. Various other assets apply surface tension based on curvature decimation. By integrating the curvature decimation approach with additional custom forces and a PBD-based strategy, we are taking a different approach in this case.

The core SOP solvers deal with the implementation of the thin sheet techniques , viscosity, surface tension and curvature control in the borders. Another feature allowing the user to generate bubbles inside the fluid simulation when required is created which can be used for underwater sims, any kind of syrups, Honey etc. Additional tests were conducted to examine the APIC method (swirly kernel) in Houdini 18.5's FLIP solver with the default one (splashy kernel), which does not employ a hole-filling strategy. The mask property regulates the weight or contribution for the cohesiveness that determines the behaviour in those tests. The red sections is where complete energy is applied to reduce the surface area, which results in spherical formations.

Re-seeding is typically used to give users more control over particle generation, but it also has an impact on mesh generation. This is mainly because re-timing can produce strange pulsing artefacts, and the only way to avoid this is to over-smooth, which can occasionally be very expensive and eliminate nice details. Boundary conditions can also cause flickering due to dense particle clusters closer to colliders. Instead of doing this, I employ a technique to redistribute the particles in order to achieve a more even distribution among them. By doing this, I am able to create very nice and smooth edges without using as many particles, as well as a mesh that is cleaner and smoother without sacrificing any details.

Various example simulations have been generated to demonstrate the tool efficiency. In the Fig 2 presumably blood simulation used to demonstrate the various properties of the tool and can be clearly portray the implementation concepts.

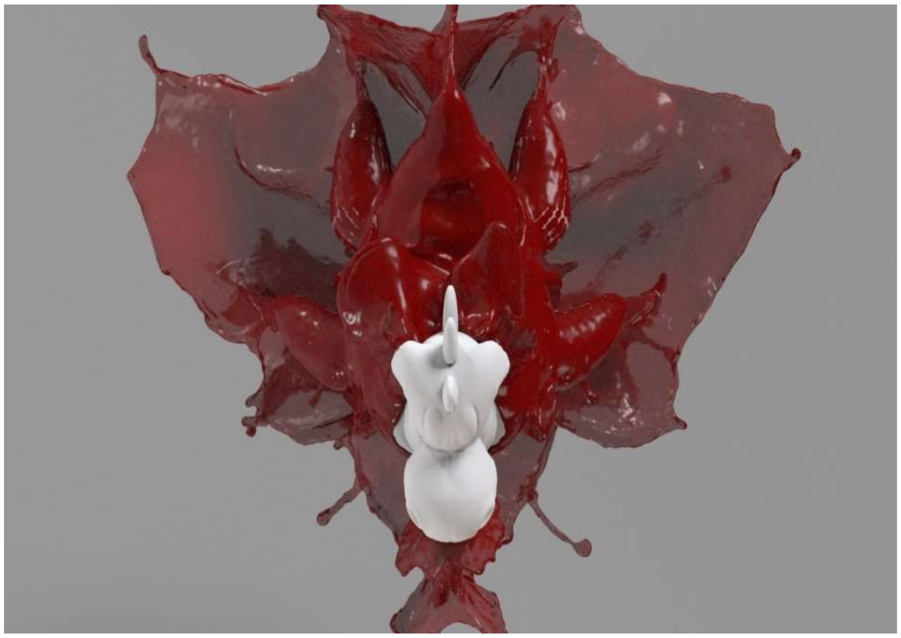

Fig 2. Simulation Parameter: Particle separation : 0.01, Thin sheet point count: 5, Thin sheet distance :0.05, Thin sheet max vel : 80, Thin sheet scale: 80, Drag :1, Reseeding features On.

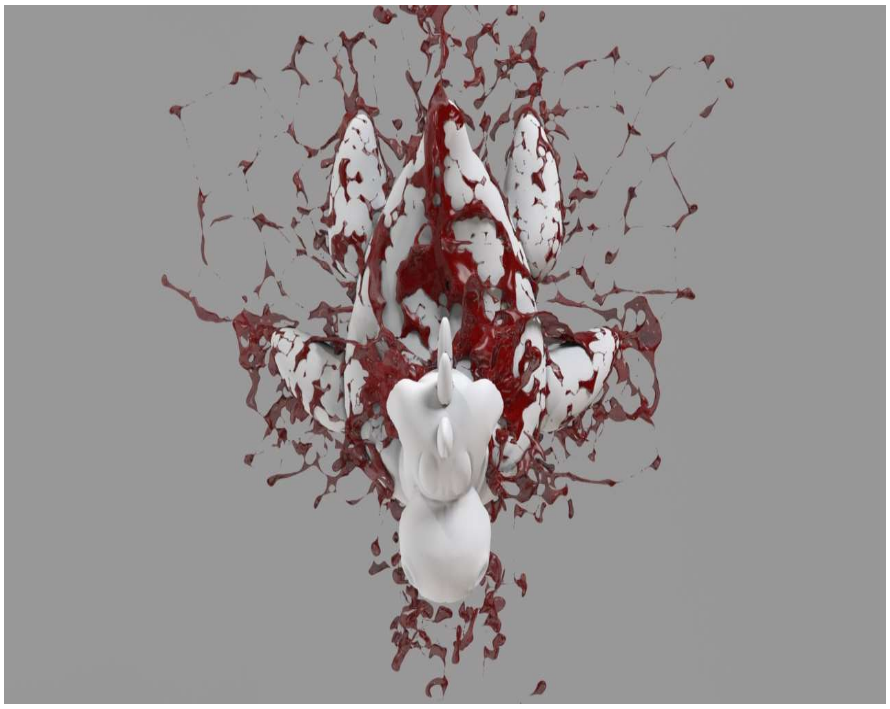

Fig 2. Simulation Parameter: Particle separation : 0.01, Thin sheet point count: 15, Thin sheet distance :0.02, Thin sheet max vel : 180, Thin sheet scale: 120, Drag :1, Reseeding features On.

Fig 3 shows the simulation of honey with also using the bubbles feature that will create air bubbles inside the fluids. Here in both fig 3 & 4 the main features such as Surface tension, Viscosity and border curvature thickness come into play.

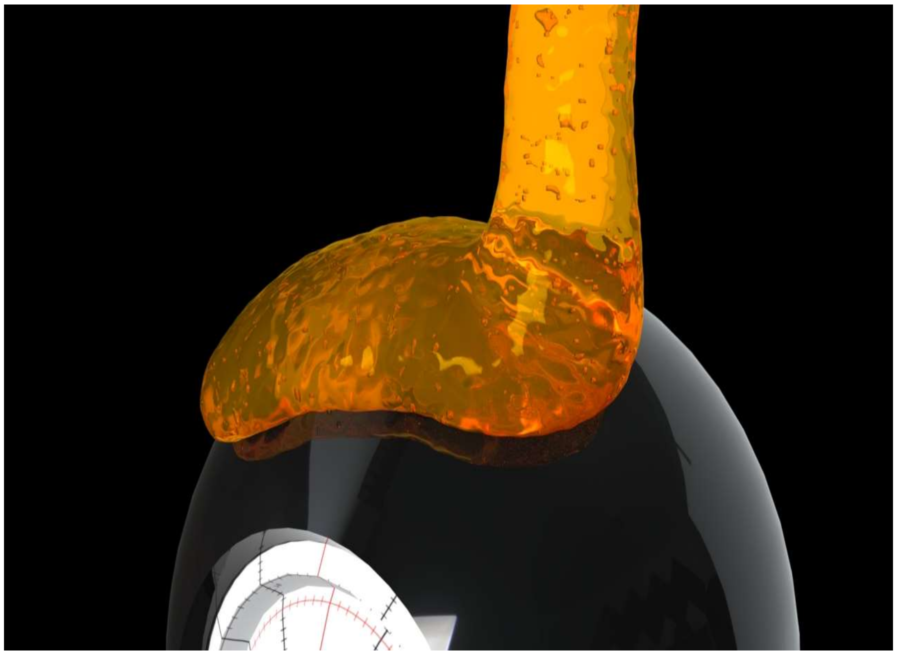

Fig 3. Simulation Parameter: Particle separation : 0.003, Thin sheet point count: 0, Thin sheet distance :0, Thin sheet max vel : 0, Thin sheet scale: 120, Viscosity : 75, Min age: 0.6, Liquidity : 40, Solidity: 60, Drag :0.001, with Detailing and Reseeding features On.

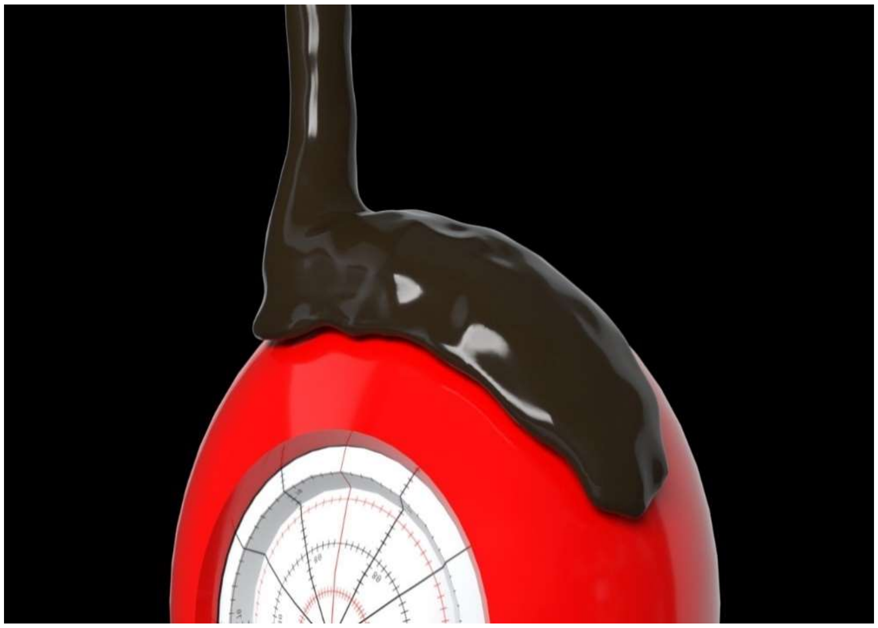

Fig 4. Simulation Parameter: Particle separation : 0.003, Thin sheet point count: 0, Thin sheet distance :0, Thin sheet max vel : 0, Thin sheet scale: 120, Viscosity : 50, Min age: 0.2, Liquidity : 60, Solidity: 40, Drag :0.001, with Detailing and Reseeding features On.

5. Results

Without any prior experience with fluid simulations, a solid foundation in knowledge and understanding of the fundamental ideas was required before any work could start. Apart from the implementation of Thin Sheet preservation techniques the tool has been extended with additional features and more control over concepts such as surface tension, viscosity, mean border curvature control. The current Houdini solvers and fluid tools were studied in addition to the scientific and technical literature review in order to gain insight into how they operate and potential modifications.

Despite the problems that had not been fixed by the time this research was finished, the fluid simulation is generally effective and producing satisfying results. The tool successfully implements thin sheet preservation techniques by detecting principal stretch and thickness of the fluid and inserting additional points based on user defined rules and cutoffs. This can be seen with a simple crown sim configuration, where the Anistropic Fluids Tools simulation time is far faster than the shelf tool. The tool also offers greater user flexibility and makes it simple to add extra details. We might achieve more attainable and effective results by addressing those problems and making additional enhancements.

6. Known Issues

Even if the majority of the objectives were met, the application could have used a little more time. Although it includes all of the necessary components of a fluid solver, they are not yet ready for commercial use. There are a lot of more complex facets to this topic. The tools at the moment is really efficient but can be made even more efficient by constructing the solver from scratch and removing any unnecessary concepts from it. In order to achieve such efficiency the anistropy can be calculated through covariance matrix singular value decomposition, Rotational iterative SVD solver can be constructed in such a way that absolute maximum non diagonal element turns to zero with every single iteration. Rotational SVD seems to be most efficient for small (3 x 3) matrices decomposition. Typical converge iterations are below 6. Mesh Flickering is observed in the final rendered output which should be further evaluated. This flickering can also be seen when the fluid gets in contact with any colliding mesh.

7. Conclusion and Further Work

In conclusion, a successful application of a particle-based algorithm that carefully re-samples particles to retain thin fluid sheets has been shown. To ensure uniform particle distribution. We adaptively sampled particles in the deep sea while keeping dense particles close to surfaces to lower the cost of neighbourhood computations. By calculating the stretch of the nearby particle distributions, we found thin fluid locations in order to keep fluid sheets from breaking. Most significantly, a thorough knowledge of solvers and fluid simulations was gained, along with expertise in using Houdini for such simulations. Due to the popularity of fluid simulations as a research area, the rapid pace of new advancements, the intricacy of the tools and microsolvers available in Houdini, and other factors, a substantial amount of time was spent researching and contrasting different approaches for this project. The tool can be made more efficient by use of covariance matrix and create a rotational iterative SVD solver. More user control over external forces can be added by allowing them to add inbuilt nodes and control the fuild sim. This could open more possibilities for creative fluid animations. Adhesion can be implemented to control the stickyness of particles to any collision meshes. Right now the tool is limited to small and medium scale level simulations with more work it can be used to generate large scale simulations as well. I believe with further more development and reconstruction of this tool will be production ready for creating complex fluid animation with all required attributes at the display and allowing user for more development can be achieved

Bibliography

Adams, B., Pauly, M., Keiser, R. & Guibas, L. J., 2007. *Adaptively Sampled Particle Fluids,* s.l.: ACM Transactions on Graphics, Vol. 26, No. 3, Article 48.

Agrotis, A., 2016. *A Fluid Implicit Particle (FLIP) Solver Built in Houdini,* s.l.: Bournemouth University, NCCA.

Akinci, N., Akinci, G. & Teschner, M., 2013. *Versatile Surface Tension and Adhesion for SPH Fluids,* New York: Association for Computing Machinery.

Ando, R., Thurey, N. & Tsuruno, R., 2012. *Preserving Fluid Sheets with Adaptively Sampled Anisotropic Particles,* s.l.: IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS.

Ando, R. & Tsuruno, R., 2011. *A Particle-based Method for Preserving Fluid Sheets,* s.l.: Eurographics/ ACM SIGGRAPH Symposium on Computer Animation (2011).

Anon., 2018. *Smoothed-particle hydrodynamics.* [Online]
Available at: https://en.wikipedia.org/wiki/Smoothed-particle_hydrodynamics
[Accessed 17 August 2022].

Anon., 2019. *Surface tension.* [Online]
Available at: https://en.wikipedia.org/wiki/Surface_tension
[Accessed 17 August 2022].

Bargteil, A. W., Goktekin, T. G., O'Brien, J. F. & Strain, J. A., 2006. *A semi-Lagrangian contouring method for fluid simulation,* s.l.: ACM Transactions on Graphics.

Brackbill, J. & Ruppel, H., 1986. *FLIP: A method for adaptively zoned, particle-in-cell calculations of fluid flows in two dimensions,* s.l.: Journal of Computational Physics, vol. 65, pp. 314–343.

Brackbill, J. U. & Ruppel, H. M., 1986. *FLIP: A method for adaptively zoned, particle-in-cell calculations of fluid flows in two dimensions,* s.l.: Journal of Computational PhysicsVolume 65, Issue 2.

Bridson, R. . E., 2008. *Fluid Simulation for Computer Graphics,* s.l.: CRC Press.

Englesson, D., Kilby, J. & Ek, J., 2011. *Fluid Simulation Using Implicit Particles,* s.l.: s.n.

Enright, D., Fedkiw, R., Ferziger, J. & Mitchell, I., 2002. *A Hybrid Particle Level Set Method for Improved Interface Capturing,* s.l.: s.n.

Evans, M. & Harlow, F., 1957. *The particle-in-cell method for hydrodynamics calculations,* s.l.: s.n.

Foster, N. & Fedkiw, R., 2001. *Practical Animation of Liquids,* s.l.: Proceedings of the 28th annual conference on Computer graphics and interactive techniques, ser. SIGGRAPH '01.

Gingold, R. A. & Monaghan, J. J., 1977. *Smoothed particle hydrodynamics: theory and application to non-spherical stars,* s.l.: Monthly Notices of the Royal Astronomical Society.

Harlow, F. H., 1962. *The Particle-in-Cell Computing Method for Fluid Dynamics,* s.l.: Methods in Computational Physics, 3, 319-343..

Hirt, C. W. & Nichols, B. D., 1981. *Volume of fluid (VOF) method for the dynamics of free boundaries,* s.l.: Journal of Computational Physics, vol. 39, pp. 201–225.

Hong, W., House, D. H. & Keyser, J., 2008. *Adaptive Particles for Incompressible Fluid Simulation,* s.l.: Computer Graphics International manuscript vol 24.

Hong, W., House, D. H. & Keyser, J., 2009. *An Adaptive Sampling Approach to Incompressible Particle-Based Fluid,* s.l.: EG UK Theory and Practice of Computer Graphics.

Irving, G., Guendelman, E., Losasso, F. & Fedkiw, R., 2006. *Efficient Simulation of Large Bodies of Water by Coupling Two and Three Dimensional Techniques,* s.l.: SIGGRAPH '06: ACM SIGGRAPH 2006 Papers.

Kim, J. et al., 2006. *Practical Animation of Turbulent Splashing Water,* s.l.: Eurographics/ ACM SIGGRAPH Symposium on Computer Animation.

Kim, J. H. & Lee, J., 2020. *Efficient preservation and breakup of liquid sheets using screen-projected particles,* s.l.: PLoS ONE 15(2): e0227590.

Lucy, L. B., 1977. *A numerical approach to the testing of the fission hypothesis,* Switzerland: Institute of Astronomy, Cambridge.

Mercier, O. et al., 2015. *Surface Turbulence for Particle-Based Liquid Simulations,* New York: Association for Computing Machinery.

Monaghan, J. J., 1994. *Simulating Free Surface Flows with SPH,* s.l.: Journal of Computational PhysicsVolume 110Issue 2.

Morris, J. P., 2000. *Simulating surface tension with smoothed particle hydrodynamics,* s.l.: International Journal for Numerical Methods in Fluids.

Muller, M., 2009. *Fast and robust tracking of fluid surfaces,* s.l.: Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation.

Müller, M., Charypar, D. & Gross, M., 2003. *Particle-Based Fluid Simulation for Interactive Applications,* s.l.: Eurographics/SIGGRAPH Symposium on Computer Animation.

Osher, S. & Sethia, J. A., 1988. *Fronts propagating with curvaturedependent speed: algorithms based on hamilton-jacobi formulations,* s.l.: Journal of Computational Physics.

Premože, S. et al., 2003. *Particle-Based Simulation of Fluids,* s.l.: EUROGRAPHICS 2003.

Raveendran, K., Wojtan, C. & Turk, G., 2011. *Hybrid Smoothed Particle Hydrodynamics,* Austria: Eurographics/ ACM SIGGRAPH Symposium on Computer Animation.

Stomakhin, A., Moffat, A. & Boyle, G., 2019. *A Practical Guide to Thin Film and Drips Simulation,* Los Angeles, CA, USA: n. In Proceedings of SIGGRAPH '19 Talks.

Strantzi, T., 2016. *Fluid Simulation Using Smoothed Particle Hydrodynamics (SPH), Master's Project,* s.l.: Bournemouth University, NCCA.

Sussman, M. & Ohta, M., 2009. *A Stable and efficient method for treating Surface Tension in Incompressible Two Phase Flow,* s.l.: SIAM Journal on Scientific Computing, Vol 31.

Thurey, N., Wojtan, C., Gross, M. & Turk, G., 2010. *A Multiscale Approach to Mesh-based Surface Tension Flows,* New York: Association for Computing Machinery.

Wang, X.-K.-K. et al., 2017. *Surface Tension Model Based on Implicit Incompressible Smoothed,* s.l.: JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 32(6):.

Wang, Z., Yang, J. & Stern, F., 2009. *An improved particle correction procedure for the particle level set method,* s.l.: Journal of Computational Physics.

Wojtan, C., Fischer, M. . M.-. & Brochu, T., 2011. *Liquid simulation with mesh-based surface tracking,* s.l.: SIGGRAPH '11: ACM SIGGRAPH 2011 Courses.

Wojtan, C., Thurey, N., Gross, M. & Turk, G., 2010. *Physics-inspired topology changes for thin fluid features,* s.l.: ACM Trans. Graph, vol. 29.

Wojtan, C. & Turk, G., 2008. *Fast Viscoelastic Behavior with Thin Features,* s.l.: ACM Transactions on Graphics, Vol. 27, No. 3, Article 47.

Yu, J. & Turk, G., 2013. *Reconstructing Surfaces of Particle-Based Fluids Using Anisotropic Kernels,* s.l.: ACM Trans. Graph. 32, 1, Article 5.

Zhu, Y. & Bridson, R. E., 2005. *Animating sand as a fluid,* s.l.: ACM Transactions on GraphicsVolume 24Issue 3.