3D Particle-Based Fluid Simulation and Rendering

YIJIN WANG, McGill University

In this paper, we propose an approach for 3D fluid animation using particle-based simulation techniques, specifically Smoothed Particle Hydrodynamics (SPH). We explore optimization techniques that reduce computational costs without compromising the quality of the results. Additionally, we present a rendering method that maps the particle space to color values in the hue-saturation-value (HSV) color space and then converts them to red-green-blue (RGB) values to produce visually appealing fluid animations. A comparison for choosing different representations of particle space is also shown.

Code: https://github.com/yijinwang7/3D_particle-based_fluid
Video: https://youtu.be/jRxDt15eg2s

CCS Concepts: • **Computing methodologies**;

Additional Key Words and Phrases: 3D SPH Fluid, hue-saturation-value (HSV), red-green-blue (RGB), rendering, texture space, screen space

ACM Reference Format:
Yijin Wang. 2023. 3D Particle-Based Fluid Simulation and Rendering. 1, 1 (April 2023), 4 pages. https://doi.org/0000001.0000001

1 INTRODUCTION

Nowadays, Smoothed Particle Hydrodynamics (SPH) and point splatting have become mature technologies to simulate and render particle-based fluid. In this paper, we start from the SPH method described in [Müller et al. 2003] for simulation. We first omit mass conservation and convective terms in the Eulerian formulation used in the grid-based fluid system. We then use the idea in [Akinci et al. 2013] to compute the surface tension and add it to the Navier-Stokes equation.

Also following the idea from [Müller et al. 2003], we will introduce methods to reduce the computational complexity. For a single particle updating, instead of searching all the other particles, we only search for its neighbours.

However, rendering by point splatting is dramatically expensive and hence the requirements for computer configuration have exceeded most of our computers. Regarding reproducibility as the most important thing, how to choose a much cheaper rendering way without the loss of basic characters of fluid has become a big problem. Inspired by surface splatting [Zwicker et al. 2001], texture mapping [Heckbert 1986] and changing coordinates in color space [Malacara 2011], we create a novel and simple rendering method. We construct this method by giving the color of particles based on an innovative mapping, which maps the particle space to hue-saturation-value (HSV) color space. Since OpenGL does not support HSV color space, we translate the HSV color space into RGB color space by the algorithm introduced in [Smith 1978]. Furthermore, since the color is determined by the mapping from the particle space, the resulting animation is influenced by the choice of representation for the particle space. We will present different results obtained by using the position and velocity of the particles as the representation.

Author's address: Yijin Wang, McGill University, yijin.wang2@mail.mcgill.ca.

2023. XXXX-XXXX/2023/4-ART $15.00
https://doi.org/0000001.0000001

2 RELATED WORK

[Müller et al. 2003] and [Akinci et al. 2013] provide detailed SPH concepts to simulate the particle-based fluid. Along with the paper [Müller et al. 2003], there is also an incomplete demo in 2D [Schuermann 2016]. Based on this demo, I add the surface tension, change the system into 3D and make it visible via OpenGL. I also change FreeGLUT to GLFW in order to make it easier to reproduce. A new rendering method is also created.

[Zwicker et al. 2001] gives a basic idea of point splatting, and [Heckbert 1986] described what is texture mapping used by point splatting. [Malacara 2011] gives an idea of transforming coordinates from rectangular form to polar form, which will be used in our mapping. [Smith 1978] shows how to transform HSV color space to RGB color space.

3 METHODS

3.1 Methods for Simulation

Firstly, we manipulate the Eulerian formulation based on [Müller et al. 2003]. We omit mass conservation and convective terms in the Eulerian formulation used in the grid-based fluid system. So the Navier-Stokes equation becomes :

$$\rho(\frac{d\boldsymbol{v}}{dt}) = f^{external} + f^{pressure} + f^{viscosity} + f^{surface} \tag{1}$$

More specifically, for the particle i, we have:

$$f_i^{external} = \rho \boldsymbol{g} \tag{2}$$

and

$$f_i^{pressure} = -\nabla p = -\sum_j m_j \frac{p_i + p_j}{2\rho_j} \nabla W(r_i - r_j, h) \tag{3}$$

and

$$f_i^{viscosity} = \mu \nabla^2 \boldsymbol{v} = \mu \sum_j m_j \frac{v_j - v_i}{\rho_j} \nabla^2 W(r_i - r_j, h) \tag{4}$$

with external force density field $\boldsymbol{g}$, particle mass m, viscosity of the fluid μ, velocity field $\boldsymbol{v}$, a density field ρ and a pressure field p. Also, W is the Smoothing Kernel and h is kernel radius defined in [Müller et al. 2003].

Secondly, as described in [Akinci et al. 2013], the surface tension force is :

$$f_i^{surface} = -\gamma m_i (\boldsymbol{n_i} - \boldsymbol{n_j}) \tag{5}$$

where γ is the surface tension constant and $\boldsymbol{n_i}$ is the normal defined by:

$$\boldsymbol{n_i} = h \sum_j \frac{m_j}{\rho_j} \nabla W(|x_i - x_j|) \tag{6}$$

Here, the SPH kernel function W is defined in [Monaghan 1992] and h is the kernel radius.

Finally, we try to reduce the time complexity. [Müller et al. 2003] suggests two efficient ways:

· When computing the density and pressure of particle i, only consider the neighbouring particles located in the ball centered at the particle i, with kernel radius h.

· When computing the surface tension, only consider the surface particles, which are identified by

$$\boldsymbol{n_i} > l \tag{7}$$

where l is a threshold parameter. With excessive experiments, we find that $l = 0.3$ is good for my program.

3.2 Methods for rendering

Firstly, let's revisit the big problem we mentioned in Section 1. Point splatting [Zwicker et al. 2001] is too expensive to reproduce, and we want a cheaper rendering method.

Basically, we could regard the process of rendering as assigning colors to every particle. In order to assign colors based on the properties of our particle system, [Zwicker et al. 2001] introduces the mappings from 2D texture space to 3D object space (our particle system) and from there to 2D screen space. As a result, these two mappings can be combined into 2D to 2D mapping, from texture space to screen space. This triggers a solution to solve the problem. That is, find a simpler mapping from 2D texture space to 2D screen space.

[Heckbert 1986] defines the texture space as follows : it can be either a texture in the usual sense (e.g. cloth, wave, wood) or more generally, a multidimensional space that is mapped to a multidimensional space. It is also motioned that a 2D texture can represent waves which can exactly be used in our fluid system. Strictly following the definition, for the texture space, we want a 2D space which can map to 3D space (our particle system). The simplest mapping we can imagine is this linear mapping :

$$(x, y) \rightarrow (x, y, z) \tag{8}$$

In our particle system, (x, y, z) can be represented by position or velocity. (Specifically, the texture space can be $(\boldsymbol{v_x}, \boldsymbol{v_y})$ derived from velocity space or $(\boldsymbol{p_x^*}, \boldsymbol{p_y^*})$ derived from position space. We will discuss the results by different representations in Section 4). (x, y) can be obtained by a projection from x-y-z space to x-y plane. Furthermore, this projection is surprisingly easy to achieve in programming. We only need to take the x-value and y-value from the particle system.

[Zwicker et al. 2001] described the screen space as a function of summing differently weighted red-green-blue (RGB) values. But evaluating this function is significantly time-consuming. Instead of using a complicated function, one may think to use the RGB color space to represent the screen space directly. However, as we explained previously, we expect the screen space to be in 2D, but the RGB color space is in 3D. This leads us to consider hue-saturation-value (HSV) color space shown in Figure 1. H(hue) is measured as angle in polar coordinates. (e.g. 0° = red, 60° = yellow, 120° = green). S(saturation) is the amount of black. And V(value) is the brightness. If we set the brightness to a constant, we could make the HSV color space to be 2D while maintaining the ability to display all the colors. So we could use the HSV color space as our screen space.

Finally, we try to find a mapping, from texture space ($(\boldsymbol{v_x}, \boldsymbol{v_y})$ or $(\boldsymbol{p_x^*}, \boldsymbol{p_y^*})$) to screen space (HSV color space). [Malacara 2011]

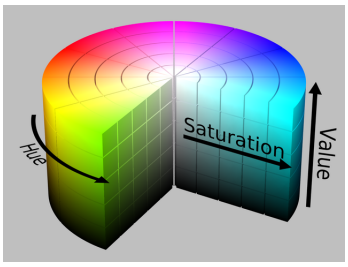

Fig. 1. HSV color space ([SharkD 2010])

introduce a method to mapping a point (x, y) in Cartesian coordinate to polar coordinate where hue is the angular component and saturation is the radial component:

$$H = tan^{-1}(x, y) \tag{9}$$

and

$$S = \|(x, y)\| \tag{10}$$

We need to scale H and S values since the HSV values are in the range of [0,1]. Also, to make our animation more clear and prettier, we set the V to 1.

However, OpenGL does not support HSV color space. So we need a transformation from HSV to RGB. We fulfill this transformation based on the algorithm introduced in [Smith 1978].

The complete algorithm to assign colors is shown in Algorithm 1.

Algorithm 1: Compute Color for Each Particle

Given: particle space (x, y)
Desired: The equivalent R, G, and B values
$H = (tan^{-1}(x, y) + \pi)/(2 * \pi) * 6$, $S = \|(x, y)\|^2$, V = 1
$M = V - S$
$K = V * (V - S * (1 - abs(H(mod2) - 1)))$
$N = V * (V - S * abs(H(mod2) - 1))$
switch H **do**
  case $H < 1$ **do**
    $(R, G, B) = (V, K, M)$
  end
  case $H < 2$ **do**
    $(R, G, B) = (N, V, M)$
  end
  case $H < 3$ **do**
    $(R, G, B) = (M, V, K)$
  end
  case $H < 4$ **do**
    $(R, G, B) = (M, N, V)$
  end
  case $H < 5$ **do**
    $(R, G, B) = (K, M, V)$
  end
  otherwise do
    $(R, G, B) = (V, M, N)$
  end
end

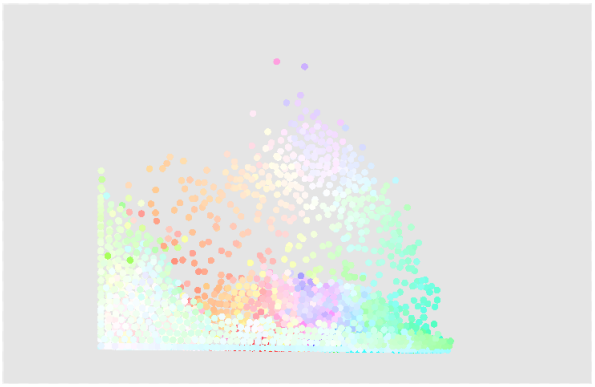

Fig. 2. fluid triggered based on velocity

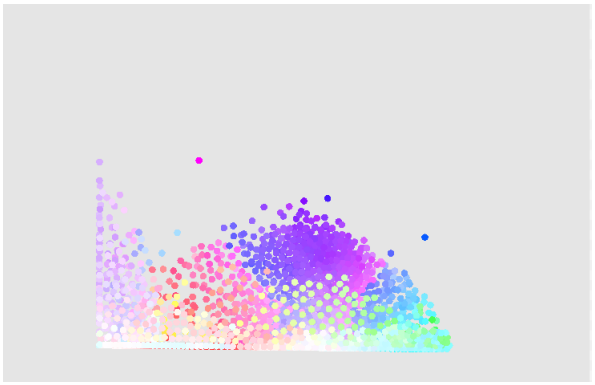

Fig. 3. fluid triggered based on velocity

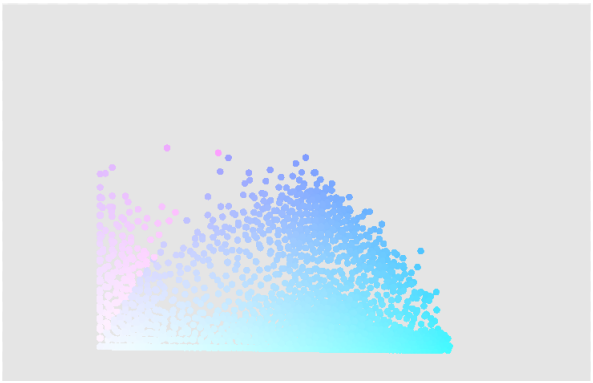

Fig. 4. fluid triggered based on position

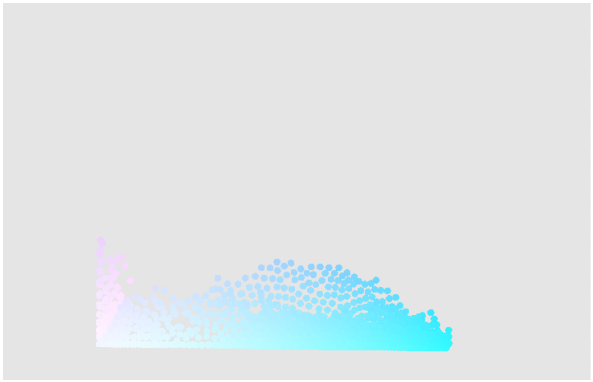

Fig. 5. fluid triggered based on position

4 RESULTS

We create a 3D fluid, imitating pouring liquid into a bottle. In order to make the result more interesting, we lean the bottle a little bit. We name the resulting animation "candy ocean", as the liquid is formed by colorful particles.

As discussed in Section 3.2, we choose velocity space (v_x, v_y) and position space (p_x^*, p_y^*) to represent the texture space (x, y). The results and comparisons are shown below:

4.1 Result of using velocity space (v_x, v_y)

The resulting animation based on velocity space is shown in Figure 2 and Figure 3.

4.2 Result of using position space (p_x^*, p_y^*)

The resulting animation based on position space is shown in Figure 4 and Figure 5.

4.3 Comparisons and Discussions

Comparing the results in Section 4.1 and Section 4.2, we find that the velocity generation performs better than the position generation, which is reasonable. As explained in Section 3.2, We only use the information on the x-y plane to determine the colors. So if particles have a similar position on the x-y plane but different positions on the z-axis, they will have a similar color.

5 CONCLUSIONS

Smoothed Particle Hydrodynamics (SPH) and point splatting have become popular methods to simulate and render particle-based fluid. We have achieved 3D simulation through SPH with some optimizations. However, due to the disadvantage of point splatting such as the expensive and high-requirement for the computer, we are not able to do the point splatting. Instead, we innovate a simpler rendering algorithm based on a mapping from 2D texture space to 2D screen space. Texture space is defined by velocity and position. And we use the HSV color space to represent the screen space. As OpenGL doesn't support HSV color, we transform the HSV color into the RGB color. Then we use OpenGL to visualize the 3D particle-based fluid.

Fluid triggered by the particle's velocity looks more three-dimensional than that triggered by the particle's position, as the x-y plane in 3D position space involves no information of the depth.

Future work could be finding a new rendering method, which is not time-consuming while having the ability to render the particles with real fluid textures.

REFERENCES

N. Akinci, G. Akinci, and M. Teschner. 2013. Versatile Surface Tension and Adhesion for SPH Fluids. *ACM Trans. Graph.* 32, 6, Article 182 (nov 2013), 8 pages. https://doi.org/10.1145/2508363.2508395

P. Heckbert. 1986. Survey Of Texture Mapping. *Computer Graphics and Applications, IEEE* 6 (12 1986), 56–67. https://doi.org/10.1109/MCG.1986.276672

D. Malacara. 2011. *Color vision and colorimetry: Theory and applications.* Vol. PM204. SPIE. https://doi.org/10.1117/3.881172

J. J. Monaghan. 1992. Smoothed particle hydrodynamics. 30 (Jan. 1992), 543–574. https://doi.org/10.1146/annurev.aa.30.090192.002551

M. Müller, D. Charypar, and M. Gross. 2003. Particle-Based Fluid Simulation for Interactive Applications. (2003), 154–159.

L. V. Schuermann. 2016. mueller-sph. https://github.com/cerrno/mueller-sph. (2016).

H. SharkD. 2010. HSV cylinder. (22 march 2010). https://commons.wikimedia.org/w/index.php?curid=9801673

A. R. Smith. 1978. Color Gamut Transform Pairs. In *Proceedings of the 5th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '78)*. Association for Computing Machinery, New York, NY, USA, 12–19. https://doi.org/10.1145/800248.807361

M. Zwicker, H. Pfister, J. Baar, and M. Gross. 2001. Surface Splatting. *Proceedings of the ACM SIGGRAPH Conference on Computer Graphics* 2001 (08 2001). https://doi.org/10.1145/383259.383300