Physically Based Rendering Techniques to Visualize Thin-Film Smoothed Particle Hydrodynamics Fluid Simulations

Aditya H. Prasad
Dartmouth College, aditya.h.prasad.21@dartmouth.edu

Physically Based Rendering Techniques to Visualize Thin-Film Smoothed Particle Hydrodynamics Fluid Simulations

Aditya Hans Prasad

Department of Computer Science

Dartmouth College

Supervisor

Professor Bo Zhu

In partial fulfillment of the requirements for the degree of

Bachelor of Arts in Computer Science

June 2021

Two bubbles found they had rainbows on their curves. They flickered out saying: ‘It was worth being a bubble, just to have held that rainbow thirty seconds.’

Carl Sandburg

American Poet

Abstract

This thesis introduces a methodology and workflow I developed to visualize smoothed hydrodynamic particle based simulations for the research paper 'Thin-Film Smoothed Particle Hydrodynamics Fluid' (2021), that I co-authored. I introduce a physically based rendering model which allows point cloud simulation data representing thin film fluids and bubbles to be rendered in a photorealistic manner. This includes simulating the optic phenomenon of thin-film interference and rendering the resulting iridescent patterns. The key to the model lies in the implementation of a physically based surface shader that accounts for the interference of infinitely many internally reflected rays in its bidirectional surface scattering function. By simulating the effect of interference on rays reflected off the surface of a thin-film as a component of a surface shader, I am able to obtain photorealistic renderings of bubbles and thin-films. This enables us to visualize complex vortical swirls and turbulent surface flows on oscillating and deforming surfaces in a physically accurate and visually evocative manner.

Contents

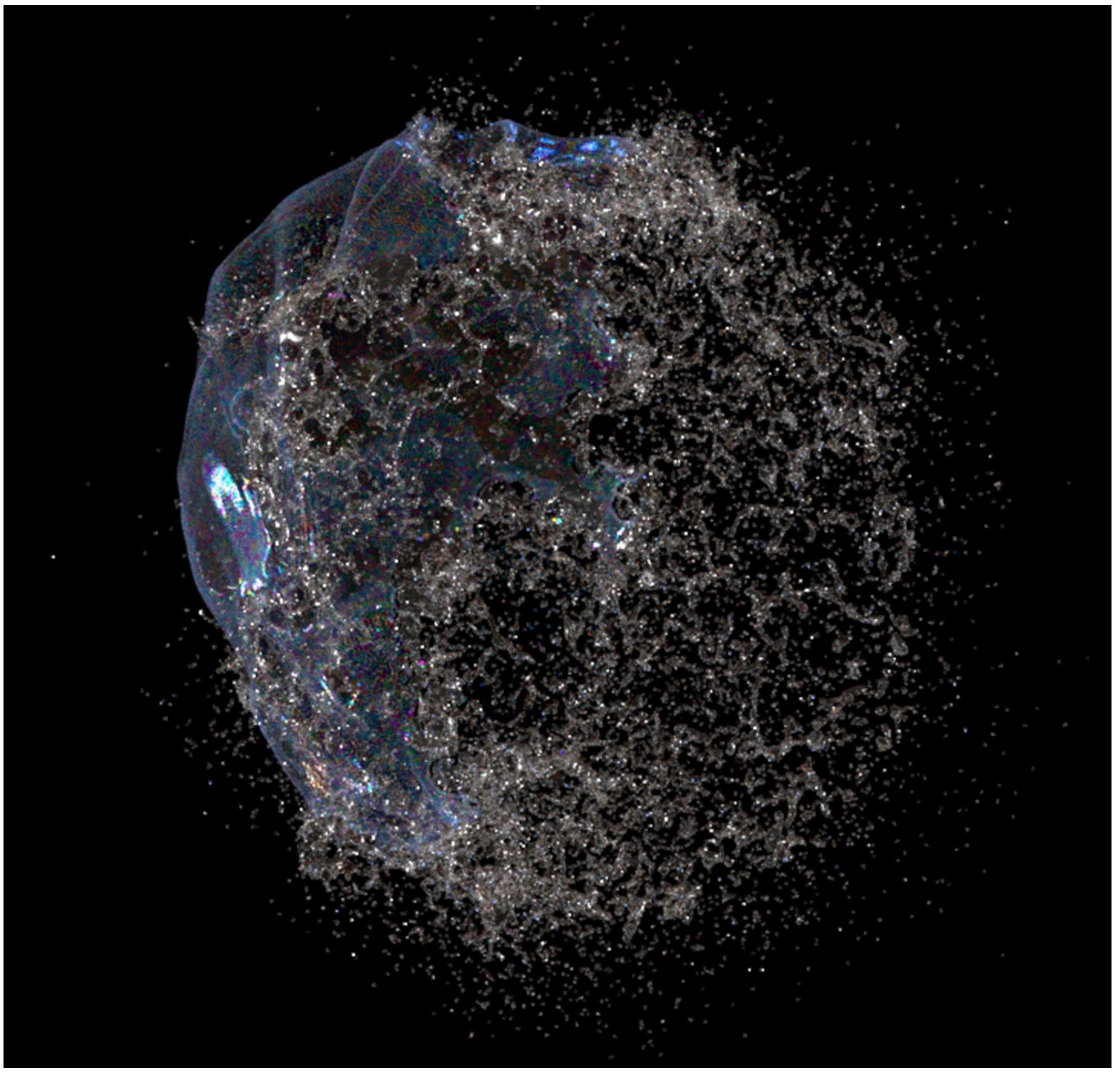

Figure 1 A render of a bursting bubble

Chapter 1

Introduction

1.1 Motivation

Bubbles are among the most beautiful natural phenomena. Their intricate surface flows, vibrant colour pallete, and unique motion make them fascinating to watch. In fact, the beauty of bubbles lies in the laws of physics that govern the way they appear, evolve, and contort. Bubbles are given their distinct shapes and forms due to surface tension. This also causes constant turbulent flows on their surface, which are mesmerizing to watch. The vivid colors we see in bubbles are actually due to an optical phenomenon called tin-film interference. Thin-film interference gives bubbles a unique appearance as it colours the delicate patterns on their surface in a rich palette of hues. The unique, evocative appearance of bubbles has captured human attention for centuries. Throughout human history, artists, painters, photographers, and scientists have made great efforts to recreate the visual beauty of bubbles.

The endeavour to capture the beauty of thin-films has been furthered by the computer graphics community. Research and innovation has brought computer graphics so far that we can accurately recreate physical phenomena, following the principles of math, physics, and nature. Harnessing the power of graphics, researchers are constantly striving to capture more accurately the vast array of phenomena that give bubbles their distinct appearance, and to recreate them computationally. There are many previous works that have made significant contributions to capturing the optical phenomena of thin-film interference [1][2][3].

The methods put forth in these works are seminal in computational approaches to recreating the beauty of bubbles.

In this paper, I present physically based rendering techniques to visualize the phenomena that govern the distinct beauty of bubbles. I visualize significant physical simulations of thin-films developed in our SIGGRAPH paper [4]. More specifically, the methods I propose here allowed the research team to visualize simulation data, represented by point clouds, as photorealistic bubbles.pip i By recreating and following the laws of optics, these methods are able to capture mesmerizing thin-film phenomenon, and create beautiful visualizations of bubble motion, turbulent flow, and various other physical thin-film phenomenon in order to computationally create realistic bubbles. The work done for this thesis is intrinsically linked to the proposed simulation methods in the larger research project [4], but can be utilized to accurately visualize related thin-film phenomenon.

1.2 Background

The goal of this paper is to highlight the rendering techniques and workflow I undertook in rendering simulations for 'Thin-Film Smoothed Particle Hydrodynamics Fluid' [4]. That paper's central goal is to computationally simulate thin-films in a physically accurate manner. As discussed before, the beauty of bubbles lies in the way they move, evolve, and appear on both a micro and macro scale, and our research project aims to capture all three of those aspects.

To recreate the optical phenomena of thin-film interference, I used a physically based rendering approach [5]. There were several reasons for this. Firstly, raytracing is a fast, well developed technique that has been implemented in a variety of different research and commercial software. Secondly, raytracing allows for a great degree of physical accuracy and photo-realism. Lastly, the mathematical principles underlying raytracing allow for many different materials to be represented computationally, in a clean and modular way. In order to recreate the effect of iridescence, I did not need to modify the entire raytracer, but simply write a new material representation of a thin-film. In this representation, I am able to capture the mathematical rules that govern how light interacts with thin-film. Therefore, I found that physically based rendering affords a degree of practical and mathematical freedom that makes it the most viable approach for photorealistic rendering.

1.3 Artistic Process

The process of rendering is just both a scientific process, as well as an artistic one. While the focus of this paper is to highlight the development of specific rendering techniques, a central portion of my research work was the creation and generation of beautiful imagery and videos. As a result, much of my late stage work required that I play the role of an artist by choosing appropriate photography techniques, colour treatments, and lighting in order to get aesthetically pleasing, visually evocative, and physically accurate rendering results. In pursuit of beautiful results, I needed visual references and inspiration in order to ground my approach.

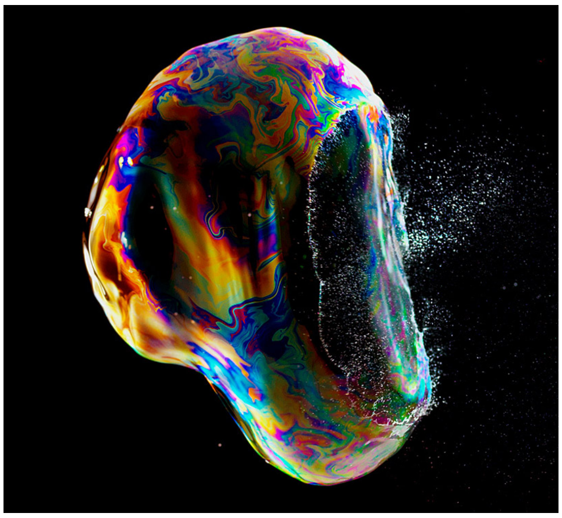

Figure 1.1 A photograph from Fabian Oefner's *Iridient*

A big source of inspiration for me was the work of Fabian Oefner, in particular his series *'Iridient'* [6]. The hyper-saturated, high contrast photographs capture the rich color pallete of iridescence, and highlight the intricate patterns formed by turbulent flows on the surface of bubbles. I set up render scenes to emulate these vibrant pictures, while doing so in a physically accurate manner.

While the above figure served as aesthetic inspiration, the underlying methods in rendering thin-film structures always maintain fidelity to photorealism to the

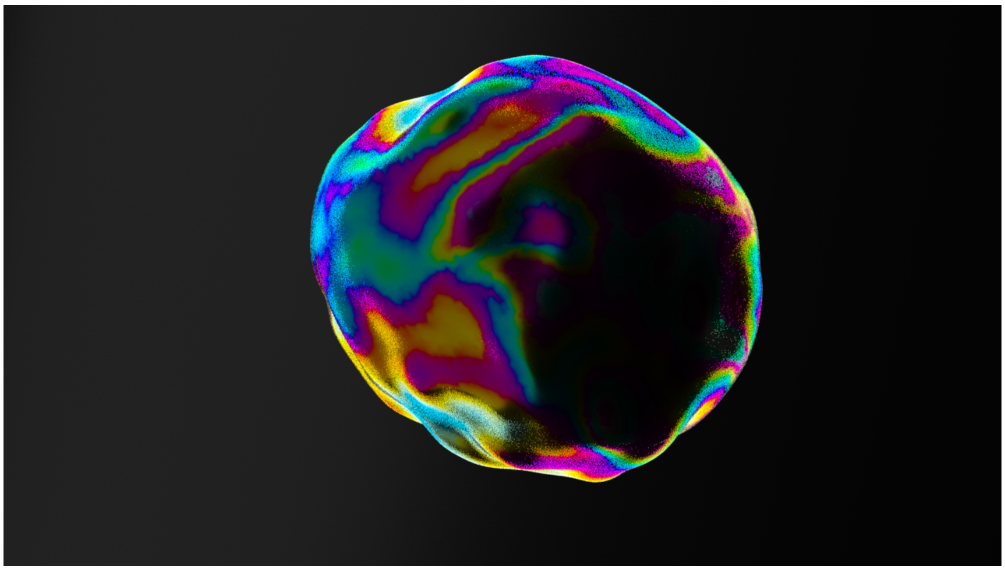

Figure 1.2 A physically based rendering emulating Oefner's style

best capacity of the raytracer.

1.4 Related Works

Mesh Based Dynamic Thin Film

There are seminal works in representing thin-film geometries, particularly area minimizing geometries. Several early works have worked on quick techniques to model the appearance of static bubble geometries [2] and clusters [7]. Other models provide great physical accuracy when rendering thin-film surfaces [8]. There are several works that have modelled in greater detail various aspects of thin-film rendering, such as visualizing non-parallel thin-film interface boundaries [9], or visualizing thin-film sheets and their evolution [10]. A significant paper which couples physics of motion with optical modelling of bubble structures is 'Chemo-mechanical Simulation of Soap Film Flow on Spherical Bubbles' [11]. There are other novel approaches to visualizing the evolution of thickness on the bubble surface [12] as well as the visualization of flows of bubble shape and form [13]. These works exhibit how deeply interlinked the physical simulation and rendering-based visualization aspects are in capturing the rich beauty of bubbles.

Thin Film Interference Surface Shaders

There are several papers highlighting techniques to model thin-film interference computationally. The earliest work to introduce thin-film interference to Computer Graphics is Smits and Meyer's seminal paper [14] to recreate iridescent patterns in image synthesis. Early works model thin-film interference based off the Fresnel Equations [15] [3]. These works do not consider multiple internal reflections, and subsequent work has dealt with these limitations. Seminal developments in computational methods of spectral rendering [16] have also greatly affected the development of thin-film interference simulation methods. There have been techniques developed to calculate thin-film interference with fast approximations, in real time [17]. There are also technical developments in representing wave phenomena, accounting for reflection off isotropic thin-film material [18], which is an apparent limitation in my approach. The defining mathematical approach for the model presented in this thesis is proposed in 'A Practical Extension to Microfacet Theory for the Modeling of Varying Iridescence' [1].

Point Set Thin Film

There are many previous works developing novel techniques for the simulation of level-set and point-set thin films [19]. While this approach converts the simulation point cloud to a mesh surface for rendering purposes, previous works are able to render point-set surfaces [20], and point cloud based SPH simulations [21] with great detail.

Houdini Rendering

There are various useful implementations of thin-film materials accounting for iridescence in several research-oriented and commercial renderers. These have been done in spectral renderers such as Arnold [22], Blender + LuxRenderer [23]. There are also high-quality shader implementations in RGB rendering methods, such as for the Disney BRDF Explorer [24]. Since Houdini's native Mantra renderer is not a spectral renderer, there are relevant approaches to implementing spectral colours in Houdini [25]. There are also implementations of static bubble rendering following Andrew Glassner's bubble rendering techniques [2] in Houdini [26] that produce stunning results, though they are not entirely physically accurate.

Chapter 2

Methods

2.1 Physics Background

Thin-film refers to an extremely thin walled layer. For the purposes of this project, we are focusing on fluid thin-films such as those formed by soap water. Thus, the thin films here have the refractive index of around 1.34 and thickness in magnitude of nanometers. When a ray of light hits such a thin film, it is partially reflected, and partially transmitted through. In fact, some part of the ray undergoes a series of internal reflections that leads to interference in the reflected rays. Figure 2.1 shows this process. The rays R0, R1, R2 constructively and destructively interfere with one another, resulting in the phenomenon of thin-film interference [27].

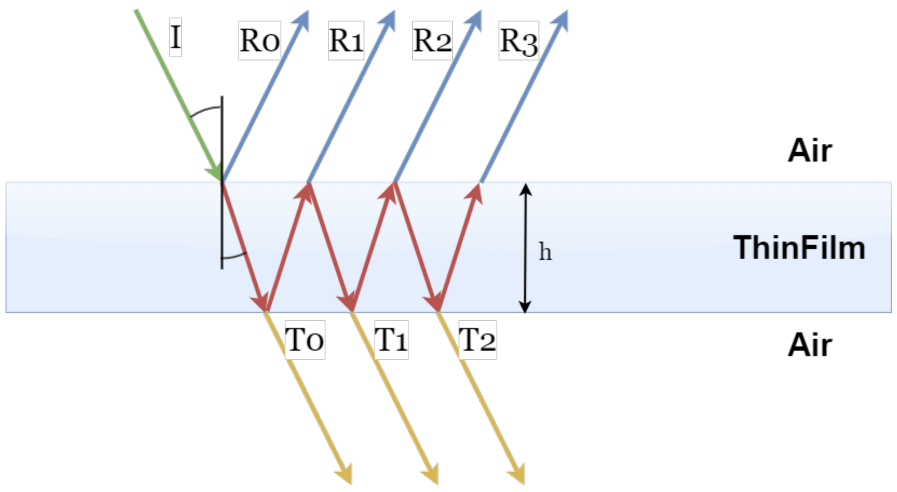

Figure 2.1 Interaction of a ray with air-thin film interface

When considering thin-film interference for a renderer, the goal is to calculate

Symbol	Meaning
$R0, R1, R2$	reflected waves
$T0, T1, T2$	transmitted waves
h	height of soap film
ν_a	refractive index of air = 1.00
ν_s	refractive index of soap film = 1.34
Θ_i	angle of incidence
Θ_r	angle of reflection
D	optical path difference
$\Delta\phi$	phase change
λ	wavelength of ray
R	Reflectance, intensity of reflected wave
T	Transmittance, intensity of transmitted wave
a_r, a_t	amplitude of reflected and transmitted ray respectively
r	complex reflection coefficient
t	complex transmittance coefficient
r_{as}, t_{as}	air-to-soap film interface coefficients
r_{sa}, t_{sa}	soap film-to-air interface coefficients

Table 2.1: A list of symbols used in our thin-film model

the intensities of visible light rays reflected off the thin film, as well as the intensities of transmitted rays. The mathematical approach used in my rendering workflow is modelled on those presented in 'A Practical Extension to Microfacet Theory for the Modeling of Varying Iridescence' [1] and 'Chemomechanical Simulation of Soap Film Flow on Spherical Bubbles' [11].

2.1.1 Derivation of Optical Path Difference

Consider a light ray I travelling towards a thin-film, in air ($\nu_a = 1.00$). The thin-film is represented by a single surface of width h and refractive index $\nu_s = 1.34$. Let the angle of incidence of the ray I on the surface with respect to the surface normal be Θ_i. Following Snell's Law, upon entering the thin-film, the angle of reflectance of the ray I can be calculated by $\sin\Theta_r = \frac{\nu_a \sin\Theta_i}{\nu_s}$. Following Figure 2.1, some waves of the ray reflect back to the air as $R0$. Additionally, some partial energy of the ray travels through the medium as $T0$. Some energy is reflected, and travels back to the medium as $R1$. It is important to note the assumption that on a microscopic scale, the walls of the thin film are parallel. Then, $T0$ exits the thin film into air at precisely the same angle as I entered, Θ_i. Since the film is so thin, the lateral shift is negligible for results, and its effect on the transmitted ray is ignored. Each of the reflected light waves undergo a difference in path length due to reflection within the thin-film. Consider the difference in path length for $R1$ vs $R0$.

For this first order reflection $R1$, the Optical Path Difference D can be calculated by

$$D = 2\nu_s h \cos\Theta_i \tag{2.1}$$

For a kth order reflection, the OPD $= kD$. This Optical Path Difference is what causes a phase shift. Let us denote this phase shift as $\Delta\phi$.

$$\Delta\phi = \frac{2\pi D}{\lambda} \tag{2.2}$$

where λ is the wavelength of the incident wave [1].

2.1.2 Calculation of Intensities of Reflected Ray and Transmitted Ray

Once we have calculated the phase shift $\Delta\phi$, we can calculate the reflectance power R and transmittance power T at the interface of the thin-film. R is the ratio between incoming and outgoing reflected light at the medium of interaction, and T is the ration between incoming and outgoing transmitted light at the medium of interaction.

$$R = \frac{|a_r|^2}{|a_i|^2} = |r|^2 \tag{2.3}$$

where a_r is the amplitude of the reflected wave and a_i is the amplitude of the incident wave. r is denoted as the complex reflection coefficient [1]. Similarly,

$$T = \frac{|a_t|^2}{|a_i|^2} = |t|^2 \tag{2.4}$$

where a_t is the amplitude of the transmitted wave and a_i is the amplitude of the incident wave. t is denoted as the complex transmission coefficient.

Thus, by calculating r and t, we are able to calculate R and T.

r and t are both determined by the phase difference $\Delta\phi$ calculated above, as well as the reflection coefficients r_{as}, r_{sa}, and the transmission coefficients t_{as}, t_{sa} [1]. These coefficients are obtained from the Fresnel Equations.

The reflection coefficient r is calculated by summing up the contributions of all reflected waves [11]:

$$r = |r_{as} + \sum_{k=0}^{\infty} t_{as} r_{sa} (r_{sa}^2 e^{i\Delta\phi})^k e^{i\Delta\phi} t_{sa}| = |r_{as} + \frac{t_{as} r_{sa} t_{sa} e^{i\Delta\phi}}{1 - r_{sa}^2 e^{i\Delta\phi}}| \tag{2.5}$$

Similarly, the transmission coefficient t is calculated by summing up the contributions of all transmitted waves [11]:

$$t = |\sum_{k=0}^{\infty} t_{as} (r_{sa}^2 e^{i\Delta\phi})^k t_{sa}| = |\frac{t_{as} t_{sa}}{1 - r_{sa}^2 e^{i\Delta\phi}}| \tag{2.6}$$

It is important to note that we calculate $\Delta\phi$ as a function of λ, the wavelength of the incident wave. Therefore, R and T are also calculated dependent on λ.

2.1.3 Spectral Sampling and Integration over Ray Energy

Therefore, to calculate the spectral energy of the incident ray I, we must calculate R and T for every wave constituting the ray. In practice, this is done by sampling the incident ray for a certain number of wavelengths determined by the parameter n. n is a parameter inherent to raytracers which determines the number of spectral samples that define a ray. By calculating R and T as a function of λ, for every λ in the spectral samples, we are able to calculate the spectral representation of the reflected ray and the transmitted ray based off the incident ray.

2.1.4 The Fresnel Equations

The Fresnel Equations are key in the calculation of thin film interference, since they define the aforementioned coefficients, r_{as}, r_{sa}, t_{as}, and t_{sa}. These equations essentially govern how much light is reflected at a given interface, which is dependent on the angle of incidence of the light ray and the respective refractive indices of the materials in the interface [27]. It is important to note that the reflection and transmission coefficients obtained from Fresnel's equations depend on the polarization components of the incident waves. However, most raytracers assume that the distribution of s and p polarized waves are equal in a light ray unless specified otherwise. In this model, we follow that assumption. As a result, we can simply average the s and p polarized coefficient components obtained from the Fresnel equations in order to obtain our reflection and transmission coefficients [11].

2.2 Implementation of Surface Shader

The crux of implementing the above model to visualize thin-film optical phenomena no doubt lies in the choice of renderer. I implemented the above model as a surface shader in three different rendering software: PBRT [5] , Mitsuba Renderer [28], and finally Houdini's Mantra Renderer. Each implementation differs slightly due to software specific parameters, but the underlying model remains the same.

For the purpose of rendering results for my team's research paper [4], I ultimately chose to work with Houdini and it's built-in renderer, Mantra. This is because it provides a straightforward workflow, preexisting methods to handle geometry and lighting, and extensive technical and artistic power.

2.2.1 Bidirectional Scattering Distribution Function

The key implementation of the mathematical model lies in a function known as the bidirectional scattering distribution function (BSDF). The BSDF is one of the principle functions of a physically based surface shader, and is associated to a specific material. The BSDF calculates the probability that a single ray of light will reflect, for a given material and given angle of incidence. Essentially, the BSDF calculates how much light reflects off the interface, and how much light transmits through. In fact, I specifically had to implement a Bidirectional Reflection Distribution Function (BRDF), a specification of the BSDF which calculates how much light reflects off the surface of the surface it is applied to [29].

The reflection of rays off the thin-film is perfectly specular since the surface

Algorithm 1 Thin-Film Surface Shader Ray Intersection Function

```
Input: $\Theta_i$, $\nu_a$, $\nu_s$, $h$, $I$
Output: $R$, intensity of reflected ray and angle of reflection
    if $\cos\Theta_i \leq 0$ then
        swap $\nu_a$ and $\nu_s$
    end if
    $\Theta_r \leftarrow \arcsin\frac{\nu_a \sin\Theta_i}{\nu_s}$
    $D \leftarrow 2\nu_s h \cos\Theta_i$
    for $\lambda$ in SpectralSamples do
        $\Delta\phi \leftarrow \frac{2\pi D}{\lambda}$
        $r_{as}^{||} \leftarrow \frac{\nu_a \cos\Theta_r - \nu_s * \cos\Theta_i}{\nu_a \cos\Theta_r + \nu_s \cos\Theta_i}$
        $r_{as}^{\perp} \leftarrow \frac{\nu_a \cos\Theta_i - \nu_s \cos\Theta_r}{\nu_a \cos\Theta_i + \nu_s \cos\Theta_r}$
        $t_{as}^{||} \leftarrow \frac{2\nu_a \cos\Theta_i}{\nu_a \cos\Theta_r + \nu_s \cos\Theta_i}$
        $t_{as}^{\perp} \leftarrow \frac{2\nu_a \cos\Theta_i}{\nu_s \cos\Theta_r + \nu_a \cos\Theta_i}$
        $r_{sa}^{||} \leftarrow \frac{\nu_s \cos\Theta_i - \nu_a \cos\Theta_r}{\nu_s \cos\Theta_i + \nu_a \cos\Theta_r}$
        $r_{sa}^{\perp} \leftarrow \frac{\nu_s \cos\Theta_r - \nu_a \cos\Theta_i}{\nu_s \cos\Theta_r + \nu_a \cos\Theta_i}$
        $t_{sa}^{||} \leftarrow \frac{2\nu_s \cos\Theta_r}{\nu_s \cos\Theta_i + \nu_a \cos\Theta_r}$
        $t_{sa}^{\perp} \leftarrow \frac{2\nu_s \cos\Theta_r}{\nu_s \cos\Theta_r + \nu_a \cos\Theta_i}$
        $r_{as} = \frac{r_{as}^{\perp} + r_{as}^{||}}{2}$
        $t_{as} = \frac{t_{as}^{\perp} + t_{as}^{||}}{2}$
        $r_{sa} = \frac{r_{sa}^{\perp} + r_{sa}^{||}}{2}$
        $t_{sa} = \frac{t_{sa}^{\perp} + t_{sa}^{||}}{2}$
        $r = |r_{as} + \frac{t_{as} r_{sa} t_{sa} e^{i\Delta\phi}}{1 - r_{sa}^2 e^{i\Delta\phi}}|$
        $t = |\frac{t_{as} t_{sa}}{1 - r_{sa}^2 e^{i\Delta\phi}}|$
        $R[\lambda] = r^2$
        $T[\lambda] = t^2$
    end for
    Angle of reflection = specular reflect($\Theta_i$)
    return $R, T$, angle of reflection
```

of the bubble is smooth. Thus, in my implementation, I do not anticipate any scattering of light. As a result, the angle of reflection of any reflected ray is equivalent to the angle of its incidence [30]. Then, the BSDF simply calculates the phase difference, the reflection coefficients, and the reflected light intensity is calculated based off the guiding Equation 2.5.

2.2.2 Sampling Wavelengths to Calculate Reflected Ray's Intensity

However, the above Equations 2.5 and 2.6 calculate the reflected and transmitted intensities of only one wave, with a single associated wavelength λ. To calculate the reflected intensity of an incident ray, we must calculate the intensities for all waves that constitute it. Therefore the implementation must sum over the reflected intensities over the spectrum of visible waves in order to account for all wavelengths in the ray.

In practice it is not possible to account for every single wave, so we choose to sample a given number of waves ranging from wavelengths of $300nm$ to $800nm$, at a fixed increment. This range is a variable, and the upper and lower bounds can be tweaked. Thus, the parameter $nSpectralSamples$ that defines how many samples it takes to represent the spectrum of a ray is very important. While the renderers PBRT and Mitsuba provide this value as a global variable based of their implementation, Houdini did not have a universal parameter for this as it is not a spectrally based renderer. Thus, the parameter $nSpectralSamples$ is a user defined parameter in our implementation. 8 or 16 works best; anything greater is a bit excessive.

Then, the reflected intensities are calculated for each wave whose associated wavelength we choose to sample, and we construct the reflected ray's spectral energy curve by compounding the reflected intensity at each sampled wavelength.

2.2.3 Conversion of Spectral Energy to Colour

For the representation of spectral energy to be visualized as colour, the resultant light intensities for all wavelengths are converted to a single RGB color value by integrating with the CIE matching functions. In practice, this integration leads the resulting colours to veer from the ground truth. This model assumes such errors to be a limitation of RGB rendering engines [1].

2.3 Bubble Geometry

The modelled surface shader can be applied to any mesh geometry. However, a significant aspect of the research paper [4] that this approach contributes to is that the Smoothed Particle Hydrodynamics based simulations produce data that is represented and contained within a point cloud. To convert a point cloud to a mesh surface, I followed the basic workflow as described in 'From Point Cloud to Durface: The Modeling and Visualization Problem' [31] that entails pre-processing, global and localized topology determination, polygon mesh surface generation, and post-processing. These methods have been encapsulated and packaged into several existing Houdini functionality.

It is important to convert the point cloud data into mesh geometries for the purpose of rendering by the above methods.

The conversion of point cloud to mesh is in Houdini, largely due to the pre-existing functionality to represent volume fields as *VDBs* [32], which allow for ease of importing point cloud data into a data format that is convertible to mesh. *VDBs* are voxel-based data structures, and are memory efficient in representing both sparse and dense volume fields [32].

The pipeline for this conversion is as follows:

1. Import point data into Houdini, with points placed at (x, y, z) coordinate in 3d space.

   Along with the position vector of the points, we also import the height-field parameter rh, as well as the direction of the surface normal N. rh contains the thickness of the thin-film at the given point.

2. Convert point data cloud to Houdini VDB.

   This step is key in representing the point cloud as a fluid. On conversion, the points are treated as 'droplets', and neighbouring points within

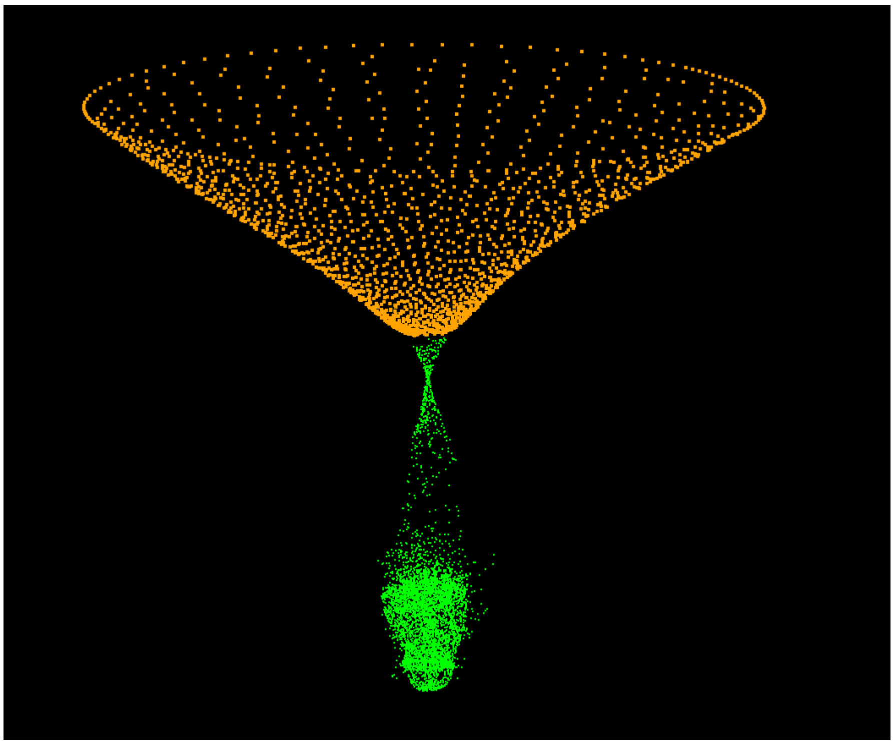

Figure 2.2 Point Cloud of a Dripping thin-film

the radius defined by parameter $InfluenceRadius$ are joined to form a fluid volume.

3. Convert Fluid VDB to a polygon mesh.

   Here it is important to implement re-sampling of the vertices, and transfer rh and N from the original point cloud to the newly sampled vertices of polygons. The mesh surfaces interpolate the transferred data based on the closest of the original points in the point cloud. Thus, the resolution of the mesh is key to making sure that no great losses are incurred by interpolating from too far a point. Otherwise, that will lead to artifacts.

This pipeline converts a point cloud to a infinitesimally thin mesh that emulates fluid, simulating the effects of 'droplets'. The thickness of the fluid is contained within the imported vertex parameter, rh. It is very important to tune the parameters as tight as possible, otherwise artifacts will be present in the mesh representation. The tuning of the governing parameters is different for each input since it depends on the sparseness of the input cloud, as well as the size of the cloud, maximal distances between neighbouring points, and also whether there are any holes present in the structure.

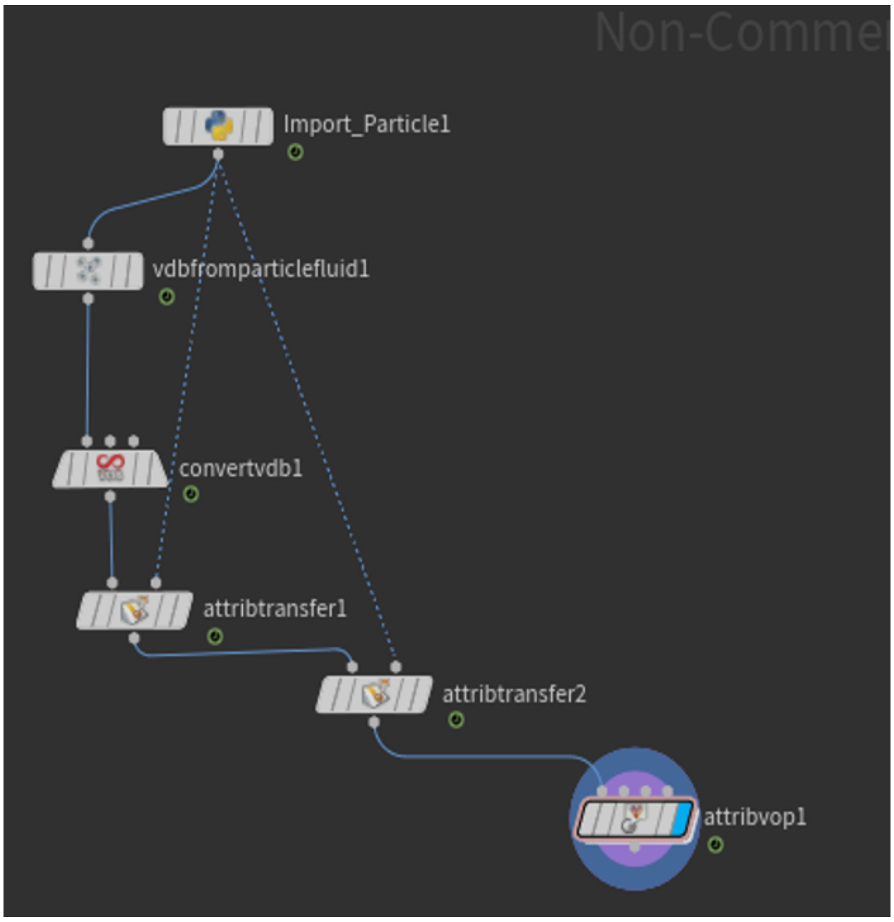

Figure 2.3 Pipeline of Fluid Surface Mesh Generation in Houdini

2.3.1 Colour based on Varying Thickness

The colours of the reflected rays are dependent on the height of the thin-film at the point of incidence. In the real world, thin-films do not have a uniform thickness. Instead, the thickness of the thin-film varies. This variation is what leads to the formation of intricate patterns and turbulent surface flows, as the varying thickness evolves and develops due to the action of surface tension, among other forces. Thus, for this model to be able to visualize surface flows, there must be a way to map variations in thickness to the mesh.

The simulation data produced by the SPH simulation method highlighted in our paper contains heightfield values rh for every point in the resultant point cloud. In order to transfer this heightfield value rh from point cloud to mesh, we import rh as a vertex attribute in Houdini. We then interpolate the attribute from the closest vertex to every point where a ray collides with the resultant mesh. This introduction of variation in height per vertex leads to the visualization of surface turbulent flows in physically accurate manner. However, as a result the parameters controlling the interpolation and resolution of the mesh conversion must be tuned very finely to avoid artifacts.

2.4 Houdini Workflow and Rendering

For the purpose of rendering the thin-film simulations, I chose to work with Houdini and its native Mantra renderer. Using all the previous discussed methods, I used the following workflow in Houdini:

1. Import point cloud and convert into mesh
2. Apply bubble material on mesh
3. Setup lighting, camera, and scene geometries.

   Tweak lighting to bring out the specific aspects that the render aims to capture.
4. Setup raytracer and render nodes.

   This includes fine tuning camera parameters such as focus, exposure, contrast, saturation etc.
5. Render animation and export.

2.5 Particle Visualization

For the purposes of scientific visualization, my collaborators Yitong Deng, Xiangxin Kong, and I also worked on a surface shader that visualizes the thin film simulation data as particles. This is particularly valuable in visualizing the sparseness of the point cloud, the surface turbulence, and keeping track of how individual points in the point cloud move. This is also useful in catching artifacts caused by the point cloud to mesh conversion method. Finally, we resort to using this method for simulations where the point cloud to surface mesh conversion method cannot perform, largely due to sparseness of points causing holes, or temporal artifacts in animation as a result of an inability to properly tune the fluid surface mesh forming parameters.

2.5.1 Rendering the Point Cloud as Geometry

Either of the following two geometries are used to visualize points in the cloud:

1. 3D spheres.

   This method is valuable to show the action and motion of individual points in the point cloud.

2. 2D circles lying orthogonal to the surface normal N.

   This method is especially valuable to detail intricate flow patterns.

Chapter 3

Results

The following still images were rendered using the devised model, implemented in Houdini with Mantra renderer.

3.1 Irregular Bubble

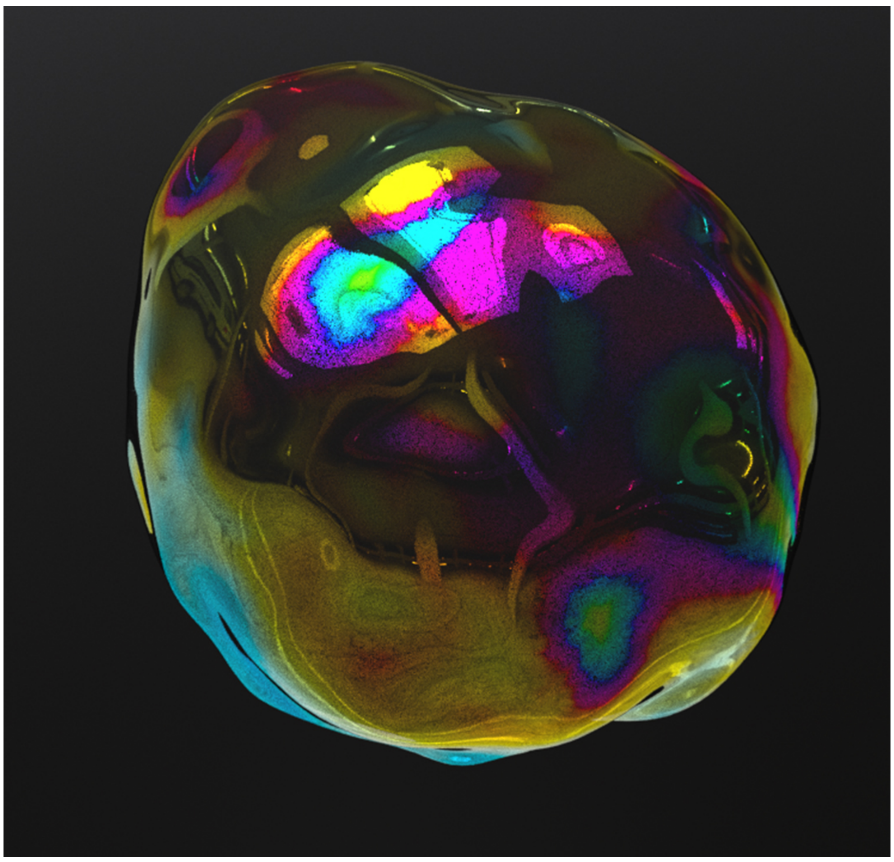

Figure 3.1 An irregular bubble

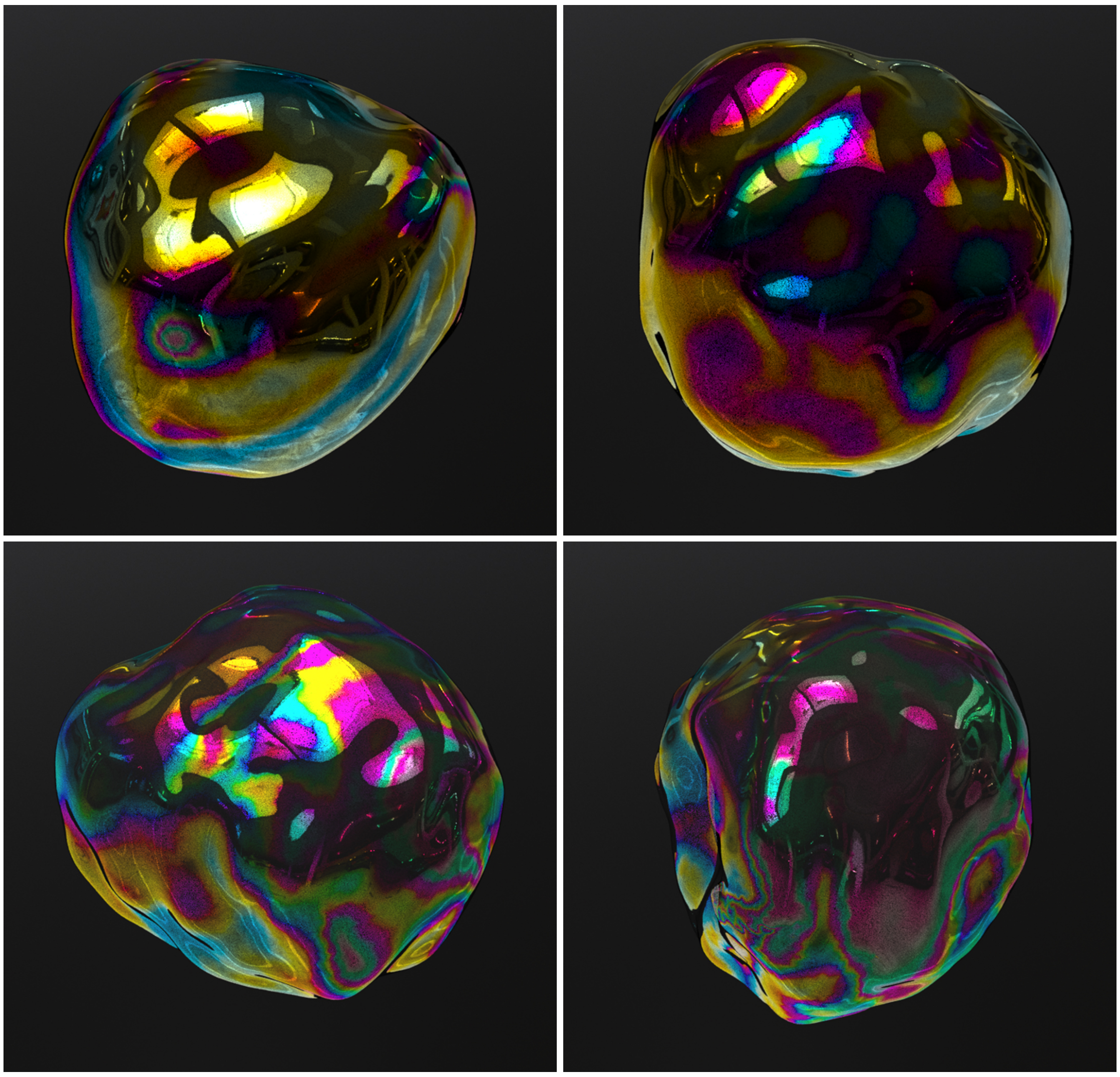

Figure 3.2 The progression of surface flow on an irregular bubble

3.2 Catenoid

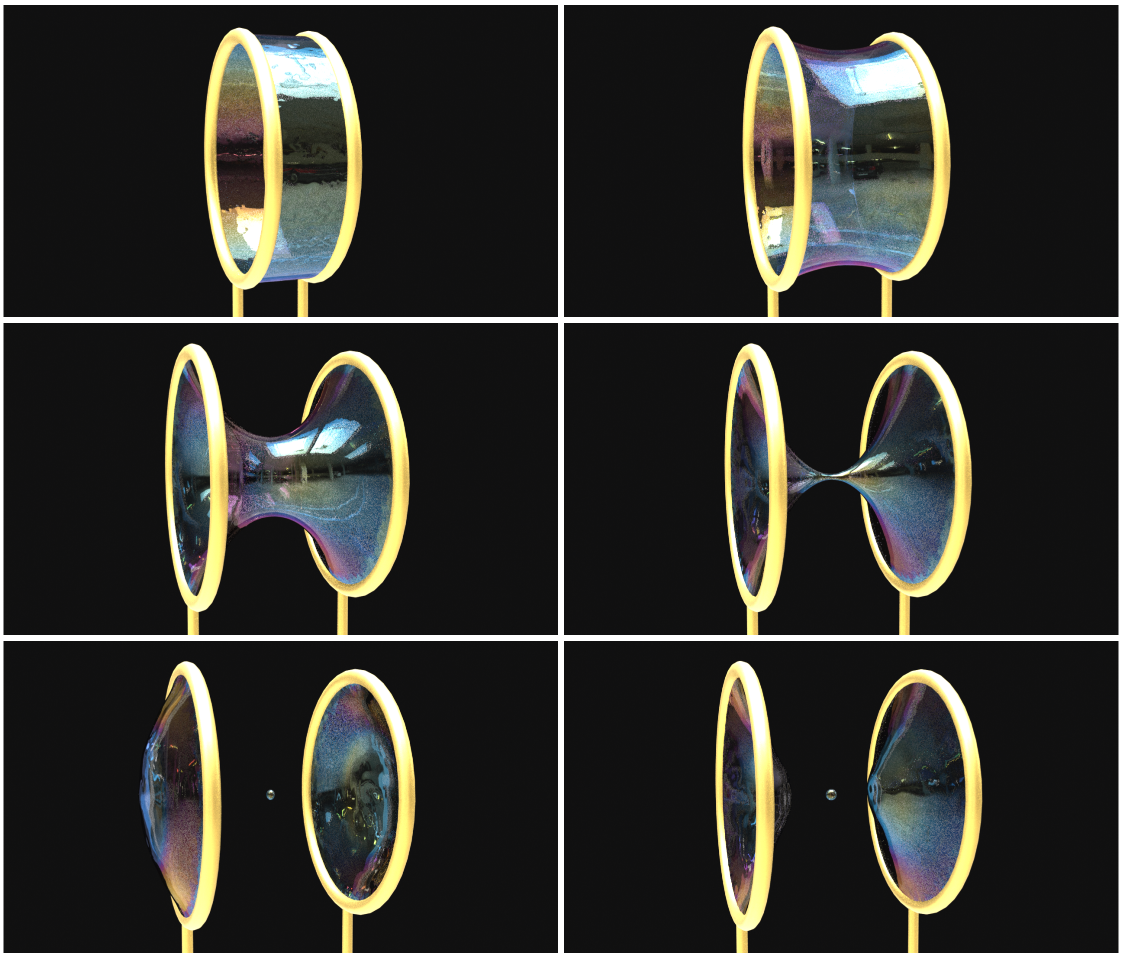

Figure 3.3 Two parallel rings connected by a thin film are pulled apart

3.3 Half bubble

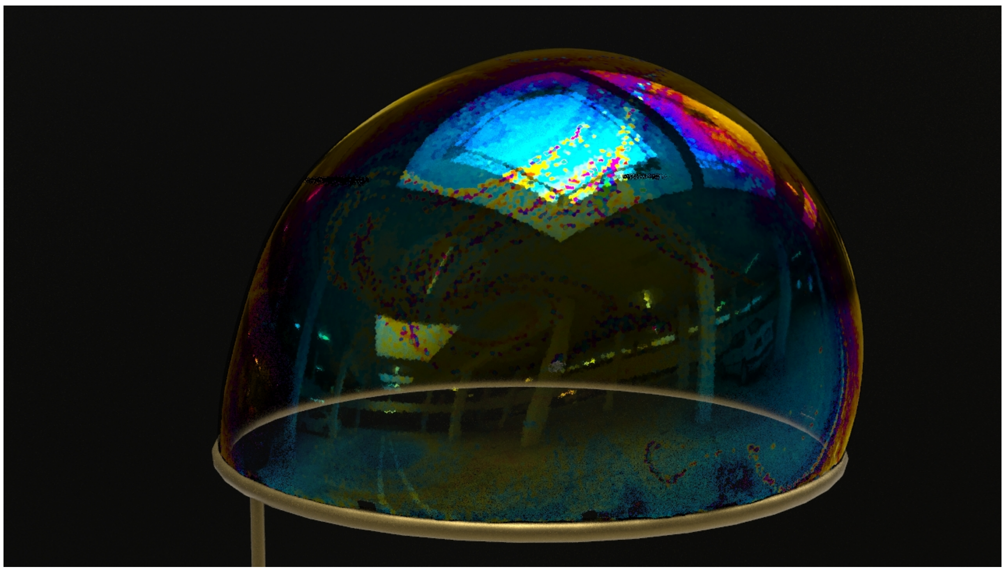

Figure 3.4 Half bubble rendered with environment light and mesh surface

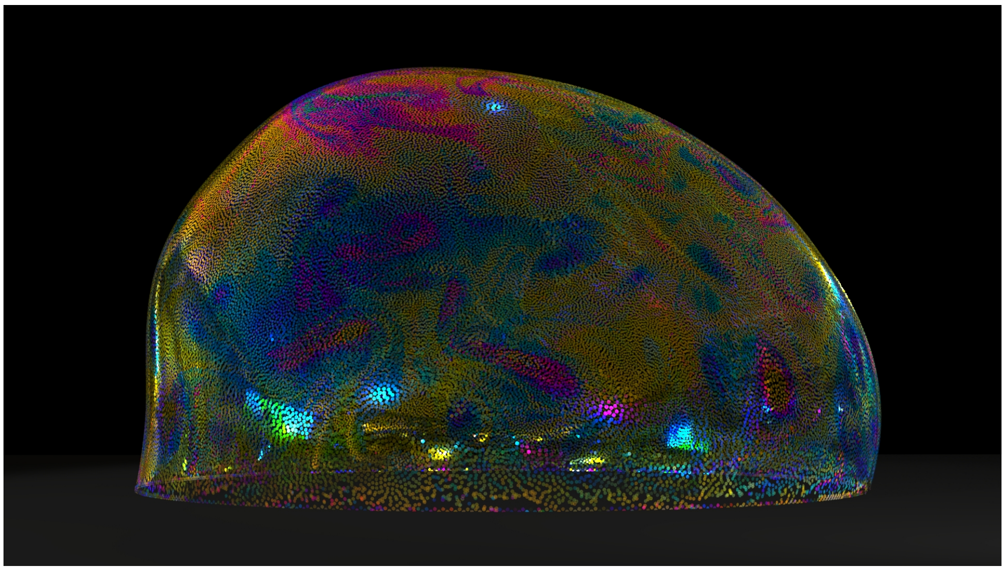

Figure 3.5 Half bubble rendered with area light and circle based surface

3.4 Dripping

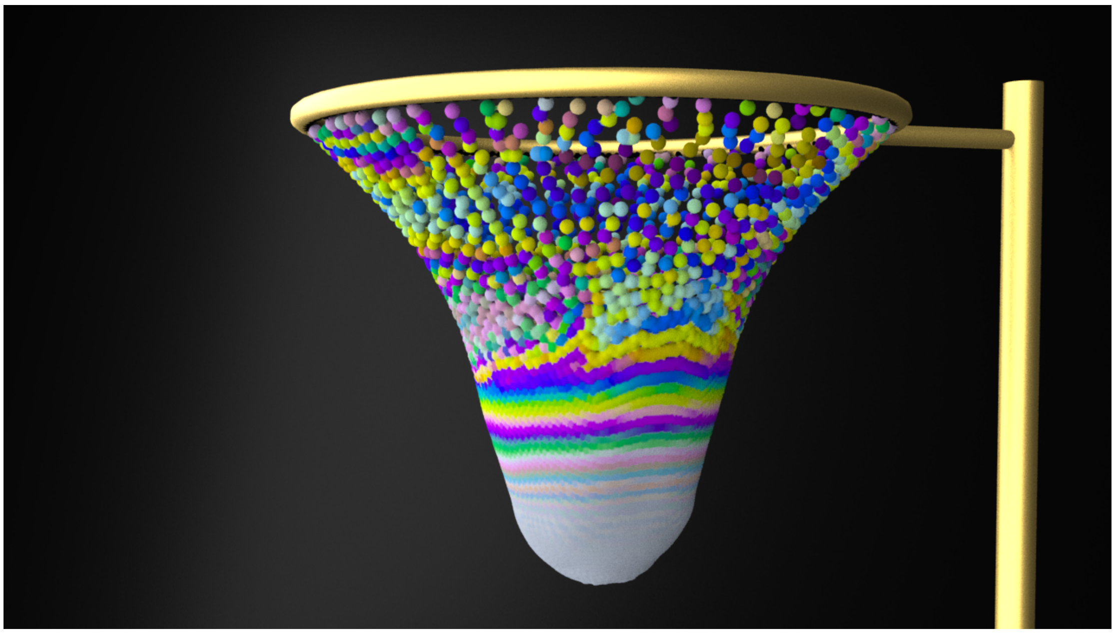

Figure 3.6 Drip rendered with spherical particle visualization

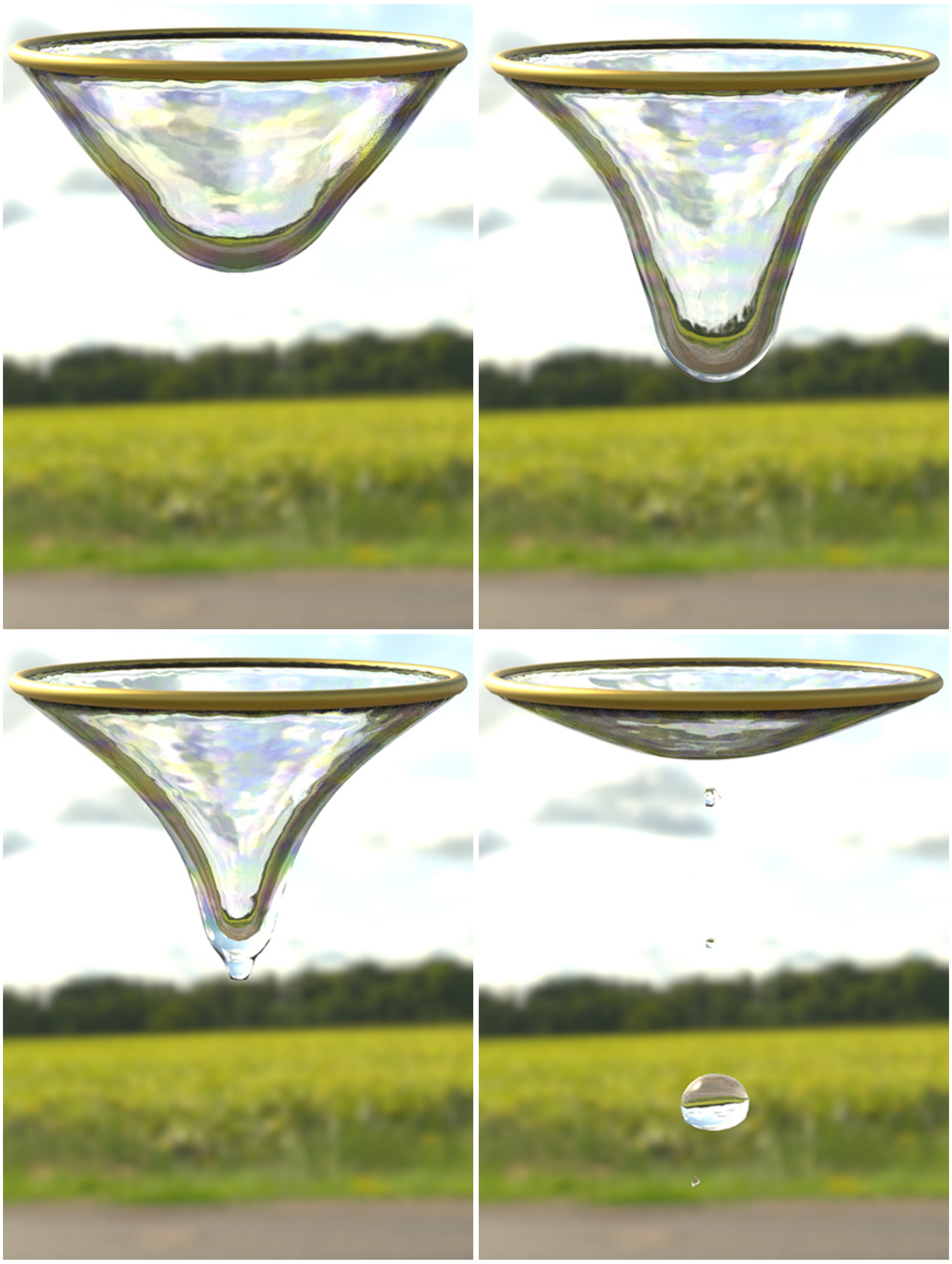

Figure 3.7 A thin film drips, and droplets separate from the film

3.5 Film

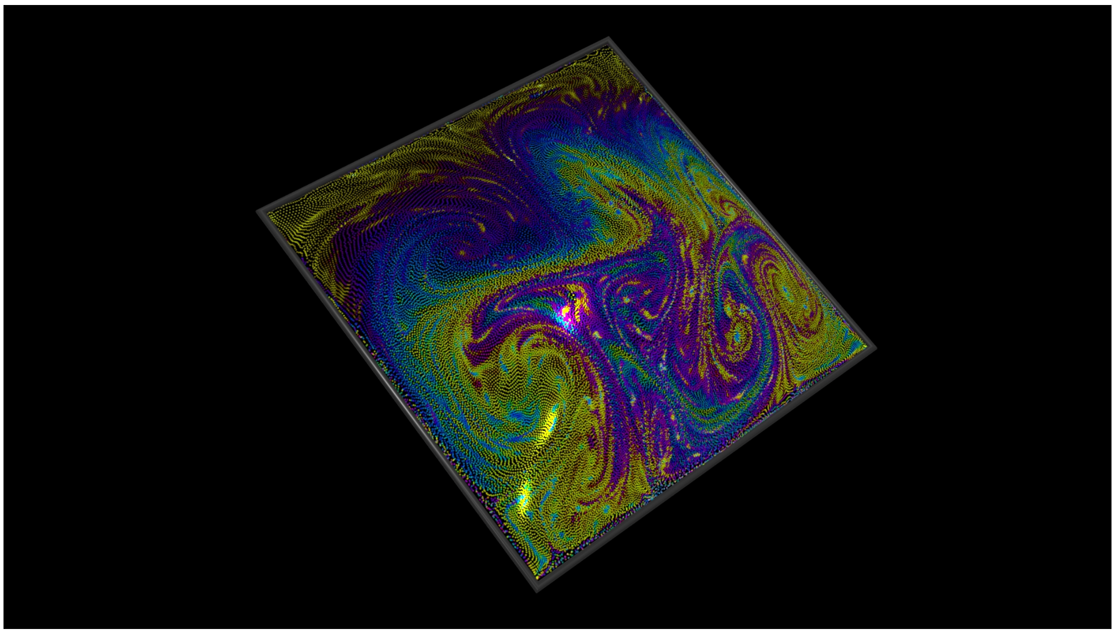

Figure 3.8 Square thin film

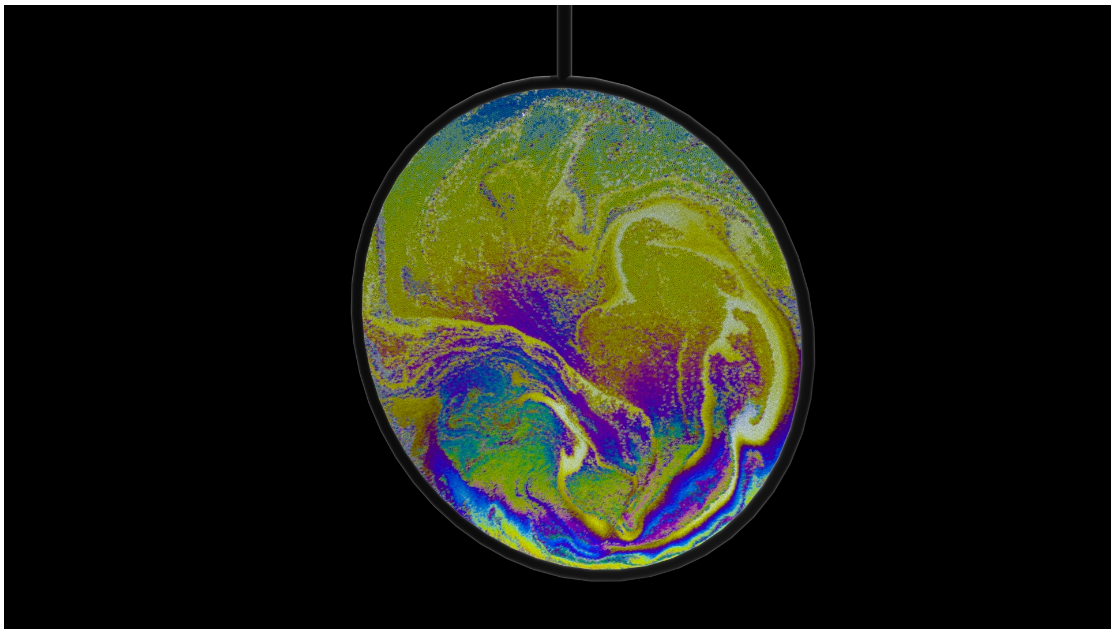

Figure 3.9 Circle thin film

3.6 Bubble Oscillation

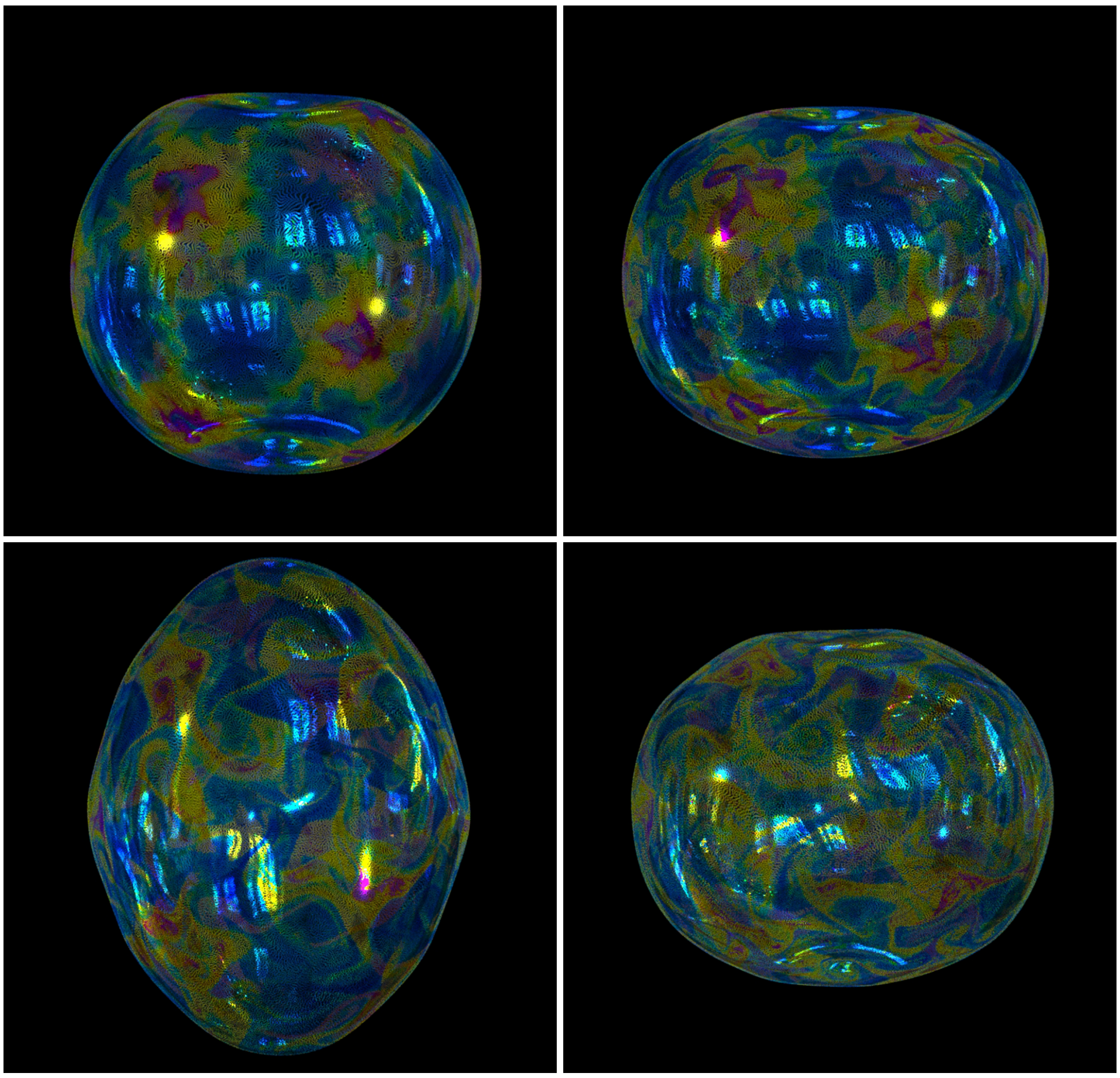

Figure 3.10 A large bubble oscillates

3.7 Limitations

There are definitely some difficulties when rendering thin films in a photorealistic way. I found that the surface mesh conversion method is very particular, and needed to be finely tuned for each instance of the input data. If not tuned, the renders were prone to a few different types of artifacts. The most prominent were temporal inconsistencies, where same points on the bubble surface would be coloured vastly different in adjacent frames, especially if the input point cloud was sparse. These inconsistencies make the surface appear jittery in rendered videos.

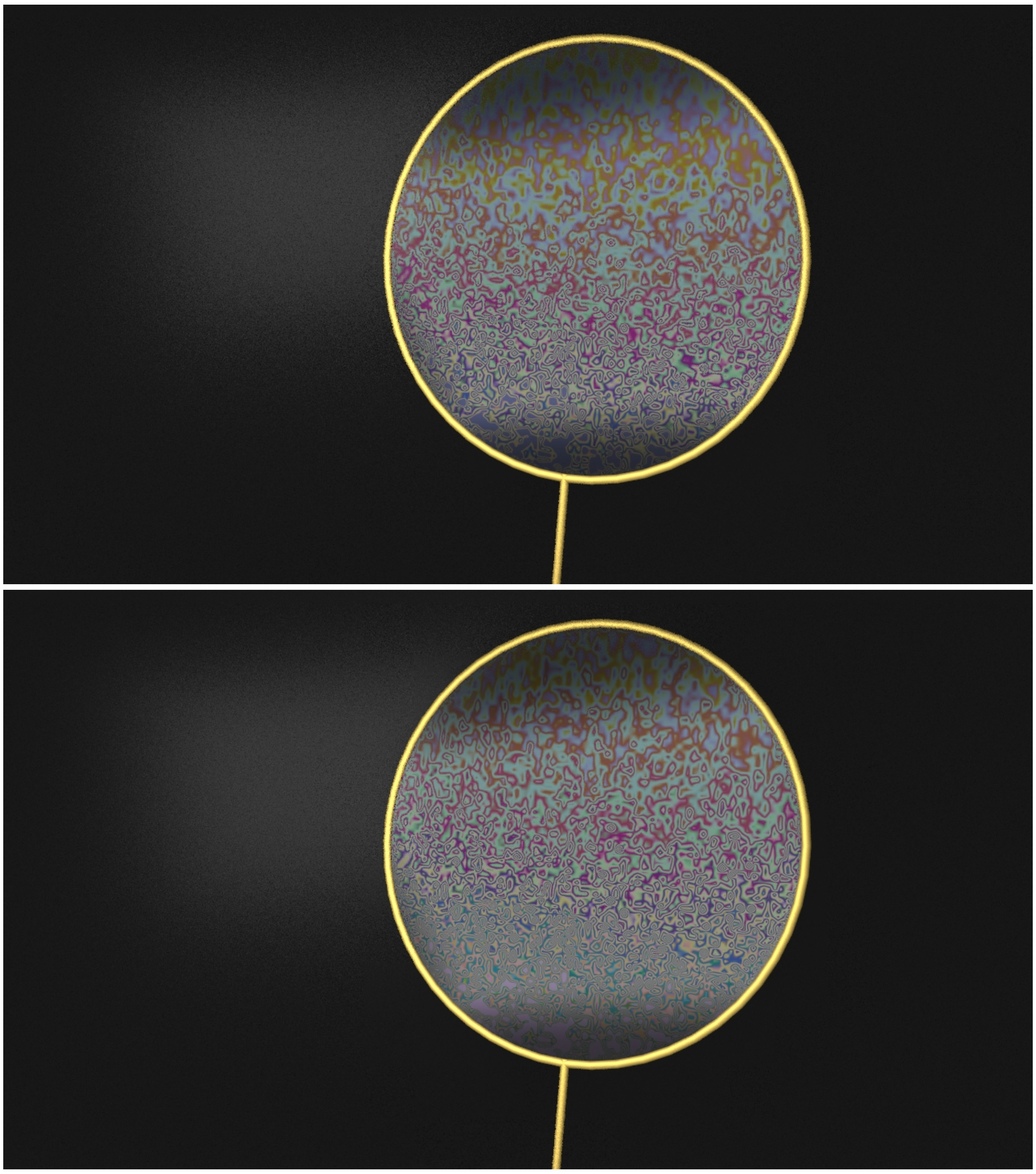

Figure 3.11 The same frame tuned with different mesh conversion parameters

As with any image render where there are many reflections, the process of rendering is quite expensive and time intensive. Thus, the renderer needs to be finely tuned to balance time, noise, and quality. As expected, the performance of the render varies significantly depending on the number of lights. For single area light setups, the method is quick. However for setups with multiple lights it can be quite expensive. Special care needs to be paid with regards to maximum reflection limit of the raytracer. The number of samples per pixel also have to be tuned finely to ensure minimization of noise, especially as the number of internal reflections increase.

Aside from limitations with the general render process, there is room for improvement in the mathematical model itself. The mathematical model doesn't capture the phenomenon of black spots that evolve on the surface of thin-film structures [33]. Black spots appear when the film is extremely thin, and additionally varies on the concentration of soap within the film. In fact, this visualization model doesn't account for the concentration of soap, and assumes a constant refractive index across the thin-film wall.

In my approach, I also assume equal polarisation of light. The mathematical model doesn't account for isotropic thin-film material. There are seminal approaches to modelling this [18], and the mathematical model could be modified to account for this when modelling gasoline and similar materials.

This model also assumes that the thin-film interface has perfectly parallel surfaces. Our mesh conversion model outputs an infinitesimally thin surface with a heightfield applied onto it. Thus we do not account for small angular changes due to the surface heightfield gradient. Expanding the model to allow for this would give us more physically accurate, albeit expensive results [9]. Finally, while our model is able to capture thin-film interference, it is unable to account for other wave phenomena such as diffraction. Perhaps expanding the model to account for other wave phenomena could provide even more realistic results [34].

Acknowledgements

I would like to express my greatest thanks to Professor Bo Zhu, who introduced me to fluid simulations and rendering research at the beginning of my undergraduate career. If not for his guidance, I would have never been able to explore the wonderful world of fluid simulations and rendering in such a hands-on manner. His encouragement and mentorship provided me the opportunity to contribute as a co-author to the paper 'Thin-Film Smoothed Particle Hydrodynamics Fluid' [4], which is one of my proudest academic achievements.

I am immensely grateful for the collaboration and support of Yitong Deng, Mengdi Wang, Xiangxin Kong, and Shiying Xiong, who invited me into their research team and allowed me to contribute to a valuable research project.

Finally, I would like to thank all my friends and family, who took an interest in my work and helped me realize many of my academic goals.

References

[1] L. Belcour and P. Barla, "A practical extension to microfacet theory for the modeling of varying iridescence", *ACM Transactions on Graphics*, vol. 36, no. 4, Jul. 2017, ISSN: 0730-0301, 1557-7368. DOI: 10.1145/3072959.3073620.

[2] A. Glassner, "Soap bubbles. 2 [computer graphics]", *IEEE Computer Graphics and Applications*, vol. 20, no. 6, 2000. DOI: 10.1109/38.888023.

[3] F. Almgren and J. Sullivan, "Visualization of soap bubble geometries", *Leonardo*, vol. 25, no. 3/4, 1992, ISSN: 0024094X. DOI: 10.2307/1575849.

[4] M. Wang, Y. Deng, X. Kong, A. H. Prasad, S. Xiong, and B. Zhu, "Thin-film smoothed particle hydrodynamics fluid", 2021. eprint: arXiv:2105.07656.

[5] M. Pharr, W. Jakob, and G. Humphreys, *Physically based rendering: from theory to implementation*, Third edition. Morgan Kaufmann Publishers/Elsevier, 2017, ISBN: 9780128006450.

[6] F. Oefner, *Iridient*, 2012.

[7] R. Ďurikovič, "Animation of soap bubble dynamics, cluster formation and collision", *Comput. Graph. Forum*, vol. 20, Sep. 2001. DOI: 10.1111/1467-8659.00499.

[8] D. Jaszkowski and J. Rzeszut, "Interference colours of soap bubbles", *The Visual Computer*, vol. 19, no. 4, 2003. DOI: 10.1007/s00371-002-0195-6.

[9] X. Granier and W. Heidrich, "A simple layered rgb brdf model", vol. 65, no. 4, pp. 171–184, Jul. 2003, ISSN: 1524-0703. DOI: 10.1016/S1524-0703(03)00042-0.

[10] F. Da, C. Batty, C. Wojtan, and E. Grinspun, “Double bubbles sans toil and trouble: Discrete circulation-preserving vortex sheets for soap films and foams”, *ACM Trans. on Graphics (SIGGRAPH 2015)*, 2015.

[11] W. Huang, J. Iseringhausen, T. Kneiphof, Z. Qu, C. Jiang, and M. B. Hullin, “Chemomechanical simulation of soap film flow on spherical bubbles”, *ACM Transactions on Graphics*, vol. 39, no. 4, Jul. 2020, ISSN: 0730-0301, 1557-7368. DOI: 10.1145/3386569.3392094.

[12] S. Ishida, P. Synak, F. Narita, T. Hachisuka, and C. Wojtan, “A model for soap film dynamics with evolving thickness”, *ACM Trans. Graph.*, vol. 39, no. 4, Jul. 2020, ISSN: 0730-0301. DOI: 10.1145/3386569.3392405.

[13] S. Ishida, M. Yamamoto, R. Ando, and T. Hachisuka, “A hyperbolic geometric flow for evolving films and foams”, *ACM Trans. Graph.*, vol. 36, no. 6, November 2017, ISSN: 0730-0301. DOI: 10.1145/3130800.3130835.

[14] B. E. Smits and G. W. Meyer, “Newton’s colors: Simulating interference phenomena in realistic image synthesis”, K. Bouatouch and C. Bouville, Eds., pp. 185–194, 1992. DOI: 10.1007/978-3-662-09287-3_13.

[15] M. Dias, “Ray tracing interference color”, *IEEE Computer Graphics and Applications*, vol. 11, no. 2, pp. 54–60, 1991. DOI: 10.1109/38.75591.

[16] R. Ďurikovič and R. Kimura, “Spectrum-based rendering using programmable graphics hardware”, SCCG ’05, pp. 233–236, 2005. DOI: 10.1145/1090122.1090161.

[17] K. Iwasaki, K. Matsuzawa, and T. Nishita, “Real-time rendering of soap bubbles taking into account light interference”, pp. 344–348, 2004. DOI: 10.1109/CGI.2004.1309231.

[18] I. Icart and D. Arquès, “An illumination model for a system of isotropic substrate- isotropic thin film with identical rough boundaries”, D. Lischinski and G. W. Larson, Eds., pp. 261–272, 1999.

[19] W. Zheng, J. Yong, and J. Paul, “Simulation of bubbles”, *Graph. Model.*, vol. 71, pp. 229–239, 2006.

[20] A. Adamson and M. Alexa, "Ray tracing point set surfaces", pp. 272–282, 299, January 2003.

[21] X. Xiao, S. Zhang, and X. Yang, "Fast, high-quality rendering of liquids generated using large scale sph simulation", *Journal of Computer Graphics Techniques (JCGT)*, vol. 7, no. 1, pp. 17–39, March 2018, ISSN: 2331-7418.

[22] Mograph, *104. creating all kinds of bubbles using thin film in arnold for maya*, May 2018. [Online]. Available: `http://mographplus.com/104-creating-all-kinds-of-bubbles-using-thin-film-in-arnold-for-maya/`.

[23] J. Walter, *Making bubbles with cycles and luxrender*, March 2014. [Online]. Available: `https://www.janwalter.org/jekyll/rendering/cycles/2014/03/28/making-bubbles-cycles-vs-luxrender.html`.

[24] Bacterius, *Thin film interference for computer graphics*, April 2013. [Online]. Available: `https://www.gamedev.net/tutorials/_/technical/graphics-programming-and-theory/thin-film-interference-for-computer-graphics-r2962/`.

[25] M. Marengo, *Image implementing spectral colours in mantra*, May 2008. [Online]. Available: `https://www.sidefx.com/tutorials/image-implementing-spectral-colors-in-mantra/`.

[26] Moritz, E. Centeno, and N. *, December 2020. [Online]. Available: `https://entagma.com/bubbles-again-simulating-soap-swirls-using-flip/`.

[27] M. Born and E. Wolf, *Principles of optics: electromagnetic theory of propagation, interference and diffraction of light*, 7th expanded ed. Cambridge University Press, 1999, ISBN: 9780521642224.

[28] W. Jakob, *Mitsuba renderer*, http://www.mitsuba-renderer.org, 2010.

[29] X. D. He, K. E. Torrance, F. X. Sillion, and D. P. Greenberg, "A comprehensive physical model for light reflection", *ACM SIGGRAPH Computer Graphics*, vol. 25, no. 4, Jul. 1991, ISSN: 0097-8930. DOI: `10.1145/127719.122738`.

[30] R. L. Cook and K. E. Torrance, "A reflectance model for computer graphics", *ACM Transactions on Graphics*, vol. 1, no. 1, January 1982, ISSN: 0730-0301, 1557-7368. DOI: 10.1145/357290.357293.

[31] F. Remondino, "From point cloud to surface: The modeling and visualization problem", *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 34, March 2004. DOI: 10.3929/ethz-a-004655782.

[32] [Online]. Available: https://www.sidefx.com/docs/houdini/model/volumes.html.

[33] A. W. Reinold and A. W. Rücker, "On the thickness of soap films", *Proceedings of the Royal Society of London*, vol. 26, pp. 334–345, 1877, ISSN: 03701662.

[34] T. Cuypers, T. Haber, P. Bekaert, S. B. Oh, and R. Raskar, "Reflectance model for diffraction", *ACM Transactions on Graphics*, vol. 31, no. 5, August 2012, ISSN: 0730-0301, 1557-7368. DOI: 10.1145/2231816.2231820.