UNIVERSIDAD NACIONAL DEL SUR

TESIS DE DOCTOR EN INGENIERÍA

Metodologías para el modelado y renderizado de interfases sólido-líquido en Computación Gráfica

Juan Miguel Bajo

BAHÍA BLANCA ARGENTINA

2020

Prefacio

Esta Tesis es presentada como parte de los requisitos para optar al grado académico de Doctor en Ingeniería, de la Universidad Nacional del Sur, y no ha sido presentada previamente para la obtención de otro título en esta Universidad u otras. La misma contiene los resultados obtenidos en investigaciones llevadas a cabo en el Laboratorio de Ciencias de las Imágenes (UNS - CONICET) durante el período comprendido entre el día 1 de Abril de 2013 y el 30 de Julio de 2020, bajo la dirección del Dr. Claudio Delrieux, Profesor Titular del Departamento de Ingeniería Eléctrica e Investigador Independiente del CONICET.

Ing. Juan Miguel Bajo
jmbajo@gmail.com
Departamento de Ingeniería Eléctrica y de Computadoras
Universidad Nacional del Sur
Bahía Blanca, 30/07/2020

UNIVERSIDAD NACIONAL DEL SUR
Secretaría General de Posgrado y Educación Continua

La presente tesis ha sido aprobada el …./…./…... , mereciendo la calificación de .....(……)

Agradecimientos

En primer lugar quiero agradecer a mi director Dr. Claudio Delrieux, por su confianza, paciencia y aporte en el desarrollo de esta tesis.

Al Dr. Gustavo Patow por su colaboración fundamental para el desarrollo del trabajo y las publicaciones.

Además, quiero hacer extensivo el agradecimiento al jurado por sus correcciones y valiosos comentarios sobre la tesis. En especial a la Dra. Silvia Castro por enseñarme Computación Gráfica.

A la Universidad Nacional del Sur y a todo el Departamento de Ingeniería Eléctrica y de Computadoras. A la Secretaria de Posgrado Mg. Andrea Silvetti por su ayuda en la preparación de la defensa de tesis junto a Gustavo Goñi.

A mis compañeros del Laboratorio de Ciencias de las Imágenes.

A CONICET por haberme otorgado una beca doctoral con la cual fue posible realizar esta tesis.

A Marina Cipolletti y Albertina Popp por su ayuda en la preparación de la disertación.

A Manlio Massiris y a Steven Martinez Vargas por su compañía los últimos años.

A mis colegas de Ebers Med® por el espacio de crecimiento laboral y las oportunidades brindadas.

A mis padres, Miguel y Viviana; a mis hermanas, Antonela, Juliana y Josefina.

A mis abuelos Oscar, Beba y Luisa; a mis tíos y primos.

A todos mis amigos, estudiantes y colegas quienes colaboraron desde su lugar para que este trabajo sea posible.

Resumen

Según el Principio de Arquímedes, un objeto total o parcialmente sumergido es afectado por una fuerza vertical hacia arriba, conocida como fuerza de empuje de magnitud igual al peso del fluido desalojado. Esta descripción simple explica un conjunto variado de efectos que son ubicuos en la naturaleza y están cada vez más presentes en videojuegos, animaciones generadas por computadora y aplicaciones gráficas interactivas en general. Si bien existen soluciones al problema del acoplamiento entre fluidos y sólidos para algunos casos particulares, es necesario el desarrollo de nuevos modelos de flotabilidad precisos y suficientemente generales. En esta tesis, se propone un método para computar de manera realista el problema del acoplamiento fluido-a-sólido que es adecuado para un amplio espectro de casos, como objetos rígidos o deformables, sólidos o huecos, permeables o impermeables y con distribución de masa variable. En el caso de objetos permeables que permiten el ingreso de líquido hacia dentro del objeto, el método presentado incorpora la dinámica del fluido en el que el objeto está sumergido y desacopla el cómputo de las magnitudes físicas involucradas en la fuerza de flotabilidad del objeto vacío respecto al líquido contenido dentro de él.

Por otro lado, la apariencia visual de ciertos materiales depende de su propiedades intrínsecas de transferencia de luz, de la iluminación presente y de otras contribuciones de ambiente. En particular, ésto se produce en el caso de materiales porosos, rugosos o absorbentes donde la presencia de líquido sobre la superficie altera de manera significativa la función de distribución de reflectancia bidireccional (BRDF) que, a su vez, resulta en

cambios notables en la apariencia visual. Por esta razón, el renderizado de materiales que alteran su apariencia bajo condiciones de humedad continúa siendo un tópico de relevancia en el área de Computación Gráfica. En esta tesis, además, se introduce una nueva técnica para modelar y renderizar los cambios de apariencia de los materiales absorbentes cuando existe líquido en su superficie.

En primer lugar, se desarrolló un nuevo método para resolver el problema de la interacción entre un medio líquido y la superficie de un objeto utilizando, para esto, sus coordenadas de textura. Seguidamente, se propuso un algoritmo para modelar los principales procesos físicos que ocurren en la superficie de un objeto sólido húmedo o mojado. Finalmente, se extiende el modelo propuesto por la literatura para explicar el cambio de apariencia de ciertos materiales absorbentes bajo condiciones de humedad y se implementa un algoritmo logrando tiempos aptos para aplicaciones en Tiempo-Real. La solución completa es diseñada teniendo en cuenta la implementación en arquitecturas superescalares y usando aceleración por GPU *(Graphics Processing Unit)*, permitiendo su integración con los *pipelines* de procesamiento gráfico actuales.

Abstract

Following the celebrated Archimedes' Principle, any object wholly or partially immersed in a fluid is subject to an upwards force, known as buoyancy force, equal to the weight of the fluid displaced by the object. This simple description is the origin of a set of effects that are ubiquitous in nature, and are becoming commonplace in games, animation, and interactive applications in general. Even though there are solutions to the fluid-to-solid coupling problem in some particular cases, to the best of our knowledge, comprehensive and accurate computational buoyancy models adequate in general contexts are still lacking. We propose a real-time GPU-based algorithm for realistic computation of the fluid-to-solid coupling problem, which is adequate for a wide generality of cases (rigid or deformable objects, solid or hollow, permeable or waterproof, and with variable masses). The method incorporates the dynamics of the fluid on which the object is immersed, and decouples the computation of the physical parameters involved in the buoyancy force of the empty or dry object from the mass of liquid contained inside. The dynamics of this mass of liquid are also computed, in a way such that the relation between the center of mass of the object and the buoyancy force may vary, leading to complex, realistic behaviors such as the one arising with a sinking boat.

The visual appearance of materials depends on their intrinsic light transfer properties, the illumination and camera conditions, and on other environmental contributions. In particular, this is the case in porous, rough, or absorbent materials, where the presence of liquid on the surface alters significantly their BRDF (bidirectional reflectance distribution

function), which in turn results in significant changes in their visual appearance. For this reason, rendering materials that change their appearance when wet continues to be a relevant topic in Computer Graphics, especially when real-time photo-realistic rendering is required in scenes involving this kind of materials in interaction with water or other liquids. In this paper we introduce a physically inspired technique to model and render the changes in appearance of absorbent materials when liquid is present on their surface. First, we develop a new method to solve the interaction between the liquid phase and the object surface using the latter's own underlying texture coordinates. Then, we propose an algorithm to model the diffusion phenomenon that occurs in the interface between a solid porous object and a translucent liquid. Finally, we extend a model which explains the change of appearance of materials under wet conditions, and we implement it achieving real-time performance. The complete model is developed using GPU (Graphics Processing Unit) acceleration, which can be seamlessly integrated with the usual rendering pipeline in the most popular graphic libraries.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

La interacción sólido-líquido[1] de objetos sumergidos en medios líquidos ha sido un tópico de investigación activo en el área de Computación Gráfica desde sus inicios, siendo de particular importancia en videojuegos, realidad virtual y otras aplicaciones interactivas. La interacción sólido-a-líquido ha recibido una significativa atención por su importancia en simuladores y contextos similares.

A pesar de la gran importancia en la investigación, un modelo interactivo y completo para la interacción líquido-a-sólido es todavía un problema abierto, ya que la mayoría de las técnicas presentadas usan geometrías 'proxies' altamente simplificadas (Gonzalez-Ochoa, 2016) o no son aplicables a soluciones de tiempo-real debido a su complejidad computacional (Lu et al., 2016). Uno de los desafíos de este problema es el requerimiento de computar de manera rápida y precisa las variables involucradas en la dinámica del movimiento, tomando en cuenta situaciones como, entre otras cosas, líquido entrando al objeto y moviéndose dentro de él.

El avance de esta tesis con respecto al problema descripto es proponer un algoritmo para el cómputo aproximado, aunque realista, del problema del acoplamiento two-way, es decir sólido-a-líquido y viceversa simultáneamente. Actualmente, existen soluciones satis-

[1] A lo largo de esta tesis nos referiremos a líquido, fluido y agua como sinónimos. Las ideas presentadas son adecuadas para otros líquidos ajustando los correspondientes valores.

factorias para solo una parte del problema correspondiente al acoplamiento sólido-a-líquido (Jeschke et al., 2018), por lo tanto esta tesis se centrará en el problema complementario, la interacción líquido-a-sólido. El concepto fundamental subyacente de la solución planteada es, que con una parametrización razonable, las texturas tradicionalmente usadas para color y otras características visuales pueden ser utilizadas para computar de manera escalable todas las magnitudes físicas necesarias para simular de manera acorde el sistema en cuestión.

Otra faceta del problema de la interacción sólido-líquido consiste en la capacidad de los seres humanos de diferenciar superficies mojadas de las secas basándose solamente en cambios sutiles de su apariencia visual. El color de muchos materiales (especialmente los porosos/rugosos) lucen más oscuros y con coloraciones más vívidas cuando están mojados. Este fenómeno, fácilmente observable, puede ser explicado por las complejas interacciones de la luz que toman lugar en la superficie del objeto cuando existe una capa de líquido traslúcido sobre su superficie. En otros términos, la presencia de una capa de líquido genera una superficie más brillante, generando una apariencia visual más vívida y, al mismo tiempo, incrementa la proporción de luz que es absorbida por el material, oscureciendo y saturando el color percibido.

Otro fenómeno que tiene lugar, fácilmente percibido por los humanos, es el líquido (especialmente el agua) goteando, drenando y secándose sobre una superficie húmeda. En particular, la evolución de este fenómeno a lo largo del tiempo da lugar a diversas apariencias que los humanos perciben ávidamente como una impresión visual única. Como ejemplo, observando un jarrón de cerámica rociado podemos percibir inmediatamente cómo se salpicó el líquido, si éste es acuoso o aceitoso, si la cerámica es brillante o porosa, cuánto líquido se está infiltrando o evaporando y muchas otras señales sobre procesos que pueden ser particularmente complejos. Otro ejemplo es la interacción agua-arena-viento-sol en las playas: nuestra percepción ofrece varios indicios sobre las propiedades de la

escena, por ejemplo, si la arena es fina o gruesa, cómo el agua de mar está drenando y evaporándose, o si la superficie puede ser resbaladiza, por mencionar solo algunos ejemplos.

Este tipo de percepciones se naturalizan a través de nuestra experiencia diaria, pero son complicadas de modelar en el área del renderizado fotorrealista. Los efectos ópticos que surgen en superficies húmedas y los modelos adecuados para la difusión, absorción y evaporación en materiales porosos se han estudiado en detalle en Física, Visión por Computadora y otras disciplinas, pero hasta ahora, solo unos pocos modelos consistentes han sido propuestos en Computación Gráfica.

En esta tesis también presentamos un modelo integral inspirado en la Física para renderizar materiales absorbentes húmedos. A pesar de que nuestra propuesta es de naturaleza fenomenológica, se pretende ser exhaustivos al tener en cuenta la mayoría de los factores relevantes que conducen a la apariencia visual final de las superficies húmedas, incluida la geometría del objeto, las propiedades microfísicas del material (por ejemplo, su porosidad y micro textura) y el comportamiento dinámico del líquido, incluida la absorción, difusión y evaporación. Esto proporciona la capacidad de modelar materiales absorbentes húmedos razonablemente de una manera intuitiva y flexible. Aunque el modelo es una simplificación de la física subyacente, los resultados son realistas y pueden ser calculados en tiempo real.

Contribuciones

- Según nuestro conocimiento, éste es el primer método capaz de calcular las magnitudes significativas necesarias para describir el movimiento de flotabilidad de un modelo 3D en tiempo real: centro de gravedad, momentos de inercia, centro de flotabilidad, rozamientos, entre otras. El método propuesto soporta objetos deformables o con forma variable, ya que puede computar las magnitudes requeridas de manera instantánea y continua.

- Proponemos también una nueva estructura de datos y un algoritmo para representar dinámicamente el líquido dentro de un objeto 3D sumergido, además de los mecanismos para calcular su ingreso y desplazamiento dentro del objeto sumergido dependiendo de su movimiento. Esto permite simular no solo la flotabilidad de un objeto sumergido, sino también su comportamiento de forma realista, lo que, según nuestro conocimiento, no es posible hasta el momento de manera general.

- La técnica propuesta no requiere de procesamiento manual de los modelos de entrada, ya que utiliza la parametrización típicamente usada para mapeo de textura en la etapa de la creación artística del modelo.

- Desarrollamos una técnica de representación y modelado de materiales húmedos inspirada en la física subyacente.

- Implementamos un método computacional para la simulación fotorrealista y la representación de la dinámica del líquido sobre una superficie, a partir de de los procesos físicos subyacentes de la difusión, evaporación y absorción.

- Finalmente, integramos estos procesos en un *pipeline* en tiempo real que aprovecha las características de los últimos motores gráficos y que es totalmente integrable en la mayoría de las bibliotecas y *frameworks* de desarrollo en Computación Gráfica.

Diseminación de los resultados

Artículos en revistas

- Bajo, J. M., Delrieux, C. A., y Patow, G. *Physically inspired technique for modeling wet absorbent materials*. The Visual Computer. (En prensa). https://doi.org/10.1007/s00371-020-01963-w.

- Bajo, J. M., Patow, G., y Delrieux, C. A. *Realistic buoyancy model for real-time applications*. Computer Graphics Forum, (Publicado on-line 19/5/2020, https://doi.org/10.1111/cgf.14013).

- Massiris Fernández, M., Álvaro Fernández, J., Bajo, J., y Delrieux, C. *Sistema automatizado para monitorear el uso de equipos de protección personal en la industria de la construcción*. Revista Iberoamericana de Automática e Informática industrial, https://10.4995/riai.2020.13243.

Artículos en conferencias

- Juan Miguel Bajo, Cristian García Bauzá, Claudio Delrieux. *Modelado de Texturas de Materiales Absorbentes Bajo Condiciones de Humedad*. Escuela y Workshop Argentino de Ciencias de las Imágenes 2014. Buenos Aires, 11 al 15 de Agosto de 2014.

- Leonardo Molas, Juan Miguel Bajo, Marina Cipoletti, Claudio Delrieux. *An Attempt to Disambiguate Saddle Points in Marching Squares*. Escuela y Workshop Argentino de Ciencias de las Imágenes 2014. Buenos Aires, 11 al 15 de Agosto de 2014.

- Massiris, M., Bajo, J., Delrieux, C. *Aproximación a la evaluación de riesgos ergonómicos mediante visión por computadora*. (2019). XXVII. Jornadas de Jovens Pesquisadores da Associação de Universidades. Grupo Montevideo, São Carlos, 23-25 de octubre de 2019.

Presentaciones en jornadas

- Juan Miguel Bajo, Claudio Delrieux. *Modelización y Renderización de la Interfase Líquido-Sólido*. Jornadas de Becarios y Tesistas 2017. Instituto de Investigaciones en Ingeniería Eléctrica 'Alfredo Desages' (IIIE). Departamento de Ingeniería Eléctrica y de Computadoras. Bahía Blanca, 24 de noviembre de 2017.

Capítulo 2

Trabajo Previo

2.1. Cálculo de Flotabilidad

La interacción de medios líquidos sobre objetos sólidos sumergidos total o parcialmente ha sido un tema de investigación en Computación Gráfica desde sus inicios (Bridson, 2015), siempre con el objetivo de lograr imágenes con alto realismo visual. En la literatura existen métodos *off-line* (no tiempo real) que proveen soluciones precisas aunque no son viables para aplicaciones interactivas. Para el caso de tiempo-real, los métodos propuestos utilizan generalmente geometrías *proxy*, es decir, simplificaciones de la geometría real del modelo simulado (Gonzalez-Ochoa, 2016; Gourlay, 2019) que típicamente sufren la falta de realismo y, en algunos casos, requieren la generación manual de geometría extra.

2.1.1. Métodos *Off-line*

Carlson et al. (2004) propusieron un método para resolver el acoplamiento bidireccional (la interacción del líquido hacia el sólido y del sólido al líquido) modelando los objetos sólidos como si fueran fluidos. La rigidez es asegurada aplicando restricciones al campo velocidad en el interior. Guendelman et al. (2005) desarrollaron un modelo para resolver el acoplamiento sólido-líquido válido para superficies de grosor infinitesimal, como telas o recipientes convexos utilizando partículas. Posteriormente, Batty et al. (2007) propusieron un enfoque variacional que permite una solución precisa en grillas regulares cartesianas relativamente gruesas, estableciendo el paso de proyección de presión como una minimización de la energía cinética. Esta idea es relativamente similar a otros enfoques al problema del contacto con cuerpos rígidos, donde un acoplamiento robusto entre líquido y sólido genera un sistema lineal simétrico bien condicionado y definido positivo que puede ser resuelto con algoritmos numéricos conocidos.

Robinson-Mosher et al. (2008) presentaron un algoritmo para el cómputo preciso del acoplamiento de fluídos a sólidos rígidos y deformables. Posteriormente, Robinson-Mosher et al. (2009) desarrollaron un método para computar las velocidades tangenciales para la interacción fluido-sólido. Akinci et al. (2012) desarrollaron un método para el acopla-

miento de fluidos modelados con partículas SPH (Smoothed-particle hydrodynamics) y objetos rígidos basados en las fuerzas hidrodinámicas. Este método conserva el momento lineal y está basado en reconstruir la superficie con partículas que interactúan con el fluido. Gerszewski and Bargteil (2013) presentaron un método basado en la combinación de incompresibilidad unilateral, el método FLIP (Brackbill and Ruppel, 1986) y límites borrosos (*blurred boundaries*) que es acorde para la animación de fluidos a gran escala, principalmente salpicaduras. En una contribución posterior Gerszewski et al. (2013) proponen un esquema de simulación *model-reduced* combinando bases *data-driven* o artísticas con bases más generales derivadas a partir de autofunciones Laplacianas.

Yin et al. (2013) propusieron un *framework* eficiente basado en un modelo de partícula unificado para el acoplamiento sólido-fluido incluyendo cuerpos rígidos y elásticos. Su técnica aplica diferentes esquemas de acoplamiento para el caso fluido-sólido y sólido-sólido respectivamente. Da et al. (2016) presentaron una técnica para computar la superficie de líquidos tomando en cuenta efectos como la tensión superficial, pero la interacción con objetos sólidos está limitada a salpicaduras solamente. Teng et al. (2016) presentaron un método Euleriano para el cómputo del acoplamiento. Simultáneamente, Lu et al. (2016) presentaron un *framework* para simulación del acoplamiento de fluidos incompresibles con objetos deformables.

Canabal et al. (2016) presentaron un método para computar olas preservando sus propiedades de dispersión que soportan interacción con obstáculos. El método computa la aceleración usando un *kernel* de dispersión como un filtro variable espacialmente el cual es posible computar de manera eficiente a partir de una pirámide de *kernels*. La interacción con obstáculos es computada modulando el kernel de dispersión.

Akbay et al. (2018) presentaron un método extendido particionado para el acoplamiento bidireccional sólido-fluido, donde el modelado individual de cada fase es tratado como un modelo caja-negra con interfaces limitadas y expuestas, lo cual facilita el reúso y la modularidad. Por su lado, Wang and Whiting (2016) presentaron un método de diseño

para la *fabricación* de objetos flotantes. En esta propuesta, el usuario elige la orientación deseada de la línea de agua, luego de lo cual se utiliza un procedimiento de optimización para computar las configuraciones internas necesarias. Luego de ser impresos en 3D, estos objetos flotan de la manera y posición deseados. Desde una perspectiva diferente, hay varios trabajos que proveen modelos computacionales bio-mecánicos para cuerpos articulados en movimiento dentro de fluidos (Kwatra et al., 2010; Si et al., 2014).

La interacción de agua absorbida por ciertos tipos de materiales ha sido extensivamente estudiada. En trabajos como por ejemplo Patkar and Chaudhuri (2013a); Lenaerts et al. (2008a); Fei et al. (2018) se proponen modelos para describir el cambio de apariencia y simular el flujo del líquido dentro de una determinada geometría. Ninguno de estos métodos fue diseñado bajo restricciones de tiempo-real, sino para ser integrados en *solvers* más precisos para cómputo *off-line*. Al contrario, las ideas presentadas en la presente tesis fueron diseñadas para ser precisas y completamente usables en entornos interactivos con restricciones de tiempo-real, como juegos y aplicaciones de realidad virtual.

2.1.2. Métodos interactivos

Como se dijo anteriormente, en la industria de videojuegos, es corriente el uso de geometrías *proxies* para el acoplamiento del agua sobre el sólido (Gonzalez-Ochoa, 2016; Gourlay, 2019).

Por otro lado, la interacción del sólido sobre el líquido ha sido también estudiada en la literatura de Computación Gráfica. Un ejemplo es el trabajo de Yuksel et al. (2007), quienes presentaron el concepto de *Wave Particles*, las cuales permiten un enfoque simple, rápido e incondicionalmente estable para la simulación de agua. El modelo convierte *Wave Particles* en una superficie que es deformada horizontalmente para modelar el flujo local inducido.

Como una extensión natural de este trabajo, Jeschke et al. (2018) presentaron *Water Surface Wavelets*, una técnica para modelar olas usando la transformada *wavelet* que dis-

cretiza el movimiento del líquido a partir de funciones de amplitud variable. Esto permite eliminar algunas limitaciones como la condición CFL y el límite de Nyquist permitiendo simular olas con gran nivel de detalle a grandes resoluciones espaciales. Entre otras extensiones, el trabajo presenta acoplamiento bidireccional sólido-líquido a partir de obstáculos como botes o boyas emitiendo ondas usando formas circulares. La interacción del fluido sobre el sólido es computada por integración sobre una superficie matemática primitiva simplificada.

Macklin et al. (2014) propusieron un método para describir el movimiento de objetos sólidos completamente sumergidos reemplazando el tensor de inercia por el tensor de Kirchoff. Este método funciona de manera precisa para objetos completamente sumergidos en los cuales el total de la superficie está en contacto con el líquido. Thürey et al. (2007) propusieron un método para simular de manera realista el comportamiento de las olas en la zona de rompiente.

Como ya fue mencionado, el objetivo de esta tesis es desarrollar métodos que mejoren los actuales al usar la geometría real del modelo y sus propiedades para realizar los cómputos. Además, se incorpora un modelo simple pero efectivo para modelar el líquido en el interior del modelo en el caso que sea necesario, logrando efectos que hasta el momento no eran posibles con técnicas integrales.

2.2. Materiales húmedos

Las propiedades ópticas y la apariencia visual de materiales frente a diferentes condiciones fue el foco de un gran número de investigaciones, tanto en renderizado en tiempo real u *off-line* (Akenine-Möller et al., 2018; Pharr and Humphreys, 2010; Dorsey et al., 2008). Los enfoques más simples, utilizados en soluciones de tiempo real, consisten en interpolar entre texturas para la apariencia seca y mojada, en vez de modelar una difusión, absorción y evaporación de líquido más precisas sobre la superficie del objeto, generando imágenes de poco realismo visual.

En Computación Gráfica, Jensen et al. (1999) desarrollaron una solución *ad-hoc* basada en un promedio pesado de dos funciones de fase Henyey-Greenstein (Kattawar, 1975), ajustando el grado de *forward scattering* para lograr el nivel deseado de humedad.

Ese método, computado a partir de un esquema de Monte-Carlo, permite simular un amplio rango de materiales como papel, piel y muchos otros. Sin embargo, la complejidad del modelo lo hace inadecuado para renderizado en tiempo-real. De manera similar, Lu et al. (2005) estudiaron el proceso de secado usando muestreo en modelos reales. Para superficies que cambian de apariencia a lo largo del tiempo, Gu et al. (2006) propusieron un método para adquirir y transferir BRDFs variables en el tiempo. Además, publicó una base de datos de la evolución de materiales bajo diferentes condiciones ambientales.

Si bien los resultados son prometedores, la dependencia del tiempo de cómputo no permite su aplicación en soluciones interactivas, donde el cambio de apariencia depende de eventos de la misma simulación y no pueden ser precomputados. Por otro lado, estos modelos generan resultados de alta calidad visual pero son específicos para ciertos materiales y condiciones (por ejemplo, madera bajo alta temperatura), por lo tanto, no proveen una solución general que pueda aplicarse a diferentes escenarios.

En Kimmel and Baranoski (2007), los autores usan técnicas de Monte-Carlo para simular la interacción de la luz con partículas individuales de arena. Su modelo usa como entrada los parámetros físicos y las características mineralógicas que describen una determinada muestra de arena, y produce las magnitudes radiométricas a partir de la reflectancia espectral y la BRDF. De manera similar a otros modelos estocásticos, este enfoque es computacionalmente costoso ya que se requieren gran cantidad de muestreos para lograr convergencia en el resultado, haciéndolo poco apropiado para aplicaciones interactivas.

En un trabajo posterior sobre simulación de la apariencia de arena, Kimmel and Baranoski (2010), los autores utilizan la técnica anterior como una herramienta de preprocesado que almacena los datos radiométricos para arena bajos diferentes condiciones ambientales

y de iluminación para realizar, posteriormente, una predicción eficiente de la apariencia final. Si bien el último paso puede usarse para aplicaciones interactivas, el pipeline completo no, ya que la primera etapa no es interactiva. Al contrario, nuestro modelo es un pipeline completo, interactivo y basado en física para materiales húmedos en general, no solamente arena.

Con el objetivo de modelar visualmente la porosidad y las micro-fracturas en la superficie, en Merillou et al. (2000) se define un método que utiliza huecos parametrizados embebidos en la superficie del objeto. En este trabajo, la humedad es dejada como trabajo futuro y retomada posteriormente en Hnat et al. (2006) donde se extiende el modelo asumiendo que los huecos de la superficie se llenan con líquido. Sin embargo, la interacción dinámica del líquido y el objeto no es estudiada. En Patkar and Chaudhuri (2013b) se presenta un modelo de mojado para medios porosos donde el fluido es modelado a partir de partículas y la interacción líquido-sólido se resuelve a partir del cómputo de colisión partícula-triángulo, a diferencia de nuestro modelo que hace uso extensivo del espacio de textura.

En Lenaerts et al. (2008b) se propone un método para modelar el flujo dentro de medios porosos que asume una descripción Lagrangiana del fluido y un correspondiente muestreo del modelo geométrico con partículas. Esto permite simulaciones altamente realistas del flujo dentro del modelo (incluyendo escurrimientos) pero los tiempos de cómputo no son aptos para tiempo real. Además el muestreo del objeto con partículas impone restricciones adicionales.

Algunos materiales específicos tienen un comportamiento particular cuando están mojados y existen técnicas para estos casos, por ejemplo, un modelo para pelo mojado es presentado en Rungjiratananon et al. (2012)). Para el caso de la dispersión de tinta en papel absorbente, Chu and Tai (2005) presentan un modelo basado en la ecuación de Lattice-Boltzmann para generar trazos de tinta de alta calidad.

Además, la interacción arena-agua ha sido un punto significativo de interés en la comunidad de Computación Gráfica. En Rungjiratananon et al. (2008) se presenta un método basado en partículas para modelar los fenómenos que ocurren en materiales granulares bajo condiciones de humedad, como líquido penetrando en los espacio inter-partículas cambiando la fuerzas internas del sistema a partir de *puentes* generados entre las partículas de material.

Este enfoque basado en partículas posibilita simular fenómenos específicos de manera realista pero carece de escalabilidad necesaria para grandes extensiones de material debido a restricciones de tiempo y complejidad computacional. Por otro lado, fenómenos como secado y cambio de apariencia no son estudiados.

En Liu et al. (2005) se propone agregar una divergencia no nula en las ecuaciones de Navier-Stokes para modelar el efecto de la penetración de agua desde la superficie al material del sustrato. Esta técnica es capaz de renderizar efectos de humedad a partir de dos texturas: una del material seco y otra totalmente húmedo. Luego, cualquier estado intermedio es obtenido a partir de la combinación lineal de ambas imágenes, lo cual resulta en efectos rápidos y de buena calidad aunque sin fundamento físico. Fenómenos de larga duración, como el desgaste y envejecimiento también han sido estudiados en diversa bibliografía (Mérillou and Ghazanfarpour, 2008; Dorsey et al., 1999, 1996). La mayoría de estos modelos no están diseñados para aplicaciones en tiempo-real ya que emulan procesos naturales lentos y su objetivo es la imagen final.

En el campo de la Visión por Computadora el tema de materiales húmedos también es de particular interés (Mall and Da Vitoria Lobo, 1995; Yacoob, 2013), aunque el problema es resolver el problema inverso, es decir, reconocer materiales secos y mojados a partir de imágenes capturadas y no la síntesis de imágenes como se propone en esta tesis.

La simulación computacional de fluidos ha sido un tema activo en diferentes áreas de Computación Gráfica. Existen dos grandes diferentes enfoques que describen fluidos de manera computacional: Euleriano y Lagrangiano. El enfoque Euleriano describe el

fluido en locaciones específicas del espacio, generalmente a partir de una discretización del espacio en forma de grilla (Stam, 1999; Bridson, 2008). Por el otro lado, el enfoque Lagrangiano describe el fluido con un conjunto de partículas cada una de ellas con un estado determinado (Müller et al., 2003; Macklin and Müller, 2013; Yan et al., 2009).

Desde el punto de la física óptica, Lekner and Dorf (1988) presentan una explicación de por qué los materiales absorbentes son percibidos más oscuros cuando están mojados. Este estudio extiende investigaciones previas (Ångström, 1925) que proponen que el oscurecimiento del color se produce por la fina capa de agua sobre la superficie del material. Esta capa de agua agrega otra interfase entre materiales, por lo tanto una proporción de los rayos de luz reflejados en la superficie son reflejados nuevamente hacia la superficie del objeto. Este proceso incrementa la proporción total de luz absorbida por el sistema generando la apariencia de colores más oscuros. El objetivo final del mencionado trabajo, no es generar imágenes sintéticas sino explicar el proceso y caracterizar este tipo de materiales.

Es particularmente notable que ninguno de los trabajos descriptos anteriormente, encara el problema de la interacción dinámica en la interfase entre la superficie de objeto genérico y el líquido en tiempo-real. En general, se asume que la superficie está previamente mojada desligándose del proceso de interacción con el líquido. Si bien el fenómeno está altamente estudiado, los formalismos subyacentes son relativamente complejos para ser implementados computacionalmente. Esto explica la causa de por qué escenas con este tipo de interacciones se han visto solo recientemente en el área de Computación Gráfica.

Capítulo 3

Simulación de Flotabilidad

En este capítulo describiremos un método novedoso para computar todas las variables involucradas en el fenómeno de la flotación y la interacción líquido-sólido de objetos sumergidos (parcial o totalmente). Estas variables incluyen centro de masa, masa total, tensor de inercia y otras. En lo referente a la interacción líquido-sólido, se presentará un método para objetos sólidos impermeables flotando en un medio líquido. En el capítulo próximo, se presentará un método para la simulación de objetos permeables teniendo en cuenta la infiltración y el movimiento del líquido dentro del objeto.

Para este caso, se necesita una parametrización $\mathbb{R}^3 \rightarrow \mathbb{R}^2$ de la malla poligonal del modelo, típicamente usada para la técnica de mapeo de texturas (Catmull, 1974). Con dicha información es posible el cómputo de las variables involucradas en la simulación de la flotabilidad de manera directa. Al emplear la parametrización utilizada para el mapeo de texturas, el método aprovecha la arquitectura multi-paralela del procesador gráfico (GPU), permitiendo computar las variables necesarias en tiempo real y de manera precisa, sin necesidad de agregar información extra a la geometría por parte del artista. El método puede ser utilizado para modelos deformables o dinámicos ya que las métricas pueden computarse interactivamente. Para la simulación de la interacción sólido→líquido

existen diferentes soluciones; particularmente, la implementación de nuestra solución es compatible con el trabajo propuesto por Jeschke et al. (2018).

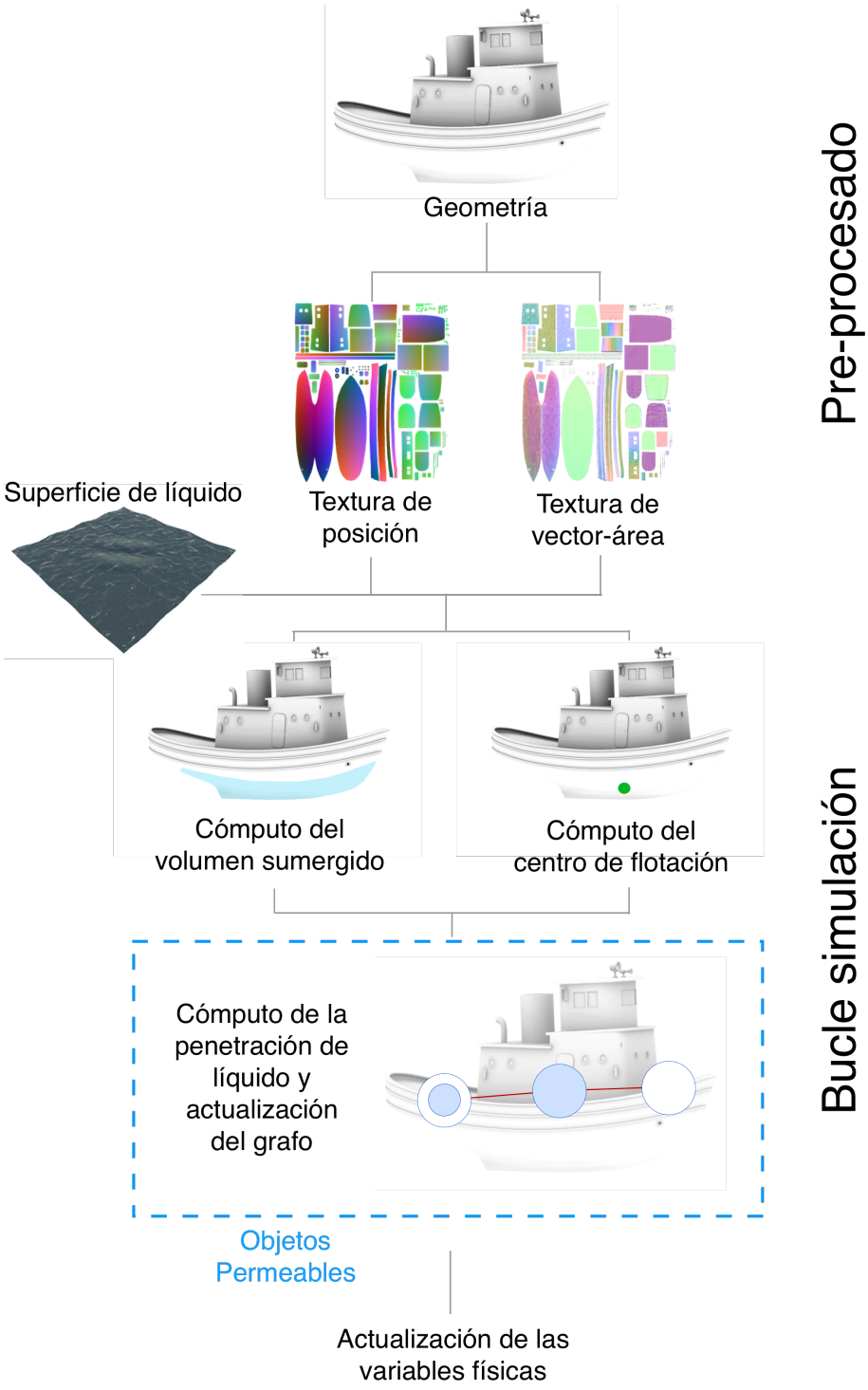


Figura 3.1: Descripción general de nuestro método: a partir del modelo de entrada parametrizado calculamos dos texturas adicionales (*i.e.*, la textura de posiciones y la de vector-área), utilizadas para calcular los parámetros del modelo físico (*e.g.*, volumen sumergido, centro de flotabilidad, tensor de inercia, etc.). Finalmente, calculamos la penetración del agua en la geometría del modelo, y usamos una estructura auxiliar, el grafo de líquido, para obtener las magnitudes físicas relevantes. Este método se explicará en el capítulo próximmo.

3.1. Introducción

Para resolver el movimiento de flotación de manera precisa, el método requiere la evaluación de ciertas magnitudes físicas correspondientes al objeto seco (sin líquido contenido en su interior) modelado, éstas son masa total, tensor de inercia y centro de masa. Estas magnitudes son automáticamente generadas y pueden ser precomputadas en caso que el objeto no se deforme durante la simulación o pueden ser calculadas en cada cuadro de la simulación, claramente con una penalización en el rendimiento.

Dado un modelo $\mathcal{M}$, las propiedades de masa son: la masa total M, el centro de masa C y el tensor de inercia I, dado por una matriz de 3x3 simétrica. Asumiendo que la superficie $\mathcal{M}$ encierra un volumen $\Omega \in \mathbb{R}^3$ que corresponde al sólido de densidad constante ρ, podemos expresar las propiedades de masa como Bächer et al. (2017):

$$\mathbf{b} = \begin{bmatrix} 1 & x & y & z & xy & yz & xz & x^2 & y^2 & z^2 \end{bmatrix}, \tag{3.1}$$

calculando la integral de volumen sobre Ω:

$$\mathbf{s}(\rho) = \begin{bmatrix} s_1 & s_x & s_y & s_z & s_{xy} & s_{yz} & s_{xz} & s_{x^2} & s_{y^2} & s_{z^2} \end{bmatrix}, \tag{3.2}$$

donde

$$\mathbf{s}_t(\rho) = \rho \int_\Omega t \; dV. \tag{3.3}$$

Los términos definidos en la ecuación 3.2 pueden ser expresados como integrales de superficie utilizado el *Teorema de la Divergencia*. Para esto identificamos un campo vectorial $\mathbf{B}$ para cada componente b de $\mathbf{b}$ tal que $\nabla \cdot \mathbf{B} = b$:

$$\mathbf{B} = \left[\begin{array}{c|ccc|ccc|ccc} 0 & x^2/2 & 0 & 0 & x^2y/2 & 0 & 0 & x^3/3 & 0 & 0 \\ y & 0 & y^2/2 & 0 & 0 & y^2z/2 & 0 & 0 & y^3/3 & 0 \\ 0 & 0 & 0 & z^2/2 & 0 & 0 & z^2x/2 & 0 & 0 & z^3/3 \end{array}\right]. \tag{3.4}$$

Luego

$$\mathbf{s}(\rho) = \rho \int_{\Omega} \mathbf{b}\ dV = \rho \int_{\Omega} \nabla \cdot \mathbf{B}\ dV = \rho \int_{\partial\Omega} \mathbf{n} \cdot \mathbf{B}\ dS, \tag{3.5}$$

donde $\mathbf{n}$ es el vector normal a la superficie.

3.2. Estructuras de datos

Como se mostró en la sección anterior, es posible expresar las componentes del vector $\mathbf{s}(\rho)$ como integrales de superficie sobre el objeto. Para poder implementar el cómputo de las integrales de manera escalable y suficientemente rápida para una solución de tiempo-real, proponemos las estructuras de datos que se detallan a continuación.

3.2.1. Textura de posiciones

Con el objetivo de poseer una técnica escalable y suficientemente rápida de obtener esa información, se define un nuevo tipo de textura de tres canales llamada *Textura de Posiciones* (TP). Este concepto requiere de la geometría del modelo sólido y su correspondiente parametrización sobre un espacio 2D; típicamente, se utilizará la parametrización del mapeo de texturas clásico. La TP es generada automáticamente a partir de la malla poligonal y su mapeo de textura UV, comúnmente provistos por el artista (o generados de manera procedimental). En ella, cada texel almacena su propia posición en coordenadas de objeto. Desde el punto de vista matemático, la TP almacena el mapeo de textura inverso ($\mathbb{R}^2 \to \mathbb{R}^3$). Es importante aclarar que el mapeo de texturas debe ser biyectivo, de lo contrario perderíamos información durante el cómputo de la función inversa debido a que un mismo texel podría tener dos posiciones de objetos diferentes. Si bien el mapeo de texturas biyectivo es un requerimiento extra, en la actualidad no es considerado excesivo.

El proceso de generación de la TP consiste en tomar cada uno de los polígonos de la malla del objeto y transformarlos al espacio de textura. Asumiendo mallas formadas

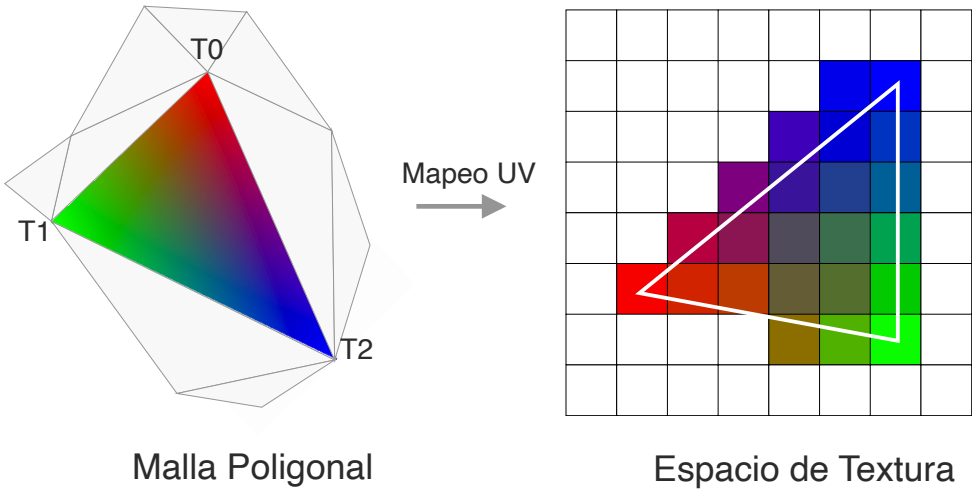


Figura 3.2: El método de generación de las texturas consiste en *rasterizar* el triángulo de la malla poligonal en una textura sintética, interpolando a partir de las coordenadas baricéntricas el atributo de los vértices que corresponde. Por ejemplo, para la textura de posiciones se interpola la posición del vértice en coordenadas de objeto. Para la textura de vector-área, se interpola el vector normal de cada vértice.

por triángulos, este proceso se puede hacer tomando los tres vértices formando una cara, proyectándolos en el espacio de textura usando el mapeo UV y luego interpolando los texels intermedios usando coordenadas baricéntricas.

Sea M un mapeo de texturas biyectivo, $P = P_0P_1P_2$ un triángulo de la malla poligonal, $P_t = M(P) = T_0T_1T_2$ el triángulo en coordenadas de textura y T un texel dentro del polígono P_t. Entonces, el valor de T es un valor vectorial dado por

$$T = \lambda_0 P_0 + \lambda_1 P_1 + \lambda_2 P_2,$$

donde λ_0, λ_1 y λ_2 son las coordenadas baricéntricas de T relativas a T_0, T_1 y T_2 (Ver Figura 3.2

La TP depende de la resolución adoptada y debe ser recalculada si cambia la resolución de la textura. Esta técnica podría considerarse similar al muestreo de objetos con partículas, pero tiene la característica importante de almacenar la información en una textura, lo cual permite accesos más eficientes durante la paralelización en la GPU. A partir de

esta textura y de las matrices de transformación clásicas (Model-View) podemos obtener la posición del texel en cualquier sistema de coordenadas de manera eficiente.

3.2.2. Textura de vector-área

Además de la TP presentada en la subsección anterior, necesitamos más información del texel para poder computar las integrales de superficie, componentes del vector $\mathbf{s}(\rho)$. En particular, necesitamos el vector normal a la superficie y el área de los segmentos de integración (dS). A partir de esto definiremos otra estructura de datos, llamada *Textura de Vector-Área* (TVA). La TVA es una textura de 4 canales (RGBA) y es una generalización del mapa de normales (Normal Map). Sea T_i el i-ésimo texel de la TVA, en él se almacena un vector 3D cuya dirección es paralela a la normal de la superficie $\mathbf{n}_i$ en dicho punto y su magnitud es el área (a_i) de la superficie ocupada por dicho texel en las coordenadas de mundo, es decir, almacena $\mathbf{A}_i = a_i\mathbf{n}_i$ como una 4-tupla:

$$T_i =< n_i^x, n_i^y, n_i^z, a_i >, \tag{3.6}$$

donde T_i, y n_i^x, n_i^y y n_i^z son las componentes del vector normal unitario del texel y a_i es el área de la superficie ocupada por el texel calculada como:

$$a_i = \frac{A_P}{N_T}, \tag{3.7}$$

donde A_P es el área de la superficie del polígono P en el cual el texel T_i está mapeado y N_T es el número de texels en los cuales P está mapeado. Claramente estos valores deben ser transformados linealmente de acuerdo a la profundidad (Bit Depth) de la imagen usada. Por ejemplo en imágenes de 8-bits por canal y tipo de dato entero, los valores serán codificados entre 0 y 255.

Es importante notar que el área proyectada en una dada dirección $\mathbf{r}$ puede ser computada como $\mathbf{A}_i \cdot \mathbf{r}$, donde $\cdot$ es producto escalar. En particular, podemos computar el área

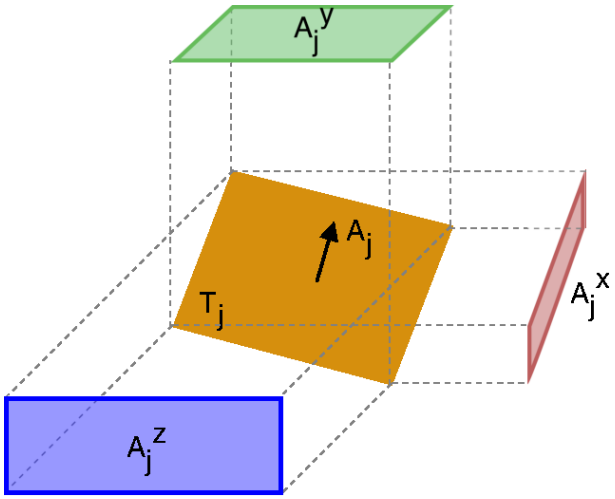


Figura 3.3: Dado un polígono T_j y su correspondiente vector-área $\mathbf{A_j}$, el área proyectada por el polígono en una dada dirección puede calcularse como la proyección del vector-área sobre un versor paralelo a la dirección correspondiente. Si bien en la imagen se muestran las proyecciones según los ejes cartesianos A_j^x, A_j^y y A_j^z, es importante notar que el método es válido para cualquier dirección.

proyectada en cualquiera de los ejes tomando la componente x, y o z correspondiente del texel, las cuales notaremos como $\mathbf{A}_i^x$, $\mathbf{A}_i^y$ o $\mathbf{A}_i^z$ para claridad. Ver figura 3.3.

3.3. Propiedades de masa

A partir de los términos definidos en la sección 3.1 y de las estructuras de datos definidas anteriormente, proponemos un método computacional para calcular las propiedades de masa: masa total, centro de masa, y tensor de inercia.

Masa: La masa de un objeto sólido limitado por un determinado volumen $\mathcal{V}$ puede ser computado como

$$M = \int_{\mathcal{V}} \rho \, dv,$$

es decir, la integral de la densidad de masa en cada punto sobre el volumen del cuerpo. En objetos de densidad homogénea ρ es constante, por lo tanto la masa total será una

constante por el volumen del cuerpo

$$M = \int_{\mathcal{V}} \rho \; dv = \rho \int_{\mathcal{V}} \; dv.$$

En función de los términos definidos,

$$M = s_1.$$

Si la superficie del objeto tiene una parametrización conveniente, esta integral se puede aproximar adecuadamente como la sumatoria sobre los texels de la textura del siguiente modo

$$M = \rho \sum_{\text{texels } i} a_i c_i y_i = \rho \sum_{\text{texels } i} \mathbf{A}_i^y y_i, \tag{3.8}$$

donde $c_i = \mathbf{n_i} \cdot (0, 1, 0)$, y_i es la coordenada y del texel en el sistema de coordenadas del mundo, n_i es la normal unitaria almacenada para el n-ésimo texel, a_i es el área del texel, y $\mathbf{A}_i^y$ es el área del texel *proyectada* sobre la dirección y ($\cdot$ representa el producto escalar).

Centro de masa: El cómputo anterior puede utilizarse de manera análoga para computar el centro de masa del objeto, resultando en la siguiente ecuación

$$\mathbf{C}_M = \frac{1}{M} \int_{\mathcal{V}} r \; dm = \frac{\rho}{M} \int_{\mathcal{V}} r \; dv \tag{3.9}$$

Ésta la podemos expresar como integral de superficie y arribar a

$$\mathbf{C}_M = \frac{1}{M} \; [s_x, s_y, s_z]. \tag{3.10}$$

Tensor de inercia: Siguiendo un razonamiento análogo, obtenemos una expresión similar para el tensor de inercia. Por definición, el mismo sigue la expresión

$$\mathbf{I} = M \begin{bmatrix} \int_{\mathcal{V}}(y^2+z^2)\ dv & -\int_{\mathcal{V}} xy\ dv & -\int_{\mathcal{V}} xz\ dv \\ -\int_{\mathcal{V}} yx\ dv & \int_{\mathcal{V}}(x^2+z^2)\ dv & -\int_{\mathcal{V}} yz\ dv \\ -\int_{\mathcal{V}} zx\ dv & -\int_{\mathcal{V}} zy\ dv & \int_{\mathcal{V}}(x^2+y^2)\ dv \end{bmatrix}, \tag{3.11}$$

la cual, a partir de los términos definidos, puede ser expresada como:

$$\mathbf{I} = \begin{bmatrix} s_{y^2}+s_{z^2} & -s_{xy} & -s_{xz} \\ -s_{xy} & s_{x^2}+s_{z^2} & -s_{yz} \\ -s_{xz} & -s_{yz} & s_{x^2}+s_{y^2} \end{bmatrix}. \tag{3.12}$$

3.4. Acoplamiento Líquido→ Sólido

Cuando un determinado objeto está sumergido en líquido, su dinámica está definida por la superficie de contacto entre ambos. El método que presentamos es independiente de la técnica usada para implementar la dinámica del líquido. Ésta puede realizarse mediante una función parametrizada simple, como se hace comúnmente en los videojuegos Gonzalez-Ochoa (2016), por ondas de Gerstner, sistemas de partículas o incluso por funciones analíticas simples. El único requisito de nuestro algoritmo es poder recuperar el nivel de la superficie del líquido f para cualquier posición del espacio, es decir, poder calcular la profundidad de cualquier punto perteneciente a la superficie del objeto simulado.

3.4.1. Cómputo del volumen sumergido

Para computar la porción de volumen sumergido, se aplica la siguiente expresión explicada gráficamente en las figuras 3.4 y 3.5:

$$S_V = \int_{\mathcal{V}_{sub}} dv = \oint_{\partial\mathcal{V}_{sub}} \mathbf{n}\cdot\mathbf{r}ds, \tag{3.13}$$

con la precaución de integrar solamente sobre la porción de volumen sumergido $\mathcal{V}_{sub}$ y no sobre el volumen total del objeto $\mathcal{V}$. A diferencia de la ecuación 3.8, en este caso los límites de integración deben tomar en cuenta la superficie de líquido para computar el volumen sumergido de manera correcta.

Es sencillo notar que el volumen sumergido $\mathcal{V}_{sub}$ está encerrado entre la superficie del líquido y la superficie del objeto ubicado debajo de él, por lo tanto, $\mathcal{V}_{sub}$ puede ser calculado por integración entre estos dos límites. Además, se observa el uso de la TVA para proyectar correctamente el área de texel en la dirección del eje y. La figura 3.5 muestra dos escenarios con texels en diferentes orientaciones y cómo el cálculo de volumen tiene en cuenta sus respectivas orientaciones.

Es importante aclarar que los vectores normales a la superficie, codificados en la TVA, contienen la información de orientación de la superficie, por lo tanto, el volumen sumergido puede computarse directamente como:

$$S_V = \sum_{texels_{sub}\, i} -A_i^y (f_i - y_i), \tag{3.14}$$

donde f_i es la posición de la superficie del fluido en la posición del texel i; A_i^y es la componente vertical del vector área correspondiente al texel i ($A_i^y = a_i n_i^y$) computada a partir de la TVA; (x_i, y_i, z_i) es la posición del texel en coordenadas de mundo (almacenado en la TP) y $texels_{sub}$ es el conjunto de texels que se encuentran sumergidos por debajo del nivel del líquido, es decir, con $y_i < f_i$. Este proceso es mostrado en la figura 3.4: en la columna de líquido izquierda el volumen generado por el texel rojo y la superficie del líquido se restaría del generado por el texel verde. Es importante tener en cuenta que situaciones como la que se muestra en la columna de líquido de la derecha en la misma figura no son casos especiales, ya que la parte superior del objeto será descartada por la prueba que verifica si el texel está debajo o sobre la superficie del agua.

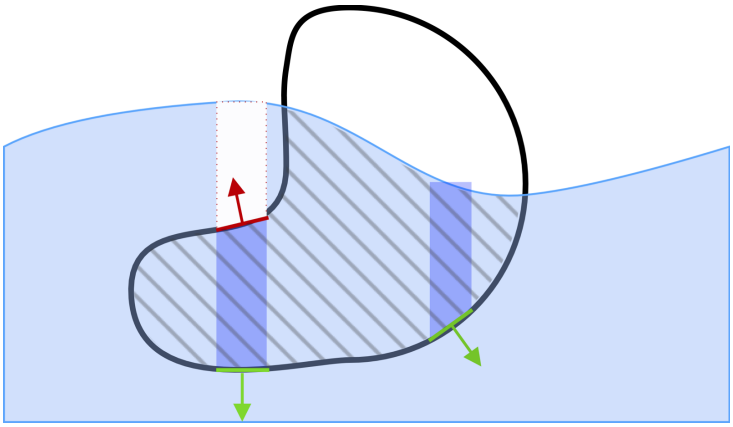


Figura 3.4: Interpretación geométrica del cómputo de volumen sumergido (región rayada): Los texels cuya normal tenga componente vertical negativa (verdes) contribuyen al volumen sumergido total. Por otro lado, los texels cuya normal tenga componente vertical positiva (rojo) restan volumen del total. Repitiendo ese cómputo y calculando la sumatoria total de todas las contribuciones de los texels calculamos una aproximación del volumen del sólido sumergido.

3.4.2. Cómputo de flotabilidad

El centro de flotabilidad se define como el centro de masa del volumen de fluido desplazado por un cuerpo sumergido y es el punto de aplicación de la fuerza de empuje que éste recibe. Otra fuerza que típicamente interviene es la atracción gravitacional debido a la masa del objeto que actúa hacia abajo aplicada desde centro de gravedad del objeto. Las magnitudes relativas de estas dos fuerzas determinan si el objeto se hunde completamente o solo parcialmente hasta que alcanza un punto de equilibrio estable, con el peso y el empuje de igual magnitud pero sentido contrario. En este caso decimos que el objeto flota. Es importante notar que estas fuerzas son siempre verticales y luego, paralelas entre sí.

Por otro lado, las posiciones relativas de los centros de flotación y masa determinan si esta flotación es estable o no (figura 3.6). La posición relativa también introduce un torque de rotación τ sobre el objeto flotante en caso que las fuerzas peso y empuje no sean colineales. Si el centro de masa se encuentra por debajo del centro de flotación (Figura 3.6a) el torque generado tenderá a alinear las fuerzas y el objeto restaurará su

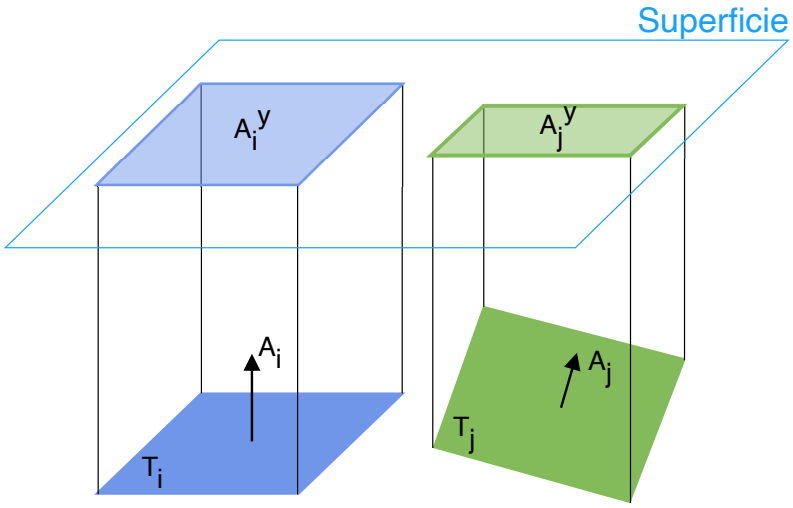


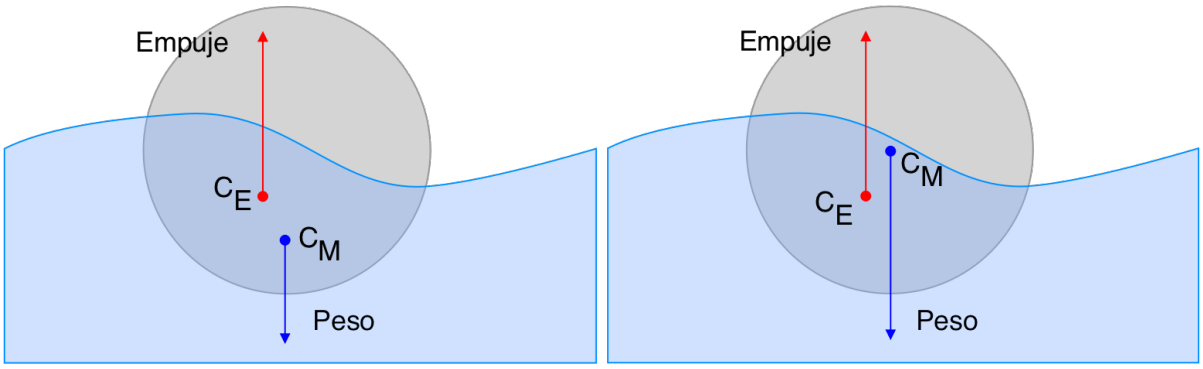
Figura 3.5: El volumen generado por un texel dado es computado como el producto de su profundidad y el área proyectada en la dirección vertical A_i^y, la cual se computa como el producto del área total del texel por el coseno del ángulo del vector normal y el eje vertical. A partir de TVA sería equivalente a $A_i^y = a_i n_i^y$

posición inicial, este caso de denomina *Flotación Estable*. Si, por el contrario, el centro de masa se sitúa por arriba del centro de flotación (Figura 3.6b), el torque generado no restaurará la posición original y, potencialmente, moverá al objeto hacia otra posición estable diferente a la inicial; esta condición se denomina *Flotación Inestable*. Claramente, la noción de estabilidad que discutimos aquí no está relacionada con la estabilidad de traslación, la cual es determinada por la diferencia de magnitudes de las fuerzas peso y empuje.

El centro de flotación es computado de forma similar a los magnitudes anteriores. Dada la TP y el modelo del fluido lo computamos a partir de los texels sumergidos

$$C_{sub} = \rho_a \sum_{texels_{sub}\, i} \mathbf{n}_i^T \mathbf{T}_i a_i, \tag{3.15}$$

donde $\mathbf{T}_i$ es la posición del texel i almacenada en la TP, a_i es el área del texel (almacenada en la TVA) y ρ_a la densidad del agua.



(a) Flotación estable. (b) Flotación inestable

Figura 3.6: Fuerzas actuando sobre un objeto flotante. La posición relativa entre los centros de masa y flotación determinan si la condición de flotación es estable o no.

Fuerza de Rozamiento: La fuerza de rozamiento es opuesta a la traslación de cada punto del objeto. La expresión para la magnitud de la fuerza de rozamiento debe ser evaluada para todos los puntos del objeto en contacto con el líquido, es decir, la superficie del objeto sumergida y se calcula como

$$D_i = \frac{1}{2}\rho C_d A_i^{\mathbf{V}_i} V_i^2,$$

dónde C_d es el coeficiente de rozamiento, comúnmente determinado de forma experimental (depende del material, rugosidad, micro-geometría, etc), ρ es la densidad del líquido, $A_i^{\mathbf{V}} = \mathbf{A}_i \cdot \mathbf{V}/|V|$ es el área de la superficie ocupada por el texel proyectada según $\mathbf{V}$, que a su vez es la velocidad relativa del texel en coordenadas de mundo respecto al fluido circundante. En este cómputo se debe tener la siguiente precaución: la velocidad de desplazamiento total es la suma de la velocidad debida a la traslación y la velocidad debida a la rotación del objeto en el punto correspondiente al texel i, además se debe tener en cuenta la velocidad del fluido en ese punto. Un caso extremo sería si el objeto se moviese con la misma magnitud y sentido que el líquido sin generar rozamiento. Es necesario remarcar que la velocidad de traslación tiene la misma magnitud y sentido en cada punto del objeto. Sin embargo, la velocidad de cada punto debido a la rotación

depende de la distancia respecto al eje de rotación $\mathbf{r}_i$. La velocidad de cada texel debido a la rotación se calcula como $\omega \times \mathbf{r}_i$, donde ω es la velocidad angular del objeto. La fuerza de rozamiento resultante es computada a partir de la suma de todas las fuerzas calculadas para cada texel. Una alternativa más eficiente para calcular la fuerza de rozamiento es computar una expresión en función de la velocidad, tanto de rotación como de traslación. De esta manera, se evita el cálculo por texel de las fuerzas y la sumatoria total, eliminando una operación de *reducción*[1] que es, claramente, la operación menos escalable en GPU de toda la solución.

3.5. Actualización de las magnitudes físicas

La simulación física de un cuerpo rígido involucra la solución de las siguientes ecuaciones diferenciales de primer orden

$$\frac{d}{dt}\mathbf{Y}(t) = \frac{d}{dt}\begin{bmatrix} x(t) \\ R(t) \\ P(t) \\ L(t) \end{bmatrix} = \begin{bmatrix} v(t) \\ \omega(t) * R(t) \\ F(t) \\ \tau(t) \end{bmatrix}, \tag{3.16}$$

donde x es la posición del objeto, R su rotación, P es la cantidad de movimiento, F es la fuerza resultante aplicada al objeto, τ es el torque actual y L es el momento angular Witkin and Baraff (1997). Estas ecuaciones son resueltas en nuestra implementación de manera acorde usando un esquema de integración numérico de Euler, pero para aplicaciones más generales podrían usarse métodos más precisos y numéricamente estables Eberly and Shoemake (2004); Millington (2010). En nuestro caso, la implementación resultó ser lo

[1] Una operación de reducción consiste en un algoritmo que convierte un conjunto de múltiples datos en un conjunto de menor cantidad utilizando un operador en cada elemento

suficientemente estable y precisa dados los requerimientos de las aplicaciones de tiempo real.

Capítulo 4

Simulación de Flotabilidad en Objetos Permeables

En este capítulo se presentan los resultados más importantes en la representación y simulación del comportamiento de objetos cuya dinámica de flotación puede cambiar a lo largo del tiempo. Esta dinámica puede ser debido a que los objetos sean de material permeable, o a que el líquido puede penetrar y moverse en su interior.

4.1. Grafo de Líquido

En el caso de objetos permeables, como por ejemplo un muñeco de peluche, el agua se difunde hacia el interior del objeto modificando las propiedades físicas del mismo. El cambio más notorio, por ejemplo, es el aumento de masa total debido a la materia líquida absorbida. Para modelar estos casos proponemos una extensión del método presentado en el capítulo anterior incorporando una nueva estructura de datos llamada *Grafo de líquido* (GL). Esta estructura es un típico grafo geométrico utilizado para aproximar la distribución de líquido dentro del objeto y simular el movimiento de éste a través de la estructura interna teniendo en cuenta su geometría. Matemáticamente, el GL se define

como

$$G_L = (N, E),$$

donde N es un conjunto de vértices o nodos N_i representando el espacio disponible dentro del objeto que puede ser llenado de líquido, ya sea espacio hueco (una habitación dentro de un barco) o espacio distribuido dentro del material (tela o felpa) y está definido por

$$N = \{N_i\},$$

$$N_i =< \mathbf{x_i}, c_i, l_i >,$$

donde $\mathbf{x_i}$ es la posición del nodo en coordenadas de objeto, c_i es el volumen de líquido que éste puede almacenar antes de saturarse (capacidad) y l_i el volumen de líquido presente en el nodo, siendo estos últimos expresados en litros.. Asimismo,

$$E = \{< N_i, N_k, p^g_{ik} >\}$$

es el conjunto de pares de nodos N_i y N_k tal que sus espacios representados están conectados y p^g_{ik} es la permeabilidad promedio del material que se encuentra entre los espacios representados por los nodos.

La figura 4.1 muestra posibles grafos de líquido para un modelo hueco del *Stanford Bunny*. Se puede observar fácilmente que los nodos tienen diferentes tamaños representando sus distintas capacidades. El movimiento del líquido dentro del objeto es modelado a partir del flujo entre los nodos del grafo.

4.1.1. Construcción automática

El *Grafo de Líquido*, si es relativamente simple, puede crearse de manera supervisada (manual) por un artista o creador del modelo. Una alternativa es implementar un algoritmo para la generación automática del grafo. Para esto, se requiere de un algoritmo de

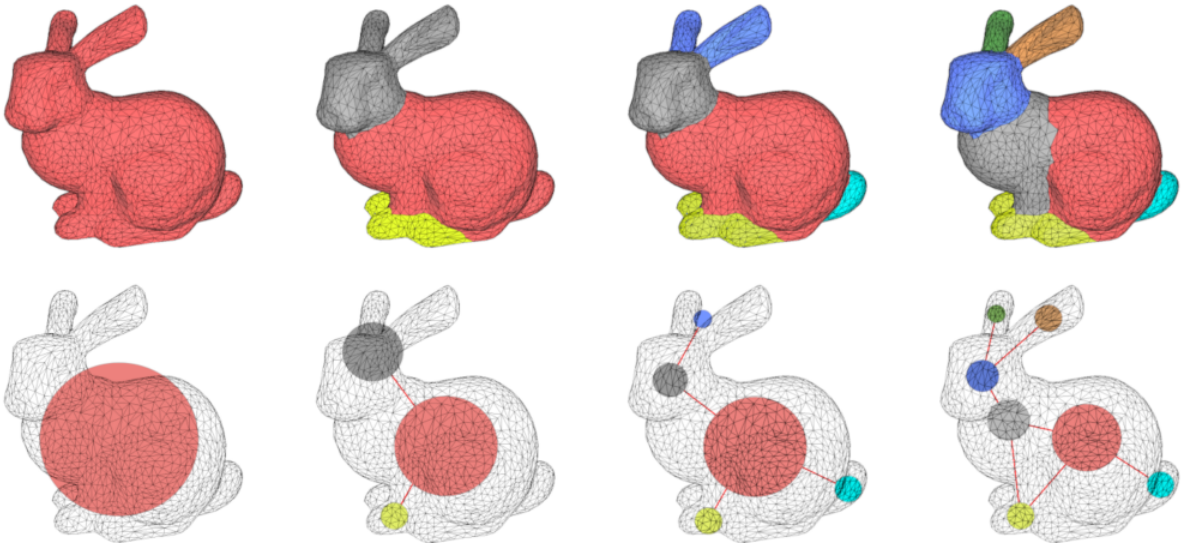


Figura 4.1: Construcción del *Grafo de Líquido*: segmentación de malla y creación de nodos para diferente cantidad de sub-mallas. Los colores indican el nodo correspondiente a cada sub-malla.

segmentación de mallas poligonales para particionar el modelo en los diferentes volúmenes que lo componen.

1. Inicialmente el algoritmo calcula una segmentación de la malla poligonal de entrada de la siguiente manera:

   a) Primero se computa la función SDF (Shape Diameter Function) para todos los vértices. Esta función escalar, definida para cada punto de una superficie cerrada, aproxima una medida del diámetro de la superficie en una vecindad del punto. En la figura 4.2 se observa la función SDF para un modelo en distintas poses.

   b) El segundo paso es aplicar un algoritmo de agrupamiento (clustering) para determinar las zonas de la malla con un diámetro calculado similar. La cantidad de grupos, dependiendo del algoritmo usado, puede ser un parámetro de entrada.

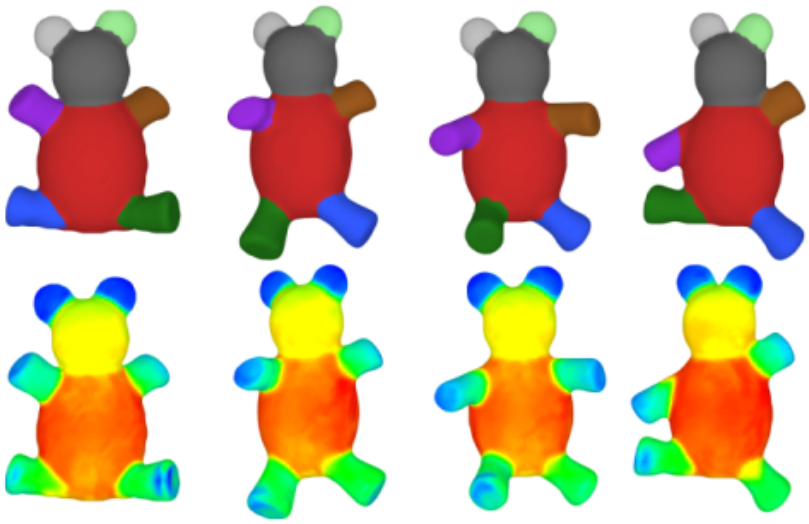


Figura 4.2: Segmentación de mallas utilizado el descriptor SDF. Puede observarse que la segmentación es suficientemente robusta ante cambios de pose (Fuente: CGAL).

   c) Por último, se aplica un algoritmo basado en *graph-cut* para agrupar vértices de la misma clase generando las diferentes sub-mallas (ver Yaz and Loriot (2018)).

2. Posteriormente, el algoritmo calcula, para cada sub-malla generada por el algoritmo de segmentación, una esfera centrada en el centro geométrico de la sub-malla tal que minimice el error cuadrático entre la superficie de la esfera y los vértices de la sub-malla.

3. Finalmente, con todos los nodos generados se agrega una arista al grafo entre dos nodos en el caso de que las sub-mallas representadas por ellos sean adyacentes, es decir, compartan algún vértice o arista en la malla original.

En la figura 4.1 se muestra un mismo modelo segmentado de manera automática con diferentes cantidad de grupos (clusters) en el algoritmo de agrupamiento, además se muestra el grafo generado a partir de cada una de las segmentaciones.

4.1.2. Actualización de la dinámica del fluido

Una vez que el grafo está creado y las capacidades de cada nodo establecidas junto a la conectividad entre ellos, ya sea de forma manual o automática con el método descripto en la sección anterior, se puede evaluar la dinámica del líquido dentro del objeto. El método está propuesto para trabajar en tiempo real, por lo tanto es viable repetirlo en cada cuadro de una animación.

La primera etapa consiste en computar la cantidad de líquido que ingresa al objeto, debido a la permeabilidad del material del objeto o simplemente por alguna situación particular, como una fisura o rotura en su superficie.

Una implementación *naive* es agregar líquido de manera arbitraria a los nodos del grafo a lo largo del tiempo de la simulación. Sin embargo, con el objetivo de incrementar el realismo y proveer interactividad en la simulación, nuestra solución define una nueva textura 2D mono-canal llamada *Textura de Permeabilidad*. Esta textura puede compartir la misma parametrización que las texturas definidas previamente, por lo tanto no se necesita de un mapeo UV ad-hoc. En cada texel se almacena un valor de permeabilidad por área por tiempo y durante la simulación se calcula la cantidad de líquido que ingresa en el área cubierta por él.

En la figura 4.3 (superior izquierda) se muestra la textura de permeabilidad aplicada a un modelo de un barco. Las zonas rojas indican mayor nivel de permeabilidad y las blancas impermeabilidad. Esta textura puede actualizarse dinámicamente permitiendo la simulación de situaciones completamente dinámicas, como por ejemplo una batalla de barcos interactiva.

Ingreso de Líquido

Con el objetivo de modelar el ingreso de líquido en modelos permeables, se utiliza una aproximación basada en el trabajo de Muskat sobre la Ley de Darcy Muskat and Wyckoff

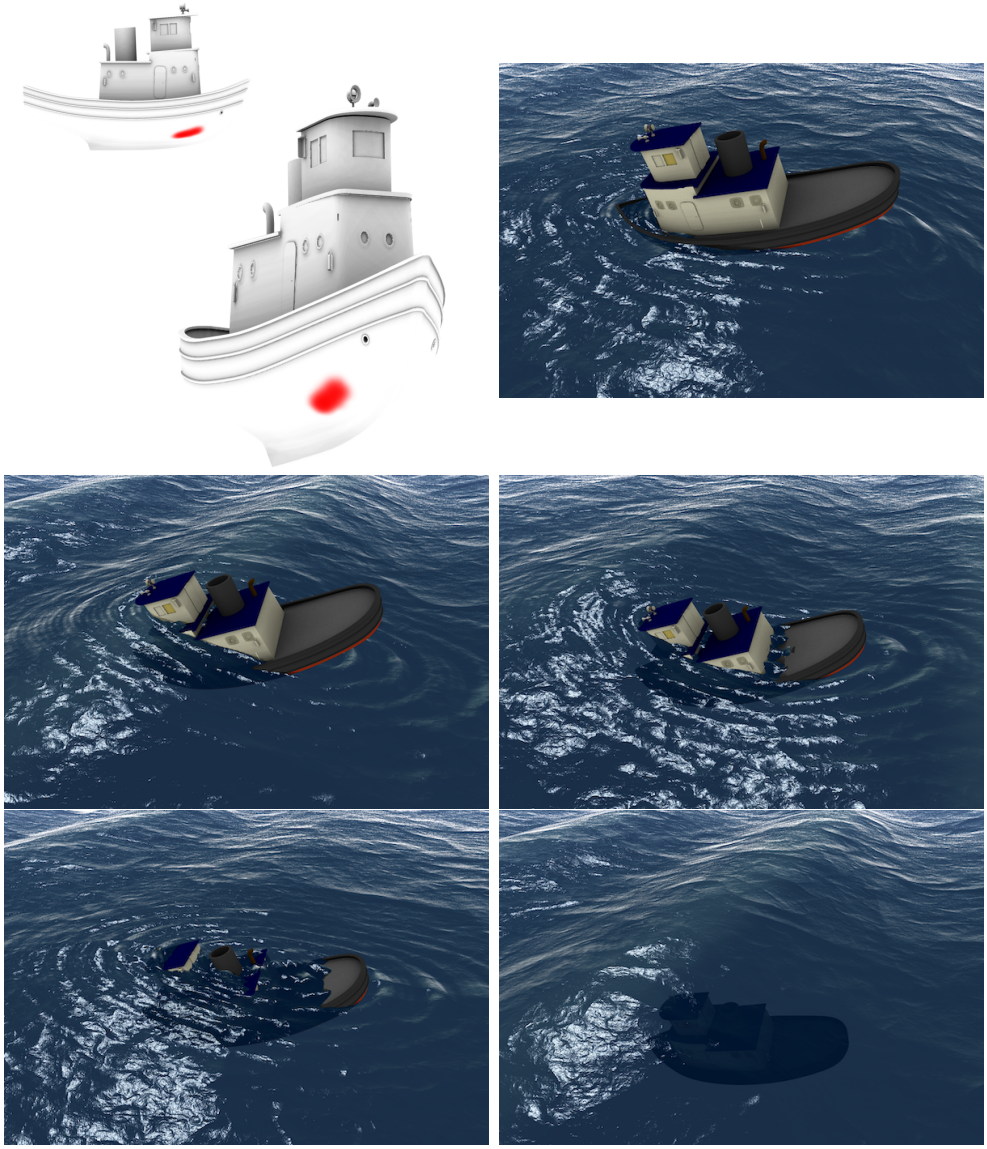


Figura 4.3: Cuadros de la simulación de un barco hundiéndose. La imagen superior izquierda muestra la *Textura de Permeabilidad* indicando con color rojo las zonas de mayor permeabilidad.

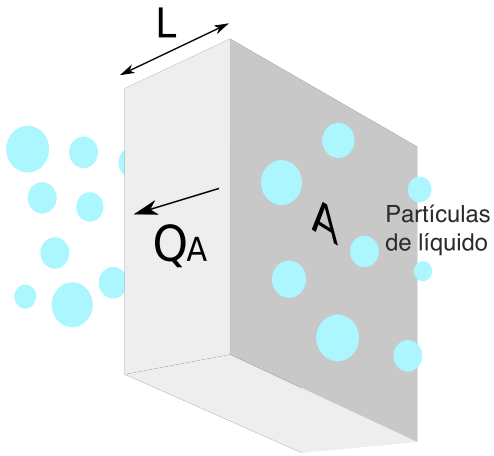


Figura 4.4: Para modelar la absorción de líquido se utiliza la *Ley de Darcy* que establece que el flujo de líquido (ϕ_A) a través de un área puede ser calculado como una función de la superficie del área (A), el largo de la sección (L) y las propiedades del material sólido y del líquido como la viscosidad y la permeabilidad.

(1937) que consiste en

$$\Phi_A = -\frac{C_{PM} A \Delta\rho}{\mu L},$$

donde Φ_A es el flujo por unidad de tiempo, C_{PM} es la permeabilidad del medio definida de manera global para todo el modelo o por texel en la *Textura de Permeabilidad*, A es el área del flujo, μ es la viscosidad del líquido y $\Delta\rho$ es la diferencia de presión en el área de contacto medida a lo largo de una sección de longitud L (Figura 4.4). El modelo puede ser simplificado bajo la consideración de que la superficie del objeto es de longitud infinitesimal, lo cual resulta en un infinitesimal $\Delta\rho$. Por lo tanto, se puede asumir que la relación entre $\Delta\rho$ y L es una constante finita, proporcional a la presión del líquido en ese punto. Bajo esta suposición, se define $P(d)$ como la presión en el punto de interfase entre la superficie y el líquido, que depende linealmente de la profundidad:

$$\frac{\Delta\rho}{L} \xrightarrow{L\to 0} P(d),$$

y, por lo tanto, puede usarse la siguiente expresión simplificada:

$$\Phi_A = -\frac{C_{PM}A}{\mu}P(d).$$

Con la información geométrica del texel, provista por las texturas TP y TVA, el modelo del fluido, la *Textura de Permeabilidad* y las estructuras de datos del modelo, se computa la cantidad de líquido por texel que ingresa al modelo. De acuerdo a la cercanía del texel con los nodos del grafo, se selecciona el nodo donde se acumulará la cantidad de líquido absorbida por el texel. Esto se realiza en dos pasadas en la GPU; primero se computa el líquido ingresado por texel y después se realiza una operación de reducción sumando los totales de cada nodo.

Distribución Interna del Líquido

Para resolver la dinámica del fluido contenido dentro del objeto, en cada cuadro de la simulación se utiliza una aproximación basada en la teoría de vasos comunicantes, la cual probó ser efectiva y práctica para situaciones típicas. Cada nodo en el grafo puede transferir líquido a cualquier otro nodo siempre que exista un camino de vasos comunicantes que los unan. Además, deben tenerse en cuenta las respectivas capacidades, cantidad de líquido y la dirección del flujo debido a la gravedad. Un caso especial (no considerado aquí) se genera con agua y algunos materiales porosos, debido a que la tensión superficial y el efecto capilar pueden generar movimientos contrarios a la dirección de la fuerza gravitatoria.

Una implementación directa de la dinámica de fluidos de un líquido dentro de un modelo puede generar soluciones complejas y no aptas para tiempo-real. En la Figura 4.5, por ejemplo, se muestran dos grafos cada uno de los cuales constituido por tres nodos con igual capacidad pero con diferentes cantidades de líquido cada uno y a diferentes alturas. La implementación directa consistiría en trasvasar líquido del nodo B al nodo C y, en una iteración siguiente, el nodo A trasvasaría líquido al nodo B. Interpretando en detalle esa situación, es fácil determinar que las partículas de líquido son indistinguibles para

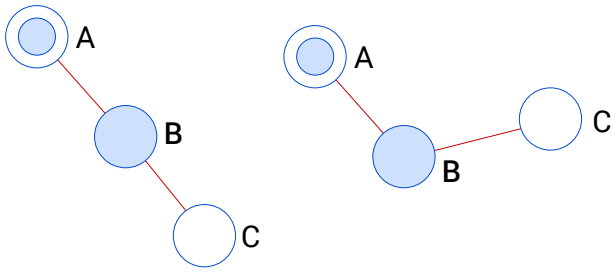


Figura 4.5: Dos grafos simples, cada uno con tres nodos de igual capacidad conectados por aristas equivalentes. En todos los casos, el nodo A tiene líquido en su interior, el nodo B está completo mientras que el nodo C está vacío. Claramente, la transferencia de líquido parte de A hasta C, pasando por B.

nuestro propósito, por lo tanto, sería más eficiente y directo que el nodo A la trasvase líquido directamente al nodo C. De esta manera evitaríamos un proceso de dos (o más) etapas.

Este proceso es implementado por un algoritmo que en cada cuadro de la animación invoca al método *buscarNodoDeDescarga* para cada nodo del grafo. El método es presentado en pseudo-código en el algoritmo 1, el cual, dado un nodo inicial n_s, busca un posible *nodo de descarga* n_d al cual le trasvasará líquido. La búsqueda se hace de manera recursiva a través de los nodos adyacentes al inicial hasta encontrar al menos un nodo de descarga. Si durante la búsqueda se encuentra un nodo que está más alto que el nodo fuente, el proceso se detiene ya que ese nodo funciona como barrera de energía y el líquido no puede pasar a través de él considerando su energía potencial. Si durante la búsqueda se encuentra un nodo con capacidad de recibir líquido, éste es seleccionado y retornado para la función. Si después de una búsqueda exhaustiva no se encuentra ningún nodo capaz de recibir líquido, el proceso se detiene y el método retorna *Nodo Nulo*, lo cual significa que el nodo inicial no puede transferirle líquido a ningún otro nodo en todo el sistema. Repitiendo este proceso en cada cuadro y para cada nodo, alterando de manera aleatoria la búsqueda de los nodos adyacentes, se asegura una solución al problema del movimiento del líquido dentro del objeto. Es claro que, cuanto mayor cantidad de veces se compute este algoritmo, la solución se acercará más a la solución real al problema. Es importante

aclarar que deben computarse la posiciones de los nodos en coordenadas de mundo, ya que el objeto puede estar rotado.

Una vez encontrado n_d, se computa el volumen de líquido para trasvasar del nodo inicial n_s a n_d, usando la Ley de Poiseuille Batchelor (2000), que establece lo siguiente:

$$\Phi_{n_s \to n_d} = \frac{\pi \Delta p r^4}{8 \mu L},$$

donde $\Phi_{n_s \to n_d}$ es el flujo de líquido desde n_s hasta n_d, Δp es la diferencia de presión entre los nodos, r el radio del segmento, L la longitud del camino $n_s \to n_d$ y μ la viscosidad del líquido. En el caso que se quiere simular un proceso de difusión, Δp debería ser reemplazado por la diferencia de concentración. El algoritmo podría ser modificado mínimamente para incluir todos los nodos de descarga posibles y no sólo uno de ellos, simplemente continuando el proceso recursivo de manera exhaustiva hasta visitar todos los nodos del grafo. En este caso, además, debería implementarse un algoritmo para priorizar los nodos que se usarán en la transferencia dependiendo de su posición.

4.1.3. Magnitudes físicas del Grafo de Líquido

Análogamente al caso anterior, es necesario computar las magnitudes físicas (*e.g.*, masa, centro de masa y tensor de inercia) del grafo, ya que serán utilizadas para modelar el sistema compuesto de sólido y líquido. Dada la naturaleza del grafo, las magnitudes físicas serán computadas como variantes discretas de las ecuaciones presentadas en la sección 3.3. La masa total del líquido absorbido puede computarse a partir del grafo como

$$M_G = \rho_f \sum_N l_i,$$

donde la suma se realiza sobre los nodos del grafo N, ρ_f es la densidad del fluido modelado y l_i es el volumen de líquido contenido en el nodo N_i. El centro de masa del líquido dentro

del objeto es:

$$\mathbf{C_G} = \frac{\rho_f}{M_G} \sum_N l_i \mathbf{x_i}.$$

Finalmente, el tensor de inercia (TI) $\mathbf{I_G}$ es calculado con la versión discreta de la ecuación previamente descripta

$$\mathbf{I_G} = \begin{bmatrix} {I_G}^{xx} & {I_G}^{xy} & {I_G}^{xz} \\ {I_G}^{yx} & {I_G}^{yy} & {I_G}^{yz} \\ {I_G}^{zx} & {I_G}^{zy} & {I_G}^{zz} \end{bmatrix}, \tag{4.1}$$

$$\begin{aligned} I_G^{xx} &= \sum_N (y_n^2 + z_n^2) m_n \quad I_G^{yy} = \sum_N (x_n^2 + z_n^2) m_n \\ I_G^{zz} &= \sum_N (x_n^2 + y_n^2) m_n \quad I_G^{xy} = I_G^{yx} = -\sum_N x_n y_n m_n \\ I_G^{yz} &= I_G^{zy} = -\sum_N y_n z_n m_n \quad I_G^{xz} = I_G^{zx} = -\sum_N x_n z_n m_n, \end{aligned} \tag{4.2}$$

dónde $m_n = \rho_f * v_n$ es la masa del nodo n y (x_n, y_n, z_n) es la posición del nodo respecto al sistema de coordenadas en el cual se está computando el TI.

4.2. Cómputo del modelo físico

Para simular correctamente el comportamiento dinámico del objeto sumergido junto con el agua ingresada, debemos combinar las magnitudes de ambos sistemas físicos, el sólido y el líquido (Figura 4.6). Las magnitudes físicas combinadas se calculan como:

$$M_s = M + M_G$$

$$\mathbf{C_s} = M\mathbf{C} + M_G\mathbf{C_G}$$

$$\mathbf{I_s} = \mathbf{I} \otimes \mathbf{I_G},$$

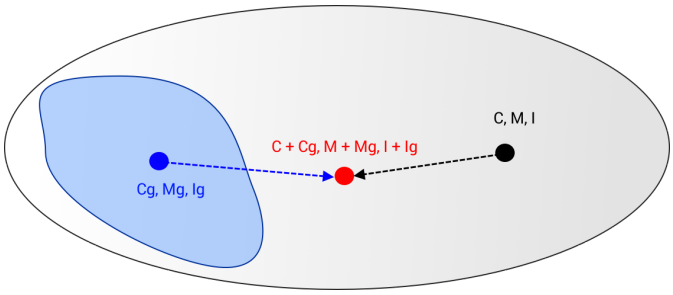


Figura 4.6: Cálculo del centro de masa (C) y tensor de inercia (I) resultantes del sistema sólido-líquido. Las magnitudes correspondientes al líquido absorbido son computadas a partir del grafo. Notar que el signo + no denota suma algebraica, ya que los centros de masa y los tensores de inercia deben ser combinados apropiadamente como se explica en el texto.

dónde M_s es la masa total del sistema, $\mathbf{C_s}$ es el centro de masa total del sistema e $\mathbf{I_s}$ es el tensor de inercia de todo el sistema. La operación de composición de tensores de inercia $\otimes$ consiste en la suma aritmética de los tensores calculados respecto al mismo sistema de coordenadas. Calcular el TI del sistema líquido respecto a un punto determinado del espacio es trivial a partir de las ecuaciones mencionadas anteriormente. En el caso del sólido es sensiblemente más complejo, ya que el TI está precalculado. Ésto se puede resolver fácilmente utilizando el teorema de los ejes paralelos Haas and Verschoyle (1928); Abdulghany (2017). A partir de ambos TI calculados respecto a un mismo punto en común se pueden sumar obteniendo el centro de masa total ($\mathbf{C_s}$) del sistema completo.

Algoritmo 1: *buscarNodoDeDescarga* - Algoritmo para buscar un nodo posible de recibir líquido desde un nodo inicial

Input:

- n_s: Nodo inicial cuyo nodo de descarga estamos buscando
- n_c: Nodo actual siendo evaluado.
- nivel: Nivel relativo de n_c respecto a n_s.
- $perm_{min}$: Permeabilidad mínima entre n_s y n_c.

Output: Un nodo de descarga para n_s.

if $n_c.visitado$ **then**
    return *null*;
end
$n_c.visitado = True$
if *nivel* $>$ *0* **then**
    return *null*; // Barrera de energía
end
if $(n_c \neq n_s)$ *&&* $noEstaLleno(n_c)$ **then**
    return n_c;
end
for *each node* n_i *adjacent to* n_c **do**
    $e = aristaEntreNodos(n_c, n_i)$
    $\Delta_l = n_i.pos.y - n_c.pos.y$
    $m = min(perm_{min}, e.permeabilidad)$
    $n_d = buscarNodoDeDescarga(n_s, n_i, level + \Delta_l, m)$;
    if $n_d \neq null$ **then**
        return n_d
    end
end

Capítulo 5

Materiales Húmedos

La apariencia visual de algunos materiales cambia significativamente cuando están mojados. Típicamente los colores se alteran aumentando su saturación y decrementando su intensidad, resultando en colores más vívidos y oscuros. En los siguientes dos capítulos introduciremos una técnica basada en física para modelar y renderizar cambios de apariencia en superficies de materiales absorbentes bajo condiciones de humedad. En este capítulo, específicamente, desarrollaremos un método para resolver la interacción entre el medio líquido y la superficie del objeto utilizando información presente en la malla poligonal. Posteriormente presentaremos un método para simular fenómenos que ocurren en la superficie de un objeto húmedo como difusión, infiltración y evaporación. En el Capítulo 6 extenderemos un modelo conocido para explicar el cambio de apariencia de materiales húmedos y propondremos una implementación adecuada para aplicaciones de tiempo real.

5.1. Introducción

El método propuesto consiste en un *pipeline* de procesamiento de tres etapas para modelar y renderizar materiales húmedos. Inicialmente, la malla poligonal es pre-procesada

y se crea una estructura de datos especial. Posteriormente, durante la simulación se calcula la dinámica del líquido absorbido y los resultados se actualizan. Finalmente, la malla es renderizada en tiempo real utilizando las estructuras de datos generadas durante la simulación, este método se explicará en detalle en el capítulo siguiente. El *pipeline* de procesamiento, mostrado en la figura 5.1, se detalla a continuación:
En la etapa de pre-procesamiento:

1. Procesar la malla poligonal y crear una textura especial denominada *Textura de Posición-Área* (TPA) la cual codifica, para cada texel, su posición junto al área de superficie cubierta por él, ambos en coordenadas de objeto. Este proceso se realiza una única vez por modelo. La textura TPA en una combinación de las dos texturas presentadas en el Cápítulo 3. En este caso, no se utiliza el vector-área para el cómputo de la simulación, por lo tanto no se incluye en las estructuras generadas. En el caso que se simule tanto la flotabilidad como el cambio de apariencia pueden usarse el par de texturas más generales TP y TVA, sustituyendo la TPA.

Durante el ciclo de simulación:

2. Actualizar el estado del fluido de acuerdo al modelo utilizado para simularlo.

3. Actualizar la *Textura de Líquido* (TL), la cual almacena la cantidad de líquido en la superficie del modelo, resolviendo la colisión líquido-objeto a partir de la TPA. Este proceso consiste en:

   a) Computar el ingreso del líquido a partir del modelo de Darcy.

   b) Computar el proceso de difusión intra-objeto de acuerdo a la ley de Fick a partir de la creación de un *Campo de densidad de líquido* (CDL) para poder realizar la simulación en el espacio 3D.

   c) Computar la evaporación del líquido.

4. Renderizar la escena utilizando un modelo acorde para las superficies húmedas.

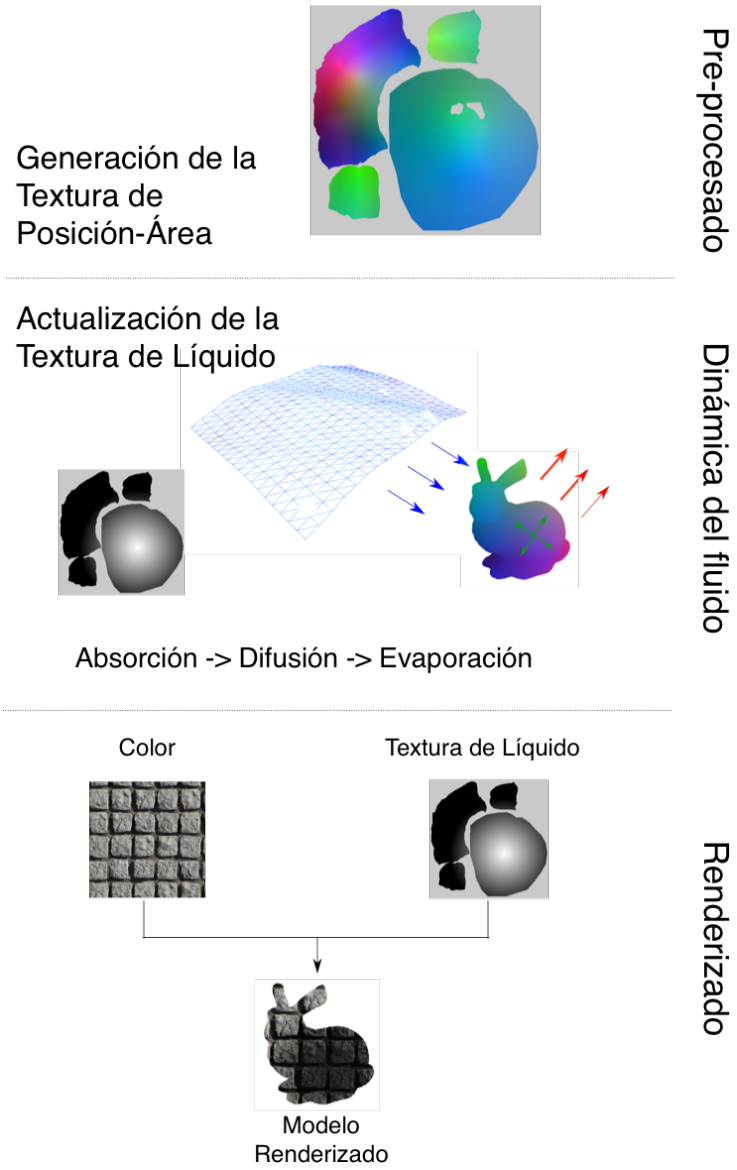


Figura 5.1: *Pipeline* de procesamiento. En primera instancia, la geometría es pre-procesada y la TPA es creada. A continuación, se actualiza la simulación de líquido absorbido de acuerdo a los fenómenos simulados actualizando la textura de densidad de líquido. Finalmente, la geometría es renderizada teniendo en cuenta la información obtenida en las etapas anteriores.

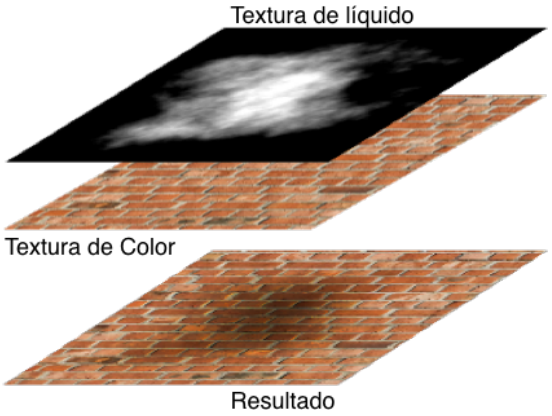


Figura 5.2: Se define una *Textura de Líquido* (TL) que almacena la cantidad de líquido por unidad de área en cada texel. Este textura es actualizada de acuerdo a la interacción del modelo con el líquido presente en la escena de manera interactiva utilizando las posiciones de los texels almacenados en la *Textura de Posición-Área*. Esta textura será utilizada para el renderizado, durante la etapa final del *pipeline*.

Si bien nuestra implementación utiliza una grilla 2D para simular el líquido, el método puede ser adaptado a otras técnicas de modelado de fluidos, como sistemas de partículas o grillas 3D. Estos detalles serán discutidos detalladamente en los Cápítulos 7 y 8. En la Tabla 5.1 presentamos de manera tabular todos los parámetros que nuestro método utiliza para la simulación.

5.2. Estructuras de datos

Con el objetivo de simular líquidos y su interacción con sólidos de manera precisa y escalable, al utilizar el espacio de textura como recurso se necesita establecer una correspondencia bidireccional entre este espacio 2D (donde se tiene almacenada la información de líquido absorbido) y el espacio del mundo 3D (donde el objeto está inmerso y los fenómenos suceden propiamente). Para esto definimos un nuevo tipo de textura, que llamamos *Textura de Posición-Área* (TPA) que permite almacenar información espacial y geométrica de cada texel. Esta textura es una simplificación que surge de combinar las TP y la TVA presentadas en el Capítulo 3.

Parámetro	Definición	Unidad
f_{env}	Condiciones ambientales	–
k_T	Temperatura ambiente	$[^\circ K]$
k_h	Humedad Ambiente	$[\%]$
k_w	Velocidad del viento	$[m/s]$
μ	Viscosidad del líquido	$[Pa \cdot s]$
k_L	Tasa de evaporación del líquido	$[m^{-1} \cdot s^{-1}]$
C_M	Permeabilidad del material	$[m^2]$
D	Coeficiente de difusión	$[m^2/s]$
r	Rugosidad del material	–
$n = \mu_1/\mu_2$	Proporción de los IOR líq-aire	–
a_λ	Color del material (componente de la reflexión difusa)	–
σ_λ	Color del líquido	–

Tabla 5.1: Parámetros usados en la simulación en unidades SI. Estos parámetros serán definidos detalladamente en el capítulo actual y en el próximo.

Además, para poder simular la dinámica del líquido, es necesario definir una estructura de datos acorde para almacenar la cantidad de líquido absorbida en cada punto de la superficie del objeto. Para esta estructura, también se utilizará la parametrización de la malla para el mapeo de textura y se almacenará la información necesaria en una textura mono-canal llamada *Textura de Líquido* (TP), en donde cada texel almacena la cantidad de líquido por unidad de área en su posición en la superficie del modelo (ver Figura 5.2). La ventaja de esta metodología, además de las conocidas del mapeo clásico de texturas, es que el proceso de renderizado puede implementarse usando técnicas de multi-texturas de manera eficiente. Efectos conocidos, como infiltración y evaporación, al ser de naturaleza local pueden ser implementados fácilmente con esta estructura. Es importante reiterar que el mapeo UV debe ser biyectivo para poder tener una relación biunívoca entre texels y posiciones sobre la superficie del modelo.

Sin embargo, algunos cómputos sobre la propagación del líquido requieren ser hechos en coordenadas de mundo 3D, por ejemplo la solución al problema de la difusión requiere acceder a una vecindad de un determinado punto en la superficie del objeto. El problema de usar una parametrización 2D de una superficie 3D radica en que se pierde la continuidad, es decir, existen puntos contiguos en la superficie original que no lo son en la superficie

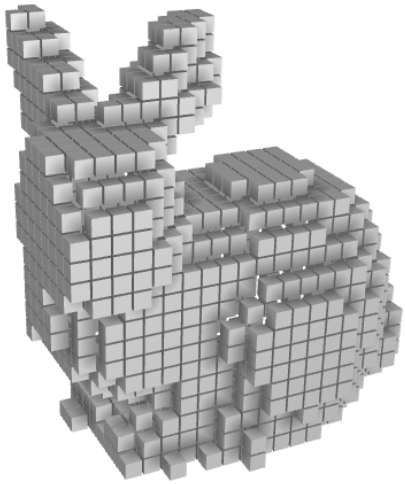


Figura 5.3: *Voxelización* del modelo *Stanford Bunny*. Solo se muestran los *voxels* que contienen porciones de la superficie del modelo, es decir, texels de la textura de posiciones. El resto de los voxels, que no contienen geometría, no participan en el proceso de difusión. Cada *voxel* contiene los texels situados dentro del volumen correspondiente y pueden ser contiguos en el espacio de textura. Sin embargo, con la estructura descripta, puede resolverse el problema de la difusión de manera rápida.

parametrizada. En el dominio del mapeo de texturas clásico, al conjunto de estos puntos se los denomina *seams* (costuras). Este problema requiere la creación de otra estructura de datos especial que permita accesos de vecindades en el mundo 3D de manera eficiente, sin perder las ventajas que las texturas proveen. En particular, definiremos un campo escalar 3D $L : R^3 \rightarrow R$ llamado *Campo de Densidad de Líquido* (CDL) tal que almacene la cantidad de líquido en cada punto del espacio formado por la superficie del modelo. Una posible implementación de esta función escalar es a partir de una discretización del espacio donde el modelo está embebido. Ésto genera una grilla tridimensional, regular y discreta de tamaño $\mathbf{S_F} = (N, M, O)$ *voxels*, (por *Volume Cells*). Cada *voxel* almacenará la cantidad de líquido en la porción de espacio ocupado por él y podemos definir una relación entre los *texels* de la TL y los *voxels* del CDL como sigue. Sea $\mathbf{P_T}$ la posición en

coordenadas del objeto del texel T, la posición del *voxel* correspondiente es $\mathbf{P_V}$, donde

$$\mathbf{P_V} = \mathbf{P_T} \dot{/} \mathbf{S_V}, \tag{5.1}$$

y $\mathbf{S_V}$ es el tamaño de los *voxels* regulares definido como

$$\mathbf{S_V} = \mathbf{S_M} \dot{/} \mathbf{S_F}, \tag{5.2}$$

la operación $\dot{/}$ es la división real de vectores componente a componente y $\mathbf{S_M}$ el tamaño del *Bounding Box* del modelo. Los *voxels* de la grilla que tengan al menos un texel en su interior serán *definidos* y participarán en la simulación, el resto serán caracterizados como *no-definidos*. El resto de ellos no serán parte de la simulación, excepto en el caso que se quiera simular la difusión en el interior, lo cual dado que el método simula cambios de apariencia no sería de utilidad. En el caso de querer simular el cambio en los parámetros físicos de un objeto ante la absorción de líquido ver los Capítulos 3 y 4. En la figura 5.3 puede verse una posible *voxelización* del modelo *Stanford Bunny*. De esta manera, a partir de la TPA y el CDL, se crea una correspondencia entre el espacio 3D y el espacio de textura.

El CDL permite resolver tres aspectos importantes de la simulación. Primero, como dijimos, el mapeo UV no es continuo en el espacio. Dos texels pueden ser vecinos en la superficie del objeto pero no en la textura, dificultado la implementación eficiente de la simulación directamente en el espacio de textura. Segundo, el área cubierta por los texels no es la misma en todos los casos, la parametrización genera deformaciones en la superficie, produciendo discontinuidades de muestreo. Estos dos tipos de discontinuidades, inherentes al mapeo de texturas, son resueltos de manera escalable a partir de la grilla regular que la *voxelización* provee. Por último, es posible compartir un mismo CDL para diferentes objetos, permitiendo la implementación de transferencia de líquido por difusión

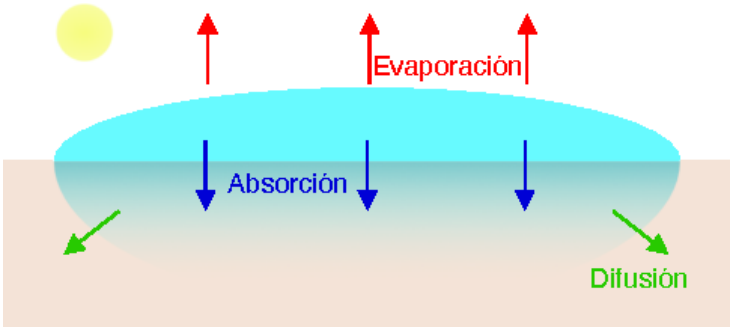


Figura 5.4: Los tres procesos simulados en el método propuesto. Cada uno de ellos depende de diversas variables como la permeabilidad del medio sólido, viento, temperatura, presión del líquido debido a la profundidad y viscosidad. Las flechas indican la dirección del flujo de líquido de acuerdo a cada fenómeno.

entre diferentes mallas poligonales, por ejemplo, las huellas de un pie húmedo sobre una superficie seca o una esponja húmeda sobre una mesa de madera.

5.3. Dinámica del líquido en material poroso

La absorción, difusión y evaporación son claramente los efectos más notorios relativos al líquido en contacto con materiales absorbentes o porosos. Por esta razón, debido a que nuestro método está enfocado en proponer un método visualmente realista en tiempo real, presentaremos una solución para su simulación. En esta sección presentaremos los modelos que se utilizarán para simular los diferentes fenómenos y cómo implementarlos de manera escalable a partir de las estructuras presentadas anteriormente. Los fenómenos que se simularán están descriptos en la figura 5.4.

5.3.1. Absorción

Con el objetivo de modelar el ingreso de líquido en modelos permeables, se utiliza una aproximación basada en el trabajo de Muskat sobre la Ley de Darcy Muskat and Wyckoff

(1937) presentada en el Capítulo 4 y repetida aquí por completitud, que consiste en

$$\Phi_A = -\frac{C_{PM} A \Delta\rho}{\mu L}, \tag{5.3}$$

donde Φ_A es el flujo por unidad de tiempo, C_{PM} es la permeabilidad del medio definida de manera global para todo el modelo o por texel en la TPM, A es el área del flujo, μ es la viscosidad del líquido y $\Delta\rho$ es la diferencia de presión en el área de contacto medida a lo largo de una sección de longitud L (Figura 4.4). El modelo puede ser simplificado bajo la consideración de que la superficie del objeto es de longitud infinitesimal, lo cual resulta en un infinitesimal $\Delta\rho$. Por lo tanto, se puede asumir que la relación entre $\Delta\rho$ y L es una constante finita, proporcional a la presión del líquido en ese punto. Bajo esta asunción, se define $P(d)$ como la presión en el punto de interfase entre la superficie y el líquido, que depende linealmente de la profundidad:

$$\frac{\Delta\rho}{L} \xrightarrow{L\to 0} P(d), \tag{5.4}$$

y, por lo tanto, puede usarse la siguiente expresión simplificada:

$$\Phi_A = -\frac{C_{PM} A}{\mu} P(d). \tag{5.5}$$

5.3.2. Evaporación

La evaporación es el proceso de cambio de fase de estado líquido a gaseoso por parte de una sustancia líquida y el transporte asociado del vapor resultante Hall and Hoff (2009). Si la tasa de evaporación es más rápida que el proceso de absorción decimos que el material está en un proceso de secado. En un caso ideal, por ejemplo agua en un plato, la capa fina de aire en contacto directo con el agua está totalmente saturada con vapor de agua, por lo tanto, su humedad es de 100 %. Si la humedad relativa de las capas de agua más alejadas es menor a 100 %, el gradiente de concentración genera un flujo de vapor que es

transportado hacia fuera de la superficie reduciendo la concentración de vapor de agua en las capas cercanas a la superficie, lo cual produce, como consecuencia, que el proceso de evaporación continúe. Eventualmente, este proceso continuo producirá la completa evaporación del líquido, dejando la superficie seca.

De acuerdo a Hall y Hoff Hall and Hoff (2009), la *Tasa de Evaporación* e puede ser modelada como

$$e = -\frac{p_{w0} D_w \rho_w M}{R\tau} \frac{dH}{dx}, \tag{5.6}$$

donde H es humedad relativa fraccionaria, ρ_w es la densidad del líquido, M la masa molar, D_w es la difusividad del líquido en el aire, R es la *constante universal de los gases* y τ la temperatura ambiental.

A partir de este modelo se desprende que la tasa de evaporación depende principalmente de tres factores: las *condiciones ambientales*, la *permeabilidad del material* del objeto y la *tasa de evaporación* del líquido; por ejemplo, bajo las mismas condiciones atmosféricas, una roca mojada se secará más lentamente que una prenda de tela deportiva. Por otro lado, el alcohol se evaporará más rápido que el agua debido que las fuerza inter-moleculares son más débiles. Por lo tanto, proponemos un modelo simplificado basado en estos tres factores. La cantidad de líquido trasladándose desde la superficie del objeto hacia el aire en un determinado instante puede ser modelado de acuerdo a la siguiente ecuación

$$\frac{\partial \phi_E}{\partial t} = f_{env} k_L C_M A, \tag{5.7}$$

asumiendo que todos los valores son constantes durante el intervalo de integración

$$\Phi_E^{\Delta t} = \int_0^T d\phi_E \approx f_{env} k_L C_M A \Delta t, \tag{5.8}$$

donde $\Phi_E^{\Delta t}$ es el líquido evaporado total en un período de tiempo Δt, k_L es la tasa de evaporación que depende exclusivamente de las fuerzas inter-moleculares del líquido, A es el área de la superficie en contacto directo con el aire, C_M es la permeabilidad del material y

f_{env} es una función modelando las condiciones ambientales en el intervalo correspondiente. En nuestra implementación, esta función es simplemente una multiplicación de tres términos lineales que modelan la temperatura del ambiente (k_T), la humedad (k_h) y la velocidad del viento (k_w). Nuestra implementación asume que las condiciones ambientales son las mismas para los diferentes objetos en la escena. Sin embargo éstas pueden variar en tiempo y espacio libremente durante la simulación.

5.3.3. Difusión

La difusión es el proceso por el cual la materia es transportada, debido al movimiento aleatorio de las moléculas, desde un punto del espacio a otro. Este proceso puede ser modelado usando la primera *Ley de Fick* Fick (1995); Crank (1956) que establece que, cuando existen regiones con diferentes concentraciones de materia, se genera un flujo desde las regiones con mayor concentración a regiones con concentración más baja. Matemáticamente, la *Ley de Fick* es expresada como

$$\mathbf{J} = -D\nabla\phi,$$

donde $\mathbf{J}$ es el vector de difusión del flujo, D es el coeficiente de difusión, que depende totalmente del material y ϕ un campo escalar representando la concentración.

5.4. Implementación

En esta sección presentaremos detalles de la implementación de nuestro modelo a partir de los fundamentos físicos descriptos anteriormente. En particular, proveeremos detalles sobre la interacción líquido-sólido, la simulación de los fenómenos de absorción, evaporación y difusión utilizando las estructuras definidas. Además, se discutirán algunos aspectos numéricos del cómputo del modelo.

La distribución de líquido dentro del objeto será modelada a partir de la TL que presentamos anteriormente. Esta textura será actualizada en todos los cuadros de la simulación a partir de la simulación de los fenómenos de absorción, evaporación y difusión. Los procesos de absorción y evaporación de líquido en la superficie del objeto son de implementación directa y escalable a partir de la información geométrica de los texels, provista por la TVA. Como nuestro modelo está basado en la cantidad de líquido y área de cada texel, se pueden realizar las cómputos directamente sobre cada uno de ellos. En el caso de la absorción, usaremos la expresión de la ecuación 5.5 para agregar el volumen de líquido correspondiente en cada iteración. Análogamente, la evaporación es computada también por cada texel utilizando la ecuación 5.8 para eliminar líquido de la superficie del objeto.

5.4.1. Simulación del proceso de difusión

La implementación del proceso de difusión es un desafío debido a que, por su naturaleza, afecta puntos espacialmente adyacentes de la malla poligonal. Una posible solución *naive* sería implementar la simulación directamente sobre los polígonos de la malla, eliminando el problema de las discontinuidades. Este enfoque ha sido utilizado en varios trabajos previos como se describió en el Cápítulo 2. Sin embargo, todos esos métodos carecen de la escalabilidad necesaria para ser implementados en soluciones de tiempo real. Además, la cantidad de polígonos en los modelos diseñados para este tipo de aplicaciones tiene una limitante por cuestiones de eficiencia. Por esa razón estos modelos hacen uso intensivo de las texturas para detalles finos. Implementar un efecto visual en el dominio de la geometría generaría resultados visualmente pobres y de poca escalabilidad.

La simulación del proceso de difusión (ver figura 5.5) comienza con la creación del CDL. La cantidad de líquido en cada *voxel* se computa usando el método de *Interpolación de Shepard* (Shepard (1968) como se detalla a continuación (ver figura 5.6). Sea $h_i = h(\boldsymbol{x_i})$, con $i = 1, 2, ..., N$ el conjunto de texels pertenecientes a la TL con posiciones $\boldsymbol{x_i}$ localizadas

dentro del volumen del *voxel* $V_{\boldsymbol{x}}$, podemos computar el valor de líquido del *voxel* $V_{\boldsymbol{x}}$ del CDL como:

$$V_{\boldsymbol{x}} = \frac{\sum_{i=1}^{N} w_i(\boldsymbol{x}) h_i}{\sum_{i=1}^{N} w_i(\boldsymbol{x})}, \tag{5.9}$$

donde

$$w_i(\boldsymbol{x}) = \frac{1}{d(\boldsymbol{x}, \boldsymbol{x_i})^p}, \tag{5.10}$$

d es la distancia euclídea y p es un número real positivo. En nuestra implementación se utilizó el valor 1.0.

Con el objetivo de evitar inestabilidades numéricas debido a divisiones por valores cercanos a cero, debe garantizarse que un texel no coincida con la posición del centro del *voxel*. La solución original del método de interpolación consiste es asumir que si existe un punto interpolante en la misma posición que el punto interpolado, este último tomará su valor. Si bien esto podría funcionar, se dificulta una implementación masivamente paralela, ya que deberíamos buscar el texel más cercano al centro del *voxel*. Debido a esto, nuestro método propone un solución eficiente que consiste en fijar una distancia mínima segura entre el texel y el *voxel*. En nuestra implementación, definimos este valor en función del tamaño del *voxel* como $Dist_{min} = 1/50 * \mathbf{S_V}$.

En el proceso de generación del campo escalar, cada texel acumula su contribución en dos ubicaciones diferentes dentro del mismo *voxel*, una para el numerador y otra para el denominador en la ecuación 5.9. Una vez creado el campo escalar, se procede a dividir el numerador y denominador calculados en la etapa anterior para obtener el valor final de cada *voxel*. A partir de esto, los diferentes *voxels* definidos en la estructura (al menos un texel en su interior) intercambian líquido entre sus *voxels* vecinos generando un flujo de líquido que depende de las cantidad de líquido distribuido en la vecindad y de los parámetros del material, como el coeficiente de difusión. Es importante aclarar que esta etapa preserva la cantidad total de líquido del sistema.

Dado un determinado *voxel* definido, se necesita calcular la cantidad de líquido en la próxima iteración que, intuitivamente, dependerá de la cantidad actual de líquido y del flujo tanto entrante como saliente debido a la naturaleza de la difusión. La cantidad de líquido de un determinado *voxel* en la siguiente iteración, puede computarse como

$$V_{\boldsymbol{x}}^{t+\Delta t} = V_{\boldsymbol{x}}^{t} + \Phi_{in} - \Phi_{out}, \tag{5.11}$$

donde Φ_{in} es la cantidad total de líquido entrante en $V_{\boldsymbol{x}}$ desde sus *voxels* vecinos y Φ_{out} es la cantidad de líquido difundido en la dirección opuesta.

Para implementar este cómputo, se define un operador local K implementado como una matriz de 3x3x3 elementos evaluado en la posición $\mathbf{x}$ de la *voxelización* como

$$K_{\mathbf{x}}(\mathbf{y}) = \begin{cases} 1 - D \cdot Count(\mathbf{x}) & \text{if } \mathbf{y} = \mathbf{x} \\ D & \text{if } \mathbf{y} \neq \mathbf{x} \text{ and } V_y \text{ is defined} \\ 0 & \text{otherwise} \end{cases} \tag{5.12}$$

donde D es el coeficiente de difusión del material, $Count(\mathbf{x})$ es la cantidad de *voxels* definidos en una vecindad de 3x3x3 definida alrededor del *voxel* en posición $\mathbf{x}$.

Luego, el valor de volumen de líquido en el *voxel* $V_{\mathbf{x}}$ en el tiempo $t + \Delta t$ está dado por:

$$V_{\boldsymbol{x}}^{t+\Delta t} = \sum_{y \in N_x} K_x(y) H^t(y)$$

donde $H^t(y)$ es el valor del *voxel* en el CDL en el tiempo t y posición y (figura 5.7).

Posteriormente a la simulación de la difusión de líquido sobre la superficie del modelo usando el CDL, la TL debe ser actualizada de manera acorde a la nueva distribución. Este proceso es implementado de manera análoga a la construcción del campo, usando la Interpolación de Shepard (ver Figura 5.6).

Dado un texel y su posición, se puede definir un cubo que lo contiene tal que sus ocho vértices sean los ocho *voxels* más cercanos a él. La idea fundamental es que el valor del texel sea computado a partir de un subconjunto de esos ocho *voxels* más próximos formados por los *voxels* definidos, es decir, que tienen un valor computado a partir de los texels en su interior y forman parte de la superficie del objeto.

Sea V_i, con $i = 1, 2, ..., N$, $1 \leq N \leq 8$ el conjunto de *voxels* definidos pertenecientes al cubo alrededor del texel T_{ij}. Entonces,

$$T_{ij}^{t+\Delta t} = \frac{\sum_{i=1}^{N} w_i(\boldsymbol{x}) V_i}{\sum_{i=1}^{N} w_i(\boldsymbol{x})}, \tag{5.13}$$

donde

$$w_i(\boldsymbol{x}) = \frac{1}{d(\boldsymbol{x}, \boldsymbol{x_i})^p}, \tag{5.14}$$

d es la distancia Euclídea y p un número real positivo.

En la solución propuesta consideramos que la interpolación de Shepard es particularmente útil debido a que puede ser implementada en paralelo, lo cual es fundamental para lograr soluciones lo suficientemente rápidas para aplicaciones interactivas. Además, este esquema de interpolación permite interpolar utilizando puntos con distribución irregular como es el caso de los texels en el espacio o los *voxels* definidos en la grilla del campo escalar.

Luego de finalizado todo el proceso, la TL queda actualizada de acuerdo a las condiciones ambientales, propiedades del material y estado del líquido, entre otras variables. A continuación, la textura es pasada a la siguiente fase en la cual se procederá al renderizado de la escena empleando un modelo acorde para las porciones de escena con mojados.

Como fue mencionado oportunamente en la definición del CDL, el método usa una grilla regular con tamaño dependiente del tamaño promedio del texel. Sin embargo, como

también fue notado, no existe un tamaño fijo de grilla que provea un balance perfecto en términos de precisión, distorsión y eficiencia ya que las texturas tienden a ser deformadas de manera no-uniforme a lo largo del todo el modelo.

La utilización del CDL permite resolver dos fenómenos importantes. En primer lugar, es que el mapeo de texturas es, en general, no contiguo en el espacio; esto significa que dos texels puede ser vecinos en la superficie del objeto pero no en el espacio de textura, dificultando el procesamiento en el espacio de textura. En segundo lugar, texels vecinos pueden cubrir diferente área en la superficie del modelo, debido a las deformaciones típicas del mapeo UV, generando discontinuidades de muestreo. Estos dos tipos de discontinuidades, inherentes la mapeo de texturas, son resueltos de manera escalable ya que el proceso de difusión es realizado sobre una grilla regular. Por otra parte, compartir el CDL entre diferentes objetos permite implementar la difusión entre mallas (ver figura. 7.13) haciendo posible el pasaje de líquido de un modelo a otro.

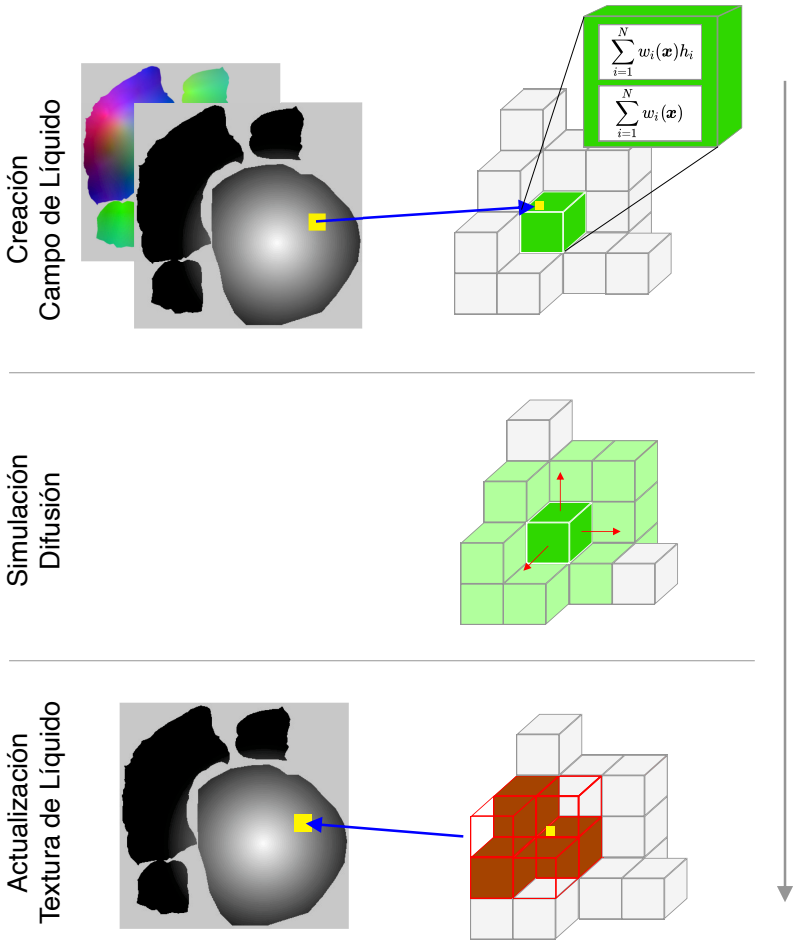


Figura 5.5: El proceso de simulación de la difusión a partir de la generación del CDL. Inicialmente, para cada texel de la TL se calcula el *voxel* correspondiente y se acumula el valor correspondiente tanto en el numerador como en el denominador. Seguidamente, para cada *voxel* se efectúa la división entre estos dos valores para hallar el valor correspondiente al *voxel* y se calcula la difusión hacia *voxels* vecinos. Por último se actualiza la TL de acuerdo al nuevo campo de líquido generado.

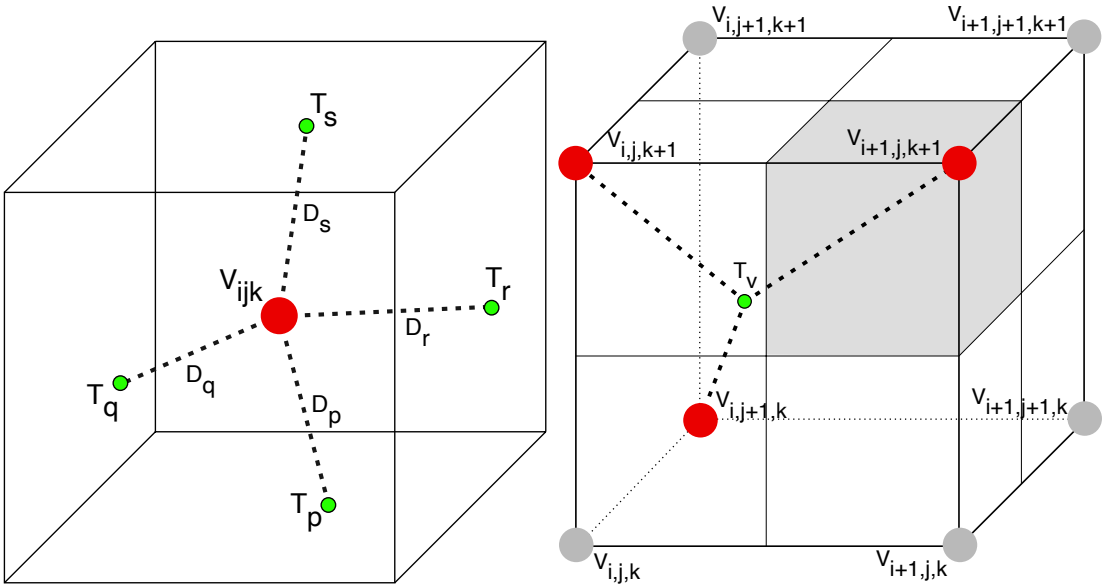


Figura 5.6: Izquierda: Con el objetivo de crear el CDL, el valor de cada *voxel* es interpolado a partir de los texels dentro del volumen correspondiente al voxel. El valor del *voxel* V_{ijk} es calculado por interpolación de los valores de los texels T dentro de su volumen y sus respectivas distancias al centro D. Los *voxels* sin texels en el interior de su volumen son *indefinidos* y no se utilizan durante el proceso de difusión. Derecha: Dado un texel y su respectiva posición, se puede definir un cubo generado por los ocho *voxels* más cercanos. El nuevo valor del texel se obtiene interpolando los valores de los *voxels* definidos (rojos). Para referencia espacial, el volumen sombreado en el *voxel* $V_{i+1,j,k+1}$ representan 1/8 del volumen de *voxel* total.

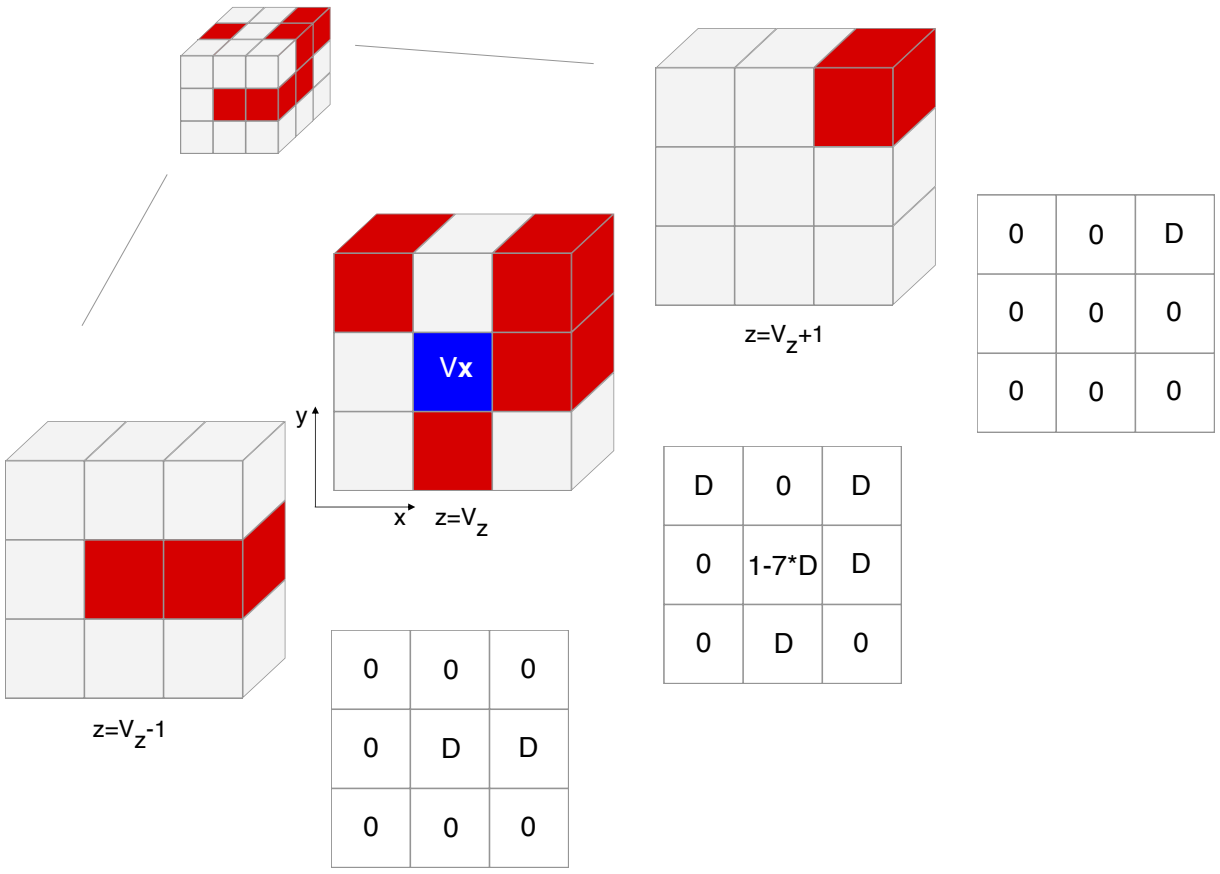


Figura 5.7: Ejemplo de la generación del operador local para simular la difusión en el CDL alrededor del *voxel* $V_{\mathbf{x}}$. El kernel computa la cantidad de líquido desde $V_{\mathbf{x}}$ hacia su vecindad y en sentido inverso. Los *voxels* rojos están definidos ya que tienen al menos un texel dentro de su volumen.

Capítulo 6

Modelo de Renderizado

El renderizado preciso de una determinada superficie requiere el modelado complejo del transporte de luz. En particular, los materiales absorbentes húmedos exhiben un particular efecto que consiste en el cambio de la apariencia del color percibido, específicamente en la saturación y el brillo. En este capítulo se describirá una explicación plausible de la causa de este fenómeno. Posteriormente, a partir de esta explicación, se propone un método de renderizado adecuado para obtener imágenes realistas de superficies de materiales húmedos en tiempo real.

6.1. Comportamiento de la luz en superficies húmedas

Con el objetivo de proponer una técnica de *renderizado* en tiempo real e inspirada en los fenómenos físicos subyacentes, este capítulo se fundamenta en el trabajo publicado por Lekner and Dorf (1988), el cual es una extensión de Ångström (1925). En ambos trabajos, los autores proponen un modelo basado en física que explica por qué los materiales absorbentes se perciben más oscuros cuando están mojados. De acuerdo a su estudio, una fracción de la luz reflejada de manera difusa en la superficie del objeto es reflejada

nuevamente hacia el interior del objeto en la interfase agua-líquido, generando un incremento de la reflexión total interna (ver Figura. 6.1) produciendo una apariencia visual más oscura. Además, los autores calculan la probabilidad de la reflexión total interna. El modelo asume que el objeto posee una superficie distinguible y rugosa.

Sin embargo, ciertos materiales formados por agregación de partículas (*e.g.* arena) también pueden ser modelados. El trabajo de Lekner es consistente con datos experimentales (Lekner and Dorf, 1988) y ha sido usado satisfactoriamente para interacción agua-arena (Zhang and Voss, 2006; Nolet et al., 2014), pavimento (Hendel et al., 2014) y aceite en arenisca (Muggeridge et al., 2014). En este capítulo extendemos el modelo para incluir líquidos coloreados. La influencia del color del líquido genera dos fenómenos importantes que deben ser modelados. Primero, la componente difusa reflejada por el objeto es filtrada por las partículas de color presentes en el líquido. Segundo, las partículas también reflejan la luz de manera difusa. Técnicamente, esto es descripto como fenómenos *in-scattering* y *out-scattering* (Pharr and Humphreys, 2010).

Siguiendo la figura 6.1, la reflexión interna total (Lekner and Dorf, 1988) puede ser calculada del siguiente modo:

$$\begin{aligned} A_\lambda &= (1 - R_{spec})(1 - a_\lambda)\sigma_\lambda[1 + a_\lambda p\sigma_\lambda^2 + a_\lambda^2 p^2 \sigma_\lambda^4 + ...] \\ &= (1 - R_{spec})(1 - a_\lambda)\sigma_\lambda \sum_{n=0}^{\infty} a_\lambda^n p^n \sigma_\lambda^{2n} \\ &= (1 - R_{spec})\frac{(1 - a_\lambda)\sigma_\lambda}{1 - pa_\lambda\sigma_\lambda^2}, \end{aligned} \tag{6.1}$$

donde R_{spec} es la reflectancia del líquido (respuesta especular), p es la fracción de luz reflejada nuevamente hacia la superficie del objeto, a_λ es la fracción de luz con longitud de onda λ reflejada difusamente por el material y σ_λ es la proporción de luz con longitud de onda λ transmitida por el líquido. Típicamente, a_λ es modelada por la componente difusa de la BRDF del material (seco) y, para aplicaciones en tiempo-real, representada por una tupla RGB. Si bien la componente difusa de la luz reflejada en la superficie

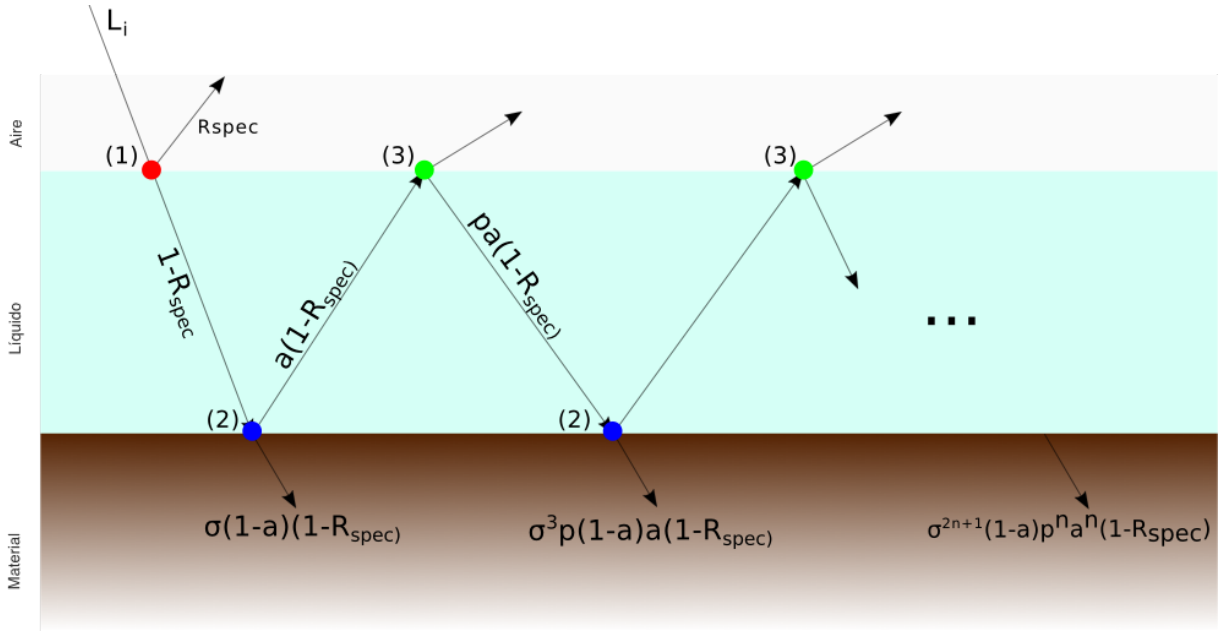


Figura 6.1: Modelo de la interacción de la luz con un material húmedo. Primero, una porción del rayo de luz entrante es reflejado en la interfase aire-líquido (punto rojo)(1). Seguidamente, la porción transmitida llega a la superficie del objeto y nuevamente es descompuesta en dos interacciones diferentes: una es absorbida por el objeto y la otra es reflejada. La proporción de luz reflejada determina el color difuso del material (albedo) (puntos azules)(2). Posteriormente, el rayo reflejado es nuevamente descompuesto en dos direcciones en la interfase líquido-aire (puntos verdes)(3), parte de la luz es trasmitida hacia fuera y otra parte es nuevamente reflejada hacia la superficie del objeto. Este proceso es repetido indefinidamente.

del objeto es comúnmente modelada por la *Ley de Lambert*, nuestro modelo puede ser integrado con modelos más específicos, como el propuesto por Merillou et al. (2000) para superficies porosas o el modelo de Oren and Nayar (1994) para superficies rugosas.

R_{spec} depende del ángulo de incidencia del rayo de luz y puede ser calculado usando la aproximación para CG de *Ley de Fresnel* propuesta por Schlick (1994):

$$R(\theta) = R(0) + (1 - R(0))(1 - \cos\theta)^5, \tag{6.2}$$

dónde $R(0)$ es la reflectancia con una incidencia normal y θ es el ángulo de incidencia.

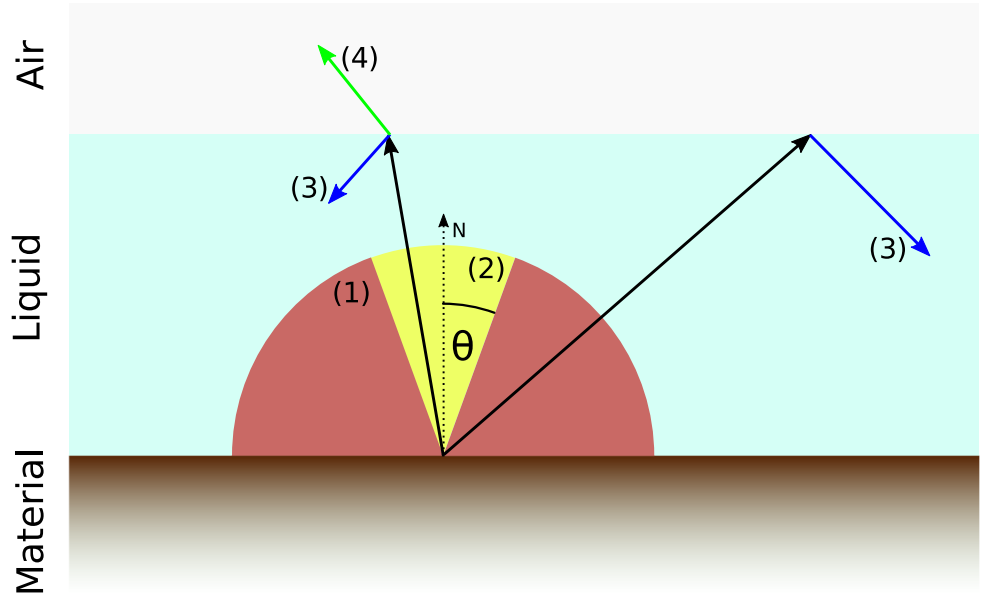


Figura 6.2: La proporción de luz reflejada en un ángulo mayor al *ángulo crítico* es totalmente reflejada hacia la superficie (área roja)(1). Una proporción de luz reflejada con ángulos menores al crítico es reflejada y otra transmitida. La relación entre una y otra está determinada por la ley de Fresnel (área amarilla). Los vectores azules (3) representan los rayos de luz que contribuyen a la reflexión total interna, mientras que los vectores verdes (4) representan los rayos que son reflejados desde la superficie húmeda y, potencialmente, llegan al observador.

Para evaluar p, es necesario determinar la proporción de luz reflejada en la interfase líquido-aire hacia la superficie del objeto. El comportamiento de la luz trasladándose entre medios con diferentes índices de refracción puede ser caracterizado a partir de las las ecuaciones de Fresnel. En el caso de la luz trasladándose desde un medio con mayor *Índice de Refracción* a otro con uno menor, existe un ángulo denominado *crítico* tal que todo rayo de luz con ángulo mayor será totalmente reflejado. El ángulo crítico es calculado como:

$$\theta_c = \arcsin(\frac{\mu_r}{\mu_i}) \tag{6.3}$$

donde μ_r y μ_i son los índices de refracción de los dos medios de la interfase. En el caso de interfase agua/aire, $\theta_c \approx 48°$ y $\mu_r < \mu_i$. Los rayos de luz con ángulo de incidencia menor

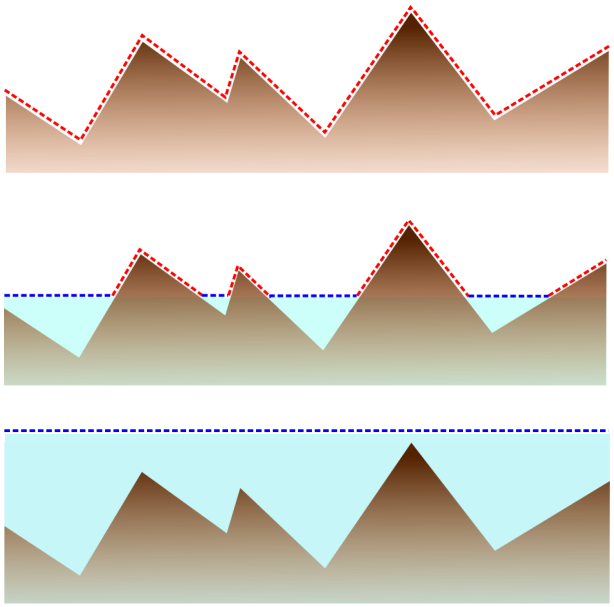


Figura 6.3: Material genérico bajo diferentes condiciones: seco, parcialmente mojado y saturado.

al ángulo crítico serán parcialmente reflejados y transmitidos. La cantidad total de luz reflejada en la interfase agua-líquido será la luz con ángulo de incidencia mayor al ángulo crítico y la porción de luz reflejada con ángulo menor al ángulo crítico (Figura 6.2).

Para superficies *lambertianas*, como la mayoría de los materiales absorbentes, la intensidad reflejada es proporcional al coseno del ángulo de emisión. Por lo tanto, la proporción de luz reflejada p en la interfase líquido-aire puede ser evaluada como:

$$p = \frac{\int_0^{2\pi} \sin(\theta)\cos(\theta)R(x,n)d\theta}{\int_0^{2\pi} \sin(\theta)\cos(\theta)d\theta} = \int_0^1 R(x,n)dx, \tag{6.4}$$

donde $x = \sin^2(\theta)$, $R(x,n)$ es la función reflectancia en la interfase entre los medios 1 y 2, con $n = \mu_1/\mu_2$. Como $R(x,n) = 1$ para todo $x > n^2$, entonces p puede ser simplificado como:

$$p = 1 - n^2 + \int_0^{n^2} R(x,n)dx. \tag{6.5}$$

Haciendo un cambio de variables (ver Lekner and Dorf (1988) y Stern (1964) para detalles), podemos expresar la expresión de la siguiente manera:

$$p = 1 - n^2[1 - \bar{R}(1/n)], \tag{6.6}$$

donde $\bar{R}(n)$ es la reflectancia media de una superficie isotrópicamente iluminada, calculada en Stern (1964) as:

$$\begin{aligned}\bar{R}(n) = \frac{3n^2 + 2n + 1}{3(n+1)^2} - \frac{2n^3(n^2 + 2n - 1)}{(n^2+1)^2(n^2-1)} + \\ \frac{n^2(n^2+1)}{(n^2-1)^2} \ln n - \frac{n^2(n^2-1)^2}{(n^2+1)^3} \ln \frac{n(n+1)}{n-1}.\end{aligned} \tag{6.7}$$

En general, debido a la micro-geometría de la superficie, los materiales absorbentes tienden a reflejar la luz de manera difusa, lo que significa que la *reflexión del cuerpo* es predominante sobre la *reflexión de la superficie*. El parámetro a_λ en la ecuación 6.1 modela la proporción de luz reflejada de manera difusa por el material, también llamado *albedo*, típicamente modelado a partir de la textura de color. En términos formales, esta respuesta es modelada por el término difuso de la BRDF.

6.1.1. Función de Distribución Reflectiva Bidireccional de la Dispersión Sub-superficial (BSSRDF)

La Figura 6.1 muestra que la luz refractada en la interfase aire-líquido entra en un punto y sale en otro diferente. La distancia entre los puntos de entrada y salida es importante porque determina la técnica de rendering necesaria para modelar adecuadamente el fenómeno (Akenine-Möller et al., 2018). Si esta distancia es menor a la escala del shading (tamaño relativo del pixel en el mundo), el proceso puede aproximarse con un modelo de *dispersión* (scattering) superficial local, el cual se modela típicamente con un término difuso en la BRDF, que no necesita tener en cuenta esta distancia. Sin embargo, cuando

la distancia entre la entrada y la salida de la luz es superior a esta escala, se requiere un modelo global de dispersión sub-superficial que modele adecuadamente el comportamiento de la luz (Akenine-Möller et al., 2018). Existen enfoques diferentes para modelar estos fenómenos y su uso, en general, queda determinado de acuerdo al tipo de material que se desea simular y la escala de observación. Algunos materiales como por ejemplo la leche o el mármol exhiben efectos de transporte de energía luminosa sub-superficial muy marcados y apreciables. Para generar imágenes realistas con este tipo de materiales, cuando la escala del shading es pequeña, se requieren técnicas mucho más complejas que el uso de una BRDF local, para tener en cuenta entre otras cosas la distancia entre la entrada y la salida de la luz.

Para describir el comportamiento de la energía lumínica en modelos con una contribución representativa de la dispersión sub-superficial, como es el caso que ocurre cuando una capa delgada de líquido cubre la superficie de un material húmedo, se requiere el uso de una función de distribución reflectiva bidireccional de la dispersión sub-superficial (BSSRDF, por *Bidirectional Scattering Surface Reflectance Distribution Function*). Más aún, si la capa de líquido cubre un material húmedo poroso (el cual inclusive podría ser inhomogéneo), se requieren más elementos para obtener una función de reflectancia plausible. En esta tesis aproximamos la interacción de la luz con el material húmedo cubierto por la capa de líquido con una BRDF usual; sin embargo, nada impide el uso de modelos de materiales más complejos combinados con la técnica presentada, de acuerdo a la relación de compromiso entre exactitud física y velocidad de cómputo. La BSSRDF $S(x_i, \vec{w_i}, x_o, \vec{w_o})$ describe la relación entre la radiancia reflejada en la posición x_o hacia la dirección $\vec{w_o}$, respecto de la luz incidente sobre la posición x_i desde la dirección $\vec{w_i}$. Reemplazando la BSSRDF en la ecuación general de reflectancia nos permite obtener:

$$L_o(x_o, \vec{w_o}) = \int_A \int_{2\pi} S(x_i, \vec{w_i}, x_o, \vec{w_o}) L_i(x_i, \vec{w_i})(\vec{n}\vec{w_i}) dw_i dA,$$

donde L_o es la iradiancia reflejada y L_i es la distribución de irradiancia incidente.

El cómputo de esta ecuación es computacionalmente muy elevado, aún para contextos *off-line*, por lo que se han ensayado en la literatura diferentes aproximaciones o simplificaciones. La aproximación por difusión del dipolo, planteada en Jensen et al. (2001), permite una simulación del transporte de luz sub-superficial de mayor eficacia. Este método calcula una componente difusa S del siguiente modo:

$$S_d(x_i, \vec{w_i}, x_o, \vec{w_o}) = \frac{1}{\pi} F_t(x_i, \vec{w_i}) R(||x_i - x_0||_2) F_t(x_o, \vec{w_o}),$$

donde F_t es el coeficiente de transmitancia de Fresnel y $R(r)$ es un perfil de difusión que depende del material y también puede ser función de la longitud de onda. La suposición subyacente a este modelo es que la distribución espacial de la energía luminosa en medios altamente dispersivos puede aproximarse con una función espacialmente isotrópica. El modelo de difusión del dipolo es computacionalmente accesible, pero no garantiza una mejora en la calidad del rendering en todos los casos. Una limitación, por ejemplo, es la que ocurre con materiales rugosos o particulados, dado que en esos casos la propiedad de isotropía geométrica no se cumple en la realidad y el modelo carece de posibilidades para representar la interacción luz-partícula, por lo que debe extenderse el modelo con elementos *ad-hoc*. Ejemplos de esto son la propuesta de Kimmel and Baranoski (2007, 2010) para superficies arenosas y la de Chen et al. (2015) para modelar la piel humana.

Para nuestro propósito utilizamos también la aproximación por difusión del dipolo porque nuestro foco central es el modelado de las reflexiones internas múltiples entre la interfase líquido-aire y la interfase sólido-líquido. Para dicha implementación seguimos la aproximación de d'Eon et al. (2007), utilizando funciones Gaussianas ponderadas para modelar el transporte sub-superficial de luz. Asumimos que el líquido conforma una capa delgada sobre la superficie del objeto humedecido, lo cual ocurre cuando el objeto está totalmente humedecido. Si la situación es "intermedia" (el objeto está parcialmente humedecido), parte de su superficie parcialmente seca está en contacto con el aire (ver Figura 6.3), por lo que se requiere a su vez una función general que contemple los estados

intermedios entre el comportamiento del material del objeto seco en contacto con el aire y el comportamiento del material rodeado por la capa delgada de líquido. Esta función general se puede definir como un material multi-capa (Weidlich and Wilkie, 2011), pero también resulta adecuada la solución adoptada en esta tesis, que consiste en modelar el comportamiento del material como una combinación convexa entre la BRDF del material seco y la BSSRDF planteada más arriba cuando el objeto está totalmente humedecido:

$$Fr(h) = \Gamma(h) \times BRDF(\cdot) + (1 - \Gamma(h)) \times BSSRDF(\cdot),$$

donde $\Gamma(h)$ es una función de blending entre $BRDF(\cdot)$ y $BSSRDF(\cdot)$, que representan la reflectancia del material seco y húmedo respectivamente y h es la humedad existente en el texel. La función $\Gamma(h)$ puede ser arbitraria, pero debe satisfacer una cantidad mínima de condiciones:

- $\Gamma(0) = 0$,
- $\Gamma(1) = 1$ y
- $\Gamma(h)$ es monótona no decreciente.

La función $\Gamma(h)$ entonces agrega un grado más de libertad al modelo de apariencia del material en condiciones variables de humedad, pudiendo ser desde una función muy sencilla (identidad, lineal a trozos), definida por el artista para modelar efectos *ad-hoc* por medio de funciones *Spline* implementadas con texturas 1D. En esta tesis investigamos fenomenológicamente varias funciones, como por ejemplo:

- $\Gamma(h) = \dfrac{log(C * h + 1)}{log(C + 1)}$,
- $\Gamma(h) = h^E$,
- $\Gamma(h) = \dfrac{1}{2}\left(\dfrac{\arctan(r * (h - 0{,}5))}{\arctan(r/2)} + 1\right)$,

siendo esta última una función sigmoidal que se comportó más satisfactoriamente en la mayoría de los casos. El parámetro libre r sirve como modelo de la rugosidad de la superficie. En un material más suave y pulido, un valor bajo de r genera una función $\Gamma(h)$ más cercana a la lineal que transiciona de una manera suave entre el comportamiento de la BRDF y la BSSRDF de acuerdo al valor de h, mientras que un valor alto de r genera una transición abrupta, lo cual es característico en los materiales rugosos.

6.1.2. Implementación

El modelo completo se implementó utilizando técnicas de aceleración por GPGPU basadas en las APIs de OpenCL y OpenGL. El diseño siguió las pautas de desarrollo de acuerdo al patrón "componente" (*Entity-Component-System*), utilizado en la gran mayoría de los motores gráficos Unity Technologies (2019) (ver Figura 7.1). De acuerdo a este patrón, un sistema se representa como una colección de entidades, donde cada una contiene un conjunto de componentes que definen su comportamiento. Las componentes pueden modificarse durante la ejecución, por lo que las entidades pueden cambiar su comportamiento en forma dinámica Nystrom (2014). De esa manera, se desacopla la representación de un objeto gráfico genérico de su descripción como interfase húmeda. Los valores de los parámetros utilizados para cada una de las simulaciones se muestran en la Tabla 5.1.

Para computar el transporte sub-superficial de energía luminosa, seguimos el modelo de Borshukov and Lewis (2003). Primero se rasteriza la irradiancia en una textura específica, utilizando las coordenadas del objeto como coordenadas para dicha textura. Esta irradiancia se suaviza con un kernel Gaussiano. Para evitar los artificios debidos a la escala no uniforme en la textura, se utiliza la textura posición-área introducida en capítulos anteriores, en la cual se define el área que representa cada texel. Suponiendo que la iluminación es aproximadamente constante, este proceso podría precomputarse para toda la escena y reutilizarse para toda la animación sin ser recalculado. Por otro lado, dada la

propiedad de separabilidad de los kernel Gaussianos, la capacidad aritmética de la GPU permite también realizarlo en tiempo real. Más aún, en casos particulares de materiales, este proceso puede hacerse más específico utilizando sumas de Gaussianas para modelar el comportamiento real, medido y cuantificado en condiciones experimentales controladas.

Capítulo 7

Resultados

En este capítulo se presentan los resultados relacionados con los diferentes métodos propuestos a lo largo de la tesis, detallando en cada uno los tiempos de cómputo requeridos para su procesamiento.

La implementación completa del pipeline, incluyendo las soluciones para flotabilidad y el renderizado de materiales húmedos, fue implementado utilizando técnicas de GPGPU a partir de las API OpenCL, para cómputo de propósito general, y OpenGL, para el renderizado. Los tiempos que se detallan fueron recolectados en un sistema Intel(R) Core(TM) i7 920 at 2.67GHz con una placa gráfica NVidia GeForce GTX 1080. Como plataforma de desarrollo se utilizó C++/Qt.

El diseño de la solución, mostrado en la figura 7.1, está basado en un diseño por componentes Wikipedia (2019). Los objetos gráficos están modelados por la clase *VisualObject* donde se abstraen las operaciones de dibujado. Las clases *MeshModel* y *Fluid* modelan visualmente los objetos gráficos basados en redes poligonales y grillas para fluidos respectivamente. Los objetos basados en mallas poligonales pueden tener una referencia a un componente de tipo *WettableComponent* que provee la funcionalidad de simular el cambio de apariencia del objeto a partir del fluido de la escena. De manera similar, un objeto basado en malla poligonal puede tener un componente de tipo *BuoyancyComponent* que

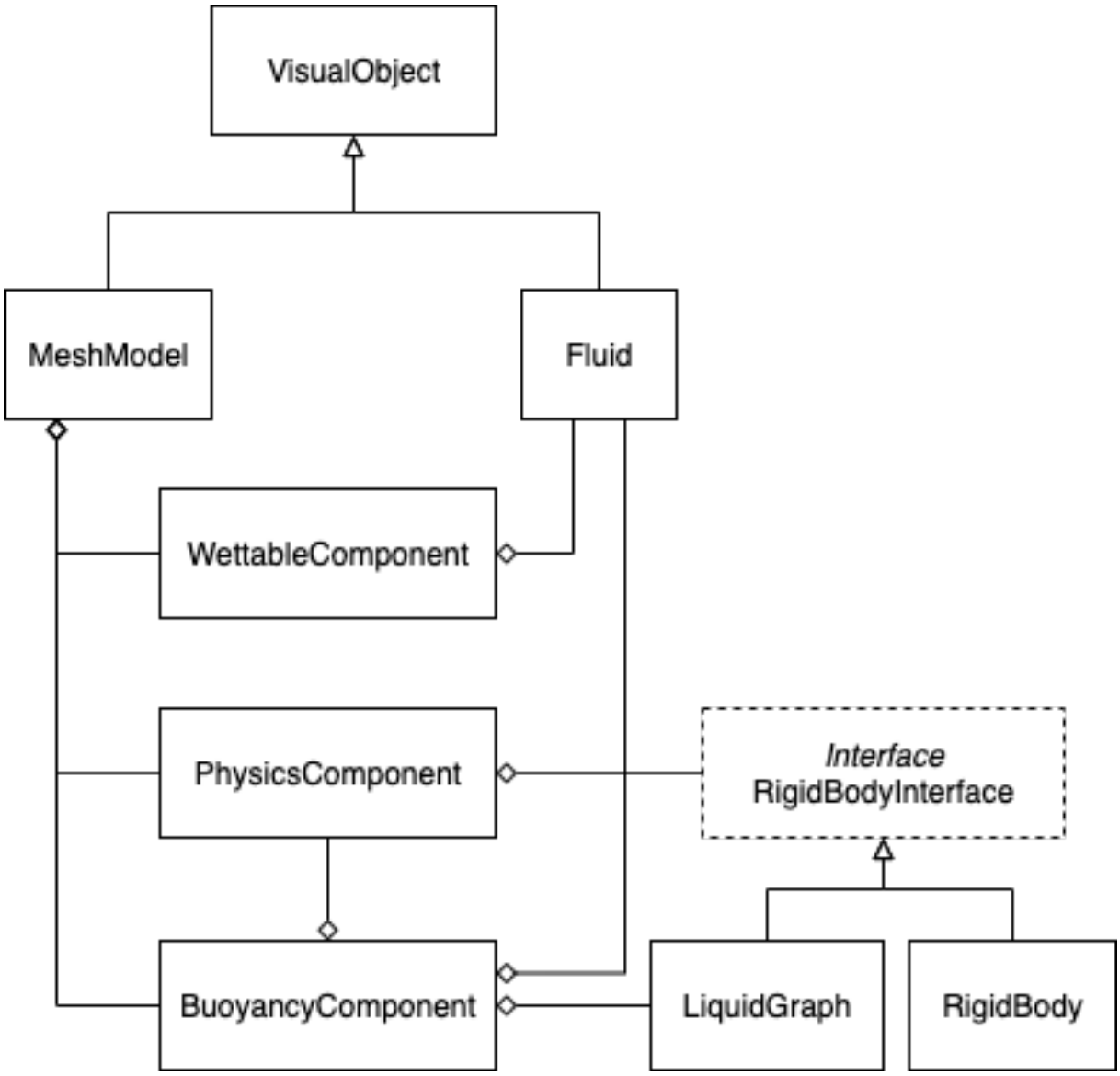


Figura 7.1: En la implementación del sistema se siguió un patrón de diseño basado en componentes para asegurar la modularización y extensibilidad de la solución, siguiendo el diseño de los motores de videojuegos modernos. Nota: Se mantuvieron los nombres de los módulos en inglés para mantener consistencia con el código fuente.

provee la funcionalidad necesaria para simular la flotabilidad. En el caso de simular un objeto permeable, este componente posee una referencia al grafo del modelo. Dado que este componente simula la dinámica del objeto, posee una referencia al componente de tipo *PhysicsComponent* que computa las magnitudes físicas relativas al movimiento a partir de las fuerzas generadas del sistema (peso, empuje, rozamientos, etc) y las diferentes masas, actualizando la posición del modelo en el mundo. La clase *LiquidGraph* modela el grafo de líquido en el caso que el objeto sea permeable e implementa la *interface RigidBody* para computar la magnitudes físicas de acuerdo a su estado (cantidad de líquido, nodos, etc).

Como ejemplo de nuestro método, las imágenes de la boya mostrada en la figura 7.2 están simuladas usando el método propuesto en la tesis para flotabilidad. La misma animación es mostrada en la figura 7.3 renderizadas con las correspondientes fuerzas superpuestas con una proyección ortográfica. Es interesante notar cómo la fuerza de empuje cambia, tanto de magnitud como de punto de aplicación, de acuerdo a la proporción de modelo sumergido. El cómputo de la fuerza de empuje use la TP y la TVA que, a su vez, son computadas a partir de la geometría de manera no supervisada.

En la figura 7.4 se muestran algunos cuadros de la animación de un barco hundiéndose. La permeabilidad del objeto es definida por texel y almacenada en la TPM simulando una sección dañada en la parte delantera derecha del casco del barco. La TPM es mostrada en la imagen superior izquierda. El mismo modelo de barco es mostrado en la figura 7.5 pero con una TPM diferente; en este caso, la sección dañada es en ambos lados de la sección trasera del casco. El tiempo de cómputo por cuadro de la animación se muestra en la Tabla 7.1. Es importante señalar que los tiempos de cómputo no dependen del tamaño de la malla poligonal del modelo dado que el método es completamente basado en texturas.

En la Figura 7.6 se muestra un conjunto de cuadros de una simulación de flotación inestable. Como el centro de flotación está situado por debajo del centro de masa del objeto, éste no tiende a recuperar su rotación inicial. Comparar este comportamiento con

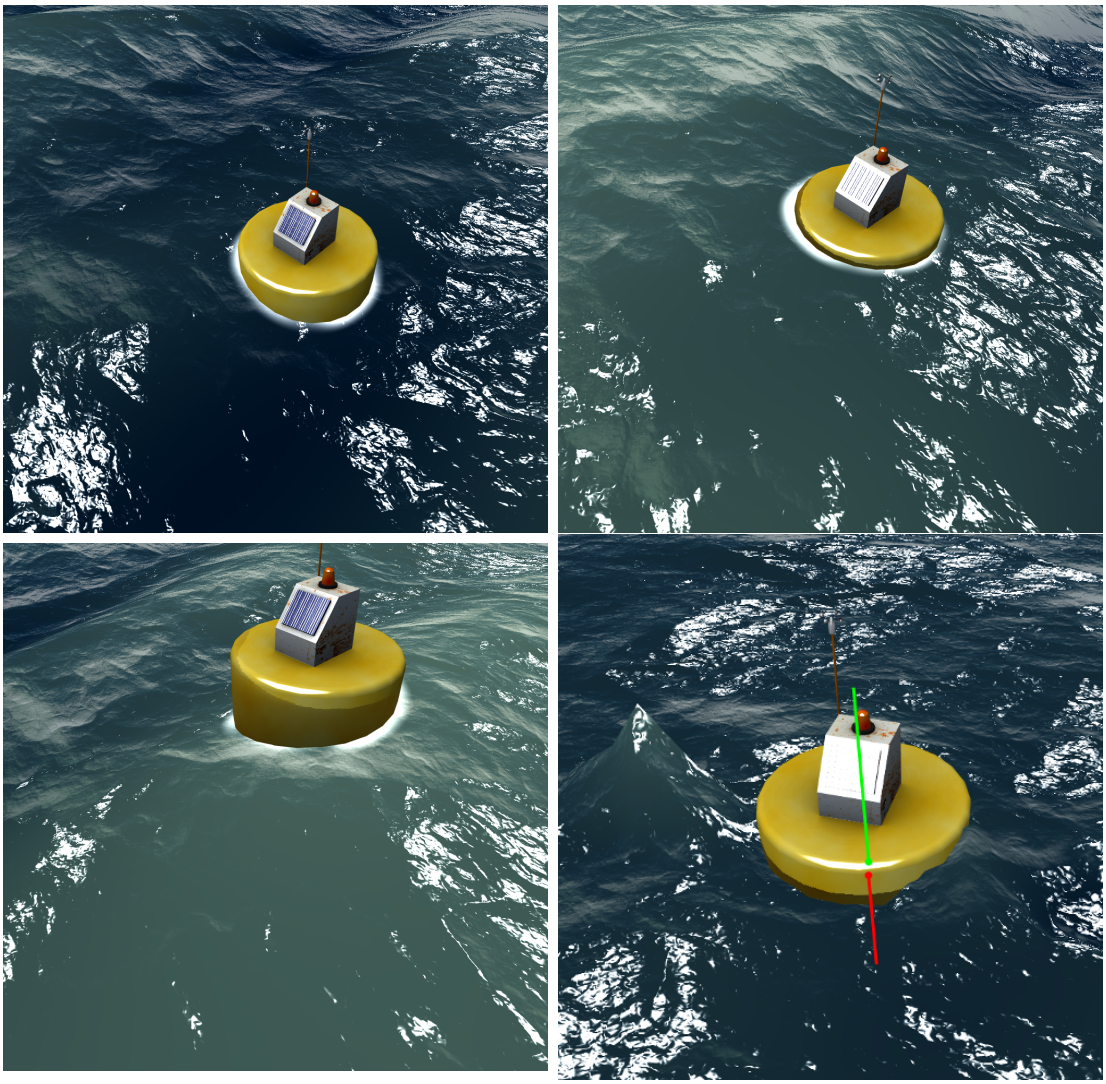


Figura 7.2: Cuadros de una animación mostrando la simulación de una boya. La boya no es permeable, es decir, no entra líquido a su interior. Utilizando la posición en el mundo de los texels, la zonas en contacto con el agua se renderizan de manera acorde con el método descripto en el capítulo 5. El cuadro inferior derecho muestra el efecto de la integración con el método para acoplamiento sólido-líquido desarrollado en Jeschke et al. (2018). Notar la perturbación que genera la boya sobre la superficie del agua.

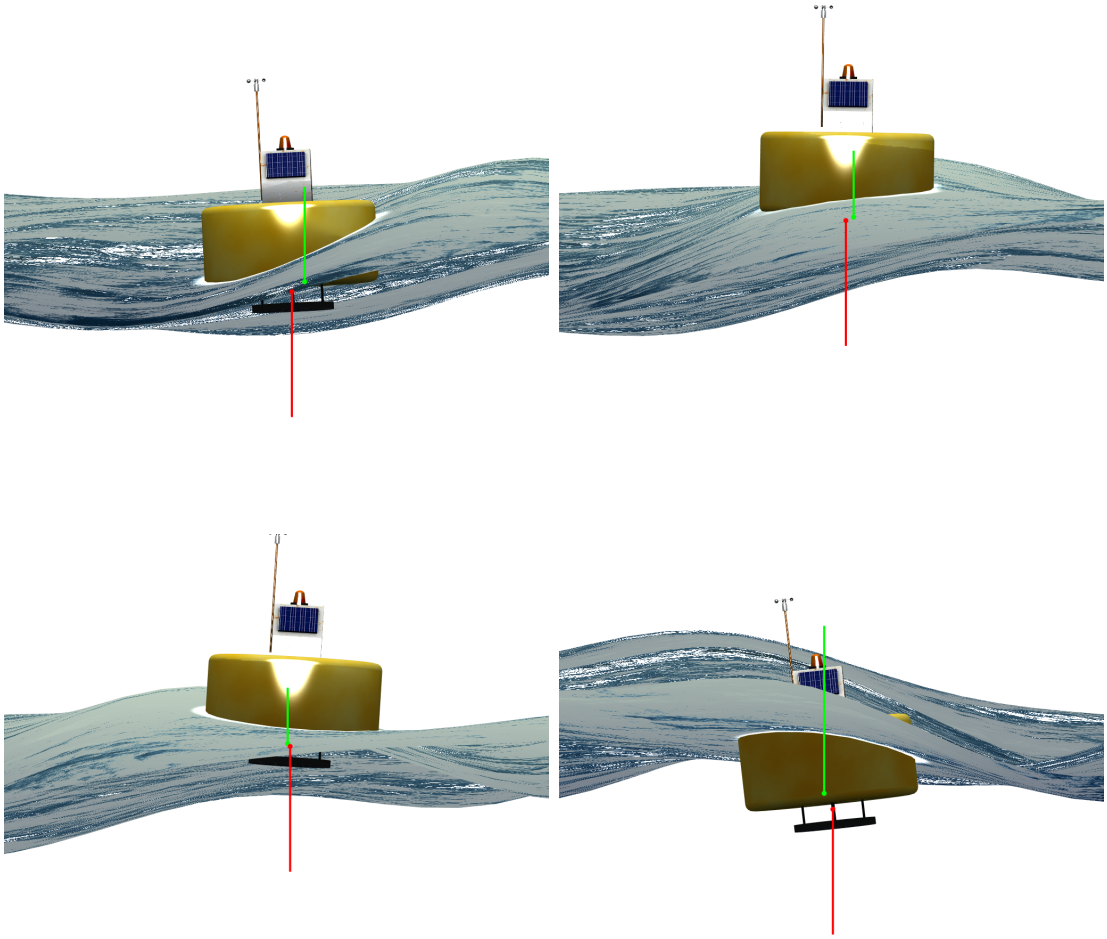


Figura 7.3: Cuadros de la animación de la boya en diferentes instantes de tiempo mostrando las fuerzas correspondientes al peso y al empuje en perspectiva ortográfica. Es posible observar las posiciones relativas y las magnitudes de las fuerza de peso (vector rojo) y el empuje (vector verde) de acuerdo a la porción sumergida del modelo. Claramente la fuerza de peso se mantiene constante a lo largo de la simulación porque no ingresa líquido al modelo; por lo tanto, su masa es constante.

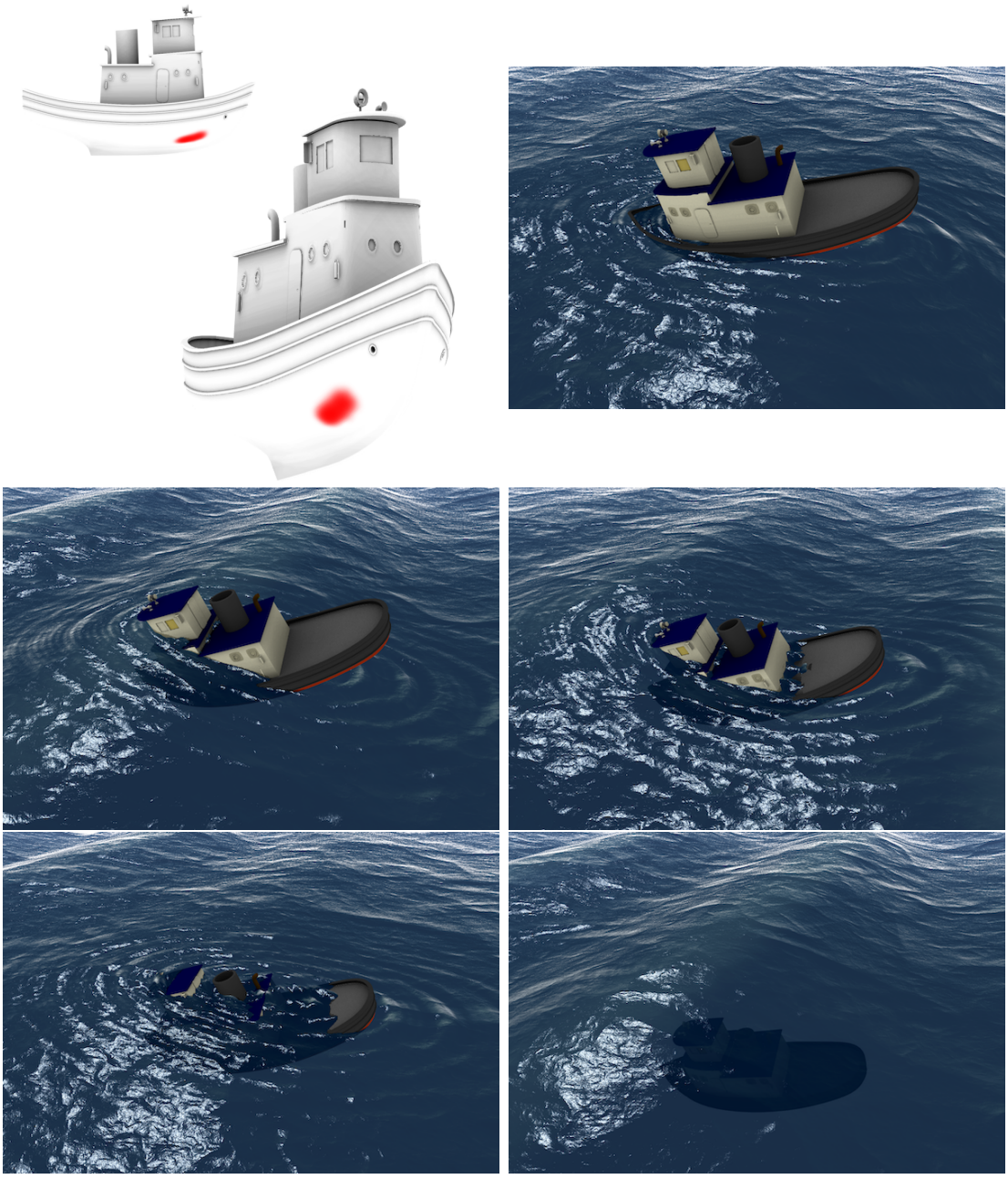


Figura 7.4: Cuadros de la simulación del hundimiento de un barco. La imagen superior izquierda muestra la TPM desde distintos puntos de vista. Los texels rojos son permeables mientras que los blancos no.

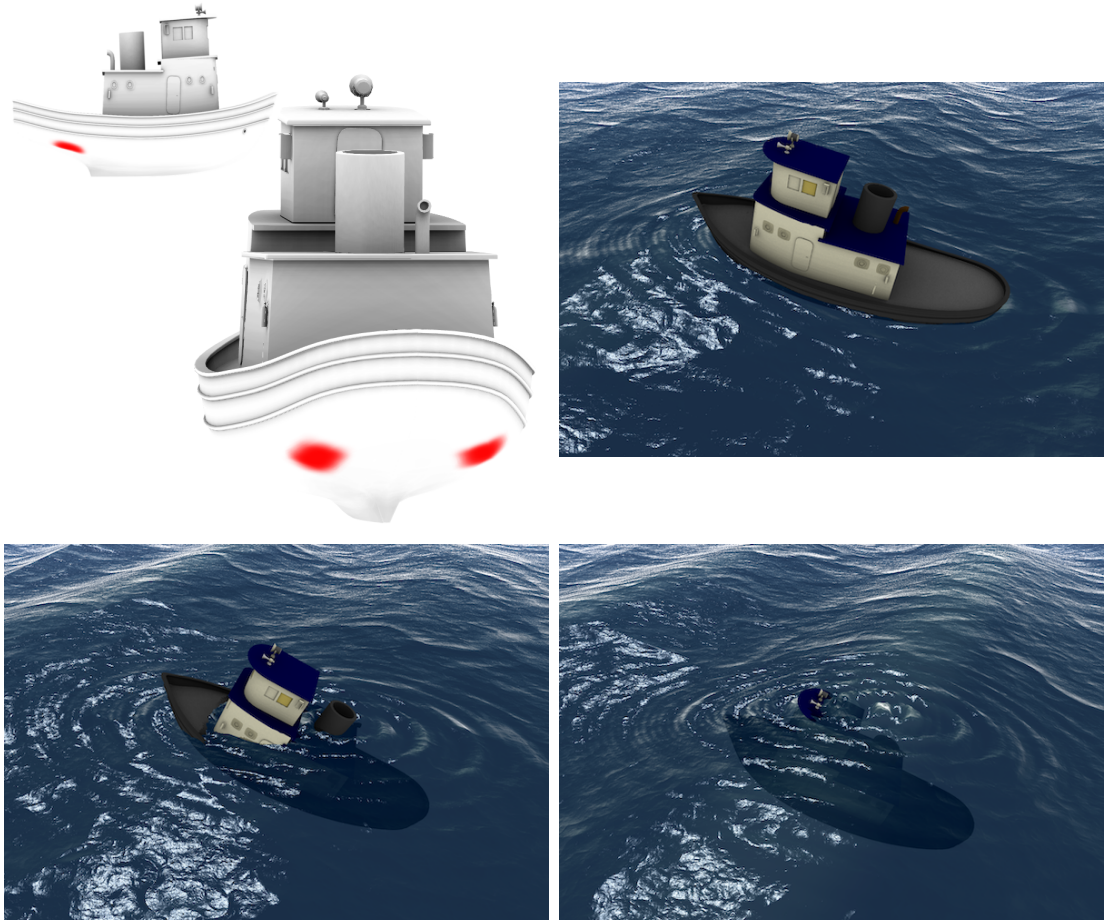


Figura 7.5: Cuadros de la simulación del hundimiento del barco con una TPM alternativa a la usada en la figura 7.4. La imagen superior izquierda muestra la TPM desde distintos puntos de vista. Los texels rojos son permeables mientras que los blancos no.

Computation times (ms)		
Texture size	Impermeable	Permeable
512	1.7	2.4
1024	2	3.1
2048	3	4.2

Tabla 7.1: Tiempos de cómputo (en milisegundos) para diferentes tamaños de TVA. Los objetos impermeables (boya) no tiene un grafo definido ya que se asume el interior sin líquido. Por otro lado, los objetos permeables (barco hundiéndose) tiene un grafo definido, por lo tanto el tiempo de cómputo para actualizar su estado es mayor. Los tiempos no dependen de la cantidad de polígonos ni de vértices en la malla del modelo. Además, el tamaño de las texturas necesarias para el procesamiento del método es independiente al de las texturas usadas para el renderizado.

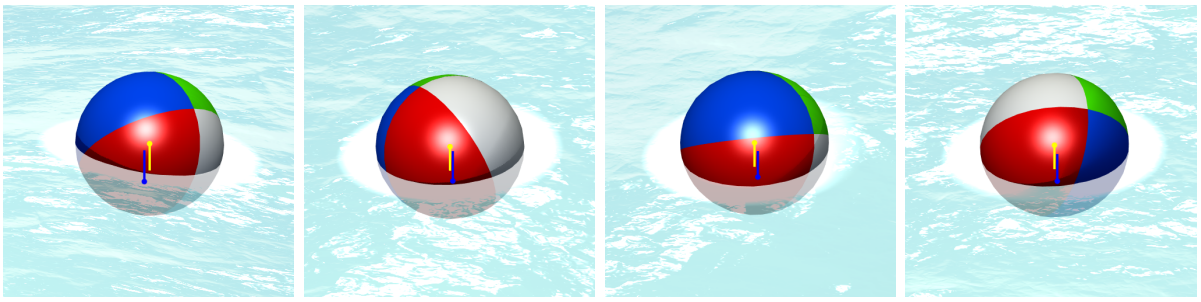


Figura 7.6: Diferentes cuadros de la simulación de flotabilidad inestable. La pelota tiene densidad uniforme de masa sobre la superficie, por lo tanto su centro de masa está localizado en el centro geométrico del objeto. Por otro lado, el centro de flotabilidad está localizado siempre por debajo del centro de masa (ya que el objeto no está totalmente sumergido) generando un movimiento inestable, es decir, el objeto no tiende a recuperar su rotación inicial.

lo detallado en la Figura 3.6b y con la simulaciones de los barcos, en las cuales el objeto a pesar de ser perturbado por las olas tendía a volver a su posición de rotación inicial. En el caso particular de esta simulación, la pelota no regresa a su rotación inicial; por esa razón, se denomina inestable.

Por otro lado, el método presentado también puede ser usado para simular objetos cuya distribución de masa cambie a lo largo del tiempo. En estos casos, las magnitudes físicas involucradas en la simulación necesitan ser computadas en cada cuadro de la simulación, ya que cambian continuamente y no pueden ser precomputadas. En la Figura 7.7

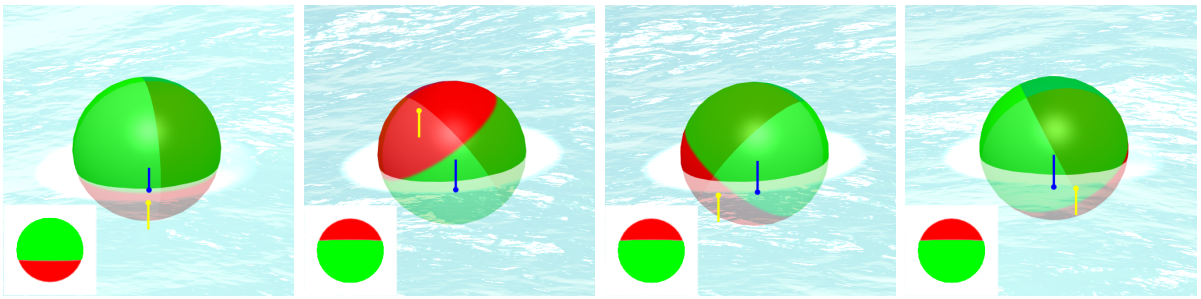


Figura 7.7: Cuadros de una simulación de un objeto con distribución de masa variable a lo largo del tiempo. El método propuesto puede ser usado también con objetos de masa variable (por ejemplo, un barco pequeño con gente moviéndose en su interior). El hecho de usar texturas como estructuras de datos permite al método calcular las magnitudes físicas del sistema en tiempo-real y de acuerdo al estado del sistema (ver la distribución de masa en el recuadro inferior izquierdo de cada cuadro). En la figura, el mapa de color muestra la función densidad de masa en cada cuadro: las zonas rojas muestran mayor densidad de masa y las zonas verdes, menor. El vector amarillo es la fuerza de peso y el vector azul la fuerza de flotación. En esta simulación, la masa total, el centro de masa y el tensor de inercia fueron computados en cada cuadro usando texturas de 1024x1024 texels. El tiempo promedio de cómputo del cuadro fue de 3,2 milisegundos.

se presenta una pelota con distribución de masa variable. La función densidad de masa es mostrada codificada en un esquema de pseudo-color sobre el modelo; las zonas rojas indican mayor densidad de masa y las zonas verdes, menor. La distribución correspondiente a cada cuadro es mostrado en un recuadro en la zona inferior izquierda en cada uno de los cuadros. En este caso, a diferencia de la simulaciones del barco y la boya donde la distribución de masa es constante en el tiempo, las magnitudes deben re-computarse en cada cuadro. Esta simulación muestra que los tiempos de cómputo, aún en un caso general donde ciertas magnitudes no pueden pre-computarse, se mantienen acordes a aplicaciones interactivas. Además, el modelo propuesto es capaz de funcionar con modelos deformables y/o animados si la TPA es computada en cada cuadro, por ejemplo, con técnicas de *render to texture*.

La Figura 7.8 muestra un objeto de felpa que tiende a absorber líquido a través de su superficie permeable siguiendo un modelo de propagación por difusión. En esta caso, el grafo de líquido es generado de manera automática con el método explicado anteriormente.

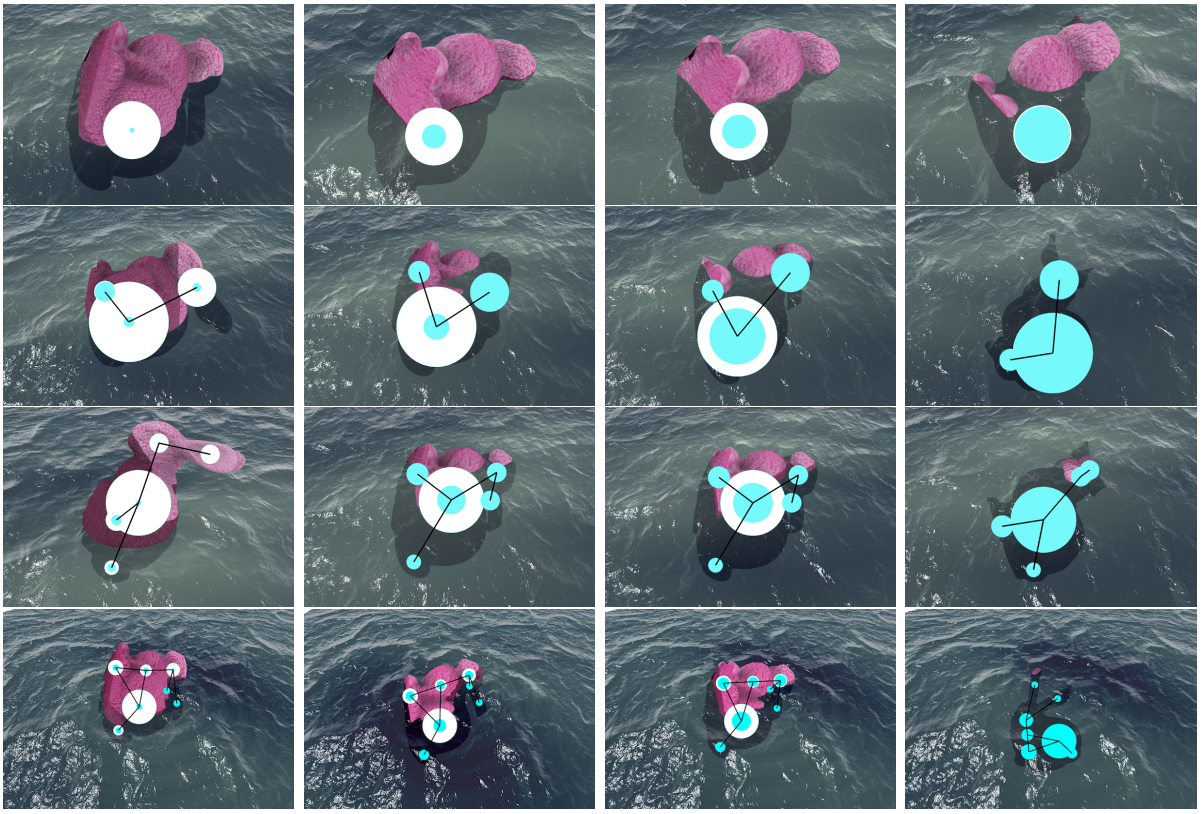


Figura 7.8: Cuadros de la simulación de un muñeco de felpa usando el grafo generado automáticamente mostrado en la figura 4.1.

En la Figura 7.9 se muestra una simulación de un modelo deformable mostrando que el método también funciona correctamente para simulaciones de objetos inmersos en medios gaseosos.

En la Figura 7.10 se muestra la precisión del método mostrando el volumen sumergido computado de una esfera de radio $R = 0{,}5$ a diferentes profundidades con diferentes tamaños de TVA y TP. Es sencillo observar el balance típico entre memoria requerida y precisión: por ejemplo, texturas de tamaño 128x128 pueden describir un volumen simple como una esfera con resultados aceptables.

En el gráfico de la derecha se compara el resultado del método propuesto para computar volumen sumergido utilizando texturas de (1024x1024) con la solución analítica y el método usado en el paquete comercial AQUAS2020 (Domatic Games, 2020) para el motor de videojuegos Unity (Unity Technologies, 2019), el cual está diseñado para modelar y simular agua como una superficie y efectos tales como la flotabilidad. AQUAS2020 computa la fuerza de empuje como la resultante de diversas fuerzas, una por cada triángulo sumergido de la malla poligonal. La magnitud de cada fuerza es calculada como el volumen desplazado por el correspondiente triángulo. Este proceso es aplicado solo para triángulos orientados hacia abajo, lo que explica el error considerable y la relación lineal entre volumen desplazado y la profundidad a medida que el objeto se sumerge totalmente. Además, el uso de polígonos para computar medidas geométricas dificulta la implementación eficiente de tales algoritmos en la GPU generando tiempos de procesamiento considerables (por ejemplo, una simulación con un modelo *Stanford Bunny* de 15k triángulos tarda 22mS aproximadamente.)

La escena de la isla mostrada en la Figura 7.11 es un ejemplo del algoritmo de simulación de materiales porosos húmedos y consiste en cuatro modelos, de 250k triángulos en total, que simulan la presencia de humedad en su superficie. El tronco y las rocas tienen texturas de 2048×2048 texels mientras que la caja de madera y la isla tienen texturas de 1024×1024 texels. En el caso de la isla, la textura de color tiene una disposición de

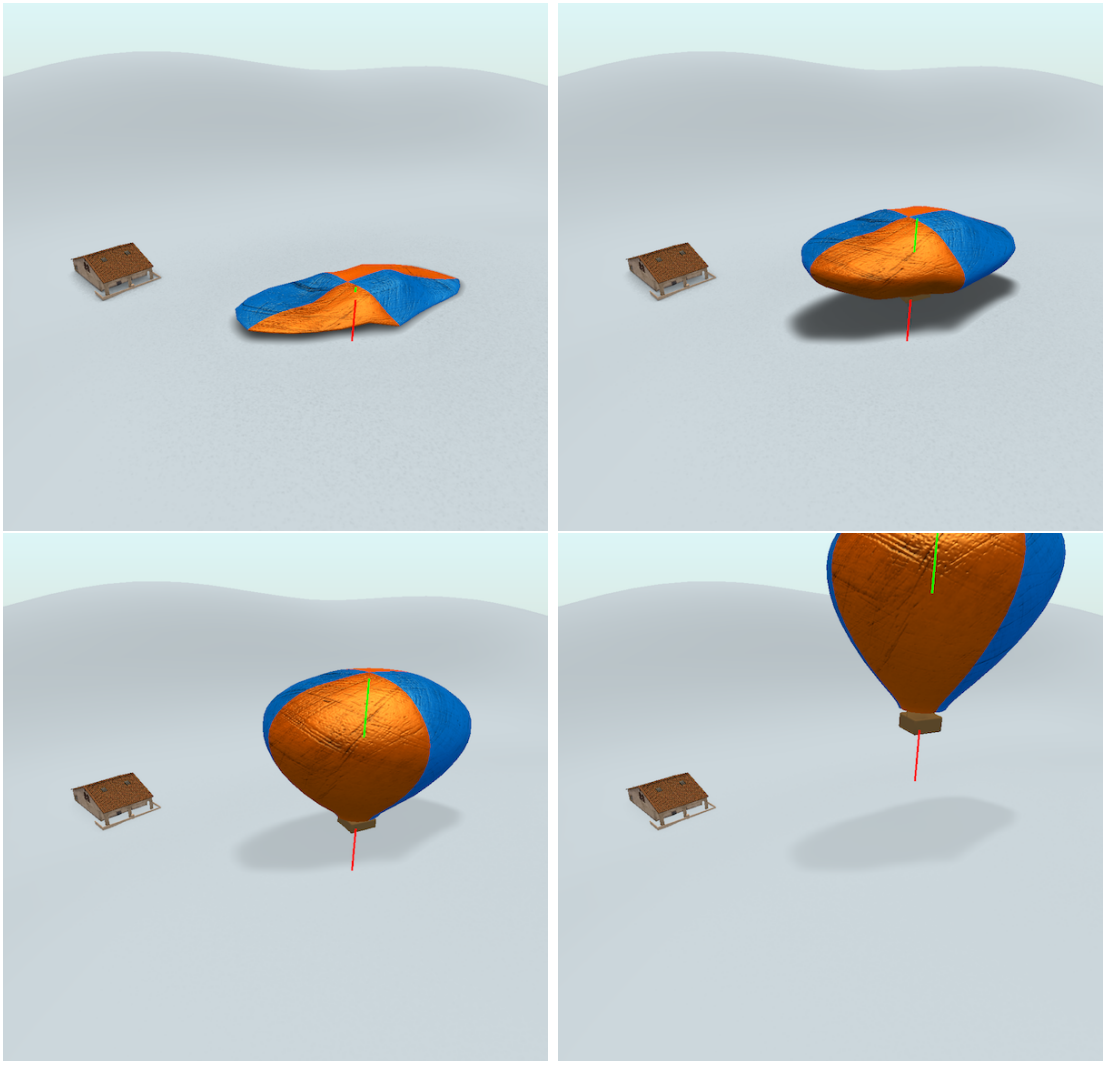


Figura 7.9: Cuadros de la simulación de un globo aerostático mostrando el funcionamiento del método propuesto con mallas poligonales animadas. En este caso el objeto está sumergido en un medio gaseoso (aire). El vector rojo muestra la fuerza de peso, mientras que el verde muestra la fuerza de empuje que depende del volumen de aire desplazado.

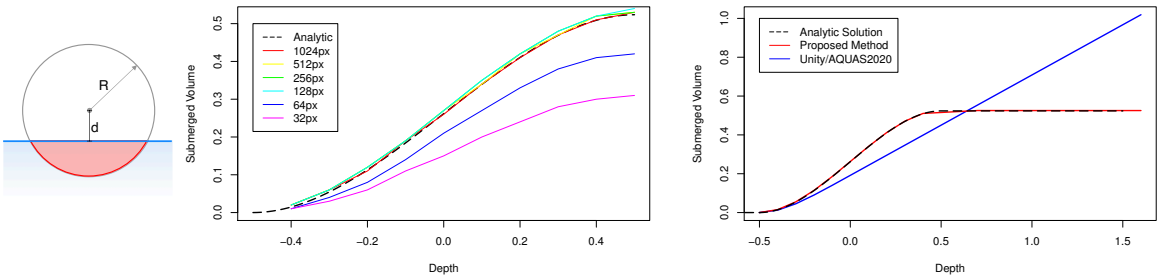


Figura 7.10: Izquierda: Volumen sumergido de una esfera de radio $R = 0{,}5$ en función de la profundidad, desde $d = -0{,}5$ (no sumergido) hasta $d = 0{,}5$ (totalmente sumergido). La superficie del agua está a profundidad $d = 0$. Centro: valor analítico y aproximaciones computadas con el método propuesto para diferentes resoluciones de TVA y TP (desde 32x32 hasta 1024x1024 texels). Derecha: resultados del método propuesto comparado con la solución analítica y el método propuesto por el paquete de simulación AQUAS2020 para el motor Unity.

mosaico (*tiled*). Además, la caja de madera y la isla implementan el algoritmo de difusión de líquido en la superficie a través de CDL de tamaño $30 \times 30 \times 30$ y $80 \times 70 \times 80$ respectivamente. Cada cuadro de animación tarda, en promedio, 11 milisegundos en ser procesado.

En algunos casos, codificar las texturas en un formato de imagen de 8 bits puede resultar en algunos defectos visuales debido a problemas de precisión. Como alternativa, pueden utilizarse formatos de imagen de punto flotante y mayor profundidad de bits, con la esperable penalización en ocupación de memoria y tiempo de cómputo.

En la Figura 7.11 se muestran algunos cuadros de la simulación de la escena de la playa. Las olas del mar, a medida que avanzan y retroceden, alcanzan y mojan diferentes zonas de la arena y de los objetos circundantes. Los fenómenos simulados pueden ser observados en las zonas donde la arena es mojada por el agua y luego secada por la condiciones ambientales, la difusión puede notarse ya que existen zonas no alcanzadas directamente por el agua pero que son humedecidas a través de la circulación del agua de sus zonas vecinas.

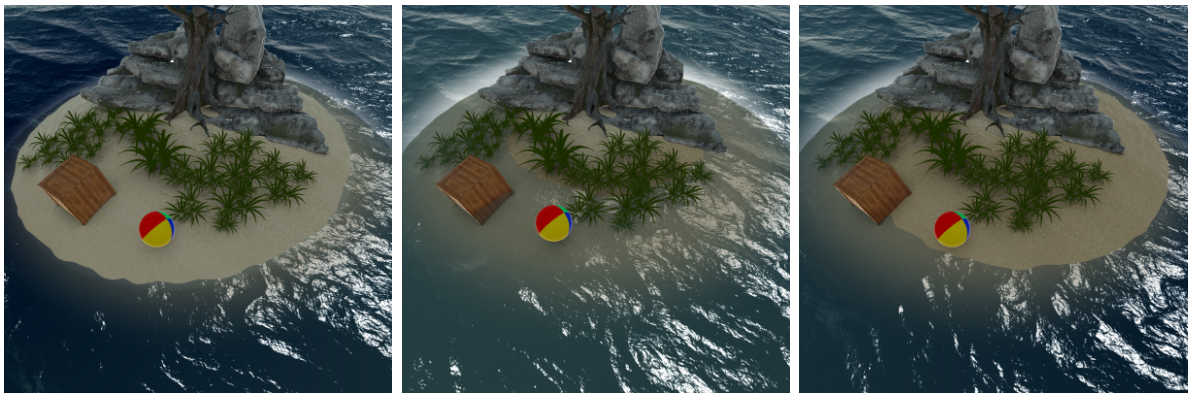


Figura 7.11: Cuadros de la simulación de la playa, mostrando el efecto del proceso de difusión. Porciones de arena fuera del alcance del agua que son humedecidas por este efecto. Los cambios en la apariencia y reflectividad entre zonas secas y mojadas son fácilmente notables.

La Figura 7.12 muestra los resultados utilizando una *Textura de líquido* creada de manera artística y el modelo de renderizado simulando líquido traslúcido coloreado. Esta imagen muestra que el método de renderizado propuesto puede ser usado de manera independiente al modelo físico propuesto para resolver la interacción líquido-sólido, en caso que se necesite una solución estática.

En la Figura 7.13 se muestran cuadros de una simulación de dos objetos compartiendo un mismo CDL. Este permite implementar el efecto de líquido difundiéndose entre diferentes mallas poligonales.

En la figura 7.14 se compara una escena del mundo real, capturada con una cámara de alta calidad, con los resultados de nuestro método. En este caso, la TL fue inicializada de manera manual por un artista y, a partir de esa situación inicial, el sistema evolucionó con el método propuesto. Para estas imágenes sintéticas, la iluminación fue reproducida utilizando un mapa de irradiancia computado a partir de 6 imágenes capturando el entorno alrededor de la maceta. Posteriormente, a partir de una muestra de la superficie de la maceta real, se generó una textura sintética emulando al material original para lograr una apariencia similar. A pesar de diferencias particulares, por ejemplo en los bordes de la

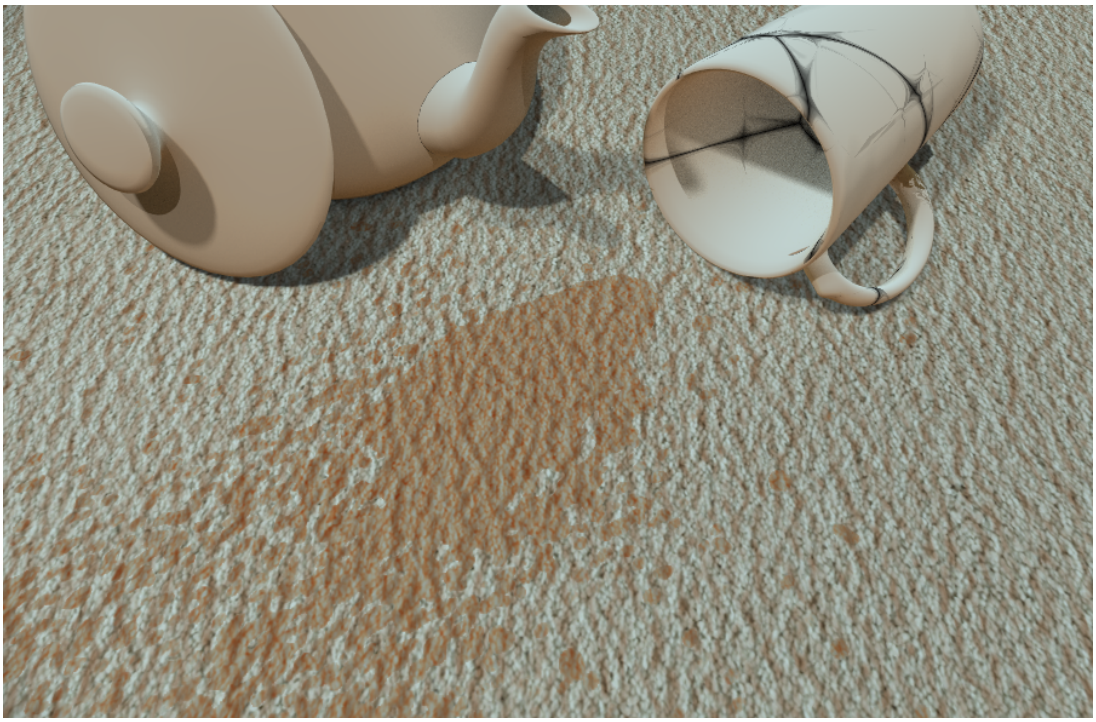


Figura 7.12: Imagen renderizada con líquido coloreado a partir de una TL especificada por un artista.

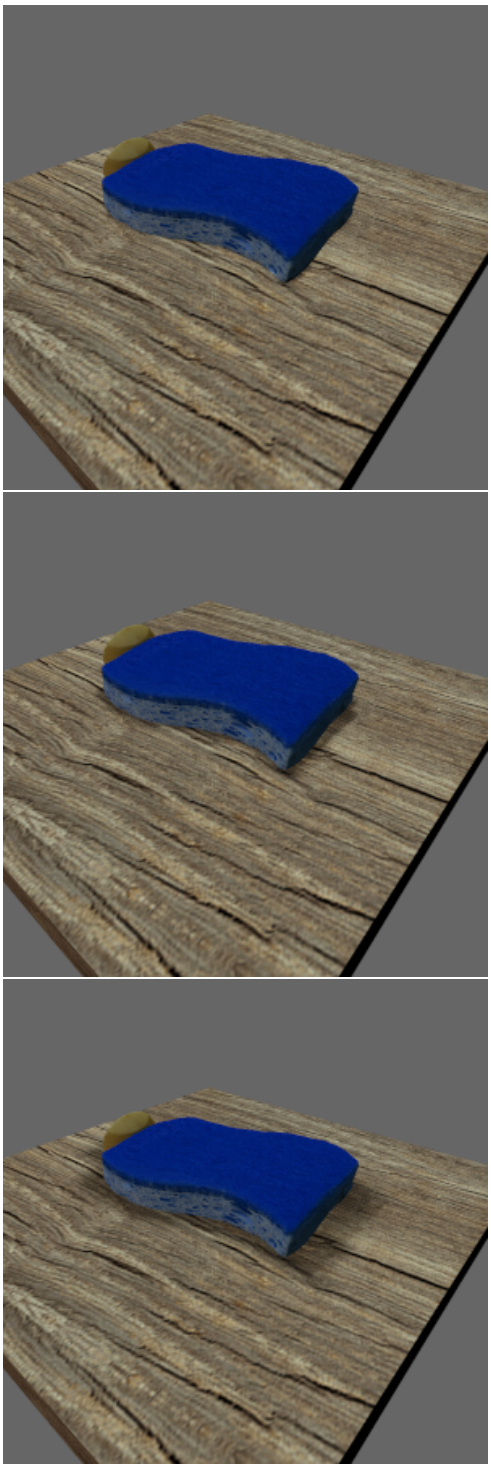


Figura 7.13: El CDL puede ser compartido por diferentes mallas poligonales. El resultado es poder simular la situación en que un objeto mojado humedece otro seco por contacto.

silueta de humedad, se observa gran similitud en la apariencia y evolución del sistema real y simulado.

En las Figuras 7.15 y 7.16 se muestra una comparación entre imágenes creadas con el modelo presentado e imágenes correspondientes al proceso de secado publicadas en la base de datos correspondiente al trabajo Gu et al. (2006). En todos los casos, la forma de la zona húmeda fue extraída usando técnicas de procesamiento de imágenes digitales con el objetivo de lograr una distribución de líquido comparable. Posteriormente, el modelo de simulación fue ejecutado con las imágenes de los materiales originales y la distribución de líquido extraída, obteniendo las imágenes mostradas en la tercera columna de las Figuras 7.15 y 7.16. Puede observarse una gran similitud entre ambos conjuntos de imágenes. Específicamente en la Figura 7.15 se muestra el proceso con imágenes de granito.

En la figura 7.16 se comparan muestras de madera de la base de datos con resultados de la simulación. En este caso, se puede ver que el modelo propuesto reproduce fielmente las características principales de la superficie húmeda, con excepción de una marcada diferencia en la saturación del color final. En cualquier caso, el método puede ser corregido al asignarle un color a la capa de líquido simulado en la superficie, ajustando el color del líquido sobre la superficie en la ecuación 6.1. Solo ese paso adicional fue requerido para lograr el resultado final de gran similitud con las imágenes del mundo real.

Los videos con las animaciones correspondientes a las diferentes simulaciones pueden ser accedidos desde `https://youtu.be/67D1GOn-5lk` y `https://youtu.be/JxkOrAjlv6M`. En caso de versión impresa, escanear los códigos QR de la figura 7.17 con un dispositivo móvil.

Figura 7.14: Imágenes generadas a partir del modelo desarrollado comparadas con capturas del mundo real. Con el objetivo de simular las condiciones reales, se tomaron en cuenta las siguientes condiciones: la textura del modelo de la maceta fue sintetizada a partir de una muestra de la textura de la superficie de la maceta real utilizando un método basado en parches implementado por el proyecto G'MIC G'MIC Project (2017). Además, las condiciones de iluminación fueron recreadas utilizando un mapa de irradiancia basado en fotografías del entorno. Las imágenes de la izquierda son fotografías del mundo real capturadas con una cámara *reflex* con una configuración supervisada en un día nublado, las imágenes centrales son resultados sintéticos obtenidos utilizando el método presentado y las imágenes de la derecha muestran los resultados sintéticos con pseudo-color mostrando la evolución de la TL.

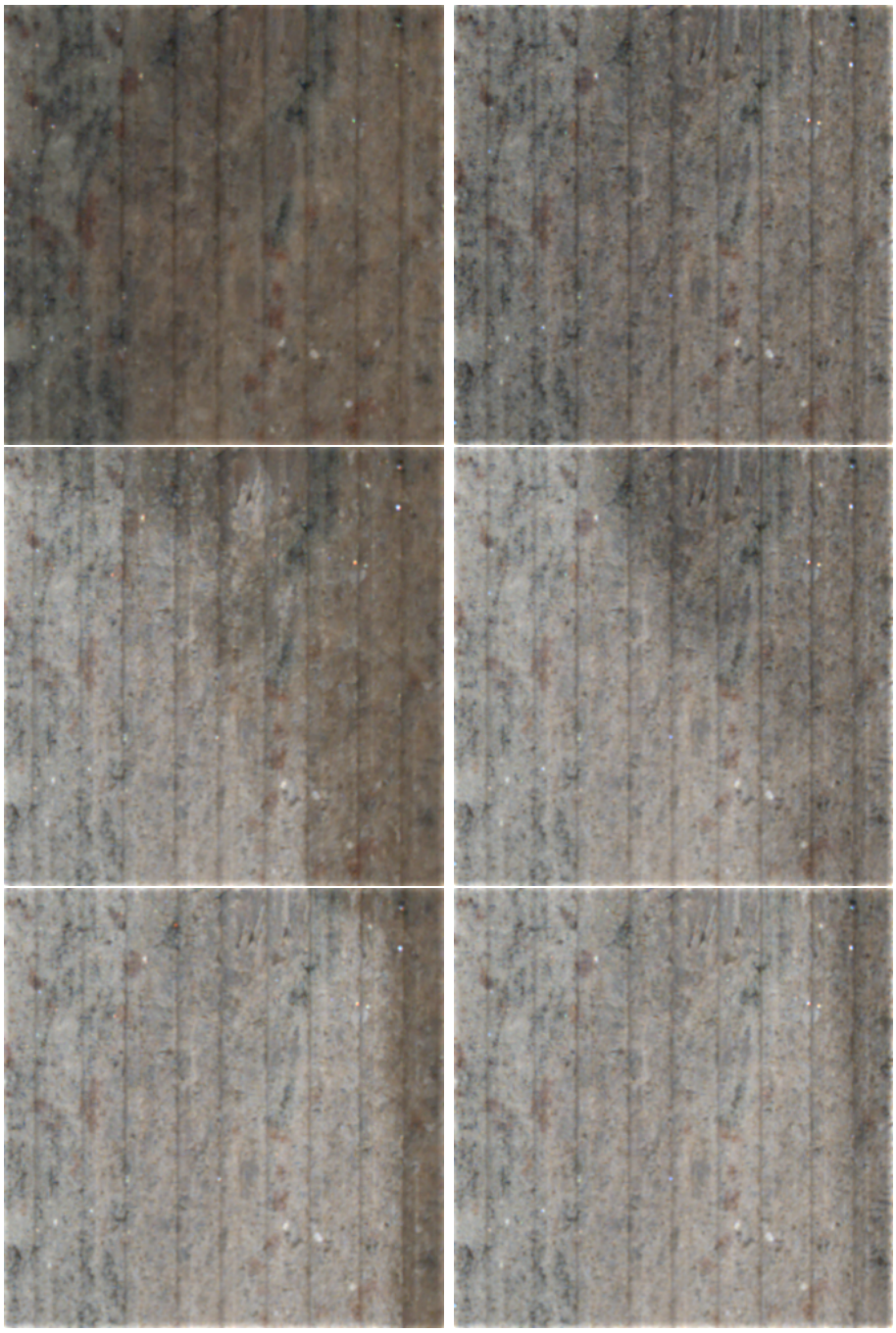


Figura 7.15: Muestras de secado de granito comparando el método presentado con capturas del trabajo de Gu et al. (2006). La columna izquierda muestra las imágenes originales de la base de datos mientras que la columna de la derecha muestra las imágenes procesadas con el algoritmo. Es notable la similitud entre ambos conjuntos de imágenes.

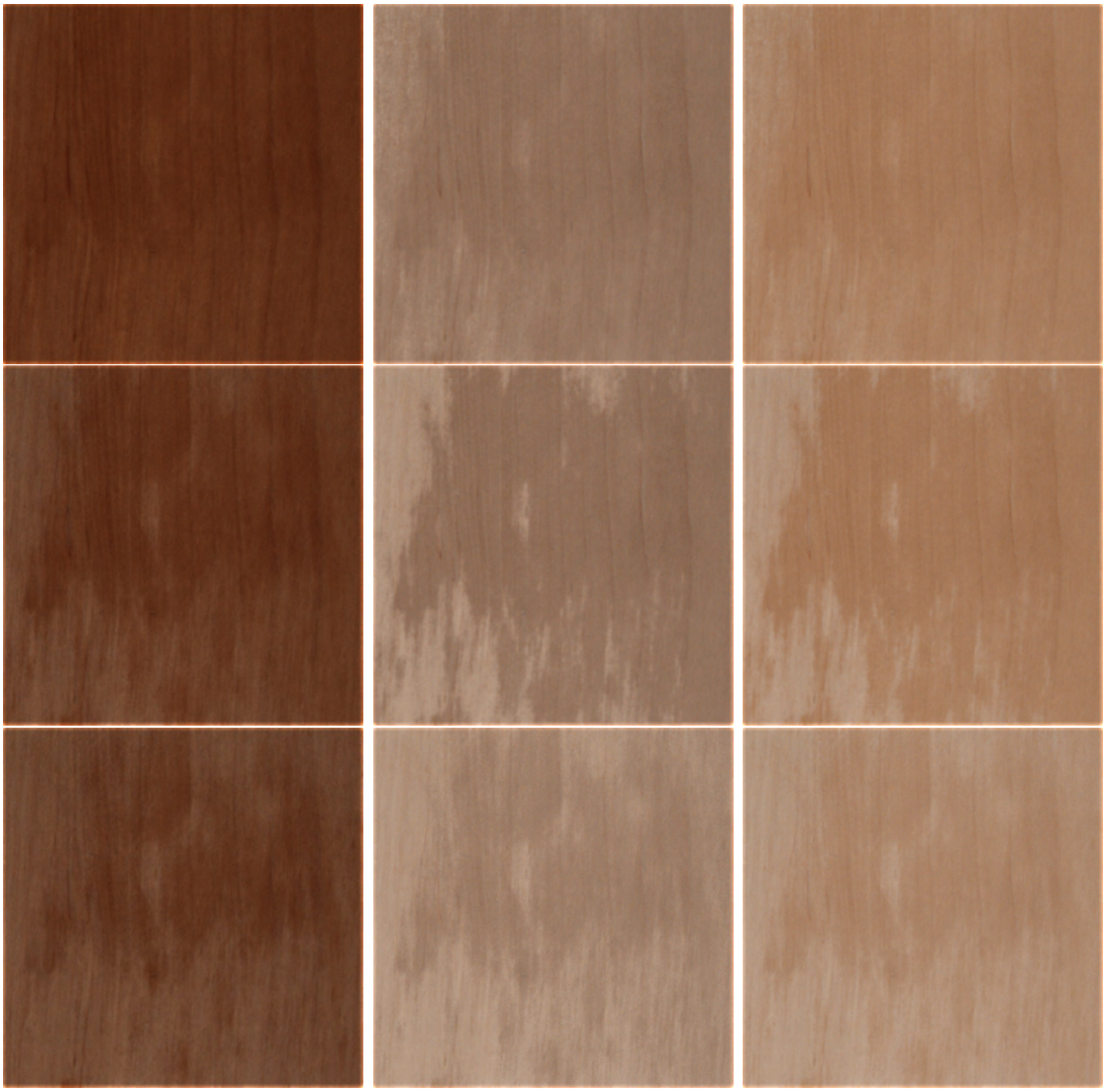


Figura 7.16: Diferentes muestras de la base de datos del trabajo de Gu et al. (2006) correspondientes al proceso de secado de la madera comparadas con los resultados del método propuesto en la tesis. La columna de la izquierda muestra las imágenes de la base de datos mencionada, la columna del centro muestra imágenes sintetizadas con nuestro método usando líquido incoloro y la columna de la derecha utiliza líquido coloreado para lograr resultados similares a las imágenes originales.

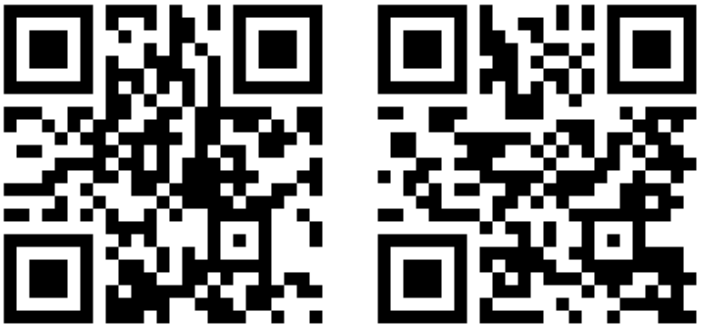


Figura 7.17: Códigos QR para acceder a los videos con las correspondiente simulaciones.

Capítulo 8

Conclusiones

En esta tesis hemos presentado una técnica que permite simular el comportamiento de objetos sumergidos en líquidos de manera interactiva y realista. Esta técnica se basa en dos componentes: un algoritmo diseñado para ser ejecutado en la GPU para el cómputo en tiempo real de las magnitudes físicas involucradas en el movimiento de un objeto (masa, centro de masa, tensor de inercia, etc.) y, por otra parte, una estructura de datos y un algoritmo basado en teoría de grafos para la simulación de objetos permeables o con interiores de relativa complejidad. Estos dos componentes permiten la simulación realista de objetos rígidos y deformables, permeables y no permeables en diversas situaciones.

El cómputo de todas las magnitudes físicas involucradas en el movimiento de un cuerpo sólido están íntimamente relacionadas y no pueden ser asignadas de forma manual a menos que se tenga un conocimiento profundo del sistema simulado. De lo contrario, la dinámica resultante será inadecuada y la simulación poco realista. La aproximación descripta en esta tesis fue desarrollada con la premisa de poder ser utilizada en aplicaciones gráficas de tiempo real, como simuladores y vídeojuegos. Si se requiere mayor precisión, existen métodos fuera del campo del tiempo real que permiten calcular estas magnitudes de manera más precisas (por ejemplo, integración de Monte-Carlo).

En importante notar la facilidad de implementar una técnica basada en Level-Of-Detail (LOD) a partir del método presentado. Simplemente ajustando el tamaño de las texturas se puede ajustar el nivel de balance entre velocidad de cómputo y precisión en los resultados.

La técnica presentada depende de una parametrización relativamente buena, así como también de un tamaño de textura razonable. Esto no es una limitación ya que, en la práctica, todos los modelos utilizados en *pipelines* modernos la poseen.

El método fue diseñado con la intención de ser completamente no supervisado. Sin embargo, el usuario puede intervenir para ajustar manualmente o definir ciertos parámetros (por ejemplo, la TL, la distribución de masa a lo largo del modelo o el GL) con el objetivo de lograr efectos específicos. Es importante notar que la TPM puede ser computada de manera interactiva, permitiendo la destrucción dinámica de objetos de acuerdo a las condiciones de la simulación. Como ejemplo, en el área de vídeo-juegos puede implementarse una batalla de barcos con destrucciones interactivas.

Respecto al pre-procesado del modelo para generar las texturas especiales que el método utiliza, debe notarse que solamente la capa externa de poligonos debe procesarse. En general los modelos pensados para aplicaciones interactivas no poseen el interior modelado, por lo tanto no es una restricción fuerte. Por otro lado, el grafo de líquido puede usarse, no solamente para representar el líquido interno al modelo, sino para modelar características internas al modelo no contempladas en la malla poligonal, por ejemplo distribuciones de masa específicas.

Por otro lado, el método propuesto no está libre de limitaciones. Probablemente la más importante sea que el cómputo en espacio de textura puede no reflejar las propiedades reales del objeto, ya que solo se computan métricas que dependen de su superficie y asumen el interior como homogéneo. Como se dijo anteriormente, una posible solución a esto es extender la idea del grafo para representar otras magnitudes internas del modelo que por defecto son consideradas homogéneas. En todos estos casos, la solución es generar el grafo

de manera manual. Es importante volver a mencionar que los modelos típicamente usados en aplicaciones interactivas no poseen información del medio interno, ni magnitudes físicas asociadas. Por lo tanto, cualquier método de simulación basado en física deberá requerir la definición de estas magnitudes. En el caso específico de la densidad de masa, puede definirse tan arbitraria como sea necesario, ya que si se mantiene constante (para el modelo sin líquido en su interior) a lo largo de la simulación, las magnitudes que dependen de ella, masa total y tensor de inercia, pueden pre-computarse y el método funciona sin cambios.

El método también requiere superficies cerradas. En el caso de objetos cóncavos, como un balde parcialmente sumergido, el algoritmo fallará ya que no se podrá modelar el líquido que queda atrapado dentro de la concavidad del objeto. Para casos particulares, una posible solución puede ser agregar polígonos invisibles pero incluidos en el mapeo UV que lo transformen en un objeto cerrado. Finalmente, cabe mencionar que el uso de un modelo basado en grafos prueba ser práctico, eficiente y útil para aplicaciones interactivas donde la precisión no es mandatoria frente al tiempo de procesamiento. Sin embargo, pueden ser derivados otros modelos más avanzados. Además del método específico propuesto, una de las contribuciones de la presente tesis es mostrar la posibilidad de diseñar soluciones basadas en cómputo acelerado sobre superficies utilizando texturas y desacoplando la simulación del interior de los objetos en caso de ser necesario.

Desde el punto de vista de la visualización, el problema del acoplamiento fue resuelto presentando una técnica diseñada para GPU para modelar y renderizar la evolución y la apariencia de superficies mojadas, incluyendo fenómenos como evaporación, difusión y absorción. Al igual que la solución para el problema de la flotabilidad, este método también hace uso extensivo de la parametrización geométrica típicamente usada para mapeo de texturas. Además, se implementó un método de renderizado basado en física que produce buenos resultados para la generalidad de materiales absorbentes en tiempos aceptables para soluciones de tiempo real. El modelo demuestra producir resultados convincentes para un gran espectro de materiales absorbentes como se ilustra en las diferentes imágenes a lo

largo de la tesis. Sin embargo, como se mencionó en el trabajo previo, existen soluciones ad-hoc para materiales específicos (por ejemplo: arena, pelo, papel/tinta).

Tomando en consideración que se han presentado comparaciones (Capítulo 7) con trabajos considerados estado-del-arte, con base de datos de materiales y con imágenes propias, consideramos que nuestra técnica está bien situada respecto a las técnicas usadas actualmente.

Sin embargo, el método propuesto sufre de algunas limitaciones particulares que es conveniente señalar. En primer lugar, si bien el método es capaz de reproducir una rango amplio de situaciones de la vida real, tiene limitaciones cuando se intenta reproducir situaciones o imágenes específicas ya que la saturación del color puede diferir en el material real del modelo de predicción. Sin embargo, con el parámetro de color en el modelo de renderizado puede acercarse al modelo real, como fue mostrado en la tabla de la figura 7.16.

Por otro lado, si bien existe la posibilidad de implementar la transferencia de líquido entre objetos diferentes de la escena, el proceso puede ser de alto costo computacional. Sin embargo, en materiales con constante de difusión relativamente baja, este fenómeno es poco notable, por lo tanto, esta etapa de procesamiento puede omitirse dejando una técnica sensiblemente más eficiente. Claramente, en este caso, solamente se simularían los procesos de absorción y evaporación.

Si bien las demostraciones de los algoritmos utilizan una superficie para representar el líquido, el método puede adaptarse fácilmente a un representación basada en partículas. Una alternativa es utilizar *Marching Squares* para recuperar la superficie del fluido; otro enfoque consiste en redefinir la profundidad de un texel utilizando las partículas del fluido. En este caso, puede asumirse el uso de una estructura de datos de particionado del espacio para optimizar las búsquedas y cada texel puede acceder en tiempo constante a las partículas relativamente cercanas a él. Existe diversas técnicas que han sido desarrolladas para hacer este cómputo rápido Hoetzlein (2014).

Bibliografía

Abdulghany, A. R. (2017). Generalization of parallel axis theorem for rotational inertia. *American Journal of Physics*, 85(10):791–795.

Akbay, M., Nobles, N., Zordan, V., and Shinar, T. (2018). An extended partitioned method for conservative solid-fluid coupling. *ACM Trans. Graph.*, 37(4):86:1–86:12.

Akenine-Möller, T., Haines, E., Hoffman, N., Pesce, A., Iwanicki, M., and Hillaire, S. (2018). *Real-Time Rendering 4th Edition*. A K Peters/CRC Press, Boca Raton, FL, USA.

Akinci, N., Ihmsen, M., Akinci, G., Solenthaler, B., and Teschner, M. (2012). Versatile rigid-fluid coupling for incompressible sph. *ACM Trans. Graph.*, 31(4):62:1–62:8.

Ångström, A. (1925). The albedo of various surfaces of ground. *Geografiska Annaler*, pages 323–342.

Bächer, M., Bickel, B., Whiting, E., and Sorkine-Hornung, O. (2017). Spin-it: Optimizing moment of inertia for spinnable objects. *Commun. ACM*, 60(8):92–99.

Batchelor, G. K. (2000). *An Introduction to Fluid Dynamics*. Cambridge Mathematical Library. Cambridge University Press.

Batty, C., Bertails, F., and Bridson, R. (2007). A fast variational framework for accurate solid-fluid coupling. *ACM Trans. Graph.*, 26(3):100.

Borshukov, G. and Lewis, J. (2003). Realistic human face rendering for "the matrix reloaded".

Brackbill, J. and Ruppel, H. (1986). Flip: A method for adaptively zoned, particle-in-cell calculations of fluid flows in two dimensions. *Journal of Computational Physics*, 65:314–343.

Bridson, R. (2008). *Fluid Simulation for Computer Graphics*. A K Peters/CRC Press.

Bridson, R. (2015). *Fluid Simulation for Computer Graphics, Second Edition*. Taylor & Francis.

Canabal, J. A., Miraut, D., Thuerey, N., Kim, T., Portilla, J., and Otaduy, M. A. (2016). Dispersion kernels for water wave simulation. *ACM Trans. Graph.*, 35(6):202:1–202:10.

Carlson, M., Mucha, P. J., and Turk, G. (2004). Rigid fluid: Animating the interplay between rigid bodies and fluid. In *ACM SIGGRAPH 2004 Papers*, SIGGRAPH '04, pages 377–384, New York, NY, USA. ACM.

Catmull, E. E. (1974). *A Subdivision Algorithm for Computer Display of Curved Surfaces*. PhD thesis. AAI7504786.

Chen, T. F., Baranoski, G. V. G., Kimmel, B. W., and Miranda, E. (2015). Hyperspectral modeling of skin appearance. *ACM Trans. Graph.*, 34(3):31:1–31:14.

Chu, N. S.-H. and Tai, C.-L. (2005). Moxi: Real-time ink dispersion in absorbent paper. *ACM Trans. Graph.*, 24(3):504–511.

Crank, J. (1956). *The Mathematics of Diffusion*. Clarendon Press.

Da, F., Hahn, D., Batty, C., Wojtan, C., and Grinspun, E. (2016). Surface-only liquids. *ACM Trans. Graph.*, 35(4):78:1–78:12.

d'Eon, E., Luebke, D., and Enderton, E. (2007). Efficient rendering of human skin. In *Proceedings of the 18th Eurographics Conference on Rendering Techniques*, EGSR'07, pages 147–157, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association.

Domatic Games (2020). Aquas 2020.

Dorsey, J., Edelman, A., Jensen, H. W., Legakis, J., and Pedersen, H. K. (1999). Modeling and rendering of weathered stone. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '99, pages 225–234, New York, NY, USA. ACM Press/Addison-Wesley Publishing Co.

Dorsey, J., Pedersen, H. K., and Hanrahan, P. (1996). Flow and changes in appearance. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, pages 411–420, New York, NY, USA. ACM.

Dorsey, J., Rushmeier, H., and Sillion, F. (2008). *Digital Modeling of Material Appearance*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Eberly, D. and Shoemake, K. (2004). *Game Physics*. Interactive 3D technology series. Taylor & Francis.

Fei, Y. R., Batty, C., Grinspun, E., and Zheng, C. (2018). A multi-scale model for simulating liquid-fabric interactions. *ACM Trans. Graph.*, 37(4):51:1–51:16.

Fick, A. (1995). On liquid diffusion. *Journal of Membrane Science*, 100(1):33–38.

Gerszewski, D. and Bargteil, A. W. (2013). Physics-based animation of large-scale splashing liquids. *ACM Trans. Graph.*, 32(6):185:1–185:6.

Gerszewski, D., Kavan, L., Sloan, P.-P., and Bargteil, A. W. (2013). Enhancements to model-reduced fluid simulation. In *Proceedings of Motion on Games*, MIG '13, pages 201:223–201:228, New York, NY, USA. ACM.

G'MIC Project (Accessed 12/12/2017). http://gmic.eu/.

Gonzalez-Ochoa, C. (2016). Advances in real-time rendering in games: Rendering rapids in uncharted 4. acm siggraph 2016 courses. In *SIGGRAPH '16: ACM SIGGRAPH 2016 Courses*, New York, NY, USA. ACM.

Gourlay, M. J. (2019). Fluid simulation for video games. https://software.intel.com/en-us/articles/fluid-simulation-for-video-games-part-9. [Online; accessed 10-January-2019].

Gu, J., Tu, C.-I., Ramamoorthi, R., Belhumeur, P., Matusik, W., and Nayar, S. (2006). Time-varying surface appearance: Acquisition, modeling and rendering. *ACM Trans. Graph.*, 25(3):762–771.

Guendelman, E., Selle, A., Losasso, F., and Fedkiw, R. (2005). Coupling water and smoke to thin deformable and rigid shells. *ACM Trans. Graph.*, 24(3):973–981.

Haas, A. E. and Verschoyle, T. (1928). *Introduction to theoretical physics*. Constable & company ltd London, 2d ed. edition.

Hall, C. and Hoff, W. (2009). *Water Transport in Brick, Stone and Concrete*. CRC Press.

Hendel, M., Colombert, M., Diab, Y., and Royon, L. (2014). Improving a pavement-watering method on the basis of pavement surface temperature measurements. *Urban Climate*, 10:189 – 200.

Hnat, K., Porquet, D., Merillou, S., and Ghazanfarpour, D. (2006). Real-time wetting of porous media. *MG&V*, 15(3):401–413.

Hoetzlein, R. (2014). Fast fixed-radius nearest neighbors: Interactive millon-particle fluid. In *GPU Technology Conference (GTC)*, Santa Clara, CA.

Jensen, H., Legakis, J., and Dorsey, J. (1999). Rendering of wet materials. In Lischinski, D. and Larson, G., editors, *Rendering Techniques 99*, Eurographics, pages 273–281. Springer Vienna.

Jensen, H. W., Marschner, S. R., Levoy, M., and Hanrahan, P. (2001). A practical model for subsurface light transport. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, pages 511–518, New York, NY, USA. ACM.

Jeschke, S., Skřivan, T., Müller-Fischer, M., Chentanez, N., Macklin, M., and Wojtan, C. (2018). Water surface wavelets. *ACM Trans. Graph.*, 37(4):94:1–94:13.

Kattawar, G. W. (1975). A three-parameter analytic phase function for multiple scattering calculations. *Journal of Quantitative Spectroscopy and Radiative Transfer*, 15(9):839 – 849.

Kimmel, B. W. and Baranoski, G. V. (2007). A novel approach for simulating light interaction with particulate materials: application to the modeling of sand spectral properties. *Opt. Express*, 15(15):9755–9777.

Kimmel, B. W. and Baranoski, G. V. (2010). Simulating the appearance of sandy landscapes. *Computers & Graphics*, 34(4):441 – 448. Procedural Methods in Computer Graphics Illustrative Visualization.

Kwatra, N., Wojtan, C., Carlson, M., Essa, I. A., Mucha, P. J., and Turk, G. (2010). Fluid simulation with articulated bodies. *IEEE Transactions on Visualization and Computer Graphics*, 16(1):70–80.

Lekner, J. and Dorf, M. C. (1988). Why some things are darker when wet. *Appl. Opt.*, 27(7):1278–1280.

Lenaerts, T., Adams, B., and Dutré, P. (2008a). Porous flow in particle-based fluid simulations. *ACM Trans. Graph.*, 27(3):49:1–49:8.

Lenaerts, T., Adams, B., and Dutré, P. (2008b). Porous flow in particle-based fluid simulations. In *ACM SIGGRAPH 2008 Papers*, SIGGRAPH '08, pages 49:1–49:8, New York, NY, USA. ACM.

Liu, Y., Zhu, H., Liu, X., and Wu, E. (2005). Real-time simulation of physically based on-surface flow. *The Visual Computer*, 21(8):727–734.

Lu, J., GEORGHIADES, A. S., Rushmeier, H., Dorsey, J., and Xu, C. (2005). Synthesis of material drying history: Phenomenon modeling, transferring and rendering. In *Eurographics Workshop on Natural Phenomena (NPH 2005)*, pages 7–16. Eurographics Association, Eurographics Association.

Lu, W., Jin, N., and Fedkiw, R. (2016). Two-way coupling of fluids to reduced deformable bodies. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA'16, pages 67–76, Goslar Germany, Germany. Eurographics Association.

Macklin, M. and Müller, M. (2013). Position based fluids. *ACM Trans. Graph.*, 32(4):104:1–104:12.

Macklin, M., Müller, M., Chentanez, N., and Kim, T.-Y. (2014). Unified particle physics for real-time applications. *ACM Trans. Graph.*, 33(4):153:1–153:12.

Mall, H.B., J. and Da Vitoria Lobo, N. (1995). Determining wet surfaces from dry. In *Computer Vision, 1995. Proceedings., Fifth International Conference on*, pages 963–968.

Merillou, S., Dischler, J. M., and Ghazanfarpour, D. (2000). A brdf postprocess to integrate porosity on rendered surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 6(4):306–318.

Mérillou, S. and Ghazanfarpour, D. (2008). A survey of aging and weathering phenomena in computer graphics. *Computers & Graphics*, 32(2):159–174.

Millington, I. (2010). *Game Physics Engine Development, Second Edition: How to Build a Robust Commercial-Grade Physics Engine for Your Game*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2nd edition.

Muggeridge, A., Cockin, A., Webb, K., Frampton, H., Collins, I., Moulds, T., and Salino, P. (2014). Recovery rates, enhanced oil recovery and technological limits. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 372(2006).

Müller, M., Charypar, D., and Gross, M. (2003). Particle-based fluid simulation for interactive applications. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '03, pages 154–159, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association.

Muskat, M. and Wyckoff, R. (1937). *The Flow of Homogeneous Fluids Through Porous Media*. International series in physics. McGraw-Hill Book Company, Incorporated.

Nolet, C., Poortinga, A., Roosjen, P., Bartholomeus, H., and Ruessink, G. (2014). Measuring and modeling the effect of surface moisture on the spectral reflectance of coastal beach sand. *PLOS ONE*, 9(11):1–9.

Nystrom, R. (2014). *Game Programming Patterns*. Genever Benning.

Oren, M. and Nayar, S. K. (1994). Generalization of lambert's reflectance model. In *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '94, pages 239–246, New York, NY, USA. ACM.

Patkar, S. and Chaudhuri, P. (2013a). Wetting of porous solids. *IEEE Transactions on Visualization and Computer Graphics*, 19(9):1592–1604.

Patkar, S. and Chaudhuri, P. (2013b). Wetting of porous solids. *IEEE Transactions on Visualization and Computer Graphics*, 19(9):1592–1604.

Pharr, M. and Humphreys, G. (2010). *Physically Based Rendering, Second Edition: From Theory To Implementation*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2nd edition.

Robinson-Mosher, A., English, R. E., and Fedkiw, R. (2009). Accurate tangential velocities for solid fluid coupling. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA'09, pages 227–236, New York, NY, USA. ACM.

Robinson-Mosher, A., Shinar, T., Gretarsson, J., Su, J., and Fedkiw, R. (2008). Two-way coupling of fluids to rigid and deformable solids and shells. *ACM Trans. Graph.*, 27(3):46:1–46:9.

Rungjiratananon, W., Kanamori, Y., and Nishita, T. (2012). Wetting effects in hair simulation. *Computer Graphics Forum*, 31(7):1993–2002.

Rungjiratananon, W., Szego, Z., Kanamori, Y., and Nishita, T. (2008). Real-time animation of sand-water interaction. *Computer Graphics Forum*, 27(7):1887–1893.

Schlick, C. (1994). An inexpensive brdf model for physically-based rendering. *Computer Graphics Forum*, 13(3):233–246.

Shepard, D. (1968). A two-dimensional interpolation function for irregularly-spaced data. In *Proceedings of the 1968 23rd ACM National Conference*, ACM '68, pages 517–524, New York, NY, USA. ACM.

Si, W., Lee, S.-H., Sifakis, E., and Terzopoulos, D. (2014). Realistic biomechanical simulation and control of human swimming. *ACM Trans. Graph.*, 34(1):10:1–10:15.

Stam, J. (1999). Stable fluids. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '99, pages 121–128, New York, NY, USA. ACM Press/Addison-Wesley Publishing Co.

Stern, F. (1964). Transmission of isotropic radiation across an interface between two dielectrics. *Appl. Opt.*, 3(1):111–113.

Teng, Y., Levin, D. I. W., and Kim, T. (2016). Eulerian solid-fluid coupling. *ACM Trans. Graph.*, 35(6):200:1–200:8.

Thürey, N., Müller-Fischer, M., Schirm, S., and Gross, M. H. (2007). Real-time breaking waves for shallow water simulations. *15th Pacific Conference on Computer Graphics and Applications (PG'07)*, pages 39–46.

Unity Technologies (2019). Unity - game engine.

Wang, L. and Whiting, E. (2016). Buoyancy optimization for computational fabrication. *Computer Graphics Forum*, 35(2):49–58.

Weidlich, A. and Wilkie, A. (2011). Thinking in layers: Modeling with layered materials. In *SIGGRAPH Asia 2011 Courses*, SA '11, pages 20:1–20:43, New York, NY, USA. ACM.

Wikipedia (2019). Component-based software engineering — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Component-based%20software%20engineering&oldid=877978926. [Online; accessed 23-January-2019].

Witkin, A. and Baraff, D. (1997). Physically based modeling: Principles and practice. In *ACM SIGGRAPH 1997 Courses*, SIGGRAPH'97, New York, NY, USA. ACM.

Yacoob, Y. (2013). Matching dry to wet materials. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 2952–2959.

Yan, H., Wang, Z., He, J., Chen, X., Wang, C., and Peng, Q. (2009). Real-time fluid simulation with adaptive sph. *Computer Animation and Virtual Worlds*, 20(2-3):417–426.

Yaz, I. O. and Loriot, S. (2018). Triangulated surface mesh segmentation. In *CGAL User and Reference Manual*. CGAL Editorial Board, 4.13 edition.

Yin, X., Shen, X., Zhang, F., and Huang, G. (2013). Particle-based simulation of fluid-solid coupling. In Tan, G., Yeo, G. K., Turner, S. J., and Teo, Y. M., editors, *AsiaSim 2013*, pages 373–378, Berlin, Heidelberg. Springer Berlin Heidelberg.

Yuksel, C., House, D. H., and Keyser, J. (2007). Wave particles. *ACM Trans. Graph.*, 26(3).

Zhang, H. and Voss, K. J. (2006). Bidirectional reflectance study on dry, wet, and submerged particulate layers: effects of pore liquid refractive index and translucent particle concentrations. *Appl. Opt.*, 45(34):8753–8763.