CLEI electronic journal, Volume 25, Number 1, Paper 4 , April 2022

A Comprehensive Method for Liquid-to-Solid Interactions.

Juan Miguel Bajo
Departamento de Ing. Eléctrica y Computadoras - Universidad Nacional del Sur and CONICET, Bahía Blanca, Argentina, 8000,
jmb@cs.uns.edu.ar

Claudio Delrieux
Departamento de Ing. Eléctrica y Computadoras - Universidad Nacional del Sur and CONICET, Bahía Blanca, Argentina, 8000,
cad@uns.edu.ar

Gustavo Patow
ViRVIG-UdG,
Girona, Spain,
gustavo.patow@udg.edu

Abstract

Realistic real-time water-solid interaction has been an open problem in Computer Graphics since its beginnings, mainly due to the complex interactions that happen at the interface between solid objects and liquids, both when objects are completely or partially wet, or when they are fully submerged. In this paper we present a method that tackles the two main aspects of this problem, namely the buoyancy of objects submerged into fluids, and the superficial liquid propagation and appearance changes that arise at the interface between the surface of solid objects in contact with a liquid. For the first problem (buoyancy) a method is proposed to realistically compute the fluid-to-solid coupling problem. Our proposal is suitable for a wide spectrum of cases, such as rigid or deformable objects, hollow or filled, permeable or impermeable, and with variable mass distribution. In the case of permeable materials, which allow liquid to pass through the object, the presented method incorporates the dynamics of the fluid in which the object is submerged, and decouples the computation of the physical quantities involved in the buoyancy force of the empty object with respect to to the liquid contained within it. On the other hand, the visual appearance of certain materials depends on their intrinsic light transfer properties, the lighting present and other environmental contributions. Thus, complementing the first approach in this paper, a new technique is introduced to model and render the appearance changes of absorbent materials when there is liquid on their surface. Also, a new method was developed to solve the problem of the interaction between the object surface and liquids, taking advantage of texture coordinates. An algorithm was proposed to model the main physical processes that occur on the surface of a wet or wet solid object. Finally, we model the change in appearance that typically arise in most materials in contact with fluids, and an algorithm is implemented achieving real-time performance. The complete solution is designed taking advantage of superscalar architectures and GPU acceleration, allowing a flexible integration with the pipelines of current graphic engines.

Keywords: Real-Time Rendering, Computer Graphics, Procedural Animation, Buoyancy, Wetting

1 Introduction

Solid-liquid interaction of objects totally or partially submerged in liquid media has been an active research topic in the area of Computer Graphics since its beginning. This topic is of particular importance in video games, virtual reality and other interactive applications, and has received significant attention for its relevance in simulators, serious games, and similar contexts. Despite its great importance in research, an interactive and complete model for the liquid-to-solid interaction is still an open problem, since most of the presented techniques use highly simplified 'proxy' geometries [1], or are not applicable to real-time solutions due to their computational complexity [2].

One of the challenges regarding this problem is the requirement to quickly and accurately compute the variables involved in the dynamics of movement, taking into account situations such as, among other things, liquid entering the object and moving within it. The first contribution presented here is to propose an algorithm for the approximate, although realistic, computation of the two-way coupling problem, that is, solid-to-liquid interaction, and vice versa, simultaneously. Currently, there are satisfactory solutions for only a part of the problem corresponding to the solid-to-liquid coupling [3], therefore here we will focus on the complementary problem, the liquid-to-solid interaction. The fundamental concept underlying the proposed solution is that with a reasonable parameterization, the textures traditionally used for color and other visual characteristics can be used to compute in a scalable way all the physical quantities necessary to simulate the system in question accordingly.

Other aspect of the liquid-to-solid interaction problem is that the color of many materials (especially porous, rough or absorbent ones) appear darker and more vividly colored when wet. This easily observable phenomenon can be explained by the complex interactions of light that take place on the surface of the object when there is a layer of translucent liquid on its surface. In other words, the presence of a layer of liquid generates a crisper surface, generating a more vivid visual appearance and, at the same time, increases the proportion of light that is absorbed by the material, darkening and saturating the perceived color. An example of this is the water-sand-wind-sun interaction on beaches. Our perception offers several clues about the properties of the scene, for example, if the sand is fine or coarse, how the sea water is draining and evaporating, or if the surface can be slippery, to name just a few examples. These types of perceptions are naturalized through our daily experience, but they are difficult to model in the area of photo-realistic rendering. The optical effects that arise on wet surfaces and the appropriate models for diffusion, absorption and evaporation in porous materials have been studied in detail in Physics, Computer Vision and other disciplines, but so far, only a few consistent models have been proposed in Computer Graphics. Thus, we also present a Physics-inspired comprehensive model for rendering wet absorbent materials. Although our proposal is phenomenological in nature, it is intended to be comprehensive by taking into account most of the relevant factors that lead to the final visual appearance of wet surfaces, including the geometry of the object, the micro-physical properties of the material (e.g. example, its porosity and micro texture) and the dynamic behavior of the liquid, including absorption, diffusion and evaporation. This provides the ability to model reasonably wet absorbent materials in an intuitive and flexible manner. Although the model is a simplification of the underlying physics, the results are realistic and can be calculated in real time.

Contributions

- To the best of our knowledge, this is the first method capable of calculating all the significant magnitudes necessary to describe the buoyancy movement of a 3D model in real time, including the center of gravity, inertia momentums, center of buoyancy, friction, among others. The proposed method supports deformable or variable-shaped objects, since it can compute the required quantities instantaneously and continuously.

- We also propose a new data structure and an algorithm to dynamically represent the liquid within a submerged 3D object, in addition to the mechanisms to calculate its entry and displacement within the submerged object depending on its movement. This makes possible to simulate not only the buoyancy of a submerged object, but also its behavior in a realistic way, which, to our knowledge, is not generally possible so far.

- The proposed technique does not require manual processing of the input models, since it uses the parameterization typically used for texture mapping at the stage of artistic creation of the model.

- We also developed a representation and modeling technique for wet materials inspired by the underlying physics.

- We implement a computational method for photo-realistic simulation and representation of liquid dynamics on a surface, based on the underlying physical processes of diffusion, evaporation and absorption.

- Finally, we integrate these processes in a real-time pipeline that takes advantage of the characteristics of the latest graphics engines and that is fully integrable in most of the development libraries and frameworks in Graphics Computing.

2 Buoyancy Simulation

In this section we will describe a novel method to compute all the variables involved in the phenomenon of buoyancy and the liquid-solid interaction of submerged objects (partially or totally). These variables include center of mass, total mass, inertia tensor, and others. Regarding the liquid-solid interaction, a method for impermeable solid objects floating in a liquid medium will be presented.

For this case, a parameterization $\mathbb{R}^3 \to \mathbb{R}^2$ of the model polygon mesh is needed, typically used for the texture mapping technique [4]. With this information it is possible to directly compute the variables involved in the buoyancy simulation. By employing the parameterization used for texture mapping, the method takes advantage of the multi-parallel architecture of the graphics processor (GPU), allowing the necessary variables to be computed in real time and accurately, without the need to add extra information to the geometry by part of the artist. The method can be used for deformable or dynamic models since the metrics can be computed interactively. For the simulation of the solid $\to$ liquid interaction there are different solutions; in particular, the implementation of our solution is compatible with the work proposed by [3].

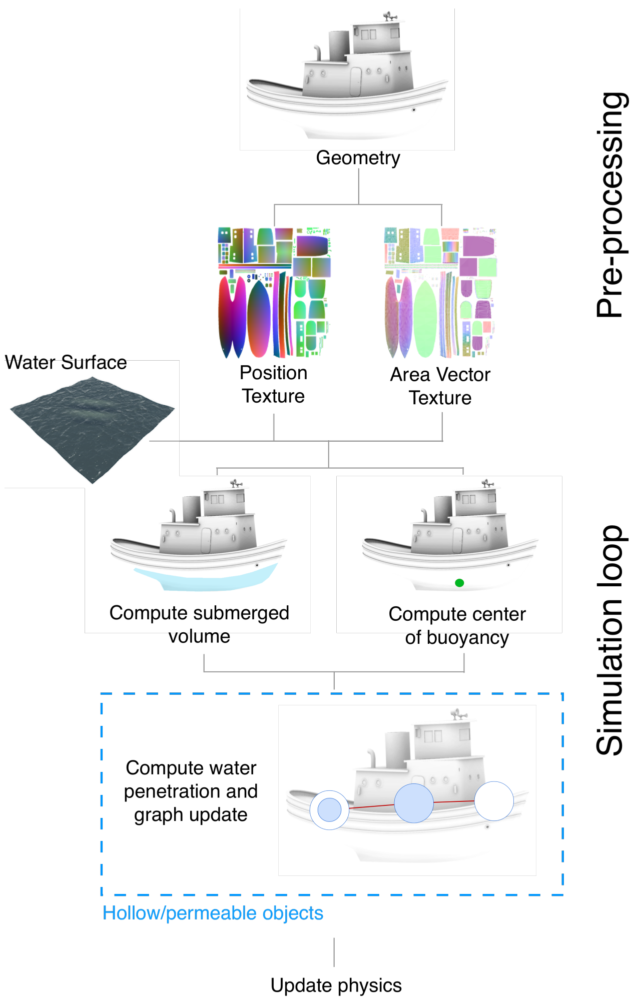

Figure 1: General description of our method: from the parameterized input model we calculate two additional textures (the position texture and the vector-area texture), used to calculate the parameters of the physical model (for example, submerged volume, center of buoyancy, tensor of inertia, etc.). Finally, we calculate the penetration of water in the geometry of the model, and we use an auxiliary structure, the liquid graph, to obtain the relevant physical quantities. This method will be explained in the next section.

2.1 Mathematical model

To solve the floating motion accurately, the method requires the evaluation of certain physical magnitudes corresponding to the modeled dry object (without liquid contained within), these are total mass, tensor of inertia and center of mass. These magnitudes are automatically generated and can be precomputed in case the object is not deformed during the simulation or they can be calculated in each frame of the simulation, clearly with a performance penalty.

Given a model $\mathcal{M}$, the properties of mass are: the total mass M, the center of mass C and the inertial tensor I, given by a symmetric 3x3 matrix. Assuming that the surface $\mathcal{M}$ encloses a volume $\Omega \in \mathbb{R}^3$ that corresponds to the solid of constant density ρ, we can express the properties of mass as [5]:

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

calculating the volume integral over Ω:

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

where

$$\mathbf{s}_t(\rho) = \rho \int_\Omega t \; dV. \tag{3}$$

The terms defined in the equation 2 can be expressed as surface integrals using the *Divergence Theorem.* For this we identify a vector field $\mathbf{B}$ for each b component of $\mathbf{b}$ such that $\nabla \cdot \mathbf{B} = b$:

$$\mathbf{B} = \left[\begin{array}{c|ccc|ccc|ccc} 0 & x^2/2 & 0 & 0 & x^2y/2 & 0 & 0 & x^3/3 & 0 & 0 \\ y & 0 & y^2/2 & 0 & 0 & y^2z/2 & 0 & 0 & y^3/3 & 0 \\ 0 & 0 & 0 & z^2/2 & 0 & 0 & z^2x/2 & 0 & 0 & z^3/3 \end{array} \right]. \tag{4}$$

Then

$$\mathbf{s}(\rho) = \rho \int_\Omega \mathbf{b} \; dV = \rho \int_\Omega \nabla \cdot \mathbf{B} \; dV = \rho \int_{\partial\Omega} \mathbf{n} \cdot \mathbf{B} \; dS, \tag{5}$$

where $\mathbf{n}$ is the normal vector to surface.

2.2 Data structures

As shown in the previous section, it is possible to express the components of the vector $\mathbf{s}(\rho)$ as surface integrals on the object. In order to implement the computation of integrals in a scalable way and fast enough for a real-time solution, we propose the data structures that are detailed below.

2.2.1 Position texture

In order to have a scalable and fast enough technique to obtain this information, a new type of three-channel texture called *Position Texture* (PT) is defined. This concept requires the geometry of the solid model and its corresponding parameterization on a 2D space; typically, classic texture mapping parameterization will be used. The PT is automatically generated from the polygon mesh and its UV texture mapping, commonly provided by the artist (or is generated procedurally). In it, each texel stores its own position in object coordinates. From a mathematical point of view, the PT stores the reverse texture mapping ($\mathbb{R}^2 \to \mathbb{R}^3$). It is important to clarify that the texture mapping must be bijective, otherwise we would lose information during the computation of the inverse function because the same texel could have two different object positions. While bijective texture mapping is an extra requirement, it is currently not considered excessive.

The process of generating the PT consists of taking each one of the polygons of the mesh of the object and transforming them to the texture space. Assuming meshes formed by triangles, this process can be done by taking the three vertices forming a face, projecting them into the texture space using UV mapping and then interpolating the intermediate texels using barycentric coordinates.

Let M be a bijective texture mapping, $P = P_0P_1P_2$ a triangle of the polygon mesh, $P_t = M(P) = T_0T_1T_2$ the triangle in texture coordinates and T a texel within the polygon P_t. In this way, T is a vector whose value is given by

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

where λ_0, λ_1 y λ_2 are the baricentric coordinates of T relative to T_0, T_1 y T_2 (See Figure 2).

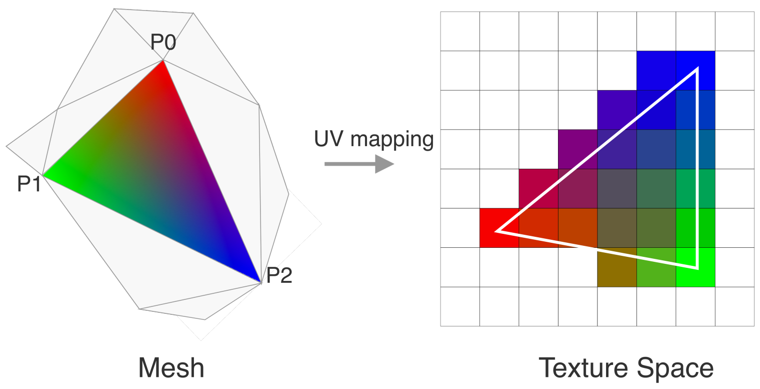

Figure 2: The texture generation method consists of *rasterizing* the triangle of the polygonal mesh into a synthetic texture, interpolating the corresponding vertex attribute from the barycentric coordinates. For example, for the position texture, the vertex position is interpolated in object coordinates. For the vector-area texture, the normal vector of each vertex is interpolated.

2.2.2 Vector-area Texture

In addition to the PT presented in the previous subsection, we need more information from the texel to be able to compute the surface integrals, components of the vector $\mathbf{s}(\rho)$. In particular, we need the normal vector to the surface and the area of the integration segments (dS). From this we will define another data structure, called *Vector-Area Texture* (VAT). Let T_i be the i-th texel of the VAT, it stores a 3D vector whose direction is parallel to the normal of the surface $\mathbf{n}_i$ at given point and its magnitude is the area (a_i) of the surface occupied by the texel in world coordinates, that is, it stores $\mathbf{A}_i = a_i\mathbf{n}_i$ as a 4-tuple:

$$T_i =< n_i^x, n_i^y, n_i^z, a_i >, \tag{6}$$

where T_i, y n_i^x, n_i^y y n_i^z are the components of the unit normal vector of the texel and a_i is the area of the surface occupied by the texel calculated as:

$$a_i = \frac{A_P}{N_T}, \tag{7}$$

where A_P is the surface area of the polygon P in which the texel T_i is mapped and N_T is the number os texels to which P is mapped.

Clearly these values must be linearly transformed according to the depth (Bit Depth) of the image used. For example, in 8-bit images per channel and integer data type, the values will be encoded between 0 and 255.

2.3 Liquid-to-solid coupling

Using the terms defined in Section 2.1 and the data structures defined above, we propose a computational method to calculate the properties of mass: total mass (M), center of mass (C_M), and inertia tensor ($\mathbf{I}$).

$$M = s_1,$$

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

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

When a certain object is submerged in liquid, its dynamics is defined by the contact surface between the two. The method we present is independent of the technique used to implement liquid dynamics.

2.3.1 Submerged volume computation

When a certain object is submerged in liquid, its dynamics is defined by the contact surface between the two. To compute the portion of submerged volume, the following expression is applied graphically explained

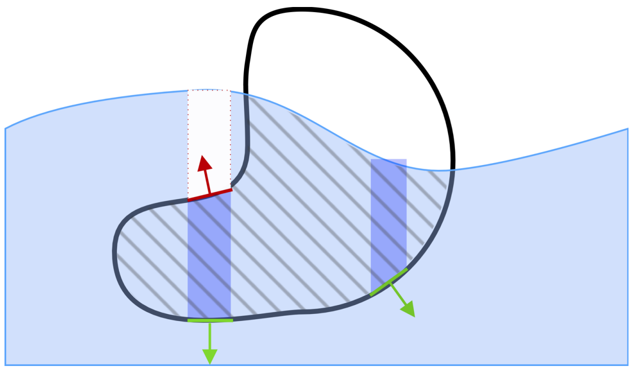
Figure 3: Geometric interpretation of the submerged volume count (hatched region): Texels whose normal has a negative vertical component (green) contribute to the total submerged volume. On the other hand, the texels whose normal has a positive vertical component (red) subtract volume from the total. Repeating this computation and calculating the total sum of all the contributions of the texels, we calculate an approximation of the volume of the submerged portion of the solid.

in Figure 3:

$$S_V = \int_{\mathcal{V}_{sub}} dv = \oint_{\partial\mathcal{V}_{sub}} \mathbf{n}\cdot\mathbf{r}ds, \tag{8}$$

with the precaution of integrating only over the submerged volume portion $\mathcal{V}_{sub}$ and not on the total volume of the object $\mathcal{V}$. It is straightforward that the submerged volume $\mathcal{V}_{sub}$ is enclosed between the surface of the liquid and the surface of the object located below it, therefore $\mathcal{V}_{sub}$ it can be calculated by integration between these two limits. In addition, the use of VAT to correctly project the texel area in the y-axis direction can be observed.

3 Buoyancy Simulation in Permeable Objects

This section presents the most important results in the representation and simulation of the behavior of objects whose floating dynamics can change over time. This dynamic may be due to the fact that the objects are made of permeable material, or because they are shallow and liquid can move inside them.

3.1 Liquid graph

To model these cases we propose an extension of the method presented above incorporating a new data structure called *Liquid Graph* (LG). This structure is used to approximate the distribution of liquid within the object and simulate its movement through the internal structure, taking into account its geometry. Mathematically, the LG is defined as

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

where N is a set of vertices or nodes N_i representing the available space within the object that can be filled with liquid, either hollow space or distributed space within the material and is defined by

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

where $\mathbf{x_i}$ is the position of the node in object coordinates, c_i is the volume of liquid that it can store before being saturated (capacity) and l_i the volume of liquid present in the node, the latter being expressed in liters. In addition,

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

is the set of pairs of nodes N_i and N_k such that their represented spaces are connected and p^g_{ik} is the average permeability of the material found between the spaces represented by the nodes.

3.2 Automatic construction

The *Liquid Graph*, if relatively simple, can be created in a supervised (manual) manner by an artist or model creator. An alternative is to implement an algorithm for the automatic generation of the graph. For this, a polygonal mesh segmentation algorithm is required to partition the model into the different volumes that compose it.

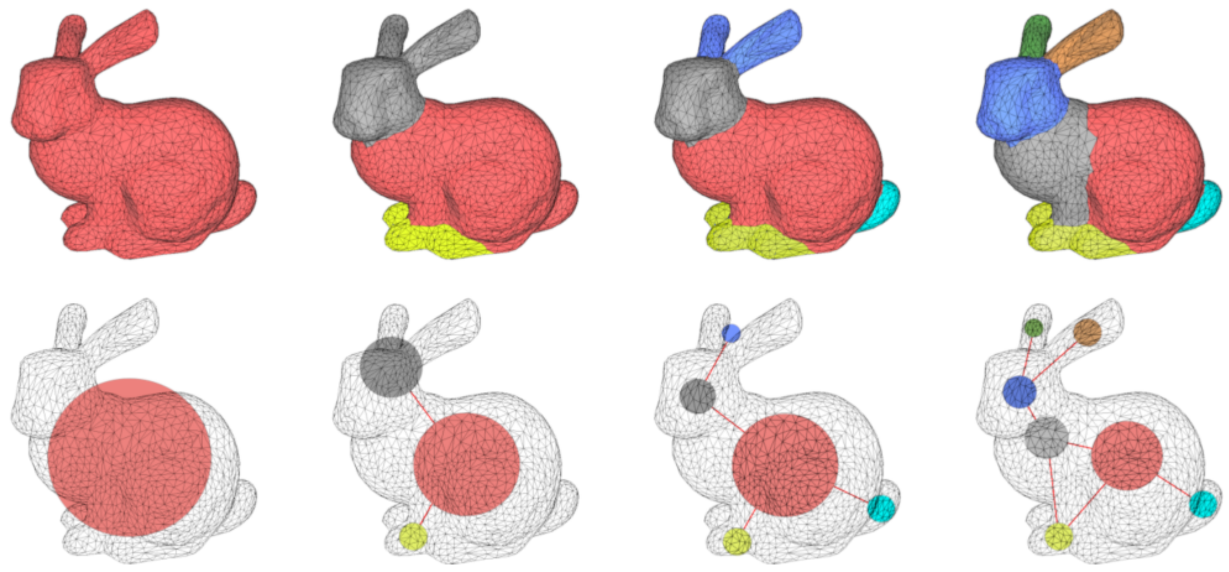
Figure 4: Construction of the *Liquid Graph*: Mesh segmentation and creation of nodes for different amount of sub-meshes. The colors indicate the node corresponding to each sub-mesh.

1. Initially the algorithm calculates a segmentation of the input polygon mesh as follows:
2. Subsequently, the algorithm calculates, for each sub-mesh generated by the segmentation algorithm, a sphere centered on the geometric center of the sub-mesh such as to minimize the quadratic error between the surface of the sphere and the vertices of the sub-mesh.
3. Finally, with all the nodes generated, an edge is added to the graph between two nodes in the event that the sub-meshes represented by them are adjacent, that is, they share some vertex or edge in the original mesh.

Figure 4 shows the same model automatically segmented with different number of groups (clusters) in the grouping algorithm, and the graph generated from each of the segmentations is also shown. Once the graph is created and the capacities of each node established together with the connectivity between them, either manually or automatically with the method described in the previous section, the dynamics of the liquid within the object can be evaluated. The method is proposed to work in real time, therefore it is feasible to repeat it in each frame of an animation.

4 Wet Materials

The visual appearance of some materials changes significantly when wet. Colors are typically altered by increasing their saturation and decreasing their intensity, resulting in darker and more vivid colors. The proposed method consists of a three-stage processing pipeline for modeling and rendering wet materials. Initially, the polygon mesh is pre-processed and a special data structure is created. Later, during the simulation, the dynamics of the absorbed liquid is calculated and the results are updated. Finally, the mesh is rendered in real time using the data structures generated during the simulation. The processing pipeline, shown in Figure 5, is detailed below:
In the pre-processing stage:

1. Process the polygon mesh and create a special texture called *Position-Area texture* (PAT) which encodes, for each texel, its position together with the surface area covered by it, both in object coordinates. This process is done only once per model. The PAT texture in a combination of the two textures presented above. In this case, the area vector is not used for the computation of the simulation, therefore it is not included in the generated structures. In the case that both buoyancy and change of appearance are simulated, the more general pair of textures PT and VAT can be used, replacing the PAT.

During the simulation cycle:

2. Update the state of the fluid according to the model used to simulate it.

3. Update the *Liquid Texture* (TL), which stores the amount of liquid on the surface of the model, solving the liquid-object collision from the PAT. This process consists of:
   (a) Compute the income of the liquid from the Darcy model.
   (b) Compute the intra-object diffusion process according to Fick's law from the creation of a *Liquid Density Field* (LDF) in order to perform the simulation in 3D space.
   (c) Compute the evaporation of the liquid.
4. Render the scene using a matching model for wet surfaces.

Although our implementation uses a 2D grid to simulate the liquid, the method can be adapted to other fluid modeling techniques, such as particle systems or 3D grids.

4.1 Data structures

In order to simulate liquids and their interaction with solids in a precise and scalable way, when using the texture space as a resource, it is necessary to establish a bidirectional correspondence between this 2D space (where the absorbed liquid information is stored) and the space of the 3D world (where the object is immersed and the phenomena happen properly). For this the PAT allows to store spatial and geometric information of each texel. This texture is a simplification that arises from combining together the PT and VAT presented above.

Furthermore, in order to simulate the dynamics of the liquid, it is necessary to define a corresponding data structure to store the amount of liquid absorbed at each point on the surface of the object. For this structure, the mesh parameterization will also be used for texture mapping and the necessary information will be stored in a mono-channel texture called *Liquid Texture* (LT), where each texel stores the amount of liquid per unit area at its position on the model surface (see Figure 6).

4.2 Liquid dynamics in porous material

Absorption, diffusion and evaporation are clearly the most noticeable effects relative to liquid in contact with absorbent or porous materials. For this reason, because our method is focused on proposing a visually realistic method in real time, we will present a solution for its simulation. In this section we will present the models that will be used to simulate the different phenomena and how to implement them in a scalable way from the structures presented above.

4.2.1 Absortion

In order to model the ingress of liquid in permeable models, an approximation based on Muskat's work on Darcy's Law is used [6], which consists of

$$\Phi_A = -\frac{C_{PM} A \Delta\rho}{\mu L}, \tag{9}$$

where Φ_A is the flow per unit time, C_{PM} is the permeability of the medium defined globally for the whole model or by texel in a *permeability texture* (PMT), A is the area of the flow, μ is the viscosity of the liquid and $\Delta\rho$ is the pressure difference in the contact area measured along a section of length L.

4.2.2 Evaporation

Evaporation is the process of phase change from a liquid to a gaseous state by a liquid substance and the associated transport of the resulting vapor [7].

Acording to Hall and Hoff [7], the *Evaporation Rate* e can be modeled as

$$e = -\frac{p_{w0} D_w \rho_w M}{R\tau} \frac{dH}{dx}, \tag{10}$$

where H is fractional relative humidity, ρ_w is the density of the liquid, M the molar mass, D_w is the diffusivity of the liquid in air, R is the *universal constant of the gases* and τ the ambient temperature.

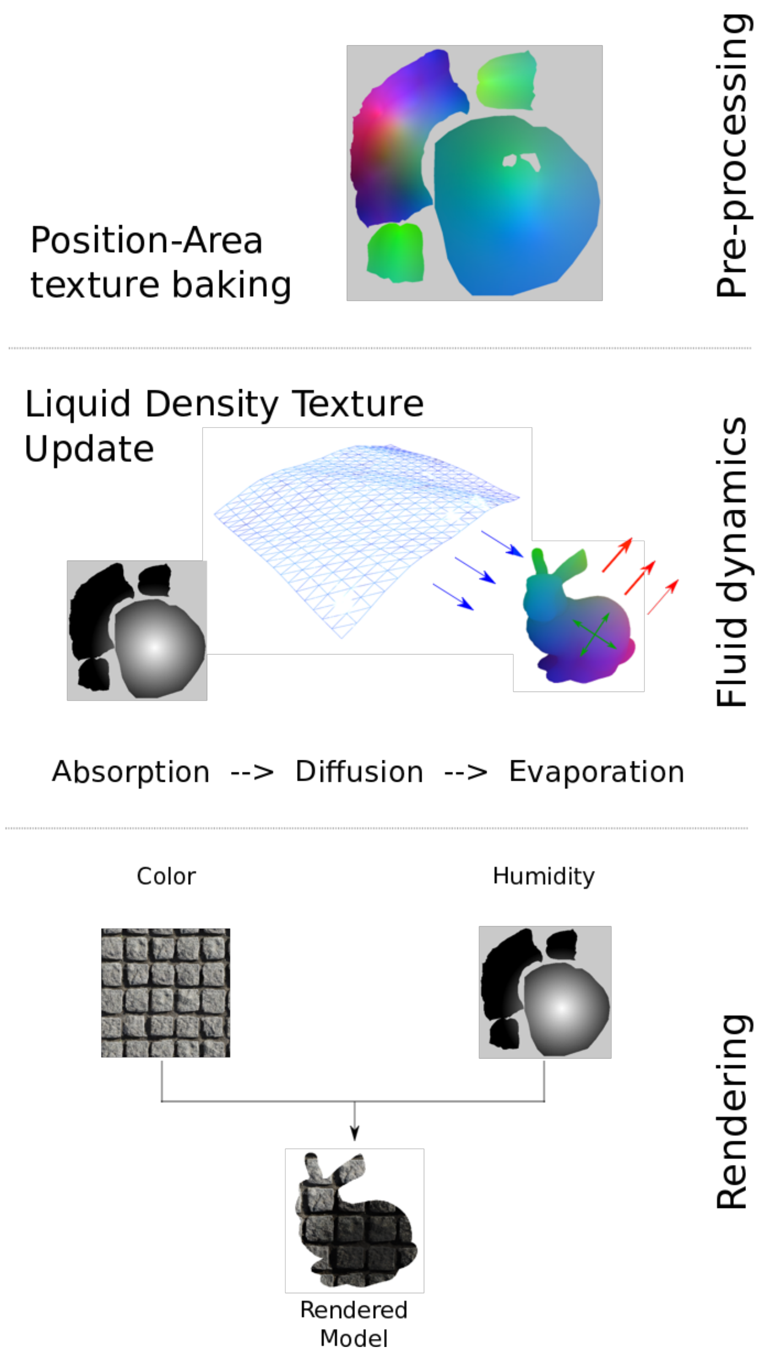

Figure 5: Processing *Pipeline*. In the first instance, the geometry is pre-processed and the PAT is created. Next, the absorbed liquid simulation is updated according to the simulated phenomena by updating the liquid density texture. Finally, the geometry is rendered taking into account the information obtained in the previous stages.

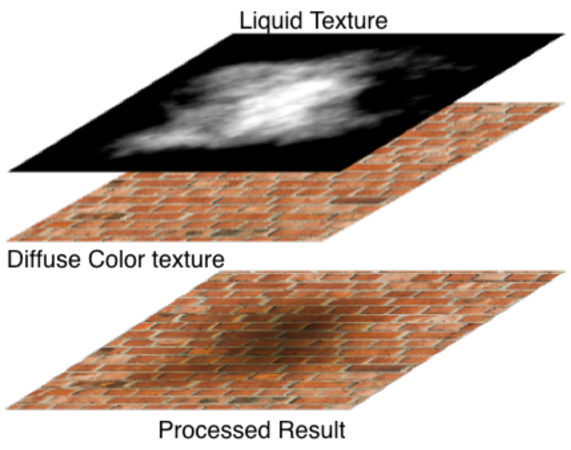

Figure 6: A *Liquid Texture* (LT) is defined that stores the amount of liquid per unit area in each texel. This texture is updated according to the interaction of the model with the liquid present in the scene interactively using the positions of the texels stored in the PAT. This texture will be used for rendering, during the final stage of the pipeline.

4.2.3 Diffusion

Diffusion is the process by which matter is transported, due to the random movement of molecules, from one point in space to another. This process can be modeled using the first *Fick's Law* [8, 9] which states that, when there are regions with different concentrations of matter, a flow is generated from regions with higher concentration to regions with higher concentration more low. Mathematically, *Fick's Law* is expressed as

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

where $\mathbf{J}$ is the diffusion vector of the flow, D is the diffusion coefficient, which depends entirely on the material, and ϕ a scalar field representing the concentration.

4.3 Render model

Precise rendering of a certain surface requires complex modeling of light transport. In particular, wet absorbent materials exhibit a particular effect that consists in changing the appearance of the perceived color, specifically in saturation and brightness. A plausible explanation for the cause of this phenomenon is related to light behavior in the interface between the object's surface and the thin liquid area. In order to propose a real-time rendering technique inspired by the underlying physical phenomena, this section is based on the work published by [10], which is an extension of [11]. In both works, the authors propose a Physics-based model that explains why absorbent materials appear darker when wet. According to these studies, a fraction of the light reflected diffusely on the surface of the object is reflected back into the object at the water-liquid interface, generating an increase in the total internal reflection (see Figure 7) producing a darker visual appearance. In addition, the authors calculate the probability of total internal reflection. In short, a larger proportion of the diffuse reflection on the object's surface is retained and bounced back by the liquid layer, therefore intensifying the object's hue (increased saturation) but decreasing the overall reflection (decreased brightness). Based on this model, a suitable rendering method is proposed to obtain realistic images of wet material surfaces in real time. The model assumes that the object has a distinguishable, rough surface. Also, our model can be extended to include colored liquids.

5 Results

As an example of our method, the images of the buoy shown in Figure 8 are simulated using the method proposed for buoyancy. The same animation is shown in Figure 9 rendered with the corresponding forces superimposed with an orthographic projection. Figure 10 shows some frames of the animation of a sinking ship. The permeability of the object is defined by texel and stored in the PMT simulating a damaged section in the right front part of the ship's hull. The PMT is shown in the upper left image. The same ship model is shown in Figure 11 but with a different PMT. In this case, the damaged section is on both sides of the rear section of the helmet. It is important to note that the computation times do not depend on the size of the polygon mesh of the model since the method is completely texture-based.

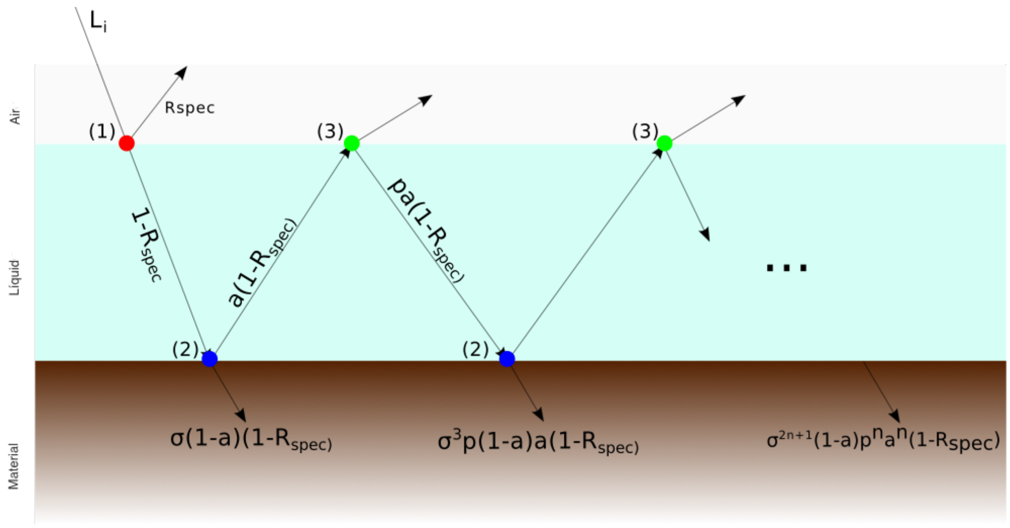
L_i
Rspec
Air
Liquid
Material
(1)
(2)
(3)
$1-R_{spec}$
$a(1-R_{spec})$
$pa(1-R_{spec})$
$\sigma(1-a)(1-R_{spec})$
$\sigma^3 p(1-a)a(1-R_{spec})$
$\sigma^{2n+1}(1-a)p^n a^n(1-R_{spec})$
...

Figure 7: Model of the interaction of light with a wet material. First, a portion of the incoming light beam is reflected at the air-liquid interface (red dot) (1). Next, the transmitted portion reaches the surface of the object and is again decomposed into two different interactions: one is absorbed by the object and the other is reflected. The proportion of reflected light determines the diffuse color of the material (albedo) (blue dots) (2). Subsequently, the reflected ray is again decomposed in two directions at the liquid-air interface (green dots) (3), part of the light is transmitted outwards and another part is reflected back towards the surface of the object. This process is repeated indefinitely.

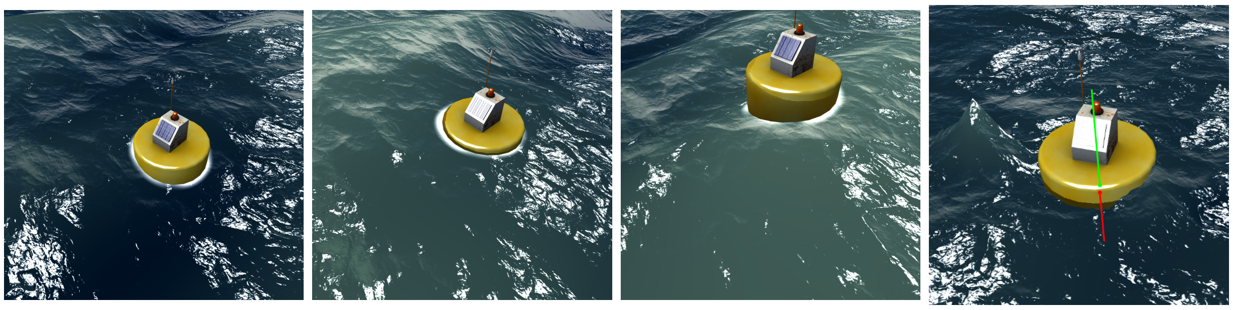

Figure 8: Frames of an animation showing the simulation of a buoy. The buoy is not permeable, that is, no liquid enters its interior. Using the position in the world of the texels, the areas in contact with water are rendered in accordance with the method described in section 4.

Figure 12 shows a set of tables of an unstable floating simulation. Since the center of buoyancy is below the object's center of mass, the object does not tend to regain its initial rotation. Compare this behavior with ship simulations, in which the object, despite being disturbed by the waves, tended to return to its initial rotational position. In the particular case of this simulation, the ball does not return to its initial rotation; for that reason, it is called unstable.

On the other hand, the presented method can also be used to simulate objects whose mass distribution changes over time. In these cases, the physical quantities involved in the simulation need to be computed in each frame of the simulation, since they change continuously and cannot be precomputed. Figure 13 shows a ball with variable mass distribution. The mass density function is displayed encoded in a pseudo-color scheme on the model; the red areas indicate a higher density of mass and the green areas, lower. The distribution corresponding to each box is shown in a box in the lower left area in each of the boxes. In this case, unlike the ship and buoy simulations where the mass distribution is constant over time, the magnitudes must be re-computed in each frame. This simulation shows that the computation times, even in a general case where certain magnitudes cannot be pre-computed, remain consistent with interactive applications. Furthermore, the proposed model is capable of working with deformable and / or animated models if the PAT is computed in each frame, for example, with *render to texture* techniques.

Figure 14 shows a plush object that tends to absorb liquid through its permeable surface following a diffusion propagation model. In this case, the liquid graph is generated automatically with the method explained above. Figure 15 shows a simulation of a deformable model showing that the method also works correctly for simulations of objects immersed in gaseous media. The island scene shown in Figure 16 is an

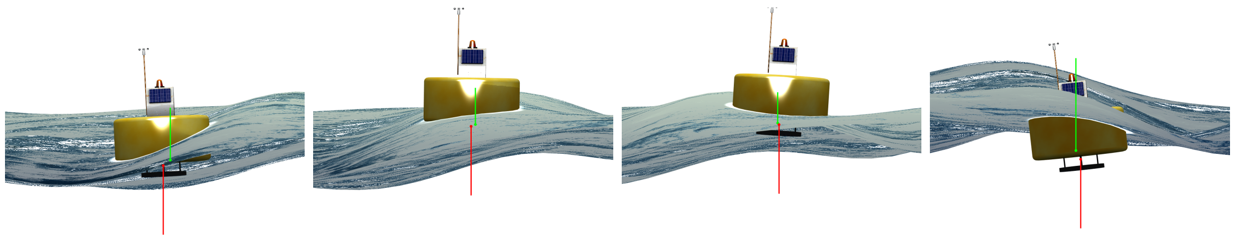

Figure 9: Frames of the animation of the buoy at different instants of time showing the forces corresponding to the weight and the thrust in orthographic perspective. It is possible to observe the relative positions and magnitudes of the weight force (red vector) and thrust (green vector) according to the submerged portion of the model. Clearly the weight force remains constant throughout the simulation because no liquid enters the model; therefore, its mass is constant.

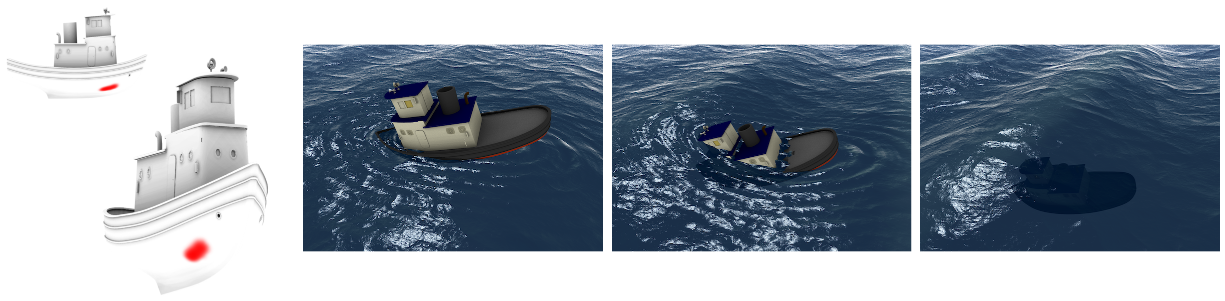

Figure 10: Frames of the simulation of the sinking of a ship. The upper left image shows the PMT from different points of view. Red texels are permeable while white ones are not.

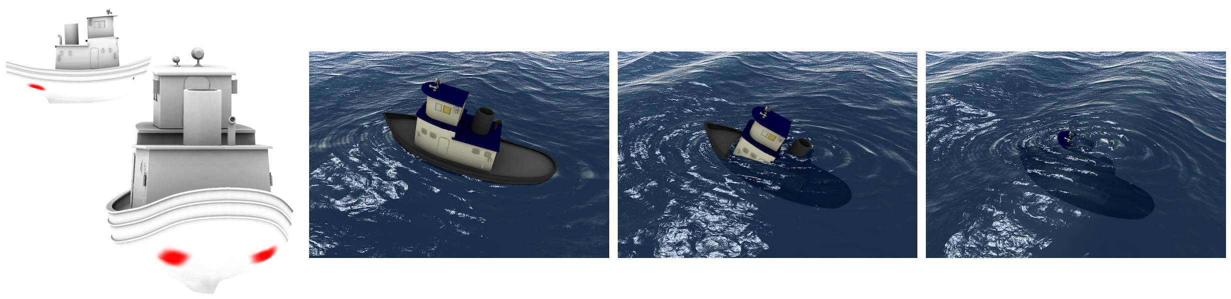

Figure 11: Frames of the simulation of the sinking of a ship with an alternative PMT to the one used in the figure 10. The upper left image shows the PMT from different points of view. Red texels are permeable while white ones are not.

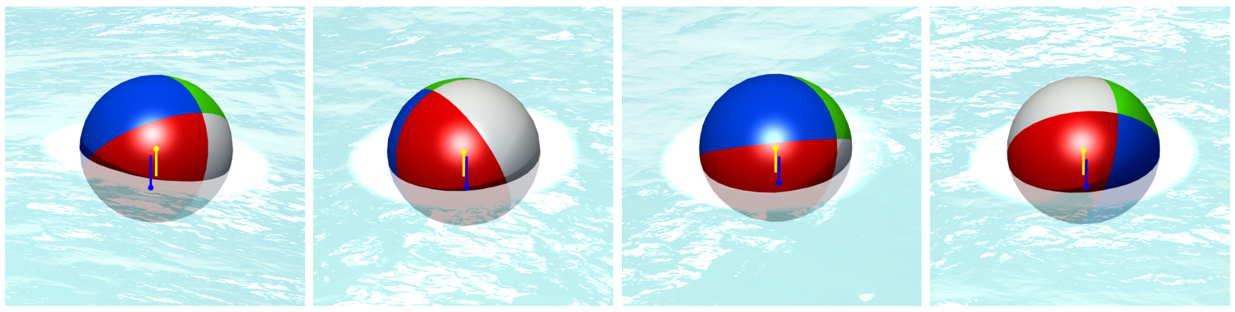

Figure 12: Different frames of the unstable buoyancy simulation. The ball has a uniform density of mass on the surface, therefore its center of mass is located at the geometric center of the object. On the other hand, the center of buoyancy is always located below the center of mass (since the object is not totally submerged) generating an unstable movement, that is, the object does not tend to recover its initial rotation.

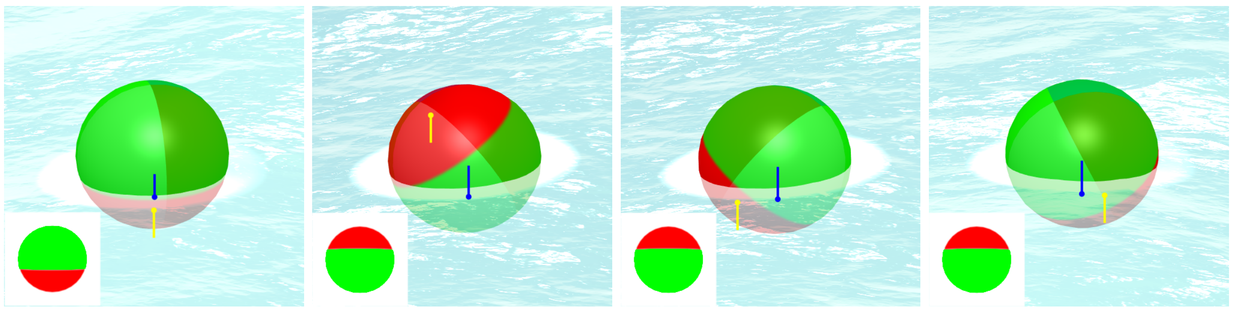
Figure 13: Frames of a simulation of an object with variable mass distribution over time. The proposed method can also be used with objects of variable mass (for example, a small boat with people moving inside it). The fact of using textures as data structures allows the method to calculate the physical magnitudes of the system in real-time and according to the state of the system (see the mass distribution in the lower left box of each table). In the figure, the color map shows the mass density function in each box: red areas show higher mass density and green areas lower. The yellow vector is the weight force and the blue vector the buoyant force. In this simulation, the total mass, the center of mass and the inertial tensor were computed in each frame using textures of 1024x1024 texels. Average frame computation time was 3.2 milliseconds

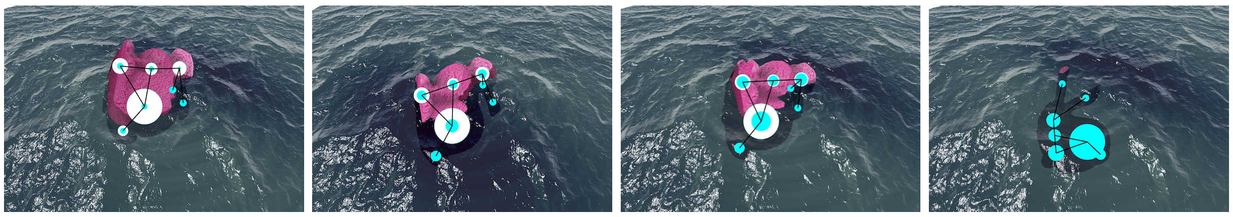
Figure 14: Frames of the simulation of a plush doll using the automatically generated graph shown in the figure 4.

example of the wet porous materials simulation algorithm and consists of four models, 250k triangles in total, that simulate the presence of moisture on its surface. Each animation frame takes, on average, 11 milliseconds to process. Figure 17 shows the results using an artistically created *Liquid Texture* and the rendering model simulating colored translucent liquid. Figure 18 compares a real-world scene, captured with a high-quality camera, with the results of our method. All the videos of the different simulations presented here can be accessed from `https://youtu.be/67D1G0n-5lk` and `https://youtu.be/JxkOrAjlv6M`.

6 Conclusion

We presented an encompassing approach to model the different interactions that occur between a liquid and a solid, consisting of two approaches, one to solve the liquid-to-solid coupling, and the other to realistically model the appearance and evolution of wet materials. With respect to the first approach, and to the best of our knowledge, this is the first method capable of calculating the significant magnitudes necessary

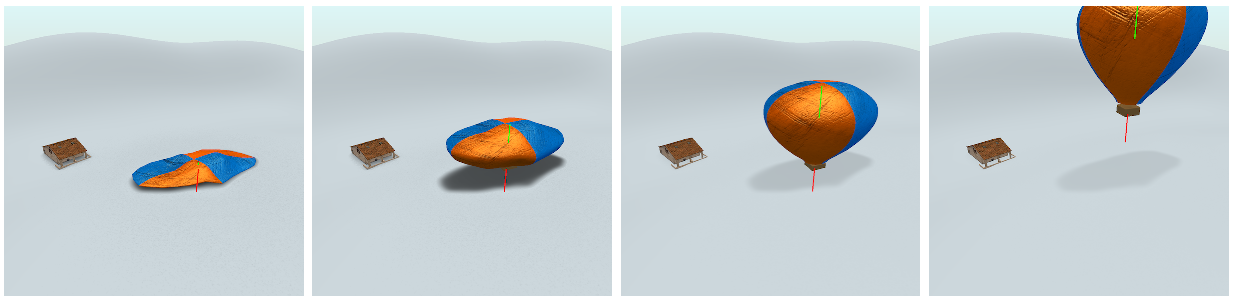
Figure 15: Frames of the simulation of a hot air balloon showing the operation of the proposed method with animated polygonal meshes. In this case the object is immersed in a gaseous medium (air). The red vector shows the weight force, while the green one shows the thrust force that depends on the volume of air displaced.

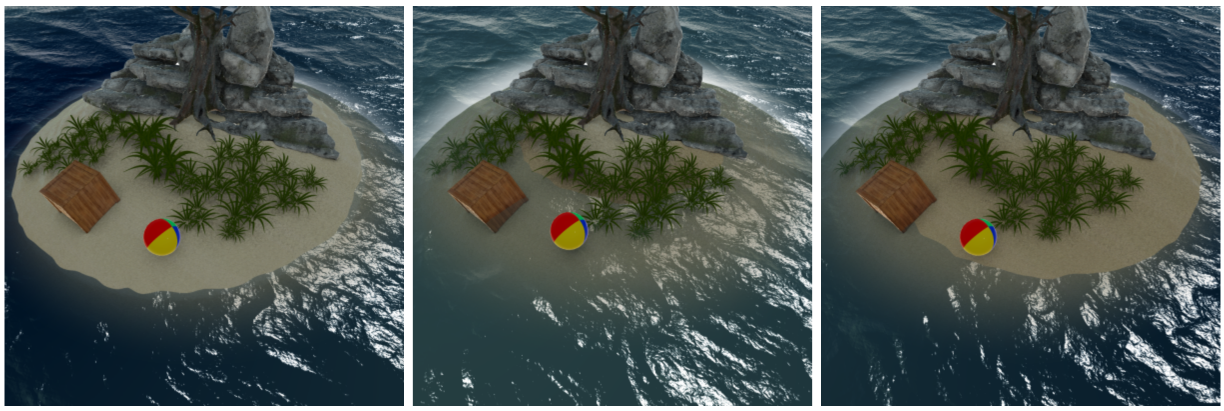

Figure 16: Frames of the beach simulation, showing the effect of the diffusion process. Portions of sand out of reach of the water that are moistened by this effect. Changes in appearance and reflectivity between wet and dry areas are easily noticeable.

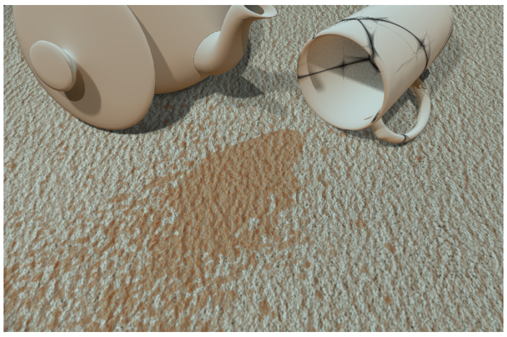

Figure 17: Image rendered with colored liquid from an artist-specified LT.

to describe the buoyancy movement of a 3D model in real time: center of gravity, moments of inertia, center of buoyancy, friction, among others. As previously mentioned, our method supports deformable or variable-shaped objects, as it can compute the required quantities instantaneously and continuously. Together with this fast calculation approach, we provide a new set of data structures, together with an algorithm, which enable dynamical representations of the liquid within a submerged 3D object. This, together with a mechanism to calculate water entry and displacement within the submerged object depending on its movement, results in a comprehensive model for buoyancy. By combining these atomic algorithms and data structures, it is now possible to simulate not only the buoyancy of a submerged object, but also its behavior in a realistic way, which, to our knowledge, is not generally possible so far.

Our second approach, a representation and modeling technique for wet materials inspired by the underlying Physics, implements a computational method for photo-realistic simulation and representation of liquid dynamics on a surface, based on the underlying physical processes of diffusion, evaporation and absorption. Finally, we integrate these processes in a real-time pipeline that takes advantage of the characteristics of the latest graphics engines and that is fully integrable in most of the development libraries and frameworks in Graphics Computing. Also, the proposed techniques do not require manual processing of the input models, since they uses the parameterization typically used for texture mapping at the stage of artistic creation of the model. Thus, this enables to use the models without any costly pre-processing stage, rendering these techniques suitable for video-games and other real-time applications.

7 Acknowledgments

This work was partially funded by project TIN2017-88515-C2-2-R from Ministerio de Ciencia, Innovación y Universidades, Spain, project 24/K083 from the SECyT, Universidad Nacional del Sur, Argentina, and a scholarship by the National Science and Technology Council of Argentina (CONICET).

References

[1] C. Gonzalez-Ochoa, “Advances in real-time rendering in games: Rendering rapids in uncharted 4,” in *SIGGRAPH ’16: ACM SIGGRAPH 2016 Courses.* New York, NY, USA: ACM, 2016.

[2] W. Lu, N. Jin, and R. Fedkiw, “Two-way coupling of fluids to reduced deformable bodies,” in *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ser. SCA’16. Goslar Germany, Germany: Eurographics Association, 2016, pp. 67–76. [Online]. Available: http://dl.acm.org/citation.cfm?id=2982818.2982829

[3] S. Jeschke, T. Skřivan, M. Müller-Fischer, N. Chentanez, M. Macklin, and C. Wojtan, “Water surface wavelets,” *ACM Trans. Graph.*, vol. 37, no. 4, pp. 94:1–94:13, Jul. 2018. [Online]. Available: http://doi.acm.org/10.1145/3197517.3201336

[4] E. E. Catmull, “A subdivision algorithm for computer display of curved surfaces.” Ph.D. dissertation, 1974, aAI7504786.

[5] M. Bächer, B. Bickel, E. Whiting, and O. Sorkine-Hornung, “Spin-it: Optimizing moment of inertia for spinnable objects,” *Commun. ACM*, vol. 60, no. 8, pp. 92–99, Jul. 2017. [Online]. Available: http://doi.acm.org/10.1145/3068766

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

[7] C. Hall and W. Hoff, *Water Transport in Brick, Stone and Concrete.* CRC Press, 2009.

[8] A. Fick, “On liquid diffusion,” *Journal of Membrane Science*, vol. 100, no. 1, pp. 33–38, mar 1995.

[9] J. Crank, *The Mathematics of Diffusion.* Clarendon Press, 1956.

[10] J. Lekner and M. C. Dorf, “Why some things are darker when wet,” *Appl. Opt.*, vol. 27, no. 7, pp. 1278–1280, Apr 1988.

[11] A. Ångström, “The albedo of various surfaces of ground,” *Geografiska Annaler*, pp. 323–342, 1925.

[12] G’MIC Project, “http://gmic.eu/,” Accessed 12/12/2017.

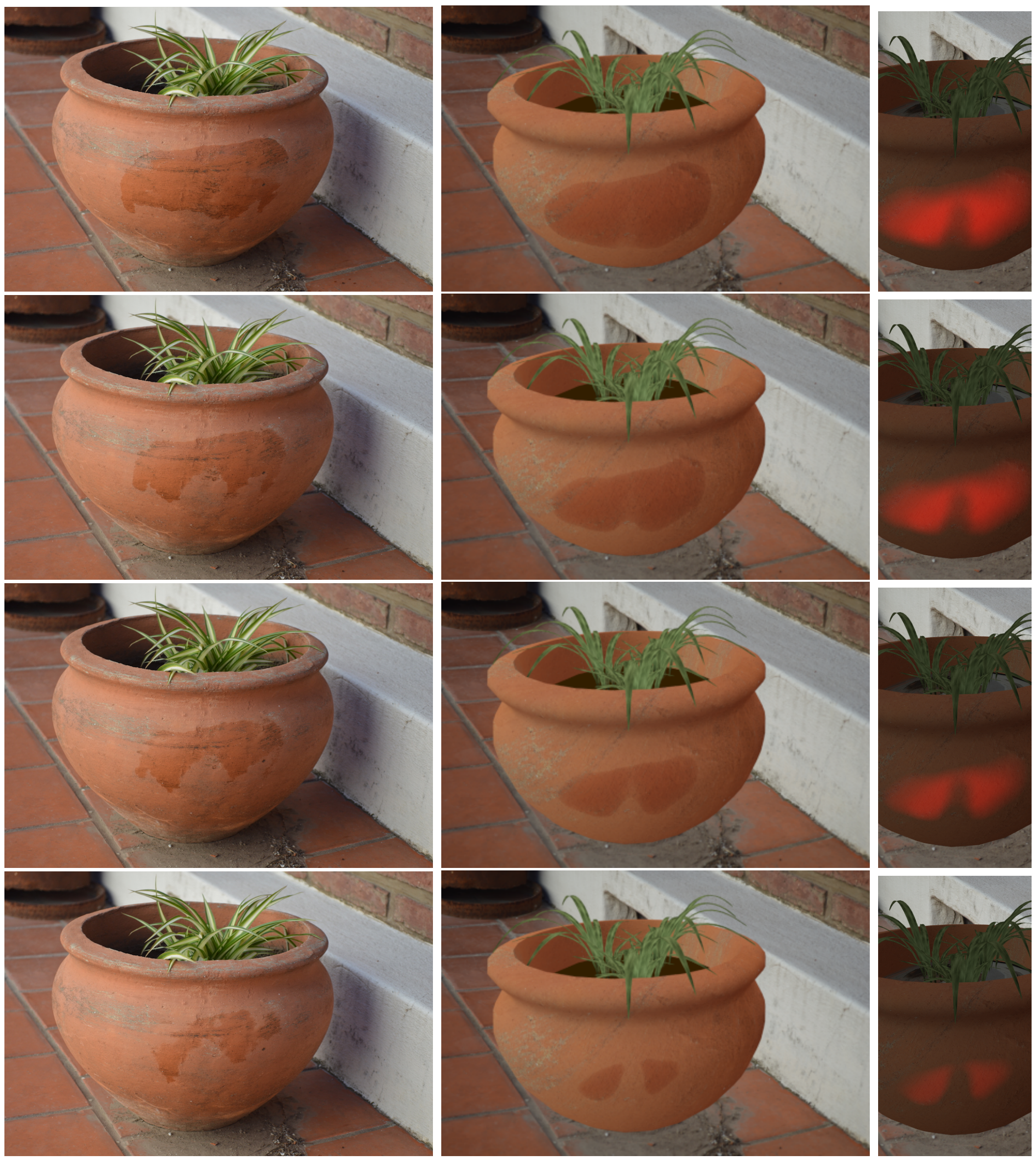

Figure 18: Images generated from the developed model compared with captures from the real world. The first column shows the actual drying process of a wet pot. The second column shows a sequence of animation frames performed with the presented simulation technique. The third column also shows the result of the simulation with pseudocolor to show the wet zone. In order to simulate the real conditions, the following conditions were taken into account: the texture of the pot model was synthesized from a sample of the surface texture of the real pot using a patch-based method implemented by the G'MIC project [12].