Interactive Fluid Animation Using Deformable Meshes and Shaders

Júlia Kubiak Melgaré
Pontifical Catholic University of Rio Grande do Sul, Polytechnic School
Porto Alegre, Brazil
julia.melgare@acad.pucrs.br

ABSTRACT

In digital games, well polished interactive 3D environments can make the user's experience a lot more fun and enjoyable. When it comes to large bodies of water, however, such level of interaction is rarely seen given the difficulty of such task to be performed in real-time. In this work, we implement and experiment upon a different abstraction for interactive large bodies of water in real-time 3D, using a deformable mesh that is able to respond to other physically simulated objects, such as rigid bodies. The results obtained with our experiments show that this approach is able to produce satisfactory results, while allowing more control of the water's behavior to the user.

Author Keywords

computer animation; games; real-time; water shader; mesh deformation; rigid bodies.

CCS Concepts

•**Computing methodologies** → **Animation; Mesh models;**

INTRODUCTION

When it comes to digital games, much of the fun of playing comes from the level of quality of their interaction with the game world. A well polished and interactive 3D environment in a game allows for many emergent behaviors and gameplay mechanics to develop organically. In that sense, rigid body physics solvers have revolutionized 3D environments because they provide so many natural interaction possibilities. Similar interactions with large bodies of water, however, are very difficult to achieve in the real-time 3D realm due to how costly fluid simulations can be.

Given this scenario, in this work we propose to implement an interactive fluid through the use of a deformable mesh, allowing it to respond to rigid bodies that collide with it accordingly. Indeed, this approach is not attempting to create or optimize a model for physically simulating fluids, but rather propose and explore a different abstraction of fluid interaction in real-time 3D environments. We believe that some applications for this approach would be its use in 3D real-time environments that allow for emergent interaction with bodies of liquid and that require a less computationally heavy approach where physical accuracy is not as important, e.g. digital games.

In this paper, we give a brief summary of other work that aim to solve a similar problem in Section 2. We describe our approach as well as some experiments to examine it's behavior in Section 3. In Section 4 we present the results obtained after implementing our method and executing the defined experiments, and finally, in Section 5, future aspects of this research are discussed.

ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

RELATED WORK

In the literature there are many methods to simplify and optimize already known water simulation models so that they can be used in real-time executions [3], while other authors can also propose their own solutions, presenting new and more optimized fluid simulation models [2]. Meanwhile, in the digital games industry, developers often find other solutions that do not necessarily involve real-time physically accurate models [1].

Regarding the use and optimization of already existing methods, Kellomäki [3] uses height maps in combination with a virtual pipe model originally created by Mei et al. [4] to simulate water and rigid body interaction with a physically accurate approach that is more optimized. The virtual pipe model is able to dynamically alter the water levels through the height map, which results in a simple but fast method for real-time water simulation to be applied in games. On the other hand, Heintz et al. [2] created their own method based on cellular automata, which divides the liquid in many cells, and has each agent be responsible for their cell and its characteristics. This allows them to generate fluid simulations in a much more organic manner, including the interactions between different kinds of liquids (e.g. oil and water).

As an example of water simulation techniques being applied in digital games, Gonzalez-Ochoa et al. [1] describe how such process was conceived in the 2012 game *Uncharted*, where different types of in-game water interaction were implemented. The techniques used include vertex shaders for rivers, geometric clipmaps for stormy oceans and offline simulations

combined with particle systems for a scene which included a flooding area.

Given these presented related work, we aim to implement an interactive and animated body of water that does not rely on physical accuracy, which in turn would give more artistic freedom to artists and animators to manipulate it accordingly. Our method will also use simpler calculations which makes it usable in 3D real-time environments such as digital games.

METHOD

This section describes the proposed methodology, which is organized in three subsections: *i*) first we explain the mesh deformation process and how it responds to rigid bodies, *ii*) after that we also describe how the shader used to render the mesh was implemented, including the vertex displacement and other visual aspects, and lasty, *iii*) we define the experiments that were done in order to analyse our model's behavior.

Mesh Deformation

The body of water animated with our method is based on a grid plane with a deformable mesh. This approach was chosen because the mesh deformation allows the water to respond to the impact and weight of the rigid bodies that float on top of it, as well as other external forces.

The deformation is achieved by applying a force to the target mesh based on the point of impact. This force is distributed to all the vertices of the mesh proportionally, according to how close each vertex is to the point of impact. Equation 1 shows the calculation for this process, where the attenuated force F_v is calculated by dividing the applied force F by the square distance of the vertex from the point of impact plus 1. We increment the square distance by 1 so that the force is at full strength when the distance is zero instead of growing towards infinity.

$$F_v = \frac{F}{1+d^2} \tag{1}$$

After obtaining the attenuated force, we turn it into a velocity for each vertex. We obtain the acceleration a using $a = \frac{F_v}{m}$, considering $m = 1$ for all vertices. Using $\Delta v = a\Delta t$, we obtain $\Delta v = F_v \Delta t$. This velocity is used to have the vertices move in a direction given by the normal of the vector between each vertex and the point of impact. This causes the object to deform when a force is applied to it, but in order for it to maintain its shape, we add a constraint where the vertices always try to move back to their original positions. This constraint works similarly to a spring system, and the speed with which the vertices return to their original place is given by a spring force parameter F_s. With these two velocities being applied to the mesh, it would constantly oscillate and never stop deforming, which is why we add a dampening effect on the velocities of the vertices so that they are decreased over time until their values reach 0.

With each alteration of the mesh due to its deformation, we also update the object's mesh collider accordingly. This allows for a smooth and organic interaction with other physically simulated objects, such as rigid bodies. Thus, whenever a rigid body collides with our deformable body of water, we start to continuously apply the colliding object's weight force (given by Equation 2, where m_o is the rigid body's mass and g is the simulated gravity) to the deformable mesh, using the point of collision as our point of impact. Figure 1 shows two examples of our mesh being deformed by different rigid bodies.

$$F_w = m_o g \tag{2}$$

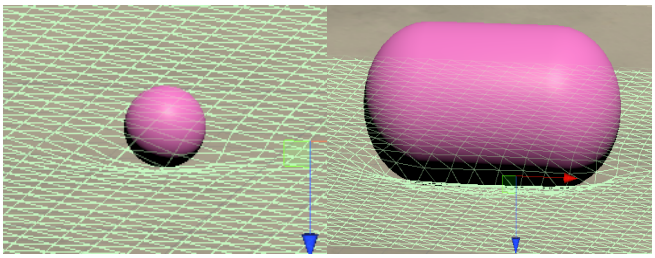


Figure 1. Examples of our mesh being deformed by two different rigid bodies.

Water Shader

We also implemented a shader which included vertex displacement for the creation of waves in our body of water, as well as color variation based on depth, foam lines and ripple effects using particle systems. Figure 2 shows an example of these effects mentioned, along with two objects interacting with the body of water.

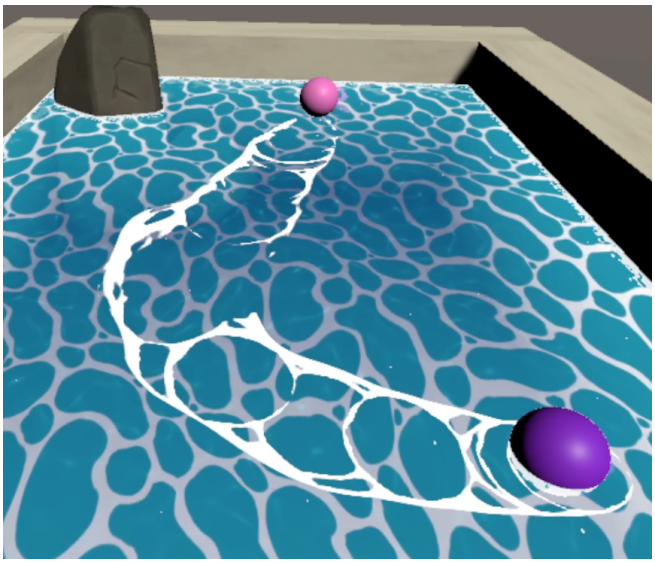


Figure 2. Two objects interacting with our mesh, which is rendered by our shader.

For the vertex displacement, we simply apply a sinusoidal function to the y-axis of the mesh's vertices. Because we use a mesh collider, these transformations also affect the rigid bodies floating atop of the water, causing them to be affected by the flow of the waves. For the color of the water, we

use a depth-based function which applies lighter colors to shallow areas and darker colors to deep areas. The foam lines around the objects who are partially submerged in the water are implemented with a similar depth-based approach, rendering a white gradient noise texture on the edges of the shallowest parts of the mesh. Lastly, for the ripple effect, we use a particle system which follows its parent object, being rendered in a separate layer and added to the rest of the shader's color.

Experiments

We implement our deformable mesh and its shader using the Unity 3D engine [1] and C#. After creating a 3D test environment, we define some experiments to examine our method's behavior.

We observe two main aspects: i) how the mesh behaves when different rigid bodies of varying mass and size interact with it, and ii) how the mesh's behavior is affected when we vary the spring force F_s. With that in mind, we propose six separate experiments by using spring force values {20, 40, 10} each with a small and a large rigid body object that have mass m_o of 1 and 2, respectively. Spring force value of 20 was chosen because it is the approximate limit where the force F_s would be higher than the large rigid body's force F_w. We then decide on 40 and 10 as they are twice as strong and half as strong as 20, respectively.

Our implemented 3D environment is composed of our deformable mesh, either a small or large-sized rigid body and also a rigid body which is controllable via keyboard interface, allowing the user/player to push around other objects floating in the water via collision.

RESULTS

In this section we present and discuss the results obtained when executing the previously defined experiments, where we aim to analyse the body of water's behavior when different sized objects are interacting with it, as well as the effects of altering the deformable mesh's spring force.

Our first experiment was to use spring force $F_s = 20$. Figure 3 shows the results obtained when executing the 3D environment with both small and large rigid bodies, and pushing them with our controllable object (the purple sphere). The deformation obtained in this scenario can more easily be seen on Figure 3(b), where the large rigid body is able to deform the water as it is moving. In comparison, Figure 4 shows the results obtained when the spring force was doubled, using a value of 40. In both sub-figures it is possible to see that the deformation happens in a more subtle manner, since a higher spring force means that the mesh will return to its original shape in a faster speed. Meanwhile, Figure 5 shows the results obtained when using spring force $F_s = 10$. As can be seen in Figure 5(b), we find that our deformable mesh starts to invert itself when a force greater than it's spring force is continuously applied to it, which causes an aspect of an agitated or stormy sea. However, this behavior could also be a limitation of our method, as we could consider that if an object's weight force F_w is greater than the spring force F_s, then it should sink underwater.

Regarding the obtained results with our experiments, we note that altering the spring force of the deformable mesh affects the water's rigidness, with it being higher as the spring force's values increases. We believe that the most satisfactory results were obtained using spring force $F_s = 20$, as it achieved the most natural looking results, since it is the value where the water can be most malleable without breaking the mesh deformation. Meanwhile, as evidenced by the experiment using $F_s = 10$, the model does not function properly when the forces continuously applied to the mesh are greater than its spring force.

FINAL CONSIDERATIONS

This paper presents an experimental study of a different abstraction method for representing large bodies of water in 3D real-time environments. Although our method does not attempt to present a physically accurate solution, we were still able to implement a body of water which is able to correctly respond to the interaction with other physically simulated objects such as rigid bodies.

For future work, the authors would like to refine the model to fix some of its limitations, such as the unexpected behavior that occurs when an interacting object's weight force is greater than the mesh's spring force. In this case, we could define that the object should be submerged in the water instead of remaining afloat. We could also allow for further interaction with the rigid bodies that are touching the water, such as applying external forces to it. An example of such feature would be the possibility of pushing down a light-weighted rigid body in order to submerge it underwater.

REFERENCES

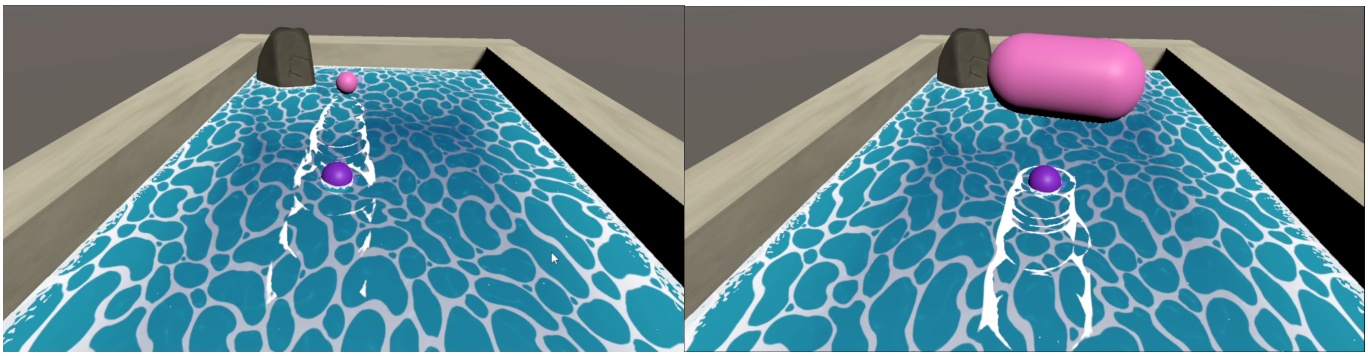
[1] Carlos Gonzalez-Ochoa, Doug Holder, and Eben Cook. 2012. From a Calm Puddle to a Stormy Ocean: Rendering Water in *Uncharted*. In *ACM SIGGRAPH 2012 Talks (SIGGRAPH '12)*. Association for Computing Machinery, New York, NY, USA, Article 3, 1 pages. DOI:http://dx.doi.org/10.1145/2343045.2343050

[2] Christian Heintz, Moritz Grunwald, Sarah Edenhofer, Jörg Hähner, and Sebastian von Mammen. 2017. The Game of Flow - Cellular Automaton-Based Fluid Simulation for Realtime Interaction. In *Proceedings of the 23rd ACM Symposium on Virtual Reality Software and Technology (VRST '17)*. Association for Computing Machinery, New York, NY, USA, Article 76, 2 pages. DOI:http://dx.doi.org/10.1145/3139131.3143415

[3] Timo Kellomäki. 2017. Fast Water Simulation Methods for Games. *Comput. Entertain.* 16, 1, Article 2 (Dec. 2017), 14 pages. DOI: http://dx.doi.org/10.1145/2700533

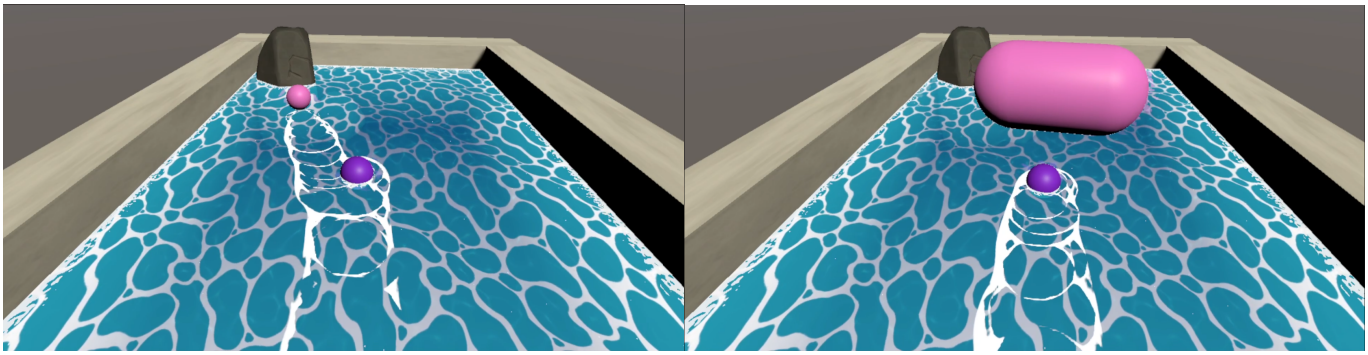
[4] Xing Mei, Philippe Decaudin, and Bao-Gang Hu. 2007. Fast Hydraulic Erosion Simulation and Visualization on GPU. In *15th Pacific Conference on Computer Graphics and Applications (PG'07)*. 47–56. DOI: http://dx.doi.org/10.1109/PG.2007.15

[1] https://unity.com/



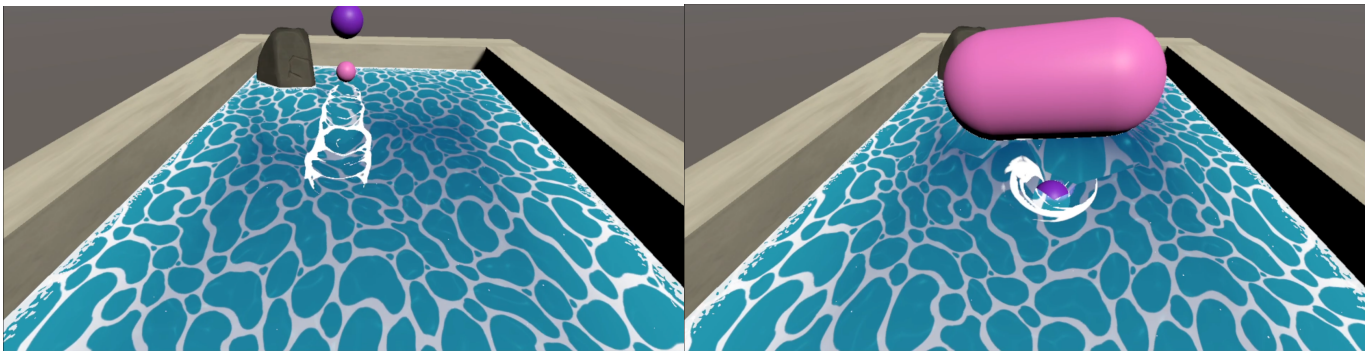
(a) Small rigid body (pink) being pushed by the player (purple) (b) Large rigid body (pink) being pushed by the player (purple)

Figure 3. Results obtained with both small and large rigid bodies using spring force $F_s = 20$.



(a) Small rigid body (pink) being pushed by the player (purple) (b) Large rigid body (pink) being pushed by the player (purple)

Figure 4. Results obtained with both small and large rigid bodies using spring force $F_s = 40$.



(a) Small rigid body (pink) being pushed by the player (purple) (b) Large rigid body (pink) being pushed by the player (purple)

Figure 5. Results obtained with both small and large rigid bodies using spring force $F_s = 10$.