

Robust and Efficient Wave Simulations on Deforming Meshes

Roland Angst¹, Nils Thuerey¹, Mario Botsch^{1,2}, Markus Gross¹

¹ Computer Graphics Laboratory, ETH Zurich; ² Computer Graphics Group, Bielefeld University

Abstract

The goal of this paper is to enable the interactive simulation of phenomena such as animated fluid characters. While full 3D fluid solvers achieve this with control algorithms, these 3D simulations are usually too costly for real-time environments. In order to achieve our goal, we reduce the problem from a three- to a two-dimensional one, and make use of the shallow water equations to simulate surface waves that can be solved very efficiently. In addition to a low runtime cost, stability is likewise crucial for interactive applications. Hence, we make use of an implicit time integration scheme to obtain a robust solver. To ensure a low energy dissipation, we apply an Implicit Newmark time integration scheme. We propose a general formulation of the underlying equations that is tailored towards the use with an Implicit Newmark integrator. Furthermore, we gain efficiency by making use of a direct solver. Due to the generality of our formulation, the fluid simulation can be coupled interactively with arbitrary external forces, such as forces caused by inertia or collisions. We will discuss the properties of our algorithm, and demonstrate its robustness with simulations on strongly deforming meshes.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism, Animation; I.6.8 [Types of Simulation]: Gaming

1. Introduction

Physically based simulations have become a crucial aspect in real-time applications, and represent a central gameplay element in a variety of modern games. While rigid bodies can be handled in large numbers, and deforming bodies or cloth are increasingly used in recent games, fluid effects are still rarely encountered. This is caused by the high computational cost of solving the underlying equations, which prevents practical applications in real-time scenarios. In particular, interesting effects can be achieved by controlling fluid sim-

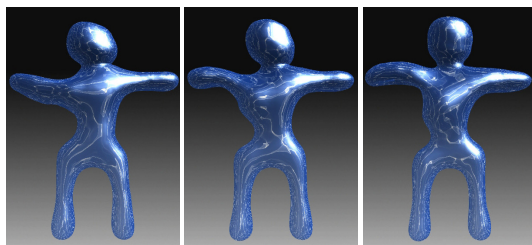


Figure 1: Waves on a simple animated character that was simulated with our approach.

ulations, e.g., to form characters. These effects, however, require even more computations, since, in addition to the fluid solver, control forces have to be computed and applied. Thus, we reduce the problem from 3D to 2D, and use the shallow water equations (SWE) to capture the effects of waves traveling on the surface of a character. As we are targeting real-time environments, the stability of the simulation is highly important. We make use of an implicit time integration scheme to ensure unconditional stability. The implicit integrators used in previous work, however, suffer from a high numerical dissipation, which causes the energy of the system to be quickly lost. For fluids, this usually means that the material looks more viscous than it should. We overcome this problem by using an Implicit Newmark scheme. Finally, we propose to use a Cholesky decomposition to solve the resulting system of equations. This guarantees a low run-time of the solver throughout the course of the simulation. The contributions of this paper are a careful linearization together with an mixed finite volume/finite element discretization of the SWE, which are tailored to a temporal integration with the Implicit Newmark scheme. This lays the foundation for energy-preserving wave simulations on deforming triangle meshes with arbitrary user controllable forces at interactive frame rates. We will demonstrate inertia and rigid body coupling as examples for the external forces. An example simulation on a moving character mesh is shown in Fig. 1.

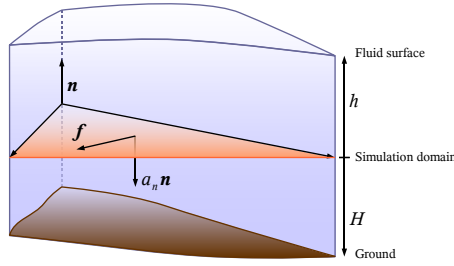


Figure 2: A fluid volume element is represented as a height-field elevation in normal direction \mathbf{n} . The ground plane is given at a depth of H below the simulation domain. The total fluid depth η equals the sum of the dynamic displacement h and the (in our case constant) ground depth H . Forces are decomposed into a tangential force component \mathbf{f} and a vertical acceleration component $a_n \mathbf{n}$.

2. Related Work

As we are surrounded by fluids and liquids in our everyday lives, these phenomena have become important for all applications recreating reality – from large scale movie productions to computer games. The use of fluid simulations in computer graphics was first demonstrated in [KM90]. While extensions of this approach, such as a coupling with particles [OH95], and rigid body coupling [CdVL95] were proposed, full three-dimensional simulations have become especially popular in the past years.

These simulations are most often based on the techniques described in [Sta99] to solve of the underlying Navier-Stokes equations. For liquids, the solver is often combined with appropriate boundary conditions and a level set based surface tracking, as demonstrated in [FF01]. In addition, full 3D simulations were lately combined with height-field techniques to reduce the computational costs for large water surfaces [IGLF06, TRS06]. Although recently a full three-dimensional Navier-Stokes based free surface solver was demonstrated by [CLT07], these computations are still too costly for practical applications in interactive applications. On the other hand, Navier-Stokes solvers using a Lagrangian representation, such as smoothed particle hydrodynamics [MCG03], do not restrict the fluid to small spatial regions, but are likewise costly to compute. In our goal to create fluid characters, our approach is similar to previous work such as [TMPS03], [SY05], and [TKPR06]. We, however, directly use the animated mesh as the domain for our simulation, instead of computing control forces from it.

Height-field based fluid simulations, which we will also use in the following, have been applied to a variety of problems in engineering applications, as well as in the computer graphics fields. In [KM90], Kass and Miller derived the ordinary wave equation by simplifying the SWE. However, they used the wave equation without terms for external forces, which are required for, e.g., inertial effects. Layton and van der Panne demonstrated a stable shallow water solver in combination with a semi-Lagrangian advection step in [LvdP02]. While two-dimensional Navier-Stokes (NS) solvers were demonstrated on Catmull-Clark surfaces [Sta03], and on static triangle meshes [SY04], we will solve the SWE instead of the NS equations on arbitrarily deforming triangle meshes. Similar to our work, Wang et. al [WMT07] solve the SWE on the surfaces of objects. They, however, make use of an Implicit Euler method and target surface

tension driven effects. They do not simulate surface waves spreading on the meshes, and thus energy preserving wave movements are not as crucial as for the goals of this paper. In contrast to [WMT07], we include the external forces in our implicit time integration scheme to achieve a robust solver. In addition, our approach allows us to directly discretize the equations on the triangle mesh. Hence, unlike [WMT07], we do not compute an additional well conditioned triangulation, which would be especially difficult for interactively deforming meshes. In conclusion, we do not impose any restrictions on the deformations of the animated mesh.

3. Shallow Water Equations

The SWE are a simplified form of the equations describing the motion of a fluid, the Navier-Stokes equations. In the following we will describe the SWE and their linearization for our problem.

Differential Formulation: The shallow water equations can be derived from the inviscid Navier-Stokes equations by assuming a negligible vertical velocity component. This in turn implies a hydrostatic pressure distribution and, furthermore, velocities that can be assumed as averaged in the vertical direction. The result is a two-dimensional tangential velocity field representation with the fluid surface given by elevations orthogonal to the tangential velocity field. In the following, H will denote the height of the ground beneath the tangential plane (which is constant in our case), h is the dynamic height displacement with respect to this plane, and hence, $\eta = H + h$ is the total height of the fluid, as shown in Fig. 2.

The derivation of the SWE can be generalized by additionally considering density force terms \mathbf{f} , which are tangential to the simulation domain, and an acceleration in vertical direction a_n . With these terms, and the fluid density ρ , the SWE take the form:

$$\eta_t + \mathbf{v} \cdot \nabla \eta = -\eta \nabla \cdot \mathbf{v} \quad (1)$$

$$\mathbf{v}_t + \mathbf{v} \cdot \nabla \mathbf{v} = a_n \nabla h + \frac{1}{2} \eta \nabla a_n + \frac{1}{\rho} \mathbf{f}, \quad (2)$$

where a t subscript denotes the temporal derivative. Usually, the SWE are derived by assuming a constant normal acceleration a_n equal to the gravitational acceleration $g = -9.81 \frac{m}{s^2}$. However, the formulation given above properly handles spatially varying normal accelerations as well. These accelerations will be used later on to include inertial effects and rigid body collisions. Note that, in contrast to, e.g., [WMT07], the direction of the constant gravitational acceleration g is chosen to always point in the same direction w.r.t. the local reference system, and thus g is opposing the direction of the surface normal.

Linearization: The SWE from (1) and (2) contain several nonlinearities. In the following, we will describe how to linearize the equations so that we can make use of the Implicit Newmark integrator. We have noted that the effect of velocities caused by surface waves is very small compared to the wave movement itself. Thus, for simulations without a global flow, which we are targeting, we can neglect the advection terms on the left hand sides of (1) and (2). We have performed tests with a solver including these terms, and found that the results were hard to distinguish from a version without the advective terms. This simplification reduces the overall amount of computations, and simplifies the system to be solved for the Implicit Newmark step.

By considering the analytical expression for the vertical acceleration of the fluid depth η_t , which is given by taking the temporal

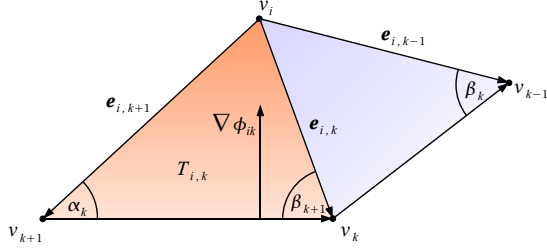


Figure 3: The discrete differential operators can be formulated as a weighted sum with cotangent weights. The gradient of the linear basis function associated with vertex v_i in an adjacent triangle $T_{i,k}$ is orthogonal to the edge in this triangle opposite to the vertex v_i . The outgoing edges of a vertex v_i associated with a triangle $T_{i,k}$ are denoted as $\mathbf{e}_{i,k}$.

derivative of the necessary terms (1), we get the second order differential equation

$$\eta_{tt} = -\frac{\partial}{\partial t} (\eta \nabla \cdot \mathbf{v}) = -\eta_t \nabla \cdot \mathbf{v} - \eta \nabla \cdot \mathbf{v}_t \quad (3)$$

$$= -\eta_t \nabla \cdot \mathbf{v} - \eta \nabla \cdot \left(a_n \nabla h + \frac{1}{2} \eta \nabla a_n + \frac{1}{\rho} \mathbf{f} \right), \quad (4)$$

where the tangential acceleration \mathbf{v}_t was replaced with the remaining terms of (2). This second order differential equation requires a proper linearization to be integrable with a non-iterative implicit scheme. As in [KM90], [LvdP02], and [WMT07] we assume the fluid depth to be approximately constant within the time interval $[t^{(n)}, t^{(n+1)}]$ and, hence, the first term in (4) vanishes.

The factor η in front of the divergence operator, however, requires us to depart from previous work, where this factor was assumed to be temporally constant in the interval $[t^{(n)}, t^{(n+1)}]$, i.e., the factor was simply fixed to the value $\eta^{(n)}$ for the computation of $\eta_{tt}^{(n+1)}$. Our experiments show that in order to obtain an unconditional stable integration scheme, we have to assume that this factor is temporally and spatially constant, i.e., this factor is set to a constant η_0 in the remaining non-linearities. Note that this gives us an acceleration term similar to the wave equation. For examples such as those presented in the following, with an initial water height η_0 that is an order of magnitude larger than the wave amplitudes, we have found that this is a good approximation. Thus, the second derivative of the fluid height is given by

$$\eta_{tt} = -\eta_0 \left(a_n \nabla^2 h + \frac{1}{2} \eta \nabla^2 a_n + \frac{3}{2} \nabla a_n \cdot \nabla h \right) - \eta \left(\frac{1}{2} \nabla a_n \cdot \nabla h + \nabla \cdot \frac{\mathbf{f}}{\rho} \right), \quad (5)$$

which is linear in the unknown fluid depth η . Note that in contrast to (4) this formulation no longer requires us to carry along the tangential velocities \mathbf{v} .

4. Discretization

In the following we will explain our discretization of the shallow water equations and present an unconditionally stable implicit integration scheme.

Spatial Discretization: The two-dimensional manifold surface, on which the simulation will be performed, is discretized as a triangle mesh with a fixed topology. The neighborhood of a vertex v is denoted as $\mathcal{N}(v)$ and depending on the context, this set consists either of the adjacent vertices v_k or adjacent triangles $T_{i,k}$. According to [DKT06], the triangle mesh induces a dual mesh whose dual vertices are given by primal triangle centers. The more robust barycentric scheme is used to define these dual vertices since the circumcenter is not guaranteed to lie inside a badly shaped triangle. Such triangles can easily emerge during a deformation of the simulation domain.

Similar to [TLHD03] and [PP02], we follow a mixed finite volume/finite element approach that is popular in the field of meteorology and oceanography. Discrete *scalar* fields are represented as piecewise linear functions $f(\mathbf{x}) = \sum_{v_i \in \mathcal{V}} f_i \phi_i(\mathbf{x})$, which corresponds to a finite element approach with linear basis functions ϕ_i . Here, \mathcal{V} denotes the set of all vertices of the mesh. The values f_i of the function are located at the vertices $v_i \in \mathcal{V}$. Discrete *vector* fields on the other hand are defined as piecewise constant functions with sampling nodes at the triangle faces. This formulation leads to the well-known discrete differential operators for the gradient ∇_{T_k} inside a triangle T_k , the divergence $\nabla_{v_i} \cdot \mathbf{v}$ at a vertex v_i , and the Laplacian $\nabla_{v_i}^2$ at a vertex v_i . As derived in [PP93] and [MDSB02] the operators are given by:

$$\nabla_{T_k} f = \sum_{v_i \in T_k} f_i \nabla \phi_{i,k}, \quad (6)$$

$$\nabla_{v_i} \cdot \mathbf{v} = - \sum_{T_k \in \mathcal{N}(v_i)} \nabla \phi_{i,k} \cdot \mathbf{v}_k |T_k|, \quad (7)$$

$$\nabla_{v_i}^2 f = \frac{1}{2A_i} \sum_{v_k \in \mathcal{N}(v_i)} (\cot \alpha_k + \cot \beta_k) (f_k - f_i). \quad (8)$$

Here, the gradient $\nabla \phi_{i,k}$ of the linear basis function ϕ_i at vertex v_i in a neighboring triangle $T_{i,k}$ is calculated with

$$\nabla \phi_{i,k} = -\frac{1}{2|T_{i,k}|} (\cot \alpha_k \mathbf{e}_{i,k} + \cot \beta_{k+1} \mathbf{e}_{i,k+1}). \quad (9)$$

In these formulas, the area of a triangle T_k is denoted as $|T_k|$ and the barycentric dual cell area associated with a vertex v_i is given by A_i . Fig. 3 shows a graphical depiction of the terms in the above equations that have not yet been defined here.

Temporal Discretization: Although explicit time integration methods can handle the nonlinearities in (1) and (2), such an explicit scheme will be subject to the *Courant-Friedrichs-Lewy* (CFL) condition. In our case, the CFL condition depends on the fluid depth as well as on the shape and size of the triangles of the simulation domain. Since our approach should work without enforcing any restrictions on the animation of the control mesh, the CFL condition might impose a very small and impractical time step in order to avoid instabilities. As interactive applications have a limited time for computations during each displayed frame, we can't arbitrarily decrease the time step size, and thus unconditional stability is crucial. While Implicit Euler methods have been used to achieve stability, these approaches quickly dissipate energy.

To overcome the problem of energy dissipation while ensuring a robust scheme, we propose to use an Implicit Newmark time integration scheme [New59]. It can be formulated as

$$\begin{aligned} \eta^{(n+1)} &= \eta^{(n)} + \Delta t \eta_t^{(n)} + \frac{1}{2} \Delta t^2 \left((1-\beta) \eta_{tt}^{(n)} + \beta \eta_{tt}^{(n+1)} \right) \\ \eta_t^{(n+1)} &= \eta_t^{(n)} + \Delta t \left((1-\gamma) \eta_{tt}^{(n)} + \gamma \eta_{tt}^{(n+1)} \right). \end{aligned} \quad (10)$$

The choice $\beta = \gamma = \frac{1}{2}$ results in an unconditionally stable method. An update step with this scheme requires a vertical velocity η_t and an acceleration term η_{tt} for the fluid elevation. Note that, although the SWE provide an analytical expression for the vertical velocity η_t with (2), the vertical velocity needs to be integrated according to the Newmark integration scheme (10) in order to preserve its unconditional stability.

Inserting the linearized vertical acceleration (5) in (10) for $\eta_{tt}^{(n+1)}$ results in a linear system in the unknown fluid depth $\eta^{(n+1)}$ at the next point in time. The sparse matrix of this implicit system is rendered into a symmetric matrix by dividing the equation corresponding to vertex v_i by a weight factor $\eta_0|a_n|/(2A_i)$ such that the off-diagonal elements of the matrix $\eta_0 a_n \nabla^2 h$ correspond to the cotangent weights of the discretized Laplacian operator (8), which are symmetric. For details, please refer to [BBK05]. The term involving $\nabla a_n \cdot \nabla h$ is symmetrized in a slightly different way, namely the summand involving h_{v_i} and a_{n,v_j} , where v_i and v_j are two adjacent vertices, is rather divided by $\eta_0|a_{n,v_i} + a_{n,v_j}|/(4A_i)$. This averaging of neighboring normal accelerations introduces some spatial smoothing of these normal accelerations. However, as external forces usually vary slowly, the effects of this symmetrization are barely noticeable. These symmetrization operations allow to use efficient solvers for sparse, symmetric systems of linear equations. In the following, we will use Cholesky Decomposition, as also proposed in [BBK05]. The decomposition can be solved efficiently, as the sparsity structure of the matrix does not change (only the cotangent weights are updated in each time step). Thus, we store a symbolic pre-factorization to speed up the Cholesky Decomposition for each time step. Here, we use the solver provided by the *CHOLMOD* package [DH05]. Note that we have found this decomposition to be advantageous over solving the problem with an iterative method, such as a CG solver. An iterative solver would require a large number of iterations for a peak in the applied forces, while our direct solvers has a runtime independent of the state of the simulation. Furthermore, our tests indicate that the speed of the *CHOLMOD* solver is superior even when only few iterations of a CG solver would be required.

The symplectic nature of the Newmark integration scheme implies conservation of energy in our case. This is a great advantage compared to the commonly used Implicit Euler integration method which suffers from severe numerical damping which renders the Implicit Euler method unusable to simulate waves on triangle meshes. A manually controllable damping of the Newmark scheme is easily introduced by slightly decreasing displacements from the equilibrium position in each time step (note that a more accurate way would be to include *Rayleigh damping* in the Newmark solver).

Since the continuity equation (1) is actually nonlinear, a linear implicit system, as used in any of the implicit approaches, such as ours and [WMT07, LvdP02], can not ensure the conservation of mass. Therefore, a simple and commonly applied correction is to uniformly scale the fluid heights with the ratio between the true mass and the mass after solving the implicit system. Note however, that this mass correction step slightly draws energy from the system, and hence the energy is no longer fully preserved. On the other hand, the damping effect is very subtle and usually only noticeably affects the simulation after a large number of time steps.

5. Results

The test cases that will be presented in this section were computed on an off-the-shelf PC with an Intel Core2 Dual CPU with 2.4 GHz,

Test case	Vertices	Triangles	FPS
Skinny Guy (cf. video)	3778	7552	32.8
Stickman (Fig. 1)	4482	8960	25.4
Hamster (Fig. 4)	8502	16800	14.4
Face (Fig. 5)	9350	19056	9.4

Table 2: Performance of our implementation for the different test cases shown in the paper and video. The timing measurements are given in frames per second (FPS) and include the simulation as well as rendering. The rendering takes ca. 10% of the time.

and a modern GPU. As our algorithm runs on the CPU, using only one of the cores, the GPU was not a bottleneck in any of the test cases presented below. In Table 2, frame rate measurements for the test cases are given together with the sizes of the simulation domain meshes. In this section, we will compare our approach to the popular Implicit Euler integrator, and show examples of simulations with external forces from inertia and rigid body collisions.

Comparison to Implicit Euler: The Implicit Euler integration is commonly used to solve the SWE, e.g., in [KM90, WMT07], and is formulated as

$$\eta^{(n+1)} - \Delta t^2 \eta_{tt}^{(n+1)} = \eta^{(n)} + (1 - \tau)(\eta^{(n)} - \eta^{(n-1)}), \quad (11)$$

where the parameter τ gives control over a manual damping in addition to the numerical damping of the implicit integration itself. As our experiments indicate that the Implicit Euler step has a high numerical damping itself, we have set $\tau = 0$ for the comparison.

Table 1 shows a comparison between these integration schemes. The test case is a smooth initial elevation on a spherical mesh with ca. 5000 triangles. The elevation results in a wave that travels around the sphere, and converges on the opposite side again. Both the Implicit Euler and Newmark integrators are unconditionally stable, but the Implicit Euler step introduces a significant amount of numerical damping, which prevents the wave from properly being visible. The Implicit Newmark integrator, on the other hand, shows very little numerical damping, so that the wave motion continues on the sphere.

Animated Meshes: The algorithm described in Section 4 allows us to use any animated mesh with fixed topology as the simulation domain. An important effect that we can easily include in our formulation is that of inertia. The acceleration at a vertex v of the simulation mesh is approximated using a backward finite difference scheme

$$\mathbf{a}^{(n)}(v) = \frac{\mathbf{x}^{(n)}(v) - 2\mathbf{x}^{(n-1)}(v) + \mathbf{x}^{(n-2)}(v)}{\Delta t^2}. \quad (12)$$

The inertia of the fluid acts against the direction of acceleration, and thus the negative of (12) is used in the following. The acceleration is split into a tangential component \mathbf{f} and a normal component a_n , which are added to (2).

Fig. 4 shows an example of a wave simulation on the surface of a complex animated mesh. The mesh has almost 17000 triangles, and the simulation runs with more than 14 frames per second. Note the strongly deforming triangles that arise during the animation, e.g., near the shoulders and below the mouth of the character.

Coupling with Rigid Bodies: The inclusion of forces from rigid

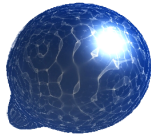
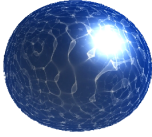
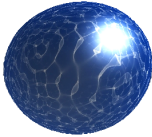
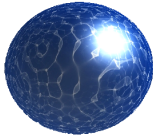
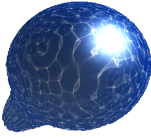
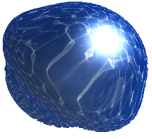
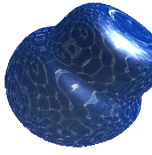
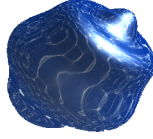
	t=0.025	t=0.6	t=1.0	t=7.0
Implicit Euler				
Implicit Newmark				

Table 1: Comparison of the energy preservation for Implicit Euler and Newmark integrators. As an initial condition, a displacement is added to the spherical simulation domain. The right column of pictures shows the simulation at a later time. While the simulation with Implicit Euler has lost all its energy, the simulation with the Newmark scheme retains its wave motion.

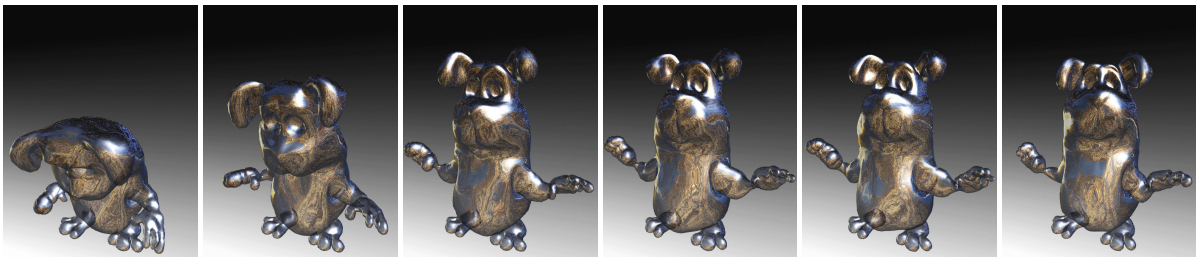


Figure 4: Waves simulated on an animated character with almost 17000 triangles, running with ca. 14 frames per second. The wave propagation can be seen best in flat areas, e.g., near the belly of the character.

bodies is also straight-forward. When collisions of a rigid body with the fluid surface are detected, we add forces according to the motion of the rigid body to the fluid simulation. These are mapped to the tangential and normal force components, and result in waves according to the shape of the colliding object. For a two-way coupling, forces proportional to the immersion depth could be integrated over the rigid body. A simulation with forces generated by rigid bodies is shown in Fig. 5. In this setup, a user can interactively throw balls at the simulated mesh. The collisions result in circular waves spreading on the fluid surface.

5.1. Limitations

The obvious limitations of our approach are those imposed by the height-field representation. We can naturally not handle effects such as splashes within our framework, and we do not prevent self-intersections from occurring. For simplicity we restricted the simulation domains to closed 2-manifolds which implies that we do not have to deal with boundary conditions. However, the formulation trivially extends to non-closed 2-manifolds. Moreover, changing mesh topologies could be handled with a simple interpolation scheme. An inaccuracy introduced by the linearization of the SWE is that a linear increase in fluid depth is a valid equilibrium state, if there are no external forces except the constant gravitational force, i.e., $\nabla a_n = 0$. In this case (5) only depends on the second order derivative of the

fluid depth. Hence, the acceleration in normal direction vanishes in this case. Fortunately, this situation rarely arises, and was not problematic for the test cases shown.

6. Conclusions

We have presented a framework that allows for robust and efficient wave simulations on arbitrarily deforming meshes. Our method is based on a finite element/finite volume discretization, and is linearized in a way that allows us to use an Implicit Newmark scheme for time integration. With our general formulation, we can easily include external forces, e.g., from inertia or rigid body collision, in our simulation. We have shown several examples of wave simulations with external forces on complex deforming characters, that run with high frame-rates. As a topic of future research, we plan to parallelize our method to run on multi-core CPUs or GPUs. As our current implementation is not fully optimized, and most parts of the algorithm can be executed in parallel, significant speed-ups should be possible on current GPU architectures. It would also be interesting to experiment with modeling other phenomena of deforming characters with our method, such as jiggling layers of skin. Another interesting aspect is the inclusion of a free surface treatment in the shallow water simulation. This will enable the simulation of interesting effects, like filling processes of fluid characters. Moreover, our method could be coupled to three-dimensional fluid simulations

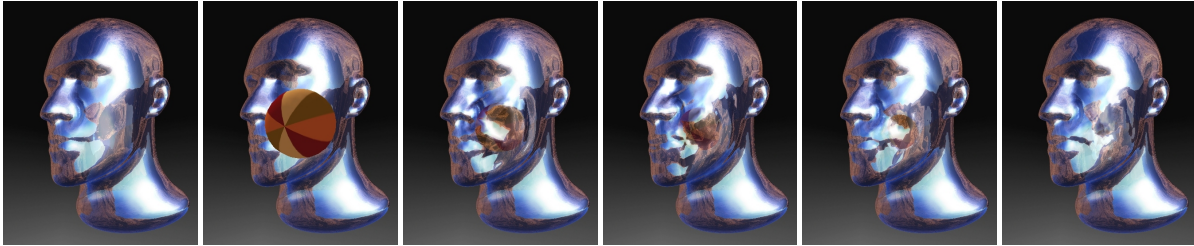


Figure 5: A complex mesh with more than 19000 triangles is hit by a rigid object. The waves resulting from the collision forces correctly spread over the curved surface.

such as smoothed particle hydrodynamics. This would make it possible to, e.g., treat splashes on the surface in a realistic way, while our simulation approach could efficiently handle the main fluid volume.

7. Acknowledgements

We would like to thank AGEIA for funding this project, and Christoph Baumann and Peter Hess for their work and helpful discussions on the topic. We also thank the anonymous reviewers for their useful comments.

References

- [BBK05] BOTSCH M., BOMMES D., KOBBELT L.: Efficient linear system solvers for mesh processing. In *IMA Conference on the Mathematics of Surfaces* (2005), pp. 62–83.
- [CdVL95] CHEN J. X., DA VITORIA LOBO N.: Toward interactive-rate simulation of fluids with moving obstacles using Navier-Stokes equations. *Graph. Models Image Process.* 57, 2 (1995), 107–116.
- [CLT07] CRANE K., LLAMAS I., TARIQ S.: Real-time simulation and rendering of 3D fluids. In *GPU Gems 3* (2007), Addison-Wesley Professional.
- [DH05] DAVIS T. A., HAGER W. W.: Row modifications of a sparse cholesky factorization. *SIAM Journal on Matrix Analysis and Applications* 26 (3) (2005), 621–639.
- [DKT06] DESBRUN M., KANSO E., TONG Y.: Discrete differential forms for computational modeling. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Courses* (New York, NY, USA, 2006), ACM Press.
- [FF01] FOSTER N., FEDKIW R.: Practical animation of liquids. In *Proc. of ACM SIGGRAPH* (2001), pp. 23–30.
- [IGLF06] IRVING G., GUENDELMAN E., LOSASSO F., FEDKIW R.: Efficient simulation of large bodies of water by coupling two and three dimensional techniques. *ACM Trans. Graph.* 25 (2006), 805–811.
- [KM90] KASS M., MILLER G.: Rapid, stable fluid dynamics for computer graphics. In *Proc. of ACM SIGGRAPH* (New York, NY, USA, 1990), ACM Press, pp. 49–57.
- [LvdP02] LAYTON A. T., VAN DE PANNE M.: A numerically efficient and stable algorithm for animating water waves. *The Visual Computer* 18, 1 (2002), 41–53.
- [MCG03] MÜLLER M., CHARYPAR D., GROSS M.: Particle-based fluid simulation for interactive applications. *Proc. of the 2003 ACM Siggraph/Eurographics Symposium on Computer Animation* (2003), 154–159.
- [MDSB02] MEYER M., DESBRUN M., SCHRODER P., BARR A.: Discrete differential-geometry operators for triangulated 2-manifolds. *Visualization and Mathematics III* (2002), 35–57.
- [New59] NEWMARK N. M.: A method of computation for structural dynamics. *ASCE J. Eng. Mech. Div.* 85 (1959), 67–94.
- [OH95] O'BRIEN J. F., HODGINS J. K.: Dynamic simulation of splashing fluids. In *CA '95: Proceedings of the Computer Animation* (1995), pp. 198–205.
- [PP93] PINKALL U., POLTHIER K.: Computing discrete minimal surfaces and their conjugates. *Experimental Mathematics* 2, 1 (1993), 15–36.
- [PP02] POLTHIER K., PREUSS E.: Identifying vector fields singularities using a discrete hodge decomposition. *Visualization and Mathematics III* (2002).
- [Sta99] STAM J.: Stable fluids. *ACM Trans. Graph.* (1999), 121–128.
- [Sta03] STAM J.: Flows on surfaces of arbitrary topology. *ACM Trans. Graph.* (2003), 724–731.
- [SY04] SHI L., YU Y.: Inviscid and incompressible fluid simulation on triangle meshes: Research articles. *Comput. Animat. Virtual Worlds* 15, 3-4 (2004), 173–181.
- [SY05] SHI L., YU Y.: Taming liquids for rapidly changing targets. *Proc. of the 2005 ACM Siggraph/Eurographics Symposium on Computer Animation* (2005), 229–236.
- [TKPR06] THÜREY N., KEISER R., PAULY M., RÜDE U.: Detail-Preserving Fluid Control. *Proc. of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2006), 7–12.
- [TLHD03] TONG Y., LOMBEYDA S., HIRANI A. N., DESBRUN M.: Discrete multiscale vector field decomposition. *ACM Trans. Graph.* (2003), 445–452.
- [TMPS03] TREUILLE A., MCNAMARA A., POPOVIC Z., STAM J.: Keyframe control of smoke simulations. *ACM Trans. Graph.* 22, 3 (2003), 716–723.
- [TRS06] THÜREY N., RÜDE U., STAMMINGER M.: Animation of Open Water Phenomena with coupled Shallow Water and Free Surface Simulations. *Proc. of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2006), 157–164.
- [WMT07] WANG H., MILLER G., TURK G.: Solving general shallow wave equations on surfaces. *Proc. of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2007), 229–238.