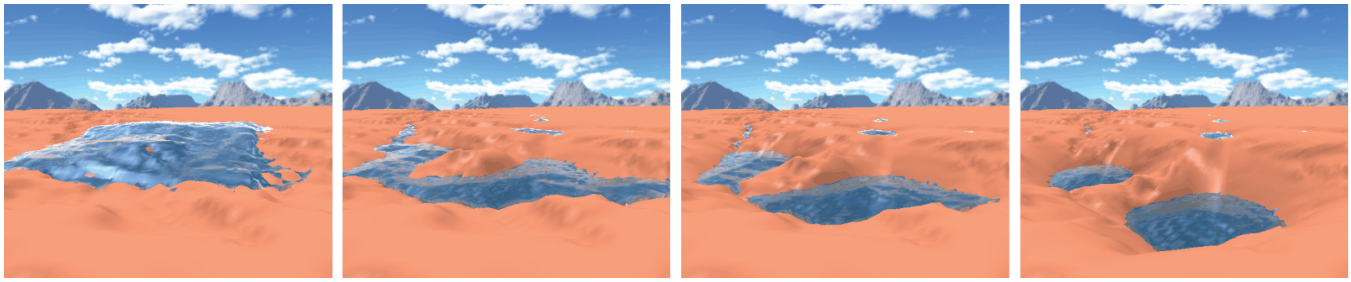
GPU Based Fluid Animation over Elastic Surface Models

*Bruno Barcellos S. Coutinho and Gilson Antonio Giraldi
National Laboratory for Scientific Computing
Petropolis, RJ, Brazil
(lsdelphi, gilson)@lncc.br

Antonio L. Apolinário Jr.
State University of Feira de Santana
Bahia, BA, Brazil
apolinario@ecomp.uefs.br



Keywords:: Real-time Animation, Cellular Automata, Fluid Simulation, Elastic Surfaces, GPU programming.

*Bruno Barcellos now is a master student in the Computer Graphics Laboratory (LCG) at Federal University of Rio de Janeiro (UFRJ-COPPE)

Abstract

In this work, we focus on flow animation in elastic surfaces described by mass-spring models for computer game applications. We propose the combination of an efficient fluid model, that does not require solution of complicated equations, with a mass-spring approach to simulate the deformable surface. Firstly, we describe the fluid model for simulating the flow and its GPU implementation. The simulation method is based on a particle system, that evolves over a lattice. This lattice is defined over the surface domain. A set of local rules determine the interaction between particles. The elastic surface is simulated by a GPU based mass-spring system, geometrically represented by a regular mesh. The fluid particles are guided by the surface topography interacting with the elastic mesh due to external, elastic and damping forces. In the experimental results we emphasize the fact that physically plausible flow/deformation phenomena can be efficiently reproduced and animated in real time by the combined technique.

1 Introduction

Physically-based techniques for the animation of natural elements like fluids (gas or liquids), flood, elastic, plastic and melting objects, among others, have taken the attention of the computer graphics community [Iglesias 2004; Nealen et al. 2005; Deusen et al. 2004]. In particular, techniques in the field of Computational Fluid Dynamics (CFD) have been applied for fluid animation in applications that involve the interaction between the fluid and deformable solids [Müller et al. 2004; Keiser et al. 2005; Genevaux et al. 2003; Chentanez et al. 2006].

A common approach in this area relies on top down viewpoints that use 2D/3D mesh based techniques in conjunction with fluid equations [Chentanez et al. 2006]. Other possibility is to apply mass-spring systems to model the elastic solid [Genevaux et al. 2003]. Mass-spring models are well suited to animation due to its flexibility to handle non-rigid solid properties, its easy manipulation and implementation. Besides, mass-spring models can be faster then their counterpart in continuous mechanics, and so, more suitable for real time applications specially when GPU capabilities are explored [Mosegaard and Sorensen 2005].

However, a common challenge is this area is the cost of the computational animation of the solid-fluid interaction. Recently, it has been demonstrated the advantages of using *bottom up* models for surface water flow simulation which are inspired in Lattice Gas Cellular Automata (LGCA) and LBM techniques [Barcellos et al. 2007; Fan et al. 2005]. These methods are cheaper than the traditional ones for fluid simulation, because there is no need to solve Partial Differential Equations (PDEs) to obtain a high level of description [Frisch et al. 1987]. Convincing animations with real time frame rates can be generated by lattice methods, as demonstrated in [Judice et al. 2008] for computer game applications involving surface flow over terrains.

In this paper we combine the mass-spring model described in [Mosegaard and Sorensen 2005] with an extension of the surface flow simulation proposed in [Barcellos et al. 2007] in order to get a general system for GPU animation of fluids over elastic surfaces. The combined technique is the main contribution of this paper. The advantages of this method over other approaches are its simplicity for implementation and the gain in computational efficiency allowing real time frame rates. The basic data structures of the model are a polygonal representation of the surface and a regular lattice with nodes $(i, j) \in L \times L$, where $L \subset \mathbb{N}$. Up to now, we consider surface flow simulation over deformable $2D$ manifolds which have a global parameterization $\varphi : D \rightarrow \mathbb{R}$ where $\varphi(x, y, t)$ is the elevation of the surface at point (x, y), at time t. So, each lattice node (i, j) is a projection of a surface point $(i, j, \varphi(i, j, t))$. The surface deforms as a mass-spring system, due to internal forces (elastic and damping ones) and external forces (gravitational, etc.). Extensions for more general parametric surfaces will be discussed in section 6. Potential applications in games are erosion effects and deformation of objects, modeled by elastic surfaces, under water accumulation.

The fluid model uses a LGCA approach. Therefore, particles can only move along the edges of the lattice and their interactions are based on local rules. Differently from traditional LGCA approaches, in our model more than one particles may share the same node position (i, j). Particles move according to the surface topography and the fluid configuration nearby. There is a counter in each lattice node used to keep the number of particles in the corresponding (i, j) position. When particles move over the lattice the particles counters change and, consequently, the flow distribution is updated and the surface deforms. The obtained result is a function $f(i, j, t)$ which gives the elevation of the free surface of the fluid, at the point (i, j), in the simulation time t.

The paper is organized as follows. Sections 2 gives a survey of works in interaction between fluids and elastic materials. In Section 3 we present our technique for simulating surface water flow over deformable surfaces. Its GPU implementation is described in Section 4. The experimental results are presented on Section 5. Finally, we present the conclusions and future works on Section 6.

2 Interaction of Fluids with Deformable Objects

The main focus of this work is the animation of fluids interacting with deformable objects. Generally speaking, this subject includes:

(1) Representation of the object geometry; (2) The modeling of mechanical behavior of elastic solid; (3) A suitable model for fluid simulation; (4) A model for interaction of the flow with the object; (5) Visualization and rendering issues.

The $3D$ object geometry is usually represented by mesh based methods that offer the support for Finite Element techniques [Müller et al. 2002; Debunne et al. 2001]. Bidimensional manifolds can be represented by using implicit surfaces [Zhu and Bridson 2005], triangulated meshes or subdivision surfaces with local parameterization for representation [Guendelman et al. 2005; Stam 2003].

The mechanical behavior of elastic objects (item (2)) can be described by the continuum elasticity theory that models how the objects deform under applied forces. In this case, constitutive laws are used for the computation of the symmetric internal stress tensor σ, and a conservation law gives the final PDE that governs the dynamic of the material [Nealen et al. 2005]. Then Finite Element methods, Boundary Element Method (BEM), Finite Differences (FDM) or SPH techniques, are applied to solve the PDE [Müller et al. 2002; James and Pai 1999; Terzopoulos and Witkin 1988; Debunne et al. 2000; Desbrun and Gascuel 1996]. Other possibility is to apply discrete models, based on mass-spring systems [Genevaux et al. 2003; Mosegaard and Sorensen 2005]. In this case, the object geometry is represented by a mesh and its nodes are treated like mass points while each edge acts like a spring connecting two adjacent nodes. The relation between mass-spring models and the continuum elasticity theory was examined in the reference [Delingette 2008]. The conclusion is that methods that are based on the continuum mechanics are more realistic than their discrete counterparts. However, mass-spring models are simple to implement and can be faster then the continuous ones, and so, more suitable for real time applications [Volino and Magnenat-Thalmann 2000; House and Breen 2000; VanGelder and Wilhelms 1997; Etzmuss et al. 2003].

The item (3) involves numerous works that can be coarsely classified in non-physically and physically based models [Iglesias 2004; Deusen et al. 2004]. Our work belongs to the later class, which can be subdivided in PDEs and Lattice based techniques [Frisch et al. 1987; Iglesias 2004].

PDEs methods involve continuous fluid equation, like the Navier-Stokes ones, and numerical techniques based on discretization approaches that can be Lagrangian (Smoothed Particle Hydrodynamics (SPH) [Liu and Liu 2003], method of characteristics [Stam 1999], Moving-Particle Semi-Implicit [Premoze et al. 2003]) or Eulerian (Finite Element) ones [Foster and Metaxas 1997]. The former class uses particle systems for discretization while in the latter model properties are computed for a set of stationary points, usually connected in a mesh, and updated to get the time evolution of the continuous media. PDE based models are called *top down* approaches because continuous mechanics concepts are applied to derive the PDE, which governs the continuous dynamics, while the particle view appears as a consequence of discretization methods.

Alternatively, lattice methods comprised by the Lattice Gas Cellular Automata (LGCA) - FHP and HPP are the most known ones - and Lattice Boltzmann (LBM) can be used [Frisch et al. 1987; Wei et al. 2004; Ye et al. 2004]. The basic idea behind these methods is that the macroscopic dynamics of a fluid is the result of the collective behavior of many microscopic particles. The LGCA will follow this idea but simplifying the dynamics through simple and local rules for particles interaction and displacements while LBM constructs a simplified kinetic model, a simplification of the Boltzmann equation, that incorporates the essential microscopic physics so that the macroscopic averaged properties obey the desired macroscopic equations [Chen and Doolen 1998]. Therefore, lattice methods do not apply PDEs to simulate fluids which can reduce the computational cost of the animation. Recently, it was demonstrated the advantages of using the philosophy behind FHP and HPP models for computer graphics applications and for surface water flow simulation [Barcellos et al. 2007]. Besides, despite of some intrinsic limitations, multiscale techniques were applied to demonstrated that the FHP and LBM models can reproduce Navier-Stokes behaviors [Frisch et al. 1987].

Particularly, the FHP model has a number of advantages over more traditional numerical methods, particularly when fluids mixing and phase transitions occur [Rothman and Zaleski 1994]. The simulation is always performed on a regular grid and can be efficiently implemented on a massively parallel computer. Solid boundaries and multiple fluids can be introduced in a straightforward manner and the simulation is performed equally efficiently, regardless of the complexity of the boundary or interface [Buick et al. 1998]. In addition there are not numerical instability issues because the evolution follows integer arithmetic. However, system parameterization (viscosity, for example) is a difficult task in such lattice models and they are less realistic than PDE based models.

The item (4), interaction between deformable solids and fluids, can be addressed by hybrid methods (fluid is a continuum medium and the solid is represented as a discrete one), SPH based techniques and variational approaches [Genevaux et al. 2003; Solenthaler et al. 2007; Batty et al. 2007]. In [Genevaux et al. 2003] the interaction problem is addressed by an hybrid technique in which the deformable solid is represented through a mass-spring network and the fluid is simulated by Navier-Stokes equations and an Eulerian method. The key idea is to apply spring forces to mass-less marker particles in the fluid and the nodes of the mass-spring network. In [Müller et al. 2004] authors proposed another hybrid method, in which the fluid is represented by a SPH approach and the solid is represented by polygonal meshes. To model the solid-fluid interaction it is used virtual boundary particles which are placed on the surface of the solid objects according to Gaussian quadrature rules. Such approach allows the computation of smooth interaction potentials that yield stable simulations at interactive rates.

A subclass of hybrid methods deals with the interaction between fluids and bidimensional manifolds modeled by a lower dimensional (moving) triangulated surface. The fluid model is a continuous one, simulated by Navier-Stokes plus SPH or grid based techniques. These approaches deals with the specific problem of preventing the leaking of fluid across the thin solids. In [Guendelman et al. 2005] it is proposed a ray cast based visibility method to perform this task and a new technique for properly enforcing incompressibility near the triangulated surface. When using the SPH method, robust point face collisions detection algorithms must be used to prevent fluid leaking [Bridson et al. 2002]. In addition, fluid flows can be simulated on $2D$ manifolds, represented by (continuous) subdivision surfaces or unstructured meshes, following traditional [Stam 2003] or LBM approaches [Fan et al. 2005]. Interaction between Navier-Stokes fluids and digital terrain models is another subclass of fluid-surface interaction [Ye et al. 1996]. Fluid equation on height fields, like shallow water equations [Thürey et al. 2006; Kass and Miller 1990], where applied for surface flow simulation. These methods and our technique share the idea of modeling the terrain and water surface as height fields. Besides, a hybrid particle and implicit surface approach to simulating water was proposed in [Foster and Fedkiw 2001], which led to the particle level set method of [Enright et al. 2002].

The interaction fluid-solid can be seen as a simplified case of two-phase systems. This is explored in [Solenthaler et al. 2007] where it is presented an unified SPH framework for the simulation of melting and solidification which can be straightforward adapted for interaction between fluids and deformable solids. The technique uses the SPH method for the simulation of liquids, deformable as well as rigid objects, which eliminates the need to define an interface for coupling different models. Additionally, a new surface reconstruction technique, based on considering the movement of the center of mass, is proposed to reduce rendering errors in concave regions.

Variational approach follows the usual philosophy for strong variational techniques: a functional (the Lagrangian) is defined such that the governing equation is the Euler-Lagrange equation for minimizing that functional. In [Batty et al. 2007] the governing equation is the pressure PDE and the functional computes the total kinetic energy of the system. The solution in this formulation is the divergence-free fluid field and compatible solid velocities that minimizes the total kinetic energy.

Finally, visualization and rendering techniques must be applied to ensure the desired level of realism or visual effect. Realistic ren-

dering methods can properly account for this task through several algorithms including path tracing, bidirectional path tracing [Heckbert 1990], Metropolis light transport [Veach and Guibas 1997], and photon mapping [Jensen and Christensen 1998]. The interested reader is also encouraged to browse interesting reviews in this area [Adabala and Manohar 2002; Iglesias 2004].

In this paper we focus on the interaction between fluids and deformable $2D$ manifolds modeled, respectively, by a lattice based technique and mass-spring systems. The work is based on a fluid model proposed in [Barcellos et al. 2007] that shares the basic features of the HPP and FHP models. Therefore, we do not apply PDEs, LBM or shallow water methods and the fluid evolution is computed by integer arithmetic. The mass-spring system is simulated by the GPU implementation presented in [Mosegaard and Sorensen 2005]. Next, we describe the proposed model and its GPU implementation.

3 Particle-Based Simulation Model

In [Barcellos et al. 2007] we propose a particle based model for flow simulation over static surfaces which is inspired in a lattice gas model [Frisch et al. 1987]. In this paper we modify that model for surface flow simulation over deformable $2D$ manifolds which have a global parameterization $\varphi : D \rightarrow \mathbb{R}$ where $\varphi(x, y, t)$ is the elevation of the surface at point (x, y), at the time t. The extension for general manifolds is discussed in Section 6.

The fluid-solid interaction model is composed by three basic elements: (1) A discretization of φ, composed by a regular lattice and the value of φ in each lattice node (i, j); (2) A mass-spring system to model the elastic surface; (3) A particle based water flow simulation.

The fluid model uses a cellular automaton approach and is based on the $2D$ regular lattice, a particle system and local rules for particles displacements. The Figure 1 illustrates the surface domain and the regular lattice which resolution is the same of that one used in the φ discretization. In this figure we highlight a (i, j) node of the lattice and its 4 neighbors given by $(i-1, j)$, $(i, j-1)$, $(i+1, j)$ and $(i, j+1)$, numbered V_1, V_2, V_3 and V_4, respectively. Besides, we can estimate the surface slope at point $(x, y) \in D$ through the projection of the surface normal over the domain D.

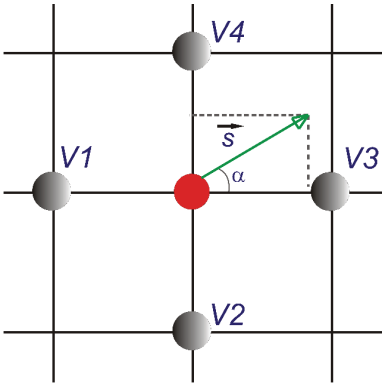


Figure 1: *Neighborhood of a lattice node.*

The particles movement are discrete in space and time; that means, each particle moves from one lattice node to another one in the neighborhood, in the time step $\triangle t$. Specifically, from the lattice topology in Figure 1, we have four directions to consider for the red node (i, j), which can be indexed as $\{1, 2, 3, 4\}$. So, we score these directions, as described by Algorithm 1. The output of this algorithm is the list *Flow Directions*, which receives the four directions sorted according to the priorities for the flow; that means, we have the first, second, third and forth flow direction.

A particle moves from a node (i, j) to a nearest neighbor $V_k = (m, n)$ if k is the first element of *Flow Directions* that satisfies $\mathbf{f}(V_k, t) < \mathbf{f}(i, j, t)$. Besides, a four bit string $n_1(i, j)$, $n_2(i, j)$,..., $n_4(i, j)$ is assigned to each node (i, j) of the lattice. We set

Algorithm 1 : Build Flows Directions ()

```
Initialization:
    L = {1, 2, 3, 4}
while L not null; do
    m ← arg min_k {arc(s⃗, V_k − (i, j)), k ∈ L};
    Remove m from L;
    Insert m in Flow Directions;
end while
```

$n_k(i, j) = 1$ if the node (i, j) has one particle to send to the neighbor k. In the actual implementation each lattice node can send at most one particle at a time, and consequently, can receive at most 4 particles in each simulation time.

For each grid node (i, j) a particle counter is associated, which is used to define the high of the free surface of the flow. The particle system is used in order to update the field of counters. The result is a function

$$f(i, j, t) = \varphi(i, j) + \beta \cdot counter(i, j, t), \tag{1}$$

which gives the elevation of the free surface flow $\mathbf{f}$, at the point (i, j) in the simulation time t (β is a scale parameter).

In this work, the surface deforms as a mass-spring system, subject to internal forces (elastic and damping ones) and external forces. Therefore, the surface elevation φ and slope field $\vec{s}$ are non-stationary fields (as well as the $\mathbf{f}$ in Expression (1)).

The mass-spring system follows the reference [Mosegaard and Sorensen 2005]. The surface nodes works as masses and the edges defines the linear springs with damping. So, given a particle i with mass m_i and position vector $\mathbf{x}_i$, the force system is composed by the elastic ($\mathbf{f}^i_{elastic}$), gravitational ($\mathbf{f}^i_{grav}$) and damping ($\mathbf{f}^i_{damp}$) forces, defined respectively, by:

$$\mathbf{f}^i_{elastic} = \sum_{j=1}^{4} k_{ij} \left(l_{ij} - \|\mathbf{x}_i - \mathbf{x}_j\|\right) \frac{(\mathbf{x}_i - \mathbf{x}_j)}{\|\mathbf{x}_i - \mathbf{x}_j\|}, \tag{2}$$

where k_{ij} is the stiffness of the spring linking the nodes $\mathbf{x}_i$ and $\mathbf{x}_j$ and l_{ij} the spring rest length;

$$\mathbf{f}^i_{grav} = (\beta_i counter(i))\mathbf{z} + m_i \mathbf{g}, \tag{3}$$

$$\mathbf{f}^i_{damp} = \gamma_i \dot{\mathbf{x}}_i, \tag{4}$$

where β_i is a scale parameter, $\mathbf{g}$ is the gravity field, γ_i is the damping factor and $counter(i)$ holds the number of particles accumulated in the corresponding position. Following Newton's Laws, we get the following evolution equation:

$$m_i \ddot{\mathbf{x}}_i = \mathbf{f}^i_{elastic} + \mathbf{f}^i_{damp} + \mathbf{f}^i_{grav}. \tag{5}$$

This system of ordinary differential equations can be efficiently solved by the Verlet integration technique [Mosegaard and Sorensen 2005]:

$$\mathbf{x}_i(t + h) = 2\mathbf{x}_i(t) - \mathbf{x}_i(t - h) + \ddot{\mathbf{x}}_i(t) h^2. \tag{6}$$

Now, let us put the fluid model and the mass-spring together to render the basic algorithm. At the initialization, the particle counters field, the initial surface geometry and velocity of the nodes must be defined. Then, in the simulation loop, the forces in Expressions (2)-(4) are computed and the differential Equation (5) is integrated through the Verlet scheme in Equation (6). Next, the new normal field is computed and projected to get the new field $\vec{s}$. Then, the Algorithm 1 is applied to obtain the *Flow Directions* field. Based on the free surface high and on the *Flow Direction*, the n_k field is built. This field is used to compute the number of particles that each node

(i, j) sends and receives from the neighbors, and, consequently, the particles counter field is updated. Finally, the deformable surface and the surface flow are visualized and the next simulation step starts. At each interaction, we re-compute the free surface elevation, given by expression (1), in order to mimic the transference of velocity from the terrain to the particles.

Besides, we can include physical effects like evaporation $E(i, j, t)$ and precipitation $P(i, j, t)$ by using an intuitive balance model given by:

$$counter(i, j, t) \leftarrow counter(i, j, t) + (P - E). \qquad (7)$$

This model is based on local rules which try to simulate some aspects of overland flows [Ye et al. 1996].

We shall emphasize that the fluid simulation algorithm does not suffer from numerical stability issues because the floating-point operations are simple (normal field and computation of expressions (1)) and the update of the counter field is based on simple comparisons and integer arithmetic. Besides, volume conservation is straightforward verified because all operations are conservative with respect to the number of particles.

4 GPU Implementation

The algorithm described in Section 3, for surface flow simulation over deformable surfaces, is based on a regular lattice and local rules for surface deformation and fluid animation. Therefore, it may be very suitable for a GPU implementation. The main idea is to encode the information needed in the simulation as textures into the video memory. The Algorithm 2 gives an overview of the whole GPU processing. The initialization step computes the initial values to the nodes particle counters, the surface geometry and the initial velocity to each node. These values are computed in the CPU and stored in different matrixes. Then, the simulation loop starts, performing, for each time step, the simulation and visualization processes. The simulation process update of the surface geometry, based on the mass-spring simulation, and the fluid simulation updates the fluid particles distribution among the lattice nodes. Finally, the visualization step begins, drawing the surface and the water surface. The loop process, involving simulation and visualization is completely performed on the GPU.

Algorithm 2 : GPU Simulation and Visualization()

```
Initialization:
   Compute
      Particles Counters.
      Surface Geometry.
      Velocity.
   Transfer Data to GPU.
loop
   Simulation:
      GPU-Based Mass-Spring Simulation();
      GPU-Based Fluid Simulation();
   Visualization:
      Draw Mass-Spring Surface;
      Draw Water Free Surface;
end loop
```

Like others GPU-based algorithms, all data needed to simulation and visualization processes are stored in GPU memory, mapped as textures. The data flow of algorithm 2 is illustrated by Figure 2.

The GPU-based mass-spring simulation follows the reference [Mosegaard and Sorensen 2005]. At each interaction of the main loop a $2D$ texture, represented in Figure 2 as **Properties texture**, encoding the automaton configuration is generated as follows. To each texture point (i, j) it is associated the index k, such that each color channel represents a different data: the terrain elevation $\varphi(i, j)$ and a particle counter, respectively channels R and B. The connectivity among masses can be implicitly represented because surface samples are organized in a regular lattice with a simple fixed 4-connected neighborhood rule. **Surface** textures stores the particles positions. In fact, as long as Verlet integration method is used (equation 6)), three surface stages must be stored during the simulation, each one representing a different time step: current (t), backward $(t-1)$ and forward $(t+1)$. The output of the spring-mass simulation includes the new surface geometry $(t + 1)$ and the **flow** texture. The **Flow** data encodes the four flow directions in each node (i, j) of the lattice, as described in section 3 and algorithm 1. Based on the flow directions and the particle counters from previous simulation step, the fluid simulation begins.

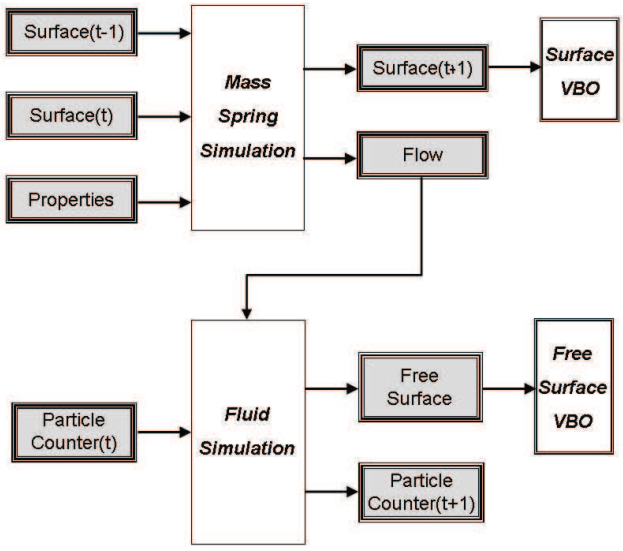


Figure 2: *The data flow involved in the simulation and visualization processes. The gray boxes represent data mapped in GPU using texture memory.*

During the fluid simulation an **Upward Flow** is build, where the $n_k(i, j)$ bit string associated with each lattice node is generated as described in section 3. In this process, given a lattice node (i, j), we sequentially check the Flow Direction texture values and take the first one, say k, that satisfies $\mathbf{f}(V_k, t) < \mathbf{f}(i, j, t)$.

Using the **Upward Flow** and the **Particle Counter** at interaction t, a new **Particle Counter** is generated at time $t + 1$. The elevations of the free surface flow $\mathbf{f}(i, j, t)$ at each (i, j) node of the lattice is also calculated, as Figures 2 illustrate.

The particle counters are updated evaluating the number of particles that the node (i, j) receives and sends, respectively. The value of received particles in a node (i, j) is computed by adding the suitable values of the **Upward Flows** at the neighbors (V_1, V_2, V_3, V_4). The number of particles the node (i, j) sends is obtained by adding the $n_k(i, j)$ values.

The elevation of the free surface at a lattice node (i, j) is finally obtained by adding the terrain elevation and the particle counter value, weighted by a scale factor at that node (equation 1). The new elevation is used to generate the **Free Surface** texture.

The two textures **Surface**$(t+1)$ and **Free Surface**, which contains, respectively, the terrain and the water surfaces, must be sent to an *Frame-Buffer Object (FBO)* [Woo et al. 1999], in order to be used to update data for visualization process.

It is important to emphasize that we need a new **Particle Counter Texture** to encode the configuration at time $t + 1$ due to the restriction of the shader-based implementation, using GLSL (OpenGL Shading Language) [Rost 2004]. In context of a shader, a texture can be strictly *read-only* or *write-only*, and the GPU parallel architecture uses several processing units to compute a field. Also, the fragment processors are able to update more than one texture in parallel through the technique called *Multiple Render Target* [Rost 2004].

The **Surface** and the **Free Surface** textures are used as input to the *Vertex Buffer Object* (**VBO**). The **VBO** will be managed as a *Vertex Array* to render the final visualization of the free surface of the

fluid. Therefore, the whole computation (simulation and visualization) is performed in the GPU. This scheme allows us to minimize the information flow between the CPU and the GPU and improves the performance of the application.

5 Experimental Results

We developed a software in $C++$ language that allows a user to deposit densities on the surfaces as well as to control parameters of the simulation. The pictures included in this paper are snapshots obtained from that program. The corresponding videos can be found in: `http://virtual01.lncc.br/~barcellos/videosSBGames2009.zip`. The rendering is implemented by standard OpenGL calls and the shaders with OpenGL Shading Language. The experiments were performed in a Intel Core 2 Duo 2.66 GHz, with 4 GB of RAM and a Video Card NVidia GeForce 8800 GTX, running Windows XP.

In these experiments we highlight aspects of our model that can be useful for computer graphics applications: simple to simulate complex configurations and computational efficiency.

We apply 2 kinds of external forces: vertical ones and wave-like forces (Section 5.1). Surface configurations include multiply connected domains (Section 5.2) and complex geometries (Section 5.3). Besides we simulate effects like evaporation and precipitation (Section 5.3) and present the CPU and GPU performance comparison (section 5.4).

The lattice resolution is 256×256 and the number of particles used in each test is given in Table 1. The intensity of the vertical forces are scaled according to the requirements of each example. The masses and springs located at the boundary of the domain are kept rigid and we set $m = 1.0$ everywhere. The time step h of the simulation was set to $h = 0.1$. We render the terrain and the water surface as height fields. The visualization was implemented in OpenGL. To increase the scene realism we applied the environment mapping technique and Fresnel effects to incorporate the physical laws of reflection and refraction for the water free surface rendering. The shaders was implemented in OpenGL Shading Language.

Table 1: *Number of particles for the experiments of Sections 5.1, 5.2, 5.3.*

Configuration	Particles
Figure 4	2377092
Figure 5	2005169
Figure 6	218700
Figure 7	218700
Figure 8	364500
Figure 9	211200
Figure 10	53550
Figure 11	218700

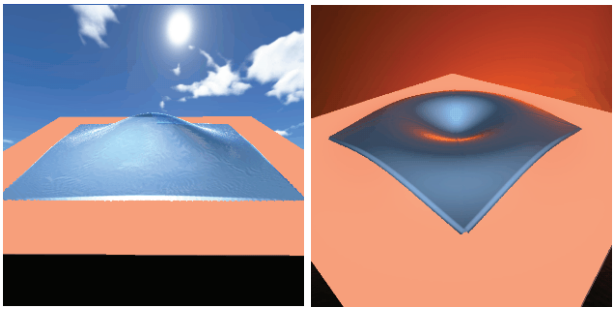
5.1 Vertical Forces

In this section we consider elastic surfaces over the influence of vertical external forces, like the gravitational one, that are applied during a time interval $[t_1, t_2]$ along the simulation, as follows:

$$\mathbf{f}_v(t) = \begin{cases} T \cdot \mathbf{z} & : \quad t_1 \leq t \leq t_2 \\ \mathbf{0} & : \quad otherwise \end{cases} \tag{8}$$

where $\mathbf{z}$ is the unitary vector in the vertical direction and T is a scale factor. The Figures 3.a-b show the initial configurations used in the experiments. Each spring has a spring rest length l_{ij} (see Expression (2)) equal to its length at the initialization.

The Figure 4 shows some snapshots of the animation which is generated starting from the initial configuration pictured in Figure 3.a. In this case, we apply a force given by Expression (8) with $T = 0.025$ $t_1 = 0$, $t_2 = 3200$. Then, we turn off this force and apply a descending one, given by $T = -0.025$ for $t \geq 3201$. The mass-spring parameters are: $\gamma = 0.05$ (damping); $k = 1.0$ (stiffness). Besides force (8) we add a gravitational force computed by:



(a) (b)

Figure 3: *Initial configurations that are used in the computational experiments: (a) Flat surface with a gaussian fluid distribution. (b) Mexican Hat surface with a uniform fluid distribution.*

$$\mathbf{f}^i_{grav} = (\rho * counter(i))\mathbf{z}. \tag{9}$$

with $\rho = -0.00005$. At the beginning of the simulation, we observe that the fluid remains at the center of the deformable surface because the surface deformation generates a lake (Figure 4.a). The force $\mathbf{f}_v$ deforms the surface in the upward direction generating a pyramid-like surface due to the fact that the surface boundary is rigid, as it can be seem in Figures 4.a-b. The descending force is applied and soon the surface starts a contraction process. The Figure 4.c shows the configuration at time $t = 4650$ in which the fluid spills out the top of the surface and deforms the surface at the same time. We observe a fully coupled interactions between three-dimensional deformation and fluid evolution. The surface geometry is modified resulting a fluid redistribution which then feed back and influence subsequent deformation. For instance, in Figure 4.d we observe the system configuration at $t = 5950$ showing new regions of fluid accumulation due to surface deformation. In these examples, the fluid particles are not allowed to go out the simulation domain. So, particles accumulate in the boundary nodes until the surface slope points inside the domain. That is why we observe some accumulations of fluid in the boundary of the surface.

Figure 5 shows the evolution of the same initial configuration depicted on Figure 3.a. However, in this case we set the damping zero and apply the force (8) at time $t = 750$ with $T = -20.0$. Each spring has a spring rest length l_{ij} (see Expression (2)) equal to its length at the initialization and $k = 50.0$.

Figures 5.a-b depicts the time step in which the fluid volume is suspended by the surface generating interesting effects of transparency and downhill flows. This configuration needs special comments. The mechanical behavior acts like a viscous fluid, such a gelatin, which flows slowly and generates pics of accumulations like the one observed. This is a consequence of our heuristic for fluid particles motion which imposes a constant horizontal velocity for particles displacements: the velocity of the particle projection is equal the ratio between the lattice edge and the time step which are constants in the model. Figure 5.c depicts the system configuration at interaction $t = 3440$. We observe a lake formation at the center of the surface as well as a large portion of fluid going down towards the central lake. The configuration shown in Figure 5.d occurs at time step $t = 4200$ and presents a portion of the fluid volume suspended by the surface, similarly to Figure 5.a. In fact, the video sequence shows periodic formations as a consequence of the symmetries in the initial configuration, the fact that we do not consider friction between the fluid and the surface and that the damping is zero in this case.

The Figure 6 shows a sequence of snapshots generated from the initial configuration depicted in Figure 3.b. We applied the force (8) with $T = -30.0$ at time $t = 1000$ and $T = 35.0$ at time $t = 3200$. The mass-spring parameters are: $\gamma = 0.075$ (damping); $k = 20.0$ (stiffness), l_{ij} is the length at the initial step. Besides

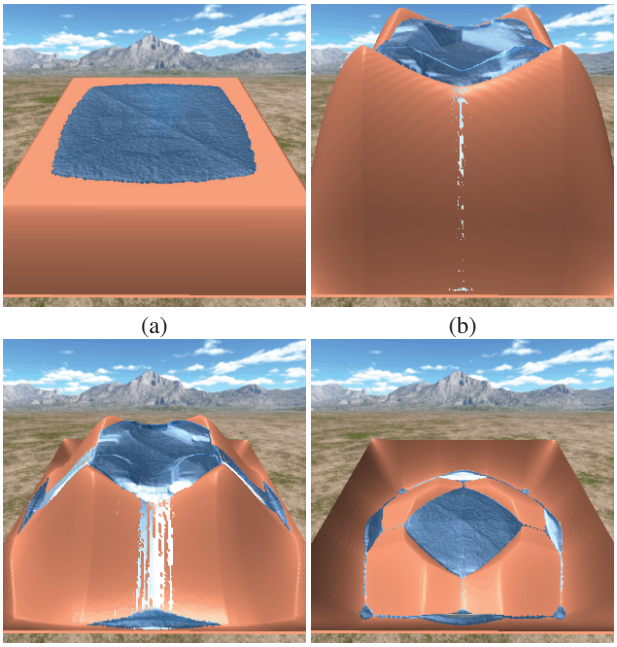


Figure 4: *(a) Flat surface with a gaussian fluid distribution at time step $t = 1400$. (b) The spring-mass system evolves generating a pyramid-like figure due to the fact that springs at the domain boundary are kept rigid and the force given by Expression (8) is applied. (c) Down hill surface flow at simulation time $t = 4650$. (d) Configuration at time $t = 5950$.*

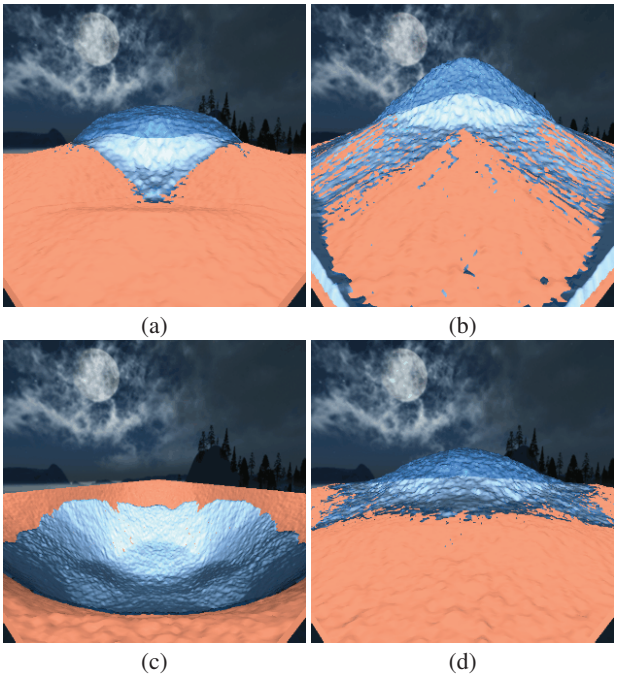


Figure 5: *Evolution of the initial configuration depicted in Figure 3, but without damping. (a) Frame at time step $t = 1910$. (b) Configuration with the fluid volume suspended by the elastic surface (time $t = 2610$); (c) Lake formation at time $t = 3440$. (d) System evolution at time $t = 4200$. Energy conservation and symmetries of the initial configuration implies periodic structures during the evolution.*

force (8) we add a gravitational force computed by expression (9) with $\rho = -0.0000025$.

The Figure 6.a shows a portion of the fluid flowing towards the boundary and another portion that generates a lake in the valley of the surface. The Figure 6.b shows this configuration some time further ($t = 448$) now depicting the fluid accumulation nearby the surface boundary and at the center. The system evolves and achieves the configuration shown in Figure 6.c in which the fluid is concentrated nearby the boundary and in the valley. Then, at the time $t = 1000$ the force (8) is applied with $T = -30.0$ deforming the system which achieves the configuration pictured on Figure 6.d. When the mass-spring system achieves zero kinetic energy, it starts to go up achieving the configurations shown in Figure 6.e and 6.f at times $t = 4648$ and $t = 7084$, respectively.

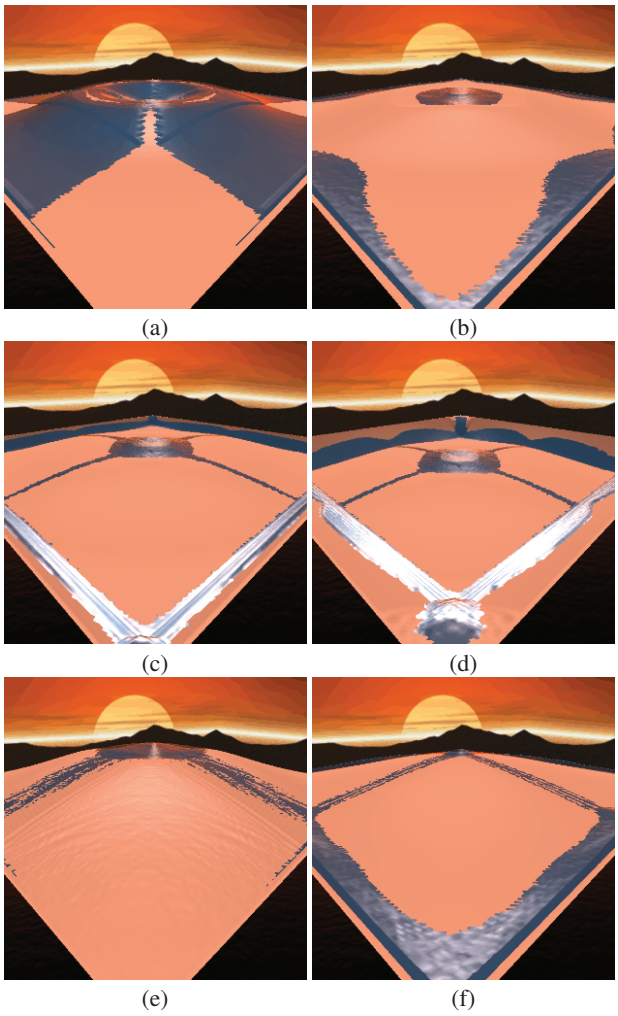


Figure 6: *(a) Mexican hat surface with a uniform fluid distribution at time step $t = 84$. (b) Configuration at time $t = 448$ showing the deformation of the surface nearby the surface valley; (c) Fluid concentration nearby the domain boundary and in the valley at time 1092. (d) Configuration at $t = 1232$ generated after the descending force is applied. (e) System in ascending movement: configuration at time $t = 4648$. (f) Snapshot obtained at time step 7084 of the simulation.*

The Figure 7 shows some snapshots for the evolution of the initial configuration depicted in Figure 3.b, but without damping. In this case we set the spring rest length $l_{ij} = 1.0$ at the initialization. The mass-spring parameters are: $\gamma = 0.0$ (damping); $k = 50.0$ (stiffness). We apply the force (8) with $T = -40.0$ at time $t = 1500$ and we add a gravitational force computed by equation (9) with $\rho = -0.00025$.

Once the rest length of the springs is $l_{ij} = 1.0$, the elastic energy at time $t = 0$ is not null. That is way we observe a little hill formation at the center in Figure 7.a, which dissipates as shown in Figure 7.b.

At time $t = 1500$ it is applied the force (8) with $T = -40.0$ generating the configuration pictured on Figure 6.c at time $t = 2260$. When the kinetic energy becomes null, the system starts ascending movement generating the configuration pictured on Figures 7.d.

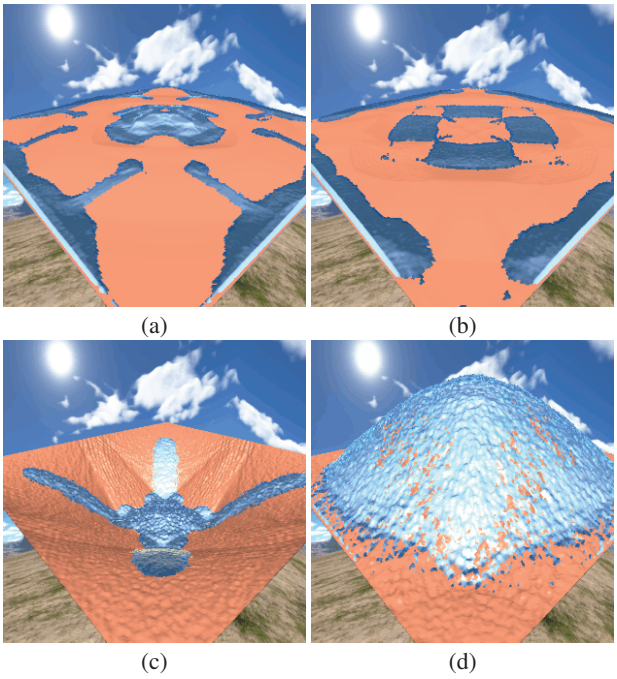


Figure 7: *(a) Mexican hat surface with a uniform fluid distribution at time step* 364*. (b) Configuration at time 560; (c) 2260. (d) 2772.*

The next example applies a force given by a periodic wave form namely a harmonic wave:

$$f_w(x, y, t) = A_0 \sin(bx - \omega t), \qquad (10)$$

where b is the wave number and ω is the angular frequency. This expression represents an unaltered propagation through a linear media in the electromagnetic theory [Corson and Lorrain 1970]. The parameters are: $\gamma = 0.2$ (damping), $k = 20.0$ (stiffness), $b = \omega = 0.1$ and $A_0 = 0.375$. We turned off the gravitational force ($\rho = 0.0$ in expression (9)).

The obtained patterns of the surface flow can be seen in the Figure 8. The initial configuration is similar the one pictured on Figure 3.a but now we take a uniform fluid distribution at the time $t = 0$. As we turned off the influence of the weight of the fluid in the spring-mass system we avoid surface deformation due the fluid weight. So, the fluid is carried by the surface waves spreading a bit due to fluid concentration (Figure 8).a. In this case the fluid is allowed to flow out the surface domain. So we do not have mass conservation, as it can be observed on Figure 8.b,d.

5.2 Complex Domains

Let us consider the simulation technique for a multiply connected domain, like the one pictured in Figure 9.a. To define the initial configuration it was generated a circular hole with radius 35.0 on the surface and an uniform fluid distribution composed by 211200 particles. We use a texture channel in order to distinguish the lattice nodes that are inside the surface domain from the outside nodes.

In this case, we have a boundary with two disconnected components. It is not required any extra mathematical machinery to deal with such topology because system rules do not undergo modifications. The parameters are: $\gamma = 0.125$ (damping), $k = 5.0$ (stiffness). We apply the force (8) with $T = -30.0$ at time $t = 200$, $T = 45.0$ at time $t = 1200$, $T = -45.0$ at time $t = 2900$ and we add a gravitational force computed by expression (9) with $\rho = -0.0000025$.

Figure 9.b shows the fluid distribution at time $t = 1120$ when system is going down. We observe a lake formation and the spring

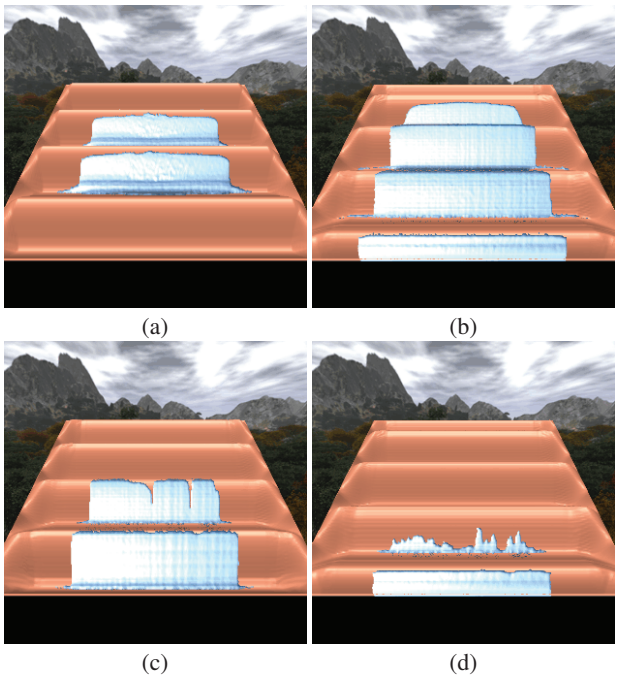


Figure 8: *(a) Flat surface with a uniform fluid distribution and harmonic waves at time step* 220*. (b) Configuration at time* 1150 *with fluid going out the domain; (c) Surface flow pattern at simulation time* 5040*. (d) Snapshot at time step* 6104*.*

mass deformation. Then, we apply the ascending force, at the time $t = 1200$. The Figure 9.c pictures a snapshot of the corresponding sequence showing a surface flow towards the surface hole. In this case the fluid is not allowed to go out the surface. Finally, at time $t = 2900$ we apply the force (8) with $T = -45.0$. As a result, the fluid nearby the inner boundary flows out and lake formations are observed, as it can be seen in Figure 9.d. If we want to predict such effects, we need to consider Navier-Stokes equations with suitable boundary conditions. However, if the aim is to explore the visual effect, we can just simulate and take the desired result at its time.

5.3 Erosion and Deformation

This section illustrates the erosion effects that can be obtained with the proposed technique. The mechanical behavior of the inviscid substrate under erosion is modeled by the spring-mass system coupled with the surface flow. The system dynamics generates a fluid redistribution that indicates high consequence areas and influence subsequent erosion modeled as deformation. A similar approach was presented in [Simpson 2004] but using a continuous thin-plate formulation.

The Figure 10 illustrates an example of erosion simulation caused by a constant rainfall. The original surface topography is pictured on Figure 10.a. We model the rainfall as a precipitation distribution over the terrain, or some part of it, as depicted in Figure 10.b. The precipitation is constant and applied during the first time step ($P(i, j, 0) = 12$ in Equation (7)). There is no evaporation ($E(i, j, t) = 0$ in expression (7)) and the parameters are: $\gamma = 0.15$ (damping), $k = 1.0$ (stiffness).

We observe in Figures 10.c-e the evolution of the terrain topography and the formation of flooded areas over the terrain. The Figure 10.f pictures the surface topography at time $t = 5300$. We can compare it with the initial configuration in Figure 10.a and observe the deformation generated.

The model can predicts high erosion potential at the lower, concave parts of hillslopes. If an almost flat area is encountered then the water simply spreads out into the flat area, deforming it, without any extra machinery. We observe this effect in the region nearby the left-hand corner of Figures 10.c-d. We shall observe that the particle model does not incorporates dissipative forces between the flow

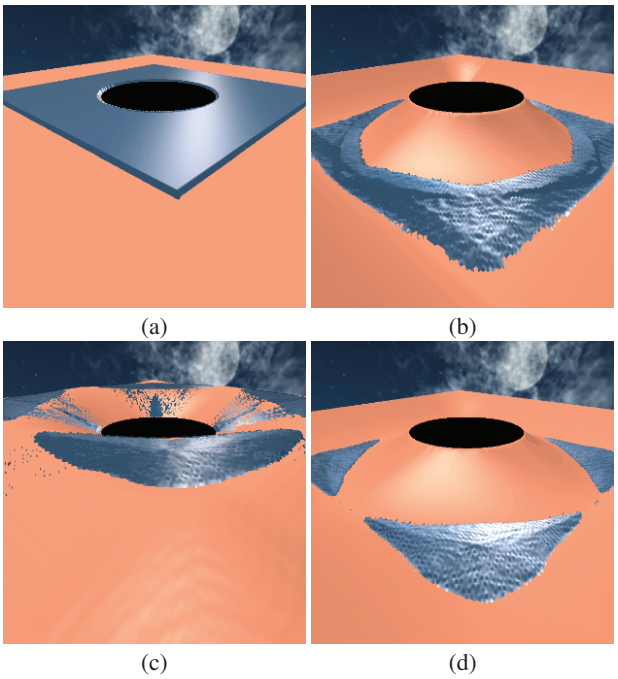


Figure 9: *(a) Multiply connected domain: Initial configuration. (b) System is going down at time step* 2380 *due to the application of force (8) with* $T = -30.0$*; (c) Configuration at time step* 2380 *when system is going up. (d) Snapshot at time step* 4676 *when system is going down again due to application of force (8) with* $T = -45.0$*.*

and the terrain. Besides, mass transport effects are not considered in our model.

Also, we can apply Expression (7) to include evaporation in the simulation. The Figure 11 shows six snapshots of such animation over the Grand Canyon topography, depicted in Figure 11.a. The evaporation is constant ($E(i, j, t_0 + \Delta t) = 1$), for all $(i, j) \in D$, with $t_0 = 2800$ and $\Delta t = 40$. The precipitation is modeled like in the previous example. The Figure 11.b is the initial step of the simulation. Figures 11.c-e show the fluid evolution and the surface deformation. In this case, there is no mass conservation due to the evaporation. So, the simulation stops when the fluid mass is null, that means, in the time step t^* in which the particle counters are zero: $counter(i, j, t^*) = 0$. Figure 11.f shows a snapshot closer to the final step. The parameters are: $\gamma = 0.15$ (damping), $k = 1.0$ (stiffness). In this example as well as in the Figure 10 each spring has a rest length given by its length at the time $t = 0$. When the fluid volume becomes null, the spring-mass will evolve towards its equilibrium configuration. That is way we stop the evolution when the fluid vanishes. More efficient mechanisms must be implemented to control the deformation in order to mimic erosion more efficiently.

5.4 CPU versus GPU Animation and Real Time

In this section we compare the performance of the CPU implementation with the performance of the GPU implementation of the animation algorithm. The Table 2 shows the rate of frames per second (**FPS**) obtained through these implementations, for the above examples. We observe a substantial improvement in the performance (more than 13 times better in all cases) with frame rates suitable for real time applications.

A major concern about using graphics hardware for general computation is the accuracy. The graphics hardware used supports 4 bytes per color channel which fits the requirement on the accuracy of the computation.

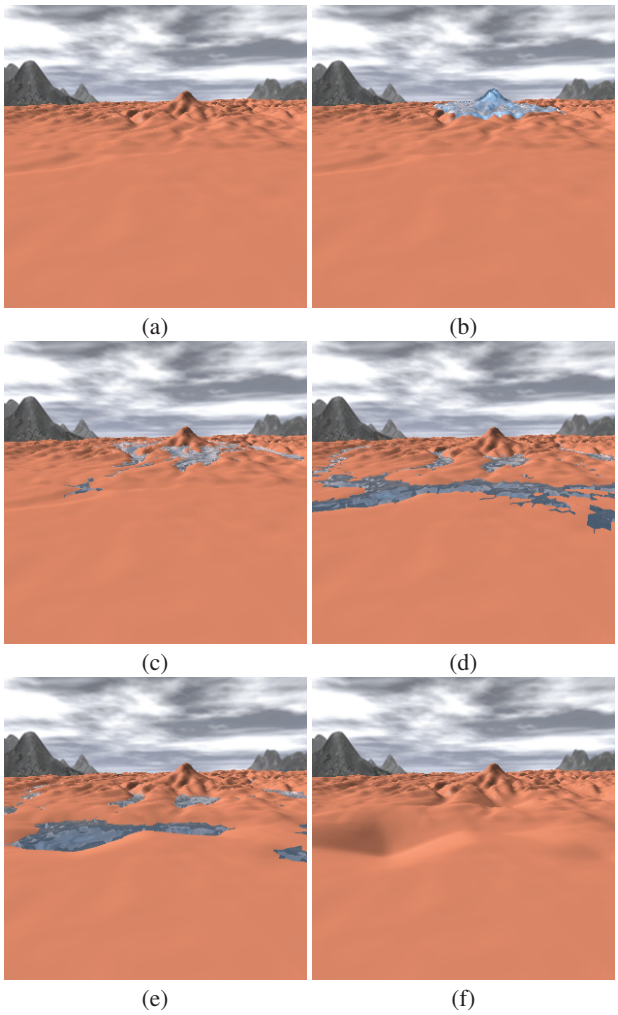


Figure 10: *(a) Puget Sound Model. (b) Fluid precipitation; (c) Modified terrain topography and surface flow at time* 196*. (d) Configuration at time step* 1500*. (e) Configuration at time step* 3800*. (f) Snapshot at time step* 5300 *showing the deformed surface topography.*

Table 2: *Frames per second rates (FPS) and number of particles for the experiments of Sections 5.1, 5.2, 5.3.*

Configuration	Particles	FPS in CPU	FPS in GPU
Figure 4	2377092	5.7	81.6
Figure 5	2005169	5.6	78.3
Figure 6	218700	5.5	83.2
Figure 7	218700	5.4	81.6
Figure 8	364500	5.3	81.6
Figure 9	211200	5.7	82.0
Figure 10	53550	5.7	78.4
Figure 11	218700	5.4	83.2

Table 3: *This table shows the FPSs for different numbers of particles when we use the same configuration of Figure 5.*

Configuration	Particles	FPS in CPU	FPS in GPU
Figure 5	2138306	5.5	79.9
Figure 5	706754	5.5	80.0
Figure 5	229522	5.5	80.1

6 Conclusions

In this work, we focused on surface flow animation in deformable surfaces described by mass-spring models for computer graphics applications. We proposed the combination of an efficient particle model for fluid simulation with a mass-spring approach to perform the animation. Both the CPU and GPU implementations were con-

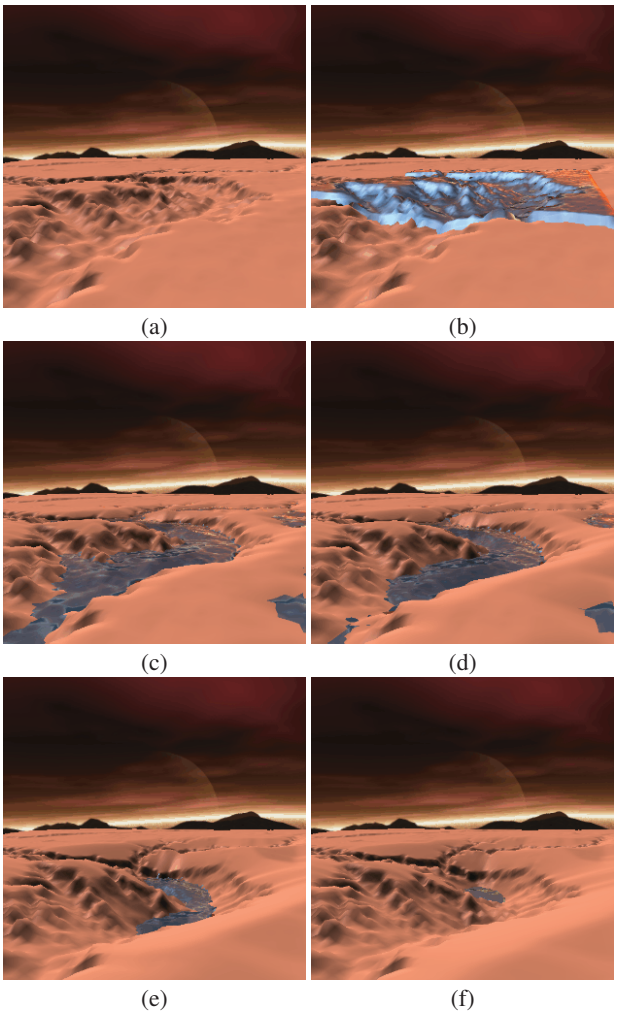


Figure 11: *(a) Grand Canyon model. (b) Initial precipitation area; (c) Snapshot at simulation time* 1568 *showing fluid distribution and surface deformation. (d) Configuration at time step* 2500*. (e) Snapshot at time step* 4060*: we observe the loss of fluid volume due to evaporation. (f) Terrain topography when fluid mass almost becomes null.*

sidered. In the experimental results we emphasize the abilities of our animation technique and the real time capabilities of the GPU implementation. Besides, we notice that the bottleneck of the whole simulation is the mass-spring system. In fact, as we can observe in Table 3, when the number of particles used is 2138306, 706754 or 229522, for the configuration picture on Figure 5, the measured FPS is almost the same.

Future directions in this work are the extension of the combined technique for general $2D$ manifolds represented by subdivision surfaces with local parameterization with special care to handle cross-patch boundary conditions. Besides, the introduction of random variables, to incorporate viscosity, and the comparison with a PDE-based technique will be performed soon.

Acknowledgements

The authors would like to thank the support provided by PCI-LNCC for this work.

References

ADABALA, N., AND MANOHAR, S. 2002. Techniques for realistic visualization of fluids: A survey. *Comput. Graph. Forum 21*, 1, 65–81.

BARCELLOS, B., GIRALDI, G. A., SILVA, R. L., APOLINRIO, A. L., AND RODRIGUES, P. S. S. 2007. Surface flow animation in digital terrain models. In *SVR 2007 - IX Symposium on Virtual an Augmented Reality*.

BATTY, C., BERTAILS, F., AND BRIDSON, R. 2007. A fast variational framework for accurate solid-fluid coupling. *ACM Trans. Graph. 26*, 3.

BRIDSON, R., FEDKIW, R., AND ANDERSON, J. 2002. Robust treatment of collisions, contact and friction for cloth animation. In *SIGGRAPH '02*, 594–603.

BUICK, J. M., EASSON, W. J., AND GREATED, C. A. 1998. Numerical simulation of internal gravity waves using a lattice gas model. *Int. J. Numer. Meth. fluids 26*, 657–676.

CHEN, S., AND DOOLEN, G. D. 1998. Lattice boltzmann method for fluid flows. *Annual Review of Fluid Mechanics 30*, 329–364.

CHENTANEZ, N., GOKTEKIN, T. G., FELDMAN, B. E., AND O'BRIEN, J. F. 2006. Simultaneous coupling of fluids and deformable bodies. In *SCA '06: Proc. of the 2006 ACM SIGGRAPH/Eurographics*, Eurographics Association, 83–89.

CORSON, D., AND LORRAIN, P. 1970. *Electromagnetic fields and waves*. W.H. Freemann, New York, USA.

DEBUNNE, G., CANI, M.-P., DESBRUN, M., AND BARR, A. 2000. Adaptive simulation of soft bodies in real-time. In *CA '00: Proc. of the Comp. Animation*, IEEE Computer Society, 15.

DEBUNNE, G., DESBRUN, M., CANI, M.-P., AND BARR, A. H. 2001. Dynamic real-time deformations using space & time adaptive sampling. In *SIGGRAPH '01*, 31–36.

DELINGETTE, H. 2008. Triangular springs for modeling nonlinear membranes. *IEEE Trans. on Vis. and Comp. Graph. 14*, 2, 329–341.

DESBRUN, M., AND GASCUEL, M.-P. 1996. Smoothed particles: a new paradigm for animating highly deformable bodies. In *Proceedings of the Eurographics workshop on Computer animation and simulation '96*, Springer-Verlag New York, Inc., New York, NY, USA, 61–76.

DEUSEN, O., EBERT, D. S., FEDKIW, R., MUSGRAVE, F. K., PRUSINKIEWICZ, P., ROBLE, D., STAM, J., AND TESSENDORF, J. 2004. The elements of nature: interactive and realistic techniques. In *ACM SIGGRAPH 2004 Course Notes*, 32.

ENRIGHT, D., MARSCHNER, S., AND FEDKIW, R. 2002. Animation and rendering of complex water surfaces. *ACM Trans. Graph. 21*, 3, 736–744.

ETZMUSS, O., GROSS, J., AND STRASSER, W. 2003. Deriving a particle system from continuum mechanics for the animation of deformable objects. *IEEE Trans. on Vis. and Comp. Graph. 9*, 4, 538–550.

FAN, Z., ZHAO, Y., KAUFMAN, A., AND HE, Y. 2005. Adapted unstructured lbm for flow simulation on curved surfaces. In *Proc. of the 2005 ACM SIGGRAPH/Eurographics Symp. on Comp. Anim.*, 245–254.

FOSTER, N., AND FEDKIW, R. 2001. Practical animation of liquids. In *SIGGRAPH '01*.

FOSTER, N., AND METAXAS, D. 1997. Modeling the motion of a hot, turbulent gas. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., 181–188.

FRISCH, U., D'HUMIÈRES, D., HASSLACHER, B., LALLEMAND, P., POMEAU, Y., AND RIVET, J.-P. 1987. Lattice gas hidrodynamics in two and three dimension. *Complex Systems*, 649–707.

GENEVAUX, O., HABIBI, A., AND DISCHLER, J.-M. 2003. Simulating fluid-solid interaction. In *Graphics Interface*.

GUENDELMAN, E., SELLE, A., LOSASSO, F., AND FEDKIW, R. 2005. Coupling water and smoke to thin deformable and rigid shells. In *SIGGRAPH '05*, 973–981.

HECKBERT, P. S. 1990. Adaptive radiosity textures for bidirectional ray tracing. In *SIGGRAPH '90*, 145–154.

HOUSE, D. H., AND BREEN, D. E., Eds. 2000. *Cloth Modeling and Animation*. A K Peters, Ltd.

IGLESIAS, A. 2004. Computer graphics for water modeling and rendering: a survey. *Future Gener. Comput. Syst. 20*, 8, 1355–1374.

JAMES, D. L., AND PAI, D. K. 1999. Artdefo: accurate real time deformable objects. In *SIGGRAPH '99*, ACM Press/Addison-Wesley Publishing Co., 65–72.

JENSEN, H. W., AND CHRISTENSEN, P. H. 1998. Efficient simulation of light transport in scences with participating media using photon maps. In *SIGGRAPH '98*, ACM, 311–320.

JUDICE, S. F., BARCELLOS, B., AND GIRALDI, G. 2008. A cellular automata framework for real time fluid animation. In *Proceedings of SBGames 08: Computing Track.*, 169–176.

KASS, M., AND MILLER, G. 1990. Rapid, stable fluid dynamics for computer graphics. In *SIGGRAPH '90*, 49–57.

KEISER, R., ADAMS, B., GASSER, D., BAZZI, P., DUTRE, P., AND GROSS, M. 2005. A unified lagrangian approach to solid-fluid animation. In *Point-Based Graphics, 2005. Eurographics/IEEE VGTC Symposium Proceedings*, 125–148.

LIU, G. R., AND LIU, M. B. 2003. *Smoothed particle hydrodynamics : a meshfree particle method*. World Scientific, New Jersey.

MOSEGAARD, J., AND SORENSEN, T. S. 2005. Gpu accelerated surgical simulators for complex morphology. In *VR '05: Proceedings of the 2005 IEEE Conference 2005 on Virtual Reality*, IEEE Computer Society, Washington, DC, USA, 147–154, 323.

MÜLLER, M., DORSEY, J., MCMILLAN, L., JAGNOW, R., AND CUTLER, B. 2002. Stable real-time deformations. In *SCA '02: Proc. of the 2002 ACM SIGGRAPH/Eurographics*, 49–54.

MÜLLER, M., SCHIRM, S., TESCHNER, M., HEIDELBERGER, B., AND GROSS, M. 2004. Interaction of fluids with deformable solids: Research articles. *Comput. Animat. Virtual Worlds 15*, 3-4, 159–171.

NEALEN, A., MULLER, M., KEISER, R., BOXERMAN, E., AND CARLSON, M. 2005. Physically based deformable models in computer graphics. In *Eurographics 2005, State of The Art Report (STAR)*.

PREMOZE, S., TASDIZEN, T., BIGLER, J., LEFOHN, A., AND WHITAKER, R. 2003. Particle-based simulation of fluids. *Computer Graphics Forum 22*, 3.

ROST, R. J. 2004. *OpenGL(R) Shading Language*. Addison-Wesley Professional, Boston, MA, USA.

ROTHMAN, D. H., AND ZALESKI, S. 1994. Lattice-gas models of phase separation: Interface, phase transition and multiphase flows. *Rev. Mod. Phys 66*, 1417–1479.

SIMPSON, G. 2004. A dynamic model to investigate coupling between erosion, deposition, and three-dimensional (thin-plate) deformation. *JOURNAL OF GEOPHYSICAL RESEARCH 19*.

SOLENTHALER, B., SCHLAFLI, J., AND PAJAROLA, R. 2007. A unified particle model for fluid-solid interactions. *Comput. Animat. Virtual Worlds 18*, 1, 69–82.

STAM, J. 1999. Stable fluids. In *Siggraph 1999*, Addison Wesley Longman, 121–128.

STAM, J. 2003. Flows on surfaces of arbitrary topology. In *SIGGRAPH '03*, 724–731.

TERZOPOULOS, D., AND WITKIN, A. 1988. Physically based models with rigid and deformable components. *IEEE Comput. Graph. Appl. 8*, 6, 41–51.

THÜREY, N., RÜDE, U., AND STAMMINGER, M. 2006. Animation of open water phenomena with coupled shallow water and free surface simulations. In *Proc. of the 2006 ACM SIGGRAPH/Eurographics Symp. on Comp. Anim. (SCA '06)*, 157–164.

VANGELDER, A., AND WILHELMS, J. 1997. Simulation of elastic membranes and soft tissue with triangulated spring meshes. Tech. rep.

VEACH, E., AND GUIBAS, L. J. 1997. Metropolis light transport. In *SIGGRAPH '97*, 65–76.

VOLINO, P., AND MAGNENAT-THALMANN, N. 2000. *Virtual Clothing: Theory and Practice*. Springer.

WEI, X., LI, W., MUELLER, K., AND KAUFMAN, A. 2004. The lattice-boltzmann method for simulating gaseous phenomena. *IEEE Trans. on Vis. and Comp. Graphics 10*, 2, 164–176.

WOO, M., DAVIS, AND SHERIDAN, M. B. 1999. *OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 1.2*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

YE, Z., MAIDMENT, D., AND MCKINNEY, D. 1996. Map-based surface and subsurface flow simulation models: An object-oriented and gis approach. Tech. rep., Center for Research in Water Resources, University of Texas at Austin, Tec. Report: http://www.crwr.utexas.edu/reports/pdf/1996/rpt96-5front.pdf.

YE, X. W., ZHAO, Y., FAN, Z., LI, W., QIU, F., YOAKUM-STOVER, S., AND KAUFMAN, A. 2004. Lattice-based flow field modeling. *IEEE Trans. on Vis. and Comp. Graphics 10*, 719–729.

ZHU, Y., AND BRIDSON, R. 2005. Animating sand as a fluid. *ACM Trans. Graph. 24*, 3, 965–972.