Real-time Interactive Animations of Liquid Surfaces With Lattice-boltzmann Engines

C. García Bauza, G. Boroni, M. Vénere and A. Clausse

Universidad Nacional del Centro and CICPBA-CONICET-CNEA, 7000 Tandil, Argentina.

Abstract. A physics-based graphic engine supporting interactive animations of free water surfaces at real-time is presented. The algorithm is based on a lattice-Boltzmann model of the shallow waters equations and the interaction between the surface and external objects is achieved by means of source terms. The engine is capable of produce scenes of ponds whose surface reacts to perturbations introduced by the user or controlled by the computer, like drizzle or the stirring of a finger.

Key words: Lattice Boltzmann, real-time animation, physics-based modeling, graphic engines, physical engines, shallow waters, rain, pond, fluid simulation.

INTRODUCTION

An efficient method to improve the realism in a virtual environment is to include algorithms based in physics laws. Physics will make objects move like they would do in the real world. A physical engine is a component of a program that computes how physical objects should move and interact with each other. Physical engines usually involve many research areas and often require high level of knowledge (Millington 2007).

Physics-based animation (PBA) has been recently recognized as one of the most important elements of computer graphics, largely because of the realism that it offers. Research in the field of PBA in computer graphics is concerned with finding new methods for the simulation of physical phenomena such as the dynamics of rigid bodies (*e.g.*, balls, feet, ships), deformable objects (*e.g.,* springs, fabric, skin) or fluid flow (smoke, clouds, surface waves). Physical engines of particles mechanics, spring-coupled objects, and collisions of rigid bodies, are increasingly available to the graphic community, for applications in film animation and the entertainment industry (Angst *et al*. 2009, Bao *et al*. 2007, Boeing *et al*. 2007, Harada *et al*. 2007 , Majkowska *et al*. 2007, Millington 2007, Tang *et al*. 2008, Thomaszewski *et al*. 2008, Treuille *et al*. 2006, Yu *et al*. 2009). The benefits of PBA should balance the computational cost associated to the number crunching effort required to transform the physical equations in good renderizations. Evidently the cost escalates with the number and complexity of objects and interactions in the virtual world, making it extremely challenging to render complex scenarios in real-time. Fortunately, a large amount of the effort in the simulation of complex systems is parallelizable, which can be exploited to improve the performance of PBA. It is also important to note that, in contrast to scientific computation where the main focus is on accuracy, the main issues for PBA are stability and speed, while keeping the visual appearance. Hence, a key target of the research on PBA is to come up with specialized methods, tailored to particular needs.

Rigid bodies were the first objects animated with physical engines. Likely, deforming bodies and cloth are increasingly used in games. On the other hand, fluid effects with physical engines are still rarely encountered. This is probably due to the computational cost of solving the underlying fluids equations, preventing practical applications in real-time scenarios. Still, since in our everyday lives water surrounds and constantly interacts with us, the animation and rendering of water is an important issue in computer graphics. Some of the potential applications of graphic simulations of fluid behavior are film making, animation design, texture synthesis, flight simulation, and scientific visualization.

Since the appearance of water is governed by the surface layer, much of the efforts were concentrated in simulating surface phenomena. Free surface flows are also important for a variety of applications, such as hydraulic engineering (Krafczyk *et al*. 2001), foaming (Korner *et al*. 2002) and bubble formation (Buwa *et al*.

Corresponding Author: C. García Bauza, Universidad Nacional del Centro and CICPBA-CONICET-CNEA, 7000 Tandil, Argentina.
E-mail: crgarcia@exa.unicen.edu.ar

2005). Early attempts of the graphics community in simulating water surfaces were not physically based and mostly focused on reduced model representations ranging from Fourier synthesis methods to parametric representations of the water surface (Masten *et al*. 1987, Schachter 1980, Tso and Barsky 1987). Fairly realistic wave scenery can be developed using these methods including the appearance of breaking waves, but ultimately they are all constrained by the sinusoidal modeling assumption present in each of them. These models are unable to deal with complex three-dimensional behaviors such as flow around objects and changing boundaries.

Physics-based model of the surface should capture the essence of water movement with the minimum computational cost, otherwise the effect is ruined for the viewer because either they are unrealistic or to slow to renderize in real time. The relevance of fluid simulations has been shown in numerous publications. Kass and Miller (1990) and Chen and Lobo (1995) were among the first to use computational fluid dynamics to calculate motions of fluids for computer graphics. Three dimensional simulations were lately combined with height-field techniques to reduce the computational costs (Irving *et al*. 2006). Free surface fluids can also be calculated by smoothed particle hydrodynamics (Muller M. *et al*. 2003, Yuksel *et al*. 2007), which does not require a fixed grid and computes the fluid properties through kernels defined on neighbourhoods of each particle. In order to track the surface the solver is often combined with appropriate boundary conditions and a level set (Foster and Fedkiw 2001). A good review of water rendering can be found in Iglesias (2004).

The straightforward method to calculate water surfaces with a physics-based model that can be applied to dynamic animation environments is the two dimensional approximation of the full 3D Navier-Stokes equations, called shallow water equations (SWE). Kass and Miller (1990) use a linearized form of the SWE to obtain a height field representation of the water surface. Chen and Lobo (1995) used a pressure defined SWE formulation to simulate fluids with moving obstacles. O'Brien and Hodgins (1995) used a height model combined with a particle system in order to simulate splashing liquids. Thon and Ghazanfarpour (2001) used a noise function for the vertical velocity in computing the horizontal velocity with SWE. Neyret and Praizelin (2001) proposed a simpler stream model using a two-dimensional Laplace equation for the bulk flow. A semi-Lagrangian treatment of the Navier-Stokes equations was introduced to the computer graphics community by Stam (1999) in order to allow the use of significantly larger time steps without hindering stability. Layton and Van de Panne (2002) and Thürey *et al*. (2007) continues with similar techniques applied to animating water waves. Foster and Fedkiw (2001) introduced a hybrid liquid volume model combining implicit surfaces and massless marker particles and the formulation of boundary conditions for moving objects in a liquid. Finally, Wang *et al*. (2007) proposes a framework for solving General Shallow Wave Equations (GSWE).

Interactive simulation is an old and important problem in computer graphics, and especially the interaction with virtual liquids is in great demand for its enormous potential. Unfortunately, the difficulty of physical simulation of fluids is an obstacle for real-time simulation. Most of the methods described above are simply not robust or fast enough to be employed in interactive virtual environments. An important step towards interactivity in graphical fluid dynamics was taken by Stam (1999) with the introduction of an unconditionally stable algorithm, allowing for long time steps and fast simulations. Subsequent works improved the calculation speed through hierarchical space decomposition (Losasso *et al*. 2004) and non-uniform meshes (Klinger *et al*. 2006), yielding impressively high resolution results, although the high computation cost prevents that method from being used in real-time contexts. Fourier (Stam 1999) and vortex methods (Park and Kim 2005), in turn, produce very fast performance in particular cases, but present difficulties handling boundaries and diffusion. Carlson *et al*. (2004) proposed a method for the coupling of rigid bodies with a fluid, by treating the rasterized rigid body velocities as if they were fluid. The latter was also done by Foster and Fedkiw (2001) for modeling slip boundary conditions.

The method proposed in the present article uses the lattice-Boltzmann method (LBM) (Thürey 2007, Thürey and Rüde 2004, Li *et al*. 2005). In contrast to solvers that directly compute solutions for the discretized Navier-Stokes equations, the LBM is a form of cellular automaton with a number of very simple local operations. When aggregated, these local operations produce an approximation to second order of the macroscopic fluid dynamics, which is sufficient for visualization purposes. Actually, LBM is a particular instance of coupled map lattices (CML), which are mappings of continuous dynamic state values to nodes on a lattice that interact with a set of other nodes in the lattice according to specified rules. CML were proposed by Kaneko (1993) for the purpose of studying spatio-temporal dynamics and chaos, and have been used extensively in the simulation of a variety of phenomena, including boiling Yanagita (1992), convection (Yanagita and Kaneko 1993), chemical reaction-diffusion (Kapral 1991), and sand ripples and dunes (Nishimori and Ouchi 1993). In the field of computer graphics, CML were introduced by Miyazaki *et al.* (2001) for the purpose of cloud animation.

Although the LBM has become an established method in fluid dynamics research, it is presently not very popular in animations of free surface flows. Thürey (2007) produced impressive animations of surface water using the LBM to solve the 3D Navier-Stokes equations and then tracking the movement of the free surface with level sets, applying special boundary conditions along the interface to calculate the appropriate values for velocity and pressure at the interface. However, these animations are not interactive, for the calculation of the 3D lattice is still too slow. Provided that faster implementations can be achieved, the properties of the LBM can be exploited to perform interactive fluid simulations.

The algorithm presented in this article is based on a lattice-Boltzmann model of the SWE (Zhou 2004). The equations are analogous to the 2D BGK-LBM scheme for the Navier-Stokes equations, which was extensively used in many applications (Sukop and Thorne 2006). In this article, an extension of Zhou's model is presented, which includes source terms to simulate the interaction of the free surface and the environment, like falling drops or moving solids.

II. The Lattice Boltzmann Model:

LBM is a class of algorithms that produce fast running simulations of fluid flows (Sukop and Thorne 2006). LBM operates on a regular grid usually designated by an identifier in the form *DdQq, d* being the number of dimensions of the lattice and *q* the number of velocity vectors that are used in a hidden mesoscopic representation of the fluid. Let us consider a regular lattice *L* defined in a *d*-dimensional space, composed by a set of nodes L_o and a set L_1 of links between two nodes (parameterized by a space step Δx). Given a node *x*, there exists a set *N(x)* of neighbouring nodes, including the node *x* itself. For the particular case *D2Q9* (Sukop and Thorne 2006) the set *N(x)* is given by the neighbourhood:

$$N(x) = \left\{ \vec{x} + \vec{v}_\alpha \Delta t, 0 \geq \alpha \geq 8 \right\} \tag{1}$$

and the following vector set (see fig. 1):

$$\vec{v}_o = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

$$\vec{v}_1 = \begin{pmatrix} v \\ 0 \end{pmatrix} \quad \vec{v}_2 = \begin{pmatrix} v \\ -v \end{pmatrix} \quad \vec{v}_3 = \begin{pmatrix} 0 \\ -v \end{pmatrix} \quad \vec{v}_4 = \begin{pmatrix} -v \\ -v \end{pmatrix} \tag{2}$$

$$\vec{v}_5 = \begin{pmatrix} -v \\ 0 \end{pmatrix} \quad \vec{v}_6 = \begin{pmatrix} -v \\ v \end{pmatrix} \quad \vec{v}_7 = \begin{pmatrix} 0 \\ v \end{pmatrix} \quad \vec{v}_8 = \begin{pmatrix} v \\ v \end{pmatrix}$$

where $v = \dfrac{\Delta x}{\Delta t}$ is a characteristic velocity given by the ratio between the space and the time steps associated

to each state change of the system.

LBM proposes that the mentioned discrete geometry is populated by mesoscopic "particles" whose state

is given by a particle density function $f_\alpha(\vec{x}, t)$ (not directly observable), representing the number of

particles in the node $\vec{x}$ at time *t* moving with velocity *vα* (which is treated as an internal variable). The

function $f_\alpha(\vec{x}, t)$ changes its value according to predetermined rulessimulating the mechanisms of transport,

collision and sources of the particles. The physical "observables" are macroscopic variables generated by

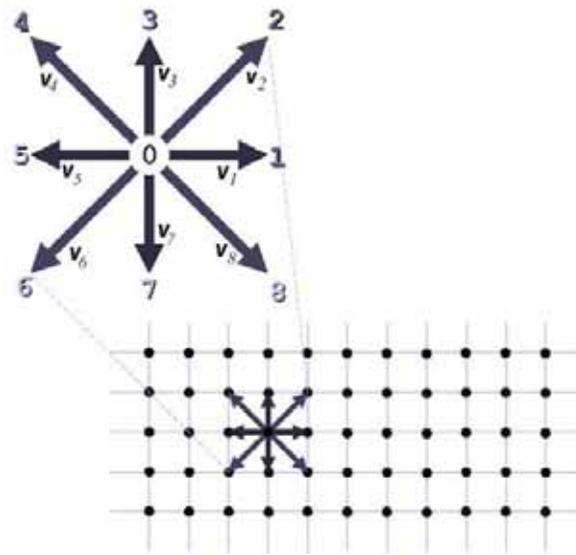moments of $f_\alpha(\vec{x}, t)$ respect to the internal variable $\vec{v}\alpha$, namely:

Fig. 1: Diagram of discrete velocities of Lattice Boltzmann in 2D

$$h(\bar{x},t) = \sum_{\alpha} f_{\alpha}(\bar{x},t) \qquad\qquad \text{Number of particles} \qquad\qquad (3)$$

$$\bar{u}(x,t) = \frac{\sum_{\alpha} \bar{v}_{\alpha} f_{\alpha}(\bar{x},t)}{h(\bar{x},t)} \qquad\qquad \text{Average velocity} \qquad\qquad (4)$$

The state of each cell change according to a scheme of explicit rules, that is (Chen and Doolen 1998):

$$f_{\alpha}(\bar{x} + \bar{v}_{\alpha}\Delta t, t + \Delta t) = f_{\alpha}(\bar{x},t) - \frac{1}{\tau}\left[f_{\alpha}(\bar{x},t) - f_{\alpha}^{eq}(\bar{x},t)\right] + S_{\alpha} \qquad\qquad (5)$$

where τ is a relaxation time that is related with the viscosity ν (Chen and Doolen 1998).

The equilibrium density function $f_{\alpha}^{eq}(\vec{x},t)$ plays the roll of a local velocity redistribution of the fluid

particles due to collisions, and it is calculated in terms of the macroscopic variables. This function is the key to the dynamics of the LBM model since its form determines the equations of motion that the macroscopic quantities will follow. To simulate free surfaces in small scenarios like ponds or rivers, the differential limit should be the shallow-waters equations:

$$\frac{\partial h}{\partial t} + \nabla \cdot (h\bar{u}) = 0 \qquad\qquad (6)$$

$$\frac{\partial (h\bar{u})}{\partial t} + \nabla \cdot \left(h\bar{u}\bar{u}^{T} + \frac{gh^2}{2}I\right) = 0 \qquad\qquad (7)$$

where h is the water height and g is the gravity acceleration. Zhou (2004) derived the following set of D2Q9 collision rules for shallow waters:

$$f_o^{eq} = h\left(1 - \frac{5gh}{6v^2} - \frac{2u^2}{3v^2}\right) \tag{8}$$

$$f_i^{eq} = w_i h\left(\frac{gh}{6v^2} + \frac{\vec{e}_i \cdot \vec{u}}{3v^2} + \frac{|\vec{e}_i \cdot \vec{u}|^2}{2v^4} - \frac{u^2}{6v^2}\right), \qquad i = 1,\ldots,8$$

where u is the modulus of the mean velocity, and:

$$w_i = \begin{cases} 1, & i = 1,3,5,7, \\ 1/4, & i = 2,4,6,8, \end{cases} \tag{9}$$

In order to complete the lattice rules, boundary conditions should be defined at the edges of the lattice. In the present case bounce-back conditions were applied. Hence, for each boundary cell, instead of copying the neighboring particle distribution function from a boundary cell during streaming, its own opposing distribution function is taken. This procedure enforces zero mean velocity at the boundary.

The interaction of the fluid with the user and the effect of rain drops are simulated by imposing pulsed localized sinks or sources of particles. Accordingly, for example, a rain drop falling at position $\vec{x}$ at time to will correspond to the addition of a constant quantity to every f_a at $(\vec{x}_o, t_o)$, that is:

$$S_\alpha(\vec{x},t) = \begin{cases} D_o & if\ (\vec{x},t) = (\vec{x}_o, t_o) \\ 0 & else \end{cases} \tag{10}$$

The larger the source D_o, the larger the drop. Moreover, since Do is uniform for all velocities, no momentum force is introduced by the drop, that is:

$$\Delta\vec{u} = \sum_\alpha S_\alpha \vec{v}_\alpha = 0 \tag{11}$$

Analogously, the introduction of a solid object of immersed height $H(\vec{x},t)$ inside the water surface is simulated by the subtraction of $H(\vec{x},t)$ to every f_a at $(\vec{x},t)$, that is:

$$S_\alpha(\vec{x},t) = -H(\vec{x},t) \tag{12}$$

These simple artifacts are fast enough for an interactive simulation, although in the case of the solid-fluid interaction cannot represent drag forces, hence being limited to slow movements.

III. Implementation:

The essence of LBM is simply a scheme of collision and transport. The junctions between cells correspond to a finite set of discrete velocities and the dynamic variables are the corresponding populations. The data structure used to represent the model is a regular grid of N × M cells, each containing the values $f_\alpha(\vec{x},t)$

This data structure is encapsulated in a class that provides the methods to access the macroscopic variables of each cell, namely $f_\alpha(\vec{x},t)$ and $\vec{u}(x,t)$, and additionally contains the methods to define properties, initial and boundary conditions.

The following methods implement the steps of the LBM described in section II:

• *Collide_stream():* Calculate and combine the advection and relaxation steps.

• *ApplyBounds():* Implement boundary conditions.
• *CalculateHU:* Calculate the macroscopic variables.
• *ComputeFeq:* Calculate the equilibrium function.

This object is created and called by a layer that controls the temporal evolution and interact with the user (Fig. 2). Following the common practice in physical engines (Seuglin and Rolin 2006, Boeing and Bräunl 2007) the implementation was structured as a Dynamic Link Library (DLL), providing easy access and interaction with computer graphic applications, independently of the programming language. The DLL contains a minimum set of basic operations preventing overloading the user interface and keeping flexible maintenance. Basically it provides the following self-described methods to actualize the data structure and to obtain the macroscopic data:
• *Initialize(tau, width, height, initValue)*
• *setBoundaryConditions(boundMode);*
• *addEvent(xHome,yHome,xEnd,yEnd,newValue)*
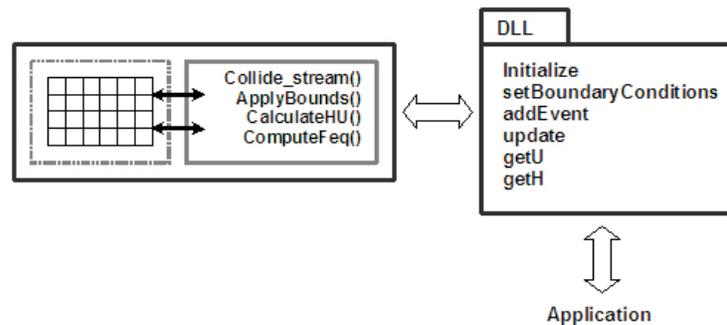• *update(deltaT)*
• *getU(i,j)*
• *getH(i,j)*

**Fig. 2:** Architecture of the LBM physical engine.

IV. Results:
The interactive LBM engine for shallow-water was tested in a square grid simulating the free surface of a small pond. Fig. 3 shows a sequence of the circular surface wave produced by the fall of a drop. The grid movement can be seen in the lower sequence of snaps and the complete animation with visual effects in the upper sequence. These 3D scenes, as all the ones presented in the article, were constructed with an application implemented over the graphic engine Impromptu (Garcia *et al.* 2008). The effects of reflection and refraction on the renderization of the water heights are constructed by means of a cubemap. The reflection and refraction vectors are calculated according to the position of the observer, and the final colors according to the Fresnel equations. Moreover, different shaders can be used in order to simulate the appearance of special liquids. The simulation was performed over a 100×100 grid, using a relaxation parameter (Eq. 5) $\tau = 0.77$. The velocity of the surface wave is around $\sqrt{gh}$, which is the analytical limit for shallow waters. As can be seen in Fig. 4 this limit is approached as the number of cells increases. The height of the waves is attenuated as they propagate following an exponential decay law $h_{max} \sim \exp\left(-\alpha \dfrac{r}{\Delta x}\right)$, where r is the propagation distance.

Fig. 5 shows the variation of the attenuation coefficient α with τ for different grid sizes. As expected, the attenuation increases with the relaxation parameter (*i.e.* with viscosity), and the reduction of the number of cells introduces numerical viscosity. Figure 6 shows a sequence of sna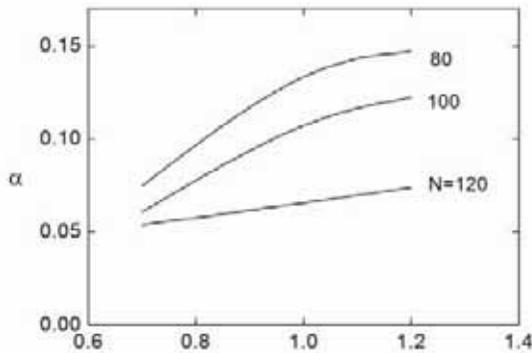ps of a rain simulation in a pond. The simulation proceeds introducing a series of random drops *Di* at random points and times $(\vec{x_i}, t_i)$. Different visual effects can be obtained varying the dropping rate and the relaxation parameter τ.
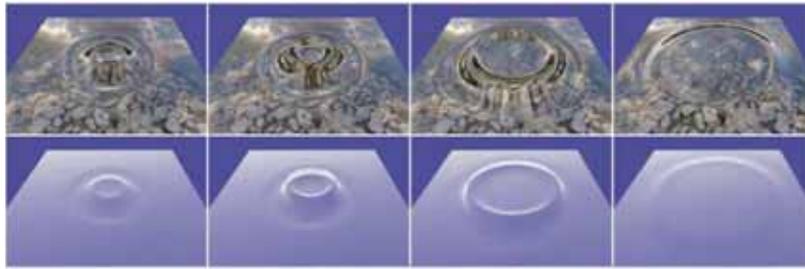
Fig. 3: Animation of the circular surface wave produced by the fall of a drop on a pond
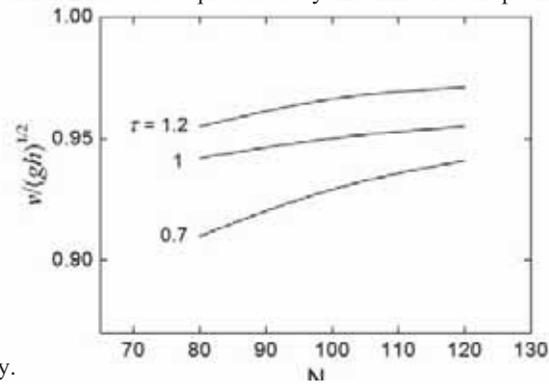

Fig. 4: Surface wave velocity.

Fig. 5: Dependence of the waves attenuation on the relaxation parameter.

Fig. 6: Simulation of rain falling on a pond.

It should be stressed that the LBM-shallow-waters algorithm is not unconditionally stable (Zhou 2004). In the continuous limit, the relaxation parameter τ should not be lower than 0.5, which corresponds to infinite Reynolds number. In finite grids this bound depends on the resolution and the amplitude of the perturbation. Fig. 7 shows the stability map in the plane (τ, N). The map was constructed for a case of a single drop on the center of a pond, with amplitude twice the stagnation level. It can be seen that the scheme is more stable as the resolution of the grid increases, which is reasonable since smaller cells can dissipate energy at higher
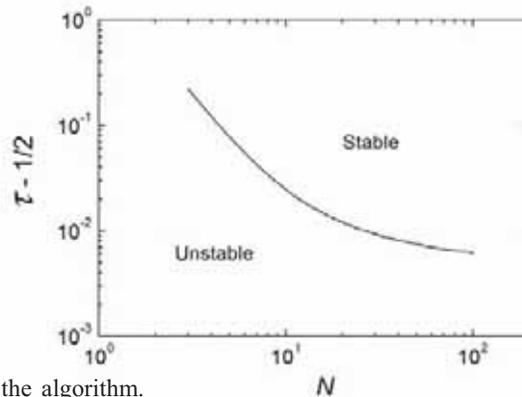
Fig. 7: Stability map of the algorithm.

frequencies. Another interesting effect can be produced by letting the user interact with the virtual fluid by means of an externally controlled source simulating a local perturbation of size δ describing a path $\vec{x}_o(t)$, that is, at every time *t*:

$$S_\alpha(\vec{x},t) = \begin{cases} -H & if \; |\vec{x} - \vec{x}_o(t)| < \delta \\ 0 & elsewhere \end{cases}$$

(13)

Figure 8 shows a sequence of snaps of a zigzag passage performed with the mouse on the virtual pond. Furthermore, using simple communications the application can be made completely interactive by enabling the introduction of the external events through touchscreen. In such case the projection perspective of the touch position should be obtained (*e.g.* using the OpenGL command "glUnproject") in order to initiate the associated perturbation event. Figure 9 shows the touchscreen application in action on a Tablet PC HP TX2500. The visual effect of interaction of objects with water surfaces were implemented following the two-path scheme described by Souza (2005). Finally, among the numerous applications that can be developed based in the interactive LBM, a special effect suitable for science-fiction features is the imprint of a transient footprint of an invisible object. Fig. 10 shows an example of a sequence of snaps of the trace left by an invisible hand in viscous water.

Fig. 8: Surface stirring of a virtual pond with the mouse.

In order to study the performance and scalability of the present interactive LBM algorithm, a series of simulations with different grid sizes were running on a standard PC with a 2.26 GHz Core2 processor and GeForce 8800 GT graphics card. Table 1 details the time consumed by the numerical calculation, grid updating and rendering. The total time is proportional to the number of cells. It can be seen that the numerics takes about 70% of the computing time, evidencing the importance of continuing the research and development of faster physics-based algorithms. Actually, the LB algorithm as was described in section II can be easily implemented on the hardware of the graphic card, which would surely improve the performance to permit simulations in larger grids (Li *et al*. 2003). Our preliminary tests in this direction showed reductions of an order of magnitude of the execution time. Nevertheless, since the quality of the images obtained on 100×100 grids is good, larger grids are not required from the visual point of view.
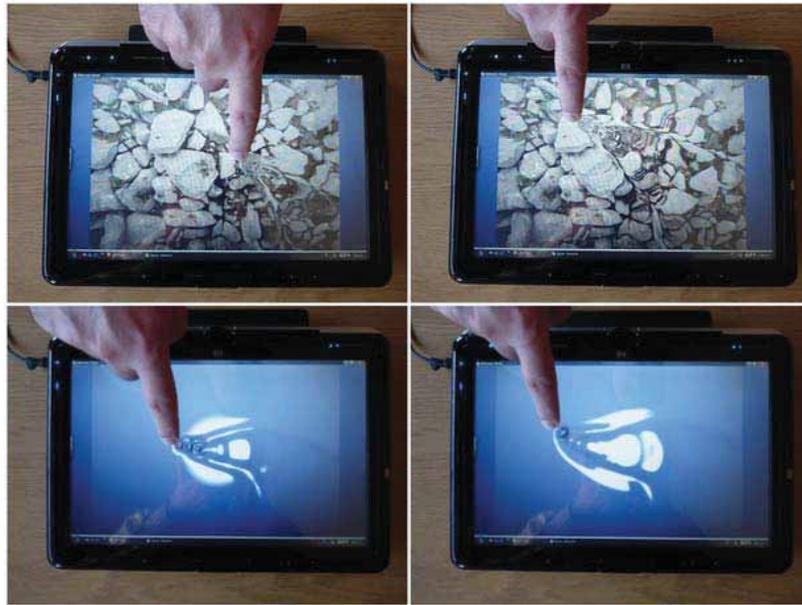
Fig. 9: Stirring of a virtual pond with the finger on a touchscreen.

Fig. 10: Special effect of the print left by an invisible hand in the surface of a viscous transparent liquid.

Table 1: Performance Test (time in ms).

GridSize (N×N)	50	80	100	120	150	180	200
LBM calculation	1.19	3.1	5	7.5	11.3	16.9	21.3
Grid updating	0.37	1	1.1	2.5	3.8	5.6	6.9
Rendering	0.1	0.2	0.3	0.4	0.6	0.7	0.9
Total Time	1.66	4.3	6.4	10.4	15.7	23.2	29.1
Frames per second	602	232	156	96	63	43	34

Conclusions:

A physical engine based on a lattice-Boltzmann model of the shallow-waters equations was presented. The tool was applied to produce interactive animations of free water surfaces at realtime. The user can interact with the virtual liquid by means of source terms that are designed to resemble perturbations of the free surface. The engine is capable of produce scenes of ponds whose surface reacts to perturbations introduced by the user or controlled by the computer, like drizzle or the stirring of a finger. The resulting algorithm is easy to implement, stable and fast enough to produce excellent real-time scenes and visual effects.

REFERENCES

Angst, R., N. Thürey, M. Botsch, M. Gross, 2009. Robust and Efficient Wave Simulations on Deforming Meshes, Computer Graphics Forum, 27: 1895-1900.

Bao, Z., J. Hong, J. Teran and R. Fedkiw, 2007. Fracturing rigid materials, IEEE Transactions on Visualization and Computer Graphics, 13: 370-378.

Boeing, A., and T. Bräunl, 2007. Evaluation of real-time physics simulation systems, GRAPHITE 2007, ACM 978-1-59593-912-8/07/0012, Perth, p. 281-288, Western Australia, December 1–4.

Buwa, V., D. Deo, V. Ranade, 2005. Eulerian-Lagrangian Simulations of Unsteady Gas-Liquid Flows in Bubble Columns, Int. J. Multiphase Flow, 32: 864-885.

Carlson, M., P. Mucha and G. Turk, 2004. Rigid fluid: animating the interplay between rigid bodies and fluid, ACM Transactions on Graphics, 23: 377-384.

Chen, J. and D. Lobo, 1995. Toward interactive-rate simulation of fluids with moving obstacles using Navier-Stokes equations. Graph. Models Image Process., 57: 107116.

Chen, S. and G.D. Doolen, 1998. Lattice Boltzmann Method for Fluid Flows. Annual Reviews Fluid Mechanics, 30: 329-364.

Foster, N. and R. Fedkiw, 2001. Practical animation of liquids, Proc. of ACM SIGGRPAH, 23-30.

García Bauza, C., M. Lazo, M. Vénere, 2008. Introduction of physical behavior in Graphic Engines. Mecánica Computacional (ISSN 1666-6070), 27: 3023-3039.

Harada, T., S. Koshizuka, Y. Kawaguchi, 2007. Smoothed particle hydrodynamics in complex shapes, Proc. of Spring Conference on Computer Graphics, pp: 26-28.

Iglesias, A., 2004. Computer graphics for water modeling and rendering: a survey, Future Generation Computer Systems, 20: 1355-1374.

Irving, G., E. Guendelman, F. Losasso and R. Fedkiw, 2006. Efficient simulation of large bodies of water by coupling two and three dimensional techniques, ACMTrans. Graph, 25: 805-811.

Kaneko, K., 1993. Theory and Applications of Coupled Map Lattices, Wiley, New York.

Kapral, R., 1991. Discrete models for chemically reacting systems, J. Math. Chem., 6: 113-163.

Klinger, B., B. Feldman, N. Chentanez and J. O'Brien, 2006. Fluid animation with dynamic meshes, ACM Transactions on Graphics, 25: 820-825.

Korner, C., M. Thies, R. Singer, 2002. Modeling of Metal Foaming with Lattice Boltzmann Automata, Advanced Engineering Materials, 4: 765-769.

Krafczyk, M., J. Tolke, E. Rank and M. Schulz, 2001. Two-Dimensional Simulation of Fluid-Structure Interaction using Lattice-Boltzmann Methods, Computers & Structures, 79: 2031-2037.

Layton, A., M. Van de Panne, 2002. A numerically efficient and stable algorithm for animating water waves. The Visual Computer, 18: 41-53.

Li, W., Z. Fan, X. Wei and A. Kaufman, 2005. Simulation with Complex Boundaries. "GPU Gems II", edited by Matt Pharr (Nvidia) and published by Addison Wesley, Chapter 47.

Li, W., X. Wei and A. Kaufman, 2003. Implementing Lattice Boltzmann Computation on Graphics Hardware, The Visual Computer, 19: 444-456.

Losasso, F., F. Gibou and R. Fedkiw, 2004. Simulating water and smoke with an octree data structure. ACM Trans. Graph, 23: 457-462.

Majkowska, A. and P. Faloutsos, 2007. Flipping with Physics: Motion Editing for Acrobatics, Eurographics ACM SIGGRAPH Symposium on Computer Animation, p. 35-44, San Diego, CA, USA, August 3-4.

Masten, G., P. Watterberg and I. Mareda, 1987. Fourier synthesis of ocean scenes. IEEE Computer Graphics and Application, 7: 16-23.

Millington, I., 2007. Game physics engine development, Elsevier.

Miyazaki, R., S. Yoshida, Y. Dobashi and T. Nishita, 2001. A method for modeling clouds based on atmospheric fluid dynamics, Proc. Ninth Pacific Conf. Comput. Graphics Appl., pp: 363-372, Tokyo, Oct. 16-18.

Muller, M., D. Charypar, M. Gross, 2003. Particle-based fluid simulation for interactive applications. Proc. of the ACM Siggraph/Eurographics Symposium on Computer Animation, 154-159.

Neyret, F. and N. Praizelin, 2001. Phenomenological simulation of brooks, Proc. Eurographics Workshop., 53-64.

Nishimori, H. and N. Ouichi, 1993. Formation of ripple patterns and dunes by wind-blown sand, Phys. Rev. Lett., 71: 197-200.

O'Brien, J., J. Hodgins, 1995. Dynamic simulation of splashing fluids, Proceedings of Computer Animation 95, p. 198-205, Geneva, Switzerland, April 19-21.

Park, S. and M. Kim, 2005. Vortex fluid for gaseous phenomena, Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pp: 261-270.

Schachter, B., 1980. Long crested wave models, Computer Graphics and Image Processing, 12: 187-201.

Seugling, A. and M. Rolin, 2006. Evaluation of Physics Engines and Implementation of a Physics Module in a 3d-Authoring Tool. Masters Thesis, Department of Computing Science, Umea University.

Souza, T., 2005. Generic refraction simulation. GPU Gems 2, 295-305. Addison-Wesley.

Stam, J., 1999. Stable fluids. In Siggraph 1999, Computer Graphics Proceedings, Rockwood A., (Ed.), Addison Wesley Longman, pp: 121-128.

Sukop, M. and D. Thorne, 2006. Lattice Boltzmann Modeling, Springer.

Tang, M., S. Curtis, S. Yoon, D. Manocha, 2008. Interactive Continuous Collision Detection between Deformable Models using Connectivity-Based Culling, Proceedings of the ACM Symposium on Solid and Physical Modeling, 25-36, Stony Brook, NY, USA, June 2-4.

Thomaszewski, B., A. Gumann, S. Pabst, W. Straßer, 2008. Magnets in motion, ACM Transactions on Graphics, 27(162): 1-9.

Thon, S. and D. Ghazanfarpour, 2001. A semi-physical model of running waters. Comput. Graph. Forum (Proc. Eurographics), 19: 53-59.

Thürey, N., U. Rüde, 2004. Free Surface Lattice-Boltzmann fluid simulations with and without level sets. Workshop on Vision, Modelling, and Visualization VMV, 199-208.

Thürey, N., 2007. Physically Based Animation of Free Surface Flows with the Lattice Boltzmann Method. PhD thesis, University of Erlangen-Nuremberg.

Thürey, N., M. Müller, S. Schirm, M. Gross, 2007. Real-time Breaking Waves for Shallow Water Equations. Proceedings of the Pacific Conference on Computer Graphics and Applications 2007; IEEE Computer Society.

Treuille, A., A. Lewis, Z. Popovic, 2006. Model reduction for real-time fluids, ACM Transactions on Graphics, 25: 826-834.

Tso, P., B. Barsky, 1987. Modeling and rendering waves: Wave-tracing using beta-splines and reflective and refractive texture mapping. ACM Transactions on Graphics, 6: 191-214.

Wang, H., G. Miller, G. Turk, 2007. Solving general Shallow Wave Equations on Surfaces. Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation, 229-238.

Yanagita, T., 1992. Coupled map lattice model for boiling, Physics Letters A, 165: 405-408.

Yanagita, T. and K. Kaneko, 1993. Coupled map lattice model for convection, Physics Let. A175, 415-420.

Yu, Q., F. Neyret, E. Bruneton and N. Holzschuch, 2009. Scalable real-time animation of rivers, Computer Graphics Forum, 28: 239-248.

Yuksel, C., D. House and J. Keyser, 2007. Wave Particles, Proceedings of SIGGRAPH 2007 (TOG), 26(3).

Zhou, J.G., 2004. Lattice Boltzmann methods for shallow water flows, Springer-Verlag.