# VISUALIZACIÓN INTERACTIVA DE SUPERFICIES DE FLUIDOS A PARTIR DE AUTÓMATAS LATTICE BOLTZMANN

**C. García Bauza[b], G. Boroni[a], M. Vénere[c] y A. Clausse[a,c]**

[a]*CONICET y Universidad Nacional del Centro, 7000, Tandil, Argentina*
[b]*CICPBA, Pcia. de Buenos Aires, Argentina*
[c]*CNEA, Libertador 8250, 1429, Ciudad de Buenos Aires, Argentina,*
*cristiangb@gmail.com, gboroni@gmail.com, venerem@exa.unicen.edu.ar, clausse@exa.unicen.edu.ar*

**Palabras clave**: Lattice Boltzmann, animación en tiempo real, motores gráficos basados en modelos físicos, Shallow waters.

**Resumen**. En este trabajo se presenta un motor gráfico basado en modelos físicos que soporta animaciones interactivas en tiempo real de superficies de aguas abiertas. La innovación presentada en este trabajo es un algoritmo basado en el modelo de Lattice Boltzmann aplicado sobre las ecuaciones de Shallow Waters y modelos físicos de interacción entre la superficie y objetos externos. Como resultado el motor gráfico es capaz de producir escenas de estanques o aguas abiertas cuya superficie reacciona a las perturbaciones introducidas por el usuario o controladas por la computadora, como por ejemplo la agitación provocada por un dedo o el efecto de la lluvia.

# 1 INTRODUCCION

Un método eficaz para mejorar el realismo en un entorno virtual es incluir algoritmos basados en leyes físicas. Estos algoritmos hacen que los objetos se muevan y comporten como lo harían en el mundo real. Un motor físico es un componente de un programa que calcula como los objetos físicos se mueven e interactúan con otros. En general estos motores aparecen en muchas áreas de investigación y su utilización requiere un alto nivel de conocimiento (Millington 2007).

La animación basada en el comportamiento físico (PBA, de *Physical Based Animation*) ha sido reconocida como uno de los elementos más importantes de Realidad Virtual, en gran parte debido al realismo que ofrece. La investigación de los PBA en computación gráfica se ha centrado en buscar nuevos métodos para la simulación de fenómenos físicos tales como la dinámica de los cuerpos rígidos (por ejemplo, pelotas, pies, buques), objetos deformables (por ejemplo, resortes, telas, piel) o el comportamiento de fluidos (humo, nubes, superficie de ondas).

Los motores físicos de mecánica de partículas, objetos acoplados por tensores (*Springs*), y la colisión de objetos rígidos son requeridos cada vez más en películas animadas y en la industria del entretenimiento (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). Al analizar los beneficios de la utilización de los PBA se debe tener en cuenta el costo computacional asociado al cálculo necesario para modelar las ecuaciones de física en una buena renderización. Evidentemente, este costo aumenta con el número y la complejidad de los objetos y las interacciones en el mundo virtual, por lo que es extremadamente difícil hacer escenarios complejos en tiempo real.

Los cuerpos rígidos fueron los primeros objetos animados por motores físicos. Posteriormente, se empezó a utilizar cuerpos deformables y telas, sobre todo en la implementación de juegos. Por otra parte, rara vez podemos encontrar efectos de fluidos en entornos virtuales, probablemente debido al costo computacional de la solución de las ecuaciones de fluidos, lo cual impide la simulación de escenarios en tiempo real. Aun así, la constante interacción del agua en la vida cotidiana hizo que la animación y renderización de la misma sea un tema de importancia en computación gráfica. Algunas aplicaciones destacables de la simulación gráfica de fluidos son el cine, el diseño de animación, simulación de vuelo y visualización científica.

## 1.1 Simulación de superficies de fluidos

La simulación de flujos de superficie es importante para una gran variedad de aplicaciones, tales como ingeniería hidráulica (Krafczyk et al. 2001), formación de espuma (Korner et al. 2002) y formación de burbujas (Buwa et al. 2005). Los primeros intentos de la comunidad de computación gráfica en la simulación de superficies de agua no se basaban en leyes físicas, y se centraba principalmente en representaciones reducidas de modelos yendo desde métodos sintéticos de Fourier a representaciones paramétricas de la superficie del agua (Masten et al. 1987, Schachter 1980, Tso y Barsky 1987). Pueden desarrollarse varios escenarios de características realistas con estos métodos, aunque finalmente todos están limitados por la hipótesis de modelado sinusoidal presente en cada uno de ellos. En general estos modelos no son capaces de hacer frente a comportamientos complejos en tres dimensiones, tales como el flujo alrededor de objetos y el cambio de las condiciones de contorno.

El modelo físico de la superficie debe captar la esencia del movimiento del agua con un costo computacional mínimo, de lo contrario el efecto se arruina para el observador ya sea

porque no es realista o porque es lenta la renderización en tiempo real. La importancia de las simulaciones de fluidos se ha demostrado en numerosas publicaciones. Kass y Miller (1990) y Chen y Lobo (1995) fueron los primeros en utilizar la dinámica computacional de fluidos para el cálculo de movimientos de fluidos para la computación gráfica. En el trabajo de Irving (Irving et al. 2006) se puede ver como las simulaciones tridimensionales pueden ser combinadas con técnicas de campos escalares de altura para reducir el costo computacional. En (Muller, M. et al. 2003) se propone una reducción del costo simulando fluidos de superficie libre mediante la técnica *Smoothed Particle Hydrodynamics* (SPH), la cual no requiere una grilla fija y permite calcular las propiedades del fluido a través de los puntos vecinos de cada partícula. Con el fin de seguir y modelar la superficie, el solver debe incorporar condiciones de contorno adecuadas y un conjunto de propiedades (Foster y Fedkiw 2001). Una buena reseña de renderización de agua se puede encontrar en (Iglesias 2004).

Un método simple para calcular un modelo físico de la superficie de agua, y que se puede aplicar a entornos de animación dinámica, es la aproximación en dos dimensiones de las ecuaciones completas de Navier-Stokes en 3D, llamado ecuaciones de Shallow Waters (SWE). En Kass y Miller (1990) se utiliza una forma lineal de SWE para obtener una representación del campo escalar de de alturas de la superficie del agua. En Chen y Lobo (1995) se utilizó la presión, definida sobre la formulación de SWE para simular fluidos con obstáculos en movimiento. O'Brien y Hodgins (1995) utilizaron un modelo de alturas combinado con un sistema de partículas para simular las salpicaduras de líquidos. En Thon y Ghazanfarpour (2001) se utiliza una función de ruido para la velocidad vertical en el cálculo de la velocidad horizontal con SWE. En Neyret y Praizelin (2001) se propuso un modelo simple usando una ecuación de Laplace en dos dimensiones para *bulk flow*. En Foster y Fedkiw (2001) se introdujo un modelo híbrido de volumen, que combina superficies y partículas de masa y la formulación de condiciones de contorno de los objetos que se mueven en un líquido.

Lamentablemente, la dificultad de la simulación física de fluidos es un obstáculo para la simulación en tiempo real. La mayoría de los métodos mencionados anteriormente no son lo suficientemente robustos o rápidos para ser empleados en entornos virtuales interactivos. Un paso importante hacia la interactividad en la dinámica de fluidos fue iniciado por Stam (1999) con la introducción de un algoritmo incondicionalmente estable, que permite intervalos de tiempo largos y simulaciones rápidas. En los trabajos posteriores se mejoró la velocidad de cálculo mediante la descomposición del espacio en jerarquías (Losasso et al. 2004) y mallas no uniformes (Klinger, al. 2006), generando impresionantes resultados de alta resolución, aunque el alto costo computacional impide que el método se utilice en tiempo real. A su vez, Fourier (Stam, 1999) y los métodos de vórtice (Park y Kim 2005) producen resultados muy rápidos para casos particulares, pero presentan dificultades en el manejo de contornos y difusión. En Carlson et al. (2004) se propone un método para el acoplamiento de los cuerpos rígidos con un fluido, descomponiendo el cuerpo rígido en una trama de velocidades como si fueran líquidos.

El método propuesto en este artículo utiliza el método de Lattice Boltzmann (LBM) siguiendo el trabajo de (Thurey de 2007.). A diferencia de los solvers que directamente calculan la solución de las ecuaciones de Navier-Stokes, el método LBM es una forma de autómata celular con una serie de operaciones locales muy simples. La suma de esas operaciones locales produce una aproximación a segundo orden de la dinámica de fluidos macroscópica, el cual es suficiente a los propósitos de visualización.

LBM es un caso particular de *coupled map lattices* (CML), en el cual se realizan asignaciones de valores de estado de la dinámica continua sobre los nodos de una red, que luego interactúa con un conjunto de otros nodos de acuerdo a determinadas reglas. CML fue propuesto por Kaneko (1993) con el fin de estudiar la dinámica espacio-temporal y caos, y ha sido

ampliamente utilizado en la simulación de una gran variedad de fenómenos, incluyendo problemas de ebullición Yanagita (1992), convección (Yanagita y Kaneko 1993), químicas de reacción-difusión (Kapral 1993), y el comportamiento de arena y dunas (Nishimori y Ouchi, 1993). En el campo de computación gráfica, CML fue introducido por Miyazaki et al. (2001) con el propósito de realizar animaciones de nubes.

Aunque el modelo LBM se ha convertido en un método característico en la investigación de dinámica de fluidos, en la actualidad no es popular en las animaciones de flujos de superficie libre. En el trabajo de Turey (2007) se pueden ver impresionantes animaciones de superficies de agua usando LBM para resolver las ecuaciones 3D de Navier-Stokes, y la aplicación de condiciones de contorno para calcular los valores adecuados de velocidad y presión. Sin embargo, estas animaciones no son interactivas, haciendo que el cálculo de la red en 3D sea aún demasiado lento.

El algoritmo presentado en este artículo se basa en el modelo de Lattice Boltzmann sobre las ecuaciones de SWE (Zhou 2004). Las ecuaciones son análogas a las 2D-BGK-LBM con las ecuaciones de Navier-Stokes, que fue ampliamente utilizado en muchas aplicaciones (Sukop y Thorne 2006). En este artículo, se presenta una extensión del modelo de Zhou, que incluye términos para simular la interacción de la superficie libre y el medio ambiente, como gotas que caen y el movimiento de objetos sólidos.

## 2 MODELO DE LATTICE BOLTZMANN

LBM es una clase de algoritmo que permite generar simulaciones rápidas de flujos de fluidos (Sukop y Thorne 2006). LBM opera en una malla o grilla regular generalmente designada por un identificador *DdQq*, donde *d* es el número de dimensiones de la red y *q* es el número de vectores de velocidad que se utilizan en una representación mesoscópica del fluido. Consideremos una red regular *L* definida en un espacio *d*-dimensional, compuesta por un conjunto de nodos $L_o$ y un conjunto de enlaces $L_1$ entre dos nodos (parametrizado por un paso de espacio $\Delta x$). Dado un nodo *x* cualquiera, existe un conjunto *N*(*x*) de nodos vecinos, incluido el nodo *x*. Para el caso particular *D2Q9* (Sukop y Thorne 2006), el conjunto *N*(*x*) está dado por

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

y el siguiente conjunto de vectores (ver Figura 1)

$$\begin{aligned} \vec{v}_o &= \begin{pmatrix} 0 \\ 0 \end{pmatrix} \\ \vec{v}_1 = \begin{pmatrix} v \\ 0 \end{pmatrix} \quad \vec{v}_2 &= \begin{pmatrix} v \\ -v \end{pmatrix} \quad \vec{v}_3 = \begin{pmatrix} 0 \\ -v \end{pmatrix} \quad \vec{v}_4 = \begin{pmatrix} -v \\ -v \end{pmatrix} \\ \vec{v}_5 = \begin{pmatrix} -v \\ 0 \end{pmatrix} \quad \vec{v}_6 &= \begin{pmatrix} -v \\ v \end{pmatrix} \quad \vec{v}_7 = \begin{pmatrix} 0 \\ v \end{pmatrix} \quad \vec{v}_8 = \begin{pmatrix} v \\ v \end{pmatrix} \end{aligned} \tag{2}$$

donde $v = \dfrac{\Delta x}{\Delta t}$ es una velocidad característica dada por la razón espacio tiempo asociada a cada estado de cambio del sistema.

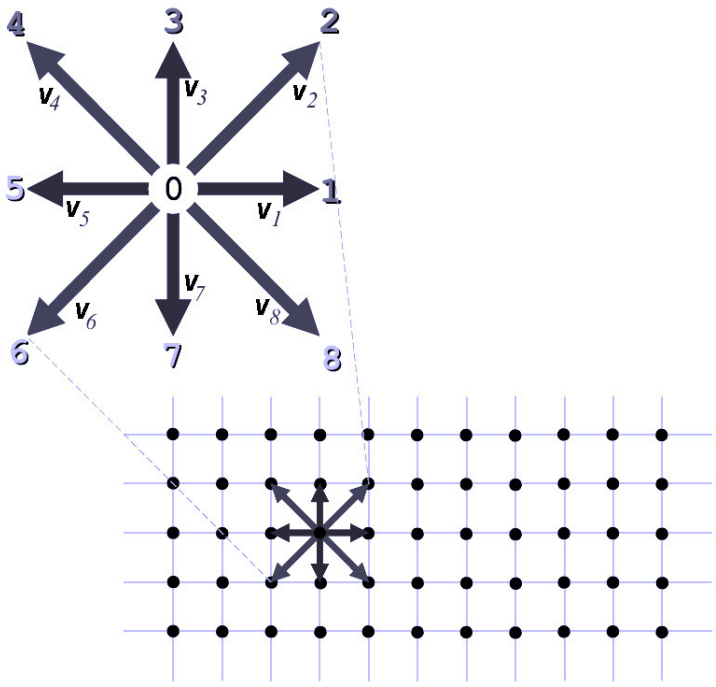


Figura 1. Diagrama de velocidades discretas para Lattice Boltzmann en 2D

El método LBM propone que la mencionada geometría discreta este poblada de partículas mesoscópicas, cuyo estado es representado por la función densidad de partículas $f_\alpha(\vec{x},t)$ (no observable directamente), que representa el número de partículas en el nodo $\vec{x}$ en el tiempo $t$ moviéndose con una velocidad $\vec{v}_\alpha$ (la cual es tratada como una variable interna). La función $f_\alpha(\vec{x},t)$ cambia su valor de acuerdo a reglas predeterminadas que simulan el mecanismo de transporte, colisión y fuentes de partículas. La física observable son variables microscópicas generadas por los momentos de $f_\alpha(\vec{x},t)$ respecto de la variable interna $\vec{v}_\alpha$:

Número de partículas

$$h(\vec{x},t) = \sum_\alpha f_\alpha(\vec{x},t) \tag{3}$$

Velocidad promedio

$$\vec{u}(x,t) = \frac{\sum_\alpha \vec{v}_\alpha f_\alpha(\vec{x},t)}{h(\vec{x},t)} \tag{4}$$

El estado de cada celda cambia de acuerdo a un esquema de reglas explícitas (Chen y Doolen 1998):

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

donde $\tau$ es una relajación del tiempo relacionada con la viscosidad $\nu$ (Chen y Doolen 1998).

La función densidad de equilibrio $f_\alpha^{eq}(\vec{x},t)$ se utiliza para redistribuir la velocidad local

del fluido de partículas debido a las colisiones, y se calcula en función de las variables macroscópicas. Para simular superficies libres en escenarios pequeños como estanques o ríos, el límite diferencial debe ser las ecuaciones de conservación de Shallow Waters:

$$\frac{\partial h}{\partial t} + \nabla \cdot \left(h\vec{u}\right) = 0 \tag{6}$$

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

donde *h* es la altura de agua y *g* es la aceleración de la gravedad. En el trabajo de Zhou (2004) se propuso para *D2Q9* el siguiente conjunto de reglas de colisión

$$\begin{aligned} f_o^{eq} &= h\left(1 - \frac{5gh}{6v^2} - \frac{2u^2}{3v^2}\right) \\ f_i^{eq} &= w_i h\left(\frac{gh}{6v^2} + \frac{\vec{e}_i \cdot \vec{u}}{3v^2} + \frac{\left|\vec{e}_i \cdot \vec{u}\right|^2}{2v^4} - \frac{u^2}{6v^2}\right), \qquad i = 1, \ldots, 8 \end{aligned} \tag{8}$$

donde *u* es el modulo de la velocidad media, y

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

Para completar las reglas del modelo se deben definir las condiciones de contorno sobre los bordes de la grilla. En este caso se aplican condiciones bounce-back, donde para cada celda de frontera en lugar de considerar la función de distribución de partículas durante la transmisión, se toma su propia función de distribución.

La interacción del fluido con el usuario y el efecto de las gotas de lluvia son simulados mediante la imposición puntual de sumideros o fuentes de partículas. En consecuencia, por ejemplo, una gota de lluvia que cae en la posición $\vec{x}_o$ en el tiempo $t_o$ corresponderá agregar una cantidad constante en todas las $f_\alpha$ en $(\vec{x}_o, t_o)$, es decir:

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

Claramente, cuanto mayor es la fuente *Do,* mayor es la caída. Además, ya que *Do* es uniforme para todas las velocidades, no hay fuerza introducida en la caída, es decir:

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

Da forma análoga, la introducción en el agua de un objeto sólido de altura inmersiva $H\left(\vec{x},t\right)$ es simulado por la substracción de $H\left(\vec{x},t\right)$ para todo $f_\alpha$ en $(\vec{x},t)$

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

Con estos artefactos simples se permiten simulaciones rápidas e interactivas, aunque en el caso de la interacción fluido-sólido no se puede representar las fuerzas de arrastre, y por lo tanto se está limitado a movimientos lentos.

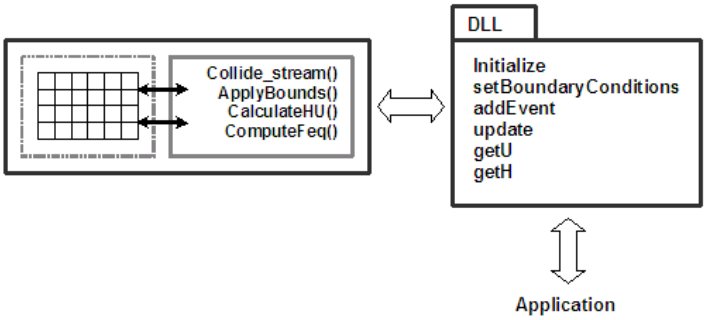


Figura 2. Arquitectura del motor físico LBM

## 3 IMPLEMENTACIÓN

La esencia de LBM es su presentación simple mediante un esquema de colisión y transporte. La unión entre las celdas correspondientes a un conjunto finito de velocidades discretas y las variables dinámicas representan las correspondientes poblaciones. La estructura de datos utilizada para representar el modelo es una malla regular de celdas, cada una conteniendo valores $f_\alpha(\vec{x},t)$. Esta estructura se encapsula en una clase que proporciona los métodos para acceder a las variables macroscópicas $h(\vec{x},t)$ y $u(\vec{x},t)$ de cada celda, y además contiene los métodos para definir las propiedades y condiciones iniciales y de contorno.

Los siguientes métodos implementan los pasos de LBM mencionados en la sección 2.

- *Collide_stream():* Calcula y combina los pasos de advección y relajación.
- *ApplyBounds():* Implementa las condiciones de contorno.
- *CalculateHU():* Calcula las variables microscópicas.
- *ComputeFeq():* Calcula la función de equilibrio.

Este objeto es creado y llamado por una capa que controla la evolución temporal e interacción con el usuario (Fig. 2). Siguiendo la práctica común en la utilización de motores físicos (Seuglin y Rolin 2006, Boeing y Bräunl 2007) la aplicación se estructuró como una librería dinámica (DLL), facilitando el acceso desde las aplicaciones de computación gráfica independientemente del lenguaje de programación. El archivo DLL contiene un conjunto de operaciones básicas:

- *Initialize(tau, width, height, initValue)*
- *setBoundaryConditions(boundMode);*
- *addEvent(xHome,yHome,xEnd,yEnd,newValue)*
- *update(deltaT)*
- *getU(i,j)*
- *getH(i,j)*

## 4 RESULTADOS

El motor interactivo de LBM para las ecuaciones de Shallow Waters ha sido probado en una grilla que simula la superficie libre de un pequeño estanque.

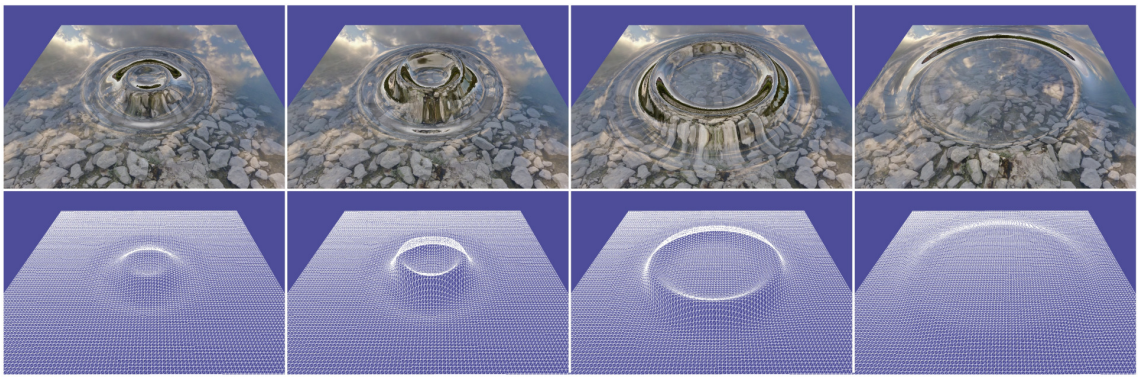

Figura 3. Animación de una onda circular causada por la caída de una gota en un estanque

La Figura 3 muestra una secuencia de ondas de superficie producida por la caída de una gota. El movimiento de la grilla se puede ver en las capturas de la parte inferior, y en la parte superior se visualiza la animación con los efectos visuales. Estas escenas 3D, como todas las presentadas en el artículo, se construyeron sobre una aplicación generada por el motor gráfico Impromptu (García et al. 2008). Los efectos de reflexión y refracción de la renderización de las alturas de agua se construyen por medio de un mapa cúbico de texturas (comúnmente conocido como *cubemap*). La reflexión y refracción de los vectores se calcula de acuerdo a la posición del observador, mientras que los colores se obtienen a partir de las ecuaciones de Fresnel.

La simulación fue realizada sobre una grilla de 100x100, usando un parámetro de relajación (Ec. 5) $\tau = 0.77$. La velocidad de onda de superficie es aproximadamente $\sqrt{gh}$ , la cual es el límite analítico para Shallow Waters (Fig. 4). La altura de onda se atenúa por la propagación de la onda, siguiendo el decaimiento exponencial $h_{\max} \sim \exp\left(-\alpha \frac{r}{\Delta x}\right)$, donde $r$ es la distancia de propagación.

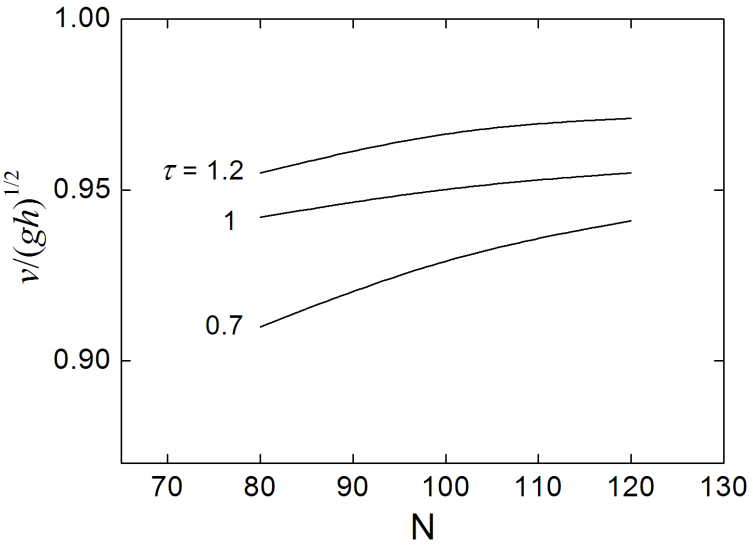


Figura 4. Velocidad de onda

En la Figura 5 se muestra la variación del coeficiente de atenuación $\alpha$ con respecto a $\tau$ para diferentes tamaños de grilla. Como es esperado, la atenuación se incrementa al aumentar el valor del parámetro de relajación (por ejemplo, mayor viscosidad), y por otra parte la reducción del número de celdas o tamaño de la grilla introduce viscosidad numérica. La Figura 6 muestra una secuencia de capturas de la simulación de una lluvia en un estanque. La simulación se realiza introduciendo una serie de gotas de azar $D_i$ en puntos y tiempos al azar $(\vec{x}_i, t_i)$. Se pueden obtener diferentes efectos visuales variando la tasa de caída y el parámetro de relajación $\tau$.

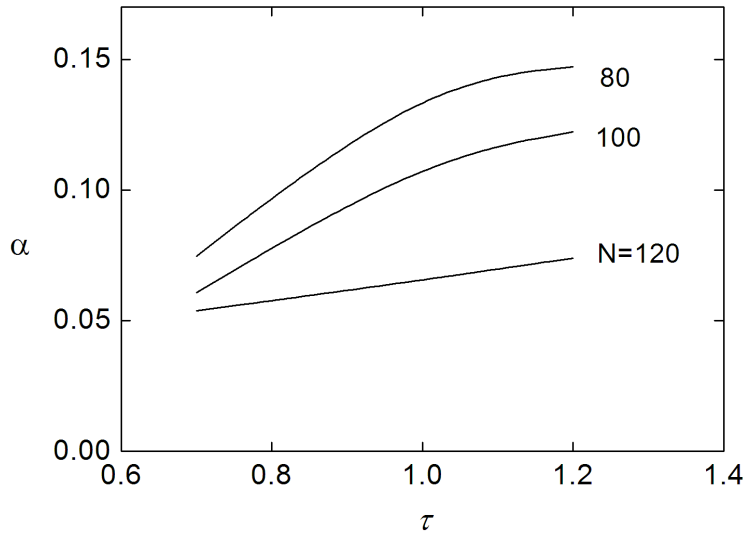


Figura 5. Dependencia de la atenuación de onda y el parámetro de relajación

Es interesante destacar que el algoritmo de LBM-Shallow-Waters no es incondicionalmente estable (Zhou 2004). En el límite continuo, el parámetro de relajación τ no debe ser inferior a 0.5, lo cual corresponde al número de Reynolds infinito. En grillas finitas este límite depende de la resolución y la amplitud de la perturbación. En la Figura 7 se muestra el mapa de estabilidad en el plano (τ, N), el cual fue generado a partir de la caída de una gota en el centro de un estanque. En dicha figura se puede ver que el algoritmo se vuelve más estable cuanto mayor es la resolución de la grilla, lo cual es razonable dado que las celdas más pequeñas permiten disipar la energía a frecuencias más altas.

Figura 6. Simulación de lluvia cayendo en un estanque

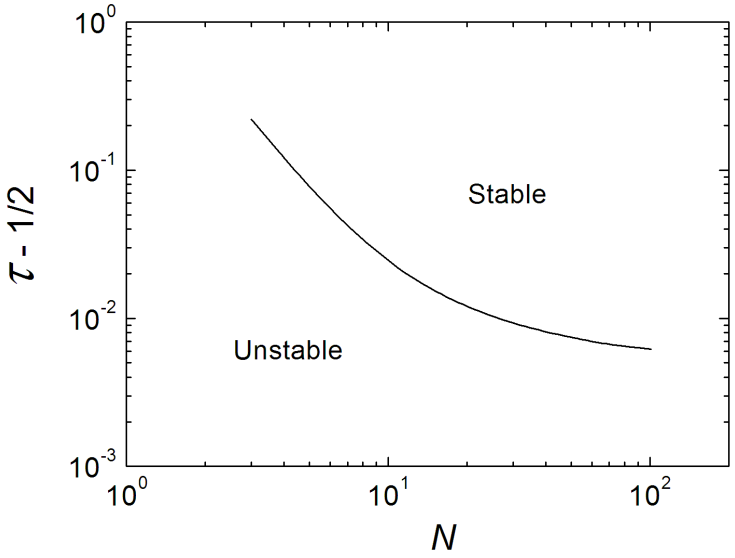


Figura 7. Mapa de estabilidad de algoritmo

Otro efecto interesante que fue simulado corresponde a la interacción del usuario con el fluido virtual por medio de una fuente externa controlada por una perturbación local de tamaño δ. Es decir, en cada tiempo *t* se aplica una fuente

$$S_\alpha(\vec{x},t)=\begin{cases}-H & if\ |\vec{x}-\vec{x}_o(t)|<\delta \\ 0 & elsewhere\end{cases} \tag{13}$$

La Figura 8 muestra una secuencia de capturas de una pasada en zigzag con el mouse sobre el estanque virtual. Claramente, al juntar la utilización de fuentes externas con interfaces interactivas simples se puede generar un ambiente completamente interactivo (por ejemplo, la introducción de eventos externos a través de una pantalla táctil).

En ese caso, se debe calcular la proyección en perspectiva de la posición del dedo en la pantalla táctil (utilizando por ejemplo, el comando de OpenGL "glUnproject") a fin de iniciar el evento de perturbación asociado. La Figura 9 muestra la aplicación corriendo en una pantalla táctil de un Tablet PC HP TX2500. El efecto visual de la interacción de los objetos con la superficie del agua se realiza siguiendo el esquema de dos trayectorias propuesto en el trabajo de Souza (Souza 2005). Por último, entre las numerosas aplicaciones que se pueden desarrollar con base en LBM, un efecto especial muy utilizado en animaciones de ciencia ficción es la huella de una presencia transitoria de un objeto invisible. La figura 10 muestra un ejemplo de una secuencia de capturas de la huella dejada por una mano invisible en un agua viscosa.

Con el fin de estudiar el rendimiento y la escalabilidad del algoritmo presentado, se realizaron una serie de simulaciones con diferentes tamaños de grilla, las cuales fueron ejecutadas sobre una PC estándar con un procesador de 2,26 GHz Core 2 Quad y una placa grafica GeForce 8800 GT.

| 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 |

Tabla 1. Resultados de Performance (tiempo en milisegundos)

La tabla 1 muestra el tiempo consumido por el cálculo numérico, la actualización de la grilla y la renderización. En dicha tabla se puede ver que el tiempo total es proporcional al número de celdas, y el tiempo de cálculo numérico consume alrededor del 70% del tiempo total, lo que evidencia la importancia de continuar la investigación para lograr algoritmos más rápidos. Siguiendo en esta línea, actualmente se están realizando implementaciones del algoritmo de LBM sobre el hardware de una placa gráfica (Rinaldi et al. 2008), lo cual permite simulaciones de grandes redes. Pruebas preliminares en esta dirección mostraron reducciones de un orden de magnitud del tiempo de ejecución. Sin embargo, dado que la calidad de las imágenes obtenidas en redes de 100×100 es excelente, no son necesarias redes más grandes desde el punto de vista visual.

Una versión ejecutable del estanque interactivo se puede descargar desde http://www.pladema.net/~cgarcia/resources/LBM.rar.

## 5 CONCLUSIONES

En este trabajo se presentó un motor físico sobre la base del modelo de Lattice Boltzmann para las ecuaciones de Shallow Waters. Esta herramienta fue aplicada para producir animaciones interactivas del agua en superficies libres en tiempo real. El usuario puede interactuar con el líquido virtual por medio de términos fuente diseñados para parecerse a las perturbaciones de la superficie libre. El motor es capaz de producir escenas de estanques cuya superficie reacciona a las perturbaciones introducidas por el usuario o controladas por la computadora. El algoritmo resultante es fácil de implementar, estable y lo suficientemente rápido como para producir excelentes escenas en tiempo real.

Figura 8. Perturbación de un estanque virtual con el Mouse

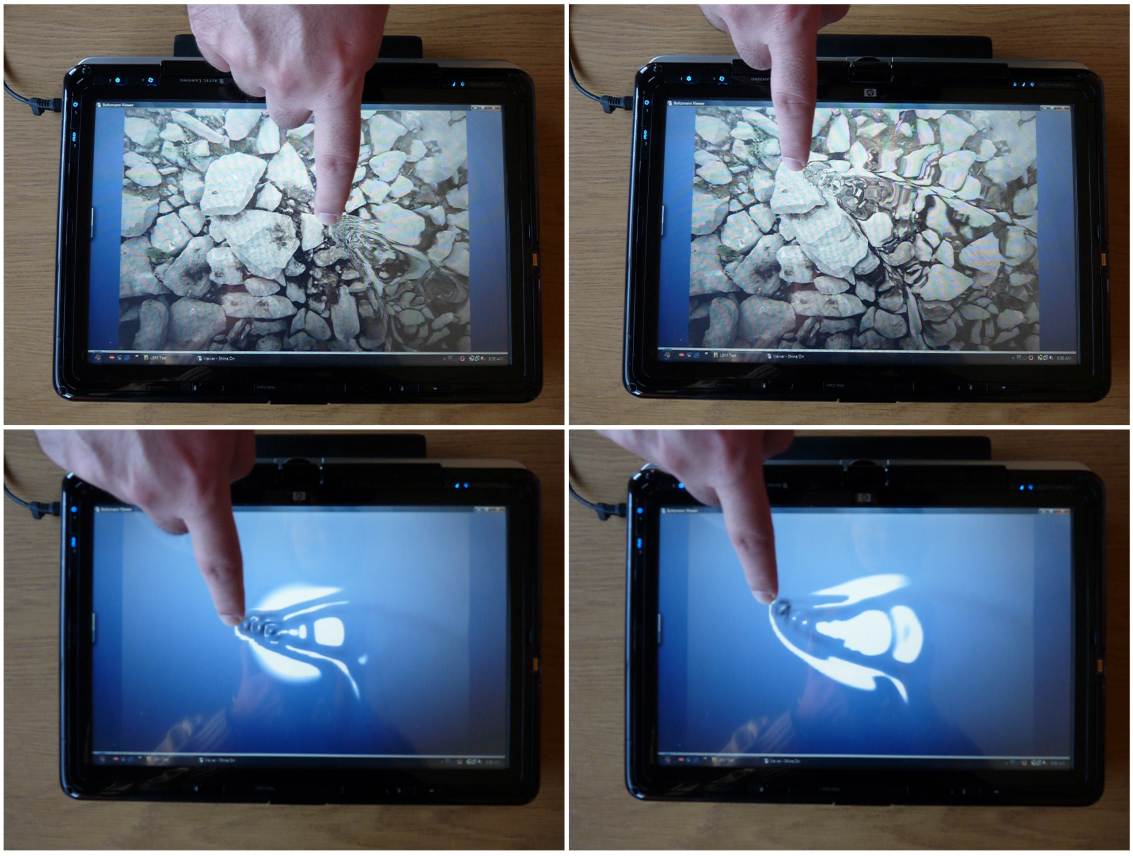

Figura 9. Perturbación de un estanque virtual con el dedo en una dispositivo táctil

Figura 10. Efecto especial de una impresión dejada por una mano invisible en la superficie de un líquido viscoso y transparente

## REFERENCIAS


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

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

Boeing A., and Bräunl T., 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, 2007.

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

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

Chen J. and Lobo D., Toward interactive-rate simulation of fluids with moving obstacles using Navier-Stokes equations. Graph. Models Image Process., 57, 107–116, 1995.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

2003.
Neyret F. and Praizelin N., Phenomenological simulation of brooks, Proc. Eurographics Workshop., 53–64, 2001.
Nishimori H. and Ouichi N., Formation of ripple patterns and dunes by wind-blown sand, Phys. Rev. Lett. 71, 197–200, 1993.
O'Brien J., Hodgins J., Dynamic simulation of splashing fluids, Proceedings of Computer Animation 95, p. 198-205, Geneva, Switzerland, April 19-21, 1995.
Park S. and Kim M., Vortex fluid for gaseous phenomena, Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation, p. 261-270, 2005.
Rinaldi P., García Bauza C., Clausse A. and Marcelo Vénere. "Paralelización de modelos de simulación de autómatas celulares sobre placas gráficas". Mecánica Computacional, Vol. XXVII, p. 2943-2957. ISSN 1666-6070, 2008.
Schachter B., Long crested wave models, Computer Graphics and Image Processing 12, 187–201, 1980.
Seugling A. and Rolin M. 2006. Evaluation of Physics Engines and Implementation of a Physics Module in a 3d-Authoring Tool. Proc. of SIGGRAPH 99, 121–128, 1999.
Souza T. Generic refraction simulation. GPU Gems 2, 295-305. Addison-Wesley, 2005.
Sukop M. and Thorne D., Lattice Boltzmann Modeling, Springer, 2006.
Tang M., Curtis S., Yoon S., Manocha D., Proceedings of the ACM Symposium on Solid and Physical Modeling, 25-36, Stony Brook, NY, USA, June 2-4, 2008.
Thomaszewski B., Gumann A., Pabst S., Straßer W., Magnets in motion, ACM Transactions on Graphics, 27, 162:1-9, 2008.
Thon S. and Ghazanfarpour D., A semi-physical model of running waters. Comput. Graph. Forum (Proc. Eurographics) 19, 53–59, 2001.
Thurey N., Physically Based Animation of Free Surface Flows with the Lattice Boltzmann Method. PhD thesis, University of Erlangen-Nuremberg, 2007.
Treuille A., Lewis A., Popovic Z., Model reduction for real-time fluids, ACM Transactions on Graphics. 25, 826-834, 2006.
Tso P., Barsky B., Modeling and rendering waves: Wave-tracing using beta-splines and reflective and refractive texture mapping. ACM Transactions on Graphics 6, 191–214, 1987.
Yanagita T., Coupled map lattice model for boiling, Physics Letters A, 165, 405-408, 1992.
Yanagita T. and Kaneko K.; Coupled map lattice model for convection, Physics Let. A175, 415-420, 1993.
Yu Q., Neyret F., Bruneton E. and Holzschuch N., Scalable real-time animation of rivers, Computer Graphics Forum, 28, 239-248, 2009.
Zhou J. G., Lattice Boltzmann methods for shallow water flows, Springer-Verlag, 2004.