Asociación Argentina
de Mecánica Computacional

Mecánica Computacional Vol XXIX, págs. 3431-3440 (artículo completo)
Eduardo Dvorkin, Marcela Goldschmit, Mario Storti (Eds.)
Buenos Aires, Argentina, 15-18 Noviembre 2010

# BORDES DINÁMICOS SOBRE EL MÉTODO DE LATTICE BOLTZMANN PARA SIMULACIÓN DE SUPERFICIES DE FLUIDOS EN TIEMPO REAL

**C. García Bauza, M. Lazo, G. Boroni, D. Dalponte, y A. Clausse**

CNEA-CONICET-CICPBA y Universidad Nacional del Centro, 7000 Tandil, Argentina
cristiangb@gmail.com

**Resumen.** Un problema típico en flujos en geometrías complejas son los bordes móviles, como por ejemplo en la simulación de flujo de sangre en arterias o el movimiento de un barco. En este trabajo se presenta una implementación especial en un autómata de Lattice Boltzmann 2D de aguas superficiales con bordes y obstáculos internos móviles. El esquema permite simular en forma flexible escenarios de ondas superficiales producidas por embarcaciones, reflejos en puentes, y otras situaciones de interés en animación.

**Palabras clave:** Lattice Boltzmann, condiciones de contorno, aguas superficiales, bordes móviles.

## 1 INTRODUCCIÓN

En los últimos años, las aplicaciones de computación gráfica en tiempo real han comenzado a incluir además de simulación de partículas, otros procesos más complejos como colisiones elastoplásticas y fluidos. Esta última, ha tenido un gran avance en el último tiempo, particularmente en el aspecto visual del fluido (Iwasaki et al. 2003, Mitchell 2005, Ernst 2005). En este sentido, el realismo visual depende de una buena representación no sólo del fluido sino también de las interacciones fluido-estructura, especialmente si la estructura se mueve. Lamentablemente, la dificultad de la detección de puntos de contacto entre los objetos y el fluido, sigue siendo un obstáculo para la simulación en tiempo real.

Un paso importante hacia la interacción de objetos sólidos y superficies de fluido fue iniciado por Takahashi (2002), que presentó un método simple donde los objetos son voxelizados para calcular las condiciones de límites sobre el fluido. Génevaux (2003) propuso un método usando puntos de referencia y simulación elastoplástica mediante cálculo de masas y uniones elásticas. Carlson (2004) presentó un método para el acoplamiento de los cuerpos rígidos con líquidos, tratando el cuerpo rígido como otro líquido inmiscible. Foster y Fedkiw (2001) aplicó este método para modelar condiciones de deslizamiento. Guendelman (2003) planteó un enfoque alternativo para incluir grillas representadas por octrees, utilizando objetos sólidos pequeños y una dinámica de sólidos arbitraria.

Cuando los bordes son complicados se suele generar ruido por la discretización, que debe ser eliminado generalmente con un cálculo auxiliar para ajustar las diferencias de presión, lo cual aumenta el costo computacional. Otra solución es usar funciones de ajuste para representar los objetos sólidos, aunque las soluciones están acotadas al número de primitivas disponibles. En este trabajo se presenta un algoritmo para modelar

contornos sólidos de forma dinámica en autómatas de Lattice Boltzmann, que permite simular la interacción entre sólidos y superficies de fluidos.

## 2 MODELO DE LATTICE BOLTZMANN (LBM)

El LBM es un autómata celular que discretiza el espacio como una grilla regular de celdas, cuyo estado está representado por una población de partículas mesoscópicas. El concepto básico es un modelo cinético con variable interna discreta representando un conjunto finito de velocidades), cuyos promedios macroscópicos cumplen con las ecuaciones de balance de un campo transportado (Chen y Doolen 1998). El estado actual de un paso de simulación depende sólo de su condición previa —*i.e.* LBM es un caso particular de autómata celular— lo que permite implementaciones rápidas en sistemas distribuidos o de computación paralela.

Consideremos una red regular $L$ definida en un espacio $d$-dimensional, compuesta por un conjunto de nodos $L_0$ 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 de 2 dimensiones espaciales y 9 velocidades, *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:

$$\begin{aligned} \vec{v}_0 &= \begin{pmatrix} 0 \\ 0 \end{pmatrix} \\ \vec{v}_1 &= \begin{pmatrix} v \\ 0 \end{pmatrix} \quad \vec{v}_2 = \begin{pmatrix} v \\ -v \end{pmatrix} \quad \vec{v}_3 = \begin{pmatrix} 0 \\ -v \end{pmatrix} \quad \vec{v}_4 = \begin{pmatrix} -v \\ -v \end{pmatrix} \\ \vec{v}_5 &= \begin{pmatrix} -v \\ 0 \end{pmatrix} \quad \vec{v}_6 = \begin{pmatrix} -v \\ v \end{pmatrix} \quad \vec{v}_7 = \begin{pmatrix} 0 \\ v \end{pmatrix} \quad \vec{v}_8 = \begin{pmatrix} v \\ v \end{pmatrix} \end{aligned} \tag{2}$$

donde $v = \dfrac{\Delta x}{\Delta t}$ es la velocidad característica de la grilla.

El estado de las partículas está dado por poblaciones $f_\alpha(x,t)$ que representan la cantidad de partículas en la celda $x$ en el tiempo $t$ moviéndose con velocidad $\vec{v}_\alpha$. Los observables físicos son variables macroscópicas que se generan a partir de los momentos de $f_\alpha(x,t)$ respecto de la variable $\vec{v}_\alpha$, a saber:

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):

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

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 de manera que se conserve la masa, la cantidad de movimiento y la energía total en cada celda. Zhou (2004) propuso la siguiente regla de colisión para *D2Q9* para simular una superficie líquida con un campo de alturas $h(\vec{x},t)$ y un campo de velocidades $\vec{u}(\vec{x},t)$:

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

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

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

En el límite diferencial estas reglas aseguran que *h* y *u* obedecen las ecuaciones de aguas superficiales:

$$\frac{\partial h}{\partial t} + \nabla\cdot(h\vec{u}) = 0 \qquad (8)$$

$$\frac{\partial(h\vec{u})}{\partial t} + \nabla\cdot\left(h\vec{u}\vec{u}^T + \frac{gh^2}{2}I\right) = 0 \qquad (9)$$

donde *g* es la aceleración de la gravedad.

Para poder ejecutar la Ec. (5) en un dominio finito es necesario definir condiciones de contorno, ya que en las celdas adyacentes a los bordes hay celdas vecinas fuera del dominio. En general, las condiciones en los bordes pueden ser abiertas o cerradas. Las condiciones abiertas se utilizan normalmente en la entrada y salida del canal, mientras que las condiciones cerradas se imponen en las paredes laterales al flujo o en las celdas que forman el borde de obstáculos. En los trabajos de Zhou (2004) y Zhaoli *et al.* (2002) puede encontrarse un tratamiento exhaustivo de bordes para LBM. En particular, cuando hay obstáculos o cuerpos sumergidos se pueden dar dos posibilidades: sin deslizamiento y con deslizamiento.

La condición sin deslizamiento impone velocidad neta cero en todas las direcciones en una superficie sólida dada. La implementación más común es el esquema

*on-grid bounce-back* (Succi 2001) en la cual el límite del dominio yace exactamente sobre el borde de la grilla. La regla básica resultante es muy simple y establece que una partícula que alcanza el borde es revertida inmediatamente hacia el fluido. El esquema es mostrado en la Figura 1.

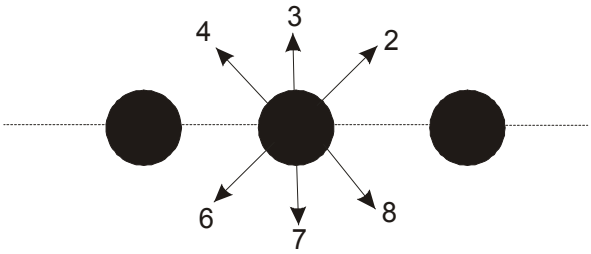


Figura 1: Condiciones de contorno on-grid bounce back.

Las partículas correspondientes a las poblaciones $f_2$, $f_3$ y $f_4$ rebotan en el borde, quedando:

$$f_8 = f_4, \qquad f_7 = f_3, \qquad f_6 = f_2 \tag{10}$$

De esta manera se asegura que la suma de los momentos de las partículas cercanas a la pared es cero.

Si bien el esquema *on-grid bounce back* es muy simple y estable, el inconveniente es que es una aproximación de primer orden, lo cual degrada la precisión de segundo orden que tiene el esquema LBM. Una modificación que conserva la precisión de segundo orden es el esquema *on-grid bounce-back modificado* (Ashyraliyev 2004) según el cual para el borde norte se tiene:

$$f_8 = f_4 - \frac{1}{2}(f_1 - f_5), \qquad f_7 = f_3, \qquad f_6 = f_2 + \frac{1}{2}(f_1 - f_5) \tag{11}$$

Esto asegura que la velocidad del fluido en el nodo sea cero, es decir:

$$\rho u = f_1 + f_2 + f_8 - f_5 - f_4 - f_6 = 0, \qquad \rho v = f_2 + f_3 + f_4 - f_6 - f_7 - f_8 = 0 \tag{12}$$

De la misma manera se procede para un nodo de la pared sur:

$$f_2 = f_6 - \frac{1}{2}(f_1 - f_5), \qquad f_3 = f_7, \qquad f_4 = f_8 + \frac{1}{2}(f_1 - f_5) \tag{13}$$

Las condiciones de contorno con deslizamiento (Zhou 2004, Succi 2001) se aplican para simular bordes con deslizamiento tangencial, por ejemplo la capa límite. La implementación *on-grid* de esta condición es la más simple y consiste en una dispersión elástica sobre la pared. La misma establece que una partícula dirigida hacia el borde del canal es reflejada hacia el fluido (Figura 2). De esta manera las incógnitas $f_2$, $f_3$ y $f_4$ después de la advección quedan determinadas por:

$$f_2 = f_8, \qquad f_3 = f_7, \qquad f_4 = f_6 \tag{14}$$

Esto asegura que el momento normal al borde sea nulo, esto es:

$$\rho v = f_2 + f_3 + f_4 - f_6 - f_7 - f_8 = 0 \tag{15}$$

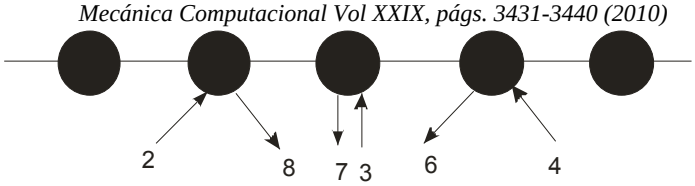


Figura 2: Condiciones de contorno on-grid free-slip.

## 2 CONDICIONES DE CONTORNO DINÁMICAS

La propuesta que se presenta en este trabajo para simular bordes dinámicos aprovecha las propiedades del esquema explícito de LBM, que permiten separar las reglas de advección de las condiciones de contorno. Para ello se descompone cada paso de tiempo en las siguientes rutinas:

1) Calcular la población de equilibrio (Ec. 6)
2) Advectar (Ec. 5) usando variables *dummy* para las referencias a poblaciones fuera del dominio de cálculo en celdas adyacentes a los bordes u obstáculos (*e.g.* asignar un valor arbitrario).
3) Calcular las nuevas posiciones de los bordes u obstáculos, ya sea por input externo del usuario o por medio de alguna ecuación de movimiento.
4) Identificar y clasificar las celdas adyacentes a los bordes u obstáculos.
5) Recalcular las poblaciones de las celdas adyacentes a los bordes u obstáculos de acuerdo al tipo de condición de contorno asignado en 4.
6) Calcular los promedios macroscópicos.

La tarea 4 es central, porque flexibiliza la representación de la interacción del fluido con estructuras sólidas. Hay ocho posibles tipos de adyacencias a una celda sólida: norte (N), sur (S), este (E), oeste (W), y las cuatro diagonales NE, NW, SE y SW. A su vez, puede elegirse dos tipos de condiciones de contorno: con y sin deslizamiento. De esta manera habrá 16 tipos de celdas adyacentes a bordes u obstáculos. Para cada uno de estos tipos se implementa un método correspondiente a la actualización de las poblaciones que en la advección requieren valores fuera del dominio. Por ejemplo, una celda tipo S sin deslizamiento debe actualizar las poblaciones $f_6$, $f_7$ y $f_8$ de acuerdo a las Ecs. 12. Nótese que una celda adyacente a un sólido puede tener más de un tipo de adyacencia, en cuyo caso simplemente se ejecutan todos los métodos correspondientes.

```
if (x>=1)&&(x<=Lx)&&(y>=1)&&(y<=Ly) {
        if ((x-1)>=1)&&((x-1)<=Lx)&&((y-1)>=1)&&((y-1)<=Ly)
                addContour(x-1,y-1, lower_left);
        if ((x+1)>=1)&&((x+1)<=Lx)&&((y+1)>=1)&&((y+1)<=Ly)
                addContour(x+1,y+1, upper_right);
        if ((x-1)>=1)&&((x-1)<=Lx)&&((y+1)>=1)&&((y+1)<=Ly)
                addContour(x-1,y+1, upper_left);
        if ((x+1)>=1)&&((x+1)<=Lx)&&((y-1)>=1)&&((y-1)<=Ly)
                addContour(x+1,y-1, lower_right);
        if ((x)>=1)&&((x)<=Lx)&&((y-1)>=1)&&((y-1)<=Ly)
                addContour(x,y-1, upper);
        if ((x)>=1)&&((x)<=Lx)&&((y+1)>=1)&&((y+1)<=Ly)
                addContour(x,y+1, lower);
        if ((x+1)>=1)&&((x+1)<=Lx)&&((y)>=1)&&((y)<=Ly)
                addContour(x+1,y, left);
        if ((x-1)>=1)&&((x-1)<=Lx)&&((y)>=1)&&((y)<=Ly)
                addContour(x-1,y, right);
};
```

Cuadro 1: Clasificación del tipo de adyacencia de las celdas que tocan bordes sólidos.

En el cuadro 1 se muestra un pseudocódigo para la tarea 4 para una celda obstáculo ubicada en una posición (x, y), de una grilla de tamaño Lx x Ly. El método "addContour(pos_x, pos_y, tipo_borde)" se utiliza para agregar a la lista "lista_contorno" un nodo con la posición y el tipo de adyacencia de una celda dada. En el cuadro 2 se presenta una implementación de las condiciones de contorno sin deslizamiento.

```
void Noslip();
…
for (i=0, i<count(lista_contorno);i++){
x=lista_contorno[i].x;
y=lista_contorno[i].y;
case lista_contorno[i].tipo
lower_left:     ftemp[2][x][y]=ftemp[6][x][y];
lower_right:    ftemp[4][x][y]=ftemp[8][x][y];
upper_left:     ftemp[8][x][y]=ftemp[4][x][y];
upper_right:    ftemp[6][x][y]=ftemp[2][x][y];
lower:          ftemp[2][x][y]=ftemp[6][x][y];
                ftemp[3][x][y]=ftemp[7][x][y];
                ftemp[4][x][y]=ftemp[8][x][y];
upper:          ftemp[6][x][y]=ftemp[2][x][y];
                ftemp[7][x][y]=ftemp[3][x][y];
                ftemp[8][x][y]=ftemp[4][x][y];
left:           ftemp[8][x][y]=ftemp[4][x][y];
                ftemp[1][x][y]=ftemp[5][x][y];
                ftemp[2][x][y]=ftemp[6][x][y];
right:          ftemp[4][x][y]=ftemp[8][x][y];
                ftemp[5][x][y]=ftemp[1][x][y];
                ftemp[6][x][y]=ftemp[2][x][y];
};
    …
```

Cuadro 2. Implementación de las condiciones de contorno sin deslizamiento.

## 2.1 Cálculo de los bordes dinámicos

La tarea 3 de asignación de los bordes sólidos en cada paso de tiempo se implementó usando información de un motor gráfico OpenGL. Cada objeto está representado con una malla 3D (*e.g.* un barco, un puente). Para identificar los puntos (x,y) del dominio ocupados por cada objeto se usa una máscara con la proyección vertical de la malla sobre la superficie del fluido, es decir una subgrilla binaria como muestra la Fig. 3. La técnica propuesta es similar a las utilizadas en Kim et al. (2006) y Crane et al. (2007) y utiliza la capacidad del buffer de profundidad de las placas gráficas (Z-Buffer). La máscara se genera simplemente con la proyección vertical del objeto (*e.g.* barco) como se muestra la Figura 4.

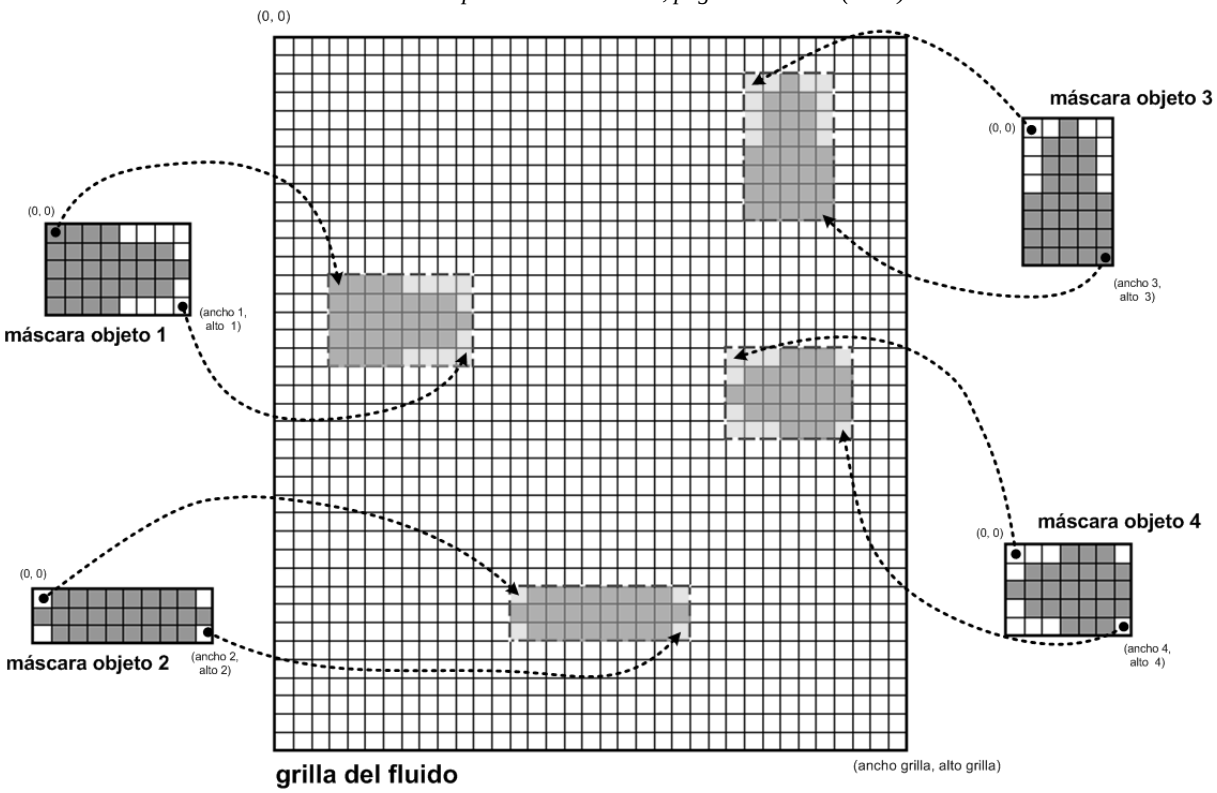


Figura 3: Correspondencia entre las máscaras generadas y la grilla del fluido.


Figura 4: Configuración de cámara y proyección

## 3 RESULTADOS

Se presentan a continuación tres casos utilizando los algoritmos correspondientes al cuadro 1 y 2, los cuales fueron implementados sobre una herramienta de simulación interactiva (García Bauza et al 2010).

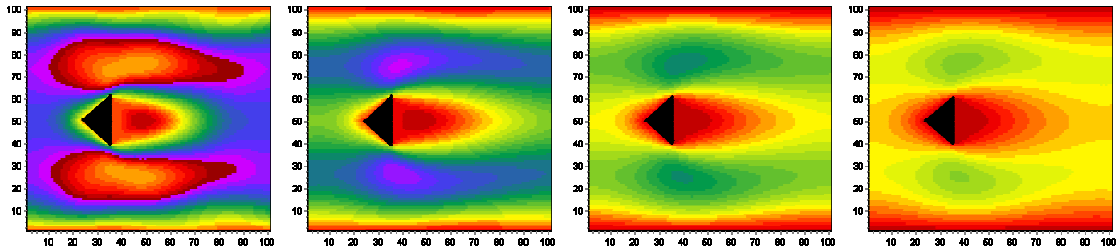

Figura 5: Primer caso de prueba, altura de agua en una corriente obstruida por una columna triangular.

La primera escena de prueba está compuesta por un canal sin pendiente con flujo de entrada constante con dirección *x*, con bordes superior e inferior utilizando condiciones de contorno No-Slip (Succi, 2001). La escena de prueba está formada por una grilla de 100x100 celdas e incluye una columna vertical de sección triangular. La Figura 5, muestra una secuencia de 4 capturas tomadas en las iteraciones 250, 500, 750 y 1000 de un total de 1000, donde la paleta de colores (de azul a rojo) representa la altura de agua.

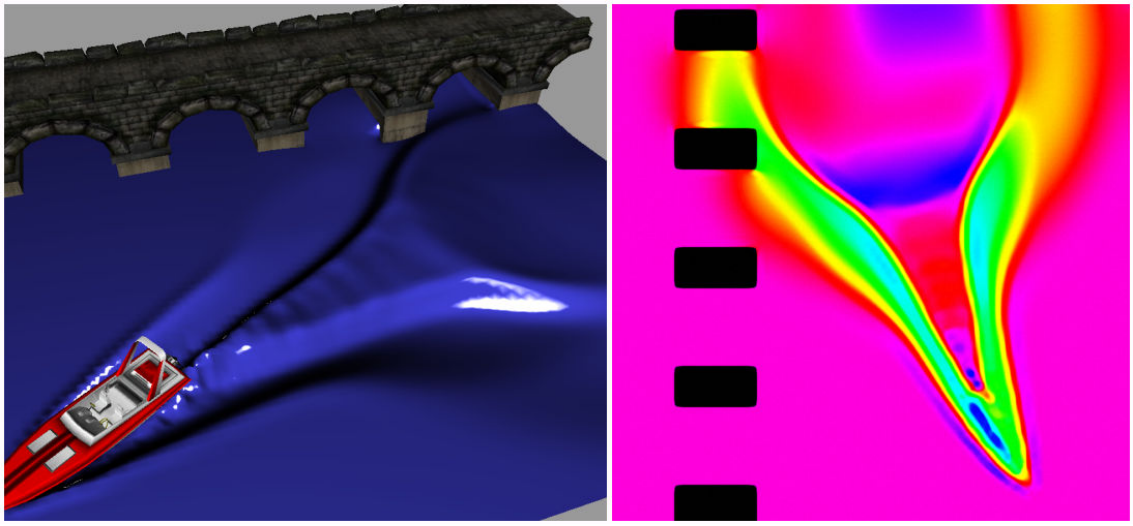

Figura 6: Segundo caso de prueba, onda generada por una lancha en un canal con un puente.

En el segundo caso se simuló la sección de un canal que contiene un puente y que es atravesado por una lancha. En la figura 6 se puede ver la onda superficial generada por el desplazamiento de la embarcación, la cual se propaga y refleja con las bases del puente.

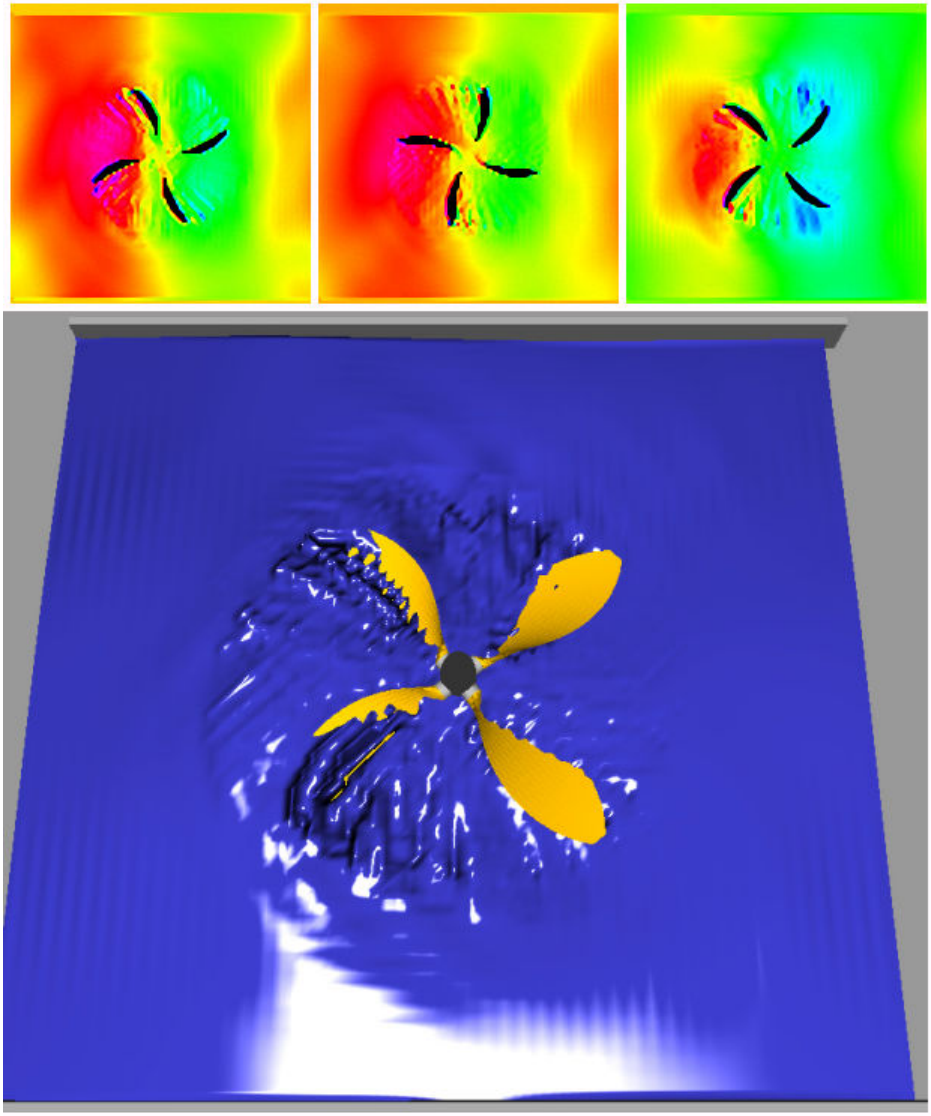

Figura 7: Tercer caso de prueba, flujo a través de una hélice rotando en sentido horario.

La tercera escena consiste de un canal sin pendiente con flujo de entrada constante en dirección *x*. Los bordes superior e inferior utilizan condiciones de contorno No-Slip (Succi, 2001). La grilla utilizada tiene un tamaño de 100x100 celdas y se incluyó en el centro del canal un obstáculo móvil en forma de hélice girando con velocidad constante en sentido horario. El borde de la hélice se genera en cada iteración como se describe en la sección 2.1. La Figura 7 muestra una secuencia de tres estados de altura de agua y la vista 3D del último de ellos.

## 4 CONCLUSIONES

En este trabajo se presentó un método para la inclusión de bordes dinámicos y obstáculos que cambian en el tiempo en un esquema de Lattice Boltzmann de aguas superficiales. Se utilizó una técnica basada en el hardware gráfico actual para generar máscaras de bordes que luego son mapeadas a una estructura fácilmente actualizable. El modelo resultante es fácil de implementar y la técnica puede ser adaptada a otros modelos de simulación de flujos. El procedimiento usado puede extenderse fácilmente a otros esquemas LBM, por ejemplo para simular las ecuaciones de Navier-Stokes en 3D.

# 5 REFERENCIAS

Ashyraliyev M. *Implicit Schemes for the Lattice Boltzmann Equation*. Tesis de Maestría en Ciencia Computacional, Universidad de Amsterdam, Agosto 2004.

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 S., Doolen G. D.*, Lattice Boltzmann Method for Fluid Flows, Annual Reviews Fluid Mechanics. 30:329-64, 1998.*

Crane K., Llamas I., Tariq S., *Real-Time Simulation and Rendering of 3D Fluids*, Chapter 30, GPU Gems 3, Addison-Wesley Professional, ISBN 978-0321515261, 2007.

Ernst M., Akenine-Moller T., Jensen H.W., *Interactive Rendering of Caustics using Interpolated Warped Volumes*, ACM International Conference Proceeding Series; Vol. 112, Proceedings of Graphics Interface 2005, p. 87 - 96, 2005.

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

García Bauza C., Boroni G., Vénere M., Clausse A. *Real-time interactive animations of liquid surfaces with Lattice-Boltzmann engines*. Aust. J. Basic & appl. Sci., 4 (8), p. 3730-3740, ISSN 1991-8178, 2010.

Génevaux, O., Habibi, A., and Dischеler, J. *Simulating fluid–solid interaction. In Graphics Interface, 31–38. 2003*

Guendelman, E., Bridson, R., and Fedkiw, R. *Non-convex rigid bodies with stacking. ACM Trans. Graph. Proc. SIGGRAPH 22, 871–878. 2003*

Iwasaki, K., Dobashi, Y. and Nishita, T. *A Fast Rendering Method for Refractive and Reflective Caustics Due to Water Surfaces*. Computer Graphics Forum, 22: 601–609, 2003.

Kim J., Kim S., Ko H., Terzopoulos D., *Fast GPU computation of the mass properties of a general shape and its application to buoyancy simulation*, The Visual Computer, Springer Berlin / Heidelberg, Vol. 22, p. 856-864, 2006.

Mitchell J., *Real-Time Synthesis and Rendering of Ocean Water*, ATI Technical. Report, April 2005.

Succi S. *The Lattice Boltzmann Equation for Fluid Dynamics and Beyond. Numerical Mathematics and Scientific Computation. Clarendon Press, Oxford, 2001.*

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

Takahashi, T., Heihachi, U., and Kunimatsu, A. *The simulation of fluid-rigid body interaction. In Proc. SIGGRAPH Sketches & applications, 2002*

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

Zhaoli G., Chuguang Z., Baochang S. *An extrapolation method for boundary conditions in lattice Boltzmann method*. Physics of Fluids, Volume 14, Number 6, 2002.