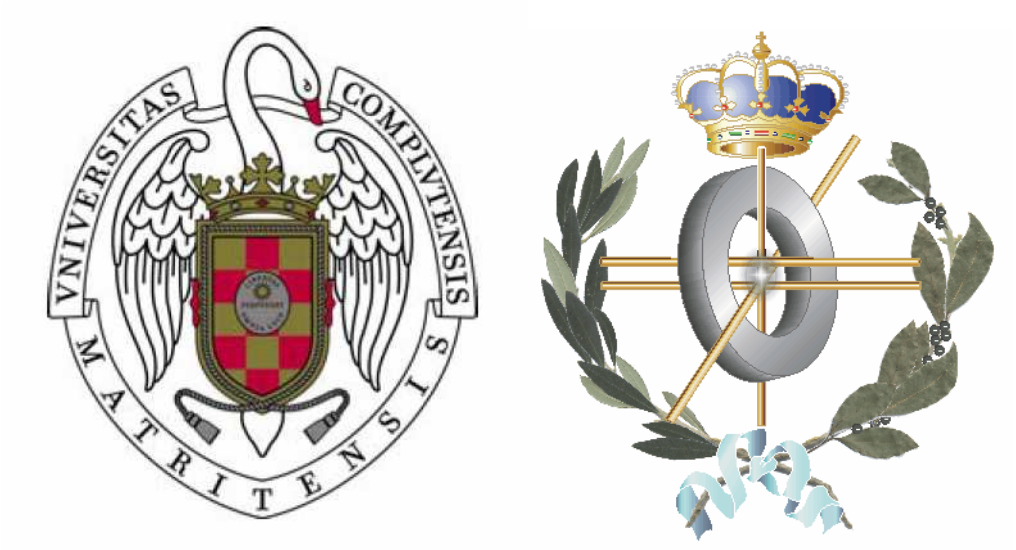
UNIVERSIDAD COMPLUTENSE DE MADRID

Facultad de Informática



SimBoltz: Simulación de fluidos mediante el método Lattice Boltzmann

Proyecto de Sistemas Informáticos

Alumno: José Luis Pérez Díaz

Profesor director: Pedro Jesús Martín de la Calle

Curso: 2009 / 2010

Resumen

El método Lattice Boltzmann permite resolver problemas de mecánica computacional de fluidos, en los cuales se simula el comportamiento de un flujo con ciertas características. Para aplicar las ecuaciones del método, debe discretizarse el espacio estudiado en cierto número de celdas por cada dimensión. En el presente desarrollo, se ha aplicado el modelo D2Q9 para 2 dimensiones en varios escenarios, unos correspondientes a problemas típicos de mecánica de fluidos y otros de propia creación, obteniendo comportamientos del flujo muy próximos a la realidad.

Abstract

Lattice Boltzmann Method solves problems of computational fluid dynamics, in which simulates the behaviour of a stream with certain characteristics. To apply the equations of the method, the studied space must be discretized in a number of cells for each dimension. In this development, D2Q9 model for two dimensions has been applied in various scenarios, some of them typical problems related to fluid mechanics and others of own creation, obtaining flow behaviours which are very close to reality.

Palabras clave

Método Lattice Boltzmann, simulación fluido incompresible, vórtice, corriente, cavidad lid-driven, Karman, bounce-back, condición periódica frontera, OpenGL.

Autorización

Se autoriza a la Universidad Complutense a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la propia memoria, como el código, la documentación y/o el prototipo desarrollado.

Fdo.: José Luis Pérez Díaz

Madrid, 2 de julio de 2010.

Índice

1. INTRODUCCIÓN 6

2. LATTICE BOLTZMANN METHOD 7

2.1. FASES DEL MÉTODO 8
2.2. CONDICIONES DE FRONTERA 10
2.2.1. Bounce-back 10
2.2.2. Condición periódica 10

3. IMPLEMENTACIÓN 11

3.1. MODELO: LBM 11
3.1.1. Representación 11
3.1.1.1. Celdas 11
3.1.1.2. Obstáculos 12
3.1.1.3. Partículas 12
3.1.2. Fases LBM 12
3.1.2.1. Fase inicial 13
3.1.2.2. Equilibrio 13
3.1.2.3. Colisión 13
3.1.2.4. Propagación 14
3.1.2.5. Cálculo de velocidad y densidad 14
3.1.3. Condiciones de frontera 15
3.1.3.1. Bounce-Back 15
3.1.3.2. Fronteras periódicas 15
3.2. VISTA: FORMULARIO OPENGL 16
3.2.1. Visualización de la escena 17
3.2.1.1. Modo Campo de Velocidades 18
3.2.1.2. Modo Animación 19
3.2.1.3. Otros modos de depuración 20
3.2.2. Menús y opciones 22
3.2.2.1. Barra de estado 22
3.2.2.2. Panel de opciones 22
3.2.2.3. Menú 23
3.3. ANIMACIÓN DE PARTÍCULAS 25
3.3.1. Comportamiento partículas 25
3.3.1.1. Copos 27
3.3.2. Visualización de partículas en el escenario 27

4. EXPERIMENTACIÓN 29

4.1. FENÓMENOS OBSERVADOS 29
4.1.1. Vórtice 29
4.1.2. Corriente 30
4.1.3. Comportamiento de las partículas 31
4.2. ESCENARIOS 32
4.2.1. Cavidad Lid-Driven 32
4.2.2. Esquina 35
4.2.3. Cruz 38
4.2.4. Cruz asimétrica 40
4.2.5. Tubería 42
4.2.6. Vórtices Karman 44
4.2.7. Nevada 47

5. TRABAJO FUTURO 50

5.1. AMPLIACIÓN A 3D 50
5.2. IMPLEMENTACIÓN EN CUDA 50
5.3. EDITOR DE ESCENARIOS 51
5.4. REALISMO DE LA ANIMACIÓN 51
5.5. OBJETOS COMPLEJOS 51

6. BIBLIOGRAFÍA 52

1. Introducción

En la mecánica de fluidos computacional (CFD) existen diversos métodos para resolver problemas sobre el flujo de líquidos y gases. En los últimos años, destacan las implementaciones del método Lattice Boltzmann (LBM) para resolver dichos problemas, debido a su naturaleza discreta y fácilmente computable, además de su alto nivel de paralelismo aprovechable en los últimos procesadores multinúcleo.

El LBM, como evolución de Lattice Gas Automata, consiste en una discretización del espacio ocupado por el fluido, y en la aplicación de ciertas ecuaciones que permiten averiguar el comportamiento del fluido en cada celda en relación a sus vecinas. El método contiene dos fases principales: propagación y colisión. La primera, describe la forma en la que el fluido se mueve de un lugar a otro; la segunda, cómo el fluido presente ya en una celda interactúa con el proveniente de sus vecinas. A lo largo de este trabajo podrán observarse con más detalle estos procesos, tanto teórica como visualmente.

Entre los problemas que pueden resolverse mediante este método, destacan los utilizados comúnmente para validar los resultados del método como la *Cavidad Lid-Driven* o los *Vórtices Karman*. Además, se han establecido otros supuestos en los que el sistema desarrollado ha mostrado comportamientos realistas y aparentemente correctos, como los escenarios *Esquina*, *Cruz*, *Cruz asimétrica* y *Tubería*, que se detallan en el capítulo de *Experimentación*.

Uno de los objetivos de estos métodos, consiste en realizar animaciones de escenas en las que intervienen fluidos, que posteriormente pueden utilizarse en el cine o videojuegos. Como aproximación a esta idea, se ha desarrollado un escenario de una *nevada*, en la que puede apreciarse el comportamiento de los copos de nieve arrastrados por las corrientes de aire.

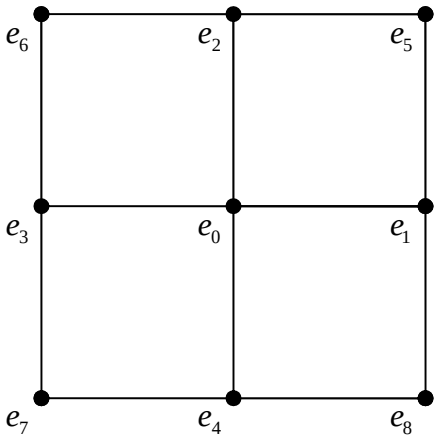
En el capítulo *Lattice Boltzmann Method*, se describen en detalle las fases del método y sus ecuaciones, la discretización del espacio, así como las condiciones de frontera que se han aplicado en el desarrollo del proyecto. El capítulo *Implementación* detalla la forma en la que se han desarrollado el modelo LBM, su visualización con OpenGL y todo lo referente a la animación de partículas en el escenario. Pueden observarse los resultados obtenidos en todos los escenarios mencionados anteriormente en el capítulo *Experimentación*, en el cual también se describe la configuración de sus obstáculos y la forma en la que el fluido se comporta inicialmente. Por último, en *Trabajo Futuro* se sugieren una serie de ampliaciones del programa desarrollado, describiendo su posible diseño.

2. Lattice Boltzmann Method

El objetivo del procedimiento Lattice Boltzmann es el de simular computacionalmente el comportamiento de un fluido incompresible a nivel macroscópico, representando la interacción entre las partículas del mismo. Para ello, se divide el espacio que quiere simularse en cierta cantidad de celdas, dependiendo del nivel de detalle requerido en los resultados, las cuales representarán la distribución de partículas del fluido en ese volumen/área determinado. A su vez, se establecen una serie de fases que, a partir de una densidad y velocidad iniciales en cada casilla, y tomando en cuenta la viscosidad característica del fluido, calculan las nuevas distribuciones de partículas en el sistema.

El fluido se representa mediante una matriz de dos o tres dimensiones según el modelo elegido, que a su vez contiene un conjunto discreto de direcciones e_i. Los modelos comúnmente utilizados son: D2Q9 para 2D (9 direcciones, utilizado en este proyecto); D3Q15, D3Q19 y D3Q27 para 3D.

El conjunto de direcciones para el modelo D2Q9 es el siguiente:



siendo:

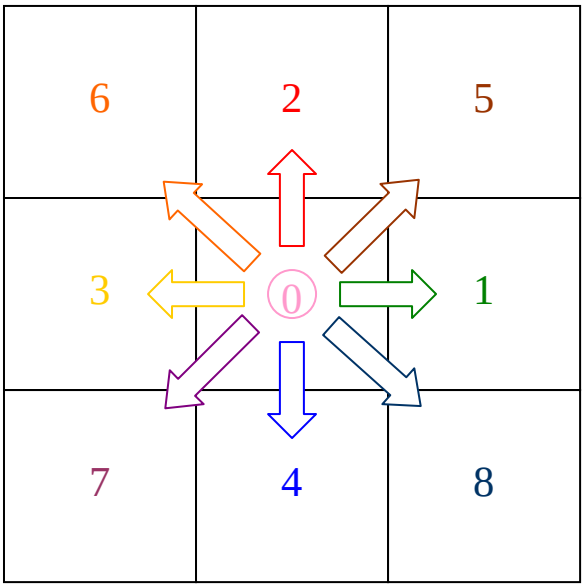
$e_0 = (0,0)$

$e_i = (\pm 1,0)$ para $i = 1,2$

$e_i = (0,\pm 1)$ para $i = 3,4$

$e_i = (\pm 1,\pm 1)$ para $i = 5,6,7,8$

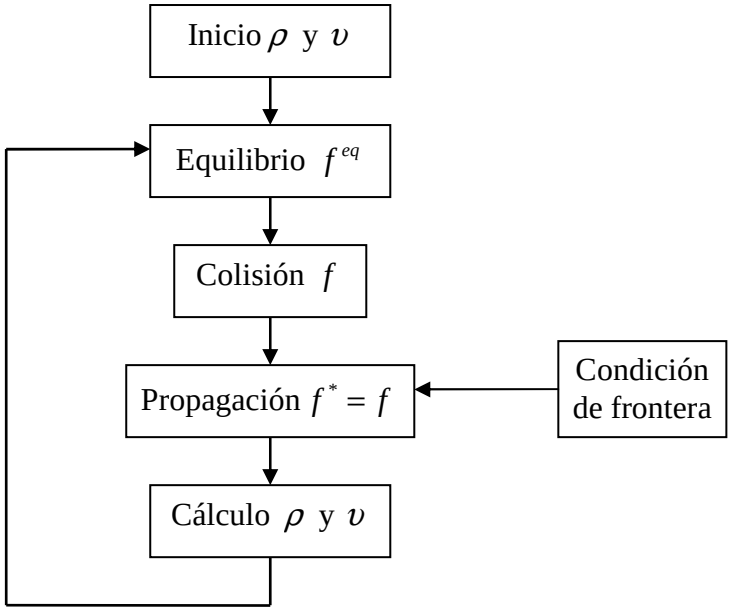
Como podemos ver a continuación, en este modelo existe una correspondencia directa entre las direcciones e_i y las celdas vecinas, lo que simplifica y acelera el cálculo de interacciones entre ellas.



Correspondencia directa entre direcciones y celdas vecinas.

2.1. Fases del método

Aunque el método original consiste únicamente en las fases de colisión y propagación, puede representarse mediante el siguiente esquema:



Inicio: La primera fase debe computarse únicamente al comienzo para establecer la densidad (ρ) y velocidad (v) iniciales en todas las celdas. La velocidad no puede superar 0.1 para una densidad de 1.0, por lo que debe cumplirse:

(1) $\rho = 1.0$ $v = 0.0$ $(-1.0 \le v \le 1.0)$

Equilibrio: esta fase se encarga de calcular la distribución de equilibrio de las partículas del sistema (f^{eq}), a partir de la densidad y velocidad de todas las celdas, que sirve para suavizar posteriormente las distribuciones. Para cada dirección i , se calcula mediante las siguientes ecuaciones:

(2) $$f_i^{eq} = \omega_i \left[\rho + 3e_i \cdot \upsilon - \frac{3}{2}\upsilon^2 + \frac{9}{2}(e_i \cdot \upsilon)^2 \right]$$, donde:

$$\omega_i = \begin{cases} 4/9 \text{ para } i = 0. \\ 1/9 \text{ para } i = 1,2,3,4. \\ 1/36 \text{ para } i = 5,6,7,8. \end{cases}$$

Como puede observarse, ω_i es un factor de peso, que pondera la distribución en función de las direcciones *i*, asignando mayor importancia a la dirección de reposo $i = 0$, después a las direcciones vertical y horizontal $i = 1,2,3,4$, y por último y con menor peso, a las direcciones en diagonal $i = 5,6,7,8$.

Colisión: Una vez calculada la distribución de equilibrio y la distribución temporal de la iteración anterior (f^*), se obtiene la nueva distribución en reposo (f), esto es:

(3) $$f_i = f_i^* - \omega \left(f_i^* - f_i^{eq} \right)$$

donde $\omega \in (0,2]$ representa la viscosidad del fluido, siendo menor para fluidos muy viscosos y mayor para flujos turbulentos. Al iniciar el sistema, tanto f_i^* como f_i deben tomar el valor de f_i^{eq} .

Propagación: este paso únicamente copia a las celdas vecinas correspondientes, las partículas que se trasladan en su dirección, esto equivale a:

(4) $$f_i^*(x + e_i, t + \Delta t) = f_i(x,t)$$

Cálculo densidad y velocidad: a partir de la distribución temporal, se suman las propiedades de todas las direcciones de la celda:

(5) $$\rho = \sum f_i^*$$ **(6)** $$\upsilon = \sum e_i f_i^*$$

2.2. Condiciones de frontera

Durante la fase de propagación, es necesario tener en cuenta las fronteras del sistema y los posibles obstáculos. Para ello se deben aplicar condiciones que determinen el comportamiento del mismo en estos extremos.

Las condiciones utilizadas en el proyecto actual han sido *bounce-back* y *condición periódica*, que se explican a continuación.

2.2.1. Bounce-back

Se trata de una condición sencilla de implementar, que consiste en cambiar la dirección de las partículas que tratan de cruzar una frontera u obstáculo, por su inversa, conservándose en la misma celda. Afecta a la fase de propagación y puede describirse como:

(7) $f_i^*(x,t+\Delta t) = f_{\tilde{i}}(x,t)$, siendo:

$\tilde{i} = i+2$ para $i = 1,2,5,6$

$\tilde{i} = i-2$ para $i = 3,4,7,8$

2.2.2. Condición periódica

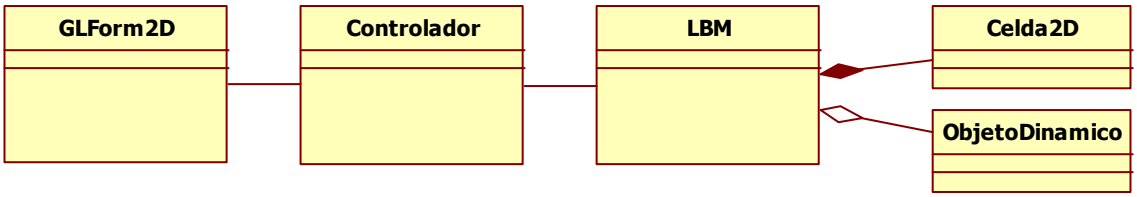
Utilizada en problemas donde es necesaria una entrada y salida de flujo en el sistema, consiste en mantener las condiciones de un extremo del dominio, iguales a las de su frontera opuesta. Esto es, para un dominio de *n x m* celdas, aplicado a las fronteras derecha e izquierda:

(8) $\forall y : 0 \le y < m : f_i^*((n-1,y),t+\Delta t) = f_i((0,y),t)$, y

$\forall y : 0 \le y < m : f_i^*((0,y),t+\Delta t) = f_i((n-1,y),t)$

3. Implementación

Como patrón de diseño, se ha aplicado un Modelo-Vista-Controlador. En el modelo, la clase principal es LBM, que ofrece métodos al controlador. La vista GLForm2D, obtiene la información necesaria para representar el estado del sistema en la interfaz de usuario del mismo controlador. El diagrama de clases simplificado de la aplicación es el siguiente:



3.1. Modelo: LBM

3.1.1. Representación

Para simbolizar el método Lattice Boltzmann, se ha creado una única clase que, a su vez, utiliza las clases Celda2D y ObjetoDinamico, explicadas en los próximos apartados.

La clase ofrece métodos públicos que devuelven toda la información necesaria del sistema para su posterior visualización, esto es: tamaño del sistema, densidad total y velocidad de las celdas, naturaleza de la celda (fluido, obstáculo), y si contiene o no partículas.

Por otro lado, permite la ejecución de iteraciones del método, el cálculo del desplazamiento de partículas, y la aplicación de diversos escenarios predefinidos en el propio LBM, entre otras tareas.

Además, existen métodos que inicializan el LBM con datos precalculados en un fichero, y a su vez, que lo guardan para futuros experimentos.

La clase dispone de una matriz de Celda2D y de una lista de ObjetoDinamico, para representar las casillas del sistema y las partículas, respectivamente.

3.1.1.1. Celdas

En la clase Celda2D, se tienen una serie de atributos para describir cada casilla del sistema. Entre ellos, las distribuciones de partículas en reposo, equilibrio y temporal, representadas mediante matrices de *float* para todas las direcciones de la celda; además de la densidad y velocidad acumulada.

3.1.1.2. Obstáculos

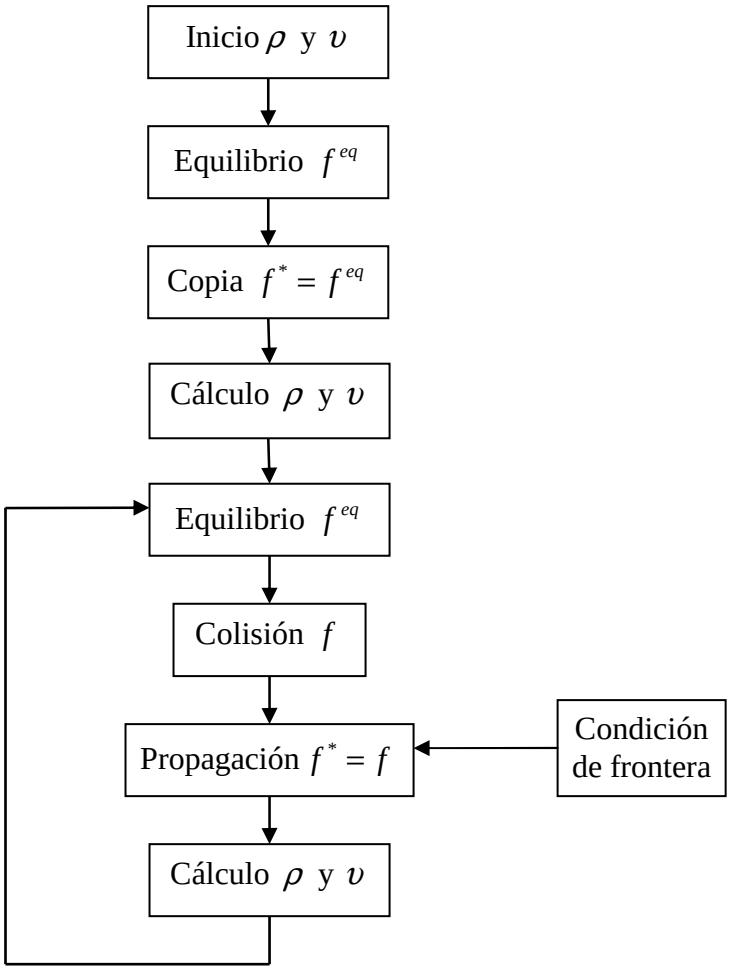
Para representar si una celda corresponde a un obstáculo o al fluido, no se utilizan clases diferentes, simplemente se diferencian en un atributo. En caso de ser un obstáculo, se almacena también el color del mismo, para una mejor visualización.

3.1.1.3. Partículas

La clase ObjetoDinamico contiene la posición de la partícula que representa, además de su color y su velocidad "acumulada". En el apartado *Animación de partículas* de esta memoria, se detalla la utilidad de dicha velocidad y la forma en la que se calculan los desplazamientos de las partículas en base al campo de fuerzas previamente obtenido según el modelo LBM.

3.1.2. Fases LBM

A continuación, se muestra un esquema de la implementación del LBM, donde puede distinguirse una fase inicial más detallada:



3.1.2.1. Fase inicial

Al crear el sistema con un tamaño ya especificado, se inicializan todas las celdas según la ecuación **(1),** con densidad 1.0 y velocidad (0.0, 0.0). Después, se ejecutan las fases de equilibrio y copia de f^{eq} a f^{*} desde la clase Celda2D, calculándose la nueva densidad y velocidad en cada celda.

3.1.2.2. Equilibrio

La distribución de equilibrio se calcula desde la propia celda, debido a que no necesita información de sus vecinas. El algoritmo basado en la ecuación **(2)** se describe así:

```
void LBM::equilibrar(){
  para (i=0; i<n; i++) // n celdas horizontal
     para (j=0; j<m; j++) // m celdas vertical
        si (!celdas[i][j].obstaculo) // si no se trata de un obstáculo
           celdas[i][j].equilibrarCelda();
}

void Celda2D::equilibrarCelda(){
  para (k=-1; k<=1; k++) // direcciones de la celda eje X,…
     para (l=-1; l<=1; l++) // …direcciones eje Y.
        prod= productoEscalar(k, l, vel.x, vel.y);
        cuad= productoEscalar(vel.x, vel.y, vel.x, vel.y);
        ei=dirección(k,l);
        Feq= peso(k,l)*(densidad +3*prod -(3/2)*cuad +(9/2)*prod*prod);
}
```

donde peso(ei) devuelve el factor de peso que corresponde a esa dirección, detallado en el apartado *Fases del Método* de esta misma memoria.

3.1.2.3. Colisión

Esta fase también se ejecuta desde la propia celda, puesto que utiliza únicamente las distribuciones temporales y de equilibrio de la misma, además de la viscosidad definida para el sistema. Basándonos en la ecuación **(3)**, puede resumirse como:

```
void LBM::colisionar(){
  para (i=0; i<n; i++) // n celdas horizontal
     para (j=0; j<m; j++) // m celdas vertical
        si (!celdas[i][j].obstaculo) // si no se trata de un obstáculo
           celdas[i][j].colisionarCelda();
}

void Celda2D::colisionarCelda(){
  para (k=-1; k<=1; k++) // direcciones de la celda eje X,…
     para (l=-1; l<=1; l++) // …direcciones eje Y.
       Freposo[k][l]= Ftemp[k][l]-(viscosidad*(Ftemp[k][l]-Feq[k][l]));
}
```

3.1.2.4. Propagación

La propagación debe ejecutarse desde la clase LBM, puesto que es la única fase en la que interaccionan las celdas del sistema.

Existen dos enfoques para esta fase, denominados *push* y *pull*. El primero de ellos consiste en recorrer todas las celdas del sistema, copiando su distribución de reposo en las distribuciones temporales de las celdas vecinas. La estrategia *pull*, por el contrario, recorre todas las celdas, copiando las distribuciones de reposo de las celdas vecinas, en su distribución temporal. La principal ventaja del enfoque *pull* consiste en que la ejecución de las siguientes fases del algoritmo puede continuar para una celda en cuanto el paso de propagación acaba para ella. En consecuencia, la eficiencia del sistema podría beneficiarse en arquitecturas de procesadores paralelos, si la ejecución del algoritmo para cada celda tiene lugar en una hebra independiente. Como el objetivo de este proyecto ha sido la implementación del método LBM en un único procesador, la elección del enfoque carece de relevancia. En nuestro caso, hemos optado por la aproximación *push*.

El algoritmo simplificado, basado en la ecuación **(4)** para un escenario sin condición periódica y con Bounce-Back en todas sus fronteras, es el siguiente (el método para Bounce-Back se detallará en el apartado *Condiciones de Frontera* de este mismo capítulo):

```
void LBM::propagar(){
  para (i=0; i<n; i++) // n celdas horizontal
     para (j=0; j<m; j++) // m celdas vertical
        si (!celdas[i][j].obstaculo) // si no se trata de un obstáculo
           propagarCelda(i,j);
}
void LBM::propagarCelda(i,j){
  para (k=-1; k<=1; k++) // direcciones de la celda eje X,…
     para (l=-1; l<=1; l++) // …direcciones eje Y.
       si(i+k>=n|| i+k<0|| j+l>=m|| j+l<0|| celdas[i+k][j+l].obstaculo)
          aplicarBounceBack(i,j,k,l);
       si no    // destino no es frontera ni obstáculo.
          celdas[i+k][j+l].Ftemporal[k][l]= celdas[i][j].Freposo[k][l];
}
```

3.1.2.5. Cálculo de velocidad y densidad

El cálculo según las ecuaciones **(5)** y **(6),** de los sumatorios de la distribución de partículas en todas las direcciones de la celda, y de los productos de la distribución y la dirección correspondiente, es inmediato. El algoritmo es el siguiente:

```
void LBM::calcularDensidadVelocidad(){
  para (i=0; i<n; i++) // n celdas horizontal
     para (j=0; j<m; j++) // m celdas vertical
        si (!celdas[i][j].obstaculo) // si no se trata de un obstáculo
           celdas[i][j].calcularDVCelda();
}
void Celda2D::calcularDVCelda(){
  densidad = vel.x = vel.y = 0;
  para (k=-1; k<=1; k++) // direcciones de la celda eje X,…
     para (l=-1; l<=1; l++) // …direcciones eje Y.
        densidad += Ftemp[k][l];
        vel.x += k*Ftemp[k][l]; vel.y += l*Ftemp[k][l];
}
```

3.1.3. Condiciones de frontera

Para la implementación del método LBM en este proyecto, se han elegido dos condiciones que describen el comportamiento del fluido al aproximarse a las fronteras del sistema y a los posibles obstáculos presentes en el escenario. La condición Bounce-Back es la elegida por su sencillez para resolver la interacción con obstáculos y fronteras. La condición periódica, utilizada en alguno de los escenarios predefinidos, se aplica en la propagación de las celdas situadas en aquellas fronteras que previamente se han marcado como periódicas.

Durante el desarrollo del proyecto, se probaron diferentes condiciones más complejas, en las que se trataba de describir el comportamiento de los fluidos de una manera más realista, pero, a pesar de su dificultad, se observaron resultados muy similares a los obtenidos aplicando Bounce-Back, pero eso sí con una eficiencia menor, puesto que requerían mayor cantidad de cálculos. Finalmente, se optó por simplificar la condición de Bounce-Back en las fronteras aun más, incluyendo obstáculos en los márgenes de los escenarios que simulan el comportamiento de la frontera. En consecuencia, no ha sido necesario distinguir entre obstáculo y frontera en el algoritmo.

3.1.3.1. Bounce-Back

Como se ha indicado, se ha unificado el comportamiento del fluido al "chocar" con un obstáculo o frontera, puesto que las celdas que conforman las fronteras del sistema (que deban ser paredes), se configuran como obstáculos.

En base a la ecuación **(7)**, el algoritmo de la condición Bounce-Back para cada celda, podría resumirse como:

```
void LBM::aplicarBounceBack(i,j,k,l){
   …
   celdas[i][j].Ftemp[k][l]= celdas[i][j].Freposo[inv(k)][inv(l)];
}
```

donde *i* y *j* corresponden a los índices de la celda en el sistema, *k* y *l* son los índices de la dirección e_i , e inv(*k*) devuelve la dirección inversa a *k*.

3.1.3.2. Fronteras periódicas

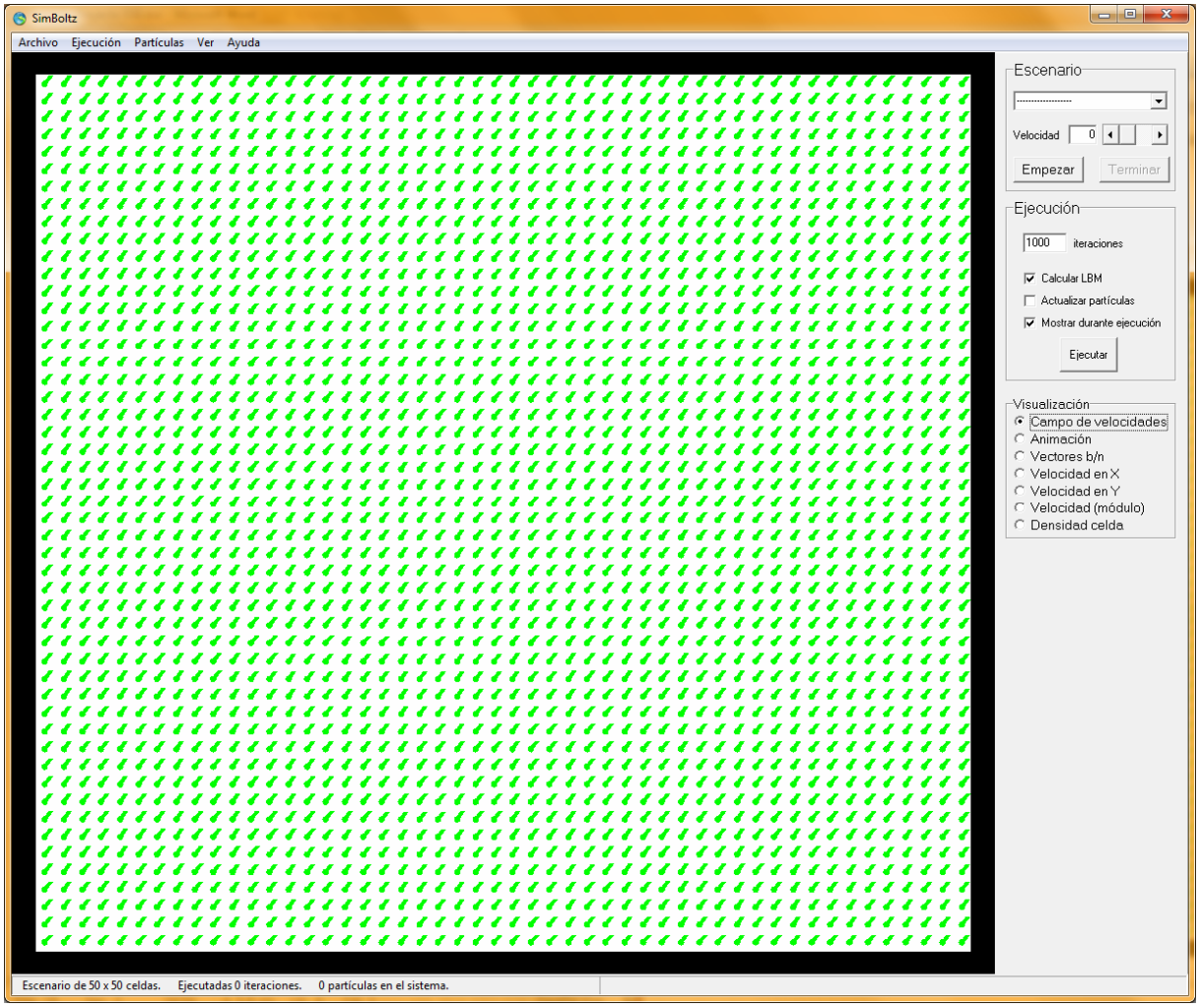
Todos los escenarios predefinidos, excepto la cavidad Lid-Driven, utilizan esta condición especial. Se aplica en la fase de propagación en aquellas fronteras que se hayan "desactivado", que en nuestros ejemplos corresponden a las fronteras derecha e izquierda, como se aplica en la ecuación **(8)**. La implementación extiende el algoritmo de Bounce-Back, como sigue:

```
void LBM::aplicarBounceBackConPeriodica(i,j,k,l){
 si ( celdas[i+k][j+l].obstaculo || paredActiva(i,j,k,l) )
        celdas[i][j].Ftemp[k][l]= celdas[i][j].Freposo[inv(k)][inv(l)];
 si no si (i==0) celdas[n-1][j].Ftemp[k][l]=celdas[0][j].Freposo[k][l];
     si no celdas[0][j].Ftemp[k][l]= celdas[n-1][j].Freposo[k][l];
}
```

donde paredActiva(i,j,k,l) indica si, para la celda (*i*,*j*) con dirección (*k*,*l*), la pared del sistema está activada o no. En el algoritmo descrito, solo se ha evaluado la condición periódica en las paredes derecha e izquierda. Las condiciones serían análogas en las paredes superior e inferior.

3.2. Vista: formulario OpenGL

El formulario está dividido en cuatro partes: el menú de opciones, el panel derecho que sirve para interactuar con el escenario, la barra de estado y el panel central donde se visualiza la escena. Este es el aspecto de la interfaz:



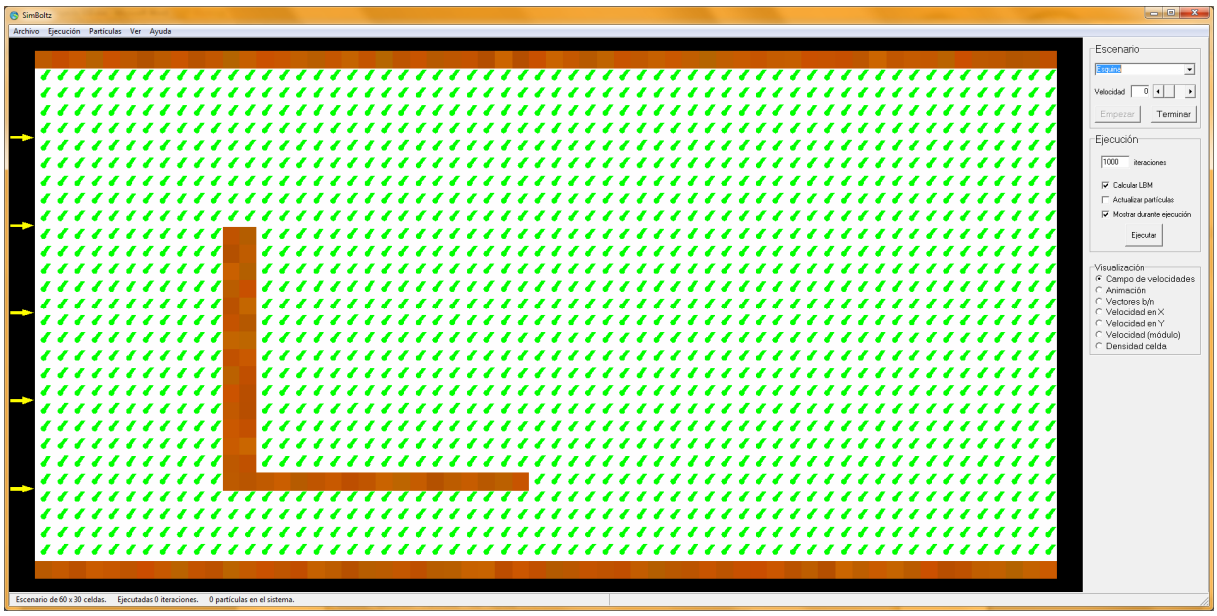
La ventana se redimensiona en base al tamaño del escenario cargado, guardando la proporción entre el número de celdas en el eje X y la cantidad en el eje Y, de tal forma que las casillas siempre tengan forma cuadrada. En todo caso, se toma en cuenta la resolución del sistema en el que se está ejecutando, aprovechando la mayor parte de la pantalla disponible.

3.2.1. Visualización de la escena

La visualización del modelo se realiza en el panel central del formulario. Se han desarrollado varios modos de visualización, que se detallan a continuación. Pueden escogerse desde el panel derecho o desde el menú.

Todos los modos tienen en común la forma en que se representan los obstáculos y las partículas. Los primeros se dibujan con el tamaño total de las celdas que ocupan y con su color propio, que se almacena en la clase *Celda2D*. Las partículas, sin embargo, se pintan mediante un *GL_POINT*, con un grosor tal que el diámetro del punto sea del 80% del tamaño de la casilla, y con el color que tienen guardado en la propia clase *ObjetoDinámico*.

En todas las vistas, se muestra la dirección de "entrada" del fluido en el sistema, dibujando flechas en el margen correspondiente como muestran los siguientes ejemplos:



3.2.1.1. Modo Campo de Velocidades

Este es el tipo de visualización más interesante de los disponibles, ya que en él se pueden identificar fácilmente todos los fenómenos acontecidos en los diversos escenarios.

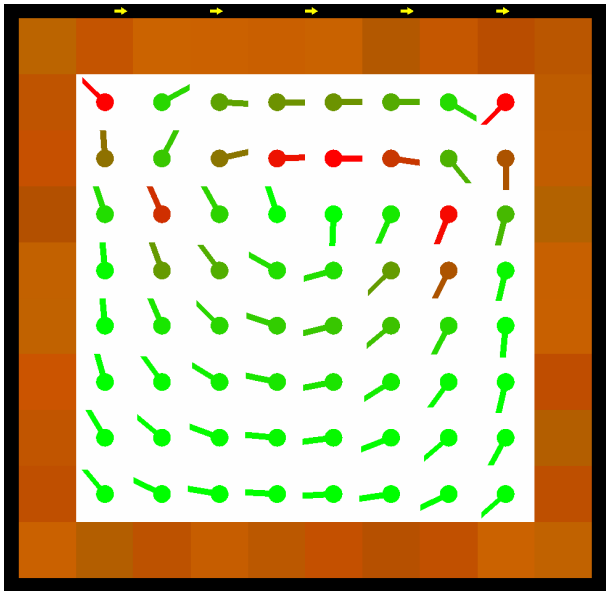
Se trata de una representación en la cual se visualiza, para cada celda del sistema, un vector con las características de la velocidad del fluido en dicha celda. En este caso, el tamaño del vector es siempre el mismo, en lugar de usar la magnitud del vector, homogeneizando así todas las casillas y evitando que en aquellas en las que la velocidad fuese nula, no se dibujase nada. No obstante, el color del vector simboliza la magnitud, mediante el siguiente convenio: el color verde representa la velocidad nula y el rojo la velocidad máxima del sistema (0.1,0.1). La dirección del vector, concuerda con la dirección de la velocidad de las partículas de la celda. El algoritmo para dibujar el campo vectorial es el siguiente:

```
void GLForm2D::pintarCampoVelocidades(){
 para (i=0; i<n; i++) // n celdas horizontal
    para (j=0; j<m; j++) // m celdas vertical
       si (LBM.celdas[i][j].obstaculo)
          … //se dibuja el obstáculo con su color en toda la celda.
       si no
          a0=i*tamCeldax – medioEspaciox + mediaCeldax;
          a1=j*tamCelday – medioEspacioy + mediaCelday;
          b0= LBM.getVelocidadx(i,j);
          b1= LBM.getVelocidady(i,j);
          modulo= sqrt(pow(b0,2.0)+pow(b1,2.0));
          c0=a0+(radioCelda/modulo)*b0;
          c1=a1+(radioCelda/modulo)*b1;
          color=modulo/ LBM.getMaxVel();
          PintarLineaColor(a0,a1,c0,c1,color);
}
```

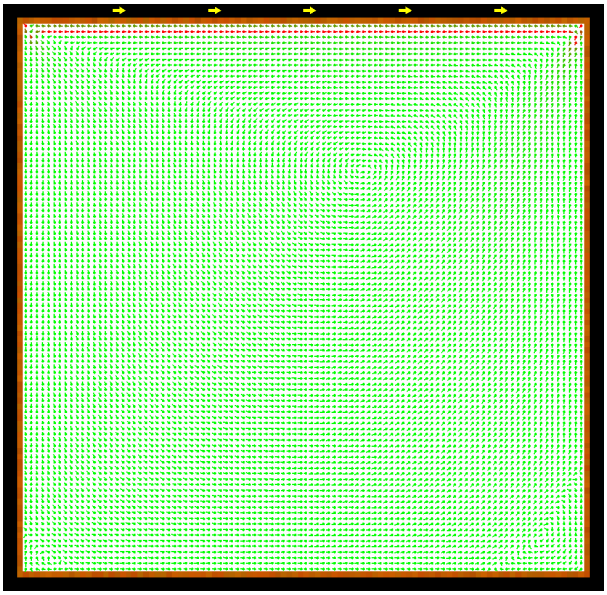
En el código *(a0,a1)* es el centro de la casilla y origen del vector, *(c0,c1)* es el destino del vector, *color* representa la magnitud de la velocidad del vector en porcentaje respecto a la máxima velocidad del sistema y *PintarLineaColor(a0,a1,c0,c1,color)* dibuja una línea desde (a0,a1) hasta (c0,c1), y un punto más grueso en (a0,a1), ambos con color RGB calculado como (0.0+pow(color,3) , 1.0- pow(color,3) , 0.0). Como puede observarse, se aplica una interpolación cúbica al color, para exagerar los extremos con menor y mayor velocidad.

Para mejorar la eficiencia del algoritmo, los valores tamCeldax, tamCelday, radioCelda, medioEspaciox y medioEspacioy, se precalculan al iniciar el programa y se actualizan únicamente en caso de redimensionar el formulario. De esta forma no es necesario calcularlos cada vez que se dibujan los escenarios, para cada una de las celdas.

A continuación se muestra un ejemplo de este modo de visualización para tamaños de LBM de 10 x 10 y 100 x 100 casillas, aplicando un mismo escenario, donde se puede apreciar como el grosor y tamaño de los vectores se adapta al tamaño de las celdas:



Sistema de 10x10 casillas.



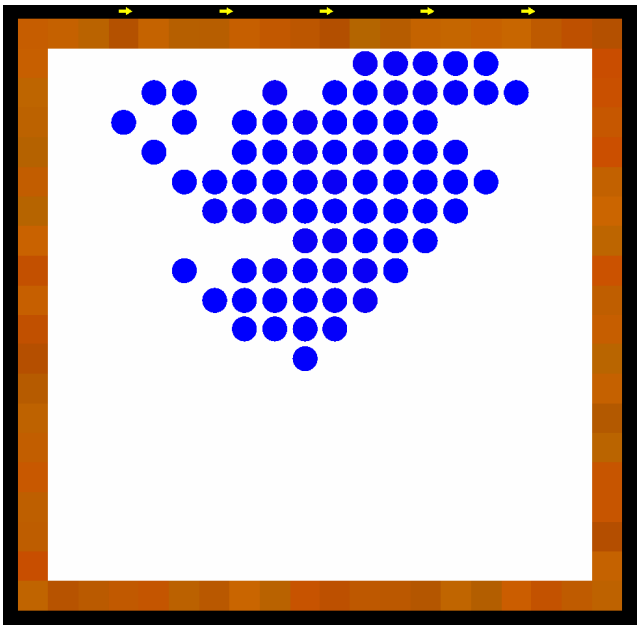
Sistema de 100x100 casillas.

3.2.1.2. Modo Animación

Este tipo de visualización está concebido únicamente con el objetivo de mostrar el escenario y el comportamiento de las partículas en él, por lo que solo se muestran los obstáculos y partículas con su propio color, sobre un fondo blanco que cubre todo el panel.

Se ha decidido no mostrar el campo vectorial del fluido ni ninguna de sus características, para acelerar la visualización de la animación.

Un ejemplo sencillo aplicando este modo de visualización es el siguiente:

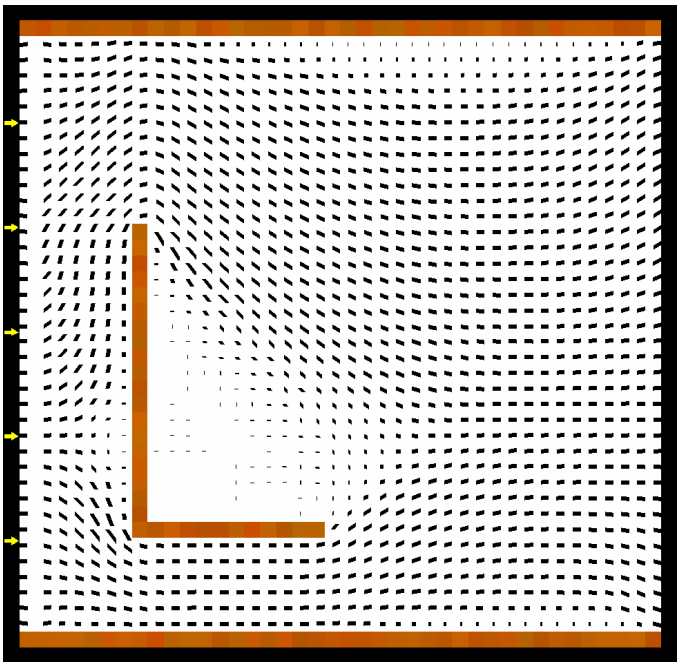


Sistema de 20x20 casillas en modo Animación.

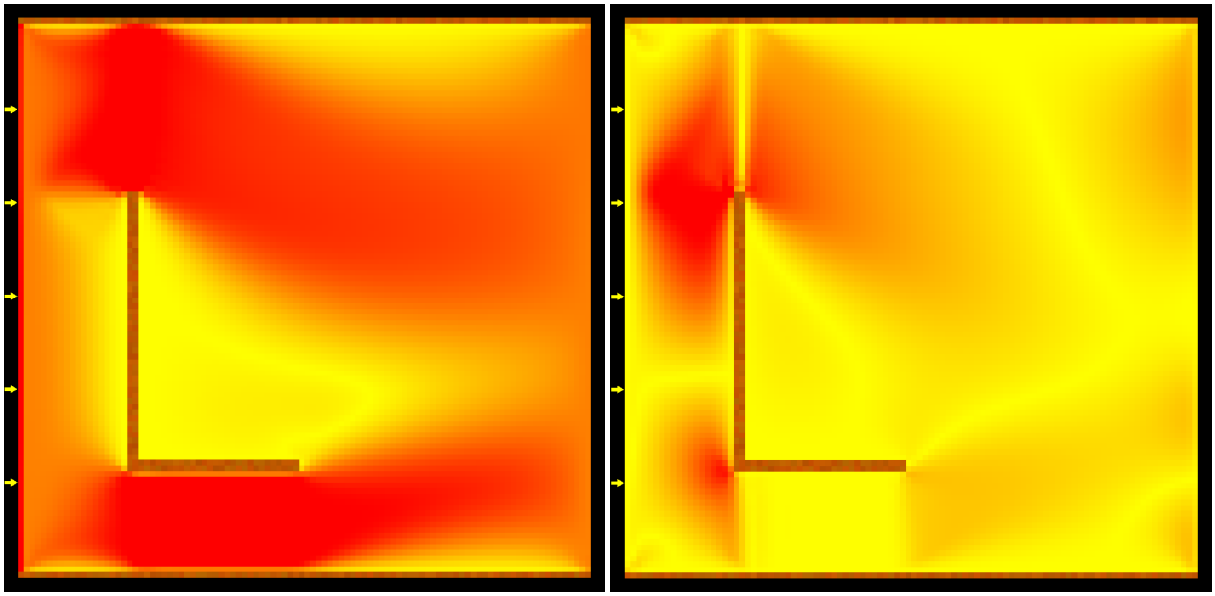
3.2.1.3. Otros modos de depuración

Durante el desarrollo del proyecto, se han utilizado otros modos de visualización para depurar y comprobar el buen funcionamiento del sistema, que se muestran a continuación.

Vectores blanco y negro: en este caso la representación del sistema es similar al modo Campo de Velocidades, con la diferencia de que la magnitud de la velocidad en cada celda se representa con el tamaño del vector y no con el color del mismo. Tiene una gran desventaja respecto a los vectores de colores, y es que precisamente donde suelen generarse fenómenos interesantes, las velocidades son de menor magnitud, por lo que deja de apreciarse la dirección de los vectores, o incluso llega a desaparecer. Por ejemplo, en la imagen se observa que detrás del obstáculo hay celdas en las que no se distingue la dirección del fluido.

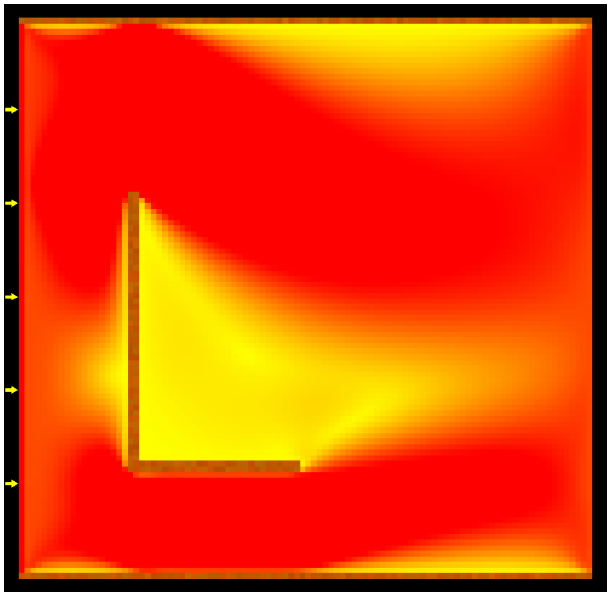


Velocidad en X o Y: orientada al análisis de la velocidad del fluido en cada celda, esta vista dibuja la casilla con cierto color, dependiendo de la magnitud de la velocidad en su componente X ó Y, según el modo seleccionado. La escala de colores se sitúa entre el amarillo y el rojo, siendo este último el de mayor magnitud, basándose en la velocidad máxima del sistema.



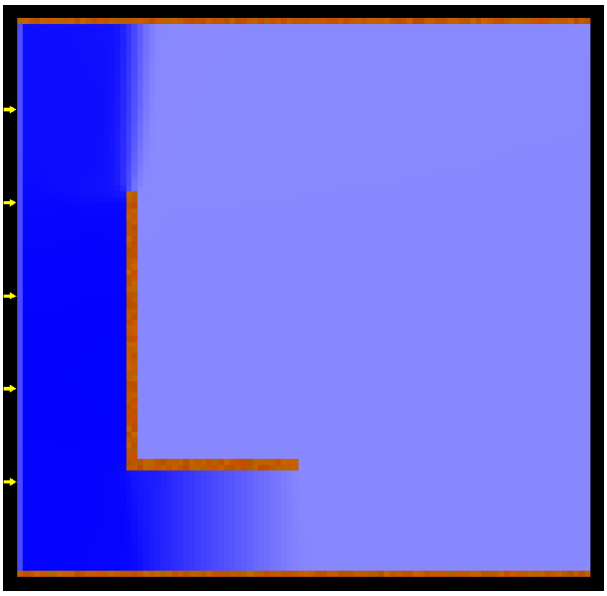
Modo visualización Velocidad en X. *Modo visualización Velocidad en Y.*

Módulo velocidad: esta vista es una conjunción de las vistas velocidad en X e Y, ya que con la misma escala de colores, representa la magnitud de la velocidad mediante la ecuación: $módulo = \sqrt{velocidadX^2 + velocidadY^2}$.



Modo visualización módulo Velocidad.

Densidad celda: en este tipo de visualización, la densidad del fluido en la celda se representa mediante otra escala de colores: blanco para una densidad nula y azul para la mayor densidad del sistema. No es especialmente interesante en ninguno de los escenarios estudiados, pero al comienzo del desarrollo fue de gran ayuda para depurar el funcionamiento del método LBM. Siguiendo con el mismo escenario, obtendríamos la siguiente imagen:

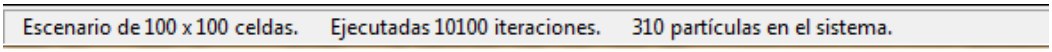


Modo visualización Densidad Celdas.

3.2.2. Menús y opciones

3.2.2.1. Barra de estado

La barra de estado está dividida en dos partes. La primera muestra información del escenario, que incluye el tamaño del sistema, el número de iteraciones ejecutadas y la cantidad de partículas que están visualizándose.



La segunda parte muestra información sobre la ejecución de iteraciones. Mientras se están realizando cálculos, aparecen mensajes informando sobre que tipo de ejecución se está realizando. Además, si se sitúa el puntero del ratón sobre el campo editable del número de iteraciones a ejecutar, se muestra el tiempo aproximado que requiere su ejecución.

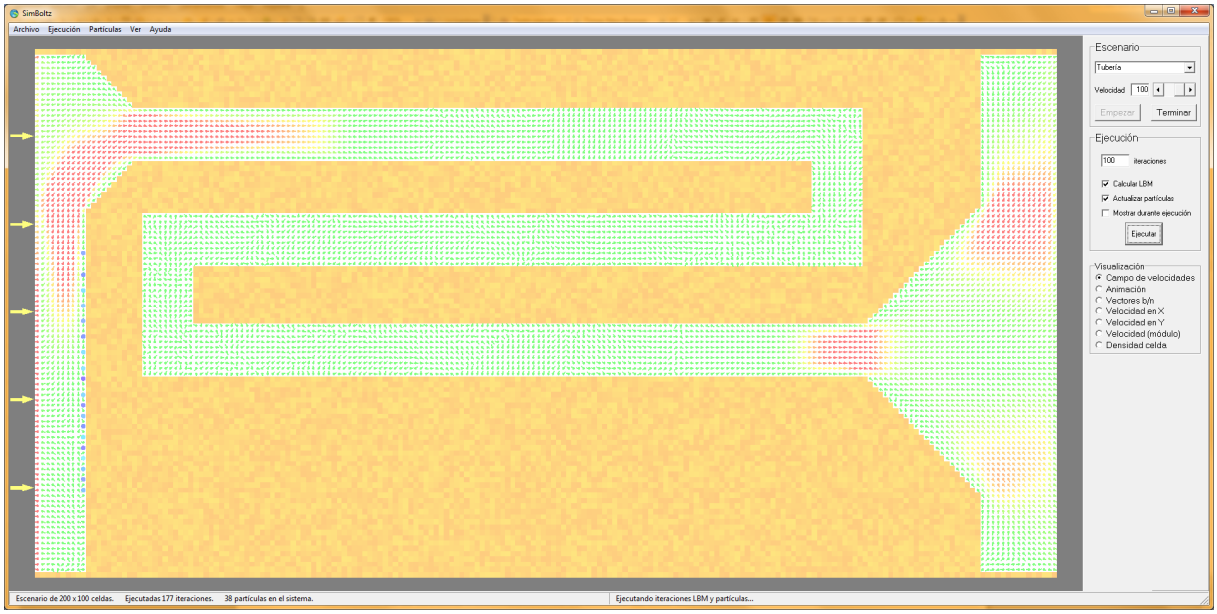
3.2.2.2. Panel de opciones

En este panel hay tres conjuntos de opciones diferenciados: *Escenario*, *Ejecución* y *Visualización*.

En el grupo *Escenario*, puede seleccionarse uno de los escenarios predefinidos. Una vez elegido, debe fijarse el porcentaje de velocidad que va a aplicarse en la entrada de fluido propia de cada escenario. Tanto la barra de desplazamiento como el campo editable, representan dicho porcentaje de 0 a 100. Después, pulsando empezar se construirá el escenario, añadiendo los obstáculos predefinidos. En sucesivas iteraciones se aplicará la velocidad seleccionada sobre las celdas que corresponda, según el escenario. Pulsando terminar, dejará de aplicarse dicha velocidad, pero se mantendrán los obstáculos en el escenario, por lo que se puede continuar iterando sin entrada de fluido en el sistema.

En la zona de *Ejecución*, puede indicarse el número de iteraciones a ejecutar, estando limitado este campo a 10.000. Si se marca la opción *Calcular LBM*, en cada una de las iteraciones antes indicadas, se recalculará el campo vectorial del fluido aplicando el método LBM, lo cual ralentiza la ejecución. Seleccionando *Actualizar partículas*, se añadirán partículas al fluido, en zonas predefinidas para cada escenario y, durante el proceso de ejecución, se actualizarán sus posiciones en base al campo de velocidades del mismo.

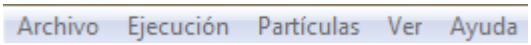
La opción *Mostrar durante ejecución* indica si deben visualizarse los cambios en el escenario tras cada una de las iteraciones, o únicamente al finalizar la totalidad. En caso de desmarcarse, se minimiza el tiempo de ejecución. Para indicar que el escenario no está actualizándose en tiempo real, se muestra una capa semitransparente sobre el sistema, como puede verse en la imagen:



En el apartado de Visualización, se puede seleccionar uno de los modos de vista, ya descritos en la sección *Visualización de la escena* de este documento.

3.2.2.3. Menú

El menú principal de la aplicación, situado en la parte superior izquierda, contiene los siguientes submenús.



Archivo

Desde aquí puede crearse un nuevo escenario seleccionando *Nuevo*, tras lo cual aparecerá una ventana en la que se debe introducir el tamaño del nuevo escenario y la viscosidad del fluido. El tamaño no debe exceder 1000x300 ni ser inferior a 10x10 celdas. Además, tal como precisa el método LBM, la viscosidad debe encontrarse entre 0.0 y 2.0. Si alguno de los valores no se encuentra en el rango especificado, se muestra un mensaje indicando los límites permitidos.

Por otra parte, puede abrirse un escenario previamente guardado seleccionando *Abrir*, o guardarlo mediante la opción *Guardar como*. En ambos casos, se puede navegar por el sistema de archivos para seleccionar un fichero donde guardar o cargar el escenario. Esto permite ahorrar

mucho tiempo al poder almacenar escenarios con una gran cantidad de celdas e iteraciones ejecutadas, con los que luego experimentar.

Por otro lado, la matriz de velocidades del fluido puede exportarse a documentos de texto, mediante las opciones *Exportar→Velocidades X e Y*, para su posible uso en otro tipo de aplicaciones de simulado.

Ejecución

En este menú puede reiniciarse el sistema, reiniciando el fluido y eliminando obstáculos y partículas. Además, pueden ejecutarse iteraciones manualmente, tanto calculando el LBM, como simplemente actualizando las partículas en el fluido. Ambas funciones, son accesibles mediante las teclas F2 y F3 respectivamente, lo que permite observar la evolución del sistema ejecutando iteración a iteración, sin necesidad de fijar un número de iteraciones a ejecutar.

Partículas, Ver y Ayuda

El menú *Partículas* permite añadir o quitar las partículas predefinidas para cada escenario, sin alterar los obstáculos ni el campo vectorial del fluido.

Desde el menú *Ver*, puede rotarse el modo de visualización del escenario, lo que puede conseguirse también con la tecla F7.

Por último, el menú *Ayuda* muestra información sobre la aplicación.

3.3. Animación de partículas

En esta sección de la memoria, se van a detallar las decisiones de diseño e implementación relacionadas con la animación de las partículas sobre el fluido. Aunque el método Lattice Boltzmann no incluye la gestión de partículas u objetos, sí genera la información necesaria para integrarlas con éxito y obtener animaciones realistas.

3.3.1. Comportamiento partículas

Tras haber generado el campo vectorial para un escenario utilizando el método LBM, la distribución del fluido en el espacio adquiere ciertas características interesantes para evaluar su interacción con partículas u objetos móviles. Para ello, es necesario modelar el comportamiento de dichas partículas.

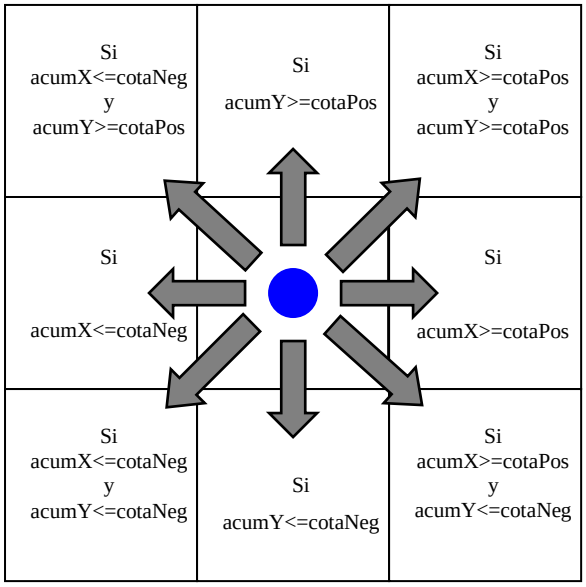
Una de las decisiones tomadas consiste en determinar que el fluido afectará directamente al movimiento de las mismas, y no viceversa. Esto simplifica considerablemente la implementación, ya que no es necesario contemplar si existen partículas o no en las fases del método LBM, como sí se hace para los obstáculos fijos mediante las condiciones bounce-back.

Tras cada iteración del modelo, y si la opción de actualizar partículas está activada, éstas deben renovar su posición en el sistema, basándose en el campo de velocidades presente en ese instante. Para ello, se ha limitado el número de celdas que puede avanzar cada partícula a una sola; es decir, si la partícula se encuentra en la celda (i, j), ésta solo puede saltar a las celdas vecinas (k,l), donde $i-1 \leq k \leq i+1$ y $j-1 \leq l \leq j+1$. Esta decisión también simplifica el cálculo de la celda destino: por una parte, es posible usar las mismas direcciones que emplea LBM; por otra, la comprobación de la validez del movimiento en cuanto a si el destino se encuentra dentro del sistema y no se trata de un obstáculo, resulta más sencilla.

En la implementación, las partículas contienen una velocidad acumulada que se utilizará para determinar cuándo y en qué dirección deben avanzar. Dicha velocidad aumenta, tras cada iteración, acumulando la velocidad que tiene el fluido en esa casilla.

Cuando alguna de las componentes de la velocidad acumulada alcanza cierta cota, la partícula se traslada a la casilla vecina correspondiente y se resetea toda su velocidad. La cota (positiva y negativa), está predefinida en cada escenario.

La obtención de la dirección o casilla a la que debe saltar es inmediata. Por ejemplo, si únicamente la componente X alcanza la cota positiva, la partícula debe saltar a su vecina derecha; mientras que, si las componentes X e Y alcanzan la cota negativa en la misma iteración, la partícula deberá desplazarse a la celda vecina inferior izquierda. En la siguiente imagen pueden observarse todas las combinaciones:



Desplazamiento de partícula a celdas vecinas.

En la decisión de salto, debe tenerse en cuenta que no exista un obstáculo en la celda destino y que ésta, se encuentre dentro de las fronteras del sistema. En caso contrario, se aplica una condición similar al Bounce-Back en LBM: la partícula quedará en la misma celda que en la iteración anterior, pero su acumulado se invertirá completamente, posibilitando que en la siguiente actualización, salte a la casilla vecina inversa.

El algoritmo simplificado de la actualización de partículas es el siguiente:

```
void LBM::actualizarParticulas(){
  para (k=0; k<nParticulas; k++) // partículas presentes en el sistema
    x=particulas[k].x;
    y=particulas[k].y;
    particulas[k].acumX+=celdas[x][y].velX;
    particulas[k].acumY+=celdas[x][y].velY;
    si (acumX≥cotaPos ||acumX≤cotaNeg ||acumY≥cotaPos ||acumY≤cotaNeg)
      si (celdas[x][y].obstaculo || !existeCelda(x,y)) //bounce-back
        particulas[k].acumX*=(acumX≥cotaPos||acumX≤cotaNeg)?-1:1;
        particulas[k].acumY*=(acumY≥cotaPos||acumY≤cotaNeg)?-1:1;
      si no // destino permitido
        particulas[k].posicionX+=(acumX≥cotaPos)?1:((acumX≤cotaNeg)?-1:0);
        particulas[k].posicionY+=(acumY≥cotaPos)?1:((acumY≤cotaNeg)?-1:0);
        particulas[k].acumX=0;
        particulas[k].acumY=0;
}
```

Existe un aspecto delicado en el algoritmo anterior: ¿qué debe ocurrir si varias partículas pretenden ocupar la misma celda? La primera posibilidad es permitir que convivan más de una partícula en la misma posición, lo cual se aplica en alguno de los escenarios y no conlleva ningún cambio en el algoritmo. La segunda, es establecer un comportamiento tras el “choque” de partículas, que consiste en que la partícula que ya se encuentra en la celda destino, absorba

la velocidad acumulada de la partícula que pretende entrar en esa casilla, de tal forma que la primera pueda continuar, mientras que la segunda permanece en la celda origen.

Para aplicar la segunda opción en algunos de los escenarios, hay que añadir al algoritmo lo siguiente:

```
…
   si (celdas[x][y].tieneParticula())
      l=celdas[x][y].particula;
      particulas[l].acumX+= particulas[k].acumX;
      particulas[l].acumY+= particulas[k].acumY;
      particulas[k].acumX=0;
      particulas[k].acumY=0;
…
```

De esta forma, en la siguiente iteración se comprobará si la partícula *l* debe desplazarse o no, quedando la partícula *k* en su celda origen.

3.3.1.1. Copos

En el escenario *Nevada* las partículas representan copos de nieve y el fluido al aire. Para mejorar el realismo del comportamiento de los copos, se ha añadido una condición de derretimiento, que elimina los copos cuando llevan cierto número de iteraciones sobre el suelo o frontera inferior del escenario. De esta forma, se evitan acumulaciones exageradas de partículas, emulando con mayor fidelidad la deposición de la nieve.

Ha sido necesario incluir un contador en la clase *ObjetoDinámico* para eliminar aquellos copos que alcancen el número de iteraciones máximo, fijado en 200 en nuestra escena.

3.3.2. Visualización de partículas en el escenario

Debido a que los objetos móviles que actualmente se han desarrollado solo pueden ocupar una celda, su visualización consiste en dibujar un punto centrado en la casilla cuya superficie es el 80% del tamaño de la celda.

El color del punto depende del escenario: en algunos se trata de un color aleatorio dentro de una gama, y en otros de un color predeterminado para cada partícula.

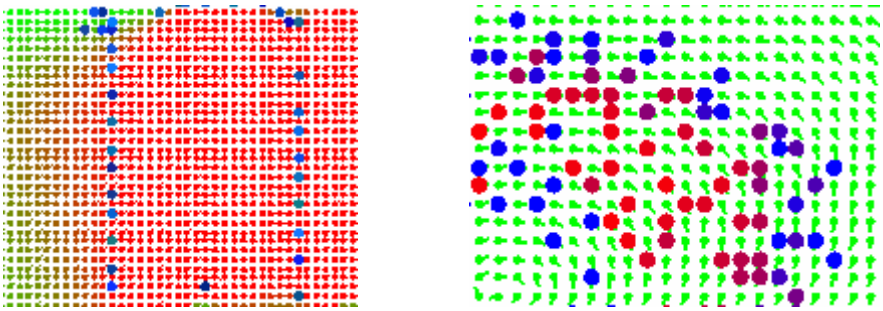
La tasa de frames por segundo (fps) de la animación de partículas depende fundamentalmente de los siguientes factores, estando los tres primeros relacionados con la complejidad del código, mientras que el último lo está con la elección de la cota usada para permitir el salto de una partícula:

- El modo de visualización escogido: en visualizaciones como *Campo de Velocidades*, deben realizarse más cálculos para dibujar los vectores, mientras que en la vista *Animación*, no existe ningún tipo de cálculo, lo que acelera considerablemente la animación.

- El tamaño del sistema: haciéndose más notable la diferencia entre modos de visualización.
- El número de partículas en el sistema.
- La elección de la cota en cada escenario: la velocidad de las partículas será inversamente proporcional a la cota, ya que a mayor cota, será necesario acumular durante más iteraciones para saltar de casilla, mientras que a menor cota, es posible que en una única iteración se desplacen a sus vecinas.

La elección de la cota en cada escenario, se detalla en el siguiente capítulo de la memoria.

Las siguientes imágenes muestran ejemplos de la visualización de partículas en el escenario:



Partículas en escenarios de distinto tamaño.

4. Experimentación

Una de las tareas fundamentales del proyecto desarrollado ha consistido en el diseño de diversos escenarios en los que se diesen ciertos fenómenos en el fluido de interés visual. Además, para lograr resultados con un alto nivel de realismo, ha sido de especial importancia la elección de parámetros adecuados en los mismos, como el tamaño del sistema, la velocidad de entrada del fluido, la cantidad de partículas a introducir y su posición, entre otros. Por ello, en este capítulo se detallan todas las decisiones tomadas en cada uno de los escenarios ideados, así como los resultados obtenidos en los mismos.

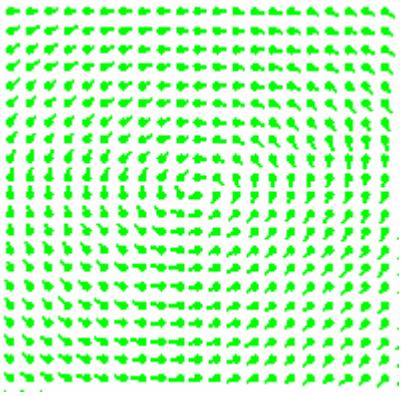
4.1. Fenómenos observados

A continuación, se describen una serie de fenómenos comunes, que aparecen en los escenarios con los que se ha experimentado durante el desarrollo del proyecto. En todos ellos se ha aplicado una viscosidad del fluido de 1,5 con la que se han obtenido resultados más vistosos, aunque puede modificarse siempre que se respete el rango [0,2) requerido por el método LBM.

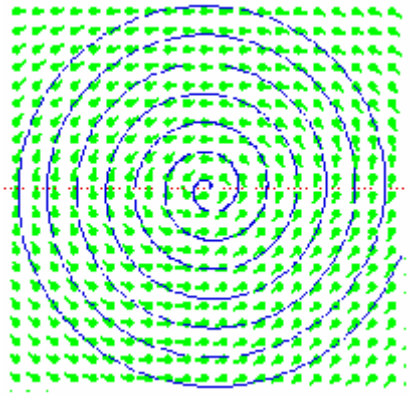
4.1.1. Vórtice

Se trata de un movimiento circular o rotatorio del fluido, en torno a un eje. Gráficamente sobre el campo vectorial del fluido, se corresponde con una espiral alrededor del centro del vórtice.

En las siguientes imágenes se muestra un vórtice en uno de los escenarios predefinidos y una espiral superpuesta al mismo vórtice, donde puede notarse una clara similitud:



Vórtice en un escenario.



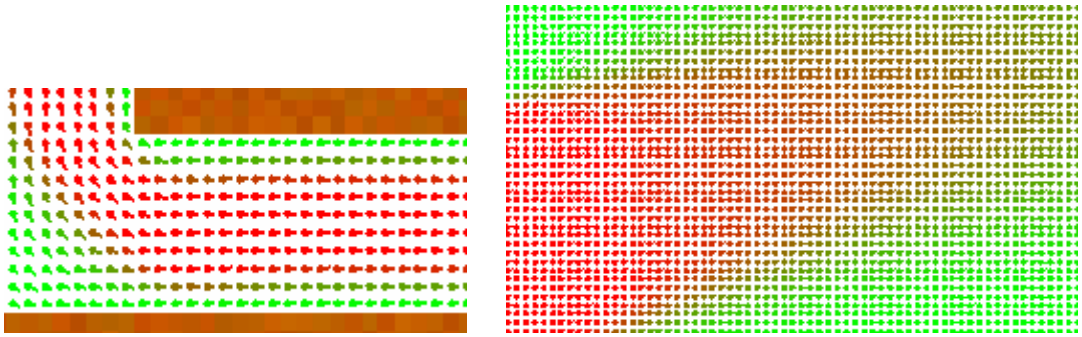
Espiral superpuesta a un vórtice.

4.1.2. Corriente

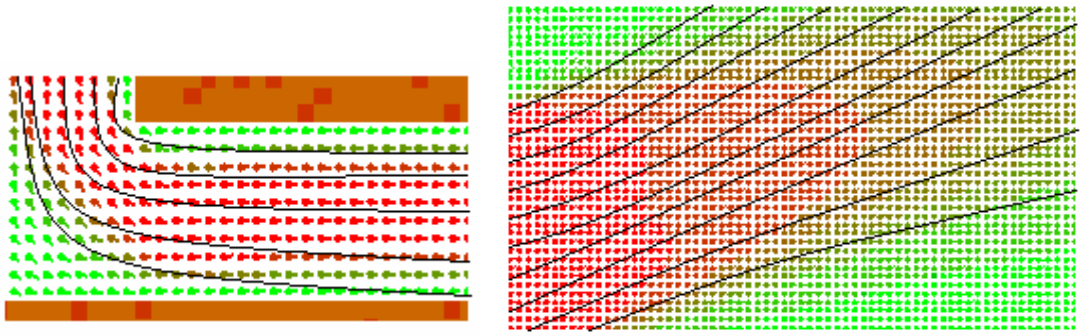
Puede entenderse como corriente un fenómeno dado en un conjunto de celdas del fluido, en el que cada una de las celdas tiene una velocidad con dirección similar a la de sus vecinas. Concretamente, tienen especial interés aquellas corrientes con magnitud de velocidad superior a la media del escenario, ya que por ellas circularán las partículas más rápidamente.

Las corrientes pueden representarse mediante líneas de corriente imaginarias, que simbolizan la trayectoria seguida por las partículas del fluido, donde la velocidad de éstas es siempre tangente a la trayectoria.

En las siguientes imágenes pueden observarse corrientes originadas en diversos escenarios predefinidos:



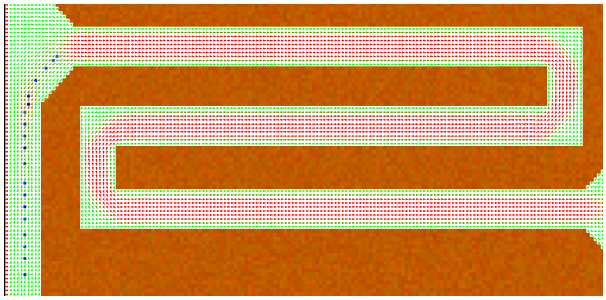
Si superponemos las líneas de corriente sobre los vectores, el resultado aproximado es el siguiente:



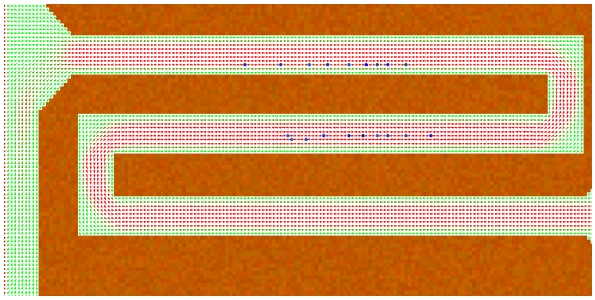
4.1.3. Comportamiento de las partículas

El comportamiento general de las partículas suele ser el de seguir la trayectoria de las corrientes, concentrándose en mayor medida en las zonas con mayor magnitud de velocidad, recorriendo éstas de forma más rápida que en el resto del escenario. De esta forma, puede verse como algunas partículas adelantan a otras, si circulan por una corriente de mayor magnitud.

En las siguientes imágenes se observa como un grupo de 19 partículas se divide en dos subgrupos de 10 y 9 partículas respectivamente, habiendo avanzado el primero más que el segundo tras 250 iteraciones:



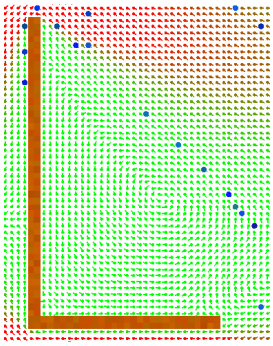
Conjunto de partículas, iteración 10.000.



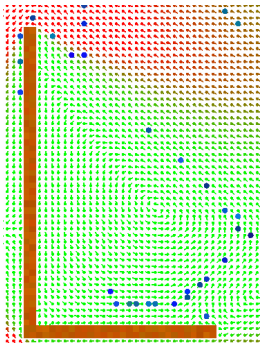
Partículas separadas, iteración 10.250.

Cuando las partículas se aproximan a un vórtice lo suficiente, entran en el mismo y, siguiendo una trayectoria circular, rodean su eje constantemente, quedando atrapadas en su zona de influencia o radio.

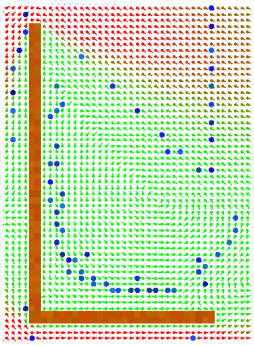
En las siguientes capturas, puede observarse como las partículas cercanas al vórtice generado tras la esquina del obstáculo, quedan atrapadas en él:



Iteración 5.070.



Iteración 5.130.



Iteración 5.270.

Si las partículas comienzan en el centro de un vórtice, éstas se alejarán de él describiendo una espiral. No obstante, puede suceder que el campo vectorial las mantenga en órbita alrededor del centro del vórtice. Esto puede explicarse mediante la existencia de dos fuerzas que se contrarrestan: la fuerza centrífuga y la centrípeta. Este efecto se observa más adelante en el escenario *Cavidad Lid-Driven*.

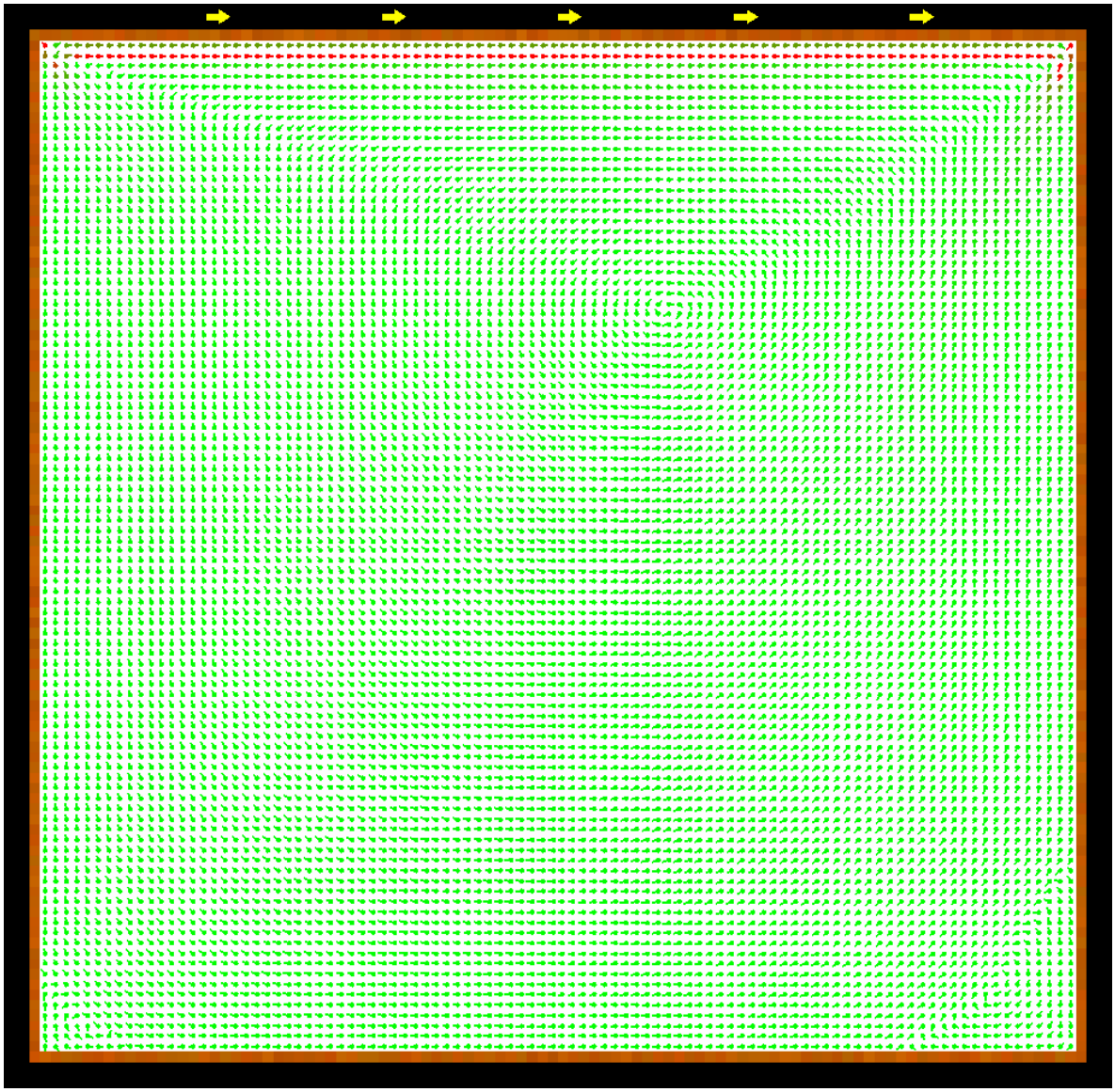
4.2. Escenarios

En los siguientes apartados se describen los escenarios diseñados y las decisiones tomadas en su implementación, así como los resultados obtenidos en los mismos.

4.2.1. Cavidad Lid-Driven

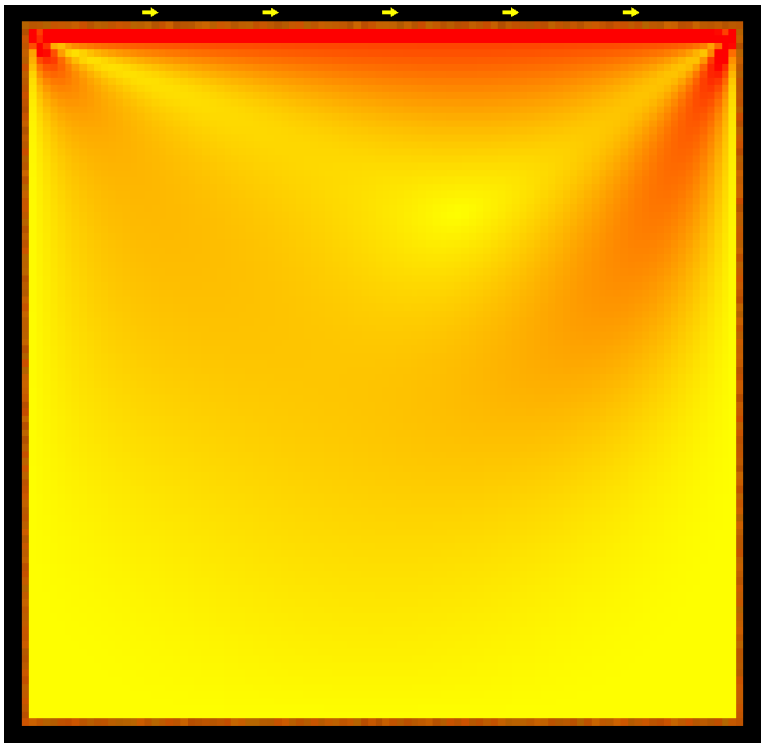
El problema de la cavidad con pared móvil se ha utilizado frecuentemente para comprobar la corrección de los resultados obtenidos en la simulación computacional de fluidos. Consiste en un escenario con paredes en todos sus extremos, completamente lleno de fluido. Las paredes izquierda, derecha e inferior, permanecen inmóviles; mientras que la pared superior mantiene una velocidad constante de izquierda a derecha. Dicha velocidad puede ajustarse desde el panel de opciones, variando entre 0,0 y 0,1.

El resultado del experimento, visualizado mediante el modo *Campo de velocidades*, en un sistema de 100 x 100 casillas, tras 10.000 iteraciones y a máxima velocidad, es el siguiente:

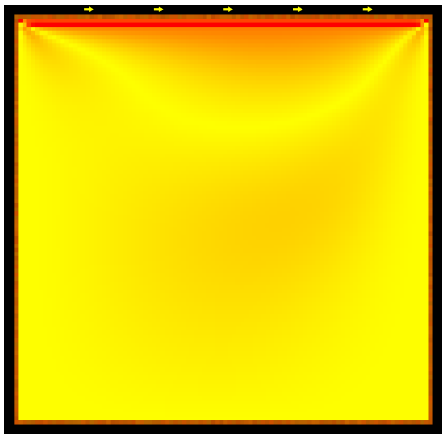


En él puede observarse un vórtice en el cuadrante superior derecho, cuya zona de influencia cubre todo el escenario. En las esquinas inferiores izquierda y derecha, también surgen sendos vórtices con menor magnitud que el principal.

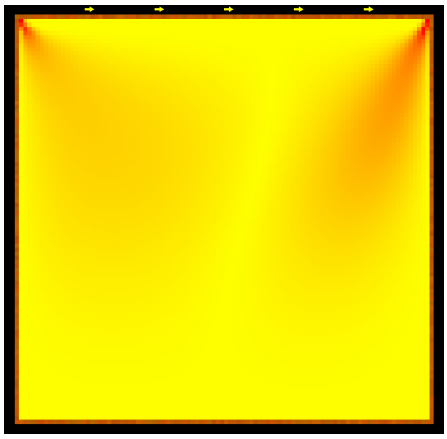
Si presentamos el escenario con los modos de visualización de velocidad, puede observarse como el centro del vórtice principal carece de velocidad, mientras que las zonas cercanas a la pared móvil tienen la velocidad máxima del sistema. Además, la velocidad X es superior en general a la velocidad Y. Los resultados son los siguientes:



Modo Módulo Velocidad.



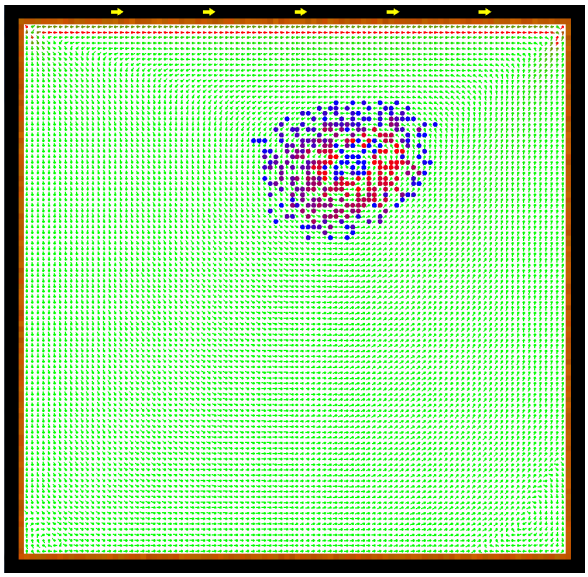
Modo Velocidad X.



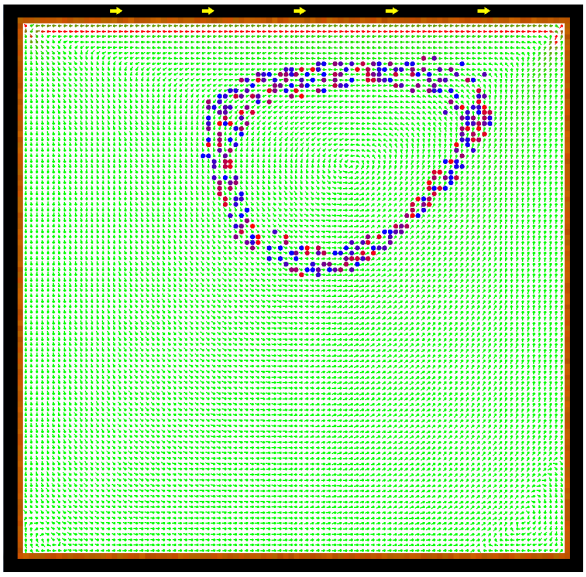
Modo Velocidad Y.

Se ha configurado el escenario para la inclusión de un total de 312 partículas en la zona cercana al centro del vórtice, con diferentes colores, que se mueven durante 100 iteraciones. La cota para el desplazamiento se ha fijado en 0,01.

Realizando una ejecución sin recalcular el LBM y con actualización de partículas, puede observarse cómo las partículas se alejan del centro del vórtice con una trayectoria espiral, hasta alcanzar cierta zona, a partir de la iteración 10.400, en la que rotan alrededor del eje permanentemente, sin alejarse ni acercarse más al mismo, como si de una órbita se tratase.



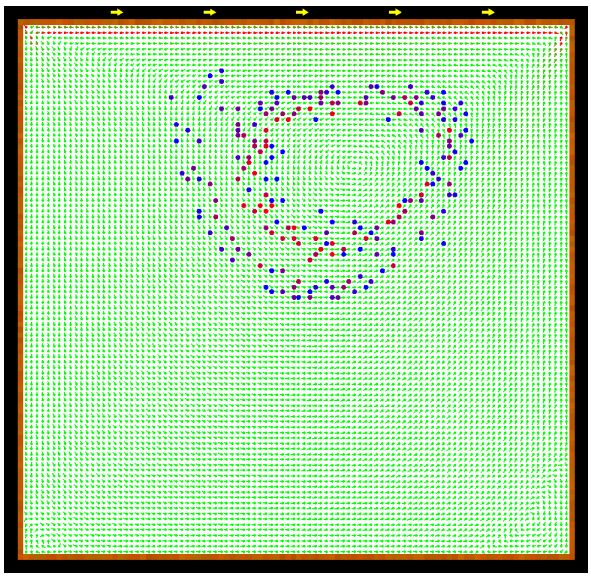
Lid-Driven con partículas recién añadidas.



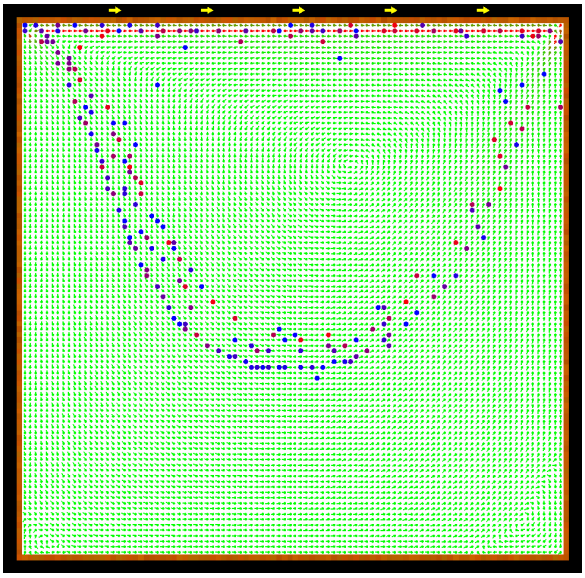
Partículas rotando alrededor del eje.

Si se aumenta la cota de desplazamiento de las partículas a 0,02, se pierde el equilibrio que las mantiene en órbita, alejándose del centro hasta chocar con las paredes.

En las siguientes imágenes, tomadas en las iteraciones 10.400 y 11.200 respectivamente, puede observarse este fenómeno:



Partículas comienzan a abandonar la órbita.



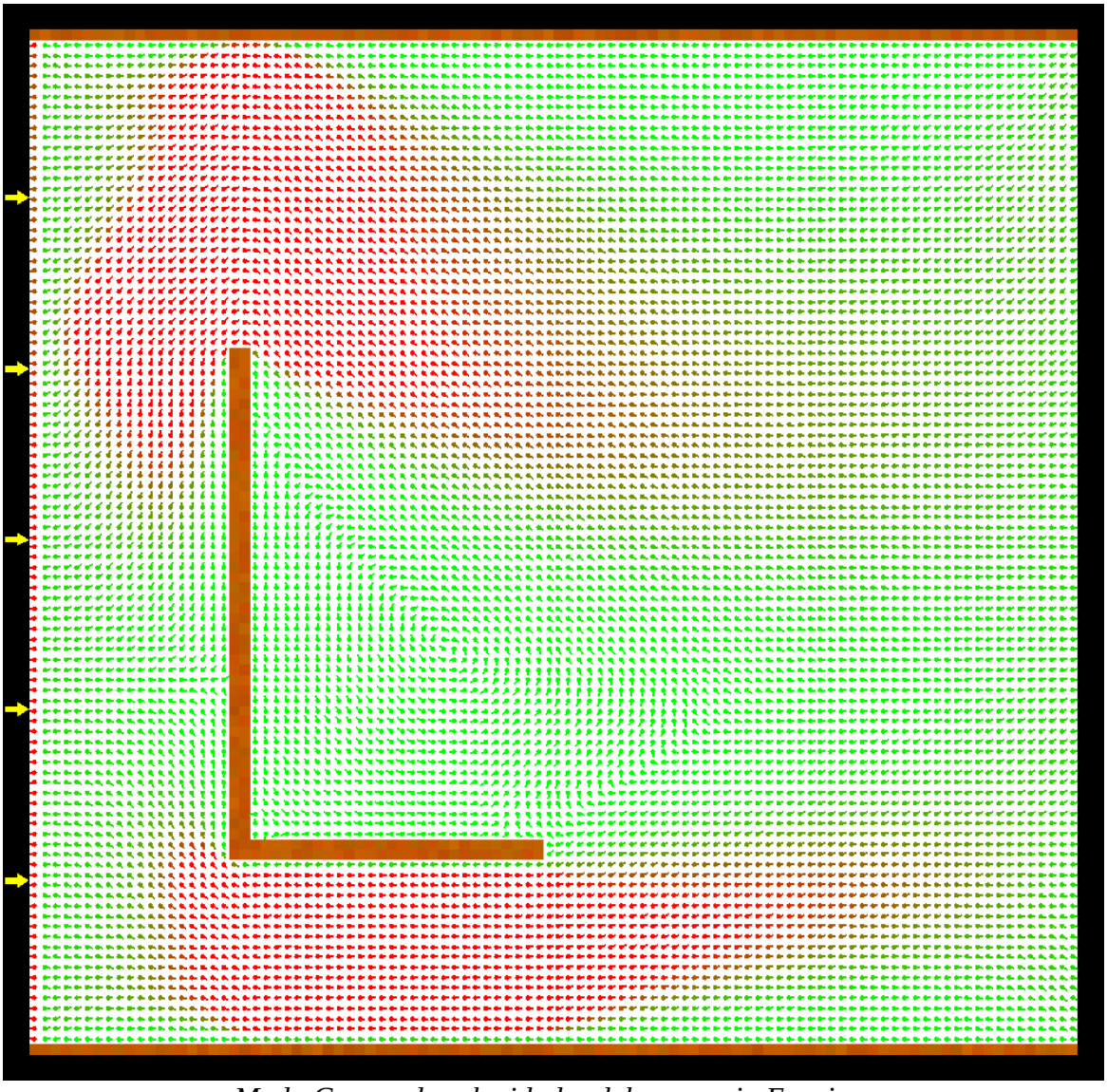
Partículas alcanzan la frontera superior.

4.2.2. Esquina

Uno de los primeros escenarios desarrollados fue la *Esquina*. Se trataba de diseñar un vórtice capaz de atraer partículas que comenzaban lejos de su centro.

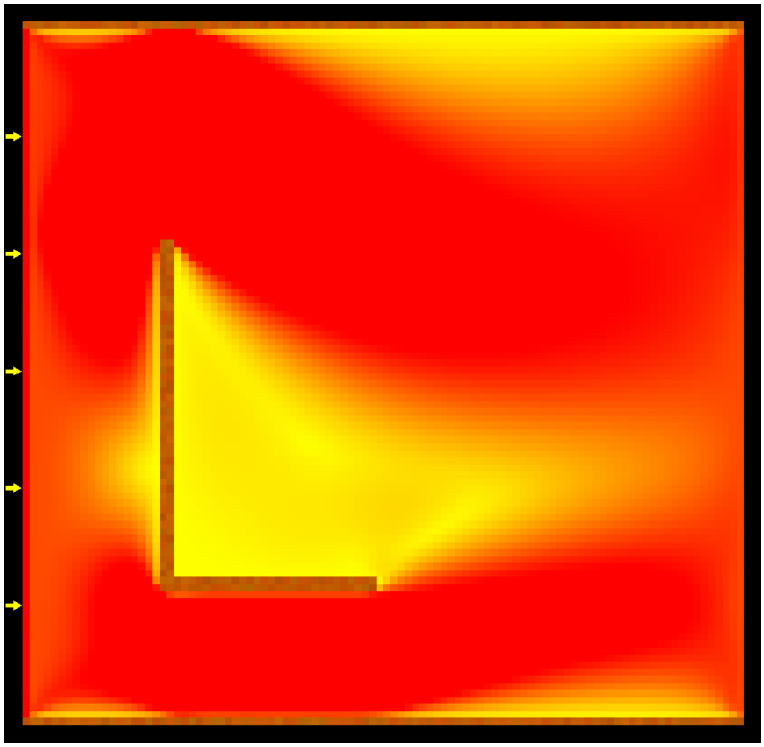
El escenario está compuesto por dos paredes fijas, superior e inferior, y un obstáculo en forma de L más cercano a la frontera izquierda del sistema. Se ha aplicado en el LBM la condición periódica en las fronteras derecha e izquierda, ya que el objetivo es simular la entrada de fluido por esta última, en dirección izquierda-derecha.

Tras 5.000 iteraciones se estabiliza el sistema, para un tamaño de 100 x 100 celdas y velocidad máxima en la entrada de fluido. En la figura pueden apreciarse dos corrientes importantes en las zonas superior e inferior del obstáculo. Además, aparece un vórtice resguardado por la esquina, que depende directamente del tamaño de ésta, y más concretamente de la proporción entre las longitudes de sus brazos vertical y horizontal.

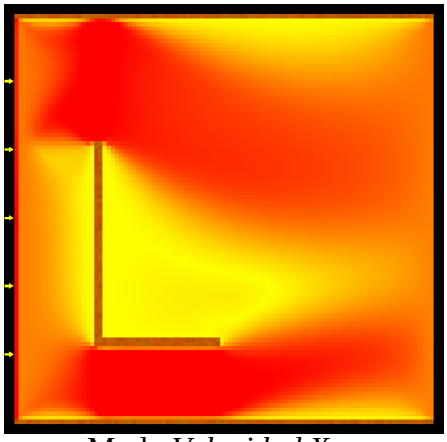


Modo Campo de velocidades del escenario Esquina.

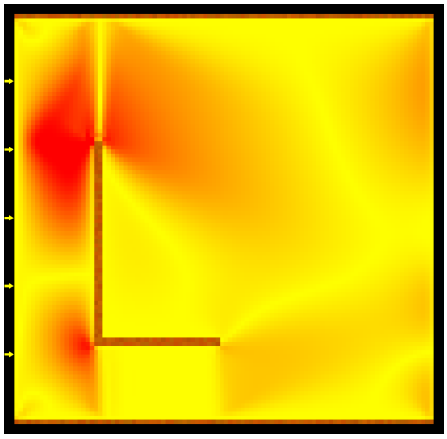
En los modos de visualización de velocidad, puede distinguirse como hay zonas con mayor actividad en el eje X o en el eje Y:



Modo Módulo Velocidad.

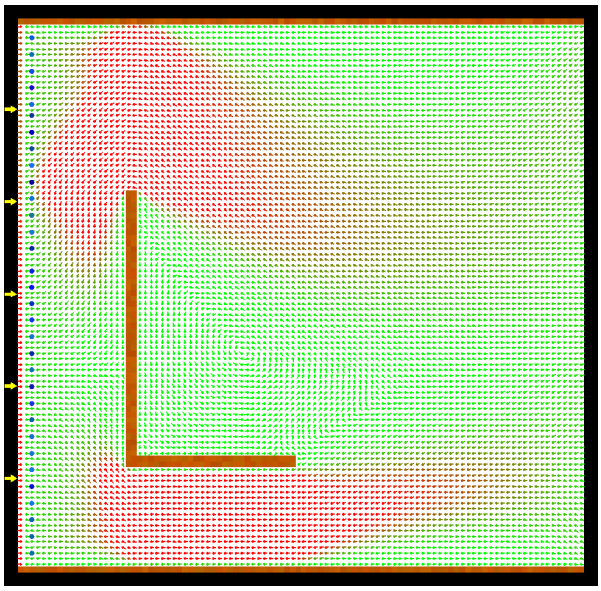


Modo *Velocidad X.*

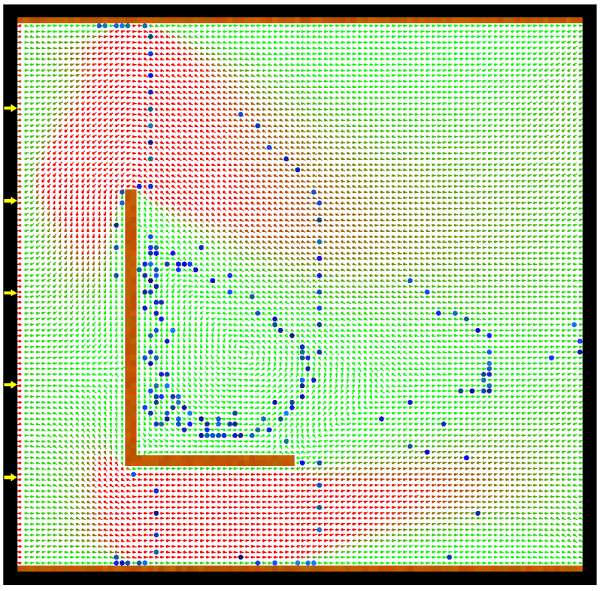


Modo Velocidad Y.

La cota para el desplazamiento de las partículas se ha fijado en 0,01 al igual que en Lid-Driven. Cada 30 iteraciones, se añaden partículas a lo largo de la frontera izquierda del sistema, cada 3 celdas, para que sean arrastradas por la corriente. Estos son los resultados:

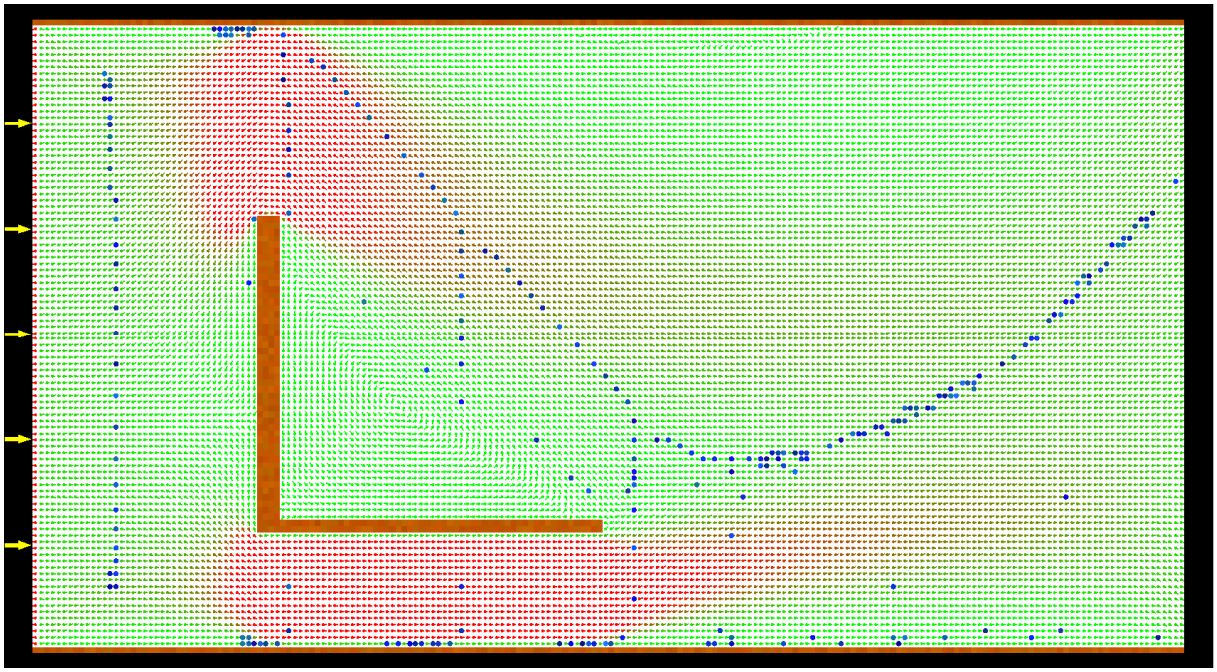


Partículas entrando en el sistema.

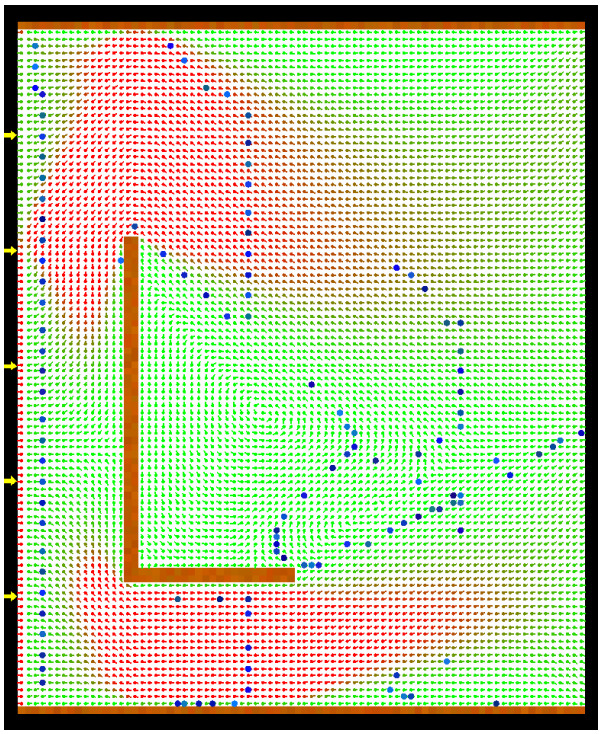


Vórtice capturando partículas.

Durante el desarrollo de este escenario se comprobó, como se ha indicado anteriormente, que la proporción entre las longitudes de los brazos que conforman la esquina, afecta directamente a la generación del vórtice. De esta forma, si se aplica el escenario *Esquina* a un sistema con 200 x 100 casillas, en el que la longitud del brazo horizontal se aproxima a la del vertical, se sigue generando un vórtice pero las partículas no llegan a entrar en él:



Por el contrario, en un sistema de 80 x 100 celdas, en el que el brazo vertical es todavía más largo que el horizontal, las partículas sí entran en el vórtice, pero salen poco después debido a un segundo vórtice:



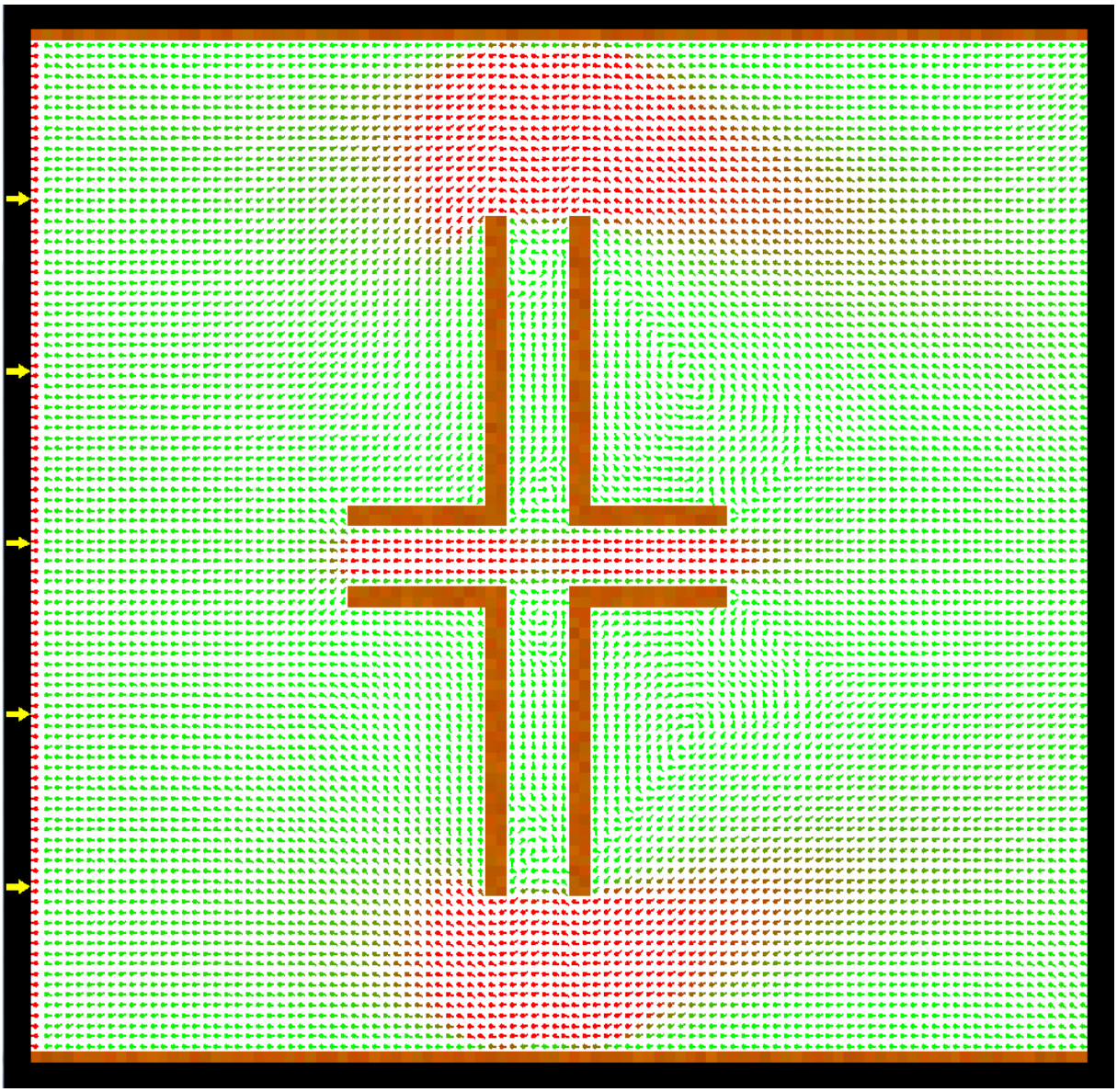
4.2.3. Cruz

Como una evolución del escenario *Esquina*, la *Cruz*, que no es más que 4 esquinas enfrentadas entre sí, centradas en el escenario y con paredes y fronteras iguales que en su predecesor, ofrece más vórtices y corrientes en el fluido.

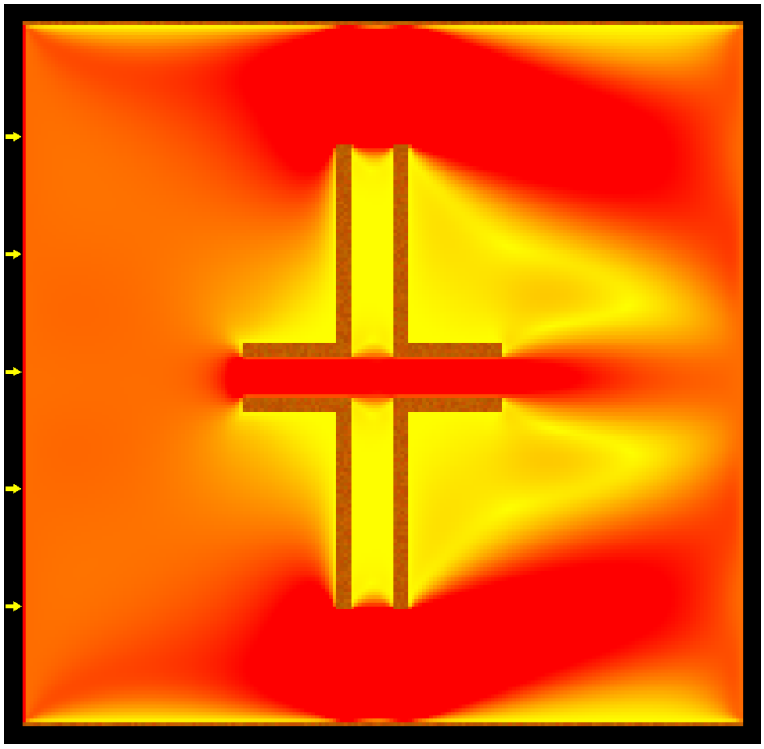
Además de las corrientes superior e inferior, destaca la corriente central en el pasillo horizontal, en la que el fluido pierde cierta velocidad en la intersección con el pasillo vertical.

Con un tamaño mínimo de 200 x 200 celdas para que puedan generarse los vórtices con suficiente nivel de detalle, este escenario requiere al menos 5.000 iteraciones para alcanzar cierta estabilidad usando la velocidad máxima en la entrada del fluido.

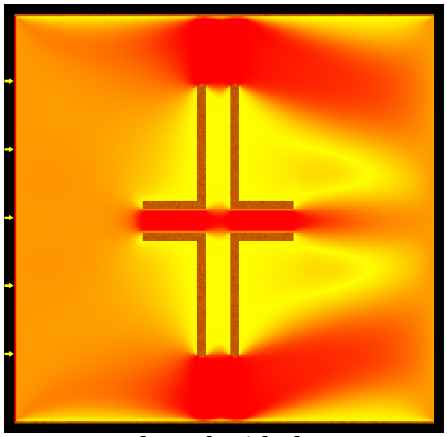
A continuación se muestra aplicado a un sistema de 100 x 100 casillas, en el que pueden apreciarse los detalles:



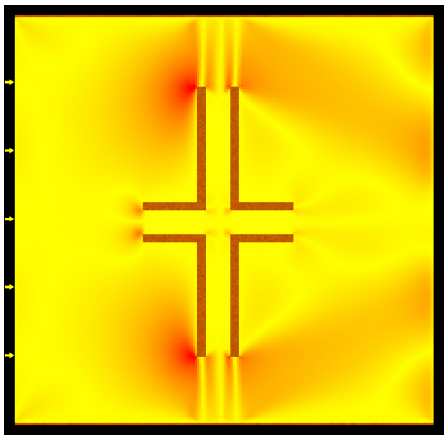
Si se observa mediante los modos de visualización de velocidad, predomina la del eje X debido a la entrada de fluido por la frontera izquierda en dirección hacia la derecha:



Modo Módulo Velocidad.

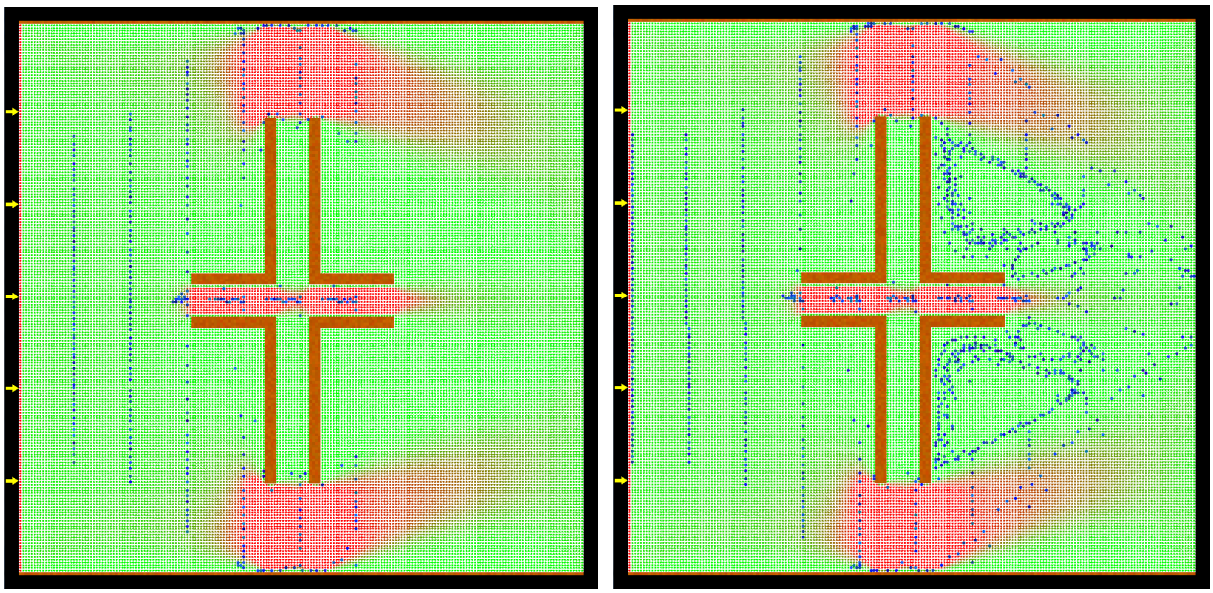


Modo Velocidad X.



Modo Velocidad Y.

Introduciendo las partículas en el escenario de la misma forma que en la *Esquina*, pero concentrándolas en el centro del sistema y cada 20 iteraciones, y fijando la cota de desplazamiento en 0,01, se obtienen 4 vórtices:

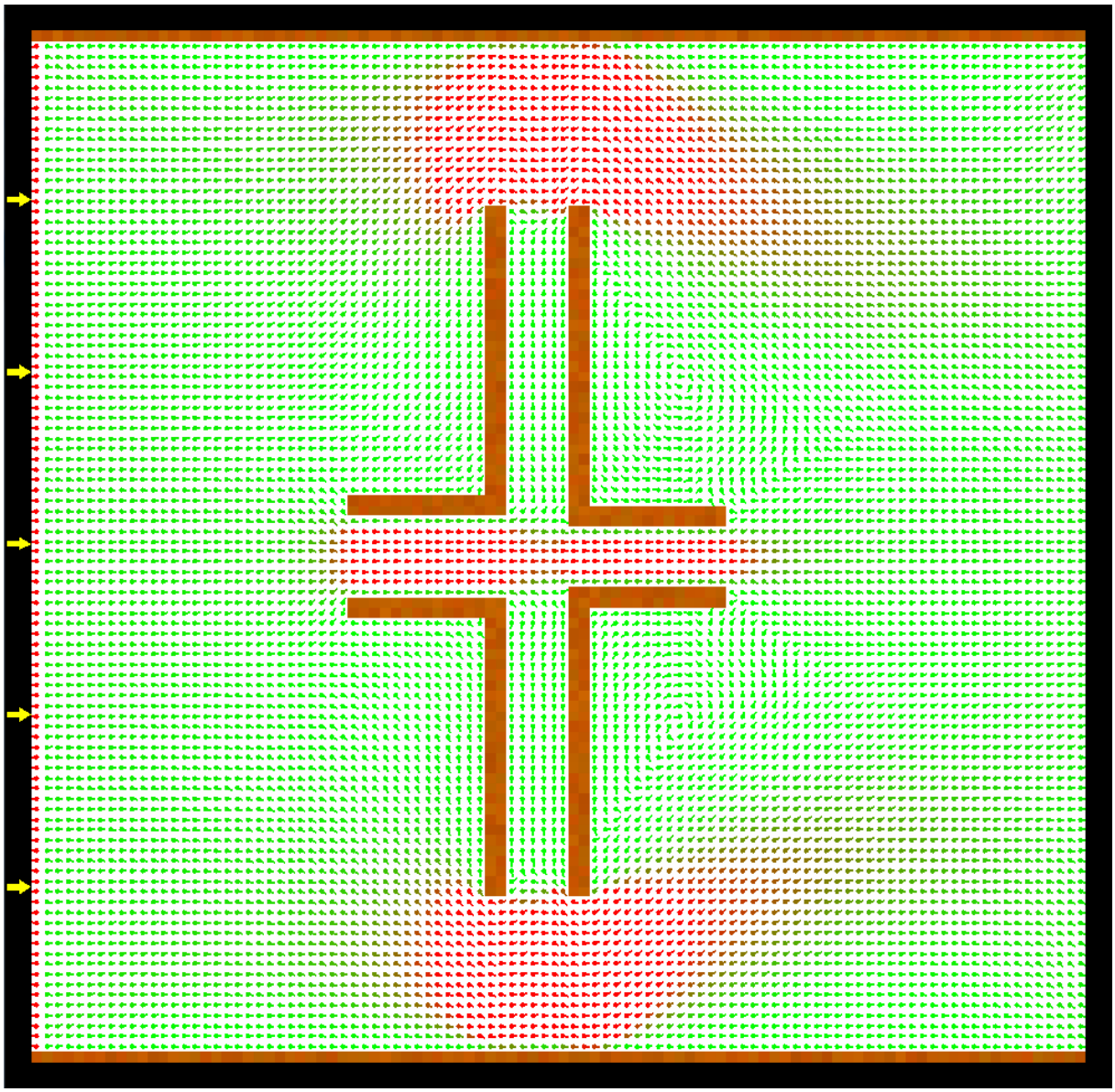


4.2.4. Cruz asimétrica

Este escenario es una modificación de la *Cruz*, en la que el pasillo horizontal se estrecha en su salida, buscando nuevos efectos en el comportamiento de las partículas.

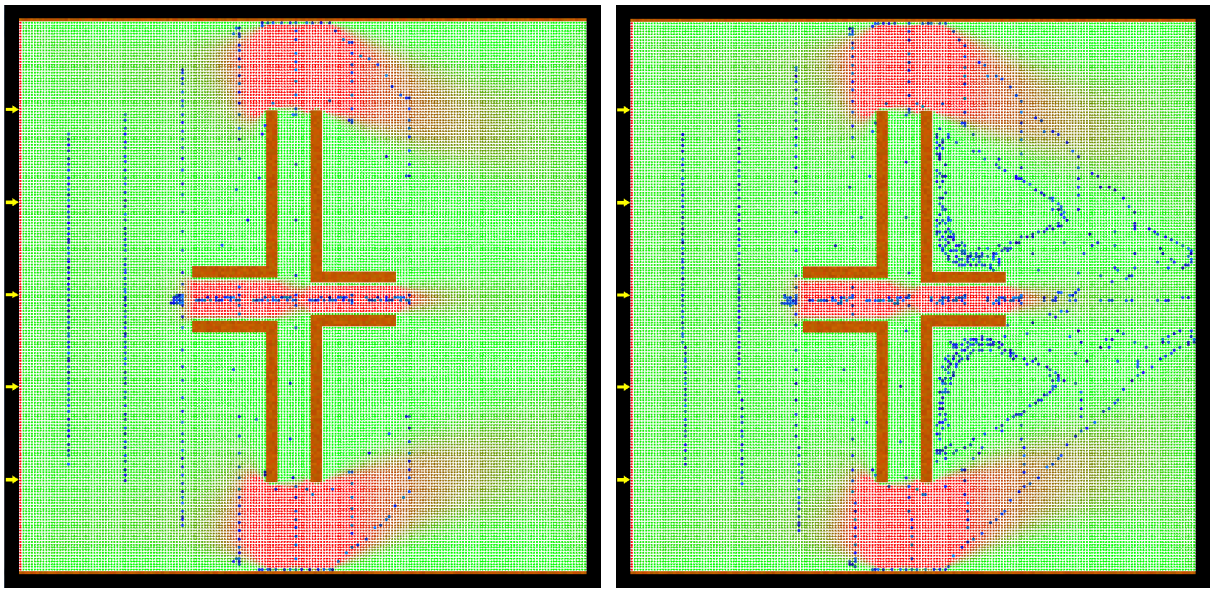
Las condiciones del escenario deben ser las mismas que en la *Cruz*, tamaño 200 x 200, velocidad máxima del fluido en la entrada, y 5.000 iteraciones para estabilizarse. La cota de desplazamiento de las partículas se mantiene en 0,01.

En la siguiente imagen puede observarse el campo de velocidades de este escenario, aplicado a un sistema de 100 x 100 casillas:



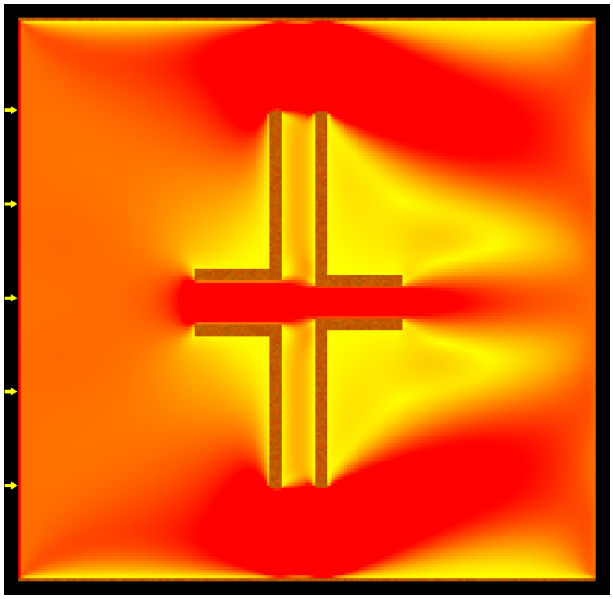
La diferencia principal con el escenario anterior, es que los pequeños vórtices que aparecían en el pasillo vertical, desaparecen, creándose corrientes de poca magnitud en las direcciones superior e inferior.

En estas capturas puede observarse el comportamiento de las partículas:



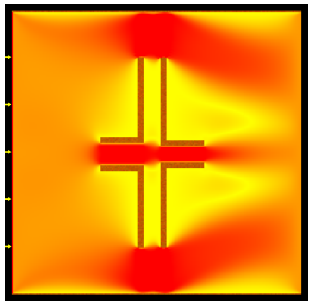
Se distinguen dos diferencias con respecto al escenario *Cruz*. La primera de ellas consiste en que cierta cantidad de partículas que se desplazan por el pasillo horizontal, se dirigen a los pasillos verticales, rodeando el obstáculo por su interior, al contrario que en la cruz original, en la que todas las partículas que entraban en dicho pasillo, salían por el otro extremo del mismo. La segunda se trata de que las partículas que rodean la cruz por su exterior, siguen formando los cuatro vórtices, pero en los dos más pequeños, la cantidad de partículas atrapadas es mucho menor, debido a que el campo vectorial varía ligeramente en esa zona.

Los modos de velocidad muestran resultados similares, aunque se distingue una mayor superficie en la entrada al pasillo central, con velocidad máxima del fluido:

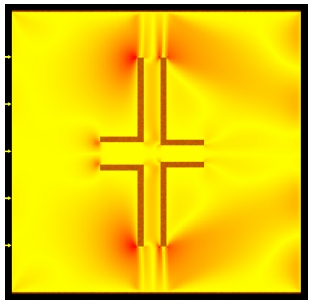


Modo Módulo Velocidad.

Modo Velocidad X.



Modo Velocidad Y.

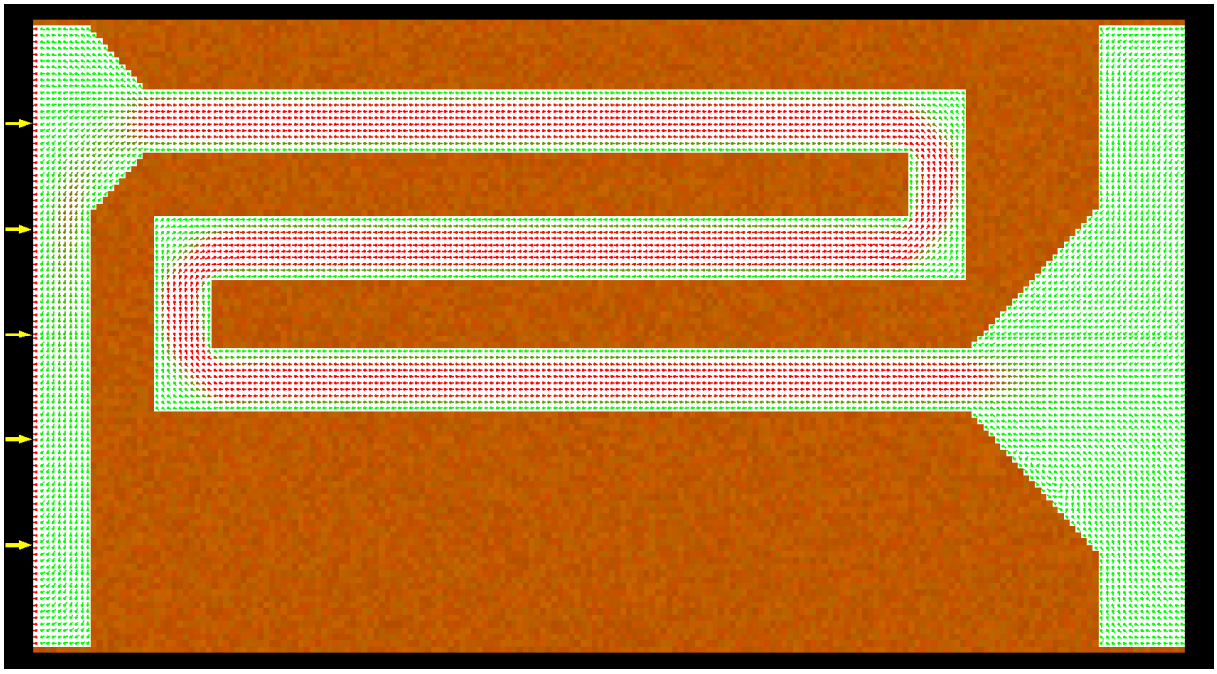


4.2.5. Tubería

Este escenario se diseñó con la idea de generar una corriente en la que el fluido tuviera que cambiar de dirección en varias ocasiones a lo largo de su recorrido. Con intención de simular una tubería con agua o un conducto de aire, se han añadido obstáculos al escenario que conforman una especie de tubo.

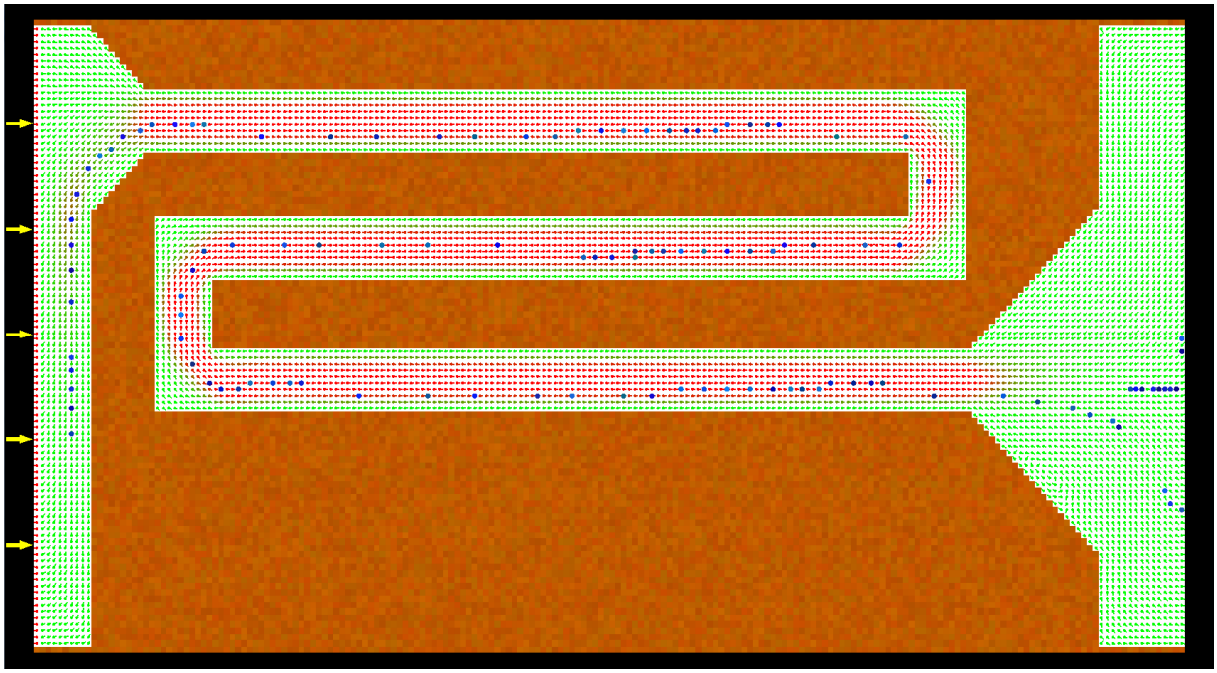
El fluido se introduce por la frontera izquierda del escenario, en dirección derecha, con una condición periódica en ambas fronteras. De esta forma, se genera una corriente que dibuja exactamente el recorrido del conducto, describiendo una curva en cada una de sus esquinas. Además, se dan mayores velocidades en el centro del conducto que en las zonas cercanas a las paredes del mismo.

El campo de velocidades que describe el fluido en un sistema de 200 x 100 celdas, es el siguiente:



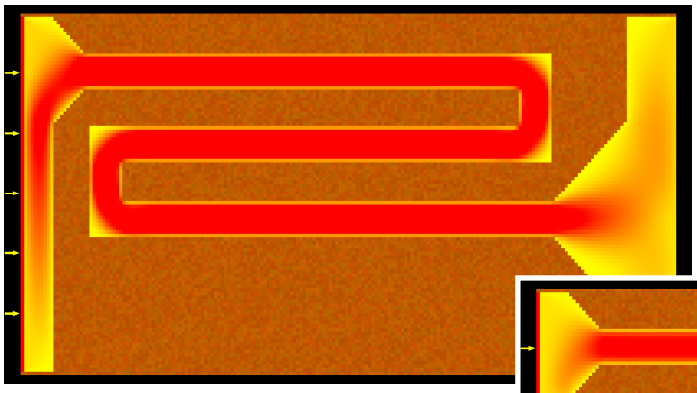
El comportamiento de las partículas se debe fijar con una cota de 0,1, para obtener los resultados más vistosos, ya que para cotas inferiores, todas las partículas recorren exactamente las mismas casillas (aquellas con mayor velocidad); y con superiores, la animación es muy lenta y las partículas se acumulan en cierto grado en las esquinas del conducto.

Las partículas se introducen en el fluido cerca de la frontera izquierda del sistema, cada 100 iteraciones, colocadas cada dos casillas, de tal forma que al entrar en el tubo, comienzan prácticamente en hilera.



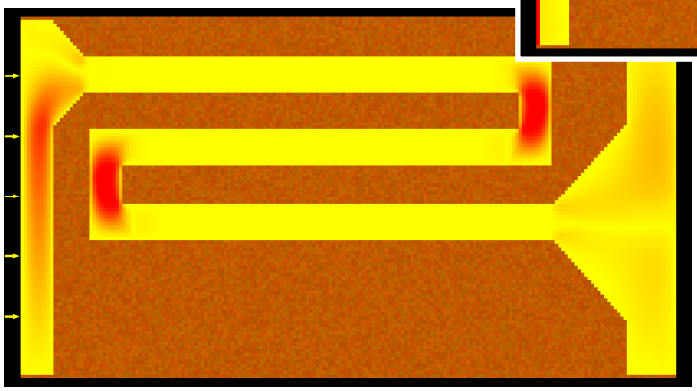
Partículas en Tubería 200 x 100, recorriendo toda su extensión.

En los modos de visualización de velocidad, puede observarse como la mayor magnitud se encuentra a lo largo de la tubería, y como las velocidades X e Y predominan en los tramos horizontales y verticales del conducto, respectivamente:



Modo Módulo Velocidad.

Modo Velocidad X.



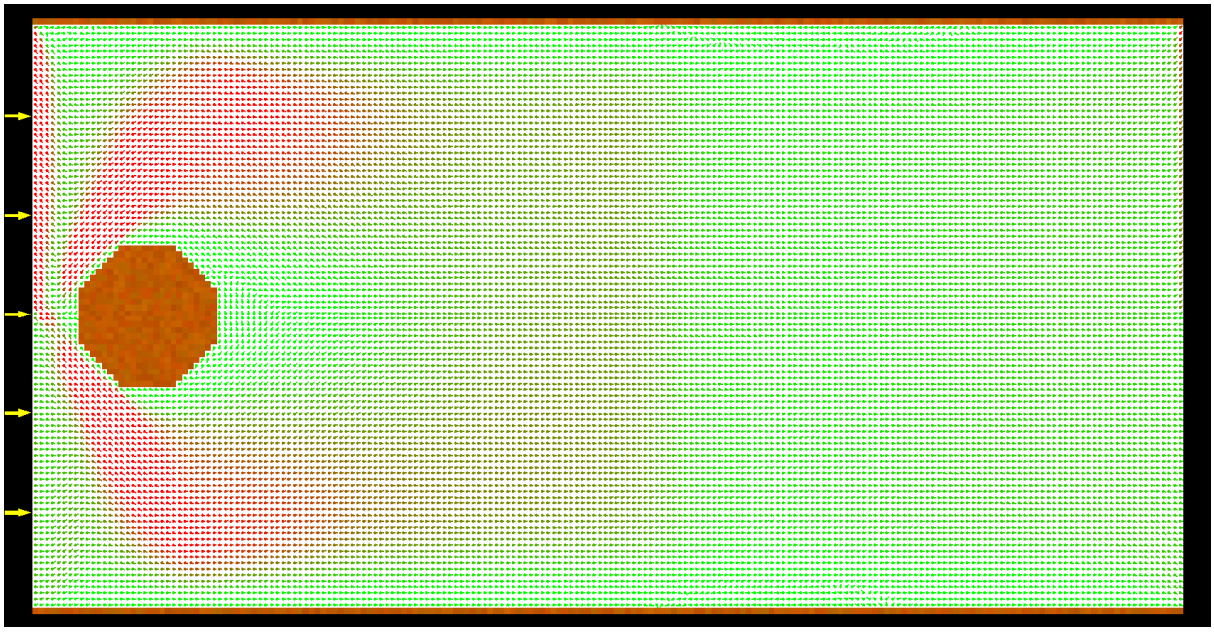
Modo Velocidad Y.

4.2.6. Vórtices Karman

Los vórtices de Von Karman constituyen un fenómeno físico muy conocido en mecánica de fluidos, que se da en numerosas situaciones al introducir un obstáculo en el curso de un flujo determinado. Este obstáculo puede tratarse de una roca en un río, e incluso de una isla en un mar u océano; o bien del pico de una montaña en la atmósfera (ver referencia bibliográfica **[19]**).

Utilizando el método desarrollado durante el proyecto, se ha creado un escenario sencillo, en el que se establece un flujo constante de izquierda a derecha, con un único obstáculo con forma octogonal, centrado verticalmente y próximo a la frontera izquierda del sistema. En sus fronteras derecha e izquierda se aplica la condición de frontera periódica para simular la continuidad del flujo. El objetivo fundamental consiste en observar la reacción del fluido al encontrar dicho obstáculo, y su influencia en las partículas que puedan flotar en el mismo.

En la siguiente captura se muestra el escenario en la iteración 500, para un tamaño de 200 x 100 celdas y velocidad del sistema al 50%:

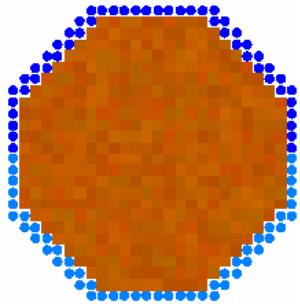


Vórtices Karman en modo Campo de velocidades.

Como puede observarse en la imagen, la velocidad en la zona superior izquierda del campo de velocidades presenta una mayor magnitud en dirección hacia abajo. Esto se debe a la forma en la que se introduce el fluido en el escenario por la frontera izquierda: durante 50 iteraciones se hace por la mitad superior, y en las siguientes 50 se introduce por la mitad inferior, alternándose constantemente. Gracias a esto, el campo de velocidades cambia durante la ejecución de iteraciones, produciendo vórtices intermitentes en las zonas superior e inferior del escenario.

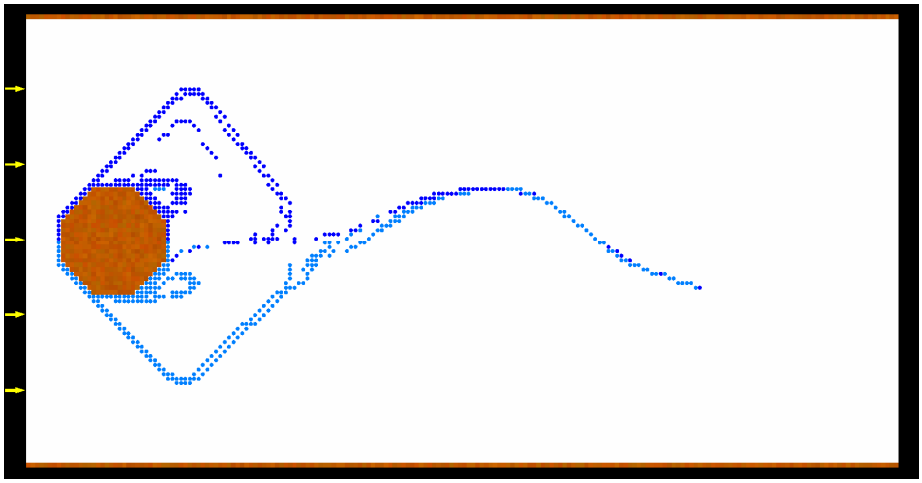
Es importante destacar que, para la generación de los vórtices, la velocidad del fluido entrante debe ser moderada. Si es muy alta, no da tiempo a generarse con suficiente intensidad, mientras que si es baja, no puede apreciarse su efecto en el fluido.

Las partículas, que podrían representar espuma sobre el agua, o nubes en la atmósfera, se introducen constantemente en el sistema alrededor del obstáculo. Se seleccionan colores diferentes para las mitades superior e inferior, a fin de distinguir su procedencia durante la animación:

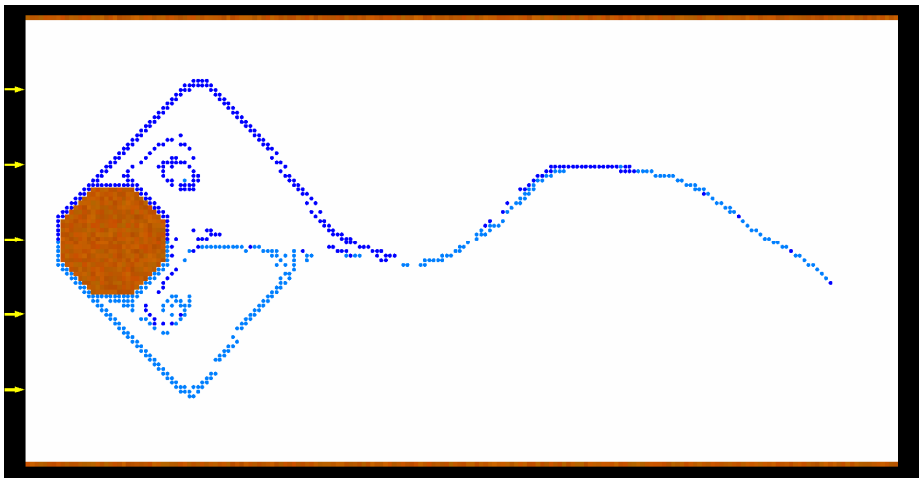


Introducción de partículas en Vórtices Karman.

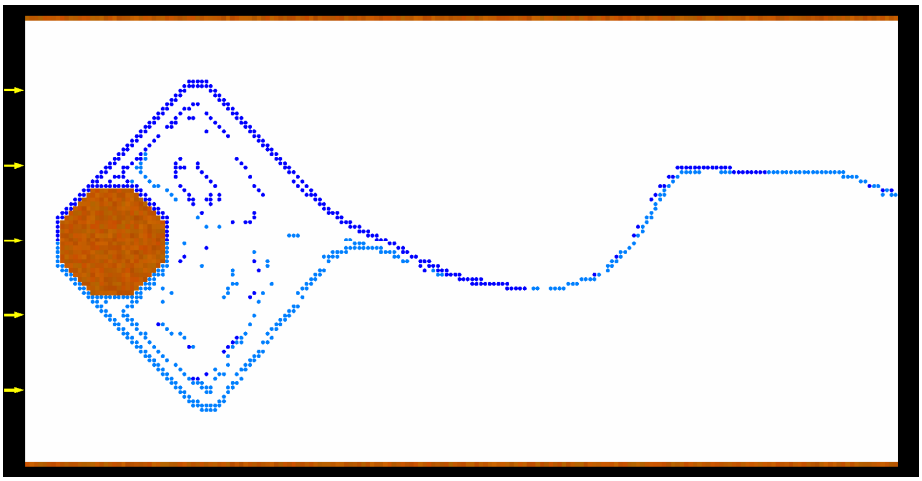
Aunque en el campo de velocidades no llegan a distinguirse los vórtices, al introducir las partículas en él, éstas sí describen las trayectorias esperadas:



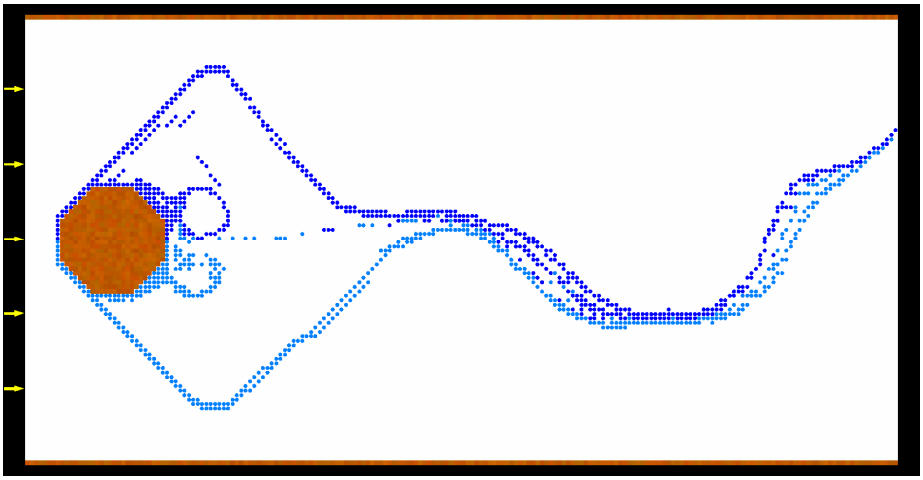
Iteración 640.



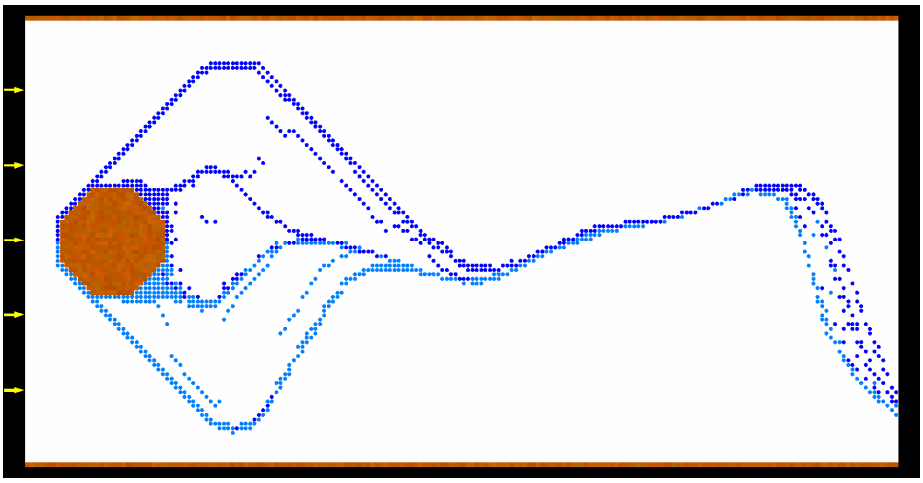
Iteración 670.



Iteración 700.



Iteración 830.



Iteración 890.

Inmediatamente detrás del obstáculo, se generan dos vórtices con intensidades diferentes, desfasados entre sí en cierto grado, que atrapan las partículas durante un cierto número de iteraciones. Además, las partículas siguen una trayectoria oscilatoria en la zona derecha del escenario, formando una curva curiosa.

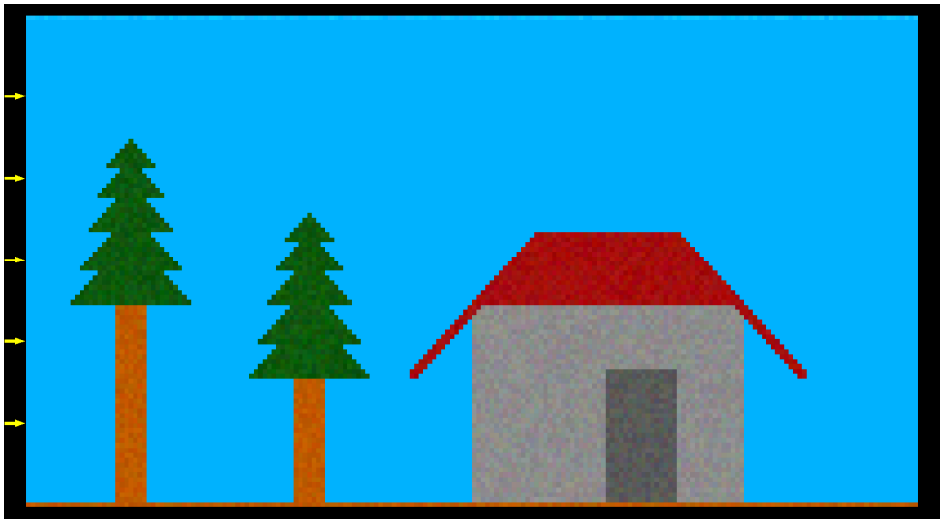
Buscando que el resultado sea lo más vistoso posible, se ha fijado la cota de desplazamiento de las partículas en 0,008, permitiendo a las mismas convivir en una misma celda. Una vez llegan a la frontera derecha del sistema, se eliminan para evitar acumulaciones simulando la continuidad del fluido.

4.2.7. Nevada

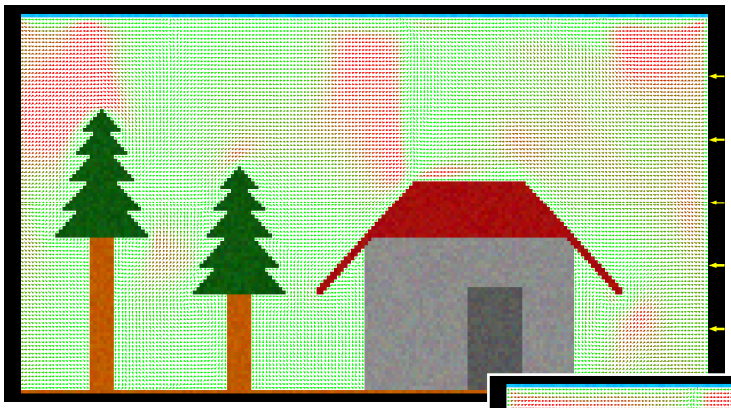
Con el objetivo de simular una escena con nieve, se crea este escenario con obstáculos con forma de árbol o casa, en el que el fluido simula el aire.

Para aumentar la complejidad del campo de velocidades y simular cierta aleatoriedad en las corrientes de viento, se introduce el mismo por las fronteras derecha e izquierda en direcciones izquierda y derecha respectivamente, durante 150 iteraciones cada dirección, para que llegue a notarse un efecto intermitente en el sistema.

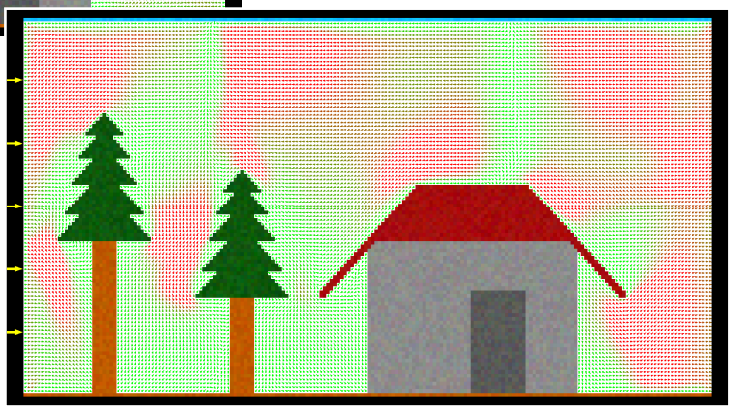
Para un tamaño de 200 x 100 celdas, la escena es la siguiente:



Debido a la variación en la dirección y origen de la entrada de viento, el campo de velocidades varía a lo largo de la ejecución de iteraciones, como puede observarse en las siguientes capturas:

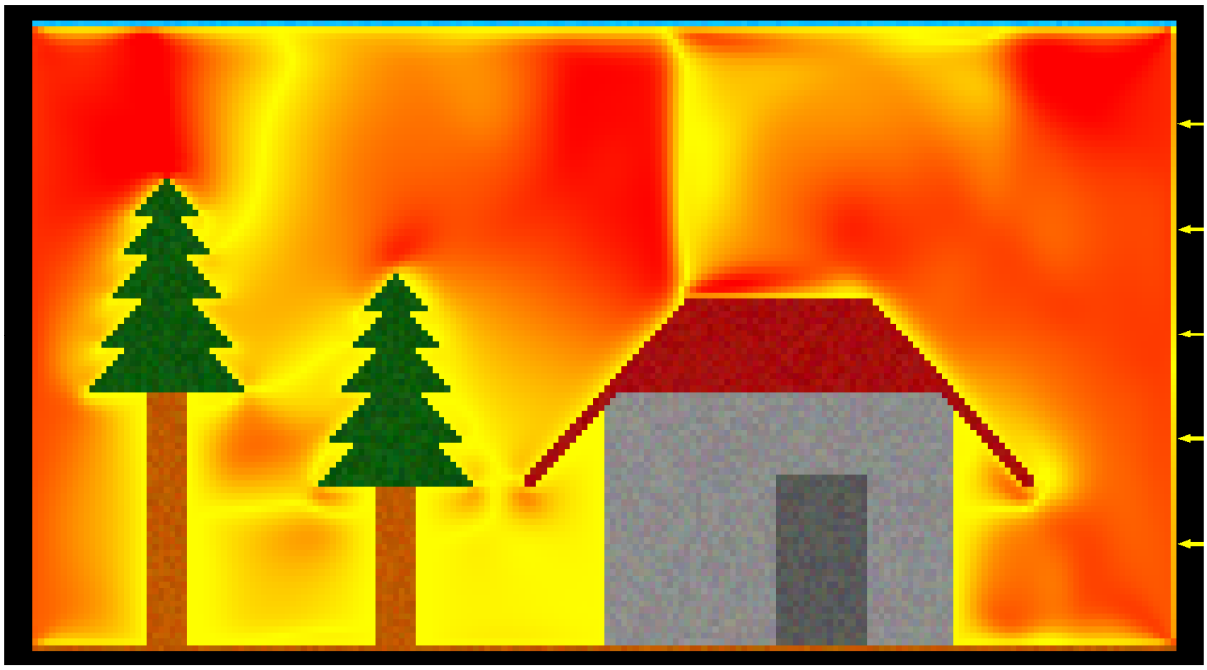


Campo de velocidades iteración 600.

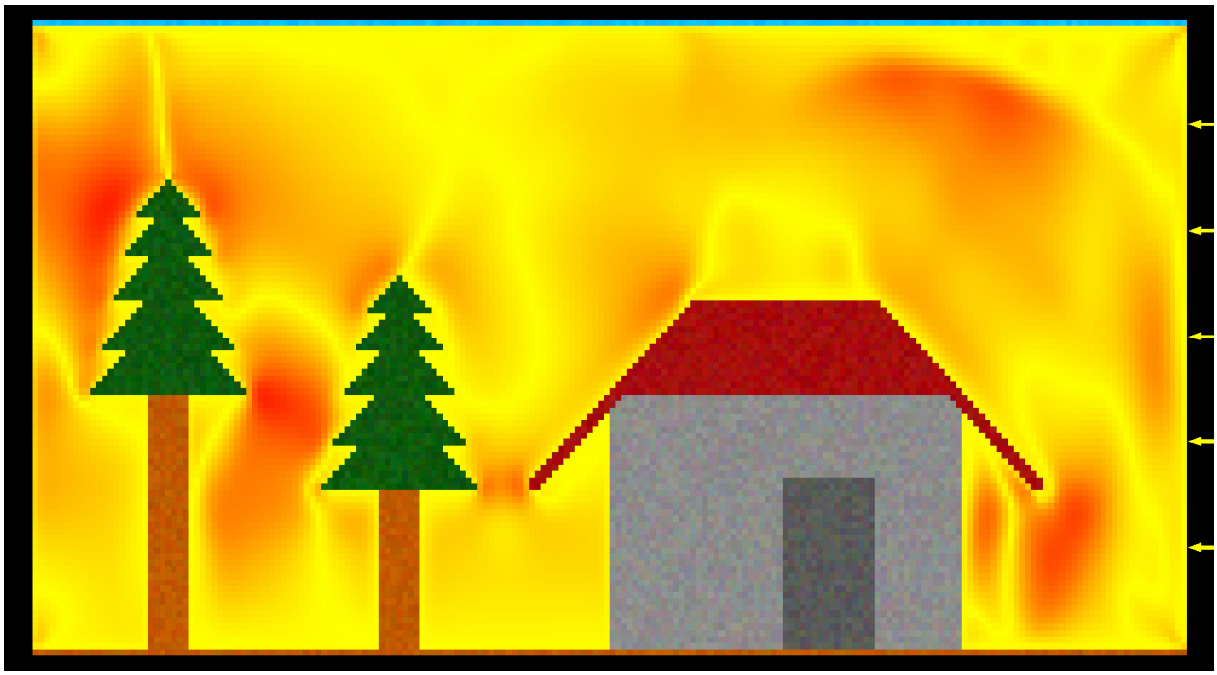


Campo de velocidades iteración 700.

Puede observarse en las vistas de velocidad en un instante dado, que predomina la velocidad X debido a la entrada de viento al sistema; no obstante, la presencia del componente vertical en la velocidad no es despreciable:



Modo visualización Velocidad X del escenario Nevada, iteración 600.



Modo visualización Velocidad Y del escenario Nevada, iteración 600.

El comportamiento de las partículas o copos de nieve en este escenario difiere del resto, ya que se ha incluido el efecto de la gravedad en los copos. Debido a que el campo vectorial del aire solo da información sobre el comportamiento del fluido, al que no afecta la gravedad a este nivel, es necesario resolver este fenómeno en la animación de los copos para conseguir un resultado mucho más realista. Para ello, en la actualización de las partículas tras cada iteración, se suma a su acumulado en la velocidad Y una constante calculada en base a la cota de desplazamiento, en sentido negativo.

Los resultados más vistosos se han obtenido con la constante igual a $\frac{1}{3}$ de la cota, la cual se ha fijado en 0,3, lo que corresponde a 0,1. Con una cota superior, la velocidad de los copos y la animación en general era muy lenta, distando mucho de la realidad; mientras que con una cota menor, la velocidad era excesiva, por lo que los copos "ignoraban" los fenómenos que se producen en el campo de velocidades.

Los copos se introducen en posiciones aleatorias de la frontera superior del sistema, en todas las iteraciones, logrando un efecto bastante realista. Además, usando la condición periódica, se logra la aparición de copos en las fronteras derecha e izquierda, provenientes de la frontera inversa, cuando son empujados por la corriente de aire, lo que supone un mayor realismo.

La condición de derretimiento explicada en el apartado *Comportamiento de partículas*, permite que en el suelo no se acumule gran cantidad de copos, simulando que los mismos se derriten individualmente, pasado cierto tiempo.

En la siguiente captura tomada en la iteración 600, puede observarse la escena con los copos sobre los obstáculos y depositándose en el suelo:

5. Trabajo futuro

En los siguientes apartados se describen tareas que podrían abordarse en futuros desarrollos para ampliar el proyecto o aplicar algunas alternativas de diseño.

5.1. Ampliación a 3D

Una posible ampliación del proyecto consistiría en extender los conceptos del método Lattice Boltzmann a 3 dimensiones, para lo cual podrían usarse las configuraciones D3Q19 ó D3Q27.

Para ello, deberían adaptarse todos los métodos correspondientes a cada una de las fases del modelo, además de ampliar las matrices de 2 a 3 dimensiones, añadiendo las direcciones e_i necesarias según la configuración elegida.

Además, el formulario tendría que mostrar un escenario en 3D, para lo cual habría que seleccionar una representación del fluido que permitiese identificar fenómenos como los vórtices o corrientes, además de seguir el comportamiento de las partículas presentes en el fluido.

Debe tenerse en cuenta que, al ampliar tanto el modelo como la vista a 3D, la complejidad de la mayor parte de los métodos, que ahora puede simplificarse en $O(n \times m)$, donde *n* es el número de celdas en el eje X y *m* en el eje Y, pasaría a estar en $O(n \times m \times p)$ siendo *p* el número de casillas del eje Z, lo cual para $n \approx m \approx p$, supone un orden cúbico. Por este motivo, se recomienda desarrollar esta ampliación en conjunto con la siguiente propuesta.

5.2. Implementación en CUDA

La ventaja principal de las GPU's frente a las CPU's de propósito general, reside en el paralelismo que ofrecen sus múltiples núcleos. CUDA aprovecha la gran cantidad de hilos que pueden lanzarse con ejecución simultánea, para obtener un gran rendimiento en aquellas aplicaciones que se ajustan al modelo de programación de streams.

Si en el modelo LBM se aplica el enfoque *pull* en la fase de propagación, como se ha explicado en el capítulo de *Implementación*, se pueden realizar tantas tareas independientes como celdas tenga el sistema debido al carácter local de las operaciones. Esto supone en definitiva acelerar el rendimiento de la aplicación al aprovechar la ejecución en múltiples núcleos.

En la referencia bibliográfica **[1]**, puede obtenerse información sobre cómo aplicar el LBM en CUDA, además de conocerse los resultados obtenidos por este grupo de investigadores de la Universidad Politécnica de Cataluña.

5.3. Editor de escenarios

Una posibilidad para ampliar la interacción con el usuario, sería la de permitir generar escenarios con nuevos obstáculos, definiendo características como la entrada del fluido, su dirección, el comportamiento de las partículas, y su posición inicial en la escena. Para ello sería necesario un editor en el que pudiesen diseñarse estos elementos sobre el propio escenario, sin necesidad de conocer cómo está implementada la aplicación. Otra alternativa sería permitir modificar los escenarios ya predefinidos. Esto ampliaría en gran medida la funcionalidad de la aplicación, abriéndose a nuevos experimentos con fluidos.

5.4. Realismo de la animación

Como se ha explicado en la sección de *Animación de partículas*, actualmente el comportamiento de las mismas se resume en la condición de salto de una celda a otra, siempre limitado a las celdas inmediatamente próximas.

Una posible mejora consistiría en permitir saltos de varias celdas, dependiendo de la magnitud y dirección de la velocidad de la celda en la que se encuentra la partícula, incluso obligando a saltar al menos una celda en cada iteración. La complejidad del código aumentaría, ya que para cada partícula situada en la celda A, cuya celda destino fuese C, tendrían que consultarse todas las casillas situadas entre A y C, comprobando que no se tratasen de obstáculos (ni partículas) para poder realizar el desplazamiento. En caso de toparse con un obstáculo en su camino, digamos en la celda B, habría que redefinir el comportamiento de la partícula, que podría consistir en desplazarla hasta la celda B-1 y aplicar Bounce-Back a la partícula en dirección totalmente inversa, con tantas casillas como faltasen de B a C. El rendimiento en la animación podría verse afectado, y queda por ver si el resultado visual sería o no más realista que el actual.

5.5. Objetos complejos

Las partículas diseñadas en este proyecto, tienen como característica principal que su tamaño está limitado al de una celda, lo que ha simplificado su implementación.

Si se permitiera generar objetos móviles complejos, podrían simularse situaciones más cotidianas, como podría ser una pluma dirigida por el viento, copos de nieve en una escena invernal o una simple pelusa rotando en un vórtice generado en un rincón.

La tarea principal en este caso, consistiría en implementar la rotación de los objetos en base al campo de velocidades del escenario y su movimiento según la velocidad de las celdas que ocupa.

6. Bibliografía

[1] J. Ojeda, A. Susín (2009). *Aceleración de simulaciones de fluidos mediante el método Lattice Boltzmann utilizando CUDA.*

[2] C. Wang, Z. Wang, T. Xia, Q. Peng (2006). *Real-time snowing simulation.* Springer-Verlag.

[3] X. Wei, Y. Zhao, Z. Fan, W. Li, S. Yoakum-Stover, A. Kaufman (2003). *Blowing in the Wind*. Eurographics Association and Blackwell Publishers.

[4] E. Flórez, I. Cuesta, C. Salueña (2008). *Flujo de Poiseuille y la cavidad con pared móvil calculado usando el método de la ecuación de lattice Boltzmann.* Ingeniería & Desarrollo.

[5] R. Begum, M. Abdul Basit (2008). *Lattice Boltzmann Method and its Applications to Fluid Flow Problems.* European Journal of Scientific Research.

[6] C. Riff (2004). *Computational Fluid Dynamics (CFD) Modeling.*

[7] P. L. Bhatnagar, E.P. Gross, M. Krook (1954). *A Model for Collision Processes in Gases. I. Small Amplitude Processes in Charged and Neutral One-Component Systems*. Physical review 94 (3).

[8] N. Thürey (2007). *Physically based Animation of Free Surface Flows with the Lattice Boltzmann Method*. Tesis Doctoral.

[9] S. Chen, G. D. Doolen (1998). *Lattice Boltzmann Method for fluid flows*. Annual Review of Fluid Mechanics.

[10] X. Shan, H. Chen (1993). *Lattice Boltzmann model for simulating flows with multiple phases and components*. Physical review 47 (3).

[11] P. R. Rinaldi, E. A. Dari, D. D. Dalponte, M. J. Vénere, A. Clausse (2009). *Métodos de Lattice Boltzmann sobre GPU para simulación de fluidos. Comparación con NS mediante elementos finitos*. Mecánica Computacional Vol. XXVIII.

[12] F. D. E. Latief, U, Fauzi (2007). *Performance Analysis of 2D and 3D Fluid Flow Modelling Using Lattice Boltzmann Method*. Indonesian Journal of Physics Vol. 18 (2).

[13] F. J. Jiménez, J. V. Giráldez, E. Gutiérrez de Ravé, F. J. Moral (2005). *Análisis de la dispersión turbulenta de las emisiones del tráfico rodado usando la combinación modelos de malla - CAD*. Ministerio de Educación y Ciencia.

[14] L. Benstead, D. Astle, K. Hawkins (2009). *Beginning OpenGL game programming*. Course Technology/Cengage Learning, 2ª Ed.

[15] M. Pharr, F. Randima (2005). *GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation*. Addison-Wesley Professional.

[16] Consulta de numerosos conceptos físicos para la elaboración de la actual memoria. http://es.wikipedia.org.

[17] Fuente de artículos sobre LBM e ideas para la visualización del fluido en el sistema. http://www.lbmethod.org.

[18] Documentación sobre OpenGL. http://www.opengl.org.

[19] Varias vistas de los vórtices de Karman desde satélites de la NASA. http://disc.sci.gsfc.nasa.gov/oceancolor/additional/science-focus/ocean-color/vonKarman_vortices.shtml.