PRÁCTICA 3: GESTIÓN EFICIENTE DE INTERACCIONES ENTRE PARTÍCULAS

Jordi Beltran y Alberto Játiva

SIMULACIÓN Universitat de València

CONTENIDO

TABLA DE ILUSTRACIONES

TABLA DE GRÁFICAS

INTRODUCCIÓN

En esta práctica, simulamos dos sistemas de partículas para tratar el funcionamiento de colisiones con diferentes métodos, además de analizar el coste computacional de utilizar diferentes estructuras de datos.

En la primera parte, se simulará un billar francés para analizar tanto la colisión plano-partícula como partícula-partícula.

En la segunda parte, se simulará un fluido viscoso en un recipiente. Aquí analizaremos el tiempo computacional entre diferentes estructuras de datos como pueden ser Grid y Hash con respecto a no utilizar ninguno, además de aplicar la fuerza del muelle al tratar las colisiones partícula-partícula.

PARTE 1. BILLAR FRANCÉS

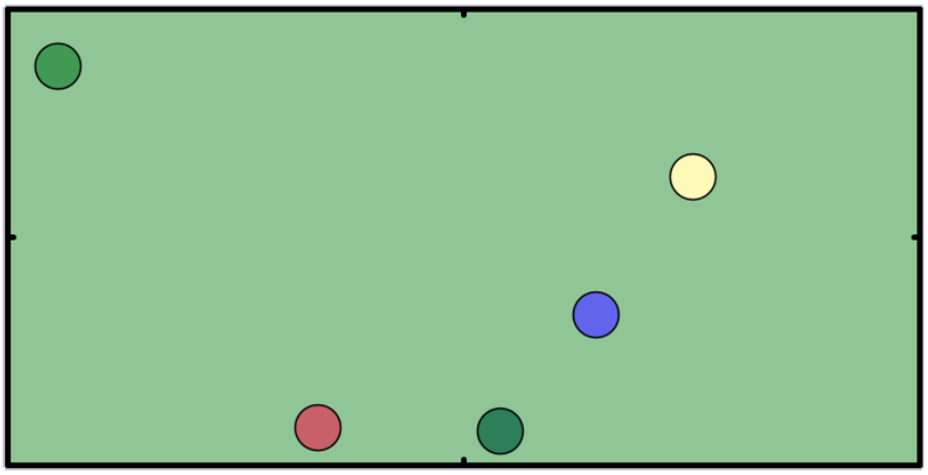


Ilustración 1. Billar Francés

¿OBSERVAS ALGUNA DIFERENCIA EN LA ESTABILIDAD DEL SISTEMA SEGÚN AUMENTA EL NÚMERO DE BOLAS?

A pesar de que el sistema se mantenga estable, ocurren comportamientos no deseados cuando las partículas ya no caben completamente en el tablero. Como no caben todas, se van empujando unas a otras fuera del tablero, estas intentan corregirse y volver al entrar en contacto con el plano o salirse de este, pero, al entrar en contacto con otra partícula de su interior, vuelven a salir o sale otra. Esto ocurre con los siguientes datos:

Número de partículas	> 125
Radio	20
Tamaño del tablero	800x400

Se puede observar un ejemplo de esto con 150 partículas en: Ejemplo 150 partículas

¿OBSERVAS ALGUNA DIFERENCIA EN LA ESTABILIDAD DEL SISTEMA AL AÑADIR UNA FUERZA F_C CON RESPECTO AL CASO ANTERIOR? ¿POR QUÉ?

Al añadir una fuerza que arrastra a las partículas a la esquina superior izquierda, añadiremos una aceleración en sentido a dicha esquina, haciendo que todas las partículas deseen ir a esa

ubicación, empujándose unas a otras para conseguir llegar. A pesar de tratar de resituar la ubicación de la partícula dentro del tablero cuando se ve arrastrada por otra que la empuja al exterior, se observan comportamientos extraños al tratar de restituirla, pues en la ubicación resultante hay más partículas colisionando. Calculamos la posición de restitución de la partícula al colisionar con el plano de la siguiente manera para que, en caso de salirse de este, vuelva al interior:

```
1. float distancia = planes.get(i).getDistance(_s);
2. if((distancia < _radius && planes.get(i).checkLimits(_s)))
3.        dist_restitucion = _radius - distancia;
4. else if(!planes.get(i).checkSide(_s))
5.        dist_restitucion = _radius + distancia;
```

En cuanto al tiempo de simulación o estabilidad del sistema, este no se ve afectado por añadir la fuerza F_C ni por incrementar las partículas, computacionalmente no está afectando, el problema es visual, ya que se puede apreciar el reposicionamiento de las partículas que salen de los planos tras la presión que hay dentro de estos debido al elevado número de partículas.

Se puede observar un ejemplo de esto con 125 partículas en: Ejemplo 125 partículas con F_C

PARTE 2. FLUIDO

Este ejercicio consiste en simular un fluido mediante un sistema de partículas que contendrá un número n de partículas de radio r y masa m dentro de un recipiente con forma específica. Estas partículas colisionan entre sí utilizando el modelo de colisiones mediante la utilización de muelles asimétricos, también colisionan con las paredes del recipiente, pero utilizando el modelo de velocidades como en el ejercicio del Billar Francés.

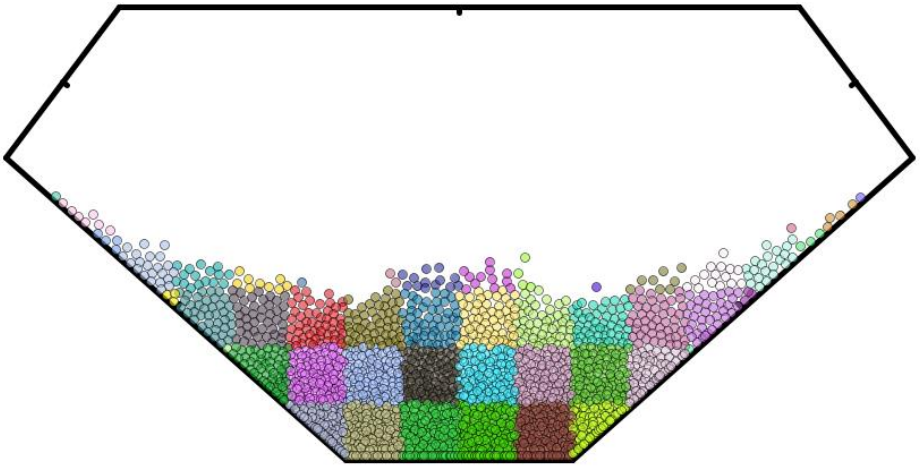


Ilustración 2. Fluido

PARÁMETROS DEL PROBLEMA

R	Radio de las partículas (m)	5
M	Masa de las partículas (kg)	0.1
G	Módulo del vector aceleración de la gravedad $(\mathrm{m/s^2})$	9.8
H	Altura del recipiente (m)	2
Cr	Coeficiente de restitución de las paredes del recipiente	0
Ke	Constante elástica para los muelles de colisión (N/m)	2
Dm	Distancia de activación de los muelles (m)	2*R
L0	Elongación de reposo de los muelles (m)	2*R
Kd	Constante de fricción cuadrática (kg/m)	0.001

ESTRATEGIAS DE SIMULACIÓN

Dado que el número de partículas puede ser elevado, se implementan dos estructuras de datos para mejorar la eficiencia computacional de la simulación. Mediante estas estructuras de datos, se agrupan las partículas vecinas en relación con cada partícula individual, lo que permite calcular las colisiones considerando únicamente este subconjunto de partículas, en lugar de evaluar todas las partículas presentes en el sistema en cada iteración.

Se implementan las estructuras de datos Grid y Hash, que presentan otros parámetros que definen su comportamiento:

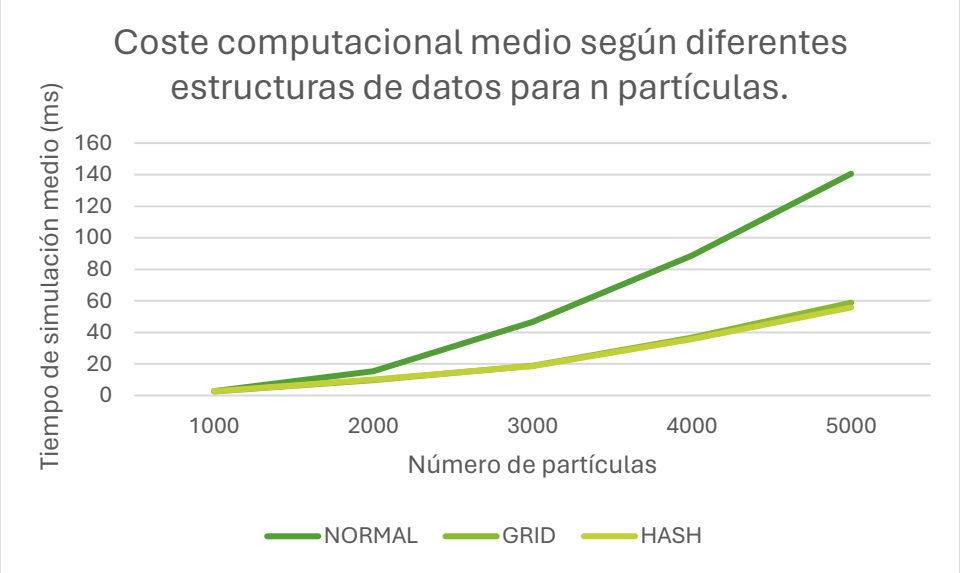
- Grid:
  - S_C → ancho y alto de la celda (m). Puesto que las celdas son cuadradas
  - El número de celdas viene dado por el tamaño de la pantalla
- Hash
  - S_C → ancho y alto de la celda (m). Puesto que las celdas son cuadradas
  - N_C→ número de celdas

ANÁLISIS DEL COSTE COMPUTACIONAL

A través de los ficheros CSV generados durante las simulaciones se generan gráficas dónde podemos analizar el coste computacional de cada simulación y compararlo con otras utilizando otros parámetros u otras estructuras de datos.

COMPARACIÓN DEL COSTE COMPUTACIONAL PARA LAS DIFERENTES ESTRUCTURAS DE DATOS UTILIZANDO DIFERENTES NÚMEROS DE PARTÍCULAS

Realizamos una comparación del tiempo necesario para simular una iteración de la simulación dependiendo del número de partículas del sistema y utilizando Grid, Hash o ninguna estructura de datos.



Gráfica 1. Coste computacional para *n* partículas

Se observa que el coste computacional aumenta significativamente con el número de partículas en todos los casos, pero también podemos apreciar como el uso de las estructuras de datos Grid y Hash reduce este coste en comparación con la simulación sin utilizar ninguna de estas estructuras de datos para optimizar la eficiencia.

- Sin utilizar estructura de datos: el coste aumenta de forma cuadrática a medida que aumenta el número de partículas y esto es debido a que cada partícula debe interactuar con todas las demás. El resultado de esto es un crecimiento cuadrático en el tiempo de simulación.
- Grid: el uso de esta estructura de datos mejora el rendimiento de la simulación, a medida que aumenta el número de partículas, también aumentará el costo computacional, sin embargo, no aumentará igual que sin utilizar ninguna estructura de datos, ya que únicamente analizará el estado con sus vecinos.

- Hash: el uso de una función de hash para distribuir las partículas en celdas mejora el rendimiento de la simulación. El coste computacional es muy similar al Grid ya que utiliza el formato de x vecinos. En este caso vemos que la utilización de Hash le otorga mucha mejora del tiempo de simulación, pero esto podría no ser así, esto se debe a la función de Hash, ya que esta debe ser una u otra según el problema en el que nos encontremos. En este caso estamos utilizando la siguiente función hash para distribuir las partículas en celdas:

$$hash(p) = 73856093 * p.x + 19349663 * p.y$$

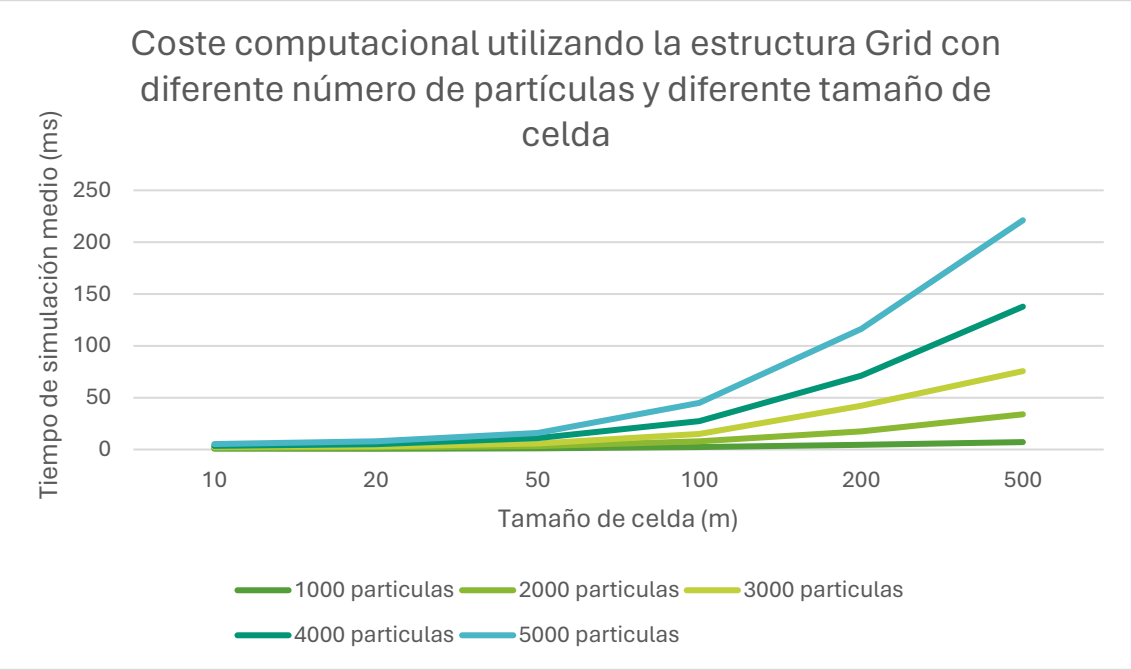
```
1. int hash(PVector _s){
2.       long xd = int(floor(_s.x) / _cellSize);
3.       long yd = int(floor(_s.y) / _cellSize);
4.       if (xd < 0) xd = 0;
5.       if (yd < 0) yd = 0;
6.       long suma = 73856093*xd + 19349663*yd;
7.       int cell = int(suma % _table.size());
8.       return cell;
9. }
```

Grid y Hash producen el mismo coste computacional, son del orden de $O(\mathrm{n} \cdot \log n)$. Esto es así ya que sólo se analiza el estado de una partícula con sus vecinas. Una ventaja del Hash con respecto al Grid es el ahorro producido en memoria, ya que Grid almacena todas las celdas de la ventana mientras que hash solo almacenará las celdas donde haya partículas. Por tanto, podemos destacar que el uso de Grid es más viable para simulaciones en las que hay partículas por todo el espacio mientras que, Hash es mejor para simulaciones donde hay partículas únicamente en regiones concretas, pues Grid almacenará gran cantidad de celdas vacías.

En cuanto a no utilizar estructuras de almacenamiento, el coste computacional es del orden de $O(n^2 - n)$. Esto se debe a que no se obtienen partículas vecinas. Para obtener las colisiones de una partícula, se debe analizar el estado de todas las partículas menos el de ella misma. Por tanto, para n partículas se deben analizar n - 1 partículas.

COMPARACIÓN DEL COSTE COMPUTACIONAL UTILIZANDO GRID CON DIFERENTES TAMAÑOS DE CELDA S_c

Se evalúa el efecto que tiene modificar el parámetro S_c del Grid, y el número de partículas en el sistema. De esta manera, podremos observar cómo varía el costo computacional según el tamaño de la celda.



Gráfica 2. Coste computacional para *x* tamaño de celda (GRID)

Se aprecia una relación directa entre el incremento del tamaño de la celda y el aumento del coste computacional medio de la simulación. Esta relación se justifica por el hecho de que, al ampliar el tamaño de la celda, se incorporan un mayor número de partículas en el cálculo del vecindario de cada partícula. Es debido a esto que, con un número elevado de partículas y un tamaño de celda reducido obtendremos mejores resultados en cuanto a coste computacional, sacrificando mayor espacio en memoria.

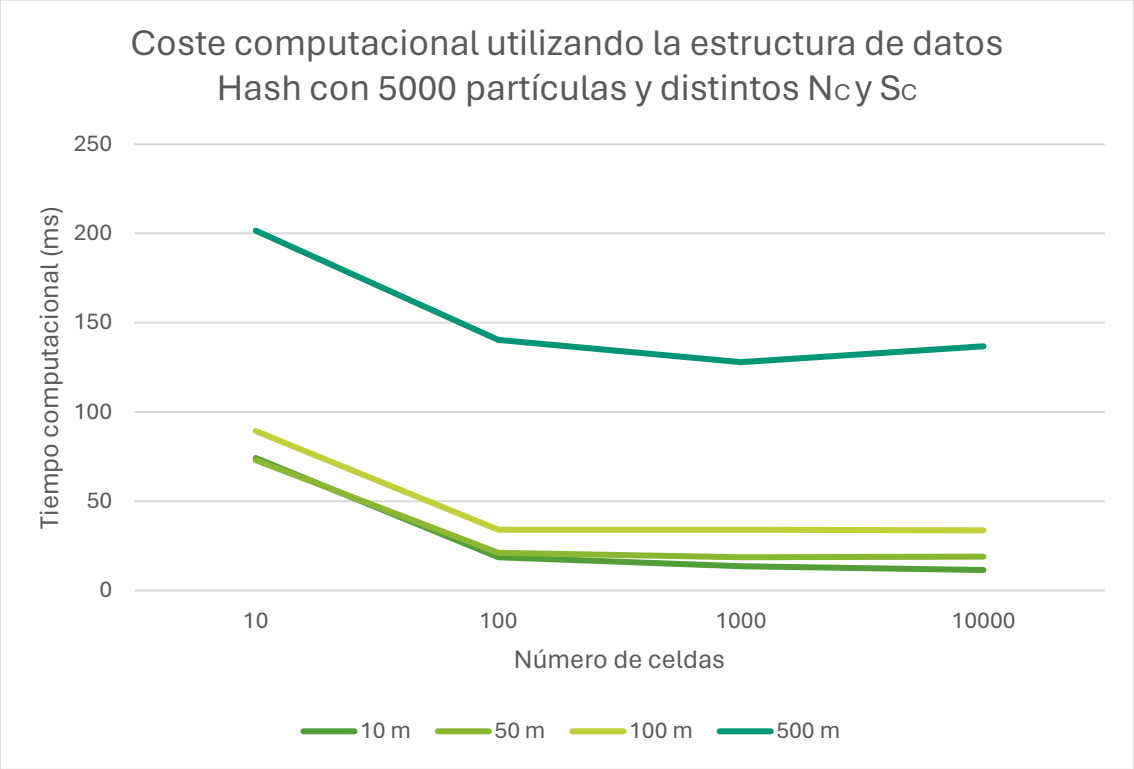
La gráfica muestra como varia el tiempo de simulacion medio en función del tamaño de la celda, a medida que se aumenta el tamaño de la celda, el tiempo de simulación medio también aumenta. Esto es debido a que, a mayor tamaño de celda, hay más partículas con las que comprobar colisión. Por tanto, nos favorece la utilización de tamaños de celdas pequeños.

El aumento de este tiempo de simulacion no solo depende del tamaño de la celda, también depende del número de partículas.

Por ejemplo, para un tamaño de ventana de 1000x1000 y un tamaño de celda 500x500, obtendremos una distribución de 2x2 celdas y, por tanto, todas las partículas serán vecinas de todas, puesto que no habrá celdas más allá de las vecinas.

COMPARACIÓN DEL COSTE COMPUTACIONAL UTILIZANDO HASH CON DIFERENTES TAMAÑOS DE CELDA S_c Y n_c

Se evalúa el efecto que tiene modificar el parámetro S_c de la tabla Hash, y el número de celdas que esta tendrá, se utiliza un número de partículas fijo de 5000 para realizar esta comparación. Veremos con la siguiente gráfica de qué manera afecta el tamaño de la celda y el número de estas al coste computacional de la simulación.



Gráfica 3. Coste computacional para *x* tamaño de celda e *y* número de celdas (HASH)

Se observa como el coste computacional aumenta al aumentar el tamaño de las celdas ya que, a mayor tamaño de celda, mayor número de partículas por celdas (como se explicó en el Grid). Además, se puede observar cómo, al aumentar el número de celdas, disminuirá el tiempo computacional, ya que la estructura donde se almacenan será mayor. Esto se debe a que las partículas se almacenarán en posiciones más repartidas por todo el vector principal de la tabla Hash, haciendo más pequeños los vectores del interior de los índices del vector principal. Al haber más índices, habrá menos partículas dentro de cada índice.

Podemos concluir que, para simulaciones con una gran cantidad de partículas, es recomendable utilizar una estructura de almacenamiento de datos para ahorrar costes computacionales (pasando del orden de n^2 a un orden logarítmico). A pesar de que Grid y Hash presentan tiempos de simulación similares,Hash es una gran alternativa frente cuando las partículas se sitúan en zonas determinadas de la ventana de simulación.