Medición de rendimiento de algoritmos de simulación NBody con ordenamiento previo basados en SFC

Diego Hinrichs, *Computer Civil Engineering Student, UACh*

***Resumen*—En este estudio se presenta una implementación de simulación N-Body, abordando la complejidad computacional inherente del problema mediante el uso de curvas de relleno de espacio, específicamente las curvas de Hilbert y Morton, así como el algoritmo Barnes-Hut. Estas técnicas permiten una mejor gestión de la memoria y preservación de la localidad espacial, lo que resulta en mejoras de rendimiento en simulaciones de gran escala. Se comparan los tiempos de simulación, mostrando que el ordenamiento previo es beneficioso en contextos con un número elevado de cuerpos. Los resultados indican que, aunque el proceso de ordenamiento introduce una sobrecarga computacional, esta es justificada en aplicaciones donde la eficiencia en el acceso a la memoria es crítica.**

I. Introducción

LAS simulaciones de cuerpos o partículas que interactúan entre sí son un campo de estudio que abarca desde la astronomía hasta lo más atómico que uno pueda imaginar. En este trabajo se aborda una simulación N-Body, donde cada cuerpo es afectado por los n-1 cuerpos restantes, debido a fuerzas que alteran la posición en el tiempo según ciertas condiciones iniciales, como la velocidad, masa del cuerpo y constante gravitacional.

Este problema tiene una complejidad computacional O(n^2), lo que sugiere que con cantidades pequeñas de cuerpos (n) su rendimiento es eficiente y el tiempo de ejecución bastante aceptable. Para valores grandes de (n), el rendimiento decrece debido a la complejidad.

Una SFC (curva de relleno de espacio) implica cubrir un espacio p-dimensional (p = 2, 3) con una curva continua. Esto nos permite ordenar los cuerpos reduciendo la dimensionalidad del espacio, asignando a cada cuerpo un índice en la curva y preservando la localidad espacial de los cuerpos, lo que permite administrar de manera eficiente el almacenamiento en RAM. Además, el algoritmo de Barnes-Hut ofrece una técnica jerárquica para reducir la complejidad computacional a O($n \log n$) mediante el uso de QuadTrees u OctTrees para agrupar cuerpos distantes y calcular las interacciones de manera aproximada.

Este trabajo revisa que el ordenamiento previo no tiene un impacto significativamente alto cuando se trata de grandes cantidades de cuerpos, pero abre puertas a algoritmos que junto a las SFC y el algoritmo Barnes-Hut, pueden llegar a mejorar el tiempo de simulación.

II. Trabajo Relacionado

Podemos encontrar diferentes enfoques para implementar una simulación N-Body, cada uno con sus propias ventajas y desventajas en términos de rendimiento y precisión. A continuación, se presentan algunos de los métodos más relevantes:

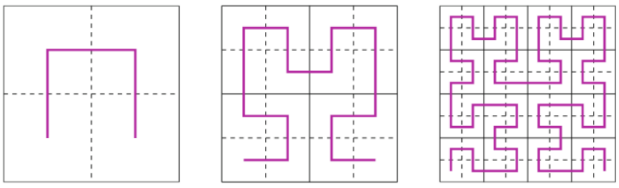

Figura 1. Ejemplo de construcción recursiva de Curva de Hilbert ilustrada por Bader[1]

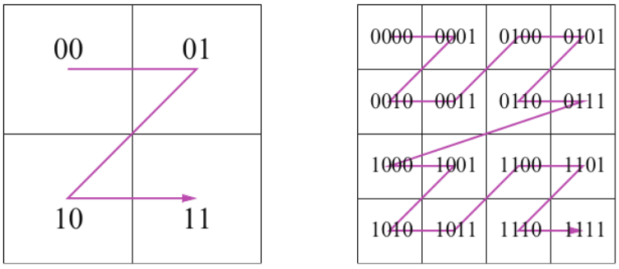

Figura 2. Ejemplo de construcción recursiva de Curva de Morton ilustrada por Bader[1]

II-A. Algoritmo de Barnes-Hut

El algoritmo de Barnes-Hut[2], propuesto por Josh Barnes y Piet Hut en 1986, es una técnica ampliamente utilizada para mejorar la eficiencia de las simulaciones N-Body. Este método reduce la complejidad computacional de O(n^2) a O($nlogn$) mediante el uso de una estructura de datos jerárquica conocida como QuadTree (para 2D) u OctTree (para 3D).

El algoritmo divide el espacio en regiones cuadrantes (QuadTree) o octantes (OctTree) y calcula las interacciones entre cuerpos de forma aproximada, agrupando cuerpos distantes en una única masa central. Este enfoque permite calcular las fuerzas gravitacionales de manera más eficiente, especialmente en simulaciones con un gran número de cuerpos.

II-A1. QuadTree: Un QuadTree es una estructura de datos jerárquica en la que cada nodo tiene cuatro hijos. Se utiliza para dividir recursivamente el espacio en cuatro cuadrantes hasta que cada cuadrante contiene un solo cuerpo o un número limitado de cuerpos. La construcción y recorrido del QuadTree permiten agrupar cuerpos distantes y tratar sus interacciones de manera aproximada, reduciendo así el número total de cálculos necesarios.

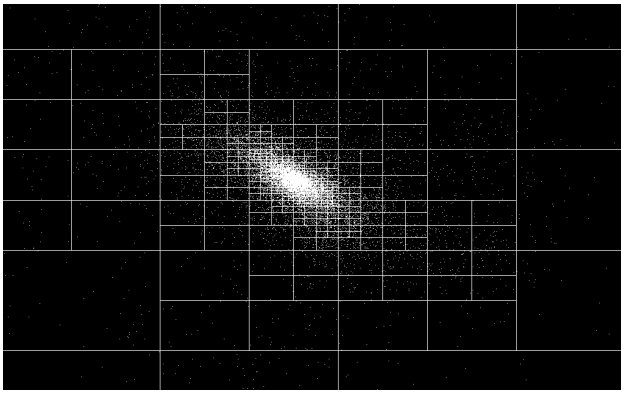

Figura 3. Simulación de Barnes-Hut utilizando un QuadTree para la construcción del espacio.

II-B. Implementación con CUDA

NVIDIA Gems3[3] proporciona una implementación paralelizada del algoritmo N-Body utilizando CUDA. Esta implementación aprovecha la capacidad de procesamiento masivamente paralelo de las GPU para mejorar el rendimiento de las simulaciones. En comparación con las implementaciones en CPU, el uso de CUDA permite manejar un mayor número de cuerpos en un tiempo significativamente menor. La implementación de NVIDIA Gems3 es altamente optimizada y demuestra cómo los recursos computacionales avanzados pueden mejorar tanto el tiempo de simulación como la precisión. Esto se abordará en un trabajo futuro para contrastar los resultados obtenidos en este estudio

II-C. Curvas de Relleno de Espacio

Existen varias curvas de relleno de espacio que han sido ampliamente estudiadas y aplicadas en diversos campos de la computación y la ingeniería. A continuación, se presentan algunas de las más relevantes:.

II-C1. Curva de Hilbert: La curva de Hilbert fue introducida por el matemático David Hilbert en 1891. Es una curva continua y fractal que pasa por cada punto de una cuadrícula 2^nx2^n . Una de sus principales ventajas es su capacidad para preservar la localidad espacial de una manera más efectiva que otras curvas, lo que la hace especialmente útil en aplicaciones que requieren acceso eficiente a la memoria.

II-C2. Curva de Morton: La curva de Morton, también conocida como orden Z, fue descrita por Guy MacDonald Morton en 1966. Su principal ventaja es la simplicidad de su implementación, que se logra mediante la intercalación de bits. Aunque no preserva la localidad espacial tan eficientemente como la curva de Hilbert, su facilidad de cálculo la hace atractiva para diversas aplicaciones.

II-D. Otras Curvas de Relleno de Espacio

Además de las curvas de Hilbert y Morton, existen otras curvas de relleno de espacio como la curva de Peano y la curva de Sierpinski, que también se utilizan para aplicaciones de preservación de localidad espacial. Estas curvas, aunque menos comunes en simulaciones N-Body, ofrecen propiedades interesantes que pueden ser exploradas en trabajos futuros.

Muchas de estas curvas e implementaciones se pueden encontrar en “Space-Filling curves” de Michael Bader[1].

III. METODOLOGÍA

Se implementarán dos ordenamientos basados en SFC, Hilbert y Morton. Para la parte experimental, se medirá el tiempo que tarda la simulación en cada paso de tiempo en contraste con ordenamiento, contrastando con la versión sin ordenar.

III-A. N-Body

Se realiza la implementación de N-Body por fuerza bruta, similar a la de NVIDIA gems3[3], este algoritmo tiene una complejidad $O(n^2)$, la idea básica es que la posición x del cuerpo n en el tiempo t, será la suma de todas las fuerzas que ejercen los $n-1$ cuerpos sobre él, esto es:

$$a_{ji} = \sum_{j=0}^{n-1} G \cdot \frac{m_j \cdot x_{ji}}{||x_{ji}||^{\frac{3}{2}}}$$

Esta forma de la aceleración se obtiene de lo siguiente, nos limitaremos a la fuerza que un cuerpo j ejerce sobre otro i:

$$F_{ji} = G \cdot \frac{m_i \cdot m_j}{||x_{ji}||^2} \hat{u}_{ji} \;\; ; \;\; \hat{u}_{ji} = \frac{x_{ji}}{||x_{ji}||}$$

Donde, $m_{i,j}$ es la masa de los cuerpos, $||x_{ji}||$ es la distancia entre los cuerpos n_j, n_i y $\hat{u}_{ji}$ es el vector unitario entre el n_j y n_i, es decir, la dirección sin considerar la magnitud.

Además, para poder conocer la posición del cuerpo en $t+1$, nos interesa conocer la aceleración y posterior velocidad que se aplicará sobre el cuerpo, utilizamos la segunda ley de Newton para obtener las otras ecuaciones.

$$a = \frac{F}{m_i}$$

Con eso podemos simplificar las masas m_i:

$$a_i \approx G \frac{m_j \cdot x_{ji}}{||x_{ji}||^{\frac{3}{2}}}$$

Integramos la velocidad en el tiempo:

$$v(t) = v_0 + a \cdot dt$$

Y actualizamos la posición del cuerpo:

$$x(t) = x_0 + v \cdot dt$$

Algorithm 1: NBody algorithm

Data: numBodies, steps, G, dt, orderMode = 0
Result: Bodies position in last step
$bodies \leftarrow initBodies(numBodies, seed)$;
$step \leftarrow 0$;
if $orderMode \neq 0$ **then**
  while $step < steps$ **do**
    $sort(bodies, numBodies, orderMode)$; `//orderMode: Hilbert or Zorder`
    $updateForces(bodies, numBodies, G, dt)$;
    $updatePosition(bodies, velocityDumping = 0.90f)$;
    $step++$;
  end
else
  if $step < steps$ **then**
    while $step < steps$ **do**
      $updateForces(bodies, numBodies, G, dt)$;
      $updatePosition(bodies, velocityDumping = 0.90f)$;
      $step++$;
    end
  end
end

Figura 4. Pseudocódigo de NBody por fuerza bruta

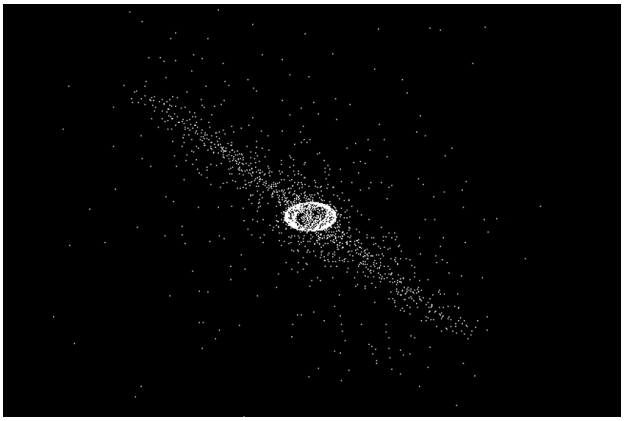

Figura 5. Simulación de N-Body, por fuerza bruta 1024 partículas.

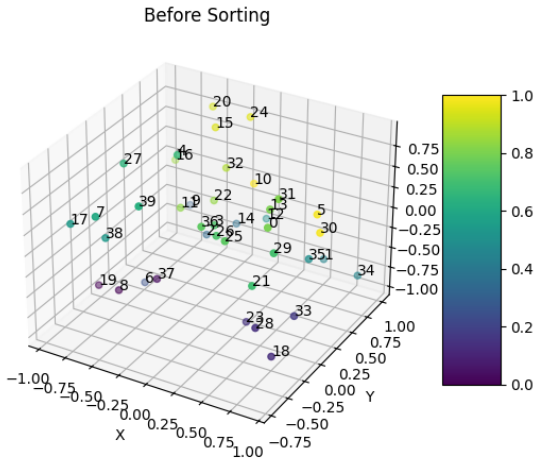

Figura 6. Cuerpos previo ordenamiento

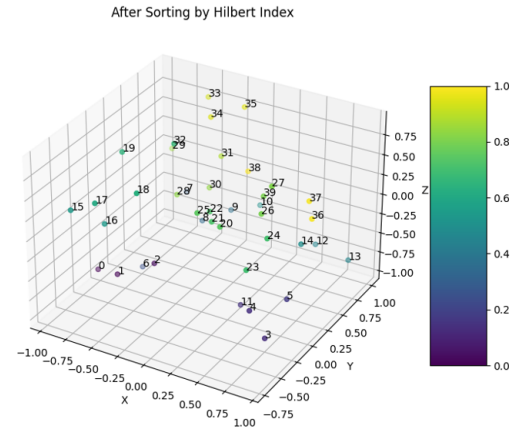

Figura 7. Ordenamiento de 40 cuerpos a partir de la curva de Hilbert

III-B. Hilbert

La curva de Hilbert[4] es una curva de relleno de espacio que pasa por todos los puntos de una cuadrícula $2^n \times 2^n$. Esta curva se construye de manera recursiva y es conocida por su excelente preservación de la localidad espacial. La implementación del ordenamiento de Hilbert cuenta con varias etapas:

1. **Normalización de coordenadas**: Las coordenadas x, y, z de cada cuerpo se normalizan para que estén en el rango [0, 1023].
2. **Intercalación de bits**: Se intercalan los bits de las coordenadas normalizadas para formar un solo índice Hilbert.
3. **Cálculo del índice Hilbert**: Utilizando la función de intercalación, se calcula el índice Hilbert para cada punto.
4. **Ordenamiento**: Los cuerpos se ordenan de acuerdo a sus índices Hilbert utilizando un algoritmo de ordenamiento.

En la fig. 7, se observa el agrupamiento de los cuerpos según el índice de Hilbert, diferenciando por colores los cuerpos que se encuentran más cercanos.

III-C. Morton

La curva de Morton, también conocida como “z-order”, es otra curva de relleno de espacio que organiza los puntos de una cuadrícula $2^n \times 2^n$. Esta curva se caracteriza por una intercalación más simple de bits en comparación con la curva de Hilbert.

1. **Normalización de coordenadas**: Las coordenadas x, y, z de cada cuerpo se normalizan para que estén en el rango [0, 1023].
2. **Intercalación de bits**: Se intercalan los bits de las coordenadas normalizadas de manera que un bit de cada coordenada se coloca en el índice resultante.
3. **Cálculo del índice Morton**: Utilizando la función de intercalación, se calcula el índice Morton para cada punto.
4. **Ordenamiento**: Los cuerpos se ordenan de acuerdo a sus índices Morton utilizando un algoritmo de ordenamiento.

La curva de Morton no preserva la localidad espacial tan

eficientemente como la curva de Hilbert.

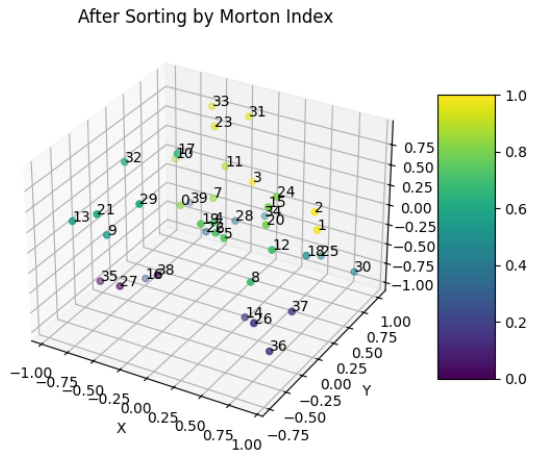

Figura 8. Ordenamiento de 40 cuerpos a partir de la curva de Morton o z-order

III-D. Barnes-Hut

El algoritmo de Barnes-Hut[2], se caracterizá por implementar estructuras jerarquicas conocidas como árboles, en este caso particular se implementará utilizando un QuadTree[5], que subdivide el espacio en 4 cuadrantes según se cumplan ciertas condiciones como la cantidad máxima de cuerpos por cuadrante y profundidad máxima del árbol.

Recorrido del Árbol: Para calcular las fuerzas sobre un cuerpo, se recorre el árbol desde la raíz.
Criterio de Aproximación: Se utiliza un criterio de apertura, generalmente basado en el tamaño de la región y la distancia al cuerpo, para decidir si se puede usar un nodo interno como una aproximación. Este criterio se expresa típicamente como θ , donde θ es un valor pequeño (por ejemplo, 0.5 valor utlizado para este estudio).

1. Si el tamaño del nodo dividido por la distancia al cuerpo es menor que θ, se considera que todos los cuerpos dentro de ese nodo pueden ser aproximados como un único cuerpo en el centro de masa del nodo.
2. Si no se cumple el criterio, el nodo se divide y se consideran sus subregiones.

Suma de Fuerzas: Se suman las fuerzas de todos los nodos y cuerpos que cumplen el criterio de aproximación.

Se debe considerar el valor de θ, puede existir un compromiso entre la exactitud y la eficiencia.

III-E. Ambiente de trabajo

La implementación se realizó en C++, utilizando paralelismo a nivel de CPU con las directivas ofrecidas por OpenMP, para paralelizar los cálculos de fuerza y posición. La CPU utilizada para realizar las pruebas fue un intel i5 de 12va generación.
Los párametros de configuración para cada experimento son los mismos $G = 1{,}2$, $dt = 0{,}016$, $\epsilon = 0{,}009$, $steps = 3000$, se modificará la cantidad de cuerpos n, el tipo de ordenamiento $0 : noSort$, $1 : Z - Order$, $2 : Hilbert$.

IV. Resultados Experimentales y Análisis

En fig. 7 y fig. 8 podemos observar que los cuerpos se ordenan con gran presición, como existe un escalamiento previo de las coordenadas, puede que existan errores de calculos en los índices, pero no deberia afectar en la simulación.

Para evaluar el rendimiento de los algoritmos de simulación N-Body con las curvas de Hilbert y Morton, realizamos varios experimentos con diferentes tamaños de problema. Los resultados se presentan en la siguiente tabla y gráficos.

Cuadro I
Resultados Experimentales N-Body fuerza bruta

N	Orden	Tiempo [s]
4096	No ordenar	23,6
4096	Morton	26,0
4096	Hilbert	23,5
8192	No ordenar	85,0
8192	Morton	96,2
8192	Hilbert	96,4
16384	No ordenar	371,3
16384	Morton	382,3
16384	Hilbert	354,5

Cuadro II
Resultados Experimentales N-Body Barnes-Hut

N	N x Nodo	Orden	Tiempo [s]
4096	8	No ordenar	1,8
4096	8	Morton	3,6
4096	8	Hilbert	3,4
8192	16	No ordenar	17,4
8192	16	Morton	24,8
8192	16	Hilbert	23,5
16384	32	No ordenar	43,5
16384	32	Morton	49,3
16384	32	Hilbert	37,2

IV-A. Tiempo de Simulación

El primer experimento mide el tiempo total de simulación para diferentes cantidades de cuerpos, comparando tres escenarios: sin ordenamiento, ordenamiento utilizando la curva de Morton (Z-Order) y ordenamiento utilizando la curva de Hilbert.

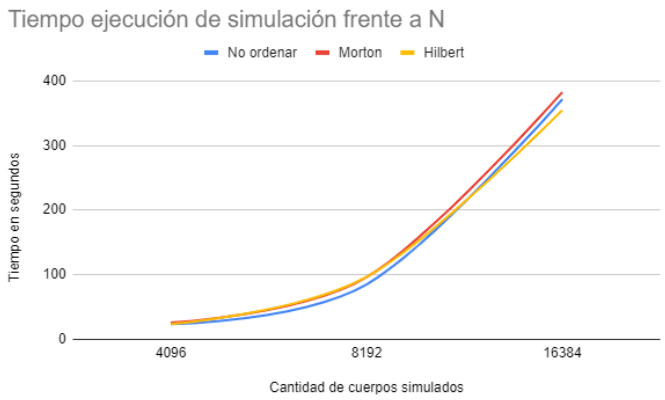

Figura 9. Comportamiento polinomial de simulación N-Body, para distintos tipos de ordenamiento.

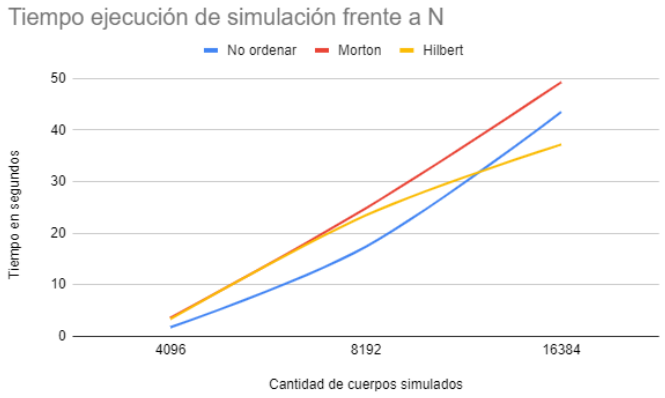

Figura 10. Tiempo de simulación utilizando Barnes-Hut

IV-B. Análisis de Resultados

Como se muestra en la Figura 9, el tiempo de simulación aumenta con el número de cuerpos en todos los casos, lo que es evidente para un algoritmo de fuerza bruta. Se observa que el ordenamiento previo no tiene un impacto relevante al momento de realizar la simulación, si bien puede añadir sobrecarga computacional, esta es casi imperceptible.

En la Figura 10, apreciamos la superioridad de la simulación con Barnes-Hut. Si bien es una aproximación, la cantidad de cuerpos simulados no afecta el rendimiento y el ordenamiento no añade una sobrecarga significativa, en algunos casos puede ser incluso mejor.

Los resultados experimentales demuestran que, aunque el ordenamiento utilizando curvas de relleno de espacio (Hilbert y Morton) introduce una sobrecarga computacional adicional, esta puede ser justificada en contextos donde la preservación de la localidad espacial y la eficiencia en el acceso a la memoria son críticas. Para simulaciones con un gran número de cuerpos, el uso de estas curvas puede ser ventajoso a largo plazo, en combinación con otras técnicas.

V. Discusión y Conclusiones

V-A. Discusión

Los resultados obtenidos en este estudio demuestran que el uso de curvas de relleno de espacio, como las curvas de Hilbert y Morton, introduce una sobrecarga computacional adicional debido al tiempo necesario para realizar el ordenamiento previo.

En nuestras simulaciones, observamos que para tamaños de problema más pequeños, el ordenamiento no aporta beneficios significativos en términos de tiempo de simulación. Esto se debe a que el costo del ordenamiento puede superar las ganancias obtenidas por la mejor gestión de la memoria. A medida que el tamaño del problema aumenta, el tiempo de ordenar se vuelve insignicante para la cantidad de cuerpos simulada, esto gracias a que ambos ordenamientos son $O(nlogn)$, con ello en mente, podemos utilizar la curva de Hilbert, que tiene un menor impacto en tiempo total de la simulación, para ordenar los cuerpos y utilizar de mejor manera el almacenamiento en caché, esto combinado con OpenMP, para paralelizar los loops más críticos del proceso.

V-B. Conclusiones

Este estudio ha demostrado que, aunque el uso de curvas de relleno de espacio introduce una sobrecarga computacional adicional, su capacidad para preservar la localidad espacial puede ser beneficiosa para simulaciones de gran escala. La optimización de estos algoritmos y la utilización de estructuras de datos más complejas, así como la evaluación en hardware especializado, representan direcciones prometedoras para futuras investigaciones.

En conclusión, la mejora del rendimiento de las simulaciones N-Body mediante técnicas de ordenamiento y optimización es un área activa de investigación. Las curvas de Hilbert y Morton ofrecen una base sólida para mejorar la eficiencia, y futuras optimizaciones pueden llevar a avances significativos en este campo.

Referencias

[1] M. Bader, *Space-Filling Curves: An Introduction with Applications in Scientific Computing*, Springer, 2012. DOI: 10.1007/978-3-642-31046-1.
[2] Wikipedia, ”Barnes–Hut simulation,.Aˆvailable: https://en.wikipedia.org/wiki/Barnes
[3] NVIDIA, ”Fast N-Body Simulation with CUDA,ïn GPU Gems 3, Available: https://developer.nvidia.com/gpugems/gpugems3/part-v-physics-simulation/chapter-31-fast-n-body-simulation-cuda.
[4] ThreadLocalMutex, ”Space-Filling Curves and Their Uses,.Aˆvailable: https://threadlocalmutex.com/?p=126.
[5] Wikipedia, ”Quadtree,.Aˆvailable: https://en.wikipedia.org/wiki/Quadtree.