**Carlos Manuel Vélez, Andres Martínez y Miguel Angel Tovar**

El objetivo de este reporte es explicar el funcionamiento de un N-Body Simulator desarrollado en Python utilizando la biblioteca PyCUDA para realizar diversas tareas en paralelo. Este simulador tiene como objetivo modelar el movimiento de un conjunto de partículas en un espacio tridimensional, teniendo en cuenta las fuerzas gravitacionales entre ellas. El uso de PyCUDA permite aprovechar la capacidad de cómputo de la GPU para acelerar significativamente los cálculos necesarios en este tipo de simulaciones.

**Descripción del Código**

El código se organiza en varias secciones, cada una de las cuales desempeña un papel fundamental en la simulación:

**Importaciones de Librerías:**

Se importan las librerías necesarias para la ejecución del código, incluyendo numpy para operaciones numéricas, pycuda para la programación en CUDA, y matplotlib para la visualización de datos.

**Definición de la Clase Particle:**

Se define una clase llamada Particle, que representa las partículas de la simulación. Cada partícula tiene una posición, una velocidad y una masa. Al inicializar la clase, se generan aleatoriamente las posiciones, velocidades y masas de un determinado número de partículas.

**Kernel de CUDA:**

El código incluye un kernel de CUDA que define dos funciones: calculateForces e integrate. Estas funciones se ejecutan en paralelo en la GPU y son responsables de calcular las fuerzas entre las partículas y actualizar sus posiciones y velocidades en cada periodo de tiempo.

**Función run\_simulation:**

Esta función coordina la ejecución de la simulación. Comienza compilando el código del kernel de CUDA utilizando la clase SourceModule de PyCUDA. Luego asigna memoria en la GPU para reservar espacio y almacenar las posiciones, velocidades, fuerzas y masas de las partículas. Durante cada iteración de la simulación, se ejecutan las funciones del kernel para calcular y actualizar el movimiento de las partículas. Se utiliza matplotlib para visualizar en tiempo real el estado de la simulación y trazar las trayectorias de las partículas al finalizar la misma.

**Detalle del Funcionamiento**

**Inicialización de las Partículas:**

Se crea un número específico de partículas con posiciones, velocidades y masas aleatorias.

**Compilación del Kernel de CUDA:**

El código del kernel de CUDA se compila utilizando la clase SourceModule, lo que permite su ejecución en la GPU y el intercambio de datos.

**Asignación de Memoria en la GPU:**

Se asigna memoria en la GPU para almacenar las propiedades de las partículas (posiciones, velocidades, fuerzas y masas).

**Ejecución de la Simulación:**

Durante un número especificado de iteraciones, se ejecutan las funciones del kernel para calcular las fuerzas entre las partículas y actualizar su movimiento. Los resultados se transfieren de vuelta a la CPU para su visualización.

**Visualización de la Simulación:**

Se utiliza matplotlib para visualizar en tiempo real el estado de la simulación, mostrando las posiciones de las partículas en un gráfico tridimensional. Además, se trazan las trayectorias de las partículas al finalizar la simulación.

**Conclusiones**

El simulador desarrollado utilizando PyCUDA proporciona una herramienta eficaz para modelar el movimiento de partículas en un espacio tridimensional. La utilización de la GPU para realizar cálculos en paralelo permite acelerar significativamente la simulación, lo que facilita el estudio de sistemas de partículas complejos y de gran tamaño. Además, la capacidad de visualización en tiempo real proporcionada por matplotlib permite una comprensión intuitiva del comportamiento de las partículas durante la simulación.

En resumen, el simulador de partículas utilizando PyCUDA es una herramienta poderosa y versátil que puede ser utilizada en una amplia gama de aplicaciones científicas y de ingeniería, desde la física de partículas hasta la simulación de sistemas biológicos y la dinámica de fluidos gracias a la posibilidad de ejecutar una gran cantidad de operaciones simultáneamente, obteniendo resultados en una menor cantidad de tiempo. Su flexibilidad y rendimiento lo convierten en una herramienta valiosa para investigadores y profesionales en diversos campos.