**Taller Unidad 4**

Se debe modelar y simular el **movimiento de partículas cargadas** bajo la influencia de un campo eléctrico utilizando las leyes de la física clásica. Deberán aplicar la **sobrecarga de operadores** para representar la interacción entre las partículas, construir un **Makefile** para automatizar la compilación y realizar pruebas de eficiencia en las simulaciones con configuraciones **Debug** y **Release.**

1. **Modelado del Movimiento de Partículas Cargadas**

Se debe crear un programa en **C++** que modele el movimiento de partículas cargadas bajo la influencia de un campo eléctrico uniforme. La ecuación básica es la siguiente:

**F=qE**

Donde:

* F es la fuerza sobre la partícula.
* q es la carga de la partícula.
* E es el campo eléctrico.

A partir de esta fuerza, los estudiantes deberán calcular la aceleración de la partícula utilizando la segunda ley de Newton:

**a=F/m**

Con la aceleración calculada, deben actualizar la posición y la velocidad de la partícula utilizando un esquema numérico simple como el método de Euler.

**2. Sobrecarga de Operadores**

Los estudiantes deberán implementar una clase **Particle** que represente una partícula. Para esto, necesitarán realizar las siguientes **sobrecargas de operadores**:

* Sobrecargar el operador **+** para sumar vectores, lo cual servirá para sumar la posición de la partícula y la velocidad.
* Sobrecargar el operador **\*** para multiplicar un vector por un escalar (por ejemplo, multiplicar la aceleración por el tiempo paso para actualizar la posición).
* Sobrecargar el operador **<<** para imprimir el estado de la partícula (posición, velocidad, etc.) de forma legible.

#### 3. ****Estructura del Proyecto****

El proyecto debe estar estructurado de la siguiente manera:

ParticleSimulation/

├── Makefile

├── main.cpp

├── Particle.h

├── Particle.cpp

└── utils.h

**main.cpp**: Contendrá el código que crea las partículas, simula el movimiento y calcula la evolución en el tiempo.

**Particle.h y Particle.cpp**: Definirán la clase Particle y sus métodos. Aquí se implementarán los operadores sobrecargados.

**utils.h**: Definirá funciones adicionales si es necesario, por ejemplo, para cálculos de eficiencia o generación de datos para pruebas.

**4. Makefile**

Los estudiantes deberán crear un **Makefile** que contenga al menos las siguientes configuraciones:

* **Debug**: Incluir símbolos de depuración (-g), desactivar optimización.
* **Release**: Activar optimización (-O3), desactivar símbolos de depuración.
* **Limpiar**: Eliminar archivos generados como .o y el ejecutable.
* **Compilación**: Incluir compilación de todos los archivos fuente en sus correspondientes archivos .o y luego enlazarlos en un solo ejecutable.

#### 5. ****Pruebas de Eficiencia****

Después de implementar la simulación, deberán medir la **eficiencia de la simulación** tanto en configuración **Debug** como **Release**. Para esto, pueden utilizar el chrono de C++ o el programa time, para medir el tiempo de ejecución en ambas configuraciones y compararlas.

**6. Requisitos Finales**

* Deben generar al menos **2 simulaciones** para **Debug** y **Release** con diferentes configuraciones de partículas y campos eléctricos.
* Al final, deberán escribir un breve reporte en el que se incluyan:
  + Resultados de la simulación.
  + Tiempo de ejecución en ambas configuraciones.
  + Posibles mejoras para optimizar el código.
  + Conclusiones sobre cómo las optimizaciones afectan el desempeño de la simulación.