Parte A: Movimiento Browniano

Generated by Doxygen 1.9.4

Chapter 1

Parte A: Simulación del Movimiento Browniano

Este directorio contiene el código y los artefactos para la Parte A del proyecto final de Física Computacional II, enfocada en la simulación del movimiento Browniano.

1.1 Descripción

Se simula el movimiento de una partícula browniana inmersa en un medio viscoso utilizando la ecuación de Langevin. La integración numérica se realiza mediante el método de Euler-Maruyama. El proyecto está estructurado con Programación Orientada a Objetos en C++.

El objetivo es observar la trayectoria característica de una partícula browniana y analizar su comportamiento difusivo.

1.1.1 Clases Principales:

- `Vector3D`: Clase auxiliar para operaciones con vectores en 3D.
- `ParticulaBrowniana`: Representa la partícula, implementando la lógica de la ecuación de Langevin.
- `SimuladorBrowniano`: Gestiona la simulación, el paso del tiempo y el guardado de datos.

1.2 Estructura del Directorio

La estructura del proyecto sigue las buenas prácticas para la organización de código C++.

```
ParteA/
|-- bin/
|   `-- movimiento_browniano
|-- include/
|   |-- ParticulaBrowniana.h
|   `-- SimuladorBrowniano.h
|-- src/
|   |-- main_browniano.cpp
|   |-- ParticulaBrowniana.cpp
|   |-- SimuladorBrowniano.cpp
|   `-- Vector3D.cpp
|-- scripts/
|   `-- plot_browniano.gp
|-- results/
|   |-- browniano_sim.dat
|   `-- (gráficas .png)
|-- documents/
|   |-- browniano_informe.tex
|   `-- (html_browniano/ y latex_browniano/)
|-- Makefile
|-- Doxyfile
`-- README.md
```

1.3 Compilación

Desde el directorio `ParteA/`, ejecuta:
```
make
```

Esto compilará todo el proyecto y creará el ejecutable `bin/movimiento_browniano`. Para una recompilación limpia:
```
make clean && make
```

1.4 Ejecución

Desde el directorio `ParteA/`, usa la regla `run` del Makefile o ejecuta el programa directamente.

Usando Makefile (recomendado):
```
make run
```

Esto ejecutará la simulación con los parámetros por defecto definidos en el `Makefile`.

Ejecución directa:
```
./bin/movimiento_browniano <tiempo_total> <dt> <semilla> [nombre_base_archivo]
```

Ejemplo:
```
./bin/movimiento_browniano 200.0 0.01 42 mi_simulacion
```

Los datos de salida se guardarán en `results/`.

1.5 Visualización

Después de ejecutar la simulación, genera las gráficas con Gnuplot usando la regla del Makefile:
```
make plot
```

Esto buscará el archivo `results/browniano_sim.dat` y creará las gráficas correspondientes en la misma carpeta.

1.6 Documentación

El proyecto cuenta con dos formas de documentación:

- **Informe Científico (LaTeX):** El informe detallado se encuentra en `documents/browniano_informe.tex`. Para compilarlo a PDF, necesitarás una distribución de LaTeX (como TeX Live o MiKTeX) y ejecutar desde `ParteA/documents/`:
  ```
  pdflatex browniano_informe.tex
  ```
- **Documentación del Código (Doxygen):** Para generar una página web navegable y un PDF con la documentación de todas las clases y funciones, usa las reglas del Makefile desde el directorio `ParteA/`:
  ```
  make doc  # Para generar solo el HTML
  make pdf  # Para generar HTML y el PDF final
  ```
  El HTML estará en `ParteA/documents/html_browniano/index.html` y el PDF en `ParteA/documents/latex_browniano/refman.pdf`.

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

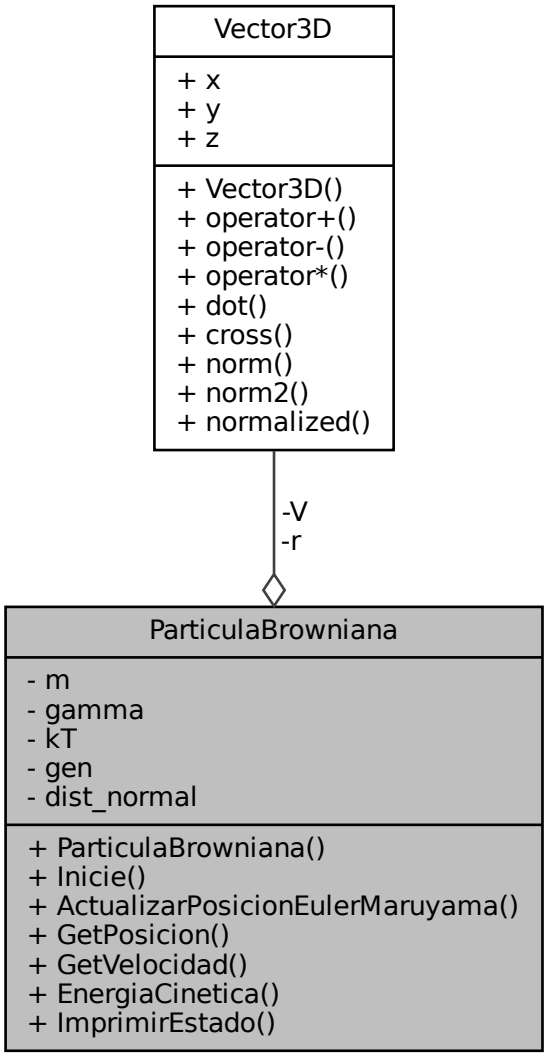
Class Documentation

4.1 ParticulaBrowniana Class Reference

Representa una partícula puntual que experimenta movimiento browniano.

`#include <ParticulaBrowniana.h>`

Collaboration diagram for ParticulaBrowniana:



Public Member Functions

- ParticulaBrowniana ()

  Constructor por defecto.
- void Inicie (double x0, double y0, double z0, double Vx0, double Vy0, double Vz0, double m0, double gamma0, double Temperatura_K, unsigned int semilla=std::random_device{}())

  Inicializa el estado completo de la partícula.
- void ActualizarPosicionEulerMaruyama (double dt)

  Avanza la simulación un paso de tiempo usando el método de Euler-Maruyama.
- Vector3D GetPosicion (void) const

  Obtiene la posición actual de la partícula.
- Vector3D GetVelocidad (void) const

Obtiene la velocidad actual de la partícula.

- double EnergiaCinetica (void) const

  Calcula la energía cinética instantánea de la partícula.
- void ImprimirEstado (std::ostream &os) const

  Imprime el estado actual de la partícula a un stream de salida.

Private Attributes

- Vector3D r
- Vector3D V

  Vector de posición (r) y velocidad (V) de la partícula.
- double m

  Masa de la partícula.
- double gamma

  Coeficiente de fricción o arrastre viscoso.
- double kT

  Energía térmica (Constante de Boltzmann $*$ Temperatura).
- std::mt19937 gen

  Generador de números aleatorios (Mersenne Twister).
- std::normal_distribution dist_normal

  Distribución normal estándar N(0,1).

4.1.1 Detailed Description

Representa una partícula puntual que experimenta movimiento browniano.

Esta clase encapsula las propiedades físicas de una partícula (posición, velocidad, masa) y las fuerzas que actúan sobre ella (arrastre viscoso y fuerza estocástica). La evolución temporal se calcula mediante el método de Euler-Maruyama.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 ParticulaBrowniana()

```
ParticulaBrowniana::ParticulaBrowniana ( )
```

Constructor por defecto.

4.1.3 Member Function Documentation

4.1.3.1 ActualizarPosicionEulerMaruyama()

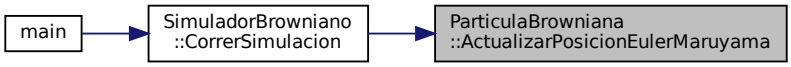
```
void ParticulaBrowniana::ActualizarPosicionEulerMaruyama (
            double dt )
```

Avanza la simulación un paso de tiempo usando el método de Euler-Maruyama.

Parameters

dt	El paso de tiempo para la integración.

Here is the caller graph for this function:



4.1.3.2 EnergiaCinetica()

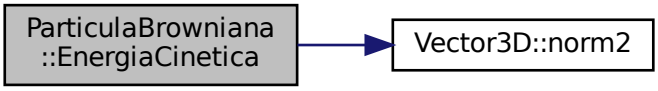
```
double ParticulaBrowniana::EnergiaCinetica (
            void  ) const
```

Calcula la energía cinética instantánea de la partícula.

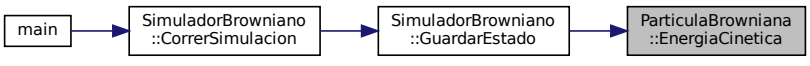
Returns

El valor de la energía cinética (0.5 ∗ m ∗ V^2).

Here is the call graph for this function:



Here is the caller graph for this function:



4.1.3.3 GetPosicion()

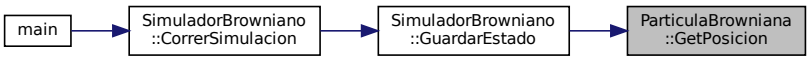
```
Vector3D ParticulaBrowniana::GetPosicion (
            void  ) const  [inline]
```

Obtiene la posición actual de la partícula.

Returns

Un Vector3D con la posición (r).

Here is the caller graph for this function:



4.1.3.4 GetVelocidad()

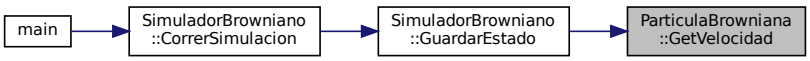
```
Vector3D ParticulaBrowniana::GetVelocidad (
            void  ) const  [inline]
```

Obtiene la velocidad actual de la partícula.

Returns

Un Vector3D con la velocidad (V).

Here is the caller graph for this function:



4.1.3.5 ImprimirEstado()

```
void ParticulaBrowniana::ImprimirEstado (
            std::ostream & os ) const
```

Imprime el estado actual de la partícula a un stream de salida.

Parameters

os	El stream de salida (ej. std::cout o un std::ofstream).

4.1.3.6 Inicie()

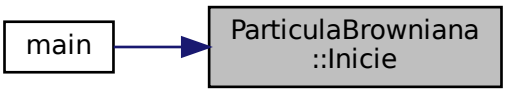
```
void ParticulaBrowniana::Inicie (
            double x0,
            double y0,
            double z0,
            double Vx0,
            double Vy0,
            double Vz0,
            double m0,
            double gamma0,
            double Temperatura_K,
            unsigned int semilla = std::random_device{}() )
```

Inicializa el estado completo de la partícula.

Parameters

x0	Componente x de la posición inicial.
y0	Componente y de la posición inicial.
z0	Componente z de la posición inicial.
Vx0	Componente x de la velocidad inicial.
Vy0	Componente y de la velocidad inicial.
Vz0	Componente z de la velocidad inicial.
m0	Masa de la partícula.
gamma0	Coeficiente de arrastre.
Temperatura_K	Temperatura del fluido en Kelvin.
semilla	Semilla para el generador de números aleatorios.

Here is the caller graph for this function:



4.1.4 Member Data Documentation

4.1.4.1 dist_normal

```
std::normal_distribution ParticulaBrowniana::dist_normal  [private]
```

Distribución normal estándar N(0,1).

4.1.4.2 gamma

```
double ParticulaBrowniana::gamma  [private]
```

Coeficiente de fricción o arrastre viscoso.

4.1.4.3 gen

```
std::mt19937 ParticulaBrowniana::gen  [private]
```

Generador de números aleatorios (Mersenne Twister).

4.1.4.4 kT

```
double ParticulaBrowniana::kT  [private]
```

Energía térmica (Constante de Boltzmann $*$ Temperatura).

4.1.4.5 m

```
double ParticulaBrowniana::m  [private]
```

Masa de la partícula.

4.1.4.6 r

```
Vector3D ParticulaBrowniana::r  [private]
```

4.1.4.7 V

```
Vector3D ParticulaBrowniana::V  [private]
```

Vector de posición (r) y velocidad (V) de la partícula.

The documentation for this class was generated from the following files:

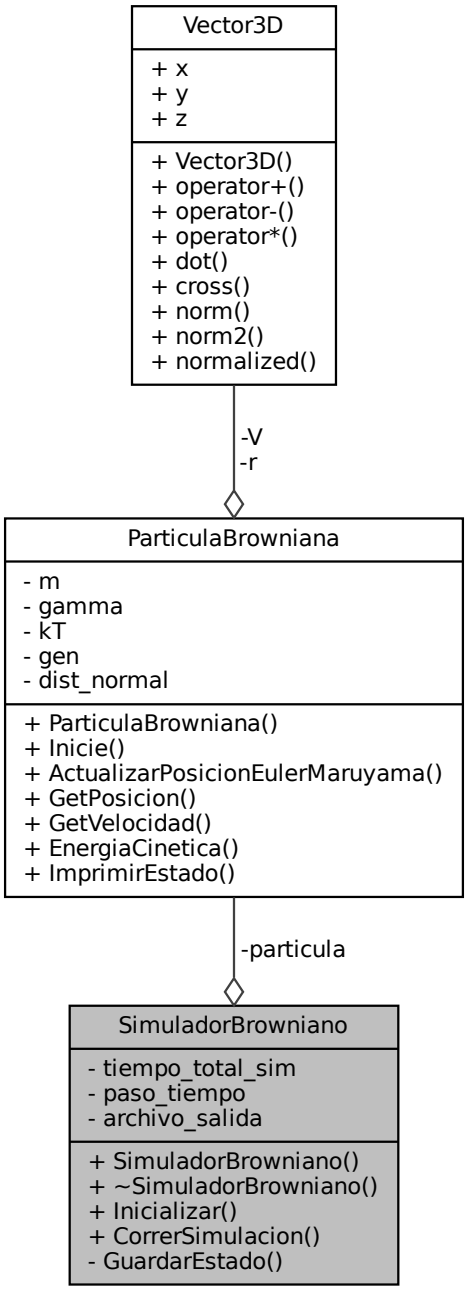
- include/ParticulaBrowniana.h
- src/ParticulaBrowniana.cpp

4.2 SimuladorBrowniano Class Reference

Orquesta la simulación del movimiento browniano para una partícula.

```
#include <SimuladorBrowniano.h>
```

Collaboration diagram for SimuladorBrowniano:



Public Member Functions

- SimuladorBrowniano ()

  Constructor por defecto.
- ~SimuladorBrowniano ()

  Destructor.

- void Inicializar (double t_total, double dt, const ParticulaBrowniana &p_inicial, const std::string &nombre_↩ base_archivo)

  Configura e inicializa la simulación.
- void CorrerSimulacion ()

  Ejecuta el bucle principal de la simulación.

Private Member Functions

- void GuardarEstado (double tiempo)

  Guarda el estado actual de la partícula en el archivo de datos.

Private Attributes

- ParticulaBrowniana particula

  La partícula que será simulada.
- double tiempo_total_sim

  Duración total de la simulación.
- double paso_tiempo

  Paso de tiempo (dt) para la integración.
- std::ofstream archivo_salida

  Stream para escribir los datos en un archivo.

4.2.1 Detailed Description

Orquesta la simulación del movimiento browniano para una partícula.

Esta clase se encarga de inicializar los parámetros de la simulación (tiempo, dt), manejar el archivo de salida de datos, y ejecutar el bucle principal que actualiza el estado de la partícula a lo largo del tiempo.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 SimuladorBrowniano()

```
SimuladorBrowniano::SimuladorBrowniano ( )
```

Constructor por defecto.

4.2.2.2 ~SimuladorBrowniano()

```
SimuladorBrowniano::~SimuladorBrowniano ( )
```

Destructor.

Cierra el archivo de salida si está abierto.

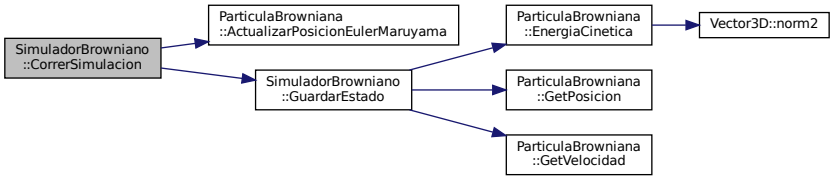
4.2.3 Member Function Documentation

4.2.3.1 CorrerSimulacion()

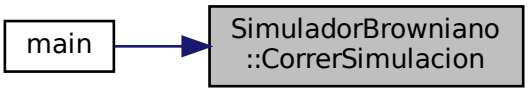
```
void SimuladorBrowniano::CorrerSimulacion ( )
```

Ejecuta el bucle principal de la simulación.

Itera desde t=0 hasta t=tiempo_total_sim, actualizando la partícula y guardando su estado periódicamente. Here is the call graph for this function:



Here is the caller graph for this function:



4.2.3.2 GuardarEstado()

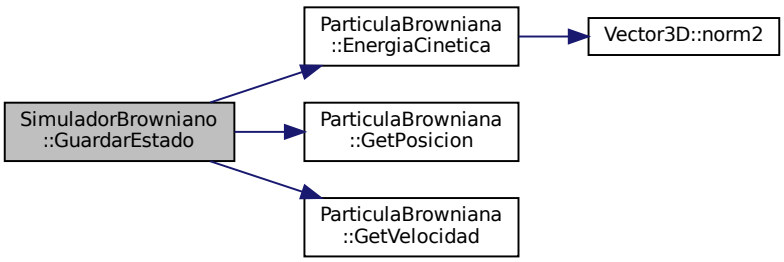
```
void SimuladorBrowniano::GuardarEstado (
            double tiempo )  [private]
```

Guarda el estado actual de la partícula en el archivo de datos.

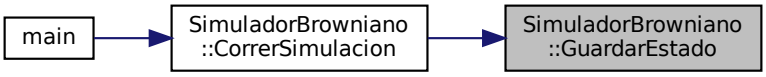
Parameters

tiempo	El tiempo actual de la simulación para registrar en la primera columna.

Here is the call graph for this function:



Here is the caller graph for this function:



4.2.3.3 Inicializar()

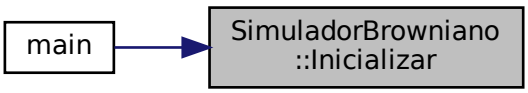
```
void SimuladorBrowniano::Inicializar (
            double t_total,
            double dt,
            const ParticulaBrowniana & p_inicial,
            const std::string & nombre_base_archivo )
```

Configura e inicializa la simulación.

Parameters

t_total	Duración total de la simulación.
dt	Paso de tiempo para la integración.
p_inicial	La partícula ya configurada que se va a simular.
nombre_base_archivo	El nombre base para el archivo de datos de salida (sin extensión).

Here is the caller graph for this function:



4.2.4 Member Data Documentation

4.2.4.1 archivo_salida

```
std::ofstream SimuladorBrowniano::archivo_salida  [private]
```

Stream para escribir los datos en un archivo.

4.2.4.2 particula

```
ParticulaBrowniana SimuladorBrowniano::particula  [private]
```

La partícula que será simulada.

4.2.4.3 paso_tiempo

```
double SimuladorBrowniano::paso_tiempo  [private]
```

Paso de tiempo (dt) para la integración.

4.2.4.4 tiempo_total_sim

```
double SimuladorBrowniano::tiempo_total_sim  [private]
```

Duración total de la simulación.

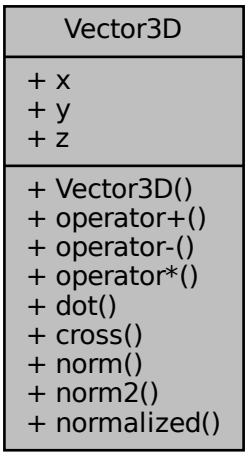
The documentation for this class was generated from the following files:

- include/SimuladorBrowniano.h
- src/SimuladorBrowniano.cpp

4.3 Vector3D Class Reference

`#include <Vector3D.h>`

Collaboration diagram for Vector3D:



Public Member Functions

- Vector3D (double x=0, double y=0, double z=0)
- Vector3D operator+ (const Vector3D &other) const
- Vector3D operator- (const Vector3D &other) const
- Vector3D operator∗ (double scalar) const
- double dot (const Vector3D &other) const
- Vector3D cross (const Vector3D &other) const
- double norm () const
- double norm2 () const
- Vector3D normalized () const

Public Attributes

- double x
- double y
- double z

Friends

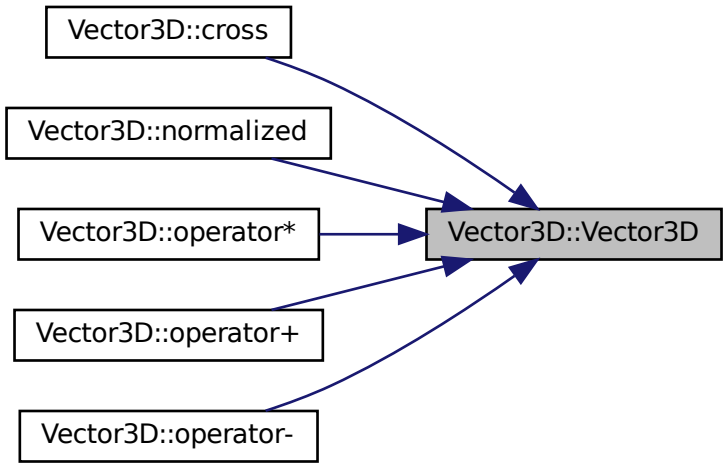
- std::ostream & operator<< (std::ostream &os, const Vector3D &vec)

4.3.1 Constructor & Destructor Documentation

4.3.1.1 Vector3D()

```
Vector3D::Vector3D (
            double x = 0,
            double y = 0,
            double z = 0 )
```

Here is the caller graph for this function:

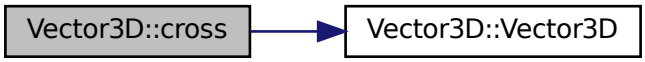


4.3.2 Member Function Documentation

4.3.2.1 cross()

```
Vector3D Vector3D::cross (
            const Vector3D & other ) const
```

Here is the call graph for this function:



4.3.2.2 dot()

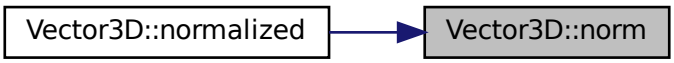
```
double Vector3D::dot (
            const Vector3D & other ) const
```

4.3.2.3 norm()

```
double Vector3D::norm ( ) const
```

Here is the call graph for this function:

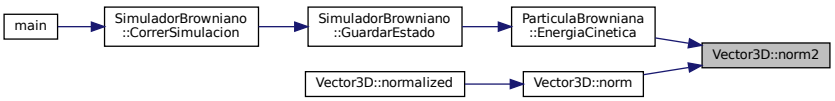
Here is the caller graph for this function:



4.3.2.4 norm2()

```
double Vector3D::norm2 ( ) const
```

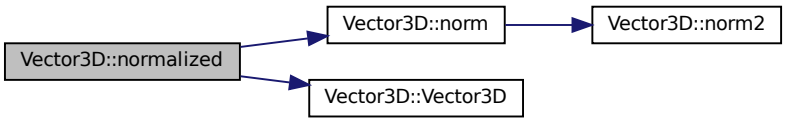
Here is the caller graph for this function:



4.3.2.5 normalized()

```
Vector3D Vector3D::normalized ( ) const
```

Here is the call graph for this function:



4.3.2.6 operator∗()

```
Vector3D Vector3D::operator* (
            double scalar ) const
```

Here is the call graph for this function:

4.3.2.7 operator+()

```
Vector3D Vector3D::operator+ (
            const Vector3D & other ) const
```

Here is the call graph for this function:

4.3.2.8 operator-()

```
Vector3D Vector3D::operator- (
            const Vector3D & other ) const
```

Here is the call graph for this function:

4.3.3 Friends And Related Function Documentation

4.3.3.1 operator<<

```
std::ostream & operator<< (
            std::ostream & os,
            const Vector3D & vec )  [friend]
```

4.3.4 Member Data Documentation

4.3.4.1 x

```
double Vector3D::x
```

4.3.4.2 y

```
double Vector3D::y
```

4.3.4.3 z

```
double Vector3D::z
```

The documentation for this class was generated from the following files:

- include/Vector3D.h
- src/Vector3D.cpp

Chapter 5

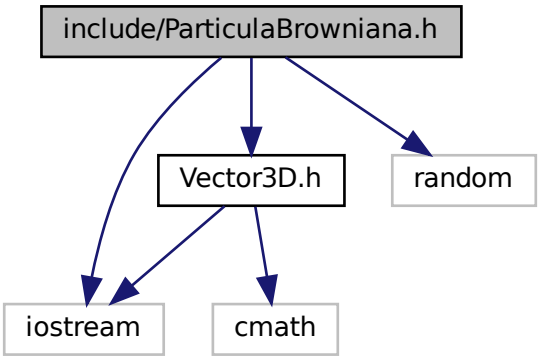
File Documentation

5.1 include/ParticulaBrowniana.h File Reference

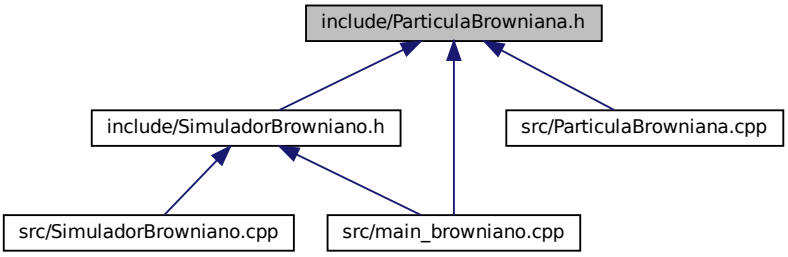
Declaración de la clase ParticulaBrowniana, que modela una partícula sometida a la ecuación de Langevin en 3D.

```
#include "Vector3D.h"
#include <random>
#include <iostream>
```

Include dependency graph for ParticulaBrowniana.h:



This graph shows which files directly or indirectly include this file:



Classes

- class ParticulaBrowniana

  Representa una partícula puntual que experimenta movimiento browniano.

5.1.1 Detailed Description

Declaración de la clase ParticulaBrowniana, que modela una partícula sometida a la ecuación de Langevin en 3D.

Author

Camilo Andres Huertas Archila

Date

2025-07-09

5.2 ParticulaBrowniana.h

Go to the documentation of this file.

```
1
9  #ifndef PARTICULA_BROWNIANA_H
10 #define PARTICULA_BROWNIANA_H
11
12 #include "Vector3D.h"
13 #include <random>
14 #include <iostream>
15
25 class ParticulaBrowniana {
26 private:
27     Vector3D r, V;
28     double m;
29     double gamma;
30     double kT;
31
32     std::mt19937 gen;
33     std::normal_distribution<> dist_normal;
34
35 public:
39     ParticulaBrowniana();
40
54     void Inicie(double x0, double y0, double z0,
55                 double Vx0, double Vy0, double Vz0,
```

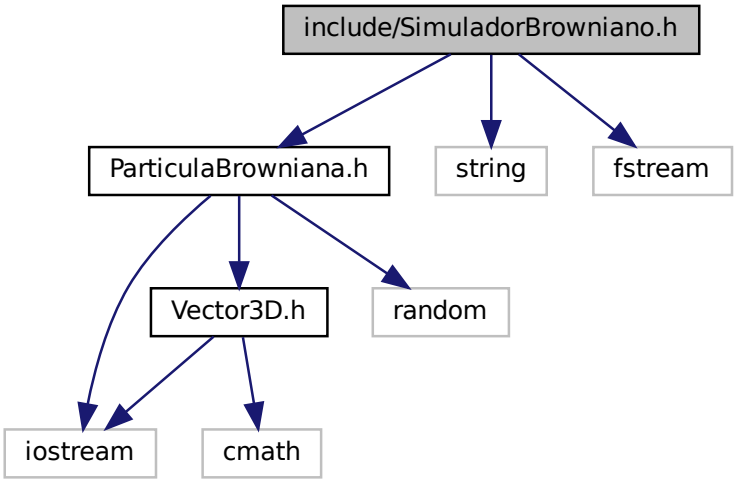
```
56                 double m0, double gamma0, double Temperatura_K, unsigned int semilla =
       std::random_device{}());
57
62     void ActualizarPosicionEulerMaruyama(double dt);
63
64     // --- Getters ---
65
70     Vector3D GetPosicion(void)const { return r; }
71
76     Vector3D GetVelocidad(void)const { return V; }
77
82     double EnergiaCinetica(void) const;
83
88     void ImprimirEstado(std::ostream& os) const;
89 };
90
91 #endif // PARTICULA_BROWNIANA_H
92
```

5.3 include/SimuladorBrowniano.h File Reference

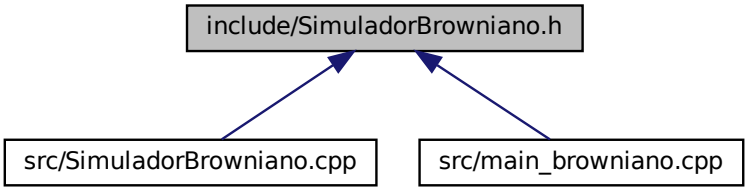
Declaración de la clase SimuladorBrowniano, que gestiona la simulación completa.

```
#include "ParticulaBrowniana.h"
#include <string>
#include <fstream>
```

Include dependency graph for SimuladorBrowniano.h:



This graph shows which files directly or indirectly include this file:



Classes

- class SimuladorBrowniano

  Orquesta la simulación del movimiento browniano para una partícula.

5.3.1 Detailed Description

Declaración de la clase SimuladorBrowniano, que gestiona la simulación completa.

Author

Camilo Andres Huertas Archila

Date

2025-07-09

5.4 SimuladorBrowniano.h

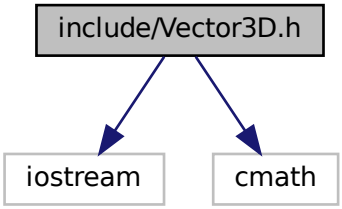
Go to the documentation of this file.

```
1
8 #ifndef SIMULADOR_BROWNIANO_H
9 #define SIMULADOR_BROWNIANO_H
10
11 #include "ParticulaBrowniana.h"
12 #include <string>
13 #include <fstream> // Para std::ofstream
14
23 class SimuladorBrowniano {
24 private:
25     ParticulaBrowniana particula;
26     double tiempo_total_sim;
27     double paso_tiempo;
28     std::ofstream archivo_salida;
29
30 public:
34     SimuladorBrowniano();
35
39     ~SimuladorBrowniano();
40
48     void Inicializar(double t_total, double dt, const ParticulaBrowniana& p_inicial, const std::string&
     nombre_base_archivo);
49
55     void CorrerSimulacion();
56
57 private:
62     void GuardarEstado(double tiempo);
63 };
64
65 #endif // SIMULADOR_BROWNIANO_H
66
```

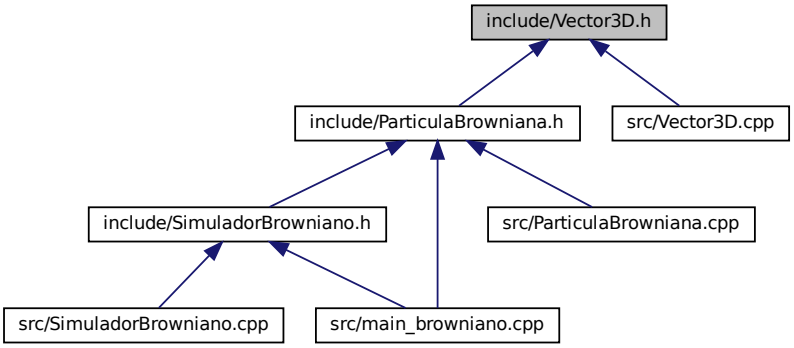
5.5 include/Vector3D.h File Reference

```
#include <iostream>
#include <cmath>
```

Include dependency graph for Vector3D.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Vector3D

5.6 Vector3D.h

Go to the documentation of this file.

```
1 // Vector3D.h
2 #ifndef VECTOR3D_H
3 #define VECTOR3D_H
4
5 #include <iostream>
6 #include <cmath> // Para std::sqrt
7
8 class Vector3D {
```

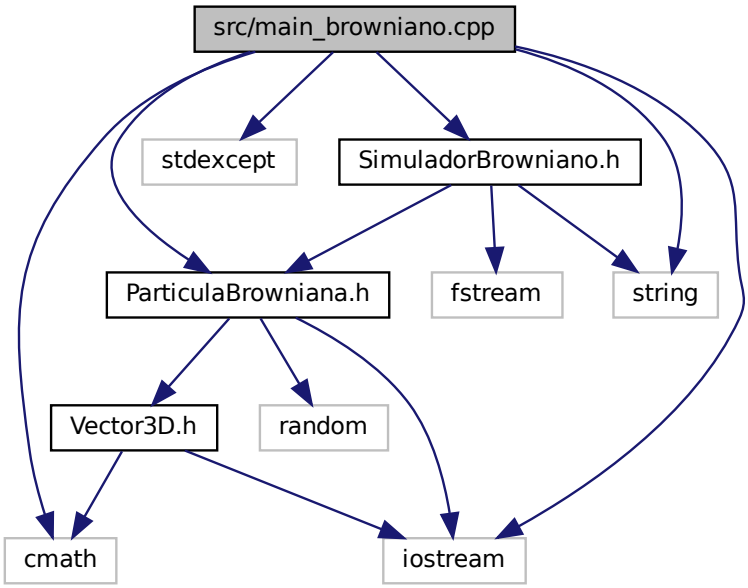
```
9 public:
10     double x, y, z;
11
12     // Constructores
13     Vector3D(double x = 0, double y = 0, double z = 0);
14
15     // Sobrecarga de operadores
16     Vector3D operator+(const Vector3D& other) const;
17     Vector3D operator-(const Vector3D& other) const;
18     Vector3D operator*(double scalar) const;
19     double dot(const Vector3D& other) const; // Producto punto
20     Vector3D cross(const Vector3D& other) const; // Producto cruz
21
22     // Otros métodos útiles
23     double norm() const; // Magnitud del vector
24     double norm2() const; // Magnitud al cuadrado
25     Vector3D normalized() const; // Vector unitario
26
27     // Sobrecarga del operador de inserción para imprimir
28     friend std::ostream& operator<<(std::ostream& os, const Vector3D& vec);
29 };
30
31 #endif // VECTOR3D_H
```

5.7 README.md File Reference

5.8 src/main_browniano.cpp File Reference

```
#include <iostream>
#include <string>
#include <stdexcept>
#include <cmath>
#include "ParticulaBrowniana.h"
#include "SimuladorBrowniano.h"
```

Include dependency graph for main_browniano.cpp:



Functions

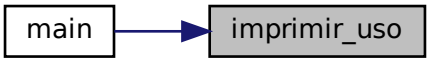
- void imprimir_uso (const char *nombre_programa)
- int main (int argc, char *argv[ ])

5.8.1 Function Documentation

5.8.1.1 imprimir_uso()

```
void imprimir_uso (
            const char * nombre_programa )
```

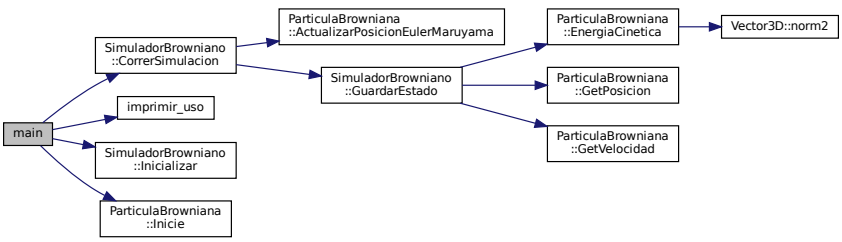
Here is the caller graph for this function:



5.8.1.2 main()

```
int main (
            int argc,
            char * argv[] )
```

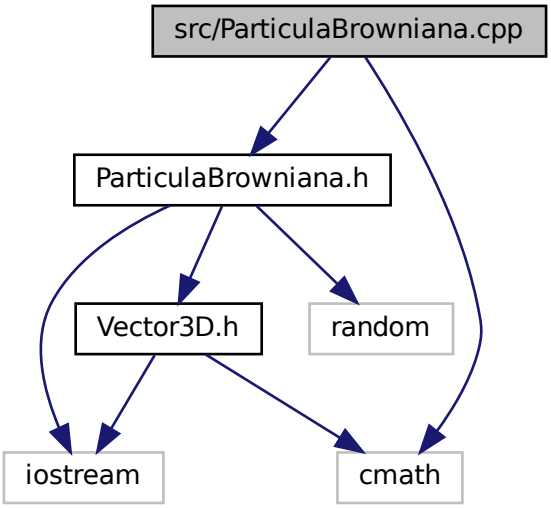
Here is the call graph for this function:



5.9 src/ParticulaBrowniana.cpp File Reference

```
#include "ParticulaBrowniana.h"
#include <cmath>
```

Include dependency graph for ParticulaBrowniana.cpp:



Variables

- const double K_BOLTZMANN = 1.380649e-23

5.9.1 Variable Documentation

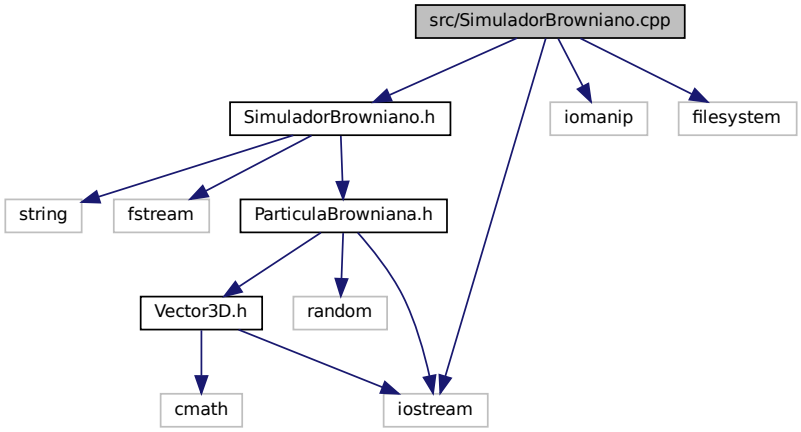
5.9.1.1 K_BOLTZMANN

```
const double K_BOLTZMANN = 1.380649e-23
```

5.10 src/SimuladorBrowniano.cpp File Reference

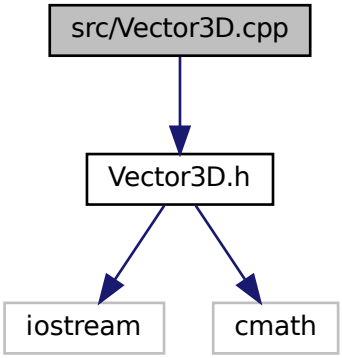
```
#include "SimuladorBrowniano.h"
#include <iostream>
#include <iomanip>
```

`#include <filesystem>`
Include dependency graph for SimuladorBrowniano.cpp:



5.11 src/Vector3D.cpp File Reference

`#include "Vector3D.h"`
Include dependency graph for Vector3D.cpp:



Functions

- std::ostream & operator<< (std::ostream &os, const Vector3D &vec)

5.11.1 Function Documentation

5.11.1.1 operator<<()

```
std::ostream & operator<< (
            std::ostream & os,
            const Vector3D & vec )
```

Index