Universidad del Bío-Bío. Sistema de Bibliotecas - Chile

Universidad del Bío-Bío
Facultad de Ciencias Empresariales
Departamento de Sistemas de Información

UNIVERSIDAD DEL BÍO-BÍO

Desarrollo de una aplicación gráfica en modalidad Cliente-Servidor para el monitoreo de simulaciones de Dinámica Molecular conducida por eventos

Julio Borja Barra

Proyecto para optar al título de
Ingeniero Civil Informático

Concepción - Chile
Agosto 2010

Universidad del Bío-Bío. Sistema de Bibliotecas - Chile

Universidad del Bío-Bío

Facultad de Ciencias Empresariales

Departamento de Sistemas de Información

UNIVERSIDAD DEL BÍO-BÍO

Desarrollo de una aplicación gráfica en modalidad Cliente-Servidor para el monitoreo de simulaciones de Dinámica Molecular conducida por eventos

Autor:	**Julio Borja Barra**
Profesores Guías:	**Sergio Bravo Silva**
	Dino Risso Rocco
Profesor Colaborador:	**Rodrigo Soto**

Concepción - Chile

Agosto 2010

Universidad del Bío-Bío. Sistema de Bibliotecas - Chile

Índice general

1. Introducción **1**

1.1. Los usuarios 1

1.2. Acerca de las simulaciones computacionales 2

1.2.1. Sobre las simulaciones de Dinámica Molecular 4

1.2.1.1. Simulaciones conducidas por tiempo 5

1.2.1.2. Simulaciones conducidas por eventos 7

1.2.1.3. Comparación entre las simulaciones conducidas por eventos y las conducidas por tiempo 11

1.3. Materiales Granulares 12

1.3.1. Aplicaciones en la Industria 12

1.3.1.1. Simulaciones de medios granulados 15

2. El problema a resolver **18**

2.1. Historia del proceso 18

2.2. El Problema a Resolver 20

3. Diseño de la solución **21**

3.1. Requerimientos 21

3.1.1. Requerimientos Funcionales 21

3.1.2. Requerimientos No-Funcionales 21

3.2. Objetivos 22

3.2.1. Objetivo General 22

3.2.2. Objetivos Específicos 22

3.3. Estudio de Factibilidad 23

3.3.1. Factibilidad Técnica 23

3.3.2. Factibilidad Operativa 23

3.3.3. Factibilidad Económica 23

Universidad del Bío-Bío. Sistema de Bibliotecas - Chile

3.4. Arquitectura del Sistema 26
3.4.1. Diagramas de Flujo de Datos 27
3.4.2. Casos de Uso 29
3.4.2.1. Descripción de Casos de Uso 29

4. Implementación de la solución **32**
4.1. Protocolo de Comunicación 32
4.2. Programas asociados al monitor de simulaciones 32
4.3. Cómo funciona el Monitor de Simulaciones 33
4.4. Comportamiento del Programa 35
4.5. Algunas Interfaces 36

5. Requerimientos **39**
5.1. Hardware 39
5.2. Dependencias de Software 40

6. Puesta en marcha y Pruebas **41**
6.1. Cambios en los procedimientos 41
6.2. Pruebas 42
6.2.1. Pruebas de Unidad 42
6.2.1.1. Conexión 42
6.2.1.2. Historial 42
6.2.1.3. Animación Gráfica con LibPlot 43
6.2.1.4. Animación Gráfica con OpenGL 43
6.2.2. Pruebas de Integración 44
6.2.2.1. Animación y Aumento del Coeficiente de Inelasticidad 44
6.2.2.2. Animación y Comenzar a medir la temperatura 45

7. Conclusiones y Proyecciones **46**
7.1. Aprendizajes Logrados 46
7.2. Proyecciones y Trabajos Futuros 48
7.3. Licencia 48

A. Diccionario de Datos (DFD) **49**

B. Documentación **53**

Universidad del Bío-Bío. Sistema de Bibliotecas - Chile

B.1. main.c Referencia del Archivo 53
B.1.1. Documentación de Definiciones 55
B.1.1.1. _client_ 55
B.1.1.2. VERSION 55
B.1.2. Documentación de Funciones 55
B.1.2.1. bienvenida 55
B.1.2.2. create_socket 55
B.1.2.3. main 55
B.1.2.4. ncurses_init 55
B.1.2.5. recv_data 56
B.1.2.6. salir 56
B.1.2.7. send_msg 56
B.1.3. Documentación de Variables 56
B.1.3.1. footer 56
B.1.3.2. hist_file_name 56
B.1.3.3. len 57
B.1.3.4. numbytes 57
B.1.3.5. posiciones 57
B.1.3.6. Puerto 57
B.1.3.7. recvbuffer 57
B.1.3.8. running_sim 57
B.1.3.9. sendbuffer 57
B.1.3.10. serv_addr 57
B.2. Command.c Referencia del Archivo 57
B.2.1. Documentación de Definiciones 58
B.2.1.1. _Command_ 58
B.2.2. Documentación de Funciones 58
B.2.2.1. cmd_count_args 58
B.2.2.2. cmd_execute 59
B.2.2.3. cmd_parse 59
B.2.2.4. cmd_to_int 59
B.2.2.5. display_help 59
B.2.2.6. glut_ 60

Universidad del Bío-Bío. Sistema de Bibliotecas - Chile

B.2.2.7. glut_cmd_to_int 60
B.3. glut.cpp Referencia del Archivo 60
B.3.1. Documentación de Definiciones 61
B.3.1.1. _grglut_ 61
B.3.2. Documentación de Funciones 61
B.3.2.1. ChangeSize 61
B.3.2.2. getPos 61
B.3.2.3. glut 62
B.3.2.4. IdleFunc 62
B.3.2.5. Menu 62
B.3.2.6. RenderScene 62
B.3.2.7. SetupRC 62
B.3.2.8. SpecialKeys 62
B.3.2.9. SubMenuQn 62
B.3.2.10. SubMenuTemp 62
B.3.2.11. Timer 63
B.3.3. Documentación de Variables 63
B.3.3.1. camara 63
B.3.3.2. globFact 63
B.3.3.3. pause_anim 63
B.3.3.4. submenuQn 63
B.3.3.5. submenuTemp 63
B.3.3.6. temp_med 63
B.3.3.7. window_id 63
B.3.3.8. windowHeight 63
B.3.3.9. windowWidth 63
B.3.3.10. X1 63
B.3.3.11. X2 63
B.3.3.12. x_GLUT 63
B.3.3.13. Y1 63
B.3.3.14. Y2 63
B.3.3.15. y_GLUT 63
B.3.3.16. Z1 63

Universidad del Bío-Bío. Sistema de Bibliotecas - Chile

B.3.3.17. Z2 63
B.3.3.18. z_GLUT 63

Universidad del Bío-Bío. Sistema de Bibliotecas - Chile

A mis padres por su entrega y sacrificio,
por su alegría y perseverancia.
Por sus errores y aciertos que me motivaron
a buscar la grandeza en las cosas pequeñas y
me enseñaron que los medios justifican los fines.

A Dino por su amor a lo que hace, por poner su ética sobre todo
y por su infinito amor y dedicación a la docencia.

Y a todos los que me han enseñado con su ejemplo
que antes de ser ingenieros, abogados o electricistas
somos humanos.

"Para pequeñas criaturas como nosotros, la inmensidad del universo es soportable sólo a través del amor" –Carl Sagan.

Julio Borja Barra

Universidad del Bío-Bío. Sistema de Bibliotecas - Chile

Universidad del Bío-Bío. Sistema de Bibliotecas - Chile

Capítulo 1

Introducción

El presente trabajo tiene como objetivo describir el proceso de investigación, diseño y desarrollo del Monitor de Simulaciones, aplicación en modalidad Cliente/Servidor con interfaz gráfica 3D, al que llamaremos SiMon (por su abreviación del inglés: “Simulation Monitor”).
La primera parte del texto está enfocada en describir a los usuarios para los cuales ha sido diseñada esta herramienta. A continuación se explica de qué se tratan las simulaciones computacionales, cuál es su relación con las experiencias reales en laboratorios, cuál es su importancia en la industria y más tarde se justifica esta investigación.
Finalmente, el escrito mostrará los motivos que hicieron necesaria la construcción de esta aplicación y las características fundamentales que debe poseer la misma para ser aceptada como primera versión.

1.1. Los usuarios

El *Grupo de Dinámica Molecular y Teoría Cinética* (GDMTC) al cual va orientado el trabajo de la presente memoria de título está conformado por los académicos Patricio Cordero S. y Rodrigo Soto B. de Universidad de Chile, y el académico Dino E. Risso de Universidad del Bío-Bío (ver URL[1]).

Este grupo de investigadores tiene una tradición de investigación en el área de la Dinámica Molecular y de la Teoría Cinética de alrededor de 15 años. Habiendo focalizando –en una primera etapa– su investigación al estudio de gases enrarecidos actualmente dirigen sus esfuerzos al estudio de **medios granulados**, donde colaboran activamente con otros investigadores a nivel mundial y con el *Laboratorio de Física No Lineal* que lidera Nicolás Mujica[2]).

(U. de Chile).

Las herramientas de análisis del GDMTC son principalmente de caracter teórico y simulacional. Particularmente en esta última área este grupo ha desarrollado un **software de simulación para partículas duras tipo esferas** de alta complejidad y eficiencia que se puede considerar como el más óptimo de su tipo [1, 2]. En este trabajo nos referiremos a ese software como “*el simulador*”.

Previo a detallar el tipo de herramientas de software y de hardware que el grupo utiliza dedicaremos

[1] http://dfi.uchile.cl/cinetica
http://maxwell.ciencias.ubiobio.cl/~drisso/
[2] http://www.dfi.uchile.cl/~nmujica/Sitio_web/Home.html

Universidad del Bío-Bío. Sistema de Bibliotecas - Chile

algunos párrafos a describir los aspectos históricos involucrados en las técnicas de simulación que son de interés para este grupo.

1.2. Acerca de las simulaciones computacionales

Las *simulaciones computacionales* surgen como una aplicación práctica inmediata de las *máquinas de procesamiento electrónico* que se desarrollaron durante y después de la Segunda Guerra Mundial. Estas máquinas se construyeron con el propósito de ayudar en los complejos y exigentes cálculos involucrados en el desarrollo de las armas nucleares. A comienzos de la década del 50, los computadores electrónicos estuvieron disponibles para propósitos civiles, lo que dió inicio a la disciplina de la *simulación computacional*. Las primeras aplicaciones ocurrieron a comienzos de 1952, sobre MANIAC (del inglés "*Mathematical Analyzer, Numerical Integrator, and Computer*" (ver Fig. 1.1) en Los Alamos, National Laboratory en University of New Mexico, lideradas principalmente por Nicholas Constantine Metrópolis, quién estaba interesado en usar estas máquinas para los problemas más diversos posibles con el objeto de evaluar su estructura lógica y sus capacidades.

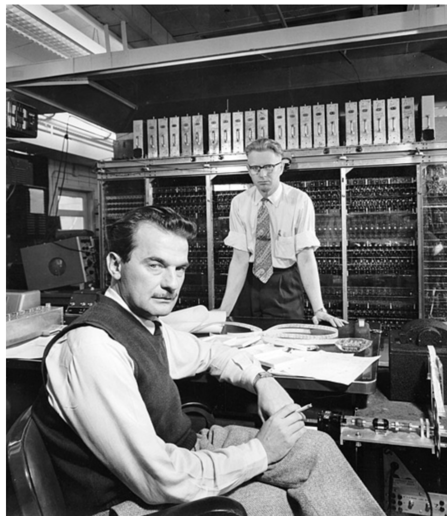


Figura 1.1: Al fondo MANIAC, en primer plano algunos físicos de la época.

Entre las primeras aplicaciones de MANIAC está el estudio de fluidos densos, donde se hace uso del actualmente conocidísimo Metodo Montecarlo introducido por Metropolis et. al [3]. El nombre Montecarlo se debe a que el método estadístico de estudio hace un uso muy intenso de números pseudo-aleatorios o pseudo-azarosos generados vía el computador.

Una segunda aplicación es lo que actualmente se conoce como Dinamica Molecular (DM). Los orígenes de la dinámica molecular ya están en los atomistas de la antiguedad, con su creencia que la materia podría subdividirse hasta un cierto grado donde ella se descomponía en objetos elementales, moléculas o átomos. Con la teoría de la Mecánica de Newton surge la posibilidad de usar el concepto de moléculas y átomos ocmo ingredientes básicos para construir una descripción de la materia a nivel macroscópico. Sin embargo esta tarea no es simple, el problema de la dinámica acoplada de N cuerpos es complejo y no es soluble analíticamente excepto para el caso particular de dos cuer-

Universidad del Bío-Bío. Sistema de Bibliotecas - Chile

pos, y en situaciones muy particulares en el problema de tres cuerpos [3]. Un tratamiento sistemático teórico del problema de N cuerpos no fue posible sino hasta Boltzmann en 1872, quíen propuso un tratamiento estadístico basado en una teoría atomística para el estudio de las propiedades de los gases enrarecidos [4]. Las primeras simulacionales de la teoría atomística vía DM ocurren recién a partir de 1956 con Alder y Wainwright quienes realizaron un estudio de las propiedades dinámicas de un conjunto de esferas [5]. Es interesante nombrar aquí que éste era un tema de interés en la época pues el sistema de esferas durás constituye una base para el estudio de líquidos y gases densos y que en la época de Alder algunos de dichos estudios se realizaban empíricamente pegando esferas de goma entre sí, disponiéndolas de la manera más aleatoria posible y estudiando las estructuras de conjunto que resultaban. Un trabajo completamente tedioso, demoroso y propenso a errores. La primera simulación computacional "realista" se deben a Vineyard en 1959 quién estudio el daño por radiación en un cristal de cobre [6], y la primera simulación "realista" de las propiedades de un líquido (Argón) se debe a Rahman en 1964 [7]. Demás está decir que desde allí en adelante se ha dado enormes avances en el estudio de las propiedades de los líquidos, pero siempre dicho estudio ha estado asistido por las simulaciones computacionales.

Previo a que aparecieran las simulaciones computacionales existía sólo una forma de predecir las propiedades de una substancia molecular: a través de un teoría que proporcionara una descripción aproximada del material. Estas aproximaciones eran necesarias pues son muy pocos los sistemas que pueden calcularse en forma analítica o exacta (ejemplos de ellos son el modelo de gas ideal, el cristal armónico, y algunos modelos de red para el magnetismo). A partir de estos modelos se pudo predecir, en forma aproximada, muchas propiedades de algunos materiales reales. En principio si se tiene suficiente información del detalle de las interacciones moleculares, estas teorías debieran entregar una buena estimación de las propiedades de interés, sin embargo, este conocimiento, incluso para las moléculas más simples es limitado. Surge aquí una dificultad cuando se quiere validar una teoría comparándola directamente con el experimento: si la teoría y el experimento no concuerdan puede deberse a que la teoría es errónea o que se tiene una estimación incorrecta de las interacciones entre moléculas, o ambas.

Es en estos aspectos que las simulaciones computacionales vienen en ayuda.

(a) por un lado pueden entregar resultados esencialmente exactos para un modelo dado, sin hacer uso de teorías aproximadas.

(b) por otro, permiten comparar los resultados de la simulación de un modelo simple con los resultados analíticos de una teoría para ese mismo modelo.

En el primer rol, si no hay concordancia, se puede intentar mejorar el modelo para que los resultados de la simulación se parezcán mas a los del experimento. En el segundo rol, la no concordancia indica una falla en la teoría. En esta última situación la simulación computacional es un test de validez de la teoría.

Esta última modalidad de aplicación es lo que se conoce como *experimento computacional*, y es una de las formas de aplicación de las simulaciones más usadas. Practicamente no hay teoría que no sea comparada con simulaciones computacionales con el objeto de validarla antes de aplicarla en el mundo real.

[3]El problema de N cuerpos encuentra sus origenes en el estudio de la dinámica del sistema solar, donde se encuentra que en general este es insoluble si hay mas de 2 cuerpos.

Universidad del Bío-Bío. Sistema de Bibliotecas - Chile

En la primera modalidad la simulación tiene la ventaja que ella entrega al investigador una imagen de la física involucrada en un problema, al mismo tiempo que le genera resultados ”exactos” de un modelo simple y que pueda ser contrastado con la teoría a ser construida. En este sentido las simulaciones computacionales no siempre buscan precisión con la realidad material en un sentido matemático estricto, sino que —buena parte de las veces— fidelidad con las leyes fisicas subyacentes.

Muchas veces la necesidad de hacer simulaciones no es bien entendida. ¿Cuál es el sentido de estudiar el punto de congelamiento de un líquido, si podemos hacer un experimento para obtenerlo?. La respuesta mas directa es que si bien se puede hacer fácilmente un experimento para medir esta propiedad en condiciones de presión de 1 atmosfera, llevar a cabo el mismo experimento a muy altas presiones (digamos por ejemplo varios miles de atmósfera) es prácticamente imposible. En cambio al realizar una simulación computacional, no hay ningún riesgo de explosión al simular un proceso que corresponde a esas presiones. Además está el hecho, de que una vez que un modelo está bien establecido, muchas de las propiedades del modelo (por ejemplo el orden estructural en las macromoléculas de un material) pueden ser obtenidas vía la simulación computacional. Por otro lado hay variables que son directamente medibles en la simulación y que en un experimento real es muy difícil, o prácticamente imposible, de obtener.

Un valor agregado de la simulación y que resulta muy importante en aplicaciones de ingeniería, es que buena parte de los estudios experimentales de una gran variedad de problemas, tienen costos asociados —en tiempo y dinero—- que suelen ser excesivos y no están al alcance de muchas empresas, mientras que los costos asociados en montar una simulación pueden ser varios ordenes de magnitud menores.

Otro aspecto importante y ligado al párrafo anterior, es que la simulaciones puede tener caracter exploratorio: en el computador es posible variar parámetros del problema simplemente cambiando un valor inicial de montaje simulacional, o incluso dinámicamente mientras la simulación corre; incluso es posible poner valores de parámetros que experimentalmente no son alcanzables hoy día, pero que podrían ser alcanzables en un futuro (quizas en un trabajo de desarrollo motivado por el resultado de las mismas simulaciones).

1.2.1. Sobre las simulaciones de Dinámica Molecular

La primera simulación de dinámica molecular fue reportada por Alder y Wainwright en 1957 [5]. El método consistía en resolver en forma exacta las ecuaciones para el movimiento clásico simultáneo de muchas esferas duras y también de partículas que son modeladas por potenciales de atracción con forma de *pozo cuadrado* (ver figura 1.2. Los autores obtuvieron propiedades de equilibrio de estos sistemas. En particular se comparo la ecuación de estado del sistema de esferas duras —que relaciona presión con temperatura y densidad— con resultados numéricos de la misma época obtenidos mediante el *Método Montecarlo* por Metrópolis [3]. En un trabajo posterior los autores publicaron una descripción de su método basado en una **estrategia de simulación conducida por eventos** (estrategia conocida también como “event driven”) [8]. En esta estrategia la simulación “salta” de la colisión de un par de partículas a la siguiente colisión de otro par de partículas. Los eventos son las colisiones propiamente tales.

Sin embargo las esferas y el potencial de pozo cuadrado son sólo un primera aproximación a la interacción entre moléculas y los investigadores se vieron pronto en la necesidad de simular sistemas con interacciones inter-molécula más realistas. La primera simulación de este tipo fue llevada a cabo por Rahman en 1964 [7]. El sistema estudiado por Rahman consistió de 864 partículas que interactúan vía

Universidad del Bío-Bío. Sistema de Bibliotecas - Chile

un potencial efectivo conocido como "*potencial de Lennard-Jones*" (ver nota[4]). El trabajo de Rahman consituyó un aporte muy importante para la época pue permitió convencer a los físicos experimentales del valor de las simulaciones computacionales y que éstas permitían una mejor comprensión de los experimentos, y en general de las propiedades microscópicas de los fluidos.

Las simulaciones de Rahman, a diferencia de las de Alder, se basan en una **estrategia conducida por tiempo**. Esto es la simulación, en vez de "saltar" de un choque al siguiente choque, evoluciona con un paso fijo de tiempo, suficientemente grande como para avanzar bastante en el tiempo pero a la vez suficientemente pequeño de manera que el el efecto en la dinámica del detalle de la dependencia con la distancia de las fuerzas resulta descrito con cierta fidelidad o precisión (en partícular no perder interacciones tipo choque).

Un paso importante lo dio en 1964 Verlet, quien propuso un esquema para la actualización de posiciones y velocidades en la estrategia conducida por tiempo, que tiene la ventaja de ser al mismo tiempo que económico —en cuanto al número de operaciones de punto flotante— preciso y lo más importante desde el punto de vista físico: "reversible" en el tiempo, que es una condición relacionada con la conservación de la energía. El algoritmo de Verlet, con variantes que lo perfeccionan, es utilizado hasta nuestros dias.

1.2.1.1. Simulaciones conducidas por tiempo

La dinámica molecular conducida por tiempo (DMCT) es una técnica aplicable a sistemas clásicos, no cuánticos, que se basa en resolver numéricamente las ecuaciones de Newton que describen la dinámica de un sistem de N cuerpos. Esto es resolver el sistema de ecuaciones:

$$m_i \frac{d^2\vec{r}_i}{dt_i^2} = \vec{F}_i$$

con $i = 1, \ldots, N$ y en que $\vec{F}_i$ es la fuerza neta que actúa sobre la partícula i-esima, dada la condición inicial de posición $\vec{r}_i(0)$ y de velocidad $\vec{v}_i(0)$ de cada una de las N partículas.

Una forma de hacer esto es usando diferencias finitas de posición. El algoritmo de Verlet [9] se apoya en escribir el desarrollo en serie de las posiciones $\vec{r} = \{x(t), y(t), z(t)\}$ como dos series de Taylor en torno al instante de tiempo t para estimar la posición en un instánte posterior $t + dt$ y previo $t - dt$ de cada partícula:

$$\begin{aligned} x(t+dt) &= x(t) + v(t)dt + (1/2)a(t)dt^2 + (1/6)b(t)dt^3 + O(dt^4) \\ x(t-dt) &= x(t) - v(t)dt + (1/2)a(t)dt^2 - (1/6)b(t)dt^3 + O(dt^4) \end{aligned}$$

Sumando ambas se tiene la relación:

$$x(t+dt) = 2x(t) - x(t-dt) + a(t)dt^2 + O(dt^4)$$

que en términos de la ecuación de movimiento corresponde a:

$$\vec{r}(t+dt) = 2\vec{r} - \vec{r}(t-dt) + \frac{\vec{F}_i(t)}{m} dt^2 + O(dt^4)$$

[4] potencial que había sido usado previamente por Wood y Parker en 1957 vía simulaciones MonteCarlo para modelar un gas denso de Argón con bastante éxito, dada su sencillez y bajo costo computacional, de manera que dicho potencial fue adoptado por la comunidad científica de la época como el modelo preferido para describir los gases nobles.

Universidad del Bío-Bío. Sistema de Bibliotecas - Chile

y que muestra que si dt es suficientemente pequeño se puede obtener —con muy buena precisión— la posición avanzada $\vec{r}(t+dt)$ de cada partícula a partir del conocimiento de la posición actual $\vec{r}(t)$, la posición previa $\vec{r}(t-dt)$, y la fuerza $\vec{F}_i(t)$ que en el instante actual actúa sobre la partícula.

El algoritmo necesita inicialmente 2 posiciones por cada partícula, la actual $\vec{r}(0)$, y la previa $\vec{r}(-dt)$ (obtenida con la información de velocidad inicial de cada partícula), y predice la siguiente posición $\vec{r}(dt)$ en el instante posterior dt. Para obtener la posicion en los instantes $2dt, 3dt, 4dt, \ldots$ se repite el procedimiento previa actualización de las posiciones que se almacenan en un arreglo.

Un aspecto interesante de este algoritmo —desde el punto de vista físico— es que es explícitamente reversible en el tiempo **tal cual lo son** las ecuaciones de Newton, lo que tiene implicancias en la esta'distica que resulta de los datos de la simulación. Desde el punto de vista de costo algorítmico, las fuerzas —que es lo más caro de evaluar— son evaluádas una sola vez por cada paso de iteración.

Existen muchas otras variantes de este tipo de algoritmos, basadas en predicciones más precisas, o con métodos predictores y correctores, así como adaptaciones del método a situaciones en que las partícuas tienen ciertas restricciones geométricas y/o termodinámicas [10].

El aspecto más complejo de este tipo de simulaciones es la evaluación de las Fuerzas sobre cada partícula. Puesto que —por ejemplo en el caso gravitacional— la interacción distante de cada una de las otras restantes $N-1$ partículas tiene un efecto sobre la parícula i-esima, el costo de evaluar las fuerzas por particula es $O(N)$, lo que conduce a un resultado, que por paso de iteración temporal es orden $O(N^2)$ (también hay que tener en cuenta que se debe actualizar las posiciones de todas las N partículas). Sin embargo existe notables algoritmos que con una estrategia de "divide y reinaras" reducen este costo a uno de orden $N\log(N)$ [11] y recientemente, con el uso de las capacidades de cálculo paralelo en las tarjetas gráficas NVidia con soporte para aceleración, a un costo mucho menor orden $\log(N)$ para el caso gravitacional y de orden constante para fuerzas de muy corto alcance [12]. Lo que conduce a tiempos CPU de pocos segundos por paso temporal de interación, en sistemas de algunos millones de partículas, en que la dinámica molecular se calcula en un ambiente de procesamiento parallelo con por ejemplo paso de mensajes entre varios procesadores con tarjetas graficas de este tipo.

Universidad del Bío-Bío. Sistema de Bibliotecas - Chile

1.2.1.2. Simulaciones conducidas por eventos

La dinámica molecular conducida por eventos (DMCE) es una técnica de simulación que a diferencia de la DMCT — en que la simulación avanza a intervalos fijos de tiempo— la simulación avanza de acuerdo a una secuencia de colisiones (*o eventos de colisión*), que no ocurren a intervalos fijos de tiempo, sino que siguen alguna distribución azaroza de tiempos.

Esta dinámica se apoya en la idea de un modelo idealizado de las partículas en que éstas son objetos duros. No hay potenciales que varíen continuamente, y en consecuencia todas las colisiones son binarias (idealmente si la precisión del computador fuera infinita), es decir no es posible tener colisiones simultáneas de 3 o más partículas.

El modelo más básico simulable con esta dinámica es el de esferas duras, y que puede considerarse como una aproximación muy gruesa a la interacción molecular descrita por el potencial de Lennar-Jonnes para el caso de una dinámica molecular conducida por tiempo:

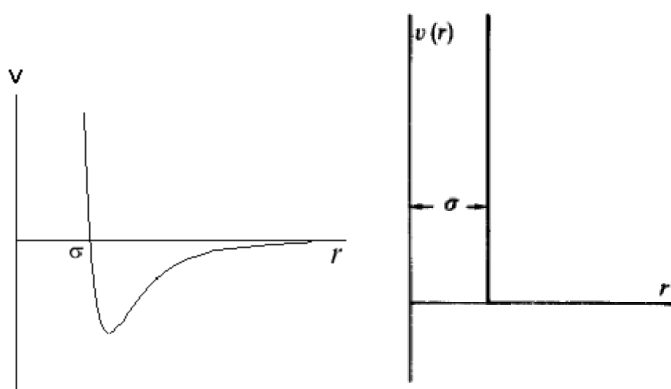


Figura 1.2: **Potenciales de interacción** (izquierda) potencial de Lennard-Jones; (derecha) potencial de esferas duras, el choque es impulsivo sólo para el momento del choque que ocurre cuando los centros de las esferas están a distancia σ entre sí.

en que sólo se considera el intenso caracter repulsivo del potencial y se reemplaza la empinada zona repulsiva por una de potencial infinito (ver figura 1.2). La idea es que las partículas sólo sienten fuerzas en el instante del choque, y que en dicho caso la interacción tiene caracter de golpe impulsivo. Entre otras propiedades físicas esto significa que la interacción conserva el momentum total y la energía mecánica total, lo que es una característica básica de las fuerzas a nivel molecular que todo modelo debe respetar.

1.2.1.2.1. El Algoritmo de Alder y Wainwright.

El algoritmo de Alder y Wainwright se basa en tener en cuenta que, para el modelo de esferas duras, al ser las colisiones binarias, si se conoce la evolución libre de las partículas, es posible determinar para cada par de ellas, si existe una posible colisión, y en que instante ocurriría dicha colisión. Esto genera para cada par de particulas (i, j) un tiempo de posible colisión que admite un valor finito $t_{i,j}$ (o un valor infinito en el caso de no colisión).

Universidad del Bío-Bío. Sistema de Bibliotecas - Chile

Dado las posiciones $\vec{r}_i$ y velocidades $\vec{v}_i$ para el conjunto de todas las partículas ($i = 1,\ldots,N$), en el instante inicial $t_{\text{last}} = 0$, el algoritmo realiza un ciclo típico de simulación a través de los siguientes pasos:

Paso 1 Para cada partícula i se obtiene el menor de lo tiempos de posible colisión con las restantes $k = 1,\ldots,N$ part'culas mediante un algoritmo de detección de la colisión (explicado más abajo). Evidentemente hay redundancia pues el tiempo de colisión de la partícula i con la j es el mismo que el de la partícula j con la i de modo que no es necesario obtener todos los tiempos $t_{i.j}$ sino que basta con para cada i determinar tiempos con un pareja $j > i$. En este proceso se obtiene el instante t^* del siguiente evento a ocurrir (que es el mínimo de todos los valores t_{ij}) y la pareja i, j de partículas que intervienen en la colisión.

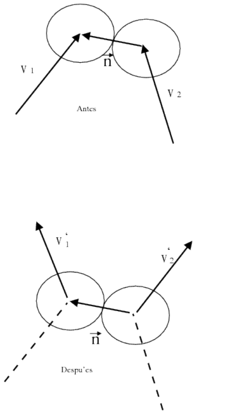
Paso 2 A continuación se actualiza las posiciones de todas las partículas usando las reglas de evolución libre para ellas:

$$\vec{r}_i \leftarrow r_i + \vec{v}_i \Delta t$$

en que Δt se refiere a la diferencia de tiempo entre t^* y el instante t_{last}.

Paso 3 Esta actualización de posiciones deja al par (i, j) en una configuración de contacto en que las dos particulas que colisionan están incidiendo con velocidades $\vec{v}_i$ y $\vec{v}_j$. La colisión misma es llevada a cabo obteniendo nuevas velocidades $\vec{v}'_i$ y $\vec{v}'_j$ para las partículas que colisionan de manera tal que ciertas reglas físicas propias de la colisión se cumplan. Estas reglas involucran típicamente exigir conservación del momentum, conservación de la energía y que la interacción entre partículas sea sin torque (lo que corresponde a una fuerza de interacción central) en cuyo caso los nuevos valores de velocidad para partículas de la misma masa que colisionan se obtienen mediante:

$$\begin{aligned}
\hat{n} &= (\vec{r}_i - \vec{r}_j)/\|\vec{r}_i - \vec{r}\| \\
\vec{\Delta} &= -(v_i - v_j)\cdot\hat{n} \\
\vec{v}'_i &= \vec{v}_i + \vec{\Delta} \\
\vec{v}'_j &= \vec{v}_j - \vec{\Delta} \\
\vec{v}_i &\leftarrow \vec{v}'_i \\
\vec{v}_j &\leftarrow \vec{v}'_j \\
t_{\text{last}} &\leftarrow t^*
\end{aligned}$$



Paso 4 Con el paso anterior las dos partículas que estaban incidiendo en la colisión ahora se alejan y es posible volver al **Paso 1** para repetir todo el procedimiento. La simulación se detiene cuando se avanza una cierta cantidad total de tiempo o cuando ha ocurrido un cierto número de colisiones.

La detección de las posibles colisiones se realiza de la siguiente manera:

A partir de las posiciones y velocidades de las partículas i y j se determina si ellas se están acercando (posible colisión) o alejando (en cuyo caso no hay colisión). Para esto revisa el signo del producto escalar $\vec{r}_{ij} \cdot \vec{v}_{ij}$ en que $\vec{r}_{ij}$ es la difererencia de posición y $\vec{v}_{ij}$ la diferencia de velocidad entre la partícula i y la j. Si el producto es negativo entonces se resuelve una ecuación de 2do grado en el tiempo, que

Universidad del Bío-Bío. Sistema de Bibliotecas - Chile

determina el instante de colisión. Si el discriminante de dicha ecuación es positivo, existen 2 soluciones reales que corresponden a la intersección de las partículas, en caso contrario no hay colisión y las partículas se cruzan pero no se tocan. La condición de positividad del determinante equivale a la comparación $(\vec{r}_{ij} \cdot \vec{v}_{ij})^2 > v_{ij}^2(r_{ij}^2 - \sigma^2)$ en que σ es el diámetro de las partículas. Si hay intersección el tiempo de choque queda dado por la expresión de menor tiempo de las dos soluciones:

$$t^* = \frac{-(\vec{r}_{ij} \cdot \vec{v}_{ij}) - \sqrt{(\vec{r}_{ij} \cdot \vec{v}_{ij})^2 - v_{ij}^2(r_{ij}^2 - \sigma^2)}}{v_{ij}^2}$$

1.2.1.2.2. Simulación eficiente: Los Algoritmos de Rapaport y Lubachebsky

El orden del costo del algoritmo de Alder y Wainwright es $O(N^2)$ por paso de colisión, lo que está relacionado con la parte más costosa de los pasos anteriores: la obtención de los tiempos t_{ij} de posibles colisiones entre partículas.

No fue sino hasta 1980, con el trabajo de D. Rapaport [13] en que se obtuvo una mejora sustancial sobre el algoritmo de Alder. La estrategia de Rapaport se basa en dos optimizaciones fundamentales, por un lado se evita recalcular tiempos de colision, guardando todos los valores t_{ij} calculados para posibles colisiones, pero descartando de esta gran lista los tiempos t_{ij} asociados a las dos partículas que acaban de colisionar, por otro optimizar la busqueda de tiempos almacenando los eventos predichos para cada par (i, j) (y sus tiempos) en un árbol de busqueda binario, lo que se puede realizar con un costo del orden $\log N$. El proceso de descarte de tiempos inválidos se realiza manteniendo para cada partícula i (el dueño del evento) listas doblemente enlazadas asociadas a las partículas j (pareja en el evento). Para minimizar la cantidad de parejas j en la lista de eventos asociadas a una partícula i se introdujo una división regular en celdas del dominio en que se mueven las partículas.

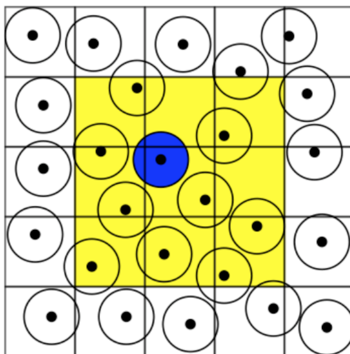


Figura 1.3: División en celdas para reducir el costo del cálculo de los eventos de colisión. La partícula en gris oscuro solo estudi posible colisiones con las partículas en las celdas de su vecindario. La reducción de complejidad del algoritmo introducel el costo adicional $O(1)$ de determinar cuando cada partícula cambia de celda. La pertenencia a un celda está determinada por la ubicación del centro de cada objeto. Las celdas deben ser de tamaño mayor o igual al diámetro de las partículas

Universidad del Bío-Bío. Sistema de Bibliotecas - Chile

La consistencia de la simulación requiere que debido a esto se introduzca un nuevo tipo de evento: la entrada o salida a una celda de cada partícula, y que se introduzca estructuras de datos para mantener, por cada celda, qué partículas se encuentran dentro de ella, asi como operaciones para mantener esta estructura actualizada.

Posteriormente Lubachebsky, en 1991 [14], propuso una estrategia simplificada que resulta bastante eficiente para densidades altas. En esta estrategia el estado de las partículas se define no sólo con su posición y velocidad, sino que también con el instante en que cada partícula colisionó por última vez. En vez de actualizar las posiciones de todas las partículas se actualiza sólo las posiciones y el tiempo de la última colisión para el par de partículas que acaban de chocar. Si se quiere obtener la posición actual siempre es posible a partir de la ultima posición, y velocidad almacenada y la diferencia de tiempo entre el instante actual y el instante de la última colisión. Para simplificar la complejidad del algoritmo Lubachebsky no guarda los eventos, sino que recalcula cada vez, pero para minimizar el costo que involucraría esto, igual que Rapaport, sólo tiene en cuenta las partículas que están en el vecindario de la celda a la que pertenece la partícula. Para optimizar la busqueda del tiempo mínimo se usa una estructura tipo 'heap' [15].

1.2.1.2.3. El Algoritmo de Marín-Risso-Cordero

En 1993 Marín, Risso y Cordero propusieron una estrategia que —a diferencia del Algoritmo de Rapaport que es eficiente a densidades bajas, y el algoritmo de Lubachebsky que es eficiente a densidades altas— resulta eficiente en todo el rango de densidades [1]. En el algoritmo de estos autores, al igual que en el de Lubachebsky, se usa un estado retardado para cada partícula, es decir se guardan las posiciones y velocidades post-colisionales y también el tiempo del último evento de colisión de la partícula. Como en el caso de Lubachebsky se divide el dominio en que evolucionan las partículas en celdas y se usa una estructura de datos para especificar que partículas hay en cada celda. Pero a diferencia de Lubachebsky todos los eventos son retenidos y almacenados, y por lo tanto no se recalcula eventos. El algoritmo de estos autores se apoya en la idea de tiempo mínimo local. De acuerdo al estudio de estos autores de todos los eventos predichos para una partícula dada ("owner" o "dueño" del evento) el de tiempo mínimo es el más probable. Para encontrar el siguiente evento se usa un arbol binario completo CBT, que es balanceado por definición. en la base del CBT se almacena el tiempo mínimo local asociado a cada "dueño". El mínimo global se obtiene a partir del CBT mediante operaciones de comparación que tienen un costo $\log N$. La lista de eventos asociada a cada partícula es salvada, y cada vez que una partícula cruza de una celda a otra, sólo se agrega a dicha lista los nuevos eventos que se generan debido a este proceso. El evento ganador en la comparación con los valores de esta lista es lo que se ingresa como evento al CBT. Cuando dos particulas colisionas las listas de eventos asociadas a estas partículas se descartan, luego de atender la colisión se obtienen nuevas listas de eventos para ellas y se vuelve a actualizar el CBT.

Sin embargo esta estrategia tiene un punto en contra, y es que en el CBT quedan almacenados eventos que debido a las colisiones que ocurren son inválidos. Sin embargo los autores plantean una estrategia muy simple que permite determinar, con una simple comparación entera al momento de atender el evento, si el tiempo minimo global es un evento válido o no. En el caso de no serlo simplemente se descarta el evento y se extrae el siguiente evento. La cantidad de eventos inválidos que llegan a la cima del CBT es baja, no excede el 15 % en el peor caso. El tamaño de las listas de eventos asociadas a cada partícula tiene una longitud con un promedio de 2.5 eventos, de modo que el costo de este algoritmo es $(1{,}15\log N)+2{,}5$ que se compara muy bien con el costo $2{,}0\log N$ del algoritmo de Rapaport, y con

Universidad del Bío-Bío. Sistema de Bibliotecas - Chile

el costo del algoritmo de Lubachebsky, que es orden $1{,}15 m \log N$ donde m es el número de particulas vecinas a un partícula dada (9 en dimension 2, y 27 en dimension 3) y que tiene que ver con el costo de recalcular eventos y no almacenarlos.

En su estudio los autores compararon estrategias, y demostraron que —en todo el rango de densidad— su estrategia es más eficiente que la que resulta de usar el algoritmo de Lubachebsky y la que resulta de usar el algoritmo de Rapaport [1].

1.2.1.3. Comparación entre las simulaciones conducidas por eventos y las conducidas por tiempo

Una pregunta que resulta pertinente aquí es ¿Por qué realizar simulaciones conducidas por eventos cuando se puede realizar simulaciones conducidas por tiempo con potenciales muy realistas?. ¿Qué se gana o se pierde al escoger una o la otra?.

Una primera respuesta tiene que ver con el modelamiento. Es mucho más simple construir teorías para sistemas de partículas duras que sistemas con potenciales contínuos, pero por supuesto todo depende del tipo de interacción que mejor se acerca al problema en estudio. Hasta el dia de hoy si se trata de simular sistemas de partículas duras, la estrategia conducida por eventos sigue siendo la mejor, particularmente porque no se gasta tiempo en simular el desplazamiento de las partículas entre colisión y colisión. Por otro lado en el caso de las estrategia conducida por tiempo siempre existe la posibilidad de que para dos partículas que se aceran a alta velocidad ocurra una colisión a una escala de tiempo mucho más pequeña que el paso temporal de la simulación y que por lo tanto esta colisión no se detecte usando esta estrategia lo que en en el fondo significa que en una simulación conducida por tiempo estas partículas se atraviesen sin verse. Esta es una fuente de error que no existe en el caso de la simulación conducida por eventos.

Sin embargo la estrategia de simulación conducida por tiempo es fácilmente paralelizable, y aunque se han generado algoritmos para paralelizar el problema de la estrategia conducida por evento estos son complejos y dificiles de mantener cuando se agrega nuevos objetos tales como paredes moviles a la simulación.

En la práctica desde el punto de vista de el tiempo CPU gastado en una simulación conducida por eventos versus la conducida por tiempo, para sistemas pequeños da hasta el orden de algunos cientos de miles de particulas duras la estrategia conducida por eventos domina sobre la estrategia conducida por tiempo, pero para sistemas más grandes las técnicas de paralelización hacen de la estrategia conducida por tiempo la estrategia favorita.

Universidad del Bío-Bío. Sistema de Bibliotecas - Chile

1.3. Materiales Granulares

Existe una gran cantidad de materiales que tiene características de materia granular y que son de uso común en nuestras vidas diarias. Estos medios son colecciones de muchas partículas macroscópicas con un tamaño típico suficientemente grande para que las fluctuaciones térmicas (que ocurren a nivel microscópico y que son propias de la naturaliza corpuscular atómica) no sean importantes. Estos incluye materiales de construcción, alimentos, materias primas y bienes de producción, entre otros. Otros manifestaciones de estos materiales y de su peculiar comportamiento ocurren en: la formación de dunas, en el potencialmetne catastrópico fenómeno las avalanchas de nieve, y hasta en la estructura que adoptan el conjunto de millones rocas sometidas a esfuerzos gravitacionales que constituyen los llamados anillos planetarios tales como los anillos de Saturno.

Figura 1.4: Material granulado en la naturaleza.

1.3.1. Aplicaciones en la Industria

Debido a su tremenda importancia práctica y tecnólogica[5], estos materiales han sido objetos de estudios por centurias en el campo de la ingeniería, y sólo muy recientemente, desde un par de décadas acá, han atraído profundamente la atención de los físicos por sus implicancias en el estudio de las propiedades de materia fuera del equilibrio.

Entre los problema de ingeniería están el transporte y almacenamiento del material granulado, así como el problema de segregación (o separación por especies) o el problema inverso de mezclado homogéneo (estos dos problemas son particularmente relevantes en la industria farmaceutica). La industria farmacéutica, por ejemplo, hace uso de cintas transportadoras para llevar las píldoras o pastillas de una parte de la línea de producción a otra. En ocasiones, estas pastillas pasan por tolvas en los que podrían quedarse atrapadas por la formación de "arcos" (Figura 1.5), que pueden retrasar el proceso o detenerlo por completo.

[5]El procesamiento de materiales granulares y agregados consume aproximadamente el 10 % de toda la energía producida en el planeta y éste tipo de materiales es el segundo tipo de materiales despues del agua en la escala de prioridades de las actividades humanas [16].

Universidad del Bío-Bío. Sistema de Bibliotecas - Chile

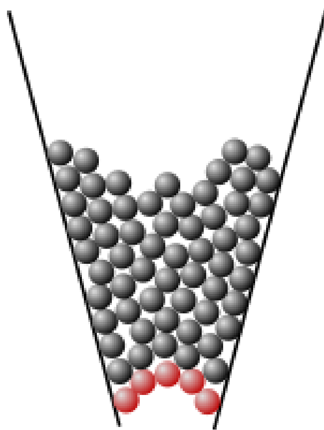


Figura 1.5: Un material granular se atasca cuando trata de pasar a través de una tolva debido a la formación de arcos (esferas rojas). Fuente: Wikipedia.

Para poner en evidencia lo complejo de la física subyacente en estos temas basta decir que aún hoy, a pesar de nuestror inmensos avances en ingeniería no todo está entendido en medios granulados, y es muy frecuente —por ejemplo— el colapso de silos de almacenamiento.

Figura 1.6: Colapso de silos de almacenamiento para material granulado.

Muchos de los procesos de material granulado usados en la industria involucran el estado fluidizado de éste. Dado que el material granulado —a diferencia del comportamiento corpuscular a nivel atómico— involucra procesos disipativos, es necesario mantener el estado fluidizado mediante una injección permamente de energía. Esto se logra por mecanismos muy diversos que van desde someterlo a vibración, fluidizarlo con injecciones de líquido o aire, e incluso aplicar fuerzas electromagnéticas.

Universidad del Bío-Bío. Sistema de Bibliotecas - Chile

En la jerga de ingeniería esta se conoce como "flujos granulares rápidos" y ha sido una de las áreas mas estudiadas [16].

Un ejemplo, de particular interés para nuestra nación, es en la industria del cobre, en la que es necesario, para sus proceso de molienda, introducir el material obtenido de la mina, en conjunto con muchos bolones de acero, en tambores de acero de caracter industrial, de algunos metros de diámetro que se hacen girar para producir la molienda[6]. El detalle de como ocurre la molienda, no se puede observar en forma directa, pues no es posible poner una ventana de observación debido a que la alta velocidad y masa de los materiales al interior destruiría dicha ventana (y las cámaras de video para observación)..

Figura 1.7: Tambores de molienda para el cobre. En el interior se mezcla material de cobre —tal como viene extraído de la mina— con bolas de acero y se somete éste a movimientos de rotación del tambor, como resultado del proceso se produce una granulado más fino del material que contiene el cobre y que es sometido posteriormente a otros procesos.

Este proceso es muy caro, pues involucra un alto costo energético: toda la energía cinética imprimida a las bolas de acero se gasta en romper el material. En otras palabras toda la energía invertida es disipada, y no es poca.

Preguntas que están abiertas en este problema son: ¿Cuál es el diámetro apropiado de estos tambores?. ¿Con qué velocidad de rotación es más eficiente el proceso?. ¿Cómo ocurre o se evita la segregación de los minerales al interior del tambor? (por ejemplo se sabe que en medios granulados existe el fenómeno de segración por tamaño conocido como "el problema de la nuez de Brasil" en que los objetos de mayor tamaño tienden a irse a la superficie (ver figura 1.8), además de un sinnúmero de otros fenómenos tales como la formación de bandas de segregación), ¿Cuál es la proporción y distribución de tamaño de las bolas más conveniente a usar?. Es claro que respuestas a este tipo de preguntas tienen una enorme incidencia económica en estos procesos.

[6]Las bolas de acero para la molienda del cobre las produce la empresa Molycop en la región del Bío-Bío

Universidad del Bío-Bío. Sistema de Bibliotecas - Chile

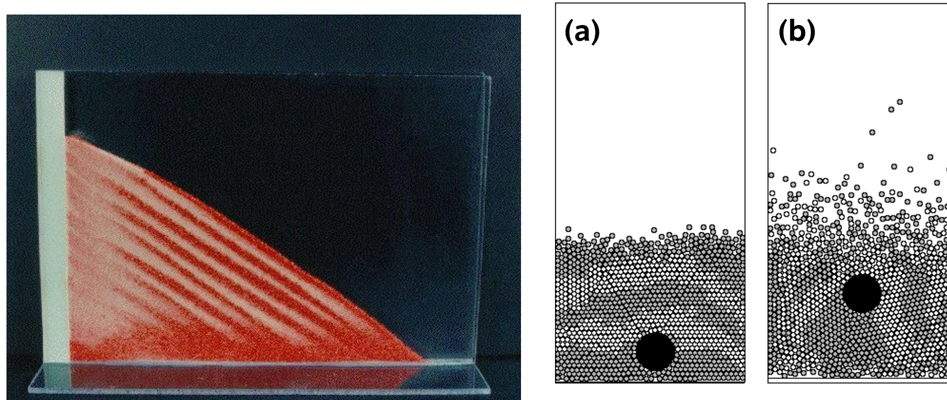


Figura 1.8: (izquierda) Formación de bandas segregación en avalanchas. (derecha (a) y (b)) El fenómeno de la nuez de Brasil.

Una dificultad del problema de la molienda es que no es es cosa de construir tambores de distinto tamaño para probar y determinar empíricamente las mejores condiciones de llenado, pues porque por un lado está la multiplicidad de variables que entran en el estudio, por otro que las dimensiones del sistema hacen prohibitiva esta alternativa. En relación a este último punto es interesante nombrar que el estudio de modelos a escala también resulta complejo pues aspectos como la electrostática y la fracción de humedad de los materiales son más relevantes a escalas pequeñas que grandes y por lo tanto los resultados no son siempre directamente comparables.

1.3.1.1. Simulaciones de medios granulados

Es aquí que el uso de los poderosos métodos de dinamica molecular conducida por eventos (que se conocen también en la literatura como *simulaciones de elmentos discretos*) viene en ayuda y han revolucionado el estudio de los materiales granulares fluidizados. Por ejemplo en la figura 1.9 se muestra resultados para un estudio sistemático del problema de la nuez de Brasil. Se aprecia que con las simulaciones (región gris en la figura) se puede barrer un amplio rango de parámetros.

El problema de la molienda es también tratable simulacionalmente y en este sentido el GDMTC tiene algun trabajo avanzado: En la figura de a continuación se muestra como ejemplo un resultado preliminar para la simulación del problema del tambor rotante obtenidos por el GDMTC mediante simulaciones conducidas por eventos

Universidad del Bío-Bío. Sistema de Bibliotecas - Chile

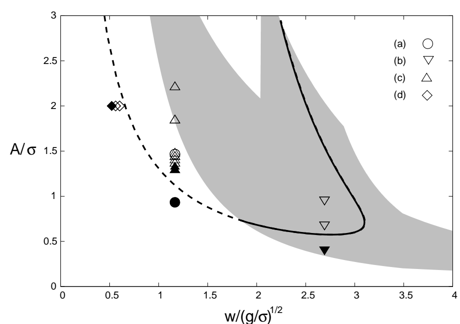


Figura 1.9: Regiones en que se observa y no se observa el fenómeno de la nuez de Brazil de acuerdo a resultados experimentales y de resultados simulacionales obtenidos por el GDMTC para el problema de un "intruso" en una cama de granos pequeños y sometido a una vibración de la base caracterizada por una Amplitud A y una aceleración a. En el eje vertical la amplitud de oscilación A/d en que d es el diámetro de los granos pequeños, y en el eje horizontal la aceleración de la base, comparada con la aceleración de gravedad g. Los puntos experimentales en negro corresponde a situaciones donde no sube el "intruso" y los blancos a situaciones en que sube. La región oscura muestra el rango de parámetros estudiado en las simulaciones.

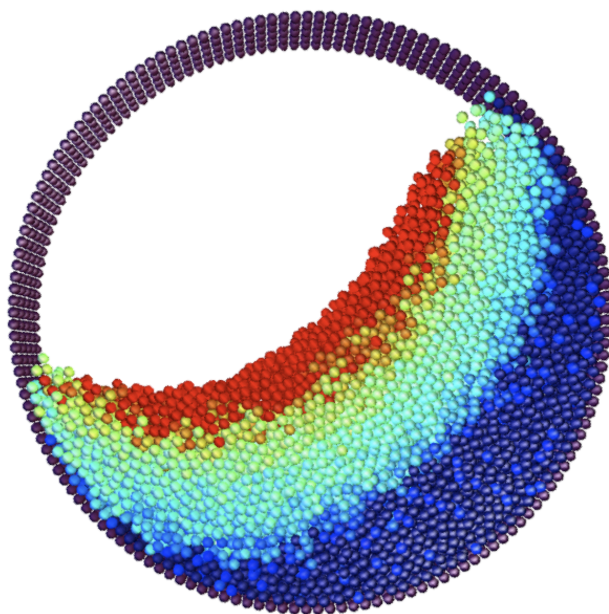


Figura 1.10: Fluidización en el problema del tambor rotante. La figura ilustra con un mapa de colores los grados de fluidización obtenidos mediante simulaciones de dinámica molecular.

Universidad del Bío-Bío. Sistema de Bibliotecas - Chile

En el caso de los resultados de la figura anterior se ha hecho uso de una técnica mejorada de simulación *"extended event driven molecular dymamics"* en que sólo las partículas fluidizadas se llevan el costo de la simulacón, mientras que las partículas adheridas a las paredes o en movimiento nulo relativo a ellas no inciden sobre la complejidad de la simulación [2].

Desde el punto de vista académico es conveniente decir finalmente que el estudio de los medios granulados significa una oportunidad única a la comunidad de físicos de poder estudiar el problema de la física que ocurre en sistemas alejados del equilibrio, pues las leyes básicas que aquí se de deducen son generalizables a muchos otros sistemas. Por ejemplo en medios granulados pueden existir ondas de choque y ser observadas simulacional y experimentalmente teniendo acceso (por ser fenómenos de caracter macroscópico) al detalle de lo que ocurre en la región de la onda de choque, mientras que no se tiene el mismo nivel de acceso a la información de lo que ocurre en ondas de choque a nivel microscópico en fluidos.

Un interesante ejemplo de interacción entre físicos experimentales, teóricos y simulacionales es el articulo [17], en que están involucrados los miembros del GDMTC, y que trata sobre la separación de fáses en sistemas soméros sometidos a vibración vertical.

Universidad del Bío-Bío. Sistema de Bibliotecas - Chile

Capítulo 2

El problema a resolver

2.1. Historia del proceso

El *simulador* fue diseñado inicialmente como parte del trabajo de Doctorado del Prof. Risso y como tema principal de la tesis de Magister del Prof. Mauricio Marín (Univ. Magallanes). Actualmente el simulador se mantiene como un código privativo del GDMTC y las sucesivas versiones de este código se almacenan bajo el sistema de control de versiones "subversion" que permite que varias personas puedan modificar y administrar el mismo conjunto de datos accediendo remotamente desde sus respectivas ubicaciones. El simulador está escrito en lenguaje C y la estrategia de modularidad en la codificación del simulador —detallada en los manuales del simulador— permite que las intervenciones de los diferentes desarrolladores se afecten en un mínimo entre sí. Hay que tener en cuenta que los problemas atacados por los distintos investigadores que hacen uso de este software son diferentes de modo que no existe un único binario ejecutable del simulador, sino que cada investigador compila el simulador directamente del código fuente activando (configurando) los aspectos que le interesen en la etapa de compilación.

Acerca de la modalidad de trabajo y monitoreo en tiempo de ejecución de las simulaciones

Una actividad muy importante de Los investigadores del GDMTC es la realización de simulaciones numéricas mediante la técnica de Dinámica Molecular. Las simulaciones son principalmente del tipo conducidas por eventos. La simulación computacional son compleja de programar porque se deben respetar leyes físicas lo que se traduce en tratamiento numérico especializado. Sin embargo, simular tiene la siguientes grandes ventajas entre las que se destacan:

- Poder realizarse en cualquier momento.
- Extenderse en el tiempo tanto como lo permita una escala de tiempo humano y la disponibilidad de tiempo de computo.
- Procesarse en lapsos prolongados sin la necesidad de supervisión.

Es común que algunas simulaciones que realiza el GDMTC tomen desde horas hasta semanas.

Universidad del Bío-Bío. Sistema de Bibliotecas - Chile

Los beneficiados de estos procesos simulacionales son principalmente los físicos del GDMTC en sus investigaciones, pero en un futuro lo pueden ser ingenieros que trabajen o esten interesados en el tipo de problemas que ataca el GDMTC, e incluso industrias de producción y extracción de materias primas.

Preparar un conjunto de simulaciones involucra típicamente las siguientes etapas:

(a) modificación del software de simulación para preparar nuevas mediciones de acuerdo al problema a estudiar. Esto signifca generar código nuevo que es incorporado al simulador y que puede ser incluido o no en una simulación, lo que se define en tiempo de compilación.

(b) preparación de una conjunto de simulaciones similares que usualmente difieren en el valor de un parámetro (a veces más de uno) y que son ejecutadas en un cluster de ordenadores (o en un servidor con muchos nucleos) en forma secuencial o simultánea dependiendo de los recursos de cómputo que el cluster ofrece. Los valores de los parámetros son proporcionadas al simulador a traves de un *archivo de entrada* "data" que especifica dichos parámetros. Ejemplos de valores de los parámetros son: la geometría del sistema a simular, el número de partículas, su tamaño, masa y el tipo de interacción. Si las partículas corresponden a medios granulados entonces se especifica también los coeficientes de restitución entre granos y de granos con las paredes del sistema. Otros parámetros que se especifican son la duración total de la simulación, cada cuanto tiempo se realizan determinados tipos de mediciones y con que frecuencia se guarda datos del estado de la simulación y de algunas de las mediciones. Como la estrategia de simulación es orientada a eventos lo que se define aquí es con que regularidad se activan los eventos asociados a estas actividades.

    Es común preparar un conjunto de 20 o más simulaciones y estar ejecutando sólo 10 de ellas mientras que las restantes 10 esperan en una cola de procesos. Las simulaciones se ejecutan en carpetas diferentes y los softwares de manejo de cola de procesos se hacen cargo de en cual ordenador del cluster corre determinado simulación y en que momento se activa. Un software de adminstración de cola de procesos típico es *Grid Engine* de la empresa SUN Microsystems.

(c) cada simulación tiene esencialmente dos etapas básicas: una fase de **termalización** y otra de **medición**. La fase de termalización tiene interés cuando el proceso físico a investigar es estacionario, en cuyo caso durante la fase de termalización ocurre el transiente desde la condición inicial de simulación hasta el estado estacionario. Durante estas fases el simulador realiza una serie de mediciones básicas que corresponde a las predefinidas en tiempo de compilación y activadas en el archivo "data" de entrada de datos . Sin embargo puesto que la estrategia de simulación es conducida por eventos, es posible activar nuevas mediciones (también definidas en tiempo de compilación, pero no activadas en la fase de termalización) lo que usualmente se realizan en la fase de medición. Si el usuario desea ver una animación de la evolución de posiciones de las partículas o granos simulados, debe especificar esto en el archivo "data". Una vez activada una animación (lo que puede ocurrir al inicio de la simulación, o despues de la fase de termalización la animación termina con la simulacíon misma, y el usuario no tiene control sobre ella una vez que parte, excepto interrumpir el proceso. Las animaciones que actualmente se dispone —son básicas— y se realizan con la libreria de software GNU *libplot* y optativamente con la interfaz GRX (librería que está descontinuada, pero que aún apera en ambientes Linux e

Universidad del Bío-Bío. Sistema de Bibliotecas - Chile

incluso Windows). Un aspecto no deseable de esto es que una animación consume recursos de cómputo de la propia CPU donde está corriendo la simulación, pues los eventos de animación son parte del simulador, y por ende la simulación misma se ralentiza. Por otro lado no siempre se desea mantener una visualización constante sobre una simulación, y menos cuando se tiene 20 a 40 de ellas corriendo en un cluster de computadores, y otras 10 a 15 corriendo en otro cluster.

Sin embargo hay que destacar que el recurso de animación es 'solo una herramienta más en los procesos de simulación. Normalmente el simulador procesa sus propios datos generando resumenes y guardando en archivos el resultado de sus mediciones con el objeto de un proceso posterior.

2.2. El Problema a Resolver

Si bien el simulador posee el que probablemente sea el algoritmo más eficiente (tanto a bajas, como a altas densidades) que existe actualmente, existen ciertos problemas a la hora de realizar las simulaciones.

Este proyecto pretende ser una ayuda para los siguientes problemas:

- La persona que monitorea la evolución de la simulación pierde una gran cantidad de tiempo dado que debe ponerle atención mientras corre, lo que se traduce en un alto costo de tiempo invertido en observar.
- Durante la observación aparecen muchas veces aspectos que no le interesan.
- Hay una gran cantidad de tiempo perdido cada vez que se desee reconfigurar parámetros o mediciones, dado que para ello es necesario detener y relanzar la simulación.
- El proceso actual es engorroso y requiere operadores con un nivel de conocimientos de programación avanzado para poder realizar cambios o introducir mediciones a la simulación. Esto limita la cantidad de personas que pueden hacer uso del simulador y la motivación que puedan tener nuevos usuarios para incluir simulaciones a sus estudios.
- La capacidad de procesamiento se reduce debido a que parte de los recursos de compúto del cluster son utilizados simultáneamente por la simulación y los procesos gráficos de visualización, ambos altamente consumidores de CPU y Memoria principal.

Se desea tener una herramienta de monitoreo que permita visualizar el estado (y evolución) desde el momento en que se interviene una simulación y cambiar parárametros dinámicamente mientras la simulación corre. También esta herramienta debiera permitir activar mediciones —a voluntad del investigador— que están programadas pero que no han sido activadas ni durante la fase de termalización ni al fin de ella.

Universidad del Bío-Bío. Sistema de Bibliotecas - Chile

Capítulo 3

Diseño de la solución

3.1. Requerimientos

Para minizar los problemas antes mencionados, se requiere un sistema computacional que posea las siguientes características:

3.1.1. Requerimientos Funcionales

El diseño Lógico debe facilitar la interacción del usuario con el sistema mediante interfaces simples y comprensibles que permitan utilizar sin conflicto toda su funcionalidad.

- La aplicación debe poseer una interfaz gráfica que permita el monitoreo en cualquier momento, desde otros equipos (clientes).
- La aplicación debe permitir a la persona que monitorea la simulación el hacer en forma simple, cambios a los parámetros (ejemplo: gravedad, densidad, etc.), asi como la activación de alguna medición existente (fuerzas, temperatura, etc.).
- Debe también poseer un intérprete de comandos que permita al usuario especificar manualmente los nuevos valores que desea darle a ciertos parámetros.
- La aplicación de monitoreo debe ejecutarse por un cliente de manera independiente, permitiendo que el servidor se dedique exclusivamente al proceso de simulación.

3.1.2. Requerimientos No-Funcionales

Además de los requerimientos de los usuarios, también se deberán considerar los siguientes requerimientos para su implementación computaciónal:

- Para posteriores actualizaciones y mejoras, debe estar escrito en lenguaje C ya que éste es conocido por el GDMTC.

Universidad del Bío-Bío. Sistema de Bibliotecas - Chile

- Debe ser capaz de permitir una visualización de las simuaciones mediante la biblioteca gráfica OpenGL[1].
- La aplicación debe correr en ambientes Unix, más específicamente en GNU/Linux, ya que es el sistema operativo que usa el GDMTC.

3.2. Objetivos

3.2.1. Objetivo General

Apoyo al Grupo de Teoría Cinética en su trabajo de simulaciones, mediante una aplicación que provea una interfaz gráfica para la visualización y control de simulaciones de medios granulados y que permita su administración remota.

3.2.2. Objetivos Específicos

1. Apoyar a los procesos de visualización de las simulaciones mediante animaciones gráficas 3D con OpenGL.
2. Desarrollar una aplicación que permita modificar los parámetros de las simulaciones mientras corren.
3. Construir una aplicación simple y modular para soportar las permanentes modificaciones que se introducen al simulador, sin afectar el funcionamiento del "visualizador".
4. Desarrollar una aplicación que permita monitorear de forma remota simulaciones que corren en un cluster de procesadores.
5. Construir un servicio[2] que interactúe con el cliente y le diga qué simulaciones están corriendo actualmente en el clúster.

El grado de éxito de este proyecto estará dado por el cumplimiento o no de los objetivos recién mencionados.

[1] En OpenGL a diferencia de LibPlot la ubicación espacial y el clipping de objetos, así como muchas otros aspectos gráficos son parte de la libreria y no debe programarlos el usuario. Además en hardware con soporte OpenGL buena parte del costo de las operaciones gráficas recae en la tarjeta gráfica y no en el procesador principal del computador.

[2] Un servicio o demonio es un programa que corre en segundo plano y que realiza una acción de manera desatendida por el usuario. En este caso tendrá que revisar si existen simulaciones corriendo.

Universidad del Bío-Bío. Sistema de Bibliotecas - Chile

3.3. Estudio de Factibilidad

3.3.1. Factibilidad Técnica

Las simulaciones computacionales pueden llegar a ser bastante exigentes en cuanto al uso de procesador, es por esto que generalmente se ejecutan en clusters de máquinas de alto rendimiento. Esto quiere decir que existe un servidor principal que administra la "cola de procesos" que se están ejecutando (o esperando a ser ejecutados) y que existen otros servidores que procesan datos y devuelven sus resultados al servidor principal. La velocidad de procesamiento de cada CPU es bastante alta, al igual que sus conexiones internas para transmisión de datos, pero si la comunicación entre cada servidor no fuera los suficientemente rápida, si la entrega de dichos resultados demorara más de cierto tiempo, podría resultar discutible realizar procesos en equipos separados. Es por esto que también es importante que todos los equipos asociados al clúster posean tarjetas de red de alta velocidad (Categoría 6) y por supuesto cables con las mísmas características.
Por el lado del cliente los requerimientos de hardware serán mucho menores. No es necesario que el cliente tenga una gran capacidad de cálculo, aunque si se desea animar una simulación con OpenGL será recomendable (aunque no vital) tener una tarjeta gráfica con un aceleración 3D.
Dado, que los investigadores asociados al estudio de medios granulares tienen años de experiencia en el tema y se han adjudicado una extensa lista de proyectos y publicaciones, poseen equipos de alto rendimiento y sus clusters tienen la capacidad de cáuclo y de transmisión de datos requerida.

3.3.2. Factibilidad Operativa

Antes de comenzar a desarrollar una herramienta de este tipo, debemos tener en cuenta que el manejo de simulaciones, ya sea para su monitoreo, inicialización o análisis, debe ser llevado a cabo por personas con vastos conocimientos en el tema. En ningún caso pretendo que cualquier persona pueda hacer uso de un monitor de simulaciones, es por esto que la herramienta debe ser construida a medida, considerando que los usuarios saben, por ejemplo, hacer uso de una línea de comandos.
Además se debe considerar que el producto final a entregar no será un binario sino el código fuente ya que, como está establecido en los requerimientos, los mismos usuarios o futuros estudiantes deben ser capaces de hacer mejoras o incorporar nuevas funcionalidades. Es por esto que la documentación del código y su estrategia deben estar también plasmados en este documento.

3.3.3. Factibilidad Económica

La factibilidad económica persigue establecer si un proyecto es rentable, es decir, si genera beneficios (VAN=valor actualizado neto). Desde un punto de vista privado, este beneficio debe satisfacer al inversionista. Desde una perspectiva social, puede considerarse satisfactorio un VAN negativo, a condicion de que hayan personas beneficiadas por el proyecto. El ejercicio que se presenta considera las siguientes variables:

I Monto inicial que se invertirá en el desarrollo de esta apliación.

i Tasa de interés anual que se considera para actualizar los flujos de ingresos y costos anuales

Universidad del Bío-Bío. Sistema de Bibliotecas - Chile

P Cantidad de años considerados para la evaluación (ciclo de vida del proyecto)

FI Flujos de ingresos anuales

CA Flujos de costos anuales.

Los resultados son expresados en las siguientes variables:

VAN Valor actualizado Neto. que la suma neta de flujos traidos a valor presente a la tasa i

TIR Tasa interna de retorno. Tasa de interés que hace cero al VAN

Los investigadores del GDMTC invierten en promedio aproximadamente **2** horas diarias haciendo uso del simulador. Este tiempo se verá reducido a la mitad (**1** hora diaria) con el uso de una herramienta con estas características. Esto significa que en 1 año, el investigador gastará aproximadamente **10** días menos con el uso de SiMon. Si el día de un investigador vale aproximadamente $45.000, esto significaría un ahorro de **$450.000** por investigador, es decir :**$1.350.000** en total (3 investigadores). Los costos anuales asociados a mantención de SiMon son de aproximadamente **$1.000.000**, que es la cantidad que podría pagar el grupo con los fondos del proyecto que se destinan a personal técnico. Tomando en cuenta estos valores y considerando que la vida del proyecto será de aproximadamente 4 años y exigiéndole al proyecto un **5 %** de interés anual, podemos construir nuestro cálculo del **VAN**, de la siguiente forma:

$$VAN = -1{,}000{,}000 + \sum_{p=1}^{4} \frac{1{,}350{,}000 - 1{,}000{,}000}{(1+0{,}05)^{p}} = 241{,}083 > 0$$

Si bien el cálculo anterior indica que el proyecto es relativamente rentable, los objetivos de este trabajo no se centran en generar beneficios económicos para el investigador. Es por esto que los mayores beneficios se encuentran al analizar la disminución de tiempo invertido en realizar los trabajos de investigación.

Los siguientes son beneficios casi inmediatos que se obtienen por la existencia de un monitor de simulaciones:
Ahorro de tiempo de aprendizaje:

- Memoristas: Estudiantes que realicen estudios sobre medios granulares normalmente tardan entre 2 y 3 semanas en aprender a usar el simulador, introducir mediciones y configurar nuevas simulaciones. Con la ayuda de un monitor de simulaciones podrían reducir este tiempo de aprendizaje a 1 semana.

- Investigadores Colaboradores: Investigadores de otras partes del mundo, que hacen uso del simulador, normalmente tardan un poco más de 2 semanas en aprender a lanzar simulaciones, este tiempo podría reducirse a 1 ó 2 días, ya que un monitor de simulaciones, con una interfaz intuitiva, podría realizar gran parte del trabajo que ellos realizan manualmente.

- Investigadores Desarrolladores: Los investigadores que trabajan desarrollando el simulador, deben invertir aproximadamente 2 semanas en enseñarle a un Colaborador cómo preparar, lanzar

Universidad del Bío-Bío. Sistema de Bibliotecas - Chile

simulaciones y extraer la información generada. Con ayuda de un monitor de simulaciones el último aspecto de este trabajo se realizaría de una manera mucho más eficiente, ya que basta con explicar la interfaz del monitor y su estrategia para permitir al Colaborador o Memorista comenzar a monitorear las imulaciones.

Versiones Demo:

- Divulgación: Un monitor de simulaciones hará posible que docentes hagan uso de él para demostraciones de fenómenos físicos en la sala de clases, en congresos o en otro tipo de actividades de divulgación científica.
- Ampliar la posibilidad de trabajar con personas no expertas: Existen personas que colaboran con el estudio de medios granulares desde otras disciplinas, que no son expertos en simulaciones o que no programan. Este proyecto hará posible que esas personas puedan lanzar simulaciones y realizar estudios sobre ellas sin necesidad de tocar código.

Universidad del Bío-Bío. Sistema de Bibliotecas - Chile

3.4. Arquitectura del Sistema

Básicamente el sistema se compone de 2 entidades fundamentales (ver figura 3.1): El cliente como receptor de información, graficador y realizador de los cambios sobre la marcha y el cluster de procesadores, en que se ejecuta la simulación y se escriben las salidas solicitadas por el cliente. Más específicamente, el cluster de procesadores, se compone de un servidor principal que maneja la cola de procesos que deben ejecutarse (por lo general varias simulaciones). Si los procesadores del servidor principal están ocupados y es necesario correr un nuevo proceso, éste será derivado a uno de los servidores secundarios, éstos realizarán los cálculos pertinentes y devolverán los resultados al servidor principal.
Esto último es transparente para quien está monitoreando una simulación, ya que para ella, todos los procesos están corriendo en el servidor principal.

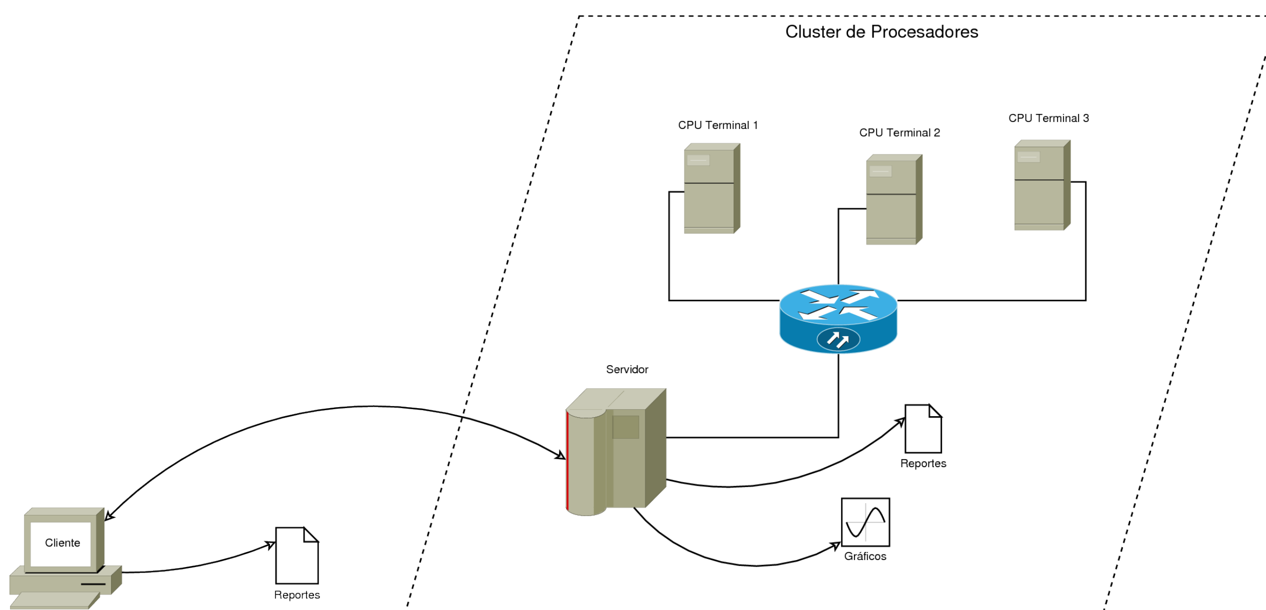


Figura 3.1: Arquitectura física del Sistema.

Universidad del Bío-Bío. Sistema de Bibliotecas - Chile

3.4.1. Diagramas de Flujo de Datos

Los siguientes diagramas muestran gráficamente cómo interaccionan los módulos del Monitor de Simulaciones con el servidor. El Sistema de Simulación comprende a SiMon como monitor de la simulación y al servidor como procesador y proveedor de datos.

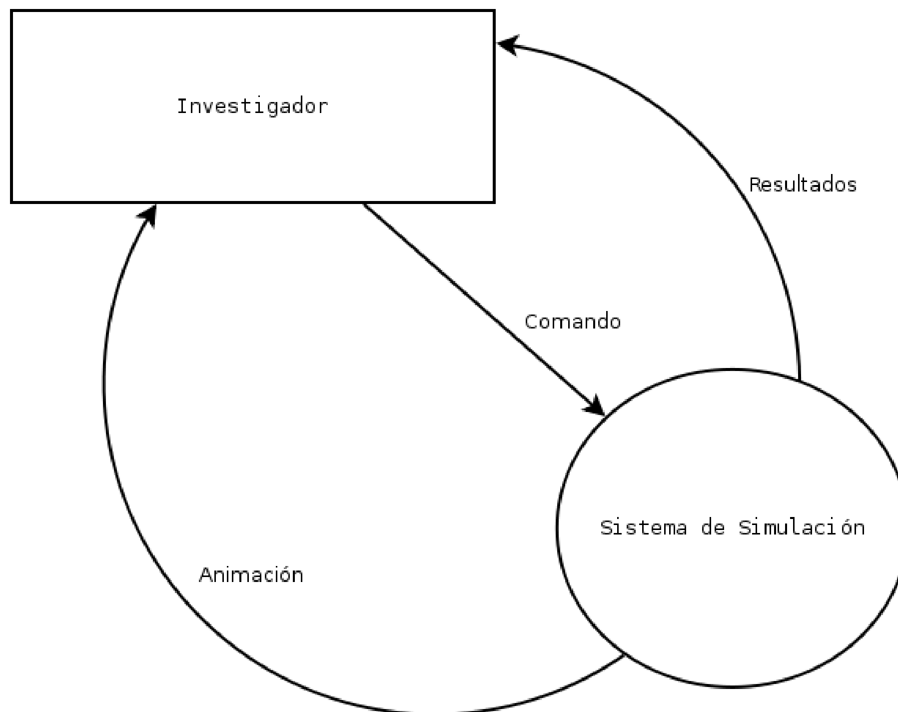


Figura 3.2: Diagrama de Flujo de Datos Nivel Contexto.

Universidad del Bío-Bío. Sistema de Bibliotecas - Chile

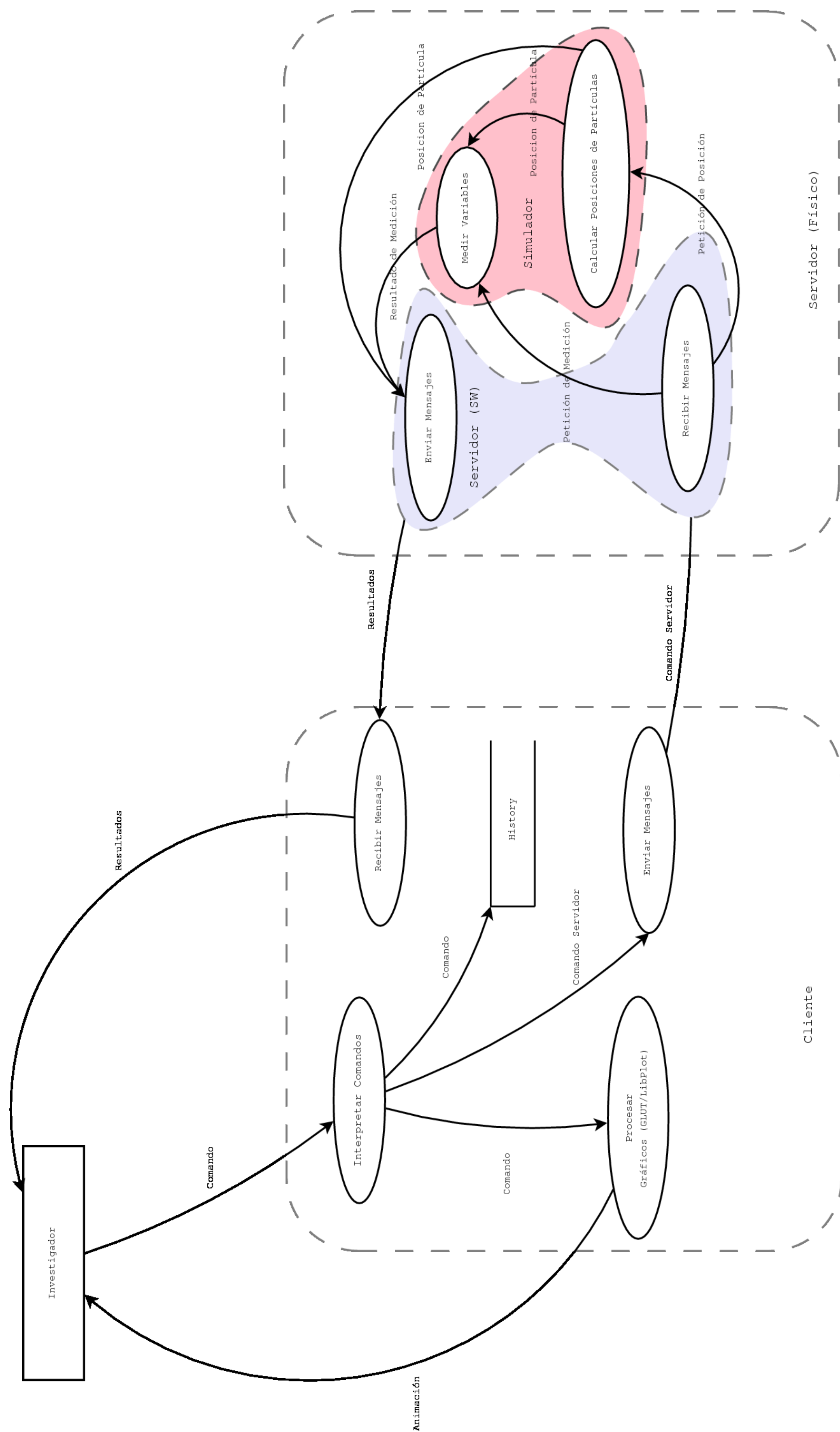


Figura 3.3: Diagrama de Flujo de Datos Nivel Superior.

Universidad del Bío-Bío. Sistema de Bibliotecas - Chile

3.4.2. Casos de Uso

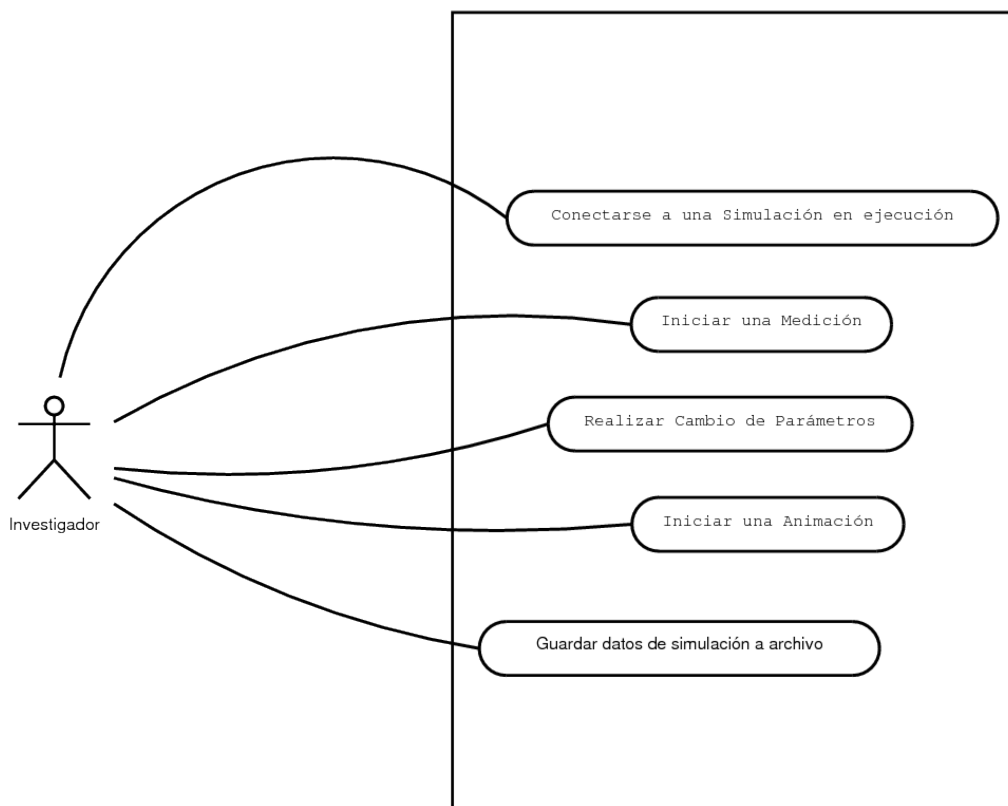


Figura 3.4: Diagrama de Casos de Uso.

3.4.2.1. Descripción de Casos de Uso

Nombre	Conectarse a una simulación.
Actor Principal	Investigador.
Precondiciones	SiMon ha sido iniciado con una IP válida.
Resultados	SiMon está ahora conectado a la simulación especificada por el usuario.
Descripción	El Investigador solicita a SiMon la conexión a una simulación específica.

Universidad del Bío-Bío. Sistema de Bibliotecas - Chile

Nombre	Iniciar una medición.
Actor Principal	Investigador.
Precondiciones	- SiMon ha sido iniciado con una IP válida. - Se ha seleccionado una simulación. - Se ha seleccionado una medición para iniciar.
Resultados	El servidor está ahora midiendo el parámetro seleccionado por el usuario y mostrando esta información por pantalla o guardandola a disco.
Descripción	El Investigador inicia una medición programada al inicio (al momento de compilación de la simulación).

Nombre	Realizar Cambio de Parámetros.
Actor Principal	Investigador.
Precondiciones	- SiMon ha sido iniciado con una IP válida. - Se ha seleccionado una simulación.
Resultados	SiMon ha enviado los cambios a realizar en el simulador y éste ha realizado los cambios. El simulador responde a SiMon el nuevo valor del parámetro. SiMon lo muestra al investigador en la ventana de estados del simulador.
Descripción	El Investigador solicita a SiMon que realice el cambio de un parametro en la simulación. Una vez que éste es realizado se le mostrará el nuevo valor por pantalla.

Nombre	Iniciar una Animación.
Actor Principal	Investigador.
Precondiciones	- SiMon ha sido iniciado con una IP válida. - Se ha seleccionado una simulación. - Se han inicializado las condiciones iniciales de la simulación. - Se ha iniciado el envío de datos desde el simulador hacia SiMon.
Resultados	SiMon ha abierto una ventana gráfica (LibPlot/OpenGL) y en su interior se muestra la evolución de la simulación.
Descripción	El Investigador solicita a SiMon que comience a animar.

Universidad del Bío-Bío. Sistema de Bibliotecas - Chile

Nombre	Guardar Datos de Simulación a Archivo.
Actor Principal	Investigador.
Precondiciones	- SiMon ha sido iniciado con una IP válida. - Se ha seleccionado una simulación. - Se ha iniciado el envío de datos desde el simulador hacia SiMon. - Se ha iniciado una medición.
Resultados	SiMon ha creado un archivo y en su interior se encuentra una lista con los valores correspondientes a la medición solicitada.
Descripción	El Investigador solicita a SiMon que guarde localmente los resultados de una simulación.

Universidad del Bío-Bío. Sistema de Bibliotecas - Chile

Capítulo 4

Implementación de la solución

4.1. Protocolo de Comunicación

Para asegurar que la gráfica presentada en el cliente sea consistente con la simulación que corre en el servidor, es necesario que la transmisión de los datos sea tan rápida como sea posible. Es por este motivo que se usará el protocolo UDP, que tiene la característica de ser bastante rápido ya que no establece una conexión, sino que incorpora en cada datagrama[1] la información necesaria para su direccionamiento. Más adelante podría ser necesario hacer cambios a la forma en que estos procesos se comunican para permitir a múltiples clientes estar monitoreando la misma simulación. Por ahora, un cliente podrá monitorear varias simulaciones (siempre una a la vez a menos que tenga una segunda instancia de SiMon en ejecución). Lo contrario, es decir, varios clientes monitoreando la misma simulación, no está soportado en esta versión.

4.2. Programas asociados al monitor de simulaciones

Para hacer posible el monitoreo de una simulación, es necesario considerar los siguientes 3 programas:

- En primer lugar, la simulación en si. A pesar de que el programa que simula ya existía fue necesario hacer ligeras modificaciones en su código para hacer posible que éste se comunicara con otro programa de manera remota. Una estrategia que permite evitar la introducción de código en muchas partes del simulador es la de usar hilos paralelos. De esta forma la simulación corre como siempre lo ha hecho y a la vez, de manera asíncrona, un servidor puede estar esperando mensajes externos y reaccionar ante éstos. Para hacer posible la comunicación, es necesario asociar un puerto disponible (que no esté siendo usado por otra aplicación) al programa. Los primeros 1024 puertos se conocen como *"Puertos Bien Conocidos"* y están reservados para aplicaciones comunes, como HTTP (puerto 80), SSH (puerto 22), FTP (puerto 21), etc. En el sitio de ICANN (Internet Corporation for Assigned Names and Numbers) es posible ver la lista

[1]Un datagrama es una agrupación lógica de información que se envía como unidad (capa de red) a través de un medio de transmisión sin establecer una conexión con anterioridad. Protocolos que usan datagramas: UDP, IPX, broadcasting de video/audio, entre otros.

Universidad del Bío-Bío. Sistema de Bibliotecas - Chile

completa de puertos "reservados" y las aplicaciones a las que corresponden. A pesar de la existencia de este estándar, los problemas que podrían ocurrir al querer usar un puerto "reservado" estarán sujetos nada más que a la existencia de otro programa en ejecución en el mismo equipo que use el puerto seleccionado.
Las simulaciones, al ejecutarse, intentarán hacer uso del puerto 24001. Si tal puerto está en uso, entonces intentarán acoplarse al siguiente (24002). Esto lo harán de manera sucesiva hasta tener éxito o hasta llegar al 24100 (lo que ocurra primero) —esto permite un máximo de 100 simulaciones corriendo en un servidor simultáneamente, lo que es bastante generoso para estos efectos— y entonces se registrarán con el demonio especificando su puerto, nombre, directorio en el que corren y la hora en la que comenzaron su ejecución.

- Si bien el servidor que corre con la simulación es capaz de responder a los comandos de un cliente, no es posible que el cliente sepa a priori en qué puerto está corriendo una simulación en particular. Es bastante común que en un servidor se ejecuten varias simulaciones de manera simultánea. Es por esto que es necesario que un demonio (servicio) esté en permanente ejecución esperando mensajes de clientes que deseen monitorear simulaciones. Como única tarea, el demonio debe responder al comando "-list", que ejecuta el cliente, con la información que éste posee sobre las simulaciones en ejecución (nombre, puerto, directorio y fecha/hora de comienzo).

- Finalmente, el usuario necesitará un cliente que le permita ejecutar comandos para realizar los cambios que desee o comenzar/pausar una simulación. El funcionamiento de SiMon es explicado en detalle en la siguiente sección.

4.3. Cómo funciona el Monitor de Simulaciones

El monitor de simulaciones (SiMon) debe ser ejecutado desde una terminal usando: "simon IP", donde **IP** corresponde a la dirección IP del servidor al que se quiera conectar, se asume que en dicho servidor el demonio está corriendo en el puerto 24000. Opcionalmente, si el usuario desea conectarse sin pasar por el demonio (directamente a una simulación cuyo puerto conoce), podrá entonces, solicitar la conexión usando: "simon IP PUERTO".
Como cualquier intérprete de comandos, el programa inicia por un loop infinito. El usuario deberá escribir en la consola los comandos necesarios para ejecutar las tareas que necesite realizar. Éstos comandos serán interpretados por la consola y se le responderá al usuario según corresponda. En caso de que el usuario cometa un error, el intérprete responderá diciendo que no ha encontrado el comando solicitado.

Básicamente, el cliente se comporta de la siguiente manera:

Universidad del Bío-Bío. Sistema de Bibliotecas - Chile

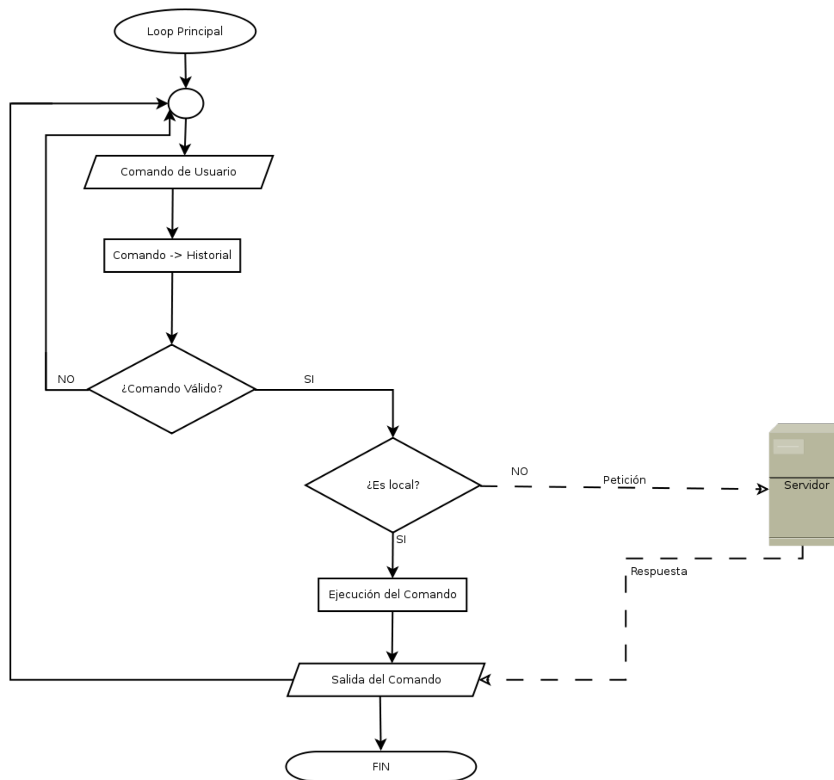


Figura 4.1: Diagrama de Flujo del Intérprete de Comandos.

Inicialmente (ya que esto cambiará en versiones posteriores), los comandos disponibles son:

glut Usando "glut start" el usuario puede iniciar una animación usando OpenGL.

plot Usando "plot start" el usuario puede iniciar una animación usando Libplot.

connect Este comando, seguido del número de puerto, permite al usuario cambiarse a un puerto específico del servidor a que está conectado.

ndiscos Este comando entrega el valor de la variable local "ndiscos", que debe ser consistente con la cantidad de partículas que participan en la simulación.

particle Este comando, seguido por el número de una partícula entrega la posición de la partícula en el instante actual. El número de la partícula debe estar entre 1 y **ndiscos**.

history Desplega la lista de comandos utilizados anteriormente. El usuario puede volver a ejecutar un comando usando "!n", donde **n** es el número que tiene ese comando en la lista mostrada, o "!-n" para hacerlo en orden inverso ("!-1" corresponde al último comando ejecutado).

quit Finaliza la ejecución de SiMon.

Universidad del Bío-Bío. Sistema de Bibliotecas - Chile

Adicionalmente, se han programado los siguientes comandos remotos, que el usuario puede usar para interactuar con la simulación:

-init Realiza la petición al servidor para que éste le entregue los parámetros iniciales que necesitará para inciar una animación.

-start Realiza la petición al servidor para que éste le entregue las posiciones de las partículas para cada instante de tiempo.

-stop Pide al servidor que deje de enviar los datos de posiciones.

-qnincr Aumenta el coeficiente de inelasticidad normal.

-qndecr Disminuye el coeficiente de inelasticidad normal.

-tempincr Aumenta la temperatura en un factor definido por el usuario.

-tempdecr Disminuye la temperatura en un factor definido por el usuario.

-tempmedon Comienza la medición de temperatura.

-tempmedoff Finaliza la medición de temperatura.

-flucdenson Comienza la medición de "autocorrelación temporal de las fluctuaciones de densidad".

-flucdensoff Termina la medición de "autocorrelación temporal de las fluctuaciones de densidad".

-quit Termina el proceso de simulación en el servidor. No podrá retomarse la simulación.

Estos comandos han sido programados inicialmente para dar cumplimiento a los objetivos planteados, pero dan la pauta para configurar todos los tipos de mediciones y peticiones de datos al servidor como lo requiera el investigador.

4.4. Comportamiento del Programa

Como se ha mencionado anteriormente, el monitor de simulaciones (SIMON por su sigla en inglés: Simulation Monitor), debe construirse en una modalidad Cliente/Servidor. La estrategia seleccionada para cumplir con este requerimiento es la de múltiples servidores (uno por cada simulación en ejecución) esperando peticiones de un cliente para comenzar a enviar datos. La secuencia de pasos típica que un usuario debería seguir para monitorear una simulación es como sigue:

- Conectar SiMon al IP del servidor que administra las simulaciones.
- Listar las simulaciones que corren en el servidor (A menos que se sepa exactamente en qué puerto está corriendo la simulación que interesa).
- Conectarse a la simulación que interesa estudiar.
- Comenzar a recibir datos y graficar si es necesario.

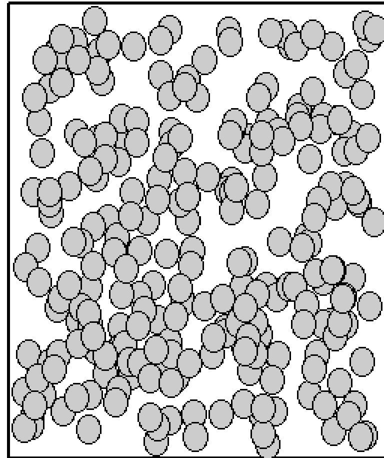
Universidad del Bío-Bío. Sistema de Bibliotecas - Chile

- Cambiar algun parámetro de la simulación, comenzar una medición.
- Terminar el envío de datos.
- Finalizar SiMon.

4.5. Algunas Interfaces

Figura 4.2: Pantalla Inicial del Intérprete de Comandos.

Universidad del Bío-Bío. Sistema de Bibliotecas - Chile



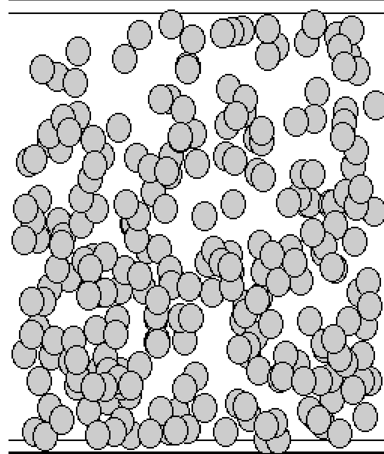


Figura 4.3: Ventana de animación GNU/Libplot.

Universidad del Bío-Bío. Sistema de Bibliotecas - Chile

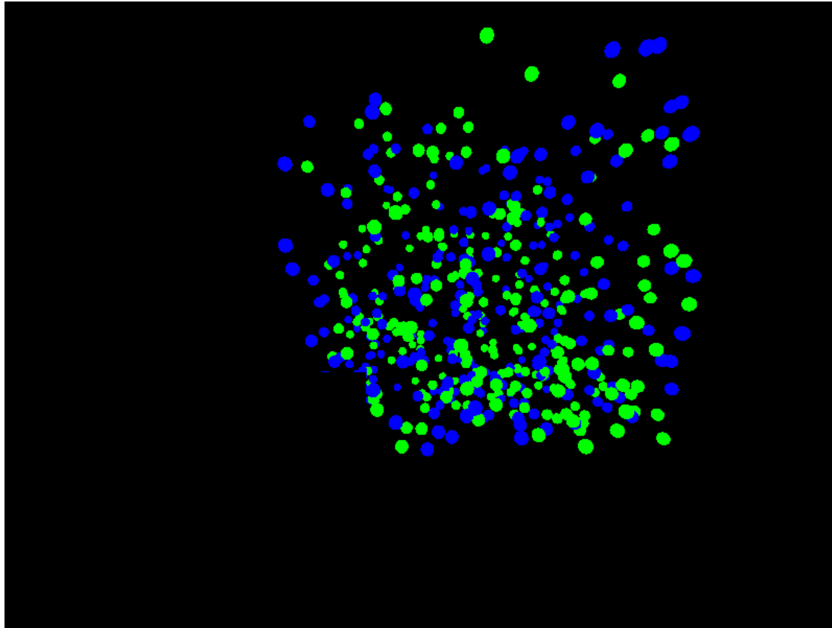


Figura 4.4: Ventana de animación OpenGL, en una caja de 15x15x15 sin mostrar sus bordes.

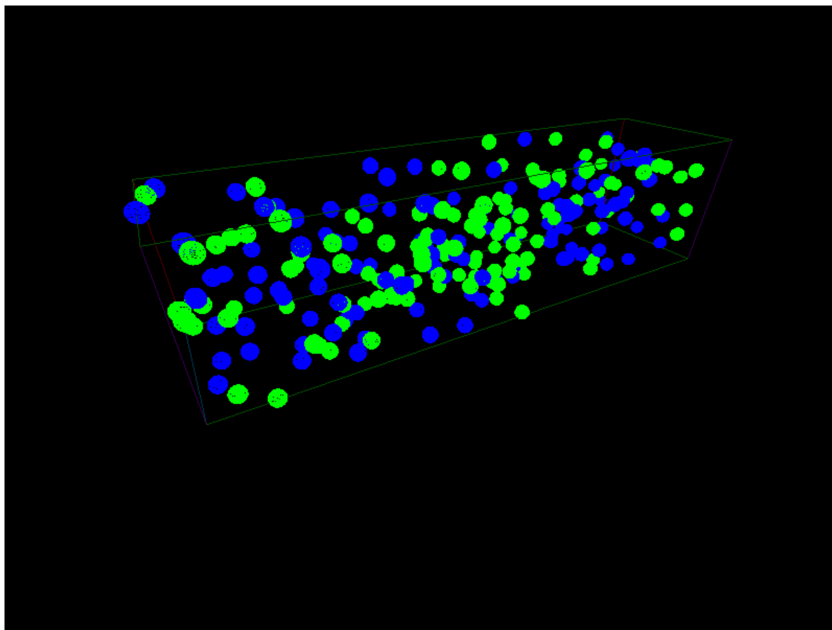


Figura 4.5: Otra ventana de animación OpenGL ahora con una caja de dimensiones 40x10x10, mostrando bordes.

Universidad del Bío-Bío. Sistema de Bibliotecas - Chile

Capítulo 5

Requerimientos

5.1. Hardware

Servidor:

- Procesador: 2,5GHz Dual Core.
- Memoria Principal (RAM): 512MB.
- Disco Duro: 20GB.
- Conexión a red (alta velicidad, Cat6 para el clúster).

Cliente:

- Procesador: 800Mhz
- Memoria Principal: 1GB.
- Tarjeta de Video con aceleración gráfica con al menos 128MB de RAM(ej: nvidia, intel, ati).
- Conexión a red.

Universidad del Bío-Bío. Sistema de Bibliotecas - Chile

5.2. Dependencias de Software

Existen 2 tipos de personas que van a interactuar con SiMon: El usuario que desea ejecutar simulaciones y monitorear su comportamiento y el desarrollador que desea introducir nuevas mediciones o comandos, hacer cambios en el comportamiento del simulador o realizar actualizaciones al código por cualquier motivo. Estas dependencias se distinguen ante la necesidad de compilar nuevamente el código (por parte del desarrollador). El usuario común (que sólo desee ejecutar SiMon) deberá tener instalado Mesa, Ncurses y freeglut3. El desarrollador deberá tener instalados estos paquetes y además el framework de desarrollo “freeglut3-dev” y las siguientes bibliotecas.

Compilador	Esta versión de SiMon ha sido compilada con el “GNU C++ Compiler” (g++) versión 5.4.0.
Libplot	Biblioteca gráfica. Este proyecto ha sido descontinuado, pero se sigue usando en este monitor de simulaciones ya que el GDMTC maneja su programación. A medida que la gráfica 3D en OpenGL vaya progresando, esta biblioteca irá teniendo menor importancia y posiblemente sea retirada del repositorio.
GLee	OpenGL Easy Extension Library.
libglut	OpenGL Utility Toolkit.
libGLU	Bibliotecas de OpenGL.
libGL	Bibliotecas de OpenGL.
libpthread	Bibliotecas de desarrollo de los hilos con el estándar POSIX. Permiten que una aplicación ejecute varios procesos en paralelo.
libncurses	Bibliotecas de desarrollo con Ncurses: Provee una interfaz que permite la creación de ventanas en modo texto.
libhistory	Bibliotecas de desarrollo con History. Usadas en este caso para la creación del archivo histórico.
libreadline	Bibliotecas de desarrollo con Readline.
libX11	Bibliotecas de desarrollo X11.

Universidad del Bío-Bío. Sistema de Bibliotecas - Chile

Capítulo 6

Puesta en marcha y Pruebas

6.1. Cambios en los procedimientos

Si bien SiMon es una herramienta que apoyará la visualización de procesos simulacionales, en una primera etapa los procedimientos no cambiarán demasiado. Deberá haber un período de “marcha blanca” en que los investigadores usarán la vieja versión del simulador y harán comparaciones en cuanto a rendimiento y resultados obtenidos con SiMon. A medida que se vayan ingresando sus mediciones y SiMon vaya teniendo más opciones gráficas, los investigadores podrán sacar mayor provecho a este monitor remoto. Cuando esto ocurra, ya no será problema modificar parámetros sobre simulaciones que están en ejecución, aunque siempre será necesario programar las mediciones al inicio. Principalmente, habrá diferencias en la forma en que se ejecutan las simulaciones, ya que previo a este trabajo los investigadores simplemente ejecutaban su simulación y esperaban a que esta concluyera o analizaban los datos que ésta iba generando. Con SiMon, previo al control que se pueda hacer de la simulación será necesario instalar el “demonio” en el servidor (aunque sólo una vez, ya que éste iniciará con el resto de los servicios del sistema operativo). Posterior a esto las simulaciones podrán ser ejecutadas de la misma forma que se ha hecho siempre.
Otro cambio que ocurrirá en el largo plazo, será que simulaciones que antes se ejecutaban por algunas horas, ahora será posible dejarlas corriendo por varios días y hacer cambios sobre ellas de manera de estudiar no solo su comportamiento ante diferentes condiciones de entorno sino también su reacción ante los cambios de estos (por ejemplo: “cuánto tiempo demora en ocurrir un fenómeno cuando el parámetro X se duplica”).

Universidad del Bío-Bío. Sistema de Bibliotecas - Chile

6.2. Pruebas

6.2.1. Pruebas de Unidad

Las siguientes pruebas se ejecutarán en módulos del sistema para descubrir posibles fallas que pudieran afectar el normal funcionamiento del mismo.

6.2.1.1. Conexión

La siguiente prueba tiene como objetivo comprobar si SiMon es capaz de conectarse al demonio que corre en el servidor y si este último es capaz de responder al comando “#list”mostrando las simulaciones que corren en ese momento.

Programa Involucrado	*Pasos a seguir para reproducir la prueba*
Demonio	Ejecutar el demonio en el servidor usando “/etc/rc.d/simd start” en el caso de distribuciones del tipo ArchLinux/Slackware o agregar “simsrv” (incluido en el servidor en el directorio ‘daemon’) a la lista de demonios en el caso de distribuciones del tipo Debian.
SiMon	Ejecutar SiMon con la dirección IP del servidor en el que corre el demonio.
SiMon	Ejecutar el comando ‘#list’.

Resultado Esperado SiMon muestra en la ventana de estado (a la derecha) la respuesta del servidor con la lista de simulaciones o un mensaje diciendo que no hay simulaciones ejecutándose en este momento.

Resultado Posible SiMon no muestra nada en la ventana de estado. Lo que indicaría que hay un error de conexión.

Resultado Obtenido SiMon muestra en la ventana de estado el mensaje: “No hay simulaciones corriendo”.

6.2.1.2. Historial

La siguiente prueba tiene como objetivo comprobar si SiMon es capaz de guardar en /home/«USUARIO»[1]/.simon_history el historial de comandos ejecutados por el usuario.

[1] «USUARIO» corresponde al nombre del usuario que ha iniciado sesión. En mi caso, mi nombre de inicio de sesión es “juboba”, lo que al crear mi usuario genera el directorio: /home/juboba.

Universidad del Bío-Bío. Sistema de Bibliotecas - Chile

Programa Involucrado	*Pasos a seguir para reproducir la prueba*
SiMon	Ejecutar SiMon con un IP cualquiera.
SiMon	Ejecutar varios comandos en el intérprete de SiMon.
S.O.	Ir al 'home'(/home/«USUARIO»/) y comprobar que existe el archivo (oculto) '.simon_history'.

Resultado Esperado El archivo existe y en su contenido se encuentra una lista de comandos ejecutados por el usuario.

Resultado Posible El archivo no existe.

Resultado Posible El archivo existe pero está vacío.

Resultado Obtenido El archivo existe y contiene la lista de comandos ejecutados por el usuario.

6.2.1.3. Animación Gráfica con LibPlot

Esta prueba tiene como objetivo comprobar que es posible seguir usando la gráfica programada por el GDMTC, pero ahora desde un cliente remoto a la simulación.

Programa Involucrado	*Pasos a seguir para reproducir la prueba*
SiMon	Ejecutar SiMon con el IP del servidor en que corre una simulación.
SiMon	Ejecutar el comando "connect [PORT]", donde PORT corresponde al puerto en el que la simulación espera.
SiMon	Solicitar la animación con libplot usando el comando "plot" con el parámetro "start".

Resultado Esperado Se abre una ventana de animación, mostrando la animación de la simulación.

Resultado Posible La ventana no se abre.

Resultado Posible La ventana se abre pero no muestra nada en su interior.

Resultado Obtenido La ventana titulada "libplot" abre correctamente y muestra la animación.

6.2.1.4. Animación Gráfica con OpenGL

Esta prueba tiene como objetivo la comprobación de que la nueva gráfica funciona y es consistente con la simulación.

Universidad del Bío-Bío. Sistema de Bibliotecas - Chile

Programa Involucrado	*Pasos a seguir para reproducir la prueba*
SiMon	Ejecutar SiMon con el IP del servidor en que corre una simulación.
SiMon	Ejecutar el comando “connect [PORT]”, donde PORT corresponde al puerto en el que la simulación espera.
SiMon	Solicitar la animación con OpenGL usando el comando “glut” con el parámetro “start”.

Resultado Esperado Se abre una ventana de animación, mostrando la animación de la simulación.

Resultado Posible La ventana no se abre.

Resultado Posible La ventana se abre pero no muestra nada en su interior.

Resultado Obtenido La ventana titulada “opengl” abre correctamente y muestra la animación.

6.2.2. Pruebas de Integración

Esta serie de pruebas pretende comprobar que los módulos son capaces de comunicarse entre sí. Estas pruebas son de mayor complejidad que las anteriores y pretenden descubrir fallos de comunicación entre módulos o de consistencia entre los datos entregados por cada uno y lo que reciben.

6.2.2.1. Animación y Aumento del Coeficiente de Inelasticidad

La siguiente prueba tiene como objetivo comprobar si es posible hacer cambio de parámetros durante la ejecución de la simulación. Esta prueba es de gran importancia porque tiene relación directa con los objetivos planteados.

Programa Involucrado	*Pasos a seguir para reproducir la prueba*
SiMon	Ejecutar SiMon con el IP del servidor en que corre una simulación.
SiMon	Ejecutar el comando “connect [PORT]”, donde PORT corresponde al puerto en el que la simulación espera.
SiMon	Solicitar la animación con OpenGL usando el comando “glut” con el parámetro “start”.
SiMon	Regresar al intérprete de comandos de SiMon y solicitar un aumento del coeficiente de inelasticidad con el comando “#qnincr”.
SiMon	Repetir el paso anterior sucesivamente.

Resultado Esperado El servidor responde con el nuevo valor del coeficiente de inelasticidad.

Resultado Posible El servidor no responde.

Resultado Posible El servidor responde siempre con el mismo valor.

Universidad del Bío-Bío. Sistema de Bibliotecas - Chile

Resultado Obtenido El servidor responde y entrega el nuevo valor. Hacer esta prueba de manera sucesiva incrementa el coeficiente hasta que es posible ver cambios en la animación (el factor de aumento está especificado en el archivo de configuración en el servidor).

6.2.2.2. Animación y Comenzar a medir la temperatura

La siguiente prueba tiene como objetivo comprobar si es posible —durante la ejecución de la simulació— niniciar una medición programada antes de ejecutarla. Esta prueba es también de gran importancia porque tiene relación directa con los objetivos planteados.

Programa Involucrado	*Pasos a seguir para reproducir la prueba*
SiMon	Ejecutar SiMon con el IP del servidor en que corre una simulación.
SiMon	Ejecutar el comando “connect [PORT]”, donde PORT corresponde al puerto en el que la simulación espera.
SiMon	Solicitar la animación con OpenGL usando el comando “glut” con el parámetro “start”.
SiMon	Regresar al intérprete de comandos de SiMon y solicitar el comienzo de la medición de temperatura con el comando “#tempmedon”.
SiMon	Dejar que mida un tiempo.
SiMon	Detener la medición haciendo uso del comando “#tempmedoff”.

Resultado Esperado El servidor muestra por pantalla y guarda en un archivo los valores de la temperatura.

Resultado Posible El servidor no responde.

Resultado Posible El servidor crea el archivo pero no guarda nada.

Resultado Obtenido El servidor muestra por pantalla y guarda en el archivo los valores de la temperatura durante el tiempo en el que estuvo activa la medición.

Universidad del Bío-Bío. Sistema de Bibliotecas - Chile

Capítulo 7

Conclusiones y Proyecciones

El presente trabajo ha cumplido con documentar y especificar el comportamiento y estrategia utilizada en el desarrollo del monitor de simulaciones SiMon. Dentro de las metas logradas es posible destacar que el GDMTC cuenta ahora con una herramienta que apoya —de manera eficiente— el estudio de medios granulados, tanto para su aplicación en investigación como para docencia. Además se ha logrado establecer una política estándar para las nuevas revisiones y versiones del Monitor de Simulaciones (SiMon). De esta forma será posible que otros estudiantes o los mismos investigadores puedan proveer modificaciones a las fuentes y que estas sean revisadas e integradas en el repositorio "oficial" como nuevos releases y así poder mantener un sistema ordenado y en constante revisión.

7.1. Aprendizajes Logrados

En lo personal, este proyecto ha resultado beneficioso en varios aspectos, entre los que se encuentran:
En primer lugar, mejorar la capacidad de trabajar en un código escrito por desarrolladores de otras áreas con una visión distinta del problema. El estudio del código del simulador significó algo más de 3 semanas, en la primera etapa del proyecto, para entender su estrategia, la estructura de directorios, los métodos a seguir para la inserción de mediciones y la planificación de los mismos.
Además, el aprendizaje logrado de OpenGL y GLUT permite que pueda seguir trabajando en proyectos de este tipo, en el desarrollo de plugins gráficos para otro tipo de aplicaciones e incluso en la creación de juegos con interfaces tridimensionales.
El desarrollo de este proyecto me ha obligado a relacionarme con personas de distinta formación académica, con motivaciones distintas a las monetarias que dedican gran parte de su tiempo a la investigación de fenómenos que aún no tienen respuesta y sus aportes son mundialmente reconocidas. Esto me ha motivado a seguir trabajando con el GDMTC, tanto con el simulador como con SiMon, para mantener un control de las futuras versiones, de esta forma cualquier intervención al código será bienvenida, pero las versiones oficiales se mantendrán bajo un control para mantener un estándar respecto al estilo de programación y las estrategias empleadas. Espero que en poco tiempo podamos tener una versión GNU disponible para todo el mundo.
Para la composición de este trabajo hice uso de un conjunto de herramientas que facilitaron la elaboración de este informe y la programación de la aplicación:

Universidad del Bío-Bío. Sistema de Bibliotecas - Chile

- LaTeXes un sistema de composición de textos, orientado especialmente a la creación de libros, documentos científicos y técnicos que contengan fórmulas matemáticas. La creación de este documento fue elaborada por completo en este lenguaje escrito por Leslie Lamport (1984).

- Editor de textos Vi: Para mi gusto el editor más poderoso y cómodo en interfaz de línea, aunque existe también GVim que posee una GUI[1]. Provee una gran versatilidad a la hora de mover bloques de texto, hacer reemplazo con expresiones regulares, etc. Incluso permite insertar directamente la salida de comandos desde el sistema operativo. Todo este informe y los códigos fuentes del programa fueron escritos haciendo uso de este editor.

- Subversion: Es un sistema de control de versiones, que permite mantener un "repositorio[2]" remoto (en un servidor, en este caso en la oficina del profesor Dino Risso). Subversion facilita bastante el trabajo de varias personas sobre las mismas fuentes, ya que se encarga de mostrar los errores de inconsistencia que puedan existir entre una versión y otra. Este documento se encuentra en su revisión 451 y es posible descargar todas las versiones anteriores (hasta llegar a la primera en que simplemente hay una página en blanco).

- OpenGL: La *"Biblioteca Gráfica Abierta"* (Open Graphics Library) provee una poderosa API para el desarrollo de aplicaciones gráficas. A diferencia de DirectX (de Microsoft), OpenGL es completamente Open Source y es posible obtener una gran cantidad de documentación y ejemplos en internet.

- El trabajo con Sockets es de gran importantancia a la hora de comunicar procesos. Durante el curso de "Comunicación de datos y Redes" obtuve algunas nociones del funcionamiento de los sockets, pero el trabajo directo con ellos a bajo nivel me permitió conocer no sólo cómo comunicar procesos en máquinas distintas sino también la comunicación de procesos en la misma máquina a través de señales y "pipes[3]".

- Software Libre: Todos los programas usados para la creación de esta memoria son de disrtibución libre.

[1] Graphic User Interface (Interfaz gráfica de usuario)

[2] Un repositorio es un sitio centralizado donde se almacena información digital. Muchos programas libres mantienen en repositorios versiones anteriores a las oficiales para permitir que los usuarios las descarguen y puedan enviar sugerencias para las liberaciones oficiales. El repositorio de esta memoria está alojado en `svn://maxwell.ciencias.ubiobio.cl/graphic_server`

[3] *"En ingeniería de software, una tubería nombrada (named pipe en inglés), también llamada FIFO por su comportamiento, es una extensión del concepto tradicional de tuberías utilizado en los Sistemas operativos POSIX, y es uno de los métodos de Comunicación entre procesos (IPC). Este concepto también se encuentra en Windows, si bien implementado con otra semántica." (Wikipedia)*

Universidad del Bío-Bío. Sistema de Bibliotecas - Chile

7.2. Proyecciones y Trabajos Futuros

Si bien la primera versión de SiMon es bastante funcional y cumple con los objetivos planteados para este proyecto, para un trabajo futuro quedan bastantes cosas que mejorar y agregar. Algunas modificaciones que deben estudiarse en el corto plazo son:

Protocolo de Comunicación El protocolo que se utiliza actualmente es UDP (User Datagram Protocol) que —como hice mención anteriormente— tiene la cualidad de ser bastante rápido, pero no hace control de transmisión. Esto implica que algunos mensajes que envía el servidor pueden simplemente no llegar por problemas de congestión, desconexión u otro. En una futura versión se pretende mejorar esto haciendo uso de un protocolo mixto con TCP o SCTP.

Animación con gráfica 3D En esta primera versión los gráficos mostrados son bastante básicos. Aún no es posible graficar nuevos objetos que se introduzcan a la simulación. En una próxima versión esto deberá resolverse, añadiendo un protocolo para objetos indicando su tamaño y características especiales, asi como distinguir por color o textura partículas con características diferentes a las demás (Multitipos).

Interfaz para modificación de parámetros En el futuro se deseará incluir en la ventana de gráfica 3D controles que permitan hacer el cambio de parámetros de una manera más amigable, con “sliders”, botones y campos de texto. Actualmente existen entradas de menú para algunas acciones, pero los máximos y mínimos que pueden alcanzar las variables asociadas no están acotados.

Además de estos cambios a corto plazo, los desarrolladores podrán hacer introducción de nuevas mediciones y nuevas funcionalidades, según lo que necesiten. Es por esto, que el código se mantendrá abierto y bien documentado.

7.3. Licencia

Por ahora el código de mantendrá propietario hasta que se tenga una versión con documentación suficiente; entonces pretendemos liberarlo bajo alguna licencia GNU[4]. Este sistema ha sido programado de manera que cualquier desarrollador pueda integrar el servidor en su programa de simulaciones y usar SiMon como su monitor simulacional.

[4]“*La Licencia Pública General de GNU o más conocida por su nombre en inglés GNU General Public License o simplemente sus siglas del inglés GNU/GPL, es una licencia creada por la Free Software Foundation en 1989 (la primera versión), y está orientada principalmente a proteger la libre distribución, modificación y uso de software. Su propósito es declarar que el software cubierto por esta licencia es software libre y protegerlo de intentos de apropiación que restrinjan esas libertades a los usuarios.*”(Wikipedia)

Universidad del Bío-Bío. Sistema de Bibliotecas - Chile

Apéndice A

Diccionario de Datos (DFD)

Nombre del Flujo	Comando.
Origen	Investigador.
Tipo de Origen	Entidad.
Destino	Intérprete de Comandos.
Tipo de Destino	Proceso.
Descripción	El Investigador le indica al proceso "Intérprete de Comandos" el comando que desea ejecutar.

Nombre del Flujo	Comando.
Origen	Interpretar Comandos.
Tipo de Origen	Proceso.
Destino	Procesar Gráficos.
Tipo de Destino	Proceso.
Descripción	El comando que ha ingresado el usuario se parsea y se envía al Procesador Gráfico para que éste despliegue la animación por pantalla.

Nombre del Flujo	Comando.
Origen	Interpretar Comandos.
Tipo de Origen	Proceso.
Destino	Enviar Mensajes.
Tipo de Destino	Proceso.
Descripción	El comando que ha ingresado el usuario se parsea y se prepara para ser enviado al servidor, añadiendo la cabecera necesaria para direccionamiento.

Universidad del Bío-Bío. Sistema de Bibliotecas - Chile

Nombre del Flujo	Comando.
Origen	Interpretar Comandos.
Tipo de Origen	Proceso.
Destino	History.
Tipo de Destino	Archivo.
Descripción	El comando se guarda en un archvio oculto (en el directorio personal del usuario) llamado ".simon_history" que servirá posteriormente para reutilizarlos más rápidamente.

Nombre del Flujo	Animación.
Origen	Procesar Gráficos.
Tipo de Origen	Proceso.
Destino	Investigador.
Tipo de Destino	Entidad.
Descripción	La animación es desplegada por pantalla lo que permite al Investigador tomar desiciones para lo que desee hacer después, que puede ser hacer cambios o guardar estados de la simulación.

Nombre del Flujo	Comando Servidor.
Origen	Recibir Mensajes (Cliente).
Tipo de Origen	Proceso.
Destino	Investigador.
Tipo de Destino	Entidad.
Descripción	Si el investigador decidió guardar datos localmente, el servidor le enviará dichos datos. Éstos pueden ser utilizados de base para gráficos y otro tipo de reportes.

Nombre del Flujo	Resultados.
Origen	Enviar Mensajes (Cliente).
Tipo de Origen	Proceso.
Destino	Recibir Mensajes (Servidor).
Tipo de Destino	Proceso.
Descripción	Todos los comandos que comienzan con un "#" son enviados sin parear al servidor para que éste los procese.

Universidad del Bío-Bío. Sistema de Bibliotecas - Chile

Nombre del Flujo	Petición de Posición.
Origen	Recibir Mensajes (Servidor).
Tipo de Origen	Proceso.
Destino	Calcular Posiciones de Partículas.
Tipo de Destino	Proceso.
Descripción	Si la simulación está en ejecución, el servidor deberá calcular las posiciones de las partículas para todo instante de tiempo. Cuando el cliente solicita animar, éstas posiciones deben ser enviadas al cliente.

Nombre del Flujo	Petición de Medición.
Origen	Recibir Mensajes (Servidor).
Tipo de Origen	Proceso.
Destino	Medir Variables.
Tipo de Destino	Proceso.
Descripción	Ante la solicitud del Investigador, el simulador debe comenzar a medir la variable anteriormente programada.

Nombre del Flujo	Posición de Partícula.
Origen	Calcular Posiciones de Partículas.
Tipo de Origen	Proceso.
Destino	Medir Variables.
Tipo de Destino	Proceso.
Descripción	Si existe una medición en ejecución se deberán enviar todos los datos de posiciones de partículas a “Medir Variables” para que éstos puedan ser procesados.

Nombre del Flujo	Posición de Partícula.
Origen	Calcular Posiciones de Partículas.
Tipo de Origen	Proceso.
Destino	Enviar Mensajes (Servidor).
Tipo de Destino	Proceso.
Descripción	Las posiciones de cada partícula deben ser enviadas al cliente para poder ser animadas. “Enviar Mensajes” las procesará previamente añadiéndoles una cabecera con la información de direccionamiento.

Universidad del Bío-Bío. Sistema de Bibliotecas - Chile

Nombre del Flujo	Resultado de Medición.
Origen	Medir Variables.
Tipo de Origen	Proceso.
Destino	Enviar Mensajes (Servidor).
Tipo de Destino	Proceso.
Descripción	Si existe una medición en ejecución se deberán enviar todos los resultados de mediciones a "Enviar Mensajes" para que éstos puedan ser procesados, añadiéndoles una cabecera con la información de direccionamiento.

Nombre del Flujo	Resultados.
Origen	Enviar Mensajes (Servidor).
Tipo de Origen	Proceso.
Destino	Recibir Mensajes (Cliente).
Tipo de Destino	Proceso.
Descripción	Todos los datos preparados para enviarse (que posean información de direccionamiento) deberán ser enviados al Cliente (Recibir Mensajes) para su posterior proceso, ya sea guardándolos, graficándolos o animándolos.

Universidad del Bío-Bío. Sistema de Bibliotecas - Chile

Apéndice B

Documentación

B.1. main.c Referencia del Archivo

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <arpa/inet.h>
#include <strings.h>
#include <signal.h>
#include <pthread.h>
#include <string.h>
#include <unistd.h>
#include <ncurses.h>
#include <readline/readline.h>
#include <readline/history.h>
#include "headers/Client.h"
```

Definiciones

- #define **VERSION** "0.2"
- #define **_client_**

Funciones

- void **ncurses_init** (void)

Universidad del Bío-Bío. Sistema de Bibliotecas - Chile

- int **main** (int argc, char ∗argv[ ])
- void **salir** (int err)
- int **create_socket** (char ∗ip, char ∗puerto)
- void ∗ **recv_data** (void ∗p)
- int **send_msg** (char ∗message)
- void **bienvenida** (void)

Variables

- int **Puerto**

  Puerto (en el servidor) para enviar datos, definido por el usuario (24000 por defecto).

- socklen_t **len**

  Variable para almacenaar el largo del socket (en bytes).

- char **sendbuffer** [128]

  Buffer de Salida, para el envío de datos hacia el servidor.

- char **recvbuffer** [128]

  Buffer de Entrada, para la recepción de datos desde el servidor.

- struct sockaddr_in **serv_addr**

  Estructura para almacenar datos de la conexión.

- int **numbytes**

  Número de bytes recibidos.

- FILE ∗ **posiciones**

  Archivo temporal para almacenar las posiciones que llegan.

- WINDOW ∗ **footer**

  Ventana Ncurses para mostrar mensajes enviados por el servidor.

- char **running_sim** [50]

  String que almacena la información entregada por el servidor de cada simulación en ejecución.

- char ∗ **hist_file_name**

  Archivo para guardar los comandos ingresados (historial). Esto hará posible recuperar comandos ya usados.

Universidad del Bío-Bío. Sistema de Bibliotecas - Chile

B.1.1. Documentación de Definiciones

B.1.1.1. #define _client_

B.1.1.2. #define VERSION "0.2"

B.1.2. Documentación de Funciones

B.1.2.1. void bienvenida (void)

Muestra la bienvenida a SIMON.

Parámetros

Nada.

Retorna

Nada.

B.1.2.2. int create_socket (char * *ip*, char * *puerto*)

Crea un socket para recibir los mensajes del servidor.

Parámetros

String IP, String Puerto.

Retorna

0 si se ejecuta con éxito, de lo contrario terminará la ejecución del programa mostrando el código de error y su descripción.

B.1.2.3. int main (int *argc*, char * *argv*[ ])

Loop Principal

B.1.2.4. int ncurses_init (void)

Inicialización de entorno Ncurses.

Parámetros

Nada.

Retorna

0 si no hay problemas.

Universidad del Bío-Bío. Sistema de Bibliotecas - Chile

B.1.2.5. void∗ recv_data (void ∗ *p*)

Recibe los mensajes del servidor. Debe correr en un nuevo thread.

Parámetros

Nada.

Retorna

Nada.

B.1.2.6. void salir (int *err*)

Termina la ejecución de SIMON. Se encarga de terminar la ejecución del receptor, cerrar el socket y archivo. Si es llamada con un código de error se encargará de mostrar el error. Además escribe un archivo con el historial de comandos ejecutados.

Parámetros

Código de error.

Retorna

Nada.

B.1.2.7. int send_msg (char ∗ *message*)

Envía mensajes al servidor si es necesario. Los mensajes que irán al servidor comienzan con un '#'.

Parámetros

String mensaje para enviar.

Retorna

0 si se ejecuta con éxito, de lo contrario terminará la ejecución del programa mostrando el código de error y su descripción.

B.1.3. Documentación de Variables

B.1.3.1. WINDOW∗ footer

Ventana Ncurses para mostrar mensajes enviados por el servidor.

B.1.3.2. char∗ hist_file_name

Archivo para guardar los comandos ingresados (historial). Esto hará posible recuperar comandos ya usados.

Universidad del Bío-Bío. Sistema de Bibliotecas - Chile

B.1.3.3. socklen_t len

Variable para almacenaar el largo del socket (en bytes).

B.1.3.4. int numbytes

Número de bytes recibidos.

B.1.3.5. FILE∗ posiciones

Archivo temporal para almacenar las posiciones que llegan.

B.1.3.6. int Puerto

Puerto (en el servidor) para enviar datos, definido por el usuario (24000 por defecto).

B.1.3.7. char recvbuffer[128]

Buffer de Entrada, para la recepción de datos desde el servidor.

B.1.3.8. char running_sim[50]

String que almacena la información entregada por el servidor de cada simulación en ejecución.

B.1.3.9. char sendbuffer[128]

Buffer de Salida, para el envío de datos hacia el servidor.

B.1.3.10. struct sockaddr_in serv_addr

Estructura para almacenar datos de la conexión.

B.2. Command.c Referencia del Archivo

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <ncurses.h>
```

Universidad del Bío-Bío. Sistema de Bibliotecas - Chile

```
#include <pthread.h>
#include <readline/history.h>
#include "headers/Client.h"
#include "../config.cfg"
```

Definiciones

- #define **_Command_**

Funciones

- char ** **cmd_parse** (char *cmd)
- int **cmd_count_args** (char **args)
- int **cmd_to_int** (char *cmd)
- void **display_help** (char *cmd)
- void **glut_** (char *glut_arg)
- void **cmd_execute** (char *cmd)

  Ejecuta el comando pedido por el usuario. Esta función hace uso de todas las otras funciones para ejecutar el comando deseado.

- int **glut_cmd_to_int** (char *cmd)

B.2.1. Documentación de Definiciones

B.2.1.1. #define _Command_

B.2.2. Documentación de Funciones

B.2.2.1. int cmd_count_args (char ** *args*)

Una función que devuelve el equivalente a "int argc".

Parámetros

String cadena de comando y argumentos.

Retorna

Un entero equivalente al número de argumentos recibidos incluido el comando (primer argumento).

Universidad del Bío-Bío. Sistema de Bibliotecas - Chile

B.2.2.2. void cmd_execute (char * *cmd*)

Ejecuta el comando pedido por el usuario. Esta función hace uso de todas las otras funciones para ejecutar el comando deseado.

Parámetros

String cadena del tipo "comando argumento1 argumento2 ... argumentoN".

Retorna

nada.

B.2.2.3. char ** cmd_parse (char * *cmd*)

Esta función hace un parsing de un gran string que incluye al comando y sus argumentos. Es decir, el equivalente a tener "char **argv".

Parámetros

String cadena del tipo "comando argumento1 argumento2 ... argumentoN".

Retorna

String[] arreglo de cadenas de la forma {"comando", "argumento1", "argumento2", ..., "argumentoN"}

B.2.2.4. int cmd_to_int (char * *cmd*)

Retorna un entero equivalente al comando ejecutado. En caso de ser un comando no reconocido retorna -1.

Parámetros

String Comando.

Retorna

Entero equivalente al comando.

B.2.2.5. void display_help (char * *cmd*)

Muestra la ayuda para el comando seleccionado.

Parámetros

String Comando.

Retorna

Nada.

Universidad del Bío-Bío. Sistema de Bibliotecas - Chile

B.2.2.6. void glut_ (char * *glut_arg*)

Ejecuta o detiene una animación en OpenGL.

Parámetros

String Comando.

B.2.2.7. int glut_cmd_to_int (char * *cmd*)

Retorna un entero equivalente al argumento ejecutado.

Parámetros

String Comando

Retorna

Entero que representa al comando.

B.3. glut.cpp Referencia del Archivo

```
#include <math.h>
#include <stdlib.h>
#include <stdio.h>
#include <pthread.h>
#include "gltools.h"
#include "glframe.h"
#include "../headers/Client.h"
```

Definiciones

- #define **_grglut_**

Funciones

- void **getPos** ()
- void **SetupRC** ()
- void **RenderScene** (void)
- void **SpecialKeys** (int key, int x, int y)
- void **ChangeSize** (int w, int h)
- void **Menu** (int value)

Universidad del Bío-Bío. Sistema de Bibliotecas - Chile

- void **SubMenuQn** (int value)
- void **SubMenuTemp** (int value)
- void **IdleFunc** (void)
- void **Timer** (int a)
- void ∗ **glut** (void ∗p)

Variables

- GLfloat **windowWidth**
- GLfloat **windowHeight**
- GLFrame **camara**
- GLfloat **x_GLUT** [10002]
- GLfloat **y_GLUT** [10002]
- GLfloat **z_GLUT** [10002]
- GLfloat **X1**
- GLfloat **X2**
- GLfloat **Y1**
- GLfloat **Y2**
- GLfloat **Z1**
- GLfloat **Z2**
- GLfloat **globFact** = 0.1f
- int **window_id**
- int **submenuQn**
- int **submenuTemp**
- int **pause_anim**
- int **temp_med**

B.3.1. Documentación de Definiciones

B.3.1.1. #define _grglut_

B.3.2. Documentación de Funciones

B.3.2.1. void ChangeSize (int *w*, int *h*)

Función que redibuja cuando la ventana cambia de tamaño.

B.3.2.2. void getPos ()

Traer posiciones de las partículas desde el simulador.

Universidad del Bío-Bío. Sistema de Bibliotecas - Chile

B.3.2.3. void$*$ glut (void $*$ *p*)

B.3.2.4. void IdleFunc (void)

Función que se ejecuta en espacios de tiempo óciosos'.

B.3.2.5. void Menu (int *value*)

Definición del Menu Principal.

B.3.2.6. void RenderScene (void)

Dibujar Escena.

B.3.2.7. void SetupRC ()

Inicialización para renderizado.

B.3.2.8. void SpecialKeys (int *key*, int *x*, int *y*)

Reaccionar ante eventos de teclado. En el caso de las flechas se deberá mover la cámara según corresponda.

Parámetros

Coordenadas de la posición del mouse y tecla presionada.

B.3.2.9. void SubMenuQn (int *value*)

Definición de Submenú de ejemplo (Coeficiente de Inelasticidad).

B.3.2.10. void SubMenuTemp (int *value*)

Definición de Submenú de ejemplo (Temperatura).

Universidad del Bío-Bío. Sistema de Bibliotecas - Chile

B.3.2.11. void Timer (int *a*)

B.3.3. Documentación de Variables

B.3.3.1. GLFrame camara

B.3.3.2. GLfloat globFact = 0.1f

B.3.3.3. int pause_anim

B.3.3.4. int submenuQn

B.3.3.5. int submenuTemp

B.3.3.6. int temp_med

B.3.3.7. int window_id

B.3.3.8. GLfloat windowHeight

B.3.3.9. GLfloat windowWidth

B.3.3.10. GLfloat X1

B.3.3.11. GLfloat X2

B.3.3.12. GLfloat x_GLUT[10002]

B.3.3.13. GLfloat Y1

B.3.3.14. GLfloat Y2

B.3.3.15. GLfloat y_GLUT[10002]

B.3.3.16. GLfloat Z1

B.3.3.17. GLfloat Z2

B.3.3.18. GLfloat z_GLUT[10002]

Universidad del Bío-Bío. Sistema de Bibliotecas - Chile

Bibliografía

[1] *Efficient Algorithms for Many-Body Hard Particle Molecular Dynamics*, M. Marín, D. Risso y P. Cordero, J. Comp. Phys. 109 306 (1993)

[2] *Extended event driven molecular dynamics for simulating dense granular matter*, S. Gonzalez, D. Risso and R. Soto, The European Physical Journal 179 (2009).

[3] *Equation of State Calculations by Fast Computing Machines*, N. Metropolis, A. W. Rosenbluth, M. N. Rosenblut, A. H. Teller y E. Teller, J. Chem. Phys. 21 (1953) 1087-1092.

[4] *Statistical Mechanics*, D. A. McQuarrie, Harper and Row Pub. N.Y. (1973)

[5] *Phase Transition for a Hard Sphere System*, B. J. Alder and T. E. Wainwright, J. Chem. Phys. 27 (1957) 1208-1209.

[6] G. H. Vinneyard, in: Interatomic Potentials and Simulation of Lattice Defects, eds P.C. Gehlen, J.R. Beeler and R. I.Jafee (Plenum Press, N.Y. 1972).

[7] *Correlations in the Motion of Atoms in Liquid Argon* A. Rahman, Phys. Rev. 136 2A (1964) 405-411.

[8] *Studies in Molecular Dynamics. I General Method*, B. J. Alder and T. E. Wainwright, J. Chem. Phys. 31 (1958) 459.

[9] *Computer "experiments" on Classical Fluids. I. Thermodynamical Properties of Lennard-Jones Molecules*, L. Verlet, Phys. Rev. 159 (1967) 9-103.

[10] *Molecular Dynamics* Wm. G. Hoover, Lecture Notes in Physics 258, Springer Verlag (1986).

[11] *A hierarchival $O(N \log N)$ force calculation algorithm*, *Nature*, 324(4) (1986) 446-449.

[12] *GPU Gems: Programming Techniques, Tips and Tricks for Real-Time Graphics*, Randima Fernando, Addison-Wesley Pub. (2004); *GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation*, Matt Pharr and Randima Fernando, Addison-Wesley Pub. (2005); *GPU Gems 3* Hubert Nguyen, Addison-Wesley Pub. (2007).

[13] *The event scheduling problem in molecular dynamics simulation*, D. C. Rapaport, J. Comput. Phys. 34 (1980), 184.

[14] *How to Simulate Billiards and Similar Systems*, B. Lubachebsky, J. Comput. Phys. 94 (1991) 255.

Universidad del Bío-Bío. Sistema de Bibliotecas - Chile

[15] *Handbook of Algorithms and Data Structures in Pascal and C* G. Gonnet y R. Baeza-Yates ed. (1992).

[16] Jaques Duran, *Sands, Powders, and Grains*, Springer Verlag, N.Y (2000).

[17] *Liquid-solid-like transition in quasi-one-dimensional driven granular media* M. G. Clerc, P. Cordero, J. Dunstan, K. Huff, N. Mujica, D. Risso and G. Varas, Nature Physics, 4 249 (2008)

[18] *OpenGL SuperBible Fourth Edition* Richard S. Wright, Jr., Benjamin Lipchak, Nicholas Haemel (2007)

[19] *NCURSES Programing HowTo* Pradeep Padala (2005)

[20] *Slackware Linux Essentials* Alan Hicks, Murray Stokely, FuKang Chen, Chris Lumens, David Cantrell and Logan Johnson (2005)

[21] *The GNU C Library Reference Manual* Richard M. Stallman, Roland McGrath, Andrew Oram and Ulrich Drepper (2007)

[22] *Redes de Computadores, Un Enfoque Descendente Basado en Internet* James Kurose, James F. Ross, Keith W. (2004)

[23] *Redes de Computadoras, Cuarta Edición* Andrew S. Tanenbaum (2003)