UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS
FACULTAD 5

“Simulación de Fluidos para el Proyecto Simulador Quirúrgico”

Trabajo para optar por el Título de Ingeniería en Ciencias Informáticas

Autores: Yaself Machado Tugores.

Yunior Miguel Almaguer Bajuelo.

Tutor: Ing. Jaime González Campistruz.

Ciudad de La Habana

Julio 2008.

Frase.

"La educación es una cosa admirable, pero es menester recordar de vez en cuando, que ninguna cosa valiosa para el conocimiento se puede enseñar."

Oscar Wilde.

Declaración de Autoría.

Declaramos que somos los únicos autores de este trabajo y autorizamos al Proyecto “Herramientas de Desarrollo para Sistemas de Realidad Virtual”, de la Facultad 5 de la Universidad de las Ciencias Informáticas, a hacer uso del mismo en su beneficio.

Para que así conste firmamos la presente a los ____ días del mes de _____ del año _____.

Autores:

Yaself Machado Tugores. Yunior Miguel Almaguer Bajuelo.

_____ _____

Tutor:

Ing. Jaime González Campistruz.

_____

Datos de Contacto.

Nombre y Apellidos: Jaime González Campistruz.

Edad: años

Ciudadanía: cubano

Institución: Universidad de las Ciencias Informáticas (UCI)

Título: Ingeniero en Ciencias de la Informática.

Categoría Docente: Profesor Adiestrado.

E-mail: jgonzalezc@uci.cu

Graduado de la UCI, con un año de experiencia en el tema de la Gráfica Computacional

Dedicatoria.

A mis padres por sus consejos, por su apoyo, por ayudarme a ser una mejor persona y sobre todo por ayudarme a conseguir este sueño.

... a mis abuelos donde quiera que estén por todo el cariño que me dieron mientras los tuve a mi lado

... a mi hermana por quererme tanto, aunque a veces no lo demuestre te quiero Tata.

... a mi hermano y su familia, por soportar sin que fuera a visitarlos, siempre la paso bien con ustedes.

... a mi hermana Baby.

Yaself.

A mi familia por todo el apoyo que me han dado, en especial a mis padres que son mi orgullo, a mis abuelos por comprenderme en todo momento, a mi hermana por estar presente en los momentos difíciles.

A mis amigos de toda la vida.

Yunior.

Agradecimientos.

A la Revolución por darnos la oportunidad de cumplir nuestro sueño, a nuestro tutor por ayudarnos en todo momento y por no volverse loco con nuestras dudas, a todos los que de una forma u otra nos apoyaron en la realización de este trabajo, en especial a Leo, Inti, Silva (El loquillo), Igr, Juan Manuel, Zenaida y a Susej.

Yaself:

Le agradezco a mi familia en especial a mis padres por estar siempre ahí cuando los he necesitado,
... a todos mis amigos los viejos y los nuevos,
En especial a los que han estado siempre a los que me han ayudado, a todos los de mi grupo,
... a Susej por tener la paciencia de enseñarme los contenidos el día antes, y por estar siempre ahí.
... a Lester "Markito" por aguantar mis pesadeces, y todos los demás del apto,
Yariel "El niche", Leonardo"Leo", Rainer, Yerandi "El cundo", Ricardo "el Ricar", Yohandri "El fácil",
Yeniel "El flaco", Alexeidis "Candyman", Reinier "el Chacal", Yausell "El médico",
Y sobre todo a Yunior "Pupi" por ser mi compañero en este trabajo por aguantarme y por ser el mejor aunque ponga las cosas en comentario pa' perderme, ah y sobre todo por la buena ortografía.

Yunior:

Le agradezco a mi familia por estar presente en todo momento, a mi novia Grisell por su comprensión en todos los momentos vividos, a mis amigos viejos, que siempre estaban en los momentos difíciles, a los amigos nuevos, que siempre creyeron en mi, le agradezco en especial a mi grupo por todas las grandes aventuras vividas en estos 5 años, en fin a todos los que confiaron en mi.

Resumen.

Los Sistemas de Realidad Virtual (SRV) han tomado gran importancia para la medicina, pues se ha logrado realizar diferentes simuladores quirúrgicos para entrenar a cirujanos y estudiantes. Un efecto muy importante y necesario para darle mayor realismo a estos simuladores es el de fluidos. Este trabajo se basa en crear un efecto que simule fluidos (en este caso la sangre) para el Simulador Quirúrgico (KHEIPROS) que actualmente se desarrolla en la Universidad de Ciencias Informáticas (UCI). El trabajo está dividido en dos partes para lograr una mayor comprensión del mismo: Simulación Visual y Simulación Física. Para la Simulación Física se utiliza el método Hidrodinámica de las Partículas Suavizadas (Smoothed Particle Hydrodynamics, “SPH”), que se usó primeramente para resolver problemas de dinámica de gases en Astrofísica, adaptándose posteriormente para aplicarse en problemas de fluidos incompresibles. Para la Simulación Visual se realizó un Sistema de Partículas Acopladas, que se complementa con el modelo físico anteriormente citado (SPH). Para la realización de este trabajo se utilizo el lenguaje C++ y la interfaz gráfica en OpenGL. El resultado permitirá un mayor acercamiento al mundo de la simulación de fluidos, y contribuirá con el nivel de realismo del Simulador Quirúrgico.

Palabras Claves.

Partículas, fluidos, efecto, hidrodinámica.

Contenido

Introducción

En el mundo actual la Realidad Virtual (RV) ha alcanzado un nivel de desarrollo significativo. El alcance de esta va más allá de los juegos interactivos; por lo cual se sitúa en el exclusivo rango de herramientas para hacer. Las simulaciones por computadora así como la animación en tiempo real han tenido un auge en los últimos tiempos debido al avance tecnológico, dando como resultado simulaciones más sofisticadas que trabajen en tiempo real. Se podría mencionar algunos ejemplos como son los simuladores de conducción, tiro, de vuelo y simuladores quirúrgicos, estos últimos permiten practicar intervenciones quirúrgicas, que minimizarán errores en procedimientos de rutina o durante operaciones complejas y optimizarán el uso de los recursos técnicos y humanos desde el momento en que los médicos entran al quirófano. En el campo de la animación por computadora, las simulaciones juegan un papel importante para dar un efecto de realismo ya sea en la realización de películas, videojuegos u otro tipo de aplicaciones.

En Cuba se está incursionando en el mundo de la simulación, actualmente se han realizado algunos simuladores como por ejemplo de conducción, de tiro, de vuelo, entre otros. En la Universidad de las Ciencias Informáticas (UCI) se está desarrollando un simulador quirúrgico para operaciones de mínimo acceso, este es de gran importancia, pues se habla de un país que tiene muchos logros en el campo de la medicina y con la realización de este simulador puede ayudar a los profesionales de la salud y a los estudiantes a entrenarse antes de realizar una operación de este tipo.

Hasta ahora el simulador está en su primera etapa de desarrollo y tiene implementados algunos módulos, como son, coordinación manos-ojos y el de cámara. En la actualidad se está trabajando en el proceso de selección y persistencia de datos, pero este simulador carece de un efecto que simule los fluidos en tiempo real, efecto muy importante para su desarrollo, pues con la inexistencia de este efecto, el simulador carecería de realismo, el cual es uno de los objetivos principales de la Realidad Virtual, y es la característica principal que requiere estos tipos de simuladores, es importante que el usuario (profesional de la medicina o estudiante de medicina) sienta la realidad en el mundo virtual, esto posibilitará que este se prepare correctamente a la hora de realizar una operación y evitará cualquier tipo de accidente indeseable.

Analizando la situación existente en el proyecto, se propone como **problema científico** a resolver en este trabajo, ¿Cómo crear un efecto que permita la simulación de fluidos en tiempo real para su uso en el Simulador Quirúrgico? Este proyecto se propone como **objetivo general**, implementar un efecto que permita la simulación de fluidos en tiempo real. El **objeto de estudio** de este trabajo es el análisis del comportamiento físico de los fluidos y el **campo de acción** las técnicas y algoritmos utilizados para la simulación de fluidos en tiempo real para su utilización en el Simulador Quirúrgico. Para el cumplimiento de los objetivos planteados se trazan las siguientes **tareas** a desarrollar:

- Analizar métodos y herramientas de simulación de fluidos existentes en el mundo.
- Estudiar la dinámica y mecánica de los fluidos.
- Elaborar el análisis y diseño del efecto de simulación del fluido.
- Implementar el efecto de simulación del fluido.

Capítulo 1: Fundamentación Teórica.

Introducción.

"En función de los avances aplicados en el desarrollo de Simuladores Quirúrgicos y de los objetivos que estos persiguen, se pueden agrupar en tres generaciones tecnológicamente secuenciales (**Fig. 0.1**):

La primera generación: Formada por los Simuladores Quirúrgicos que únicamente consideran la naturaleza geométrica de la anatomía humana.

La segunda generación: Que está constituida por aquellos que, además, permiten la interacción física con las estructuras anatómicas.

La tercera generación: Que además de las características ya mencionadas de la segunda generación, tienen en cuenta la naturaleza funcional de los órganos." **[1]**

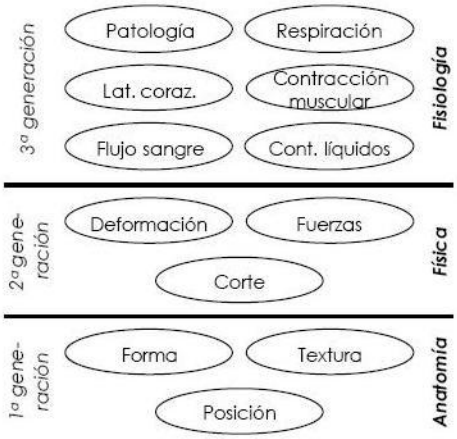


Fig. 0.1 Distribución cronológica de las diferentes generaciones de Simuladores Quirúrgicos.

Como se puede observar, lograr la optimización de la simulación de los fluidos usados en los simuladores quirúrgicos, es cosa que hoy en día se investiga con profundidad. Esta no es tarea fácil, pues se requiere de una serie de ecuaciones físicas para describir el comportamiento de los mismos, pero que sin la presencia de un efecto de este tipo cualquier Simulador Quirúrgico carecería de realismo.

Este capítulo profundizará en los diferentes métodos de simulación y visualización existentes en el mundo para la creación de los fluidos, así como también conceptos generales de las leyes físicas que rigen estos fenómenos. Para lograr esto, se ha dividido el proceso de simulación en dos partes, métodos de simulación física y métodos de visualización del efecto, con el objetivo de dar mayor claridad al texto y dejar demostrar que diferentes métodos de simulación y visualización pueden ser combinados a gusto del desarrollador.

1.1 Los fluidos. Conceptos fundamentales.

Para una mejor comprensión de este capítulo se comenzará dando varias definiciones físicas de los fluidos.

“Se conoce como fluido a todo cuerpo que carece de elasticidad de forma. Es decir no tiene una forma propia y se puede adaptar al recipiente que lo contiene. No presenta fuerzas internas tangenciales o estas son muy pequeñas. Los movimientos relativos entre partículas fluidas no realizan trabajo.” **[2]**

“Un fluido es una sustancia o medio continuo que se deforma continuamente en el tiempo ante la aplicación de una solicitación o tensión tangencial sin importar la magnitud de esta. También se puede definir un fluido como aquella sustancia que, debido a su poca cohesión intermolecular, carece de forma propia y adopta la forma del recipiente que lo contiene.” **[3]**

1.1.1 Características de los fluidos.

Los fluidos poseen 4 características fundamentales, estas son:

- La posición relativa de sus moléculas puede cambiar continuamente.
- Tienen viscosidad.
- Dependiendo de su viscosidad fluyen a mayor o menor velocidad. La viscosidad y la velocidad son inversamente proporcionales.
- Poseen una densidad bastante alta.

El movimiento de los gases y los líquidos puede estudiarse en forma aproximada mediante las ecuaciones de la dinámica de fluidos.

1.2 Antecedentes.

Esta tesis se enfocará en la simulación y visualización de un sistema dinámico compuesto por un fluido incompresible. Un sistema dinámico es un sistema complejo que presenta un cambio o evolución de su estado en un tiempo. El comportamiento en dicho estado se puede caracterizar determinando los límites del sistema, los elementos y sus reacciones; de esta forma se pueden elaborar modelos que buscan representar la estructura del mismo. Con el advenimiento de las computadoras de alta tecnología y los métodos computacionales que las acompañan, la solución de problemas de los sistemas dinámicos ha llegado a ser una realidad. A pesar de todo ese poder de cómputo, los métodos aún están lejos de ser perfectos, pero los estudios e investigaciones actuales, están logrando desarrollar mejores métodos. En **[4]**, mencionan algunos métodos relevantes para dar solución a los sistemas dinámicos.

Los primeros métodos estaban enfocados a simplificar el cómputo utilizando Síntesis de Fourier o proporcionando soluciones especializadas a problemas específicos. Por otro lado, los campos de altura junto con las ecuaciones diferenciales parciales de sombra de agua, se usaron para representar la superficie de un fluido y describir el movimiento de este, respectivamente. Más tarde, a los campos de altura se le añadió un sistema de partículas para representar el movimiento del fluido con efecto de goteo, lo cual no fue abordado en métodos previos. En ese mismo tiempo se introdujo el método "Marke-and-cell" (MAC)[I] para resolver las ecuaciones de Navier-Stokes[1], que describen el movimiento de un fluido. Este método es tridimensional y es capaz de simular un fluido cayendo y salpicando. Otro método propuesto para simular fluidos, se basa en el uso de una convección semi-Lagrangiana que permite intervalos de tiempos mucho más grandes, permitiendo estabilidad en el fluido.

En el campo del Gráfico por Computadoras existen métodos alternativos para la simulación de fluidos, los cuales han sido descritos usando simulaciones basadas en partículas[2]. Desde el surgimiento de ambos, la partícula base de Lagrange y el enfoque basado en la red Euleriana se han utilizado para la simulación de fluidos. Otro ejemplo, es el método llamado Hidrodinámica de las Partículas Suavizadas (SPH), del nombre en inglés Smoothed Particle Hydrodynamics. En un principio estuvo enfocado a simular problemas astrofísicos incluyendo colisiones galácticas y gravitacionales. El SPH ha sido

[1] *En 1822 Claude Navier (Claude Louis Marie Henri Navier 1785-1836) y en 1845 George Stokes (Sir George Gabriel Stokes 1819-1903) formularon las ecuaciones de Navier-Stokes que describen la dinámica de los fluidos.*

[2] *En 1983 T. Reeves introdujo los sistemas de partículas como una técnica para modelar una clase de objetos difusos.*

adaptado recientemente a muchos problemas de ingeniería, incluyendo transferencia de calor y de masa, dinámica molecular y mecánica de sólidos y fluidos. Este es un método Lagrangiano flexible que puede capturar fácilmente grandes deformaciones de interfaces, rotura, mezclado y salpicado. El SPH es utilizado para calcular el movimiento de las partículas que son revestidas en un campo potencial.

Aunque el método SPH es flexible, este fue creado para resolver el flujo de un fluido compresible. Se han propuesto algunas extensiones para permitir las simulaciones de fluidos incompresibles con SPH. Recientemente, fue desarrollando otro método llamado Movimiento de Partículas Semi-implícito (MPS) para resolver las ecuaciones de Navier-Stokes para fluidos incompresibles. El método MPS es capaz de simular una amplia variedad de problemas de flujo de fluidos incluyendo transiciones de fase, flujo multifase, estructuras elásticas, etc.

1.3 Introducción a la Simulación de fluidos.

1.3.1 Dinámica de los fluidos.

El flujo de los fluidos puede ser caracterizado por una serie de aspectos que gobiernan su comportamiento **[5]**:

1. Flujos Comprensibles, como opuesto de los flujos incomprensibles, ocurren cuando la variación de la presión es lo suficientemente grande como para incurrir cambios substanciales en la densidad.
2. Flujos Viscosos, como opuesto de los flujos no viscos, es usado para modelar fluidos en los cuales la fricción interna tiene un efecto significante. La cantidad de fricción es descrita por la viscosidad del fluido.
3. Flujos Estables (Flujos Laminares), como opuesto de los inestables, es usado cuando el tiempo de cambio de las propiedades del fluido es cero, es decir cuando las propiedades del fluido se mantienen constantes.
4. Flujos Turbulentos, son aparentemente caóticos y son causados por los inestables vórtices que aparecen en muchas escalas e interactúan entre ellos. Los fluidos con carencia de turbulencia son llamados laminares, aparentemente suavizados y estables.

Para que exista una mayor comprensión de este tema consideraremos otra bibliografía:

Algunas características generales del flujo de los fluidos son **[6]**:

1. El flujo de los fluidos puede ser estacionario o no estacionario. Describamos al flujo en término de los valores de variables tales como la presión, la densidad, y la velocidad del flujo en cada punto del fluido. Si estas variables son constantes en el tiempo, se dice que el flujo es estacionario. Los valores de estas variables cambian por lo general de un punto a otro, pero no cambian con el tiempo en cualquier punto en particular. A menudo puede conseguirse esta condición a velocidades de flujo bajas; una corriente que fluya continuamente es un ejemplo. En el flujo no estacionario, como una ola grande provocada por la marea, las velocidades v son funciones del tiempo. En el caso del flujo turbulento, tal como los rápidos de un río o en una caída de agua, las velocidades varían erráticamente de un punto a otro así como en el tiempo.
2. El flujo de un fluido puede ser comprensible o incomprensible. Si la densidad ρ de un fluido es constante, independiente de (x, y, z) y t, su flujo se llama flujo incomprensible. Puede considerarse usualmente que los líquidos fluyen incomprensiblemente; pero aún en un gas altamente comprensible la variación de la densidad puede ser insignificante, y para objetos prácticos podemos considerar que el flujo es incomprensible. Por ejemplo al volar a velocidades mucho menores que la velocidad del sonido en el aire (se describe como aerodinámica subsónica), el flujo del aire sobre las alas es casi incomprensible.
3. El flujo de los fluidos puede ser viscoso o no viscoso. En el movimiento de los fluidos la viscosidad es el análogo de la fricción en el movimiento de los sólidos. Cuando un fluido fluye de un modo que no disipe energía por medio de fuerzas viscosas, se dice que el fluido es no viscoso. En muchos casos, como en problemas de lubricación, la viscosidad es extremadamente importante; por ejemplo, los aceites para motor se denominan de acuerdo a su viscosidad y a su variación con la temperatura. En otros casos la viscosidad puede ser de poca importancia relativamente, y al despreciarla podemos emplear una descripción más sencilla en términos de flujo no viscoso.
4. El flujo de los fluidos puede ser rotatorio o no rotatorio. Si un elemento del fluido en movimiento no gira en torno a un eje que pase por el centro de masa del elemento, se dice que el flujo es no rotatorio. Podemos imaginar a una pequeña rueda de paletas sumergidas en el flujo en movimiento (**Fig. 1.3.1**). Si la rueda se mueve sin girar el movimiento es no rotatorio; de otro modo será rotatorio. Nótese que un elemento particular del fluido puede moverse en una trayectoria circular y experimentar también un flujo no rotatorio; una analogía es el movimiento de las góndolas colgantes de una “rueda gigante” de feria: aún cuando la rueda gire, las personas que viajan en la góndola no giran respecto a sus centros de masa. El remolino que se

forma cuando el agua fluye por el drenaje de la bañadera es un ejemplo de esta clase de flujo no rotatorio.

Fig. 1.3.1 El flujo de los fluidos puede ser rotatorio o no rotatorio.

Pequeños objetos en forma de rueda de paleta que flotan libremente en un líquido al fluir. Si la rueda gira, llamamos al flujo rotatorio; si no, el flujo es no rotatorio.

1.3.2 Dinámica Computacional de los Fluidos (CFD).

A la rama que estudia los fluidos de manera computacional se le conoce como CFD. Es la aplicación de computadoras para analizar o resolver problemas en la dinámica de los fluidos, que gracias al exponencial incremento de la potencia del hardware, se ha aumentado en experiencia y resultado. Su aplicación varía entre investigaciones físicas complejas y simulaciones en la industria de efectos especiales, tanto para juegos como para películas.

Para desarrollar la simulación, existen diferentes algoritmos, pero el más común de todos es discretizar el dominio del espacio de simulación en una malla de pequeñas celdas. Cada celda contiene el estado del fluido en la parte del dominio que le corresponde, que usualmente son estados como velocidad, densidad, temperatura y presión. Estos estados son después actualizados con un paso de tiempo discreto, con el objetivo de estudiar cómo el fluido se desarrolla al paso del tiempo. La simulación gobierna la evolución del campo de velocidad del fluido en el dominio de desarrollo. El paso de actualizar las celdas, es típicamente hecho resolviendo las ecuaciones de Navier-Stokes, que son capaces de modelar flujos incomprensibles, viscosos, inestables y turbulentos. Los datos obtenidos pueden ser visualizados o analizados dependiendo del objetivo con el cual fue desarrollado.

1.3.3 Conocimientos matemáticos previos.

Para simular el comportamiento de un fluido, se ha de tener una representación matemática de la situación del líquido en un momento dado. Lo más importante es representar la velocidad del fluido, ya que esta determina los movimientos de los fluidos en sí. La velocidad del líquido varía en el tiempo y espacio, por lo que se representa como un campo vectorial.

Un campo vectorial es un mapeo de un vector valor de la función en un espacio parametrizado, o sea se considera campos vectoriales, los que asocian un vector a cada punto en el espacio **[7]**; ejemplo: El flujo del agua en una piscina es un campo vectorial: a cada punto asociamos un vector de velocidad.

La velocidad del campo vectorial en un fluido se define, para toda posición $x = (x, y)$ tiene asociado una velocidad en el tiempo $t, u(x,t) = (u(x,t), v(x,t), w(x,t))$, como se muestra en la (**Fig. 1.3.2**).

La clave para la simulación de fluidos es, para un intervalo de tiempo y en cada intervalo de tiempo, determinar correctamente la velocidad actual. Se puede resolver utilizando un conjunto de ecuaciones que describen la evolución de la velocidad en el tiempo, bajo la acción de un conjunto de fuerzas.

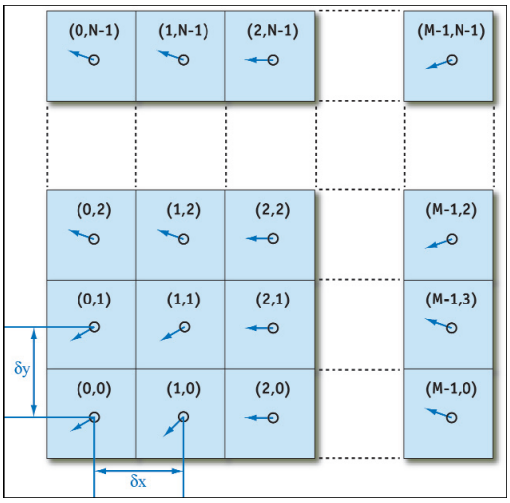


Fig. 1.3.2 Matriz Velocidad del fluido.

El estado de la simulación de fluidos está representado en una matriz M × N como la que se muestra aquí. Las flechas representan la velocidad.

1.4 Métodos de Simulación Física.

Recientemente, debido al avance en las investigaciones y el crecimiento del poder de procesamiento de las computadoras, los métodos han sido más enfocados en el comportamiento físico de los fenómenos que en los comportamientos definidos por el programador como se hacía en tiempos

pasados. A continuación se presentan una serie de métodos de simulación física que son usados a la hora de simular fluidos.

1.4.1 Ecuaciones de Navier-Stokes para un flujo incomprensible.

Como se analizó anteriormente un flujo incomprensible tiene una densidad constante (ρ) independientemente de $x, y, z, t,$ y estos pueden representarse como un campo vectorial, las ecuaciones Navier-Stokes son las ecuaciones de derivadas parciales que permiten describir el comportamiento de flujos incomprensibles a través de un campo vectorial y han sido muy usadas en el campo de CFD desde su surgimiento.

Estas ecuaciones son:

$$\frac{\partial u}{\partial t} = -(u \cdot \nabla)u - \frac{1}{\rho}\nabla p + v\nabla^2 \mathrm{u} + \mathrm{F}$$

$$\nabla \cdot u = 0$$

Ec. 1.4.1

En la primera ecuación el primer término es la derivada parcial de u con respecto al tiempo, u es el campo de velocidad del fluido, ρ es la densidad del fluido, v es el coeficiente de viscosidad y F representa cualquier fuerza externa que actúe con el fluido ($F = (fx, fy)$).

La segunda ecuación afirma que el campo de velocidad debe ser cero divergente, lo que implica que la masa debe conservarse. Las ecuaciones son usualmente definidas en un dominio computacional denominado D en el cual permanece el fluido, ya sea para un espacio 2D o 3D.

1.4.2 Términos en las ecuaciones de Navier-Stokes

Los términos que están en la parte derecha de la primera ecuación representan la advección, presión, difusión y fuerzas como se muestra en la figura (**Fig. 1.4.1**).

$$\frac{\partial \mathbf{u}}{\partial t} = \underbrace{-(\mathbf{u} \cdot \nabla)\mathbf{u}}_{\text{Advección}} \underbrace{- \frac{1}{\rho}\nabla \mathrm{p}}_{\text{Presión}} \underbrace{+ v\nabla^2 \mathbf{u}}_{\text{Difusión}} \underbrace{+ \mathbf{F}}_{\text{Fuerzas}}$$

Fig. 1.4.1 Términos en las ecuaciones de Navier-Stokes.

Estos términos determinan las principales características de los fluidos. A continuación conoceremos cada término por separado. **[7]**

Advección: No es más que el movimiento propio del fluido, la velocidad del fluido causa la transportación de objetos, densidades y otras cantidades junto con el fluido, lo cual hace que, si se derrama tinta sobre un fluido en movimiento, esta siga el movimiento del fluido. En las bibliografías existentes se le conoce como *self-advection* o *término de* advección y en las ecuaciones de Navier-Stokes es el primer término de la parte derecha de la primera ecuación.

Presión: Cuando una fuerza es aplicada al fluido, esta no se propaga instantáneamente por todo el fluido. En lugar a esto, lo que ocurre es que las moléculas cercanas al lugar donde se emitió la fuerza empujan aquellas moléculas lejanas al lugar. Debido a que la presión es la fuerza por unidades de área, cualquier presión en el fluido es una aceleración. En la primera ecuación, es el segundo término de la parte derecha.

Difusión: Es el término que determina la densidad de un fluido, algunos fluidos son más o menos densos que otros. Se dice que un fluido denso tiene una alta viscosidad, que significa en que medida es resistente el fluido al flujo. Esta resistencia es resultado de la difusión del impulso y por lo tanto, velocidad. El tercer término de la ecuación es el llamado término de difusión.

Fuerzas externas: Los cuatro términos encapsulan aceleración debido a las fuerzas externas aplicadas al fluido; pueden ser *fuerzas locales* o *fuerzas de cuerpo.* Las fuerzas locales son aquellas que actúan en una específica región. Las fuerzas de cuerpo, como la fuerza de gravedad, se aplican en el fluido entero.

Al transcurso del tiempo muchas soluciones numéricas se han planteado para solucionar estas ecuaciones, pero la mayoría de ellas requieren de mucho tiempo de procesador para solucionarla, debido a que tradicionalmente el trabajo con la CFD ha sido más enfocado a la precisión que a la eficiencia, porque su aplicación a sido más en los campos de la ingeniería y la industria que para la RV, siendo el factor gobernante la exactitud numérica y no la calidad visual.

Los modelos clásicos son inestables, lo que significa que el rendimiento de la simulación se deteriora rápidamente o diverge cuando el paso de tiempo se torna muy grande, lo que los hace imposibles de usar en aplicaciones de tiempo real, por lo que se han desarrollado otras técnicas u nuevas modificaciones a dichos métodos con tal de solucionar estos problemas como por ejemplo el Método Estable de Jos Stam.

1.4.3 Método estable de Jos Stam.

En 1999 Jos Stam introduce un nuevo método para darle solución a la ecuación de Navier-Stokes, este fue publicado en su trabajo "*Stable Fluids*", este método es usado para resolver fluidos incomprensibles y homogéneos, es decir para fluidos cuya densidad se mantienen constante en el espacio y tiempo. En este método Stam sacrifica exactitud numérica por calidad visual y rapidez. Según Stam "la forma y el comportamiento de los fluidos son de interés primario, en tanto que la exactitud física es secundaria o, en algunos casos, irrelevantes. La solución que propone, para simular fluidos, debería proporcionar a los usuarios una herramienta que permita visualizarlos como efectos en tiempo real. Esto es más importante que la estricta exactitud física, lo que exigiría demasiada potencia computacional." **[8]**

Stam simuló la Dinámica de los Fluidos en una malla cartesiana con coordenadas espaciales denotadas por x**,** que para la simulación de fluidos en dos dimensiones es $X = (x, y)$ y para simular fluidos en tres dimensiones sería $X = (x, y, z)$. Con una variable tiempo t representó el campo de velocidad como $u = (x, t)$, y el campo escalar de presión como $p = (x, t)$. Las ecuaciones son aproximadas, con el objetivo de establecer el cálculo del campo de velocidad sobre un espacio de tiempo Δt. De acuerdo con la teoría de la descomposición Helmoltz-Hodge cualquier campo vectorial w puede ser descompuesto de la siguiente forma:

$$W = u + \nabla q$$

Ec. 1.4.2

Donde u representa la conservación de la masa ($\nabla \cdot u = 0$) del campo vectorial y q representa el campo escalar. En otras palabras, cualquier campo vectorial puede ser descompuesto en la suma de sus componentes, un campo vectorial no divergente y el gradiente de un campo escalar. Un operador de proyección P que proyecte cualquier campo dentro de este campo no divergente puede ser definido como:

$$u = P_{(w)} = W - \nabla q$$

Ec. 1.4.3

Aplicando esta proyección en la ecuación original de Navier-Stokes se obtiene:

$$\frac{\partial v}{\partial t} = P(-(v \cdot \nabla)v + \mu \nabla^2 v + F)$$

Ec. 1.4.4

Donde hemos usado el hecho de que $P(v) = v$ ya que el campo de velocidades v es no divergente y $P(\nabla v) = 0$ pues ∇v es el gradiente (conservativo) de un campo escalar. La ecuación diferencial (**Ec.**

1.4.4) es la base del método de Stam. Para evaluar el campo velocidad se comienza por el estado inicial $u\ (x\ ,0)$, el procedimiento cuenta con 4 pasos para llagar a la solución en un tiempo Δt. Tenemos $W_0(x)\ =\ u(x,t)$. Los términos deben ser evaluados secuencialmente en el siguiente orden:

$$\begin{aligned} W_0(x) &\longrightarrow W_1(x) && Fuerzas\ Externas \\ W_1(x) &\longrightarrow W_2(x) && Advección \\ W_2(x) &\longrightarrow W_3(x) && Difusión \\ W_3(x) &\longrightarrow W_4(x) = u(x, t\ +\ \Delta t) && Presión \end{aligned}$$

Ec. 1.4.5

Fuerzas Externas: Las fuerzas externas tal como especifica la fuerza por unidad de masa, es simplemente integrar en el tiempo $\Delta \boldsymbol{t}$ usando el método adelantado de Euler.

$$W_1(x) = W_0(x) + \Delta t f(x,t)$$

Ec. 1.4.6

Advección: Stam propone la técnica semi-lagrangiana("*semi-lagragian*"), basada en método de las características ("*method of characteristics*")[II] usado para darle solución a las ecuaciones diferenciales parciales. En este caso se considera una partícula del fluido x en el tiempo t. En el último espacio de tiempo Δt se traslada a su ubicación actual por el campo velocidad. Para encontrar la velocidad de la partícula actual nos movemos hacia atrás en el campo velocidad desde el punto $x\ en\ \Delta t$, la traza define un camino $p(x,t)$ que corresponde a la línea del flujo parcial del campo velocidad que la partícula sigue durante el último espacio de tiempo. La nueva velocidad de la partícula en x es ajustado a la velocidad que tenía en su anterior ubicación en un tiempo atrás:

$$W_2(x) = W_1(p(x, -\Delta t))$$

Ec. 1.4.7

Una forma más intuitiva de ver esta ecuación sería ver el campo de velocidad como un conjunto de partículas centradas en cada celda de la malla. Cada partícula es trazada hacia atrás en un paso de tiempo, utilizando su velocidad actual para encontrar la posición previa que la partícula debe haber tenido para terminar en la posición actual. La velocidad en la posición previa es entonces interpolada entre las celdas vecinas para obtener la nueva velocidad actual en la posición actual. La (**Fig. 1.4.2**) muestra claramente este proceso.

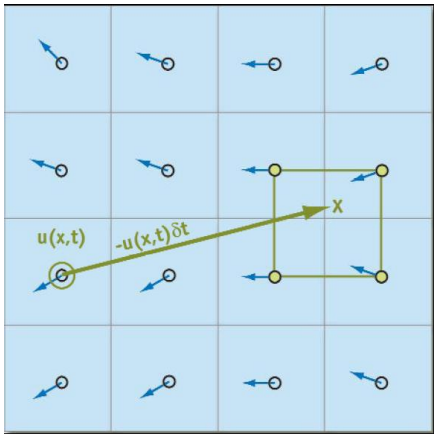


Fig. 1.4.2 Proceso de interpolación.

A pesar de ser una aproximación, este método tiene por lo menos dos claras ventajas sobre la utilización de diferenciales finitos. Y lo que es más importante sin condiciones estables. La velocidad nunca puede tomar valores excesivamente grandes ("explotar"), puesto que la velocidad máxima del nuevo campo nunca es mayor que el valor máximo del campo anterior. Esto permite utilizar medidas de tiempo tan grande como se desee, a costa de la precisión. Otra ventaja es que el algoritmo es fácil de implementar y optimizar, lo que es ideal para los algoritmos en tiempo real.

Difusión: Después de que el término de advección ha sido evaluado, el término de difusión es adicionado. La viscosidad describe la resistencia del fluido al moverse, causado por la fricción existente dentro de él. Igual que en el paso anterior, en la difusión también se aplica la misma funcionalidad:

$$(I - v\Delta t\nabla^2)W_3(x) = W_2(x)$$

Ec. 1.4.8

Donde I es la matriz de identidad. La ecuación es usada para encontrar el nuevo campo de velocidad. Esta ecuación encapsula un sistema de ecuaciones lineales que pueden ser resueltos iterativamente con los métodos de Gauss-Seidel o el método de Jacobi.

Proyección: Después que los términos fuerza, advección y difusión hayan sido adicionados, el campo de velocidad que se obtiene es divergente, por lo que la conservación de la masa no será válida y se hace una proyección multiplicando ambos miembros por la ecuación (**Ec. 1.4.2**) con un ∇ y recordando que $P\ (\nabla v) = 0$ se obtiene las siguientes ecuaciones:

$$\nabla^2 q = \nabla \cdot W_3$$
$$W_4 = W_3 - \nabla q$$

Ec. 1.4.9

La primera ecuación, es la ecuación de Poisson para un campo escalar q con la condición de frontera de Neumann. Como en el paso anterior (*Difusión*) la ecuación encapsula un sistema de ecuaciones lineales. Cuando este paso se haya completado, hemos avanzado en el campo de velocidad un intervalo de tiempo Δt.

Usando el campo de velocidades calculado, el campo de densidad escalar puede ser evaluado en la misma forma. El campo densidad puede representar el espesor del humo caliente creciente en el aire o la representación de un fluido que se mezcla con otro.

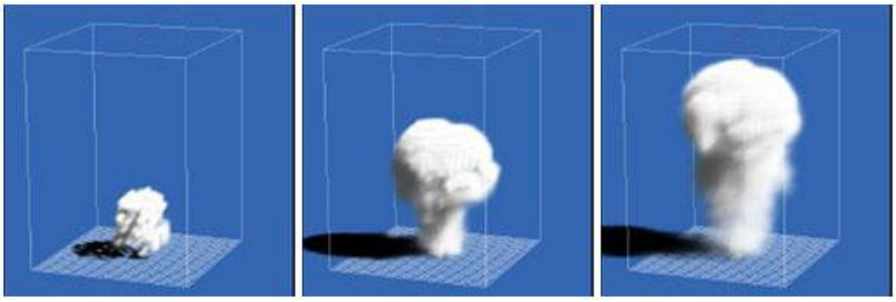


Fig. 1.4.3 Evolución de una nube creciente.

Este método estable de Stam es una importante contribución a la investigación de la dinámica de los fluidos por gráficos por computadoras. Sin embargo este método es muchas veces muy aproximado, debido a esto, en muchas ocasiones no puede ser usado en aplicaciones que requieren un comportamiento físico necesario. Ha sido utilizado en el contexto de simulación quirúrgica por Z'atonyi y otros, para visualizar la disolución de la sangre en un líquido transparente. **[8]**

1.4.4 Hidrodinámica de las partículas Suavizadas (SPH).

La Hidrodinámica de las Partículas Suavizadas, en abreviatura SPH (por su nombre en inglés) es una forma de sistemas de partículas acoplado. Fue desarrollado en 1977 para la simulación de los fenómenos astrofísicos tales como la formación de galaxias y las colisiones. Se trata básicamente de una discretización lagrangiana de las ecuaciones de Navier-Stokes con ayuda de una función peso. El procedimiento transforma las ecuaciones en derivadas parciales en ecuaciones diferenciales ordinarias que se pueden interpretar como ecuaciones de movimiento para un conjunto de partículas que interaccionan con leyes de fuerza prescritas. Es decir, la técnica permite resolver ecuaciones diferenciales parciales con códigos de Dinámica Molecular. (En realidad el método es aplicable a otras ecuaciones en derivadas parciales, no necesariamente hidrodinámicas).

Detalles del modelo SPH

El SPH es una técnica de simulación lagrangiana libre de malla en la que el fluido está representado por pseudo-partículas que interactúan entre ellas, moviéndose con el flujo y transportando en su movimiento toda la información computacional relacionada con el fluido. Las propiedades del fluido se interpolan entre las partículas. El método está basado en dos conceptos matemáticos: el interpolante integral y la aproximación de la suma de Monte-Carlo. El interpolante integral de cualquier función $f\ (r)$ se define como la siguiente integral extendida a todo el espacio:

$$\langle f(r)\rangle = \int_{\Omega} f(r')\, W(r - r', h) dr'$$

Ec. 1.4.10

La función *kernel* o núcleo, $W\ (r, h)$ debe ser una función bastante cercana a cero, para que se aproxime a una función delta de Dirac a medida que $h \to 0$ donde la longitud de suavizado *h* representa la anchura efectiva del kernel y es equivalente a la anchura de la celda del mallado en los métodos de diferencias finitas. Algunos de los kernels más usados en la literatura son el kernel Exponencial, kernel Super-Gaussiano, kernel Spline, la función de Lucy, la función de Monaghan y diversos polinomios. Uno de los primeros kernels que se utilizó fue el Gaussiano:

$$W(r, h) = \frac{1}{\sqrt{\pi h}} \exp\{-\frac{r^2}{h^2}\}$$

Ec. 1.4.11

Generalmente el kernel es esféricamente simétrico, con segunda derivada continua y soporte compacto, de tal manera que sólo contribuye a la integración el volumen dentro de una esfera de diámetro $2h.$ Además el kernel debe cumplir la condición de normalización.

$$\int W(r, h) dr = 1$$

Y

$$\lim_{h \to 0} \langle f(r)\rangle = f(r)$$

Ec. 1.4.12

Si se utiliza la aproximación de la suma de Monte Carlo, que converge a medida que el número de puntos N (o partículas en la nomenclatura SPH) tiende a infinito, y si los puntos de integración Rj están distribuidos uniformemente, se tiene:

$$\langle f(r)\rangle = \sum_{j=1}^{N} \frac{m_j}{\rho_j} f(R_j) W(r - R_j, h)$$

Ec. 1.4.13

Donde el factor $\frac{m_j}{\rho_j}$ es el elemento de volumen asociado a la partícula j, es decir, la razón entre su masa y la densidad de masa. Si se aplica la última ecuación (**Ec. 1.4.15**) al campo de densidad másica tenemos:

$$\langle \rho(r)\rangle = \sum_{j=1}^{N} m_j\, W(r - R_j, h)$$

Ec. 1.4.14

Por lo tanto la densidad de masa asociada a la partícula situada en R_i es:

$$\rho_i \equiv \langle \rho(R_i)\rangle = \sum_{j=1}^{N} m_j\, W(R_i - R_j, h)$$

Ec. 1.4.15

En esta expresión se aprecia que cada partícula de masa m_j está suavizada en el espacio de acuerdo con la función núcleo, considerada como su distribución de densidad numérica espacial. La densidad en un punto del espacio se calcula sumando para dicho punto las contribuciones de todas las partículas del sistema (dentro del rango del *kernel*). Precisamente, el nombre de la técnica SPH proviene de esta interpretación.

Además, en SPH los gradientes se calculan por diferenciación de la Ecuación (**Ec. 1.4.13**)

$$\langle \nabla f(r)\rangle = \sum_{j=1}^{N} \frac{m_j}{\rho_j} f(R_j) \nabla W(r - R_j, h)$$

Ec. 1.4.16

Para la simulación física de fluidos se necesita una ecuación de estado que se refiera a la presión y a los cambios de densidad de los fluidos. Una de las ecuaciones de estado estándar utilizado en SPH fue sugerido por Monaghan y adopta la forma:

$$P(\rho) = B\left(\left(\frac{\rho}{\rho_0}\right)^{\gamma} - 1\right)$$

Ec. 1.4.17

Donde ρ_0 es la densidad inicial de la partícula, $\gamma = 7$ y B es constante[3]. Se puede observar que una pequeña variación en la densidad producirá una gran variación en la presión. Este es el comportamiento que se quiere cuando se simulan fluidos incomprensibles como agua o sangre. Con el fin de utilizar esta ecuación de la densidad tiene que ser evaluado en el centro de cada partícula. Esto puede llevarse a cabo mediante la siguiente ecuación:

$$\rho(r) = \sum_j m_j W(r - r_j, h)$$

Ec. 1.4.18

Pero esto trae algunos inconvenientes. En la práctica se utiliza otro método basado en el cálculo de la tasa de cambio de densidades para cada centro de cada una de las partículas utilizando la ecuación de continuidad. Esta tasa de cambio de la densidad puede ser integrada en el tiempo para cada partícula usando algún estándar como Runge-Kutta o la integración por salto de rana (Leap-Forg Integration)[III].

En fluidos no viscosos, la fuerza en cada elemento del fluido es proporcional al gradiente de la presión local. Para obtener una ecuación sobre la base de presiones se evalúa en la ecuación de estado sugerida por Monaghan obtenida anteriormente, por lo tanto, puede ser derivada para actualizar la velocidad y la ubicación de cada partícula del fluido. A menudo se añade al final de la ecuación de movimiento fuerzas adicionales, términos como la viscosidad y la gravedad artificial.

La popularidad de la SPH en los recientes años es el resultado de una serie de investigaciones con respecto a las propiedades de estabilidad de esta.

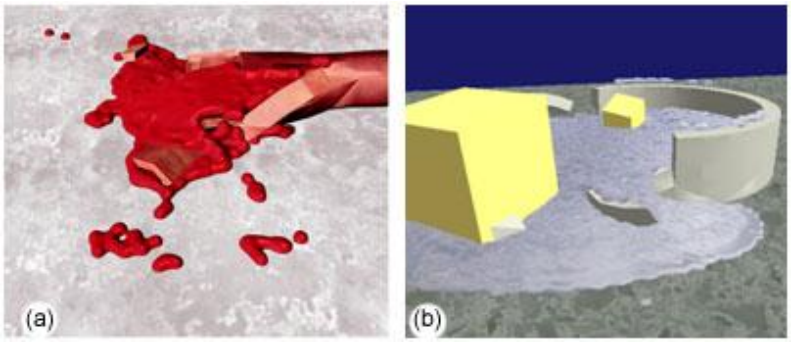
La mayoría de las ventajas que tiene el SPH son es un método Lagrangiano (basado en partículas). Por ejemplo, es fácil de definir en una forma arbitraria de fronteras y hay pocas restricciones en la forma el fluido se puede deformar o separada en piezas disjuntas. Trabajos recientes en este tema incluye una adaptación para un mejor rendimiento en tiempo real desarrollado por Müller. Este usa núcleos suavizados especiales para mejorar la estabilidad a fin de permitir mayor medidas de tiempo y para aumentar el rendimiento.

La SPH tiene una gran aplicación en los simuladores quirúrgicos pues se han desarrollados simulaciones en tiempo real de la sangre fluyendo a través de los vasos sanguíneos y salpicaduras de

[3] *Términos introducidos por J.J. Monaghan en Simulating Free Surface Flows with SPH. s.l. : Journal of Computational Physics, 1994:110:399–406.*

la misma. Esta ha sido una muy breve introducción a las ecuaciones fundamentales de SPH para más detalles puede ver **[9]**.

Un ejemplo de esto es el trabajo de Müller del cual mostramos las siguientes figuras (**Fig. 1.4.4**) **[9]**



(a) Sangre en un vaso sanguíneo cortado.

(b) Fuga de agua.

Fig. 1.4.4 (a) y (b) Imágenes del trabajo de Möller.

1.5 Métodos de Simulación Visual.

1.5.1 Los Sistemas de Partículas.

Desde que se presentó oficialmente los gráficos por computadora a la comunidad por Reeves en 1983, los sistemas de partículas han sido un instrumento importante para efectos en tiempo real. Los sistemas de partículas pueden ser tanto acopladas como no vinculadas. Los sistemas de partículas no vinculadas, cada partícula se mueve independientemente, usualmente acorde las leyes de Sir Isaac Newton. En los sistemas de partículas acopladas, las partículas interactúan entre sí, con el fin de simular un comportamiento más realista. Otra característica común, pero no necesaria, es que a menudo suelen utilizar atributos definidos, como por ejemplo, la velocidad inicial o el color de la partícula.

Los sistemas de partículas han sido usados tanto para describir técnicas de visualizado como para describir tipos de animaciones específicas. Debido a esto, su definición depende del tipo de aplicación que se va a usar. El criterio general de definición para cualquier tipo de aplicación es el siguiente **[10]**:

1. **Es una colección de partículas:** El sistema está compuesto por una o más partículas individuales, donde cada una de ellas tienen un conjunto de atributos que de manera directa o

indirectamente afectan el comportamiento de la partícula. A menudo las partículas son primitivas gráficas como puntos o líneas, pero en realidad no existe límite para esto.

2. **Define atributos o propiedades estocásticas:** Otra característica en común que tienen los tipos de sistemas de partículas es que introducen en ellas algún tipo de elementos o variables aleatorias, los cuales pueden ser usados para controlar la variación aleatoria de la velocidad, o el cambio de color, incluso su posición. Usualmente estos elementos aleatorios son controlados por algún tipo de límite estocástico predefinido, como límites o tipos de distribución.
3. **Tienen un ciclo de vida:** Cada partícula tiene tres estados distintos en su tiempo de vida, Nacimiento, en el cual cada partícula en el sistema es generada de manera aleatoria, así como también puede ser la generación de su forma. Vida, en el cual los atributos de las partículas cambian al pasar del tiempo, estos atributos pueden depender del tiempo, de otro atributo o de ambos. Y Muerte, que es cuando el tiempo de vida de la partícula llegó a su fin o cuando está fuera de los límites o cuando, por ejemplo, el color de la partícula se torna negro, lo que hace que no sea visible.
4. **Forma de las partículas:** Las partículas pueden tener cualquier tipo de forma, forma esférica, forma de caja, punto, pero generalmente tienen forma de un plano como se muestra en la (**Fig. 1.5.1**).

Este método puede resultar tener un amplio código fuente, por eso es importante diseñar bien la estructura de datos que representará nuestra partícula para mejorar el rendimiento del sistema. En el procesamiento de miles de partículas, un milisegundo de más, puede agrandar el tiempo de ejecución, por eso, cuando se diseña un sistema de partículas lo primero que se debe tener en cuenta es que los sistemas de partículas incrementan grandemente el número de polígonos visibles. La mayoría de los sistemas de partículas usan cuatro vértices y dos triángulos para representar una partícula (**Fig. 1.5.1**). Esto conlleva a que un sistema que use más de 2000 partículas incrementa la cantidad de polígonos en la escena a más de 4000 que hay que visualizar y recalcular los datos dentro del buffer en cada “frame”. Una solución para optimizar un sistema de partículas, es utilizar lo menos posible operaciones de memoria, como asignación y eliminación de memoria a objetos. Cuando una partícula muere, no eliminarla de memoria sino solamente marcarla como “muerta”, así cuando todas las partículas del sistema mueran entonces se liberan todas juntas **[10]**. Esto permite reutilizar partículas muertas, es decir memorias asignadas, como por ejemplo en la simulación de flujos constantes como el fuego, puedes revivir partículas ya muertas tan sólo inicializando sus parámetros.

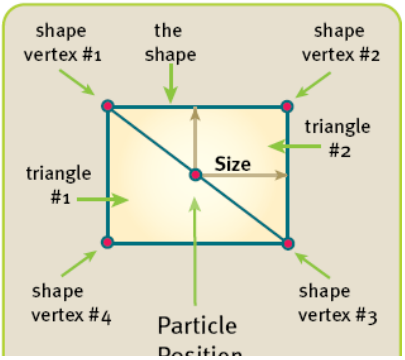
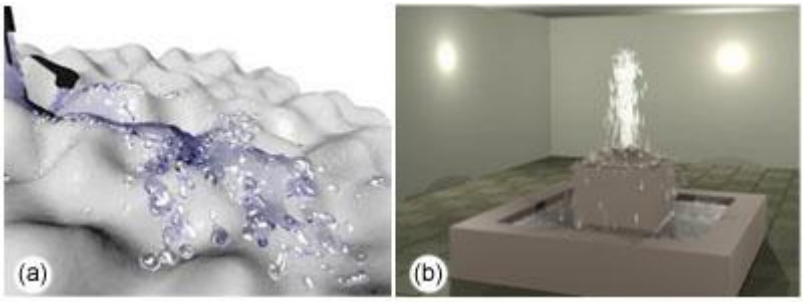


Fig. 1.5.1 Forma de la partícula.

Otro aprovechamiento que permite aumentar el rendimiento de los sistemas de partículas es el uso de los niveles de detalles (LODs “Levels Of Detail”). Este método está basado en una subdivisión física que genera una jerarquía aproximada de modelos en movimientos o niveles de detalles de la simulación (SLODs). En cada paso de tiempo se actualiza los SLODs y es seleccionando el más apropiado para reducir el costo computacional. La idea general es que se sustituye un grupo de partículas por una que represente toda el área abarcada por el grupo en dependencia de lo lejos que sea la simulación de acuerdo con la cámara. Mientras más lejos esté, menos nivel de detalle se necesita, por tanto menos partículas. Mientras más cerca esté, mayor nivel de detalle, y por tanto mayor número de partículas.

1.5.2 Sistemas de Partículas no Vinculadas.

Los sistemas de partículas no vinculadas han sido usados en juegos de computadoras desde hace ya varios años. Unos de los primeros ejemplos es el clásico juego de la Guerra de las Galaxias, desarrollado en un PDP-1 en 1961, inicialmente mediante un osciloscopio convertido como equipo de visualización; aunque en realidad no se basa estrictamente en ningún principio físico. Los sistemas de partículas no vinculadas siguen siendo utilizados con frecuencia en los juegos de ordenador para visualizar desde la sangre, hasta la caída de la lluvia y el humo de las explosiones. Gracias a los simples principios utilizados para actualizar cada partícula, un gran número de partículas se puede utilizar, lo que ayuda a que los efectos sean visualmente más convincentes. Sin embargo, dado que no existe ninguna conexión entre las partículas, los sistemas de partículas no vinculadas no son ideales para la simulación de las sustancias en las que las fuerzas internas tengan un notable efecto en su comportamiento. Sin embargo a pesar de esto, los sistemas de partículas no vinculadas, con frecuencia, han sido utilizados para modelar este tipo de sustancias con buenos resultados visuales (**Fig. 1.5.2**).



(a) Partículas de fluido que fluye sobre una superficie desigual con detección de colisiones y de la fricción.

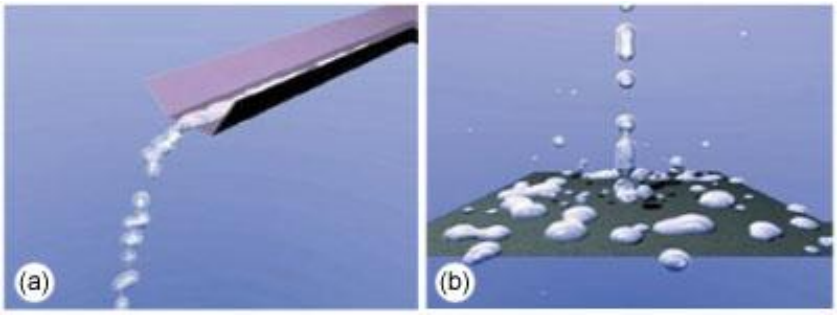
(b) Fuente de agua. Para subrayar su movimiento y dirección, las partículas se extendieron a lo largo de su vector de velocidad.

Fig. 1.5.2 (a) y (b) Imágenes de Sistemas de Partículas no Vinculadas.

1.5.3 Sistemas de Partículas acopladas.

El sistema de acoplamiento permite que las partículas interactúen entre sí por lo que genera muchas posibilidades, este hace que el sistema de partículas se comporten como un todo más realista. Por ejemplo, una manera simple de hacer que el sistema actúe como un flujo continuo sería dejar que las partículas se afecten entre sí por algún tipo de potencial Lennard-Jones, permitiendo a las partículas rechazarse unas con otras cuando ellas estén muy cerca, atraerse cuando están a una distancia media y hacer la fuerza cero a medida que la distancia vaya aumentando. Este modelo simple puede ampliarse mediante la incorporación de funciones de emulación de efectos tales como la fricción interna y mecánica de amortiguación. Este enfoque fue adoptado por Murta y Miller en [11] para simular el movimiento de goteo y salpicaduras de líquidos. Utilizan la interacción Lennard-Jones, como se ha descrito anteriormente y un simple esquema de Euler con pequeñas medidas tiempo para integrar las aceleraciones de las partículas resultantes. Para permitir la interacción entre los fluidos y objetos estáticos, la intersección rayo polígono (“ray-polygon”) se realiza para cada partícula a medida que este avanza.

Si la ruta de una partícula intercepta una superficie a gran velocidad, la velocidad de la partícula a lo largo de la superficie se invierte. Además se seleccionará al azar una desviación que será añadida al vector velocidad para fomentar un comportamiento de salpicaduras más natural. La partícula que alcance una superficie de baja velocidad no es probable que posea suficiente energía cinética para superar las fuerzas de adherencias generada. Sus velocidades son ajustadas lo largo de la superficie. Una fuerza adicional se agrega para simular la fricción entre el líquido y la superficie (**Fig. 1.5.3**).



(a) El agua que corren hacia abajo en una barra de metal, irrumpiendo en piezas separadas, ya que cae a través del aire.

(b) Otro ejemplo. Salpicaduras de líquidos y la separación de gotas, ya que golpea una superficie.

Fig. 1.5.3 (a) y (b) Imágenes del trabajo de Murta y Miller [11].

1.5.4 Representación Volumétrica (Volumen Rending).

La representación volumétrica es un método para visualizar imágenes de volúmenes (datos 3D)de manera completa sin uso de una segmentación este método tiene la ventaja de que toma en cuenta la información de los datos en su totalidad y de esta manera logra una representación 3D muy completa. La información que generalmente se requiere son imágenes de cortes transversales de algún volumen. Cada corte transversal está formado por un arreglo bidimensional de valores de intensidad. El volumen se forma agrupando las imágenes, es decir considerando que cada punto de la imagen tiene coordenadas (x, y), si se agrega la coordenada z, por medio del número de corte se obtiene un conjunto de coordenadas tridimensionales confinadas en un paralelepípedo. Cada coordenada se representa por un vóxel, el cual es un cubo cuyo ancho, alto y espesor, tiene tamaño de un píxel. El valor del vóxel que se encuentra en el punto (x, y, z) tendrá el valor del punto (x, y) en el corte z. Esta técnica de representación tridimensional toma todo el volumen de datos y suma la contribución de cada vóxel a lo largo de una línea o "rayos" (ray en términos anglosajones) desde el ojo del observador a través del volumen de datos y representa la composición resultante para cada píxel de la pantalla. La incorporación de información de todo el volumen de datos supone una mayor fidelidad a estos datos; sin embargo, para manejar estos volúmenes de información son necesarios procesadores muy potentes. El valor del píxel se obtiene considerando los valores de los "rayos" como la representación de una variación de opacidades. (Se define la opacidad en este caso como el grado en el cual la luz no puede penetrar en un objeto). Distintos valores de opacidad se asignan a los diferentes valores del vóxel, lo cual puede representar las diferentes propiedades de los tejidos, como por ejemplo la densidad. El efecto es reproducir los objetos de alta opacidad más claramente visibles sobre los objetos menos opacos, los cuales aparecen transparentes en mayor o menor grado. El resultado es

la posibilidad de ver diferentes tipos de tejidos, en vez de ver sólo el primer tejido; la representación volumétrica permite ver simultáneamente objetos con diferentes propiedades (**Fig. 1.5.4**).

Opacidad 0: Se asigna a los vóxeles[IV] transparentes, por lo que no se verán en la imagen.

Opacidad 1: Se asigna a los vóxeles totalmente opacos que no transmiten luz pero la reflejan totalmente y obtienen así una apariencia sólida.

Opacidad intermedia: Los vóxeles se muestran de forma semitransparente. La representación volumétrica se puede obtener en tres formas: sombreado en blanco y negro, sombreado a color y sombreado a color de múltiples objetos.

Sombreado en blanco y negro: El valor de sombreado de un vóxel se define por su opacidad. El resultado final es un amplio porcentaje de valores de vóxeles en cada rayo. La contribución más significativa viene dada porque lo vóxeles localizados cerca de la superficie donde las propiedades del tejido son las mismas. Los bordes son visibles como una línea oscura debido a que el rayo atraviesa un gran número de vóxeles. Es útil para estudios de vasculares y óseos.

Sombreado a color: El valor de sombrado de un vóxel se define por su opacidad y la orientación local de la superficie definido su localización. El color se basa en el valor del vóxel.

Sombreado a color de múltiples objetos: Al igual que el sombreado a color el valor del sombreado para el vóxel se define por su opacidad y la orientación local de la superficie por la localización del vóxel. El color se basa en el color asignado a cada objeto u objetos en caso de selección múltiple.

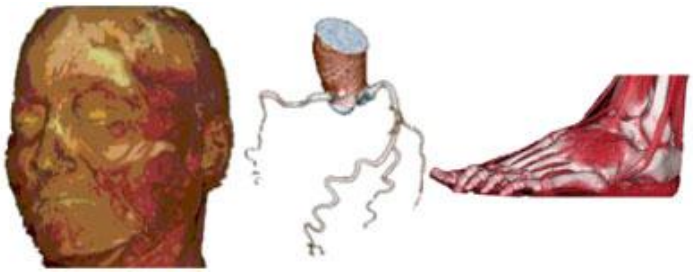


Fig. 1.5.4 Ejemplos de representaciones volumétricas.

1.5.5 Metaball (Metabolas).

Metaball es el nombre de una técnica de gráficos realizada por ordenador para simular interacción orgánica**,** (pertenecientes a seres humanos y animales) entre diferentes objetos n-dimensionales

(como gotas de mercurio mezclándose por su superficie) y fue introducido por Jim Blinn a principios de los años 1980, se basan en el concepto de superficie implícita. **[12]**.

Una superficie implícita es una superficie dada como los ceros de una función $f(x, y, z)$ de tres variables. Por ejemplo, la esfera admite una definición como superficie implícita:

$$x^2 + y^2 + z^2 - 1 = 0.$$

Ec. 1.5.1

Hacía tiempo se conocía el hecho de que cuando tienes dos funciones implícitas $f(x, y, z) = 0$ y $g(x, y, z) = 0$, que son continuas, la superficie que define $f + g = 0$ es una combinación de las dos superficies. En la (**Fig. 1.5.5**) tenemos ejemplos de Metaball con distintos cargas (aquí la carga ha de interpretarse como la fuerza de atracción, la cual determina la forma de la isosuperficie[V]):

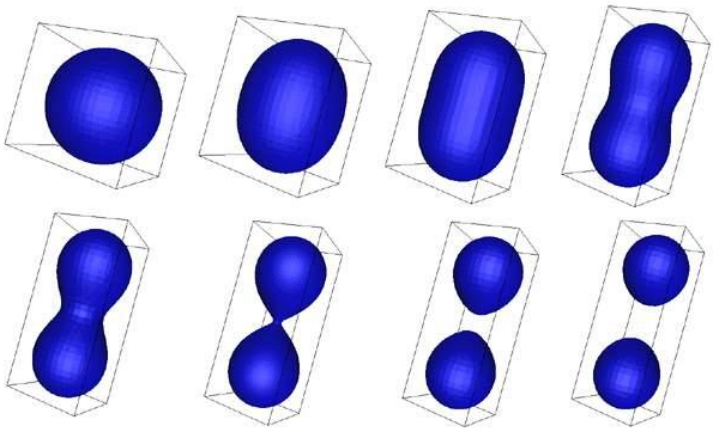


Fig. 1.5.5 Ejemplos de Metaball con distintas cargas.

En general, la forma de las Metaball depende de la distancia; en la práctica se toma el cuadrado de la distancia, que evita los problemas numéricos.

Los algoritmos para generar las Metaball son lentos. Al observar la ecuación de la isosuperficie no se puede sacar, en general, una representación paramétrica que permitiría reconstruir por muestreo la isosuperficie con la resolución deseada, con lo único que se dispone es con la ecuación implícita de la isosuperficie y ésta sólo permite contestar a la pregunta de si un determinado punto está dentro, fuera o sobre la isosuperficie. Se hace necesario implementar un algoritmo de reconstrucción que use árboles octales adaptativos para obtener la isosuperficie. Hay dos algoritmos basados en esta idea, el

"*marching cubes*" [VI] y el "*marching tetrahedrons*" [VII]. El segundo es una mejora del primero, el cual tiene una complejidad muy alta.

Las Metaball dan problemas a la hora de definir las normales porque producen errores pues sus derivadas cambian muy bruscamente. Esto trae como consecuencia que el asignar texturas sea difícil. Hay que hacer un tratamiento previo para suavizar las normales antes de aplicar las texturas.

Fig. 1.5.6 Ejemplo de simulación de la sangre usando Metaball.

1.6 Condiciones Fronteras

Para obtener resultados reales en la simulación de un fluido se tiene que establecer en primer lugar un espacio de trabajo. Estableciendo dicho espacio, se definen las fronteras en las que se va a simular el fluido, en este caso, la sangre. El espacio de trabajo del fluido se puede modelar utilizando dos técnicas diferentes [**13**].

- Partículas de Frontera.
- Superficies de Frontera.

La elección de la técnica para modelar las condiciones de frontera está basada en la complejidad del espacio de trabajo, es decir, qué resulta más fácil y práctico utilizar para su construcción. En ocasiones, es más rápido realizar la implementación del espacio u obstáculos utilizando las partículas de frontera, que implementarlas con las superficies de frontera, pero el precio a pagar es el aumento de partículas dentro del sistema dinámico, provocando el incremento del tiempo de cómputo en el cálculo de las características de cada una de ellas.

1.6.1 Partículas de Frontera

Las partículas de frontera se utilizan mucho en las simulaciones de fluidos para delimitar el área de trabajo de este, debido a que su introducción dentro de la simulación del fluido es más fácil y rápida. La diferencia entre las partículas del fluido y las partículas de frontera radica principalmente en la actualización de sus características, o sea las partículas fronteras tendrían características específicas, no tendrán aceleración ni velocidad y la posición es fija durante la simulación del fluido (**Fig. 1.6.1**). Dichas partículas tendrían además características variables que sería necesario ir modificando a medida que avanzara la simulación como son la densidad y la presión ya que estas son las que van a repeler a las partículas del fluido.

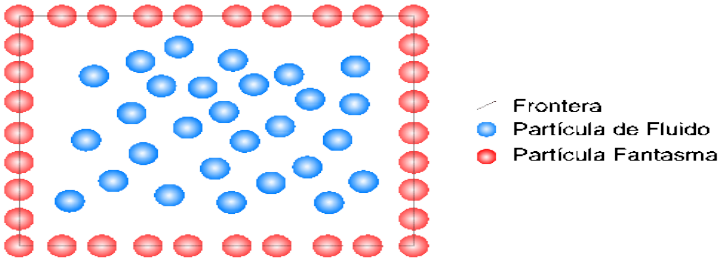


Fig. 1.6.1 Ejemplo de sistema utilizando partículas de fronteras.

Al implementar las partículas de frontera en el sistema dinámico se tiene la desventaja de que el número de partículas dentro del sistema se eleva dependiendo de que tan complejo sea el espacio de trabajo. Al elevar el número de partículas en el sistema dinámico implica más tiempo de cómputo en el cálculo de las características de las partículas haciendo que la velocidad de la simulación disminuya, alejándose del tiempo real.

1.6.2 Superficies de Frontera

Las superficies de frontera son una mejor alternativa a las partículas de frontera para reducir el tiempo de simulación. Estas son superficies compuestas por formas geométricas planas (en el espacio bidimensional) o por poliedros (en espacio tridimensional) como se representa en la (**Fig. 1.6.2**) Para el caso del espacio tridimensional, los poliedros se pueden construir utilizando cualquiera de los siguientes métodos: mallas de triángulos o mallas simples. A diferencia de las partículas de frontera, las superficies de frontera no aumentan el número total de partículas del fluido, ya que se utilizan los modelos matemáticos que describen la superficie de las figuras geométricas. Por otro lado, las superficies cuentan con un solo atributo, su posición, la cual no se actualiza, ya que para cuerpos no deformables la superficie se considera fija en todo momento de la simulación del fluido.

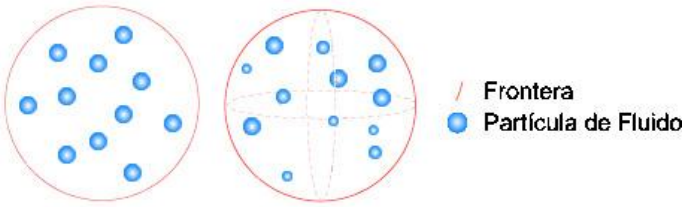


Fig. 1.6.2 Ejemplo superficies fronteras bidimensionales y tridimensionales.

La forma de trabajar con las superficies de frontera es totalmente diferente de las partículas de frontera, ya que las superficies introducen una fuerza externa con la superficie en la partícula del fluido, capaz de mantener a dicha partícula dentro del espacio de trabajo. Esta fuerza debe tener una magnitud igual o mayor que la fuerza interna de la partícula, pero en dirección contraria (**Fig. 1.6.3**). La forma en que se sabe cuando se tiene que aplicar dicha fuerza en la partícula, es detectando el choque entre la partícula del fluido y la superficie de frontera; para esto se utilizan los detectores de colisiones.

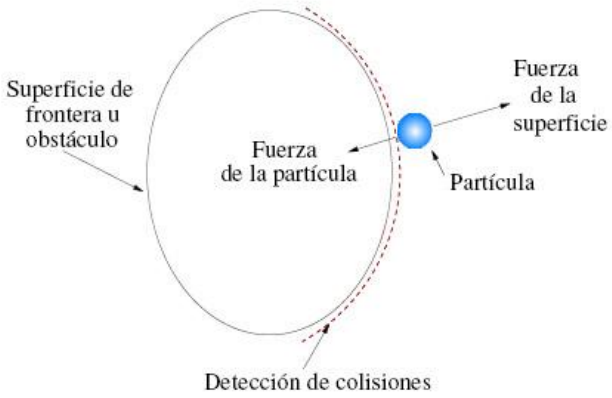


Fig. 1.6.3 Detección de colisiones y fuerza externa aplicada a la partícula.

Capítulo 2: Descripción de la Solución Propuesta.

Introducción

En el presente capítulo se propone soluciones técnicas para crear fluidos, partiendo de la dinámica de los fluidos en la parte física y del sistema de partículas para la visualización.

2.1 Método de Visualización.

Como solución se propone el uso de un sistema de partículas como método de visualización para la simulación de los fluidos. Un sistema de partículas capaz de asimilar cualquier modelo matemático que responda al comportamiento de cada partícula y que defina un efecto en sí. Se ha escogido este método de visualización porque en comparación con los planteados en el capítulo anterior, tiene mejores características para el trabajo en tiempo real, que es uno de los objetivos principales de la investigación, además de su fácil uso y manejo. El sistema estará compuesto por:

1. Una estructura partícula: que describirá aquellos atributos necesarios en cada partícula. Además será capaz de soportar tantas definiciones de estructuras de partículas como el desarrollador quiera, dando vía a la extensibilidad del sistema.

2. Métodos de inicialización: que se encargarán de inicializar los atributos de las partículas.

3. Modelos matemáticos: que empaquetarán la física que rige el movimiento de cada partícula.

4. Será capaz de hacer un control centralizado del conjunto de partículas independientemente del modelo que se haya definido.

2.2 Simulación Física.

Para la simulación física de sangre se utilizará el método Hidrodinámica de las Partículas Suavizadas (SPH) mencionado en el capitulo anterior, ya que este método como es libre de malla permite el movimiento libre del fluido en una superficie. Esta es la ventaja frente a las técnicas numéricas convencionales, evitar la conectividad rígida que exigen estas, sustituyendo los elementos que caracterizan las mallas por subdominios asociados a cada uno de los puntos en que se discretiza el dominio.

El método SPH presenta una serie de ventajas y desventajas que lo hacen especialmente idóneo para el análisis de ciertos fenómenos y totalmente desaconsejable para otros. Entre las ventajas se puede destacar:

- Las principales ventajas de esta técnica se derivan de su naturaleza lagrangiana. SPH sigue automáticamente flujos complejos y puede fácilmente mantener constante la resolución de la masa. Se pueden mejorar los resultados si se utiliza una longitud suavizada *h* que tenga una variación espacial, manteniendo un número constante de vecinos dentro de un radio aproximadamente igual a $2h$.
- Se manejan fácilmente los problemas multidimensionales así como los que no tienen ninguna simetría.
- El método sólo calcula donde la masa está localizada, por tanto, no se gastará tiempo en el tratamiento de los espacios vacíos.
- Es fácil incluir cualquier proceso físico adicional en el código base.
- El método produce buenos resultados para un gran número de problemas de distinta naturaleza.

Siendo las desventajas más importantes:

- Las condiciones de contorno son normalmente difíciles de implementar. Uno de los problemas que puede aparecer debido a un mal tratamiento de los contornos es la penetración de las partículas fluidas en los contornos, lo cual debe evitarse. Los tipos de contornos fácilmente manejables con el SPH son:

  1) Contornos naturales (donde la densidad cae a cero) lejos de las regiones de interés. Entonces, los contornos pueden ser ignorados.

2) Condiciones de tipo periódico por ejemplo, en las cajas (donde se puede conseguir que las partículas “vean” los vecinos en la cara opuesta de la caja).

- SPH trabaja mejor si el ancho del kernel es variable, así la resolución será mejor en las regiones de alta densidad. Las regiones de baja densidad estarían peor resueltas.

2.2.1 SPH enfocado a fluidos.

Para aplicar el modelo matemático del SPH a mecánica de fluidos con superficie libre, se partirá de las ecuaciones de Navier-Stokes, escribiéndolas para una partícula sin considerar la viscosidad ni las fuerzas externas sobre esta.

En primer lugar se tiene la ecuación de continuidad, la cual hace referencia a la masa de la partícula, donde esta no se crea ni se destruye durante el proceso, solamente se conserva.

$$\frac{d}{dt}\rho = -\rho \cdot \nabla v$$

Ec. 2.2.1

Por otro lado, la aceleración de la partícula está dada por:

$$\frac{d}{dt}v = \frac{1}{\rho} \cdot \nabla p$$

Ec. 2.2.2

Haciendo la relación entre la presión y la densidad de la partícula, se tiene:

$$\nabla\left(\frac{\mathrm{p}}{\rho}\right) = \frac{\nabla \mathrm{p}}{\rho} - \frac{\mathrm{p}}{\rho^2}\nabla\rho$$

Ec. 2.2.3

Por lo tanto, la ecuación (**Ec. 2.2.2**) queda:

$$\frac{d}{dt}v = -\nabla\left(\frac{\mathrm{p}}{\rho}\right) - \frac{\mathrm{p}}{\rho^2}\nabla\rho$$

Ec. 2.2.4

Al aplicar la función de interpolación que utiliza el método SPH (Suma Monte Carlos) a la expresión (**Ec. 2.2.4**), se obtiene la ecuación de movimiento para una partícula.

$$\frac{d}{dt}v = -\sum_j m_j \left(\frac{p_i}{{\rho_i}^2} + \frac{p_j}{{\rho_j}^2}\right) \cdot \nabla_i W_{ij}$$

Ec. 2.2.5

La ecuación (**Ec. 2.2.5**) da como resultado la aceleración de la partícula $\boldsymbol{i}$ teniendo en cuenta a las partículas $\boldsymbol{j}$ del sistema. Por otro lado, para calcular la densidad de la partícula $\boldsymbol{i-ésima}$, se obtiene de la ecuación (Suma Monte Carlos):

$$\rho_i = \sum_j m_j \cdot W_{ij}$$

Ec. 2.2.6

Pero esta aproximación arroja resultados erróneos al momento en que las partículas se aproximan a la frontera. Este error ocurre en el modelado de los fluidos incompresibles (como la sangre), ya que la densidad de esta tiende a cero cuando se acerca a la superficie (frontera) y la ecuación (**Ec. 2.2.6**) suaviza la densidad en la longitud $2h$, lo que representa que las partículas que se aproximan a la superficie tendrán una densidad incorrecta. Además de provocar este error, también afecta a la ecuación de estado (de la cual se hablará más adelante), que introducirá presiones incorrectas. Para solucionar esto, se propone la ecuación de continuidad (Gingold y Monaghan) a partir de la expresión (**Ec. 2.2.1**):

$$\frac{d}{dt}\rho_i = \sum_j m_j \, v_{ij} \cdot \nabla_i W_{ij}$$

Ec. 2.2.7

Donde v_{ij} es la diferencia entre las velocidades de las partículas i y j, es decir, $v_{ij} = v_i - v_j$. Para poder usar la ecuación (**Ec. 2.2.7**), se tiene que tener en cuenta la inicialización de la densidad para cada partícula del sistema al inicio de la simulación. Para completar la ecuación de la aceleración de la partícula (**Ec. 2.2.5**), es necesario incluir y explicar dos términos importantes, el primero es la función de aproximación kernel (W_{ij}) y el segundo es la viscosidad artificial.

2.2.2 Kernel.

La función de aproximación tipo kernel, tendrá las siguientes propiedades:

1. Ser simétrica cuando r = 0 y la función Delta de Dirac disminuya cuando el límite h se aproxima a cero

$$\lim_{h\to 0} W(|r|,h) = \delta(|r|)$$

Ec. 2.2.8

2. La normal de la función debe ser igual a uno

$$\int W(|r|,h)dr = 1$$

Ec. 2.2.9

3. El kernel se hace cero para los vecinos que tienen una distancia de más de $2h$, $|r| > 2h$

$$W\big((|r| > 2h), h\big) = 0$$

Ec. 2.2.10

lo cual indica que en la búsqueda de los vecinos se debe tomar en cuenta que la distancia entre la partícula i y la partícula j sea a lo más $2h$ para que estas se consideren vecinas.

En el trabajo se utilizara el kernel más utilizado en la literatura es el B-Spline, propuesto por Monaghan y Lattanzio, el cual se define como:

$$W(r,h) \triangleq Cf(x) = \begin{cases} 1 - \frac{3}{2}q^2 + \frac{3}{4}q^3, & si\ 0 \le q < 1 \\ \frac{1}{4}(2-q)^3, & si\ 1\ \le q < 2 \\ 0 & de\ lo\ contrario \end{cases}$$

Ec. 2.2.11

Donde

$$q = \frac{r_{i,j}}{h}\ y\ r_{i,j} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2}$$

Ec. 2.2.12

La variable C es un escalar cuyo valor depende de la dimensión en que se esté trabajando. Este valor para el caso tridimensional es $\frac{1}{\pi h^3}$.

A continuación se muestra la representación de forma gráfica de la función.

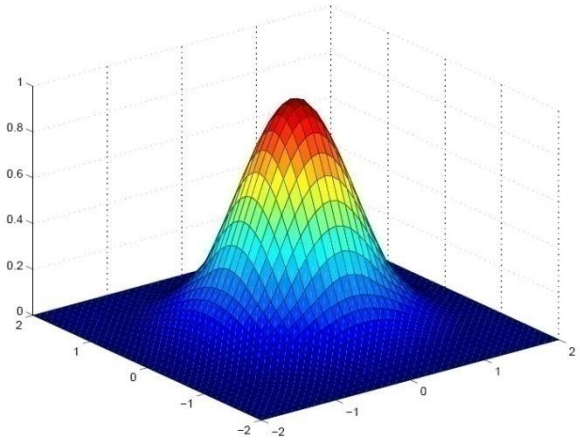


Fig. 2.2.1 Gráfica de la función del kernel en 3 dimensiones.

Como se puede observar en la (**Fig. 2.2.1**), el área donde existe interacción entre las partículas está comprendida entre −2 y 2 cuando la función está centrada en el origen. Es decir, cuando la distancia entre la partícula i (que se está evaluando) y la partícula j es mayor a $2h$, no hay ninguna interacción entre ellas. En caso contrario, cuando la distancia entre ellas es menor de $2h$, entonces, existe interacción y si esta distancia es muy pequeña, la interacción es más grande y fuerte.

Como se trabajará con el gradiente del kernel es necesario obtener las derivadas parciales de la función kernel. Las derivadas parciales se efectúan con respecto a las coordenadas (x, y, z) que maneja la distancia entre las partículas (ri, j).Para obtenerlas, aplicamos la regla de la cadena empezando por ri, j , por lo tanto el gradiente del kernel en función de sus derivadas parciales quedará de la siguiente manera:

$$\nabla_i W(r,h), cuando\ 0 \leq q < 1 = \begin{cases} \dfrac{\partial}{\partial x_i} W(r,h) = -\dfrac{C}{h^2}(3 - \dfrac{9}{4}q)\Delta x_{i,j} \\ \dfrac{\partial}{\partial y_i} W(r,h) = -\dfrac{C}{h^2}(3 - \dfrac{9}{4}q)\Delta y_{i,j} \\ \dfrac{\partial}{\partial z_i} W(r,h) = -\dfrac{C}{h^2}\left(3 - \dfrac{9}{4}q\right)\Delta z_{i,j} \end{cases}$$

$$\nabla_i W(r,h), cuando\ 1 \le q < 2 = \begin{cases} \frac{\partial}{\partial x_i} W(r,h) = -\frac{3}{4}\frac{C}{h \cdot r_{i,j}}(2-q)^2 \Delta x_{i,j} \\ \frac{\partial}{\partial y_i} W(r,h) = -\frac{3}{4}\frac{C}{h \cdot r_{i,j}}(2-q)^2 \Delta y_{i,j} \\ \frac{\partial}{\partial z_i} W(r,h) = -\frac{3}{4}\frac{C}{h \cdot r_{i,j}}(2-q)^2 \Delta y_{i,j} \end{cases}$$

$$\nabla_i W(r,h), cuando\ q \ge 2 = \begin{cases} \frac{\partial}{\partial x_i} W(r,h) = 0 \\ \frac{\partial}{\partial y_i} W(r,h) = 0 \\ \frac{\partial}{\partial z_i} W(r,h) = 0 \end{cases}$$

Ec. 2.2.13

Estas ecuaciones se dividen en los tres casos que plantea la función de interpolación (**Ec. 2.2.11**). Estas son las que se tendrán que considerar en el momento de resolver el gradiente del kernel. Otro aspecto importante que se tiene que considerar dentro de la ecuación (**Ec. 2.2.5**) es el término de la viscosidad artificial.

2.2.3 Viscosidad artificial.

En la ecuación de movimiento (**Ec. 2.2.5**), se contempla un fluido no viscoso. Esto produce que el fluido presente oscilaciones y colisiones no físicas en la simulación. Para resolver este problema que existe en el fluido, se introducirá un término denominado viscosidad artificial. Existen muchas propuestas sobre este término, pero la que se usará es la viscosidad **[14]**.

$$\prod\nolimits_{i,j} = \begin{cases} \frac{-\alpha \overline{C_{i,j}} \mu_{i,j} + \beta {\mu_{i,j}}^2}{\overline{\rho_{i,j}}} & para\ v_{i,j} \cdot r_{i,j} < 0 \\ 0 & para\ otro\ caso \end{cases}$$

Ec. 2.2.14

Donde

$$\mu_{i,j} = \frac{h v_{i,j} \cdot r_{i,j}}{{r_{i,j}}^2 + 0.001h^2}, \overline{\rho_{i,j}} = \frac{1}{2}(\rho_i + \rho_j) y\ \overline{C_{i,j}} = \frac{1}{2}(C_i + C_j)$$

Ec. 2.2.15

la variable c describe el promedio de la velocidad del sonido. Las constantes α y β son constantes de viscosidad, las cuales toman valores de $0.1 - 0.01\ y\ 0$, respectivamente **[15]**. La constante α produce valores viscosos en el volumen del fluido. Por otro lado, la constante β, amortigua las colisiones entre partículas cuando la velocidad de la partícula es muy alta.

En el denominador de la primera expresión se contempla el término $0.001h^2$ para evitar divergencias cuando la distancia entre las partículas $i\ y\ j$ sea mínima. La expresión de la viscosidad artificial, se introduce en la ecuación de movimiento (**Ec. 2.2.5**). Por lo tanto, esta quedará de la siguiente manera:

$$\frac{d}{dt}v = -\sum_j m_j \left(\frac{p_i}{{\rho_i}^2} + \frac{p_j}{{\rho_j}^2} + \prod_{i,j}\right) \cdot \nabla_i W_{ij}$$

Ec. 2.2.16

Al introducir la viscosidad artificial en la ecuación (**Ec. 2.2.5**), además de evitar los efectos de oscilaciones y colisiones no reales, se incluye la disipación de energía (fricción) entre las partículas al momento de existir colisiones entre ellas.

Por último, para completar la ecuación de movimiento (**Ec. 2.2.16**), se agregan las fuerzas externas que existen en las partículas del fluido. La fuerza externa contempla los efectos de las superficies de fronteras y los de la gravedad G. Por lo tanto, la ecuación quedará se la siguiente forma:

$$\frac{d}{dt}v = -\sum_j m_j \left(\frac{p_i}{{\rho_i}^2} + \frac{p_j}{{\rho_j}^2} + \prod_{i,j}\right) \cdot \nabla_i W_{ij} + F$$

Ec. 2.2.17

Cabe aclarar que las fuerzas externas se normalizan con respecto a la masa de las partículas, por lo que al final las fuerzas externas se convierten en aceleraciones externas.

2.3 Cálculo Fuerzas Externas.

Como se había comentado, las superficies de frontera inyectan una fuerza externa en las partículas del fluido cuando éstas chocan contra la superficie. Para calcular esta fuerza se parte de la ecuación de movimiento (**Ec. 2.2.17**) para una partícula dentro del fluido utilizando el método HSP.

$$\frac{d}{dt}v = -\sum_j m_j \left(\frac{p_i}{{\rho_i}^2} + \frac{p_j}{{\rho_j}^2} + \prod_{i,j}\right) \cdot \nabla_i W_{ij} + Fint + Fext$$

Ec. 2.3.1

Donde la fuerza interna (F_{int}), es la fuerza de gravedad y la fuerza externa (F_{ext}) o fuerza de colisión (F_c) para este caso, es la fuerza que se inyecta a la partícula dada la colisión existente en la superficie. Cabe mencionar que estas fuerzas deben estar expresadas por unidad de masa $(F_{ext} = F_c/m)$, por lo que en realidad son aceleraciones.

Ahora se desarrollará el tratamiento de las fuerzas para una partícula que choca contra una superficie.

Se puede modelar la fuerza que la superficie ejerce a la partícula con dos componentes $F_c = F_d + F_f$, una perpendicular a la superficie (F_d), debida principalmente a los esfuerzos elásticos de deformación mecánica de la superficie y una componente tangencial a la superficie (F_f) debida principalmente a los esfuerzos de fricción. Esta última tendrá la dirección tangencial de la velocidad de la partícula al momento del choque, pero en sentido contrario.

Así pues, es necesario definir estas direcciones. Esto puede hacerse definiendo los vectores unitarios en estas direcciones.

2.3.1 Fuerza Normal a la Superficie.

Sea $\lambda_N \in R_3$ un vector unitario en la dirección normal a la superficie de contacto (**Fig. 2.3.1**).

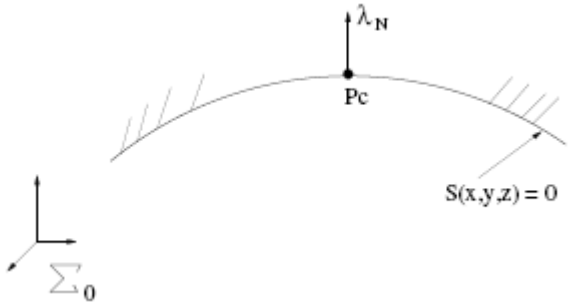


Fig. 2.3.1 Representación de la Fuerza normal a la superficie.

Sea $S(x, y, z) = 0$ una función en el espacio euclidiano tridimensional con coordenadas cartesianas, que define una superficie cualquiera. Sea el punto $Pc = [(x_c, y_c, z_c)]^T$, el punto en el espacio euclidiano donde ocurre el choque, con coordenadas (x_c, y_c, z_c) con respecto a un referencial inercial $\sum_0$.

El vector $\lambda_N$puede ser calculado a partir del gradiente normalizado de la superficie $S(x, y, z)$ en el punto P_c:

$$\lambda_N = \frac{\nabla_s(\mathrm{P_c})}{\|\nabla_s(\mathrm{P_c})\|} = \begin{pmatrix} \lambda_x N \\ \lambda_y N \\ \lambda_z N \end{pmatrix}$$

Ec. 2.3.2

Donde

$$\nabla_s = \left(\frac{\partial_s}{\partial_x}, \frac{\partial_s}{\partial_y}, \frac{\partial_s}{\partial_z}\right)^T \text{ y } \|\lambda_N\| = 1$$

Ec. 2.3.3

Por las características de las superficies, el vector unitario λ_N apuntaría hacia la dirección cóncava de la superficie $S(P_c)$, por lo que el signo debe ajustarse dependiendo de la partícula, si está de un lado o del otro. Se puede suponer que este vector normal apunta hacia el lado de la superficie con el que la partícula ha chocado.

2.3.2 La Fuerza de Deformación Normal.

De acuerdo a la (**Fig. 2.3.2**) definimos la posición de deformación de la partícula al chocar con la superficie como $X = P - P_c$, donde $P(x, y, z)$ es la posición de la partícula en el espacio. La posición normal a la superficie puede calcularse como la magnitud (escalar) de dicha deformación en la dirección del vector unitario normal como:

$$X_N = \|X_N\|\lambda_N = \lambda_N\|X_N\|$$

Ec. 2.3.4

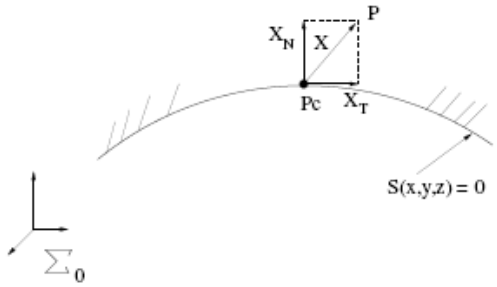


Fig. 2.3.2 Representación de la Fuerza de Deformación.

La magnitud de este vector de deformación es obtenida por la proyección geométrica del vector de deformación total sobre la dirección normal a la superficie, es decir, usando el producto punto o reescribiendo la relación por medio de álgebra lineal $\|X_N\| = \lambda_N \cdot X = \lambda_N^T \cdot X$. Así, la deformación tangencial se obtiene por la relación siguiente:

$$X_N = \lambda_N \lambda_N^T X = [\lambda_N \lambda_N^T] X$$

Ec. 2.3.5

Donde el operador lineal está dado por:

$$[\lambda_N \lambda_N^T] = \begin{bmatrix} \lambda_{xN}^2 & \lambda_{xN}\lambda_{xyN} & \lambda_{xN}\lambda_{zN} \\ \lambda_{xN}\lambda_{yN} & \lambda_{yN}^2 & \lambda_{yN}\,\lambda_{zN} \\ \lambda_{xN}\lambda_{zN} & \lambda_{yN}\lambda_{zN} & \lambda_{zN}^2 \end{bmatrix}$$

Ec. 2.3.6

Con la siguiente propiedad:

$$[\lambda_N \lambda_N^T]^2 = \|\lambda_N\|^2 [\lambda_N \lambda_N^T] = [\lambda_N \lambda_N^T]$$

Ec. 2.3.7

De la misma manera, se puede calcular una velocidad normal, usando velocidades en lugar de posición. Así se define una velocidad de deformación como: $\dot{X} = \dot{P} - \dot{P}_c$, donde $\dot{P}$ y $\dot{P}_c$ son las velocidades lineales de la partícula y del punto de contacto, respectivamente, de una superficie rígida (si la superficie se considera rígida y sin movimiento, la velocidad del punto de contacto es nula, $\dot{P}_c = 0$). Entonces la componente de la velocidad de contacto, normal a la superficie es:

$$\dot{X_N} = \lambda_N \lambda_N^T \dot{X} = [\lambda_N \lambda_N^T]\dot{X}$$

Ec. 2.3.8

Entonces, la fuerza que la partícula ejerce sobre la superficie puede modelarse por $F_{dp} = k_s X_N + b_s \dot{X_N}$, donde la k_s y b_s son coeficientes de rigidez y disipación viscosa de la superficie de contacto, respectivamente. Los signos de los términos de la fuerza F_{dp} están dados por la hipótesis que el vector unitario normal a la superficie es positivo apuntando hacia el interior del cuerpo cuya superficie es la superficie de contacto de la partícula. Por la tercera ley de Newton, la fuerza que ejerce la superficie a la partícula es entonces la fuerza de misma magnitud, pero de sentido opuesto. Esta es calculada entonces como:

$$F_d = -k_s X_N - b_s \dot{X}_N$$

Ec. 2.3.9

Haciendo las sustituciones adecuadas, esta fuerza puede ser calculada únicamente con una sola proyección al espacio normal de la superficie en el punto de contacto por:

$$F_d = -k_s X_N - b_s \dot{X}_N = -[\lambda_N \lambda_N^T]\left(k_s X + b_s \dot{X}\right)$$

Ec. 2.3.10

Los coeficientes de la superficie deben ser calculados en función del material de contacto. El coeficiente de rigidez debe ser equivalente al coeficiente de deformación elástica del material en cuestión $k_s = k_e$. El coeficiente de disipación viscosa es una aproximación lineal de la disipación energética durante el choque de la partícula. Así un coeficiente $b_s \geq 2\sqrt{k_s m_p}$, donde m_p representa la masa de la partícula, equivalente a un material "absorbente". Esto quiere decir que la velocidad de la partícula durante el choque, normal a la superficie, será completamente absorbida por el material, dando como resultado que la partícula quede "pegada" a la superficie de contacto. Por el contrario, coeficientes $b_s \leq 2\sqrt{k_s m_p}$, representan materiales no absorbentes, donde las partículas serán rebotadas después del choque. **[16]**

Esta noción puede ser verificada bajo el siguiente razonamiento. Supongamos que sólo hay una partícula y que la ecuación de esta está definida por la segunda ley de Newton:

$$\frac{d}{d_t}(m_p\dot{X}) = \sum F_{ext}$$

Ec. 2.3.11

Donde F_{ext} representa la suma de todas las fuerzas exógenas a la partícula que modifican su comportamiento. Para el caso de la partícula durante el choque estas están dadas por la suma $F_c = F_d + F_f$. Sin embargo, se sabe que las fuerzas de deformación y de fricción son perpendiculares. Si la masa de la partícula m_p es constante, la variación de la cantidad de movimiento de la partícula $(m_p\,\dot{X}\,)$ queda como:

$$m_p\ddot{X} = F_d + F_f$$

Ec. 2.3.12

Separando la componente normal a la superficie de contacto por el operador$[\lambda_N\lambda_N^T]$ la ecuación (**Ec. 2.3.12**) queda como: $[\lambda_N\lambda_N^T]m_p\ddot{X} = F_d$ Donde se asume que el producto $[\lambda_N\lambda_N^T]F_f = 0$, ya que la fuerza de fricción es ortogonal a la fuerza de deformación.

Sustituyendo la expresión de la fuerza de deformación (**Ec. 2.3.9**) en la ecuación de movimiento de la partícula (**Ec. 2.3.12**) en la dirección normal de la superficie queda de la siguiente forma:

$$[\lambda_N\lambda_N^T]m_p\ddot{X} = -k_sX_N - b_s\dot{X}_N$$

Ec. 2.3.13

Reordenando los términos y sustituyendo la aceleración en la dirección normal a la superficie, esta última queda de la forma lineal de segundo orden:

$$m_p\ddot{X} + b_s\dot{X}_N + k_sX_N = 0 \text{ Con Contacto.}$$

$$m_p\ddot{X} = 0 \text{ Sin Contacto}$$

Ec. 2.3.14

La solución de la primera ecuación es exponencialmente estable al origen (deformación nula de la superficie de contacto). Sin embargo, si el contacto se pierde antes que esta ecuación llegue a su estado estacionario, la ecuación de movimiento estará regida por la segunda ecuación. Entonces, el tipo de

absorción estará dado por la relación de los coeficientes m_p, b_s y k_s que definen el tipo de solución de la ecuación de segundo orden en respuestas subamortiguada, críticamente amortiguada y sobreamortiguada **[17]**. Respuestas del tipo amortiguada o sobreamortiguada no permitirán salir a la partícula de la superficie, mientras que la respuesta subamortiguada sí lo hará durante el primer sobretiro de la respuesta. El valor crítico del coeficiente de disipación para el cual la respuesta es críticamente amortiguada puede obtenerse por inspección de la ecuación de segundo orden. Esta es: $b_{scrit} = 2\sqrt{K_s m_p}$ para obtener una respuesta subamortiguada es necesario que el coeficiente de disipación sea definido como $b_s \doteq 2\varepsilon\sqrt{K_s m_p}$ donde ε es la relación de amortiguamiento y su valor debe ser estrictamente $\varepsilon < 1$.

2.3.3 Fuerza de Fricción Tangencial

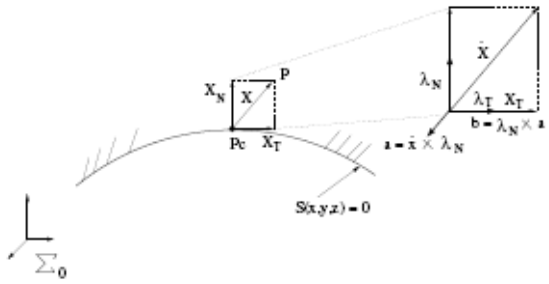


Fig. 2.3.3 Representación de la Fuerza de Fricción en la Superficie.

Para determinar la fuerza de fricción sobre la superficie tangencial de contacto, hay que determinar la dirección en la que ocurre este fenómeno. Así que lo primero es calcular el vector unitario con la dirección de la velocidad tangencial de la partícula, en la superficie de contacto. Para ello definimos dos vectores auxiliares (**Fig. 2.3.3**) a y b de la siguiente manera. El vector a debe ser perpendicular al vector normal λ_Ny a la componente tangencial de la velocidad $\dot{X}_T$. Como es así, también es perpendicular al vector de la velocidad relativa de la partículaẊ , por lo que puede ser calculado como el producto vectorial (producto cruz) de la velocidad relativa de la partícula y el vector normal como $a = \dot{X}\ x\ \lambda_N$. El segundo vector auxiliar b se define en la dirección de la velocidad tangencial por el producto cruz entre el vector normal λn y el vector auxiliar a como $b = \lambda_N\ x\ a$. Así, este vector tiene magnitud cualquiera, pero la dirección adecuada, en el sentido y dirección de la velocidad tangencial de la partícula sobre la superficie de contacto:

$$b = \lambda_N \ (\dot{\mathrm{X}} \mathrm{x} \lambda_N) = -\lambda_N (\lambda_N \mathrm{x} \dot{\mathrm{X}})$$

$$\frac{b}{\|b\|} = \frac{\dot{X_T}}{\|X_T\|}$$

Ec. 2.3.15

Entonces, el vector unitario en la dirección tangencial puede definirse como:

$$\lambda_T = \frac{\dot{b}}{\|b\|}$$

Ec. 2.3.16

Donde se deduce que la magnitud de este vector es efectivamente la unidad

$$\|\lambda_T\| = 1.$$

Ec. 2.3.17

Con este vector, se puede calcular la componente tangencial de la velocidad relativa, mediante el mismo razonamiento usado para las componentes normales, reemplazando el vector unitario normal por el vector unitario tangencial:

$$X_T = \lambda_T \lambda_T^T \dot{X} = [\lambda_T \quad \lambda_T^T] \dot{X}$$

Ec. 2.3.18

Entonces, la fuerza de fricción tangencial al contacto de la partícula con la superficie, puede modelarse muy simplemente como la fuerza de Coulomb, en el sentido contrario a la velocidad de contacto, por la ecuación $F_f = \mu_k \|F_N\| (-\lambda_T)$. Donde μ_k es el coeficiente de fricción dinámico entre la superficie y la partícula. Por otro lado, F_N es la fuerza normal con la que la partícula interacciona con la superficie. Esta última no es otra sino la fuerza de deformación de la superficie (**Ec. 2.3.10**). Entonces, la fuerza de fricción puede calcularse como:

$$F_f = -\mu_k \ \|F_d\| \frac{\dot{X_T}}{\|\dot{X_T}\|}$$

Ec. 2.3.19

Esta relación puede calcularse sin necesidad de evaluar la componente de velocidad tangencial, ya que el vector unitario tangencial puede ser calculado tanto por la componente de la velocidad $\dot{X}_T$como por el vector auxiliar b. Sin embargo, un resultado interesante es saber que el vector auxiliar b es, de hecho, la componente tangencial de la velocidad relativa de la partícula $b = \dot{X}_T$.

2.4 Herramientas de desarrollo y lenguaje utilizado.

Las herramientas que se proponen en este epígrafe serán puestas en marcha en todas las fases de desarrollo del módulo, tal como el lenguaje que se referencia.

2.4.1 Visual Studio 2003.

Microsoft Visual Studio 2003 es un software desarrollado por la compañía Microsoft, entre las características fundamentales que presenta se encuentran algunas como, su capacidad de crear software profesional con rapidez, reducir los costos de funcionamiento de tecnologías de la información y además se integra con una amplia gama de aplicaciones, sistemas y dispositivos, la plataforma sobre la que se ha desarrollará este trabajo es el entorno de programación Visual C++, debido a que la programación correspondiente a la parte gráfica y el Modelo matemático se realizan en C++, además de ser un lenguaje con posibilidad de desarrollo en plataforma libre.

2.4.2 Rational Rose.

Es una herramienta de desarrollo del software para el Modelado Visual. Rational tiene gran ventaja para el equipo de desarrollo ya que los unifica a través del modelamiento el cual está basado en el Unified Modeling Language™ (UML). El UML es la notación estándar para arquitectura de software. Esto significa que con Rational Rose, todo el equipo puede comunicarse con un lenguaje y una herramienta. Rational Rose domina el mercado de herramientas para el análisis, modelamiento, diseño y construcción orientado a objetos. De acuerdo a International Data Corporation (IDC), la participación de Rational en 1998 con respecto a los ingresos del mercado es mayor que las participaciones de los siguientes cuatro competidores directos combinados. Por cuatro años consecutivos IDC ha nombrado a Rational Rose como "La Herramienta Líder en Análisis, Diseño y Construcción Orientada a Objetos", con base en los ingresos del producto. Rational Rose permite visualizar, entender, y refinar los requerimientos y arquitectura antes de enfrentar el código. Esto permite, evitar esfuerzos desperdiciados en el ciclo de

desarrollo. El modelo arquitectónico puede ser rastreado hacia el modelo de procesos de negocios y los requerimientos de sistema. Esta herramienta permite especificar, analizar, diseñar el sistema antes de codificarlo. Tiene varias características que lo distinguen como son el de chequear la sintaxis UML, mantiene la consistencia de los modelos del sistema del software, genera la documentación automáticamente, la generación de código a partir de los modelos, permite la ingeniería inversa (crear modelo a partir código) entre otras.

2.4.3 C++.

Existen principalmente tres lenguajes que se utilizan para desarrollar aplicaciones gráficas profesionales en 3D: Lenguaje Ensamblador, C y C++, por ser los que con más velocidad ejecutan el código (menor costo de ejecución del programa). A éstos se ha unido recientemente el Java como una opción para el desarrollo de este tipo de aplicaciones. Es decisión de la entidad cliente Simulador Quirúrgico, implementar este proyecto mediante el lenguaje C++, que ha estado en su línea de trabajo con magníficos resultados. Es el lenguaje en que se tiene mayor experiencia por parte del equipo del Simulador Quirúrgico, futuros desarrolladores del sistema que ocupa este proyecto. Si se estudian las características de este lenguaje, se puede apreciar lo acertado de la elección, dado que C++ es un lenguaje de programación de propósito general, especialmente indicado para la programación de sistemas por su flexibilidad y su potencia.

Capítulo 3: Características del Sistema.

Introducción.

En este capítulo se obtiene una visión más específica del sistema que se va a desarrollar, partiendo fundamentalmente de las soluciones técnicas. De inicio se obtiene una panorámica desde el punto de vista del negocio dando continuidad a crear la concepción práctica del producto a elaborar partiendo de las necesidades del cliente desde el punto de vista de que es lo que debe hacer el sistema.

3.1 Reglas del Negocio.

El fichero de textura a cargar para las partículas debe ser de extensión BMP, en caso de no existir el fichero en el camino indicado, ser de otro formato, estar corrupto o no tener textura el modelo, no constituye un error, solo se debe indicar lo sucedido y cargar el modelo sin textura. El programador que emplee esta herramienta debe tener conceptos previos de simulación de partículas y fundamentalmente de la dinámica de los fluidos.

3.2 Modelo de Dominio.

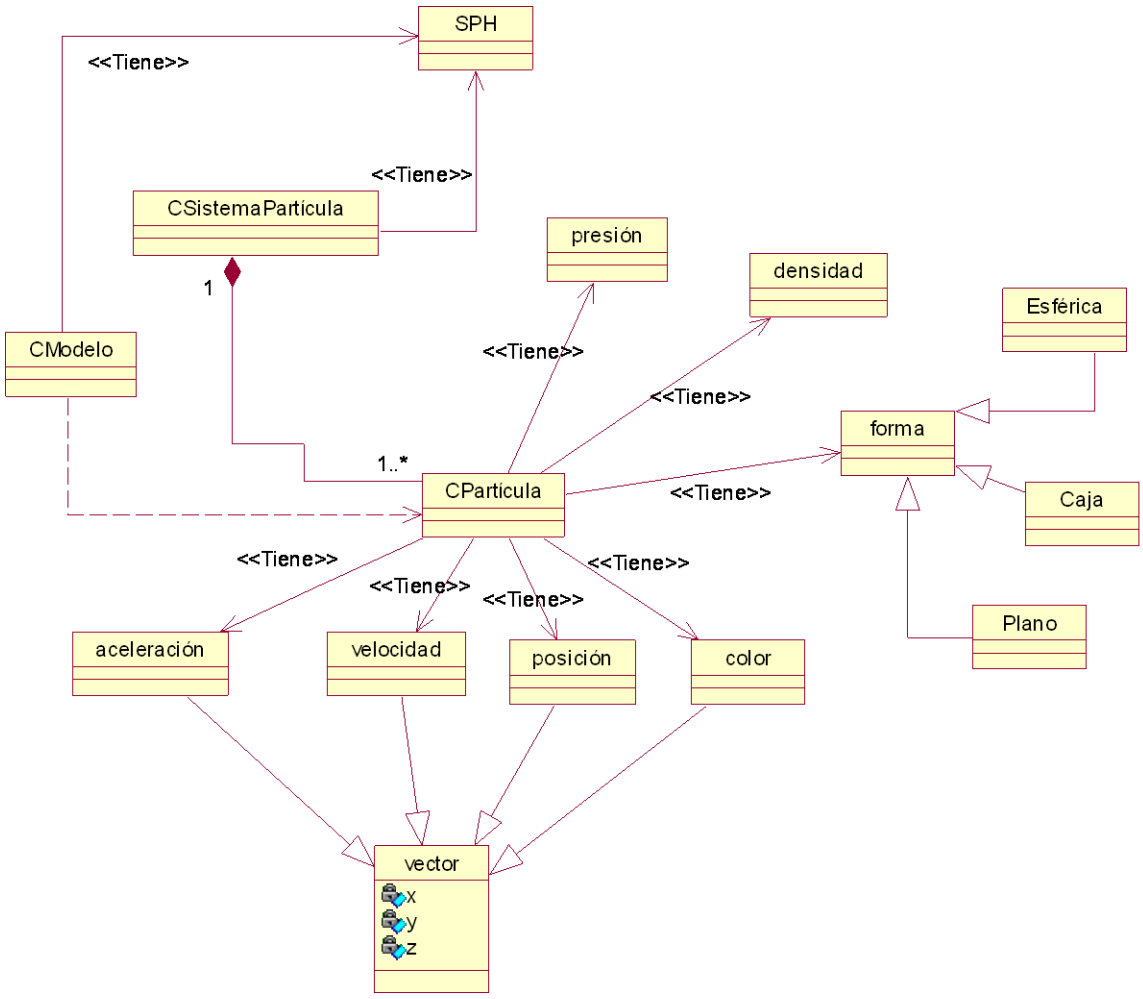


Fig. 3.2.1 Diagrama del Modelo del Dominio.

3.3 Glosario de términos del Dominio.

Partícula: Es un registro que contiene algunos atributos.

Aceleración: Es un vector de aceleración velocidad que utiliza el movimiento de la partícula para poder simular los efectos.

Posición: Es un vector que proporcionará la posición de la partícula en el espacio.

Velocidad: Es un vector velocidad que se utiliza en el movimiento de la partícula.

Forma: Puede ser de cualquier tipo, esférica, forma de caja, forma de un plano, etc.

Presión: Es una propiedad física del fluido.

Densidad: Es una propiedad física del fluido.

Color: Define el color como un RGB (para los píxeles). Puede usarse para almacenar el índice de un arreglo de bitmaps creado previamente para colorear cada partícula.

Vector: Vector de 3 componentes flotantes (x, y, z).

3.4 Captura de requisitos.

A continuación se expondrán los requisitos funcionales y no funcionales del sistema.

3.4.1 Requisitos Funcionales.

1- Definir atributos de partículas.

2- Crear partícula.

3- Definir la forma de la partícula.

4- Seleccionar fichero de textura.

5- Verificar extensiones de fichero de textura.

6- Cargar textura.

7- Definir cantidad de partículas a utilizar en el fluido.

8- Definir las fronteras a utilizar en el fluido.

9- Calcular elementos necesarios que se utilizarán en la simulación de las partículas (pre acción).

10- Actualizar las partículas.

11- Crear las estructuras de la simulación física.

12- Inicializar las variables con los valores físicos (difusión, densidad, presión, masa, etc.).

13- Dibujar las partículas (render).

14- Modificar las partículas con nuevos atributos.

15- Ejecutar el fluido para su visualización.

16- Eliminar las estructuras realizadas en la simulación física.

17- Eliminar las estructuras realizadas en la simulación del visual.

3.4.2 Requisitos no Funcionales.

Usabilidad: Los futuros usuarios del sistema serán programadores con conocimientos básicos con respecto al tema de sistema de partículas y fluidos.

Rendimiento: Debe ser una aplicación en tiempo real donde tenga gran alto de velocidad de procesamiento de los cálculos, tiempo de respuesta y recuperación del sistema.

Hardware: Compatibilidad con tarjetas gráficas de la familia NVIDIA, y tarjeta de video de más de 128 megabyte de RAM.

Diseño e implementación: Debe trabajar con la biblioteca gráfica de OpenGL. Se harán llamadas a dichas bibliotecas desde Leguaje C++. Se regirá por la filosofía de Programación Orientada a Objetos.

3.5 Modelo de caso de uso.

En este epígrafe se darán a conocer los actores del sistema a desarrollar, y a partir de los requisitos funcionales del epígrafe anterior se obtendrán los casos de usos del sistema con sus respectivas descripciones.

3.5.1 Definición de los Actores del Sistema.

Tabla 3.5.1 Justificación de los actores.

Actores	Justificación
KHEIPROS	

	Es el que se beneficiará con las funcionalidades que ofrece la “simulación de fluidos” donde obtendrá: crear fluidos, ejecutar fluidos, etc.

3.5.2 Casos de uso del sistema.

1. Crear fluido.
2. Ejecutar fluido.
3. Actualizar fluido.
4. Eliminar fluido.
5. Localizar fluido.
6. Modificar fluido.

3.5.3 Diagrama de casos de uso.

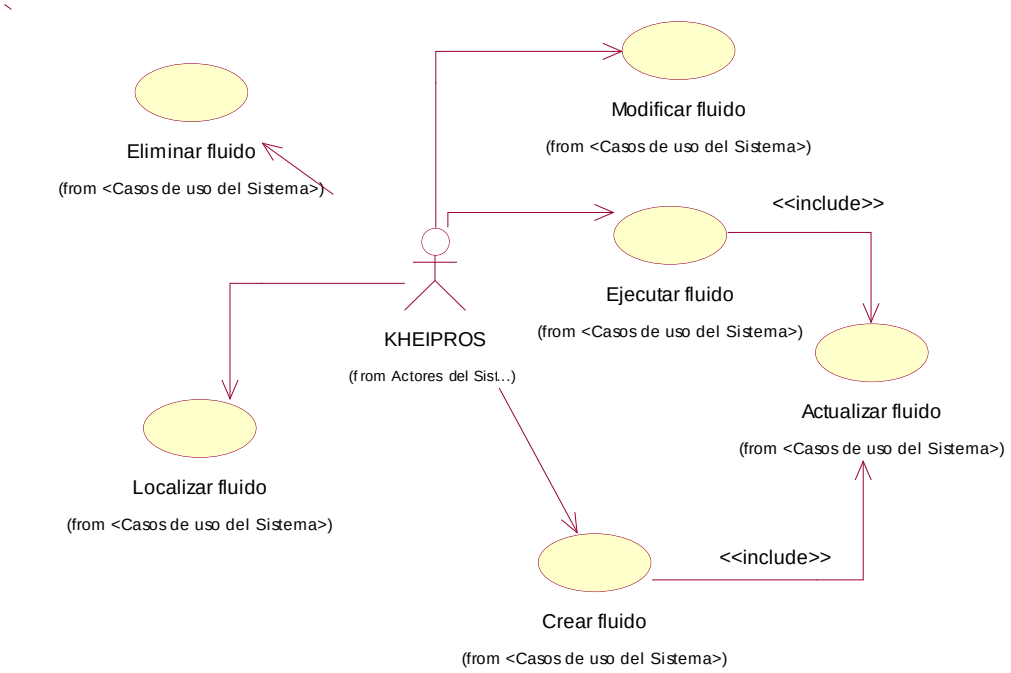


Fig. 3.5.1 Diagrama de Casos de Usos del Sistema.

3.5.4 Descripción de los casos de uso en el formato expandido.

Tabla 3.5.2 CU "Crear Fluido".

Nombre del caso de uso.	**Crear fluido.**
Actores.	KHEIPROS.
Propósito.	Crear el fluido.
Resumen: Se inicia cuando el actor solicita crear el fluido. De ocurrir algún error en el proceso se le informa al actor. El CU finaliza cuando el efecto es creado con todos sus objetos y variables.	
Referencias.	R5, R6, R7, R8, R9, R10, CU3 *(include).*
Precondición.	-
Poscondición.	Fluido creado.

Curso Normal de los eventos.	
Acción del actor.	Respuesta del sistema.
1- Solicita crear el fluido.	1.1- Crea la lista de partículas con la cantidad de partículas.
	1.2- Inicializa el origen del sistema de partículas.
	1.3- Crea la simulación física con los parámetros físicos.
Curso Alterno.	
Prioridad: Crítico.	

Tabla 3.5.3 CU "Actualizar Fluido".

Nombre del caso de uso.	**Actualizar Fluido.**
Actores.	
Propósito.	Actualizar todas las variables de las partículas como son la posición, velocidad etc.
Resumen: El Caso de Uso se inicia cuando el CU "Crear fluido" o el CU "Ejecutar fluido" solicitan actualizar todas las partículas que se emitirán en un fluido, disminuyendo los valores del tiempo de vida.	
Referencias.	R12.
Precondición.	**-**
Poscondición.	Todas las partículas estén actualizadas.
Curso Normal de los eventos.	

Acción del actor.	Respuesta del sistema.
1- Indica actualizar los valores.	1.1- Se le da una acción previa a las partículas.
	1.2-Se calculan todas las variables del fluido.
	1.3- Se cambian los valores de las variables calculadas.
Curso Alterno.	
Prioridad: Crítico.	

Tabla 3.5.4 CU "Ejecutar Fluido".

Nombre del caso de uso.	**Ejecutar Fluido.**
Actores.	KHEIPROS.
Propósito.	Ejecutar Fluido.
Resumen: El CU se inicia cuando el actor solicita ejecutar el fluido, donde el fluido será visualizado.	
Referencias.	CU6(include)
Precondición.	Que el fluido este creado.
Poscondición.	Fluido ejecutado.
Curso Normal de los eventos.	
Acción del actor.	Respuesta del sistema.
1-Se solicita ejecutar el fluido.	1.1- Si el fluido está creado se ejecuta.

	1.2- Se actualizan los valores físicos.
	1.3- Se dibujan las partículas que se encuentren en el sistema de partículas.
	1.4- Se actualizan los valores del sistema de partículas.
Curso Alterno.	
Línea 1.1	
	1.1- Si el fluido no está creado se crea y se visualiza.
Prioridad: Crítico.	

Capítulo 4: Diseño del Sistema.

Introducción.

A continuación se presentarán los diagramas de clases de diseño el cual tendrá dos Paquetes uno de “Simulación Visual” y el otro “Simulación Física” con las relaciones existente entre ellas que pueden ser de agregación, generalización/especialización y composición. Además se presentan los diagramas de secuencia de los casos de usos que intervendrán en el primer ciclo de desarrollo de software.

4.1 Modelo de Diseño.

Los artefactos de diagrama de clases de diseño y los diagramas de secuencias por casos de usos del sistema, se darán a conocer en los siguiente sub epígrafes.

4.2 Diagrama de Clases de Diseño.

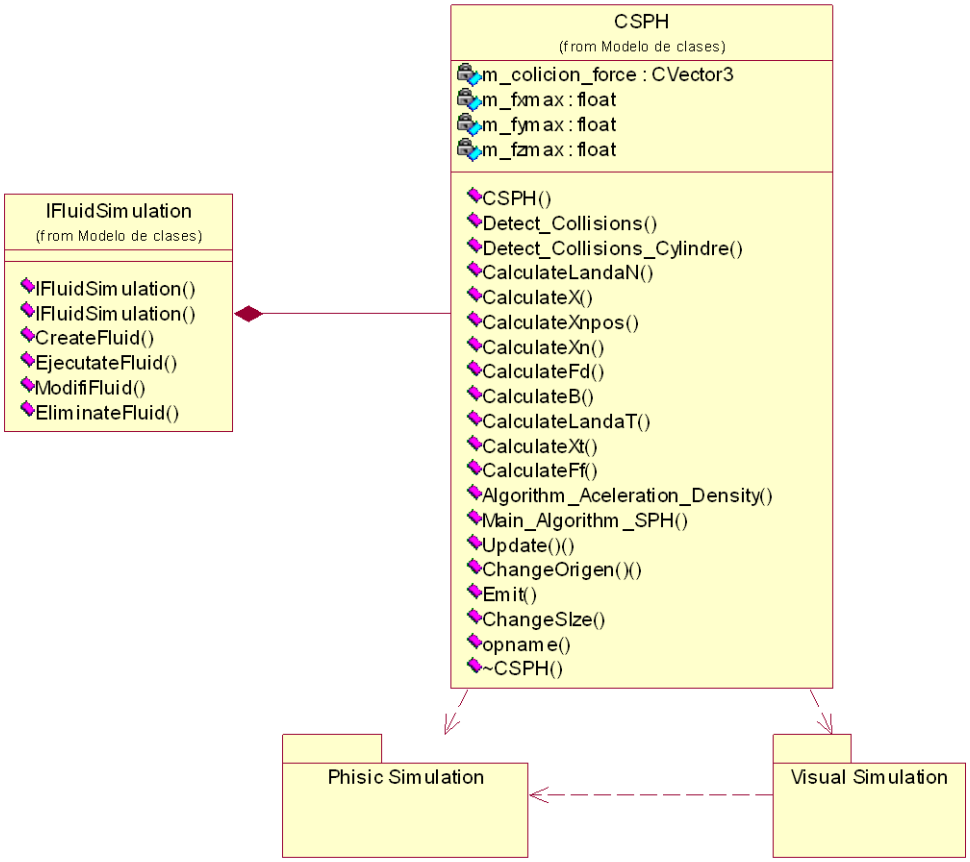


Fig. 4.2.1 Diagramas de paquetes de Clases de Diseño.

4.2.1 Paquete “Visual Simulation”.

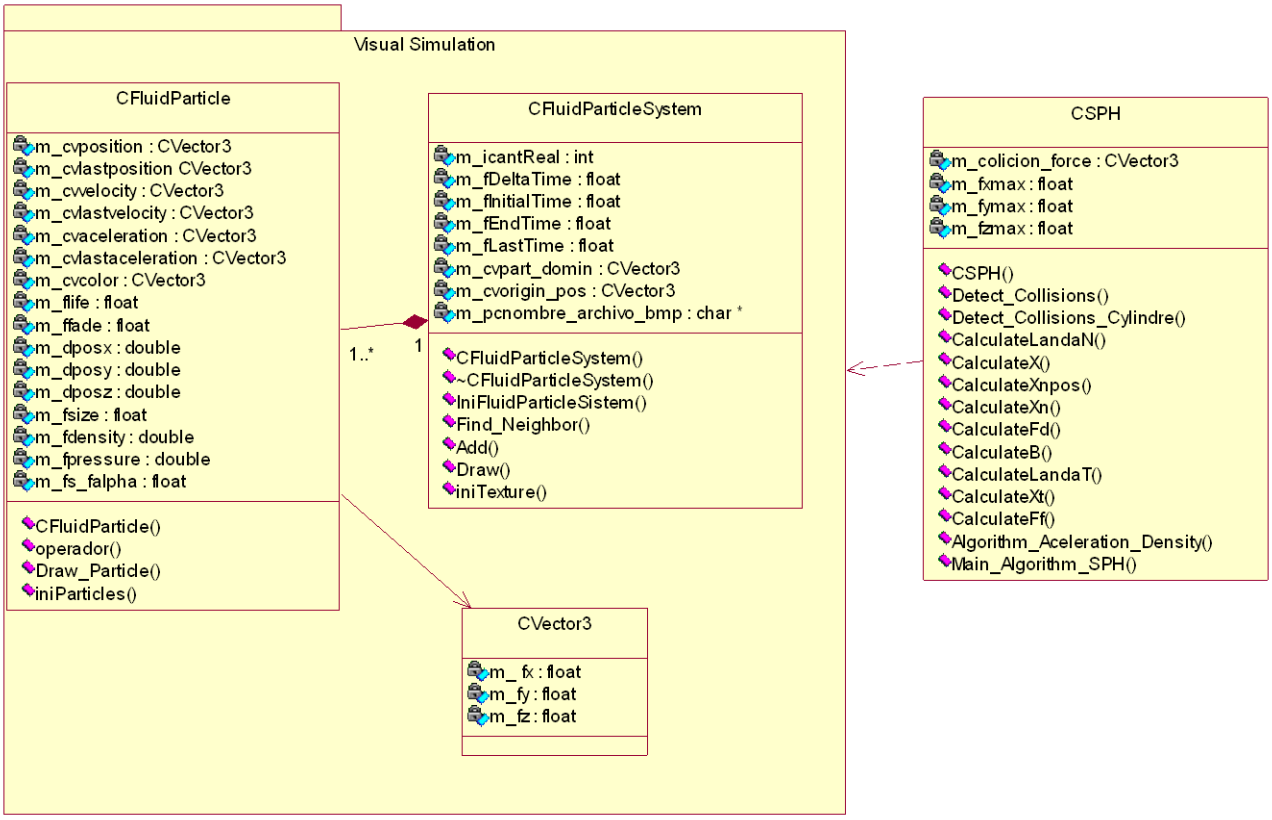


Fig. 4.2.2 Diagrama de Clases del Paquete Simulación Visual.

4.2.2 Paquete “Phisic Simulation”.

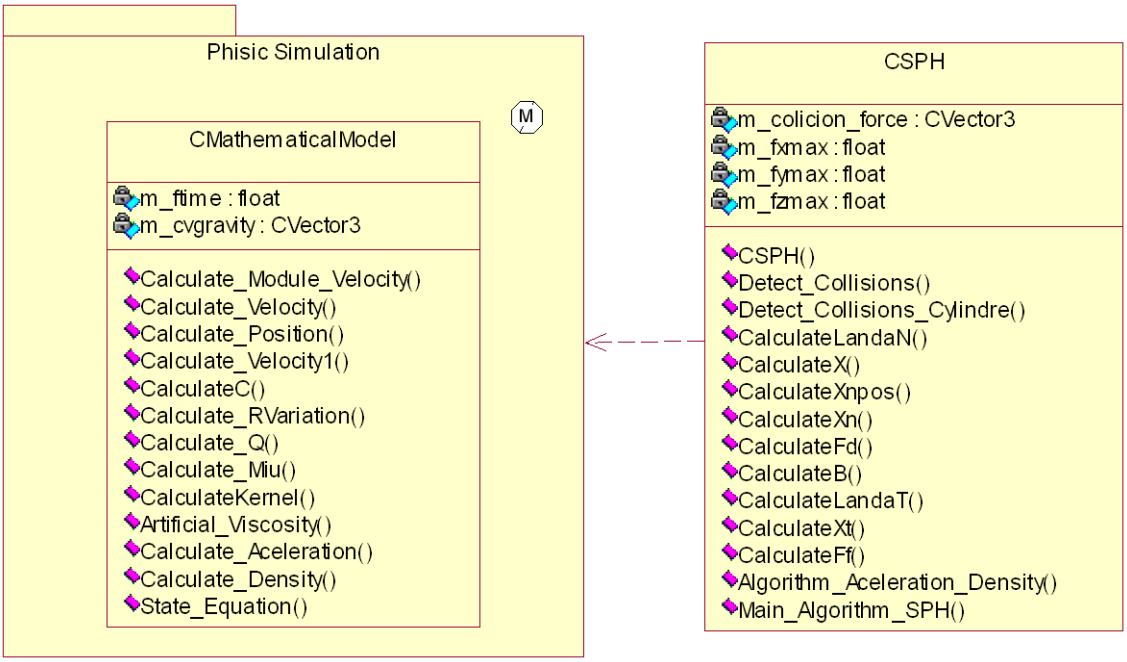


Fig. 4.2.3 Diagrama de Clases del Paquete Simulación Visual.

4.3 Diagramas de interacción del diseño.

4.3.1 Realización del caso de uso “Crear Fluido”.

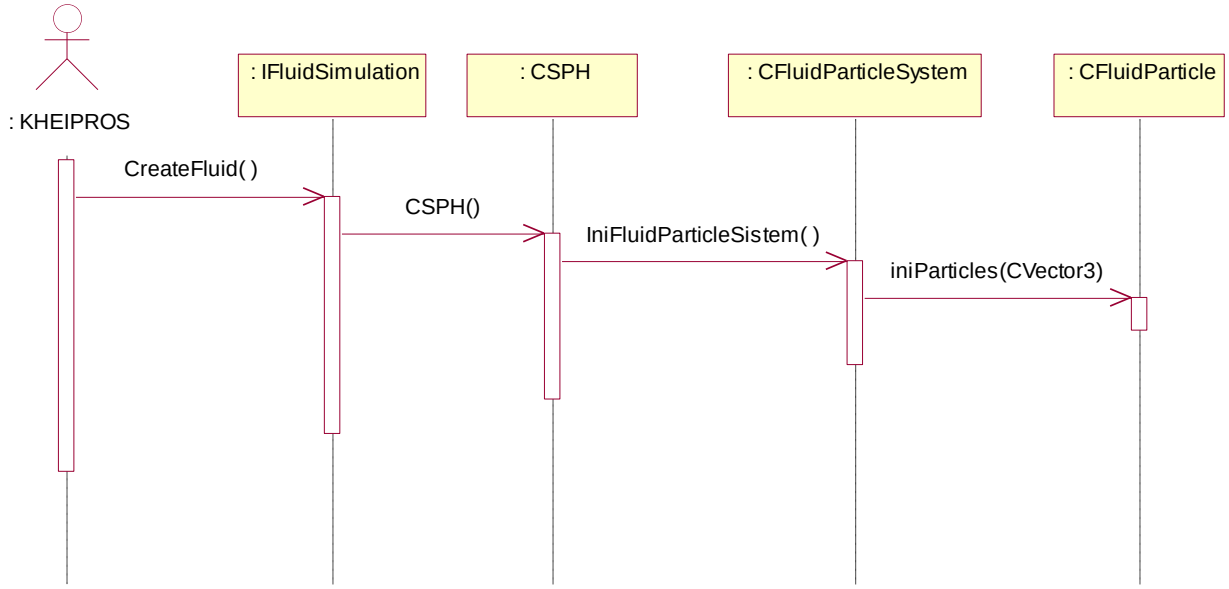


Fig. 4.3.1 Realización del Caso de Uso “Crear Fluido”.

4.3.2 Realización del caso de uso “Ejecutar Fluidos”

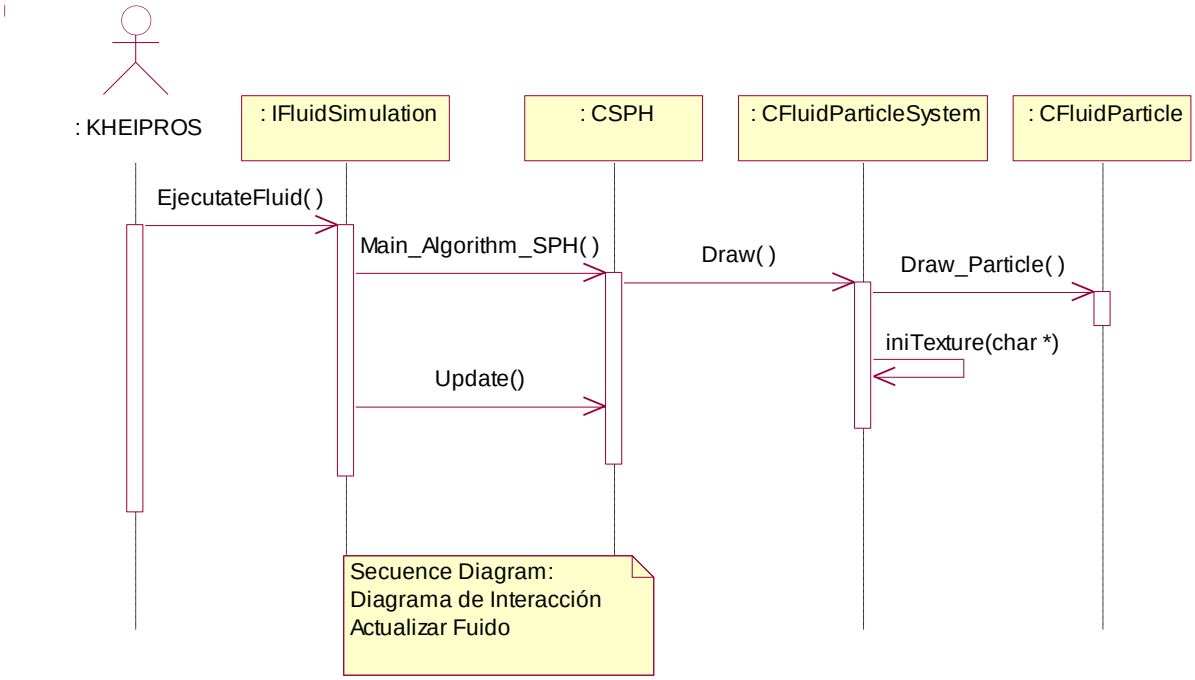


Fig. 4.3.2 Realización del Caso de Uso “Ejecutar Fluido”.

4.3.3 Realización del caso de uso “Actualizar Fluidos”

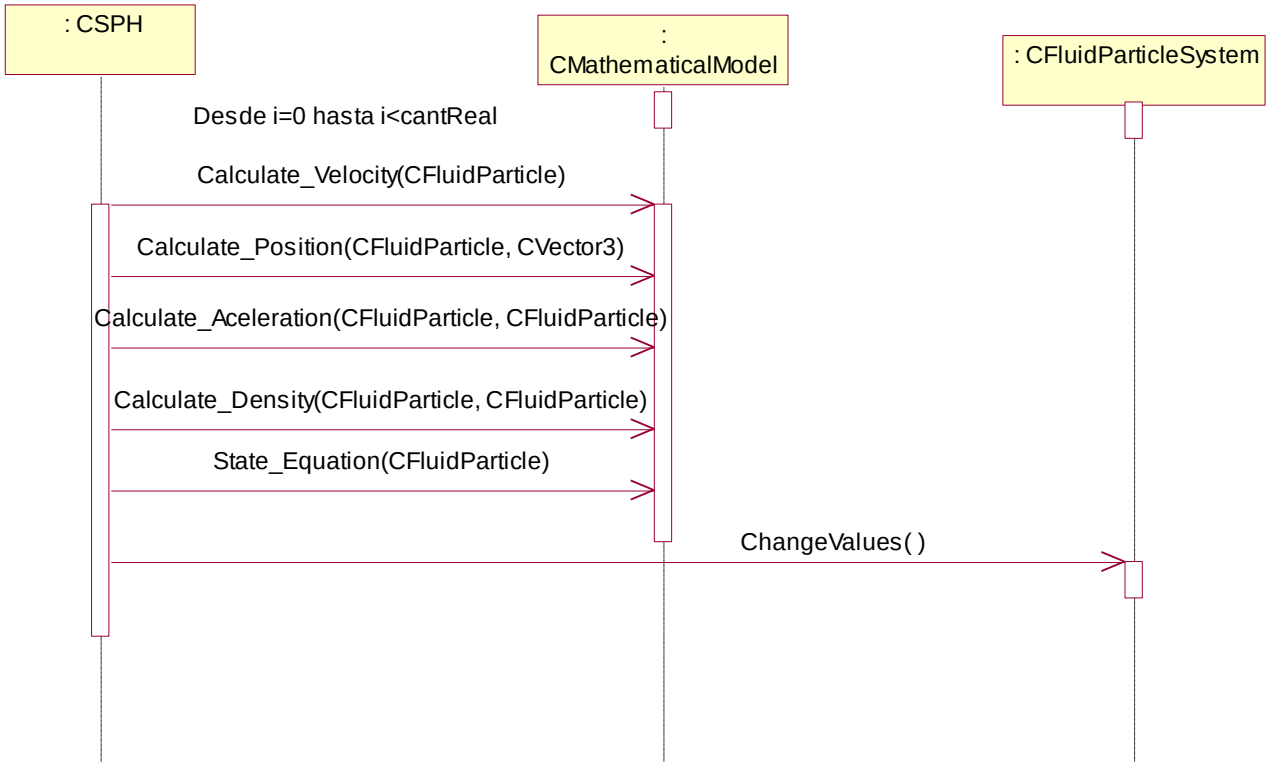


Fig. 4.3.3 Realización del Caso de Uso “Actualizar Fluido”.

4.3.4 Realización del caso de uso “Eliminar Fluidos”

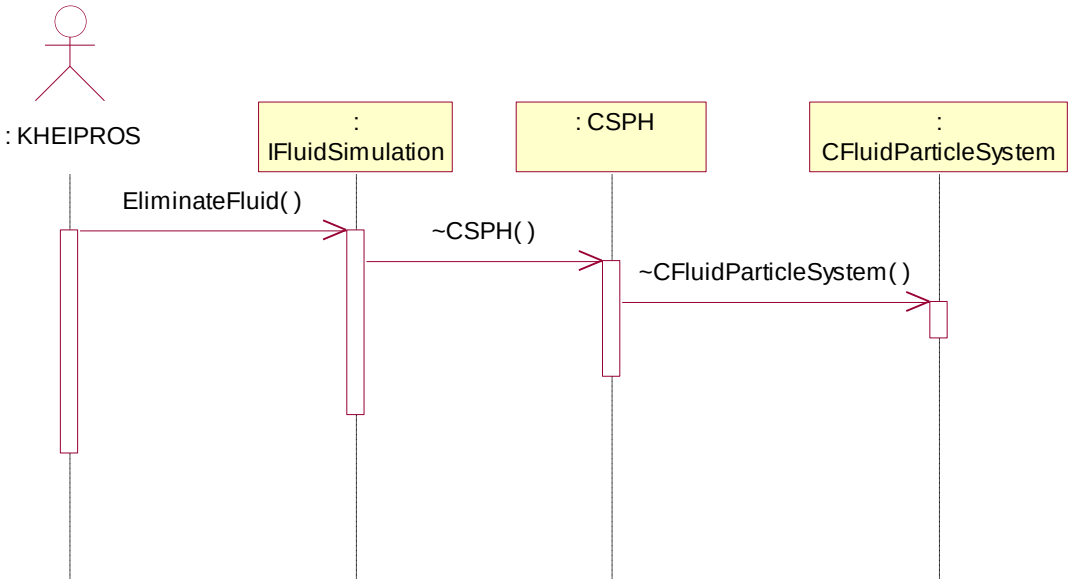


Fig. 4.3.4 Realización del Caso de Uso “Eliminar Fluidos”.

4.4 Descripción de clases.

4.4.1 Clase Controladora

Tabla 4.4.1 Clase “IFluidSimulation”.

Nombre: IFluidSimulation	
Tipo de clase:	
Atributo	**Tipo**
`pkSPH`	`CSPH *`
Para cada responsabilidad:	
Nombre:	CreateFluid()
Descripción:	Crea el Sistema de partícula para comenzar la simulación.
Nombre:	EjecutateFluid()
Descripción:	Ejecuta y visualiza el fluido.

Nombre:	ElimateFluid()
Descripción:	Termina la simulación.

Tabla 4.4.2 Clase "SPH".

Nombre: *SPH*	
Tipo de clase:	
Atributo	**Tipo**
m_pkParticleSistem	CFluidParticleSystem*
m_pkMathematicalModel	CMathematicalModel*
m_pkCylinder	CCylinder*
m_colicion_force	CVector3
m_fxmax	float
m_fymax	float
m_fzmax	float
Para cada responsabilidad:	
Nombre:	Algorithm_Aceleration_Density()
Descripción:	Algoritmo que calcula y cambia la aceleración y la densidad de todas las partículas del sistema de partículas.
Nombre:	Main_Algorithm_SPH()
Descripción:	Algoritmo principal del método SPH en el cual llama a los demás algoritmos de este método.
Nombre:	CVector3 cvColisionForce(){return m_colicion_force;}
Descripción:	Retorna la fuerza de colisión.
Nombre:	Detect_Collisions(CVector3 arg_cvpos)
Descripción:	Detecta las colisiones en un cubo.
Nombre:	Detect_Collisions_Cylindre(CVector3 arg_cvpos)
Descripción:	Detecta las colisiones en un cilindro
Nombre:	CalculateLandaN(CVector3 arg_cvpos)
Descripción:	Calcula la normal

Nombre:	CalculateX(CVector3 arg_cvlastPos, CVector3 arg_cvposA)
Descripción:	Calcula el Vector X=P-Pc.
Nombre:	CalculateXnpos(CVector3 arg_cvpos, CVector3 arg_cvlastpos)
Descripción:	Calcula la posicion Relativa Xn=Landan*(landan.dotprodut(X)).
Nombre:	CalculateXn(CVector3 arg_cvvel,CVector3 arg_cvpos)
Descripción:	Calcular la velocidad Relativa Xn=Landan*(landan.dotprodut(X)).
Nombre:	CalculateFd(CVector3 arg_cvvel,CVector3 arg_cvpos,CVector3 arg_cvlastpos)
Descripción:	Calcula la fuerza de deformación.
Nombre:	CalculateB(CVector3 arg_cvvel,CVector3 arg_cvpos)
Descripción:	Calcula la bariable auxiliar b=-landaN X crosProdut(X crosProdutlandaN).
Nombre:	CalculateLandaT(CVector3 arg_cvvel,CVector3 arg_cvpos)
Descripción:	Calcula la landa t.
Nombre:	CalculateXt(CVector3 arg_cvvel,CVector3 arg_cvpos)
Descripción:	Calcula la Velocidad Relativa.
Nombre:	CalculateFf(CVector3 arg_cvvel,CVector3 arg_cvpos,CVector3 arg_cvfd)
Descripción:	Calcula la fuerza de fricción Ff=-miu* modulo(Fd)* Xt/Modulo(Xt)

4.4.2 Paquete “Visual Simulation”.

Tabla 4.4.3 Clase “CFluidParticle”.

Nombre: *CFluidParticle*	
Tipo de clase:	
Atributo	**Tipo**
m_cvposition	*CVector3*
m_cvlastposition	*Cvector3*
m_cvvelocity	*Cvector3*
m_cvlastvelocity	*CVector3*
m_cvcolor	*CVector3*
m_cvlastaceleration	*CVector3*

m_cvaceleration	*CVector3*
m_flife	*float*
m_ffade	float
m_fsize	float
m_fs_falpha	float
m_dposx	double
m_dposy	double
m_dposz	double
m_fdensity	double
m_fpressure	double
Para cada responsabilidad:	
Nombre:	operator = (const CParticle &particula)
Descripción:	Operador de asignación.
Nombre:	Draw_Particle ()
Descripción:	Dibuja una partícula
Nombre:	iniParticles(CVector3 arg_cvaPos)
Descripción:	Inicializa la partícula.

Tabla 4.4.4 Clase "CFluidParticleSystem".

Nombre: *CFluidParticleSystem*	
Tipo de clase:	
Atributo	**Tipo**
ParticleList	vector<CFluidParticle>
m_icantReal	*int*
m_pcnombre_archivo_bmp	*char**
ktextura	COGLTexture
m_cvpart_domin	*CVector3*
m_cvorigin_pos	*CVector3*

m_fDeltaTime	*float*
m_fLastTime	float
m_fInitialTime	float
m_fEndTime	float
Para cada responsabilidad:	
Nombre:	IniFluidParticleSistem ()
Descripción:	Inicializa el sistema de partículas
Nombre:	TPartList ()
Descripción:	Retorna la lista de Partículas.
Nombre:	iCantReal ()
Descripción:	Retorna la cantidad real de partículas con la que el usuario desea hacer la simulación.
Nombre:	Add(CFluidParticle arg_kpartitula)
Descripción:	Adiciona una partícula a la lista.
Nombre:	Draw()
Descripción:	Dibuja el sistema de Partículas.
Nombre:	Find_Neighbor(int aX,int &CantResult)
Descripción:	Retorna una lista de las partículas que son vecinas.

4.4.3 Paquete “Phisic Simulation”.

Tabla 4.4.5 Clase”CMathematicalModel”.

Nombre: CMathematicalModel	
Tipo de clase:	
Atributo	**Tipo**
m_ftime	float
m_cvgravity	*CVector3*
Para cada responsabilidad:	
Nombre:	Calculate_Velocity(CFluidParticle &arg_kpart1)

Descripción:	Calcula la velocidad a la que se mueve una partícula y cambia la velocidad de la partícula por la calculada.
Nombre	Calculate_Velocity1(CFluidParticle &arg_kpart1)
Descripción	Retorna la velocidad a la que se mueve una partícula.
Nombre:	Calculate_Module_Velocity(CFluidParticle &arg_kpart1)
Descripción:	Retorna el Modulo de la velocidad a la que se mueve una partícula.
Nombre:	Calculate_Position(CFluidParticle &arg_kpart1,CVector3 arg_cvaVel)
Descripción:	Calcula la Distancia a la que se va a mover una partícula.
Nombre:	CalculateC()
Descripción:	Retorna Una constante del método SPH
Nombre:	Calculate_RVariation(CFluidParticle &arg_kpart1,CFluidParticle &arg_pkart2)
Descripción:	Retorna la distancia en que se encuentran dos partículas.
Nombre:	Calculate_Q(CFluidParticle &arg_kpart1,CFluidParticle &arg_kpart2)
Descripción:	Retorna la variable Q
Nombre:	Calculate_Miu(CFluidParticle &arg_kpart1,CFluidParticle &arg_kpart2)
Descripción:	Retorna la variable MIU
Nombre:	CalculateKernel(CFluidParticle &arg_kpart1,CFluidParticle &arg_kpart2)
Descripción:	Retorna el gradiente del Kernel del método SPH
Nombre:	Artificial_Viscosity(CFluidParticle &arg_kpart1,CFluidParticle &arg_kpart2)
Descripción:	Retorna la Viscosidad Artificial.
Nombre:	Calculate_Aceleration(CFluidParticle&arg_kpart1,CFluidParticle &arg_kpart2)
Descripción:	Retorna la aceleración con que se mueve una partícula
Nombre:	Calculate_Density(CFluidParticle &arg_kpart1,CFluidParticle &arg_kpart2)
Descripción:	*Retorna la Densidad de una Partícula*
Nombre:	State_Equation(CFluidParticle &arg_kpart1)
Descripción:	*Retorna la presión de la partícula.*

Capítulo 5: Implementación del Sistema.

Introducción.

Esta etapa del proyecto constituye el paso del diseño de clases a la creación de componentes físicos, que se traducen en ficheros .h y .cpp correspondiente a la implementación en C++. También se dan a conocer los estándares de codificación.

5.1 Estándares de programación.

Nombre de los ficheros: Los nombres de los ficheros .h y .cpp utilizan las iniciales del nombre de la herramienta (STK).

ej.: **STK**NameOfUnits.cpp

Constantes: Las constantes se nombran con mayúsculas, utilizándose "_" para separar las palabras.

ej. const int MY_CONST_ZERO = 0;

Enumerados: Para los enumerados se utiliza el indicador "**E**" en el nombre del tipo, y en las constantes se utilizan las iníciales del nombre del enumerado. Las constantes van en mayúsculas.

ej1: enum **E**MyEnum {**ME**_VALUE, **ME**_OTHER_VALUE };

ej2: enum **E**Node**T**ype {**NT**_GEOMETRYNODE, **NT**_GROUPNODE...};

Estructuras: Se utiliza el indicador "**S**" para indicar que es una estructura. Las variables miembros de la estructura se nombrarán igual que en las clases, leer más adelante.

ej: struct **S**MyStruct {…};

Clases: Se utiliza el indicador "**C**" para indicar que es una clase. Ver más adelante la nomenclatura de las variables miembros.

ej: class **C**ClassName;

Interfaces: Se utiliza el indicador "**I**" para indicar que es una interfaz.

ej: **I**MyInterfaceName

Listas e iteradores de la *std*: Para los tipos de datos utilizados de la librería estándar de C++ (*vector*, *map*, *multimap*, etc.), se utiliza el indicador "**T**", con los sufijos **List**, **Map** y **MultiMap** según la estructura, así como el sufijo **Iter** para los iteradores. Además el nombre lleva el tipo de dato a almacenar en la estructura en cuestión:

*ej:*vector<CNode> **T**Node**List**;

TNodeList::iterator TNodeList**Iter**;

*ej:*map<> **T**Name**Map**;

TNameMap::iterator **T**name**MapIter**;

*ej:*multimap<> **T**Name**MultiMap**;

TNameMultiMap::iterator **T**Name**MultiMapIter**;

Declaración de variables: Los nombres de las variables comienzan con un identificador del tipo de dato al que correspondan, como se muestra a continuación. En el caso de que sean variables miembros de una clase, se le antepone el identificador "**m**_" (en minúscula), si son globales se les antepone el identificador "**g**_", y en caso de ser argumentos de algún método, se les antepone el prefijo "**arg**_".

Tipos simples:

ej: bool **b**VarName;

int **i**Name;

unsigned int **ui**Name;

float **f**Name;

char **c**Name; char* **ac**Name; // arreglo de caracteres char*

pcName; // puntero a un char char**

aacName; // bidimensional char**

apcName; // arreglo de punteros bool

m_b MemberVarName; // variable miembro de una estructura o clase char

g_c GlobalVarName; // variable global short **s**Name; void* **pv**Name;

Instancias de tipos creados:

ej: EMyEnumerated **e**Name;

SMyStructure **k**Name;

CClassName **k**ObjectName; // objeto de una clase

CClassName* **pk**Name; // puntero a objeto

CClassName* **ak**Name; // arreglo de objetos

CClassName* **m_ak**Name; // variable miembro de clase

IMyInterface* **pI**Name; // puntero interfaces

Métodos: En el caso de los métodos, se les antepone el identificador del tipo de dato de devolución, y en caso de no tenerlo (void), no se les antepone nada. Los constructores y destructores, como lo exigen los compiladores, llevan el nombre de la clase.

Constructor y destructor:

*ej:*CClassName (bool arg_bVarName, float& arg_fVarName);

~CClassName ();

Funciones:

ej: bool bFunction1 (…);

int* piFunction2 (…);

CClassName* pkFunction3 (…);

Procedimientos:

ej: void Procedure4 (…);

Métodos de acceso a miembros: Los métodos de acceso a los miembros de las clases no se nombrarán "Gets" ni "Sets", sino como los demás métodos, pero **con el nombre** de la variable a la que se accede y sin el prefijo "**m**_": *ej:* Para la siguiente variable, los métodos de acceso serían:

int **m**_**i**MyVar; //variable

int **i**MyVar(); //"Get"

void MyVar(int arg_**i**MyVar) //"Set"

int& **i**MyVar(); //"Get" y "Set"

5.2 Diagrama de despliegue.

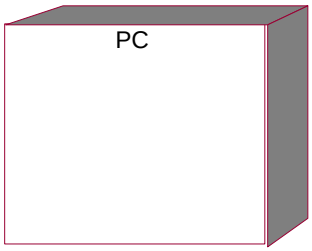


Fig. 5.2.1 Diagrama de despliegue.

5.3 Diagramas de componentes.

5.3.1 SubPaquetes de Componentes.

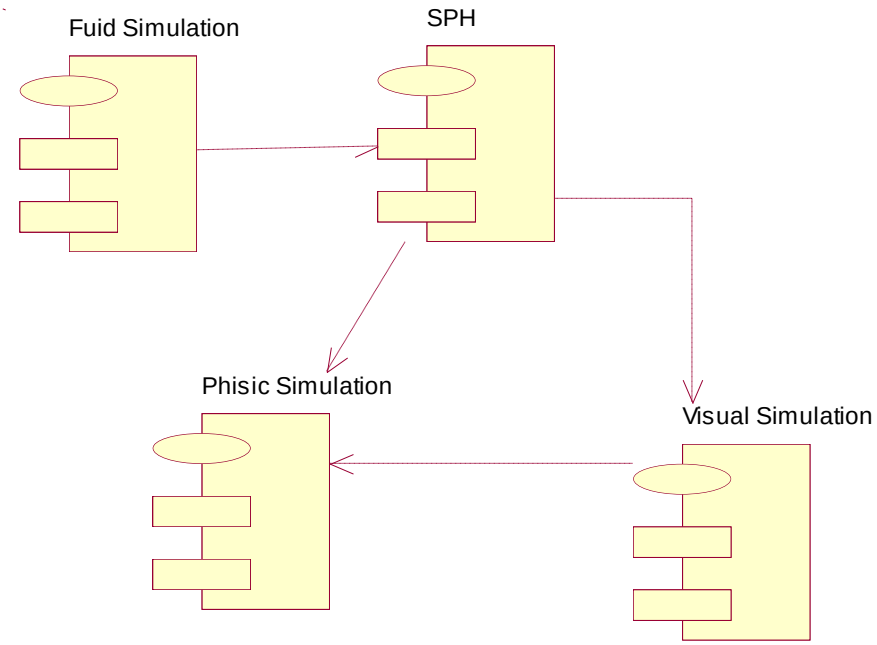


Fig. 5.3.1 Diagrama de SubPaquetes de componentes.

5.3.2 Diagrama de Componentes “Controller”.

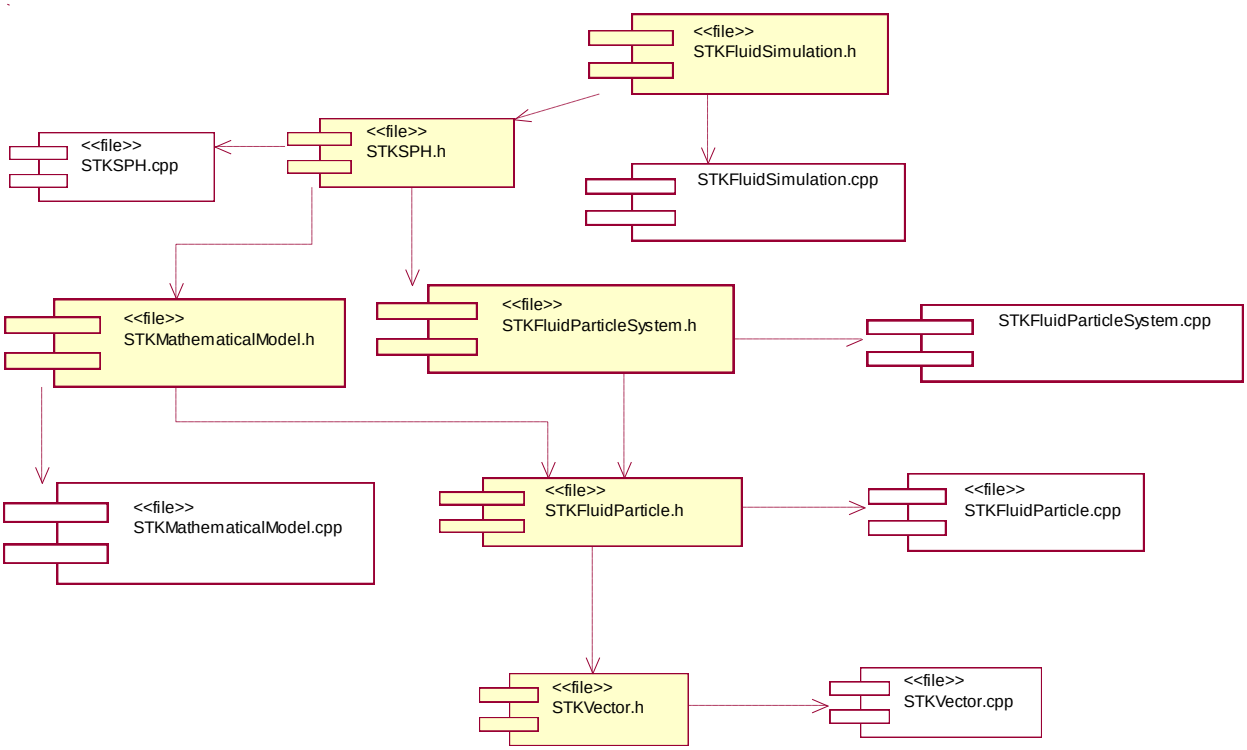


Fig. 5.3.2 Diagrama de componente “Controller”.

5.3.3 Diagrama de componentes “Phisic Simulation”.

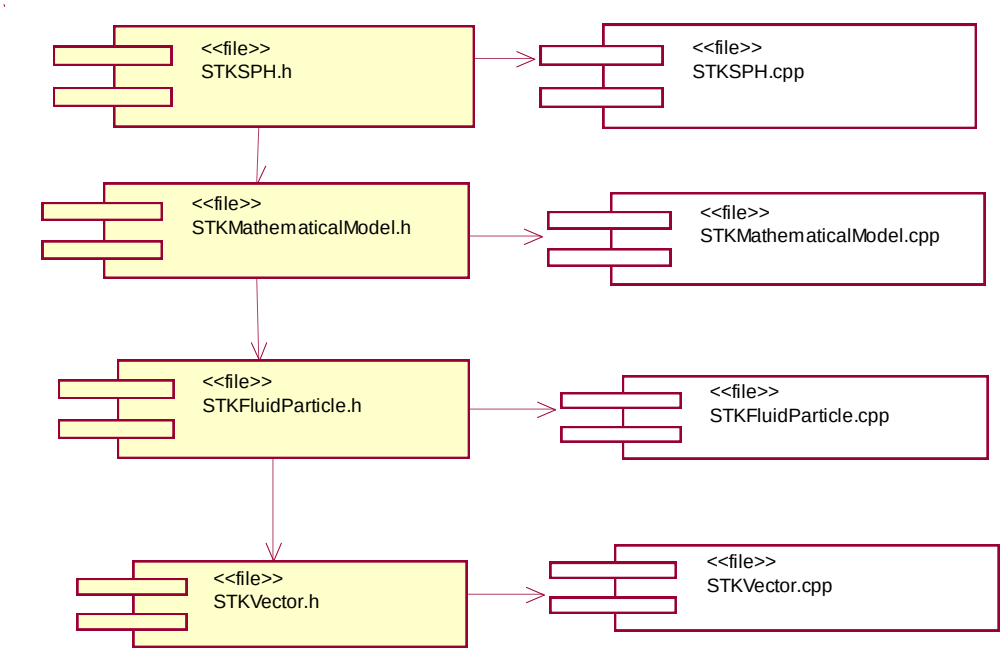


Fig. 5.3.3 Diagramas de Componentes de la Simulación Física.

5.3.4 Diagrama de componente “Visual Simulation”.

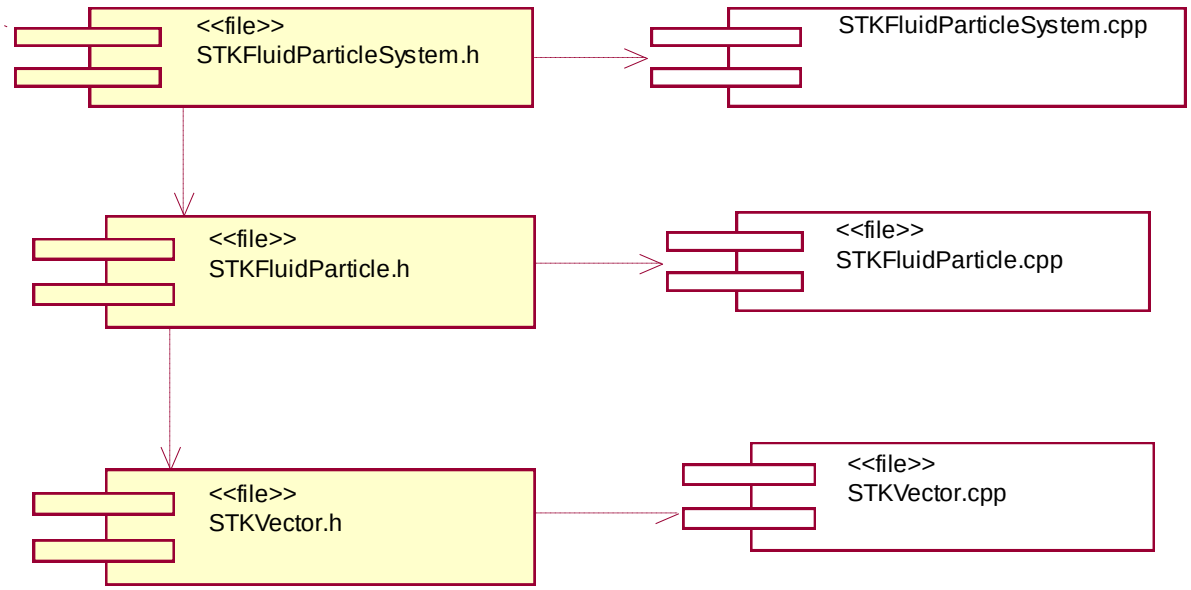


Fig. 5.3.4 Diagrama de componentes de la Simulación Visual.

Conclusiones Generales.

Para darle cumplimiento a los objetivos propuesto, en correspondencia con las exigencias del cliente, fue necesario hacer, en primer lugar, un estudio de la Mecánica y Dinámica de los fluidos y los diferentes métodos de Simulación Física y de Simulación Visual, A partir de esta investigación se propone una solución factible.

Para la construcción del sistema se utilizó el Proceso Unificado de Software (RUP) como metodología de desarrollo, a partir del cual se obtuvo un pequeño módulo que permite ver la efectividad de los métodos escogidos para la simulación de fluidos.

Además se le dio cumplimiento a las tareas planteadas para el logro del proyecto.

- Analizar herramientas de simulación de fluidos existentes en el mundo.
- Estudiar la dinámica y mecánica de los fluidos.
- Elaborar el análisis y diseño del efecto de simulación del fluido.
- Implementar el efecto de simulación del fluido.

Se obtuvo un efecto de simulación de fluido que con una cantidad de 150 partículas se ejecuta en tiempo real dependiendo su velocidad de la cantidad de vecinos que tenga una partícula en cada iteración.

Recomendaciones.

Se recomiendan los siguientes aspectos al trabajo:

- Estudiar otros métodos de Simulación Visual.
- Estudiar formas para optimizar la aplicación teniendo en cuenta la implementación de Estructuras de Datos Espaciales para la búsqueda de vecinos entre las partículas.
- Estudiar y llevar a cabo la futura integración del efecto en la herramienta Scene Toolkit(STK).
- Lograr que el fluido interactúen con objetos deformables.
- Profundizar en las condiciones fronteras por las que se mueve el fluido.

Referencias bibliográficas

1. **U.Meier, C. Monserrat, M. Alcañiz,MC. Juan, V. Grau, JA. Gil.** *Simulación Quirúrgica.* [ed.] Medical Image Computing Laboratory(MedlCLab). s.l. : Universidad Politécnica de Valencia, 2003.

2. **Madrid(UCM), Universidad Complutense.** [Online] (Curso 2007-2008). http://www.ucm.es/info/catmosf/docencia/mecanica2/fluidos_1.PDF.

3. **Suárez López Joaquin, Martínez Abela Fernando, Puertas Agudo Jerónimo.** [Online] 2005. http://www.google.com.cu/books?id=TrKVuY_NvlwC&pg=PT36&dq=definicion+de+fluido&sig=ACfU3U3u7BzuCUJQpJI8FTD4vpYtsJ1zbg#PPT36,M1.

4. **Simon Premoze, Tolga Tasdizen, James Bigler, Aaron Lefohn, and Ross T. Whitaker.** *Particle-based simulation of fluids.* s.l. : EUROGRAPHICS, 22(3), 2003.

5. **David Halliday, Robert Resnick, Kenneth S. Krane.** *"Física" .* Cuarta Edición. La Habana : Félix Varela,2003. Vol. 1.

6. **Nguyen D. Q., Fedkiw R., Jensen H. W.** Physically based modeling and animation of fire. . [Online] 2002. http://graphics.ucsd.edu/~henrik/papers/fire/.

7. **J., Harris. Mark.** Fast fluid dynamics simulation on the gpu. [Online] 2004. http://download.developer.nvidia.com/developer/SDK/Individual_Samples/DEMOS/OpenGL/src/gpgpu_fluid/docs/GPU_Gems_Fluids_Chapter.pdf..

8. **Stam, Jos.** *Stable Fluids. .*

9. **Teschner, Matthias M¨uller. Simon Schirm .Matthias.** *Interactive Blood Simulation for Virtual Surgery Based on Smoothed Particle Hydrodynamics.* s.l. : Computer Graphics Laboratory - ETH Zurich.

10. **Richard, S. Bezemer.** *Real Time Simulation of Natural Phenomenon Using Particle Systems and Noise Functions.* 2002.

11. **Miller, Alan Murta. James.** *Modelling and Rendering Liquids in Motion.* s.l. : Univercity of Manchester: Departament of Computers Science.

12. [Online] http://www.eui.upm.es/%7Efmartin/Docencia/A3D/Teoria/Tema3/index.htm.

13. **Youquan Liu, Xuehui Liu, and Enhua Wu.** *Real-time 3d fluid simulation on gpu with complex obstacles. Proceedings of the Computer Graphics and Applications.* 2004.

14. **Morris., J.P.** *An overview of the method of smoothed particle hydrodynamics.* Universitat Kaiserslautern. Internal Report : s.n., 1995.

15. **Monaghan, J. J.** *Simulating Free Surface Flows with SPH.* s.l. : Journal of Computational Physics, 1994:110:399–406.

16. **Goldstein, Herbert.** *Mecánica Clásica.* s.l. : Reverté, 1987.

17. **Robert Resnick, David Halliday, and Kenneth Krane.** *Física.* s.l. : CECSA, 1998.

Bibliografía consultada

1. **U.Meier, C. Monserrat, M. Alcañiz,MC. Juan, V. Grau, JA. Gil.** *Simulación Quirúrgica.* [ed.] Medical Image Computing Laboratory(MedlCLab). s.l. : Universidad Politécnica de Valencia, 2003.

2. **Madrid(UCM), Universidad Complutense.** [Online] (Curso 2007-2008). http://www.ucm.es/info/catmosf/docencia/mecanica2/fluidos_1.PDF.

3. **Suárez López Joaquin, Martínez Abela Fernando, Puertas Agudo Jerónimo.** [Online] 2005. http://www.google.com.cu/books?id=TrKVuY_NvlwC&pg=PT36&dq=definicion+de+fluido&sig=ACfU3U3u7BzuCUJQpJI8FTD4vpYtsJ1zbg#PPT36,M1.

4. **Simon Premoze, Tolga Tasdizen, James Bigler, Aaron Lefohn, and Ross T. Whitaker.** *Particle-based simulation of fluids.* s.l. : EUROGRAPHICS, 22(3), 2003.

5. **David Halliday, Robert Resnick, Kenneth S. Krane.** *"Física" .* Cuarta Edición. La Habana : Félix Varela,2003. Vol. 1.

6. **Nguyen D. Q., Fedkiw R., Jensen H. W.** Physically based modeling and animation of fire. . [Online] 2002. http://graphics.ucsd.edu/~henrik/papers/fire/.

7. **J., Harris. Mark.** Fast fluid dynamics simulation on the gpu. [Online] 2004. http://download.developer.nvidia.com/developer/SDK/Individual_Samples/DEMOS/OpenGL/src/gpgpu_fluid/docs/GPU_Gems_Fluids_Chapter.pdf..

8. **Stam, Jos.** *Stable Fluids. .*

9. **Teschner, Matthias M¨uller. Simon Schirm .Matthias.** *Interactive Blood Simulation for Virtual Surgery Based on Smoothed Particle Hydrodynamics.* s.l. : Computer Graphics Laboratory - ETH Zurich.

10. **Richard, S. Bezemer.** *Real Time Simulation of Natural Phenomenon Using Particle Systems and Noise Functions.* 2002.

11. **Miller, Alan Murta. James.** *Modelling and Rendering Liquids in Motion.* s.l. : Univercity of Manchester: Departament of Computers Science.

12. [Online] http://www.eui.upm.es/%7Efmartin/Docencia/A3D/Teoria/Tema3/index.htm.

13. **Youquan Liu, Xuehui Liu, and Enhua Wu.** *Real-time 3d fluid simulation on gpu with complex obstacles. Proceedings of the Computer Graphics and Applications.* 2004.

14. **Morris., J.P.** *An overview of the method of smoothed particle hydrodynamics.* Universitat Kaiserslautern. Internal Report : s.n., 1995.

15. **Monaghan, J. J.** *Simulating Free Surface Flows with SPH.* s.l. : Journal of Computational Physics, 1994:110:399–406.

16. **Goldstein, Herbert.** *Mecánica Clásica.* s.l. : Reverté, 1987.

17. **Robert Resnick, David Halliday, and Kenneth Krane.** *Física.* s.l. : CECSA, 1998.

18. *Las matemáticas de los fluidos: torbellinos, gotas y olas.* **Diego Córdoba, Marco Antonio Fontelos y José Luis Rodrigo.** 2005, LA GACETA DE LA RSME, Vol. 8.3 , pp. 53–83.

19. *Utilización del método sph en la simulación del sloshing.* **Pérez., S Abril.** Technical Report No. 822., s.l. : Universidad de la Rioja, 2005.

20. **Anderson., J.D.** *Computational Fluid Dynamics, The Basics with Applications.* s.l. : McGraw-Hill, 1995.

21. **Batchelor, G.K.** *An Introduction to Fluid Dynamics.* s.l. : Cambridge University, 1974.

22. **Kaufmann, Morgan.** *Real-Time Collision Detection.* s.l. : Christer Ericson, 2005.

23. **Quinn, David A. Fulk and Dennis W.** An analysis of 1-d smoothed particle hydrodynamics kernels. *Journal of Computational Physics.* 1996, 126(1), pp. 165–180.

24. **Garcia., Alejandro L.** *Numerical Methods for Physics.* s.l. : Prentice Hall, 1999.

25. *On criterions for smoothed particle hydrodynamics kernels in stable field.* **Xin, Jin Hongbin and Ding.** 202(2), 2005, Journal of Computational Physics, pp. 699–709.

26. **Belton, D.A. Jones and D.** Defence Science and Technology Organisation – Australia. [Online] 2006. http://dspace.dsto.defence.gov.au/dspace-oai/request] (Australia), Technical Report.

27. **Lars., Andersson.** *Real-time fluid dynamics for virtual surgery. Master's thesis.* s.l. : Chalmers University of Technology, 2005.

28. **Youquan Liu, Xuehui Liu, and Enhua Wu.** *Real-time 3d fluid simulation on gpu with complex obstacles. Proceedings of the Computer Graphics and Applications,.* s.l. : 12th Pacific Conference on (PG'04), 2004. IEEE Computer Society.

29. *Simulating Free Surface Flows with SPH.* **Monaghan., J. J.** 110, 1994, Journal of Computational Physics, pp. 399–406.

30. *Sph without a tensile instability.* **Monaghan., J. J.** 159(2), 2000, J. Comput. Phys., pp. 290–311.

31. *Smoothed particle hydrodynamics.* **Monaghan., J. J.** 68, 2005, Rep. Prog. Phys., pp. 1703–1759.

32. *Interactive blood simulation for virtual surgery based on smoothed particle hydrodynamics. .* **Matthias M¨uller, Simon Schirm, and Matthias Teschner.** 2(1), 2004., Technol. Health Care, , pp. 25–31.

Apéndices.

Índice de Figuras, Tablas y Ecuaciones.

Índice de Figuras.

Índice de Tablas.

Glosario de términos.

[I] El primer intento de tratamiento de superficies libre fue desarrollado a partir del método MAC (Marker- and- Cell). Marcadores Lagrangeanos que definen a los fluidos y su interface son utilizados para trazar el movimiento de las partículas de la superficie libre.

[II] El método de la características ha sido una idea muy utilizada para la integración del termino convectivo en la ecuación de transporte puro y goza de gran tradición en la Mecánica de Fluidos Computacional.

[III] La integración del salto de rana o (Leap Frog Integration) es muy usado en juegos por computadora y muy fácil para el trabajo con colisiones. Trabaja mejor que la integración con Euler, pero también pudieran probar con Verlet.

[IV] Vóxel o voxel (la palabra proviene de la contracción del término en inglés "volumetric pixel") es la unidad cúbica que compone un objeto tridimensional. Constituye la unidad mínima procesable de una matriz tridimensional y es, por tanto, el equivalente del píxel (o pixel) en un objeto 2D.

[V] Una superficie equipotencial es el lugar geométrico de los puntos de un campo escalar en los cuales el potencial de campo es constante. Las superficies equipotenciales pueden calcularse empleando la ecuación de Poisson.

El caso más sencillo puede ser el de un campo gravitatorio en el que hay una masa puntual: las superficies equipotenciales son esferas concéntricas alrededor de dicho punto. El trabajo realizado por esa masa siendo el potencial constante, será pues, por definición, cero. En el caso del campo magnético generado por un conductor rectilíneo, las superficies equipotenciales serán cilindros concéntricos cuyo eje será precisamente el del conductor. Las curvas de nivel de estos cilindros son las que generan las Líneas equipotenciales en el plano x-y.

[VI] Es un algoritmo de grafico por computadoras publicado en 1987 por Lorens y Cline para extraer los voxeles de una isosuperficie.

[VII] Es un algoritmo de grafico por computadora utilizado para renderear superficies implícitas este algoritmo es una mejora del Marching cube.