UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

Facultad de Ingeniería

Visualización 3D de partículas en tiempo real, en ambientes de cómputo de alto rendimiento.

TESIS QUE PARA OBTENER EL TÍTULO DE

Ingeniero en Computación

P r e s e n t a

Edgar Arce Ayala

Director de Tesis

José Luis Villarreal Benítez

Ciudad Universitaria – 2009

Reconocimientos

- *A mi madre Guadalupe, por todo su cariño, confianza y apoyo incondicional. Eres un ejemplo de vida y entereza para todos nosotros.*

- *A mi padre Jorge (QPD.), por sus consejos y ser una guía en mi desarrollo profesional.*

- *A mis hermanas Gina e Itzel, por ser una parte fundamental en mi vida. Flaca gracias por creer en mí y por todo tu apoyo que finalmente trajo como consecuencia este trabajo final. Munis eres genial, tienes personalidad y una chispa que espero nunca pierdas.*

- *A los motores que conducen y guían mi vida, mi esposa Nidia y mi pequeña Sara. Amor sabes lo que significas en mi vida, simplemente todo, gracias por estos años a tu lado y por darme a la peque que representa la luz de nuestra casa.*

- *A mis abuelos, tíos y primos que me han dado todo su afecto y amistad.*

- *A mis suegros Guillermina y Ricardo, así como a mis cuñados Riqui y Alma, por su amistad y permitirme ser parte de su familia.*

- *A la Facultad de Ingeniería y al Departamento de Visualización de la Dirección General de Servicios de Cómputo Académico por todas sus enseñanzas y el apoyo brindado para poder realizar este trabajo.*

- *A la Univesidad Nacional Autónoma de México por darme el privilegio de formarme como profesionista en la Máxima Casa de Estudios.*

- *Finalmente un agradecimiento muy especial al arquitecto de este trabajo, mi Director de Tesis José Luis Villarreal, por tenerme la confianza de ponerme este reto enfrente. Gracias José Luis por tu paciencia y consejos siempre estaré en deuda contigo.*

Visualización 3D de partículas en tiempo real, en ambientes de cómputo de alto rendimiento.

Edgar Arce Ayala
División de Ingeniería Eléctrica - Ingeniería en Computación

Director de Tesis
José Luis Villarreal Benítez
Departamento de Visualización, DCI-DGSCA

Resumen

La visualización de campos escalares y vectoriales permite el estudio de una gran variedad de fenómenos de flujos. Las soluciones de los campos (obtenidos por simulaciones, experimentos u observaciones) son muy costosas (computacionalmente o en instrumentos) y generalmente se realizan en modo *batch*; sin embargo, la visualización de estos conjuntos de datos también requiere cálculos que suelen ser complejos. Pero es posible lograr visualizaciones en tiempo real, gracias a nuevos algoritmos y tecnologías de cómputo. La visualización en tiempo real permite ganar intuición rápida a través de la retroalimentación instantánea, sobre el fenómeno en estudio y conseguir las condiciones iniciales de la visualización de forma interactiva, evitando retrasos por espera de una visualización inadecuada, realizada en modo batch.

Prefacio

En la UNAM y en general en México, existe una comunidad de científicos e ingenieros que trabajan Dinámica de Fluidos Computacionales (CFD) de gran escala (en ambientes de cómputo de alto rendimiento); generando grandes cantidades de datos para cada caso que estudian, los cuales pueden ser interpretados a través de visualizaciones en tiempo real.

Partiendo de esta premisa, en el presente trabajo se muestra un análisis muy completo en donde se evalúan las alternativas de visualización de partículas 3D, documentadas en la literatura especializada que sirvieron como estrategia de implementación con criterios de eficiencia y eficacia para desarrollar un **Sistema de Visualización 3D de Partículas en Tiempo Real**.

La literatura especializada presenta en sus resultados, las ventajas y desventajas de cada aproximación. Lo cuál, generalmente es un compromiso en el que los desarrolladores deciden qué aspecto sacrificar según los objetivos que se persigan en los tipos de casos en CFD.

A través de la experiencia de usuarios del DepVis (Departamento de Visualización) DGSCA UNAM, se tomaron estas decisiones para lograr un sistema que cubra las necesidades de los problemas de la comunidad universitaria y de problemas de importancia social.

Así mismo, se consideró cubrir las necesidades de visualización de los problemas clásicos en la materia, ya que éstos son bien conocidos y sirven para evaluar los resultados de un sistema nuevo, con los ya encontrados con otras aproximaciones reportadas en la literatura.

La implementación esta basada en la Programación Orientada a Objetos (C++) utilizando técnicas de cómputo de alto rendimiento, con un diseño modular que permite un fácil mantenimieto y reusabilidad del código, así como OpenGL para lograr el mejor rendimiento en cuanto al despligue Gráfico.

Resultados Esperados con el Sistema

i. Exploración y análisis de simulaciones de fluidos computacionales CFD, con fines científicos y de ingeniería. Por ejemplo: dispersión de partículas contaminantes.

ii. Una herramienta para explorar los conceptos básicos en CFD, a través de ejemplos y visualizaciones ilustrativas. Por ejemplo: tetrahedrización, path line, streak line, binarización, look-up table, métodos de integración.

Índice general

Parte I
FUNDAMENTOS

Capítulo 1

Introducción a la Visualización Científica

1.1. Principios de la Visualización

En los últimos años, los avances en la tecnología de la información han facilitado la obtención de grandes cantidades de datos provenientes principalmente de simulaciones computacionales e instrumentación. El estudio de los fenómenos naturales a través de estas herramientas modernas ha traído como consecuencia la necesidad de explotar esta gran cantidad información; sin embargo, estos datos por si solos no son útiles si no existe una forma fácil y eficiente de analizarlos e interpretarlos. Es aquí en donde la Visualización juega un papel fundamental.

1.1.1. La necesidad de utilizar la Visualización

Usualmente cuando se almacenan los datos se incluyen varios parámetros, resultando datos multidimensionales con altos niveles de complejidad, lo que representa que al buscar información relevante en ellos o una representación útil, es una tarea difícil. Con los sistemas actuales de administración de datos, solo es posible ver porciones muy pequeñas de los mismos. Si los datos son presentados textualmente, la cantidad de éstos que pueden ser mostrados están en el rango de algunos cientos de registros, pero esto se complica enormemente cuando se trata de conjuntos de datos que contengan millones de registros. No teniendo la posibilidad de explorar adecuadamente las grandes cantidades de datos que han sido coleccionados debido a su utilidad potencial, éstos se tornan inservibles y las bases de datos se convierten simplemente en "almacenes" de datos.

La Visualización facilita el entendimiento a pequeña y gran escala de características de los datos. Simplifica el análisis y la comunicación de modelos y conceptos. Emplea las potencialidades del sistema visual que es un buscador de patrones de extrema fuerza y sutileza. El hecho de que el resultado sea una imagen, posibilita una mayor comprensión, claridad y aprovechamiento de éste. Se estima que el 50 % de las neuronas está dedicado a la visión. Además, la densidad de información por unidad de área en una imagen es notablemente mayor a la de un texto. Por otro lado, la visualización permite observar lo que "no es posible ver directamente", debido, entre otras razones, al gran volumen de los datos o a que éstos no tengan una representación gráfica asociada.

Existe una especialización dentro de la Visualización denominada **Visualización Científica**, que ha evolucionado rápidamente con el paso del tiempo, por la relevancia que esta tiene en el desarrollo actual de la Investigación y la Ciencia. La Visualización Científica busca constantemente encontrar nuevas y más eficientes técnicas para el análisis y la interpretación de datos.

1.1.2. *Pipeline* de la Visualización

El *Pipeline* de la Visualización describe los pasos que se deben seguir para el procesamiento de los datos, hasta lograr la representación visual de los mismos. La siguiente figura, ilustra los pasos que conforman al pipeline.

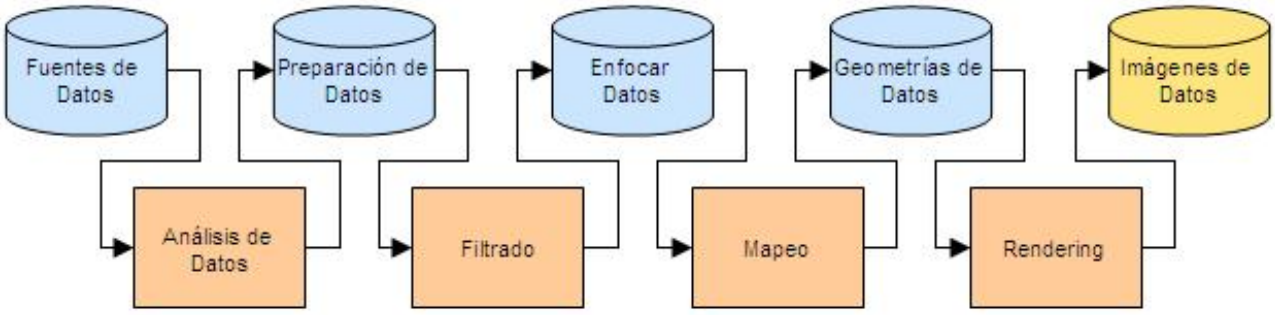


Figura: *Pipeline de la Visualización*

- **Análisis de datos**: En esta etapa, los datos son preparados para la visualización (p. ej., interpolaciones para datos faltantes, correciones de mediciones erróneas, etc.). Este paso, generalmente está centrado en los equipos de cómputo y hay una pequeña o casi nula interacción con el usuario.
- **Filtrado**: En esta etapa, de acuerdo a los objetivos que se buscan con la visualización, se hace una selección de datos a ser visualizados. Usualmente este paso, está centrado en el Usuario.
- **Mapeo**: En esta etapa, los datos seleccionados, son mapeados a geometrías primitivas (puntos, lineas, etc.), así como sus atributos (color, posición, tamaño). Este es el paso más crítico para lograr los objetivos de una visualización.
- **Render**: Esta es la última etapa en donde finalmente los datos geométricos son transformados en imágenes.

Un punto importante a mencionar del pipeline de la Visualización, es que al ser un procesamiento lineal, cada paso depende del resultado del anterior, y al ser volúmenes de información muy grandes que pueden provenir de diversas fuentes, el tiempo de inicio a fin puede ser muy grande y el costo computacional muy alto, por lo que la Visualización

demanda constantemente técnicas y metodologías más rápidas y eficientes.

1.2. Importancia de la Visualización Científica

La visualización científica posibilita reconocer patrones de comportamiento de los datos, ver en una sola imagen o en una secuencia de éstas (animación) una gran cantidad de datos y facilita la comprensión de algunos conceptos, sobre todo de tipo abstracto. Por ejemplo, si se diera el caso de que tuviéramos una serie de datos, obtenidos de una estación meteorológica, al mostrarlos en forma de tabla, sería muy difícil distinguir a simple vista alguna relación entre los mismos, pero al conformar una gráfica de los valores veríamos si siguen cierto patrón de comportamiento.

Entre las ventajas de la visualización científica está el poder representar datos de varias dimensiones o variables, lográndose visualizar cuatro o más variables al mismo tiempo apoyándose en algunos métodos. Por ejemplo, el plano cartesiano puede mostrar dos variables, si agregamos otro, podremos ver tres, si agregamos colores, tendremos cuatro, si se hace alguna animación de la gráfica podremos apreciar una quinta variable o dimensión. Otra gran ventaja es la independencia del lenguaje, ya que la idea principal del problema está representada de forma gráfica.

También posibilita a las personas la interacción directa con los datos. La visualización puede ser hecha sin mayor dificultad en datos no homogéneos o que no se conozca en detalle su estructura. La exploración visual es intuitiva, no requiere de complicados conocimientos matemáticos, estadísticos o de otra índole. Otra gran ventaja consiste en la gran cantidad de información que puede ser rápidamente interpretado, y generar conocimiento de los fenómenos estudiados.

1.2.1. Propósitos de la Visualización Científica

La visualización científica se emplea con varios propósitos que se agrupan en tres modos de uso muy particulares que se pueden describir de esta forma:

Análisis exploratorio.

Se tiene un conjunto de datos sin una hipótesis específica. Estos datos se someten a un proceso de búsqueda interactiva de información que va a arrojar como resultado una visualización que soporte una hipótesis sobre el conjunto de datos.

Análisis confirmativo.

Se tiene un conjunto de datos sobre los que se plantea una hipótesis. Se realiza un procesamiento de dichos datos que genera una visualización mediante la cual se pueda validar o refutar la hipótesis que se tenía de ellos.

Presentación de un resultado.

Se conocen hechos específicos sobre los datos, se realiza un proceso que de como resultado una visualización que enfatice en la veracidad de dichos hechos.

1.3. Evolución de la Visualización Científica

Uno de los primeros pasos dentro de la visualización científica fue la creación de gráficas y modelos en dos dimensiones. Posteriormente, estas evolucionaron hacia modelos de tres, cuatro, cinco ó más dimensiones. Los modelos tridimensionales se iniciaron como objetos construidos con líneas, a los que posteriormente se les dio volumen por medio de la generación de imágenes o renderización (del inglés digujado), que no es más que la generación de una imagen a partir de un modelo. Este modelo es una descripción de un objeto tridimensional en un lenguaje bien definido o en una estructura de datos.

Más tarde se diseñaron métodos para manipular, modificar y animar estos modelos. Algunos métodos gráficos permiten, mediante la manipulación del plano, representar más de dos variables en un solo plano.

El primer impulso por utilizar las computadoras en la representación de datos fue en la década de 1960, de ahí en adelante la visualización científica ha ido de la mano del desarrollo de la computación.

A principios de los noventa las técnicas de visualización científica empezaron a atraer a una serie de científicos, ingenieros, médicos, entre otros, al estudio de la amplia variedad de conjuntos de datos disponibles. El desarrollo de las capacidades de cómputo dio posibilidad de mejorar los resultados y facilitar la visualización. Se vuelve común usar una computadora de gran capacidad y rendimiento para el procesamiento o generación de grandes cantidades de información y después una estación de trabajo para la presentación gráfica de los resultados, de esta forma se pueden aprovechar las ventajas de ambos equipos. Posteriormente, las imágenes obtenidas pueden almacenarse en discos o grabarse en videos para facilitar su distribución y presentación.

Uno de los últimos avances en visualización es el uso de la realidad virtual. Por medio de ésta se puede generar "fácilmente" una representación tridimensional de objetos o lugares que no se podrá lograr con una computadora y una pantalla de video normal. Por ejemplo se pude ver el funcionamiento de órganos o sistemas animales desde dentro del mismo, o se pueden tener representaciones en tres dimensiones de objetos de cuatro dimensiones, sin que se vean reducidos a dos al presentarlos en una pantalla normal.

Figura 1.1: *Realidad Virtual en la Visualización Científica*

1.4. Problemáticas de la Visualización Científica

Las capacidades de manejo y visualización de estos datos, aún es muy inferior a la demanda de científicos e ingenieros. La dificultad radica en i) la gran escala de los datos, ii) la fusión de datos generados por diferentes grupos de investigación que comparten intereses, iii) la complejidad de los datos por su multidimensionalidad y por la naturaleza de los fenómenos, iv) la variedad y poco conocimiento que sobre estos se tiene.

Para responder a esta problemática, se han abordado las visualizaciones con nuevas estrategias que llevan hacia posibilidades de manejo de estos datos a través de imágenes y la capacidad de adquirir información y entendimiento para seguir avanzando en el conocimiento.

La solución al problema de la visualización de datos de gran escala requiere de una aproximación de sistemas integrados. A bajo nivel, es crítico desarrollar mecanismos de manejo de datos coherentemente de forma eficiente, que soporten diversas representaciones de los datos y patrones de acceso típicos para los cálculos de visualización (principalmente para datos dispersos geográficamente). Así que se necesitan nuevos diseños en las áreas de a) bases de datos escalables, b) sistemas de almacenamiento jerárquico, y c) I/O paralelos.

1.4.1. Estrategias a problemáticas de la Visualización Científica

A un nivel alto, las estrategias se han centrado en visualización paralela escalable, interfaces de usuarios, aproximaciones de visualización en tiempo real.

Visualización paralela escalable

Debido a la necesidad de visualizar datos de gran escala en la resolución más alta posible, se hace indispensable recurrir al poder de procesamiento y memoria de computadoras paralelas. Un problema de esta aproximación es que cada paso del *pipe* de visualización debe ser paralelizado para evitar cuellos de botella.

Interfaces de usuarios

Los costos de las visualizaciones de datos de gran escala son muy altos, por lo que es necesario diseñar herramientas que permitan usar la experiencia que se ha ganado con las técnicas tradicionales y lo que se puede aprender de los datos con visualizaciones tradicionales y las de gran escala. Dentro de las alternativas se cuenta las interfaces de usuarios y los programas para visualizaciones inteligentes; así como la posibilidad de reuso y compartir la experiencia a partir de mapas guías, macros y herramientas visuales para representar las operaciones exitosas.

Como se puede apreciar de las estrategias expuestas, la visualización de datos de gran escala requiere de una buen diseño que aproveche todas estas herramientas en diferentes fases de su *pipe*.

Visualización en tiempo real

Los costos computacionales de las simulaciones son muy altos, así como los cálculos necesarios para las visualizaciones de los datos resultantes (*ver tabla anexa*). Tradicionalmente estas visualizaciones son llevadas a cabo en modo batch; definiendo los parámetros

y características particulares de una visualización (puntos de vista de la cámara, posición inicial de los inyectores de partículas, etc.), se realizan todos los cálculos necesarios para la visualización, sin ningún control por parte del investigador; posteriormente se forma una animación predefinida con estos cálculos y es hasta este momento en el que el investigador visualiza el resultado y decide los cambios de los parámetros y características mencionados, para mejorar los resultados de la visualización.

Tamaño de los problemas en visualización. Modelación computacional.

Número de elementos	Archivo de malla	+	Archivo de solución	=	Total por cada tiempo	*Número de pasos	=	Total por simulación
250×10^3	4 MB		5 MB		9 MB	5000		135 GB
900×10^3	16 MB		20 MB		36 MB	90000		3240 GB
1.3×10^6	23 MB		29 MB		52 MB	1450		226 GB
2.8×10^6	45 MB		56 MB		101 MB	1000		101 GB
3.2×10^6	53 MB		66 MB		119 MB	10000		1190 GB

La visualización en tiempo real permite interactuar durante los cálculos de mapeos de datos a imágenes, cambiando parámetros y características particulares de la visualización. Permitiendo así, tomar decisiones para mejorar los resultados de la visualización. Esta estrategia evita cálculos innecesarios y una mejor interacción con los datos; además de ganar intuición y experiencia para las futuras visualizaciones.

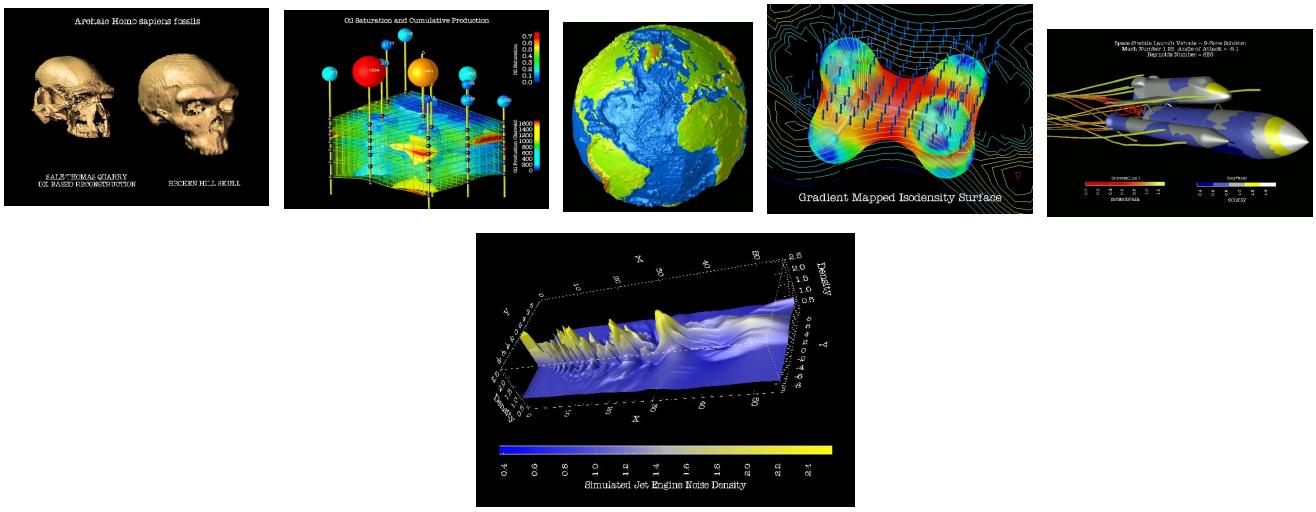
Para lograr visualizaciones en tiempo real, es necesario implementar estrategias de cómputo de alto rendimiento que mejoren el desempeño en todas las faces; desde el acceso y carga de datos, los algoritmos para los cálculos visuales, hasta los despliegues gráficos (3D) y las herramientas de interacción.

1.5. Aplicaciones de la Visualización Científica

Debido a la gran utilidad que tiene la visualización científica, existe una amplia gama de aplicaciones, para la investigación como en cualquier otra área.

Algunos campos específicos donde se aplica la visualización son:

- Medicina
- Geofísica
- Geografía
- Bioquímica
- Dinámica de Fluidos



Figuras 1.2: *Imágenes de visualizaciones en diferentes campos de la Visualización Científica*

La visualización científica es usada en la astronomía para visualizar objetos que se sabe que existen pero no han sido observados directamente como los Agujeros Negros. Se utiliza en la medicina para mostrar el crecimiento de las células, para asistir en diagnósticos médicos. Un ejemplo común de aplicación está en los programas del estado del tiempo, los cuales combinan datos atmosféricos con colores y otra serie de parámetros específicos de la visualización científica. En la Química se aprecian varios usos, como la representación de moléculas y/o estructuras moleculares. Otras aplicaciones importantes se encuentran en la rama de la vulcanología, como la visualización de la actividad sísmica en un área.

Fuera del área de la investigación, la visualización tiene aplicaciones en la ingeniería y en el diseño (diseño de ropa, diseño industrial, diseño de automóviles y aviones), la ingeniería genética, la exploración mineral y de combustibles, la animación para la producción de efectos especiales en la industria audiovisual, entre otros campos.

Capítulo 2

Simulaciones de fluidos

En México y especialmente en la UNAM, se estudian fenómenos naturales en los centros de investigación. La gran mayoría de ellos pertenecen al grupo de la Mecánica ó Dinámica de Fluidos, y se vuelve necesario el uso de simulaciones para analizar su comportamiento.

La ciencia moderna, permite que estos modelos puedan ser representados y calculados por métodos computacionales utilizando a la Visualización como un mecanismo para la interpretación y entendimiento de los mismos. Es por ello que se vuelve importante conocer a grandes razgos, el tipo de problemas que se pueden estudiar con la Dinámica de Fluidos, sus propiedades, y sobre todo, el papel fundamental que juega la Dinámica de Fluidos Computacional (CFD) en la simulación de fluidos que sirven de insumo para la Visualización Científica.

2.1. Principios de la Simulación de fluidos

Existen diferentes alternativas a la hora de simular un fluido. La opción que se escoja dependerá de los requisitos de la aplicación, teniendo que decidir en primer lugar si se quiere enfocar la simulación a una aplicación gráfica (juegos, cine), o a una simulación física (ingeniería, investigación, etc.).

Cuando basta con que la simulación "parezca" un fluido (en una aplicación orientada a gráficos), muchas veces es preferible recurrir a técnicas procedurales más sencillas, más eficientes y que permiten un mayor control de la simulación, en lugar de la dinámica de fluidos. Esto último es importante, ya que hay que tener en cuenta que en una simulación únicamente gobernada por leyes físicas, es difícil predecir y controlar el resultado de la simulación.

La Visualización Científica utiliza a la silumación física como entrada, por lo que en este punto, nos vamos a centrar únicamente en explicar los principios básicos de la simulación física.

2.1.1. Objetivos de la Simulación física

Los métodos de simulación física proporcionan una herramienta para poder implementar en una computadora simulaciones de fenómenos físicos regidos por ecuaciones conocidas. En el fondo, estos métodos no son más que técnicas para implementar en un programa de computadora un sistema de ecuaciones diferenciales, que son las más

habituales en la física y que permiten obtener aproximaciones numéricas en lugar de tener que desarrollar soluciones analíticas (ya que para la mayoría de los casos reales éstas no existen) y, lo que también es muy importante, permiten integrar esta solución en el tiempo, pudiendo obtener una secuencia temporal del estado de las variables que rigen el sistema.

En el caso concreto de la simulación física de fluidos, normalmente se tratan de implementar las ecuaciones de ***Navier-Stokes*** y, por tanto, el objetivo se centra en determinar las *velocidades y presiones del fluido, así como su variación a lo largo del tiempo*, que son las características que se analizan con estas ecuaciones.

2.1.2. Características deseables de una Simulación física

No todos los métodos son igual de válidos, y habrá que escoger en cada caso el que más se adecue a los requisitos del sistema. En general, estos son los factores que se deben tener en cuenta:

- **Eficiencia**: en muchos casos interesa que el método sea lo más rápido posible, para poder generar animaciones en tiempo real. En este caso, la eficiencia hace que se vean afectados otros factores, como la precisión y la estabilidad.

- **Estabilidad**: no todos los métodos pueden garantizar la estabilidad de la simulación. Cuando una simulación se vuelve inestable, el resultado de la simulación diverge, perdiendo por completo el sentido de la simulación, al alejarse totalmente de los resultados esperados. Un sistema inestable provocará efectos impredecibles.

- **Precisión**: los métodos numéricos implican siempre aproximaciones, por lo que en la mayoría de los casos no podemos asegurar que nuestra solución sea exacta. La desviación entre los valores que se desean obtener y los que se obtienen realmente se denomina disipación numérica, y en muchas aplicaciones es un objetivo prioritario minimizarla en la medida de lo posible.

- **Control**: si se deja a una simulación física "correr" sin más, es difícil predecir su comportamiento. Hay aplicaciones en las que puede interesar dirigir de alguna manera la simulación para producir un efecto visual concreto. En estos casos, es posible que se tengan que introducir otros factores que complementen la simulación física para aumentar el control del usuario. Un ejemplo de ello es la **Visualización en Tiempo Real**.

Teniendo como punto de partida estas características deseables, en los siguientes apartados se hace un análisis a grandes razgos de las técnicas y métodos computacionales que se han desarrollado en la Dinámica de Fluidos Computacional (CFD), así como en el comportamiento y características físicas que son de interés en el estudio de los fluidos para la Visualización Científica.

2.2. Principios básicos de la Dinámica de fluidos

Un **fluido** se define como una sustancia que se deforma continuamente bajo la aplicación de un esfuerzo de corte (tangencial). Éstos a su vez se clasifican por su estado, en líquidos y gases.

La **Dinámica de Fluidos** en la rama de la mecánica de medios continuos (que a su vez es una rama de la física), que estudia el movimiento de los *fluidos* así como las fuerzas que lo provocan. La característica fundamental que los define, es su incapacidad para resistir esfuerzos (lo que provoca que carezcan de una forma definida). También estudia las interacciones entre el *fluido* y el contorno que lo limita.

Hipótesis del medio continuo

Uno de los cimientos de la dinámica de fluidos es la hipótesis del medio continuo, que consiste en asumir que el fluido es continuo a lo largo de todo el volumen que ocupa. Esto simplifica mucho el problema, ya que se puede asumir que todas las magnitudes (densidad, temperatura, etc.) del fluido que se quieren estudiar van a regirse siempre por funciones continuas.

2.2.1. Propiedades de los fluidos

Estas son algunas de las principales propiedades de los fluidos.

- **Viscosidad**: Esta propiedad, esta ligada a la resistencia que opone un fluido a deformarse continuamente cuando se le somete a un esfuerzo de corte.
- **Compresibilidad**: La compresibilidad representa la relación entre los cambios de volumen y los cambios de presión a los que está sometido un fluido.
- **Presión de vapor**: Está definina como la presión para una temperatura dada, en la que la fase líquida y el vapor se encuentran en equilibrio dinámico. Cuando un líquido disminuye su presión hasta un punto que comienza un estado de ebullición se dice que ha alcanzado la presión de vapor.
- **Tensión superficial**: En física se define a la tensión superficial de un líquido como la cantidad de energía necesaria para aumentar su superficie por unidad de area. Esta propiedad produce el fenómeno por el cual la superficie de un líquido tiende a comportarse como si fuera una delgada película elástica.
- **Capa límite**: La capa límite o capa fronteriza de un fluido es la zona donde el movimiento de éste es perturbado por la presencia de un sólido con el que está en contacto. La capa límite se entiende como aquella en la que la velocidad del fluido respecto al sólido en movimiento varía desde cero hasta el 99 % de la velocidad de la corriente no perturbada.

2.2.2. Clasificación de los fluidos

Los diversos tipos de problemas encontrados en la dinámica de fluidos pueden ser clasificados en función de la observación de las características físicas de sus flujos. Con base en ello, el siguiente diagrama muestra una clasificación de los tipos de fluidos que son los más utilizados en el desarrollo de simulaciones de CFD, y su visualización científica.

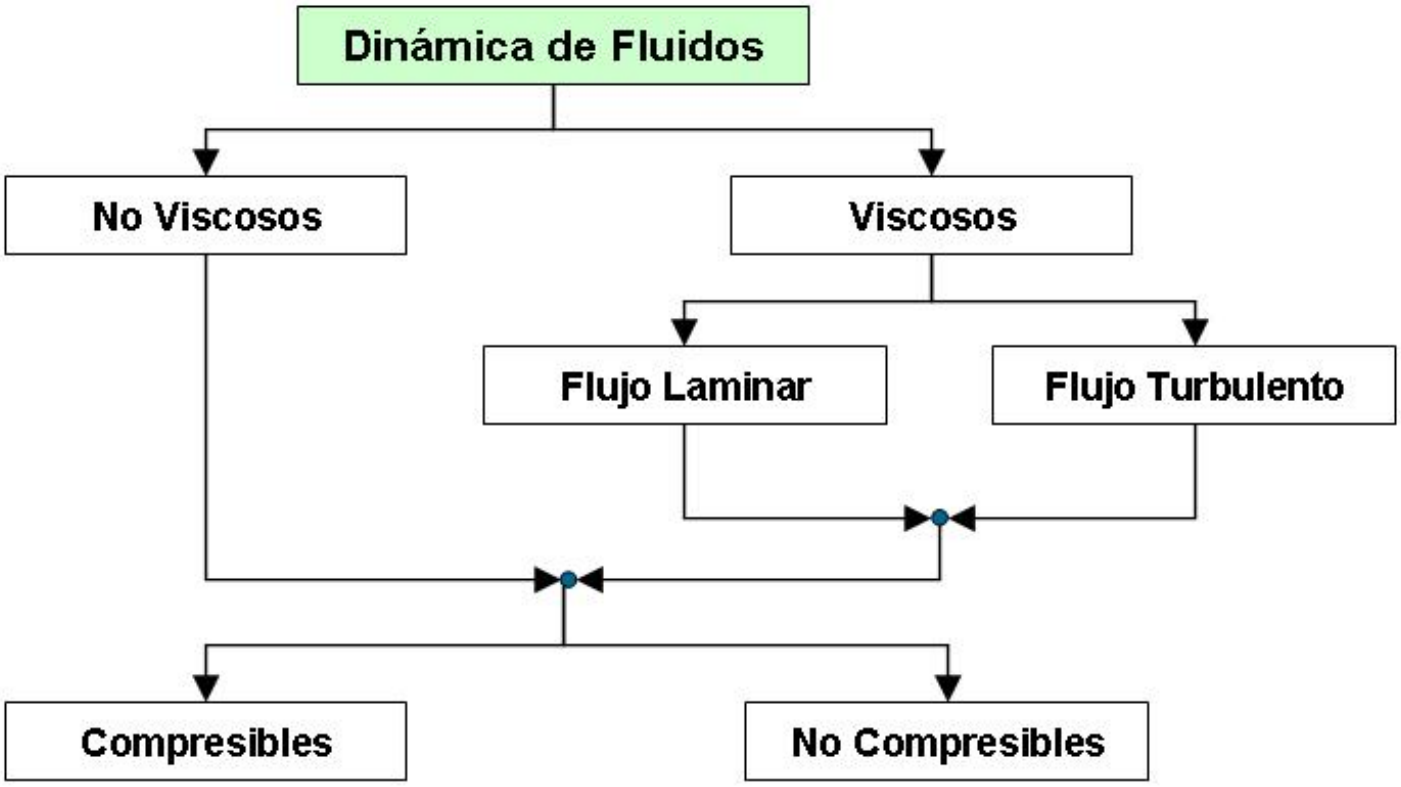


Figura 2.1: *Diagrama de clasificación de fluidos*

- **Fluidos Compresible - No Compresibles**: Todos los fluidos son compresibles en mayor o menor medida, por lo que en realidad cuando hablamos de fluidos no compresibles nos referimos a fluidos en los que esta propiedad es despreciable.

  Un fluido es más compresible cuanto mayor es el cambio que sufre en su densidad al aplicarle una presión externa. Por tanto, los gases son fluidos muy compresibles, mientras que los líquidos en general se suelen considerar no compresibles.

- **Fluidos Viscosos - No Viscosos**: Un fluido es viscoso si las fuerzas de roce internas son apreciables. Otra forma de decirlo es que disipa una cantidad apreciable de energía debido al roce. Si la viscosidad del fluido no juega un papel importante en su movimiento, se le llama no-viscoso.

- **Fluidos Viscosos con flujo Laminar**: Es uno de los dos tipos principales de los flujos en un fluido. Se llama flujo laminar o corriente laminar, al tipo de movimiento de un fluido que es perfectamente ordenado, de manera que el fluido se mueve en láminas paralelas, suave, sin entremezclarse si la corriente tiene lugar entre dos planos paralelos. Se dice que este flujo es aerodinámico. En el flujo aerodinámico, cada partícula de fluido sigue una trayectoria suave, llamada línea de corriente.

- **Fluidos Viscosos con flujo Turbulento**: En mecánica de fluidos, se llama flujo turbulento o corriente turbulenta al movimiento de un fluido que se da en forma caótica, en que las partículas se mueven desordenadamente y las trayectorias de las partículas se encuentran formando remolinos.

Figura 2.2: *El humo de cigarro comienza como flujo laminar y termina como turbulento.*

Existen otras dos clasificaciones para los fluidos que son importantes y se mencionan a continuación:

- **Flujos estacionarios - no estacionarios**: Se dice que un fluido es estacionario, si la velocidad (como vector) del fluido en cualquier punto dado es constante en el tiempo. No significa que la velocidad sea la misma en todos los puntos del fluido. Si el flujo no cumple esta propiedad, se le llama no-estacionario.

- **Flujos newtonianos - no newtonianos**: Un fluido newtoniano es aquel en el que su viscosidad no varía en función de la tensión que se le aplica. Por tanto, su viscosidad es un valor constante.
  En los fluidos no newtonianos, la viscosidad no es constante, por lo que podríamos decir que en cierto modo se comportan como sólidos cuando se les aplica una presión, y como líquidos cuando la presión es menor.

2.2.3. Estudio del comportamiento de los fluidos

El estudio de los fluidos consiste en conocer *las velocidades, presiones y fuerzas internas* de las partículas que forman el fluido, en función de las fuerzas externas que se aplican sobre él, y del contacto con otros medios (por ejemplo, el recipiente que contiene el fluido).

Cuando hablamos de contacto con otros medios, destaca la interacción entre el fluido y un sólido, teniendo en cuenta que el sólido se puede encontrar estático o en movimiento, y que además, en el caso más complejo, el sólido podría ser deformable. Este último caso podría dar lugar a una variación del volumen disponible para el fluido, lo cual en algunos casos (como en los gases) obliga también al estudio de las variaciones de densidad del fluido.

En cualquier caso, el estudio de la dinámica del fluido puede hacerse bien de forma analítica o de forma numérica, siendo este último caso el que nos interesa desde el punto de vista de la visualización.

Tipos de Enfoques para el estudio del movimiento de los fluidos

Si se quiere conocer el movimiento de las partículas del fluido, se puede abordar el problema desde dos enfoques totalmente diferentes.

La primera idea se basa en estudiar el movimiento de cada partícula en concreto, siguiendo su recorrido según se desplaza por el fluido. Este sería el *enfoque lagrangiano*. (Fig. 2.3)

La segunda idea es elegir un punto concreto del espacio que abarca el fluido, y estudiar los movimientos que se producen debido a las partículas que atraviesan ese punto fijo. Este sería el *enfoque euleriano*.

En la siguiente figura se muestra que en el enfoque Lagrangiano (izquierda) los componentes de la discretización son las propias partículas del fluido. En el enfoque Euleriano (derecha) se discretiza el volumen que ocupa el fluido, y cada punto por el que pasen las partículas será un elemento discreto.

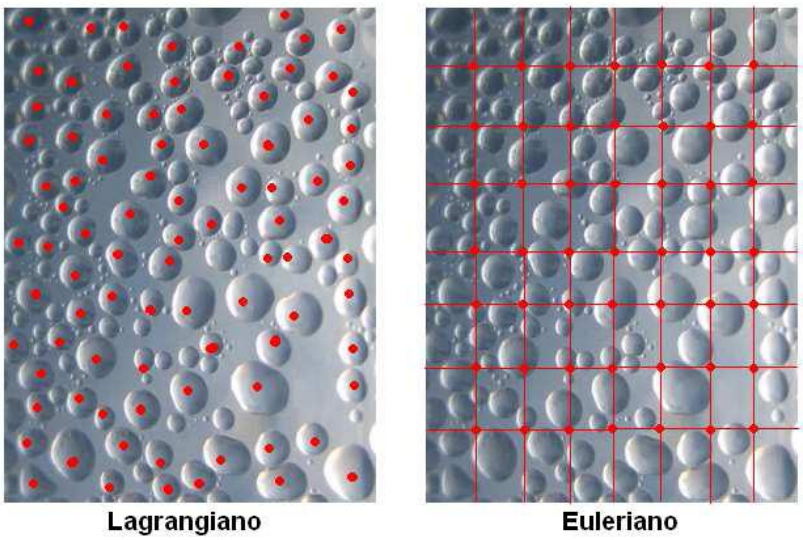


Figura 2.3: *Enfoque lagrangiano y euleriano*

Ecuaciones de Navier-Stokes

Las ecuaciones de Navier-Stokes son las ecuaciones más populares que describen el comportamiento dinámico de un fluido. Estas ecuanciones abarcan la atmósfera terrestre, las corrientes oceánicas y el flujo alrededor de vehículos o proyectiles y, en general, cualquier fenómeno de todo tipo de fluidos.

Derivan de aplicar los principios de conservación de la mecánica y de la termodinámica al fluido, de donde se obtiene una formulación integral que generalmente se suele transformar en una formulación diferencial más práctica.

El siguiente es un ejemplo en donde se muestra una ecuación de *Navier-Stokes* para el movimiento de fluidos no compresibles y densidad uniforme:

$$\frac{du}{dt} = F - \frac{\nabla p}{\rho} + v\nabla^2 u \tag{2.1}$$

En el lado izquierdo de la ecuación tenemos el movimiento (variación del desplazamiento en el tiempo), y en el derecho los siguientes términos:

- **F**: Fuerzas aplicadas al fluido.
- **p**: presión en el fluido.
- ρ: densidad del fluido.
- **v**: viscosidad del fluido.
- **u**: desplazamiento.

Aunque las ecuaciones de Navier-Stokes son las más utilizadas para el estudio de los fluidos, existen otras que también sirven para este fin, como las ecuaciones de *Bernoulli y Euler*.

Todas estas ecuaciones diferenciales, son la entrada para el desarrollo de los métodos y técnicas en la Dinámica de Fluidos Computacional (CFD), y son fundamentales en los objetivos que persigue la Simulación física de los fluidos.

2.3. Dinámica de Fluidos Computacional (CFD)

Como ya se ha mencionado, en la actualidad en muchos campos es imposible recurrir a soluciones analíticas debido a la tremenda complejidad de los sistemas que estudia la dinámica de fluidos, por lo que se recurre a soluciones numéricas que pueden ser procesadas por computadoras. Surge así una rama de la dinámica de fluidos denominada **dinámica de fluidos computacional, o CFD (por sus siglas en inglés)**, que se basa en aproximaciones numéricas de las ecuaciones físicas empleadas en la dinámica de fluidos (Navier-Stokes, Bernulli, etc.).

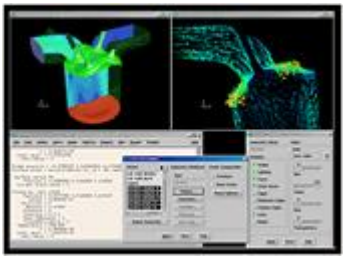
De esta forma se pueden definir tres niveles de estudio de la dinámica de fluidos computacional.

- Por un lado están los aspectos fundamentales que resultan en un conjunto de ecuaciones generalmente bastante complicadas que describen lo que físicamente ocurre en determinada situación. Puede decirse que este aspecto pertenece enteramente a la dinámica de fluidos como rama de la ciencia.

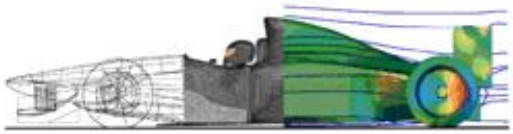
$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x_i}(\rho u_i) = S_m$$

$$\frac{\partial}{\partial t}(\rho u_i) + \frac{\partial}{\partial x_j}(\rho u_i u_j) = -\frac{\partial p}{\partial x_i} + \frac{\partial \tau_{ij}}{\partial x_j} + \rho g_i + F_i$$

- Después está el problema de resolver esas ecuaciones usando la computadora. Aquí intervienen conceptos de otras disciplinas como el análisis numérico, desarrollo de software, visualización de imágenes, etc.



- Finalmente, está el aspecto práctico, en el que el usuario generalmente usa el software creado como resultado de los dos puntos anteriores y lo aplica a problemas reales. En este caso, siempre existe la necesidad de verificar los resultados numéricos con resultados medidos cuidadosamente en un sistema de prueba. Esto es lo que se conoce como validación.



2.3.1. Enfoques de la CFD

Existen dos enfoques utilizados concretamente en la dinámica de fluidos computacional, que dan lugar al desarrollo de técnicas muy diferentes en función de los objetivos buscados. Estos dos enfoques se podrían denominar: *gráficos y simulación*.

La CFD aplicada a gráficos

En estas aplicaciones se busca un modelo que estéticamente dé buenos resultados, no siendo necesario que el comportamiento del modelo se ajuste exactamente al comportamiento de un fluido real.

Si bien en el cine no es un requisito, en el caso de los juegos si que es necesario el *tiempo real*, por lo que estas técnicas, muchas veces tienen que ser rápidas y eficientes.

La CFD aplicada a simulación

A veces no basta con un modelo que "parezca" comportarse como un fluido, sino que se necesita conocer de forma muy exacta cómo se comporta un fluido realmente. Los modelos de CFD que se utilicen en cualquier campo de la ingeniería deben ser muy estrictos con la precisión del modelo, procurando que éste se asemeje lo máximo posible al comportamiento real del fluido.

Siempre que se habla de métodos numéricos, como los utilizados en CFD, estamos hablando de aproximación, por lo que muchas veces será imposible garantizar que el modelo es exacto, y todos los esfuerzos se deben centrar en acotar el error y hacer los modelos tan precisos como sea posible. Esto da lugar a que estas técnicas generalmente no sean en *tiempo real*. Sin embargo, la Visualización en este tipo de simulaciones, permiten que el *tiempo real* pueda ser empleado en este tipo de enfoque cuidando los aspectos de aproximación y controlando el error. Un ejemplo de ello, es la **técnica de Visualización con seguimiento de partículas para fluidos**.

2.3.2. Discretización del espacio de solución de fluidos en la CFD

Como las ecuaciones que gobiernan el movimiento de los fluidos son ecuaciones diferenciales parciales, que resultan de la combinación de las variables del fluido, como son las componentes de velocidad y presión, así como sus derivadas, las computadoras no pueden utilizar estas ecuaciones de manera directa para generar la solución debido a las limitaciones que tienen y que se pueden dividir en cuatro.

- La primera limitación es que las computadoras sólo pueden realizar operaciones aritméticas (+,-,*,/) y operaciones lógicas (falso, verdadero). Esto significa que las operaciones no aritméticas tales como las derivadas o integrales, deben ser representadas en función del tipo operaciones que las computadoras pueden ejecutar.

- La segunda limitación es que las computadoras representan números mediante un número finito de dígitos. Esto significa que existen errores de redondeo, y que estos deben ser controlados.

- La tercera limitación es que las computadoras tienen memorias limitadas, lo que significa que sólo se pueden obtener soluciones en un número finito de puntos en el espacio y tiempo.

- Por último las computadoras realizan un número finito de operaciones por unidad de tiempo. Esto significa que los procedimientos de solución deben reducir al mínimo el tiempo de computadora necesario para realizar una tarea de cálculo mediante la utilización completa de todos los procesadores disponibles y reducir al mínimo el número de operaciones.

Debido a estas limitaciones, se vuelve indispensable la discretización del dominio en el fluido que se quiere analizar, para poder resolverlo por cálculos computacionales. Para ello existen tres *Métodos de Discretización* que permiten generar las soluciones de las ecuaciones diferenciales parciales que son **Diferencias Finitas, Elemento Finito y Volúmen Finito**, los cuales se explican a continuación:

- **Método de Diferencias Finitas**: El método de diferencias finitas es una aproximación para encontrar la solución numérica de las ecuaciones que gobiernan el modelo matemático de un sistema continuo. Básicamente, las derivadas son reemplazadas por aproximaciones en diferencias finitas (*utilizando la serie de Taylor*), que propone un conjunto de ecuaciones para definir las derivadas de una variable como la diferencia entre valores que esta variable puede tomar para varios puntos en el espacio o tiempo. Esto permite realizar una discretización en un espacio conformado por puntos interconectados.

- **Método de Elemento Finito**: El método se basa en dividir el cuerpo, estructura o dominio (medio continuo) sobre el que están definidas ciertas ecuaciones integrales que caracterizan el comportamiento físico del problema, en una serie de subdominios no intersectantes entre sí denominados *elementos finitos*. El conjunto de elementos finitos forma una partición del dominio y dentro de cada elemento se distinguen una serie de puntos representativos llamados *nodos*. Dos nodos son adyacentes si pertenecen al mismo elemento finito; además, un nodo sobre la frontera de un elemento finito puede pertenecer a varios elementos.

- **Método de Volumenes Finitos**: Este es probablemente el método más popular de discretización utilizado en la CFD. Lo que se considera, es que en cada punto/nodo que conforma la estructura que representa el fluido, se construye un volumen de control que no se traslapa con los de los puntos vecinos. De esta forma el volumen total de fluido resulta ser igual a la suma de los volúmenes de control considerados. La ecuación diferencial a resolver se integra sobre cada volumen de control, lo cual entrega como resultado una versión discretizada de dicha ecuación.

La *metodología* que se utiliza para lograr la discretización del espacio de solución de los fluidos, se basa en los siguientes pasos:

- **Discretizar el dominio**: El dominio espacial y temporal continuo del problema debe ser sustituido por uno discreto en celdas y niveles de tiempo. La discretización ideal utiliza el menor número de puntos de celdas y niveles de tiempo para obtener soluciones con la precisión deseada. Este espacio de solución es conocido como **malla**.

- **Discretizar las ecuaciones diferenciales parciales**: Las ecuaciones diferenciales parciales que rigen el problema deben ser reemplazadas por un conjunto de ecuaciones algebraicas con los puntos/celdas de la malla y los niveles de tiempo como su dominio. En situaciones ideales, las ecuaciones algebráicas por *diferencias finitas, elemento finito o volumen finito* que se utilizaron para construir la malla, deberían describir el mismo ámbito físico que el descrito por las ecuaciones diferenciales.

- **Especificar el algoritmo**: El algoritmo paso a paso mediante el cual se obtienen soluciones en cada punto/celda de la malla se deriva de las ecuaciones de *diferencia finita, elemento finito o volumen finito* cuando se pasa de un nivel de tiempo al siguiente. Idealmente, el algoritmo deberá garantizar no sólo soluciones precisas, sino también el uso eficiente de la computadora.

2.3.3. Importancia de construir *Mallas* en la CFD

Despúes de haber realizado una breve introducción sobre los métodos de discretización, el siguiente paso consiste en la construcción de una malla de puntos contenidos en el volumen del fluido. Esto puede ser interpretado como *la discretización del espacio en el cuál el fluido está contenido.*

También, por el apartado de discretización sabemos que el método de diferencias finitas consiste de mallas integradas por puntos, que el método de volúmen finito considera puntos que forman volúmenes y que las mallas para el método del elemento finito consideran subdominios conocidos como elementos en los cuales las variables son determinadas para puntos fijos denomiados nodos. Por lo que podemos concluir, que en general las mallas estan compuesta por: **puntos o nodos, volúmenes o celdas y elementos** que se encuentran relacionados entre sí.

Necesidad de dividir una *malla* en regiones.

Cada problema de fluidos contendrá una amplia variedad de características en el dominio. Características como la vorticidad, la capa límite, regiones con alta velocidad y cambios en la presión, y que deben estar contenidas en el modelo de la Dinámica de Fluidos. Si se tiene una malla capaz de representar estas características entonces se cuenta con la capacidad de delimitar regiones en donde se pueda centrar el análisis.

Para estas regiones consideradas como críticas se requiere una gran concentración de puntos en la malla. De otra forma si el fluido tiene variaciones abruptas en el espacio, como sucede en las regiones críticas, es necesaria la generación de una malla más fina que permita observar estas variaciones con mayor presición. Gracias a este proceso de determinación de regiones críticas se generan *mallas adaptivas.*

Clasificación de las mallas.

Existen principalmente tres tipos de malla (*Mallas Estructuradas, Mallas No Estructuradas y Mallas Híbridas*)

- **Mallas estructuradas o regulares**: En las mallas estructuradas, los elementos que la conforman están conectados de manera regular, es decir, sus elementos forman una estructura de renglones y columnas. Es por ello que existe una restricción geométrica en los tipo de elementos que la pueden conformar (*cuadros* en mallas 2D y *hexaédros* en mallas 3D).

- **Mallas no estructuradas o irregulares**: En este caso los elementos que la conforman están conectados de manera irregular, por lo que no hay una restricción en la geometría de sus elementos (*triángulos, cuadrados, rectángulos, etc* en 2D y *tetraédros, cubos, prismas rectangulares, etc* en 3D). Este tipo de malla es de gran ayuda para geometrías complejas. Sin embargo se puede incrementar en mucho el voúmen de datos que se necesitan calcular para generarla.

- **Mallas híbridas o adaptiva**: Este tipo de malla está compuesto tanto por mallas estructuradas, como no estructuradas. Son utilizadas para representar geometrías en donde la malla se divide en regiones que no se traslapan entre sí. A este tipo de malla también de le conoce como *malla multibloque*.

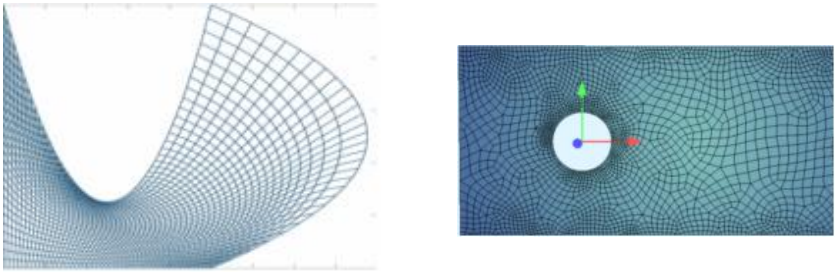


Figura 2.4: *Ejemplo de malla estructurada y no estructurada en 2D*

En el caso de las mallas *estructuradas* en 3D, existe una sub-clasificación de acuerdo a la geometría de sus elementos y la cuál se describe a continuación:

- **Cartesianas** (i, j, k) − típicamente conocidas como mallas de voxeles, los elementos son cúbicos y están alineados a los ejes.

- **Regulares** $(i \times dx, j \times dy, k \times dz)$ − las celdas son prismas rectangulares idénticos y alineadas a los ejes.

- **Rectilíneas** $(x[i], y[j], z[k])$ − las celdas son prismas rectangulares alineadas a los ejes, pero de diferentes dimensiones.

- **Curvilíneas** $(x[i, j, k], y[i, j, k], z[i, j, k])$ − las celdas que lo conforman son hexáhedros o prismas rectángulares deformados para llenar un volumen o cubrir, alrededor de un objeto.

- **Bloques estructurados** − varios sistemas de mallas curvilíneas en el mismo conjunto de datos volumétricos, usados para superar algunas liminationes en la topología de cada categoría o sistema.

Consideraciones con el uso de mallas.

La estructura de la malla está muy relacionada al método numérico, por ejemplo diferencias finitas requiere de un malla regular mientras que elemento finito puede ser utilizado con estructuras irregulares. En teoría el método de volumen finito puede utilizar tanto mallas regulares como irregulares. En las aplicaciones actuales es muy común encontrar solución de mallas regulares, esto debido a que los cálculos son efectuados con mayor facilidad. De manera contraria, cuando se dispone de mallas irregulares hay un incremento en el tiempo de cálculo.

Aunque en la vida real las geometrías que se presentan resultan mucho más complejas, resulta a veces imposible ajustar estas a mallas regulares, y es por eso que el uso de mallas irregulares es socorrido. Algunas aplicaciones de CFD en la actualidad combinan ambos tipos de mallas para resolver geometrías y en algunos casos mejoran los tiempos de cálculo en donde el modelado lo permita.

2.4. Software de análisis para la simulación de fluidos

En este apartado se presenta el software más popular que se utiliza en la simulación de fluidos (solvers), así como algunas herramientas para la construcción de mallas. Sin embargo el problema común en ellas, es que se encuentran limitadas en las técnicas de Visualización en Tiempo Real.

2.4.1. Software para análisis de fluidos *(Solvers)*

Fluent

Fluent es el paquete de simulación en dinámica de fluidos computacional (Solver) de propósito general más popular que existe. La estructura de Fluent le ha permitido incorporar una gran cantidad de modelos para diferentes procesos físicos y químicos que le dan una enorme versatilidad. De esta manera, no sólo se podrán realizar simulaciones de flujos laminares o turbulentos, newtonianos o no newtonianos, compresibles o incompresibles, sino también procesos de transferencia de calor, así como procesos de fundición y con reacciones químicas, como combustión de gases, líquidos y combustibles sólidos. Ref. (*http://www.fluent.com*).

Figura 2.5: *Fluent como solver en la CFD.*

Polyflow

Polyflow es un software CFD de aplicación general basado en la técnica numérica de elemento finito enfocado principalmente para el análisis de procesamiento de polímeros y elaboración de vidrio.

Polyflow es bien conocido por su extensa biblioteca de modelos para fluidos viscoelásticos. Posee una capacidad única de diseño inverso que permiten ser más eficientes en los métodos tradicionales de construcción, prueba y error, lo cual se traduce en reducción de costos y tiempo.

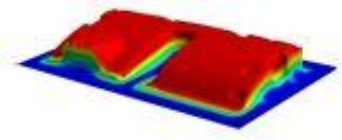


Figura 2.6: *Polyflow como solver en la CFD.*

Fidap

Abundantes modelos físicos y métodos de solución eficientes hacen a Fidap una herramienta ideal de modelación para aplicaciones como procesamiento de polímeros, películas, biomedicina, crecimiento de cristales, metalurgia y procesamiento de vidrio. Fidap es notable por su capacidad de modelar fluidos no newtonianos.

Basado en el método de elemento finito, Fidap ofrece flexibilidad y robustez en el mallado así como cálculos eficientes. Con su especial lenguaje de scripts fáciles de editar, Fidap facilita los estudios de parametrización donde se requiera modificar las propiedades del material o las condiciones de frontera. Fidap está disponible en plataformas Unix/Linux y Windows tanto en serie como para procesamiento en paralelo.

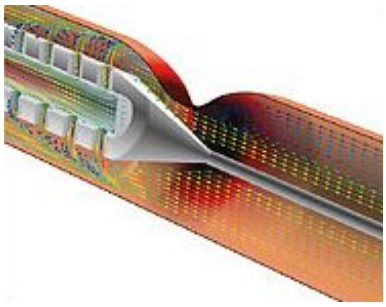


Figura 2.7: *Fidap como solver en la CFD.*

2.4.2. Software para construcción de mallas

Gambit

Es el software de Fluent para generación de geometrías y mallas. La interfaz simple para la creación de geometría y mallado reúne la mayoría de las tecnologías de preprocesamiento de fluent en un ambiente. Herramientas avanzadas para bitácoras permiten editar y convenientemente repetir las sesiones de construcción para estudios paramétricos.

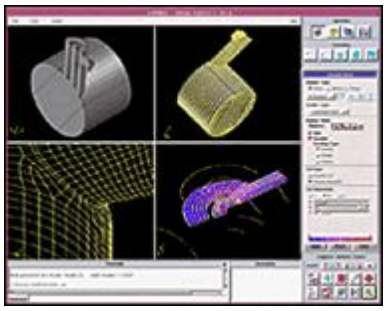


Figura 2.8: *Fluent-Gambit como herramienta para construcción de mallas.*

TGrid

TGrid es un preprocesador especializado que se utiliza para crear mallas tetraédricas no estructuradas de superficies muy grandes y complejas. TGrid ofrece avanzadas herramientas de creación de mallas, con detección de colisiones. Además incluye algoritmos automatizados que permiten ganar tiempo de preprocesamiento y generar mallas de alta calidad.

Capítulo 3

Técnicas de Visualización de datos

La visualización científica está basada en una combinación de múltiples técnicas, las cuales han sido desarrolladas para representar valores y campos escalares, vectoriales o tensoriales.

Estas van desde la visualización de volúmenes, isocontornos e isosuperficies, líneas de flujo de escalares o vectores, topología de vectores o tensores, funciones sobre superficies y características de partes de un conjunto de datos a partir de analogías geométricas o representaciones iconográficas.

A continuación se presentan las principales técnicas que se usan para generar elementos visuales en 3D a partir de datos, para su exploración, interpretación o para su difusión.

3.1. Isocontornos e Isosuperficies

Esta técnica consiste en la extracción de contornos o superficies que representan los puntos de un valor constante (presión, temperatura, velocidad, densidad) dentro de un volúmen de espacio y su evolución en el tiempo.

La extracción se basa en la evaluación de cada celda o cubos que conforman el espacio volumétrico de solución, para determinar las intersecciones con el isocontorno de interés y la conectividad que tienen estos puntos entre sí, para trazar los polígonos que conforman a la superficie.

Las Isosuperficies normalmente se utilizan como métodos de visualización de datos en dinámica de fluidos computacional (CFD), permitiendo el estudio de las características de un fluido (gas o líquido) en torno a objetos, tales como las alas de los aviones. Una isosuperficie puede representar una onda de choque individual en vuelo supersónico, o varias isosuperficies se puede generar para mostrar una secuencia de valores de presión en el aire que fluye alrededor de un ala. Las Isosuperficies tienden a ser una forma popular de la visualización de conjuntos de datos de volumen, ya que pueden ser representadas por un modelo poligonal simple, que se puede dibujar en la pantalla muy rápidamente.

En las imágenes médicas, las isosuperficies pueden utilizarse para representar las regiones de una densidad particular en tres dimensiones ó en una en tomografía computarizada que permite la visualización de órganos internos, huesos u otras estructuras.

El método más utilizado para esta técnica es el *Algoritmo de Marching Cube.*

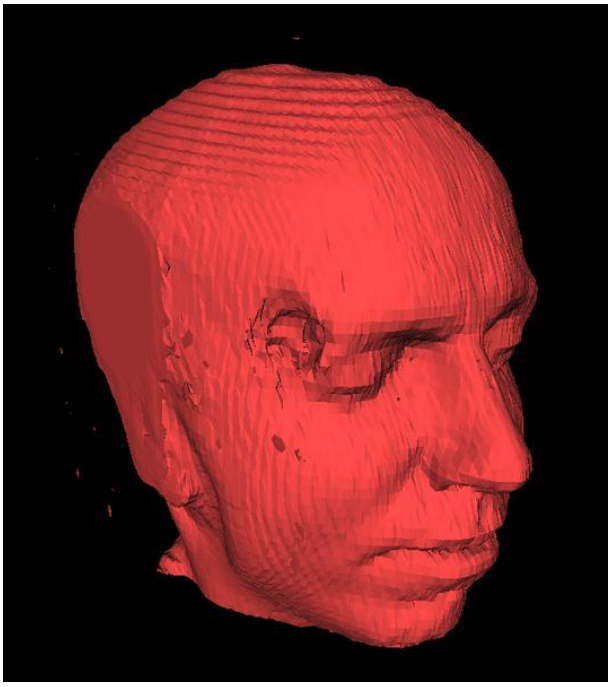


Figura 3.1: *Visualización utilizando la técnica de Isosuperficie por Marching Cube.*

3.2. Volume Render

El *volume render* es una técnica de proyección que produce imágenes de datos volumétricos con el aspecto tridimensional, a partir de datos almecenados como slides o capas de texturas en 2D. Su principal característica es que produce una vista dependiente del punto de visión pero no extrae la información geométrica de los datos.

Esta técnica se basa en atravezar un rayo de luz por pixel, siguiendo el punto de vista de los datos y acumulando o extrayendo información de los valores de intensidad de los voxeles o celdas. La precisión de la imagen final está directamente relacionada con la cantidad de slides en 2D utilizados, es decir, entre más datos almacenados se utilicen, mejor será el resultado final.

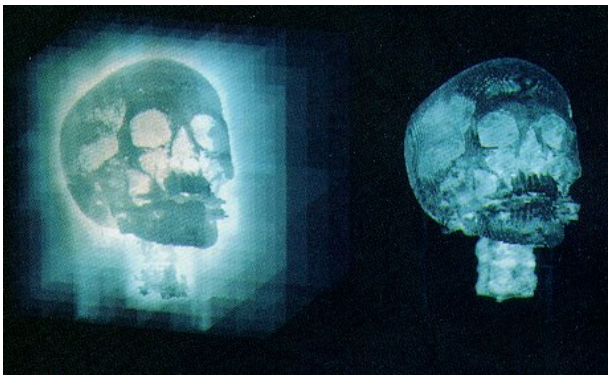


Figura 3.2: *Visualización utilizando la técnica de Volume Render.*

3.3. Campos de velocidades

Las técnicas para la visualización de campos de velocidades, son las más utilizadas para las simulaciones de CFD que tienen como objetivo buscar principalmente *turbulencia, vórtices y otro tipo de estructuras* en los fluidos.

Como revisamos en el capítulo anterior, los fluidos pueden tener diversos comportamientos y características, y con base en ello, se explican las diferentes técnicas que existen para ello.

3.3.1. Visualización con el uso de Glifos

La forma más simple de visualización que existe para representar el comportamiento de un campo de velocidades, es con la utilización de un elemento visual llamado *Glifo*.

Un *Glifo* se representa como una flecha que indica la dirección y la magnitud del campo de velocidades en cada *nodo* existente en la malla del espacio de solución.

Esta técnica es recomendable para simulaciones de CFD constantes en el tiempo y cuya variación en las magnitudes de los nodos de la malla, no sea muy grande, debido a que se perderían importantes detalles.

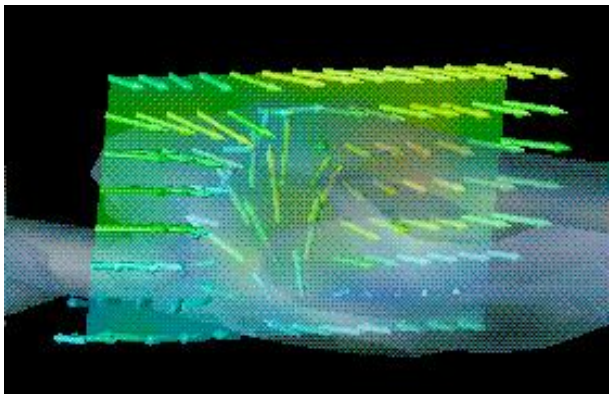


Figura 3.3: *Visualización de un fluido utilizando Glifos*

3.3.2. Visualización con Seguimiento de Partículas

Las Técnicas de Visualización más completas para el análisis de fluidos son las de seguimiento de partículas, ya que a diferencia de los Glifos (en donde se tiene una limitante para fluidos no estacionarios), las técnicas de seguimiento de partículas sirven tanto para simulaciones de fluidos estacionarios, como no estacionarios en mallas de 2D y 3D.

Con estas técnicas también podemos observar, además de la magnitud y la dirección en un fluido, propiedades escalares en las partículas o trayectorias mediante un mapeo de colores o texturas. Las partículas pueden ser representadas por esferas y las trayectorias por tubos ó pixeles y líneas, respectivamente.

Las condiciones iniciales y características de estas técnicas de visualización son las siguientes:

- Contar con los datos del espacio de solución (malla) en donde se ejecuta la simulación de visualización.
- Contar con los datos del campo de velocidades que rige al espacio de solución para poder ejecutar los cálculos de las posiciones de las partículas en el tiempo mediante interpolaciones y métodos numéricos.
- Las partículas conceptualmente tienen *masa cero*, con el fin de que su movimiento en el espacio de solución, dependa única y exclusivamente del comportamiento del campo de velocidades.
- Para el inicio de la simulación, se debe definir la posición inicial de las partiículas en el espacio de solución, ya sea de forma manual a través de un archivo, o mediante un inyector de partículas.

Las técnicas de Seguimiento de Partículas existentes son las siguientes:

Generación de stream lines

Esta es una técnica común que está basada en las simulaciones de fluidos estacionarios y consiste en calcular la ruta para cada partícula con trayectorias que son tangenciales al campo de velocidades.

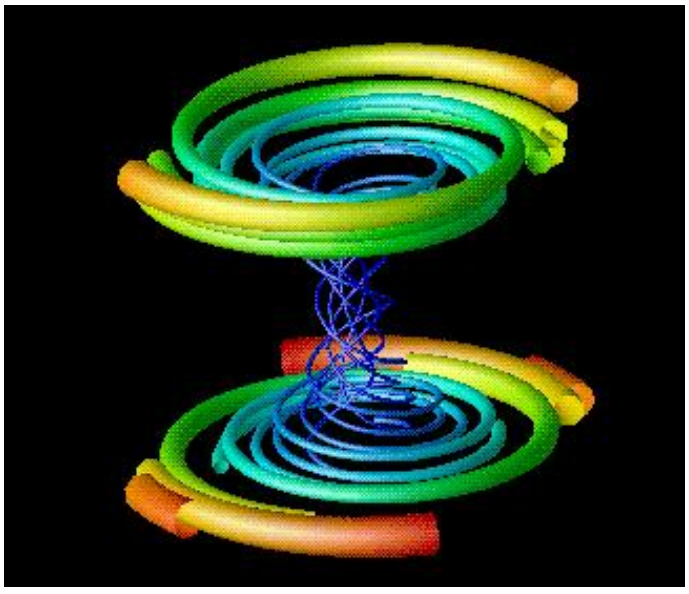


Figura 3.4: *Seguimiento de partículas por generación de stream lines con tubos.*

Generación de streak lines

Esta es la técnica más popular de seguimiento de partículas para simulacones de fluidos no estacionarios y consiste en la inyección continua de partículas desde una posición fija, para el cálculo de las trayectorias de cada una de ellas.

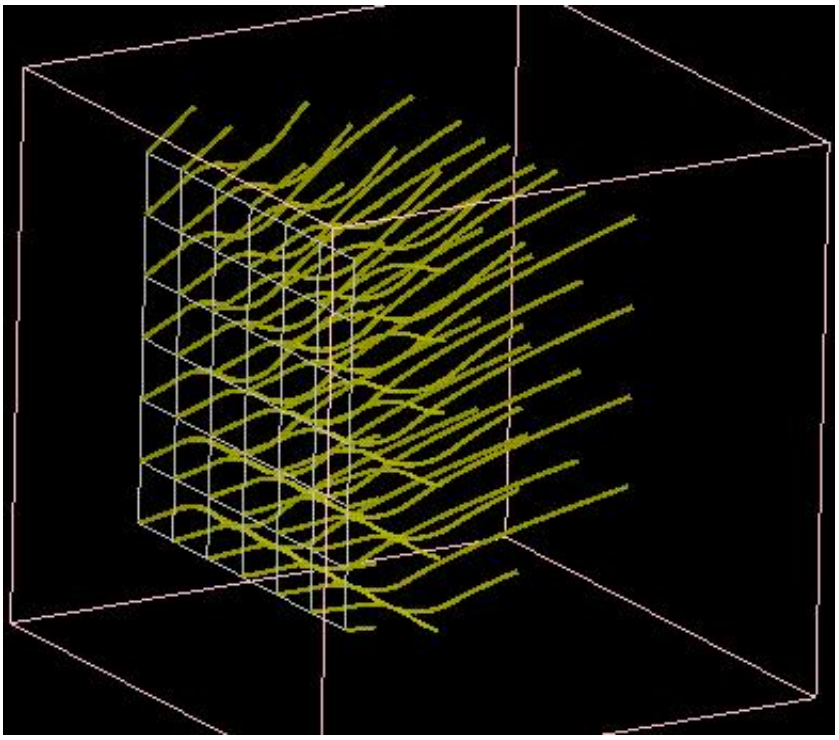


Figura 3.5: *Seguimiento de partículas por generación de streak lines.*

Generación de path lines

Esta técnica se puede emplear en simulaciones de fluidos estacionarios y no estacionarios. Como su nombre lo indica, consiste en trazar la trayectoria de una partícula a lo largo de una simulación.

Generación de time lines

Esta técnica consiste en mostrar la posición de un grupo de partículas que fueron liberadas simultáneamente, en cada instante del tiempo a diferencia del resto de las técnicas de seguimiento, en donde se trazan trayectorias.

Parte II
DESARROLLOS

Capítulo 4

Algoritmos y técnicas de seguimiento de partículas en Tiempo Real

Es importante mencionar que antes de entrar con el diseño y desarrollo del Sistema, se deben explicar con detalle los algoritmos y técnicas empleados para poder llevar a cabo la visualización en tiempo real, así como las ventajas que tienen en este tipo de esquemas.

Para ello se deben considerar varios criterios como la rapidez de los cálculos, técnicas eficientes de búsqueda y pintado en pantalla, así como el reuso de memoria y manipulación de datos. También se explican algoritmos empleados en visualización de partículas pero que no conviene utilizarlos en tiempo real. Ésto con el propósito de hacer una comparación que permita un mejor entendimiento del desarrollo final.

4.1. La importancia de las Técnicas de Visualización en Tiempo Real de fluidos no estacionarios.

Los fluidos no estacionarios representan el porcentaje mayoritario de los casos de estudio de la CFD, y es por ello que se vuelve fundamental el desarrollo de técnicas de visualización para su estudio e interpretación.

Se han presentado algunos algoritmos para el análisis de *fluidos estacionarios* que tienen una relativa extensión a poder utilizar *fluidos no estacionarios*. Sin embargo esta extensión no es trivial porque la variación natural del fluido y la malla en el tiempo, añade complejidad a casi cada paso del algoritmo que se utiliza.

Otro punto importante es que los cálculos para las trayectorias de las partículas liberadas en el espacio de solución, se procesan en modo *batch* y posteriormente son visualizados. Un problema significativo con este enfoque es que la posición de las partículas en el espacio de solución debe ser elegido de antemano. Con ello se forza al investigador a liberar una cantidad excesiva de partículas para poder prevenir pérdidas importantes en las características del fluido.

El mejor enfoque que existe para resolver este problema, consiste en poder interactuar en la visualización, eligiendo la posición y la cantidad adecuada de las partículas que se desean liberar mediante inyectores y analizar su comportamiento en *Tiempo Real*.

Se puede argumentar entonces, que el objetivo que se debe perseguir con la visualización en tiempo real para seguimiento de partículas en fluidos, debe tener las siguientes características:

- Permitir la interacción del usuario con simulaciones de fluidos (*no estacionarios y estacionarios*) en la visualización, que puedan contemplar grandes volúmenes de datos.
- Utilizar técnicas y algoritmos de alto rendimiento (interpolaciones, búsquedas, etc.) que sean rápidos y eficientes en sus cálculos para permitir un volumen adecuado de partículas en la visualización, con el fin de poder analizar el comportamiento de los fluidos sin perder características importantes.
- Flexibilidad en el uso de todas las técnicas de seguimiento de partículas en tiempo real (*streak line, path line, time line*).
- Flexibilidad en el uso de fluidos conformados por mallas 3D de tipo *curvilíneas, rectilíneas, regulares y cartesianas.*

Las **técnicas de seguimiento de partículas en tiempo real** implementadas, están basadas en este enfoque y características, que se explican a detalle en los siguientes apartados.

4.2. El espacio físico *vs*. el espacio computacional.

Las mallas curvilíneas son muy utilizadas en modelos de geometrías complejas. Algunos solvers de CFD internamente transforman estas mallas en un espacio cartesiano uniforme llamado **espacio computacional**, la principal ventaja de hacer esta tarea, es que los cálculos se realizan de forma más simple y rápida. Una vez que se tiene la solución, hay que regresar a la malla original en el **espacio físico** para su análisis y visualización posterior. Al igual que los solvers para fluidos, los algoritmos para el trazado de partículas pueden operar en ambos espacios. Los esquemas en el espacio computacional requieren de una malla curvilínea y el campo de velocidades asociado que serán remapeados al mismo espacio en donde las partículas son analizadas. Cuando este mapeo se realiza en un preprocesamiento para toda la malla, la técnica de seguimineto de partículas se hace muy eficiente.

La principal desventaja de hacer el trazado en el espacio computacional, es que se utilizan técnicas de transformación como en el caso de la matriz Jacobiana, que es una técnica de aproximación solamente, y produce errores significativos en mallas curvilíneas de 3D muy distorcionadas o en fluidos no estacionarios con mallas que sufren modificaciones en su geometría durante la simulación.

Los esquemas para el espacio físico no tienen esas características ya que los procesos de interpolación e integración se realizan sobre la malla curvilínea y eliminan la necesidad de usar técnicas de transformación entre espacios, por lo que son preferidos para fluidos no estacionarios con mallas estáticas o dinámicas. La única desventaja en estos casos, es la localización de la partícula en la malla porque es un proceso complicado y por consiguiente

computacionalmente muy costoso.

Existe una técnica de localización que resuelve este último problema de una forma eficiente y rápida. Esta técnica está basada en mallas 3D construidas con celdas que son divididas a través de una **Descomposición Tetraédrica** con lo que se obtienen las ventajas de ambos espacios (físico y computacional), haciendo factible su utilización en esquemas de visualización en **tiempo real**.

4.3. Algoritmo principal de seguimiento de partículas en fluidos no estacionarios.

Sabiendo las ventajas y desventajas de realizar los cálculos de seguimiento de partículas en un fluido no estacionario en el espacio físico o computacional, se puede concluir que la mejor opción es utilizar los esquemas en el espacio físico.

En estos algoritmos, primero se realiza la localización de la celda en donde se encuentra la partícula a través de alguna técnica de búsqueda. Una vez encontrada, se calcula su velocidad en esa ubicación haciendo una interpolación con las velocidades de los nodos que conforman la celda que la contiene.

Se sabe que para los fluidos no estacionarios, el campo de velocidades no es constante, por lo que se hace una muestra dividiendo el tiempo total que dura la simulación en intervalos de tiempo definidos por el usuario. Sin embargo, es muy común que estos intervalos sean muy grandes por los altos costos computacionales que implica calcularlos. Con esto se vuelve necesario que para tener un mejor detalle del comportamiento del fluido en la visualización, se haga un análisis de éste en intervalos de tiempo más pequeños. Es por eso que en los fluidos no estacionarios, es necesario realizar interpolaciones temporales, tanto para las velocidades, como para las posiciones de los nodos si la malla que se está analizando sufre cambios en el tiempo.

Para obtener la trayectoria de una partícula se utiliza la siguiente ecuación diferencial.

$$\frac{dr}{dt} = v(r(t), t) \tag{4.1}$$

en donde r es la posición de la partícula y v es la velocidad en el tiempo t.

Integrando tenemos:

$$r(t + \Delta t) = r(t) + \int_{t}^{(t+\Delta t)} v(r(t), t)dt \tag{4.2}$$

Los términos del lado derecho de la ecuación pueden ser evaluados utilizando un método numérico de integración, considerando tanto la precisión como la rapidez de los mismos.

El algoritmo de seguimineto de partículas en fluidos no estacionarios se resume en los siguientes pasos:

1. Especificar un punto de inyección en el espacio físico (x, y, z, t).
2. Encontrar las partículas en el espacio de solución.
3. Evaluar las velocidades de las celdas en el tiempo t, utilizando un método de interpolación (entre pasos de simulación).
4. Realizar una interpolación espacial para obtener la velocidad de la partícula en la posición actual (x, y, z).
5. Utilizar el método de integración para determinar la nueva localización de la partícula en el tiempo $t + \Delta t$.
6. Repetir del paso (2) en adelante hasta que la partícula salga del espacio de solución o termine la simulación.

Es importante denotar, que al estár analizando fluidos no estacionarios, el método de integración depende también del tiempo, por lo que el paso (5) involucra a los pasos (3) y (4) tantas veces como sea necesario. Esto depende del método de integración que sea utilizado.

Para el sistema de visialización se utilizó el **Método de Runge-Kutta (cuarto orden)** en el que se requieren tres repeticiones para llegar de t a $t + \Delta t$.

En los siguientes puntos se explican de forma detallada los algoritmos y métodos de interpolación utilizados en el desarrollo principal, así como algunos ejemplos (cuando se considere necesario). Ref. (*Interactive time-dependent particle tracing using tetrahedral decomposition. IEEE*).

4.3.1. Algoritmo de localización en celdas tetraédricas.

Como ya se mencionó anteriormente el principal problema de la técnica de seguimiento de partículas, es que dado un punto arbitrario en el espacio físico se encuentren sus coordenadas en el espacio computacional o natural. Las funciones de interpolación trilinear más utilizadas proveen el mapeo opuesto para la localización, que consiste en determinar las coordenadas de un punto en el espacio físico dadas (f, g, h), en el espacio computacional. Desafortunadamente éste no puede ser invertido fácilmente lo que implicaría realizar cálculos extras que computacionalmente serían muy costosos y no se podrían emplear en esquemas de tiempo real.

Sin embargo existe una técnica de localización alternativa basada en elementos tetraédricos, en donde las coordenadas naturales pueden ser evaluadas directamente desde las coordenadas en el espacio físico. Los elementos tetraédricos permiten usar funciones de interpolación lineal y realizar el mapeo de coordenadas naturales a coordenadas físicas y visceversa debido a que no contienen ningún término no lineal.

Para su explicación, debemos considerar lo siguiente:

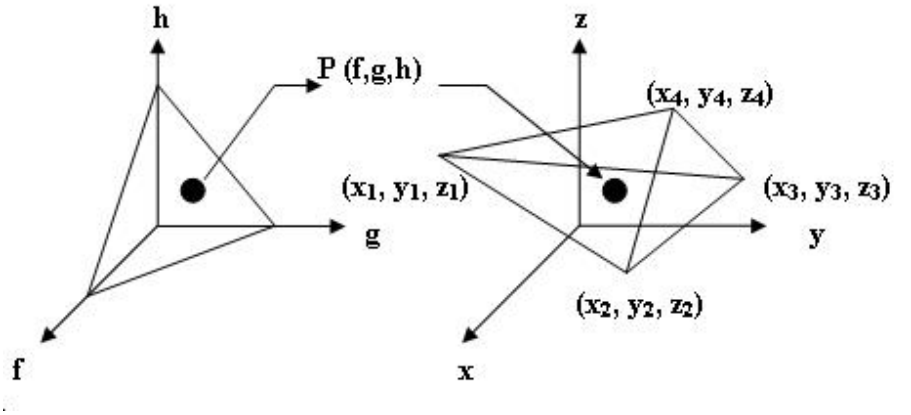


Figura 4.1: *Geometría de un Tetraédro en el espacio físico y computacional.*

Si (f, g, h) son las coordenadas de un punto arbitrario P en el sistema natural y $(x_1, y_1, z_1), (x_2, y_2, z_2), (x_3, y_3, z_3)$ y (x_4, y_4, z_4) son las coordenadas de los vértices del tetraedro en el espacio físico, entonces las coordenadas (x, y, z) de este punto en el sistema físico están dadas por la transformación

$$x = x_1 + (x_2 - x_1)f + (x_3 - x_1)g + (x_4 - x_1)h \tag{4.3}$$

$$y = y_1 + (y_2 - y_1)f + (y_3 - y_1)g + (y_4 - y_1)h \tag{4.4}$$

$$z = z_1 + (z_2 - z_1)f + (z_3 - z_1)g + (z_4 - z_1)h \tag{4.5}$$

Por el contrario, si tenemos las coordenadas (x, y, z) de un punto arbitrario en el sistema físico, sus coordenadas en el sistema computacional están dadas por la transformación inversa

$$f = \frac{1}{|A|}[A_{11}(x - x_1) + A_{21}(y - y_1) + A_{31}(z - z_1)] \tag{4.6}$$

$$g = \frac{1}{|A|}[A_{12}(x - x_1) + A_{22}(y - y_1) + A_{32}(z - z_1)] \tag{4.7}$$

$$h = \frac{1}{|A|}[A_{13}(x - x_1) + A_{23}(y - y_1) + A_{33}(z - z_1)] \tag{4.8}$$

en donde:

$$A_{11} = (y_3 - y_1)(z_4 - z_1) - (z_3 - z_1)(y_4 - y_1) \tag{4.9}$$

$$A_{12} = -[(y_2 - y_1)(z_4 - z_1) - (z_2 - z_1)(y_4 - y_1)] \tag{4.10}$$

$$A_{13} = (y_2 - y_1)(z_3 - z_1) - (z_2 - z_1)(y_3 - y_1) \tag{4.11}$$

$$A_{21} = -[(x_3 - x_1)(z_4 - z_1) - (z_3 - z_1)(x_4 - x_1)] \tag{4.12}$$

$$A_{22} = (x_2 - x_1)(z_4 - z_1) - (z_2 - z_1)(x_4 - x_1) \tag{4.13}$$

$$A_{23} = -[(x_2 - x_1)(z_3 - z_1) - (z_2 - z_1)(x_3 - x_1)] \tag{4.14}$$

$$A_{31} = (x_3 - x_1)(y_4 - y_1) - (y_3 - y_1)(x_4 - x_1) \tag{4.15}$$

$$A_{32} = -[(x_2 - x_1)(y_4 - y_1) - (y_2 - y_1)(x_4 - x_1)] \tag{4.16}$$

$$A_{33} = (x_2 - x_1)(y_3 - y_1) - (y_2 - y_1)(x_3 - x_1) \tag{4.17}$$

y

$$\begin{aligned} |A| = (x_2 - x_1)[(y_3 - y_1)(z_4 - z_1) - (z_3 - z_1)(y_4 - y_1)] - \\ (x_3 - x_1)[(y_2 - y_1)(z_4 - z_1) - (z_2 - z_1)(y_4 - y_1)] + \\ (x_4 - x_1)[(y_2 - y_1)(z_3 - z_1) - (z_2 - z_1)(y_3 - y_1)] \end{aligned} \tag{4.18}$$

Podemos observar, que esta interpolación se fija a una matriz de (3×3), y debido a que en ella muchas expresiones se repiten, computacionalmente los cálculos pueden ser más rápidos con el uso de variables, por lo que se vuelve muy recomendable en esquemas de Tiempo Real.

4.3.2. Técnica de descomposición tetraédrica.

Una vez explicado el método de interpolación que permite hacer el mapeo de un espacio a otro usando tetraedros, podemos explicar esta técnica. La descomposición tetraédrica consiste en dividir cada elemento de la malla (hexaedro) en cinco tetraedros cuyas caras deben coincidir tanto dentro del hexaedro, como con el resto de sus elementos vecinos. Es por ello que se manejan dos tipos de configuración (*Par e Impar*). Estas configuraciones son necesarias, ya que dos elementos de una misma configuración no podrían ir juntos, debido a que sus caras no coinciden entre sí.

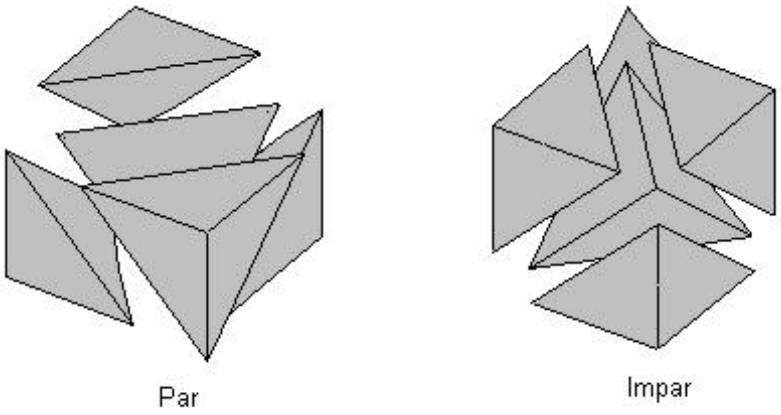


Figura 4.2: *Configuración par e impar de un hexaedro.*

El propósito principal de esta técnica es determinar la localización exacta de la partícula tanto del elemento como del tetraedro que la contiene. Para ello se tienen que evaluar las ecuaciones descritas en el punto anterior y determinar las coordenadas naturales (f, g, h) a

partir de las coordenadas físicas (x, y, z). Hay cuatro condiciones que se deben cumplir para determinar esto:

$f >= 0 \quad : g >= 0 \quad : h >= 0 \quad : 1 - f - g - h >= 0$

Si alguna de éstas no se cumple, entonces el punto se encuentra fuera del tetraedro. En el algoritmo de seguimiento de partículas, esto sucede cuando una partícula cruza por alguna de las caras del tetraedro. El problema central entonces, es determinar hacia donde se mueve. Esto es fácil en el espacio computacional, por ejemplo si $f < 0$, quiere decir que la partícula cruzó la cara en donde $f = 0$. De manera similar, si $g < 0$ ó $h < 0$ quiere decir que la partícula cruzó las caras en donde $g = 0$ ó $h = 0$. Si no se cumple la cuarta condición es decir que $(1 - f - g - h) < 0$, entonces la partícula ha cruzado la cara de la diagonal del tetraedro.

Se puede dar el caso de que dos o más condiciones no se cumplan, sobre todo si la partícula cruza cerca de una esquina, o si atraviesa varias celdas a la vez. En este caso para predecir hacia donde se movió la partícula, se debe tomar el peor valor (el más negativo) y esto hará que siempre se sigua la dirección de la ruta óptima y se encuentre rápidamente el tetraedro correcto.

Localización de partículas por "Look-Up Tables"

El proceso que le indica al algoritmo de búsqueda para donde moverse, se encuentra explicado en tres tablas (**"Look-Up Table"**) las cuales determinan tanto el tetraedro como el elemento en donde se está buscando mientras se encuentra el punto. El siguiente diagrama muestra la descomposición de dos tetraedros en cinco tetraedros con sus respectivas configuraciones.

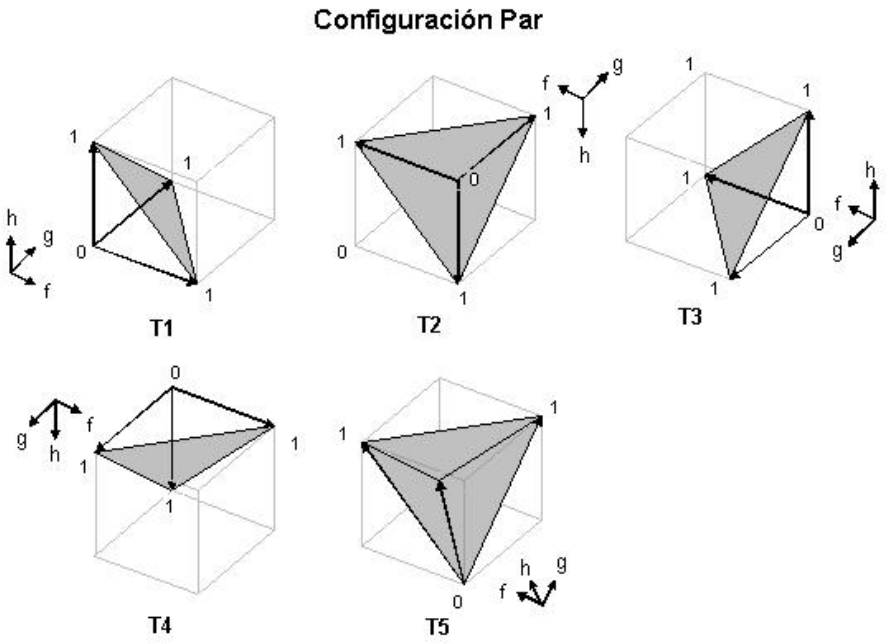


Figura 4.3: *Configuración par de un hexaedro.*

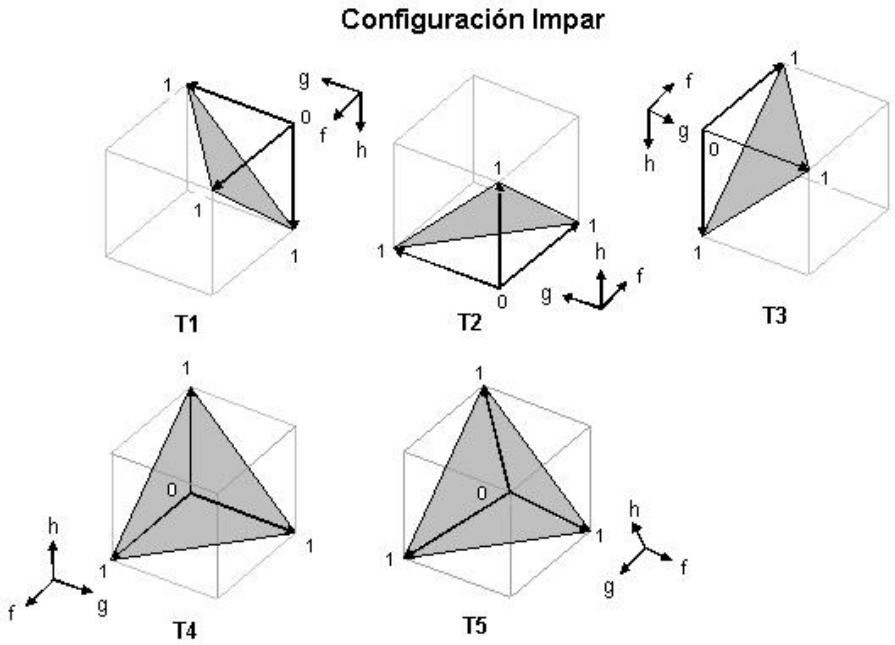


Figura 4.4: *Configuración impar de un hexaedro.*

La primera tabla predice a qué tetraedro se movió la partícula tomando como referencia el tetraedro en el que se encuentraba y la ó las condiciones que no se cumplieron para determinar la cara por la que la partícula salió.

LOCALIZACIÓN DE NUEVO TETRAEDRO					
Condición	***Tetra 1***	***Tetra 2***	***Tetra 3***	***Tetra 4***	***Tetra 5***
$f < 0$	#2	#3	#4	#1	#3
$g < 0$	#4	#1	#2	#3	#1
$h < 0$	#3	#2	#1	#4	#2
$1 - f - g - h < 0$	#5	#5	#5	#5	#4

Cuadro 1: *Localización de nuevo tetraédro.*

Las otras dos tablas se utilizan para actualizar los índices (i, j, k) de las celdas y determinar el nuevo hexaedro en el que probablemente se encuentra la partícula. Esto es debido a que ésta puede avanzar entre los tetraedros de la celda sin salir de ella. Al igual que en el caso anterior, se toma en cuenta la ó las condiciones que no se cumplieron, el tetraedro en donde se encontraba y de forma anexa si el elemento tenía configuración par o impar.

TABLAS DE ACTUALIZACIÓN DE ÍNDICES

CONFIGURACIÓN PAR					
Condición	***Tetra 1***	***Tetra 2***	***Tetra 3***	***Tetra 4***	***Tetra 5***
$f < 0$	$i = i - 1$	$i = i + 1$	$i = i + 1$	$i = i - 1$	misma
$g < 0$	$j = j - 1$	$j = j - 1$	$j = j + 1$	$j = j + 1$	misma
$h < 0$	$k = k - 1$	$k = k + 1$	$k = k - 1$	$k = k + 1$	misma
$1 - f - g - h < 0$	misma	misma	misma	misma	misma

Cuadro 2: *Tetraédro con configuración par.*

CONFIGURACIÓN IMPAR					
Condición	***Tetra 1***	***Tetra 2***	***Tetra 3***	***Tetra 4***	***Tetra 5***
$f < 0$	$j = j + 1$	$j = j - 1$	$j = j - 1$	$j = j + 1$	misma
$g < 0$	$i = i + 1$	$i = i + 1$	$i = i - 1$	$i = i - 1$	misma
$h < 0$	$k = k + 1$	$k = k - 1$	$k = k + 1$	$k = k - 1$	misma
$1 - f - g - h < 0$	misma	misma	misma	misma	misma

Cuadro 3: *Tetraédro con configuración impar.*

Un ejemplo práctico:

Para entender de forma clara esta técnica, las siguientes figuras muestran un ejemplo sencillo para determinar la posición actual de una partícula (P) en el espacio físico, que se encuentra en el tetraedro uno de una celda par, y posteriormente se mueve al tetraedro tres de una celda impar, realizando una serie de pasos que explicamos de forma detallada a continuación.

Lo primero que debemos hacer es evaluar si la partícula se encuentra en la celda actual de búsqueda y de no ser así, validar las condiciones que no se cumplieron para determinar su posición. Para nuestro caso, la celda en la que vamos a iniciar es lógicamente aquella en donde la partícula estaba antes de moverse (T1 en una celda Par).

Utilizando la función de interpolación para mapeo entre espacios, determinamos las coordenadas de la partícula en el espacio computacional (f, g, h) y evaluamos las condiciones que no se cumplieron. En la **figura 4.5**, el lado derecho explica la posición de (P) en el espacio computacional, y se ve claramente que la condición que no se cumple es la de $g >= 0$, por lo que ya tenemos toda la información para determinar el nuevo tetraedro de búsqueda y los índices de la celda en donde éste se encuentra.

En la primera tabla (Tabla 1), si $g < 0$ y estamos en T1(Tetraedro 1), el nuevo tetraedro de búsqueda es T4, es decir la partícula cruzó el plano g = 0 de T1. Posteriormente debemos actualizar los índices del nuevo tetraedro y para ello utilizamos la tabla Par debido a que T1 se encuentra en una celda con esta configuración. Entonces, para la condición $g < 0$ y T1 vemos que j se decrementa una unidad, lo que quiere decir que la partícula abandonó la celda actual.

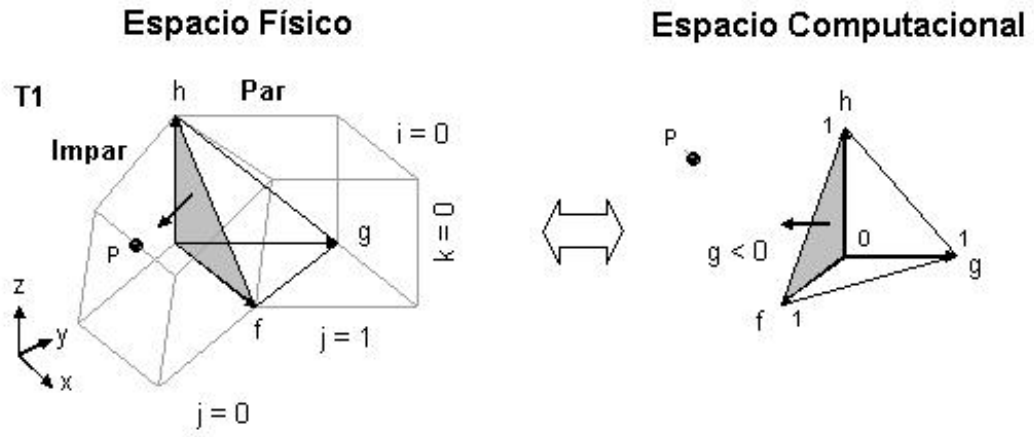


Figura 4.5: *El algoritmo de búsqueda se desplaza de T1 en una celda par a T4 en una celda impar.*

Una vez que se determina el nuevo tetraedro, se debe realizar exactamente lo mismo que el paso anteriror, es decir primero evaluar (f, g, h) y en caso de que alguna de las condiciones no se cumpla, consultar las tablas para determinar un nuevo tetraedro de búsqueda. Estos pasos se repiten hasta que el tetraedro contenga a la partícula o determinar que ésta se encuentra fuera del espacio de solución.

Siguiendo con nuestro ejemplo, el nuevo tetraedro es T4 y se encuentra en una celda de configuración Impar. Entonces evaluando (f, g, h) la condición que no se cumple es $1 - f - g - h >= 0$ y de tablas tenemos que para T4 y la condición $1 - f - g - h < 0$, el nuevo tetraedro de búsqueda debe ser T5 y no hay actualización de índices puesto que se avanza dentro de la misma celda. **Figura 4.6**

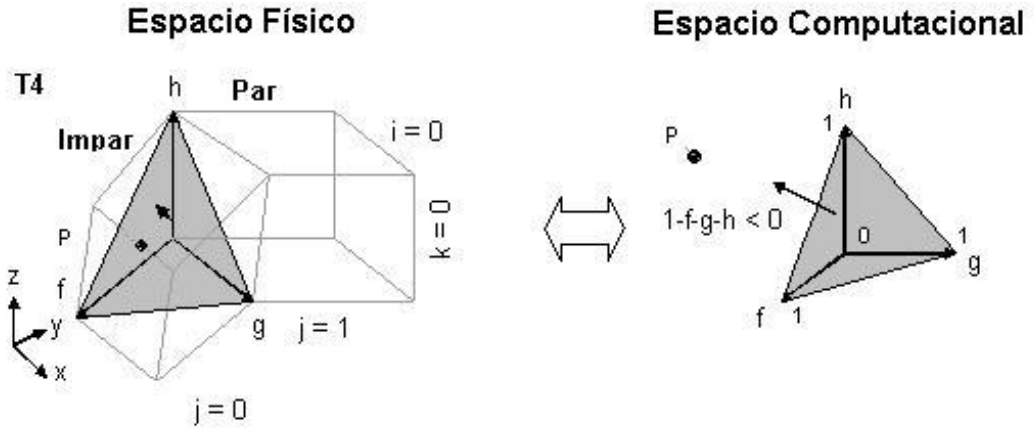


Figura 4.6: *El algoritmo de búsqueda se desplaza de T4 en una celda impar a T5 en la misma celda.*

Nuevamente se evalúa la partícula (P) con respecto al nuevo tetraedro (T5) en el espacio computacional y obtenemos que no se cumple la condición de $f >= 0$, por lo que de tablas determinamos que el nuevo tetraedro de búsqueda es T3 en la misma celda. **Figura 4.7**.

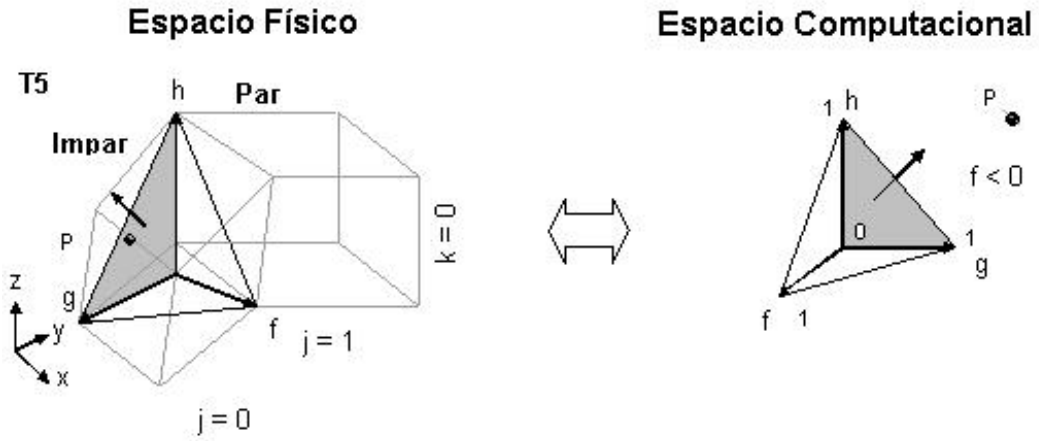


Figura 4.7: *El algoritmo de búsqueda se desplaza de T5 en una celda impar a T3 en la misma celda.*

Finalmente estando en T3 de la celda Impar y evalunado (f, g, h), se determina que $f >= 0$, $g >= 0$, $h >= 0$ y $1 - f - g - h >= 0$, por lo que la partícula (P) se encuentra en el tetraedro tres de la celda cuyos índices son $i = 0$, $j = 0$ y $k = 0$. **Figura 4.8**.

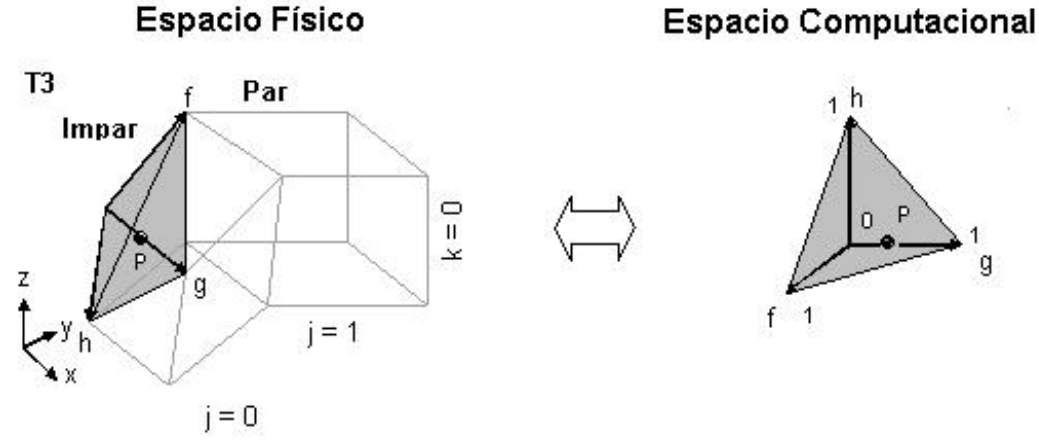


Figura 4.8: *El algoritmo de búsqueda encuentra la partícula en T3 de la celda impar*

4.3.3. Algoritmo de acercamiento para inicio de localización en celdas tetraédricas

Cabe mencionar que el algoritmo propuesto para la localización de partículas en celdas tetraédricas, es muy eficiente, debido a que se realizó pensando en esquemas de tiempo real, sin embargo hay un problema que se debe resolver, ya que el algoritmo por sí mismo no puede. Si analizamos a fondo la técnica, siempre se parte de que el tetraedro inicial de búsqueda se encuentra muy cerca de la partícula, esto implica que sea muy rápida puesto que la localización de ésta se hace de forma inmediata, aunque el espacio esté muy deformado. Pero que pasa si este tetraedro, se encuentra muy lejos de la partícula y tenemos una malla muy deformada, lo más probable, es que no la encuentre aunque ésta exista en el

espacio de solución.

El algoritmo de acercamiento tiene como objetivo encontrar una celda en el espacio físico que esté lo más próxima a la partícula objetivo y que a partir de dicha celda se inicie la búsqueda utilizando el algoritmo de localización. Para poder lograrlo, se busca la distancia mínima entre la partícula objetivo y una serie de puntos que se seleccionan en el espacio físico como consecuencia de realizar búsquedas en el espacio natural.

Sabemos que el espacio computacional se conforma de celdas unitarias y por ello podemos asegurar que un punto $P(f, g, h)$ localizado en este espacio, automáticamente indica las coordenadas de un nodo en la malla unitaria. ——— Con esto, conocer la posición de la celda que contiene este nodo/punto, es una labor muy sencilla. El algoritmo de acercamiento se basa en este concepto para lograr su objetivo.

Básicamente lo que se hace, es trazar cuatro diagonales que conectan las esquinas de la malla en el espacio computacional con el propósito de poder desplazarse a través de ellas (normalización del vector de las diagonales) y encontrar *la posición y número de elemento* de las celdas que se atraviezan con estas diagonales en la malla.

Como la estructura de la posición de celdas en el espacio computacional es el mismo que en el espacio físico, se puede decir que una celda con posición $C(X, Y, Z)$ en la malla del espacio computacional, se corresponde con otra celda $c(f, g, h)$ con la misma posición en la malla del espacio físico. Es por ello que localizando la posición de estas celdas en el espacio computacional, podemos encontrar las celdas que les corresponden en el espacio físico.

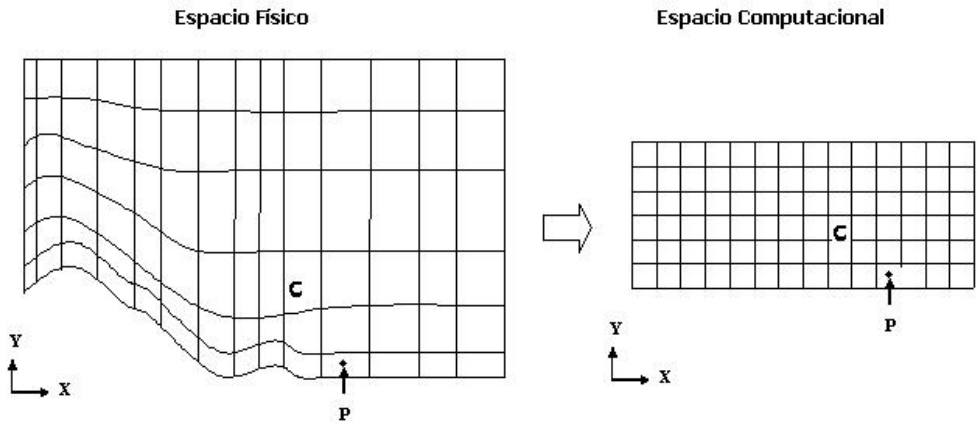


Figura 4.9: *Mapeo de malla en el espacio físico al computacional.*

Teniendo estas características como referencia, los puntos seleccionados para comparar la distancia mínima con la partícula objetivo están contenidos en las celdas que se corresponden con aquellas que fueron seleccionadas en el espacio computacional. De ahí que la celda que se elige para iniciar la búsqueda de la partícula en el espacio físico es aquella que contiene el nodo más cercano a la misma.

4.3.4. Algoritmo de interpolación de la velocidad en un tetraedro

En los fluidos no estacionarios, sabemos que el campo de velocidades cambia con en el tiempo, y debido a que éstos son calculados para los nodos que conforman la malla del espacio de solución y para determinados intervalos de tiempo, se vuelve necesario que el campo de velocidades sea interpolado tanto en el tiempo como en el espacio. En el algoritmo que se está presentando, estas interpolaciones se manejan de manera separada.

Interpolación temporal

La interpolación temporal, se ejecuta usando una función lineal que se aplica para un intervalo de tiempos. Un ejemplo de esto es considerar un tiempo t que se encuentra dentro de un intervalo denotado por t_l y t_{l+1}. La velocidad u para un nodo (i, j, k) arbitrario en la malla está dado por la siguiente expresión:

$$u^t_{i,j,k} = (1 - \delta)u^{t_1}_{i,j,k} + \delta u^{t_{l+1}}_{i,j,k} \tag{4.19}$$

en donde la fracción de tiempo δ está dada por $(t - t_l)/(t_{l+1} - t_l)$.

Es importante mencionar que esta interpolación sólo se puede utilizar para el cálculo de la velocidad en un intervalo de tiempo, no sirve para la interpolación en el espacio de solución.

Interpolación en el espacio

Existen diversas técnicas de interpolación en el espacio de velocidades, pero la más eficiente para este caso, es la interpolación de función base lineal, que además se reutiliza para el cálculo de las coordenadas naturales durante la localización de la partícula.

Por lo tanto, la interpolación se representan de la siguiente manera:

$$u(f, g, h) = u_1 + (u_2 - u_1)f + (u_3 - u_1)g + (u_4 - u_1)h \tag{4.20}$$

en donde f, g, h son las coordenadas naturales calculadas y u_1, u_2, u_3 y u_4 son los vectores de velocidad en los nodos.

4.3.5. Método de integración (Runge-Kutta)

Runge-Kutta de cuarto orden, es uno de los métodos de integración más conocidos que existen. Este método es ideal para calcular el desplazamiento de las partículas, debido a que considera fluidos no estacionarios con geometrás dinámicas. El método está denotado por la siguiente forma:

$$a = v(r(t), t)\Delta t$$

$$b = v\left(r(t) + \tfrac{a}{2}, t + \tfrac{\Delta t}{2}\right)\Delta t$$

$$c = v\left(r(t) + \tfrac{b}{2}, t + \tfrac{\Delta t}{2}\right)\Delta t$$

$$d = v(r(t) + c, t + \Delta t)\Delta t$$

$$r(t + \Delta t) = r(t) + \frac{1}{6}(a + 2b + 2c + d) \tag{4.21}$$

en donde r es la posición de la partícula, v el vector de la velocidad en esa posición y Δt es el paso en el tiempo definido para los cálculos.

Capítulo 5

Implementación computacional

En este capítulo se explica de manera detallada el sistema de visualización implementado. Se presenta el enfoque funcional y técnico sobre el que se trabajó, así como las herramientas de programación utilizadas para su desarrollo. Es importante recalcar que todos los algoritmos y técnicas usados en esta implementación, son los que la literatura especializada recomienda para seguimiento de partículas en tiempo real.

5.1. Enfoque funcional del sistema de visualización.

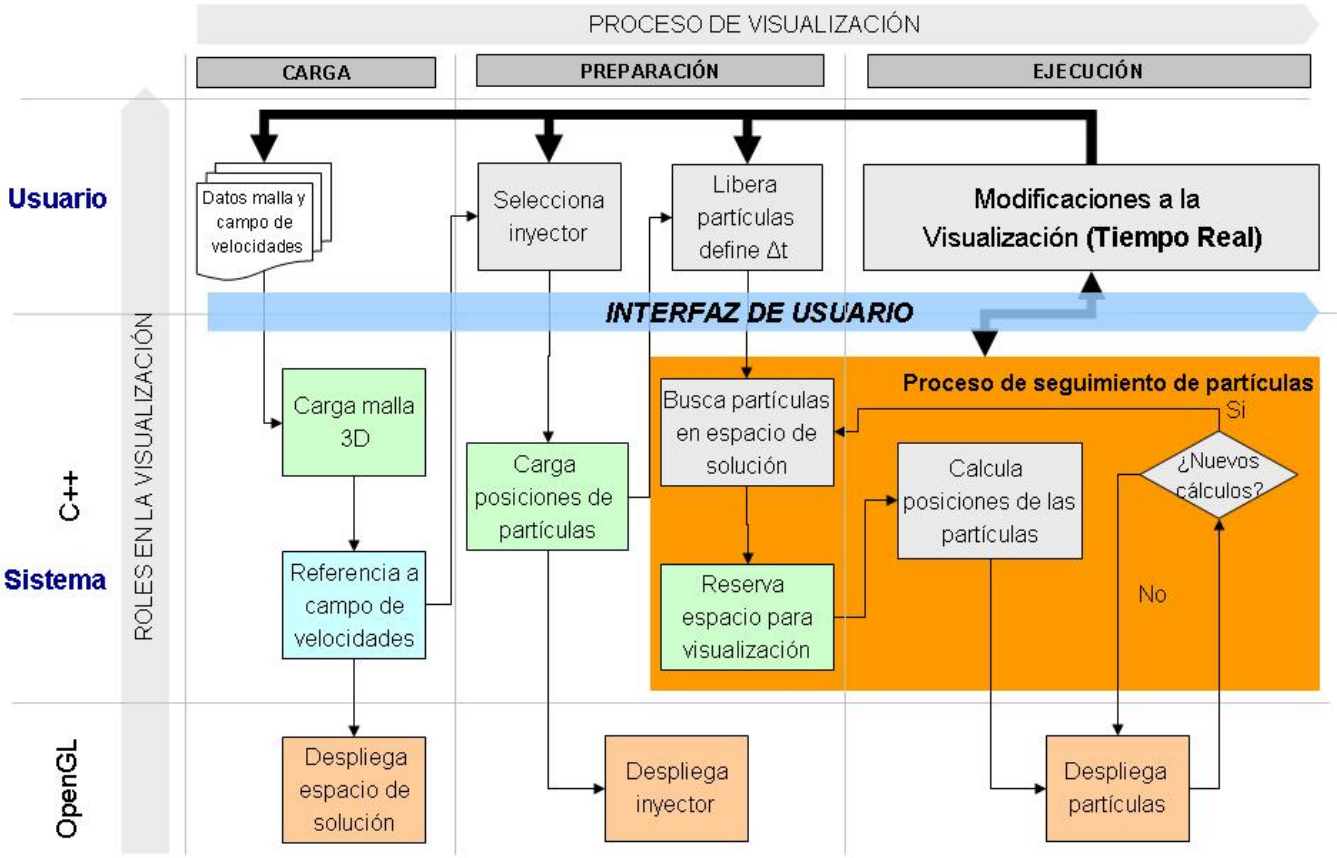


Figura 5.1: *Diagrama funcional del sistema de visualización.*

Este diagrama muestra el enfoque funcional del sistema de visualización, en donde se pueden identificar fácilmente las fases que contiene, las actividades que se llevan a cabo en cada una de ellas y los actores que ejecutan dichas actividades. Es por ello que se tomará este esquema como referencia para explicar el sistema completo (punta a punta).

Existen dos formas para interpretar este diagrama (*Vertical y Horizontal*), la primera de ellas se enfoca en las fases que componen al proceso de visualización (*carga, preparación y ejecución*), y la otra se enfoca en los tipos de actividad que el (*usuario y sistema*) llevan a cabo en el proceso.

5.1.1. Descripción general de los componentes del diagrama funcional

Proceso de Visualización - Interpretación Vertical

Como ya comentamos, esta interpretación muestra las fases que intervien en el proceso punta a punta del sistema de visualización y estas son sus principales características:

- **Fase de carga**: Como su nombre lo indica, esta es la fase en donde se cargan los datos de la simulación del fluido que se va a visualizar (*malla y campo de velocidades*). También en esta fase se controla el tipo de malla que se va a utilizar (*Cartesiana, Regular, Rectilínea o Curvilínea*), así como el despliegue gráfico del espacio de solución en el sistema.
- **Fase de preparación**: Esta fase contempla las actividades previas que se deben realizar antes de iniciar la visualización. En ella se selecciona el tipo de inyector (*Tubo, Rejilla o Círculo*) que se utilizará para la visualización, así como el control de su despliegue en el sistema. Otro punto importante de esta fase es que contiene actividades que forman parte del proceso de seguimiento de partículas, como la definición del tamaño del paso Δt que se va a utilizar y la asignación de memoria para la visualización.
- **Fase de ejecución**: Esta es la última fase del sistema y contempla dos puntos importantes; el cálculo de los datos para la visualización y las acciones que el usuario requiera ejecutar a través de la interfaz con el sistema durante la visualización (**Tiempo Real**). Estas acciones pueden ser: "definir la técnica de visualización (*streak line, path line o time line*)", "agregar inyectores nuevos", "cambiar el tamaño del paso utilizado en la visualización", "reinicio", etc.

Roles en el proceso de la Visualización - Interpretación Horizontal

La interpretación, horizontal tiene como objetivo mostrar las actividades y responsabilidades que tienen tanto el *usuario* como el *sistema* en el desarrollo del proceso de visualización:

- **Usuario**: Es el responsable de indicarle al sistema lo que debe hacer, sus actividades se centran en cargar los datos de entrada, definir los parámetros para la visualización e interactuar con el sistema para realizar modificaciones en Tiempo Real.

- **Sistema**: Es el responsable de la validación y procesamiento de los datos, así como del control de la memoria y el despliegue gráfico. Debido al planteamiento técnico de desarrollo, el sistema se divide en dos partes (*C++, OpenGL*), por el papel que desempeña cada uno de ellos en el sistema. C++ contiene el código responsable del procesamiento de datos y OpenGL se encarga del control del despliegue gŕafico.

5.2. Enfoque técnico del sistema de visualización

El éxito de desarrollar una buena aplicación de software, pasa por varias etapas como análisis, diseño y desarrollo. Se sabe que hoy en día las herramientas que existen para desarrollar, son tan amigables y poderosas, y minimizan en gran medida los tiempos en esta atapa. Sin embargo, resulta crucial escoger el lenguaje de programación adecuado para que se puedan cumplir correctamente los objetivos planteados en las dos etapas previas, así como tener una arquitectura de solución adecuada.

Basados en estas premisas, se presentan a continuación las herramientas que se utilizaron para el desarrollo y el diagrama de objetos implementado.

5.2.1. Herramientas de desarrollo utilizadas para el sistema de visualización

Los sistemas de visualización en tiempo real, por su naturaleza, consideran los siguientes objetivos:

- Deben responder a eventos o señales del sistema o del ambiente y que los cambios de estado no dependan de cálculos impredecibles.
- Deben contener algoritmos alto rendimiento para eficientar y acelerar los cálculos durante la visualización.
- Deben hacer un eficiente uso de los recursos de la computadora (memoria, disco, procesadores), con el fin de que las actividades se repartan adecuadamente y se puedan procesar volúmenes importantes de datos.
- Deben tener una herramienta eficiente para el despliegue de los componentes gráficos.
- Deben tener una interfaz amigable para interactuar con los usuarios durante la visualización.

Pero también es, conocido que además de cumplir con estas expectativas técnicas, los desarrollos debe ser mantenibles, reusables, ser multiplataforma y estar estructurados (módulos) de tal forma que puedan evolucionar.

Programación Orientada a Objetos con C++

La programación orientada a objetos (*POO*) promete mejoras de amplio alcance en la forma de diseño, desarrollo y mantenimiento del software, ofreciendo una solución a los problemas y preocupaciones que han existido desde el comienzo del desarrollo de software: la falta de portabilidad del código y reusabilidad, código que es difícil de modificar, ciclos de desarrollo largos y técnicas de codificación no intuitivas.

Un lenguaje orientado a objetos ataca estos problemas teniendo tres características básicas: debe estar basado en objetos, basado en clases y ser capaz de manejar la herencia de clases.

Podemos definir un objeto como un conjunto complejo de datos y programas que poseen estructura y forman parte de una organización. Esta definición especifica varias propiedades importantes de los objetos. En primer lugar, un objeto no es un dato simple, sino que contiene en su interior cierto número de componentes bien estructurados. En segundo lugar, cada objeto no es un ente aislado, sino que forma parte de una organización jerárquica o de otro tipo.

Con ello se vuelve muy sencillo, poder realizar desarrollos con base en objetos que tengan atributos propios que los describen, así como métodos y funciones que determinan cómo éste interactúa con el resto de los elementos. Con esto tenemos una arquitectura de sistema modular fácil de mantener y evolucionable.

C++ es el lenguaje pionero de la *POO* y ha sido por muchos años el lenguaje concentido de los desarrolladores, hasta la aparición de java. Sin embargo este lenguaje utiliza de una manera casi natural, el API de OpenGL para los despliegues gráficos. También C++ ofrece portabilidad, es multiplataforma y tiene métodos y funciones eficientes para el acceso a archivos binarios y no binarios, así como el manejo de la memoria.

Despliegue Gráfico con OpenGL

OpenGL (Open Graphics Library) es una especificación estándar que define una API multilenguaje y multiplataforma para escribir aplicaciones que produzcan gráficos 2D y 3D. La interfaz consiste en más de 250 funciones diferentes que pueden usarse para dibujar escenas tridimensionales complejas a partir de primitivas geométricas simples, tales como puntos, líneas y triágulos.

Otra característica importante es que tiene dos propósitos principales:

- Ocultar la complejidad de la interfaz con las diferentes tarjetas gráficas, presentando al programador una API única y uniforme.
- Ocultar las diferentes capacidades de las diversas plataformas hardware, requiriendo que todas las implementaciones soporten la funcionalidad completa de OpenGL (utilizando emulación software si fuese necesario).

También OpenGL se define como una máquina de estados, es decir, se pueden alterar un conjunto de estados (o modos), los cuales mantienen un cierto efecto en la escena hasta que estos estados cambian su valor. Por ejemplo, el color es una variable de estado y puede estar asignada de inicio con un valor, por lo que todos los objetos que sean creados después de esta definición se ven afectados por esta variable.

Con estas características generales podemos concluir que OpenGL es una herramienta de despliegue gráfico muy eficiente e ideal para cumplir con los requerimientos técnicos que persiguen los visualizadores en tiempo real.

5.2.2. Diagrama de objetos del sistema de visualización

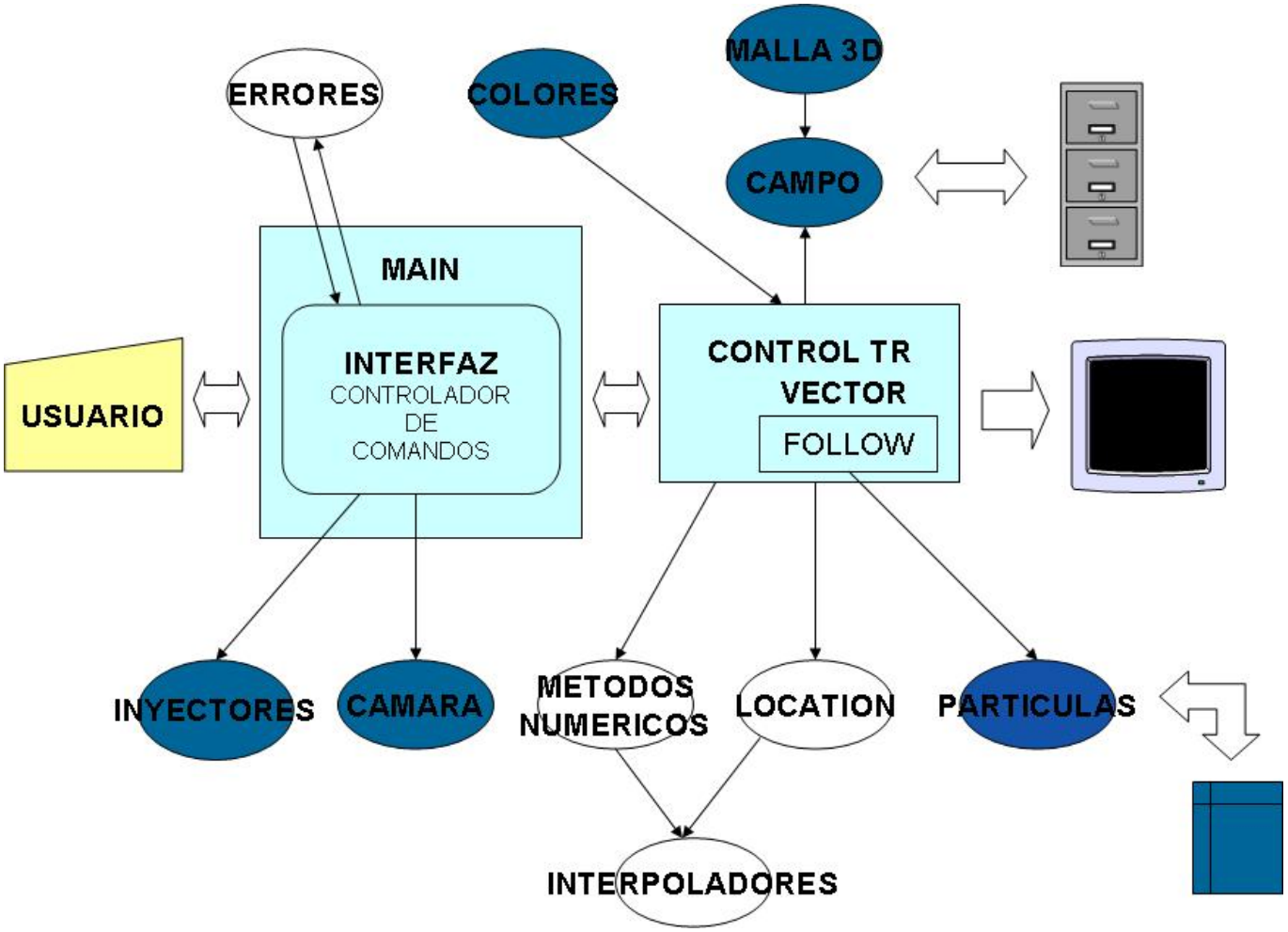


Figura 5.2: *Diagrama técnico de objetos del sistema de visualización.*

El diagrama de objetos, muestra el enfoque técnico utilizado para el sistema de visualización, así como la relación que tienen cada uno de los elementos entre ellos. Se divide en tres grandes bloques por funcionalidad (*control principal, funcionalidad gráfica, uso común*). A continuación se da una breve explicación de cada uno de ellos.

Objetos de control principal

- **Interfaz**: Este objeto controla todas las tareas de interacción entre el usuario y el sistema.
- **Control**: Este objeto se puede considerar como el cerebro del proceso de visualización, ya contiene toda la lógica de control del sistema. También gestiona el uso de memoria y le indica al usuario sobre el estado actual de la visualización a través de la interfaz.
- **Follow**: Este objeto es el que controla y calcula los datos derivados del algoritmo de seguimiento de partículas para su posterior despliegue.

Objetos con funcionalidad gráfica

- **Inyectores**: Este objeto gestiona todas las tareas que tienen que ver con los inyectores que se utilicen en la visualización.
- **Camara**: Este objeto controla la posición de la cámara en la escena de la visualización. Contiene los métodos para poderse mover a través del espacio de soluciones.
- **Partículas**: Este objeto controla las técnicas de visualización que se quiere mostrar, así como su despliegue en el sistema.
- **Malla3D**: Este objeto controla la lógica para cargar la malla en memoria, así como los métodos para encontrar elementos en la misma.
- **Campo**: Este objeto hereda los métodos del objeto Malla3D y agrega el control de los campos de velocidades en los nodos de la malla, así como la gestión con OpenGL para su despliegue.
- **Colores**: Este objeto es el responsable de controlar el estado del color de las partículas en el sistema. Utiliza (RGB).

Objetos de uso común

- **Location**: Este es el objeto que ejecuta el algoritmo de búsqueda y localización de las partículas con el algoritmo de descomposición tetraédrica.
- **Interpoladores**: Este objeto contiene las funciones de interpolación (*tiempo, espacio*) que se utilizan para el algoritmo de seguimiento de partículas.
- **MétodosNumericos**: Contiene las funciones del método de integración utilizado (*Runge-Kutta*).
- **Errores**: Contiene la biblioteca de funciones para la validación de datos de entrada.

Importancia de una arquitectura modular

Como se puede observar en el *diagrama técnico de objetos* (*Figura 5.2*), la arquitectura técnica empleada está basada en funcionalidades independientes, lo que permite:

- Fácil mantenimiento.
- Evolución de la solución con nuevas funcionalidades.
- Desarrollar una interfaz independiente sin afectar el núcleo del programa, uso de APIs.
- Reusabilidad de funcionalidades en otros sistemas.
- Definición bibliotecas para el control de errores y funcionalidades comunes.

- Control de versiones por módulo.

Con ésto se puede asegurar que el sistema además de ser multiplataforma por la característica natural de C++ y OpenGL, está modularizado por lo que también es *evolucionable, reusable y mantenible.*

5.3. Sistema de visualización de partículas en tiempo real

En este apartado se verá a detalle el sistema desarrollado, las condiciones (reglas de negocio) que se definieron para su correcto funcionamiento, así como las mejoras que éste tiene con respecto a un esquema tradicional de visualización. Es importante recordar que el sistema se presenta siguiendo como base el diagrama funcional visto.

5.3.1. Componentes del sistema de visualización

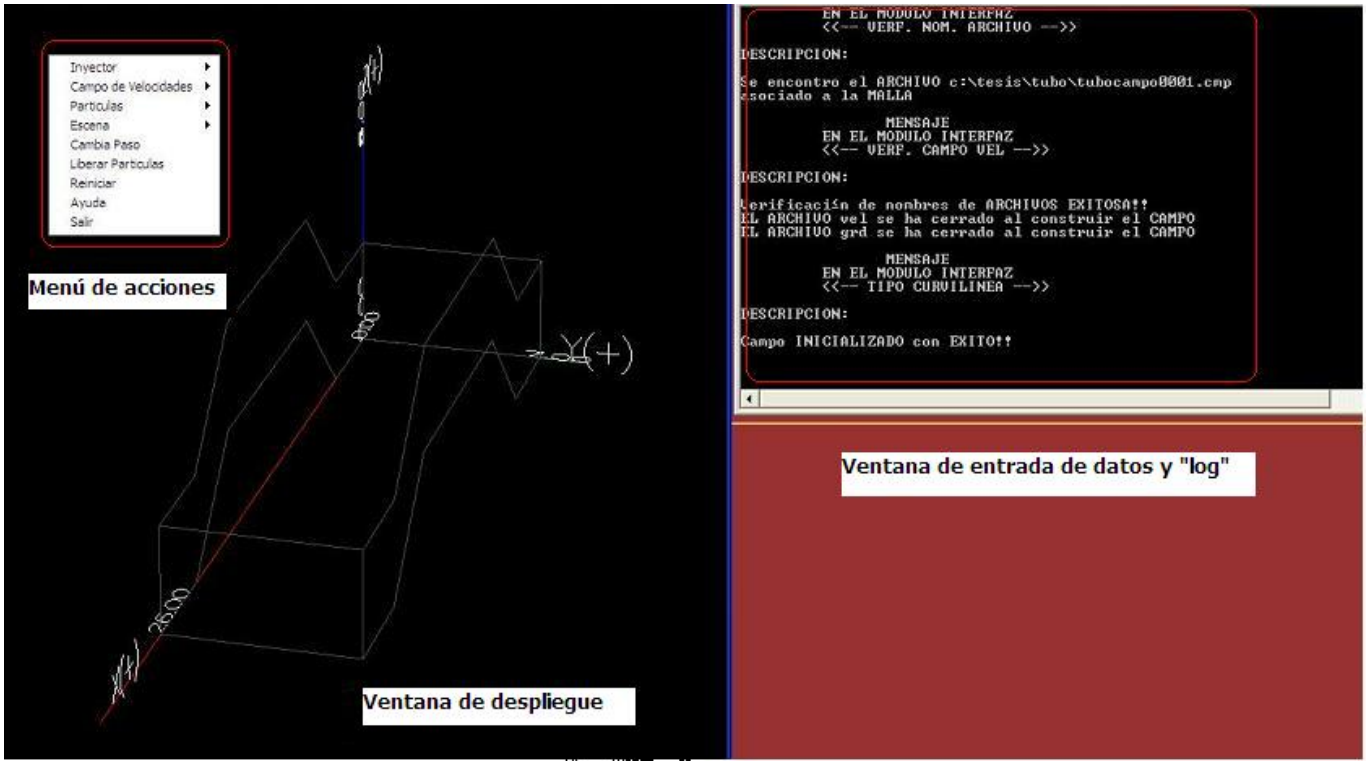


Figura 5.3: *Componentes del sistema de visualización.*

El sistema de visualización de partículas, se compone de tres módulos principales (*Menú de acciones, Ventana de despliegue, Ventana de entrada de datos*) que se explican a continuación:

- **Menú de acciones**: Este menú permite la interacción del usuario con el sistema, tanto en la fase de preparación, como en la fase de ejecución. A través de él, se solicitan las acciones que el usuario quiere ejecutar para modifocar algo durante el proceso de visualización (*mover cámara, agragar inyectores, editar fronteras de campo de velocidades, etc.*).

- **Ventana de despliegue**: En la consola de despliegue se proyectan los gráficos de la visualización. Se puede ver el progreso de la ejecución y de igual forma interactuar con el *teclado y mouse* para mover el ángulo de la cámara en la posición deseada.

- **Ventana de entrada de datos y log**: Esta consola se activa siempre y cuando sea necesario parametrizar un valor debido a una acción que el usuario ejecutó. En ella se introducen los valores que se van a utilizar. De igual forma se activa al inicio del sistema, cuando se cargan los datos de la malla y/o los campos de velocidades. Otra funcionalidad importante es que en ella también se despliegan todas las acciones importantes que el usuario ejecuta (sirve como "log" del sistema).

5.3.2. Fase de carga en el sistema de visualización

Esta fase representa el inicio del sistema y tiene los siguientes objetivos: 1) seleccionar el tipo de malla e indicar la ruta en donde se encuentran los archivos que contienen los datos del campo de velocidades; 2) mostrar el espacio de solución en la ventana de despliegue.

El siguiente diagrama muestra el proceso en la fase de carga y describe los pasos que el sistema solicita para despleguar el espacio de solución.

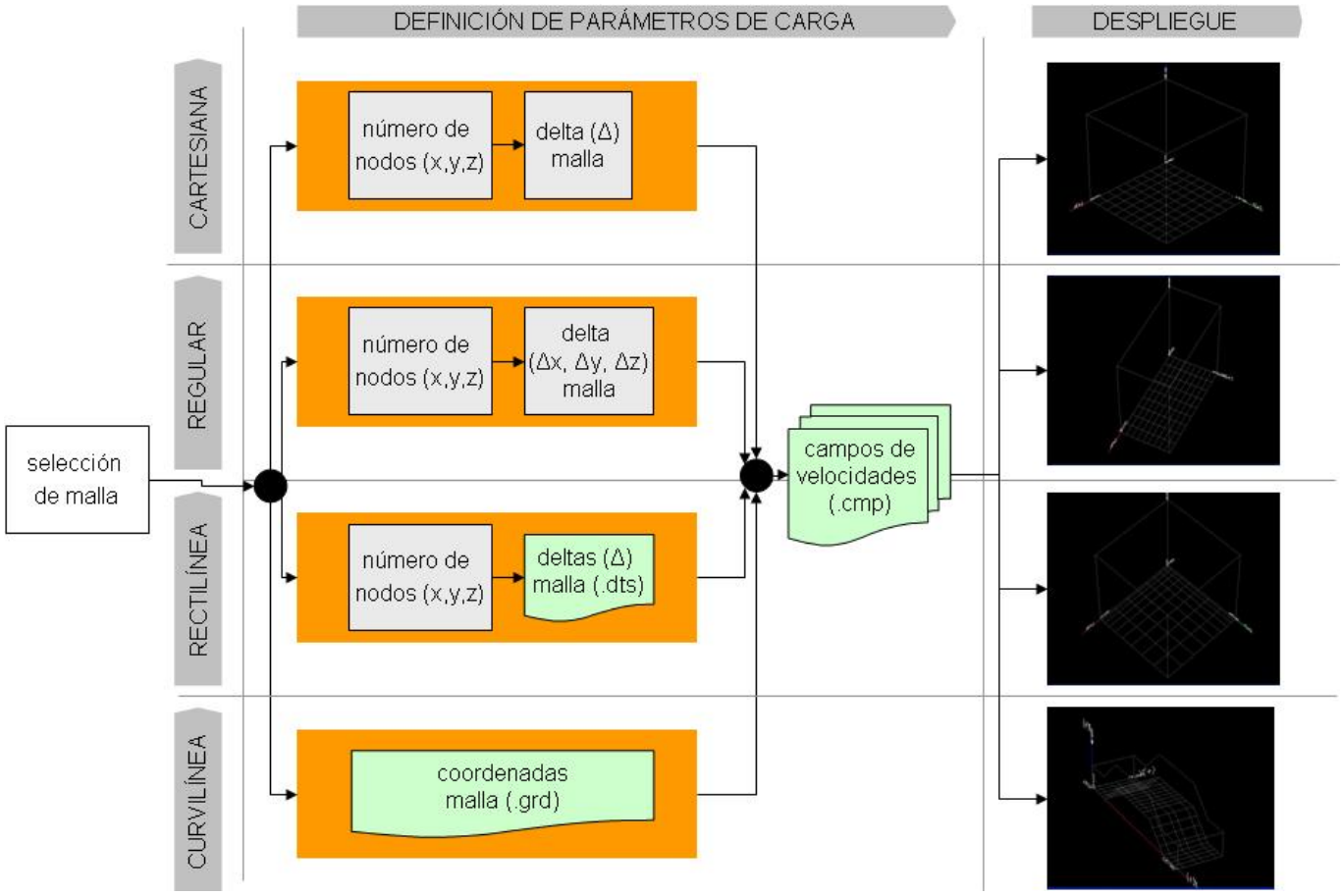


Figura 5.4: *Diagrama de la fase de carga del sistema de visualización.*

Las características importantes en esta fase son:

En el procesamiento.

- El sistema soporta cuatro tipos de mallas estructuradas (*Cartesiana, Regular, Rectiliena y Curvilínea*), y en función de ello pide una serie de parámetros.
- Se tienen controles de calidad de datos, tanto para los parámetros que describen a la malla, como para el campo de velocidades que se va a cargar.
- Los archivos que contienen los datos del campo de velocidades están en binario, lo que permite realizar accesos de manera directa en la ejecución de la visualización.
- Se definió una nomenclatura para el nombre de los archivos, con el fin de poder hacer búsquedas de manera automática.

  Nomenclatura para el nombre de archivo del campo de velocidades:

  {Nombre_Simulación} + **campo** + {Número de Archivo (dddd)} + {**.cmp**}.
  Ejemplo: *TUBOcampo0001.cmp*

  Nomenclatura para el nombre del archivo de la malla: (Esto sólo aplica para mallas curvilíneas)

  {Nombre_Simulación} + **grid** + {**.grd**}.
  Ejemplo: *TUBOgrid.grd*

  Nomenclatura para el nombre del archivo que contiene las deltas (ΔM) de una malla rectiínea:

  {Nombre_Simulación} + **delta** + {**.dts**}.
  Ejemplo: *TUBOdelta.dts*

En el despliegue.

- La posición de la cámara en el despliegue de la escena, se calcula automáticamente considerando las dimensiones del campo de solución que se va a visualizar.
- El espacio de solución muestra: la referencia de la malla con respeto al origen, las dimensiones de la malla.
- El sistema permite dibujar/ocultar las fronteras de la malla.

5.3.3. Fase de preparación en el sistema de visualización

Esta fase representa a los pasos previos que se deben realizar antes de ejecutar la visualización en el sistema y tiene los siguientes objetivos: 1) seleccionar el tipo de inyector que se va a utilizar y mostrarlo en el espacio de solución; 2) elegir el paso de tiempo que se va a utilizar y 3) iniciar la visualización.

El siguiente diagrama muestra el proceso en la fase de preparación y describe los pasos que el sistema solicita antes de iniciar la visualización.

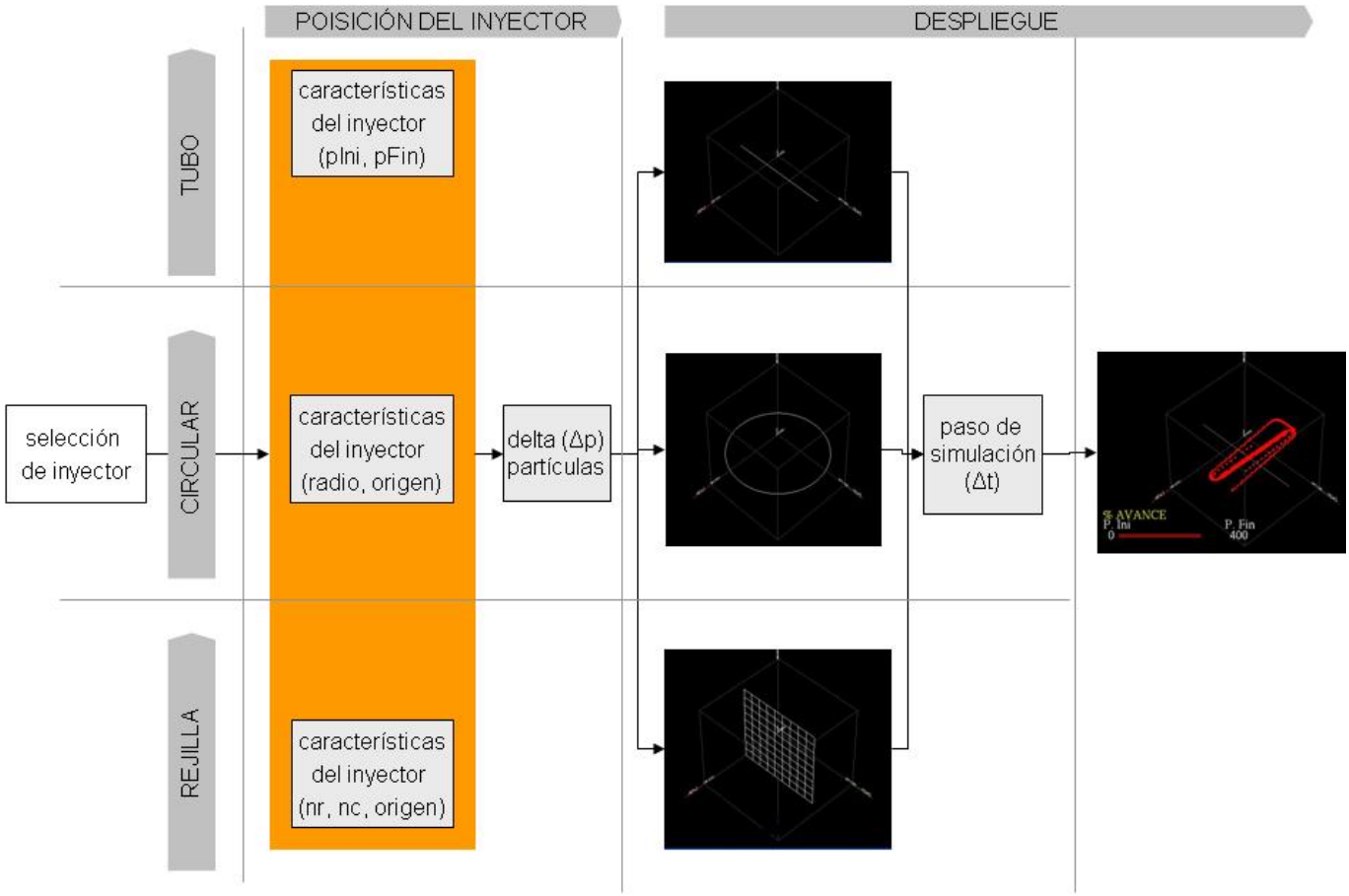


Figura 5.5: *Diagrama de la fase de preparación del sistema de visualización.*

Las características importantes en esta fase son:

En el procesamiento.

- El sistema soporta tres tipos de inyectores (*Tubo, Círculo y Rejilla*), y en función de ello pide una serie de parámetros.
- El sistema permite editar los parámetros de los inyectores en caso de que hubiera algún error en la cración.
- Para la ejecución de la visualización, el sistema hace un cálculo del paso de simulación

($\Delta t/100$ ó $\Delta t/10$ si el número de pasos $> 50,000$) de manera automática y lo muestra como una recomendación, sin embargo el usuario puede tomar esta valor ó definir el suyo.

- En esta fase de ejecuta el algoritmo de acercamiento y de búsqueda inicial, y en base a ello se determina cuántas partículas están dentro del espacio de solución y se reserva memoria para guardar sus cálculos.

En el despliegue.

- La posición de la cámara en el despliegue de la escena, se calcula automáticamente considerando las dimensiones del campo de solución que se va a visualizar.
- El sistema permite cambiar a los inyectores de posición en cualquiera de las direcciones de los ejes (X, Y, Z) con el teclado y mouse.
- El sistema permite dibujar/ocultar los inyectores en el espacio de solución.

5.3.4. Fase de ejecución en el sistema de visualización (*Tiempo Real)*

En esta fase la visualización ya se está ejecutando en el sistema y tiene el único objetivo de *que el usuario pueda interactuar con la visualización en tiempo real.*

El siguiente diagrama muestra las acciones más significativas que puede realizar el usuario en Tiempo Real.

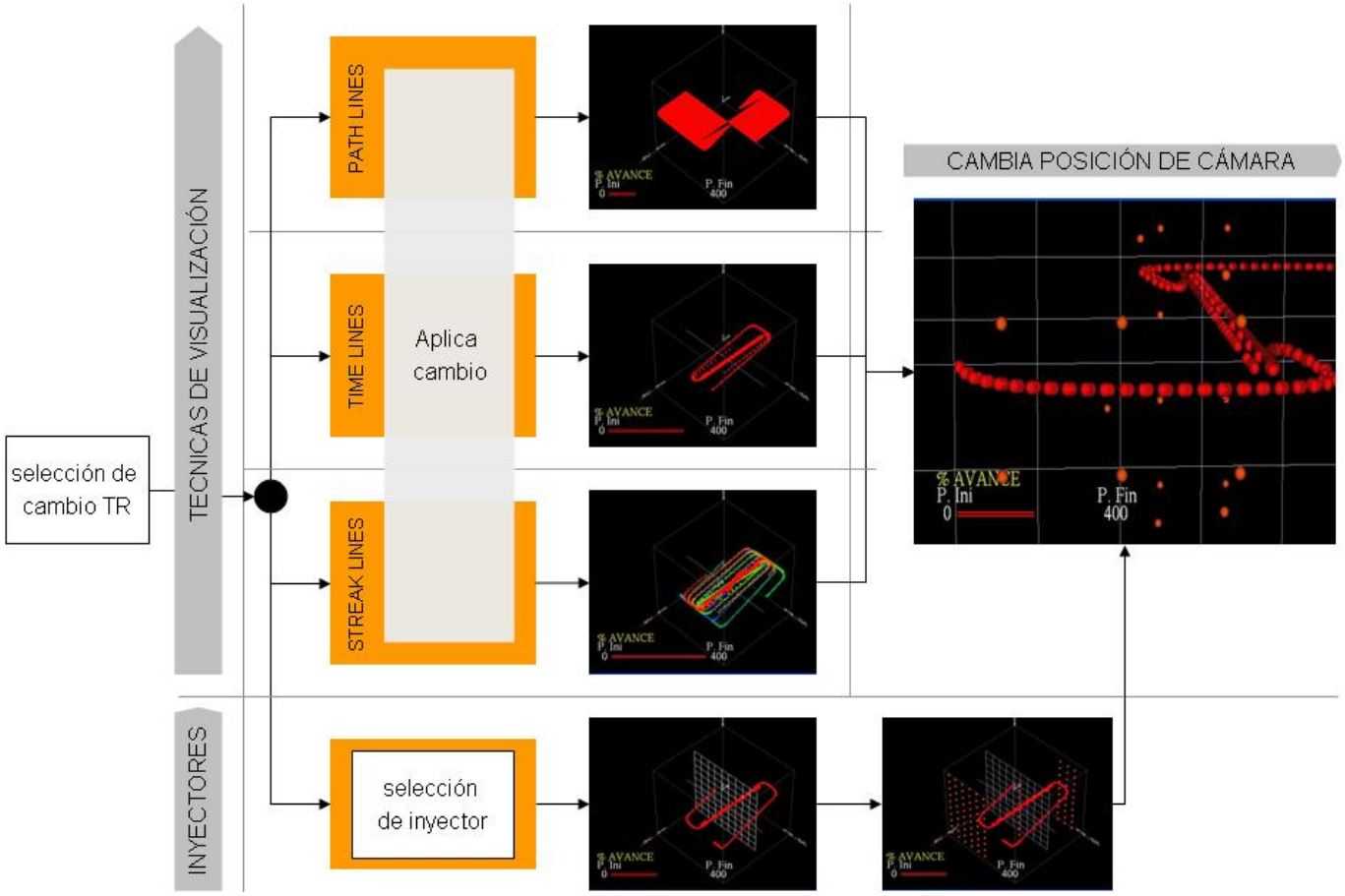


Figura 5.6: *Diagrama de la fase de ejecución del sistema de visualización (*Tiempo Real*).*

Las características importantes en esta fase son:

En el procesamiento.

- El sistema permite que se modifiquen parámetros como cambio del paso de simulación y técnicas de visualización en tiempo real.

- El sistema permite que se puedan agregar y/o eliminar inyectores durante la visualización teniendo un máximo de siete. Esto permite liberar partículas en el momento que se desee, o eliminar las que no sean de interés para el usuario.

- El sistema permite que se reinicie la visualización tomando encuenta otra simulación, liberando los recursos de la máquina como memoria RAM.

En el despliegue.

- El sistema permite que el usuario pueda mover libremente la cámra en todas las direcciones, con la finalidad de que pueda tener diferentes ángulos de la visualización y tomar mejores desiciones.
- Está implementado en el sistema el cambio automático de pixel a esfera cuando el usuario acerca la cámara a una distancia considerable. Esto se hace para trabajos futuros en donde además de la trayectoria, sea interesante poder visualizar características adicionales como texturas o la rotación de las partículas que me signifiquen otras características del fluido.
- **Optimización gráfica**: Como ya se ha mencionado en este apartado, el sistema está preparado para visualizar características independientes a la trayectoria, es por ello que se ha colocado un control de distancia, que permite cambiar de un pixel a esfera y poder apreciar otro tipo de comportamientos en la visualización. Con esto se optimizan los recursos para el depliegue gráfico, ya que una esfera representa un costo computacional mayor que un pixel.

  Para que esto ocurra, simplemente se debe acercar la cámara al objetivo que se desea y el sistema realizará el cambio de forma automática como se ve en la figura 5.6.

De esta forma se ha mostrado el sistema de visualización implementado y en la siguiente sección se muestran algunas de las simulaciones que han sido visualizadas con este sistema.

Parte III
RESULTADOS

Capítulo 6

Aplicaciones

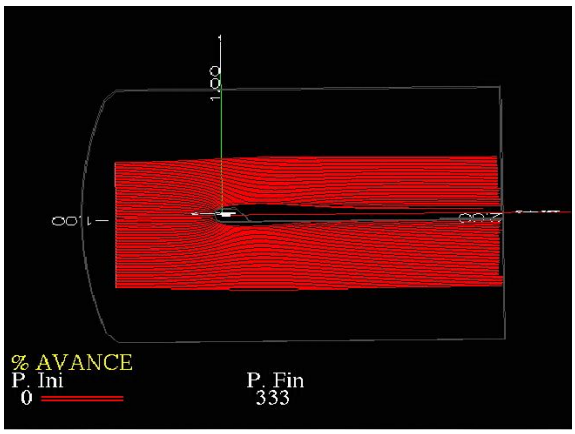
Este capítulo muestra algunas de las visualizaciones que se han realizado con el sistema. Se da una breve descripción de la simulación y se indican los parámetros utilizados.

6.1. Visualización de fluidos estacionarios

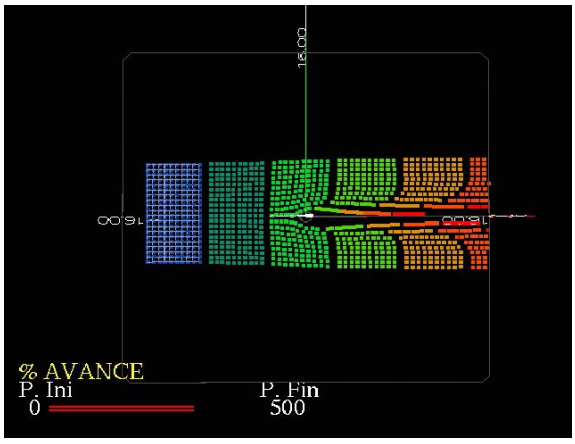
FLUJO LAMINAR

Descripción: Se muestran simulaciones de fluidos con flujo laminar en donde las partículas pasan alrededor de un ala delta y un cilindro.

FLUJO LAMINAR - ALA DELTA	
Parámetro	***Valor***
Técnica de visualización	path lines
Tipo de malla	curvilínea
Tipo de inyector	tubular
No. de partículas	5000
Tiempos de Simulación	333



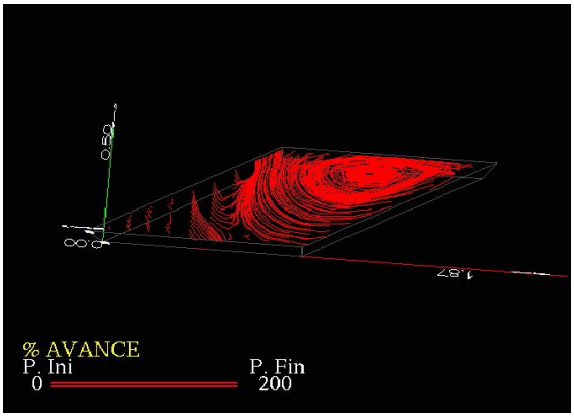
FLUJO LAMINAR - CILINDRO	
Parámetro	***Valor***
Técnica de visualización	streak lines
Tipo de malla	curvilínea
Tipo de inyector	rejilla
No. de partículas	1600
Tiempos de Simulación	500



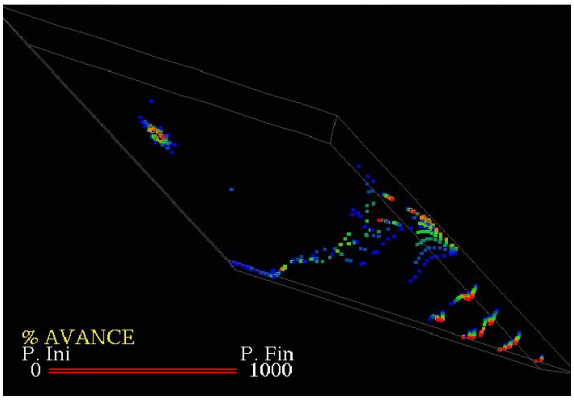
FLUJO TURBULENTO

Descripción: Se muestran diferentes simulaciones de flujos turbulentos en donde las partículas forman remolinos.

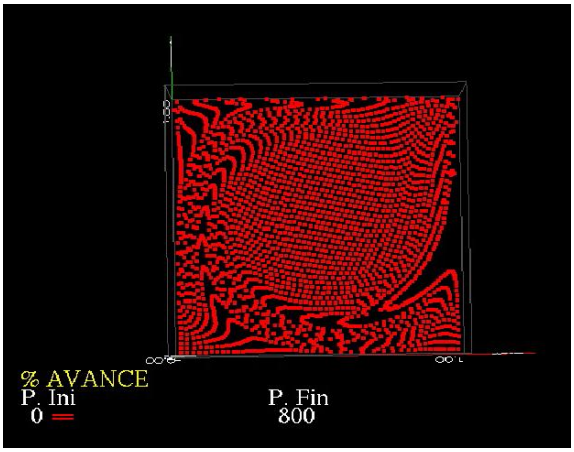
TURBULENCIA - REMOLINO	
Parámetro	***Valor***
Técnica de visualización	path lines
Tipo de malla	curvilínea
Tipo de inyector	rejilla
No. de partículas	5000
Tiempos de Simulación	200



TURBULENCIA - REMOLINO	
Parámetro	***Valor***
Técnica de visualización	streak lines
Tipo de malla	curvilínea
Tipo de inyector	rejilla
No. de partículas	15000
Tiempos de Simulación	1000



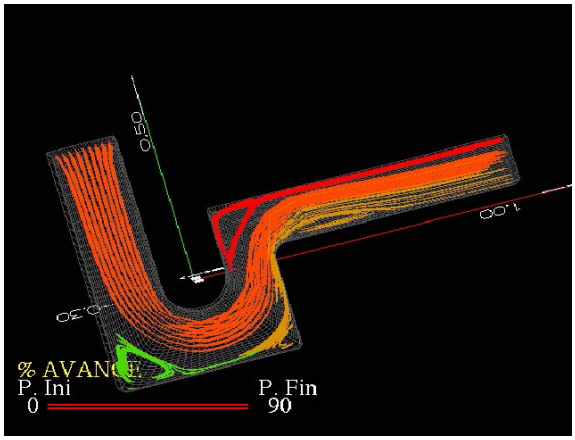
TURBULENCIA - REMOLINO	
Parámetro	***Valor***
Técnica de visualización	time lines
Tipo de malla	cartesiana
Tipo de inyector	rejilla
No. de partículas	5000
Tiempos de Simulación	800



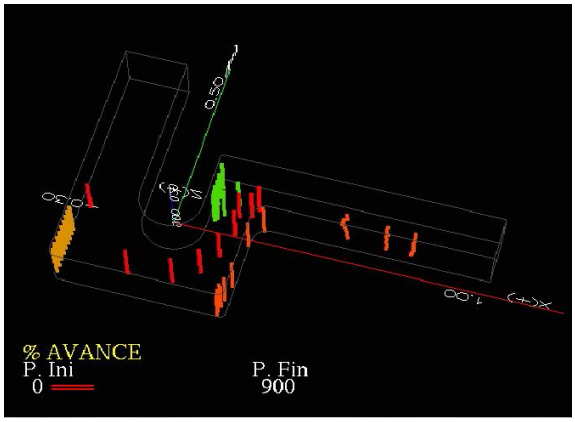
T U B E R I A

Descripción: Se muestra la formación de vórtices en una tubería con inyectores en distintas posiciones.

T U B E R I A	
Parámetro	***Valor***
Técnica de visualización	path lines
Tipo de malla	curvilínea
Tipo de inyector	rejilla
No. de partículas	5000
Tiempos de Simulación	90



T U B E R I A	
Parámetro	***Valor***
Técnica de visualización	TIME lines
Tipo de malla	curvilínea
Tipo de inyector	rejilla
No. de partículas	5000
Tiempos de Simulación	900

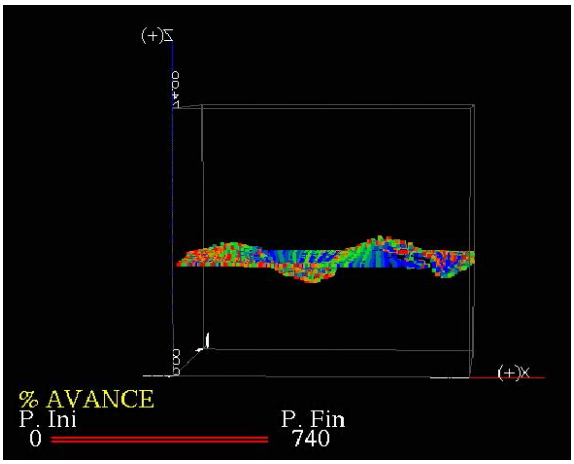


6.2. Visualización de fluidos no estacionarios

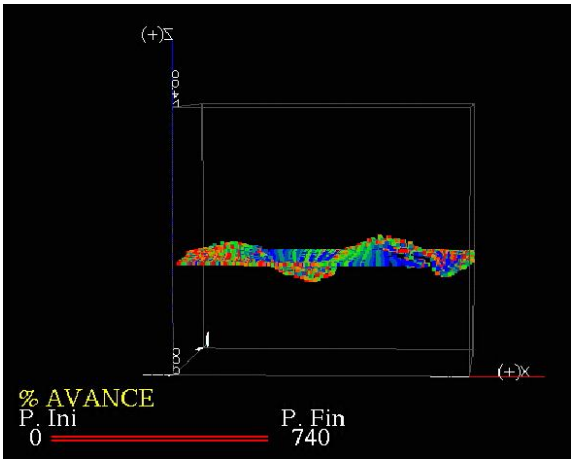
EXPLOSIÓN DE CHIMENEA DE AMONIACO - JETT

Descripción: Se visualiza el fenmeno de la explosin o salida violenta de un tubo o turbina.

EXPLOSIÓN - JETT	
Parámetro	***Valor***
Técnica de visualización	streak lines
Tipo de malla	cartesiana
Tipo de inyector	rejilla
No. de partículas	30000
Tiempos de Simulación	740



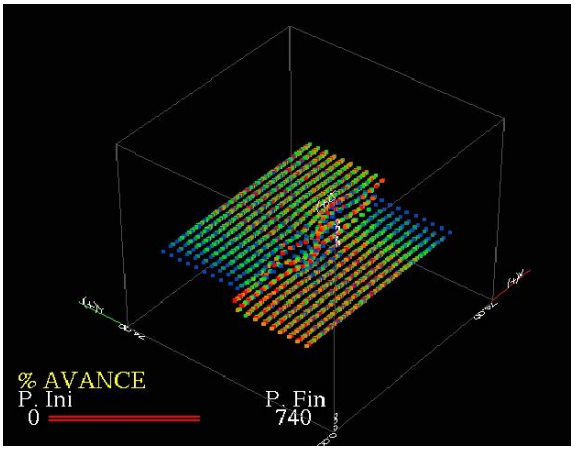
EXPLOSIÓN - JETT	
Parámetro	***Valor***
Técnica de visualización	time lines
Tipo de malla	cartesiana
Tipo de inyector	rejilla
No. de partículas	10000
Tiempos de Simulación	740



ROCE DE PLACAS

Descripción: Se muestra la simulación de fluido en donde dos placas se desplazan en direcciones opuestas y rozan.

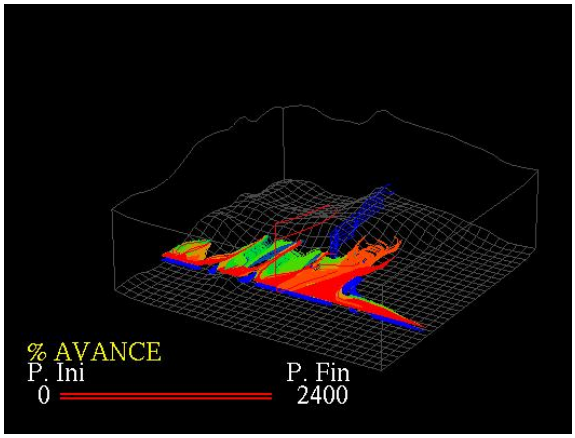
ROCE DE PLACAS	
Parámetro	***Valor***
Técnica de visualización	streak lines
Tipo de malla	cartesiana
Tipo de inyector	rejilla
No. de partículas	20000
Tiempos de Simulación	740



DISPERSIÓN DE PARTÍCULAS CONTAMINANTES

Descripción: Se visualiza la dispersión de partículas contaminantes a causa del viento.

DISPERSIÓN PARTÍCULAS	
Parámetro	***Valor***
Técnica de visualización	path lines
Tipo de malla	curvilíneas
Tipo de inyector	tubular
No. de partículas	70000
Tiempos de Simulación	2400



Capítulo 7

Resultados y evaluaciones

Para poder realizar un análisis del rendimiento de la solución técnica, se desarrollaron dos variantes: la primera consiste en cargar en memoria (*RAM*) el campo de velocidades de la simulación (archivos ascii) y posteriormente iniciar los cálculos; la segunda consiste en acceder de manera directa a los datos que contienen el campo de velocidades utilizando accesos directos en archivos binarios en la fase de ejecución *(I/O)*.

7.1. Casos de evaluación *RAM vs. I/O*

Dentro de los dos enfoques propuestos (*RAM e I/O*) se analizaron dos casos; el primero fué en la variación del número de inyectores, y el segundo en la variación de la cantidad de partículas.

La siguiente tabla muestra la nomenclatura definida para las actividades que se realizan en el ciclo de la visualización en términos de procesamiento:

Nomenclatura	
RMM	Lectura y carga de la malla 3D en memoria.
RMCV	Lectura y carga de todos los archivos de muestreo con las velocidades en cada nodo de la malla en memoria.
CPI	Cálculo de las posiciones iniciales de las partículas que se van a liberar en el Inyector.
RMPI	Reserva memoria para almacenar las posiciones iniciales del Inyector.
CPS	Cálculo del número de pasos a simular en base a la delta de tiempo y número de archivos de muestreo.
RMV	Reservar memoria para almacenar las posiciones de todas las partículas del inyector para cada paso de simulación.
LAV	Lecura en los archivos de muestreo, de velocidades en los nodos del tetraedro donde se encuentran las partículas.
CPP	Cálculo de la nueva posición de las partículas en cada paso de simulación.
APR	Almacenamiento de las posiciones de la partícula en memoria.
PP	Pintado de las partículas.

Tabla 1: *Tabla que describe las actividades en un ciclo de procesamiento de la visualización.*

7.1.1. Variación en el número de inyectores.

Las condiciones iniciales para realizar esta pruebas fueron:

- Equipo SGI Onyx 2 con cuatro procesadores y 512Mb de memoria RAM.
- Un tiempo de muestreo de 27 minutos.

- 740 pasos de simulación.

- Nmero de partículas por inyector 10000.

- Técnica utilizada *time lines*.

El siguiente tabla se muestran los pasos de procesamiento que el ciclo de la visualización sigue en función del número de inyectores.

Simulación con carga de campo de velocidades en mem. RAM

	Carga Inicial		Simulación														
			Ciclo I													Ciclo II	
			Inyector I						Inyector II							Inyector I	Inyector II
	RMM	RMCV	CPI	CPS	RMPI	RMV	CPP	APR	CPI	CPS	RMPI	RMV	CPP	APR	PP	PP	PP
PROC	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
RAM	x	x			x	x			x			x	x				
IO	x	x															

Simulación acceso directo a los datos del campo de velocidades (I/O)

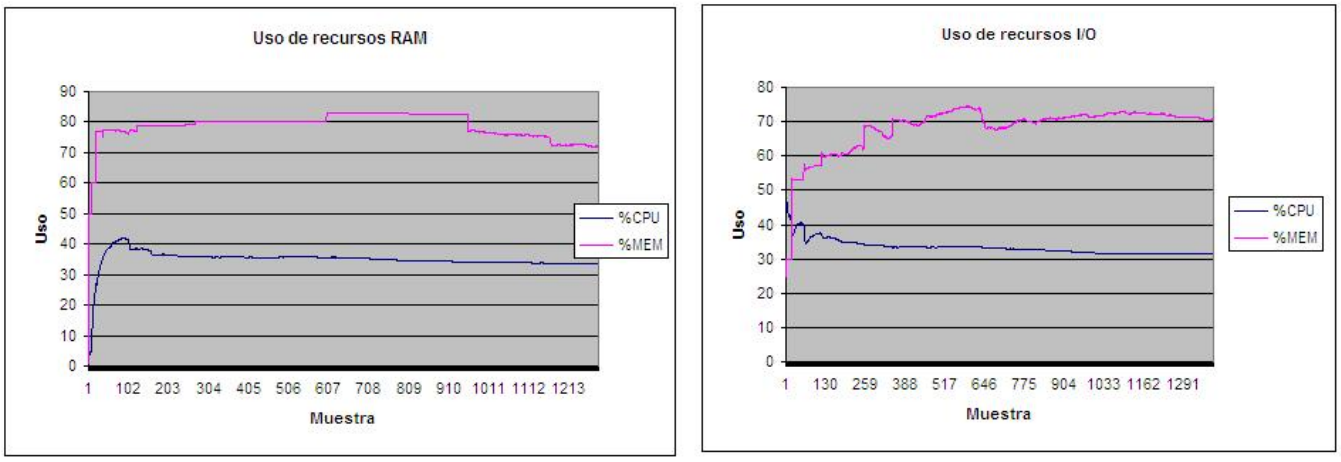
	rga Inic	Simulación																	
		Ciclo I																Ciclo II	
		Inyector I								Inyector II								Inyector I	Inyector II
	RMM	CPI	CPS	RMPI	RMV	LAV	CPP	APR	PP	CPI	CPS	RMPI	RMV	LAV	CPP	APR	PP	PP	PP
PROC	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
RAM	x				x								x						
IO	x					x								x					

Tabla 2: *Pasos de procesamiento de la visualización en función del número de inyectores.*

Los resultados obtenidos son los siguientes:

- El tiempo de procesamiento de CPU en ambos casos de comporta de una forma similar en este tipo de equipos, es decir que acceder a archivo de forma directa no representa un sobre esfuerzo en el procesamiento de datos (Gráfica 1 izquierda).

- El tiempo de procesamiento cuando se carga el campo de velocidades en memoria RAM, si muestra un incremento con respecto al cuando se varía la cantidad de inyectores y no hay una mejora considerable con respecto al rendimiento de la visualización (Gráfica 1 derecha).



Gráfica 1: *Resultado del monitoreo RAM/IO para variación de inyectores.*

7.1.2. Variación en el número de partículas.

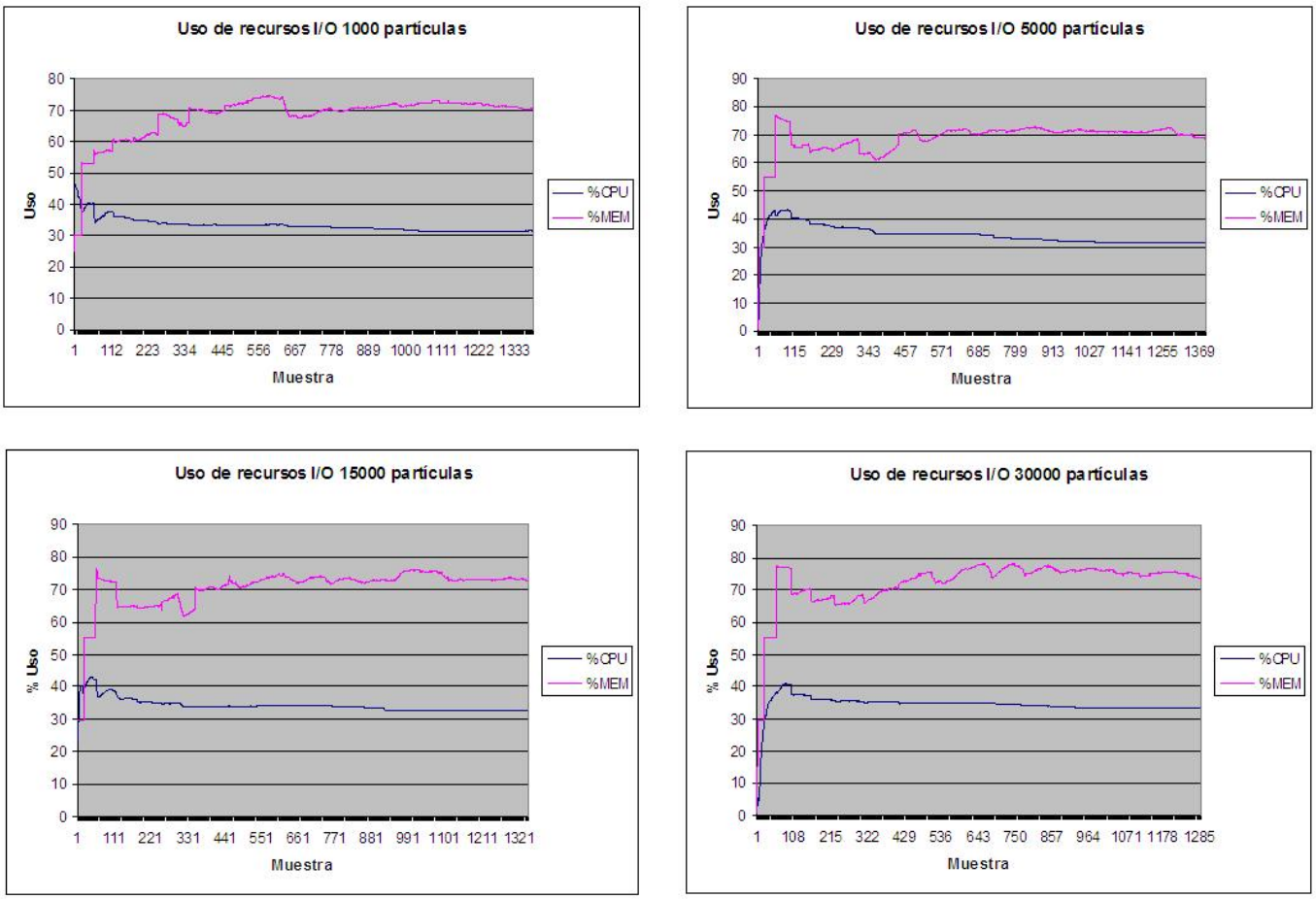
Debido a los resultado en el esquema anterior, se dicidió realizar un muestreo exclusivamente de (I/O) para un número imcremental de partículas.

Las condiciones iniciales para realizar esta pruebas fueron:

- Equipo SGI Onyx 2 con cuatro procesadores y 512Mb de memoria RAM.
- Un tiempo de muestreo de 27 minutos.
- 740 pasos de simulación.
- Nmero de partículas Rango de [1000 - 30000].
- Técnica utilizada *time lines*.

Los resultados obtenidos son los siguientes:

- El porcentaje de uso para el procesamiento de CPU y MEM se mantienen estable con el incremento gradual de partículas.
- El No hay una variación significativa en el tiempo de visualización con 30000 partículas a 740 pasos de simulación.



Gráfica 1: *Resultado del monitoreo RAM/IO para variación de inyectores.*

Con esto se concluye que el mejor escenario es la utilización de acceso directo a archivos binarios (I/O), ya que el esquema de RAM en equipo más pequeños tendrá una limitante inmediata en memoria. Al utilizar el campo de velocidades en archivos binarios, no sólo implica una ganancia en memoria sino que implica que se puedan procesar volúmenes de muestra considerables.

Capítulo 8

Conclusiones

Después de haber desarrollado el presente trabajo, se puede concluir que:

- Se ha cumplido satisfactoriamente con el objetivo inicialmente planteado, el cuál era crear un sistema de visualización de partículas en tiempo real, utilizando algoritmos y herramientas de cómputo de alto rendimiento.

- Se pudo comprobar que con el sistema desarrollado, es posible observar la representación de la simulación numérica de un fenómeno y ganar intuición con las técnicas de visualización en tiempo real de forma inmediata, decidir en dónde explorar y con qué técnica hacerlo y entonces lanzar otra visualización y así llegar al entendimiento del fenómeno y lograr la visualización definitiva.

- Los inyectores desarrollados implementados en el sistema, permiten una exploración más efectiva y eficiente, ya que pueden liberarse partículas que abarquen regiones amplias, que se ajusten a la geometría de la malla y también que revelen el fenómeno en regiones particulares, sin distracciones sobre áreas ya revisadas o de poco interés.

- Pudo responder a las necesidades de procesamiento requeridas para tiempo real, con visualizaciones que abarcan un volúmen importante de datos y con un número significativo de cálculos para la visualización.

- El sistema cumple con el objetivo de servir como una herramienta de apoyo para los servicio de visualización que ofrece la DGSCA, así como ser un sistema flexible que pueda utilizarse como herramienta docente para quien esté interesado en la visualización de fluidos en tiempo real.

- El sistema representa una mejora sustancial con respecto a una visualización de fluidos en modo *batch*, ya que se pueden tomar decisiones en el momento y no es necesario esperar que se terminen los cálulos.

- El sistema puede soportar visualizaciones de fluidos no estacionarios y estacionarios.

- El algotitmo utilzado para la técnica de seguimiento de partículas, es muy eficiente para visualizaciones en tiempo real, tanto en la parte de rendimiento computacional, como en la eficiencia del resultado de sus cálculos.

- El sistema de visualización cumple con los requerimientos de portabilidad entre plataformas, alta calidad y rendimiento en el despliegue de gráficos tridimencionales, alta interactividad, adecuado manejo del hardware de la máquina (administración de la memoria RAM, acceso directo a archivos binarios, técnicas de programación eficientes), código flexible para un fácil mantenimiento de futuras modificaciones y versiones.

- El sistema es muy interactivo y permite realizar cambios en el Modelo de Proyección de las diferentes técnicas de visualización, elegir la posición inicial de los inyectores, control de la cámara para tener el ángulo de proyección deseado, liberar partículas durante la visualización, modificar el paso de simulación, eliminar inyectores que no sean significativos y el cambio de geometrías para futuras visualizaciones en donde se haga un mapeo de texturas para indicar otras características del fluido.

BIBLIOGRAFÍA

1. Anderson, John D. Jr., *Computational Fluid Dynamic, The basics with aplications*, USA, Mc Graw Hill, 1995, 543 pp., ISBN 0-07-001685-2.

2. Blohm, Ch.; Kuhlmann, H. C., "The Two-Sided Lid-Driven Cavity: Experiments on Stationary and Time-Dependent Flows", *Journal of Fluid Mechanics*, vol. 450, USA, Cambrige University Press, january, 2002, pp. 67-95.

3. Bjarne, Stroustrup, *The C++ Programming Language*, USA, Addison-Wesley, 1997, special edition, 969 pp., ISBN 0-201-88954-4.

4. Das, Debopam; Arakeri, Jaywant H., "Transition of Unsteady Velocity Profiles with Reverse Flow.", *Journal of Fluid Mechanics*, vol. 374, USA, Cambrige University Press, june, 1998, pp. 251-283, .

5. Ferziger, J. H.; Perić, M., *Computacional Methods for Fluid Dynamics*, Germany, Berlin, Springer-Verlag, 1997, third edition, 421 pp., ISBN 3-540-42074-6.

6. Fox, Robert W.; McDonald, Alan T., *Introducción a la Mecánica de Fluidos*, México, Mc. Graw Hill, 1992, 4ta. edición, 916 pp., ISBN: 970-10-0669-0.

7. Habashi, Wagdi G., *Solution Techniques for Large-Scale CFD problems*, UK, Wiley, Chichester, 1995, 454 pp., ISBN: 978-0-471-95810-9.

8. Kenwright, David N.; Lane, David A., "Interactive Time-Dependent Particle Tracing using Tetrahedral Decomposition", *IEEE Transactions on Visualization and Computer Graphics*, vol. 2, Num. 2, USA, june, 1996, pp. 120-129, ISSN 1077-2626.

9. Kilgard, Mark J., *The OpenGL Utility Toolkit (GLUT) Programming Interface, API Version 3*, [en línea], USA, November 13, 1996, Dirección URL: **http://www.opengl.org/resources/libraries/glut/glut-3.spec.pdf**.

10. Mathews, John H.; Fink, Kurtis D., *Métodos Numéricos con Matlab*, México, Prentice Hall, 2000, 3ra edición, 736 pp., ISBN: 84-8322-181-0.

11. Puente, Jorge, *SIGGRAPH - Introducción a la Visualización Científica*, [en línea], México, Noviembre 14, 1995, Dirección URL: **http://www.siggraph.org.mx/sm-esp/boletin/sm-bol14.html**.

12. Ramos, E.; Cisneros, G; Fernández-Flores, R.; Santillán-González, A., *Computacional Fluid Dynamics, Proceedings of the Fourth UNAM Supercomputing Conference*, USA, New Jersey, World Scientific, 2001, 300 pp., ISBN 981-02-4535-1.

13. Watt, A.; M. Watt, *Advanced animation and rendering techniques*, USA, New York, Addison-Wesley, 1992, 472 pp., ISBN 0-201-54412-1.

14. Woo, Mason; Neider, Jackie; Davis, Tom, *OpenGL Programming Guide - The Official Guide to Learning OpenGL, Version 1.2*, USA, Addison Wesley, 1999, third edition, 730 pp., ISBN 0-201-60458-2.

15. Wright, Helen, *Introduction to Scientific Visualization*, UK, Springer, 2006, 147 pp., ISBN 1-84628-494-5.

16. Shaw, C. T., *Using computacional fluid dynamics*, UK, Prentice Hall International, 1992, 300 pp., ISBN 0-13928-714-5

17. Zienkiewicz, O. C.; Morgan, K, *Finite elements and approximation*, USA, New York, Jonh Wiley & Sons, 1983, 328 pp.

18. N.D., *Engineering Simulation for the 21st Century*, [en línea], USA, September, 2009, Dirección URL: **http://www.fluent.com/**.