**Universidad de las Ciencias Informáticas**

**Facultad 5 de Entornos Virtuales**

# “Propuesta de un modelo para representación de las estelas de humo en entorno virtual en tiempo real”

**Trabajo de Diploma para optar por el Titulo de Ingeniero en Ciencias Informáticas**

**Autores:**

Yoandris Arias Álvarez

Reiner Becerra Martínez

**Tutor:**

Ing. Yusleidys Guelmes

**Ciudad de La Habana, Cuba**

**Julio 2007**

## DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

Yoandris Arias Alvarez

Firma del Autor

Reiner Becerra Martínez

Firma del Autor

Ing. Yusleidys Guelmes

Firma del Tutor

**DATOS DEL CONTACTO**

Tutor: Ing. Yusleidy Guelmes León

Instructor recién graduado

Municipio: Ciego de Avila

Provincia: Ciego de Avila

Correo: yguelmes@uci.cu

**AGRADECIMIENTOS**

Nuestro agradecimiento a las personas que de una manera u otra hicieron posible este trabajo que da culminación a nuestra formación profesional a lo largo de estos cinco años.

A nuestros tutores, por la ayuda que nos han brindado, por sus orientaciones y preocupación para que el trabajo quedara con la calidad deseada.

A nuestros profesores, por todo lo que nos han enseñado tanto en el ámbito profesional como el lo personal

A nuestras familias, por su apoyo incondicional en todo memento y por darnos un ejemplo a seguir.

A Todos nuestros amigos, por estar haber estado cuando los hemos necesitado durante estos 5 años.

A todos aquellos que confiaron en nosotros desde el principio por mantener siempre esa confianza para con nosotros.

A nuestra Revolución y a Fidel por hacernos partícipes de este proyecto tan emprendedor.

## DEDICATORIA

Dedicamos este trabajo a nuestros padres y abuelos quienes nos han apoyado constantemente durante todo este tiempo que nos hemos superado profesionalmente.

A nuestras hermanas, hermanos y familiares en general, que siempre nos han ayudado a conseguir nuestros sueños.

A nuestros profesores y amigos que siempre nos han acompañado durante todo este tiempo.

“La mayoría de las ideas fundamentales de la ciencia
son esencialmente sencillas y, por regla general pueden
ser expresadas en un lenguaje comprensible para todos ”.

***Albert Einstein***

**Resumen**

Este proyecto esta enmarcado en la simulación dinámica de una *estela* de humo dejada por un cohete en un entorno virtual. Durante el desarrollo del trabajo se tienen en cuenta la nueva generación de hardwares gráficos con GPUs programables la cual fomenta muchas maneras para lograr tales efectos en juegos de computadora y ambientes virtuales.

El trabajo proporciona un breve análisis de las diversas técnicas disponibles para la simulación de fenómenos gaseosos y presenta una comparación entre estas con el objetivo de determinar la más conveniente para lograr el efecto deseado. El modelo propuesto a partir de las ventajas vistas en la comparación representa visualmente el efecto de una *estela* de humo en un mundo virtual, capaz de obrar iterativamente con el mismo. Además de ser bastante rápido y funcionar eficazmente en un ambiente simulado.

Así, el trabajo pretende ser útil como herramienta de soporte al *simulador* de tiro anti-aéreo de SIMPRO con el objetivo de lograr una mejor formación de los soldados. Al finalizar el mismo se dan algunas recomendaciones para su desarrollo futuro.

**Palabras Clave:**

Simulador, Entorno Virtual, Estela, Humo, Sistemas de partículas, Emisor, Particula, Fluido, Dinámica fluida computacional.

## Índice

## Introducción

Con el desarrollo acelerado de las tecnologías informáticas, la información y las comunicaciones. Las actuales tendencias del mundo en que nos desarrollamos se encuentran revolucionadas creando un mundo interdependiente, caracterizado por la competitividad multifacética e impredecible, que obliga a las organizaciones a la creación de productos que cada vez requieran de mayor calidad y facilidad de manipulación al cliente.

Los *simuladores* han existido desde la década de los 80, históricamente relacionados con distintos aspectos de la vida cotidiana y militares, como por ejemplo los ya conocidos *simuladores* de tiro, de vuelo y conducción. Es a partir de los años 2000 cuando el avance de estas técnicas desde el punto de vista profesional ha evolucionado los *simuladores* hasta el punto de crear verdaderos mundos virtuales capacitados con la calidad necesaria para el aprendizaje de sus usuarios con diferencias muy insignificantes a como lo harían en el mundo real.

Han tomado un gran auge en los últimos tiempos llegando a convertirse en una necesidad para mejorar la organización, control y eficiencia de las entidades no solo a nivel mundial sino también en las cubanas. Es por ello que se hace necesaria la búsqueda de una alternativa que incluya la utilización de las nuevas tecnologías de la información en la perfección de estos *simuladores*.

Para poder completar con éxito la elaboración de un buen *simulador*, se necesita tener un control riguroso sobre todo los elementos que lo conformaran, como por ejemplo, modelo matemáticos, físicos y mecánicos. Actualmente en los productos desarrollados por los informáticos de la empresa de SIMPRO se han identificado elevados números de inconformidades por la necesidad de introducir dentro del *simulador* de tiro anti-aéreo, las *estelas* de humo que le darían un mayor grado de realismo al desplazamiento de los cohetes de la defensa anti-aérea particularmente.

Por lo que una de las preguntas más importante en estos momentos es ¿Cómo lograr simular el comportamiento de las *estelas* de humo en un entorno virtual en tiempo real?

Por lo que se tiene como objetivo principal de este trabajo investigativo proponer un modelo a partir de la comparación y análisis de varios modelos respecto a sus metodologías y resultados para la simulación de las *estelas* de humo de un cohete C10 en un mundo virtual en tiempo real.

Para cumplir este objetivo las tareas a realizar en esta investigación incluyen:

1. Obtener información visual sobre el lanzamiento del cohete.

2. Buscar he investigar los fundamentos teóricos del objeto de estudio.

3. Investigar sobre la documentación existente sobre técnicas utilizadas para la representación de fenómenos gaseosas en un entorno virtual.

4. Estudiar los modelos físicos y matemáticos sobre el comportamiento del humo que compone la *estela* dejada por un objeto.

5. Entrevistar a especialistas que trabajen en el campo del *simulador* de tiro anti-aéreo en la empresa de SIMPRO.

6. Valorar los modelos existentes y seleccionar el adecuado.

7. Determinar los software más adecuados para llevar a cabo la implementación del modelo.

8. Determinar los estándares de codificación mas adecuados para el modelo seleccionado.

Se tiene como campo de acción “Modelar la representación de *estelas* de humo dentro de los *simuladores* de tiro anti-aéreo”, que es la parte del objeto de estudio que se va a investigar.

Se esperan aportes fáciles y flexibles para diferentes situaciones a la hora del lanzamiento del cohete que servirán para lograr una mayor eficiencia y realidad del ambiente simulado. Logrando una mayor posibilidad del el éxito en el ejercicio de tiro. La inserción de este componente al *simulador* imprime una insignia de eficiencia y calidad, pues fomentara el renacimiento y adquisición de nuevas tácticas que estarían encaminadas al servicio y conocimientos de los usuarios.

Para esto este proyecto investigativo se consta de 3 capítulos:

Capitulo 1: El capítulo dará una descripción de los distintos tipos de ambientes virtuales que existen, además de abordar temas referentes a la simulación y a las diferentes técnicas para producir animaciones de fenómenos gaseosos.

Capitulo 2: Este capitulo proporcionara una descripción de la tecnología propuesta para la programación del modelo seleccionado.

Capitulo 3: Este capítulo muestra la comparación entre las principales técnicas conocidas para la representación de los fenómenos fluidos y la proposición de una de estas, seleccionada o a partir de sus

ventajas con relación a las demás. Además de dar una descripción de cómo puede ser implementado el modelo seleccionado para simular las *estelas* de humo.

## Capítulo 1: Fundamentación Teórica

Este capítulo hace alusión a los entornos virtuales y a la simulación, apunta a dar una descripción de las técnicas convenientes para producir animaciones de fenómenos gaseosos, que pueden ser usadas para lograr la creación de *estelas* de humo discutiéndose sus fuerzas y debilidades relativas. En este además se abordaran temas como los antecedentes del problema, balance crítico de los logros y limitaciones de las teorías e investigaciones anteriores.

### 1.1 Representación de entornos virtuales:

#### 1.1.1 ¿Qué entendemos por entornos virtuales?

El cine ha sido uno de los primeros medios en rentabilizar la creación de realidades virtuales, creando escenas imposibles como es la filmación de los dinosaurios o el hundimiento del Titanic, o la aparición de cientos de robots que no existen, e incluso la resucitación de actores ya fallecidos que vuelven a actuar ante las cámaras. Esta técnica *infográfica* nos ha permitido estar presentes en lugares y momentos antes nunca pensados.

El “entorno real” ha sido siempre el espacio de socialización en el que se producen las situaciones de aprendizaje. Los agentes socializadores tradicionales, familia y escuela, vieron cómo la aparición de los grandes medios de comunicación de masas ha ido tomando ventaja en esta labor hasta el punto que se convirtieron en los agentes hegemónicos de transmisión de valores comunes.

En la actualidad este “entorno real” está de nuevo perdiendo presencia frente a nuevas formas de representación que se denominan entornos virtuales. Es lo que el filósofo vasco Javier Echeverría ha denominado “el tercer entorno”, un nuevo espacio de comunicación y por tanto de socialización.(ESCLAPEZ 2005)

#### 1.1.2 ¿Puede el entorno virtual reemplazar el entorno social real?

Los entornos virtuales, además de representar la realidad, también pueden construir un mundo irreal. Krauss, R. afirma que “estamos rodeados, se aduce, no por la realidad misma, sino por el efecto de realidad, producto de la simulación y los signos”.

El entorno virtual permite interactuar, colaborar, elegir, comunicar, independencia y autonomía, es un nuevo modelo comunicativo: más participativo, interactivo, divertido y colaborativo. Pero lamentablemente está sometido a estructuras jerárquicas tal como lo está la vida real. El usuario tiene los permisos que le haya otorgado el programador según el perfil que le asigna. Por lo que seria un prácticamente imposible que en un entorno virtual se pueda tener la libertad de tomar decisiones que se puede tener en la realidad y por consiguiente queda claro que el entorno social real no puede ser remplazado por el entorno virtual.(ESCLAPEZ 2005)

### 1.1.3 Tipos de entornos virtuales desde el punto de vista tecnológico:

Lucio Margulis realiza una clasificación de los entornos virtuales según su naturaleza tecnológica:

*Simuladores de sistemas:* son las aplicaciones que emulan el funcionamiento de programas informáticos que se utilizan como tutoriales del manejo de los mismos. Estas aplicaciones guían y corrigen el comportamiento del usuario, creando un entorno virtual previo al entorno real. En este tipo de aplicación, el usuario suele relacionarse con la parte lógica de la máquina.

*Simuladores de entornos:* Se trata de escenarios virtuales creados mediante infografía cuyo objetivo es parecerse lo máximo posible a los escenarios reales y en donde el usuario puede interactuar modificando y creando nuevos elementos. Estos entornos virtuales pueden ser compartidos por varios usuarios a través de la red. Por ejemplo los *simuladores* de vuelo.

*Simuladores de situaciones:* Se suelen producir mediante aplicaciones especiales para animación en 3D y su principal objetivo es introducir al usuario en situaciones relacionales en las que debe tomar decisiones para su posterior evaluación. El entorno laboral virtual en el que el participante debe realizar tareas similares a las del entorno real (atender clientes, conversar a través del teléfono, contratar o despedir personal o realizar los presupuestos) supone un sistema de valoración de las competencias humanas y profesionales de los “jugadores”. Este tipo de programas están diseñados tanto para la interacción entre el usuario y la máquina, así como entre varios usuarios. Un ejemplo es el denominado "Desafío SEBRAE Argentina 2003 ", un juego virtual de simulación empresaria para estudiantes.(ESCLAPEZ 2005)

## 1.2 Simulación

Una definición de simulación es la formulada por R.E. Shannon al plantear que: " es el proceso de diseñar un modelo de un sistema real y llevar al término experiencias con el mismo, con la finalidad de comprender el comportamiento del sistema o evaluar nuevas estrategias dentro de los limites impuestos por un cierto criterio o un conjunto de ellos para el funcionamiento del sistema".

Para lograr una buena simulación hay que tener una definición exacta del sistema que se desea simular, es necesario hacer primeramente un análisis preliminar del mismo, con el fin de determinar la interacción con otros sistemas, las restricciones del sistema, las variables que interactúan dentro del sistema y sus interrelaciones, las medidas de efectividad que se van a utilizar para definir y estudiar el sistema y los resultados que se esperan obtener del estudio.

Con la denominación genérica de sistema de simulación, se encuadran todos aquellos artificios que, de alguna forma u otra, sirven para representar una situación real o virtual, la cual requiere toma de decisiones o bien la ejecución de un determinado tipo de acción.(WIKIPEDIA)

Los sistemas son empleados en las primeras instancias del adiestramiento, y como un medio para adquirir una cierta experiencia previa al empleo de equipos y materiales, cuyo costo de operación es realmente elevado. Estas presentan varias ventajas y desventajas que son enunciadas a continuación.

Ventajas:

- Son los medios de ayuda más adecuados para alcanzar un alto nivel de instrucción.
- Disminuyen los costos de las ejercitaciones en el terreno.
- Prolongan los ciclos de vida útil del material provisto en la Fuerza.
- Contribuyen a la preservación del medio ambiente.
- Reducen los riesgos de accidentes y sus consecuencias.
- Los sistemas del tipo "de salón" permiten el entrenamiento en espacios reducidos.
- Los sistemas del tipo "de duelo" ofrecen un mayor realismo en el entrenamiento.

Desventajas:

- No reemplaza a la ejecución de ejercitaciones en el terreno, ni al tiro con munición de guerra, por más sofisticado y perfecto que sea el sistema de simulación empleado.
- Nunca reemplaza a la realidad, por más que la simulación logre excelencia.
- Inclusive en los sistemas tipo duelo, se puede caer en la creencia que se está realizando un juego.
- El costo de los *simuladores* crece en proporción geométrica, respecto de la calidad de los mismos.(PONZI)

### 1.3 Propósitos de un simulador

Un *simulador* es un aparato que permite la simulación de un sistema, reproduciendo su comportamiento. Los *simuladores* reproducen sensaciones que en realidad no están sucediendo.

El mismo pretende reproducir tanto las sensaciones físicas (velocidad, aceleración, percepción del entorno) como el comportamiento de los equipos de la máquina que se pretende simular. Para simular las sensaciones físicas se puede recurrir a complejos mecanismos hidráulicos comandados por potentes ordenadores que mediante modelos matemáticos consiguen reproducir sensaciones de velocidad y aceleración. Para reproducir el entorno exterior se emplean proyecciones de bases de datos del terreno. A este entorno se le conoce como "Entorno Sintético".

Los *simuladores* más complejos son certificados por las autoridades competentes. En el caso de los *simuladores* de vuelo el certificado lo realiza la organización de aviación civil de cada país, que proporciona a cada *simulador* un código indicando su grado de realismo. En los *simuladores* de vuelo de mayor realismo las horas de entrenamiento contabilizan como horas de vuelo reales y capacitan al piloto para realizar su labor.

El objetivo principal de un *simulador* es que sus usuarios aprendan a utilizar máquinas reales sin tener que utilizar la propia maquinaria. Por lo que todas las características de este apuntan a lograr un ambiente lo mas real posible de la situación que se simula.(WIKIPEDIA)

Principales Características

- Escenarios virtuales con imágenes reales

- Reproducción del sonido real durante la simulación.
- Modelo matemático diseñado para cada simulación.
- Reproducción de cabina real simuladas.
- Simulación de inercias para dar respuesta a las acciones del usuario, a impactos.
- Interfaz de instructor con acceso a información sobre sesiones anteriores y control sobre todos los elementos del sistema.
- Diseño instruccional que permite su máximo aprovechamiento.
- Manual del puesto de instructor.
- Manual de ejercicios.
- Sistema de base de datos de operarios, tanto aprendices como avanzados
- Sistema de seguimiento del programa de aprendizaje
- Obtención de informes y resúmenes
- Visualización de la simulación en el puesto de instructor
- Prácticas guiadas por medio de mensajes en la pantalla del *simulador*
- Registro de operaciones no permitidas y de maniobras peligrosas.

### 1.4 Técnicas convenientes para la producción animaciones de fenómenos gaseosos

Puesto que se desea simular una *estela* de humo que es un tipo de fluido, es importante definir el fluido dando percepción de este.

El fluido efectúa desde una gama de efectos de pequeña escala, tales como agua vertida en un cristal y un humo que se levantan de un cigarrillo, hasta a los efectos de gran escala, tales como sistemas de clima, explosiones y *estelas* de humo. Para dar una idea de los efectos fluidos, se dará una lista de ejemplos:

- *Estela* de humo de un cohete.

- Explosiones: De explosiones locales, por ejemplo de una granada, a las explosiones grandes, creando ondas de la ráfaga.
- Líquidos de gran escala: Ríos, océanos y cascadas.
- Fuego: De velas a los edificios ardientes.
- Humo: Por ejemplo levantándose de una hoguera.
- Niebla: Niebla sin rumbo.

Ver figuras en el anexo 3.

Puesto que la naturaleza de muchos de estos efectos es muy diferente, es impráctico simularlos de la misma manera. Describir la simulación de todos los efectos mencionados sería un trabajo engorroso. Aunque muchos de estos efectos están relacionados.

Para la representación de la *estela* de humo es necesario profundizar en la forma de representar fenómenos gaseosos dentro de un entorno Virtual. La animación realista en tiempo real de fenómenos gaseosos como el humo es un problema importante y desafiador en gráficos de computadora.

El elevado costo computacional asociado a modelar de manera exacta los procesos físicos en tiempo real ha incitado a investigadores a explorar muchas y diversas técnicas.

Las simulaciones pueden estar basadas en la física o pueden no estar basadas en esta. Las simulaciones no basadas en la física tienden a carecer de la regeneración dinámica y no reproducen satisfactoriamente los efectos visuales altamente complejos considerados en el flujo fluido. Una simulación basada en la física es un acercamiento más natural, pero el costo computacional asociado a modelar la geometría del volumen de esta forma presenta un problema cuando es una simulación en tiempo real que es la requerida para juegos de computadoras u otros usos interactivos.

### 1.3.1 Técnicas basadas en la física

Las simulaciones de alta calidad visual se han producido usando soluciones basadas en la física, la mayoría de la investigación en esta área se encamina hacia crear las herramientas de los animadores para soluciones que no son en tiempo real. Diversas técnicas se han aplicado a la modelación de los volúmenes de fluidos con muchos grados de éxito, sin embargo, las capacidades de hardware para

gráficos han aumentado y la llegada de los shaders programables para animaciones en tiempo real eleva a un nivel más realista los gráficos de tales fenómenos complejos.

El área de las matemáticas que describe el flujo fluido es llamada dinámica fluida, y se basa en las ecuaciones formuladas por el físico francés e ingeniero Claude Louis Navier y el matemático irlandés George Gabriel Stokes en el siglo XIX. Las ecuaciones Navier-Stokes son un sistema de ecuaciones diferenciales no lineares que describen la conservación del dinamismo en los líquidos.(POLLITT 2006)

***Navier-Stokes***

Las ecuaciones diferenciales Navier-Stokes son un sistema de las ecuaciones diferenciales parciales, que describen el movimiento de fluidos incompresibles viscosos. Esto significa que todas las características del fluido son descritas exclusivamente por su viscosidad y densidad. Aunque el aire no es realmente incompresible, la mayoría de los efectos del aire observados diariamente en la vida mantienen esta característica. Por lo que se asume que el aire presenta un comportamiento incompresible.

Las ecuaciones Navier-Stokes se basan en la teoría que los líquidos se pueden describir como colecciones de partículas. Los términos describen las fuerzas que actúan en una partícula, derivada observando el comportamiento en un cubo de la unidad alrededor de la partícula. Esto se hace asumiendo que el fluido dentro de este cubo de la unidad se comporta uniformemente.

Las Ecuaciones Navier-Stokes fueron derivadas de la segunda ley de Newton's, que se declara

$$f = m * a \tag{1.1}$$

Donde m es la masa, a es aceleración, y f la fuerza. Esta describe los cambios en un campo de la velocidad, es decir la aceleración del fluido, como suma de las fuerzas actuando en el fluido incluyendo las fuerzas introducidas por el movimiento del propio fluido. En una notación compacta del vector, según lo utilizado en muchos de los trabajos referidos, las ecuaciones Navier-Stokes son:

$$\frac{\partial u}{\partial t} = \frac{1}{\rho} f - (u * \nabla)u + v\nabla^2 u - \frac{1}{\rho}\nabla p, \tag{1.2}$$

$$\nabla * u = 0 \tag{1.3}$$

En lo siguiente, se explica los términos de estas ecuaciones, para dar una intuición de la contribución de cada término al movimiento de un fluido.

Fuerzas Externas

Un fluido se puede afectar por las fuerzas externas. Éstas son las fuerzas, tales como gravedad y viento, que no son generadas por el movimiento fluido, pero afectan el fluido debido a las condiciones externas. La contribución de fuerzas externas es descrita por el término

$$\frac{1}{\rho} f = \frac{1}{\rho}(f^x, f^y, f^z)^T \qquad (1.4)$$

donde el campo de la fuerza $f = (f_x, f_y, f_z)^T$ es la suma de todas las fuerzas externas que trabajan en el fluido, y $\rho$ es la densidad del líquido, que describe la masa de un cubo de la unidad del fluido.

Advección

La fuerza del movimiento fluido que trabaja en el mismo fluido se llama advección. Puede ser considerada como el rebote mutuo entre las moléculas del fluido. Si una molécula topa con otra molécula, la otra molécula se afecta y comenzará a moverse. La contribución de la advección se describe por

$$-(u * \nabla)u \qquad (1.5)$$

donde $\nabla$ es el operador del gradiente, y $u = (u^x, u^y, u^z)^T$ es la velocidad.

Difusión

La difusión ocurre cuando la parte del fluido pasa por un obstáculo, o cuando hay otra parte del fluido con una velocidad diferente. Esto retrasa el fluido y los *Vorticity* aparecen. La contribución de la difusión es descrita por el término

$$\nu \nabla^2 u \qquad (1.6)$$

Donde $\nu$ es la viscosidad cinemática del fluido, que describe el "grueso" del fluido. De ahí, la difusión también es referida como el efecto de viscosidad.

Presión

El fluido que se mueve dentro y fuera del cubo observado de la unidad causa la presión de cambio. Las diferencias de presión entre el cubo de la unidad y sus alrededores afectan la velocidad según lo descrito por

$$-\frac{1}{\rho}\nabla p \qquad (1.7)$$

donde $\rho$ es todavía la densidad del líquido y p es la presión.

Incompresibilidad

Para asegurar que el volumen del fluido se mantiene constante, el flujo neto del cubo de unidad debe ser 0, indicando que las cantidades de entrada de fluido y salida del cubo son iguales. Esto está descrito por la constante de incompresibilidad

$$\nabla * u = 0 \qquad (1.8)\ (\text{RØRBECH 2004})$$

A principios de los 60, Fromm, Harlow y Welch desarrollaron varios *algoritmos* basados en las ecuaciones Navier-Stokes y en el campo de la dinámica fluida computacional (CFD) para simular el flujo fluido en las computadoras electrónicas. Este campo ha avanzado rápidamente mientras que la energía de computadoras ha crecido exponencialmente, a pesar de esto muchos problemas siguen siendo sin resolver, la dinámica fluida computacional ahora abarca un arsenal extenso de diversas técnicas.

Sobre las últimas dos décadas, los investigadores en gráficos de computadora han comenzado la experimentación con muchas de las técnicas de la literatura de la (CFD). Estos campos tienen metas que se diferencian. El (CFD) apunta para una simulación numérica altamente exacta para el estudio científico, mientras que los gráficos de computadora buscan solamente una simulación visualmente exacta. Muchas técnicas del (CFD) se han modificado y se han simplificado para el uso en animación de los flujos fluidos.

Los primeros en modelar el movimiento del humo simulando las ecuaciones de la dinámica fluida fueron Kajiya y Von Herzen en 1984. Desafortunadamente el poder de las computadoras en este tiempo los limitó para trabajar en un punto pobre de detalle y las simulaciones fueron sumamente lentas.

Stam uso un esquema semi-Lagrangian implícito de la integración que es incondicionalmente estable para cualquier paso de tiempo, permitiendo por lo tanto una simulación mucho más rápida. El esquema implícito soluciona las ecuaciones iterativas modificando una solución de ensayo inicial hasta que converge dentro a una gama especificada. Las simulaciones están adicionalmente perfeccionadas por coordenadas de la textura que da advertencia a través del volumen junto con la densidad, creando así el efecto de un flujo complejo, detallado incluso en rejillas bajas de la resolución. Usando un hardware simple dado a lo largo de su solucionador de fluido, Stam pudo producir simulaciones interactivas en tiempo real en cuadrículas de baja resolución.

Si embargo a pesar de ser rápidas, las simulaciones de Stam carecen de los *Vorticity* a pequeña escala y de los movimientos que hacen al flujo fluido tan interesante. El método semi-Lagrangian que aproximaba el flujo fluido sufre de disipación numérica excesiva que hacía los *Vorticity* de pequeña escala se humedecen y desaparecen demasiado rápido.

Este problema fue tratado parcialmente por Stam, Fedkiw y Jensen que tomaron prestada otra técnica de la literatura de (CFD) llamada confinamiento del *Vorticity*. La energía perdida del sistema debido a la disipación numérica es re-inyectada usando un término dinámico el cual aumenta los *Vorticity* del flujo y mantiene en pequeña escala remolinos animados. La adición de confinamiento del *Vorticity* obtiene sólo un costo computacional pequeño y la simulación permanece estable mientras la magnitud del término fuerza permanece debajo de un cierto umbral. Combinaron la solución con un mapa del fotón dado para producir animaciones imponentes de humo, aunque ningunas de las simulaciones eran en tiempo real.

A pesar del éxito de los métodos basados en la rejilla de Eulerian, siguen siendo inadecuados para crear muchos tipos de efectos. La introducción del confinamiento del *Vorticity* por Fedkiw fue una cierta manera a tratar los problemas de la disipación numérica inherentes en el esquema semi-Lagrangian de la integración, pero fue limitada solamente a aumentar la magnitud de *Vorticity* preexistente de la rejilla. Para los efectos altamente turbulentos las resoluciones de la rejilla usadas son inadecuadas debido a que estas son típicamente demasiado gruesas para capturar adecuadamente el nivel necesario del detalle en el flujo, y el confinamiento del *Vorticity* no es verdaderamente eficaz. Por esta razón, mientras que los métodos basados en la rejilla pueden ser muy eficaces para pequeña escala y efectos de humo de baja densidad, no son generalmente capaces de capturar la escala grande o fenómenos violentos tales como explosiones o *estelas* de humo.(POLLITT 2006)

### 1.3.2 Técnicas no basadas en la física

Mientras que las técnicas del (CFD) proporcionan el modelo más exacto de la animación fluida, el costo computacional de solucionar la las ecuaciones sobre la rejilla eficazmente requiere grandes costo para el calculo cuando es usado en aplicaciones en tiempo real.

Una técnica que no esta basada en la física que proporciona una representación visual eficaz, rápida, eficiente, y versátil; es una solución más realista en el hardware de hoy. Los *sistemas de partículas* se pueden considerar una técnica muy eficaz en este aspecto.

***Sistemas de la partícula***

Los sistemas de la partícula son una técnica no basada en la física extremadamente versátil que se puede aplicar a la simulación del humo y de otros fenómenos gaseosos. Un sistema de la partícula modela el volumen sin forma como una colección de partículas primitivas que nacen en el sistema y en un cierto plazo pueden moverse y cambiar la forma antes de que expire el sistema con el tiempo. La animación del volumen total se alcanza usando funciones para controlar cada partícula individualmente. Los sistemas de la partícula tienen una historia larga del uso en gráficos de computadora.

Un *Sistema de Partículas* es una colección de muchas partículas diminutas que en conjunto representan un objeto difuso. A lo largo de un intervalo de tiempo, se generan partículas nuevas dentro del sistema, y luego se mueven, cambian y mueren dentro del sistema. Para generar cada *frame* en una secuencia de movimiento, se deben seguir los siguientes pasos:

(1) Se generan nuevas partículas en el sistema.

(2) A cada nueva partícula se le asignan sus atributos individuales.

(3) cualquier partícula cuyo tiempo de vida en el sistema haya expirado es eliminada.

(4) El resto de las partículas son desplazadas y transformadas de acuerdo a sus atributos dinámicos.

(5) Finalmente se despliega (*render*) una imagen de las partículas vivas en el buffer de cuadros de imagen (*frame buffer*).

El *sistema de partículas* puede ser programado para ejecutar cualquier conjunto de instrucciones en cada paso. Debido a su enfoque procedimental, esta técnica puede incorporar cualquier modelo computacional que describa la apariencia o dinámica del objeto. Por ejemplo, los movimientos y transformaciones de las

partículas pueden ser atados a la solución de un sistema de ecuaciones diferenciales parciales, o los atributos de las partículas pueden ser asignados sobre la base de mecánicas estadísticas. Por lo tanto, es posible tomar ventaja de modelos que han sido desarrollados en otras disciplinas científicas.

Para controlar la forma, apariencia y dinámicas de las partículas dentro del *sistema de partículas*, el diseñador del modelo tiene acceso a un conjunto de parámetros. Los procesos *estocásticos* que aleatoriamente seleccionan la apariencia y movimiento de cada partícula están restringidos por estos parámetros. En general, cada parámetro especifica un rango en el cual la partícula debe permanecer. Normalmente, un rango es especificado proveyendo su media y su varianza máxima.

Las sub-secciones siguientes describen con mayor detalle el modelo básico de un *sistema de partículas*, y como se controlan y especifican a nivel de software.

**Generación de la Partícula**

Las partículas se generan en el *sistema de partículas* mediante procesos *estocásticos* controlados. Un proceso determina el número de partículas que entran al sistema durante cada intervalo de tiempo, esto es, en un *frame* dado. El número de partículas generadas es importante debido a su enorme influencia en la densidad del objeto difuso.

El diseñador del modelo puede escoger controlar el número de partículas que entran al sistema de una de dos formas. En el primer método, el diseñador controla la media del número de partículas generadas en cada *frame* y su varianza. El número de partículas generadas en el *frame f* es:

*NPart = MediaPart + Rand() x VarPart,*

donde *Rand* es un procedimiento que retorna un numero aleatorio uniformemente distribuido entre –1.0 y +1.0, *MediaPart* es la media del numero de partículas, y *VarPart* es la varianza.

En el segundo método, el número de partículas nuevas depende del tamaño en pantalla del objeto. El diseñador del modelo controla el número medio de partículas generadas por unidad de área de pantalla y su varianza. El *sistema de partículas* puede determinar los parámetros de la vista en un *frame* particular, calcular el área aproximada que cubre y establecer el nuevo numero de partículas. La ecuación correspondiente es:

*NParts= ( MediaPart + Rand() x VarPart ) x AreaPantalla*

donde *MediaPart* es la media por área de pantalla, *VarPart* es su varianza y *AreaPantalla* el área de pantalla del *sistema de partículas*. Este método controla el nivel de detalle del *sistema de partículas* y, por lo tanto, el tiempo requerido para desplegar su imagen. Por ejemplo, no hay necesidad de generar 10.000 partículas en un objeto que solo cubre 4 pixeles de la pantalla.

Para que un *sistema de partículas* pueda crecer o decrecer en intensidad, el diseñador puede variar en el tiempo la media del número de partículas generadas por *frame*, utilizando una simple funcional lineal:

*MediaPart = MediaPartInicial + DeltaMediaPart ( f – f0 )*

donde *f* es el *frame* actual y *f0* es el primer *frame* durante el cual el *sistema de partículas* esta vivo. *MediaPartInicial* es la media del número de partículas en ese *frame*, y *DeltaMediaPart* es el radio de cambio. *VarPart* es constante para todos los *frame*s. Es sencillo introducir variaciones para la media y la varianza que sean cuadráticas, cúbicas o inclusive estocásticas.

Para controlar la generación de partículas en el sistema, el diseñador especifica f0 y los parámetros *MediaPartInicial*, *DeltaMediaPart* y *VarPart*.

**Atributos de la Partícula**

Para cada nueva partícula generada, el sistema de la partícula debe determinar los valores para las cualidades siguientes:

(1) posición inicial,

(2) velocidad inicial (velocidad y dirección),

(3) tamaño inicial,

(4) color inicial,

(5) transparencia inicial,

(6) forma,

(7) curso de la vida.

Varios parámetros del *sistema de partículas* controlan la posición inicial de sus partículas. Un *sistema de partículas* tiene una posición en el espacio tridimensional que define su origen. También tiene una orientación, la cual esta determinada por dos ángulos de rotación con respecto al origen de sus sistema de

coordenadas. Un *sistema de partículas* también tiene una forma de generación que define la región alrededor de sus orígenes dentro de la cual se colocan las nuevas partículas generadas.

La forma de generación del *sistema de partículas* también describe la dirección inicial en la cual las partículas se mueven. En una forma de generación esférica, las partículas se mueven alejándose del origen del *sistema de partículas*. En una forma rectangular o circular las partículas se mueven hacia arriba del plano XY, pero se les permite variar en dirección vertical de acuerdo a cierto ángulo de eyección que es otro parámetro. La velocidad inicial de la partícula esta determinada por:

*VelInicial = MediaVel + Rand( )x VarVelo,*

donde *Mediavel* y *VarVelo* son dos parámetros del *sistema de partículas*, la velocidad media y sus varianza.

Para determinar el color inicial de una partícula, se le provee al sistema de un color promedio y la desviación máxima para ese color. La transparencia y el tamaño de la partícula también se determinan a partir de valores medios y desviaciones máximas. Las ecuaciones son similares a la que se utiliza para calcular la velocidad inicial. Un *sistema de partículas* también tiene un parámetro que especifica la forma de cada partícula que genera.

Un sistema de partícula tiene un parámetro que especifique la forma de cada una de las partículas que genera. Las formas de la partícula implementadas son esférica rayadas. Este se utiliza para el movimiento-borroso de las partículas.

**Dinámica de la Partícula**

Las partículas individuales dentro del *sistema de partículas* se mueven en el espacio tridimensional y cambian con el tiempo sus atributos de color, transparencia y tamaño.

Para mover una partícula de un *frame* a otro simplemente se le suma un vector de velocidad a su vector de posición. Para agregar más complejidad, un *sistema de partículas* también usa un factor de aceleración para modificar la velocidad de sus partículas de un *frame* a otro. Con este parámetro el diseñador del modelo puede simular gravedad y causar que las partículas se muevan en arcos parabólicos en lugar de en línea recta. El color de una partícula cambia a través del tiempo controlada por el parámetro de cambio de color. La transparencia y el tamaño de las partículas se controlan exactamente de la misma manera. En

muchas implementaciones, estos parámetros de cambio son globales para todas las partículas en el sistema, pero también podrían ser *estocásticos*.

**Extinción de la Partícula**

Cuando se genera, una partícula se determina en un tiempo de vida moderado en el marco. Mientras que se computa cada *frame*, este tiempo de vida decrece. Se elimina una partícula cuando su tiempo de la vida sea cero.

**Rendering de la Partícula**

Una vez que los parámetros de apariencia y posición de todas las partículas han sido calculados para un *frame*, el *algoritmo* de despliegue dibuja la imagen en pantalla. El problema de desplegar partículas es tan complicado como el de desplegar objetos compuestos por primitivas como polígonos o superficies. Por ejemplo, las partículas pueden bloquear otras partículas que están detrás de ellas y producir sombras o bien pueden ser transparentes y es posible ver a través de ellas. Más aún, las partículas pueden coexistir en la escena con objetos modelados con primitivas basadas en superficie e interceptarlos.

Para reducir la complejidad del *algoritmo* de despliegue se pueden asumir ciertas condiciones en la escena. Primero, es posible asumir que las partículas no interceptan con los objetos basados en primitivas de polígonos, y por lo tanto el *algoritmo* de despliegue solo necesita manejar partículas. Los objetos modelados usando otras técnicas se componen junto con los del *sistema de partículas* en una etapa de post-despliegue. Para que el *sistema de partículas* intercepte otros objetos o este detrás de ellos, el sistema de despliegue debe partir la imagen del *sistema de partículas* en sub-imágenes basadas en los planos de corte definidos en el espacio de coordenadas del objeto. Estas sub-imágenes se combinan con las otras imágenes en la etapa de composición.

Los píxeles que cubre cada partícula están determinados por la transformación de la vista, el tamaño de la partícula y su forma. Todas las formas de las partículas se dibujan con anti-escalonado (*antialiased*) para prevenir el escalonado temporal.

Con este *algoritmo* y estas asunciones, no se necesita reordenar las partículas, ya que simplemente se despliegan en el *frame buffer* en el orden en que sean generadas.

### Jerarquía de la Partícula

El diseñador del modelo puede crear un *sistema de partículas* en el cual las partículas sean *sistemas de partículas* también. Cuando el *sistema de partículas* padre se transforma, también sus *sistemas de partículas* descendiente y las partículas en estos. La media y varianza del color del *sistema de partículas* padre se utiliza para seleccionar la media y varianza del color de los sistemas descendientes usando las mismas ecuaciones presentadas anteriormente. El número de nuevos *sistemas de partículas* generados para cada *frame* está basado en la tasa de generación del sistema padre. De forma similar, el resto de los parámetros del padre afecta a los de los hijos.(COTO 2004)

### Conclusiones

El capitulo concluye después de dar una panorámica de los ambientes virtuales y los tipos de simulación existente. Además de dar una descripción sobre las técnicas basadas y no basadas en la física utilizadas en la simulación de fenómenos gaseosos.

## CAPITULO 2: Tecnologías a Utilizar

En este capítulo se mencionan las tecnologías utilizadas en el proceso de investigación para logar la simulación y rendering de una *estela* de humo. En el desarrollo del mismo también se propone una variada tecnología teniendo en cuenta sus ventajas, características y compatibilidad con los objetivos del trabajo.

### 2.1 Internet

Para la búsqueda de información se utiliza en gran medida Internet debido a la falta de documentación del tema.

Internet es un método de interconexión de redes de computadoras implementado en un conjunto de protocolos denominado *TCP/IP* y garantiza que redes físicas heterogéneas funcionen como una red (lógica) única. De ahí que Internet se conozca comúnmente con el nombre de "red de redes", pero es importante destacar que Internet no es un nuevo tipo de red física, sino un método de interconexión. Aparece por primera vez en 1969, cuando ARPAnet establece su primera conexión entre tres universidades en California y una en Utah. También se usa el término Internet como sustantivo común y por tanto en minúsculas para designar a cualquier red de redes que use las mismas tecnologías que Internet, independientemente de su extensión o de que sea pública o privada.

Cuando se dice *red de redes* se hace referencia a que es una red formada por la interconexión de otras redes menores.(WIKIPEDIA)

### 2.2 Microsoft Windows XP

Sistema operativo (Conocido como *SO*) es un conjunto de programas destinados a permitir la comunicación del usuario con un computador y gestionar sus recursos de una forma eficaz. Comienza a trabajar cuando se enciende el computador, y gestiona el hardware de la máquina desde los niveles más básicos.

Un sistema operativo se puede encontrar normalmente en la mayoría de los aparatos electrónicos que se pueden utilizar sin necesidad de estar conectados a una computadora y que utilicen microprocesadores

para funcionar, ya que gracias a estos se puede entender la máquina y que ésta cumpla con sus funciones (teléfonos móviles, reproductores de DVD y computadoras).

Los sistemas operativos, motivados por su condición de capa software que posibilita y simplifica el manejo de la computadora, desempeñan una serie de funciones básicas esenciales para la gestión del equipo. Entre las más destacables, se pueden reseñar las siguientes:

- Gestionar los recursos del equipo ejecutando servicios para los procesos (programas)
- Brindar una interfaz al usuario, ejecutando instrucciones (comandos) (WIKIPEDIA)

El sistema operativo que se encuentra corriendo en la PC que se lleva a cabo el trabajo es el *Microsoft Windows* que es el sistema que predomina en la mayoría de las computadoras del centro aunque la universidad esta abogando masificación del sistema operativo Linux debido a que pertenece a la familia de software libre. Pese a que *Microsoft Windows* no es un software de código abierto presenta un gran cantidad de ventajas al utilizarse. Seguidamente se da una breve descripción de este sistema operativo.

*Windows XP* es un sistema operativo que fue hecho público el 25 de octubre de 2001 por Microsoft. Se considera que están en el mercado 400 millones de copias funcionando. Las letras "*XP*" provienen de la palabra *experience* ("experiencia" en español).

Windows XP es una línea de sistemas operativos desarrollada por Microsoft, orientada a cualquier entorno informático incluyendo computadoras domésticas o de negocios, computadoras portátiles, las llamadas "Tablet PC" y *media center*. Windows XP es el sucesor de Windows 2000 y Windows ME, y el primer sistema operativo de Microsoft orientado al consumidor que se construye con un núcleo y arquitectura de Windows NT y que se encuentra disponible en versiones para PC de 32 y 64 Bit.

Windows XP a diferencia de sus versiones anteriores presenta mejoras en la estabilidad y de la eficacia de Windows. Presenta una Interfaz gráfica de usuario (GUI) perceptiblemente reajustada, un cambio de Microsoft promovido para un uso más fácil que en las versiones anteriores de Windows. (WIKIPEDIA)

Requerimientos de WindowsXP:

| | Minimo | Recomendado |
|---|---|---|
| Procesador | 233 MHz | 500 MHz o mayor |
| Memoria | 64 MB RAM (funcionamiento limitado) | 256 MB RAM o mayor |
| Video | Super VGA (800 x 600) | Super VGA (800 x 600) con 8 mbs de video o mayor |
| Espacio en Disco Duro | 1.5 GB | 10.0 GB o mayor |
| Unidades | CD-ROM o DVD-ROM | DVD-ROM o mayor |
| Dispositivos | Teclado y mouse | Teclado y mouse |
| Otros | Tarjeta de Sonido, Altavoces, y Auriculares | Tarjeta de Sonido, Parlantes y Auriculares |

### 2.3 Microsoft Office

*Microsoft Office* es una suite ofimática creada por la empresa Microsoft. Funciona oficialmente bajo los sistemas operativos Microsoft Windows y Apple Mac OS. Además de aplicaciones incluye servidores y servicios basados en Web.

Microsoft Office es considerado el estándar "de facto" en programas de productividad, siendo la interoperabilidad con Office uno de los principales escollos para la competencia, ya que los documentos creados por defecto utilizan formatos propios cerrados. Se trata de software no libre.

Microsoft Office incluye:

- Microsoft Word (procesador de texto)
- Microsoft Excel (planilla de cálculo u hoja de cálculo)
- Microsoft PowerPoint (programa de presentaciones)
- Microsoft Access (programa de bases de datos)
- Microsoft Outlook (agenda y cliente de correo electrónico)
- Microsoft FrontPage (editor de páginas Web visual)
- Microsoft Photo Manager (editor fotográfico)
- Microsoft Publisher (editor para crear varios tipos de publicaciones como tarjetas, pancartas, etc.)

- Microsoft InfoPath
- Microsoft OneNote (capturar, organizar y reutilizar las notas en ordenadores portátiles, de escritorio o Tablet PC)
- Microsoft Project (gestión de proyectos)
- Microsoft Visio (editor de flujogramas)
- Microsoft photoDraw – Desfasado
- Microsoft Draw – Desfasado (WIKIPEDIA)

De estos programas se utiliza *Microsoft Word* para la confección del informe de la tesis con el objetivo de que toda la información recopilada y procesada a lo largo de la investigación pueda ser consultada y utilizada por otros especialistas. *Microsoft Word* es un software que presenta una gran cantidad de ventajas al utilizarse. Seguidamente se presenta una descripción del mismo.

*Microsoft Word* es un procesador de texto creado por Microsoft, y actualmente integrado en la suite ofimática Microsoft Office. Originalmente desarrollado por Richard Brodie para el ordenador de IBM con el sistema operativo DOS en 1983. Se crearon versiones posteriores Apple Macintosh en 1984 y Microsoft Windows en 1989, siendo esta última versión la más difundida en la actualidad, llegando a ser el procesador de texto más popular.

Un *procesador de textos* no es más que un programa informático que nos permite editar, dar formato, guardar y modificar documentos escritos en las PC. También son conocidos como procesadores de palabras (permiten la corrección de las mismas).(WIKIPEDIA)

### 2.4 Adobe Photoshop

Por las cualidades y ventajas presentadas por el software, se propone al Photoshop para la creación de las texturas utilizadas para la representación del una *estela* de humo en un entorno virtual. Las texturas son guardadas en formato .bmp para su uso en la simulación. Seguidamente se ven algunas de de las características mas importantes.

Adobe Photoshop es una aplicación informática de edición y retoque de imágenes bitmap, jpeg, gif, elaborada por la compañía de software Adobe inicialmente para computadores Apple pero posteriormente también para plataformas PC con sistema operativo Windows.

Photoshop en sus primeras versiones trabajaba en un espacio bitmap formado por una sola capa, donde se podían aplicar toda una serie de efectos, textos, marcas y tratamientos. En cierto modo tenía mucho parecido con las tradicionales ampliadoras. En la actualidad lo hace con múltiples capas.

Aunque el propósito principal de Photoshop es la edición fotográfica, este también puede ser usado para crear imágenes, efectos, gráficos y más en muy buena calidad.

Photoshop soporta muchos tipos de archivos de imágenes, como BMP, JPG, PNG, GIF, entre otros, pero tiene ciertos formatos de imagen propios como lo son:

- *PSD* (Photoshop Documentó): Es un formato que guarda una imagen como un grupo de capas, métodos de fusión, colores, textos, máscaras, canales de color, canales alfa, trazados, formas, configuración de tonos, entre otras. Éste es un formato muy popular que incluso es soportado por programas de la competencia. Este formato permite trabajar con distintas capas después de haber cerrado el programa, al contrario que el jpeg, pero ocupa mucho más espacio y no se puede abrir con programas como el visor de imágenes y fax de Windows, luego es necesario tener un programa que lea más formatos de imagen si no tienes Photoshop.

- *PSB*: Es una nueva versión del formato PSD, introducida en la versión CS2, diseñado especialmente para archivos mayores a 2 GB.

- *PDD*: Es una versión del PSD que solo soporta las opciones del programa descontinuado PhotoDeluxe.(WIKIPEDIA)

### 2.5 Lenguaje Unificado de Modelado

Para el modelado de los aspectos del sistema implementado se propone utilizar *UML* (Lenguaje Unificado de Modelado). Con el objetivo de facilitar un mejor entendimiento al lector del trabajo.

El *Lenguaje Unificado de Modelado* (*UML*, por sus siglas en inglés, *Unified Modeling Language*) es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad; aún cuando todavía no es un estándar oficial, está apoyado en gran manera por el OMG (Object Management Group). Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales

como procesos de negocios y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables.

Es importante remarcar que UML es un "lenguaje" para especificar y no un método o un proceso, se utiliza para definir un sistema de software, para detallar los artefactos en el sistema y para documentar y construir es el lenguaje en el que está descrito el modelo. Se puede aplicar en una gran variedad de formas para soportar una metodología de desarrollo de software (tal como el Proceso Unificado de Racional) pero no especifica en sí mismo qué metodología o proceso usar.(WIKIPEDIA)

Ofrece nueve diagramas en los cuales se pueden modelar sistemas de software.

- Diagramas de Casos de Uso para modelar los procesos de negocio.
- Diagramas de Secuencia para modelar el paso de mensajes entre objetos.
- Diagramas de Colaboración para modelar interacciones entre objetos.
- Diagramas de Estado para modelar el comportamiento de los objetos en el sistema.
- Diagramas de Actividad para modelar el comportamiento de los Casos de Uso, objetos u operaciones.
- Diagramas de Clases para modelar la estructura estática de las clases en el sistema.
- Diagramas de Objetos para modelar la estructura estática de los objetos en el sistema.
- Diagramas de Componentes para modelar componentes.
- Diagramas de Implementación para modelar la distribución del sistema

Se resalta que para el desarrollo de la ingeniería de software de esta investigación, se utilizo la metodología RUP. Una metodología define Quién, Cómo, Qué, Cuándo y hacer un proyecto.

| | |
|---|---|
| Trabajadores (“quién”) | Define el comportamiento y responsabilidades (rol) de un individuo, grupo de individuos, sistema automatizado o máquina, que trabajan en conjunto como un equipo. Ellos realizan las actividades y son propietarios de elementos. |
| Actividades (“cómo”) | Es una tarea que tiene un propósito claro, es realizada por un trabajador y manipula elementos. |
| Artefactos (”qué”) | Productos tangibles del proyecto que son producidos, modificados y usados por las actividades. Pueden ser modelos, elementos dentro del modelo, código fuente y ejecutables. |
| Flujo de actividades (“Cuándo”) | Secuencia de actividades realizadas por trabajadores y que produce un resultado de valor observable. |

### 2.6 Rational Rose

Seria conveniente el uso de la herramienta Rational Rose por su compatibilidad con el sistema, por su claridad a la hora de representar los artefactos, por la organización en que agrupa los eventos. Además de permitir al usuario facilidad para su uso y ahorrar tiempo de trabajo.

El Rational Rose es una herramienta de modelado visualmente energética para auxiliar en el análisis y diseño de software orientado a objetos. Se usa para modelar el sistema antes de escribir cualquier código, así se puede tener la seguridad de que el sistema sea arquitectónicamente bueno desde el principio.

### 2.7 Microsoft Visual Studio.Net 2005

El Visual Studio.NET, seria una buena propuesta debido a que este entorno contiene el lenguaje de programación Visual C++, ya que es este el lenguaje de programación en que se desarrolla el *simulador*

que se esta elaborando en la empresa de SIMPRO, debido a que es el mas adecuado para trabajar en el ambiente gráfico. Además de muchas de las librerías graficas que se encuentran en la Internet son compatibles con este entorno.

*Visual Studio .NET* es un IDE que vio la luz en Noviembre del 2005. Es para el sistema operativo Microsoft Windows y está pensado, principal pero no exclusivamente, para desarrollar para plataformas Win32. Es un programa compuesto por un conjunto de herramientas para un programador. Puede dedicarse a un sólo lenguaje de programación o bien, poder utilizarse para varios.

- Incorpora .NET Framework 2.0
- Hay más ediciones diferenciadas por el precio y las características.
- Ayuda con refactorización.
- El desarrollo de páginas con ASP.NET ha cambiado.
- Soporte para el nuevo software servidor Team System.
- Añadido soporte de tests para todo tipo de aplicaciones.

Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica GUI.

Visual Studio.NET presenta varios componentes para el uso de sus usuarios, tales como:

- Un editor de texto.
- Un compilador.
- Un intérprete.
- Herramientas de automatización.
- Un depurador.
- Posibilidad de ofrecer un sistema de control de versiones.
- Factibilidad para ayudar en la construcción de interfaces gráficas de usuarios.

Se debe tener en cuenta que a la hora de crear un proyecto se utilizan varios tipos de fichero Estilos de los ficheros. Tales como:

| **Nombre de Clase** | **Tipo de fichero** | **Extensión** |
|---|---|---|
| Cada clase del programa dispondrá de dos ficheros | Un fichero de cabecera<br>Un fichero de implementación | .h o' .hpp<br>cpp |
| App | Aplicación | exe |
| | Librería de enlace dinámico | dll |
| | Librería estática | Lib |

Para la Construcción de una aplicación básica para cualquier proyecto en VISUAL STUDIO.NET se tienen que tener en cuenta los pasos siguientes.

- Abrir la herramienta "**VISUAL STUDIO.NET"**.
- Crear un nuevo proyecto. Desde el menú "**File**", en la opción "**New**" y luego "**Project**".
- Seleccionar lenguaje de programación
- Seleccionar objetivo del proyecto
- Nombrar el proyecto y guardar.

### 2.8 Lenguaje de programación C++

Se debe implementar el modelo en C++ debido a una serie de cuestiones, la principal es que es el lenguaje utilizado en el *simulador* de tiro anti-aéreo en SIMPRO.

El *C++* es un lenguaje de programación, diseñado a mediados de los años 1980, por Bjarne Stroustrup, como extensión del lenguaje de programación C.

Actualmente existe un estándar, denominado ISO C++, al que se han adherido la mayoría de los fabricantes de compiladores más modernos. Existen también algunos intérpretes como ROOT (enlace externo). Las principales características del C++ son el soporte para programación orientada a objetos y el soporte de plantillas o programación genérica (*templales*). Se puede decir que C++ es un lenguaje que abarca tres paradigmas de la programación:

- La programación estructurada,
- La programación genérica
- La programación orientada a objetos.

Además posee una serie de propiedades difíciles de encontrar en otros lenguajes de alto nivel:

- Posibilidad de redefinir los operadores (sobrecarga de operadores)
- Identificación de tipos en tiempo de ejecución (*RTTI*)

*C++* está considerado por muchos como el lenguaje más potente, debido a que permite trabajar tanto a alto como a bajo nivel.(WIKIPEDIA)

Una de las ventajas del C++ es el gran número de compiladores que soportan el leguaje. Algunos de estos son:

- Apple C++.
- Bloodshed Dev-C++.
- Borland C++
- Codeblocks
- Codewarrior C++
- Comeau C++
- Cygwin (GNU C++)
- Digital Mars C++
- DJ Delorie's C++ development system for DOS/Windows (GNU C++)
- Edison Design Group C++ Front End

- Green Hills C++
- HP C++ para UNIX y HP C++ para OpenVMS.
- IBM C++
- Intel C++
- MINGW - Minimalist GNU for Windows.
- The LLVM Compiler Infrastructure.
- Mentor Graphics/Microtec Research C++
- Microsoft C++
- Microsoft Visual C++ Toolkit 2003
- Paradigm C++
- The Portland Group C++
- SGI C++
- Sun C++
- Sun Studio.
- WindRiver's Diab C++

También se tiene en cuenta que el C++ es muy flexible a la hora de incluirle las librerías graficas de OPENGL. En el caso de este trabajo fue necesario incluir a la librería GLAUX.LIB.

*OpenGL* es una especificación estándar que define una API multilenguaje y multiplataforma para escribir aplicaciones que produzcan gráficos 2D y 3D. Fue desarrollada por Silicon Graphics Inc. (*SGI*) en 1992. Su nombre viene del inglés *Open G*raphics *L*ibrary, cuya traducción es biblioteca de gráficos abierta (o mejor, libre, teniendo en cuenta su política de licencias).

OpenGL se utiliza en campos como CAD, realidad virtual, representación científica y de información, *simuladores* o desarrollo de videojuegos, en el que su principal competidor es Direct3D de Microsoft Windows.

A grandes rasgos, OpenGL es una especificación, es decir, un documento que describe un conjunto de funciones y su comportamiento exacto. A partir de ella, los fabricantes de hardware crean implementaciones (bibliotecas de funciones creadas para enlazar con las funciones de la especificación OpenGL, utilizando aceleración hardware cuando sea posible). Dichos fabricantes tienen que superar pruebas específicas que les permitan calificar su implementación como una implementación de OpenGL.

Existen implementaciones eficientes de OpenGL suministradas por fabricantes para Mac OS, Microsoft Windows, Linux, varias plataformas Unix, y PlayStation 3. Existen también varias implementaciones software que permiten que OpenGL esté disponible para diversas plataformas sin soporte de fabricante. Es de señalar la biblioteca de software libre / código abierto Mesa 3D, una API de gráficos basada totalmente en software y completamente compatible con OpenGL. Sin embargo, para evitar los costes de la licencia para ser denominada formalmente como una implementación de OpenGL, afirma ser simplemente una API muy similar.(WIKIPEDIA)

OpenGL 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 el conjunto completo de características de OpenGL (utilizando emulación software si fuese necesario).

La operación básica de OpenGL es aceptar primitivas tales como puntos, líneas y polígonos, y convertirlas en píxeles. Este proceso es realizado por una pipeline gráfica conocida como la Máquina de estados de OpenGL. La mayor parte de los comandos de OpenGL emiten primitivas a la pipeline gráfica o configuran cómo la pipeline procesa dichas primitivas.

OpenGL es una API basada en procedimientos de bajo nivel que requiere que el programador dicte los pasos exactos necesarios para renderizar una escena. Esto contrasta con las APIs descriptivas, donde un programador sólo debe describir la escena y puede dejar que la biblioteca controle los detalles para renderizarla. El diseño de bajo nivel de OpenGL requiere que los programadores conozcan en profundidad la pipeline gráfica, a cambio de la libertad ofrecida en la implementación de *algoritmos* novedosos de renderizado.

## Conclusiones

En este capítulo se han enunciado las características de la tecnología utilizada en el trabajo, además de dar una breve caracterización de estas y se da una descripción de porque utilizar en la implementación del modelo tecnologías como el Visual Studio.NET 2005, que contiene el Lenguaje de Programación Orientado a Objeto Visual C++, además del Lenguaje Unificado de Modelado (UML) y el Proceso Unificado de Desarrollo, el Proceso Unificado de Rational (RUP).

## Capitulo 3: Propuesta del Modelo para la Representación de las Estelas de Humo

Este capítulo muestra la variedad de aspectos que se tienen en cuenta para la selección de la técnica *sistema de partículas* y da una breve introducción para la implementación de esta, en vista a simular las *estelas* de humo, este proceso se hace invisible a la vista del usuario. Para mostrar en la pantalla del ordenador la simulación de este evento solo se lleva a cabo un proceso denominado rendering.

### 3.1 Ventajas del sistema de partícula respecto a las técnicas de la CFD

Por lo que a expresado el documento en capitulo 1 se tiene en consideración que lo mas adecuado para la representación de una *estela* de humo seria la aplicación de un *sistema de partícula*, además se debe de tener en cuenta de que a pesar que la *estela* de humo tiene una gran importancia dentro de la calidad desde el punto de vista visual no requiere de una elevada exactitud a la hora de representar el efecto. Por lo que seria poco práctico gastar tanto recurso en un efecto que no es tan importante dentro de la parte del funcionamiento del *simulador* de tiro de la empresa de SIMPRO. En este epígrafe el documento pretende aclarar la comparación de los requisitos que presenta el modelo a proponer en el trabajo con respecto a las otras técnicas y sus ventajas. Estos se enuncian a continuación:

1- Capacidad de la técnica a usar, esta debe ser capaz de producir simulaciones interactivas en tiempo real.

Algunas técnicas puestas en práctica de la CFD son capaces de resolver este requisito en espacios pequeños, pero las técnicas son demasiado costosas computacionalmente para ser aplicadas a ambientes 3d más grandes en tiempo real. Dado que el efecto se piensa usar para un *simulador* de tiro donde la velocidad en tiempo real es una premisa importante, se quita el uso de una técnica de la CFD. Dando lugar al uso de una técnica no basada en la física (*Sistemas de partículas*).

2- Conservación del volumen del humo.

El volumen de la *estela* de humo debe permanecer al paso del cohete de manera realista y no desvanecerse de manera inmediata, sino con el paso del tiempo como lo haría en la vida real. Mientras que la mayoría de las técnicas de la CFD el mantener la conservación de la masa del volumen de humo es muy costoso computacionalmente con *sistema de partícula* esto se puede resolver con muy pocos

recursos ya que cada partícula al nacer dentro del sistema se le atribuye el tiempo de vida que se desea lo que permite controlar el volumen del sistema muy fácilmente.

3- Adaptabilidad de las técnicas para una variedad de efectos.

La mayoría de las técnicas de la CFD puestas en práctica son generalmente específicas para un efecto en particular y no son fácilmente adaptables. Un modelo basado en *sistemas de partículas* produciría una implementación práctica y versátil además de ser muy flexibles a la hora de adaptarlos a otros efectos paralelos.

4- Puesta en práctica en el hardware GPU.

Los sistemas de partícula y las soluciones CFD basadas en la rejilla son particularmente conveniente para su implementación en el hardware paralelo. Una implementación eficiente del GPU hace uso de la naturaleza paralela del procesador, esto es esencial para producir efectos de alta calidad en tiempo real.

Una técnica no basada en la física apoyada en la implementación de *sistema de partículas* es una opción ideal. Un *sistema de partículas* bien implementado en el GPU debe hacer frente a los requisitos de la velocidad, y ofrece además la ventaja de mayor flexibilidad.

Las simulaciones basadas en la CFD son un área interesante de investigación y su aplicabilidad a juegos y *simuladores* aumentará indudablemente cuando el rendimiento de memoria y la capacidad de procesamiento sigan mejorando. El logro de Stam de simulaciones fluidas basadas en la física en tiempo real y las extensiones implementadas por Krüger y Westermann son impresionante, pero todavía demasiadas caras computacionalmente para ser usado en ambientes complejos como juegos de computadoras, *simuladores* o efectos de grande escala. Además de carecer de la versatilidad de los *sistemas de partículas* debido a la cual su implementación es fácilmente adaptable para producir diferentes efectos.

5- Precisión de cálculo.

Debe ser observado que el modelo que se desea no apunta hacia el nivel del realismo que se puede alcanzar por los sistemas basados en la física que llevan horas para producir las animaciones que solamente tendrán una duración de algunos segundos. En lugar de esto se apunta a producir una clase de animación "semi-realista".

Una animación semi-realista de una *estela* de humo en un ambiente virtual dinámico no es nada nuevo pero si un problema desafiador, Un factor que se debe tener en cuenta debe ser la calidad visual del efecto producido en el ambiente. Con esto se lograría una animación visualmente agradable capaz de un buen funcionamiento en tiempo real con un costo computacional razonable.

Como resultado de la comparación hecha anteriormente en el epígrafe se escoge el modelo basado en *sistemas de partículas* para el proceso de simular *estelas* de humo en un entorno virtual en tiempo real. Esta técnica ha despertado gran motivación debido a la sencillez, simplicidad y eficacia que caracteriza a esta en tiempo de ejecución. Mediante esta técnica se pueden crear escenas realistas con menos gastos de recursos que los métodos físicos estudiados anteriormente.

### 3.2 Estándares de codificación

Para la implementación del sistema de partícula mediante la creación de componentes físicos, que se traducen en ficheros .h y .cpp correspondientes a la implementación en C++. Se recomienda seguir los estándares de codificación del C++ dado que es el leguaje de programación en que se esta implementando el *simulador* de SIMPRO.

*Estándares de Codificación.*

El código sigue los mismos estándares de la Herramienta. El conocimiento de los estándares seguidos para el desarrollo del sistema permitirá un mayor entendimiento del código,

*Nombre de los ficheros:*

Se nombrarán los ficheros .h y .cpp de la siguiente manera:

NameofUnits.h

NameofUnits.cpp

*Constantes:*

Las constantes se nombrarán con mayúsculas, utilizándose el "_" para separar las palabras: MOTOR_COHETE_VIDA_PARTÍCULAS 100000;

*Estructuras:*

Se nombrarán las de la siguiente manera, indicando con "**T**" que es una estructura:

Struct TNameofStruct {…};

*Clases:*

Se nombrarán las de la siguiente manera:

class NameofClass;

*Declaración de variables:*

Los nombres de las variables comenzarán con un identificado del tipo de dato al que correspondan, como se muestra a continuación:

Tipos variable:

bool VarName;

int VarName;

*Métodos*

En el caso de los métodos, se les antepondrá el identificador del tipo de dato de devolución, y en caso de no tenerlo (void), no se les antepondrá nada. Solamente los destructores comenzarán con "~".

bool Function (…);

void Function (…);

destructor:

En caso de los destructor son antepuesto por un "~".

~NameofClass ();

Seguidamente se da una descripción de cómo pueden ser declarados en C++ un las partículas y el emisor de partículas que los pilares del sistema de partícula para crear una *estela* de humo.

### 3.3 Declaración de las partículas

Las partículas que se utilizan para crear la *estela* de humo del motor de un cohete, no son en realidad mas que pequeños planos formados por dos triángulos lanzados en gran número a partir de un punto llamado emisor de partículas, con la particularidad de que estos planos se van haciendo transparentes a medida que pasa el tiempo hasta que la transparencia es total y entonces son borrados de la memoria. Cada partícula creada el sistema tendrá una posición inicial en el espacio tridimensional, un color, una edad, una edad máxima y punteros a la partícula anterior y a la siguiente.

Primero se define la partícula y sus atributos.

```
struct TPartícula
{
        GLfloat posicion[3];
        GLfloat color[4];
        GLuint  edad, edad_max, textura_id;
        TPartícula *sig, *ant;
};
```

### 3.4 Declaración del Emisor partículas

Se define como el corazón del *sistema de partículas* al "emisor de partículas".El emisor es el responsable de la creación del sistema y por lo tanto es el objeto que se debe replicar en la escena tantas veces como se desee replicar el efecto.

El emisor controla el número de partículas y la dirección en la que deben ser emitidas las partículas así como otros parámetros globales.

A cada emisor se le asocian los parámetros que definen la vida de las partículas que libera, el tiempo que tarda en emitir cada una, la textura que le va a asociar a las partículas que genera, si está persiguiendo a un objeto y el tiempo que tarda en morir el emisor de partículas.

El constructor del emisor requerirá de la cantidad máxima de partículas, la posición, la velocidad, la aceleración, el intervalo de tiempo en que tarda en soltar una partícula y el id de la textura a utilizar también requiere el puntero del objeto al que va a perseguir y la posición relativa al mismo.

El parámetro emisor inmortal es para saber si el emisor vive para siempre o no, en el caso de que sea mortal debido a que es destruido el parámetro vida emisor nos da el tiempo de vida de este. La función inserta partícula insertara las partículas en el sistema y la actualiza partícula actualiza el emisor para un intervalo de tiempo. No hace falta implementar el destructor, porque todas las propiedades de la clase son estáticas y las partículas que quedan en la lista de partículas se destruyen solas cuando se les acaba el tiempo de vida. La clase emisor quedaría:

```
class emisor_partículas{

    public:

        GLfloat         posicion[3];
        GLfloat   ultima_posicion[3];
        GLfloat        velocidad[3];
        GLfloat       aceleracion[3];
        GLfloat posicion_relativa[3];
        emisor_partículas(GLfloat *pos, GLfloat *vel, GLfloat *ace, unsigned int inter, unsigned int vid, GLuint t_id, unsigned int v_emisor=0);
        emisor_partículas(Objeto *obj, GLfloat *pos, unsigned int inter, unsigned int vid, GLuint t_id, unsigned int v_emisor=0);

        bool actualiza();
```

```
        ~emisor_partículas();

    private:

        unsigned int vida_partículas;
        unsigned int tiempo_pasado;
        unsigned int intervalo;
        GLuint textura_id;
        bool sigue_a_objeto;
        Objeto *obj;
        bool emisor_inmortal;
        int vida_emisor;

        void inserta_partícula();

};
```

Para ver el código del .cpp de esta clase ver anexo 2.

### 3.5 Dibujar partículas

Lo más complicado para las partículas en general es el dibujado, ya que hay que ordenarlas en orden de profundidad antes de pintarlas para que no se machaque el *z-buffer* unas con otras, de tal manera que el *algoritmo* de dibujado quedaría una cosa parecida a la siguiente para cada *frame*:

- Eliminación de partículas que hayan superado su tiempo de vida.

- Actualización de la transparencia de cada partícula en función del tiempo pasado entre el *frame* anterior y el nuevo.
- Cálculo de la distancia de cada partícula a la cámara.
- Ordenación de todas las partículas en función de esa distancia (en la práctica se ha utilizado el *algoritmo* de ordenación quick sort).

- Dibujado en orden de todas las partículas empezando por las que están mas lejos y terminando por las que estén más cerca orientando cada una para que mire a la cámara de frente.

En esta función primero se halla la distancia a la cámara de cada partícula, se ordenan de más lejos a más a cerca y cuando se vayan a pintar hay que rotarlas una a una para que se queden mirando a la cámara de frente, también se controla que cuando se le acabe el tiempo de vida se borre de la memoria. La función dibujar partículas se declara:

void Dibuja_partículas();

Para ver el código de esta función ver anexo 3.

**Conclusiones**

En este capitulo se ha argumentado el porque la propuesta del *sistema de partículas* sobre otras técnicas para la representación de fenómenos gaseosos. Se da una idea de cómo puede ser declarada la estructura partícula y la clase emisor de partícula puntos pilotos en la implementación para la obtención de una *estela* de humo. También se da una secuencia de la función dibujar partícula debido a su importancia para la representación del efecto.

## Conclusiones Generales

Este trabajo ha presentado una propuesta para lograr la simulación de una *estela* de humo. Por las valoraciones hechas se ha llegado a la conclusión que la forma mas eficiente de representar este efecto es a través del uso de *sistemas de partículas* capaz de obrar recíprocamente con un ambiente dinámico complejo, puesto en ejecución enteramente en el GPU usando todas las técnicas actualizadas convenientes disponibles. Se ha demostrado cómo los sistemas de la partícula satisfacen idealmente su puesta en práctica en hardwares gráficos.

Los sistemas de la partícula son una técnica altamente versátil capaz de producir muchos efectos variados. La puesta en práctica demuestra que un sistema de la partícula puede simular con éxito la interacción de un efecto gaseoso con un ambiente dinámico de una manera semi-realista conveniente para el uso en un ambiente de *simuladores* para computadoras.

El efecto que resulta, aunque no capaz de igualar la calidad producida por animaciones basadas en la física, es comparable con las mejores técnicas puestas en práctica de la CFD para tiempo real. A pesar que se tiene una pérdida de detalle visual se logra una demanda mucho menor de recursos para lograr el efecto. La puesta en práctica de los *sistemas de partículas* aumentan la flexibilidad y la velocidad de la implementación que las técnicas perteneciente a la CFD.

Hay un número de áreas en que los *sistemas de partículas* pueden ser puesto en práctica debido a que tienen gran potencial, por lo que estos todavía pueden seguir siendo extendidos y mejorados. Sin embargo, los requisitos de la base del proyecto se han resuelto con éxito. Además, el sistema de la partícula puesto en ejecución proporciona una plataforma rápida, eficiente y versátil a partir de la cual el trabajo futuro puede ser emprendido.

## Recomendaciones

Con el propósito de seguir mejorando el efecto que causa la *estela* de humo dentro del *simulador* se dan las siguientes recomendaciones.

- Buscar el perfeccionamiento para la implantación del método sistema de partícula para que sea más eficiente y gaste menos recursos.
- Valorar su aplicación en los distintos *simuladores* de SIMPRO que necesiten representar la *estela* de humo dejada tras un objeto.
- Hacer pruebas de rendimiento del método para verificar con exactitud el tiempo y los recursos que la máquina consume.
- Estudiar otros *algoritmos* de ordenación a parte del propuesto (quick sort), prestando gran atención a merge sort dado que presenta gran potencial para reducir los recursos utilizados para la representación de las partículas.

**Referencias bibliográficas**

ESCLAPEZ, T. C. *FORMAS DE REPRESENTACIÓN EN ENTORNOS VIRTUALES*, 2005. [2006]. Disponible en: http://www.madrid.org/escueladeanimacion/noticias/escotono2002/escotono2005/memoria%2005/DOCUMENTACION/mesanntt_entornos_virtuales.pdf

PONZI, T. L. A. *Sistemas de simulación y entrenamiento*, 2006]. Disponible en: http://www.rs.ejercito.mil.ar/Contenido/Nro649/Revista/sistemasimulacion.htm

WIKIPEDIA. *Simulación*, . [2006]. Disponible en: http://es.wikipedia.org/wiki/Simulaci%C3%B3n

WIKIPEDIA. *Simulador*, . [2007]. Disponible en: http://es.wikipedia.org/wiki/Simulador

POLLITT, E. *Real Time Smoke Simulation on Graphics Hardware*, 2006. [2006]. Disponible en: http://www.dcs.shef.ac.uk/intranet/teaching/projects/archive/ug2006/pdf/u3ep.pdf

RØRBECH, M. *Real-Time Simulation of 3D Fluid Using Graphics Hardware*, 2004. [2007]. Disponible en: http://ieeexplore.ieee.org/iel5/9337/29648/01348355.pdf

COTO, E. *Introducción a los Sistemas de Partículas*, 2004. [2007]. Disponible en: http://lcg.ciens.ucv.ve/~ernesto/nds/CotoND200403.pdf

WIKIPEDIA. *Internet*, . [2007]. Disponible en: http://es.wikipedia.org/wiki/Internet

WIKIPEDIA. *Sistema Operativo*, . [2007]. Disponible en: http://es.wikipedia.org/wiki/Sistema_Operativo

WIKIPEDIA. *Microsoft Windows XP*, . [2007]. Disponible en: http://es.wikipedia.org/wiki/Microsoft_Windows_XP

WIKIPEDIA. *Microsoft Office*, . [2007]. Disponible en: http://es.wikipedia.org/wiki/Microsoft_office

WIKIPEDIA. *Microsoft Word*, . [2007]. Disponible en: http://es.wikipedia.org/wiki/Microsoft_Word

WIKIPEDIA. *Adobe Photoshop* [2007]. Disponible en: http://es.wikipedia.org/wiki/Adobe_Photoshop

WIKIPEDIA. *Lenguaje Unificado de Modelado*, . [2007]. Disponible en: http://es.wikipedia.org/wiki/Lenguaje_Unificado_de_Modelado

WIKIPEDIA. *C*++, . [2007]. Disponible en: http://es.wikipedia.org/wiki/C++

WIKIPEDIA. *OPENGL*, . [2007]. Disponible en: http://es.wikipedia.org/wiki/Opengl

**Bibliografía consultada**

ROBERT, D. S. *Exploring Parallelism for Real-Time Smoke Visualisation*, 2005. [2007]. Disponible en: http://www.iam.unibe.ch/publikationen/techreports/2005/iam-05-003/file/at_download

BRYAN E. FELDMAN, J. F. O. B., BRYAN M. KLINGNER. *Animating Gases with Hybrid Meshes*, 2005. [2007]. Disponible en: http://portal.acm.org/ft_gateway.cfm?id=1073281&type=pdf&coll=&dl=&CFID=15151515&CFTOKEN=6184618

HARRIS, M. J. *Fast Fluid Dynamics Simulation on the GPU*, . [2007]. Disponible en: http://leri.univ-reims.fr/~nocent/doc/gpugems_chap38.pdf

MARKHARRIS. *harris.SimulationGPU* [2007]. Disponible en: http://www.gpgpu.org/s2004/slides/harris.SimulationGPU.ppt

RØRBECH, M. *REAL-TIME SIMULATION OF SMOKE USING GRAPHICS HARDWARE*, . [2007]. Disponible en: http://image.diku.dk/projects/media/roerbech.04.pdf

RONALD FEDKIW_, J. S., HENRIKWANN JENSEN. *Visual Simulation of Smoke*, 2001. [2007]. Disponible en: http://www-graphics.stanford.edu/~fedkiw/papers/stanford2001-01.pdf

## Anexos

### Anexo 1 Clase emisor partículas

```
class emisor_partículas{

        public:

                GLfloat         posicion[3];
                GLfloat   ultima_posicion[3];
                GLfloat         velocidad[3];
                GLfloat       aceleracion[3];
                GLfloat posicion_relativa[3]; //para saber la pos relativa con respecto al objeto que emite la estela.

                emisor_partículas(GLfloat *pos, GLfloat *vel, GLfloat *ace, unsigned int inter, unsigned int vid, GLuint t_id, unsigned int v_emisor=0);

                emisor_partículas(Objeto *obj, GLfloat *pos, unsigned int inter, unsigned int vid, GLuint t_id, unsigned int v_emisor=0);
                //el constructor requiere la cantidad máxima de partículas, la posición, la velocidad, la aceleración, el intervalo de tiempo en que tarda en soltar una partícula  y el id de la textura a utilizar también en la segundad variante requiere el puntero del objeto al que va a perseguir y la posición relativa al mismo.
                  bool actualiza();
                ~emisor_partículas();
```

```
        private:

                unsigned int vida_partículas;
                unsigned int tiempo_pasado;  //variable necesaria para el calculo de la emisión de las
partículas.
                unsigned int intervalo;
                GLuint textura_id;
                bool sigue_a_objeto;
                Objeto *obj;
                bool emisor_inmortal;  //para saber si el emisor vive para siempre o no
                int vida_emisor; //en el caso de que sea mortal es el tiempo en que vive.


                void inserta_partícula();

};
```

**Anexo 2 Función dibuja partículas**

En esta función primero se lleva a cabo todo el proceso para dibujar las partículas del sistema.

```
 void Dibuja_partículas()
{
        TPartícula *p_aux;
        int tiempo = mirar_tiempo();
```

```
	unsigned int j = 0;
	GLfloat pos_camara[3];
	GLfloat vx1, vy1, vz1, vx2, vy2, vz2;
	GLfloat modulo1, modulo2;
	GLfloat angulo_y, angulo_z;

	if (num_partículas_activas == 0) return;
	Consulta_Posicion_camarafv(pos_camara);
	p_aux = primero;
	// j al final contendrá el numero de  elementos del vector
	while (p_aux != NULL) //lo voy pasando a un vector para ordenarlo y calcular la distancia a la cámara
	{
		vector_ordenacion[j].partícula = p_aux;
		vector_ordenacion[j].distancia = pow((pow((double(pos_camara[0] - p_aux->posicion[0])), 2) + pow((double(pos_camara[1] - p_aux->posicion[1])), 2) + pow(double(pos_camara[2] - p_aux->posicion[2]), 2)), 0.5);
		j++;
		p_aux = p_aux->sig;
	}
//llamada al *algoritmo* de ordenación quick sort
	qsort((void *) vector_ordenacion, j, sizeof(TVector_ordenacion), comparacion);

	Activar_transparencia();
```

```
	glMatrixMode(GL_MODELVIEW);
	if (luces) glDisable(GL_LIGHTING);

	 //empieza el siclo para pintarlas todas
	textura_activa = 0;

//	glBindTexture(GL_TEXTURE_2D, 0);
	for (unsigned int i=0; i<j; i++)
	{
		p_aux = vector_ordenacion[i].partícula;



		p_aux->edad += tiempo;

		if (p_aux->edad < p_aux->edad_max)  // pasa de pintarla si se le paso el tiempo
		{

			if (textura_activa != p_aux->textura_id)   //sólo cambia la textura activa si es
necesario.
			{
			glBindTexture(GL_TEXTURE_2D, p_aux->textura_id);
				textura_activa = p_aux->textura_id;
			}
```

```
//Hay que hallar el ángulo que hay que rotar la partícula para que mire hacia la cámara.
//primero hallo los vectores que me hacen falta pal calculo
vx1 = pos_camara[0] - p_aux->posicion[0] ;
vy1 = pos_camara[1] - p_aux->posicion[1] ;
vz1 = pos_camara[2] - p_aux->posicion[2] ;

vx2 = 0.0f;  //se inventa el vector pensando que la partícula este mirando al z positivo
vy2 = 0.0f;
vz2 = 1.0f;

//se normalizan dividiéndolos por el mayor de los tres

Normalizaf(&vx1, &vy1, &vz1);

//se hallan los módulos
modulo1 = (GLfloat) sqrt(pow(vx1, 2) + pow(vz1, 2));
modulo2 = vz2;
//al no tener componente "x" ni "y" es el mismo que la "z"

//saco el ángulo con la formula x*y=|x|*|y|*cos(ángulo)

GLfloat pio = (vz1*vz2)/(modulo1 * modulo2);
```

```
angulo_y = (GLfloat) acos(pio);
//el acos devuelve en radianes, hay que pasarlo a grados

angulo_y = (angulo_y * 180.0f) / PI;

//ahora igual pero para el ángulo z
modulo1 = (GLfloat) sqrt(pow(vy1, 2) + pow(vz1, 2));
pio = (vz1*vz2)/(modulo1 * modulo2);
angulo_z = (GLfloat) acos(pio);
angulo_z = (angulo_z * 180.0f) / PI;



glLoadIdentity();

glTranslatef(p_aux->posicion[0], p_aux->posicion[1], p_aux->posicion[2]);
glRotatef(angulo_y, 0.0f, 1.0f, 0.0f);
glRotatef(angulo_z -90.0f, 1.0f, 0.0f, 0.0f);

//se arregla esto, para saber el ángulo que hay que rotar la partícula.

glBegin(GL_TRIANGLES);

    //glNormal3f(0.0f, 0.0f, 1.0f);
```

```
                glColor4f(1.0f, 1.0f, 1.0f, p_aux->color[3]);

                glTexCoord2f(0.0f,1.0f);
                glVertex3f(-0.5f,0.5f,0.0f);

                glTexCoord2f(0.0f,0.0f);
                glVertex3f(-0.5f,-0.5f,0.0f);

                glTexCoord2f(1.0f,0.0f);
                glVertex3f(0.5f,-0.5f,0.0f);

                glTexCoord2f(1.0f,1.0f);
                glVertex3f(0.5f,0.5f,0.0f);

                glTexCoord2f(0.0f,1.0f);
                glVertex3f(-0.5f,0.5f,0.0f);

                glTexCoord2f(1.0f,0.0f);
                glVertex3f(0.5f,-0.5f,0.0f);

        glEnd();
```

```
                //regla de tres que actualiza el alpha de la textura según su edad.
p_aux->color[3] = 1.0f - (((GLfloat) p_aux->edad) / p_aux->edad_max);
        }
        else    //si ya se le ha pasado la edad y aun se sigue pintando la se borra de la memoria
        {
                TPartícula *ant, *sig;
                if (p_aux->ant != NULL)  //manejo básico de listas enlazadas
                {
                        ant = p_aux->ant;
                        sig = p_aux->sig;
                        ant->sig = sig;
                        if (sig != NULL) sig->ant = ant;
                        if (p_aux == ultimo) ultimo = p_aux->ant;
                        free (p_aux);
                        num_partículas_activas;
                }
                else
                {
                        if (p_aux->sig != NULL)
                        {
                                primero = p_aux->sig;  //manejo básico de listas enlazadas.
                                free (p_aux);
                                primero->ant = NULL;
```

```
                        num_partículas_activas--;
                    }
                    else
                    {
                        free (p_aux);
                        primero = NULL;
                        ultimo = NULL;
                    }
                }

            }

        }
        textura_activa = 0;
        if (luces) glEnable(GL_LIGHTING);
        Desactivar_transparencia();
}
```

## Anexo 3 Imágenes

Estela de humo.

Explosiones.

Edificios ardientes.

Océanos.

Humo de una hoguera.

Niebla.

## GLOSARIO DE TÉRMINOS

*(API):* Una *API* (*Application Programming Interface* **-** *Interfaz de Programación de Aplicaciones*) es el conjunto de *funciones* y *procedimientos* (o métodos si se refiere a programación orientada a objetos) que ofrece cierta librería para ser utilizado por otro software como una capa de abstracción.

*Algoritmo:* Secuencia de pasos computacionales que transforman una entrada en una salida (efecto caja negra) Herramienta computacional para resolver un determinado problema, en el cual, debe estar bien especificada la relación entre la entrada y la salida. El algoritmo materializa (efectúa) dicha relación. Un algoritmo es un resolvedor de un problema determinado.

Para solucionar computacionalmente un problema se requiere obtener datos de alguna materia.

1. Seleccionar un modelo matemático – computacional adecuado para el problema (representación del modelo)

2. Concebir con respecto a dicho modelo un algoritmo que dé solución al algoritmo (diseño del algoritmo)

3. Programar el algoritmo en algún lenguaje de programación y ejecutar el programa en una computadora (programación del algoritmo)

Estructura Básica:

1. inicio
2. variables(operaciones básicas)
3. ingresar datos
4. proceso de operaciones
5. mostrar resultados

fin

*(CFD):* No es más que dinámica fluida computacional (Computational Fluid Dynamics).

*Estela:* Rastro que deja tras de si un objeto en movimiento.

*Estocásticos:* es aquel sistema que funciona, sobre todo, por el azar. Las leyes conocidas de causa-efecto no explican cómo actúa el sistema (y de modo reducido el fenómeno) de manera determinista, sino en función de probabilidades.

*Framebuffer*: es un dispositivo virtual del sistema operativo que se presenta ante las aplicaciones de diferentes maneras en función del sistema del cual se hable, aunque generalmente aparece como un archivo o un bloque de memoria RAM reservado en la computadora, y que puede ser accedido en lectura/escritura por uno o más procesos; en este archivo o zona de memoria especial cualquier escritura modifica directamente las imágenes desplegadas en el dispositivo de vídeo, para que de esa manera los programas puedan mostrar información en pantalla sin preocuparse de los detalles de implantación, ni de la interacción real entre el ordenador y el dispositivo de vídeo.

*Frame*: es una imagen independiente, una sucesión de frames compone una animación. Esto viene dado por las pequeñas diferencias que hay entre cada uno de ellos que producen a la vista la sensación de movimiento.

*Infográfica*: Información gráfica.

*Simulador:* Es un aparato que permite la simulación de un sistema, reproduciendo su comportamiento. Los simuladores reproducen sensaciones que en realidad no están sucediendo.

*Sistema de partículas:* Una partícula es un objeto que tiene masa, posición, velocidad y puede estar afectada por fuerzas exteriores, como por ejemplo la aceleración de la gravedad, la resistencia del aire o el viento. Para definir nuestra partícula, se necesita lo siguiente; su posición, velocidad, aceleración, color, "tiempo" de vida y su masa, que solo se tendrá cuando las partículas interaccionan entre ellas**.**

*TCP/IP:* es un protocolo que proporciona transmisión fiable de paquetes de datos sobre redes.

*Vorticity:* El vorticity es un concepto matemático usado en la dinámica fluida. Puede ser relacionado con la cantidad de "circulación" o de "rotación" (o más terminantemente, el índice angular local de la rotación) en un líquido. Está a menudo, pero no siempre, definido como el enrollamiento de la velocidad:

$$\vec{\omega} = \vec{\nabla} \wedge \vec{v}$$

*z-buffer:* es la parte de la memoria de un adaptador de video encargada de gestionar las coordenadas de profundidad de las imágenes en los gráficos en tres dimensiones (3-D), normalmente calculados por hardware y algunas veces por software.