03063

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

10
2ej

Unidad Acádemica de los Ciclos Profesional y de Posgrado del
Colegio de Ciencias y Humanidades

MODELADO Y ANIMACION DE GASES
USANDO ESPACIOS SOLIDOS

T E S I S

Que para obtener el titulo de

MAESTRO EN CIENCIAS DE LA COMPUTACION

P r e s e n t a:
ACT. ANGELICA GUADALUPE SU RAMOS

Asesor de tesis: MAT. ANA LUISA SOLIS C.

MEXICO, D.F. 1995

Dedicado a:

mis padres Rafael Sú y Blanca Ramos con admiración y cariño por el amor, apoyo y comprensión que me han brindado durante toda la vida, por que gracias a la confianza que me han inyectado he logrado realizar mis metas.

mi abuelita Lupita por su cariño y comprensión.

mis hermanos Erika, Rafael, Javier y Arturo por todos los momentos hermosos que hemos compartido.

Moy, compañero inseparable, por su amor, apoyo y comprensión que me ha demostrado en todo momento y por la confianza que ha depositado en mí.

Adriana por la alegría que brinda a cada uno de mis días.

Ana Luisa por el tiempo, conocimientos y consejos que me ha proporcionado, los cuales han permitido realizar el presente trabajo.

todos mis amigos.

INDICE

INTRODUCCION

En graficación por computadora es importante generar imágenes realistas, un aspecto que hay que considerar para lograr esto, es la simulación visual de objetos y fenómenos naturales que vemos cotidianamente, tales como: los árboles, el agua, el fuego, la tierra, los gases, los animales y nosotros mismos.

Sin embargo las técnicas estándar que existen para la generación de imágenes no son suficientes para generar imágenes de escenas que contengan dichos fenómenos, ya que éstos no tienen superficies ni límites bien definidos. Por lo tanto es necesario desarrollar modelos que permitan generarlos.

Los modelos que se han desarrollado al respecto pueden ser clasificados como: empíricos, físicos, estructurales, morfológicos, impresionistas y autodefinibles. Estos modelos consideran los siguientes aspectos: la forma, y la interacción que existe entre la luz y el objeto. A su vez la interacción se divide en dos categorías, *efectos atmosféricos*, en donde el objeto que interactúa con la luz no tiene forma propia, y *modelos de iluminación* para superficies.

Además de estos modelos, se tienen algunos principios y conceptos que hacen más eficiente el modelado de los objetos naturales.

Una de las herramientas más utilizados en muchos de los algoritmos que se han propuesto para modelar éstos, son las texturas, de aquí la importancia de explicarlas con mayor profundidad.

Dentro de los fenómenos naturales encontramos a los fenómenos gaseosos, tales como las nubes, el humo, el vapor y la niebla. Estos son elementos que al ser agregados a las escenas incrementan el grado de realismo de las mismas, por ejemplo agregar niebla en un escena al aire libre, o vapor a una taza de café, o nubes a una escena. Además pueden ser usadas en diversas áreas de aplicación como son: meteorología, entretenimiento, arte, simulación de vuelos, física, ingeniería. De lo anterior se ve la importancia de simularlos.

Para simular los fenómenos gaseosos en graficación por computadora se consideran los siguientes aspectos: el modelado, el *"render"* (proceso por medio del cual se genera la imagen que será presentada en pantalla, en él se consideran: el modelo geométrico, el modelo de iluminación, la cámara de visión, el manejo de sombras y otros parámetros) y la animación de los gases. A través del modelado se determina la forma del gas, con el *"render"* se obtiene la apariencia visual del gas que será presentada en la pantalla, y con la animación damos movimiento a los gases.

Al respecto se han realizado varios trabajos, algunos de ellos se han enfocado sólo en la simulación de nubes, tal como: Gardner [4] que utilizó texturas sólidas en elipsoides huecas, Max [10] que usó campos de altura, Sakas[19] que utilizó síntesis espectral. Otros contemplan la simulación de todo tipo de fenómeno gaseoso, tal es el caso de: Ebert[2], que utilizó "render" de volúmenes y "scan line a-buffer" (una técnica para realizar el "render" línea por línea de una imagen) y Ebert[1] que utilizó espacios sólidos.

El objetivo que se persigue en el presente trabajo es:
La presentación e implantación de un algoritmo que permita modelar y animar fenómenos gaseosos.

1. MODELADO DE FENOMENOS NATURALES

Introducción

El mundo es muy complejo. Está formado por millones de partículas, aproximadamente de 10^{60} a 10^{70}, que a su vez forman parten de otros objetos. Aún si quisiéramos modelar en solamente una porción de éste, digamos lo que vemos por la ventana, tendríamos que modelar 10^{30} partículas. Posiblemente haya una manera más fácil de modelar cada partícula. Aunque después de todo no es necesario modelar todas las propiedades de los objetos. De hecho, en muchos casos lo que se pretende es obtener la *apariencia* de los objetos, en tal caso un modelo con superficies es suficiente. Esto es precisamente por lo que en muchos modelos de graficación por computadora se utilizan superficies. Sin embargo con esto no se resuelve el problema, ya que hay objetos con superficies muy complejas. Esto lo podemos ver al mirar a nuestro alrededor y ver nuestra piel, nuestras telas, los océanos. Los modelos que usan polígonos y superficies paramétricas no funcionan bien para este tipo de objetos. Claramente realizar el diseño, el almacenamiento de la información, las transformaciones, y el render de los objetos es una tarea muy difícil. Por tal razón es necesario desarrollar modelos dentro de grafciación por coputadora que faciliten esta tarea.

1.1. Características de los objetos naturales

A continuación se presentan las características de los objetos naturales:

- La mayoría de los objetos tienen formas complejas. Algunos objetos no pueden ser descritos con superficies. Por ejemplo los flujos turbulentos, el agua, el fuego y el humo.
- Se pueden necesitar en grandes cantidades. Por ejemplo para generar la imagen de un bosque, se necesitan miles de árboles, cada uno con miles de hojas.

- Se mueven de forma muy compleja. La mayoría de los objetos artificiales se mueven rígidamente, así que con las transformaciones básicas de la graficación por computadora es posible realizarlo. Pero con estas transformaciones sería muy difícil lograr el movimiento de una nube por ejemplo. Por tal razón el movimiento tiene que ser considerado al momento de elaborar el diseño.
- Y finalmente, estamos familiarizados con la apariencia de los objetos naturales, y nuestro sistema visual está acostumbrado a sus características. Así que cualquier discrepancia entre la apariencia del objeto generada por los modelos y la apariencia *real del objeto* será detectada fácilmente.

1.2. Tipos de modelos

Las técnicas tradicionales de modelado usan primitivas simples para modelar puntos, aristas, o más comúnmente, superficies. En particular se usan superficies paramétricas para representar a los objetos. La *forma* es primordial en estos casos. Para los objetos con volumen, se han creado técnicas de modelado de objetos sólidos, y en particular CSG (*constructive solid geometry*). Para el modelado de objetos naturales, lo más importante es la *apariencia*. En graficación por computadora se han realizado infinidad de modelos, desde modelos que se derivan de la física hasta modelos que no toman en cuenta ninguna fuerza natural. La siguiente tarea es realizar una clasificación de los modelos. Esta es una tarea difícil, pues hay veces que las cosas que se están clasificando pueden ponerse en diferentes categorías, o bien dar una categorización que no sea posible utilizar. Lo que se debe considerar al realizar una clasificación es si cada cosa tiene su propia categoría, si cada categoría señala las similaridades y diferencias, y principalmente, si ayudan a seleccionar o eliminar posibilidades cuando se diseña o se selecciona un modelo.

Es importante notar que los tipos de modelos y las primitivas usadas para construir los modelos son muy diferentes.

La clasificación de modelos usada es la siguiente: empírica, físico, morfológico, estructural, impresionistas y autodefinibles.

Modelos empíricos

Una forma de modelar objetos naturales se logra utilizando la *digitización*. Por ejemplo un terreno puede ser modelado fácilmente con los datos que se obtuvieron al medir el terreno, o bien con los datos obtenidos por satélite.

Otra aproximación puede realizarse si se utiliza el contorno del terreno. En este caso si las superficies se requieren para realizar el "render", los algoritmos para generar las superficies a partir de los contornos son necesarios.

La misma aproximación ha sido usada para objetos tan complejos como los árboles.

Los modelos empíricos tienen varias ventajas . La principal es que los datos se obtienen de la *digitización*, o pueden ser obtenidos automáticamente o semi-automáticamente a partir de ésta. Como los datos están dados, modelar las primitivas es relativamente fácil, por lo tanto pueden ser integradas fácilmente a un sistema convencional de "render". Otra ventaja es que se pueden modelar muchos objetos.

Los inconvenientes de estas técnicas también son muchos. Una tarea tediosa es obtener los datos. Esto no está tan mal si el modelo tiene un larga "vida", en vez de usarse una sola vez. Además estos modelos, pueden producir grandes bases de datos, que no son muy flexibles. Aquí la flexibilidad tiene dos aspectos. Uno es adaptarlos al modelo. El otro aspecto es el problema de los niveles de detalle. Como los datos se obtienen al digitizar, el problema es que hacer para obtener una representación menos detallada, para combinar las primitivas, o para seleccionar buenas representaciones. Y finalmente, el objeto tiene que existir y estar disponible para poder modelarlo.

Modelos físicos

Un modelo físico es aquel en el que se usan representaciones matemáticas para generar un objeto. Ejemplos de estos modelos físicos los encontramos en áreas como: física, química, mecánica de fluidos, etc.

Una ventaja de estos modelos es que se requiere poco esfuerzo en el modelado, y además se toman algunas características del comportamiento real del objeto. También son fáciles de parametrizar al cambiar algunas "constantes" involucradas. Otra ventaja muy fuerte es que casi siempre el tiempo es una variable dentro de la ecuación y por lo tanto el modelo se puede usar directamente para la animación. El área donde se ilustran los modelos físicos se conoce como visualización científica.

Las desventajas de los modelos físicos son varias. Para un fenómeno complejo las ecuaciones no tienen una solución única, y regularmente se tiene que recurrir a un sistema de ecuaciones diferenciales que tienen que ser resueltas numéricamente con una gran cantidad de puntos en el espacio 2-D o 3-D. El rango de los fenómenos que se pueden modelar está restringido. Para obtener ecuaciones manejables, frecuentemente se tienen que hacer simplificaciones lo cual hace que el rango de condiciones donde se pueden aplicar los resultados es limitado. Y finalmente los modelos físicos dan respuesta a preguntas que no se hacen en graficación por computadora, tales como la presión, la temperatura, el balance de la energía. El costo de obtener estas respuestas puede ser mucho mayor que el costo de lo que realmente se quiere: la apariencia del fenómeno. Por ejemplo, Nelson Max propuso un modelo para ondas en graficación por computadora [14] que rechaza el modelo "cicloide" y adopta el modelo de Stokes, porque se generan ondas que generan flujo irrotacional, y esto corresponde más a la realidad. En este caso seria mejor tener un modelo que no sea irrotacional pero que genere ondas con apariencia real, ya que ésta propiedad no tiene un impacto visual.

Jim Kajiya y Von Herzen [8] usaron un modelo para nubes que se basa directamente en ecuaciones diferenciales para el vapor contenido en el aire como una función de la altitud, la velocidad del viento, y la temperatura. Las ventajas de tal solución son que las ecuaciones son conocidas, y se pueden resolver en función del tiempo para obtener una animación con alto grado de realismo. Otra ventaja es que los resultados están en términos de la densidad del vapor, al cual se le puede realizar el "render" , si existe un modelo de iluminación disponible. Las desventajas son que los cálculos incluyen la resolución de ecuaciones diferenciales, las que tienen que hacerse numéricamente en una malla de cuatro dimensiones (espacio + tiempo).Otra desventaja es que no se generan las primitivas tradicionales, por lo tanto se tienen que encontrar o desarrollar técnicas para hacer el "render".

Al aparecer computadoras más poderosas, el costo computacional dejará de ser una desventaja para el desarrollo de futuros modelos aunque es obvio que aparecen otros tipos de desventajas. Otro factor es el desarrollo de procesadores paralelos, el cual cambiará el concepto en el que están basados estos modelos.

Modelos morfológicos

Los modelos morfológicos son los que dan directamente la forma del modelo. Un ejemplo trivial es una esfera para una burbuja de jabón. La mayoría de los modelos son modelos morfológicos. Cuando se usan superficies bicubicas "B-spline" para modelar el toldo de un carro, al modelo se le pide nada más la forma.

Thompson en su libro "*Of Growth and Form*" [26], propuso, (y frecuentemente sucede), que muchas de las formas de la naturaleza y la explicación del crecimiento de los mismos (cuernos, conchas, cuerpos, etc.) se logran con simples restricciones de geometría. La concha de mar es un ejemplo, el animal va buscando compartimientos más grandes y los va agregando a la concha, así es como crece, y le da la forma general de una espiral logarítmica.

Esto da un camino directo para modelar estas formas, imitando las transformaciones del trabajo. Thompson distingue tres parámetros para determinar la forma de algunas clases de conchas. La primera, α, es el ángulo entre la tangente a la espiral y el radio en algún punto (el cual es la constante en una espiral logarítmica), el segundo, β, es el ángulo formado por la concha en la dirección de los ejes de rotación, y γ, que mide la diferencia en la velocidad de crecimiento entre el exterior y el borde interior de la espiral.

Estos tipos de modelos son bastante fáciles de implementar y pueden ser hechos usando primitivas tradicionales. Hay sin embargo un rango limitado de formas y fenómenos, que cumplan con las características del modelo. También son difíciles de animar, y por lo tanto se utilizan sólo para objetos fijos.

Modelos estructurales

En los modelos estructurales, tal como el nombre lo indica, solamente es dada la estructura del objeto. En su forma más simple los modelos estructurales dan solamente la topología del objeto modelado. Mientras que en otras aplicaciones ésta es la principal característica del objeto, en algunos es lo más simple.

La ventaja de estos modelos es especialmente obvio cuando los objetos modelados tienen una estructura fuerte, tal como los árboles, y las plantas en general.

La desventaja es que la interpretación geométrica es necesaria para todo.

Basado en el trabajo de un botánico/agrónomo, Philippe de Reffye [21], un grupo de investigadores [22] diseñó e implementó modelos de plantas. El núcleo de crecimiento de las plantas son los *meristemos*, tejidos especializados de un botón. El trabajo de Reffye caracteriza y simula las funciones de los meristemos. Su actividad puede ser resumida en tres pasos: *crecimiento*, *ramificación*, y la *muerte* (no hay más actividad). Al conocer la velocidad de crecimiento, los tiempos de ramificación y los eventos de muerte permite tener un modelo muy versátil del crecimiento de las plantas, y por lo tanto tener modelos estáticos de las plantas en varias etapas de su desarrollo.

La misma aplicación se ha realizado con hojas y flores. En este caso las primitivas que se deben usar son superficies en vez de segmentos de ramas.

Modelos impresionistas

Hay modelos que no tienen conexión física alguna con el fenómeno modelado, o ninguna interpretación geométrica de la forma correspondiente a la superficie del objeto modelado, sino solamente hacen una aproximación del mismo. A estos modelos se les conoce como impresionistas porque tienen similaridad con los métodos y propósitos de un pintor impresionista, y por la analogía que hace Gardner[1].

Los modelos impresionistas son buenos cuando lo que interesa es la apariencia. Por ejemplo han sido usados en simulaciones de vuelos.

Sus desventajas son que tienen que ser diseñados por prueba y error, tienen capacidades limitadas en rango dinámico, y son difíciles o imposibles de animar

[1] Actualmente muchos pintores impresionistas se basan en principios científicos, especialmente aplican la luz y la visión; aquí no se debatirá el subjetivismo u objetivismo, o la naturaleza de la verdad. Esto es sólo graficación por computadora.

(pensemos como animar las ramas o la evolución de las nubes con estos modelos). Su flexibilidad está también limitada, un ejemplo claro es el caso de los árboles. Los parámetros de las primitivas y las texturas no están relacionadas con el modelo, y por lo tanto no ayuden a generar variantes usuales.

Modelos autodefinibles

Los modelos autodefinibles son aquellos donde el modelo se define por si mismo. Este es el caso de los objetos matemáticos a los cuales se les puede diseñar una geometría. Al respecto, la graficación por computadora ha logrado desplegar muchos objetos matemáticos que antes eran inimaginables. Los más claros ejemplos son el conjunto de Mandelbrot y el conjunto de Julia.

Modelos mixtos

Claramente no todos los modelos pertenecen estrictamente a una de las categorías antes mencionadas, esto se debe a que la mayoría de las veces se realiza el modelo con "sub-modelos" y cada uno de estos pertenecen a diferentes tipos. Por ejemplo un modelo estructural de árboles puede usar un modelo empírico para las hojas, o viceversa, ó una textura digitizada puede ser usada dentro de un modelo morfológico para dar la apariencia de realzado al objeto ("bump mapping").

1.3 Técnicas y principios básicos

En esta sección se verán los conceptos básicos y las herramientas que se han desarrollado para hacer más eficiente el modelado de objetos naturales. Primero se describirán los principios generales: *ampliación de bases de datos*, y *variables de nivel de detalle*. Después se verá cual es el rol del *tiempo* en un modelo, y el uso de elementos *estócasticos*, lo que es el *modelado estócastico*. Finalmente se describirán herramientas especificas: *texturas*, *sistemas de partículas* y *fractales*.

1.3.1. Ampliación de bases de datos

La ampliación de bases de datos se refiere a un grupo de técnicas donde el modelo original es compacto, pero permite generar grandes cantidades de primitivas geométricas o de despliegue. Por supuesto esta idea es utilizada en las representaciones de curvas y superficies, donde pocos coeficientes son suficientes para generar una gran cantidad de puntos. El problema aquí es que estas representaciones estándar generan objetos simples, superficies suaves y continuas. Ahora considérese la figura 1.1. El árbol de la derecha se obtiene a partir de los dos tipos de segmentos de la izquierda, y de dos aplicaciones sucesivas de la *regla de sustitución*. En cada paso, cada segmento se reemplaza por el patrón de la rama dado, escalado, rotado y traslado de tal forma que coincide con las extremidades del árbol. Aquí se tiene un patrón inicial, una regla simple, muy fácil de describir en una base de datos y unas pocas aplicaciones de la regla para obtener objetos más complejos. La misma técnica puede ser utilizada para generar texturas en dos o tres dimensiones.

Todos estos ejemplos pueden ser dados como *gramáticas formales*, donde una gramática formal es un conjunto de símbolos y reglas que pueden ser aplicadas para transformar los símbolos. El nuevo elemento aquí es que se quiere una *representación pictórica* para los símbolos. Hay muchas formas de lograr esto. Una es tener a los símbolos como elementos de la imagen. Este es el caso en el ejemplo de texturas. Los símbolos manipulados son patrones de pixels, y estos patrones forma la imagen.

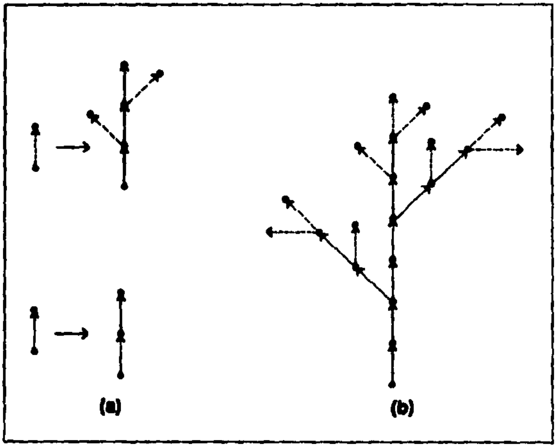


Fig. 1.1
árbol generado
(a) segmentos iniciales y reglas
(b) árbol generado después de aplicar 2 veces las reglas

Otra forma para generar una imagen a partir de una gramática es agregar una representación geométrica. Los símbolos pueden ser interpretados como instrucciones geométricas tales como escalar, rotar, trasladar, etc. Estas técnicas han sido usadas para generar un amplio rango de texturas e imágenes de plantas y árboles.

1.3.2. Niveles de detalle

En una imagen del mundo se tienen varias escalas, por ejemplo hay árboles que están a millas de distancia y hay otros que están a muy pocos metros. Es obvio que nos es práctico hacer el "render" de todos ellos al mismo nivel de detalle. Es por tanto importante que los modelos generen primitivas que controlen los niveles de detalle.

Los modelos pueden controlar esto de cuatro formas: controlando el número de aplicaciones que se producen en un modelo gramático, parametrizando dentro de un modelo procedural, separando las representaciones discretas en un modelo, y extrayendo automáticamente las representaciones de este modelo.

Generación controlada de un modelo gramático

Una condición importante, que no siempre se cumple, es que en cada paso donde se aplican las reglas, el objeto que se obtiene es una versión "simplificada" del objeto que se está desarrollando. Está técnica se usa frecuentemente asociada a un proceso estócastico, donde los detalles generados incluyen elementos aleatorios.

Parametrización

Un ejemplo simple del segundo método es la representación de un círculo con un polígono. En este caso es fácil para el sistema determinar el número de lados necesarios para que el polígono se despliegue en pantalla con la forma de un círculo. El procedimiento para crear el círculo tiene como parámetros el número de lados y el "nivel de detalle". Esto es usado comúnmente para calcular curvas paramétricas y superficies.

Modelos separados

La tercera técnica involucra la generación de bases de datos separadas una para cada nivel de detalle. Esto generalmente se hace "manualmente". Un problema entonces es asegurar que sea suave la transición entre dos niveles. Usualmente los dos modelos son combinados, lo que hace que la imagen final sea una suma de los dos. Esta ha sido la técnica favorita en la simulación de vuelos.

Extracción automática de los detalles

La cuarta técnica, la extracción automática de diferentes niveles de detalle, es la más difícil, y aún no se ha realizado satisfactoriamente en muchas representaciones. Esta es común al usarse conjuntamente con modelos empíricos, ya que les permite definir niveles de detalle variables.

En todas las técnicas los niveles correctos pueden ser determinados globalmente, esto se hace a partir de toda la imagen o determinados *adaptativamente* de una base local. El segundo método es frecuentemente necesario, ya sea dentro de la escena, o aún en el mismo objeto, los niveles que se requieren pueden ser muy diferentes, especialmente por la perspectiva.

Para determinar cual nivel de detalle es suficiente, tenemos que tomar en cuenta las características de nuestro sistema visual.

Un problema que se tiene al implementar este concepto es asegurar una transición suave entre los diferentes niveles usados. Esto se vuelve crítico cuando diferentes técnicas (mapeo de texturas, "bump mapping", "displacement mapping") y varios modelos de iluminación se juntan.

1.3.3. El rol del tiempo en los modelos

Es verdad que si un modelo quiere ser animado, el tiempo debe ser un parámetro del modelo. Pero cabe mencionar que muchos de los modelos propuestos no pueden ser animados razonablemente ya que los diseñadores olvidaron este principio.

Un ejemplo de modelos donde el tiempo está ausente desde el principio, y por lo tanto es difícil de animar, son las texturas utilizadas en el modelo de nubes de Gardner y las funciones de turbulencia ("turbulence") y ruido ("noise") usadas en el trabajo de Perlin.

Una posibilidad con modelos multidimensionales, especialmente con procesos estocásticos, es generarlos en una dimensión más grande que la que se necesita para desplegar, y usar la dimensión extra como el tiempo. Esto es válido solamente si el comportamiento temporal del proceso es similar a su comportamiento espacial. Una animación convincente de flamas usando movimiento Browniano fraccional tridimensional o cuatro-dimensional puede ser llevado a cabo con esta técnica.

El tiempo (o más exactamente la animación) se comporta muy difícil con los modelos. Los modelos funcionan muy bien al generar imágenes fijas pero revelan algunas fallas cuando son animados. Esto se debe a la falta de consistencia entre los niveles, los cambios repentinos en los parámetros, los movimientos irreales, etc. Esto se debe también al hecho de que si hay algún problema con el modelo, éste se manifestará tarde o temprano al ser animado.

1.3.4. Modelado estócastico

Se puede conservar el poder de las técnicas antes vistas e introducirle las variaciones necesarias usando elementos *estocásticos*. "Estocástico" es un sinónimo de aleatorio. El término aleatorio para nuestros propósitos se definirá como una propiedad que es impredecible para una sola ocurrencia, pero se puede obtener un comportamiento promedio después de muchas observaciones. Un ejemplo relevante en los fenómenos naturales es el siguiente: el peso exacto del primer pino que está en frente de mí al estar en un bosque no puede predecirse, pero si los pesos de muchos pinos son medidos y se promedian, alguien con conocimientos en esta especie de árboles puede dar los límites dentro de los cuales este promedio puede ser encontrado. Al revés, los pesos reales para los árboles pueden ser generados creando pesos "aleatorios" conociendo el promedio obtenido y la distribución de los pesos medidos.

La mayoría de los modelos para fenómenos naturales usados incorporan elementos aleatorios. Aunque la mayoría de ellos se resuelven determinísticamente, tales como los modelos estructurales basados en gramáticas. Los elementos estocásticos pueden aparecer en muchos pasos: en la generación de ruido blanco común en la mayoría de las aproximaciones que usan Síntesis de Fourier, agregando pequeñas fluctuaciones aleatorias a los parámetros de una interpretación geométrica, usando una gramática estocástica, usando distribuciones de probabilidad para eventos estructurales o más directamente usando funciones estocásticas.

Un proceso estocástico es un proceso que genera variables aleatorias. Por lo tanto el modelo para algún fenómeno que incluye elementos aleatorios tendrá que incluir la simulación de un proceso estocástico. Fournier y Fusell[3] llamaron *modelado estocástico* al grupo de técnicas que combinan modelos ordinarios y procesos estócasticos. El más popular proceso estocástico en graficación por computadora son los *fractales*.

1.3.5. Texturas

Una textura es simplemente una función en el espacio, $F(x,y)$ en dos dimensiones, o $F(x,y,z)$ en tres. La función puede tener un valor escalar (un simple valor) o un vector de valores, como un vector de color tridimensional. En la mayoría de las aplicaciones las coordenadas x, y ó x, y, z son coordenadas discretas (es también llamado espacio de la textura), y pueden verse en la textura como un arreglo discreto de valores, indexado por las coordenadas.

El uso de texturas en forma general, y en el modelado de fenómenos naturales en particular, tiene dos aspectos: el modelado de texturas, y su aplicación (mapeo de texturas).

Cuando las texturas son usadas para modelar fenómenos naturales, el rango de técnicas usadas para otras clases de objetos también se aplican a ellas. En este caso en particular, las imágenes digitizadas de objetos reales, texturas reales, son fáciles de usar como base de la textura, ya que ambas son imágenes en dos dimensiones. Esto es sólo una versión más fácil de un modelo empírico.

Las texturas generadas proceduralmente son también muy usadas. Sus principales ventajas son que pueden ser generadas con niveles de detalle controlables, pueden estar en una *banda-limitada* para evitar los problemas de "aliasing", y requieren pocos parámetros.

Una importante y poderosa variación de la aproximación empírica consiste en usar un modelo de textura parametrizada y aplicar un procedimiento apropiado para simular una textura real dada, con un criterio de error mínimo. Esto fue desarrollado por Ma y Gagalowicz.[6]. Ellos primero asumen que un pequeño conjunto de niveles de grises es suficiente. Después asumen que para la mayoría de las texturas el dominio donde las estadísticas tienen que ser aproximadas es aún más pequeño (en término del campo de visión). Ellos usan estadísticas tales como la autocovarianza, el histograma y los momentos de la distribución de los valores en la escala de grises. Una vez que las estadísticas se calcularon, se genera el ruido blanco a través del histograma, y después iterativamente se modifica cada uno de los pixeles para minimizar el error en cada una de las estadísticas usadas. La desventaja es que el número de parámetros es bastante grande.

1.3.6. Sistemas de partículas

Los sistemas de partículas los introdujo W. T. Reeves[19], primero para modelar fuego, después para generar algunas de las imágenes más reales del bosque [20]. Los sistemas de partículas son por lo tanto una nueva manera de modelar primitivas.

En los sistemas de partículas las primitivas básicas son puntos (partículas). La creación, destrucción y trayectoria de estos puntos se controla de acuerdo a las características de los objetos modelados. Un objeto está representado por esas partículas, ya sea por su posición en un tiempo dado, como en el caso del fuego, fuegos artificiales, o como parte de sus trayectorias, como en el caso de la hierba y los árboles. Para la mayoría de las aplicaciones se agregarán elementos estocásticos que producirán las variaciones necesarias. En aplicaciones no estructuradas, habrá pocos parámetros determinísticos. En el caso de los árboles, podría haber arriba de 30 parámetros, los cuales controlan los ángulos entre las ramas, las longitudes de las ramas, etc.. Una ventaja importante de un sistema de partículas, es que comparten datos con otros métodos, además como sus primitivas geométricas son puntos, son fáciles de transformar, y sus trayectorias son fáciles de filtrar, tanto en el espacio como en el tiempo. Por lo tanto el "antialiasing" puede aplicárseles sin mucho costo computacional. Las principales desventajas son el gran número de primitivas que se necesitan para un tiempo dado, el hecho de diseñarse a prueba y error, y los cálculos especiales que se tienen que realizar para el sombreado.

Los sistemas de partículas han sido usados en el área de graficación llamada modelado físico.

1.3.7 Modelos fractales

Algunos objetos naturales parecen tener detalles con detalles, y así (aparentemente) infinitamente. Otra forma de ver este fenómeno es considerando muchas de las funciones matemáticas clásicas. La definición de las derivadas de una función es el límite de la tangente a la curva o superficie que representa la función. Si se aplica este concepto a un litoral, por ejemplo, el "límite" no tenderá a nada en particular. De estas observaciones, y de la existencia de objetos matemáticos que comparten estas propiedades paradójicas, Benoit Mandelbrot [10] introdujo el concepto de *fractal*, y así produjo una nueva forma de ver los objetos matemáticos y algunos fenómenos naturales que pueden ser descritos con estos objetos.

El universo de los fractales es muy grande pero para nuestros propósitos pueden ser divididos en fractales determinísticos y los fractales estocásticos, donde las propiedades de los fractales se aplican a las características de las variables aleatorias. Se discutirá solo uno de los fractales estócasticos, *fractional Brownian motion (fBm)*. Este ha sido introducido por Mandelbrot y Van Ness [12] como una generalización del movimiento Browniano, el cual por sí mismo genera propiedades fractales. Un objeto natural que se ha modelado con el fBm es un terreno. Existen dos formas. Una de ellas es calcular el "render" de muestras de fBm, y desplegarlas para simular el color y el sombreado del terreno, y revisar visualmente si son satisfactorias. Podemos confirmar esta revisión midiendo las propiedades estadísticas del terreno real, y ver si corresponden a las características del fBm. La otra forma es construir un modelo para la creación del terreno, que describa la superficie en términos del fBm. Con la primera forma, se tiene que usar el criterio. En la segunda, los resultados son mejores. El terreno ciertamente tiene características fractales, pero no están en un rango medible. Después de todo esto no es sorprendente, ya que existen fuerzas y fenómenos que afectan la forma del terreno, y es demasiado pedirle a una simple función matemática que modele la suma de estos efectos.

Los procesos fractales no han sido usados solamente en el modelado de terrenos, sinó también el modelado de nubes, agua, texturas para árboles, fuego, etc.

La ventaja principal de fBm en el modelado de un terreno es el tamaño reducido de la representación. Dependiendo de cómo se incluyan los datos determinísticos, la base de datos puede tener desde dos números hasta unos cuantos cientos, y ésta representa terrenos que contienen miles o millones de polígonos. La segunda ventaja, debida a la naturaleza del fractal, es la ilimitada cantidad de detalles que se pueden generar. Las desventajas son el hecho de que la generación de una superficie con subdivisiones recursivas no es suficiente, y esto complicará los algoritmo de subdivisión, además tiene flexibilidad limitada.

2. MODELADO DE TEXTURAS POR PROCEDIMIENTOS

Introducción

El modelado por procedimientos se refiere al hecho de usar un programa para generar un modelo geométrico, y son dos los motivos por los que se han usado. Primero, se pueden modelar objetos o fenómenos a través de un procedimiento con parámetros, pudiéndose incorporar al tiempo como uno de estos parámetros, como consecuencia obtener una animación del modelo. El segundo motivo es el bajo-costo de obtener imágenes con gran complejidad visual. Esto es, podemos obtener diferentes instancias del fenómeno modelado con tan solo variar los parámetros del procedimiento.

Muy relacionado con el modelado por procedimientos está el hecho de usarlos para definir texturas. A esta forma de definir texturas se le conoce como texturas sólidas o texturas tridimensionales.

2.1. Texturas tridimensionales

Tradicionalmente el mapeo de texturas se ha realizado por medio de funciones definidas en el espacio bidimensional, el problema es que realizar éste en objetos de forma arbitraria se vuelve muy difícil. Esto se debe a dos razones:

El mapeo de texturas bidimensional basado en sistemas de coordenadas de superficie puede producir grandes variaciones en la compresión de la textura lo cual se refleja en la variación de la curvatura de la superficie.

El mapeo de texturas sobre la superficie de un objeto que tiene una topología no trivial puede volverse muy difícil. Además, mantener la continuidad de la textura a través de los elementos de la superficie, los que pueden ser de diferentes tipos y estar conectados de alguna manera, también es difícil de mantener.

El mapeo tridimensional de texturas, texturas sólidas, vence estos problemas ya que la única información que se requiere para asignar a un punto del objeto el valor de la textura, es su posición en el espacio. Entonces para asignar una textura a un objeto se tienen que evaluar los puntos de la superficie del objeto en la función tridimensional de textura.

El método es definir por medio de procedimientos una función para la textura en el espacio del objeto. Dado un punto (x, y, z) en la superficie de un objeto, el color es definido como $T(x, y, z)$, donde T es la función de textura.

Esto es como si se esculpiera o tallara un objeto a partir de un bloque de un material.

La desventaja es que aunque se eliminan los problemas de mapeo, los patrones de texturas están limitados a definiciones que se puedan construir analíticamente. Esto contrasta con las texturas bidimensionales; aquí se puede utilizar cualquier textura, desde una fotografía hasta imágenes de vídeo.

"Noise" tridimensional

Una de las funciones más usadas dentro de las texturas por procedimientos es la función "noise". Con esta función se puede simular una cantidad sorprendente de efectos reales, por lo que se considera una función primitiva.

Perlin [17] fue el primero que trabajó con el concepto de "noise", el definió a la función noise, como una función tridimensional, que recibe como entrada la posición, y regresa un valor escalar. Idealmente la función debe tener las siguientes propiedades:

1. invarianza estadística bajo rotación
2. invarianza estadística bajo traslación
3. una limitada banda de frecuencias

Las primeras dos condiciones aseguran que la función "noise" es manejable, esto es, no pasa nada si movemos u orientamos la función noise en el espacio, se garantiza que su apariencia general será la misma. La tercera condición nos permite obtener una muestra de la función "noise" sin problemas de "aliasing".

El método de Perlin para generar noise es definir una malla de enteros, o un conjunto de puntos en el espacio, situados en las localidades (*i, j, k*) donde *i, j* y *k* son enteros. Cada punto en la malla tiene un número aleatorio asociado. Esto puede ser hecho usando una tabla de valores. El valor de la función "noise", en un punto en el espacio que coincide con un punto en la malla, es este número aleatorio. Este número aleatorio para los puntos del espacio que no coinciden con un punto dentro de la malla, se obtiene por la interpolación de los puntos cercanos.

A continuación se muestra una implementación de la función noise por medio de un spline tricubico pseudoaleatorio. Dado un vector en $\Re^3$, regresa un valor entre -1.0 y 1.0. Hay dos sugerencias para hacer que el programa sea más eficiente:

- Precalcular un arreglo de gradientes pseudo-aleatorios por unidad de longitud g[n].
- Precalcular un arreglo de permutaciones de los primeros n enteros.

Dados lo dos arreglos, cualquier entero en la malla de puntos (*i, j, k*) puede ser mapeado rápidamente al vector gradiente por la siguiente relación:

g[(p[p[i] + j] % n] + k) % n]

Extendiendo los arreglos g[] y p[], g[n+i] =g[i] y p[n+i] = p[i], la búsqueda anterior puede ser reemplazada por

g[p[p[i] + j] + k]

Entonces para cualquier punto en $\Re^3$ se tienen que hacer los siguientes dos pasos::

(1) Obtener el gradiente para cada uno de los 8 puntos que lo rodean en la malla.

(2) Hacer una interpolación tricúbica

El segundo paso es evaluar el punto en la función $3t^2 - 2t^3$.

```
/* la función noise en ℜ³  - implementado por un spline tricúbico pseudoaleatorio */
#include  <stdio.h>
#include <math.h>

#define DOT(a,b) (a[0]*b[0] +a[1]*b[1] +a[2]*b[2])

#define B 256

static p[B + B + 2];
static float g[B + B + 2][3];
static start = 1;

#define setup(i, b0, b1, r0, r1) \
        t = vec[i] +10000; \
        b0 = ((int)t) & (B-1); \
        b1 = (b0+1) & (B-1); \
        r0 = t-(int)t; \
        r1 = r0 -1;
```

```
float noise3(vec)
float vec[3];
{
        int bx0, bx1, by0, by1, bz0, bz1, b00, b10, b01, b11;
        float rx0, rx1, ry0, ry1, rz0, rz1, *q, sy, sz, a, b, c, d, t, u, v;
        register i,,j;


        if (start) {
                start = 0;
                init();
        }
        setup(0, bx0, bx1, rx0, rx1);
        setup(1, by0, by1, ry0, ry1);
        setup(2, bz0, bz1, rz0, rz1);

        i = p[ bx0 ];
        j = p[ by0 ];

        b00 =  p[ i + by0 ];
        b10 = p[  j + by0 ];
        b01 = p[ i +  by1 ];
        b11 = p[  j + by1 ] ;

#define at(rx, ry, rz) ( rx *q[0] + ry * q[1] + rz * q[2] )

#define s_curve(t) ( t *t *(3. - 2. *t ) )

#define lerp(t, a, b) ( a + t * (b - a) )

        sx = s_curve)rx0);
        sy = s_curve(ry0);
        sz = s_curve(rz0);

        q = g[ b00 + bz0 ]; u = at(rx0, ry0, rz0);
        q = g[b10 + bz0]; v = at(rx1, ry1, rz0);
        b = lerp(sx, u, v);

        c = lerp(sy, a, b);     /* interpolando en y */

        q = g[ b00 + bz1 ]; v = at(rx0, ry0, rz1);
        q = g[ b10 + bz1 ]; v = at(rx1, ry0, rz1);
        a = lerp(sx, u, v);
```

```
        q = g[ b01 + bz1]; u= at(rx0, ry1, rz1);
        q = g[ b11 + bz1]; v = at(rx1, ry1, rz1);
        b = lerp(sx, u, v);

        d = lerp(sy, a, b);     interpolando en y */

        return 1.5 * lerp(sz, c, d); /interpolando en z */
}

static intit()
{
        long random();
        int y, j, k;
        float v[3], s;

        /* crear un arreglo de vectores de gradientes sobre una esfera unitaria */
        srandom(1);
        for (i=0; i<B; y++) {
                do{
                        for (j=0;  j<3; j++)
                                v[j] = (float)((random() % (B + B) / B;
                        s = DOT(v,v);
                } while (s > 1.0);
                s = sqrt(s);
        for (j=0; jz3; j++)
                g[i][j] = v[j] / s;

        /* Crear las permutaciones pseudoaleatorias de [1..B] */
        for (i=0; i<B; i++)
                p[i]=i;
        for (i=B; i>0; i-=2)[
                k = p[i];
                p[i] = p[j = random() % B];
                p[j] = k;
        }

        /* Extendiendo los arreglos g y p para indexar más rápido */
        for (i=0; i<B+2; i++){
                p[B+1]=P[I];
                for(j=0; j<3; j++)
                        g[B+i][j] = g[i][j];
        }
}
```

Simulación de turbulencia

Con la función "noise" es posible simular un gran número de efectos. Para dar mayor versatilidad a las aplicaciones se usa la función turbulencia la cual toma una posición x y regresa un valor escalar. Esto es escrito en términos de la siguiente progresión:

$$turbulence(x) = \sum_{i=0}^{k} abs\left[\frac{noise(2^i x)}{2^i}\right]$$

La sumatoria se trunca en k , que es el número entero más pequeño que satisface:

$$\frac{1}{2^{k+1}} < \textit{el tamaño del pixel}$$

Al truncar la función se asegura la propiedad de "anti-aliasing".

Con la definición de la función turbulencia sólo se resuelve la mitad del problema. Hacer el "render" de la función turbulencia directamente genera un patrón homogéneo que puede no ser muy natural. Esto se debe a que la mayoria de las texturas de los objetos naturales tienen caracteristicas no homogéneas y no pueden ser simuladas solamente con turbulencia. Tomemos el mármol, por ejemplo, en el cual se distinguen fácilmente venas de color que fueron hechas por flujo turbulento antes de que el mármol se solidificara durante la era geológica. A la luz de este hecho se pueden identificar dos pasos en el proceso de simular turbulencia - ellos son:

1. La representación de lo básico, las caracteristicas estructurales de una textura a través de algunas funciones.
2. Agregar más detalles usando turbulencia para perturbar los parámetros de la función.

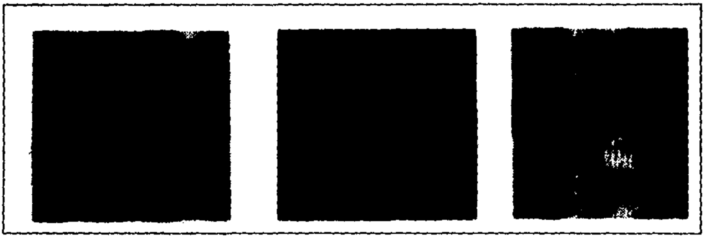
El ejemplo clásico, descrito por Perlin [17], es la turbulencia realizada con la función seno para dar la apariencia de mármol. Lo cual lo hace de la siguiente manera:

$$mármol(x) = mármol\_color(\mathrm{sen}(x + turbulence(x)))$$

El mapeo de color color_mármol() es un "spline", que mapea un valor escalar a una intensidad. En la fig. 2.1(a) se muestra una rebanada del mármol que se obtiene al hacer el "render" con el spline mostrado en la fig. 2.1(b). Después se agrega turbulencia:

$$mármol(x) = color\_mármol(\mathrm{sen}(x + turbulence(x)))$$

para obtener una simulación más convincente del mármol, ver fig. 2.1(c).



(a) (b) (c)

Fig 2.1

Simulando mármol. (a) Una rebanada del material que se obtiene al usar el "spline" de la fig. 2.1(b). (b) Spline del color. (c) Sección del mármol que se obtiene al aplicar turbulencia al spline

Por supuesto, el uso de la función turbulencia no se restringe solo a modular el color de un objeto. A ciertos parámetros que afectan el aspecto del objeto se les puede aplicar esta función. Por ejemplo se han realizado trabajos con la transparencia.

En el siguiente listado se muestra una implementación de la función "turbulence" usado la función "noise" antes mencionada:

```
/* lofreq es la frecuencia más baja de la turbulencia
   hifreq se usa para asegurar que la turbulencia este por debajo del tamaño
del pixel  */
float turbulence(point, lofreq, hifreq)
float point[3], freq, resolution;
{
        float noise3(), freq, t, p[3];

p[0] = point[0] +123.456;
p[1] = point[1];
p[2]  = point[2];

t = 0;
for (freq = lofreq; freq < hifreq; freq*=2.){
        t +=fabs(noise3(p)) / freq;
        p[0] *=2.;
        p[1] *=2.;
        p[2]*=2.;
}
return t-0.3;
}
```

Animando turbulencia

Para finalizar la discusión sobre turbulencia se presenta un ejemplo de como animarla. La función turbulencia puede ser definida en el tiempo también como en el espacio agregando una dimensión extra que represente el tiempo dentro de la malla de enteros de la función "noise". Así que la malla de puntos estará especificada ahora por los índices (*i, j, k, l*) lo cual permite extender los parámetros de noise(*x, t*) y similarmente en *turbulence(x, t)*.

Se quiere simular fuego, así que lo primero que se tiene que hacer es tratar de representar su forma básica por medio de funciones, es decir su forma de flama.

Una región de una flama está definida en el plano x-y por el rectángulo con coordenadas (-b, 0), (b.h). Dentro de esta región el color de la flama está dado por (ver fig. 2.2):

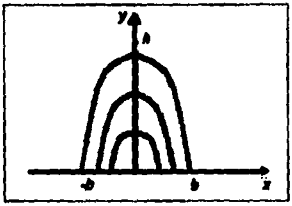


Fig. 2.2.
Función flama

La función *color_flama(x)* consiste de tres "splines", uno para cada uno de los componentes de color RGB, que mapea un valor escalar x a un vector color. Cada uno de los "splines" tienen una intensidad máxima en $x = 0$ el cual corresponde al centro de la flama . Los splines para el verde y el azul tienden más rápido a cero que el rojo. El color que regresa *color_flama()* es proporcional a la altura, la distancia desde la base de la flama. Se hace el "render" de la flama aplicando la función *flama()* al polígono rectangular que cubre la región de la flama. Para dar una apariencia perturbada de la flama se agrega la función turbulence:

$$flama(x, t) = (1 - y/h)color\_flama(abs(x/b)) + turbulence(x, t)$$

Como el "render" se está realizando en dos dimensiones, el proceso puede ser visualizado con una parte del noise tridimensional, al reemplazar el eje z con el tiempo. Simplemente se está haciendo el "render" de rebanadas de noise las cuales son perpendiculares al eje del tiempo. Es como si estuviéramos trasladando el polígono a través del eje del tiempo. Sin embargo, el usar sólo la traslación no es suficiente; los detalles en la flama, aún cuando cambia la forma con el tiempo, permanecen estáticos en el espacio. Esto es porque hay un sentido general de dirección asociado con una flama; envía los detalle hacia arriba. Esto fue simulado moviendo el polígono hacia abajo sobre el eje *y* y através del tiempo, como se muestra en la fig. 2.3. La construcción final es por lo tanto:

$$flama(x, t) = (1 - y/h)color\_flama(abs(x / b) + turbulence(x + (0, t\Delta y, 0) , t))$$

donde Δy es la distancia recorrida en *y* por el polígono relativa al noise por unidad de tiempo.

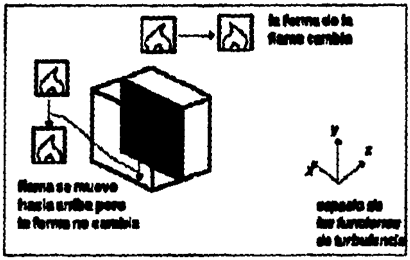


Fig. 2.3

Animando la turbulencia para un objeto en dos dimensiones

Síntesis de Fourier

Síntesis de Fourier es una de las técnicas más poderosas y comunes en la generación de texturas.

Schatter [24] fue el primero que utilizó la síntesis de Fourier para generar un patrón de textura bidimensional usada en la simulación de vuelos. En esta aplicación, la amplitud de las frecuencias *F(u, v)*, algunas veces llamadas la cresta de las ondas, son usadas dentro de un procedimiento para generar una textura bidimensional que posteriormente es puesta sobre un plano para simular la textura de un terreno. Un patrón se produce sumando los componentes de ondas al variar su frecuencia y su fase.

3. ALGORITMOS PARA LA SIMULACION VISUAL DE FENOMENOS GASEOSOS

Introducción

En nuestra vida cotidiana observamos fenómenos gaseosos tales como: humo, vapor, niebla y nubes. El realismo y el humor de escenas al aire libre, tales como un oscuro y melancólico bosque puede ser incrementado bastante agregando elementos tales como humo. El vapor y otros gases pueden también incrementar el realismo de una escena, tal es el caso del vapor saliendo de una taza de café o saliendo de un lago en una fría mañana.

Se tienen serios problemas para generar los fenómenos gaseosos, las técnicas estándar que existen para la generación de imágenes no son suficientes, ya que éstos no tienen superficies ni límites bien definidos.

La graficación por computadora ha ido superando los obstáculos que se le han presentado. Los árboles, sombras, reflexiones son ejemplos de obstáculos que sido superados por innovaciones realizadas en la simulación por computadora. Un obstáculo más es la simulación visual de los fenómenos gaseosos. Para conseguir esta meta se han desarrollado varios métodos, algunos se han enfocado a la simulación de nubes, y otros abarcan todo tipo de fenómenos gaseosos.

3.1. Algoritmos enfocados a la simulación de nubes

Se tienen serios problemas para generar nubes utilizando las técnicas estándar que existen para la generación de imágenes por computadora pues las nubes no tienen superficies ni límites bien definidos. Además, las nubes tienen varios grados de translucidez, y su estructura amorfa puede cambiar con el tiempo.

Las nubes son elementos críticos en el combate aéreo. Por más de dos décadas, la Generación de Imágenes por Computadora (GIC) ha sido usada para visualizar la simulación de vuelos, y en ella no han sido tomada en cuenta las nubes. Las nubes son importantes en la simulación de sistemas inteligentes de armas que identifican blancos aéreos. En el campo de la meteorología tener una simulación realista de las nubes es también importante. Como las nubes son objetos cotidianos, es deseable simularlas en otras aplicaciones, tales como entretenimiento, anuncios y arte. Por tal razón es necesario desarrollar métodos que permitan realizar la simulación visual de ellas.

3.1.1. Modelo de Gardner para la simulación de nubes

En esta sección se describe una técnica para simular nubes usando superficies planas y curvas cuya sombra y translucidez es modulada por una función matemática de textura.

Modelo

Para simular escenas con nubes, debemos poder modelar diferentes tipos de nubes vistas desde diferentes distancias y ángulos. Afortunadamente la clasificación estándar de los tipos de nubes está basada en su apariencia. Las tres clases fundamentales son: cirros ("rizo del cabello"), estratos ("capa"), y cúmulo ("montón"). Las nubes cirros son nubes que están en grandes alturas (5-13km.). Las nubes estratos son nubes que no tienen detalles distintos, i.e. no se puede distinguir donde termina una y empieza la otra, y se encuentran altitudes bajas (0-2km.). Las nubes cúmulo son nubes amontonadas, también se localizan en bajas altitudes. Las combinaciones de estas clases de nubes son usadas para describir nubes con características y altitudes combinadas de los tipos básicos. "Cirroestrato" se refiere a un estrato de nubes cirro a gran altura. "Estratocúmulo", "Altocúmulo", y "Cirrocúmulo" se refiere a bajos, intermedios y altos estratos de nubes separados, respectivamente. "Altoestrato" se refiere a un denso estrato de nubes en altitudes intermedias (2-8km.) "Nimboestrato" se refiere a grandes cúmulos de nubes que producen un aguacero.

Para desarrollar un modelo de simulación visual de nubes se debe considerar la formación de la mismas. Dicha formación puede ser horizontal (estratos de nubes) o vertical (nubes cúmulo).

El modelo se compone de tres aspectos:

1. Un plano para el cielo
2. Elipsoides
3. Una función matemática para generar textura.

Se define el plano para el cielo paralelo al plano del suelo en una altitud especificada y se le utiliza para modelar un estrato de nubes de dos dimensiones. Se define el plano del suelo en coordenadas de la escena (X,Y,Z), de esta forma el plano del

$$P(X,Y,Z) = Z - A = 0 \qquad \text{... Ec. (3.1)}$$

suelo es el plano (X-Y). Por lo tanto el plano del cielo puede ser escrito como:
donde A es la altitud del plano

Se usan elipsoides para modelar el grosor de la estructura tridimensional de las nubes. Una elipsoide típica es expresada como:

$$Q(X,Y,Z) = Q_1 X^2 + Q_2 Y^2 + Q_3 Z^2 + Q_4 XY + Q_5 YZ + Q_6 XZ + Q_7 X + Q_8 + Q_9 + Q_0 = 0 \quad \ldots \text{Ec.}(3.2)$$

Se usa una función matemática para generar la textura y por medio de ella se modelan los detalles de las nubes modulando la intensidad del sombreado y translucidez del plano del cielo y las elipsoides. Para definir esta función se utilizan series de Fourier y la función seno, obteniendo la siguiente ecuación:

$$T(X,Y,Z) = k\sum_{i=1}^{n}\{C_i Sen(FX_i X + PX_i) + T_0\}\sum_{i=1}^{n}\{C_i Sen(FY_i Y + PY_i) + T_0\} \qquad \ldots \text{Ec. } (3.3)$$

Para producir un patrón de textura con apariencia natural las frecuencias y coeficientes se escogen con las siguientes relaciones:

$$FX_{i+1} = 2FX_i$$

$$\ldots \text{Ec. } (3.4)$$

$$FY_{i+1} = 2FY_i$$

$$C_{i+1} = .707C_i \qquad \ldots \text{Ec. } (3.5)$$

PX_i y PY_i son cambios de fase para agregar aleatoridad y están dados por las siguientes relaciones:

$$PX_i = \pi / 2 \operatorname{sen}(.5FY_i Y) = \pi / 2 \operatorname{sen}(FY_{i-1} Y)$$

$$\ldots \text{Ec. } (3.6)$$

$$PY_i = \pi / 2 \operatorname{sen}(.5FX_i X) = \pi / 2 \operatorname{sen}(FX_{i-1} X)$$

Los cambios de fase producen un efecto pseudo aleatorio al poner los senos de PX_i como una función de Y y los de PY_i como una función de X.

Para dar variaciones tridimensionales a la textura de las elipsoides, los cambios de fase se incrementan con una función seno que dependa de Z:

$$PX_i' = PX_i + \pi \operatorname{sen}(FX_i Z / 2)$$

...Ec. (3.7)

$$PY_i' = PY_i + \pi \operatorname{sen}(FX_i Z / 2)$$

T_0 es un parámetro que controla el contraste del patrón de textura, y k está calculada de tal forma que $T(X,Y,Z)$ tenga un valor máximo de 1.

La función de textura modula la intensidad de sombreado (iluminación) del plano correspondiente al cielo o de las elipsoides, utilizando la siguiente ecuación:

$$I = (1-a)\{(1-t)[(1-s)I_d + sI_s] + tI_t\} + a \qquad \text{...Ec. (3.8)}$$

donde

a es la reflexión de la superficie debida a la luz ambiental

t es sombra producida por la textura

s es la fracción definida por la reflexión especular

I_d es la intensidad debida a la reflexión difusa

I_s es la intensidad debida a la reflexión especular

I_t es la intensidad que contribuye la función de textura

($I_t = T(X,Y,Z)$).

I es la intensidad de la superficie

Los valores de I_d e I_s son calculados de la forma usual, utilizando las relaciones entre la normal a la superficie, dirección de la luz y observador.

· Se modula la translucidez del plano correspondiente al cielo, definiendo un valor inicial para la función de textura. La translucidez está dada por la siguiente ecuación:

$$TR = 1 - (I_t - T_1) / D$$

... Ec. (3.9)

$$0 < TR < 1$$

donde T_1 es el valor inicial

D es un rango de valores para la función de textura a través del cuál la translucidez varía de 0 a 1.

TR es la translucidez

Se modula la translucidez de las elipsoides de una manera similar, pero hay que variar el valor inicial para incrementar la translucidez en los límites de la elipsoide. Para hacer esto se usa la ecuación de la orilla de la elipsoide, la proyección elíptica de la silueta de la elipsoide en el plano (X-Z). Esta ecuación es escrita en coordenadas de la imagen de la siguiente manera:

$$f(x,z) = a_1x^2 + a_2z^2 + a_3xz + a_4x + a_5z + a_6 = 0 \qquad \text{... Ec. (3.10)}$$

donde z es la coordenada vertical de la imagen y

x es la coordenada horizontal de la imagen.

Para los puntos que se encuentran en el interior de la curva correspondiente a la orilla, que son los puntos de la elipsoide, $f(x,z)$ es diferente de cero, obteniendo el valor máximo en el centro de la curva. Si se divide $f(x,z)$ entre su valor máximo se obtiene una función normalizada de la curva, $g(x,z)$, la cual varia de 0 a 1. Con esta función normalizada es posible modificar la ecuación (3.9), ecuación utilizada para la translucidez en el plano, de tal manera que la translucidez en los límites de la elipsoide se incremente. Utilizando esta función normalizada a partir de la ec. (3.9) obtenemos:

$$TR = 1 - (I_t - T_1 - (T_2 - T_1)(1 - g(x,z))) / D$$

...Ec. (3.11)

$$0 < TR < 1$$

Modelo bidimensional de nubes

Un modelo eficiente de un estrato de nube puede ser obtenido al poner textura al plano del cielo (Ec. (3.1)). Se puede colocar el plano en alguna altitud, *A*, y se puede especificar la solidez y densidad del estrato de nube por medio de los parámetros de translucidez T_1 y D (Ec. (3.9)). Se puede definir el contenido espectral del patrón de la nube a través de los parámetros de la función de textura C_i, FX_i, FY_i, y T_0 (Ec. (3.3)).

Por ejemplo, se puede modelar un estrato de cirros en una altitud de 10 km. Se representa la característica de nubes cirros especificando FX_i como el doble de FY_i y usando valores grandes para T_1 y D para producir grandes regiones de translucidez y gradualmente obtener nubes opacas.

Se puede modelar una nube estrato en una altitud típica de 2 km. incrementando T_0 en la función de textura, para reducir la translucidez y el contraste.

Se puede usar el modelo bidimensional para representar la formación de una nube cúmulo.

El plano del cielo con textura puede ser usado para simular la dinámica de las nubes variando los parámetros de la función de textura en el tiempo. Se puede simular el desarrollo de una nube decrementando los valores de T_1 y D. Se puede también mover el patrón de textura a través del plano para simular viento.

El uso del modelo bidimensional es limitado porque no tiene profundidad vertical. Por esta razón debe ser visto en una distancia, sirviendo como fondo para escenas tridimensionales. Otra limitación es que el mismo patrón de textura se amplia a través de todo el plano con lo cual se incluye las formaciones de nubes en diferentes partes del cielo.

Modelo tridimensional de nubes

Se ha escogido la elipsoide como bloque de construcción básico para modelar la estructura tridimensional de las nubes. Una elipsoide puede se usada para definir un volumen con solamente nueve parámetros, tres para especificar el tamaño y la forma, tres para especificar la posición, y tres para especificar la orientación angular. Como en el modelo bidimensional se utiliza la función de textura (Ec (3.8) y (3.11)) para simular el sombreado y translucidez. El modelo de textura puede ser definido de una manera directa especificando las frecuencias y amplitudes de ondas de la función seno para producir el contenido espectral deseado y seleccionando los valores iniciales de translucidez que produzcan la densidad deseada de las nubes.

Observando las nubes, es posible notar que algunas de ellas tienen una forma similar a una elipsoide. Sin embargo, en general hay una gran variedad de formas que no pueden ser modeladas de una forma exacta por medio de elipsoides. Para contemplar esta variedad, este modelo tridimensional nos brinda la facilidad de asociar varias elipsoides para crear formas más complejas. En la aproximación original para juntar las superficies cuadráticas, se escoge insertar planos limitados en donde se juntan las cuadráticas. Esta estrategia fue para evitar el cálculo de curvas de cuarto orden que se producen en la intersección de las superficies cuadráticas. Esta aproximación trabaja bien para objetos sólidos, pero tiende a producir límites no planos no naturales entre objetos altamente traslúcidos. Para evitar estas anomalías, se han incluido en el modelo de nubes tridimensional la capacidad de manejar la intersección de las superficies cuadráticas sin planos limitados. Para hacer esto se han extendido los algoritmos utilizados en la generación de imágenes para el cálculo de los puntos visibles en la curva de intersección de las dos superficies cuadráticas. Para minimizar el costo computacional que se ha agregado, se realiza el cálculo de forma "scan line" (línea a línea de la imagen a ser generada) y ordenamos los objetos por profundidad. Usando los puntos que se calcularon en la intersección, se determinan regiones del "scan line" en la cual la superficie que se está analizando es constante. Entonces se reordenan las superficies de adelante para atrás en la región del "scan line".

Formaciones horizontales de nubes

Para modelar un amplio rango de formaciones horizontales, se definen grupos para modelar escenas. Para cada grupo se define una elipsoide patrón especificando los parámetros de tamaño y orientación. Se define un límite elíptico para cada grupo, y la altitud de la nube. Se realiza una réplica de la elipsoide patrón dentro del límite definido en las posiciones que se van generando. El tamaño, la posición y la orientación de las elipsoides se varían aleatoriamente. A todas las elipsoides del grupo se les asigna un color común y los mismos parámetros de textura. Se puede modelar una clase específica de nubes seleccionando la forma de las elipsoides en el grupo, la altitud del grupo, la frecuencia de la función de textura y los parámetros de translucidez. Por ejemplo, se usan elipsoides redondas y un patrón de textura con un alto contraste para generar nubes cúmulo, y usar elipsoides huecas y un patrón de textura con bajo contraste para nubes estratos.

Para facilitar el modelado de escenas complejas de nubes, se utilizan macrogrupos, o grupos de grupos. Cada macrogrupo es definido de la misma forma como se hace con un grupo, en una región horizontal, una elipsoide patrón, una altitud, un color, y un patrón de textura.

Formaciones verticales de nubes

Se puede modelar el desarrollo vertical de las nubes de diferentes maneras. Se puede representar un desarrollo vertical de las nubes simplemente incrementado el tamaño de las elipsoides. Variando el tamaño de la elipsoide de una manera controlada, se puede modelar tanto el desarrollo vertical como horizontal.

Un desarrollo vertical debido a un viento que sopla hacia arriba puede ser simulado moviendo las elipsoides hacia arriba. Para esto se usa una técnica que fue planeada originalmente para árboles en una colina, la cual permite definir objetos "terrenales" sobre los cuales serán colocados los objetos que se vayan creando. Al ir creando cada objeto, su posición (X,Y) es comparada contra la misma posición (X,Y) de la escena creada con objetos terrenales. Si un objeto "terrenal" ocupa la posición (X,Y) del objeto creado éste ajusta en altitud para lo que se agrega la altura del objeto "terrenal" al valor de la posición (X,Y) del objeto creado.

3.1.2. Algoritmo basado en síntesis espectral

En esta sección se describe un método también para nubes basado en la teoría de la turbulencia espectral. La animación se realiza por medio de un cambio de fase en el dominio de las frecuencias según la ley de Kolmogorov [16]. Los parámetros del modelo de turbulencia son intuitivos.

3.1.2.1. Bases de la síntesis espectral

Síntesis espectral estocástica

En contraste a los métodos que están en el espacio Euclidiano, la síntesis espectral define la textura en el espacio de Fourier o dominio de frecuencias generando el espectro apropiado, para lo cual toma la función que está en el espacio Euclidiano y la transforma al espacio de Fourier.

La representación de una función de textura en el espacio Euclidiano y en el espacio de Fourier se acoplan al existir la Transformada de Fourier (TF). La tabla 1 contiene una explicación de la notación reflejada

.Tabla 1. Lista de símbolos

Símbolo	Explicación
$g(x, y)$	función en el espacio Euclidiano
$G(u, v)$	función en el espacio de Fourier
N	resolución del espectro y de la imagen
λ	longitud de onda
f, k	frecuencia, número de onda
ε	disipación
ν	exponente de viscosidad cinemática
ρ	densidad de la nube
Λ	tamaño de la granularidad
L, f_L	tamaño y frecuencia de la perturbación más grande
—	velocidad del viento
u'	velocidad de la turbulencia
u_L	la velocidad de la perturbación más grande
β	exponente espacial
κ	exponente de turbulencia

$$g(x,y)=\sum_{u=-N/2}^{N/2-1}\ \sum_{v=-N/2}^{N/2-1} G(u,v)\,e^{i2\pi(ux+vy)/N}$$

$$x,y=0,1,\mathrm{K}\,,N-1$$

$$G(u,v)=\frac{1}{N^2}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} g(x,y)\,e^{-i2\pi(ux+vy)/N}$$

$$u,v=-\frac{N}{2},\Lambda\ \frac{N}{2}- \qquad \text{... Ec. (3.12)}$$

La transformada discreta de Fourier descompone un campo discreto real o complejo (función o muestra) en series de senos y cosenos, u ondas senoidales, de amplitud variable, fases, y frecuencias. Las dos representaciones de la función son equivalentes, lo que significa que es posible transformar una función dada en el espacio de frecuencias o definir una función en el dominio de frecuencias y obtener la función en el espacio Euclidiano. De acuerdo a la notación utilizada en la Ec. (3.12) la transformada de Fourier de una función real se representa por series de factores complejos (coeficientes) *a+ib*, o en coordenadas polares:

$$a+ib=re^{i\phi}, \qquad r=\sqrt{a^2+b^2}, \qquad \phi=\arctan\left(\frac{b}{a}\right) \qquad \text{... Ec. (3.13)}$$

La magnitud *r* de un coeficiente de Fourier es la amplitud del término correspondiente (onda) en la transformada de Fourier, por el cual el ángulo de fase f determina el cambio de la onda con respecto al origen del sistema de coordenadas. La distancia Euclidiana del origen al coeficiente del espectro es llamada su frecuencia. En el caso de una señal donde varia el tiempo, la frecuencia *f* tiene unidades de ciclos por unidad de tiempo; en el caso de una señal donde varía el espacio, la frecuencia es llamada onda número *k* y es medida en ciclos por unidad de longitud. Los conjuntos de todas las magnitudes y fases de una representación de Fourier, como una función de la frecuencia, son llamados *amplitud espectral y fase del espectro*, respectivamente. En el caso de una campo discreto definido por una muestra de *N* puntos, ambos espectros son discretos y tienen una resolución igual a la resolución del campo. Este caso se da cuando tenemos imágenes "raster" en 2-D o "voxels" en 3-D. El espectro de una función estocástica es caracterizada por la distribución de los (amplitud y fase) coeficientes como una función de la frecuencia.

Para este trabajo se consideran solo funciones reales, tales como imágenes "raster". Para funciones reales, lo siguiente es válido:

$$G(u) = G^*(-u), \qquad \text{... Ec. (3.14)}$$

donde $G^*(-u)$ es el conjugado complejo del valor de $G(u)$. Esto significa que para funciones reales solamente una mitad del espectro tiene que ser definido, mientras que la otra mitad se obtiene de acuerdo a la Ec. (3.13); (ver Fig 3.1). Esta propiedad de simetría es usada para acelerar el calculo de la transformada en campos reales, tales como imágenes "raster". Por lo tanto, una traslación en el espacio Euclidiano resulta un cambio de fase en el espacio de Fourier y viceversa.

$$g(x - x_0) = G(u)e^{-2\pi i u x_0/N} \qquad \text{... Ec. (3.15)}$$

El valor promedio de la función está dado por $G(0)$:

$$\mu = \overline{g(x)} = \frac{1}{N}\sum_{x=0}^{N-1} g(x) = G(0) \qquad \text{... Ec. (3.16)}$$

El valor al cuadrado es:

$$\overline{g^2(x)} = \frac{1}{N}\sum_{x=0}^{N-1} g^2(x) = \sum_{\mu=0}^{N-1} |G(u)|^2 \qquad \text{... Ec (3.17)}$$

La varianza es:

$$AR(g(x)) = \overline{g^2(x)} = \mu^2 = \sum_{u=1}^{N-1} |G(u)|^2 \quad \text{... Ec. (3.18)}$$

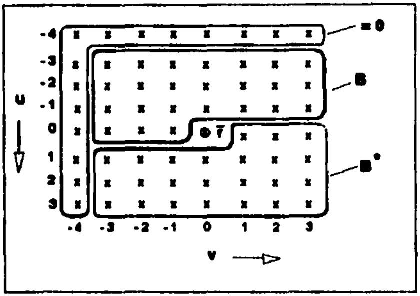


Fig 3.1

Espectro de una función discreta en 2-D con sus partes: independiente (B) y conjugado complejo (B*) f es el valor promedio

Bases de la teoría de turbulencia

La teoría espectral estática, homogénea, localmente isotrópica, con turbulencia no intermitente, fue presentada en el trabajo de Kolmogorov, Obukhov y Reynolds. Esta teoría describe globalmente a la turbulencia usando métodos estadísticos en vez de ecuaciones de mecánica de fluidos. Los resultados que se obtienen con esta aproximación frecuentemente son limitados, pero el método da una solución global razonable y se evitan los cálculos excesivos requeridos para resolver ecuaciones diferenciales dinámicas. Para mayores referencias se recomienda a Panchev [16] es un libro de texto excelente y fácil de leer, y para un análisis más profundo a Frost y Moulden[5] y a Tennekes y Lumley[25].

De acuerdo a Reynolds, la velocidad de una corriente turbulenta puede ser considerada como la superposición de dos movimientos: $u = \overline{U} + u'$, donde $\overline{U}$ es un promedio de la velocidad traslativa y u' un movimiento aleatorio de fluctuación. $\overline{U}$ es responsable del movimiento de translación global, mientras que u' causa un cambio contínuo en la estructura del flujo. Este movimiento de fluctuación surge al existir simultáneamente perturbaciones (remolinos) de diferentes tamaños λ, y cada una de éstas tiene una velocidad de u_λ. La superposición de esas perturbaciones da como resultado un movimiento caótico que es característico del flujo turbulento. Las perturbaciones construyen un cascada a través de la cual la energía es transferida del movimiento global traslativo al movimiento turbulento y se transforma en calor debido a la fricción. La energía cinética transformada a calor por unidad de tiempo y masa es $\varepsilon = \nu u_\lambda^2/\lambda^2 = u_\lambda^3/\lambda$. De acuerdo a la hipótesis de Kolmogorov, la influencia de fricción en una corriente turbulenta es insignificante en las perturbaciones excepto en las de menor escala. Esto significa que la energía que se obtiene del movimiento principal no se pierde durante la transformación de las perturbaciones de tamaño intermedio, sino solamente se redistribuye entre las otras perturbaciones que están en la cascada, la fricción y la difusión de calor se activan solamente en las perturbaciones de menor escala. Consecuentemente la disipación ε permanece constante dentro del subrango inercial. El rango de escalas en el cual esta hipótesis es válida es llamado el subrango inercial. Dentro del subrango inercial, la velocidad de oscilación asociada a las perturbaciones de escala l está dada por la siguiente ecuación:

$$u_\lambda = (\varepsilon\lambda)^{1/3} \qquad \text{... Ec. (3.19)}$$

Kolmogorov teóricamente estimó el espectro de densidad-energía correspondiente a la distribución de las velocidades de las perturbaciones que están en el subrango inercial como:

$$S(k) = 1.22\varepsilon^{2/3}k^{-5/3} = 1.22u_L \,/\, L^{2/3}k^{-5/3} \approx u_L^2 \qquad \text{... Ec (3.20)}$$

Expresando el espectro de amplitudes correspondiente, i.e. las magnitudes de los vectores de velocidad, para el caso n-dimensional resulta en:

$$(k_1, K, k_n) = 1.1\varepsilon^{1/3}\sqrt{k_1^{\,2} + K\, k_n^{\,2-(5/3+n-1)/2}} \approx u_L \sqrt{k_1^{\,2} + K + k_n^{\,2-(k+n-1)/2}} \quad \text{... Ec (3.21)}$$

Otras expresiones para el espectro que han sido dadas por Botcheler, Goltsin, Hessenberg, Kármán, Ogura, Pao y Yaglom están basadas en datos experimentales, las que pueden ser encontradas en Panchev[16]. Una característica común de todos los espectros es que incluyen un amplio subrango inercial con una dependencia de la frecuencia exponencial del tipo $S(f) \approx 1/f^{k}$; por lo tanto, el espectro de Kolmogorov es un caso especial. Para los propósitos del modelado por computadora se considera solamente la parte del subrango inercial del espectro y es ignorado el movimiento turbulento de las perturbaciones de tamaño mayor a uno dado.

$$D(k) \approx \rho^2 k^{-7/3} \approx \rho^2 f^{-k} \quad \text{... Ec. (3.22)}$$

$$A(k_1, K, k_n) \approx \rho\sqrt{k_1^{\,2} + K\, k_n^{\,2-(7/3+n-1)/2}}$$

$$\approx \rho\sqrt{k_1^{\,2} + K\, k_n^{\,2-(\beta+n-1)/2}} \quad \text{... Ec. (3.23)}$$

¿Qué es una perturbación?

Como se mencionó en la descripción de turbulencia, una perturbación es una alteración local de cierto tamaño y velocidad dentro del campo turbulento. El término alteración local significa que una perturbación tiene asociada una localización espacial dentro de la turbulencia, aún si su posición precisa no está dada o no es de interés. Una perturbación puede ser visualizada como un vórtice, aunque las perturbaciones no muestren necesariamente movimientos rotacionales. De otra manera, cuando se usa la teoría espectral, una perturbación es representada como el componente de velocidad del espectro. Por lo tanto, cada coeficiente de un espectro (discreto), tal como los usados en la síntesis de Fourier, es asociado con una perturbación de cierto tamaño (longitud de onda).

Hay una diferencia significativa entre esas dos representaciones: un coeficiente de Fourier en una longitud de onda dada es, por definición un valor promedio que incorpora las contribuciones de todas las perturbaciones del mismo tamaño independientemente de la localización de las mismas. Por lo tanto, un coeficiente de Fourier no tiene sentido de la posición en el espacio Euclidiano y, por consiguiente no puede ser expresado como una estructura espacial local. Así que, un coeficiente de Fourier no debe ser confundido con una simple perturbación. No obstante, al emplear síntesis espectral, una persona se refiere a los coeficientes espectrales como perturbaciones. En este caso, el campo es generado sobreponiendo ondas senoidales de diferentes tamaños y amplitudes, de la misma forma como un campo turbulento es considerado como la superposición de perturbaciones de diferentes escalas y velocidades.

3.1.2.2. Simulación visual

En el modelado del movimiento de gases turbulentos usando síntesis espectral el diseñador puede distinguir entre frecuencia del espectro, expresada como funciones de $\omega = 2\pi f$ que describen el comportamiento temporal de la turbulencia (i.e. como estructuras que cambian su forma con el tiempo) y número de onda del espectro, expresado como funciones de k que describen la estructura del campo turbulento. Si $\overline{U} \gg u'$, la hipótesis de turbulencia congelada de Taylor puede ser aplicada, y ambos espectros pueden ser considerados iguales. Esto es usado en el método de rebanadas, en el que el último eje de un fractal (n+1)-dimensional es considerado como el tiempo.

La simulación visual se describe a continuación. Una nube se construye sobreponiendo ondas senoidales con amplitudes definidas a partir de la ec. (3.23) y fases aleatorias. Se genera la amplitud y fase del espectro para definir la estructura inicial espacial de la nube, especificando la distribución de la densidad del gas (tal como vapor y humo) en el espacio. Debido a la presencia de viento con velocidad $\overline{U}$, la nube como un todo tiene que moverse traslativamente (i.e. sin cambiar en su estructura espacial) en la dirección del viento. Cada onda con una frecuencia mayor que la frecuencia dada f_L es considerada como una perturbación y tiene una velocidad adicional de fluctuación con una orientación aleatoria y una amplitud característica de su tamaño (longitud de onda), como está dado en la ec. (3.21) (la frecuencia f_0 de los remolinos más pequeños siempre es adaptada al tamaño del pixel). Este movimiento de fluctuación u' para ondas turbulentas es sobrepuesto al viento traslativo $\overline{U}$ aplicado a todas las ondas. Por lo tanto durante la animación cada una de las ondas en el espacio se mueven de un cuadro a otro a una velocidad que resulta de la suma de los componentes de traslación y fluctuación. En otras palabras, la "ensalada de ondas" inicial es mezclada a cada momento del tiempo, y el correspondiente campo turbulento es calculado. Finalmente, los valores calculados para cada cuadro deben ser mapeados a los colores de los pixeles o textura usando un método de visualización. Estos tres pasos (generación, animación y mapeo) serán discutidos en las siguientes secciones.

Generación

Mandelbrot [8] y Lavejoy [6] proponen que las imágenes estáticas de campos turbulentos pueden ser considerados como fractales con un exponente de Hurt $H \approx 0.7$. Es interesante notar que el valor teórico $\beta = 7/3$ de ec.(3.23) resulta en un exponente de Hurst $H = 2/3 = 0.66$, el cual corresponde a los valores obtenidos experimentalmente de las medidas de turbulencia. Voss [20] Y Peitgen y Saupe [13] introdujeron el movimiento Browniano fraccional en la graficación por computadora con la finalidad de modelar imágenes realistas de nubes con fractales aleatorios. El movimiento Browniano fraccional tiene una amplitud espectral de $1/f^{\beta}$ y una fase del espectro aleatoriamente distribuida. Aunque ambas (amplitud y fase) pueden ser cambiadas aleatoriamente en este trabajo solamente se cambia aleatoriamente la fase y a la amplitud se le da un valor de r/f^{β}.

Animación en el dominio de frecuencias

La razón más importante para emplear síntesis espectral es la forma natural de animar funciones estáticas. Como la función es expresada como la suma de todas las ondas senoidales o remolinos del espectro, la función puede ser animada si cada una de las ondas es animada. Como se puede ver en la ec. (3.15), una traslación (movimiento) en el espacio Euclidiano produce un cambio de fase en el dominio de frecuencias. La velocidad u_f de un pixel/cuadro para cada onda dada con frecuencia f y el correspondiente cambio de fase $\Delta\phi_f$ entre dos cuadros es acoplado por:

$$\Delta\phi_f = \frac{u_f 2\pi f}{N} = \frac{u_f \omega}{N} \qquad \text{... Ec. (3.24)}$$

El cambio de fase para el viento traslativo U es calculado como

$$\Delta\phi_{f,\tilde{U}} = \frac{\tilde{U} 2\pi f}{N} \cos(\vec{U}, \vec{f}) \qquad \text{... Ec. (3.25)}$$

El término coseno $\cos(\vec{U}, \vec{f})$ resulta de la proyección del vector viento en la dirección de propagación de cada onda, fig 3.2 Este cambio de fase es calculado para todas las ondas en la parte superior de la mitad del espectro y es aplicado a todas las ondas. Los valores del cambio de fase son almacenados en un campo y recalculados solamente después de un cambio en la dirección o magnitud del viento.

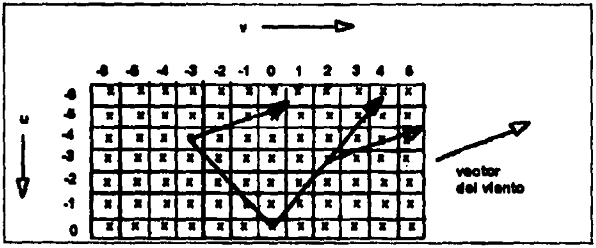


Fig. 3.2

Proyección del vector del viento en la dirección de la propagación de onda

Las ondas con frecuencia mayor a f_L tienen un movimiento de fluctuación adicional u'. La magnitud de u' resultante del espectro de Kolmogorov y expresada como un cambio de fase, es:

$$\Delta\phi_{f,u'} = \frac{u_L(f/f_L)^{-(k+n-1)/2}\,2\pi f}{N} \qquad f \geq f_2 \qquad \text{... Ec. (3.26)}$$

En el paso de inicialización, el diseñador genera una dirección aleatoria y un valor u'_f para cada onda $w(u,v)$ que está en la mitad superior del espectro B (ver fig. 1), las cuales tienen una frecuencia f que es mucho mayor que la frecuencia f_2 (frecuencia de las perturbaciones más grandes) usada en la ec (3.26). Como las ondas de Fourier pueden moverse solamente es su dirección de propagación, el vector $\vec{u}$ tiene que ser dividido en dos componentes, uno paralelo y uno perpendicular a la dirección de la onda de propagación. El componente paralelo es asignado a la $w(u,v)$ y el componente perpendicular a la onda $w(-v,u)$, como se muestra en la fig. 3.3.

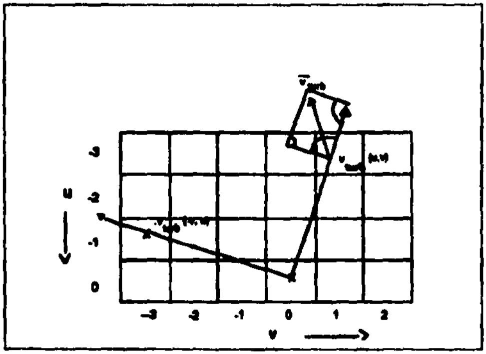


Fig. 3.3

Dividiendo la velocidad aleatoria del vector u' a lo largo de las ondas w(u,v) y w(-v,u)

Otra vez, estos valores son almacenados y recalculados después de cada cambio de f_L, u_L, o k. Los componentes resultante del cambio de fase son:

$$\Delta\phi(u,v) = \frac{u_L(f/f_L)^{-(k+n-1)/2}\,2\pi f}{N}\cos(\vec{u},\vec{f}) \qquad f \geq f_2$$

$$\Delta\phi(-u,v) = \frac{u_L(f/f_L)^{-(k+n-1)/2}\,2\pi f}{N}\,\mathrm{sen}(\vec{u},\vec{f}) \qquad \text{... Ec. (3.27)}$$

Para prevenir el "aliasing", el valor de un cambio de fase entre dos cuadros adyacentes debe ser menor que π. Los valores independientes son asignados solamente a la mitad superior, i.e. el espectro B; la otra mitad B^* se completa con los conjugados complejos de las ondas. El modelo de turbulencia descrito anteriormente utiliza siete parámetros:

(1) Dimensión fractal D o exponente de Hurst H (exponente β).

(2) Estructura espacial de tamaño L.

(3) Velocidad traslativa $\bar{U}$.

(4) Dirección de traslación $\vec{w}$ (dirección del viento).

(5) La longitud de onda L o frecuencia f_L.

(6) La velocidad de las perturbaciones más grandes u_L.

(7) El exponente k, correspondiente a la velocidad del espectro

La dimensión fractal describe la "dureza" de la nube inicial y L su granularidad. H y L determinan la estructura espacial de la nube, mientras que f_L y el exponente k controlan el comportamiento temporal de la turbulencia. La turbulencia global puede obtenerse decrementando f_L, y por medio de una perturbación pequeña se puede generar turbulencia local. Valores grandes de k atenúan las frecuencias altas. La velocidad de la turbulencia u_L no afecta la estructura de la turbulencia, solamente define la velocidad de las transiciones. Por lo tanto puede ser comparada con la rapidez de "play-back" de una videograbadora que graba el movimiento turbulento que viaja con la rapidez del viento $\bar{U}$. Es importante notar que el diseñador puede escoger diferentes exponentes para las estructuras de la turbulencia temporal y la espacial, con lo cual se incrementa la flexibilidad del diseño.

Métodos de visualización

Los valores de la textura estocástica generada son números reales positivos y negativos, así que se requiere realizar una transformación a dichos valores antes de realizar el "render". La textura generada puede ser mapeada a los atributos de un cuerpo (superficie o volumen). Para ilustrar, se mapean los valores de la textura en el color, no obstante, cualquier otro atributo (tal como transparencia y densidad) puede manejarse bastante bien.

Para visualizar nubes en 2D, usamos una paleta de 256 colores, que va desde el negro al blanco. El método usual es asignar al negro el valor mínimo y al blanco el valor máximo del campo aleatorio y calcular los valores intermedios por medio de una interpolación lineal. Desafortunadamente, los valores máximos y mínimos del campo estocástico cambian durante la secuencia de la animación, lo cual provoca transiciones abruptas en la densidad de las nubes entre cuadros adyacentes. Para evitar esto, se generar los valores, de tal forma que la mayoría de los valores estén en el intervalo de 0-255, con lo cual se podrá accesar directamente la paleta de colores. Para valores fuera de este intervalo se toma al valor que está en el límite inferior o superior del intervalo, i.e. valores mayores a 255 son tomados como 255 y valores menores a cero son tomados como cero, dando como resultado islas de densidad máxima y mínima. Se define el *offset* (valor significativo) de la función como G(0,0) Ec. (3.16). Los valores grandes dan como resultado una nube con alta densidad, valores pequeños producen un estrato delgado. Cambiando el valor de la función durante la animación y recalculando la paleta de colores es posible simular el efecto de condensación o disipación en tiempo real.

La varianza es una medida que indica la dispersión de la función con respecto a la media, por lo tanto, da el contraste de la nube. Utilizando la Ec. (3.18) para un espectro de la forma r/f^{β}.

$$AR(g(x)) = \sum_{s=1}^{N-1}(r/f^{\beta})^2 = r^2\sum_{s=1}^{N-1}(1/f^{\beta})^2 \sim r^2 \qquad \text{... E. (3.28)}$$

Por lo tanto, la variación del contraste de la nube se obtiene al variar r.

En el caso 3-D, los "voxels" generados son asignados como textura sólida a un cuerpo. El cuerpo define el espacio que ocupa la turbulencia, mientras que la textura determina la distribución del humo o vapor en el cuerpo.

3.1.3. Modelo de difusión de luz a través de nubes

3.1.3.1. Geometría de nubes usando campos de altura

Las nubes en pequeña escala se forman por un fractal, lo mismo ocurre para las nubes en escalas grandes. Pero en una escala media la forma de las nubes, es causada por ondas periódicas debidas a las perturbaciones atmosféricas. Aquí se modela la distribución a gran escala por medio de una polinomial, y las escalas media y pequeña por series sobrepuestas de ondas senoidales, con diferentes vectores de onda, amplitudes, y fases. En el límite cuando hay suficientes términos incluidos, esas series tienden a ser fractales.

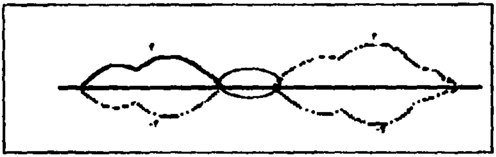


Fig. 3.4
La función f(x,.y) y -f(x,y) tienen orillas redondeadas y son simétricas con respecto al plano de la nube.

Los campos de altura definidos por funciones matemáticas producen algoritmos eficientes de superficies ocultas, ya que las evaluaciones de la función son vectorizables. Una combinación de polinomios, raíces cuadradas, valores absolutos, y términos trigonométricos pueden ser usados para calcular una función *f(x,y)* que defina la altura de las nubes arriba de un plano, y también la profundidad de las nubes abajo del mismo plano. Donde *f* es negativa las nubes no existen. Tal esquema crearía nubes con simetría de espejo tomando como referencia el plano $z = H$, tal como se muestra en la Fig. (3.4). Por lo tanto, *f(x,y)* se modifica para generar dos nuevas funciones *g(x,y)* y *h(x,y)*, las cuales tienen derivada infinita donde $f(x,y) = 0$, así se tiene que ellas se suavizan en el plano de referencia, ver Fig. (3.5) La función *g(x,y)* define la altura de las nubes arriba del plano de referencia y la función *h(x,y)* define la profundidad de las nubes debajo del mismo. Para aplastar los fondos de las nubes, *h(x,y)* tiene un límite finito cuando *f(x,y)* se incrementa.

Las fórmulas para *g* y *h* en términos de *f* son la siguientes:

$t = (\mathrm{abs}(f) + c)^2$

$g = \mathrm{sign}(\sqrt{t - c^2}, f)$ ...Ec. (3.29)

$h = -g / \sqrt{t + c^2}$ ...Ec. (3.30)

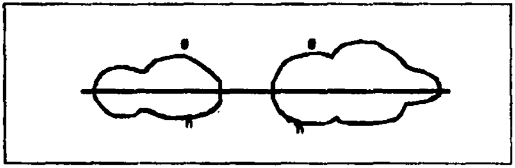


Fig. 3.5
La función g(x,y) y h(x,y) tienen orillas redondeadas y los bordes de abajo aplanados.

Aquí *c* es una constante positiva, independiente de *x* y de *y*; la función sign(*a*,*b*) es el signo de la función $\mathrm{sign}(a,b) = |a|b/|b|$. Efectivamente,

$$g = \sqrt{f^2 + 2cf}$$

así que

$$\frac{dg}{df} = \frac{f + c}{\sqrt{f^2 + 2cf}}$$

la cual tiende a infinito cuando *f* tiende a 0. Similarmente, *dh / df* tiende a infinito cuando *f* tiende a 0. Por lo tanto el plano de referencia intersecta a la nube con una tangente vertical, como se muestra en la Fig. (3.5) El uso del valor absoluto de *f* y el signo de la función asegura que *g* estará definida y será negativa cuando *f* sea negativa. Esto facilita los cálculos de los campos de altura, los cuales se interpolan con los valores de la función calculados en muestras predeterminadas en un plano vertical. Se establece un vector {*y*} con valores de la forma $y_i = H/(\sigma(maxi+1-i))$, así se tiene que la distribución de las alturas del plano $\beta = (H + g(\alpha y_i, y_i))/y_i$ es aproximadamente σ por pixel, para nubes cercanas y también para nubes lejanas.

Para f positiva y h se tiene la forma $-\sqrt{(f^2+2cf)/(f^2+2cf+2c^2)}$, la cual tiende a -1 cuando *f* tiende a infinito, aplanando los fondos de las nubes. La constante *c* afecta la curvatura en el eje.

3.1.3.2. Cálculos de dispersión simple de la luz

Para estos cálculos se asume que el sol da directamente de frente, la densidad es constante, el observador está en el origen (ver 3.6), el plano de la pantalla es perpendicular al eje *y*, y que las intensidades de la nube se calcularán a lo largo de la línea vertical x_s = α. El rayo que pasa a través del pixel (x_s, y_s) = (α, β) tendrá un vector de dirección *V* = (α, 1, β) de longitud $\gamma = \sqrt{1+\alpha^2+\beta^2}$. Un punto *P* en este rayo tiene coordenadas (α*y*, *y*, β*y*).

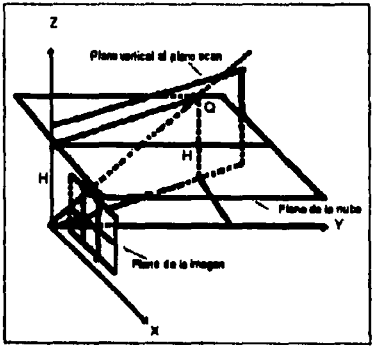


Fig. 3.6
El rayo desde el origen a través del punto P = (α,1, β) en la pantalla y atraviesa el plano de la nube en Q=(x, y, H)

Primero se considera el caso donde este rayo intersecta a la nube en un solo segmento, $\overline{QR}$, donde $Q = (\alpha y_1,\ y_1,\ \beta y_1)$ y $R = (\alpha y_2,\ y_2,\ \beta y_2)$. La figura 3.7 muestra la proyección en el plano yz del plano $x = \alpha y$.

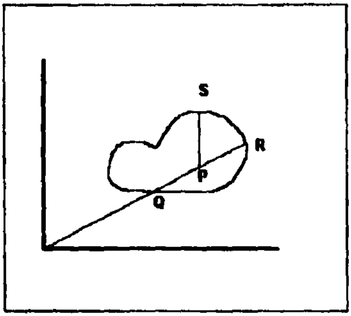


Fig. 3.7
El rayo que parte del observador y que intersecta a la nube en Q y R. La luz que llega del sol atraviesa a la nube a lo largo del segmento SP

Sea $P = (\alpha y,\ y,\ \beta y)$ un punto en el rayo QR. El rayo vertical que va desde el punto P a el sol intersecta la parte superior de la nube en el punto $S = (\alpha y,\ y,\ g(\alpha y, y)+H)$.

La luz del sol que llega al punto S es atenuada por la dispersión a través del rayo SP. Se modela la absorción por la densidad óptica ρ por unidad de longitud. La ensidad total que está en SP es $\rho\overline{SP}$, o $\rho(g(y)^{*} + H - \beta y)$, y la fracción de la luz del sol que le llega a P es $\exp(-\rho(g(y) + H - \beta y))$.

* $g(\alpha y, y) = g(y)$

Dado que *ds* es un elemento de longitud a lo largo del rayo *QR*. Entonces la fracción de energía esparcida por los puntos *P* que están en *ds* hacia el ojo es:

$$\rho\omega\varphi(a)ds\,,$$

donde ω es una albedo, y φ(a) es el factor de fase que depende del ángulo a formado entre *SP* y *PQ*.

Esta luz dispersa es por lo tanto atenuada en el rayo *PQ* por el factor:

$$\exp(-\rho\overline{PQ}) = \exp(-\rho\gamma(y-y_1))$$

Por lo tanto la contribución total de la luz desde *ds* es

$$I_0\exp(-\rho(g(y)+H-\beta y))\rho\omega\varphi(a)\exp(\rho\gamma(y-y_1))ds$$

donde I_0 es la intensidad de la luz del sol incidente en la parte superior de la nube. Para obtener la intensidad de la nube $I(y_1,y_2)$ a lo largo del rayo QR debe integrarse de y_1 a y_2. Así se tiene que reemplazando *ds* por γ*dy*, se obtiene

$$
\begin{aligned}
I(y_1,y_2)&\\
&=\int_{y_1}^{y_2} I_0\exp(-\rho(g(y))+H-\beta y)))\rho\omega\varphi(a)\exp(\upsilon\gamma(y-y_1))\gamma dy\\
&=I_0\rho\omega\varphi(a)\gamma\int_{y_1}^{y_2}\exp(-\rho(g(y)+H-\beta(y-y_1)-\beta y_1+\gamma(y-y_1)))dy\\
&=I_0\rho\omega\varphi(a)\gamma\exp(-\rho(H-\beta y_1))\int_{y_1}^{y_2}\exp(-\rho(\gamma-\beta)(y-y_1)-\rho g(y))dy\\
&=K\int_{y_1}^{y_2}\exp(-\delta(y-y_1))\exp(-\rho g(y))dy
\end{aligned}
$$

donde

$$K = I_0\,\rho\,\omega\,\varphi(a)\,\gamma\exp(-\rho(H-\beta y_1)) \qquad \text{y} \qquad \delta = \rho(\gamma-\beta)$$

Ahora se aproxima la función exponencial por una polinomial, para lo cual se usa la siguiente aproximación

$$\exp(x) \cong s\exp(x) = 0.25(max(x+2.0))^2$$

$$= 1 + x + 0.25x^2 \qquad \text{si } x \geq -2$$

$$= 0 \qquad \text{si } x \leq -2$$

Aplicando esta aproximación a las exponenciales en $I(y_1,y_2)$, se tiene

$$\exp(-\delta(y-y_1)) \cong s\exp(-\delta(y-y_1))$$

$$= 1 - \delta(y-y_1) + 0.25\delta^2(y-y_1)$$

$$= 1 + \delta y_1 + 0.25\delta^2 y_1^2 + (-\delta - 0.58\delta^2 y)y_1 + 0.25\delta^2 y^2$$

$$= a + by + cy^2$$

donde *a*, *b*, y *c* son expresiones de δ y y_1. Similarmente, se tiene

$$\exp(-\rho g(y)) \cong (-s\exp(-\delta g(y))$$

$$= 1 - \delta g(y) + 0.25y^2(g(y))^2$$

$$= 1 + dg(y) + e(g(y))^2$$

Por lo tanto

$$I(y_1,y_2) = K\int_{y_1}^{y_2} s\exp(-\delta(y-y_1))s\exp(-\rho g(y))dy$$

$$= K\int_{y_1}^{y_2}(a+by+cy^2)(1+dg(y)+e(g(y))^2)dy$$

$$= K\left[\int_{y_1}^{y_2}(a+by+cy^2)dy + ad\int_{y_1}^{y_2} g(y)dy + ae\int_{y_1}^{y_2}(g(y))^2 dy\right.$$

$$+bd\int_{y_1}^{y_2} yg(y)dy + be\int_{y_1}^{y_2} y(g(y))^2 dy + cd\int_{y_1}^{y_2} y^2 g(y)dy$$

$$\left.+ce\int_{y_1}^{y_2} y^2(g(y))^2 dy\right]$$

El primero de estos términos puede ser integrado trivialmente. Los otros términos son de la forma

$$m_{i,j} = \int_{y_1}^{y_2} y^i (g(y))^j dy$$

Estas integrales son momentos de la región debajo de la curva $z = g(y)$, entre y_1 y y_2.

Como se interpreta que los valores negativos de g(y) son regiones transparentes donde la densidad es 0, se reemplaza g(*y*) por max(0, g(*y*)). Las cantidades $y^i \max(0, g(y))^j (y_k - y_{k-1})$ pueden ser calculadas con la siguiente integral

$$\int_0^{y_k} y^i \max(g(y,0)^j dy$$

que puede ser aproximada por

$$M_{ij}(k) = \sum_{l=1}^{k} y_l^{\,i} \max(g(y_l),0)^j (y_l - y_{l-1})$$

La integral definida puede ser aproximada sustrayendo $M_{ij}(k1)$ de $M_{ij}(k2)$ donde y_{k1} y y_{k2} son valores de *y* precalculados cercano a y_1 y y_2. Conceptualmente, se tabula el momento acumulado como una integral indefinida, y entonces se determina la integral definida por la sustracción. Cada entrada de la tabla se obtiene agregando un nuevo término a la entrada anterior, de esta forma la tabla se calcula rápidamente. La eficiencia del algorimo se debe a que se reutilizan las mismas integrales indefinidas para cada pixel en una línea vertical.

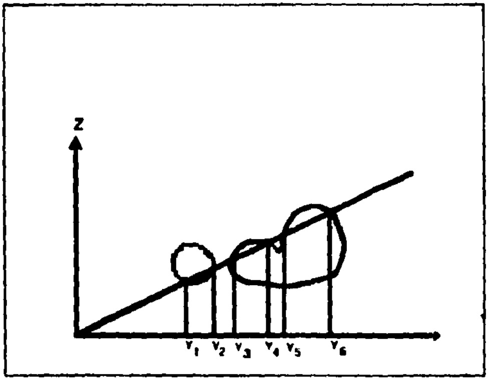


Fig. 3.8
Proyección en el plano YZ de una sección de la nube, con un rayo que la intersecta en seis puntos

Ahora considérese el caso general, donde el rayo intersecta la nube en más de un segmento, proyectando en el eje y varios intervalos, supóngase $[y_1, y_2]$, $[y_3, y_4]$, $[y_5, y_6]$, como se muestra en la Fig. 3.8. Recuérdese que β representa la pendiente dz/dy del rayo. Los intervalos pueden encontrarse usando un algoritmo que calcule al polígono en el plano (β, y) aproximando los contornos de la nube. Como β representa la pendiente dz/dy del rayo, los vértices de este polígono son $(\bar{g}_k, y_k)$ y $(\bar{h}_k, y_k)$, donde $\bar{g}_k = (H + g(\alpha y_k, y_k))/y_k$ y $\bar{h}_k = (H + h(\alpha_k, y_k))/y_k$.

El lado izquierdo del intervalo corresponde al punto final $y_L(\beta)$ y para cada pixel es inicializado con 0 para tomar en cuenta la posibilidad de que el punto de vista este dentro de la nube, un factor de transmisión $T(\beta)$ es inicializado en 1, y una intensidad $J(\beta)$ es inicializado en 0.

Las orillas del polígono para $\bar{g}_k$ y $\bar{h}_k$ son procesadas incrementando k, y la intensidad total $J(\beta)$ y la transmisión $T(\beta)$ se acumulan de adelante para atrás. Cada "cara delantera" de la orilla es usada para actualizar el lado izquierdo $y_L(\beta)$ para los pixeles afectados, y cada "cara trasera" se utiliza para crear el lado derecho del intervalo. Como se conocen ambos lados del intervalo, la integral $K(y_L(\beta), y_R(\beta))$ se aproxima como se mencionó antes, el valor $J(\beta)$ es reemplazado por $J(\beta) + T(\beta)K(y_L(\beta), y_R(\beta))$ y el valor de $T(\beta)$ es reemplazado por $T(\beta)\exp(-\gamma\rho(y_L(\beta) - y_R(\beta)))$.

3.1.3.3. Dispersión de la luz en la niebla

Si hay niebla en el aire debajo de un estrato de nubes, el patrón de luz y la sombra causada por las nubes será visible en la niebla como columnas de rayos, aparentemente convergiendo al sol. Un esquema similar al usado anteriormente puede ser usado para simular este efecto.

Supóngase que la niebla tiene una densidad γ. En la fig. 3.9 se asume por simplicidad que la función $h(x, y)$ es cero, así que las nubes están completamente arriba del plano $z = H$. Se Considera que el rayo EP parte del ojo E, en el origen, con dirección $(\alpha, 1, \beta)$ y se intersecta con el plano de la nube en el punto $P = (\alpha y_0, y_0, H)$. Se tiene que $Q = (\alpha y, y, \beta y)$ es un punto en el rayo, y $R = (\alpha y, y, H)$ es un punto que está en el plano de la nube justamente arriba de Q.

La cantidad de luz que pasa a través de la nube en el punto R es $I_0 \exp(-\rho g(y))$. La luz que se esparce en el rayo sobre el punto Q es $\gamma \omega \varphi(a) \gamma dy$. La longitud de la trayectoria formada por los segmentos $\overline{RQ}$ y $\overline{QE}$ es $\overline{RQ} + \overline{QE} = H - \beta y - \gamma y = H + (\gamma - \beta) y$, y la absorción de la niebla a través de esta trayectoria multiplica a la intensidad por un factor $\exp(-r(H + (\gamma - \beta)\gamma))$. Por lo tanto, la intensidad total que se esparce en la niebla es

$$I_H = \int_0^{y_0} I_0 \exp(-\rho g(y)) r \omega \varphi(a) \gamma \exp(-r(H + (\gamma - \beta)\gamma)) dy$$

$$= I_0 \exp(-rH) r \omega \varphi(a) \gamma \int_0^{y_0} \exp(-\rho g(y) - \varepsilon y) dy$$

Esta integral puede ser calculada como se mencionó en la sección anterior.

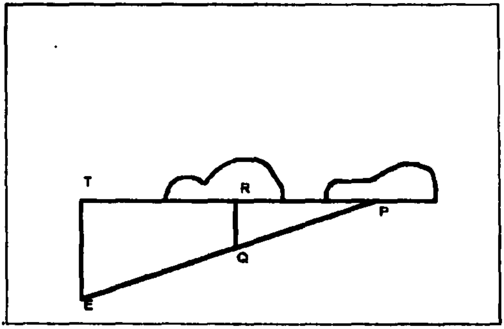


Fig. 3.9
Rayo que parte del observador E a una nube en el punto P con rayos verticales TE y RQ

3.2 Algoritmos para la simulación visual de todo tipo de fenómeno gaseoso

3.2.1. Modelo basado en "render" de volúmenes y "scanline a-buffer"

En esta sección se describe una técnica que combina las técnicas de render de volúmenes y el "scanline a buffer". Esta es ideal para realizar el render de escenas que contengan fenómenos gaseosos tales como nubes, niebla y vapor.

La geometría de los gases se obtiene usando texturas sólidas y se animan utilizando flujo turbulento. Se presenta un modelo de iluminación que considera el "self-shadow" de los volúmenes.

3.2.1.1. El render

El render *descrito* en este trabajo combina el "render" "scanline a-buffer" (almacenamiento de información para cada línea que pertenece a la imagen) y el *render* de volúmenes. El primero de ellos se usa para la superficies definidas por los objetos en la escena, y el segundo para los volúmenes de dichos objetos.

El algoritmo primero crea un "a-buffer" *(anti-aliased, área promedio, buffer acumulador)* para una línea de la imagen que contiene una lista para cada pixel y los fragmentos que cubren parcial o totalmente a ese pixel.

-

La estructura del fragmento usada es la siguiente

Estructura del fragmento
Valores máximos y mínimos de z
porcentaje que cubre
vector normal
apuntador al objeto padre
mascar de bits para la geometría
color
valores de atenuación para cada luz

Si el volumen está activo para un pixel, se realiza el *"render"* de volúmenes de la forma siguiente. Primero, se calcula el rayo que va desde el observador al pixel. Después cada fragmento que está en el "a-buffer" se mapea al espacio tridimensional. Con la geometría del volumen, la posición de esos fragmentos en el espacio tridimensional, los planos que delimitan la escena, permiten calcular el "render" del volumen. El punto inicial para el volumen al cual se le va calcular el "render" es el valor máximo entre del plano más cercano al observador y el punto más cercano de la intersección del rayo con el volumen. El punto final es determinado por las intersecciones del rayo con el volumen, el plano que está atrás y delimita la escena, y los fragmentos del "a-buffer" para este pixel.

Cada fragmento en la lista de fragmentos también determina un punto inicial o final para los elementos del volumen que están separados. Para obtener efectos correctos, el volumen al cual se le va ser el "render" se debe partir en secciones que están entre, en frente de y detrás de los fragmentos "a-buffer".(Ver Fig 3.10). Por ejemplo, si hay un objeto transparente cubriendo un pixel que contiene humo, se debe crear un fragmento en el "a-buffer" para el humo que está en frente, dentro (si se desea), y detrás del objeto transparente. Esto es necesario, ya que si solo se creará un fragmento, el fragmento sería

ordenado en frente, dentro, o detrás del objeto transparente de acuerdo a su valor promedio, trayendo como consecuencia un valor incorrecto para el color de ese pixel.

Para cada uno de los fragmentos que están en la lista se realizan los cálculos necesarios para determinar el sombreado de los volúmenes.

Después de realizar el "render" de volúmenes para cada pixel, se toma en cuenta la geometría de la mascara de bits para determinar cuales fragmentos son visibles.

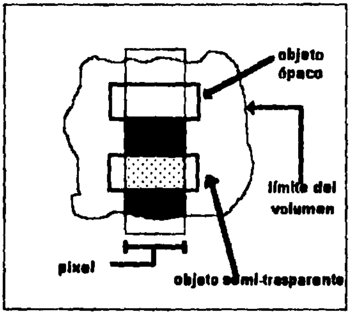


Fig. 3.10
Las áreas sombreadas forman parte del volumen

Render de volúmenes

Como se mencionó arriba, solamente la porción visible del volumen se le calculará el "render". Si el volumen está completamente cubierto por una pared, por ejemplo, no se realizará el "render" del volumen. El rayo que se envía desde el observador al pixel se traza tomando en cuenta la geometría del volumen. Como se describió antes, se realiza el seguimiento del volumen hasta que se ha logrado por completo la cobertura del pixel. Si hay alguna cobertura parcial del pixel por algunos fragmentos, el volumen se parte en secciones en frente y detrás de los fragmentos hasta que se logre la cobertura total del pixel. Es necesario realizar esto porque cada una de las secciones se convierte en un nuevo elemento en la lista de fragmentos "a-buffer". Para cada paso que se va dando a través del volumen, se va evaluando la función de densidad. Se utilizan algoritmos para el "render" ligeramente diferentes si las funciones de densidad de volumen representan a objetos sólidos, como son la funciones de hipertextura, que si las funciones de densidad de volumen representan a un gas. Los dos algoritmos difieren en los cálculos de la iluminación y acumulación de densidades.

El algoritmo básico para realizar el "render" de los volúmenes sólidos es el siguiente:

```
Determinar 2 direcciones mutuamente ortogonales a el rayo
Para cada sección del volumen
  Obtener el color, densidad, y opacidad
  Obtener la densidad en las dos direcciones mutuamente ortogonales
  Determinar la normal a la superficie basada en la densidad previa,
   la densidad actual, la densidad en la dirección 1, la densidad en la dirección 2
  Si hay sombreado_propio
        Para cada fuente de luz
          Trazar el rayo a la luz obteniendo el factor de atenuación de la luz
          color = calcular la iluminación de este volumen
          usando la normal y el color del objeto
          t1 = opacidad*(1-sum_opacidad);
          color_final = color_final + t1*color;
          sum_opacidad = sum_opacidad + t1;
          Si sum_opacidad = 1
                Se detiene el seguimiento que se está haciendo
          Incrementar muestra_pt
          densidad_previa = densidad
  Crear el fragmento en el "a-buffer"
```

La opacidad se determina a partir de la densidad usando la siguiente formula:

$$opacidad = 1 - (1 - densidad)^{c \cdot tama\tilde{n}o\_paso}$$

donde *c* es una constante normalizada.

El algoritmo para realizar el "render" en volúmenes gaseosos es el siguiente:

```
Para cada sección del gas
  Para cada incremento a través del rayo
        Obtener color, densidad, y opacidad
        Si hay sombreado_propio
          Para cada fuente de luz
                Trazar el rayo a partir de la luz y
                calcular el factor de atenuación
          color = calcular la iluminación del gas
                 usando la opacidad, densidad y el
                 modelo apropiado
          color_final = color_final + color;
          sum_densidad = sum_densidad + densidad;
          Si (transparencia = 0.01)
            Terminar
          Incrementar muestra_pt
  Crear el fragmento en el "a-buffer"
```

En este caso, la opacidad a través del rayo se calcula aproximando la siguiente integral:

$$opacidad = 1 - e^{-r\tau \int_{t_{cerca}}^{t_{lejos}} \rho(x(t),y(t),z(t))dt}$$ donde *r*

es la profundidad óptica del material, $\rho()$ es la densidad del material, t^{cerca} es el punto inicial del volumen, y t^{lejos} es el punto final.

La integral es aproximada de la siguiente manera:

$$opacidad = 1 - e^{-r\tau \sum_{t_{cerca}}^{t_{lejos}} \rho(x(t),y(t),z(t)) \times \Delta t}$$

Modelo de iluminación para los fenómenos gaseosos

Se ha implementado un modelo de iluminación de bajo-albedo. El modelo de iluminación es el siguiente

$$B = \sum_{t_{near}}^{t_{far}} e^{-\tau \sum_{t_{near}}^{t_{far}} \rho(x(u),y(u),z(u)) \times \Delta_u} \times I \times \rho(x(t), y(t), z(t)) \times \Delta_t$$

donde *I* es:

$$\sum_{i} I_i(x(t), y(t), z(t)) \times phase(\theta)$$

Phase(θ) es la función de fase, la función caracteriza el total de brillo de una partícula y está en función del ángulo entre la luz y el ojo. $I_i(x(t),y(t),z(t))$ es la cantidad de luz reflejada y que parte de la fuente de luz I. El sombreado propio del gas se incorpora en este termino, atenuando el brillo de la luz.

Sombreado del gas

El camino más simple para sombrear el gas es seguir un rayo desde cada volumen a la luz, determinando la opacidad del material a lo largo del rayo usando la ecuación para opacidad descrita arriba. Este método es similar a los cálculos de sombreado desarrollados en el método de "ray tracing" y puede ser muy lento. Dependiendo de la composición de la escena (cantidad de gas en la escena), experimentos que se han realizado muestran que "self-shadowing" llevado a cabo de esta manera pueden tomar de un 75-95% del tiempo total de cómputo.

Kajiya habla de la importancia del "self-shadowing" para corregir la visualización de datos. Sin embargo, muestra que modelos con bajo albedo para gases que tienen albedos mayores que 30% producen un alto "self-shadowing". Si el gas al que se le realizará el "render" tiene un alto albedo, los efectos de "self-shadowing" serán despreciables comparados con el esparcimiento de la luz. Por lo tanto, sino se calcula el "self-shadowing" para gases con alto albedo se pueden obtener resultados realistas., sin tener altos costos computacionales. Para gases con brechas de baja intensidad, el sombreado puede ser acelerado sino se calcula para elementos donde la densidad es menor a un valor inicial.

Para incrementar la velocidad de los cálculos una tabla precalculada puede ser usada. El uso de una tabla precalculada es definitivamente más rápido para gases que no se mueven de un cuadro a otro. Sin embargo, aunque el gas se mueva de un cuadro al siguiente, el "render" en los volúmenes se realizará más rápido.

Cálculo de la tabla de sombras

Para calcular la tabla de sombras primero hay que calcular una tabla de las mismas dimensiones que contenga valores funcionales. Esto se hace para evitar repetidas evaluaciones de la función de densidad. Lo siguiente, es calcular la distancia cuadrada de cada punto en la tabla a la luz. Despúes se ordenan estas distancias, de acuerdo al orden en que deben ser calculados los valores de la tabla de sombras. Para calcular los valores de la tabla de sombras se empieza con los puntos más cercanos a la luz y se prosigue con los más lejanos, solamente se necesita una interpolación bilineal para determinar cada valor. Para determinar el valor que le corresponde a una entrada de la tabla, se calcula el rayo que va desde ese punto a la luz (Ver fig. 3.11). El rayo que parte del punto p_{ijk} a luz pasará a través de una de las caras del paralelepipedo formado por las entradas de la tabla (i+1, j+1, k+1), (i+1, j-1, k+1), (i+1, j-1, k-1), (i-1, j+1, k+1), (i-1, j-1, k+1), (i-1, j-1, k-1). Ahora usando la distancia que existe entre los puntos de la tabla y el vector normalizado a la luz, se puede determinar la cara del cubo que rodea este elemento de la tabla y que será intersectada por el rayo trazado a la luz puede ser determinada. Una vez que es determinada la cara, el punto de intersección entre el rayo y el plano al que pertenece ésta, puede ser determinado rápidamente, puesto que la tabla está alineada a los ejes del espacio de objetos. Este punto de intersección cae entre las 4 entradas de la tabla que contienen la información de la sombra para esos cuatro puntos puesto que las entradas son calculadas en orden. Agregando los valores funcionales de las entradas regula el tamaño del paso a sus valores de la tabla de sombras e interpolando bilinealmente esas sumas, el valor de la sombra para ese elemento puede ser encontrado.

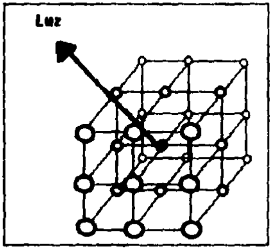


Fig. 3.11
Tabla de sombras calculadas

Para utilizar el valor de la tabla cuando se está realizando el "render" de un volumen, se tiene que localizar el punto del volumen al cual se le está haciendo el "render" dentro de la tabla de sombras. Este punto estará dentro de un paralelepipedo formado por ocho entradas de la tabla. Estas 8 entradas son interpoladas tri-linealmente para obtener la suma de las densidades entre este punto y la luz. Para determinar la cantidad de atenuación para la luz, se usa la siguiente fórmula.

$$luz\_atenuadada = 1 - e^{-\tau \times suma\_densidades \times tam\_paso}$$

Este método es más rápido que trazar una rayo a cada luz si las múltiples evaluaciones en la función de densidad son más lentas que una interpolación tri-lineal más una fracción de tiempo necesitado para crear la tabla.

3.2.2. Modelo basado en Espacios Sólidos

En esta sección se describirá un método para la simulación visual de fenómenos gaseosos usando espacios sólidos.

Para modelar dichos fenómenos se usan espacios sólidos basados en funciones de densidad de volumen, y la animación de los mismos se realiza usando funciones de flujo turbulento dentro del espacio sólido.

3.2.2.1 Definición de Espacios Sólidos

Los espacios sólidos son espacios tridimensionales asociados con un objeto para tener control sobre un atributo de un objeto. Ejemplo de estos son: las texturas sólidas, espacio del "noise" y el espacio de "turbulence" todos ellos descritos en el capitulo 2.

3.2.2.2. Modelado de los gases

La geometría de los gases es modelada usando flujo turbulento basado en funciones de densidad del volumen. Las funciones de densidad del volumen toman un punto del espacio del objeto, encuentran su posición correspondiente dentro del espacio de turbulencia (espacio tridimensional), es decir se evalúa el punto que está en el espacio del objeto en la función "turbulence". El valor regresado por está función se usa como base para la densidad del gas y este valor aplicado en funciones matemáticas nos da la forma del gas. Las funciones matemáticas que se usan más comúnmente son: función potencia, función seno y función exponencial.

La forma básica del gas se logra elevando a una potencia el valor que se obtiene al evaluar el punto en la función "turbulence", es decir utilizando la función potencia.

$$\text{Densidad} = (\text{turbulence}(\text{punto}))^{\text{exponente}}$$

Otra forma de modelar los gases es utilizando texturas sólidas con transparencia. En este caso, cada punto del objeto se mapea al espacio de turbulencia, y este valor se utiliza como base para calcular la transparencia correspondiente a ese punto del objeto.

3.2.2.3. Animación de los gases

Para realizar la animación de los gases se tienen dos opciones:

1.- Cambiar la definición del Espacio Sólido a través del tiempo.

2.- Mover el punto dentro del espacio sólido.

La primera opción toma al tiempo como parámetro y va cambiando la definición del Espacio Sólido conforme el tiempo transcurre. Esta es una forma muy natural y obvia de animar.

La segunda opción mueve el punto en el volumen u objeto sin cambiar la definición del Espacio Sólido. El movimiento del gas se crea moviendo el punto a lo largo de una trayectoria dentro un espacio del gas (Espacio Sólido). Cada punto de la pantalla se mapea al espacio 3-dimensional. Después este punto es mapeado al espacio del gas. Finalmente, se mueve dentro del último espacio.

El control de la animación se puede realizar a través de trayectorias primitivas o bien auxiliándose de tablas tridimensionales que permiten usar trayectorias más complejas, dichas tablas serán descritas en la sección siguiente.

Tablas tridimensionales generales

Hay dos tipos de tablas tridimensionales para controlar la animación de los gases: Tablas de campos vectoriales y Tablas de campos con flujo funcional. Cada tabla rodea el volumen del gas en el espacio tridimensional (Ver Fig. 3.12). Los valores son almacenados para cada punto que pertenece a la malla. Los valores en estos puntos son los cálculos realizados en esa región del gas. Para determinar los valores de las otras localizaciones en el gas, las ocho entradas de la tabla que forman el paralelepipedo que rodea el punto son interpolados.

Las tablas son incorporadas a las funciones de densidad de volumen para controlar la forma y el movimiento del gas.

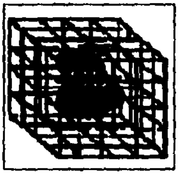


Fig. 3.12
Tabla tridimensional alrededor de un objeto

Accesando las entradas de las tablas

Para accesar los valores de las tablas al momento de realizar el *"render"*, la localización del punto dentro de la tabla se determina. Este punto estará dentro de un paralelepipedo formado por las ocho entradas de la tabla que rodea al punto. La localización dentro de la tabla se determina primero mapeando el punto de la pantalla al espacio del objeto. La siguiente información se calcula para cada cuadro de la animación: la localización de la entrada inicial de la tabla en el espacio del objeto (*inicio_tabla*), el tamaño entre las entradas de la tabla (*paso_tabla*), y el inverso de este tamaño (*inv_paso_tabla*). La siguiente formula se utiliza para encontrar la localización del punto dentro de la tabla:

*ptabla.x = (punto.x - inicio_tabla.x) * inv_paso_tabla.x*

*ptabla.y = (punto.y - inicio_tabla.y) * inv_paso_tabla.y*

*ptabla.z = (punto.z - inicio_tabla.z) * inv_paso_tabla.z* ...

Ptabla es la localización del punto dentro de la tabla tridimensional, que fue calculada a partir del punto tridimensional.

Tablas con campos vectoriales

Las tablas de campos vectoriales permiten generan campos tridimensionales que controlen el movimiento del gas. Contienen la siguiente información: vector de dirección, valor de escalamiento en la densidad, un vector de porcentaje de uso.

El vector de dirección define la trayectoria para el movimiento del gas. El factor de escalamiento en la densidad permite que la densidad del gas se decremente. El "vector de porcentaje de uso" especifica cuanto se va a usar de la dirección que está en la tabla y cuanto se va a usar de la trayectoria por default que tiene el gas, lo que permite una transición suave entre la trayectoria definida por los valores de la tabla y la trayectoria por default del gas. Este valor "porcentaje de uso" es un factor de ponderación para combinar el vector de la trayectoria por default y el vector de la tabla. Estos vectores de trayectorias se combinan como se describe a continuación:

trayectoria_final = *pocentaje_de_uso* x *vector_dirección_tabla_vectores*
+ (1 - *porcentaje_de_uso*) x *dirección_trayectoria_default*

Las tablas de campos vectoriales pueden generar un amplio rango de resultados para mover los gases en visualizaciones científicas. Estas tablas, por tanto, proveen una herramienta flexible para visualización científica.

Tablas con campos de flujo funcional

El otro tipo de tablas usadas para controlar el movimiento de los gases son las tablas con campos de flujo funcional. Estas tablas definen para cada región del gas cual función se evaluará para controlar su movimiento. Cada entrada de las tablas con campos de flujo funcional puede contener sólo una función, o bien un lista de funciones para determinar la trayectoria del movimiento del gas. Las funciones evaluadas por las tablas de campos de flujo funcional regresan la siguiente información vector de dirección, valor de escalamiento para la densidad, vector de porcentaje de uso, y velocidad.

Como se puede ver, estas funciones regresan la misma información que las tablas de campos vectoriales, con la adición del factor de escalamiento de la velocidad. La ventaja de las funciones de flujo funcional sobre las tabla con campos vectoriales es que nos dan un detalle muy fino del movimiento del gas; ya que son evaluadas para cada punto del volumen al que se le va a ser el "render", no son almacenadas para un resolución fija. La desventaja de las tablas con campos de flujo funcional es que evaluar las funciones es más costoso que realizar una simple interpolación lineal entre los valores de la tabla.

El "vector de porcentaje de uso" se utiliza otra vez para tener una transición más suave entre la trayectoria por default del gas y la trayectoria de las funciones de campos de flujo.

Funciones de campos de flujo

1. Trayectorias helicoidales

Este tipo de trayectorias pueden ser usadas para crear diferentes efectos de animación para los gases. Por ejemplo una trayectoria helicoidal que va hacia abajo produce el efecto del gas saliendo. Una trayectoria helicoidal horizontal puede crear el movimiento del gas.

2. Trayectorias de atractores y repulsores

Los atractores y repulsores son funciones primitivas que dan un amplio rango de efectos. Cada atractor tiene un valor máximo y mínimo de atracción. En la animación con la localización y la fuerza del atractor se pueden llevar a cabo diferentes efectos. Efectos tales como una brisa que está soplando. Los atractores esféricos simplemente crean trayectorias que se acerca hacia el centro de atracción y los repulsores esféricos crean trayectorias que se alejan del centro de repulsión.

Otros atractores que se pueden obtener al variar un atractor esférico son los siguientes: atractores de movimiento, atractores con ángulo limitado, atractores con una variable de máxima atracción, atractores no esféricos, y, por supuesto, combinaciones de los anteriores.

3. Funciones de vórtices espirales

Las funciones de vórtices pueden tener una variedad de usos tales como la simulación de vórtices físicos para crear perturbaciones en el flujo como una aproximación del flujo turbulento. La función vórtice está basada en una simple función en coordenadas polares bidimensionales:

$$r = \theta$$

la cual traslada a coordenadas tridimensionales como

$$x = \theta \times \cos(\theta)$$

$$y = \theta \times \operatorname{sen}(\theta)$$

La tercera dimensión en normalmente solo un movimiento lineal en el tiempo a lo largo del tercer eje. Para animar esta función, θ es relativo al número de cuadro. Para incrementar la acción del vórtice, un multiplicador escalar para los términos coseno y seno basado en la distancia del vórtice al eje se añade.

4. IMPLANTACIÓN Y RESULTADOS

Introducción

Para incrementar el realismo en las imágenes generadas por computadora, es importante considerar los fenómenos gaseosos. El problema es que las técnicas tradicionales para modelar (por medio de polígonos) y animación no pueden ser usadas en estos objetos.

Dada la importancia de los fenómenos gaseosos, es necesario desarrollar métodos en donde se logre la simulación visual de estos.

Para lograr la simulación visual de los fenómenos gaseosos, en graficación por computadora se deben considerar tres aspectos: modelado (forma del gas), *"render"* (generación de la imagen del gas tomando en cuenta el modelo de iluminación, modelo geométrico, las cámaras de visión, etc.), y la animación (moviento al gas).

El método implantado demuestra como modelar y animar tales fenómenos

4.1. Método

4.1.1. El *"render"*

Para realizar el proceso de "render" se decidió utilizar una biblioteca llamada SIPP, ya que por las propiedades que tiene permite generar imágenes con gran calidad de despliegue.

Esta biblioteca permite crear escenas tridimensionales y realizar el *"render"* de éstas, por medio de un algoritmo "scan-line z-buffer". Una escena se construye con objetos a los cuales se les pueden aplicar transformaciones como rotación, traslación y escalamiento.

La bibilioteca también nos da la opción de manejar multiples fuentes de luz, que pueden ser direccionales, puntuales, o con un ángulo de proyección. Además hace el manejo de la cámara de visión.

La principal ventaja de SIPP es el uso de "shaders", procedimientos en los que se puede especificar por medio de algunas funciones matématicas, y funciones de textura ("noise", "turbulence") el sombreado y textura que tendrá el objeto.

4.1.2. Modelado

Para modelar los gases éste trabajo se utilizó el algortimo espacios sólidos [2], el cual utiliza flujo turbulento basado en funciones de densidad del volumen para obtener la forma del gas. Las funciones de densidad del volumen toman un punto del espacio del objeto, encuentran su posición correspondiente dentro del espacio de turbulencia (espacio tridimensional), esto es evalúa el punto del espacio tridimensional en la función "turbulence". El valor regresado se usa como base para la densidad del gas y este valor aplicado en funciones matemáticas nos da la forma del gas. Las funciones matemáticas que se usan más comúnmente son: función potencia, función seno y función exponencial.

Para lograr tanto el modelado como la animación se escribieron "shaders" en donde se especifica las características generales que tendrá el gas.

Forma básica del gas

En la primera aproximación que se hace del gas se usa la función potencia, tal como se muestra en el siguiente procedimiento:

```
static void
gas(p, color, gd)
      Vector *p;
      Color *color;
      Gas1_desc *gd;
{
         double x, t;
         float x, t;
         int i;
         float density;
         float tmp;
         Vector direc;

         tmp = noise(p);
         turb = turbulence(&tmp, 2);        /* se está mapeando al espacio de
                                               turbulencia */

         if (turb < 0) turb = turb * -1;
         i= (int) (turb*(.5*(POW_TABLE_SIZE-1)));
         density = gd-> pow_table[i]*120;    /* se encuentra su posición dentro
                                                de la tabla de densidades generada */

         if (density > 1.0) density = 1;   /*  como la densidad del gas se maneja
                                               como transparencia se cuida que el rango
                                               tenga como limite máximo 1 */

         color->red = density;
         color->grn = density;
         color->blu = density;
}
```

El procedimiento recibe como parámetros a *p*, vector de textura correspondiente al punto del objeto, *color*, donde se coloca la transparencia en el punto *p*, *gd* (gas descripción), es la descripción del shader del objeto, en el cual se especifica : el "*shader*" utilizado, la superficie, la escala que se le aplicará a la textura, la luz ambiental, la especularidad del objeto.

El arreglo *pow_table* , se calcula una sola vez dentro del programa principal, con la finalidad de acelerar el proceso de cómputo. En éste almacenan las densidades que podrá tener el gas, para su cálculo se utiliza la función potencia, y el arreglo *parms*, *parms[0]* se considera la densida máxima que alcanzará el gas, y *parms[1]* es el exponente que se utiliza en la función. A continuación se muestra su implantación:

```
for (i= POW_TABLE_SIZE -1; i>= 0; i--)
        pow_table[i] = (float) pow((double)(i) / (POW_TABLE_SIZE-1) *
                                parms[0]*2.0, (double)parms[1])
```

Apariencia de nubes

Otra forma de modelar fenómenos gaseosos se logró mapeando el punto p al espacio de turbulencia (aplicar la función turbulence), y aplicando al valor obtenido la función seno. Al aplicar la función seno obtenemos la apariencia de una nube, como se muestra en la fig. 4.1.

```
static void
        gas1(p, color, gd)
                Vector *p;
                Color *color;
                Gas_desc *gd;
{
        double x, y t;

        x = p->x + turbulence(p,15) *5 ;    / se está mapeando al espacio de
        x = sin (x);
        if (x < 0.0) {
           color->red = 0.0;
           color->grn = 0.0;
```

```
            color->blu = 0.0;
        }
        else
        {
            color->red = x;
            color->grn = x;
            color->blu = x;
        }
}
```

Fig. 4.1
Nubes usando las funciones turbulencia y seno.

Al igual que en el procedimiento anterior, en este procedimiento se está calculando la transparencia en cada punto del gas. Los parámetros son p, punto del gas, color, tranparencia en ese punto, gd, descripción del gas.

4.1.2. Animación

Para animar los espacios sólidos existen dos formas:

Cambiando la definición del espacio sólido en el tiempo.

Moviendo el punto al cual se le va realizar el "render" dentro del espacio sólido.

La primera de ellas tiene cómo parámetro al tiempo, ésta aproximación es la técnica tradicional usada en graficación por computadora.

En la segunda de ellas el movimiento es creado moviendo cada punto del objeto a lo largo de una trayectoria definida dentro del espacio de turbulencia. Esto es se mapea cada punto al espacio de turbulencia, se le aplican las funciones y parámetros que ayudan a crear la trayectoria, y después se evalúa en la función *"turbulence"*. Por lo tanto, la dirección de la trayectoria tendrá un efecto visual invertido. Por ejemplo, al aplicar a un objeto una trayectoria helicoidal hacia abajo, el efecto visual que se percibirá será un movimiento arremolinado hacia arriba.

Texturas sólidas con transparencia

En este trabajo se utilizaron texturas sólidas con transparencia para simular el aspecto de niebla. En este caso al igual que en la simulación que utiliza funciones basadas en densidad de volumen, se mapea primero al espacio de turbulencia y después se obtiene la transparencia correspondiente al punto que se está evaluando.

Para animar esta imagen se utizó la segunda opción de animación, para lo cual se creó una trayectoria helicoidal dentro del espacio de turbulencia. En el siguiente procedimiento se muestra como se hizó la implantación de lo arriba mencionado:

```
static void
gas3(p, color, gd)
      Vector *p;
       Color *color;
       Gas_desc *gd;
{
        double x, opac;
        Vector punto, direc;

        punto.x = p->x  + 2*turbulence(p,10);  /* se está mapeando al espacio */
        x = noise(p);                           /* de turbulencia */
        punto.y = p->y + 4 + x;
        punto.z = p->< -2 - x;

        /* se genera la trayectoria que tendrá el fenómeno gaseoso dentro del
           espacio de turbulencia, tomando en cuenta el número de frame que
           se está generando */
        direc.x = punto.x + gd->cy.x;
        direc.y = punto.y + gd->num_frame*DOWN_AMOUNT;
        direc.z = punto.z + gd->cy1.z;

        opac = turbulence(&direc, 12);
        opac =  (1-0-(opac)*(opac)*.275);
        opac = opac*opac*opac;

        if (opac < 0.0) {
                color->red = 0.0;
                color->grn = 0.0;
                color->blu = 0.0;
        } else {
                color->red = opac;
                color->grn = opac;
                color->blu = opac;
        }
}
```

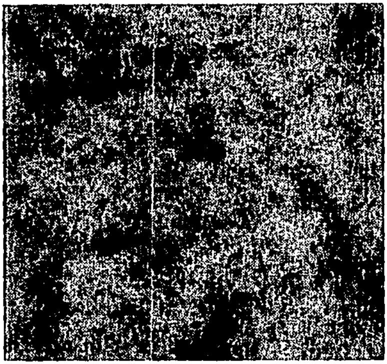


Fig 4.2

Niebla simulada por medio de texturas sólidas con transperencia

Con el procedimiento anterior se logra mover la niebla con un movimiento "arremolinado" hacia arriba., ver fig. 4.2.

Humo saliendo de una tetera

La escena que fue creada contempla a una tetera puesta sobre un piso y el humo que sale del pico de ésta. Para simular el efecto del humo se creó un superficie justo arriba del pico de la tetera, y a ésta se le aplicó el procedimiento para obtener la forma básica del gas, ver fig. 4.3.

Para animarlo se utilizó una trayectoria helicoidal hacia abajo para dar el efecto de humo "arremolinado" hacia arriba.

Para lograr lo anterior se agregó al código lo siguiente:

```
tmp = noise(p);      /* con la finalidad de agregar mayor aleotoridad a la textura */
direc.x = p.x + cos_theta2+tmp;   /* se está creando la trayectoría */
direc.y = p.y + p.y - down + tmp;
direc.z = p.z + sen_theta2 + tmp;

opac = turbulence(direc, 10);
```

Las variables down, cos_theta2, y seno_theta2 son calculadas una vez para cada cuadro de la animación dentro del programa principal, como se muestra a continuación:

```
theta = (num_frame%SWIRL_FRAMES)*SWIRL_AMOUNT
down = (float)num_frame*DOWN*3.0+4.0;
cos_theta = RAD1 * cos(theta) + 2.0;
sen_theta = RAD2*sen(thet) -2.0;
```

Donde RAD1 y RAD2 son el radio el radio en y y z respectivamente de la trayectoría helicoidal, mientras que SWIRL_FRAMES y SWIRL_AMOUNT nos ayudan a crear el efecto de un gas que se mueve alrededor de 360 grados cada SWIRL_FRAMES cuadros.

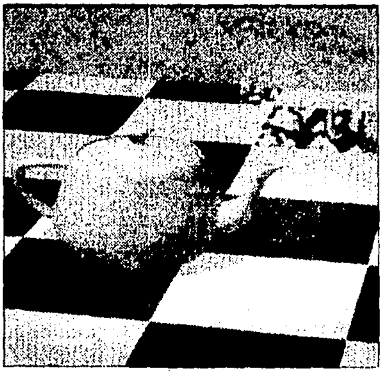


Fig. 4.3
Humo saliendo de la tetera usando texturas sólidas con transparencia

CONCLUSIONES

En el presente trabajo se realizó:

- La presentación de una clasificación de los modelos que se han desarrollado para modelar fenómenos naturales, así como los principios y las herramientas básicas para mejorar la eficiencia de éstos. Ésta presentación fue necesaria ya que ante todo hay que considerar que los fenómenos gaseosos son fenómenos naturales.

- La descripción de texturas por procedimientos, ya que estas son básicas en el algoritmo que se seleccionó para la implantación.

- La explicación de varios algoritmos para la simulación de fenómenos gaseosos. Los que pueden ser dividos en algoritmos enfocados a la simulación de nubes y algortimos para simular todo tipo de fenómeno gaseoso.

- La implantación de un algoritmo para la simulación de fenómenos gaseosos. Para ésta se tomó como base el algoritmo que utiliza espacios sólidos (en particular funciones de densidad de volúmenes y texturas sólidas con transparencia). Esta decisión se tomó por tres razones, la primera de ellas es que este algoritmo está dentro de una de mis áreas de interés dentro de la graficación por computadora, *el modelado basado en funciones*, la segunda es que los espacios sólidos son una extensión de las texturas tridimensionales (texturas sólidas), de las cuales ya he desarrollado algunos trabajos, y la última de ellas es que el algoritmo permite simular todo tipo de fenómeno gaseoso.

Con los tres primeros puntos se logró dar un sustento teórico a la implantación realizada. Mientras que con la implantación se logró realizar una simulación visual de fenómenos gaseosos, .generando imágenes de escenas que contienen dichos fenómenos, y realizando la animación de las mismas.

Las imágenes generadas tiene gran calidad de despliegue, no presentan próblemas de "aliasing" (.i.e. no se perciben frecuencias, orillas o patrones que no formen parte de la escena), y tienen un alto grado de realismo, también es importante mencionar que las animaciones se presentan en tiempo real. Con esta implantación se puede simular una cantidad enorme de fenómenos gaseosos, y se generan imágenes realistas que pueden utilizarse en diferentes áreas de aplicación, tales como: meteorología, entretenimiento, arte, simulación de vuelos, fenómenos físicos, e ingeniería.

Por lo tanto se cumple el objetivo planteado al inició del mismo

Presentar e implementar un algoritmo que permita modelar y animar fenómenos gaseosos.

Del trabajo presentado se puede ver que la implantación realizada nos da una simulación adecuada para fenómenos gaseosos, y además nos da pauta para que se inicien nuevos trabajos de investigación tales como:

- Extender los espacios sólidos para modelar líquidos.
- Agregar al modelo implantado variables físicas tales como: la temperatura y el viento, y simular el comportamiento que tendrían.
- Generar escenas en donde se tengan fenómenos gaseosos y otro tipo de fenómenos naturales, por ejemplo fuego y el humo que éste produce, un lago en una mañana fría y el vapor que se genera.
- Modelar gases con un nivel de detalle más fino, para lo cual se propone hacer una reconstrucción del gas por medio de "voxels", antes de realizar el "render".

GLOSARIO

Agoritmo Scan line

Una técnica para realizar el "render" de una imagen líneas por línea. El algoritmo reduce el problema de tridimensional a un problema bidimensional para cada línea que se está examinando.

Aliasing

Introducción de frecuencias, orillas, o patrones que no forman parte de la excena. Esto resulta al tratar de reproducir señales de alta frecuencia (orillas, detalles de texturas) con un número insuficiente de muestras (pixels).

Anti-aliasing

El proceso de reducir, quitar, o evitar los artefactos visuales tales como orillas, patrones o detalles pérdidos de la imagen.

Bump mapping

El proceso de crear y desplegar una textura donde los valores de la textura son usados para modificar la normal a la superficie del objeto. El resultado es un aparente realzado a la superficie generado únicamente por medio del algoritmo de sombreado. el cual calcula los colores con base a la dirección normal a la superficie.

Constructive Solid Geometry (CSG)

Un modelo basado en un conjunto de formas geométricas que pueden ser transformadas y combinadas por medio de operaciones booleanas. El modelo resultante es un árbol cuyas primitivas son las hojas, las operaciones booleanas son los nodos que no son hojas, y las orillas son las transformaciones.

Difuso

La luz que se refleja igualmente en todas las direcciones desde un punto en la superficie de un objeto.

Digitización

El proceso de construir un modelo geométrico a partir de dibujos, objetos físicos, o modelos de objetos. Las coordenadas tridimensionales se obtienen de la fuente.

Especular

El componente de iluminación visto en una punto de la superficie de un objeto el cual se produce por la reflección en la superficie normal. Esto depende de la posición del observador.

Fractal

Una estructua geométrica que tiene la propiedad que su frecuencia sea la misma a pesar de aplicarsele una factor de escala. Los ejemplos más citados son las nubes y las costas.

Pixel

Es el mínimo elemento direccionable que puede desplegarse en la pantalla.

Raster

Un patrón predeterminado de líneas examinadas que proveen una cobertura uniforme de una área de despliegue.

Raster Scan

Una técnica donde la imagen es examinada en una pantalla por medio de una secuencia raster, i.e., como una sucesión de líneas equidistantes, cada línea se forma de pixels.

Render

El proceso de generar una imagen tomando en cuenta el modelo geométrico, el modelo de iluminación, una camara de visión, manejo de sombras y otros parámetros. La elección del algoritmo depende de la representación del modelo y del grado de realismo que se quiera.

Spline

Curva generada a paritr de algunos puntos de control.

Textura

Patrón bidimensional o tridimensional que puede ser mapeado a la superficie del objeto.

Voxel

Un elemento tridimensional, usualmente un cubo o un paralelepipedo. Este tiene una valor (o densidad) y seis lados y representa un punto de tamaño finito en una descomposición regular del espacio.

Z-buffer

Una técnica de "render" en la cual los objetos se examinan pixel por pixel, creando un arreglo de profundidad para cada uno de estos pixels. Cuando la profundidad del objeto que está analizando para el pixel encuestión, es menor que el valor que está almacenado, éste se reemplaza. La ventaja del Z-buffer es que los objetos no necesitan ser procesados en un orden espacial.

BIBLIOGRAFIA

[1] Ebert, David. *Modeling and Animating Gases and Fluids.* ACM SIGGRAPH 1992 Procedural Modeling and Rendering Techniques Curso de Notas 23: 4.1-4.38.

[2] Ebert, David, y Parent, Richard. *Rendering and Animation of Gaseous Phenomena by combining Fast Volume and Scanline A-buffer Techniques.* Proceedings of SIGGRAPH'90 Computer Graphics 24(Agosto 1990), 4:357-366.

[3] Fournier Alan, Fusell Carpenter D. *"computer rendering of estocathic models".* COMM of ACM 25 (junio 82) 6:371-384.

[4] Fournier Alan. *The Modelling of Natural Phenomena.* ACM SIGGRAPH 1994. Modelling Natural Phenomena Curso de Notas 22.

[5] Frost W, Moulden *Handbook of turbulence Y: Fundamentals and applications.* Plenum-Press, New York. (1977).

[6] Gagalowicz A, Song D. Ma. *"Model driven synthesis of natural textures for 3-D scenes".* CGB 10(1986), 2:161-170

[7] Gardner, Geoffrey Y. *Visual Simulation of Clouds.* ACM SIGGRAPH 1988 Functional Based Modeling Curso de Notas 28:87-93.

[8] Kajiya James, y Von Herzen, Brian. *Ray Tracing Volume Densities.* Proceedings of SIGGRAPH'84. Computer Graphics 18(Julio 1984),3:165-174.

[9] Lovejoy S. Mandelbrot. *Fractal properties of rain, an a fractal model.* Tellus (1985) 37 A(3):209-232.

[10] Mandelbrot, *The Fractal Geometry of Nature*. W.H. Freeman and Co. (1982).

[11] Mandelbrot B.B. , *On the geometry of homogeneous turbulence, with stress on the fractal dimension of the iso-surfaces of scalars*. J. Fluid Mech (1975) 72(2):401-416.

[12] Mandelbrot, B.B and Van Ness J. W. *"Fractional Brownian Motion, Fractional Noises and Applications"*. (Octubre 1968) SIAM 10(4) pp 422-437.

[13] Max, Nelson. *Light Diffusion Through Clouds And Haze*. Computer Vision, Graphics, and Image Processing (1986) 33:280-292.

[14] Max, Nelson. *Vectorized Procedural Models for Natural Terrain: Waves and Islands in the Sunset*, Computer Graphics 15(Julio 1981) 3: 317-324.

[15] Nishita, Tomoyuki, Miyawaki, Yasuhiro, y Nakamae, Eihachiro. *A shading Mode for Atmospheric Scattering Considering Lumminuos Intensity Distribution of Light Sources*. Proceedings of SIGGGRAPH'87. Computers Graphics 21,4 (Julio1987):303-310.

[16] Panchev S. *Random functions and turbulence*. Int Ser. Monogr Nat Philos, pag. 32.

[17] Perlin, K.. *An Image Synthesizer*, , Proceedings SIGGRAPH '85. Computer Graphics, 19(3), 287-96.

[18] Peitgen HO, Saupe D. *The science of fractal images* (1988).Springer Verlag Berlin Heidelberg New York

[19] Reeves W.T. "Particle Systems - *A Technique for Modelling a class of Fuzzy Objects*". Computers Graphics 17(3) pp 359-376. (Julio 1985).

[20] Reeves W. T. an Blao R. *"Aproximate and Probabilistic Algorithms for shading and Rendering Structured Particle Systems"*. Computer Graphics 19(3) pp. 313-322 (Julio 1985).

[21] Reffye P. , *Modelisation de l'architecture des arbres par des processus stochastiques: simulation spatiale des models tropicaus sous l'effect de la pesanteur. Application au Coffea Robusta*. PhD Thesis Universidad de Paris (1979).

[22] Reffye P., Edelin C. Francon J. , Jaeger M., y Puech. C. *Plant Models Faithful to Botanical Structure and Development* . Computer Graphics 22 (Agosto 1988) 4:151-158.

[23] Sakas, Georgios. *Modeling and animating turbulent gaseous phenomena using spectral synthesis*. Visual Computer (1993) 9:200-212.

[24] Schachter, B. J. *Long Crested Wave Models*, Computer Graphics and Image Processing, 12, 187-201.

[25] Tennekes H, Lumley J. A first course in turbulence. MIT Press, Cambridge. 1972.

[26]Thompson, D. W. *On Growth and Form* Cambridge University Press (1961)

[27] Voss, Richard. *Fourier Synthesis of Gaussian Fractals: I/f noises, langscapes, and flakes*. SIGGRAPH'83:Tutorial on State of the Art Image Synthesis (1983), vol. 10, ACM SIGGRAPH.

[28] Voss Richard. *Random Fractal forgeries*. ACM Comput Graph. (1985).

[29] Watt, Alan y Watt *Mark Advanced Animation and Rendering Techniques Theory and Practice*. Addsion-Wesley Publishing Company. pag. 202-218. 1992.