Universidad de las Ciencias Informáticas

Facultad 5

REPRESENTACIÓN DE LAS NUBES EN EL SIMULADOR VIRTUAL DE TIRO C-10.

Trabajo de Diploma en Opción al Título de Ingeniero en Ciencias Informáticas

Autor

Janiel Reyes Leyva

Tutor

ING. Yusleidy Guelmes León

Ciudad de la Habana

Mayo 2007

Dedicatoria

A ti, mi SEÑOR JESUS… *sea toda la gloria y toda la honra, desde ahora y para siempre.*

"Y dijo al hombre:
He aquí que el temor del Señor es la sabiduría,
Y el apartarse del mal, la inteligencia"*.*
La Biblia [Job 28:28]

Agradecimientos

A mi hermano Alberto por su gran ayuda.

A mi tutora por la atención que me ha dado.

A mi madre por su preocupación y ruegos.

A todos mis hermanos en la fe, por su apoyo y por ser un ejemplo a seguir.

A mis profesores y compañeros.

A todos ustedes, gracias.

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los 10 días del mes de mayo del año 2007

Janiel Reyes Leyva.	Yusleidy Guelmes León .
AUTOR	TUTOR

OPINIÓN DEL USUARIO DEL TRABAJO DE DIPLOMA

El Trabajo de Diploma, titulado **“Representación de las Nubes en el Simulador Virtual de Tiro C-10”**, fue realizado en la **Universidad de las Ciencias Informáticas**. Esta entidad considera que, en correspondencia con los objetivos trazados, el trabajo realizado le satisface

- ☐ Totalmente
- ☐ Parcialmente en un ____ %

Los resultados de este Trabajo de Diploma le reportan a esta entidad los beneficios siguientes (cuantificar):

Como resultado de la implantación de este trabajo se reportará un efecto económico que asciende a _____

Y para que así conste, se firma la presente a los ____ días del mes de _____ del año _____.

_____ _____

Representante de la entidad Cargo

_____

Firma Cuño

OPINION DEL TUTOR

Título: Representación de las Nubes en el Simulador Virtual de Tiro C-10

Autor(es): Janiel Reyes Leyva

Tutor(es): Yusleidy Guelmes León

El tutor del presente Trabajo de Diploma considera que durante su ejecución el estudiante mostró las cualidades que a continuación se detallan.

Por todo lo anteriormente expresado considero que el estudiante está apto para ejercer como Ingeniero Informático; y propongo que se le otorgue al Trabajo de Diploma la calificación de _____

_____

Firma

Resumen

El trabajo que se presenta a continuación está relacionado con escenarios virtuales. El desarrollo de la ciencia ha permitido procesar y llevar a la pantalla de un ordenador muchos procesos que ocurren en la vida real. El querer similar estos procesos trajo consigo lo que se conoce como simulador. En la realidad toda escena al aire libre consta de una atmósfera, la cual puede o no tener presencia de nubes.

En nuestro centro se está desarrollando un simulador de tiro para realizar entrenamientos militares. Se quiere lograr que el usuario se encuentre inmerso en un ambiente real. Este trabajo fue inspirado debido a la carencia de nubes en este simulador. Cuenta con imágenes de nubes , las cuales se ponen en moviendo y dan una apariencia de realidad , pero lo que se quiere lograr es un comportamiento real de la misma en el simulador*.*

Para la solución de este problema se hizo necesaria la realización de un estudio de los modelos matemáticos, que hicieran posible la simulación de las nubes. Se encontró el algoritmo Automata Celular, el cual permitió la simulación real de las nubes. El trabajo se centra en proponer un modelo que hiciera posible la animación realista de las nubes.

Se realizó una aplicación que permite la visualización en pantalla de las nubes, con el objetivo de probar que el modelo matemático era eficiente. Para esta tarea se hicieron necesarios dos procesos: simulación y rendering (representación). La parte que se va a simular la define el Autómata Celular, esta acción se hace invisible a la vista, luego se toma este volumen simulado y a través del proceso de rendering se representa en pantalla.

Este trabajo propone una idea eficiente para lograr un comportamiento real de las nubes en el simulador virtual de tiro. El uso de la tecnología hizo posible el logro de las nubes virtuales, dando al usuario una sensación de realidad.

Índice

Introducción

Con el surgimiento de las computadoras, el hombre ha utilizado los beneficios que brinda la misma para representar y llevar situaciones de la vida real a la computadora, a través de realidad virtual. El desarrollo del Hardware y la implementación de modelos matemáticos eficientes han permitido un impulso de los escenarios virtuales.

La Realidad Virtual es considerada en muchos aspectos como la interfase definitiva entre los seres humanos y el ordenador. Básicamente consiste en simular todas las posibles percepciones de una persona como los gráficos para la vista, sonido, tacto e incluso sensaciones de aceleración o movimiento. Todas esas sensaciones diferentes deben ser presentadas al usuario de forma que se siente inmerso en el universo generado por el ordenador, hasta el punto de dejar de percibir la realidad y sentirse transportado (al otro lado de la pantalla) como si de un universo nuevo se tratase.

Puede ser de dos tipos: *inmersiva* y *no inmersiva*. Los métodos inmersivos de realidad virtual con frecuencia se ligan a un ambiente tridimensional creado por un ordenador, el cual se manipula a través de cascos, guantes u otros dispositivos que capturan la posición y rotación de diferentes partes del cuerpo humano. La realidad virtual no inmersiva utiliza medios como el que actualmente nos ofrece Internet, en el cual podemos interactuar en tiempo real con diferentes personas en espacios y ambientes que en realidad no existen sin la necesidad de dispositivos adicionales al ordenador

La realidad virtual no inmersiva ofrece un nuevo mundo a través de una ventana de escritorio. Este enfoque no inmersivo tiene varias ventajas sobre el enfoque inmersivo como son el bajo coste y fácil y rápida aceptación de los usuarios. Los dispositivos inmersivos son de alto coste y generalmente el usuario prefiere manipular el ambiente virtual por medio de dispositivos familiares como son el teclado y el ratón que por medio de cascos pesados o guantes.

La Universidad de las Ciencias Informáticas (UCI) ha establecido relaciones con la institución SIMPRO la misma se dedica al desarrollo de simuladores virtuales, con el objetivo de utilizarlo en entrenamientos militares, debido a que disminuye los gastos y no hay riesgo de vidas humanas. La escuela en

coordinación con esta institución ha establecido un local donde se está desarrollando el simulador[1] virtual de tiro, centrado en la parte de defensa antiaérea. Los compañeros de SIMPRO aportan los conocimientos por los cuales se rigen en su formación como militares, la UCI aporta la tecnología y también el personal que se encarga de procesamiento de estos conocimientos.

Para el mismo se desea representar el comportamiento de las nubes ante una defensa antiaérea. En la actualidad, cuenta con planos que representan a las nubes, estos planos son imágenes de nubes, que son usados para lograr así una mayor realidad en cuanto al entorno virtual. Por lo que se necesita la existencia de nubes que no solo parezcan reales sino que se comporten similares a la realidad, con sus cambios de estado y variaciones.

Problema Científico

Inexistencia de un modelo matemático en el proceso de simulación, que de a las nubes un comportamiento real en el simulador de tiro.

Objeto de estudio

Simulación virtual de gases

Campo de acción

Simulación de nubes a través de un modelo matemático.

Objetivo de la investigación

Como obtener un comportamiento real de las nubes en el simulador de tiro mediante la implementación de un modelo matemático

Idea a defender

La utilización de un modelo matemático para simular el comportamiento de las nubes proporcionará mayor realismo al simulador de tiro C-10.

Tareas a desarrollar

1- Realizar una investigación sobre modelos matemáticos utilizados en la simulación.

2- Proponer un modelo matemático para simular el comportamiento de las nubes.

3- Estudiar lenguaje de programación VISUAL C++.net.

4- Estudiar componentes gráficos utilizados en el proceso de Simulación y rendering de las nubes.

Organización del documento

Este trabajo ha sido organizado de la siguiente manera: El Capitulo 1 presenta los conocimientos referente a la simulación así como algunos modelos matemáticos que se utilizan en la simulación de envolturas. En el Capitulo 2 se hace mención a la tecnología utilizada en el desarrollo de la aplicación. En el Capitulo 3 se da la solución al problema, aquí se encuentra la implementación del modelo y los detalles que hicieron posible lograr la visualización de las nubes.

CAPITULO 1 Simulación

1.1 Introducción

Thomas H. Naylor[2] la define así: "***Simulación*** es una técnica numérica para conducir experimentos en una computadora digital. Estos experimentos comprenden ciertos tipos de relaciones matemáticas y lógicas, las cuales son necesarias para describir el comportamiento y la estructura de sistemas complejos del mundo real. "(WIKIMEDIA FOUNDATION 2006).

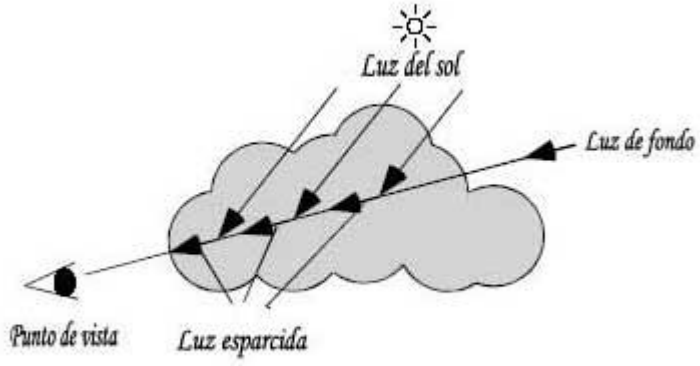


Figura #1 Especificando la luz de la nube

Las nubes desempeñan un papel importante al hacer las imágenes para el vuelo simulado o escenas al aire libre. Su color y cambio de formas dependiendo de la posición del sol y del observador influye mucho en que las nubes tengan una apariencia real. La distribución de la densidad de las nubes se debe definir en espacio tridimensional para crear imágenes realistas. Se han desarrollado varios modelos matemáticos para exhibir las nubes. Usando estos métodos, las imágenes de nubes que son extremadamente realistas pueden ser generadas a través de una computadora.

Este capítulo introduce al conocimiento y funcionamiento de algunos modelos matemáticos utilizados en la *simulación* de mundos virtuales, luego de un estudio acerca de estos modelos, se hace necesario llegar a una conclusión de cual será la mejor opción.

[2] Profesor honorario de la economía en la Universidad de Duke, es escritor y activista político.

Los **modelos matemáticos**, son ecuaciones matemáticas, que se encargan de encontrar soluciones analíticas a los problemas, permitiendo la predicción del comportamiento del sistema a partir de parámetros y de condiciones iniciales.

1.2 ¿Cuándo es posible simular?

Simular con computadora no siempre es posible, esto depende de algunas condiciones específicas. Por tanto se necesitan algunas condiciones, que se muestran a continuación:

1. Cuando se tiene el modelo matemático definido.

2. Cuando se tiene una formulación exacta del sistema.

3. Cuando se tienen las fórmulas analíticas y se necesita un modelo para ponerlas a funcionar.

4. Cuando el costo o la corrida de un modelo no es costosa.

5. Cuando al ver un proceso físico, el cual nosotros queremos conocer, la simulación es la única forma que tenemos para conocer el comportamiento real.

Ejemplo: fenómeno climático.

6. Cuando se requiere acelerar o retrazar el tiempo de los procesos dentro de un sistema.

7. Cuando se quiere por medio de la simulación encontrar o hacer estudios y/o experimentos.

Simular la realidad es mucho más que mostrar una imagen que la represente en una pantalla. La realidad es dinámica. Cambia con el tiempo y está sujeta a una serie de leyes físicas que determinan los cambios que en ella se producen. En este caso se quiere simular el comportamiento de las nubes.

1.3 Tipos de simulación

Simulación física

La creciente potencia de cálculo de los ordenadores ha permitido que actualmente sea factible el uso de técnicas avanzadas de ***simulación física*** en cualquier equipo doméstico de gama media.

La ***simulación física*** desempeña un papel importante en múltiples campos actualmente en auge, como son la realidad virtual y los videojuegos. También se hace uso de la simulación física en el nuevo cine de animación, donde el realismo de los elementos que componen la escena y sus interacciones alcanza unos niveles inimaginables hace pocos años.

Simulación interactiva

La ***simulación interactiva*** es una clase especial de simulación física, en la cual las simulaciones físicas incluyen a operadores humanos, tales como dentro de un simulador de vuelo.

1.4 Modelos de simulación (MOREA 2006)

CRITERIOS QUE SE DEBE TENER EN CUENTA PARA QUE UN MODELO DE SIMULACION SEA BUENO.

1. Fácil de entender por el usuario.

2. Tenga el modelo metas y objetivos.

3. Que el Modelo no me de respuestas absurdas.

4. Que sea fácil de manipular, la comunicación entre el usuario y la computadora debe ser sencilla.

5. Que tenga por lo menos las partes o funciones más importantes del sistema.

6. Que podamos modificarlo, adaptarlo, y actualizarlo

1.4.1 Propiedades de los modelos de simulación

FUNCIONES DEL MODELO

- Comparar
- Predecir

ESTRUCTURA DEL MODELO

El modelo se puede escribir de tal forma

E = F (Xi, Yi)

Donde

E: Es el efecto del comportamiento del sistema

Xi: Son las variables y parámetros que nosotros podemos controlar

Yi: Las variables y los parámetros que nosotros no podemos controlar

F: Es la ***función matemática*** con la cual relacionamos Xi con Yi con el fin de modificar o dar origen a E

Las ***funciones matemáticas*** a las que se hace referencia son las siguientes:
Función Trigonométrica
Función Cuadrática
Función Lineal)
Función Logarítmica

Función Exponencial
Función Polinómica

1.4.2 Características de los modelos

1. VARIABLES:
Pueden ser de dos tipos (*Exógenos*, *Endógenos*)
- Exógenas: Entradas que son originadas por causas externas al sistema.
- Endógenas: Son producidas dentro del sistema que resultan de causas internas, las cuales pueden ser de *Estado* o de *Salida.*

Estado: Muestran las condiciones iniciales del sistema
Salida: Son aquellas variables que resultan del sistema

Estadísticamente a las variables exógenas se las denomina como variables independientes

2. PARÁMETROS:

Son cantidades a las cuales el operador del modelo puede asignarle valores arbitrarios lo cual se diferencia de las variables.

Los parámetros una vez establecidos se convierten en constantes.

3. RELACIONES FUNCIONALES:

Describen a los parámetros de tal manera que muestran su comportamiento dentro de un componente o entre componentes de un sistema.

Las relaciones funcionales pueden ser de tipo *determinísticos* o *estocásticos.*

- ***Determinísticas***: Se relacionan ciertas variables o parámetros donde una salida del proceso es singularmente determinada por una estrada dada.

- ***Estocásticas***: Cuando el proceso tiene una salida indefinida, para una entrada determinada las relaciones funcionales se representan por ecuaciones matemáticas y salen del análisis estadístico matemático.

4. RESTRICCIONES:

Estas son limitaciones impuestas a valores de las variables las cuales pueden ser de dos formas:

- ***Autoimpuestas***: cuando son asignadas por el mismo operador.

- ***Impuestas***: cuando son asignadas manualmente por el mismo sistema.

1.4.3 Clasificación de los modelos

Los modelos matemáticos se pueden clasificar en forma general, pero los modelos de simulación se pueden clasificar en forma más específica.

MODELOS FISICOS:

Son los que mas se asemejan a la realidad, se encargan de modelar procesos los cuales pueden ser:

- MODELOS ANALOGICOS:

Se encargan de representar una propiedad determinada de un objeto o sistema

- MODELOS DENOMINADOS JUEGOS ADMINISTRATIVOS:

Ya empieza a involucrarse al comportamiento del ser humano.

Ej: modelos de planeación, estrategias militares.

CLASIFICACION DE LOS MODELOS DE SIMULACION

Dentro de los modelos de simulación están:

1. MODELOS DETERMINISTICOS

Ni las variables endógenas ni exógenas se pueden tomar como datos al azar. Aquí se permite que las relaciones entre estas variables sean exactas, es decir, que no entren en ellas funciones de probabilidad. Este tipo determinístico quita menos tiempo de cómputo que otros modelos.

Ejemplo: Modelos Estocásticos

2. MODELOS ESTOCASTICOS

Cuando por lo menos una variable es tomada como un dato al azar las relaciones entre variables se toman por medio de funciones probabilísticas, sirven por lo general para realizar grandes series de muestreos, quitan mucho tiempo en el computador son muy utilizados en investigaciones científicas.

3. MODELOS ESTATICOS

Es que en ellos no se toma en cuenta el tiempo dentro del proceso, por ejemplo: los modelos de juegos, modelos donde se observa las ganancias de una empresa

Ejemplo: Arquitectónicos: líneas de teléfono, tubos de agua.

4. MODELOS DINAMICOS

Si se toma en cuenta la variación del tiempo, ejemplo: la variación de la temperatura, del aire durante un día, movimiento anual de las finanzas de una empresa.

5. MODELOS A ESCALA

Son los modelos sencillos de maquetas -> casa -> baño, cuartos, etc. También se pueden tener a tamaño natural a menor o mayor escala, bidimensional, tridimensional.

.

1.5 Modelos matemáticos utilizados en la simulación de envolturas

La *simulación* de entornos virtuales siempre se mueve en los límites de la tecnología. Siempre aprovecha toda la capacidad de cómputo disponible. Cuanto más rápidos son los ordenadores, más realista es la simulación. Mientras más rápido sea el hardware y se tengan algoritmos más avanzados, permitirá simular entornos con mayor lujo de detalles.

Simulación física

- Solve Navier-Stokes Equation
- Finite difference method
- Finite element method
- Multigrid method

Simulación heurística

- Cellular Automata [DKY+00,DNO98]
- Coupled Map Lattice

1.6 Breve reseña sobre dichos modelos

1.6.1 Solve Navier-Stokes Equation

Es un sistema de ecuaciones diferenciales que describen el movimiento de las sustancias fluidas tales como líquidos y gases, este sistema es desarrollado por **Claude-Louis Navier** (ingeniero y físico franceses). (WIKIPEDIA 2006)

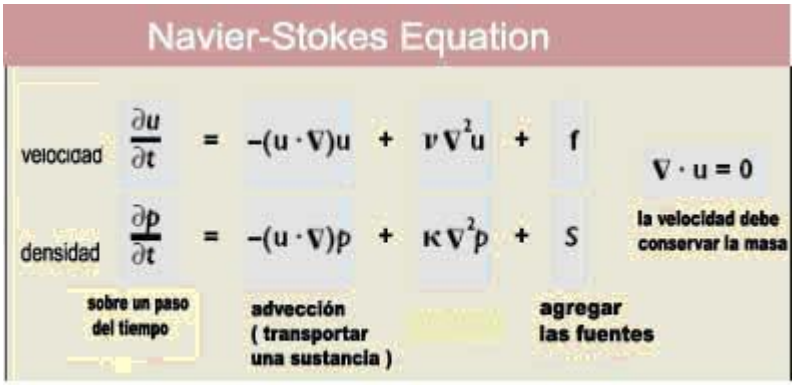


Figura #2 Ecuación de Navier-Stokes

El sistema de ecuaciones se deriva de principios de la conservación de:

- Masa
- Energía
- Ímpetu (aceleración)
- Ímpetu angular

La conservación de la masa se escribe:

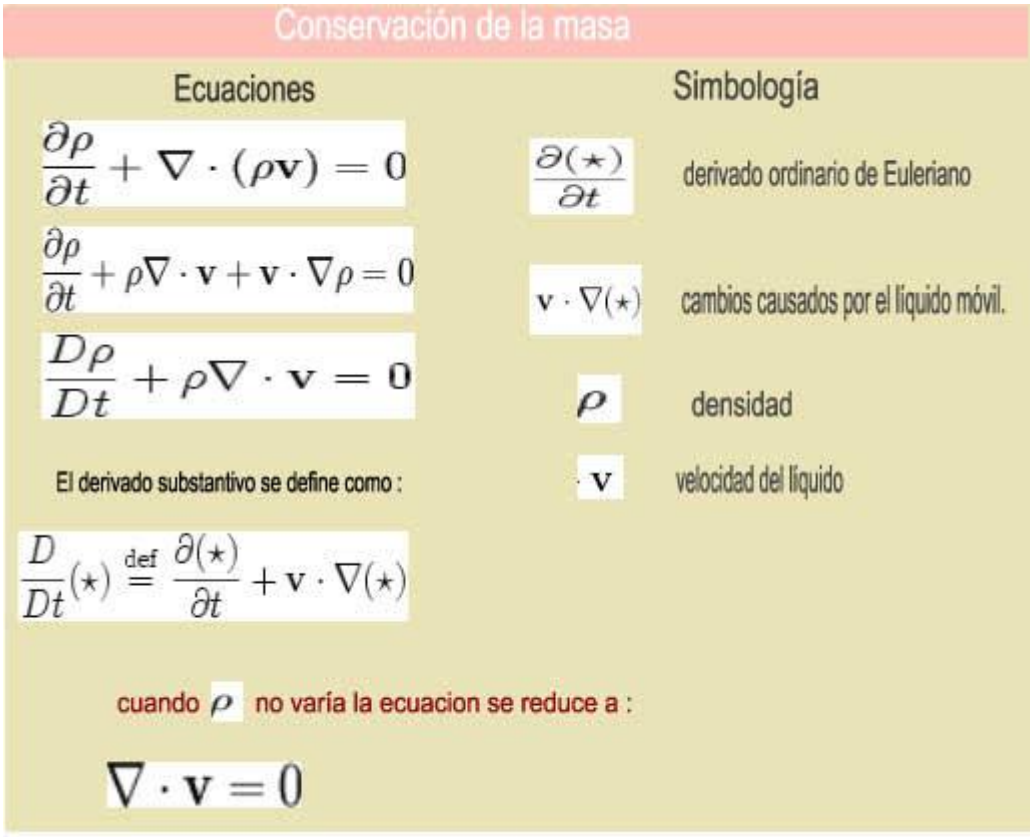


Figura #3 Principios de la ley de conservación de la masa

VENTAJAS

Es uno de los sistemas de ecuaciones más útiles, porque describen la física de una gran cantidad de fenómenos del interés académico y económico.

Se utilizan para modelar el tiempo, corrientes del océano, el movimiento de estrellas dentro de una galaxia. También se utilizan en el diseño del avión y de los coches, el estudio del flujo de la sangre, el diseño de las centrales eléctricas y otros casos.

DESVENTAJAS

- Es un modelo matemático de difícil comprensión,
- Gran cantidad de cálculo matemático.
- Utiliza mucho tiempo de procesamiento.

1.6.2 Finite Difference Method (Método de Diferencia Finita)

Una diferencia finita es una expresión matemática del − *f* (*x* +a) de la forma *f* (*x* + *b*). La aproximación de derivados por diferencias finitas desempeña un papel central en los métodos finitos de la diferencia para la solución numérica de ecuaciones diferenciales parciales. (WIKIMEDIA FOUNDATION 2006)

Solamente tres formas se consideran comúnmente:

Una diferencia delantera es una expresión de la forma

$$\Delta[f](x) = f(x+h) - f(x).$$

Una diferencia posterior se presenta cuando *h* es substituido por el −h:

$$\nabla[f](x) = f(x) - f(x-h).$$

Finalmente, la **diferencia central** es el promedio de las diferencias delanteras y posteriores.

$$\delta[f](x) = \frac{\Delta[f](x) + \nabla[f](x)}{2} = \frac{f(x+h) - f(x-h)}{2}.$$

El **derivado de una función** *f* en un punto *x* es definido por el límite:

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}.$$

Básicamente, en una solución por diferencias finitas, las derivadas son reemplazadas por aproximaciones en diferencias finitas, convirtiendo entonces un problema de ecuaciones diferenciales en un problema algebraico fácilmente resoluble por medios comunes (especialmente matriciales).

Los diversos métodos finitos pueden ser: (PETROBRAS 2007).

(a) Método implícito:

Las ecuaciones diferenciales parciales pueden ser solucionadas indirectamente solucionando un sistema de ecuaciones lineares simultáneas.

(b) Método explícito:

Las ecuaciones diferenciales se pueden solucionar directamente usando las condiciones de límite apropiadas.

Figura #4 Método de Diferencia Finita

VENTAJAS

- Es una manera clásica y directa, que numéricamente da solución a las ecuaciones diferenciales parciales.
- Este método se puede utilizar para solucionar cualquier ecuación diferencial parcial (PDE) encontrada generalmente en asuntos financieros.

DESVENTAJAS

Las aproximaciones dan resultados muy imprecisos en los valores de las derivadas, salvo que se elijan pasos muy pequeños, haciendo el cálculo excesivamente largo en el tiempo.

Los errores de redondeo pueden conducir a inestabilidades de la solución numérica.

1.6.3 Finite element method (Método de Elemento Finito)

Método de Elementos Finito (*FEM*), es una técnica matemática compleja, utilizada para modelar algunos sistemas físicos y mecánicos, cuya finalidad es dividir espacios de naturaleza contínua, sobre los cuales es posible realizar análisis numéricos para comprender, por medio de un modelo discreto, el comportamiento de sistemas analógicos.

VENTAJAS

FEM es de amplia utilización en análisis de sistemas y espacios físico-mecánicos donde el objetivo sea comprender la resistencia de materiales, la dinámica de partículas y en general el comportamiento y la interacción de los elementos base del sistema en el espacio.

DESVENTAJAS

- La complejidad de aplicar FEM sobre algunos sistemas es tal, que resulta difícil lograr modelos que describan con precisión sus comportamientos. Es una técnica matemática compleja para modelar algunos sistemas físicos y mecánicos.
- Existen muchos sistemas complejos y de diversa naturaleza en los cuales no es convencional aplicar esta técnica, por ejemplo, sistemas químicos, biológicos, evolutivos, genéticos, eléctricos, computacionales e inclusive otros físicos y mecánicos.

1.6.4 Método Multigrid

El primer *Método Multigrid* fue introducido por **Fedorenko** in 1961. Su eficacia real primero fue observada por **Brandt** en 1973, utilizándola en diversos problemas. En 1976, independiente a su desarrollo, Hackbusch introdujo ***Multigrid método***.

Los *Métodos Multigrid* son un grupo de algoritmos (procedimiento matemático) para solucionar ecuaciones diferenciales usando un nivel de ***discretizaciones*****.** (WIKIMEDIA FOUNDATION 2007)

La simulación por discretización de ecuaciones diferenciales

La simulación por discretización de ecuaciones diferenciales es un método en donde es esencial el concepto de *"discretización"*, acción de transformar la realidad de la naturaleza continúa en un modelo discreto aproximado, esto es llevar las ecuaciones diferenciales parciales a forma más fácil de trabajar que son las matrices.

También es esencial el concepto de *"interpolación"*, es la aproximación de una función complicada por una más simple.

.

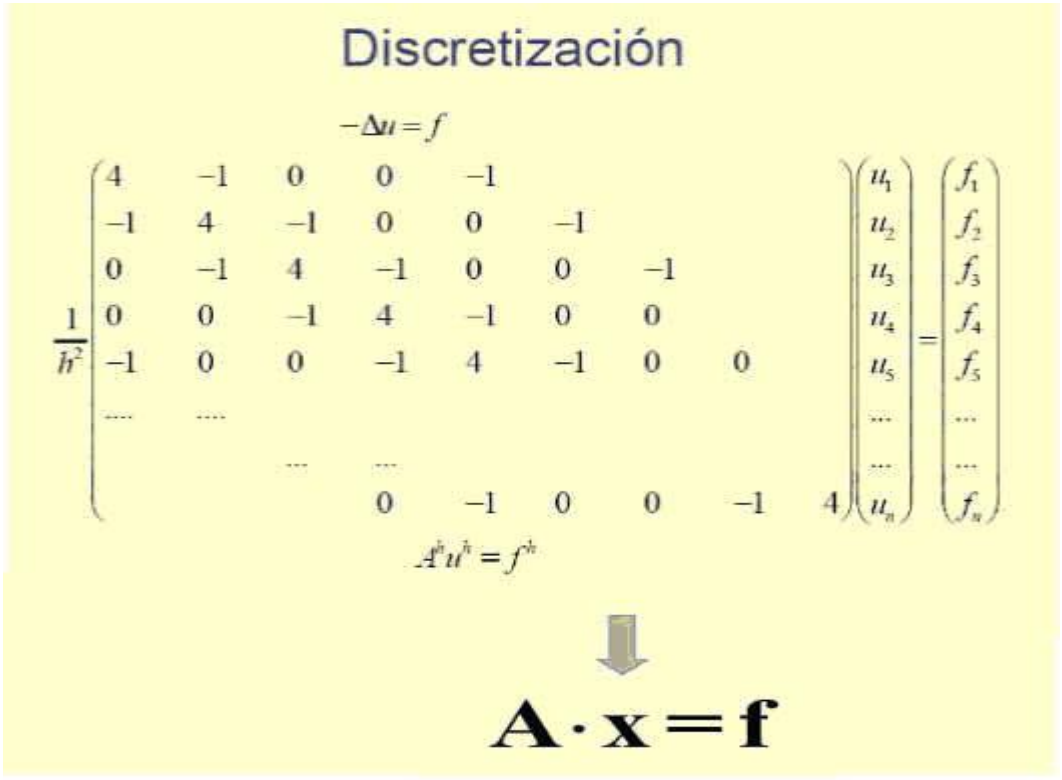


Figura #5 Discretización de Ecuaciones Diferenciales

Interpolación* o *prolongación.

A partir de una *interpolación linear*, podemos llegar a una *bilineal.* Este proceso puede realizarse en una o' dos dimensiones.

La ***interpolación linear*** puede ser escrita como:

$$P = \begin{bmatrix} 0 & 1/2 & 1/2 \\ 1/2 & 1 & 1/2 \\ 1/2 & 1/2 & 0 \end{bmatrix}.$$

En la ***interpolación lineal unidimensional***, se parte de un segmento grueso de la rejilla definido por $u_H \in \mathbb{R}^{n_H}$ a el segmento fino de la rejilla, que se define de la forma $u_h \in \mathbb{R}^{n_h}$, $n_h = 2n_H + 1$. Esto se explica a continuación con un ejemplo, figura #6.

$$
\begin{aligned}
(u_h)_1 &= (P\,u_H)_1 = 0.5\,(u_H)_1, \\
(u_h)_{n_h} &= (P\,u_H)_{n_h} = 0.5\,(u_H)_{n_H}, \\
(u_h)_i &= (P\,u_H)_i = \begin{cases} (u_H)_i & : \ i \text{ uniforme} \quad i = 2,\ldots,n-1, \\ 0.5\,((u_H)_{\frac{i-1}{2}} + (u_H)_{\frac{i+1}{2}}) & : \ i \text{ impar} \quad i = 3,\ldots,n-2 \end{cases}
\end{aligned}
$$

$u_H \in \mathbb{R}^{n_H}$
$u_h \in \mathbb{R}^{n_h},\ n_h = 2n_H + 1$
P prolongación $H = 2h$

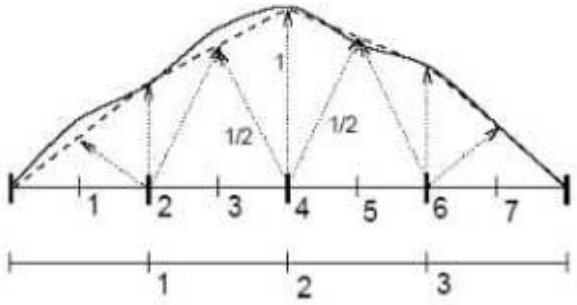


Figura #6 Interpolación Unidimensional

Interpolación lineal bidimensional

En el ejemplo mostrado, los componentes $(u_h)_{(2,2)}, (u_h)_{(3,3)}, (u_h)_{(3,4)}$ del segmento fino de la rejilla $u_h = P\,u_H$ son procesados por la computadora según se muestra a continuación:

$$
\begin{aligned}
(u_h)_{(2,2)} &= 0.5\,[\,(u_H)_{(1,1)} + (u_H)_{(2,2)}\,], \\
(u_h)_{(3,3)} &= (u_H)_{(2,2)}, \\
(u_h)_{(3,4)} &= 0.5\,[\,(u_H)_{(2,2)} + (u_H)_{(2,3)}\,].
\end{aligned}
$$

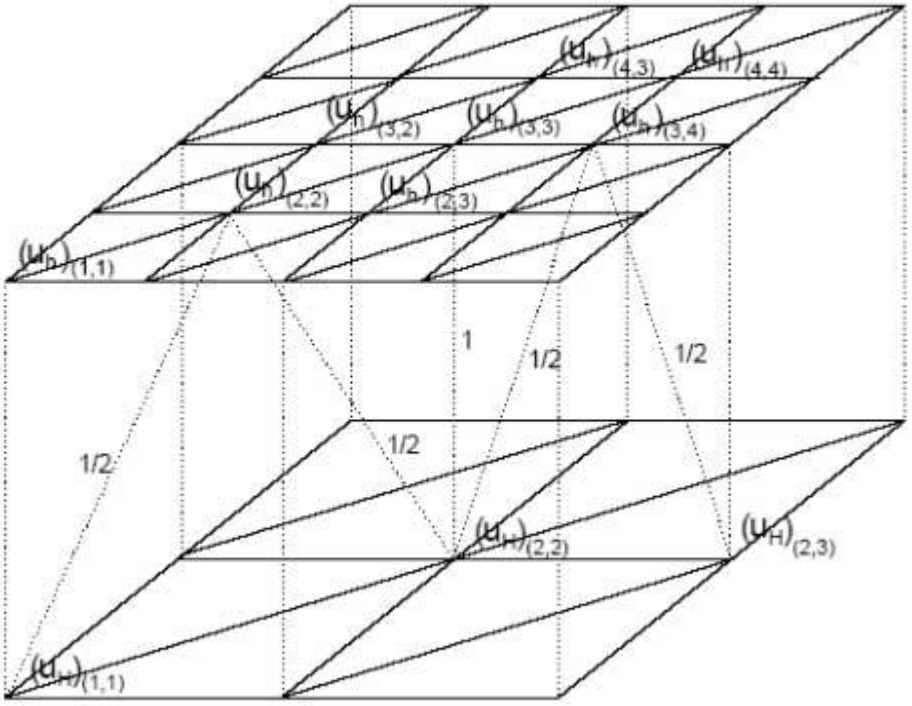


Figura #7 Interpolación Lineal Bidimensional

La ***interpolación bilineal*** es dada por:

$$P = \begin{bmatrix} 1/4 & 1/2 & 1/4 \\ 1/2 & 1 & 1/2 \\ 1/4 & 1/2 & 1/4 \end{bmatrix}$$

Después de un proceso matemático se tiene un resultado semejante a la figura #8. Para así asignar a cada rejilla un estado.

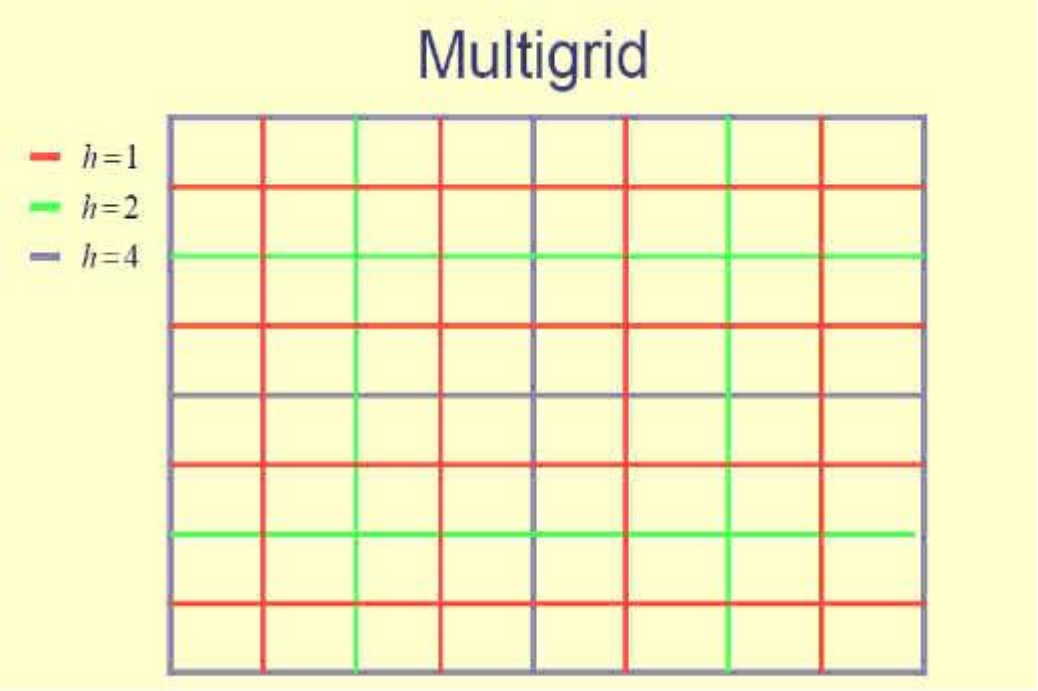


Figura #8 Modelo del Método Multigrid

VENTAJAS

Es uno de los algoritmos más rápido posible con grado de complejidad **O(n).**

Es adaptable al método de Elemento Finito.

Se puede emplear en refinaciones de malla

DESVENTAJAS

Presenta muchas aproximaciones de valores, por lo que no ocurre una transformación exacta.

Presenta difíciles operaciones de cálculo.

1.6.5 Autómata Celular

Los autómatas celulares fueron inventados a fines de los cuarenta por Stanislaw Ulam (1909-1984) y John Von Neumann (1903-1957). Ulam fue principalmente matemático, inventó el método de simulación "Monte Carlo" y aportó importantes contribuciones a la teoría de los números y al análisis matemático. **Von Neumann** trabajó en los más variados campos. Colaboró en los fundamentos de la teoría de la mecánica cuántica, incursionó en economía y en la teoría de los juegos

*Un **autómata celular** es una herramienta computacional que sirve para demostrar el comportamiento emergente que se puede producir en un sistema matemático, está basado en modelos biológicos (comportamiento de células). Presenta un conjunto finito de reglas que son aplicadas a cada celdas o' células. Se establecen un sistema de celdas que varían su estado dependiendo de las células vecinas.(COMPUTACIÓN 1999)*

Un autómata celular unidimensional consta de un arreglo lineal finito de celdas o células, figura #9.

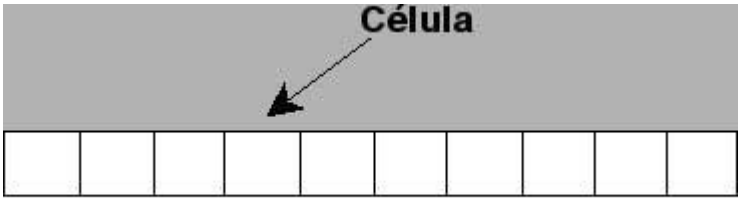


Figura #9 Autómata Celular Unidimensional

Arreglo de 10 células.

Estructura de un Autómata

*Según **Muñoz**[3], **estructura** de un Autómata Celular:*

[3] Doctor Ingeniero de Telecomunicación en la Universidad Politécnica de Madrid.

- Un plano bidimensional o un espacio n-dimensional dividido en un número de subespacios homogéneos, conocidos como *celdas*. Esto se le conoce como *Teselación homogénea*.
- Cada celda puede estar en uno de un *conjunto finito o numerable* de **estados̃**
- Una *Configuración C*, la que consiste en asignarle un estado a cada celda del autómata.
- Una *Vecindad* definida para cada celda, la que consiste en un conjunto contíguo de celdas, indicando sus posiciones relativas respecto a la celda misma.
- Una *Regla de Evolución*, la cual define cómo debe cada celda cambiar de estado, dependiendo del estado inmediatamente anterior de su vecindad.

Según **Toffoli** y **Margolus**[4], se define un *Autómata Celular* sólo si se tiene que todas las celdas:

- Tienen el mismo *Conjunto de Estados* posibles.
- Tienen la misma forma de *Vecindad*.
- Tienen la misma *Regla de Evolución*.

Consideraciones adicionales

Un *Autómata Celular* puede ser construido definiendo su *teselación*, los posibles *estados*, las *vecindades* y la *regla de evolución*; no obstante, se tienen unas consideraciones y posibilidades con estos componentes, las que permitirán cierta flexibilidad en el momento de construir el *autómata*.

- El *autómata* puede ser de 1, 2, 3,..., n dimensiones.
- La *teselación* puede ser finita o infinita, con condiciones de fronteras abiertas o periódicas.
- El *conjunto de estados̨* no necesita tener ninguna estructura algebraica adicional.
- La *vecindad* puede ser simétrica o no y puede incluir o no a la propia celda.

[4] Hicieron nuevos aportes a los autómatas celulares, agregándole mayor eficiencia.

- La *regla de evolución* se declaran conforme a la necesidad con que se va a utilizar ese modelo.

Algunos ejemplos

Tal vez, lo más llamativo e interesante de los *Autómatas Celulares* es el comportamiento presentado por el modelo en tiempo de ejecución y la similitud de éste con la complejidad de la naturaleza contínua. *"Life"* o *"El Juego de la Vida"*, por ejemplo, simula la existencia de diferentes "formas de vida" sobre un espacio bidimensional, las cuales presentan singular comportamiento a través del tiempo; *"Evolución"* es un autómata que simula cómo un conjunto de microbios sobreviven comiendo bacterias; y *"Mayoría Alineada"* muestra cómo es el comportamiento de la tensión superficial entre líquidos no permeables. A continuación, algunos ejemplos de *Autómatas Celulares*.(GONZALES 1999)

- *"Life"* o *"El Juego de la Vida"*, de John Hourton Conway.
- *"Mayoría Alineada"*. Modelo Celular de Dedwdney.
- *"Evolución"*. Modelo Celular de Dewdney.
- *"Reacción Química de Belousov-Zhabotinski"*. Modelo Celular de Dewdney.
- *"HPP-GAS"* (modelo de dinámica de fluidos), de Hardy, de Pazzis y Pomeau

1.7 Implementación de un autómata celular. Descripción

El "**juego de la vida"** es el mejor ejemplo de un ***autómata celular***, diseñado por el matemático británico **John Horton Conway** en 1970.

Descripción

El “juego de la vida” es en realidad un juego de cero jugadores, lo que quiere decir que su evolución está determinada por el estado inicial y no necesita ninguna entrada de datos posterior. El "tablero de juego" es una malla formada por cuadrados ("células") que se extiende por el infinito en todas las direcciones. Cada célula tiene 8 células vecinas, que son las que están próximas a ella, incluso en las diagonales. Las células tienen dos estados: están "vivas" o "muertas" (o "encendidas" y "apagadas"). El estado de la malla evoluciona a lo largo de unidades de tiempo discretas (se podría decir que por turnos). El estado de todas las células se tiene en cuenta para calcular el estado de las mismas al turno siguiente. Todas las células se actualizan simultáneamente. (WIKIPEDIA)

Las transiciones dependen del número de células vecinas vivas:

- Una célula muerta con exactamente 3 células vecinas vivas "nace" (al turno siguiente estará viva).
- Una célula viva con 2 o 3 células vecinas vivas sigue viva, en otro caso muere o permanece muerta(por "soledad" o "superpoblación")

"El Juego de la Vida"

Teselación: Cuadrícula homogénea.

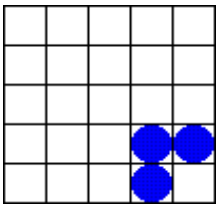
Estados: Dos estados:

- *Vivo* (azul)
- *Muerto* (blanco)

"Life"

Estado Inicial: Configuración aleatoria.

Vecindad: Cada célula tiene 8 células vecinas, que son las que están próximas a ella, incluso en las diagonales.



Modelo celular de Conway

Regla de Evolución: Una célula muerta con exactamente 3 células vecinas vivas "nace" (al turno siguiente estará viva).

Una célula viva con 2 o 3 células vecinas vivas sigue viva, en otro caso muere o permanece muerta(por "soledad" o "superpoblación")

Este autómata celular esta modelado en dos dimensiones, cada célula admite dos posibles estados: ***0*** *= célula muerta y* ***1*** *=célula viva. La evolución de los estados de las células ocurre de manera simultánea; de manera que la evolución del estado de cada célula depende del estado de ella y de los estados de sus*

vecinas inmediatas. La siguiente figura muestra la célula ***c*** *y las células vecinas inmediatas que denotamos por* ***v.***

$$\begin{matrix} v & v & v \\ v & c & v \\ v & v & v \end{matrix}$$

La ley que gobierna este juego es como sigue:

Si una célula muerta tiene exactamente tres vecinas vivas, estas se reproducen y le dan vida.

Si una célula viva tiene dos o tres vecinas vivas, ella permanece viva; pero si hay menos de dos vecinas vivas, muere por aislamiento. Si una célula tiene más de tres vecinas vivas, muere por superpoblación si estaba viva, y permanece muerta si ya lo estaba.

La figura a continuación muestra como las células vecinas se reproducen para darle vida a la célula **c** que estaba muerta.

$$\begin{matrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{matrix} \longrightarrow \begin{matrix} ? & ? & ? \\ ? & 1 & ? \\ ? & ? & ? \end{matrix}$$

Los signos de interrogación que aparecen en los lugares ocupados por las vecinas de c indican la imposibilidad de asignarles un valor, no se conocen (en este caso) los estados de las células en sus respectivas vecindades inmediatas.

En **conclusión**, un *Autómata Celular* está compuesto por:

- Una *Teselación Homogénea*.
- Un conjunto finito de *Estados* para cada celda.
- Una *Vecindad* para cada celda.
- Una *Regla de Evolución o´ transición.*

1.8 Conclusiones

Luego de un estudio realizado acerca de los algoritmos matemáticos utilizados en la simulación, se ha llegado a la conclusión de escoger el modelo matemático **Autómata Celular**, para el proceso de simular las nubes.

El interés que ha despertado esta técnica radica en la sencillez y en la simplicidad que caracteriza la construcción de los modelos; además, en la particularidad de los patrones de comportamiento presentados por el *Autómata* en tiempo de ejecución.

Refleja el proceso físico de la formación de nubes en parte, y es de cómputo más eficiente y más fácil poner en ejecución que los métodos físicos anteriores de la simulación. Mediante este método se pueden crear escenas nubladas más realistas que los métodos físicos estudiados anteriormente.

Es una técnica simple y eficaz para la animación realista de las nubes, aunque no es una simulación física sino heurística; debido a que las simulaciones físicas cumplen con ciertas leyes de la física. Las simulaciones heurísticas proporcionan métodos para obtener soluciones aproximadas de problemas reales, en un tiempo razonable. Es un Autómata Celular se generan comportamientos complejos a partir de reglas muy sencillas.

CAPÍTULO 2 Tecnología utilizada

2.1 Introducción

En este capítulo se hace mención de la técnica utilizada en el proceso de *simulación* y *rendering* de las nubes.

No se puede desconocer que la presencia de la computadora en nuestra vida es hoy una realidad palpable e innegable. Está presente en casi todos nuestros actos diarios, sean éstos de información, de trabajo, de esparcimiento o de estudio.

La razón de esa presencia es su capacidad para adquirir, procesar y almacenar información y de la posibilidad de combinar esa capacidad con la de los medios de comunicación, conformando redes que potencian aún más sus capacidades.

2.2 Recursos Tecnológicos Adoptados

Para el desarrollo de este trabajo se ha hecho uso de variada tecnología. Se ha utilizado un laboratorio de informática que cuenta con una apreciable cantidad de computadoras, conectadas en red y con acceso a Internet.

Se ha conformado un grupo o comunidad virtual por parte de la universidad, conocida como "**Teleformación**", esta plataforma de aprendizaje ha sido útil para el desarrollo del trabajo de diploma, la misma presta los siguientes servicios:

- Dispone de un medio centralizado de intercambio de mensajes, consultas y opiniones entre docentes y alumnos y entre estos últimos, utilizando el correo electrónico, para informar sobre algún tema determinado.
- Centralizar la exposición de apuntes, lecciones, artículos, trabajos y demás material bibliográfico digitalizado.
- Plantear los trabajos y recibir las respuestas de los alumnos con un plazo de entrega.

- Mantener un sitio con los documentos necesarios para desarrollar el trabajo de diploma.
- La plataforma permite generar y mantener una base da datos, no modificable por los alumnos, donde se comprueban los conocimientos y al mismo se tiempo se evalúan, permitiendo la publicación de las respuestas.

Debido a los avances de la tecnología, cada vez existen más posibilidades de actualización con respecto a la educación. En el área de la informática, específicamente, esto ocurre con mayor frecuencia.

2.3 Sistema Operativo

Un **sistema operativo** (SO) es un conjunto de programas destinados a permitir la comunicación del usuario con un ordenador y gestionar sus recursos de manera eficiente. Comienza a trabajar cuando se enciende el ordenador, y gestiona el hardware de la máquina desde los niveles más básicos.

Un sistema operativo se puede encontrar normalmente en la mayoría de los aparatos electrónicos que podamos utilizar sin necesidad de estar conectados a un ordenador y que utilicen microprocesadores para funcionar, ya que gracias a estos podemos entender la máquina y que ésta cumpla con sus funciones (teléfonos móviles, reproductores de DVD, y computadoras). (WIKIPEDIA 2007c)

Microsoft Windows es un sistema operativo (SO) gráfico para computadoras personales cuyo propietario es la empresa Microsoft. Actualmente, este sistema cuenta con diferentes versiones:

- Windows NT
- Windows NT 3.1
- Windows NT 3.5/3.51
- Windows NT 4.0
- Windows 95

- Windows 98
- Windows 98 Second Edition
- Windows Millenium Edition
- Windows 2000
- Windows XP
- Windows Server 2003
- Windows Vista

De estos sistemas operativos, se escogió el Windows XP, debido al conocimiento que se tiene sobre el mismo, ha sido el SO utilizado durante este tiempo de estudiante. Por su efectividad y los beneficios que brinda ha sido el ideal para este trabajo.

Windows XP

Windows XP usa el núcleo de Windows NT. Incorpora una nueva interfaz y hace alarde de mayores capacidades multimedia. Además dispone de otras novedades como la multitarea mejorada, soporte para redes inalámbricas y asistencia remota. (WIKIPEDIA 2007b)

2.4 Herramienta utilizada en la programación

En la programación de la *simulation* y *rendering* de las nubes se hizo uso de la herramienta **Visual Studio .NET 2003**.

Visual Studio .NET es un entorno de desarrollo integrado (IDE) desarrollado por Microsoft a partir del año 2002. Es para el sistema operativo Microsoft Windows. Es un programa compuesto por un conjunto de herramientas para un programador. Puede dedicarse a un sólo *lenguaje de programación* o bien, poder utilizarse para varios.

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

Como se muestra en la figura #10 los IDEs proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación tales como Visual C++, Visual Basic, Visual C#, Visual J#, Object Pascal.

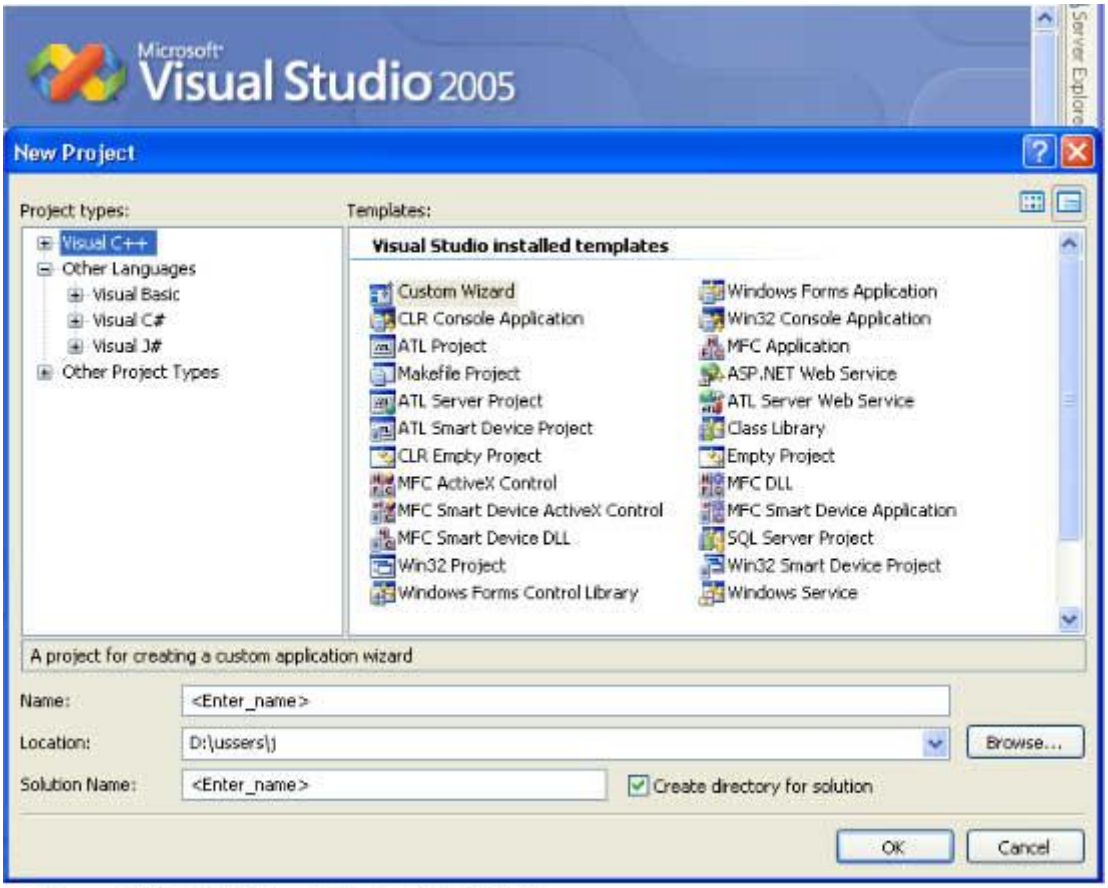


Figura #10 IDE Visual Studio 2005. NET

2.4.1 Componentes de Visual Studio.NET

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

2.4.2 Ejemplos de IDE

Algunos ejemplos de *entornos integrados de desarrollo* (IDE) son:

- Anjuta (GTK, pensado para escritório GNOME).
- C++Builder y TurboC++ Explorer de Borland (ahora de CodeGear)
- Delphi y Turbo Delphi de Borland
- Eclipse
- JBuilder de Borland
- JDeveloper de Oracle_Corporation
- KDevelop (QT, pensado para escritorio KDE)
- MS Visual Studio .NET y Visual Studio Express de Microsoft
- Turbo C y Turbo C++ de Borland

- Turbo Pascal de Borland

De estos ejemplos fue necesario utilizar el Visual Studio.NET, debido a que este entorno contiene el Lenguaje de programación Visual C++, porque es en este lenguaje de programación que se esta desarrollando el simulador, porque es el mas adecuado para trabajar un ambiente gráfico. Porque muchas de las librerías graficas que se encuentran en la Internet son compatible con este entorno de desarrollo.

2.4.3 Lenguaje de Programación Visual C ++

1. **INTRODUCCIÓN - ¿QUE ES VISUAL C++?**

Como sabemos, Windows es el entorno más popular de interfaz gráfico de usuario (GUI). Desde este punto de vista, Windows es un entorno multitarea basado en ventanas, que representan programas, y que permite ejecución concurrente.

Para desarrollar programas, Windows provee una librería de rutinas y funciones (SDK - Kit de desarrollo de software) que permiten gestionar componentes como menús, diálogos, ventanas, etc.

VISUAL C++ es un entorno integrado de desarrollo (IDE) que permite la programación orientada a objetos (POO). Al ser un entorno integrado VISUAL C++ incluye, entre otras, las siguientes herramientas de desarrollo:

- Editor de texto
- Compilador/Enlazador
- Depurador
- Visor de datos y dependencias (Browser)

CONCEPTOS PRELIMINARES

1. ¿Que es C ++?

Como todos sabemos, "C" es un lenguaje de alto nivel, basado en funciones, que permite desarrollos estructurados. Entre otras muchas características contempla la definición de estructuras de datos, recursividad o indirecciones a datos o código (punteros).

"C ++", por su parte, es un superconjunto de "C", al que recubre con una capa de soporte a la POO. Permite por tanto la definición, creación y manipulación de objetos.

2. ¿Que es la Programación Orientada a Objetos?

La POO es una nueva filosofía de programación que se basa en la utilización de objetos. El objetivo de la POO es "imponer" una serie de normas de desarrollo que aseguren y faciliten la mantenibilidad y reusabilidad del código.

Objetos. Un objeto es una entidad que tiene unos atributos particulares (datos) y unas formas de operar sobre ellos (los métodos o función miembro). Es decir, un objeto incluye, por una parte una serie de operaciones que definen su comportamiento, y una serie de variables manipuladas por esas funciones que definen su estado. Por ejemplo, una ventana Windows contendrá operaciones como "maximizar" y variables como "ancho" y "alto" de la ventana.

3. Estilos de los ficheros.

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

2.4.4 Construcción de una aplicación básica

Seguiremos los siguientes pasos:

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

2.5 Microsoft Office

Microsoft Office es un paquete de programas diseñados para solucionar necesidades que se presentan diariamente en el trabajo con la computadora. Este software cuenta con varias versiones, por ejemplo, office 2000, office XP, office 2003 y otros. De estos productos ha sido utilizado el office 2003, por la facilidad de uso y los beneficios que brinda en cuanto a su funcionalidad. Este paquete de programas contiene el siguiente software: Microsoft Word, Microsoft PowerPoint, los cuales han sido necesarios para el desarrollo de este trabajo. Microsoft Word es el ***procesador de textos***.

Un **procesador de textos** es un programa informático que nos permite editar, dar formato, grabar y modificar documentos escritos en nuestro computador. Es el actual sustituto de las máquinas de escribir, aunque con mayor capacidad, ya que pueden incluirse imágenes y mezclar otros datos. También son conocidos como *procesadores de palabras (permiten la corrección de las mismas).*

2.5.1 Otros ejemplos de Procesadores de textos

- StarOffice Writer
- WordPerfect

Software libre

- AbiWord
- Kword
- EZ Word

De estos procesadores de textos se escogió el Microsoft Word 2003, por ser compatible con el sistema operativo utilizado (Microsoft Windows) .Ambos software fueron elaborados por la misma compañía, llamada **Microsoft.**

Microsoft Word

Actualmente, este procesador cuenta con varias versiones. La primera versión de Word salió para MS-DOS a finales de 1983. No tuvo mucho éxito, y las ventas fueron muy inferiores a los de productos de la competencia como WordPerfect.

Con el tiempo han surgido distintas versiones de Microsoft Word hasta llegar a la versión Microsoft Word 2003, esta les brinda muy buenas posibilidades a los usuarios de crear documentos de forma sencilla y fácil. Por las características que brinda este software, ha motivado a hacer uso de él, para la realización de este trabajo de diploma.

2.6 Herramienta Rational Rose

Rational Rose es un programa informático y la vez una herramienta que permite modelar el Lenguaje Unificado de Modelado (UML). UML es un lenguaje para especificar, construir, visualizar y documentar los artefactos de un sistema de software orientado a objetos (OO). Un artefacto es una información que es utilizada o producida mediante un proceso de desarrollo de software. (MARTÍNEZ 1997).

2.6.1 UML y sus Diagramas

UML contiene un conjunto de notaciones y diagramas estándar para modelar sistemas orientados a objetos, y describe la idea esencial de lo que estos diagramas y símbolos significan. (*Modelado de Sistemas con UML*)

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

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

Para el desarrollo de la ingeniería de software de este trabajo, se estableció la metodología RUP. Una metodología define Quién, Qué, Cuándo y Cómo hacer un proyecto. Esta metodología utiliza para su desarrollo el lenguaje UML.

Se hizo uso de la herramienta Rational Rose por su compatibilidad con el sistema, por su claridad a la hora de representar los artefactos, por la organización en que agrupa los eventos. Permite al usuario facilidad de uso.

2.7 Conclusiones

En este capítulo se han enunciado las características de la tecnología que sustenta el trabajo que se realiza. Las nuevas Tecnologías de la Información y las Comunicaciones (TIC), el Entorno de Desarrollo Integrado Visual Studio.NET 2005, el cual contiene el Lenguaje de Programación Orientado a Objeto Visual C++, el Lenguaje Unificado de Modelado (UML), el Proceso Unificado de Desarrollo, el Proceso Unificado de Rational (RUP), forman las bases para el desarrollo de este trabajo.

CAPÍTULO 3. Simulación y Representación de las Nubes

3.1 Introducción

Este capítulo introduce la implementación del *Automata Celular* para simular las nubes, este proceso se hace invisible a la vista del usuario, y se necesita mostrar en la pantalla del ordenador el volumen simulado, a este evento se le denomina *rendering.*

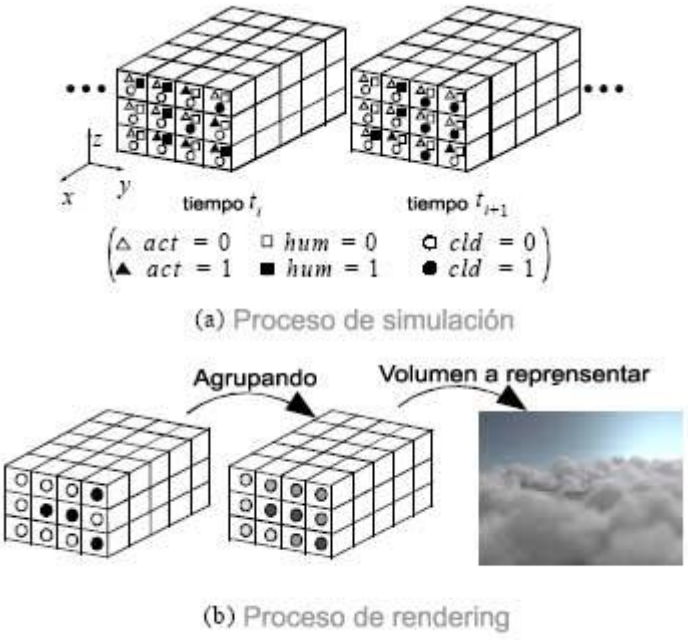


Figura #11 Descripción del método

Autómata Celular es un método eficiente en la simulación. Particularmente, el método es conveniente para la animación realista de nubes.

3.2 Lógica del método Autómata Celular

Simula el comportamiento de las nubes como sucede realmente. Se supone un terreno, sobre el cual se encuentra burbujas de aire caliente, a medida que este aire asciende se va enfriando, de aquí se define el

estado de *humedad* (hum). Luego el aire se condensa dando lugar al estado de *transición* o' *activación* (act), y como resultado final se llega al estado de *nube* (cld).

Este proceso tiene las características siguientes:

- El método de simulación (autómata celular) crea el movimiento realista de la nube con pequeña cantidad de cómputo. Puede simular el movimiento con operaciones boleanas simples.

- El método de representación (metaball) realiza un rápido procesamiento de imágenes. Puede calcular rápidamente sombras y posiciones de las nubes, permite proporcionar el color a la nube, logrando una mejor visualización del entorno gráfico.

En la figura #11 se muestra el método de trabajo para llegar a representar las nubes. Consiste en dos procesos, simulación y representación.

3.3 Simulación de las nubes utilizando un Automata Celular

El espacio de la simulación es un cubo (clase "AutomataNube") dividido en celdas (clase "TEstado") y se realiza en 3 dimensiones (dimensión X, dimensión Y, dimensión Z), figura #11. La posición de cada celda se expresa por las variables ***i, j, k***, donde ***i*** es la posición en el eje X, j es la posición en el eje Y, ***k*** es la posición en el eje Z. Las celdas corresponden a las células usadas en el autómata celular. Este algoritmo matemático presenta una estructura, la cual veremos a continuación.

3.3.1 Estados del Autómata Celular

En cada célula hay tres variables lógicas, la **humedad** (hum), las **nubes** (cld), y la **fase de transición o' activación** (act). Cada variable presenta dos *estados* que pueden ser **0** o' **1**. Cuando un estado está en **0** significa que está *inactivo* y cuando está en 1 significa que está *activo, figura #12.*

3.3.2 Vecindad para las células del Autómata

La vecindad se va a tener en cuenta solo en el estado de activación (**act**). Para la célula (**c)** que se analiza se tomaron dos células vecinas (**v)**, en todas las direcciones, teniendo así 11 células vecinas, excepto en la dirección vertical que se tomo solo una, debido a que las nubes evolucionan de abajo hacia arriba, como se explica en el epígrafe 3.1.

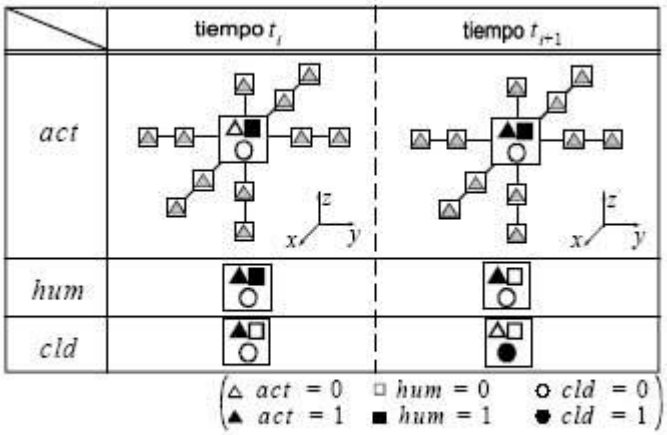


Figura #12 Reglas de transición

3.3.3 Reglas de transición

La evolución de la nube se simula aplicando simples reglas de transición, que son comunes para todas las celdas o' células, esto se hace en iteraciones consecutivas cada cierto intervalo de tiempo. Las reglas de transición representan la formación de las nubes y su extinción. Las reglas se expresan por operaciones boleanas. Las variables utilizadas se almacenan en una misma celda para ahorrar el coste de la memoria. A continuación se observan las reglas de transición declaradas para el autómata. Aquí se encuentran las variables **i, j**, **k** las cuales representan el movimiento en los tres ejes de coordenadas dentro de la matriz que va a simular el autómata.

$$hum(i,j,k,t_{i+1}) = hum(i,j,k,t_i) \wedge \neg act(i,j,k,t_i)\,, \quad (1)$$

$$cld(i,j,k,t_{i+1}) = cld(i,j,k,t_i) \vee act(i,j,k,t_i)\,, \quad (2)$$

$$act(i,j,k,t_{i+1}) = \neg act(i,j,k,t_i) \wedge hum(i,j,k,t_i) \wedge f_{act}(i,j,k)\,, \quad (3)$$

$hum(i,j,k,t_i)$: Representa el estado de humedad en la posición i, j, k de la matriz para un tiempo inicial, es decir, la primera iteración.

$hum(i,j,k,t_{i+1})$: Representa el estado de humedad en la misma posición de la matriz hablada anterior pero en un tiempo después, es decir, en la siguiente iteración (repetición)

Esta situación va a ocurrir con todos los estados del autómata, recordar que cada celda de la matriz tridimensional (implementada en 3 dimensiones), va a contener todos los **estados**.

Descripción de las reglas

Regla # 1 Existirá humedad en la siguiente iteración, si el estado de la celda en la iteración anterior estaba en humedad y no estaba en estado de transición ó activación.

Regla # 2 Existirá nube en la siguiente iteración, si el estado de la celda en la iteración anterior estaba en nube o' estaba en estado de transición ó activación.

Regla # 3 Existirá activación o' transición en la siguiente iteración, si el estado de la celda en la iteración anterior no estaba en activación y estaba en estado de humedad y estaba activa la función de activación.
En este caso se toma en cuenta la vecindad de cada celda.

$f_{act}(i, j, k)$ es la función de activación o' transición, es de tipo boleana (devuelve verdadero o' falso) y su valor es calculado a partir del estado de las células vecinas a la celda que esta en la posición i, j, k en

la iteración anterior. Se utiliza considerando el hecho de que las nubes crecen hacia arriba y horizontalmente.

$$
\begin{aligned}
f_{act}(i,j,k) = {} & act(i+1,j,k,t_i) \vee act(i,j+1,k,t_i) \\
& \vee act(i,j,k+1,t_i) \vee act(i-1,j,k,t_i) \vee act(i,j-1,k,t_i) \\
& \vee act(i,j,k-1,t_i) \vee act(i-2,j,k,t_i) \vee act(i+2,j,k,t_i) \\
& \vee act(i,j-2,k,t_i) \vee act(i,j+2,k,t_i) \vee act(i,j,k-2,t_i).
\end{aligned} \tag{4}
$$

Esta función tiene como responsabilidad, buscar cada célula vecina (2 células en cada dirección) y verificar si al menos una está en estado de activación (act = 1).

A continuación aparece implementada la función transición.

```
bool AutomataNube::fTransicion(int x, int y, int z)
{
	bool f0=false,f1=false,f2=false,f3=false,f4=false,f5=false,f6=false,f7=false,f8=false,f9=false,f10=false,f11=false;
	if (x+1<dimensionX)   // Verifico que la celda vecina de X a la derecha  se encuentre dentro de la dimensión del cubo
		f0 = nube[x+1][y][z].transicion;
	if (y+1<dimensionY)                    Verifico lo mismo pero para el eje Y
		f2 = nube[x][y+1][z].transicion;
	if (z+1<dimensionZ)
		f3 = nube[x][y][z+1].transicion;  //Se verifican todas las celdas del cubo en la dimension Z  para saber si esta en transición
	if (x-1>=0)
		f4 = nube[x-1][y][z].transicion;
	if (y-1>=0)
		f5 = nube[x][y-1][z].transicion;
	if (z-1>=0)
		f6 = nube[x][y][z-1].transicion;
	if (x-2>=0)
		f7 = nube[x-2][y][z].transicion;
	if (y-2>=0)
		f8 = nube[x][y-2][z].transicion;
	if (z-2>=0)
		f9 = nube[x][y][z-2].transicion;
	if (x+2<dimensionX)
		f10 = nube[x+2][y][z].transicion;
	if (y+2<dimensionY)
		f11 = nube[x][y+2][z].transicion;
	bool rtn = f0||f1||f2||f3||f4||f5||f6||f7||f8||f9||f10||f11;   // La función devuelve verdadero si al menos un valor está en transición
	return rtn;
}
```

Al implementarse el Automata Celular partiendo de las reglas (1, 2, 3), se presenta un problema en el logro de los resultados; pues los estados de las variables se van mantener estáticos, es decir, si en una celda hay una nube nunca dejará de existir y si no hay nube nunca se obtendrá. El resultado de esto sería una imagen fija. Ante el problema se tomaron las siguientes soluciones

Dinámica de las nubes

Para lograr nubes dinámicas se hizo el razonamiento representado en las reglas 5,6 y 7.

- La regla 5 plantea que existirá una *nube* en la iteración siguiente – si en la iteración anterior existe una nube **y** se cumple la función **IS (e)**.

- La regla 6 plantea que existirá *humedad* en la iteración siguiente – si en la iteración anterior existe humedad **o'** se cumple la función **IS (e)**.

- La regla 7 plantea que existirá un *estado de activación* en la iteración siguiente – si en la iteración anterior existe una nube y se cumple la función **IS (e)**.

$$cld(i,j,k,t_{i+1}) = cld(i,j,k,t_i) \wedge \mathrm{IS}(rnd > p_{ext}(i,j,k,t_i)), \quad (5)$$
$$hum(i,j,k,t_{i+1}) = hum(i,j,k,t_i) \vee \mathrm{IS}(rnd < p_{hum}(i,j,k,t_i)), \quad (6)$$
$$act(i,j,k,t_{i+1}) = act(i,j,k,t_i) \vee \mathrm{IS}(rnd < p_{act}(i,j,k,t_i)), \quad (7)$$

IS (*e*): función boleana que retorna **verdadero** si la expresión ***e*** se cumple y si no se cumple retorna falso. Más adelante aparece implementada.

Rnd: random de números aleatorios, es una función que el lenguaje de programación trae implementada. La función ***rand()*** retorna un número entero seudo-aleatorio que puede estar en el intervalo de 0 a RAND_MAX que puede ser de al menos 32767(DAVIDSON 2001). Se tiene por conocimiento que el mayor valor posible es de 65535. Este procedimiento se utiliza para resolver el problema que se presentaba a la hora de simular con las reglas 1, 2 y 3 solamente, recordar que el problema consistía en que después de iterar los *estados* del autómata permanecían estáticos.

Para lograr la dinámica de las nubes se implementó la función **IS (e)**
como **bool AutomataNube::randFunction(double prob)**, **bool** representa el tipo de dato que retorna la función (verdadero o' falso), **AutomataNube** es el nombre de la clase a la cual pertenece la función, y **ranFunction** es el nombre de la propia función la cual se le pasa un valor de probabilidad.

Se tomó en cuenta una probabilidad para cada *estado*: probabilidad de extinción (p_{ext}), probabilidad de humedad (p_{hum}) y probabilidad de activación (p_{act}), esta probabilidad es introducida por el usuario y va a tener un valor que va desde **0** a **1**. Finalmente para darle solución al problema presentado se compara el número generado por la función ***rand()*** para ver si es menor que la probabilidad establecida, para luego establecer nuevos comportamientos, pero como los valores devueltos por la función van a ser números muy grandes entonces fue necesario llevar esos valores a una escala que oscile entre 0 y 1. A continuación se muestra la implementación del procedimiento:

```
//comprobar que se genere un numero aleatorio que sea menor que la probabilidad (prob)
bool AutomataNube::randFunction(double prob)
{
        int hi = rand()%10000;
        int lo = rand()%10000;
        unsigned int number = hi*10000 + lo;
        double rnd = (double)number/(double)100000000;
        return (rnd<prob);
}
```

Implementación de las reglas de transición (5, 6,7)

Se declaró la estructura "**TEstado**" la cual va a contener los estados **humedad** (hum**)**, **activación** (act**)** y **nube** (nube).Para ir simulando y pasando de una iteración a otra, se hizo necesario declarar dos matrices tridimensionales (dimensión X, dimensión Y, dimensión Z). En programación, las matrices constituyen un tipo de dato abstracto (TDA). Puede afirmarse que las matrices son un recurso de programación conocido como estructuras de datos, donde todos los elementos van a ser del mismo tipo.

Las matrices declaradas fueron las siguientes:

TEstado***nube (Matriz que va a contener los valores de la primera iteración)

TEstado***temp (Esta matriz va a recibir los valores de la matriz nube, cuando ella vaya a hacer la próxima iteración, esto se hace para no perder los valores que van a ser utilizados consecutivamente.)

A continuación aparece la implementación de las reglas 5, 6 y 7 ya anteriormente explicadas.

```
temp[x][y][z].nube = (nube[x][y][z].nube || nube[x][y][z].transicion) && !randFunction(pExt);
temp[x][y][z].transicion = (!nube[x][y][z].transicion && nube[x][y][z].humedad && fTransicion(x,y,z))  || randFunction(pTrs);
temp[x][y][z].humedad = (nube[x][y][z].humedad && (!nube[x][y][z].transicion)) || randFunction(pHum);
```

3.4 Rendering de las nubes

Para el proceso de *rendering* se hizo uso de la técnica **Metaball,** es el nombre de una técnica de gráficos realizada por ordenador para simular interacción orgánica entre diferentes objetos n-dimensionales y fue inventado por Jim Blinn a principios de los años 1980.(WIKIPEDIA 2007a), véase la fig #13.

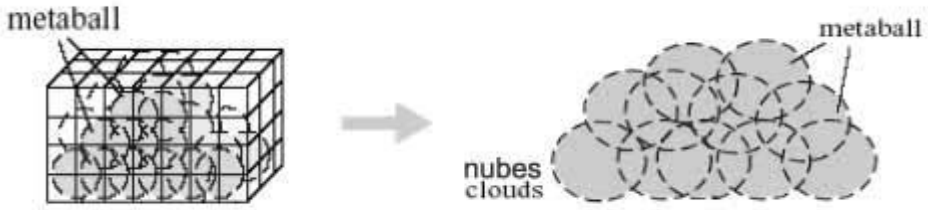


Figura #13 Rendering de las nubes utilizando Metaball

La utilización de esta técnica es necesaria para representar el proceso de simulación de las nubes en la pantalla (**rendering**); debido a que se hace complicado mandar a pintar cada célula o celdas del autómata, esto tomaría mucho tiempo de procesamiento y haría muy lento la visualización en pantalla de las nubes, pues se verían en forma de imágenes y no tendrían semejanzas a la realidad.

Metaball consiste en agrupar dentro esferas un conjunto de objetos ya simulados, en este caso agruparía un conjunto de celdas, en las cuales se encuentran las nubes, y luego se procede a pintar cada metaball que puede tener el automata nube, figura #13.

En el autómata se encuentran definidos varios metaball, y en cada metaball encontramos varias nubes.

3.4.1 Diseño de la aplicación

El trabajo realizado se considera una **aplicación** y no un **sistema de software**, pues un sistema de *software* es un conjunto de programas, instrucciones y reglas informáticas para ejecutar ciertas tareas en una computadora. Y una *aplicación* es un programa preparado para una utilización específica. En este caso se hizo para probar el método **Autómata Celular** y comprobar su visualización en la pantalla.
Lo que se pretende es validar esta técnica para luego ser incorporada a sistemas muchos más grandes, por ejemplo, el simulador de tiro que se desarrolla en nuestra Universidad.

No se aplicó una metodología de ciclo completo, solo se explicará el diseño de las clases utilizadas como se muestra en la figura #14.

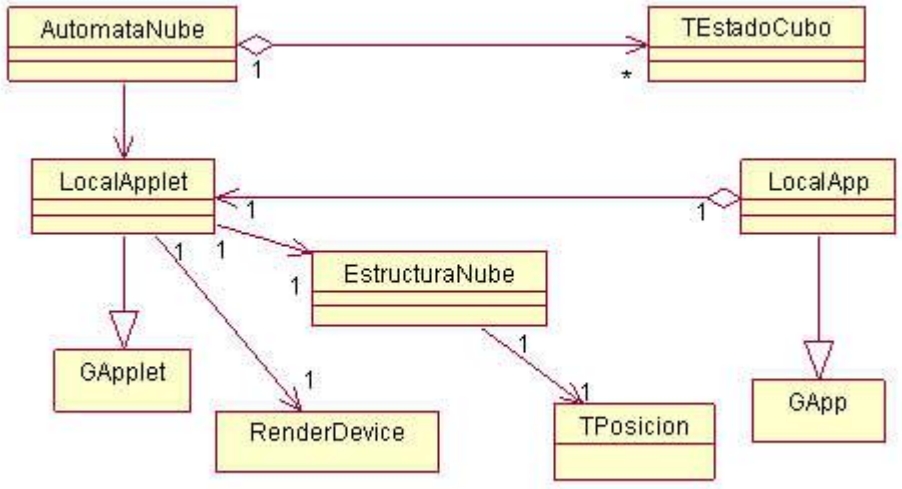
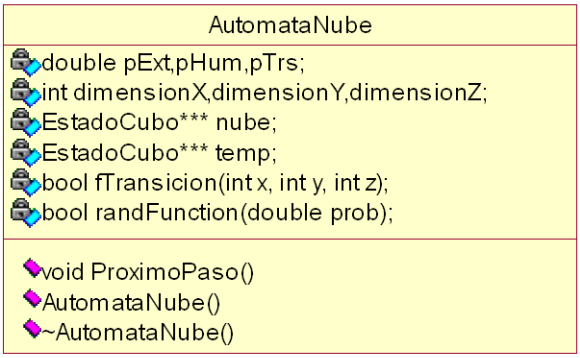


Figura #14 Diagrama de clases del diseño

Descripción del diagrama de clases del diseño

La clase "**AutomataNube**"es la responsable de realizar la simulación de las nubes, contiene **6 *atributos*** *y* **3 *funciones***; ellas son "*AutomataNube* ", "*ProximoPaso" y* "~AutomataNube".

La función *"AutomataNube*" es el constructor de la clase. En la programación OO cada clase tiene un constructor, aquí es donde se inicializan los valores que van a tener los atributos.



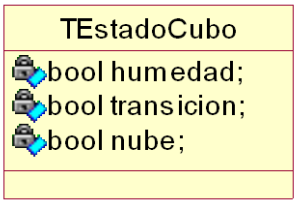
La función "*ProximoPaso"*, es la que va a determinar la situación del automata en la siguiente iteración (repetición). EL estado de cada célula en la iteración siguiente (tiempo t_{i+1}) depende del estado de las células en la iteración anterior (tiempo t_i). Para lograr este funcionamiento se declararon 2 *matrices tridimensionales* como atributo, siendo ellas: TEstadoCubo***nube** y TEstadocubo***temp**. La matriz **nube** va a contener el estado de cada celda del automata en la primera iteración y luego se intercambia la información para la matriz **temp** (temporal), entonces la matriz **nube** analizará el estado de las células de la siguiente iteración a partir de la matriz temporal y así se hará consecutivamente.

La función "**fTransicion"** es la responsable de verificar para una célula (c) si alguna de las 11 células vecinas (v) tiene activo el estado de *transición* y el resultado que devuelve es verdadero o' falso.

La función **"~AutomataNube**" es el destructor de la clase. Su objetivo es realizar operaciones como liberación de memoria.

Clase TEstadoCubo

La clase TestadoCubo tiene la responsabilidad de contener los estados de cada célula del *autómata celular,* que van a ser los atributos: humedad, transición y nube .Cada estado puede tener dos valores posible **1** si está activo y **0** si está inactivo.

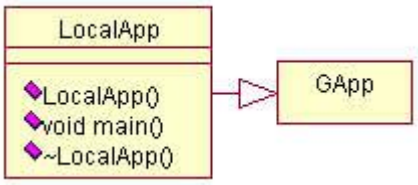


El autómata nube va a contener muchos objetos de tipo TEstadoCubo. Estos atributos son de tipo bool, contienen información que puede ser **true (verdadero)** o' **false (falso)**, estos tipos de datos vienen ya definido por el lenguaje de programación.

Esta clase no tiene funciones. Es declarada para ser usada por la clase "*AutomataNube*".

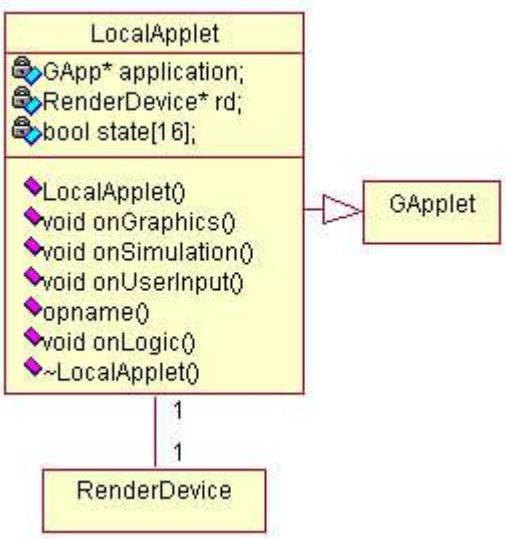
Las clases analizadas anteriormente ("AutomataNube" y "TEstadoCubo") son las que se utilizan en el proceso de simulación. Para desarrollar el rendering de las nubes, fue necesaria la utilización de las siguientes clases.

Clase LocalApp



Esta clase pertenece a la parte gráfica del trabajo, tiene la responsabilidad de controlar la aplicación. Hereda funciones de la clase "***Gapp***" ya implementadas en el lenguaje de programación. Esta clase incluye a la clase "***LocalApplet***"

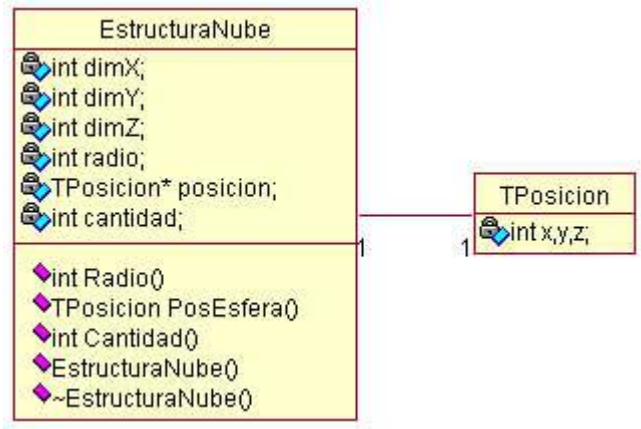
Clase LocalApplet



Esta clase es la responsable de controlar los gráficos, hereda funciones de la clase "***GApplet***". LocalApplet utiliza un objeto "***RenderDevice"***; la cual contiene el entorno gráfico de la librería *OpenGL***,** también se le conoce como motor grafico. También tiene relación de *uno* a *uno* con la clase "***AutomataNube"*** la cual se encarga de la simulación de las nubes.

La función *"onGraphics*" es un evento de G3D para pintar. Se ejecuta en un ciclo infinito.
La funcion "*onSimulation*" es un evento de G3D para simular el proceso. Se ejecuta en un ciclio infinito después del "onGraphics".Las demás funciones no se utilizan en el trabajo.

Clase EstructuraNube



Esta clase tiene la responsabilidad de guardar la estructura de la nube y saber donde están los metaball. Recordar que un metaball tiene forma de esfera y va a agrupar cantidades de nubes para evitar tener que pintar nube a nube, entonces se pintarían los metaball. Esta clase tiene relación de *uno* a *uno* con la clase "***TPosicion".***

Nubes obtenidas

Se decidió no ponerles luces a las nubes, debido a que esta propiedad toma mucho tiempo de procesamiento para el ordenador. Y el objetivo que se quiere es lograr las nubes considerando las restricciones del Hardware. La aplicación permite cambiar la densidad de las nubes, así como la velocidad de transformación.
A continuación se muestran las imágenes de las nubes logradas en la aplicación, una menos densa que la otra,

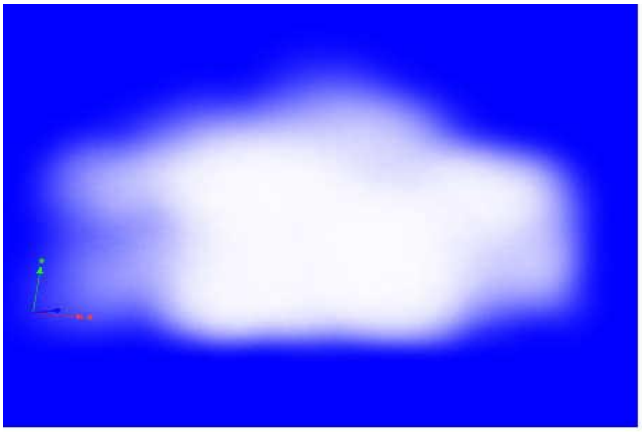


Figura #15 Nube obtenida de la simulación utilizando un Autómata Celular

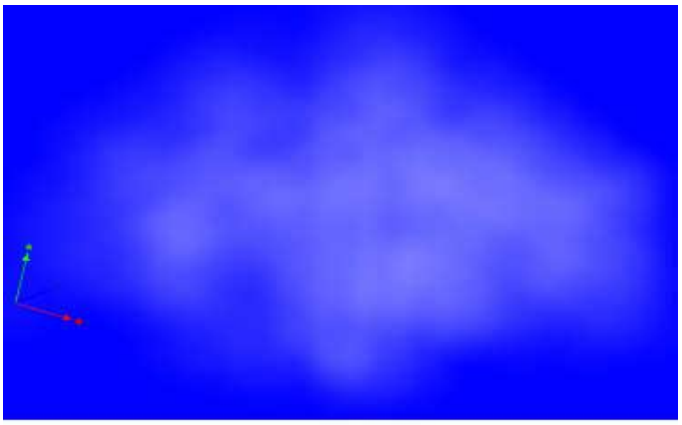


Figura #15 Nube con menos densidad

3.5 Conclusiones

En este capitulo se ha declarado la estructura de un Automata Celular implementado para la obtención de las nubes. A este modelo se le hizo el aporte de definir toda su estructura (estados, vecindad, reglas de transición). Fue necesaria la utilización de la técnica Metaball, esta permite representar la interacción de objetos en el espacio (1.2.3…n dimensiones).

Se hizo una descripción de las clases necesarias para la implementación. Se obtuvo el diagrama de clases de diseño de la aplicación. En este capitulo se le da solución a los objetivos trazados para este trabajo de diploma.

Conclusiones Generales

Se comenzó investigando sobre los modelos matemáticos que hicieran posible la simulación de las nubes, luego se escogió el algoritmo matemático llamado “Automata Celular”. Luego se continúo analizando cuales serian las propiedades a utilizar de este algoritmo que hicieran posible la obtención de las nubes. Este modelo matemático solo se encarga de la simulación, proceso que se hace invisible a la vista del usuario. Por lo que se hace necesario representar el volumen simulado, a este proceso se le denomina rendering.

Para obtener la visualización de las nubes en la pantalla, se hizo necesaria la utilización de la técnica Metaball. La cual se encarga de agrupar en esferas cantidades de nubes obtenidas en la simulación. Esto se hace con el objetivo de que el trabajo sea realizado más rápido, utilizando menor tiempo de procesamiento.

El uso de la tecnología para este trabajo de diploma, permitió probar que a través de un modelo matemático (Autómata Celular) era posible la obtención de nubes. Esto le da una respuesta positiva a la idea a defender que se planteó al inicio de la investigación.

El objetivo de la investigación no se ha cumplido completamente porque aún no se ha probado en el simulador las nubes logradas. Este modelo permite la simulación de las nubes de forma sencilla y eficiente.

Recomendaciones

- Continuar perfeccionando la programación del método para que sea más eficiente.
- Hacer pruebas de rendimiento para verificar con exactitud el tiempo y recursos que la máquina consume.
- Valorar su aplicación en los distintos simuladores de nuestra facultad que necesiten representar nubes.
- Utilizar estándares de codificación (nombre de la clases, arreglos).
- Código en Ingles, para facilitar la publicación del trabajo.

Referencias Bibliográficas

COMPUTACIÓN, A. D. Una Introducción a los Autómatas Celulares, 1999.

DAVIDSON, S. R. *Función rand ANSI C*, 2001 [Disponible en: http://www.conclase.net/c/librerias/funcion.php?fun=rand

GONZALES, L. F. *Una Introducción a los Autómatas Celulares*, 1999. [Disponible en: http://yupana.autonoma.edu.co/publicaciones/yupana/005/autocelular/Automatas.html#7

La Biblia [Job 28:28].

MARTÍNEZ, G. M. *Ingeniería de Software UML*, 1997. [Disponible en: http://www.monografias.com/trabajos5/insof/insof.shtml

Modelado de Sistemas con UML. Disponible en: http://es.tldp.org/Tutoriales/doc-modelado-sistemas-UML/multiple-html/c12.html

MOREA. *Simulación de Sistema*, 2006. [Disponible en: http://www.monografias.com/trabajos20/simulacion-sistemas/simulacion-sistemas.shtml#top

PETROBRAS. *Recursos de Internet para el método finito de la diferencia para PDE*, 2007. [Disponible en: http://translate.google.com/translate?hl=es&sl=en&u=http://math.fullerton.edu/mathews/n2003/finitediffpde/FiniteDifferencePDEBib/Links/FiniteDifferencePDEBib_lnk_1.html&sa=X&oi=translate&resnum=3&ct=result&prev=/search%3Fq%3D%2B%2BFinite%2Bdifference%2Bmethod%26hl%3Des%26lr%3D%26sa%3DG

WIKIMEDIA FOUNDATION, I. Método Multigrid, 2006.

---. *Multigrid Method*, 2007. [Disponible en: http://en.wikipedia.org/wiki/Multigrid

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

WIKIPEDIA. *Juego de la vida*. Disponible en: http://es.wikipedia.org/wiki/Juego_de_la_vida

---. *Metaball*, 2007a. [Disponible en: http://es.wikipedia.org/wiki/Metaball

---. *Microsoft Windows*, 2007b. [Disponible en: http://es.wikipedia.org/wiki/Microsoft_Windows

---. *Navier alimenta ecuaciones*, 2006. [Disponible en: http://translate.google.com/translate?hl=es&sl=en&u=http://en.wikipedia.org/wiki/Navier-Stokes_equations&sa=X&oi=translate&resnum=1&ct=result&prev=/search%3Fq%3DSolve%2BNavier-Stokes%2BEquation%26hl%3Des%26lr%3D

---. *Sistema operativo*, 2007c. [Disponible en: http://es.wikipedia.org/wiki/Sistema_operativo

Glosario de términos

Autómata: Instrumento o aparato que encierra dentro de sí el mecanismo que le imprime determinados movimientos.

Compilador: Es un programa que, a su vez, traduce un programa escrito en un lenguaje de programación a otro lenguaje de programación, generando un programa equivalente. Acelera el proceso de construcción de los programas

Depurador: Es un programa que permite depurar o limpiar errores de otro programa informático

Editor de texto: Resalta las palabras claves que admite el lenguaje de programación.

Gráficos: En informática, es el nombre dado a cualquier imagen generada por un ordenador.

Hardware: Se utiliza generalmente para describir los artefactos físicos de una tecnología

Matrices: Es una Estructura de datos, que se utiliza programación OO.

Modelos matemáticos: son ecuaciones matemáticas, que se encargan de encontrar soluciones analíticas a los problemas, permitiendo la predicción del comportamiento del sistema a partir de parámetros y de condiciones iniciales.

Límite: una función tiene un límite si progresivamente alcanza un número y se define como: $\lim_{x-x0} f(x) = L$.

Ordenador: Es un sistema digital capaz de procesar datos a partir de un grupo de instrucciones denominado programa.

Operador: Es aquel que manipula el funcionamiento del trabajo que se está realizando.

Planeación: La planeación estratégica no trata de tomar decisiones futuras, ya que éstas sólo pueden tomarse en el momento.

Programa: Es la unión de una secuencia de instrucciones que una computadora puede interpretar y ejecutar.

Software: Conjunto de programas, instrucciones y reglas informáticas para ejecutar ciertas tareas en una computadora.

Simulador: Aparato que reproduce el comportamiento de un sistema en determinadas condiciones, aplicado generalmente para el entrenamiento de quienes deben manejar dicho sistema.

Tecnología: Conjunto de teorías y de técnicas que permiten el aprovechamiento práctico del conocimiento científico.

Virtual: En informática, significa 'algo simulado', creado por el ordenador para llevar a cabo determinado fin.