UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

SIMULADOR DE DISPERSION DE CONTAMINANTES POR AUTOMATAS CELULARES

TESIS PROFESIONAL

QUE PARA OBTENER EL TITULO DE:

INGENIERO EN COMPUTACION

P R E S E N T A :

MIGUEL MARIN FLORES

DIRECTOR DE TESIS:
DR. VICTOR M. CASTAÑO MENESES

CIUDAD UNIVERSITARIA 1995

Agradecimientos

A ti Señor.

A mis padres con todo cariño, como reconocimiento a su incansable apoyo y enorme confianza que han depositado en mi.

A mis hermanas Matilde, Isabel, Georgina y Margarita, las quiero a todas.

A ti Claudia, por todo lo que eres y representas para mi, este trabajo no es sólo mío, sino tuyo también.

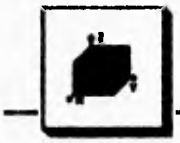
A todos mis familiares, amigos y muy especialmente a mis "compadres" por brindarme su invaluable amistad, gracias:

A mi asesor de tesis Dr. Victor M. Castaño por su infinita paciencia y amistad incondicional.

Miguel.

Índice

Introducción



En años recientes, el estudio de la contaminación ambiental ha tomado gran importancia, debido principalmente, a los efectos nocivos que los diversos contaminantes (del agua, aire, suelo, etc.) causan en las formas de vida de nuestro planeta, siendo afectados, de igual forma, los materiales inorgánicos.

El problema de la contaminación ambiental siempre ha existido, sin embargo, con el crecimiento desmedido de la población y sus altas concentraciones en pequeñas regiones geográficas, el problema aumenta y puede llegar a tomar dimensiones catastróficas si no se atacan a tiempo los diversos y muy variados factores (sociales, naturales, etc.) que causan el incremento alarmante de las concentraciones de los diversos contaminantes en agua, aire, suelo, etc.

La computación se ha convertido en una herramienta indispensable para el trabajo de investigación, especialmente, donde existe un cálculo intensivo de operaciones matemáticas y el estudio de resultados se realiza en forma gráfica. El estudio de los contaminantes ambientales no es la excepción, siendo las simulaciones por computadora (o programas de simulación), las más utilizadas y donde la aplicación de nuevas herramientas computacionales permitirán mejorar los resultados obtenidos.

Contaminación Atmosférica y Autómatas Celulares

Los efectos que sobre el medio ambiente causan los diversos contaminantes suspendidos en la atmósfera son muchos, siendo algunos de ellos, de fuertes repercusiones sobre las diversas formas de vida del planeta, el análisis de estos contaminantes (y de las demás formas de contaminación) es entonces prioritario para la conservación del equilibrio ecológico y de las muchas formas de vida que debido a la contaminación se encuentran en peligro de extinción.

Los estudios realizados sobre los contaminantes suspendidos en la atmósfera abarcan un amplio espectro, algunos estudios se enfocan al análisis de las recombinaciones a nivel molecular de los contaminantes con los componentes naturales de la atmósfera; otros se encargan de analizar, a nivel macroscópico, el comportamiento de estos contaminantes, enfocándose específicamente a los fenómenos de dispersión.

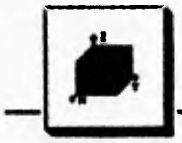
Los estudios de dispersión permiten predecir el comportamiento de estos contaminantes en diferentes situaciones atmosféricas, basándose generalmente, en modelos matemáticos y simulaciones del fenómeno en cuestión. Los modelos matemáticos representan generalmente situaciones específicas del comportamiento de contaminantes en la atmósfera; la complejidad de dichos modelos hace que en la mayoría de los casos se resuelvan únicamente en forma numérica.

Las simulaciones enfocadas a la dispersión de contaminantes, intentan, básicamente, mostrar de una manera aproximada el movimiento de un contaminante específico en la atmósfera, estas simulaciones están basadas en modelos matemáticos no muy complejos.

En años recientes han surgido nuevas herramientas computacionales cuyas características los hacen ideales para describir fenómenos físicos, una de estas herramientas de gran interés son los llamados *Autómatas Celulares* (AC).

El concepto de *Autómata Celular* fue introducido por John Von Neumann para una tarea en particular: proveer la existencia de una computadora universal que se autoreproduzca en un espacio definido. El estudio de los AC ha generado una amplia teoría que ha resultado en la creación de diversos tipos, los cuales son aplicados, de acuerdo a sus características, en fenómenos físicos, biológicos, etc.

Los AC son un tipo de sistemas dinámicos no lineales discretos, construidos por un número grande de autómatas finitos idénticos, donde



cada uno recibe datos de entrada de otro autómata finito siguiendo una regla de vecindad predefinida. Cada autómata finito usa el patrón de entrada y su estado actual para calcular su estado siguiente y así, generar un patrón de salida que enviará a sus vecinos inmediatos. Los autómatas son arreglados típicamente en un patrón celular, de aquí el nombre de autómatas celulares.

Los AC han sido aplicados a diversos problemas caracterizados por tener un comportamiento dinámico importante y una variabilidad espacial relevante, uno de estas aplicaciones se muestra en [Guariso, 1991], donde proponen una formulación especial de AC para simular el comportamiento de un contaminante en la atmósfera. Éste, ha sido tomado como base para el desarrollo del presente trabajo, de acuerdo a los objetivos planteados en la siguiente sección.

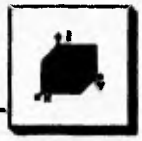
Objetivos Generales

Basándonos en el trabajo de Giorgio Guariso y Vittorio Maniezzo [Guariso, 1991] se establecen los siguientes objetivos generales:

1. Diseñar a partir de los paradigmas fundamentales (conceptos de física, química, computación) un modelo físico de dispersión de contaminantes.

2. Desarrollar un algoritmo a partir de elementos básicos de la teoría de autómatas celulares y del modelo físico propuesto.

3. Implementación del algoritmo en un programa de computadora; usando técnicas de programación orientada a objetos.



4. El programa debe ejecutarse óptimamente en una computadora personal.

5. Sentar las bases de futuros desarrollos en el tema, recalcando el papel que un ingeniero en computación pudiese tener.

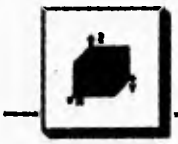
La realización de los puntos anteriores serán desarrollados en forma gradual a lo largo de los capítulos que conforman este trabajo.

Organización de la Tesis

El presente trabajo de tesis está organizado en 6 capítulos y dos apéndices, El capitulo uno es una pequeña introducción al problema de los contaminantes atmosféricos, aquí realizamos una pequeña revisión histórica del tema, desde los problemas originados por la combustión del carbón hasta nuestros días, se realiza así mismo, una pequeña clasificación de los tipos de contaminantes atmosféricos más comunes y finalmente introducimos algunos conceptos básicos de los modelos de dispersión de contaminantes.

En el capitulo dos mencionamos algunas características importantes que presenta la *computación emergente* y los *autómatas celulares*, en este capitulo explicamos exhaustivamente el modelo (basado en autómatas celulares) propuesto por G. Guariso y V. Maniezzo; el desarrollo y análisis del modelo presenta ciertas inconsistencias las cuales nos llevan a deducir un modelo físico alternativo.

El capitulo tres contiene una pequeña introducción del desarrollo de sistemas de software bajo la técnica de Programación Orientada al Objeto



así como las principales consideraciones de desarrollo del programa simulador de dispersión de contaminantes.

En el capitulo cuatro se realizan varias simulaciones para la prueba y comprobación del modelo propuesto y del programa de simulación desarrollado, el objetivo de este capítulo es encontrar los efectos que los parámetros del modelo de simulación tienen sobre el comportamiento del autómata celular.

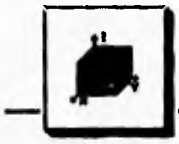
En el capitulo cinco se establece una situación de contaminación basada en estudios realizados por investigadores del Instituto de Física de la UNAM, dicha situación se simula y analiza mediante el programa desarrollado, comparando los resultados respectivos contra los esperados de la situación de dispersión de contaminantes real. Finalmente, el capitulo seis presenta las conclusiones de este trabajo.

El apéndice A aborda la teoría básica de los *autómatas celulares*, explicando de manera sencilla los conceptos básicos de la misma. El apéndice B desarrolla brevemente la teoría general del paradigma de programación conocido como *Programación Orientada al Objeto* explicando sus conceptos más relevantes.

1

Contaminantes Atmosféricos:

Una introducción



México es uno de los países biológicamente hablando, más diversos del mundo. Su variada fauna y flora es similar o mayor a la de otros países. Ésta se encuentra compuesta de aproximadamente 30,000 especies de plantas, 500 especies de mamíferos, 1,000 especies de aves y miles de especies de otros vertebrados e invertebrados. México junto con Colombia, Brasil, Indonesia, Zaire, Australia y Madagascar, albergan entre el 60% y 80% de todos los organismos vivos del planeta.

Desgraciadamente existen serios problemas ambientales que ponen en peligro a cientos de especies vegetales y animales. La deforestación de miles de hectáreas de selvas y bosques, la desecación de pantanos, humedales, esteros y otros cuerpos de agua, la contaminación del suelo, aire, agua, la desaparición de miles de especies por la destrucción de su hábitat, el tráfico y cacería ilegales, etc. son algunas de las consecuencias del avance desenfrenado de las fronteras urbana, agrícola y ganadera.

Para citar un ejemplo, las selvas de la región mexicana conocida como *El Bajío* que ocupaba una área de aproximadamente 20,000 Km. cuadrados ha desaparecido en más del 95% de su extensión original.

Entre los estudios que se encargan del estudio de la contaminación, se encuentran los relacionados con los fenómenos atmosféricos y en especial los enfocados al de la calidad del aire, los cuales han despertado de mucho tiempo atrás el interés de la comunidad científica; estos estudios abarcan desde el análisis químico hasta modelos de dispersión atmosférica.

La actividad industrial ha generado una serie de sustancias que son denominadas contaminantes porque modifican la composición del ambiente o afectan la salud de los seres vivos. La legislación mexicana ha dado una definición de contaminante, como toda la materia, sustancia o sus combinaciones y compuestos, así como los derivados químicos y/o biológicos y toda forma de energía térmica, radiaciones ionizantes, vibraciones o ruido, que al actuar en la atmósfera, el agua, el suelo, la

flora, la fauna o cualquier elemento ambiental alteren su composición o afecten la salud humana.

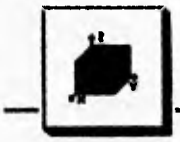
Con base en lo anterior, y para fines prácticos, definiremos el termino de *Contaminación del aire* como "cualquier circunstancia la cual añade o elimina de los usuales constituyentes del aire aquellos que puedan alterar sus propiedades físicas o químicas lo suficientemente para ser detectado por los ocupantes del medio" [Chambers, 1968].

1.1 - Perspectiva Histórica

La composición del aire ha sufrido grandes cambios cualitativos en los 2 billones de años o más desde que la primera unidad anaerobica de importancia se creara, de hecho, las primeras células primitivas no hubiesen ocurrido o continuando existiendo en una atmósfera compuesta de constituyentes químicos que ahora vemos como normales e indispensables para la vida.

De hecho, todas la evidencias existentes indican que el oxígeno casi no existía en forma libre en la atmósfera de ese entonces, y se presume que el oxígeno existente se ha acumulado como resultado de procesos fotosintéticos realizados por especies no dependientes de oxígeno, esta acumulación fue en ese entonces una forma de contaminación atmosférica tal y como lo es la emisión de CO_2, CO, NO_2 y otros componentes químicos resultantes de actividades de las formas de vida actuales, especialmente el hombre.

Este tipo de cambios atmosféricos pudieron influir en la extinción de los tipos de vida de ese entonces, tales como plantas y animales con limitada capacidad de adaptación; estos y muchos otros factores indican que la calidad de la atmósfera sobre la cual las formas de vida dependen se convierte en una variable de importancia para su estudio y control.



1.2 - El primer problema: el carbón

Esta importancia sin embargo fue ignorada en los primeros trece o catorce siglos de la era cristiana hasta el advenimiento del carbón, el uso del carbón que desde el siglo XIV a principios del XX era la fuente principal de energía, trajo consigo problemas de tipo ambiental que llamaron la atención de la sociedad de ese tiempo, tal es el caso de lo sucedido en Gran Bretaña durante el reinado de Edward II (1307 - 1327), donde un hombre fue puesto bajo tortura por llenar el aire con un "Hedor Pestilente" debido a el uso del carbón; o bien bajo el reinado de Henry V (1418 - 1422) donde las medidas para prevenir la contaminación fueron mediante la aplicación de severos impuestos, así como la creación de una comisión especial con el fin de vigilar el movimiento del carbón en Londres.

A inicios del siglo XIX el riesgo del humo en Londres y otras ciudades inglesas era sumamente preocupante, tal es el caso de lo sucedido en 1873 donde el crecimiento gradual del humo producido por el uso del carbón provocó gran mortandad en la población de Londres, así como en 1952 donde el problema del humo culminó en la muerte de 4000 personas en pocos días.

América también ha sido escenario de desastres similares, tal es el caso de lo ocurrido en el pueblo industrial de Donora, Pennsylvania, en 1948, donde la contaminación del aire provocó la muerte de 20 personas y causó grandes daños a la población.

Desafortunadamente, en estos como en otros casos de desastres ambientales, acciones positivas contra la contaminación del aire - o de otros como agua, etc. - han sido rara vez anticipadoras, en lugar de esto, estas acciones han ocurrido después de grandes desastres.

1.3 - El surgimiento del petróleo

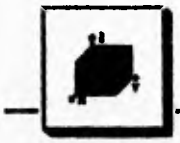
A partir de la segunda guerra mundial el uso de los productos del petróleo como gasolina, aceites, etc. han sido tremendamente acelerados, empleándose principalmente en motores de combustión interna, ya sea en fabricas, aeropuertos, ferrocarriles y transportes marítimos.

Si bien el uso de estos productos ha reducido en gran medida el riesgo que implica la contaminación debido a la combustión del carbón, el problema de la contaminación ambiental no se ha reducido del todo y su uso indiscriminado ha generado grandes problemas en las grandes ciudades del mundo.

Muchos de los contaminantes relacionados con el uso del petróleo y sus productos, tienen un bajo nivel de tóxicos intrínsecos o elementos irritantes. Por el contrario, sus reacciones fotoquímicas pueden afectar sistemas biológicos a muy bajas concentraciones. La variedad de contaminantes resultantes de la exposición a la luz solar de olefinos y otros reactivos producidos por el empleo de productos del petróleo es muy extensa y dependen de las condiciones atmosféricas tales como humedad, temperatura, intensidad de la radiación solar y otros gases reactivos.

1.4 - Contaminantes

Los tipos de estudios relacionados con la contaminación atmosférica son muy variados y dependen principalmente del tipo de contaminante. Debido a la gran variedad de contaminantes atmosféricos es difícil realizar una clasificación de los mismos, ya que estos pueden presentarse como partículas sólidas, pequeñas gotas de agua, gases o mezclas complejas de estas formas; en un esfuerzo para clasificar los contaminantes hasta ahora conocidos, es conveniente considerar dos grupos generales:



a) Aquellos emitidos directamente de fuentes identificables.

b) Aquellos producidos en el aire por la interacción de dos o más contaminantes primarios, o por reacción con constituyentes atmosféricos normales, con o sin fotoactivación.

El estudio de las cualidades de una fuente de aire contaminada, es sumamente complicado, esto se debe principalmente a que ninguna de las entidades contaminantes retienen sus características después de entrar en la atmósfera.

Aún así y considerando los dos grupos antes mencionados, el estudio de la calidad del aire se enfoca principalmente a lo que se conoce como *emisiones primarias* (también conocidos como contaminantes primarios) y *emisiones secundarias* (contaminantes secundarios).

Emisiones primarias

Las emisiones primarias son a menudo clasificadas de acuerdo a criterios diversos y que generalmente dependen del punto de vista del autor que las clasifica. Una posible clasificación de este tipo de emisiones puede ser:

- Partículas finas (menos de 100 μm de diámetro).
- Partículas gruesas (mayores de 100 μm de diámetro).
- Componentes de azufre.
- Componentes orgánicos.

- Componentes halógenos.

- Componentes radioactivos.

Las partículas finas generalmente son producidas por aerosoles los cuales incluyen partículas de metal, carbón, alquitrán, resina, polen, óxidos, nitratos, sulfatos, silicatos y una gran cantidad de otros componentes los cuales ocultan muchos otros de categorías más específicas.

Como partículas, estas se esparcen de acuerdo a leyes físicas bien conocidas, así mismo, rigen a un grado substancial la condensación y unión de otras partículas y gases. Químicamente, estas partículas son sumamente tóxicas para plantas y animales o son corrosivas a metales y otros materiales, posteriormente, depositadas en forma de ceniza, ya sea debido a la acción de la gravedad o atracción electrostática, ensucian ropa, construcciones, etc. constituyendo un riesgo general.

Las partículas gruesas presentan los mismos problemas pero a un menor grado, esto se debe a varios factores tales como la gravedad, que permite que dichas partículas sean rápidamente eliminadas de la atmósfera o bien los mecanismos anatómicos de defensa que impiden que dichas partículas penetren a los pulmones de animales y seres humanos.

El interés por los componentes de azufre ha sido prolongado e intenso, esto se debe a el importante papel que este jugó en los desastres de 1952 en Londres. La quema de combustibles que contienen azufre generan grandes cantidades de SO_2, algunas cantidades de SO_3 y en algunos procesos industriales se genera H_2S. Todos estos componentes afectan desfavorablemente a plantas y animales, a diferentes, pero generalmente bajas concentraciones.

Los componentes orgánicos son emitidos principalmente como vapores aunque algunos componentes volátiles podrían ocurrir como pequeñas

gotas de agua o partículas sólidas, estos tienen en general bajo potencial de riesgo como contaminantes atmosféricos a excepción de algunos que contienen fósforo y flúor.

Los componentes del nitrógeno mas abundantemente generados y liberados son oxido nítrico, dioxido de nitrógeno y amoniaco. Los primeros dos son producidos en combustiones de alta temperatura y otras operaciones industriales por la combinación de oxígeno atmosférico normal y nitrógeno; si bien el NO_2 es irritante a los tejidos a bajas concentraciones, el principal interés en estos componentes se enfoca a su participación en reacciones atmosféricas fotoquímicas.

El dióxido de carbono y el monóxido de carbono surgen en grandes cantidades de la quema completa e incompleta de combustibles carbonaceos respectivamente. La habilidad del CO de degradar la capacidad de acarreo de oxígeno de la hemoglobina le da una importancia especial como contaminante primario. El dioxido de carbono en muy altas concentraciones afecta el mecanismo de control vascular humano, y aunque la cantidad requerida para causar daños importantes debe ser grande, es también una variable de importancia en el análisis de los contaminantes atmosféricos.

En cuanto a los componentes radioactivos es de sobra mencionar los peligros que estos encierran, basta mencionar el desastre ocurrido en la planta ex-sovietica de Chernovyl, donde la explosión de un reactor nuclear provocó daños considerables al medio ambiente, y cuyos efectos se extendieron a gran parte de Europa.

Emisiones secundarias

Estos contaminantes atmosféricos se caracterizan principalmente por ser química y fisicamente inestables. Las velocidades, reacciones y pasos intermedios involucrados en los procesos químicos y físicos están influenciados por muchos factores tales como concentraciones relativas de reactantes, grados de fotoactivación, fuerzas dispersoras meteorológicas

variables, influencias de la topografía local y cantidades relativas de humedad.

Las reacciones fotoquímicas involucradas en la contaminación del aire han sido analizadas exhaustivamente para así entender su participación en el smog de las grandes ciudades. Los contaminantes secundarios producidos en las reacciones fotoquímicas son los más problemáticos e incluyen componentes reactivos tales como ozono, hidroperoxidos orgánicos, varios aldehidos entre otros.

1.5 - Factores que influyen en la contaminación del aire

En adición a las recombinaciónes químicas, varios factores más regulan el impacto de los contaminantes primarios y secundarios. El principal entre estos son los procesos de condensación, sedimentación y otros fenómenos de limpiado de aire los cuales tienden a eliminar sustancias de la atmósfera, estos y otros procesos meteorológicos pueden diluír los reactantes o tender a concentrarlos.

El estudio de estos factores es sumamente complejo, tal es el caso de las fuentes de aire en las cuales se involucran variables como magnitudes de movimiento del viento ya sea en forma vertical y horizontal, grado de turbulencia inducida por convección y flujo no lineal, etc. En general todos los fenómenos de tipo meteorológico involucran factores los cuales están gobernados por fuerzas sinópticas externas y por influencias térmicas y topográficas.

La existencia de patrones usuales de movimiento de aire sobre áreas geográficas específicas ha sugerido la creación de modelos matemáticos para el control y predicción de acumulación y dispersión de contaminantes.

La creación de estos modelos de dispersión permitirá en gran medida evitar desastres similares a los ocurridos en el pasado y permitirán, de acuerdo a los resultados que de estos se obtengan, tomar decisiones en caso de emergencias ambientales.

Como puede verse, la importancia del viento y su comportamiento en el problema de la contaminación del aire es evidente, ya que los problemas más agudos se presentan cuando estos patrones usuales de movimiento y en general, patrones meteorológicos comunes no ocurren.

1.6 - Modelos de dispersión

El estudio de estos fenómenos atmosféricos y en especial el relacionado con los procesos de dispersión ha ido en aumento. Hace algunas décadas, antes que las computadoras digitales tuvieran grandes capacidades de almacenamiento o grandes velocidades de procesamiento, la física de dispersión era expresada a través de sub-modelos simples que se resolvían por tablas o bien mediante las modestas computadoras de ese entonces.

"Ahora los investigadores usan supercomputadoras para resolver modelos matemáticos complejos, así como laboratorios especiales para simular comportamientos atmosféricos. Tales resultados pueden ser inclusive más confiables que los resultados obtenidos de mediciones directas en la atmósfera" [Wingaard, 1991].

Estas simulaciones nos han ayudado ha comprender de una mejor manera los procesos de dispersión. Al mismo tiempo, las supercomputadoras nos permiten atacar dichos problemas de dispersión a gran escala, incluyendo simulaciones gráficas en tres dimensiones.

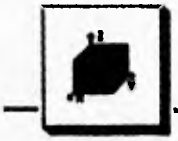
El uso de la computadora para aplicaciones gráficas y el avance tecnológico creciente de estas han permitido la introducción de estaciones

de trabajo gráficas (graphics superworkstations). Máquinas de esta clase trabajan a un factor de 10 de la velocidad de las supercomputadoras más avanzadas pero su costo es considerablemente menor. Estas son además de fácil mantenimiento y no requieren de sistemas de enfriamiento u otros tipos de condiciones ambientales restrictivos.

Si bien el uso de estas estaciones de trabajo gráficas es la opción más utilizada por los investigadores, no podemos ignorar el rápido y creciente poder que la computadora personal ha ido ganando desde su introducción en Septiembre de 1981. Estas computadoras personales han tenido un crecimiento asombroso que aunado a la disminución drástica de su costo las hacen una opción ideal para procesos que no requieran cálculos numéricos complejos o bien gráficas de muy alta resolución.

Los resultados obtenidos del estudio de procesos de dispersión han generado una gran cantidad de modelos matemáticos los cuales atacan dicho fenómeno en forma regional o a una escala media. Dichos modelos difieren entre si en diversos aspectos entre los que se encuentran:

- La entrada que requieren.
- Dimensionalidad - Capacidad de trabajar en 2D o 3D.
- Dominio del modelo - Distancias que cubre el modelo.
- Técnica de solución - Tipo de teoría matemática aplicada.
- Sistema coordenado.
- Condiciones iniciales.



- Otros.

"El modelo meteorológico de escala regional NCAR/Penn State (PSU) y el sistema de modelado atmosférico regional de la universidad del estado de Colorado (CSU RAMS) son considerados como las herramientas de modelado meteorológico más generales de entre muchas otras existentes, tales como el modelo de San José, el modelo MASS, el modelo Drexel, el modelo los Álamos y el modelo ERT CMC/PBL" [Pielke, 1991].

Si bien el avance en el estudio de estos modelos ha sido acelerado se continúan buscando alternativas que de alguna u otra forma mejoren los resultados que hasta ahora se han obtenido; tal es el caso del modelo presentado en este trabajo, donde con una formulación especial de una herramienta computacional llamada autómatas celulares, se intentará realizar simulaciones del comportamiento de algún contaminante específico suspendido en la atmósfera.

1.7 - Situación Actual

Las alteraciones climáticas y atmosféricas que causan los contaminantes (primarios y secundarios) son preocupantes, la contaminación causada por los clorofluorocarbonados usados en los sistemas de aire acondicionado, refrigeración, extinguidores, aerosoles en bote y en la fabricación de materiales de espuma de estireno o unicel, está dañando seriamente la capa de ozono de la atmósfera. Esta capa se encuentra a docenas de kilómetros de altura y ha sido destruida entre un 3% y 5% por estos compuestos formados por cloro, flúor y carbono, una sola molécula de ellos dura activa en la atmósfera un promedio de 150 años y en ese tiempo destruye casi 100,000 moléculas de ozono. El agujero de ozono cubre ya 26 millones de kilómetros cuadrados, lo que equivale a tres veces el tamaño de la República Mexicana.

A nivel mundial, se generan cada año 1,414,000 toneladas de clorofluorocarbonados y México produce 15000 toneladas, lo que representa un poco más del 1% del total. La importancia que tiene este hecho radica en un aumento en la incidencia de los casos de cáncer de piel, y algunos estudios sugieren, que el aumento en la radiación ultravioleta afecta el sistema inmune del ser humano, reduciendo su capacidad de defensa a las enfermedades. En Australia, se ha encontrado disminución en los niveles de vitaminas C y E en el organismo y se presume que las enfermedades infecciosas aumentarán en 50% en los próximos diez años.

Por otro lado, los vegetales son muy sensibles al incremento de los rayos ultravioleta, lo que disminuirá la producción de comida y esto mismo podría ocurrir con el fitoplanctón[1], el daño a las especies vegetales por los

[1] El fitoplanctón son los microorganismos que encabezan la cadena alimenticia de los mares.

efectos de los contaminantes, podría afectar la producción de oxigeno en el planeta, principal elemento en la formación de la vida.

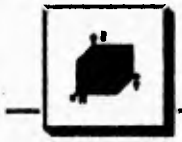
Otro problema que está ocurriendo en la actualidad, es el calentamiento de la atmósfera, producido por gases tales como bióxido de carbono, los fluorocarbonados, metano, bióxido de azufre, óxidos nitrosos, entre otros, causando el llamado efecto de invernadero El calentamiento de la atmósfera va a ocasionar sequías y una expansión de los desiertos, haciendo la producción de alimentos más precaria, especialmente en África. El recalentamiento global, producirá también deshielo en los polos, con aumento del nivel de los mares e inundaciones en vastas extensiones de tierra agrícola situadas a nivel del mar y anegar algunas de las mayores concentraciones del mundo de población urbana, por ejemplo Egipto o los países bajos.

Una alteración del nivel de los océanos significará para nuestro país, la pérdida de la península de Yucatán, y afectación en los puertos de Veracruz, Mazatlán, Tuxpan y Salina Cruz. Se calcula que podrían llegar a haber 750 millones de refugiados climáticos a nivel mundial en los próximos 20 años, desplazados de sus lugares de origen por falta de alimentos suficientes para sus sustento, lo que daría origen a un incremento en la mortalidad y morbilidad de los habitantes del planeta.

El problema de la contaminación atmosférica es un tema que si bien ha tomado importancia recientemente en nuestro país, este ha estado presente de mucho tiempo atrás. El estudio del comportamiento de estos contaminantes permitirá, sin lugar a dudas, predecir los efectos que estos contaminantes tengan sobre el medio ambiente y, dependiendo de estas predicciones, tomar las acciones necesarias para corregirlos.

2

Modelo de simulación con Autómatas Celulares



2.1 - Introducción

La computación en paralelo ha enfatizado que los sistemas pueden ser descompuestos en subunidades independientes con interacciones mínimas, por ejemplo, muchos sistemas de procesamiento en paralelo tienen la capacidad de identificar segmentos de programas que pueden ser ejecutados simultáneamente, de tal forma que las interacciones entre varios segmentos de programa son manejados directamente.

El principal problema que surge de esta arquitectura es que requiere de una gran cantidad de recursos del sistema para manejar y coordinar las actividades de varios procesadores, lo que disminuye considerablemente la velocidad y eficiencia que este tipo de arquitecturas ofrece.

Sin embargo en años recientes ha habido un crecimiento explosivo en la investigación y desarrollo de varios tipos de herramientas de software y algoritmos que puedan explotar al máximo todo el poder de cálculo que las arquitecturas de computadoras vectoriales y paralelas ofrecen, tal es el caso de la llamada computación emergente.

Computación Emergente

Los investigadores en varios campos han comenzado a explotar modelos computacionales en los cuales el comportamiento del sistema completo es, en un sentido, más que la suma de sus partes, estos campos de investigación incluyen: modelos biológicos, autómatas celulares, algoritmos genéticos, redes neuronales artificiales entre otros. Estos sistemas tienen como principal característica el que su comportamiento global emerge de algunas interacciones locales, por esta razón, estos sistemas se encuentran en un campo llamado computación emergente.

La premisa de la computación emergente se basa en el hecho de que interesantes y útiles sistemas computacionales pueden ser construidos al explotar las interacciones que existen entre componentes primitivos, y

más aún, para algunos tipos de problemas (i.e. el modelado de comportamiento inteligente) puede ser el único método disponible.

En los últimos años, se ha incrementado el uso de nuevas técnicas de computación para describir fenómenos físicos en términos de sus propiedades de procesamiento de información. Esta cualidad de la computación emergente permite describir fenómenos en donde la ausencia de información a niveles bajos puede existir a nivel de actividad colectiva, por ejemplo, los autómatas celulares permiten describir un fenómeno complejo, el cual resulta de la interacción de los componentes que lo forman.

Entre las principales características de la computación emergente, tenemos:

1. **Los modelos están formados de una colección de agentes, cada uno siguiendo instrucciones explícitas.**

2. **Existen interacciones entre los agentes (de acuerdo a las instrucciones), los cuales forman patrones globales implícitos a nivel macroscópico.**

3. **Existe una interpretación natural de los resultados pudiendose utilizar técnicas avanzadas de visualización.**

Como puede observarse, la computación emergente tiene, debido a sus características, un paralelismo inherente que permitirá una mejor utilización de las arquitecturas de computadoras paralelas que existen en la actualidad.

Actualmente, muchos sistemas de computación emergente son interpretados directamente por la persona que realiza el experimento, es decir, el análisis de resultados y su interpretación depende de la experiencia y conocimiento del investigador, esto se debe, principalmente, a que la presentación de resultados en este tipo de sistemas son esencialmente gráficos.

La auto-organización, fenómenos colectivos y comportamiento cooperativo son tres de las más importantes características de comportamiento que presentan los sistemas de computación emergente. Auto-organización significa el orden espontaneo (por orden entendemos un comportamiento completamente definido en cuanto a organización se refiere) que emerge de un sistema que inicialmente se considera aleatorio. Fenómenos colectivos son aquellos en los cuales existen muchos agentes, muchas interacciones entre esos agentes, y un gran énfasis en los patrones globales de comportamiento. Un tercer componente de computación emergente es la noción de comportamiento cooperativo, esto es, que el todo es de alguna forma más que la suma de sus partes.

Uno de los más importantes ejemplos de la computación emergente podemos tomarlo de la naturaleza, "*las colonias de hormigas*". Las acciones de cualquier hormiga como ente individual son muy limitadas y aparentemente aleatorias, pero la organización y comportamiento colectivo de la colonia es altamente sofisticado, logrando actividades tales como comunicación en masa y cooperación para la construcción de nidos. En la ausencia de cualquier control centralizado, la entidad colectiva (la colonia) puede "decidir" (la decisión por si misma es emergente) cuando, donde y como construir un nido, auto-organizandose.

Claramente, muchas de las actividades en una colonia de hormigas involucran procesamiento de información, tales como trazar rutas desde el nido a lugares donde exista una fuente potencial de comida, comunicando la calidad y cantidad de comida a un lugar en particular, etc.

La aplicación de los modelos de computación emergente se ha incrementando notablemente, su aplicación ha sido exitosa en varias áreas tales como aprendizaje [Quian, 1990; Minsky, 1969], modelado de fenómenos físicos [Toffoli, 1990; Hopfield, 1982; Kirkpatrick, 1983] e incluso proponiendo un cambio en el paradigma del diseño de sistemas computacionales que podrían conducir potencialmente a una nueva arquitectura de computadoras [Hills, 1984].

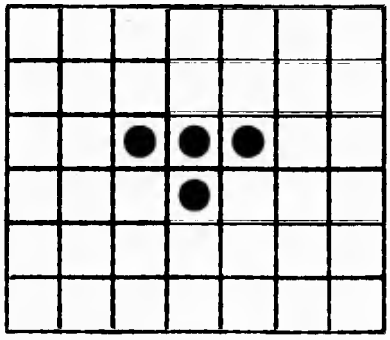
No todos los intentos que se han realizado para aplicar los conceptos de la computación emergente han resultado exitosos. La red Internet (una red de área amplia para el intercambio de correo electrónico) fue diseñada de tal forma que los mensajes puedan ser ruteados de una manera aleatoria (existen usualmente muchas rutas diferentes que un mensaje puede tomar entre dos hosts Internet). El propósito es que el tráfico de mensajes sea eventualmente distribuído a través de toda la red.

No obstante, en algunas circunstancias los mensajes han sido encontrados auto-organizandose en una estructura de alto nivel llamada anillo de paso de token (token-passing ring), esto ha originado que los mensajes se concentren en un nodo, los cuales entonces, son pasados a el siguiente nodo de el anillo. En este caso, la Auto-organización es altamente perjudicial para el correcto funcionamiento (performance) de la red.

Las áreas en las que los modelos de computación emergente pueden ser aplicados son muchas y muy variadas, actualmente su principal área de aplicación es en la investigación, sin embargo, y como se mencionó anteriormente, ya existen intentos de aplicación de estos modelos cuyos resultados, satisfactorios o no, influirán en la búsqueda de nuevos modelos. Uno de los modelos que ha llamado poderosamente la atención de los investigadores son los autómatas celulares, estos, han sido estudiados exhaustivamente y han generado una amplia teoría cuyas bases fundamentales son revisadas brevemente en el apéndice A de este trabajo. En la siguiente sección se hará una breve revisión de las principales características de los autómatas celulares así como de los campos en los que han sido aplicados con éxito.

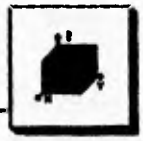
2.2 - Autómatas Celulares

Una definición no formal de un Autómata celular[1] y que consideramos adecuada para propósitos introductorios es la propuesta por Françoise Fogelman e Yves Robert en[Fogelman, 1987], "Una red de autómatas puede ser definido, en una forma general, como un conjunto (generalmente muy grande) de células (autómatas finitos), interconectadas localmente, el cual puede evolucionar a intervalos discretos de tiempo a través de interacciones mutuas. Desde una punto de vista matemático, las redes de autómatas son *sistemas dinámicos discretos*: espacio, tiempo y las células mismas son básicamente discretas." [Fogelman, 1987].

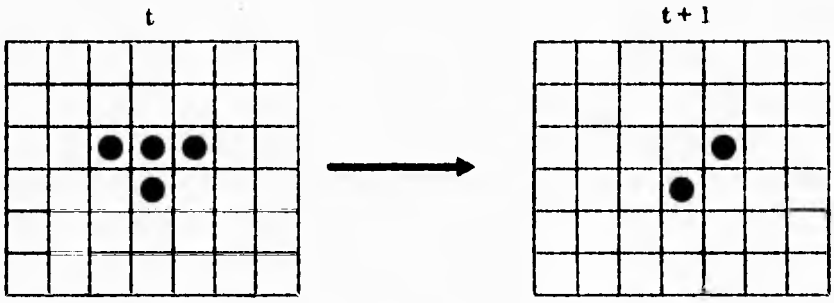


Ejemplo de Autómata Celular de dos dimensiones

[1] También conocidos como redes de autómatas.

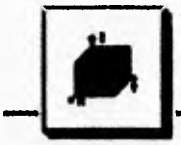


Cada célula de el espacio definido para el autómata celular, se encuentra en uno de un número finito de estados en cualquier instante de tiempo. La evolución del autómata celular se da principalmente a nivel *local*; donde *local* significa que el estado de una célula a un tiempo t+1 está en función de su propio estado y los estados de sus células vecinas inmediatas. Del mismo modo, se dice un autómata celular es totalmente *determinístico*, con determinístico queremos dar a entender que una vez que el comportamiento local ha sido definido y los estados iniciales de las células han sido establecidos, su evolución futura esta completamente determinada.



Una vez definidas las reglas locales y los estados iniciales del autómata celular su comportamiento es completamente determinístico

Los autómatas celulares son de sumo interés debido a que aún con un autómata relativamente simple y reglas lógicas simples, el autómata celular es usualmente capaz de generar una amplia variedad de comportamientos, la gran mayoría complejos. Von Neuman , E. F. Codd entre otros [Codd, 1968], han demostrado que ciertos autómatas celulares son capaces de autoreproducirse. Así también se sabe que esta clase de sistemas tienen lo que en la teoría matemática computacional se conoce como *computación universal*.



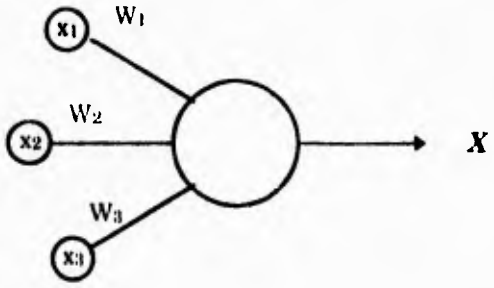
Los autómatas celulares son herramientas sumamente útiles para atacar problemas de inferencia complejos [Quian, 1990], así también estos "pueden proveer una solución alternativa a los modelos clásicos basados en *sistemas dinámicos continuos*, esto es, ecuaciones diferenciales ordinarias y ecuaciones diferenciales parciales" [Fogelman, 1987].

Los autómatas celulares como modelos de computación emergente han sido aplicados exitosamente en varios problemas que se caracterizan por un comportamiento dinámico y una variabilidad espacial relevante siendo sus principales campos de aplicación la física y biología [Fogelman, 1987].

En física los autómatas celulares han sido aplicados exitosamente como herramientas para el establecimiento de leyes mecánicas y termodinámicas, tal es el caso del modelo llamado "*spin glass*" [Kirkpatrick, 1979], el cual es un modelo clásico de las propiedades termodinámicas de sistemas desordenados, aquí las partículas son representadas por valores opuestos (+1 o -1) y colocados en los nodos de una malla (que puede ser de una, dos o n dimensiones). Las interacciones entre los nodos de malla son aleatorios. Estos sistemas (*spin glasses*) tienen muchas configuraciones diferentes de baja energía (o estados aterrizados) los cuales son sus estados estables. Este tipo de modelo ha sido también propuesto para modelar *redes neuronales*.

La simulación es un área en la que los autómatas celulares han tenido gran utilización. Tal es el caso de el diseño de modelos discretos en mecánica de fluidos para modelar turbulencias, los cuales prueban ser sumamente eficientes, compitiendo fuertemente con la aproximación de elementos finitos de las ecuaciones diferenciales parciales de Navier - Stokes [Frish, 1986], este modelo es de suma importancia para nuestro trabajo, ya que involucra un tipo especial de autómatas celulares llamado *lattice gases*, cuya principal característica es el modelado de sistemas físicos orientados al problema de la difusión de partículas, los cuales son modelados tradicionalmente por ecuaciones diferenciales parciales.

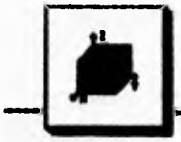
En biología, las redes boleanas aleatorias han sido usadas para modelar sistemas genéticos [Thomas, 1979], dichos sistemas están generalmente modelados, de forma tal, que cada autómata binario determinístico de nuestra red están conectados de manera aleatoria. El estudio de las redes neuronales, fue originalmente introducido por W. S. McCulloch y W. Pitts [McCulloch, 1943], en donde sus esfuerzos por simular las células del sistema nervioso (neuronas) los llevaron a introducir la noción del *autómata de umbral* para después ser formalizado en lo que ellos llamaron *perceptrón*, los estudios iniciales de W. S. McCulloch y W. Pitts han generado una gran cantidad de estudios posteriores creándose un campo de investigación independiente conocido como *inteligencia artificial*.



$$x = \begin{cases} 1 & \text{si } \sum_j W_j X_j \geq 0 \\ 0 & \text{de otro modo} \end{cases}$$

El Autómata de umbral o Perceptrón, ha generado incluso u área independiente llamada *inteligencia artificial*

La simulación del comportamiento de organismos vivientes (como nuestro ejemplo de las hormigas mencionado anteriormente), ha sido también



materia de interés, tal es el caso de el juego matemático *life* definido por J. Conway en 1970, el cual es realizado mediante autómatas finitos simples, cada uno siguiendo reglas perfectamente establecidas. Este autómata celular no obstante su simplicidad es capaz de generar un comportamiento sumamente interesante.

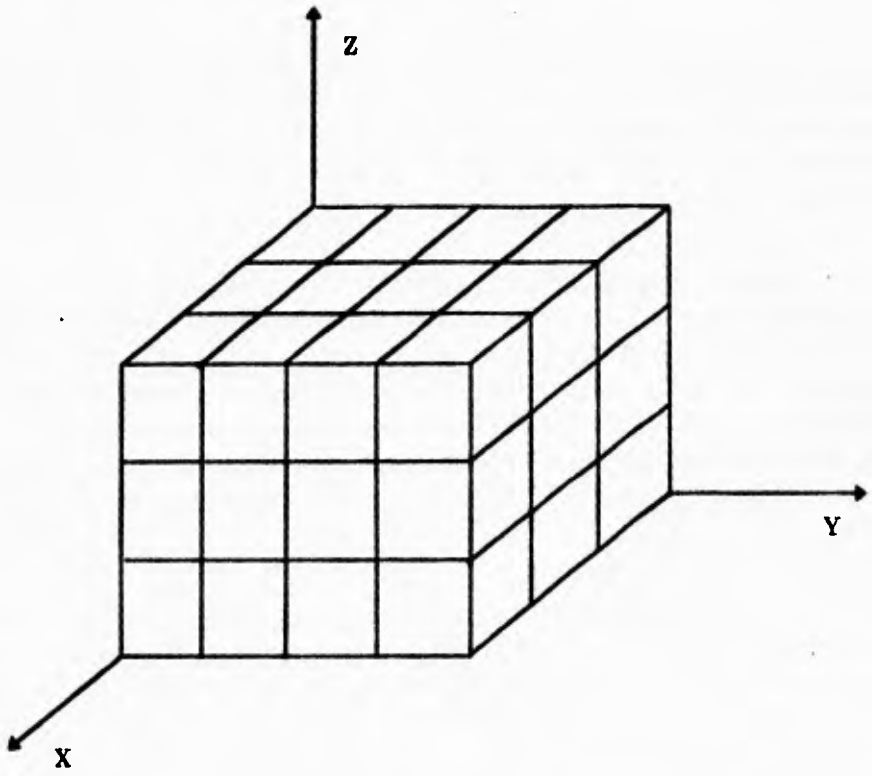
2.3 - Desarrollo del modelo

Como se mencionó con anterioridad el objetivo del presente trabajo, es la aplicación y demostración del modelo de dispersión de contaminantes atmosféricos presentado por Giorgio Guariso y Vittorio Maniezzo [Guariso, 1991], donde desarrollan y aplican una formulación especial de autómatas celulares.

En el modelo propuesto por G. Guariso y V. Maniezzo sugieren "la posibilidad de usar una formulación de autómatas celulares a un nivel muy diferente de la representación espacial. En particular, hemos aplicado una formulación de autómatas celulares a el problema de la difusión de un contaminante atmosférico con un dominio de cómputo de varios kilómetros cúbicos y en donde una celda en particular representa una porción de espacio de varios cientos de metros cúbicos" [Guariso, 1991].

El modelo propuesto esta fuertemente sustentado en la teoría de los autómatas celulares, por lo que antes de continuar con la lectura recomendamos consultar el apéndice A para una breve revisión de los conceptos básicos de la teoría general de estos.

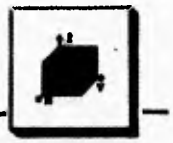
El modelo propuesto por G. Guariso y V. Maniezzo, consiste básicamente de un autómata celular en tres dimensiones, dicho autómata representa una región en el espacio real donde cada célula del autómata se considera es una pequeña región dentro de dicho espacio con un comportamiento perfectamente definido.



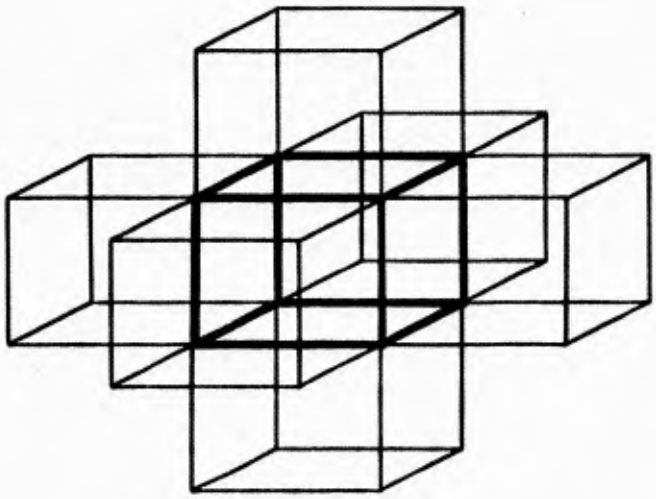
Autómata Celular de tres dimensiones

Cada célula $C_{ijk}(t)$ del autómata celular se identifica por su posición i, j, k en los ejes coordenados (i a lo largo del eje *x*, j a lo largo del eje *y* y k a lo largo del eje *z*), donde C representa el estado de la célula i,j,k del autómata celular a un tiempo t.

El estado de la célula del AC representa la concentración de algún contaminante, dicha concentración es representada por un número



(contenido en la célula) que varía entre 0 y M, donde M representa la maxima concentración de dicho contaminante en la célula. El autómata finito de cada célula esta en este caso representado por una *regla de actualización*, esta regla de actualización, que tiene la función del autómata finito dentro de las células, calcula los estados de cada célula de acuerdo a los valores de los estados de sus células vecinas, que se muestran a continuación.



Cada célula de nuestro autómata tiene 6 células vecinas las cuales afectan su comportamiento

La *regla de actualización* perteneciente a cada célula del autómata considera los intercambios de masa que ocurren entre las células vecinas, esta *regla de actualización* se encuentra afectada por los siguientes factores físicos:

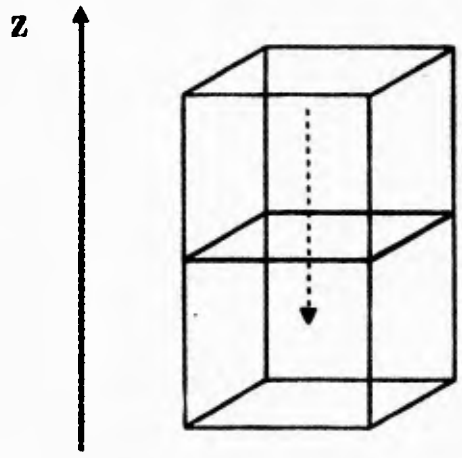
Gravedad

Esta afecta el intercambio de masa (cantidad de contaminante) entre celdas que se encuentran en la misma columna.

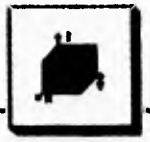
La fórmula que modela este comportamiento es:

$$\Delta g C_{ijk} = g(C_{ijk-1} - C_{ijk}) \qquad (1)$$

Esta fórmula representa el balance de masa debido a la gravedad, donde Δg es la variación de concentración de contaminante en la celda considerada en una unidad de tiempo debido a la gravedad y g es el parámetro que representa la aceleración de la gravedad, es importante hacer notar que este parámetro no es la aceleración misma (9.81 m/seg^2), solo representa el efecto de esta fuerza sobre las partículas.



El efecto de la gravedad afecta únicamente a las particulas de las células paralelas al eje z



Transporte

El cual se encuentra determinado por la velocidad del viento en las tres direcciones de las abscisas.

La fórmula del balance de masa debido al transporte es:

$$\Delta t C_{ijk} = w_i(C_{ijk-1} - C_{ijk}) + w_j(C_{ij-1k} - C_{ijk}) + w_k(C_{i-1jk} - C_{ijk}) \quad (2)$$

y donde w_i, w_j y w_k, representan la velocidad del viento, teniendo valores positivos en estos parámetros cuando la dirección del viento sigue la dirección positiva de los ejes coordenados y negativos en caso contrario.

Difusión

La cual afecta los intercambios de masa entre celdas vecinas debido únicamente a los gradientes de concentración.

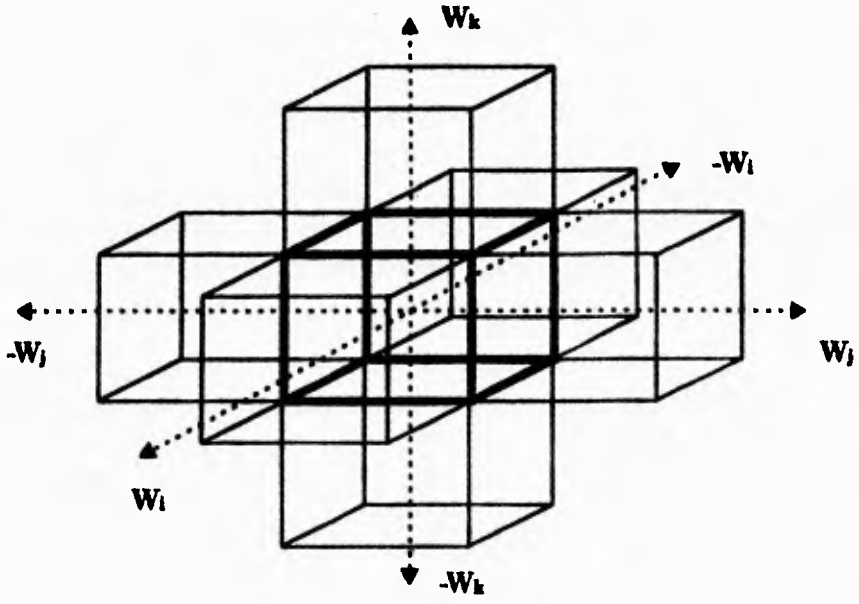
La fórmula del balance de masa debido a la difusión es:

$$\Delta d C_{ijk} = d_z(C_{ijk-1} - C_{ijk}) + d_y(C_{ij-1k} - C_{ijk}) + d_x(C_{i-1jk} - C_{ijk}) +$$

$$d_z(C_{ijk+1} - C_{ijk}) + d_y(C_{ij+1k} - C_{ijk}) + d_x(C_{i+1jk} - C_{ijk}) \quad (3)$$

La característica principal de esta fórmula es el hecho de que se asume una difusión isotrópica[2].

[2] En este caso con difusión isotrópica nos referimos al hecho de que la difusión de masa en las tres direcciones es idéntica.

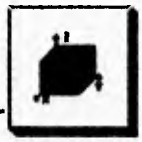


La velocidad del viento en las tres direcciones afectan a las células paralelas a la dirección del mismo

Combinando las tres fórmulas anteriores tenemos:

$$\Delta C_{ijk} = \alpha C_{ijk} + \beta C_{ijk+1} + \gamma C_{ij-1k} + \psi C_{i-1jk} + d_x C_{i+1jk} +$$

$$d_y C_{ij+1k} + d_z C_{ijk-1} \qquad (4)$$



La fórmula (4) representa finalmente a la *regla de actualización*, dicha regla calcula entonces, los estados internos de las células de nuestro autómata celular, los cuales, en cooperación con el resto de las células de el autómata, modelan el comportamiento deseado. Las fórmulas anteriores fueron tomadas literalmente del desarrollo expuesto por G. Guariso y V. Maniezzo, sin embargo, la comprobación de el desarrollo de las ecuaciones ha dado pie a la localización de posibles errores, los cuales se comentarán a continuación.

Análisis Detallado de la Fórmula

Partiendo del hecho de que las ecuaciones (1) (2) y (3) son correctas, podemos entonces realizar la combinación de estas, para así tratar de llegar a la *regla de actualización* propuesta originalmente (ecuación 4), para tal consideraremos un comportamiento lineal.

Sumando entonces (1), (2) y (3)

$$\begin{aligned}\Delta C_{ijk} = {} & gC_{ijk-1} - gC_{ijk} + w_iC_{ijk-1} - w_iC_{ijk} + w_jC_{ij-1k} - w_jC_{ijk} + \\ & w_kC_{i-1jk} - w_kC_{ijk} + d_zC_{ijk-1} - d_zC_{ijk} + d_yC_{ij-1k} - d_yC_{ijk} + \\ & d_xC_{i-1jk} - d_xC_{ijk} + d_zC_{ijk+1} - d_zC_{ijk} + d_yC_{ij+1k} - d_yC_{ijk} + \\ & d_xC_{i+1jk} - d_xC_{ijk} \qquad (5)\end{aligned}$$

Simplificando la expresión (5)

$$\begin{aligned}\Delta C_{ijk} = {} & (-g - w_i - w_j - w_k - 2d_z - 2d_y - 2d_x)C_{ijk} + \\ & (g + w_i + d_z)C_{ijk-1} + (w_j + d_y)C_{ij-1k} + (w_k + d_x)C_{i-1jk} +\end{aligned}$$

$$d_x C_{ijk+1} + d_y C_{ij+1k} + d_x C_{i+1jk} \qquad (6)$$

Comparando los parámetros de la ecuación (4), con el resultado obtenido en la ecuación (6) tenemos que para α, β, γ y ψ:

α $= -g - w_i - w_j - w_k - 2d_x - 2d_y - 2d_x$

β = No tiene equivalencia.

γ $= w_j + d_y$

ψ $= w_k + d_x$

Analizando la expresión (6) y comparándola término a término con la expresión (4), nos podemos percatar de una pequeña variación de los términos resultantes. En dicha comparación podemos ver que, el término C_{ijk+1} se encuentra operando en la fórmula original (ecuación 4) con el parámetro β, más sin embargo en la fórmula resultante del desarrollo anterior, dicho término se encuentra multiplicado por d_x[3].

[3] Cabe mencionar que los parámetros α, β, γ y ψ no pueden ser verificados contra nuestro desarrollo, debido a la falta de los mismos en el trabajo original.

Término	Ecuación 4	Ecuación 6
C_{ijk}	α	$(-g - w_i - w_j - w_k - 2d_z - 2d_y - 2d_x)$
C_{ijk+1}	β	d_z
C_{ij-1k}	γ	$(w_j + d_y)$
C_{i-1jk}	ψ	$(w_k + d_x)$
C_{i+1jk}	d_x	d_x
C_{ij+1k}	d_y	d_y
C_{ijk-1}	d_z	$(g + w_i + d_z)$

Una posible explicación (y quizá la mas viable) es el de un posible error tipográfico en la edición de dicho trabajo, esta conclusión es hasta cierto punto lógica, debido a que la diferencia en las ecuaciones (4) y (6) es únicamente en dos términos, los cuales posiblemente hayan sido intercambiados al momento de escribirla.

Para fines prácticos, la regla de actualización resultante de el análisis anterior con los parámetros α, β, γ y ψ de la ecuación (4) será:

$$\Delta C_{ijk} = \alpha C_{ijk} + \beta C_{ijk-1} + \gamma C_{ij-1k} + \psi C_{i-1jk} + d_x C_{i+1jk} + d_y C_{ij+1k} + d_z C_{ijk+1} \quad (7)$$

Es importante mencionar que el análisis del comportamiento del autómata celular se realizará con base en los parámetros α, β, γ, ψ, dx, dy y dz que pertenecen a la regla de actualización interna de las células que lo forman, el análisis de los posibles valores que de acuerdo a estos parámetros tomen las variables independientes tales como g, w_i, etc. no se realizará en este trabajo de tesis, recordando que los valores que puedan tomar estas variables, dependen en gran medida del contaminante atmosférico cuyo comportamiento desee simularse.

En el capitulo siguiente se introducirán algunos aspectos del Análisis, Diseño e Implementación del sistema que se encargará de realizar la simulación, y en el capítulo 4 y 5 se mostrarán los resultados obtenidos de simulaciones realizadas para la *regla de actualización* desarrollada en este capítulo.

3

Desarrollo del Programa de Simulación

Una vez concebido el modelo basado en autómatas celulares, el paso inmediato es la implementación de dicho modelo en un programa de computadora. Escribir dicho programa comprende varios pasos que van desde la formulación y especificación del problema, el diseño de la solución, su implementación, prueba y documentación, hasta la evaluación de la solución.

El desarrollo de sistemas es un proceso usualmente lento, el cual puede tomar varios años desde el inicio hasta su terminación. Para el desarrollo de estos, se tienen usualmente diversas metodologías cuyo principal fin es la creación y producción de buenos sistemas. Pero ¿Qué es realmente un buen sistema? esta cuestión puede ser contestada parcialmente desde un punto de vista externo y parcialmente desde un punto de vista interno al sistema. El punto de vista externo es de quienes de algún modo usan el sistema. Ellos quieren que el sistema entregue rápidamente resultados correctos, que sea confiable, efectivo, simple de aprender y de usar, etc.

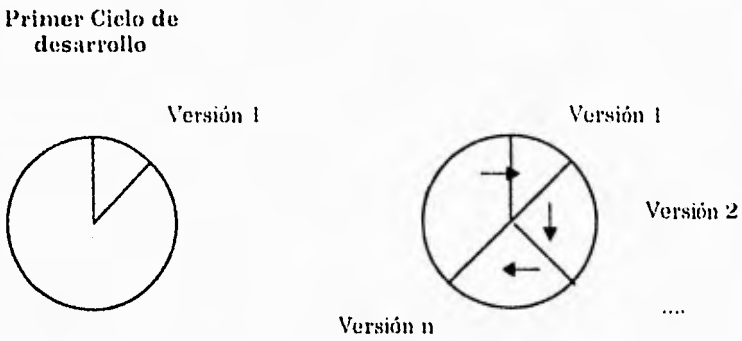
Un buen sistema es evaluado por el usuario de acuerdo a sus características de presentación, utilidad, etc.

El punto de vista interno es el de los desarrolladores del sistema y el de quienes tienen que darles mantenimiento. Ellos quieren que el sistema sea simple de modificar y de hacerle adiciones, simple de entender, que contenga partes reutilizables, que sea simple de probar, compatible con otros sistemas, portable, poderoso y sencillo de manufacturar.

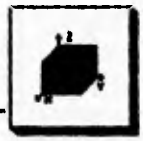
Un buen sistema desde el punto de vista del desarrollador, debe ser simple de entender, simple de modificar, con partes reutilizables, etc.

La definición de lo que es un buen sistema varía en muchos aspectos entre diferentes aplicaciones. En algunas, la eficiencia es lo importante y en otras, una interfaz de uso amigable es crucial. Un buen sistema depende de la estructura del sistema mismo, por ejemplo, si es distribuido o centralizado. De esto concluímos que resulta imposible dar una caracterización general de las propiedades que debe tener un buen sistema; sin embargo, algo común a todos los sistemas, ya sean grandes o pequeños, es que deben ser modificados en su ciclo de vida, por lo tanto, un buen sistema debe por regla general, tener un esquema de modificación posible.

Todos los sistemas cambian durante su ciclo de vida. Muchos métodos de desarrollo actuales se enfocan a los nuevos desarrollos, mientras que se dedica muy poco esfuerzo al trabajo destinado al mantenimiento. Esto ocurre a pesar de que se sabe que los cambios constituyen la parte principal del costo del ciclo de vida total de muchos sistemas. Un sistema se desarrolla normalmente en varias versiones que contienen diferentes cambios. Un desarrollo nuevo es, desde este punto de vista, sólo un caso especial de la primera versión. Si bien esta no es regla general, gran cantidad de los desarrollos se basan en este principio.



La primera versión de un sistema representa una porción mínima del consumo de recursos durante el ciclo de vida del sistema



3.1 - Programación Orientada al Objeto

De las ramas de la industria, la de la computación (refiriéndonos a software y hardware) es quizá la más dinámica y extensa; el desarrollo de la computación en materia de hardware ha sido impresionante, debido principalmente, a los avances tecnológicos de las últimas décadas. El desarrollo de software, sin embargo, adolece de prácticas y métodos bien establecidos, que impiden un crecimiento estable de esta industria.

El desarrollo de software, como toda actividad humana, se rige de acuerdo a los paradigmas vigentes (ideas, conceptos y conocimientos de la época), estos paradigmas alteran la conceptualización que el individuo tiene de su entorno, representando una especie de filtro, permitiendo que sólo aquellos conceptos e ideas que van de acuerdo con el paradigma vigente sean tomados como válidos. El desarrollo de software, como la mayoría de las áreas en las que la búsqueda de soluciones es parte fundamental, también se encuentra afectado por el paradigma de programación vigente.

Thomas Kuhn, en su libro *La estructura de la revolución científica* [Kuhn, 1970], considera un cambio de paradigma, como un cambio (conceptual, de conocimientos, etc.) que el individuo tiene de su entorno. Para entender en este sentido, lo que significa paradigmas de programación, tendremos que apreciar que el programador, operando bajo un paradigma nuevo (o al menos para él, diferente) experimenta una alteración en su conceptualización del proceso de resolución de problemas por medio de la computadora. Hay una modificación esencial de perspectiva sobre lo que es un problema y lo que representa una solución.

A finales de la década de los 80's, un nuevo paradigma cambió por completo los conceptos de programación y los métodos de solución de problemas que hasta entonces existían, a este paradigma se le llamó Programación Orientada a Objetos (POO), y ha generado un cambio radical en la industria del software que se le compara con lo realizado por la programación estructurada en la década de los setentas.

Existen, sin embargo, otros paradigmas de programación distintos a la POO y Programación Estructurada, todos ellos con características propias, las cuales son ideales para solucionar problemas específicos. Entre los principales paradigmas de programación tenemos: *programación procedimentista, programación orientada al objeto, programación lógica, programación funcional* y *programación concurrente* [Greiff, 1993].

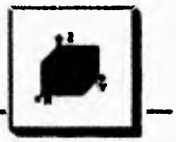
En la programación procedimentista, tenemos procedimientos (comandos en si) los cuales potencialmente forman parte de un procedimiento de más alto nivel. Entonces el programa se conceptualiza como una secuencia de comandos que manipulan datos del mismo y se comunican con el exterior. Este esquema implica, entonces, la existencia de una especie de *todopoderoso* capaz de interpretar los comandos y llevarlos a cabo.

En la POO, el universo deja de estar poblado por procedimientos responsabilizándose por tareas sobre datos ajenos en favor de objetos responsabilizándose por si mismos. Los objetos, se comunican entre sí, por medio de mensajes, pidiendo unos a otros que efectúen tareas que a ellos les corresponde o que provean información sobre su estado actual. La unidad básica de operación sigue siendo el comando, y la ejecución de un programa, una secuencia de acciones. Las acciones son definidas como métodos de objetos que corresponden al paso de mensajes en vez de procedimientos invocados por llamados.

En la POO, la postura mental de un programador trabajando bajo este paradigma prescinde del todopoderoso de la programación procedimentista. Se atribuye la responsabilidad para la interpretación y ejecución de los comandos a los mismos objetos[1]. Smalltalk, el primer

---

[1] Esto, se puede decir, es cierto de los lenguajes que se aferran al ideal del paradigma Orientado a Objetos donde toda operación, por lo básica que sea, esta conceptualizada como un mensaje enviado a otro objeto; en lenguajes menos



lenguaje conscientemente diseñado como un lenguaje orientado a objetos, produjo una explosión del desarrollo de lenguajes Orientados a Objetos y extensiones híbridas de lenguajes originalmente concebidos dentro de paradigmas diferentes.

Conceptos tales como *encapsulamiento*, *herencia*, *polimorfismo*, *clase*, etc. son la base de la POO, el estudio y correcta aplicación de estos, nos permitirá realizar software de alta calidad y con características tan ansiadas dentro de la industria del software como la *reusabilidad*, objetivo cuyo planteamiento ha estado presente desde 1968, en la conferencia de la OTAN sobre la crisis de producción de software, donde se planteó la necesidad de generar bibliotecas de componentes para facilitar la producción industrial de software.

La base para la reusabilidad es una buena biblioteca de componentes reutilizables, Las bibliotecas de componentes deben facilitar el proceso de reutilización de componentes proveyendo:

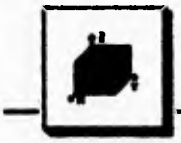
- Un buen nivel de abstracción que permita entender fácilmente que hace un componente.

- Mecanismos de selección para localizar, comparar y seleccionar componentes.

- Mecanismos de especialización de componentes usando parámetros, transformaciones, restricciones o algún otro tipo de refinamiento.

---

fieles al ideal, los comandos básicos requieren todavía de un todopoderoso para su manejo.



- **Mecanismos de integración para combinar la colección de componentes seleccionados en un sistema.**

La POO trajo consigo una nueva forma de pensar y de atacar los problemas a resolver mediante la computadora, la influencia que esta ha tenido en la ingeniería de software ha sido enorme [Quintanilla, 1993], generando una amplia gama de metodologías de desarrollo de sistemas de software que trabajan bajo este paradigma.

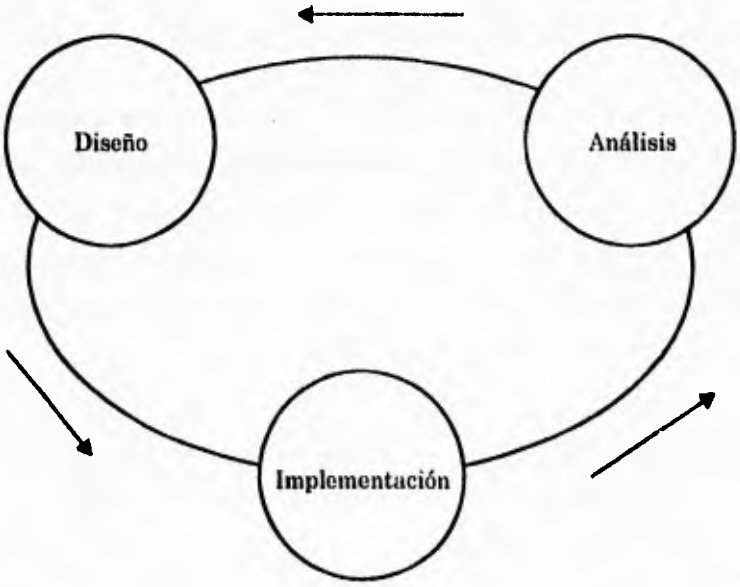
En el paradigma orientado a objetos, la idea central es efectuar una abstracción del mundo real en términos de *objetos* más que de las operaciones involucradas en el problema, para crear un modelo, en que se identifican y organizan conceptos importantes relacionados con la solución del problema a resolver.

Entre las principales características que tiene la POO, podemos resumir:

- **El proceso de desarrollo es más continuo. En el enfoque orientado a objetos se definen las propiedades de los objetos en etapas iniciales y se continúan extendiendo durante todo el proceso. Hay una separación menos clara entre las fases. En las metodologías OO, el modelo de objetos desarrollado en el análisis se refina en el diseño y se implementa. El trabajo consiste en ir refinandolo en niveles progresivos de detalle, en vez de convertirlo a otras representaciones como en las metodologías tradicionales.**

- **Cambia el esfuerzo de desarrollo al análisis. Es el análisis mismo quien capta el mayor esfuerzo. Esto tiene la ventaja de que el software resulta más claro y adaptable a futuros cambios.**

- Se pone énfasis en las estructuras de los datos más que en las funciones. Las estructuras de los datos y sus relaciones son menos vulnerables a cambios en los requerimientos que las acciones efectuadas en ellos. Organizar un sistema alrededor de los objetos proporciona al desarrollo mayor estabilidad.



La Orientación a Objetos permite refinar el modelo en niveles progresivos de detalle en las etapas del desarrollo de software.

3.2 - Desarrollo del simulador

El desarrollo del sistema simulador de dispersión de contaminantes ambientales es parte medular en este trabajo, este sistema tiene como objetivo principal la implementación del autómata celular mostrado en el capitulo anterior.

Especificaciones funcionales

Básicamente el sistema debe cumplir con los siguientes requisitos:

a) El sistema debe ser capaz de leer de un archivo la configuración de el autómata celular, dicho archivo el cual llamaremos archivo de configuración del autómata, contendrá los valores de los estados iniciales de las células que forman al autómata celular, así como los valores de los parámetros de la *regla de actualización* que calcula los estados siguientes de las mismas.

b) El sistema podrá realizar simulaciones paso a paso o bien de manera automática, permitiendo establecer condiciones de finalización de la simulación.

c) El despliegue gráfico es fundamental, por lo que el sistema deberá desplegar los resultados de la simulación en forma gráfica.

d) El sistema debe ser capaz de mostrar regiones específicas del autómata celular.

El sistema asimismo debe de proporcionar ayuda en línea que permita a usuarios con poca experiencia en la operación del simulador, poder consultarla en cualquier momento. De la misma forma debe cumplir al máximo con todos (o casi todos) los requisitos de diseño de sistemas de software en lo referente a las interfaces humano-computadora [González, 1994].

A nivel desarrollo, este sistema debe ser claro y fácil de comprender en su funcionamiento interno, pensando principalmente en la etapa de mantenimiento al sistema y la posible generación de nuevas versiones del sistema. Finalmente, el simulador debe operar en computadoras PC's con procesadores 80x86 o compatibles.

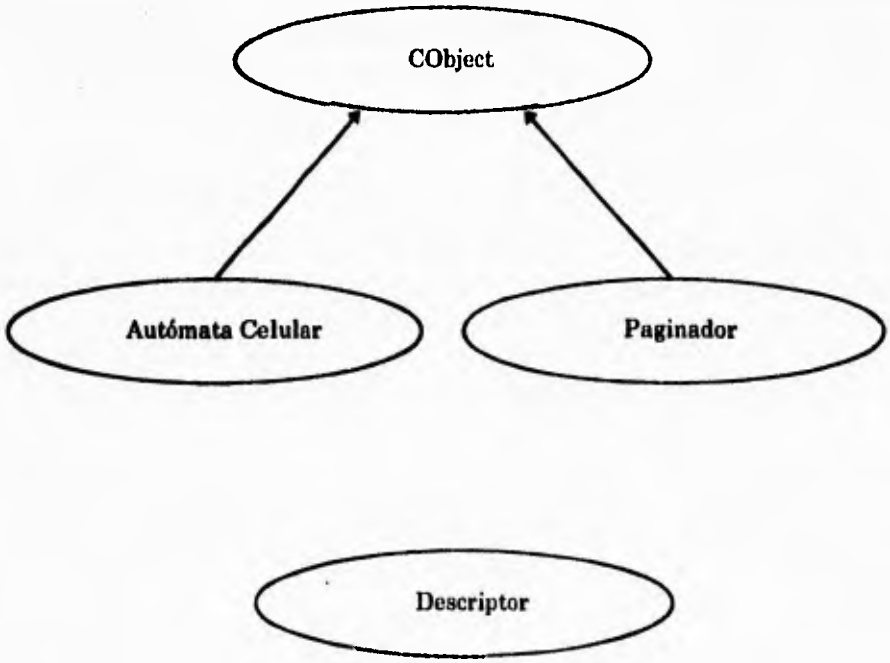
Estructura General del Sistema

Siguiendo un esquema orientado a objetos, el sistema final cuenta con un objeto (o clase, en términos OO) principal llamado **Autómata Celular**, este objeto representa al autómata celular (células, reglas de actualización, funciones de vecindad, etc.) que presentamos en el capitulo dos de este trabajo. La estructura interna de este objeto es:

Autómata Celular
Paginador *Descriptor* Células en X Células en Y Células en Z Tiempo

Estructura Interna del Objeto *Autómata Celular*

Paginador y *Descriptor* son a su vez objetos que colaboran con el objeto *Autómata celular*. A continuación se presenta la jerarquía de objetos que son parte relevante del sistema.



Jerarquía de objetos del Simulador de dispersión de Contaminantes

Autómata Celular

Este objeto representa al autómata celular (AC) mismo; puede leer y analizar los datos provenientes de algún archivo de configuración construyendo la región de acuerdo a este archivo, además realiza el cálculo de los estados internos de cada célula que lo forman de acuerdo a la regla de actualización propuesta y su configuración de vecindad. Finalmente este autómata "sabe" como desplegar cualquier región que se le indique.

Paginador

Este objeto, utiliza un algoritmo de paginación de datos específicamente desarrollado para esta aplicación, el principal objetivo de este objeto es almacenar los estados del autómata celular y de acuerdo a peticiones de otros objetos, entregar el valor del estado interno de la célula requerida.

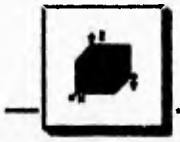
Descriptor

Este objeto, contiene la descripción global del *autómata celular*. Tiene información referente al número total de células del AC, los estados de inicio de cada célula y los valores de los parámetros de la regla de actualización del AC correspondientes a cada célula.

CObject

Este objeto pertenece a la biblioteca de objetos correspondiente al compilador en el que se desarrollo el sistema, y representa a la superclase en la jerarquía de objetos del sistema.

Existen obviamente otros objetos presentes en el sistema, sin embargo, los mencionados anteriormente juegan un papel sumamente importante en la construcción del sistema, por lo que consideramos importante su mención.



Consideraciones de la implementación

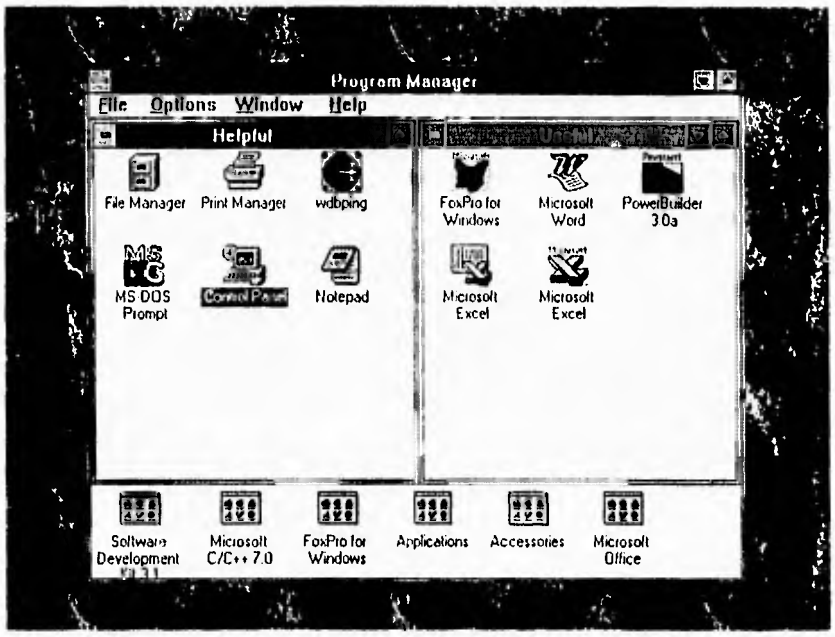
La plataforma que se utilizó para el desarrollo de nuestro simulador de dispersión de contaminantes atmosféricos fue Microsoft Windows, dicha elección se realizó en base a las características gráficas de este sistema y principalmente a la existencia de poderosas herramientas de desarrollo (compiladores, generadores de dialogo, depuradores, etc.), bibliotecas de objetos genéricas, etc. que permiten un desarrollo de aplicaciones sumamente acelerado.

En lo que respecta a la interfaz gráfica seleccionada (MS/Windows), fueron su amplia popularidad en la industria de cómputo y sus capacidades de interfaz humano-computadora los que finalmente influyeron en su elección. La interfaz humano-computadora, como todos sabemos es sumamente importante en la realización de sistemas de software, un diseño apropiado de esta, puede implicar diferencias substanciales en tiempo de entrenamiento, rendimiento, tasas de error y satisfacción de los usuarios.

Existen diversos métodos de control de diálogo entre el humano y la computadora los cuales están íntimamente ligados a la tecnología disponible en el equipo. Así, las primeras interfaces humano-computadora, implementadas en teletipos, se basan en lenguajes de comandos; posteriormente se emplearon las terminales alfanuméricas de rayos catódicos, con lo que se dio paso a implementar los diálogos entre el usuario y la computadora a través de menús de opciones y a través de teclas de función; subsecuentemente, con la disponibilidad de terminales gráficas y de dispositivos apuntadores como el *Mouse*, el *track ball*, y el *joystick*, surgió el método de manipulación directa; actualmente disponemos de equipo que permite al usuario la entrada manuscrita de datos y comandos y ya se encuentran en el mercado algunas interfaces que reconocen la voz humana para recibir comandos.

La manipulación directa, que se refiere a la posibilidad de seleccionar algo - frecuentemente un icono que representa un objeto o colección de objetos

en el dominio de la aplicación - y después decir que se desea hacer con él, proporciona una metáfora de interacción humano - computadora que puede hacer la comunicación más intuitiva. A esto se debe la popularidad de las interfaces gráficas como Macintosh, MS/Windows, X-Windows, etc. que se hacen apropiadas para usuarios con experiencia o no en el manejo de la computadora.



Las interfaces gráficas deben su popularidad a su capacidad de manejo intuitivo

Otro punto importante a tratar con respecto a la implementación del simulador, es el lenguaje de programación y la técnica de programación utilizada. Para seleccionar el lenguaje de programación buscamos aquel

que permitiera programar en la plataforma seleccionada (MS/Windows), que fuera un lenguaje que explotara todas las características esenciales de la plataforma de desarrollo, que fuera un lenguaje de propósito general y, lo más importante, que soportara programación orientada a objetos.

Como se mencionó anteriormente en este capitulo, la POO permite un mejor, más rápido y comprensible desarrollo de los sistemas de software, debido principalmente, a la aplicación de los conceptos pertenecientes al dominio del problema en el mundo real directamente en el desarrollo del sistema. De entre los lenguajes de desarrollo existentes, C++, desarrollado por Bjarne Stroustrup de los laboratorios Bell [Cox, 1987], cumple con los requisitos anteriores, sus capacidades inherentes [Gorlen, 1991] y el hecho de ser el lenguaje de desarrollo de facto en esta plataforma, lo hacen ideal para el desarrollo del simulador.

El compilador de C++, incluye una buena biblioteca de objetos, tanto para el desarrollo en MS/Windows como de propósito general, dicha biblioteca llamada *Microsoft Foundation Classes*, mejor conocida como MFC, es casi un estándar, siendo proporcionada por varias compañías desarrolladoras de compiladores de C++ [Icaza, 1994].

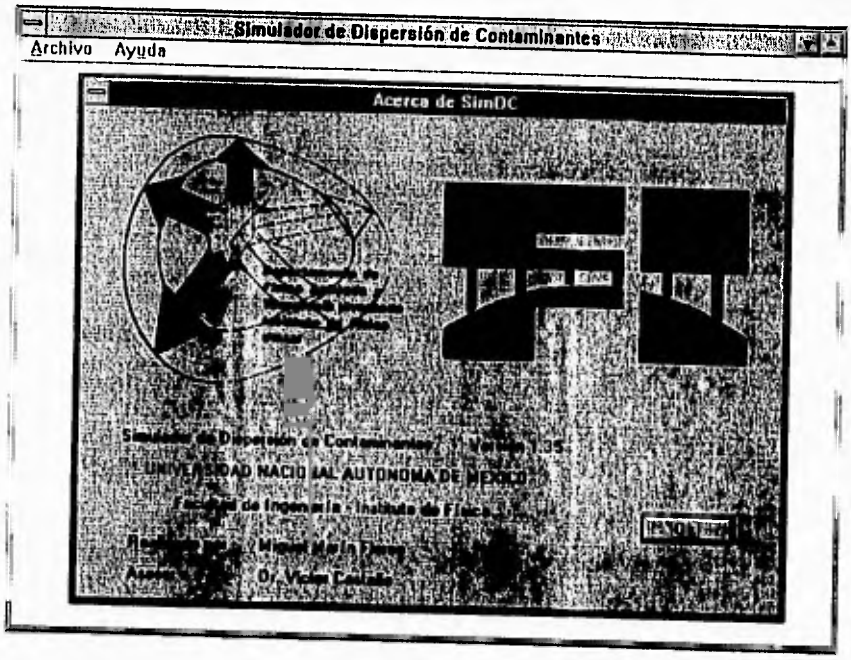
3.3 - Simulador de dispersión de contaminantes atmosféricos

El sistema simulador de contaminantes atmosféricos, tiene entre sus características:

- Interfaz humano-computadora intuitiva.
- Definición, mediante un archivo de configuración, de el autómata celular sobre el cual se realizará la simulación.
- Despliegue de regiones del autómata celular predeterminadas y definidas por el usuario.
- Ejecución de la simulación de manera automática (ejecutar varias iteraciones del autómata celular sin intervención del usuario) o a decisión del usuario, realizar la simulación paso a paso.
- Ayuda en línea.
- Impresión de las regiones mostradas en pantalla de la simulación en ejecución.

Es importante mencionar que el sistema de simulación al operar en un ambiente estándar (en este caso MS/Windows), "hereda" las principales características de este ambiente, tales como, cuadros de diálogo estándar, manejo de menús, teclas de función, operación de dispositivos apuntadores (i.e. Mouse), etc. Esto a nivel usuario es benéficioso, ya que no tiene que

operar con interfaces desconocidas o aprenderse comandos complejos para trabajar con el simulador.



Pantalla que muestra el despliegue del cuadro de información del simulador

La principal vía de interacción del usuario con el simulador es mediante el uso del menú de opciones que aparece en la parte superior de la ventana del sistema; con el uso de estas opciones el usuario puede, por ejemplo:

- Abrir el archivo de configuración del autómata celular (menú Archivo - Abrir).

- **Desplegar regiones predeterminadas dentro del simulador (menú Vistas - Vistas Externas).**

- **Desplegar regiones definidas por el mismo (menú Vistas - Región).**

- **Ejecutar un paso de la simulación (menú Ejecuta - Paso).**

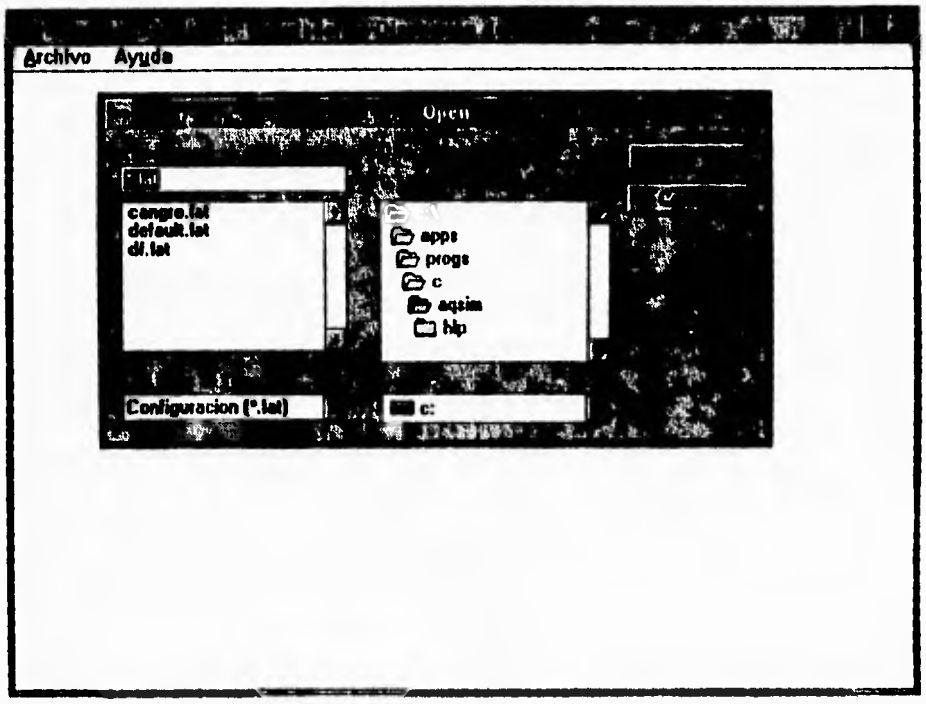
- **Ejecutar varios pasos en forma consecutiva (menú Ejecuta - Animación).**

- **Ver información acerca del simulador de dispersión de contaminantes (menú Ayuda - Acerca de SimDC).**

- **Consultar la ayuda en línea (menú Ayuda - Contenido).**

El esquema general (entradas/salidas) del programa de simulación se muestra a continuación:

Esquema de Entradas/Salidas del Simulador de Dispersión de Contaminantes

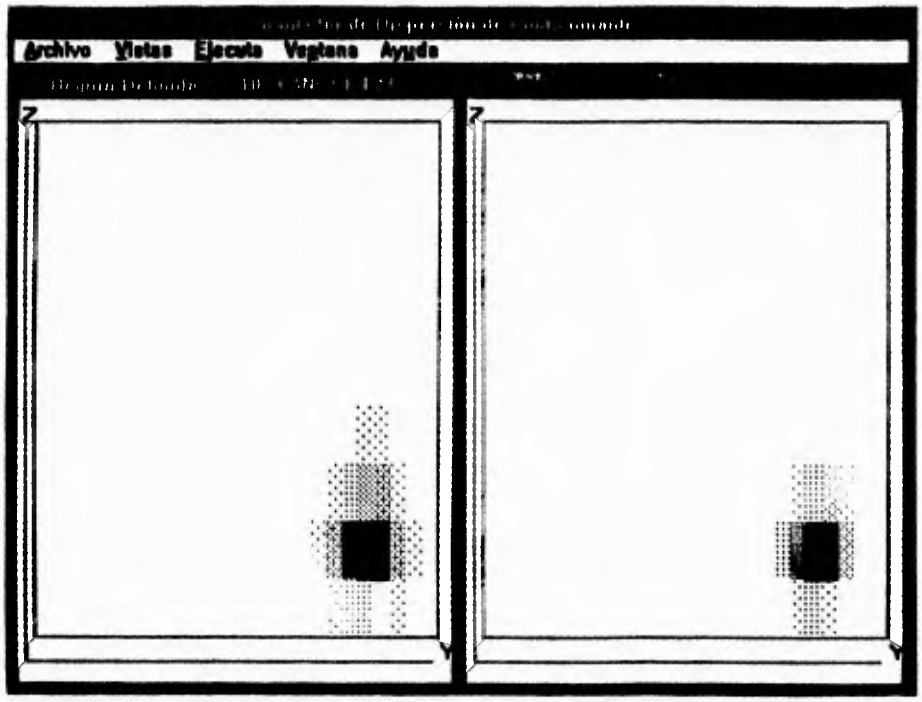


El sistema "hereda" características de la plataforma de desarrollo, tal es el caso del cuadro de dialogo que se usa para abrir un archivo de configuración

Una parte importante del desarrollo del simulador fue el diseño del archivo de configuración del autómata celular, este es un archivo tipo texto (el cual puede ser escrito con cualquier utilería de edición, i.e. el programa Edit del MS-DOS) en el que podemos especificar, de acuerdo a una sintaxis perfectamente definida, todas las características que debe cumplir el autómata celular a especificar, tales como: el valor de los parámetros de la regla de actualización ya sea para un conjunto de células

o en forma individual, la cantidad de células que forman al autómata celular, el valor máximo de los estados de las células, etc.

Debido a la importancia de este archivo de configuración y a que en lo sucesivo utilizaremos este archivo para mostrar la configuración de los autómatas celulares utilizados en este trabajo, consideramos conveniente realizar una pequeña revisión de sus características más importantes.

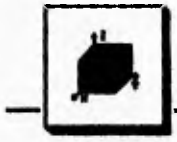


Vista de una simulación del autómata celular definido en el archivo cangre.lat

El archivo de configuración del AC

Un ejemplo de un archivo de configuración se muestra a continuación:

```
; Archivo de configuracion de un Autómata Celular
; Archivo de Prueba
; Abril 24 de 1995

[ 3, 5, 2 ]
Estado = 90000000
Alfa = 1
Beta = 0
Gama = 0
Fi = 0
dx = 0
dy = 0
dz = 0

[ 0, 0, 0 -> 6, 6, 6]
Estado = 23
Alfa = 0.3

[Global]
CelulasX = 14
CelulasY = 17
CelulasZ = 9
Alfa = 0.3
Beta = 0.1
Gama = 0.3
Fi = 0.3
dx = 0.0
dy = 0.0
dz = 0.0
Estado = 0
ValorMaximo = 100000000
```



La organización básica del archivo de configuración es la sección, una sección comienza generalmente con una palabra o una serie de números encerrados entre paréntesis cuadrados (o corchetes) y termina al inicio de otra sección o al terminar el archivo. El cuerpo de una sección (las líneas de texto que se encuentran entre una sección y otra, o una sección y el fin del archivo) esta compuesta de variables que definen las características principales del autómata, las cuales veremos más adelante.

Secciones en el archivo de Configuración

En la primera sección llamada *Global* se especifican las características que tendrá todo el autómata celular a definir, aquí se especifica la dimensión del AC, los valores de los parámetros de la regla de actualización para cada célula, el valor máximo que tendrán las células del AC y el estado inicial de todas las células del AC. Esta sección se define como [Global] y es la última sección definida en la configuración de ejemplo que se muestra en páginas anteriores.

En ocasiones cuando se está definiendo un AC, deseamos que una célula tenga características diferentes a las demás, cuando se quiere definir una célula con características diferentes, podemos usar una sección especial y especificar en esta sección sus características particulares (i.e. un estado de inicio diferente al de las demás, etc.). La sección para definir características especiales a una célula le llamaremos *sección por célula*, siendo la sintaxis para escribir este tipo de sección:

[X, Y, Z]

Donde:

X, Y, Z corresponden a las coordenadas en x,y,z donde se encuentra la célula dentro del AC

Si quisiéramos especificar las mismas características pero a un conjunto de células, utilizaremos lo que llamamos *sección por región*, cuya sintaxis es:

[Xi, Yi, Zi -> Xf, Yf, Zf]

Donde:

Xi, Yi, Zi son las coordenadas de inicio de la región a definir

Xf, Yf, Zf son las coordenadas finales de la región a definir

Además

$Xi \leq Xf,\ Yi \leq Yf,\ Zi \leq Zf$

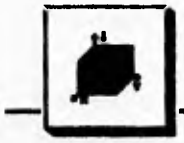
Para asignar valores a cualquiera de las variables que podemos especificar dentro de una sección, utilizaremos la siguiente sintaxis:

Variable = Número

Las variables que podemos definir dentro de las secciones del autómata celular son:

CelulasX

Esta variable indica cuantas células a lo largo del eje X tiene el AC, esta variable solo puede definirse dentro de la sección Global.



CelulasY

Esta variable indica cuantas células a lo largo del eje Y tiene el AC, esta variable solo puede definirse dentro de la sección Global.

CelulasZ

Esta variable indica cuantas células a lo largo del eje Z tiene el AC, esta variable solo puede definirse dentro de la sección Global.

ValorMaximo

Esta variable contiene el valor máximo que puede alcanzar el estado de cualquier célula del AC, esta variable sólo se puede definir dentro de la sección Global.

Estado

Esta variable puede definirse en cualquier sección del archivo de configuración, contiene el valor del estado inicial de las célula, región o autómata celular dentro de la cual se encuentre definida.

Alfa, Beta, Gama, Fi, Dx, Dy, Dz

Estas variables corresponden a los parámetros de la regla de actualización definida en el capítulo anterior, sus efectos dependen, al igual que la variable Estado, de la sección dentro de la cual se encuentren definidas.

El estado y los parámetros de la regla de actualización son asignados de acuerdo a una precedencia específica, por ejemplo, para el archivo de configuración mostrado en páginas anteriores, tenemos:

- **El estado inicial de la célula [1, 1, 1] es 23.**
- **El valor del parámetro Beta para la célula [1, 1, 1] es 0.1.**
- **El valor de Alfa para la célula [3, 5, 2] es 1.**
- **El estado inicial de la célula [10, 10, 7] es 0.**

Como podemos ver del ejemplo, los parámetros que corresponden a la regla de actualización de cada célula del AC, tienen asignado de inicio los valores de las variables definidas en la sección *Global*, sin embargo, si una célula se encuentra dentro de una *sección por región*, los parámetros de la regla de actualización de ésta toma los valores de las variables definidas en la región, finalmente, si una célula tiene una sección especial (*sección por célula*), los parámetros de la regla de actualización de esta célula toman el valor de las variables definidas en esta sección, no importando si la célula se encuentra dentro de la definición de una *sección por región*.

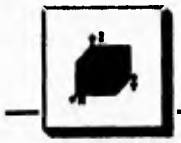
Para escribir comentarios en el archivo de configuración, estos deben comenzar con un punto y coma (;) al inicio de la línea donde se escribirá (vea el archivo de configuración de ejemplo mostrado en páginas anteriores).

4

Análisis y Pruebas del Modelo de Simulación

FALTA PAGINA

No62a la63



En el capítulo dos, mencionamos ciertas inconsistencias del modelo propuesto (específicamente la regla de actualización de estados del autómata celular) por Giorgio Guariso y Vittorio Maniezzo [Guariso, 1991], estas inconsistencias nos llevaron a proponer una ecuación alternativa al modelo original, la cual, una vez implementada (capítulo 3), analizaremos más detalle en este capítulo.

Una vez obtenido el modelo y desarrollado el programa de simulación, realizaremos un análisis de los efectos que tienen los parámetros de la regla de actualización sobre el comportamiento total del autómata celular, una vez conocidos tales efectos, podemos entonces variar dichos parámetros para lograr que el autómata celular se comporte de acuerdo a características deseadas. En el capítulo 5 mostraremos un caso práctico en el cual se aplican los resultados obtenidos para simular una situación real de contaminantes dispersos en la atmósfera.

4.1 - Análisis de comportamiento del AC

El análisis de comportamiento del autómata celular se realizará con base en los parámetros de la regla de actualización de la ecuación (7) desarrollada en el capítulo 2; debido a que la diferencia entre las ecuaciones (4) y (7) es de un intercambio en el orden en el que operan los parámetros β y dz, el comportamiento del autómata celular con la regla de actualización (4) es la misma al de la ecuación (7) si intercambiamos dz por β.

Para analizar el comportamiento del autómata celular se acordó fijar todos los parámetros a un valor tomado de manera aleatoria y variar solo uno de ellos. Los valores de los parámetros fueron fijados en valores entre

0 y 1[1], esto se debe principalmente a que el uso de valores mayores a uno, ocasiona un incremento más acelerado de ΔC_{ijk}, lo que hace que el autómata alcance rápidamente lo que llamaremos nivel de saturación[2].

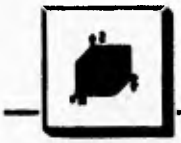
La configuración del autómata celular fue la siguiente:

- El número de células a lo largo del eje X = 13.
- El número de células a lo largo del eje Y = 17.
- El número de células a lo largo del eje Z = 9.
- Una fuente constante de contaminación en la celda X = 3, Y = 5 y Z = 2 (la denotaremos como [3, 5, 2]), ésta se logra fijando el valor del parámetro Alfa de la regla de actualización de dicha célula en uno y los demás parámetros se fijan en 0.
- Un valor de M = 100,000,000.

---

[1] Recuerde que el valor de los estados internos de las células del autómata celular se encuentra entre 0 y un valor muy grande M (vea el capítulo 2).

[2] El nivel de saturación del AC indica que el estado de una o varias células que forman parte del AC, han alcanzado o sobrepasado el valor M.



- **El estado inicial de la célula [3, 5, 2] fue 90,000,000, este valor fue fijado de forma tal que el autómata celular evolucionara rápidamente en un número pequeño de iteraciones.**

- **El estado inicial de todas las demás células fue 0.**

- **El número de pasos o iteraciones del autómata celular fue de 10, valor tomado después de realizadas varias simulaciones y en el cual, el autómata celular mostraba ya una clara tendencia en su comportamiento, siendo no necesarias más iteraciones.**

El archivo de configuración del Autómata Celular que se usa como entrada en el programa de simulación (vea el capítulo 3) es:

```
; Archivo de definición del AC
; Archivo de Prueba para al analisis de comportamiento
; del Autómata Celular

; Fuente de Contaminación Constante
[ 3, 5, 2 ]
Estado = 90000000
Alfa = 1
Beta = 0
Gama = 0
Fi = 0
dx = 0
dy = 0
dz = 0

; Características Globales del AC
[Global]
CelulasX = 14
CelulasY = 17
CelulasZ = 9
Alfa = 0.3
```

Beta = 0.3
Gama = 0.3
Fi = 0.3
dx = 0.0
dy = 0.0
dz = 0.0
Estado = 0
ValorMaximo = 100000000

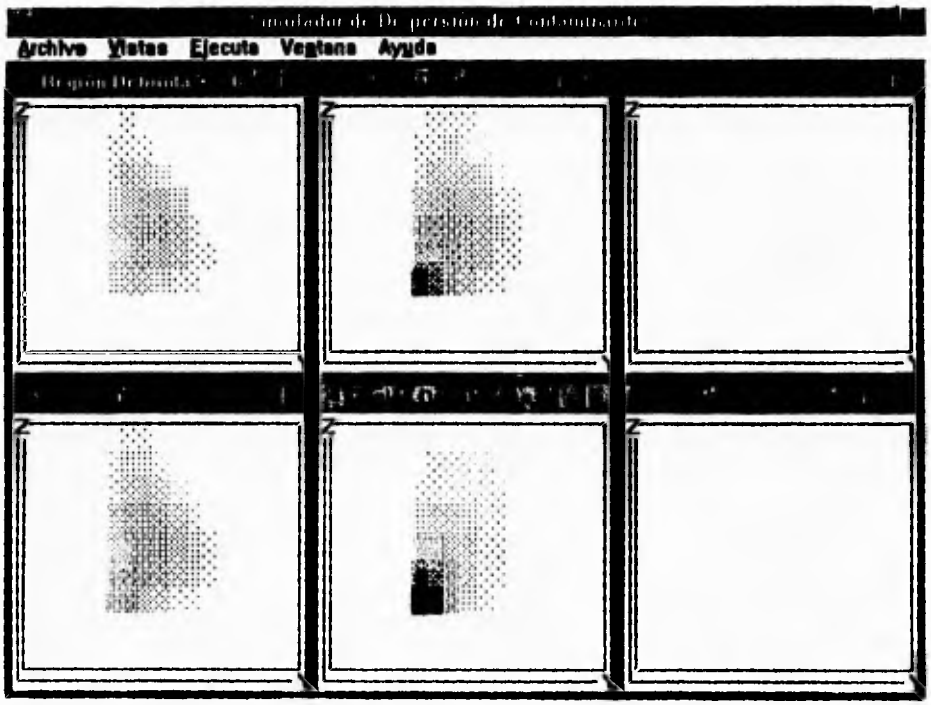
A continuación presentamos los resultados de las primeras simulaciones realizadas con la configuración del autómata celular mostrada anteriormente, en estas simulaciones, se varió únicamente el valor de un parámetro (Alfa y después Beta) fijándose los demás parámetros en valores tomados al azar.

Primera Simulación - Variando Alfa

Alfa = 0.0
Beta = 0.3
Gama = 0.3
Fi = 0.3
dx = 0.0
dy = 0.0
dz = 0.0

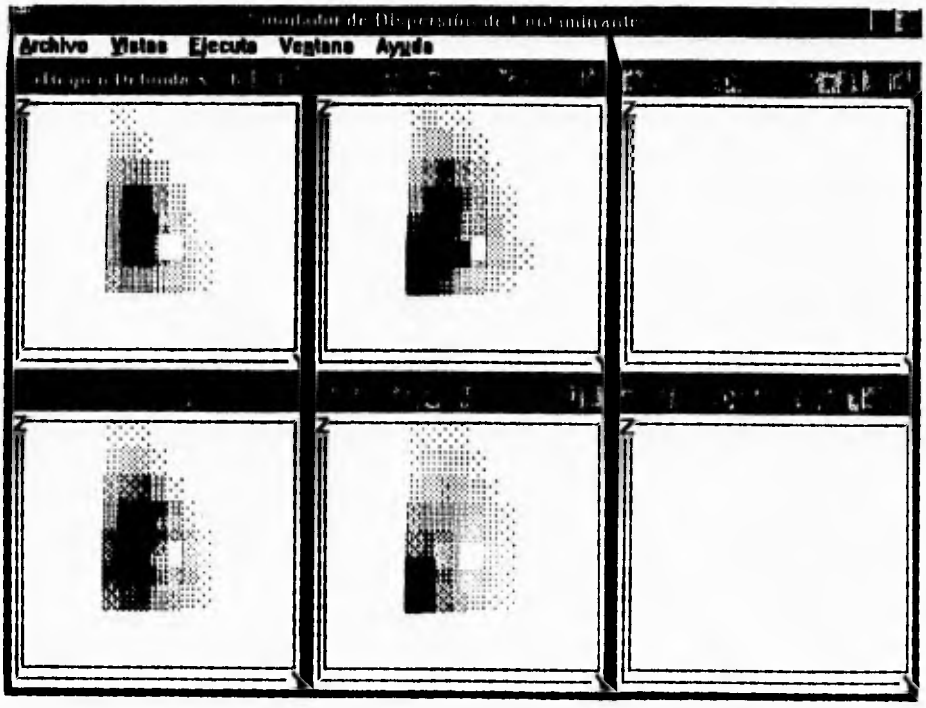
Segunda Simulación - Variando Alfa

Alfa = 0.1
Beta = 0.3
Gama = 0.3
Fi = 0.3
dx = 0.0
dy = 0.0
dz = 0.0



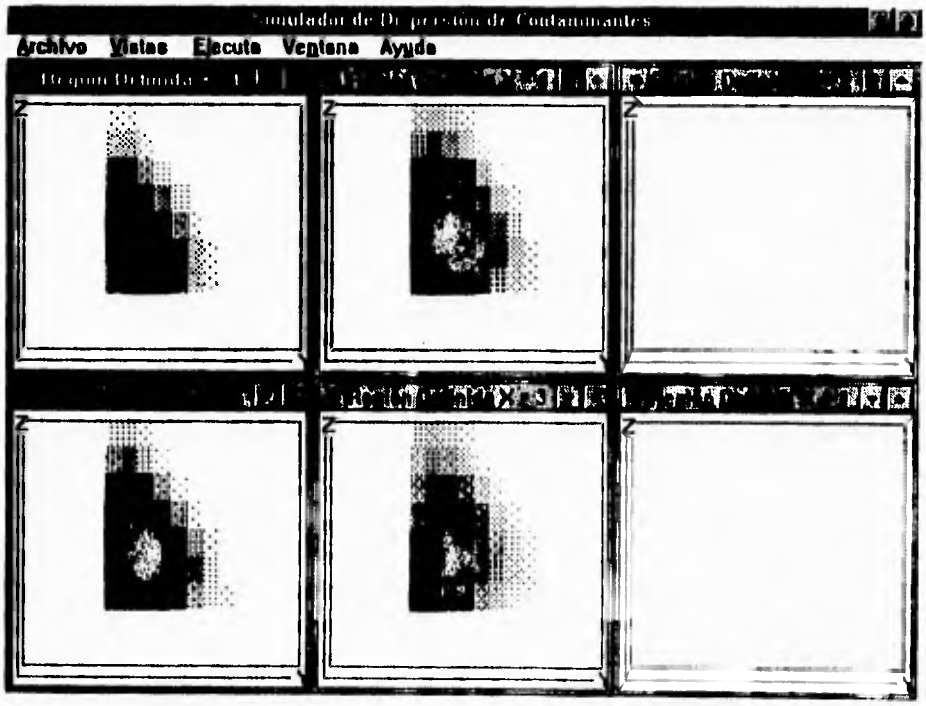
Tercera Simulación - Variando Alfa

Alfa = 0.3
Beta = 0.3
Gama = 0.3
Fi = 0.3
dx = 0.0
dy = 0.0
dz = 0.0



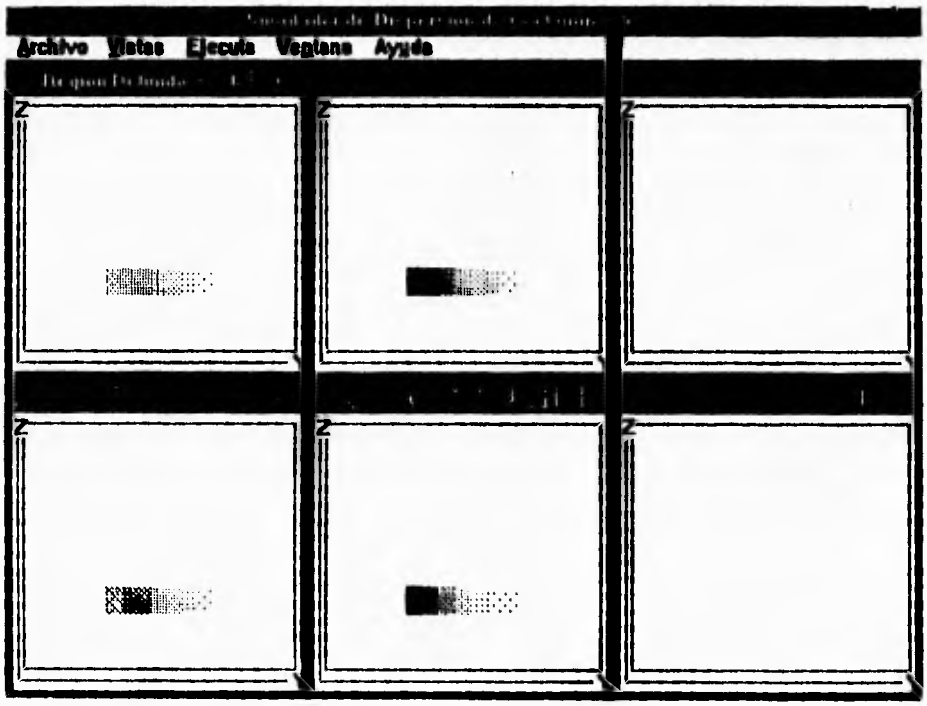
Cuarta Simulación - Variando Alfa

Alfa = 0.6
Beta = 0.3
Gama = 0.3
Fi = 0.3
dx = 0.0
dy = 0.0
dz = 0.0



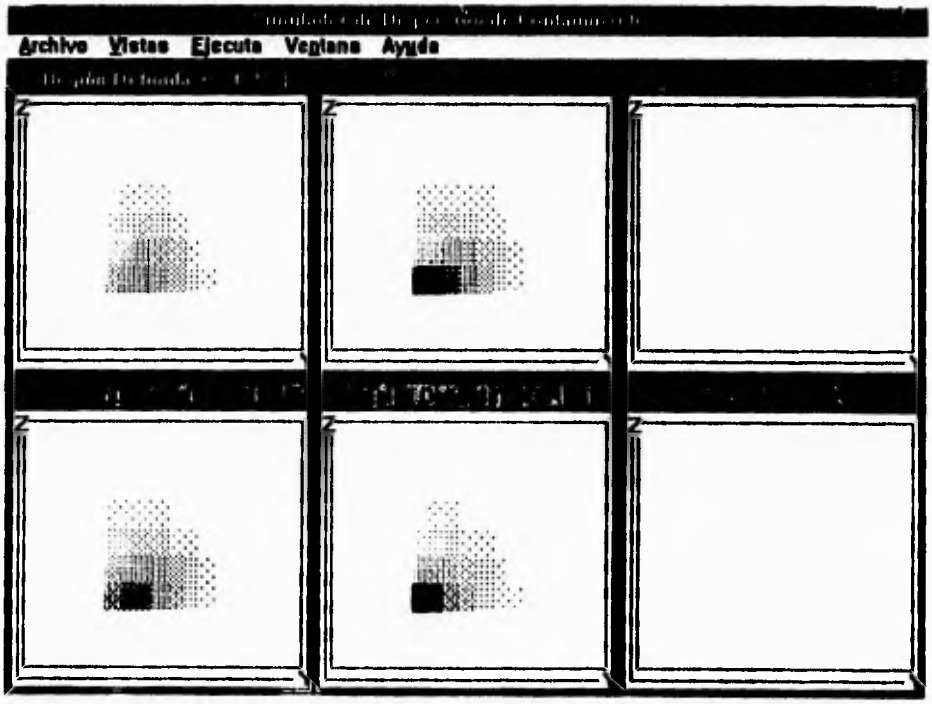
Primera Simulación - Variando Beta

Alfa = 0.3
Beta = 0.0
Gama = 0.3
Fi = 0.3
dx = 0.0
dy = 0.0
dz = 0.0



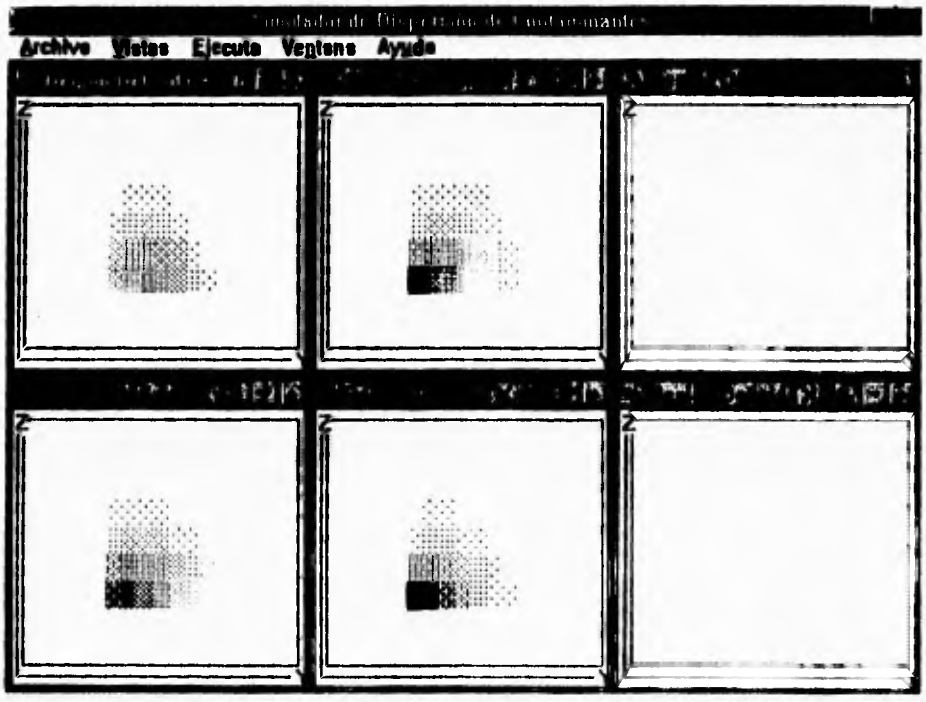
Segunda Simulación - Variando Beta

Alfa = 0.3
Beta = 0.1
Gama = 0.3
Fi = 0.3
dx = 0.0
dy = 0.0
dz = 0.0

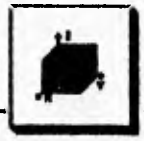


Tercera Simulación - Variando Beta[3]

Alfa = 0.3
Beta = 0.6
Gama = 0.3
Fi = 0.3
dx = 0.0
dy = 0.0
dz = 0.0



[3] No se realiza simulación para Beta = 0.3, ya que esta fue reali



De acuerdo a lo observado en las simulaciones anteriores, existen claros comportamientos del autómata celular que dependen directamente de la variación de los parámetros de la regla de actualización interna de las células que lo conforman.

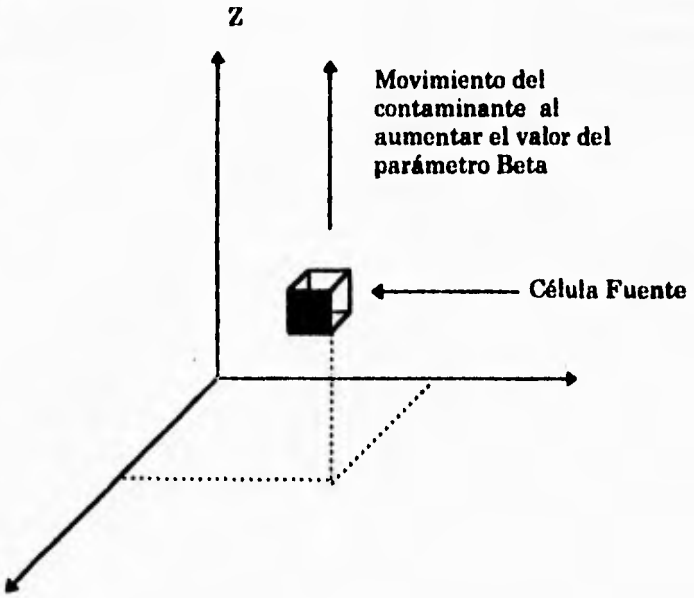
Debido a lo extenso de las pruebas realizadas, en las siguientes páginas resumiremos los resultados obtenidos de las simulaciones realizadas sobre el autómata celular de prueba propuesto en páginas anteriores.

Variación de Alfa

El parámetro Alfa, como podemos observar, influye directamente en la concentración del contaminante por célula, en otras palabras, un valor mayor de Alfa para una célula en específico, significa una incremento notable en la concentración de contaminante en ésta conforme avanza la simulación (mayor número de iteraciones).

Variación de Beta

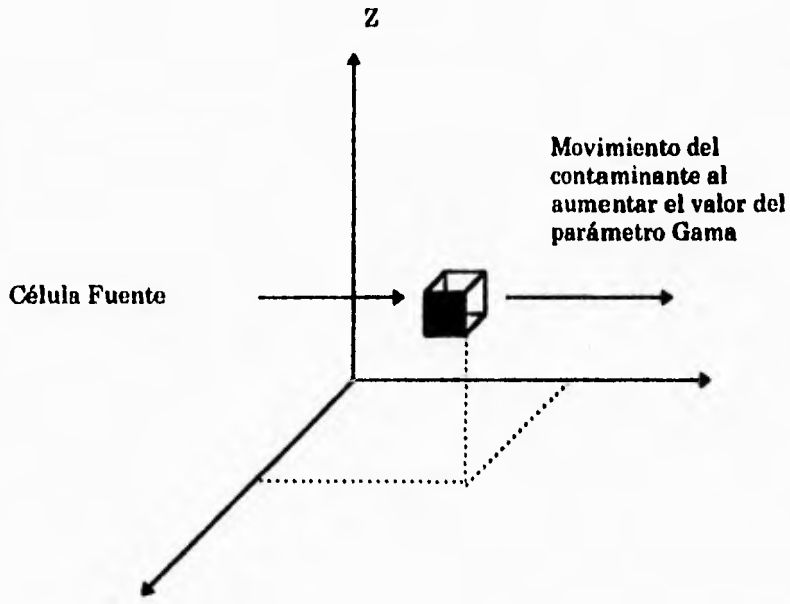
En las simulaciones donde variamos el parámetro Beta, podemos observar que se afecta significativamente el movimiento del contaminante a lo largo del eje Z, a valores mayores de Beta, se acelera la dispersión del contaminante hacia las capas superiores del autómata celular; de manera gráfica, el comportamiento del contaminante de acuerdo a variaciones del parámetro Beta puede resumirse como se muestra en la siguiente figura:



Efecto del Parámetro Beta en el movimiento del contaminante de la célula fuente

Variación de Gama

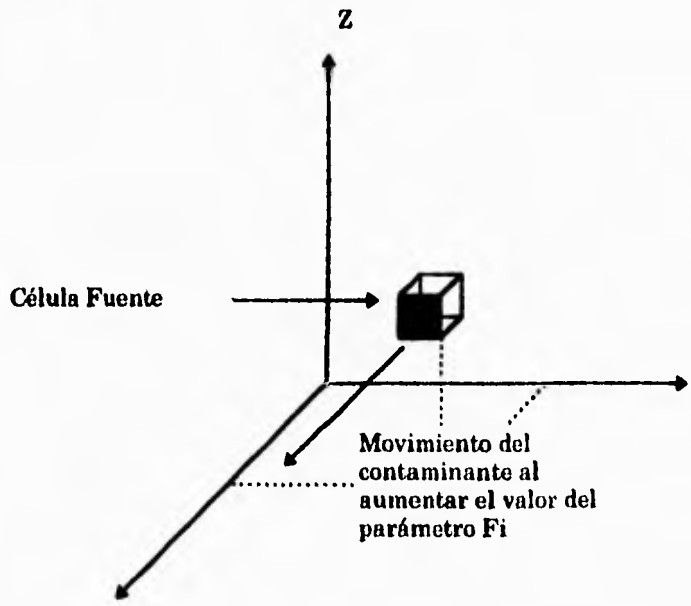
Gama afecta el movimiento del contaminante a lo largo del eje Y, a valores mayores de Gama, el contaminante tiende a alejarse del origen del sistema coordenado (en dirección de Y positiva) formado por las células del autómata celular, gráficamente tenemos:



Efecto del Parámetro Gama en el movimiento del contaminante de la célula fuente

Variación de Fi

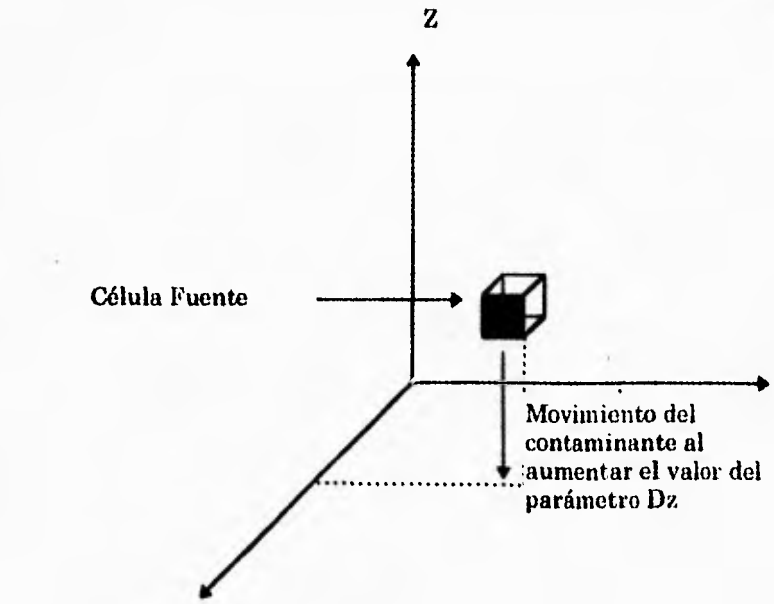
Fi afecta el movimiento del contaminante a lo largo del eje X, a valores mayores de Fi, el contaminante tiende a alejarse del origen del sistema coordenado (en dirección de X positiva) formado por las células del autómata celular, gráficamente tenemos:



Efecto del Parámetro Fi en el movimiento del contaminante de la célula fuente

Variación de Dz

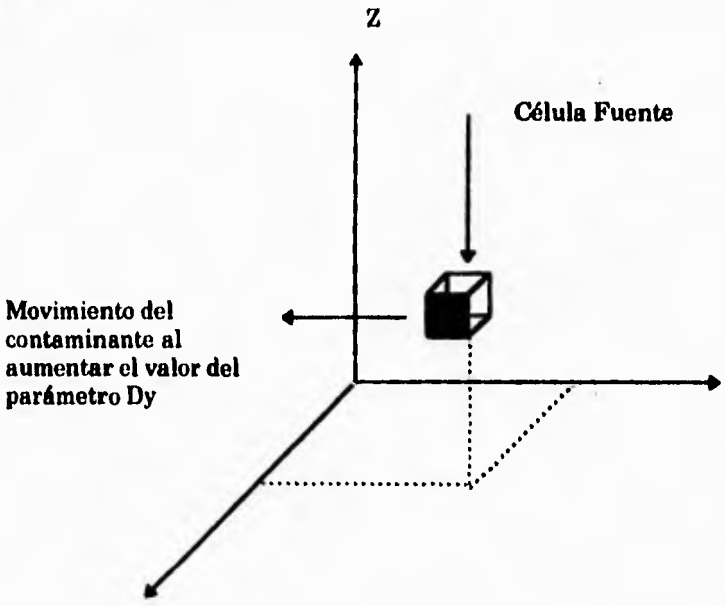
Dz tiene un efecto contrario al realizado por Beta, esto es, al aumentar el valor de Dz, el contaminante tiende a desplazarse hacia el origen del sistema coordenado formado por el autómata celular (esto obviamente a lo largo del eje Z).



Efecto del Parámetro Dz en el movimiento del contaminante de la célula fuente

Variación de Dy

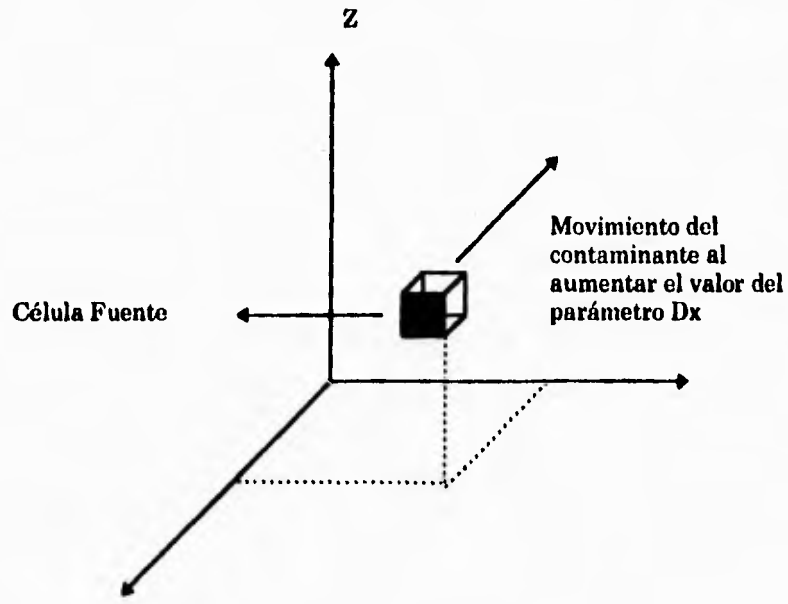
Dy tiene un efecto contrario al realizado por Gama, esto es, al aumentar el valor de Dy, el contaminante tiende a desplazarse hacia el origen del sistema coordenado formado por el autómata celular (esto obviamente a lo largo del eje Y).



Efecto del Parámetro Dy en el movimiento del contaminante de la célula fuente

Variación de Dx

Dx tiene un efecto contrario al realizado por Fi, esto es, al aumentar el valor de Dx, el contaminante tiende a desplazarse hacia el origen del sistema coordenado formado por el autómata celular (esto obviamente a lo largo del eje X).



Efecto del Parámetro Dx en el movimiento del contaminante de la célula fuente

Una vez conocido el comportamiento del autómata celular con base en los parámetros que forman la regla de actualización, podemos, de acuerdo a un comportamiento específico deseado, variar los valores de dichos parámetros para simularlo; en el capítulo siguiente se presentan los resultados obtenidos de una simulación realizada dada una situación de dispersión de contaminantes real.

5

Aplicación del Modelo

FALTA PAGINA

No 82 a la 83

Para evaluar la funcionalidad del modelo de dispersión de contaminantes así como del programa de simulación desarrollado, el paso inmediato es tratar de aplicarlo a una situación real donde los resultados obtenidos de las simulaciones realizadas deben, si bien no ser las mismas, semejar el comportamiento de la situación a modelar.

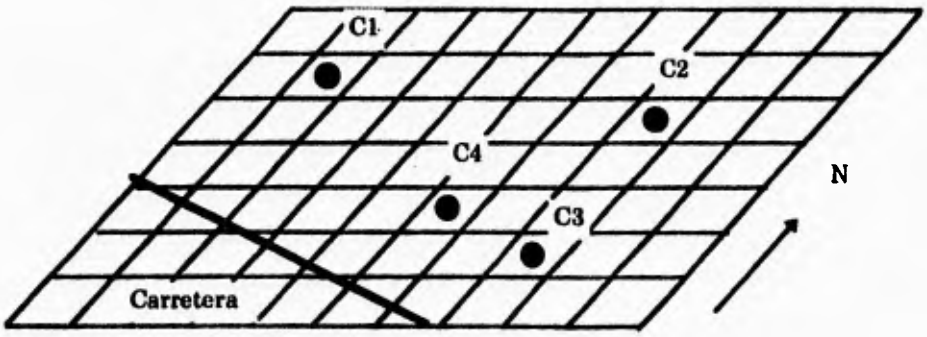
5.1 - Situación Real de Dispersión de Contaminantes

El fenómeno de dispersión elegido para ser simulado, se basa en varios estudios realizados por investigadores del Instituto de Física de la Universidad Nacional Autónoma de México en las instalaciones de PEMEX Petroquímica en Coatzacoalcos, Ver. [Martínez, 1994].

Dicho estudio, realizado entre otros por el Dr. Víctor Castaño director de este trabajo de tesis, consiste principalmente en el análisis de los efectos que tienen los diversos contaminantes dispersos en la atmósfera sobre las estructuras de los complejos petroquímicos, proponiendo, con base en estos estudios, una estrategia de protección contra la corrosión atmosférica del lugar.

PEMEX Petroquímica tiene tres plantas importantes en Coatzacoalcos: los complejos La Cangrejera, Pajaritos y Morelos, cada complejo es estudiado por separado mediante el análisis de placas de acero de exposición controlada, platos de sulfatación, etc., las cuales fueron colocadas en diversos puntos geográficos de cada uno de los complejos petroquímicos antes mencionados.

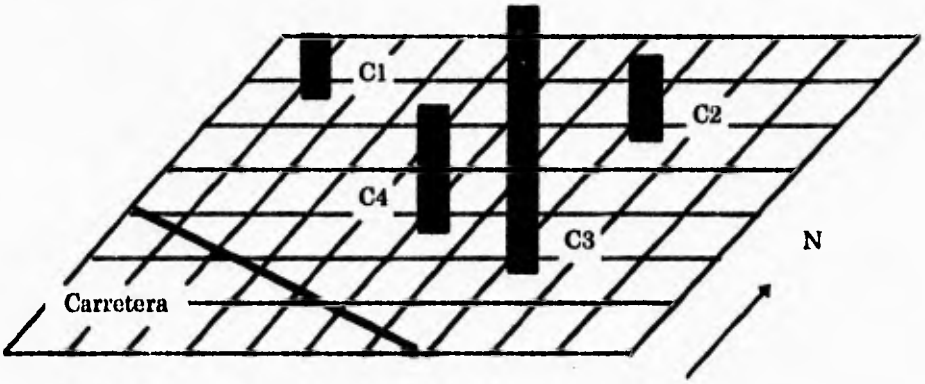
La gráfica que a continuación se presenta, muestra la distribución geográfica aproximada de los platos de sulfatación colocados en el complejo petroquímico La Cangrejera, esta representación es únicamente esquemática y sólo es usada con fines ilustrativos.



Distribución de los platos de sulfatación para el complejo petroquímico La Cangrejera

El análisis de los platos de sulfatación y placas de acero, después de un periodo de exposición determinado, se enfocó principalmente a estudiar las principales características del material tales como pérdida de peso, color y forma de la corrosión así como la presencia de azufre, cloro y fósforo en las mismas.

En la siguiente gráfica se muestra de manera esquemática, la distribución de las concentraciones de bióxido de azufre obtenidas del análisis de los platos de sulfatación de las estaciones C1 a C4.

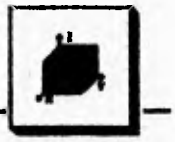


Distribución de bióxido de azufre en los platos de sulfatación del complejo petroquímico La Cangrejera

Para comprender mejor esta gráfica, mostramos a continuación una tabla donde se muestran las cantidades de bióxido de azufre (SO_2) en cada una de las estaciones de monitoreo atmosférico.

Estaciones	Bióxido de azufre
C1	8.03
C2	8.72
C3	24.53
C4	9.42

Tabla de valores de SO_2 depositados en las estaciones de monitoreo atmosférico en La Cangrejera



Si bien el estudio realizado arroja más resultados para las diferentes estaciones de monitoreo de este complejo petroquímico, solo tomaremos este resultado para realizar la simulación del fenómeno de dispersión de este contaminante (bióxido de azufre).

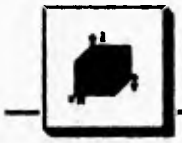
Para el caso del complejo petroquímico La Cangrejera, es pertinente realizar algunas observaciones, las cuales son sin lugar a dudas, de suma importancia para la configuración del autómata celular que realizará la simulación, entre estas tenemos:

- **Cerca de la estación de monitoreo atmosférico C3, se localizan tanques de crudo, a los cuales se les puede asociar el alto nivel de azufre en esta estación debido, principalmente, a los desprendimientos de sulfuro de hidrógeno de los crudos.**

- **Cerca de C4 existe una torre de enfriamiento y un turbogenerador, la torre de enfriamiento emana rocío lo que quizá contribuye a la fijación de componentes de cloro y azufre en los platos de sulfatación y placas de acero de esta estación de monitoreo.**

5.2 - Simulación

Una vez establecida la situación real a simular, el paso siguiente es configurar el autómata celular que realizará la simulación del fenómeno de dispersión presentado en la sección anterior.



Debido a la inexistencia de datos que nos indiquen las extensiones geográficas del complejo La Cangrejera, utilizaremos una escala basada en la gráfica de distribución de estaciones de monitoreo atmosférico presentada en la sección anterior.

Las características principales del autómata celular que se definió para realizar las simulaciones son:

- La dimensión del autómata celular es [14, 25, 9].

- Se selecciona la célula [11, 22, 1] como la fuente principal de contaminación, este punto intenta simular los tanques de crudo que se encuentran cerca de la estación de monitoreo atmosférico C3.

- Se fijan valores para los parámetros de la regla de actualización, de forma tal que simulen viento constante de Sur a Norte y una dispersión hacia el Este también constante.

- Existe una célula con todos los valores de sus parámetros en cero, esto con el fin de simular la presencia de los tanques de crudo, justo por debajo de la fuente constante de contaminación [11, 22, 0].

- A la células que forman la región [8, 15, 0 -> 9, 16, 2], se les asigno un valor al parámetro Alfa mayor que el de las demás células del autómata, este valor de Alfa permitirá simular la

fijación de componentes de azufre y cloro por el rocío emanado de la torre de enfriamiento de la que se habló en páginas anteriores.

- Un valor de M = 100,000,000.

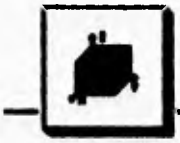
- El estado inicial de la célula [11, 22, 1] fue 90,000,000, valor tomado de las simulaciones de prueba mostradas en el capítulo anterior.

- El número de iteraciones realizadas fue de 45.

Una primera aproximación del autómata celular, se muestra mediante el archivo de configuración de entrada para el simulador[1].

```
; Archivo de definicion del AC
; Para el complejo La Cangrejera

[ 11, 22, 1 ]
Estado = 90000000
Alfa = 1
Beta = 0
Gama = 0
Fi = 0
dx = 0
dy = 0
dz = 0
```

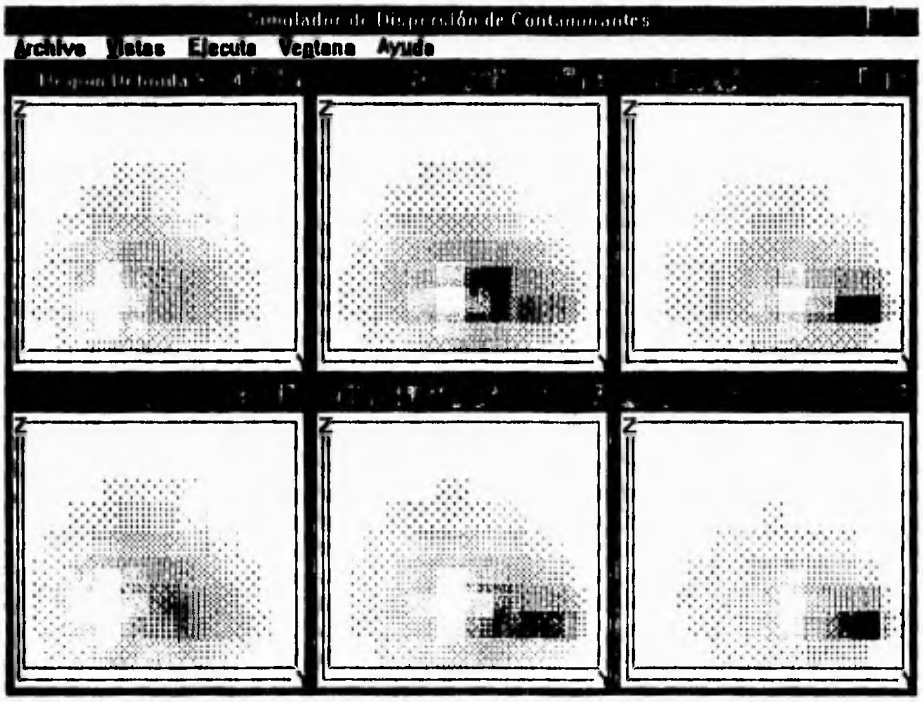
[1] Las características mencionadas anteriormente y el archivo de definición del AC, fueron seleccionados después de varios intentos de configuraciones posibles, por motivos de espacio, mostramos sólo dos archivos de configuración y sus resultados correspondientes.



[11, 22, 0]
Estado = 0
Alfa = 0
Beta = 0
Gama = 0
Fi = 0
dx = 0
dy = 0
dz = 0

[8, 15, 0 -> 9, 16, 2]
Estado = 0
Alfa = 0.45

[Global]
CelulasX = 14
CelulasY = 25
CelulasZ = 9
Alfa = 0.30
Beta = 0.05
Gama = 0.05
Fi = 0.05
dx = 0.2
dy = 0.36
dz = 0.02
Estado = 0
ValorMaximo = 100000000



Primera aproximación para la simulación del fenómeno de dispersión atmosférica en La Cangrejera

El análisis de esta primera aproximación permite observar ciertos comportamientos no deseados como:

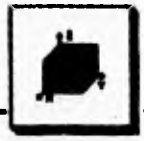
1. **La simulación de fijación de contaminantes en la región específica [8, 15, 0 -> 9, 16, 2], es excesiva, lo que hace que exista una presencia mayor de contaminantes en las regiones cercanas al origen del AC.**

2. La dispersión a lo largo del eje X en dirección al origen del AC es ligeramente mayor a la esperada.

3. El contaminante tiende a permanecer mucho tiempo en las capas superiores del autómata, por lo que debe de afectarse el movimiento a lo largo del eje Z, de forma tal que la dispersión hacia las capas inferiores del autómata (valores de Z cercanos a cero) debe ser más significativa.

Después de varias pruebas mas sobre este archivo de configuración, llegamos al autómata celular que a nuestro juicio simula de la manera más aproximada el fenómeno observado, el archivo de configuración del AC y su salida son:

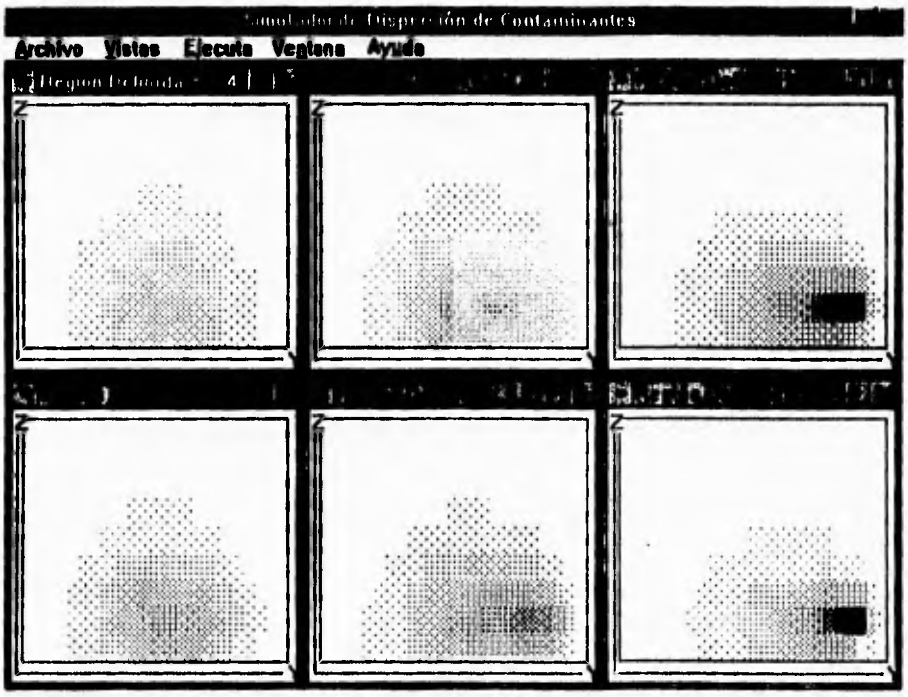
```
; Archivo de definicion del AC
; Para el complejo Cangrejera

[ 11, 22, 1 ]
Estado = 90000000
Alfa = 1
Beta = 0
Gama = 0
Fi = 0
dx = 0
dy = 0
dz = 0


[11, 22, 0]
Estado = 0
Alfa = 0
Beta = 0
Gama = 0
Fi  = 0
```



dx = 0
dy = 0
dz = 0

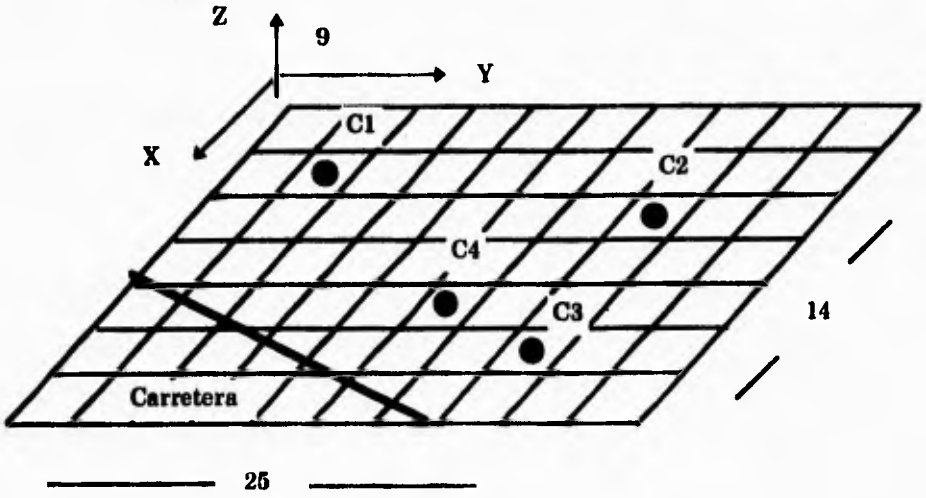
[8, 15, 0 -> 9, 16, 2]
Estado = 0
Alfa = 0.36

[Global]
CelulasX = 14
CelulasY = 25
CelulasZ = 9
Alfa = 0.30
Beta = 0.05
Gama = 0.05
Fi = 0.05
dx = 0.185
dy = 0.32
dz = 0.08
Estado = 0
ValorMaximo = 100000000



Aproximación final para la simulación del fenómeno de dispersión atmosférica en La Cangrejera

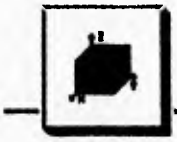
Para poder comprender la salida del simulador de dispersión de contaminantes, recuerde que las ventanas mostradas, en este caso, corresponden a planos paralelos al plano formado por los ejes YZ del autómata celular, indicándose la posición del plano sobre el eje X en el título de la ventana correspondiente (i.e. Región Definida X = 4), en la siguiente figura, se muestra la relación de los ejes coordenados del AC con la región geográfica del complejo petroquímico La Cangrejera mostrado en páginas anteriores.



Posición de los ejes del AC con respecto a la región geográfica del complejo petroquímico La Cangrejera

Como podemos observar, el autómata celular simula los aspectos más importantes del fenómeno de dispersión real, así mismo, muestra información adicional acerca del fenómeno, por ejemplo, la simulación realizada, muestra una región de alta concentración de contaminantes en la región que se encuentra, en el esquema original, entre las estaciones de monitoreo C2 y C1, tal y como se observa en la figura correspondiente a la última simulación realizada (vea las ventanas correspondientes a las regiones definidas en X=5 y X=4).

Sin lugar a dudas, la parte más importante en este modelo, es la calibración de los parámetros que corresponden a la regla de actualización del autómata celular, el conocimiento de los efectos que cada uno de los



parámetros tienen sobre el comportamiento del AC, permite un más rápido desarrollo del modelo final.

Una vez obtenida la configuración del autómata celular, podemos entonces, variar algunos valores de los parámetros de la regla de actualización, de forma tal que podamos simular diferentes comportamientos para la misma región, por ejemplo:

1. Variando el parámetro Alfa de la región definida como [8, 15, 0 -> 9, 16, 2], podemos simular una mayor cantidad de rocío emanado de la torre de enfriamiento (aumentando el valor de Alfa) o bien, simular que no existe la torre de enfriamiento (Alfa = 0).

2. Podemos variar el parámetro dz para simular un mayor o menor efecto de la fuerza de gravedad sobre las partículas dispersas (aumentando el valor del parámetro dz, simulamos partículas más pesadas).

3. Un incremento o decremento de la velocidad del viento ya sea en sentido positivo o negativo del eje X (lo mismo podemos realizar en el eje Y).

4. Colocar varias fuentes de contaminación.

5. Asignar valores de cero a los parámetros de una región específica, lo que nos permitirá simular irregularidades en el terreno, presencia de obstáculos (tales como muros) o bien, obstáculos mayores como montañas etc.

Estas posibilidades de variación en el comportamiento del autómata celular (especialmente el punto 5), le dan al sistema desarrollado una gran flexibilidad para la simulación de situaciones diversas, así como la posibilidad de aplicarse en una gran cantidad de fenómenos meteorológicos complejos, recordemos únicamente que los fenómenos deben ser a gran escala, esto es, con un dominio de cómputo de varios kilómetros cúbicos, y que (obviamente) dicho fenómeno se encuentre afectado por los factores físicos mencionados en el capítulo 2.

6

Conclusiones

FALTA PAGINA

No 98 a la 99

El desarrollo de programas de computadora enfocados a la simulación - especialmente de fenómenos físicos o químicos - requiere de la interacción de varias disciplinas tanto científicas como tecnológicas; tal es el caso del simulador de dispersión de contaminantes desarrollado en este trabajo, cuya base teórica se encuentra fundamentada en un tipo especial de sistemas dinámicos conocidos como autómatas celulares y de metodologías de desarrollo de sistemas de cómputo tales como el diseño y programación orientada a objetos.

Entre las principales características que presenta el programa de simulación tenemos:

- **Se desarrolló en una plataforma de hardware basada en sistemas personales con procesadores Intel 80x86, lo que le permite ser utilizado en universidades o instituciones dedicadas al estudio de fenómenos atmosféricos, con pocas posibilidades de tener computadoras con grandes capacidades de cálculo.**

- **El simulador de dispersión de contaminantes trabaja en un ambiente gráfico estándar en el mercado, esto le da al programa de simulación características especiales, tales como: ambiente de trabajo amigable, interacción estándar con el usuario (cuadros de diálogo, menúes, ventanas), despliegue puramente gráfico, etc.**

- **El programa fue desarrollado mediante técnicas de programación orientada a objetos, característica sumamente importante ya que permite un desarrollo más rápido y mejor, reusabilidad de código y por ende, una mayor flexibilidad en el caso de cambios en el programa para la realización de futuras versiones del mismo.**

- El diseño del programa contempló aspectos tales como la interacción usuario-computadora, la presentación de resultados, los tipos de entradas al programa, entre otros.

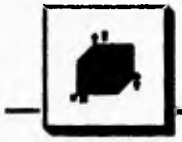
- La velocidad de proceso y despliegue de resultados es esencial en un sistema de simulación, el programa utiliza toda la memoria disponible en el sistema donde se ejecute, teniendo un excelente tiempo de respuesta, a pesar del proceso numérico realizado.

Esta y otras características dan al programa de simulación gran versatilidad en sus posibles aplicaciones, pudiendo ser usado como producto final en empresas, donde el estudio del comportamiento de los contaminantes atmosféricos que estas liberan es actividad prioritaria, o bien, en áreas de investigación ya sea en universidades o institutos.

Tanto el modelo como el programa de simulación de dispersión de contaminantes presentados en este trabajo, sientan las bases para futuras investigaciones y desarrollos en esta área, las cuales pueden enfocarse principalmente a:

- Implementar al programa de simulación un sistema de despliegue gráfico de resultados en tres dimensiones.

- Con respecto al modelo propuesto, el siguiente paso sería encontrar las relaciones que existen entre los fenómenos físicos involucrados (aceleración de la gravedad, viento, dispersión) y los parámetros de la regla de actualización.



- **Una vez encontradas las relaciones de los fenómenos físicos y los parámetros de la regla de actualización, configurar al AC desde el programa de simulación no en base a los parámetros sino en base a los fenómenos físicos que afectan al AC.**

- **Un cambio posible, y quizá el más interesante de todos ellos es, la formulación de un modelo, en donde cada célula pudiese tener una regla de actualización diferente para el cálculo de sus estados internos, este hecho, implica la posibilidad de tener comportamientos sumamente interesantes del autómata celular a nivel general.**

Para terminar, solo basta agregar que la gran diversidad y excelente calidad de conocimientos que el ingeniero en computación posee, permite al profesional en esta área de ingeniería el correcto y eficiente desarrollo de sistemas de software, especialmente, en el desarrollo de herramientas de simulación, donde sus formación en áreas físico-Matemáticas le permite una comprensión más a fondo del problema a resolver.

Apéndice A

Autómatas Celulares

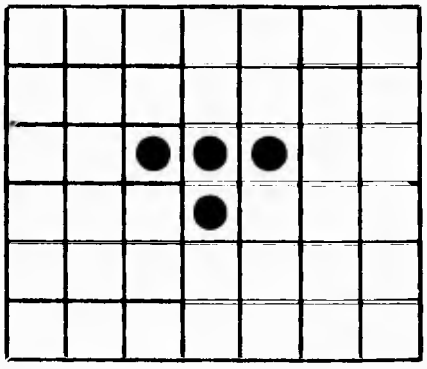
FALTA PAGINA

No 10.3 la.....

A.1 - Autómatas Celulares, un caso práctico

Uno de los autómatas celulares mejor conocidos es *life*, este es un juego matemático creado por John Horton Conway, matemático en Gonville y Caius College de la universidad de Cambridge. Este juego fue llamado así por su analogía con el origen, alteraciones y muerte de una sociedad de organismos vivos. Debido a lo simple de sus reglas evolutivas y comportamiento impredecible *life* se convierte en un extraordinario ejemplo de los conceptos generales de los autómatas celulares.

Life fué publicado por Martin Gardner en 1970 [Gardner, 1970] . La idea básica es iniciar con una simple configuración de fichas (organismos vivos) colocados sobre una superficie cuadriculada teóricamente infinita donde cada organismo ocupa únicamente una celda, entonces, observar como dichos organismos evolucionan de acuerdo a la aplicación de las llamadas *leyes genéticas* de Conway para nacimientos, muertes y supervivencia de organismos vivos.



Una posible configuración de organismos para *life*

Las reglas genéticas que Conway utilizó, fueron ideadas de tal forma que el comportamiento de la población de organismos fuese impredecible, estas reglas son:

1. *Supervivencia.* Cada organismo con dos o tres organismos vecinos sobrevive a la siguiente generación.

2. *Muertes.* Cada organismo con cuatro o más vecinos muere (es eliminada de la malla) por sobrepoblación. Cada organismo con sólo un vecino o ninguno muere por aislamiento.

3. *Nacimiento.* En cada celda vacía de la malla, adyacente a exactamente 3 organismos vecinos, no más no menos, nace, en dicha celda, un nuevo organismo.

Es importante hacer notar que todos los nacimientos y muertes ocurren simultáneamente; y juntos, constituyen lo que Conway llama una generación. Para jugar *life* Conway recomienda el siguiente procedimiento:

1. Comience con un patrón consistente de fichas negras.

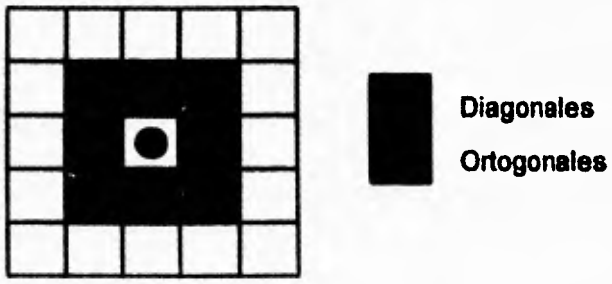
2. Localice todas las fichas que morirán. Señálelas poniendo una ficha negra encima de estas.

3. Localice las celdas vacías donde ocurrirán los nacimientos. Ponga una ficha blanca en estas celdas.

4. Después que el tablero ha sido verificado y doblemente verificado para asegurarse de que no existan errores, quite todas las fichas muertas (pilas de dos fichas negras) y reemplace todos los organismos nuevos (fichas blancas) por fichas negras.

Como mencionamos anteriormente, las muertes y nacimientos ocurren simultáneamente, los organismos nacientes no juegan ningún papel en la muerte o nacimientos de otros organismos en la generación en que ellos son creados.

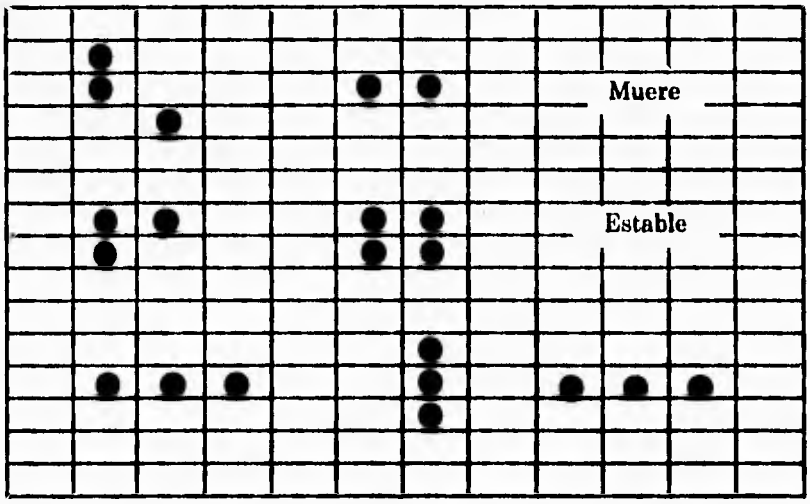
Con respecto a las celdas vecinas, cada organismo tiene ocho de ellas, cuatro adyacentes ortogonalmente y cuatro adyacentes diagonalmente.



Celdas vecinas que influyen en el nacimiento, supervivencia o muerte de la celda

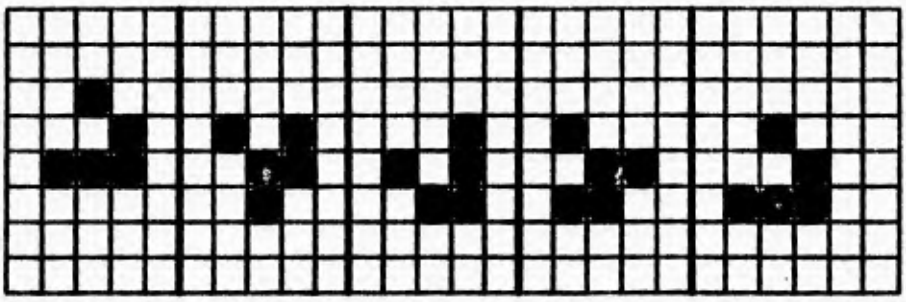
El comportamiento de diversas configuraciones de celdas es impredecible, en algunos casos la sociedad muere eventualmente, sin embargo esto no pasa hasta después de una gran cantidad de generaciones. Muchos patrones iniciales alcanzan figuras estables - Conway los llama *still lifes* - o patrones que oscilan por siempre. Patrones sin simetría inicial llegan a ser simétricos y una vez que la simetría se alcanza esta no puede perderse, más aun, puede enriquecerse.

Movimientos



Evolución de 3 configuraciones distintas

Existen ciertas configuraciones que debido a su comportamiento han llamado fuertemente la atención de los investigadores, tal es el caso del *glider* o *deslizador*.



Glider (Deslizador)

Esta configuración tiene la característica de moverse suavemente en forma diagonal, y regresar a su configuración inicial cada cuatro generaciones, lo que le permite trasladarse libremente a lo largo de la malla.

En las siguientes secciones se presenta la teoría básica de los autómatas celulares, se presupone que el lector tiene conocimiento de conceptos tales como Autómatas Finitos y Máquina de Turing.

Más adelante retomaremos nuevamente al "juego" *life* para dar una definición formal de este juego de acuerdo a la teoría de los autómatas celulares.

A.2 - Inicios de los autómatas celulares

A finales de la segunda guerra mundial John. von Neumann, quien estuvo involucrado en el diseño de las primeras computadoras, propuso diseñar la teoría de las redes de autómatas como una herramienta matemática para el estudio de la computación en paralelo.

Finalmente en 1948 introdujo la teoría de autómatas celulares para comparar el autómata natural (seres vivos) y el autómata *artificial* (computadoras). John von Neumann trató en un intento por abstraer la estructura de la vida, diseñar una red de autómatas que pudiera exhibir una de las mayores características de la vida: *autoreproducción*.

En dicho trabajo John. von Neumann introduce la noción de espacio celular, y discute sus principales comportamientos y características tales como computación universal, y autoreproducción. El espacio celular utilizado en dicho trabajo tiene las siguientes características.

1. Un plano infinito dividido en cuadros pequeños.

2. Cada cuadro contiene una copia del mismo autómata finito (el cuadro y el autómata juntos forman una célula).

3. Asociado con cada célula esta su *vecindad*, que consiste de sus 4 vecinas no diagonales.

4. El estado de una célula al tiempo t+1 está determinada únicamente por su *estado de vecindad* al tiempo t, junto con la función de transición de estados f del autómata asociado a cada célula.

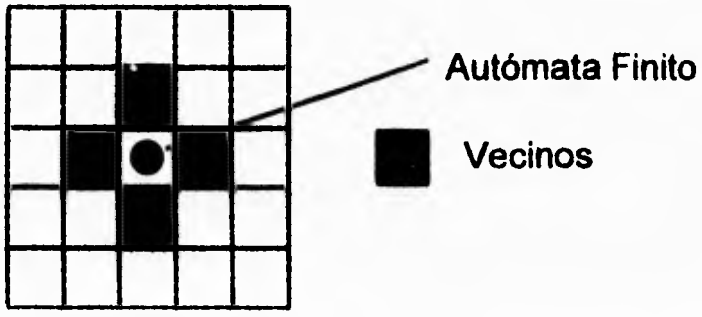
5. El autómata finito asociado a cada célula tiene un estado *Vo* llamado *Estado quieto* tal que:

$$f(V_o, V_o, ..., V_o) = V_o$$

6. A cada instante de tiempo un número finito de células se encuentran en el estado quieto.

7. El número de estados distintos para el autómata finito asociado con cada célula es 29.

8. La función de transición esta diseñada de tal forma que realice ciertos cálculos y propiedades de construcción.



Vecindad en el espacio celular de John von Newmann

Para este espacio celular se especifica un estado inicial por cada célula, dicha configuración inicial (o patrón de estados) junto con las propiedades locales mencionadas anteriormente determinan la configuración de el espacio celular a cada subsecuente estado de tiempo.

Las preguntas que se planteó y que fueron contestadas afirmativamente por el modelo antes descrito fueron:

1. **¿Puede una máquina universal de Turing ser insertada en el espacio?**

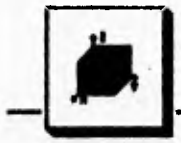
2. **¿Es posible insertar en el espacio a un autómata A con la propiedad que, dando las especificaciones de cualquier autómata construible B, A pueda construír a B y entonces colocar a B libre para trabajar independientemente de A?**

3. **¿Puede un autómata A tener la capacidad de satisfacer los requerimientos anteriores pudiendo construir un autómata igual a si mismo?**

La configuración que John von Neumann encontró, consiste de un espacio celular bidimensional donde los autómatas de cada célula consisten de 29 estados con una vecindad de 5[1].

---

[1] **Es importante mencionar que John von Newmann considera a el autómata en cuestión parte de su propia vecindad, dando como resultado que cada autómata tenga 4 vecinos ortogonales que junto con el mismo forman la vecindad de 5 autómatas.**



A partir del trabajo inicial de J. von Neumann se ha generado una amplia teoría de Autómatas Celulares, tal es el caso del trabajo de Codd [Codd, 1968], el cual mostró la existencia de un autómata celular bidimensional de 8 estados y vecindad 5, el cual tiene computación y construcción universal.

Conforme fue avanzando el estudio de los autómatas celulares, algunos esfuerzos han sido realizados para explotar su paralelismo inherente, siendo aplicados con éxito en varios campos, esto a su vez ha permitido la creación de diversos tipos de autómatas celulares que de acuerdo a sus características pueden ser aplicados a problemas específicos, incluso, dicha teoría ha generado la creación de una nueva arquitectura de computadoras [Hills, 1984].

A.3 - Autómata Celular Determinístico

A continuación daremos una breve explicación de la teoría generada con respecto a los autómatas celulares determinísticos y algunas de sus propiedades más importantes como computación universal, construcción universal y autoreproducción, si bien la revisión no pretende ser exhaustiva se tratará de dar un panorama general de dicha teoría.

Los Autómatas Celulares son un tipo de sistemas dinámicos discretos no lineales construidos de un número grande de autómatas de estado finito idénticos, cada uno recibiendo entradas de otros autómatas dentro de una vecindad predefinida. Cada autómata usa el patrón de entrada y su actual información de estados para actualizar su propio estado así como para determinar la salida que enviará a otro autómata dentro de su vecindad. El autómata esta dispuesto en un patrón celular, de aquí el nombre de autómatas celulares.

Formalmente un *Autómata Celular Determinístico* **A** está especificado por un espacio celular discreto **U**, cuatro conjuntos finitos **X**, **Y**, **Q**, **N**, y cuatro funciones γ, δ, β y Ω, donde:

i. El espacio celular U es usualmente un toroide tal que cada punto en U esta indexado por una N-ada z, donde:

$$z = (\mu 1, \mu 2, \ldots, \mu N) \bmod (n1, n2, \ldots, nN)$$

ii. X es un conjunto finito de símbolos de entrada.

iii. Y es un conjunto de símbolos de salida.

iv. Q es un conjunto de símbolos de estados internos.

v. N es la lista de relaciones de vecindad, la cual determina las posiciones relativas de cada vecino en la red para un autómata de cualquier lugar en U, posiblemente conteniendo el autómata mismo.

vi. γ es la función de configuraciones de estados de vecindad, y es tal que mapea cualquier lugar de U en la red a la vecindad de U de acuerdo a lo especificado en las relaciones de vecindad definidas por N.

vii. δ, es la función de transición de estados, la cual determina el estado siguiente q, de cualquier lugar en la red de acuerdo a la

entrada recibida junto con la configuración de estados de vecindad definidos por N.

viii. β, es la función de salida, tal que la configuración de estados de vecindad γ(u) de cualquier lugar en la red u en U siempre produce una salida en $Y = \beta(\gamma(u))$.

ix. Ω, es la función de salida y transición global, actúa sobre la red completa U, transformando cualquier configuración de estados global S en el instante de tiempo t a una nueva configuración S′ a el siguiente instante de tiempo t+1.

La dinámica de cada célula depende de la función de entrada que comunica a el autómata con su medio ambiente externo. La función de salida representa las medidas tomadas en el sistema, por ejemplo: valor del estado del autómata, etc.

Cada célula de este espacio está en uno de un número finito de estados a un tiempo t. La física de este universo lógico es determinístico y local. *Local* quiere decir que el estado de una célula a el tiempo t+1 es una función únicamente de su propio estado y los estados de sus vecinos inmediatos a el tiempo t. *Determinístico* significa que una vez que una función local y un estado inicial de un autómata celular ha sido escogida, su evolución futura está completamente determinada.

La formulación anterior de autómata celular puede ser generalizada en lo que se conoce como *Autómata Celular Estocástico*, esto es:

Un autómata celular estocástico A sobre un conjunto (U, X, Y, Q, N) está definido por la función de configuración de estados de vecindad γ así como

por dos funciones probabilisticas **F** y **G**, donde **U**, **X**, **Y**, **Q**, **N** y γ retienen su significado respectivo que en el caso determinístico.

Las funciones **F** y **G** como puede observarse reemplazan a las funciones δ y β respectivamente, tal que:

I. **F** es un mapeo estocástico de estado siguiente tal que mapea la entrada actual **X** y la configuración de vecindad **N** en el estado siguiente **Q** con una probabilidad **F(U, N, X)**.

II. **G** es un mapeo estocástico de salida, tal que la configuración de vecindad **n** de cualquier lugar en la red, producirá la salida **y** con una probabilidad **G(n, y)**.

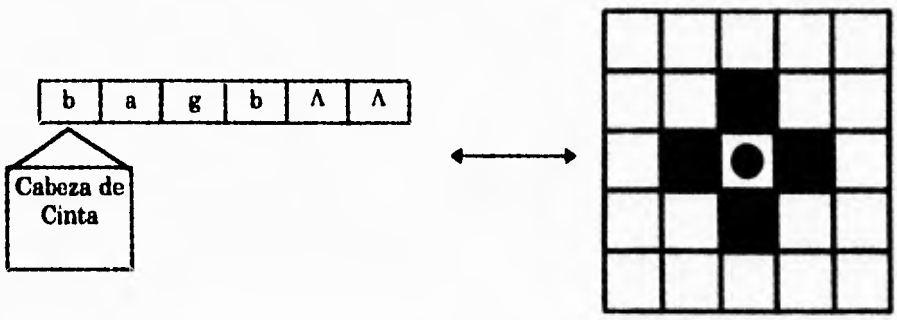
La función Ω correspondiente puede también definirse, obteniéndose directamente de F y G, y es por tanto, no esencial para la definición del autómata celular estocástico.

Entre las características más importantes que los autómatas celulares presentan son computación universal, construcción y autoreproducción, la teoría que se ha generado con respecto a estos tres importantes conceptos se introduce a continuación.

Computación Universal

Es bien conocido en las ciencias de la computación que, si un algoritmo puede ser ejecutado por cualquier máquina (por máquina podemos entender un autómata finito), entonces, dicho algoritmo puede ser realizado por una máquina universal de Turing, por tal motivo decimos que la máquina universal de Turing tiene computación universal.

Para establecer computación universal en un espacio celular, debemos por tanto establecer un vínculo estrecho entre máquinas de Turing de un lado y Autómatas celulares del otro. Para que exista una correcta correspondencia entre ambos conceptos, se deben tomar en cuenta algunas consideraciones, por ejemplo, debemos considerar que una máquina de Turing realiza únicamente manipulaciones lógicas en su cinta, verifica símbolos, los imprime y ejecuta operaciones lógicas elementales, mientras que un autómata celular puede ser capaz de reconocer componentes, moverlos por la malla y unirlos.



Se debe establecer una correspondencia entre máquinas de Turing y Automátas Celulares para establecer la Computación Universal

El concepto de computación universal en un espacio celular puede enunciarse de la siguiente manera:

Un espacio celular A con una configuración S, tiene computación universal si existe un dominio de Turing T y si para cualquier función parcial

calculable por Turing Ψ de T hacia T, existe una configuración S disjunta de T tal que S calcula Ψ.

El dominio de Turing es un conjunto infinito de cintas, las cuales están en correspondencia efectiva uno a uno con el conjunto de enteros no negativos.

En otras palabras, un espacio celular P tiene computación universal si (y en un sentido apropiado) cada función parcial calculable por Turing es calculable en P.

Construcción

Cuando J. von Neumann introdujo la teoría de autómatas celulares uno de los objetivos principales que se planteó se refleja en su pregunta:

2. **¿ Es posible insertar en el espacio a un autómata A ...tal que, A pueda construir a B y ... ?**

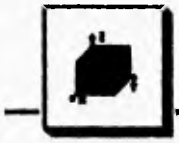
El concepto generado por dicha teoría se formaliza de la siguiente manera:

Un autómata P se dice construye a otro autómata P′ si, iniciando con una configuración finita S donde todas las celdas excepto aquellas de P están en estado quieto, existe un tiempo t, tal que:

P′ es una subconfiguración de $F^t(S)$ y

$F^t(S)$-P′ no pasa información a P′.

donde :



$F^t(S)$ es la función global de transición a un tiempo t.

Un constructor-computador universal (universal computer-constructor) es un autómata celular que puede realizar cualquier cálculo (similar a cualquier máquina de Turing), y el cual dada la descripción de cualquier autómata B, puede construir una copia de B en cualquier región vacía designada de el espacio celular.

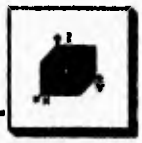
Autoreproducción

El paso inmediato de un autómata celular con construcción y computación universal es el de autoreproducción. La forma natural para definir autoreproducción en espacios celulares es entonces derivándolo como un caso especial de construcción; así, la configuración S es auto reproducible si existe una translación ϕ, tal que S construye S_ϕ.

Teniendo ya los conceptos básicos de la teoría de los autómatas celulares, podemos entonces formalizar de acuerdo a tales conceptos el juego matemático inventado por John Conway, *life*.

***Life* formalmente hablando es un autómata celular bidimensional donde:**

U	**Es un toro bidimensional.**
X	**No existe.**
Y	**es simplemente una configuración 0, 1 del toro.**
Q	**Únicamente consiste de dos estados 0 , 1.**



N Es el conjunto de 8 vecinos rodeando a la celda en cuestión[2].

δ Es la función boleana tal que:

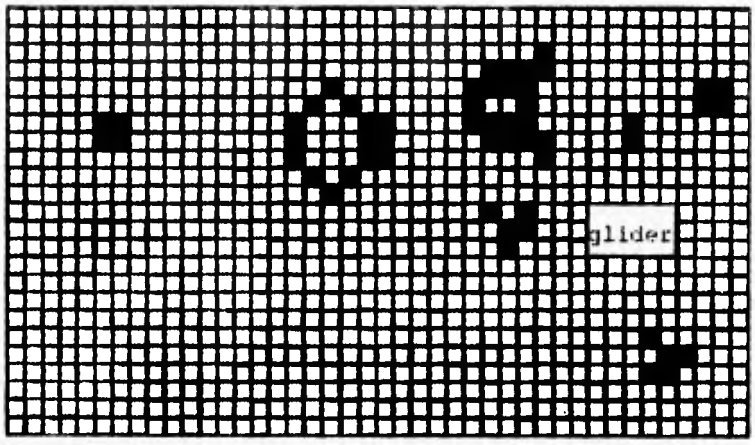
$$X_{ij}\ ((\ X_{ij}\ \text{and}\ \#N = 2\ )\ \text{or}\ \#N = 3\ )$$

Esto significa que el estado de cada celda i,j es puesta en 1 si su valor y el de únicamente 2 celdas entre las ocho vecinas fué 1 en el ciclo anterior o si tenía 3 vecinas con valor a 1.

β Es la función de salida y nos entrega un valor de 0 o 1 a la salida.

Este autómata cumple con las características de computación universal y construcción que se mencionaron anteriormente, un ejemplo de una configuración que construye un *glider* o *Deslizador* cada 30 generaciones se muestra a continuación.

[2] Nótese que en este caso, el organismo en cuestión no forma parte de su vecindad como en el caso de el espacio celular de John von Newmann.



Configuración que produce un glider cada 30 generaciónes, a esta configuración se le conoce como *gun* o *Pistola*

El comportamiento complejo y paralelismo inherente que los autómatas celulares presentan, lo convierten en un poderoso sistema de computo apropiado para problemas complejos de inferencia, de simulación de procesos fisicos, biológicos, etc.

Un tipo de Autómatas Celulares de particular interés para la simulación de dinámicas de fluidos son los conocidos como *C-A Lattice Gas*, que de hecho han generado una área de investigación autónoma, la teoría generada ha sido aplicada no solo a la dinámica de fluidos, sino también a la simulación de ondas de sonido, de procesos de difusión, de efectos de viscosidad, etc.

A.4 - C-A lattice Gas

En años recientes un tipo importante de autómata celular conocido como lattice gases (gases en malla)[3] han sido usados exitosamente para modelar una gran variedad de sistemas físicos, los cuales eran tradicionalmente modelados por ecuaciones diferenciales parciales. C-A Lattice gases deben su origen posiblemente a las investigaciones realizadas por J. Hardy, O. de Pazzi y Y. Pomeau, los cuales estudiaron la dinámica molecular en un espacio bidimensional [Hardy, 1976], o bien el trabajo de Frish, Hasslacher y Pomeau que descubrieron un A-C Lattice gas particular que simula las ecuaciones bidimensionales de Navier-Stokes.

Para realizar una introducción de este tipo especial de autómatas celulares se presentará una breve descripción del trabajo de J. Hardy y O. de Pazzis [Hardy, 1976].

Lattice gases son un tipo especial de autómatas celulares para el cual hay alguna asociación entre los bits de un lugar en la malla (Lattice) y sus vecinos inmediatos. Usualmente, los bits en la malla son tomados como partículas cuyos movimientos están asociados con sus direcciones correspondientes. En adición, puede haber un número de *partículas inmóviles*, esto es, bits que están asociados únicamente con un lugar específico y donde estas residen.

La evolución de un Lattice gas para un instante de tiempo puede entonces dividirse en dos pasos:

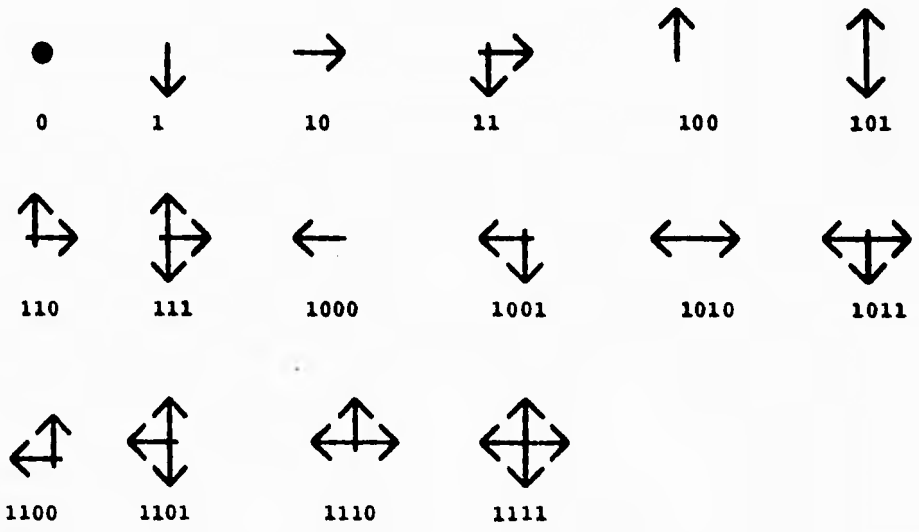
---

[3] Debido a lo impreciso de la traducción al español del término lattice gases, se usará en lo sucesivo el término en inglés.

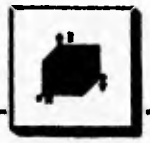
i. **Una fase de transición en el cual el valor de cada bit se mueve a el vecino con el cual está asociado (las partículas inmóviles no se ven afectadas).**

ii. **Una fase de colisión en el cual todos los bits en la malla alteran sus valores.**

Supongamos un espacio bidimensional dividido en pequeños cuadros, en cada cuadro pueden permanecer cuatro partículas como máximo, teniendo cada una diferentes velocidades; existiendo por tanto 16 posibles situaciones para las partículas de cualquier cuadro en la malla.



Posibles situaciones en un vértice



Llamaremos a E_{ij} al conjunto permitido de posiciones y velocidades de las partículas en la malla y que se denota como E_{ij} donde:

$$0 \langle = E_{ij} \rangle 1111$$

donde los índices i,j denotan un lugar en la malla que es la intersección de la i-ésima columna y el j-ésimo renglón.

De acuerdo a lo expresado en los puntos 1 y 2 anteriores podemos definir la fase de transición como:

$$(T_o E)_{ij} = d_{i\,j+1}\ c_{i+1\,j}\ b_{i\,j-1}\ a_{i-1\,j}$$

donde T_o es el operando de translación.

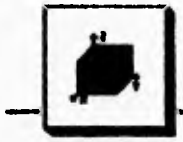
La fase de colisión entonces puede expresarse como:

$$CE_{ij} = \begin{cases} 0101 & \text{si } E_{ij} = 1010 \\ 1010 & \text{si } E_{ij} = 0101 \\ E_{ij} & \text{de otro modo} \end{cases}$$

Donde C es el operador de colisión.

Usualmente los valores del los bits en cada lugar de la malla después de la fase de colisión depende de sus valores anteriores, esto es, la colisión es un mapeo de n bits en la malla hacia n bits en cada cuadro de la misma. Existen, sin embargo, Lattice gases cuya fase de colisión es estocástica.

Una última característica distintiva de los lattice gases es la conservación de ciertas cantidades que son lineales en los valores de los bits, por ejemplo, en algunos modelos, el número total de partículas es una



cantidad conservada, tal es el caso del ejemplo anterior, donde la velocidad en su componente horizontal es conservada sobre cada renglón y su componente vertical es conservada en cada columna.

Lattice gases, como un caso especial de autómatas celulares, pueden ofrecer una alternativa útil para los métodos de punto flotante[4] que resuelven algunas ecuaciones diferenciales parciales de interés en física o matemáticas, principalmente para simulaciones cuya solución por medio de métodos continuos (o de variable continua) es sumamente compleja o para problemas en los cuales la evolución hidrodinámica depende de fluctuaciones microscópicas en condiciones iniciales.

Los conceptos generales así como la dinámica de comportamiento de los Lattice gases y autómatas celulares son similares, más sin embargo, la tendencia es reservar el término lattice gas para el autómata celular en el cual la metáfora de *gas* o movimiento de partículas puede utilizarse.

A.5 - Características Generales de los AC

Las características principales que los autómatas celulares presentan son:

1. Los autómatas celulares son un paradigma general para la computación distribuída ya sea en computadoras vectoriales o paralelas.

---

[4] Llamamos métodos de punto flotante a aquellos modelos que resuelven algunas ecuaciones diferenciales parciales usando variables continuas.

2. Los autómatas celulares son sistemas dinámicos discretos y pueden ser estudiados con técnicas bien conocidas, tratando de identificar propiedades tales como estabilidad, estacionieridad, ergodicidad, etc.

Los autómatas celulares pueden representar modelos originales para fenómenos físicos, los cuales pueden competir con los modelos continuos existentes.

Apéndice B

Programación Orientada a Objetos

FALTA PAGINA

No 126 la 127

B.1 - Antecedentes

El desarrollo de sistemas de software es una industria relativamente nueva, que no ha alcanzado el nivel de madurez que encontramos en las ramas tradicionales de la industria. Si bien el desarrollo de software ha evolucionado a pasos agigantados aún en la mayoría de los casos, las técnicas de desarrollo hacen énfasis en el proceso creativo.

La evolución acelerada del hardware, la demanda creciente de nuevos programas aunada a nuestra incapacidad por dar mantenimiento a los ya existentes han obligado al estudio de nuevos métodos y tecnologías que permitan disminuir los efectos en contra que los factores arriba mencionados producen en la industria del desarrollo de software.

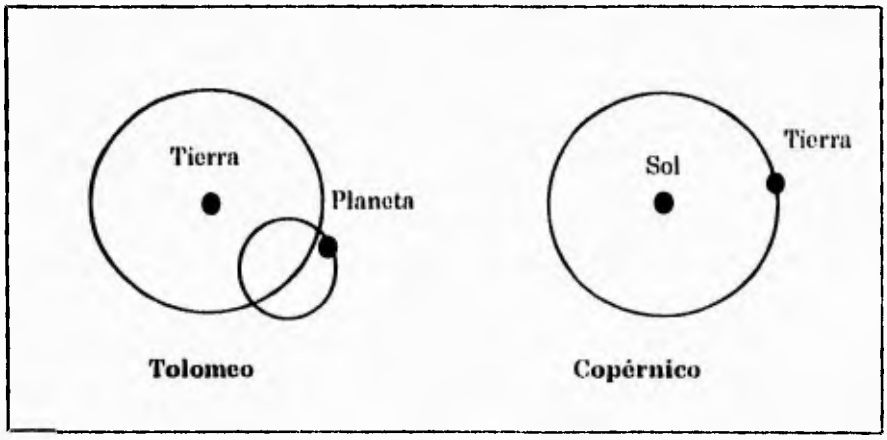
La Ingeniería de Software ha sido una de las disciplinas cuya misión principal es proveer de métodos y procedimientos perfectamente definidos para cada una de las fases del desarrollo de software.

B.2 - Programación Orientada a Objetos, el nuevo paradigma

El diccionario define paradigma como "ejemplo, modelo, ejemplar", si bien esta definición es etimológicamente correcta, en el contexto en el cual nos encontramos no es del todo acertada, sin embargo, si queremos trazar las raíces etimológicas del término en nuestro contexto, nos conviene considerar la obra de Thomas Kuhn *La Estructura de la Revolución Científica* [Kuhn, 1970], en esta obra Kuhn analiza las revoluciones científicas como lo que el llama **Paradigm Shifts**.

Para Kuhn, un cambio de paradigma "altera la conceptualización de entidades" con que trabaja el profesionista en el campo y "en el proceso, desplaza la red de teoría por medio de la cual se trata con el mundo".

En otras palabras un paradigma es el filtro que permite o no el paso de información que va acorde a la conceptualización que se tenga de un fenómeno en cuestión y el desplazamiento o cambio de paradigma es un cambio en la forma de conceptuar dicho fenómeno, por ejemplo, consideremos el caso de Nicolai Copérnico, donde su conceptualización del universo lo lleva a proponer el hecho de que los planetas giran alrededor del sol y no alrededor de la tierra como centro del universo tal y como la astronomía Toloméica lo proponía.



Las órbitas de los planetas bajo paradigmas distintos

El cambio de mentalidad que los desplazamientos de paradigmas producen es enorme y en la mayoría de los casos, se requiere de mucho tiempo para que el nuevo paradigma sea aceptado. Los cambios de

paradigmas no son exclusivos de las áreas científicas, estos son aplicables a todas las actividades en las que interviene el hombre.

La resolución de problemas mediante la computadora no difiere de otras actividades científicas. Los conceptos y actividades realizadas en y durante el proceso de desarrollo de software así como la forma en que se conceptualizan los problemas y sus soluciones, todos son propiciados (y de igual manera, limitados) por el paradigma vigente.

En años recientes un nuevo paradigma ha revolucionado el desarrollo de software, el impacto que este paradigma ha tenido en la ingeniería de sistemas computacionales solo puede ser equiparado con el impacto que tuvo la programación estructurada en los años 70's.

Durante las últimas 2 décadas el esquema dominante para lo programación en computadoras ha sido la programación estructurada. Dentro del paradigma que engloba la programación estructurada se visualiza la solución de un problema en términos de una jerarquía de procedimientos para la manipulación de los datos, en este esquema los datos juegan un papel subordinado al de los procedimientos. De este modo, la organización de un sistema esta inicialmente pensada, después implantada y finalmente descrita en términos de procedimientos.

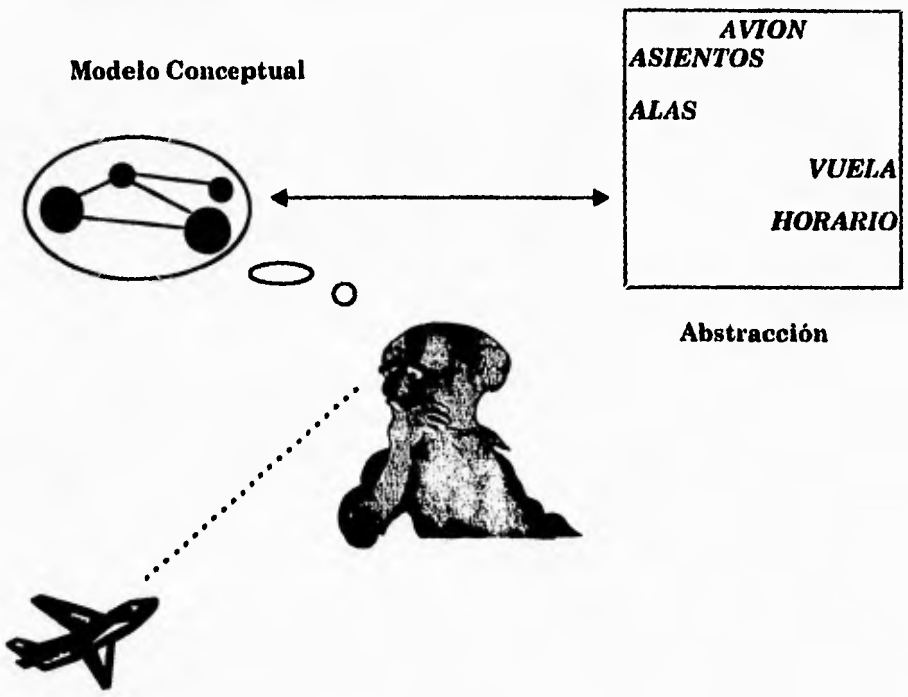
Hoy el nuevo paradigma se llama **Programación Orientada a Objetos (POO)**, el objeto desplaza al procedimiento, y tal como el centro del sistema solar, la organización de un sistema gira ahora alrededor de los datos. Las entidades que manipula el programador en el proceso de diseñar, implementar y documentar sus sistemas son los objetos.

B.3 - ¿Porqué Programación Orientada a Objetos?

La programación Orientada a Objetos induce un cambio en la cultura computacional, que va mucho más allá de la incorporación de nuevos mecanismos de abstracción en los lenguajes de programación. El verdadero impacto de la POO está en el cambio de perspectiva del desarrollo de sistemas, que pasa de ser una actividad basada en una boutique personalizada de subrutinas, a ser una industria de componentes reutilizables.

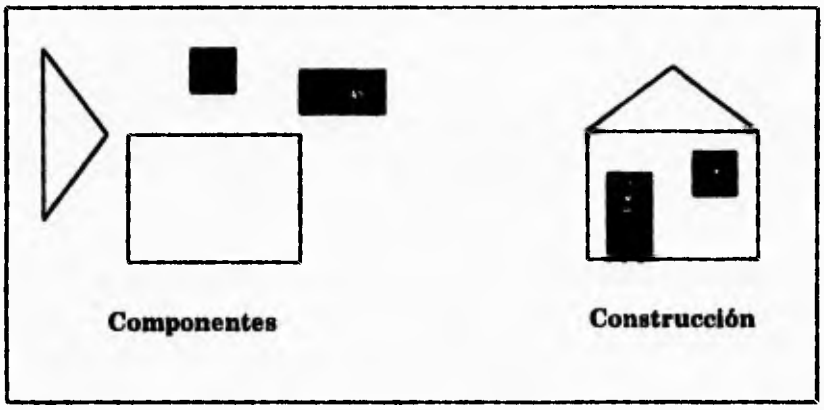
Un componente es un modulo cuya interfaz con el exterior está claramente especificada. Un componente es la abstracción representada en algún lenguaje de programación que se realiza de un objeto del mundo real y cuya descripción enfatiza **QUE** es lo que hace dicho componente ocultando todos los detalles de implementación que son irrelevantes para el entendimiento de la abstracción.

Dicha abstracción es fundamental en la POO ya que permite identificar las características estructurales y de comportamiento del objeto del mundo real pudiéndose asociar dicha abstracción con un componente computacional que refleja directamente las características de la abstracción.



La POO abstrae las características principales de los objetos del mundo real

En esta nueva perspectiva, el desarrollo de sistemas es una actividad que consiste en generar componentes que puedan ser reutilizados. Un sistema se construye en base a componentes previamente diseñados e implementados.

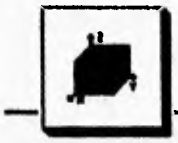


Uso de Componentes para la construcción de Sistemas

El objetivo principal es, a largo plazo, amortizar el costo de desarrollo de sistemas mediante la reutilización de componentes. Esta perspectiva cambia, también, el ciclo de desarrollo de sistemas, así como las metodologías de análisis y diseño, la administración de sistemas y, en general, la cultura de desarrollo de sistemas.

El desarrollo de sistemas con la metodología orientada a objetos tiene un doble propósito: por un lado la elaboración de herramientas que solucionen la problemática planteada y, por otro, la generación de componentes que puedan ser integrados a bibliotecas de objetos, aumentando así el capital de software.

La adopción de una metodología de desarrollo orientada a objetos contribuye esencialmente en el soporte a largo plazo, documentación y reutilización, teniendo además un impacto económico que sólo es observable después de un lapso considerable de tiempo de haber adoptado tal metodología. [Sagols, 1994].



B.4 - Programación Orientada a Objetos

Como se mencionó anteriormente, en el paradigma orientado a objetos, la idea central es efectuar una abstracción del mundo real en términos de objetos, más que de las operaciones involucradas en el problema para crear un modelo en que se identifican y organizan conceptos importantes relacionados con la solución del problema a resolver.

La gente tiene una apreciación de su ambiente en términos de objetos. Por lo tanto es simple pensar de la misma manera cuando se diseña un modelo - y más aun si ese objeto puede representarse como tal en un ambiente de programación - existiendo sólo una pequeña distancia semántica entre la realidad y el modelo representado en la computadora.

A continuación se explicaran los conceptos básicos, haciendo referencia a un ejemplo cotidiano, no trataremos de enfocar el ejemplo a ningún lenguaje de programación ni a ningún método de desarrollo en especifico[1].

La realidad que describiremos involucra el desarrollo de un sistema gráfico. Nuestra tarea es intentar modelar este sistema. Veremos también que la construcción del mismo se realiza de manera muy natural.

Objetos

El primer y más importante concepto que describimos es por supuesto, el concepto de **objeto**. De manera informal, un **objeto** es una entidad capaz de almacenar un estado (información) y que ofrece varias operaciones (comportamiento) que nos permiten examinar o afectar su estado.

---

[1] Algunos métodos de desarrollo se verificarán más adelante.

Generalmente el desarrollo de un sistema orientado a objetos consiste de varios objetos; Los objetos corresponden usualmente a entidades de la vida real, tales como personas, un automóvil, o una flor. Cada objeto tiene su información individual (por ejemplo cada automóvil tiene su número de registro).

Para nuestro ejemplo necesitamos un sistema gráfico que despliegue figuras geométricas tales como círculos, triángulos, rectángulos, líneas, etc. Como ejemplo de la información que posiblemente queremos tener para cada objeto están color, estilo, etc. Todas estas figuras tienen información y comportamiento diferente. Para tener acceso a esta información o ser capaces de afectarla, debemos definir para cada objeto un conjunto de operaciones de configuración que pueden afectar o leer la información almacenada. Podemos definir también operaciones que quizá no afecten la información interna, pero que den lugar a un comportamiento en especifico (por ejemplo, moverse, dibujarse).

La única parte del objeto que podemos ver son sus operaciones, el interior está oculto para nosotros, esta es una característica importante dentro de la POO llamada **encapsulamiento**, la cual discutiremos más adelante.

Línea

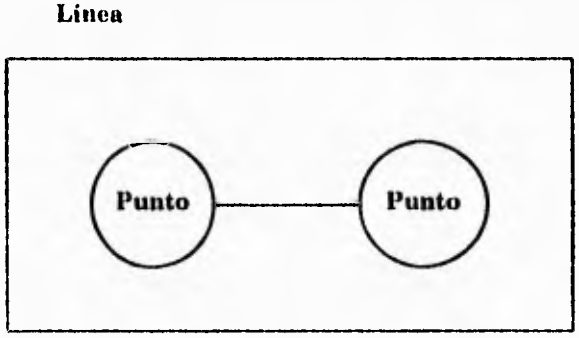
Operaciónes
Moverse
Dibujarse
Rotarse
¿Color?

Las operaciones únicamente son visibles para el objeto Línea

Dentro de la información de un objeto se especifican las asociaciones con otros objetos, por ejemplo todas las figuras saben la Posición en

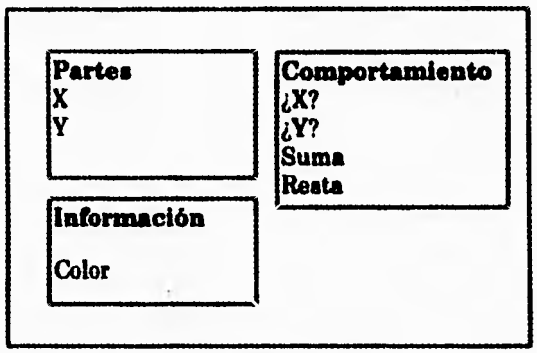
pantalla donde se encuentran. Esto significa que los objetos tienen a su vez relaciones con otros objetos, estas relaciones pueden ser de dos tipos, una de ellas se conoce como **relaciones estáticas**, estas existen durante periodos prolongados de tiempo, lo que significa que dos objetos que tienen una relación estática entre sí saben de su mutua existencia. También tenemos las **relaciones dinámicas**, que son relaciones por medio de las cuales los objetos se comunican entre sí.

Debido a esto, los objetos pueden estar compuestos por otros objetos que le auxilian en la consecución de su objetivo, por ejemplo, la clase línea puede estar compuesta de 2 **Puntos** los cuales indiquen su posición inicial y final de la pantalla[2].



El objeto Línea puede descomponerse en dos objetos Punto

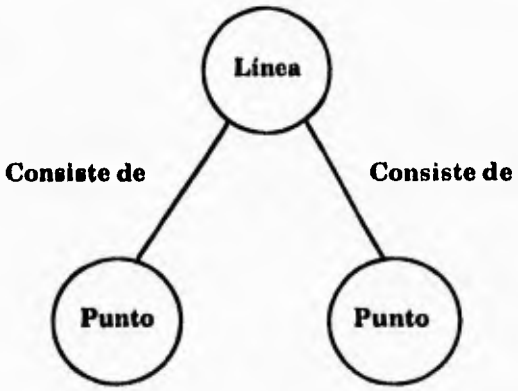
[2] En este caso **Punto** es un objeto cuya utilidad principal es conocer las coordenadas de un punto perteneciente a la pantalla de vídeo.



Interior del objeto Punto

Mediante su composición podemos estructurar al objeto línea en partes. La razón de ser de la estructura puede depender de muchos factores. Frecuentemente esta depende de la combinación que deseamos detallar en el proyecto, el nivel de conocimiento a alcanzar y el deseo de obtener partes reutilizables.

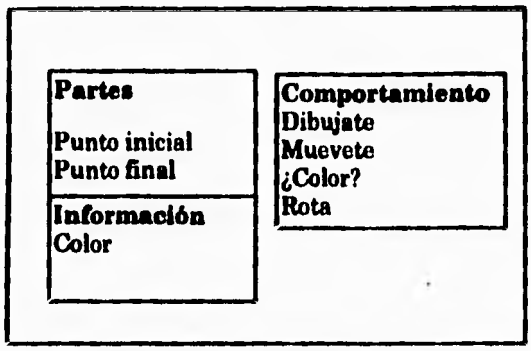
Existen así mismo otro método similar que nos permite juntar partes diferentes de un objeto llamado jerarquías de partición. En este método un objeto se construye a partir de otros objetos, donde las relaciones son llamadas frecuentemente relaciones consiste de.



Jerarquía de Partición para el objeto línea

Encapsulamiento

Si miramos al interior del objeto, veremos tanto su estructura de información como el funcionamiento de sus operaciones, podemos ver también la información que el objeto necesita almacenar, las partes de que consiste el objeto y como se define el comportamiento de sus operaciones.



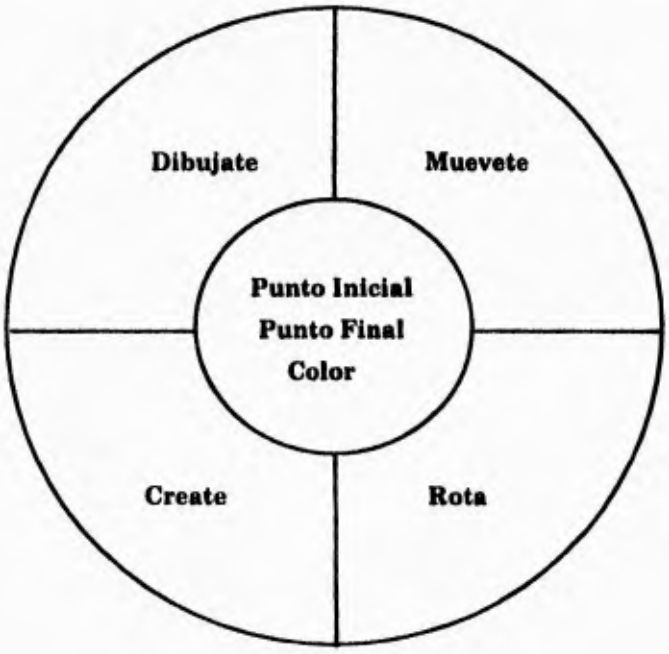
Interior del objeto Línea

Como se mencionaba anteriormente, el **encapsulamiento** es una de las características importantes dentro de la POO, este "Es el mecanismo que permite ocultar los detalles de la representación interna de un componente, presentando la interfaz disponible para el usuario. La interfaz de un componente cumple con una doble tarea; por un lado especifica los servicios que ofrece un objeto y los requerimientos para su correcta ejecución y, por otro, oculta todo aquello que no se encuentre explícitamente en la interfaz" [Quintanilla, 1994].

El **encapsulamiento** tiene varias implicaciones en la construcción de sistemas de software, "su contribución es restringir los efectos del cambio[3] colocando un muralla de código alrededor de cada pieza de datos. Todo

[3] Con cambio nos referimos a las posibles modificaciones que un sistema tenga en sus requerimientos originales.

acceso a los datos es manejado por procedimientos colocados ahí para mediar el acceso a los datos" [Cox, 1987].



La encapsulación permite acceso a los datos del objeto solo mediante los procedimientos definidos en el.

La dinámica en un modelo orientado a objetos es creada mediante el envío de **estímulos**. Con el concepto estímulo nos referimos al evento que se

presenta cuando un objeto se comunica con otro objeto[4]. Esto indica que se estimula algún comportamiento para que tome lugar, el cual al ser recibido por un objeto causa que se efectúe una operación en el objeto receptor. De esta manera si deseamos que un objeto ejecute un comportamiento enviamos un estimulo a ese objeto.

Consideremos el caso de nuestro objeto línea, el cual queremos que sufra una rotación, entonces enviamos el estímulo *Rota*. Cuando la línea recibe este estímulo, lo interpreta y ejecuta lo que ha sido definido para esta cuando recibe este estímulo. La línea entonces estimula sus puntos inicial y final para que se muevan a una posición de pantalla especifica.

Clases e Instancias

En el sistema que estamos construyendo deseamos que existan varios objetos del mismo tipo en pantalla, por ejemplo, desplegar líneas de diferentes tamaños y colores o bien en diferentes posiciones con ángulos de rotación distintos.

Todos estos objetos tienen comportamientos y estructuras de información similares aunque su apariencia sea diferente. Estos objetos se dice que tienen el mismo molde o plantilla o bien que pertenecen a una clase específica.

---

[4] En el contexto de programación orientada a objetos, algunos autores lo llaman también mensajes.

> ***Una clase** representa una plantilla para varios objetos y describe la forma como estos objetos se estructuran internamente. Los objetos de la misma clase tienen la misma definición tanto para sus operaciones como para su estructura de información.*

Mediante el concepto de clase, podemos asociar ciertas características con un grupo completo de objetos. Podemos considerar la clase como una abstracción que *describe* todas las características comunes de los objetos que forman parte de la clase.

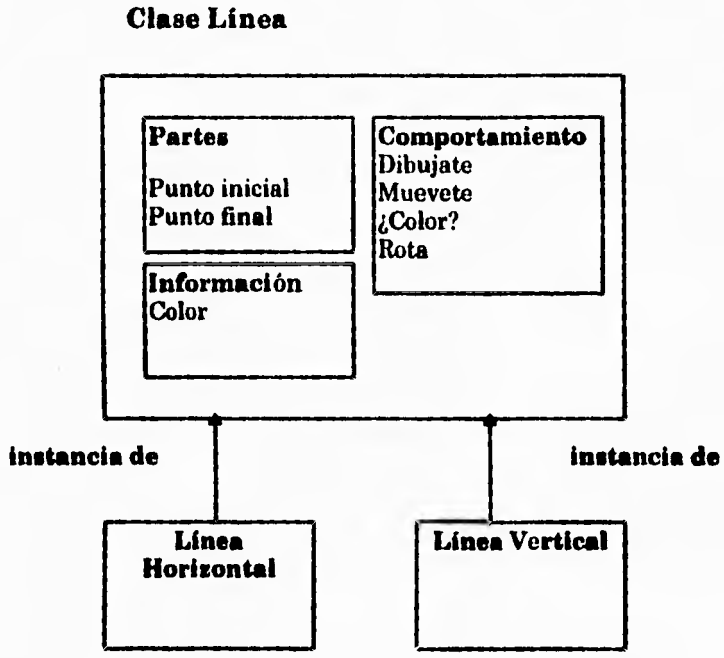
En los sistemas orientados a objetos, a cada objeto pertenece una clase. Un objeto que pertenece a cierta clase es llamado una instancia de esa clase. Por lo tanto usamos frecuentemente los conceptos de instancia y objeto como sinónimos.

> ***Una instancia** es un objeto creado a partir de una clase. La clase describe la estructura (de comportamiento e información) de la instancia, mientras que el estado actual de la instancia está definido por las operaciones ejecutadas sobre la instancia.*

El comportamiento de la instancia y su estructura de información es así definido por su clase. Cada instancia tiene también una identidad única. Varias instancias diferentes pueden ser creadas de una cierta clase, donde cada instancia es manipulada por las operaciones definidas en la clase. Diferentes instancias pueden ser manipuladas por las diferentes

secuencias de operaciones y, como resultado, tener diferentes estados internos. Ahora no necesitamos describir cómo debe aparecer cada operación, pues esta descripción se hace en un único lugar, es decir, en la clase.

Resumiendo, "Una clase es una colección de objetos que tienen una estructura de estado y un comportamiento común. La agrupación de objetos en clases es una forma de ordenar y organizar el dominio de objetos que conforman un sistema. A un objeto, el cual pertenece a una clase, se le denomina también instancia de esa clase. La clase se interpreta también como una abstracción del concepto de objeto" [Oktaba, 1993].

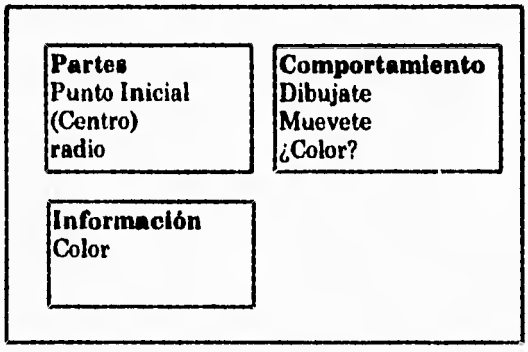


Línea Horizontal y Línea Vertical son instancias de la clase Línea

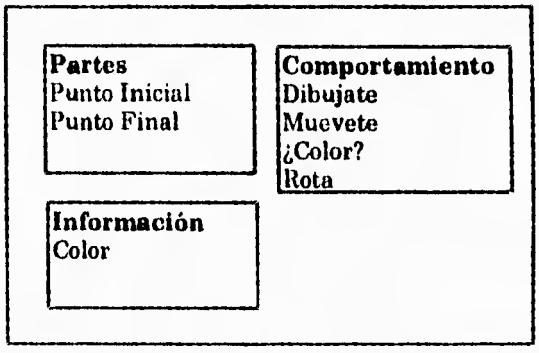
Herencia

Veamos la estructura de información de 2 objetos más que usaremos en nuestro sistema gráfico. Estos dos serán el *Rectángulo* y un *Círculo*, note que el objeto punto forma parte de la estructura de información de los nuevos objetos.

Clase Círculo



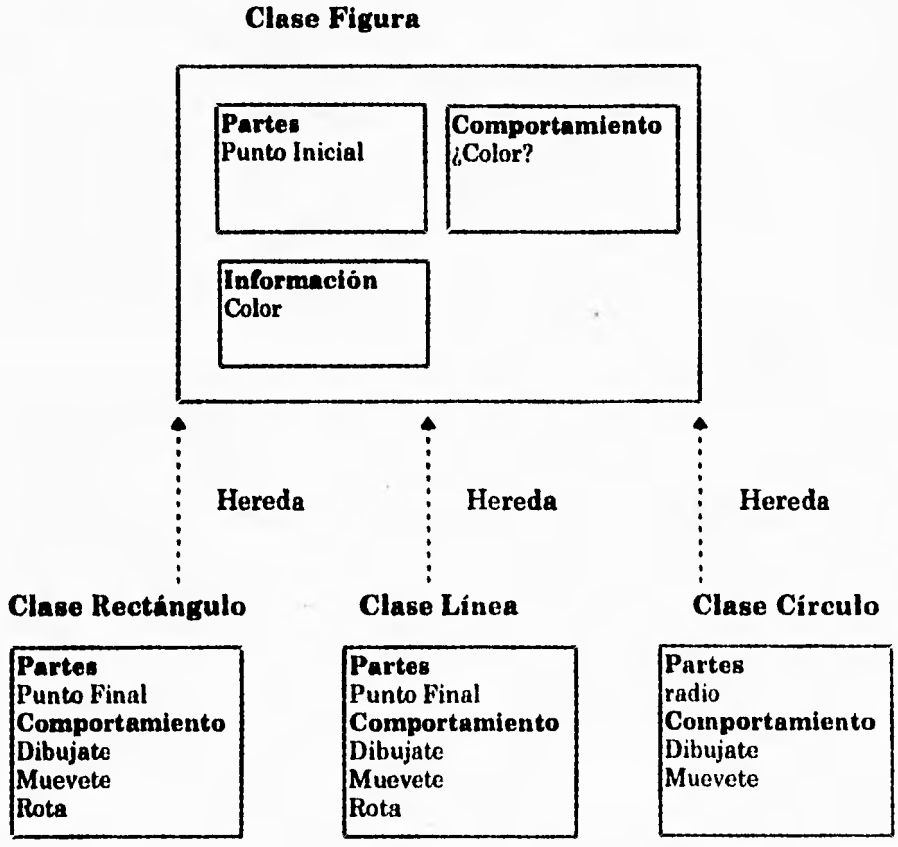
Clase Rectángulo



Estructura de Información de la clase Círculo y Rectángulo

Cuando describimos nuestras clases, rápidamente notamos que muchas clases tienen características comunes (Comportamiento y estructura de información). Por ejemplo, las tres clases de nuestro ejemplo gráfico tienen muchas similitudes entre si. Estas similitudes pueden ser compartidas entre las clases extrayéndolas y colocándolas en una clase separada **Figura**. En Figura, describimos cualquier cosa que sea común tanto para las líneas, Círculos y Rectángulos. De esta manera se pueden compartir características comunes entre varias clases. Colectamos las características comunes en una clase específica y permitimos que las clases originales **hereden** esta clase. Círculos, Líneas y Rectángulos pueden heredar las características de Figura y ahora sólo necesitamos describir las características que son nuevas a estas dos clases.

La herencia "es la parte más innovativa porque no es proporcionada por ningún método de programación. Esta es una herramienta para extender automáticamente código, a clases diseñadas por diferentes miembros de un equipo. Programadores ya no iniciarán cada módulo con una página en blanco, el lugar de esto, escribirán una simple sentencia que referencíe a alguna clase que ya se encuentra disponible en la librería de objetos. Cada sentencia subsecuente describe como la nueva clase difiere de la obtenida de la librería" [Cox, 1987].



Clases con herencia

Por medio de la herencia podemos por tanto mostrar similitudes entre clases y describir esas similitudes en una clase, que otras clases pueden heredar; entonces podemos reutilizar las descripciones comunes.

El uso de la herencia tiene otra ventaja importante. Si queremos modificar algunas características en Figura, (por ejemplo, añadir un color de relleno a la figura), es suficiente efectuar esta modificación en un lugar, Así, se hace una modificación de este tipo, todas las clases descendientes heredarán esta nueva característica de color. A la clase heredera normalmente se le llama *descendiente* o *subclase* y a la clase de la cual se heredan sus características *ancestro* o *superclase*.

Cuando se describe a una clase, si deseamos usar características de dos o más clases existentes, podemos hacerlo. Llamamos a esto **Herencia Múltiple**. Esto significa que una clase puede tener más de un ancestro directo. La herencia múltiple es sin embargo controversial en la comunidad de la POO.

La herencia múltiple es con frecuencia justificada como una manera de brindar descripciones reutilizables. Si una descripción de una clase incluye un comportamiento que existe parcialmente en otras dos clases, entonces este comportamiento puede ser extraído de modo que podemos heredar el comportamiento común de dos clases.

En resumen "La relación de herencia impone una estructura jerárquica al conjunto de clases de un sistema. Bajo la relación de herencia, una clase puede heredar propiedades de otra clase. A la clase que hereda se le conoce como *superclase* y a la clase heredera se le denomina *subclase*. Una subclase puede añadir y/o modificar operaciones a su comportamiento, formando así una especialización de la superclase. Una clase puede ser subclase de una o más superclases. En el primer caso se habla de herencia sencilla y en el segundo de herencia múltiple" [Oktaba, 1993].

Polimorfismo

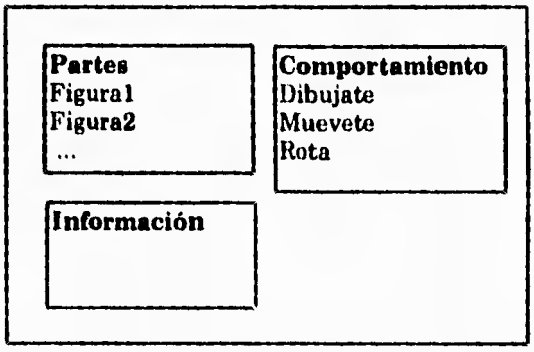
Las instancias, creadas a partir de clases, conjuntamente nos dan el comportamiento dinámico que deseemos modelar. El comportamiento del sistema ocurre cuando estas instancias empiezan a comunicarse entre sí. Una instancia puede saber sobre otras instancias a las que puede enviar

estímulos. Si una instancia envía un estímulo a otra instancia, pero no está consciente de cuál es la clase a la que pertenece la instancia receptora, decimos que tenemos **Polimorfismo**. Polimorfismo significa, al menos en el contexto orientado a objetos, que la instancia emisora no necesita saber la clase de la instancia receptora y esta clase puede ser cualquiera.

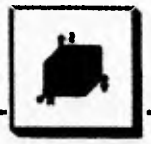
> ***Polimorfismo** significa que el emisor de un estímulo no necesita conocer la clase de la instancia receptora. La instancia receptora puede pertenecer a una clase arbitraria.*

Supongamos que, en nuestro ejemplo del sistema gráfico deseemos un objeto el cual consista de diversas figuras tales como líneas, rectángulos, círculos, etc. La estructura de información de dicho objeto al cual llamaremos **Collage** puede ser como el mostrado en la figura.

Clase Collage



Estructura de Información de la clase Collage



Como podemos observar Figura1 y Figura2 pueden ser instancias de cualquiera de las clases definidas anteriormente (Circulo, Línea o Rectángulo) y por lo tanto la clase Collage no sabrá con cual clase será asociada Figura1, Figura2, etc. Aquí la referencia debe ser polimórfica. De esta manera el polimorfismo significa no sólo que se pueden asociar diferentes instancias sino también que estas instancias pueden pertenecer a clases diferentes.

Un estímulo puede ser interpretado de manera distinta, dependiendo de la clase receptora. Es por tanto, la instancia que recibe el estímulo la que determina su interpretación, y no la instancia emisora. El transmisor de un estímulo necesita saber únicamente que otra instancia puede ejecutar cierto comportamiento, más no necesita conocer cuál es la clase a la que la instancia pertenece y por ende no requiere saber cuál es la operación encargada de realizar el comportamiento. Esta es una herramienta extremadamente fuerte que permite el desarrollo de sistemas flexibles. Por lo tanto, sólo tenemos especificado lo *qué* deberá ocurrir y no *como* deberá ocurrir. Mediante la delegación de lo que deberá ocurrir se obtiene un sistema resistente y flexible. Si deseamos agregar un objeto a una clase nueva, esta modificación afectará únicamente al nuevo objeto, y no aquellos que le envían estímulos.

Esquema Final

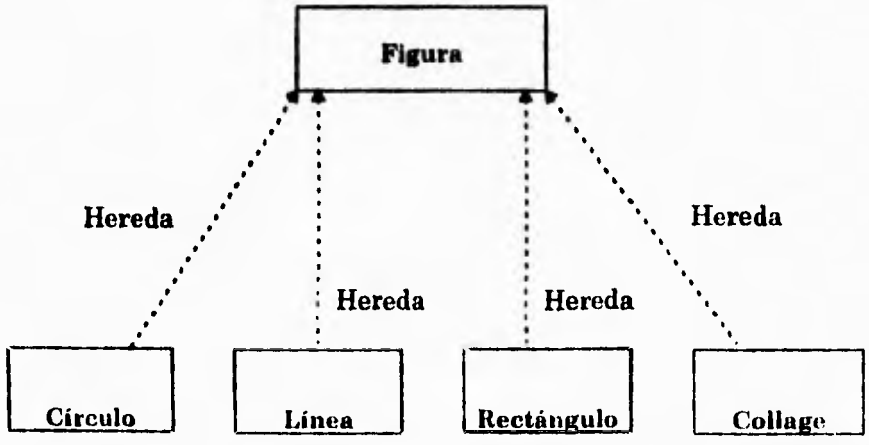
Hasta este momento tenemos 3 clases descendientes de una clase general llamada Figura, y una clase llamada Collage cuya principal característica es que puede contener instancias de las clases Círculo, Línea y Rectángulo pudiendo Figura1 o Figura2 ser una instancia de cualquiera de las clases anteriores, si esto es posible, entonces ¿De qué instancias de clase son Figura1 y Figura2?.

La respuesta es hasta cierto punto sencilla y predecible, Figura1 y Figura2 son instancias de la clase Figura, la cual es el ancestro (o superclase) de las clases Círculo, Línea y Rectángulo, esto significa que una instancia de la clase Figura puede, a su vez, ser instancia de

cualquiera de sus descendientes. Esta característica polimórfica, como se mencionó anteriormente, es una de las herramientas más poderosas de la POO, la cual da como consecuencia lo que en los lenguajes de programación orientados a objetos se le conoce como **Vinculación Dinámica.**

La Vinculación Dinámica, en resumen, permite que un estímulo sea asignado a una operación en la clase de la instancia receptora sólo hasta que el estímulo sea enviado.

Debido a que la clase collage tiene una estructura de información y un comportamiento similar a la clase Figura, podemos hacer a Collage descendiente de la clase Figura. Esta jerarquización permite a Collage además de tener instancias de las clases Círculo, Línea y Rectángulo, tener instancias de ella misma.



Jerarquía final de clases

Con la estructura de información y comportamiento de los objetos modelados, solo corresponde implementarlos en un lenguaje de programación orientado a objetos lo cual no será realizado, ya que el

propósito principal de este apéndice es la introducción de los conceptos de orientación a objetos.

B.5 Erase una vez... Historia de la POO

La programación orientada a objetos, es un paradigma de programación que se encuentra íntimamente ligado a la evolución de los lenguajes de programación, esto nos permite revisar la historia de la POO basándonos en la evolución de los lenguajes de programación. La historia de los lenguajes de programación es relativamente corta (a la par con la historia de la computación) teniendo un desarrollo sumamente acelerado en su corta vida.

Los lenguajes de programación han evolucionado aceleradamente en la relativa breve historia de la programación. El lenguaje **Algol-60**, dado a conocer al inicio de los años 60's, dio la pauta al desarrollo de la metodología de la programación estructurada. El precursor de los lenguajes Orientados a Objetos (OO), **SIMULA-67**, nació en Noruega, donde un grupo de investigadores, encabezados por Dahl y Nygaard, buscaba modificar a los lenguajes ya existentes, diseñados para aplicaciones numéricas, con el objetivo de hacerlos aptos para programar las simulaciones discretas de los problemas del mundo real. Los noruegos extendieron a **Algol-60** con los conceptos de objetos, clases de objetos y jerarquías de herencias entre clases, que caracterizan hoy en día a la POO.

Varios años después, nació **SMALLTALK-80**, este lenguaje de programación es el resultado de una década de investigación del grupo de conceptos de software en Xerox PARC. **SMALLTALK** ha influenciado a una cantidad grande de sistemas, tales como los sistemas Macintosh y una gama amplia de sistemas de información.

Desde el inicio los diseñadores de **SMALLTALK** enfatizaron un número de ideas innovativas y controversiales acerca de lo que la computación podría ser como:

1. **Computación Personal**: Las computadoras son baratas la gente es costosa. La computadoras no deben ser compartidas. Dando a cada persona su propia computadora se podría incrementar el poder de computo y permitir una mayor eficiencia hombre-máquina.

2. **Interacción**: El ambiente de la computadora debe realizar todas las acciones posibles de forma tal que provea al usuario de la suficiente retroalimentación por cada acción que el realice en la computadora.

3. **Gráficos**: La gente esta más adaptada a un pensamiento visual. El texto debe ser reemplazado por imágenes gráficas tan pronto como sea posible.

4. **Programación Orientada a Objetos**: La gente trabaja en conceptos que se encuentran dentro del dominio del problema, mientras el hardware trabaja con diferentes conceptos (operadores/operandos).

Los diseñadores de **SMALLTALK-80** fueron inspirados por la visión futurista de Alan Kay, la *Dynabook*. Esta es una poderosa computadora personal, tan avanzada como la más poderosa estación de trabajo gráfica disponible pero del tamaño de una pequeña libreta. Así entonces nació

SMALLTALK-80 el cual fue diseñado para ser el lenguaje de programación este tipo de computadoras.

Debido a las características de SMALLTALK-80, éste comenzó a ganar popularidad imponiendo una nueva terminología, apareciendo el término OO comenzando el *boom* de la POO. Es importante notar que los conceptos de la OO no fueron aplicados sino hasta después de una década del nacimiento de Simula-67, esto quizá se debió a la fuerte propaganda que se hizo de SMALLTALK-80, siendo también que por esa época (finales de 70's) hubo un desarrollo importante del software base (compiladores, sistemas operativos, etc.) y un fenómeno importante en la industria del software que hoy se conoce como la *crisis del software* [Pressman, 1992].

La segunda mitad de los 80's y los 90's se han distinguido por una gran cantidad de propuestas de nuevos lenguajes orientados a objetos. Entre los más importantes se encuentra la extensión del lenguaje C, conocida como C++ de Stroustrup, el más apegado a la herencia de SIMULA, EIFFEL de Meyer; y una extensión OO de LISP, conocida como CLOS. También cabe resaltar las modificaciones a Pascal como OBJECT Pascal o recientes versiones de TURBO Pascal.

"La POO es la esperanza de la ingeniería de software. Su método de diseño de sistemas requiere de un enfoque totalmente distinto de lo tradicional, esto requiere de un esfuerzo notable de actualización y capacitación por parte del personal académico, los profesionistas y los estudiantes. Pero es un reto que aseguramos vale la pena intentar" [Quintanilla, 1994].

Bibliografía

FALTA PAGINA

No 154 la 155

- Boghosian, Bruce.: *Lattice Gases*, 1990 - Lectures in Complex Systems SFI Studies in the sciences of complexity, Lect. Vol II, Cambridge MA. Addison Wesley, pp 293-323.

- Carruthers, D. J.; J.C.R. Hunt; et. al.: *AirFlow and dispersion over complex terrain*, Air Pollution Modeling and its Application VII Volume 13 (NATO Committee on the challenges of nodern society), New York USA, Plenum Press, 1991, pp 515-518.

- Chambers, Leslie A.: *Classification and Extent of Air Pollution Problems*, Air Pollution Volume I, New York USA, Academic Press, Third Edition, 1968, pp 3-21.

- Clarke, L. George.: *Ecología*, Barcelona, Omega, 1980.

- Codd, E. F., *Cellular Automata*, Academic Press, 1968.

- Cox, B. J.: *Object Oriented Programming - An evolutionary approach*, USA, Addison-Wesley, 1987.

- Eubank, S. and D. Farmer.: *An Introduction to chaos and Randomnes*, Lectures in complex systems 1989, USA, Addison-Wesley, 1989, 75-112.

- Fogelman, Françoise and Robert Yves.: *Automata networks in computer science Theory and Applications*, Great Britain, Manchester University Press, 1987.

- Forrest, S.: *Emergent Computation*, MIT Press Edition, 1991.

- Frish, U.; B Hassler and Y. Pomeau.: *A Lattice gas automaton for the Navier-Stokes equation*, Phys. Rev. Lett, 1986, 1505.

- Gardner, M.:*The fantastic combination of John Conwy's New Solitary game "life"*.Scientific American 223, April 1970, pp 120-123.

- González, Jorge S.; Patricia Sandoval J.: *Nueve reglas de oro en el diseño de interfaces humano-computadora*, México, Soluciones Avanzadas, Junio 1994, pp 10-14.

- Gorlen, E. K.; Sanford; et. al.: *Data Abstraction and Object Oriented Programming in C++*, Great Britain, John Wiley & Sons, 1991.

- Greiff, Warren R.: *El ocaso del sol a través de una ventana*, México, Soluciones Avanzadas, Septiembre-Octubre 1993, pp 35-40.

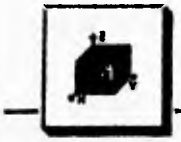
- Guariso, Giorgio y Vittorio Maniezzo. *Air quality simulation through cellular automata*, Environmental Software 7, 1991, pp 131-141.

- Gutowitz, H. A.:*Maps of recent cellular automata and lattice gas automata literature*, PHYSICA D 45, 1990, pp 477-479.

- Hardy, J.; O. de Pazzis and Y. Pomeau: *Molecular Dynamics of a clasical lattice gas: Transport properties and time correlation function*, PHYSICAL REVIEW A 13, 5, 1976, pp 1949-1961.

- Hills, D.: *The Connection Machine: A computer architecture based on cellular automata*, PHYSICA 10 D, 1984, pp 213-228.

- Hopfield, J. J.: *Neural Networks and Physical systems with emergent collective computational abilities*, Proc. Nat. Acad Sci., 1982, 2554-2612.



- Icaza, Miguel de: *Compiladores de C/C++ para DOS/Windows*, México, Soluciones Avanzadas, Noviembre 1994, pp 46-49.

- Kirkpatrick, S.: *Models of disordered materials*, III Condensed Matter, Les Houches, North Holland, 1979.

- Kirkpatrick, S., C. D. Gelatt and M. P. Vecchi.: *Optimization by simulated annealing*, Science 220, 1983, pp 671-750.

- Kuhn, Thomas S.: *The structure of scientific revolution*, Chicago USA, University of Chicago Press, 1970.

- López, Amparo; Guadalupe Ibargüengoitia: *Análisis Orientado a Objetos Mediante la Técnica de Modelado de Objetos OMT I*, México, Soluciones Avanzadas, Junio 1994.

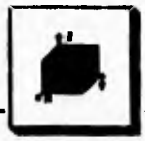
- López, Amparo; Guadalupe, Ibargüengoitia: *Análisis Orientado a Objetos Mediante la Técnica de Modelado de Objetos OMT II*, México, Soluciones Avanzadas, Octubre 1994.

- Manna, Zohar.: *Mathematical theory of computation*. USA, McGraw-Hill, 1974.

- Martínez, Lorenzo M.: Estrategia de protección contra la corrosión atmosférica en los complejos de PEMEX petroquímica en Coatzacoalcos, Instituto de Física, UNAM, 1994

- McCulloch, W. S. and W. Pitts.: *A logical calculus of the ideas immanent in nervous activity*, Bull. Math. Biophys 5, 1943, pp 115-148.

- Minsky, M and S. Papert.: *Perceptrons, an introduction to computational geometry*, Cambridge Mass, MIT Press, 1969.



- Ocaña Servin, H. L.: ***Medio Ambiente, Impacto atmosférico y vías respiratorias***, México, Vesalio Difusión Médica S.A. de C.V., Diciembre 1993.

- Odum, E. P.: *Ecología*, México, Publicaciones Cultural, 1980.

- Oktaba, Hanna.: *Programación Orientada a Objetos ¿Moda o Realidad?*, México, Soluciones Avanzadas, Abril-Mayo 1993, pp 39-42.

- Pielke, R.A.; Lyons, W. A.; et. al,: ***Regional and mesoscale meteorological modeling as applied to air quality studies***, Air Pollution Modeling and its Application VII Volume 15 (NATO Committee on the challenges of nodern society), New York USA, Plenum Press, 1991, pp 259-284.

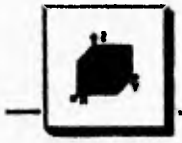
- Pressman, Roger S.: *Ingeniería de software, un enfoque práctico*, México, McGraw Hill, Segunda Edición, 1992.

- Quian, S.; Y. C. Lee; R. D. Barnes; et. al.: *Adaptive Stochastic cellular automata: Theory*, PHYSICA D 45, 1990, pp 159-180.

- Quian, S.; Y. C. Lee; R. D. Barnes; et. al.: *Adaptive Stochastic cellular automata: Applications*, PHYSICA D 45, 1990, pp 181-188.

- Quintanilla, Gloria: *El impacto en la ingeniería de software de la programación orientada a objetos*, México, Soluciones Avanzadas, Septiembre-Octubre 1993, pp 43-46.

- Rumbaugh, J.; Blaha M; Premerlani W; et. al.: *Object Oriented Modeling and Design*, USA, Prentice Hall, 1991



- Sagols, Feliú: *Ingeniería de Software con la Metodología de Objetos*, México, Escuela de verano computación 1994, 1994.

- Schueneman, Jean J.: *Air Pollution Control Administration*, Air Pollution Volume III, New York USA, Academic Press, Second Edition, 1968, pp 719-723.

- Thomas, R.: *Kinetic Logic*, Lecture Notes in Biomathematics, vol 29, Springer Verlag, Berlin, 1979.

- Toffoli, T and H. N. Margolus.: *Invertible cellular autumata: A review*, PHYSICA D 45, 1990, 229-253.

- Toledo, Alejandro.: *Cómo destruir el paraiso*, México, Oceano, 1984.

- Wyngaard, Jhon C.: *New Developments in dispersion parameterization and modeling*, Air Pollution Modeling and its Application VII Volume 13 (NATO Committee on the challenges of nodern society), New York USA, Plenum Press, 1991, pp 417-431.