DATA BUS

Número 3 *Abril 1991*
Depósito legal VG-87-90

• Facultad de Informática para Orense

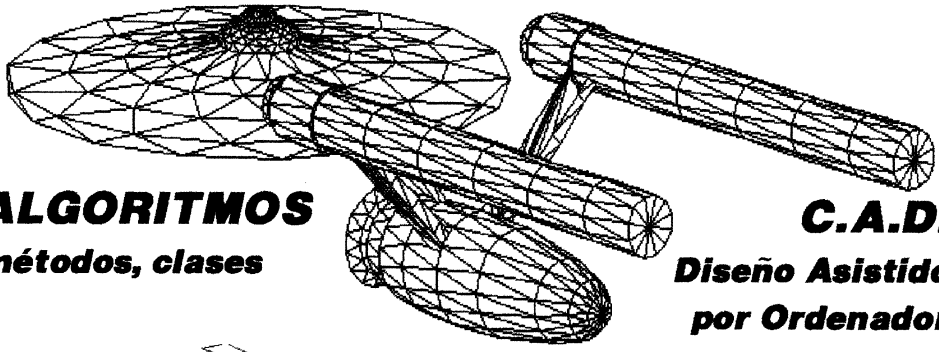
(Vigo). Pese a la protesta del Consejo de la Juventud de Vigo y de nuestra asociación, AJIV, la Xunta de Galicia se ha decantado a favor de Orense para la ubicación de la futura Facultad de Informática. Los motivos de la elección de Orense aún son desconocidos, pero desde luego no se han tenido en cuenta que las solicitudes procedían en su mayoría de Vigo y su comarca. El siglo XXI está cada vez más lejos...

HISTORIA DE LA INFORMATICA

sus orígenes, descubrimientos

IMPRESORAS

El brazo derecho del ordenador

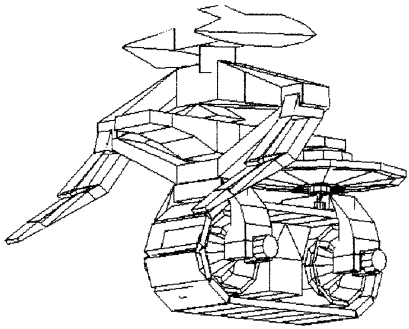


ALGORITMOS

métodos, clases

C.A.D.

Diseño Asistido por Ordenador



¿CÓMO COLABORAR CON A.J.I.V.?

(Pag.3)

DATA BUS

Número 3
Abril de 1.991

DIRECTOR
Jose Manuel Suárez Pousa

REDACTORES
Jesús Cea
Nacho Agulló

COLABORADORES
Telmo Lago
Pablo Lobariñas
Simón Gómez

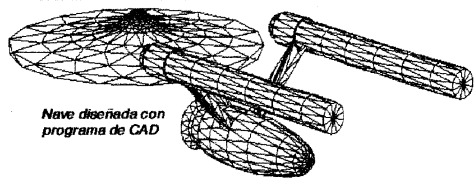
DISEÑO
Jose Manuel Suárez Pousa

MAQUETACION
Jesús Cea

EDITA
A.J.I.V.
Asociación juvenil para la Informática Viguesa
c/Caldas de Reis 12-6 Izq.
Tel: 23 18 35
41 01 13

Depósito legal:
VG-87-90

PORTADA DEL MES



Nave diseñada con programa de CAD

Los ordenadores son unas herramientas de trabajo muy poderosas y versátiles. Uno de los campos en donde han entrado con más fuerza ya desde sus inicios ha sido, sin lugar a dudas, en el C.A.D.

Por medio de los sistemas C.A.D. se simplifica notablemente el proceso de diseño y elaboración de formas, reduciendo enormemente el tiempo de desarrollo que se precisaría en el caso de utilizar los métodos tradicionales.

INDICE

EDITORIAL

Ante todo queremos agradeceros la increible acogida de que haceis objeto nuestra (vuestra) revista. El más que caluroso seguimiento de DATA BUS número a número nos ha hecho reeditar los dos meses anteriores. Nuestro más sincero agradecimiento por vuestro apoyo. GRACIAS.

Aunque hasta ahora sólo han salido a la calle 3 números de DATA BUS, hemos recorrido ya un largo camino juntos. Esperamos estar contribuyendo positivamente a la difusión de la informática como hobby, descubriendo sus secretos e interioridades para vosotros. Siguiendo este afán didáctico van a aparecer en próximos meses nuevas secciones para todos los gustos: microprocesadores, emulación de sistemas físicos, informática y matemáticas, etc.

Naturalmente seguiremos manteniendo nuestras secciones ya clásicas, como "Algoritmos", "Historia de la informática", "Curiosidades informáticas", "Nuevas tecnologías", ... Es posible que alguno note en falta las secciones de "Música por ordenador" y "Guía del profano", por lo que pensamos contar con ellas nuevamente en un futuro próximo.

Esperamos que este número os guste tanto como a nosotros. Y recordad que aguardamos vuestras colaboraciones...

la Redacción

¡¡¡ ATENCION !!!

Si deseas colaborar con nosotros aportando ideas, críticas, artículos, etc., no tienes más que hacernos llegar tu trabajo. Para ello, puedes enviarlo a la dirección de la asociación o ponerte en contacto con la persona que te suministró este ejemplar.

¡ CONTAMOS CON VOSOTROS !

FE DE ERRATAS :

- En la página 7 dice que el Arquímedes es francés, cuando en realidad es inglés.
- En la página 18 varias lineas aparecen con la última letra repetida. Disculpadnos el error.
- En las segundas ediciones de los números 1 y 2 de DATA BUS, algunas imágenes salieron borrosas o muy oscuras. Lamentamos este hecho que no volverá a suceder.

NUEVAS TECNOLOGIAS

Telmo Lago

El chip en 3 Dimensiones

A medida que se va evolucionando en el campo de la fabricación y diseño de circuitos integrados (campo siempre en constante cambio, por otra parte) se va buscando una mayor potencia y capacidad de proceso en el mínimo espacio. Pero de un tiempo a esta parte los técnicos están empezando a toparse con un obstáculo nada deseable y que hasta hace cierto tiempo no era más que una mera suposición o una pregunta hecha para obtener una respuesta teórica. Y cuando no se pueda hacer más pequeño, ¿qué? He aquí la cuestión.

La escala tan alta de integración que se ha llegado a conseguir en diversos circuitos ha llegado a hacer realidad este problema, insalvable debido a las propias leyes físicas. Por ejemplo, reduciendo el ancho de las pistas conductoras aumenta la resistencia, y la conducción eléctrica se hará con más dificultad. Haciendo lo mismo con las capas aislantes, aumenta la intensidad de los campos eléctricos, con lo que podría dañarse el circuito. Y otra: el sobrecalentamiento como consecuencia de la falta de aislamiento entre componentes. Proporcionalmente, la cantidad de calor disipada por un chip es similar a la de una plancha al rojo. Por último, las partículas alfa pueden llegar a invertir el estado de los transistores, que funcionan como biestables (0 o 1). Y suponiendo que un transistor estuviera a 1, lo pondría a 0, o viceversa, lo que causaría graves daños en la maquina. Estos efectos se hacen más patentes a medida que se reduce el tamaño de los circuitos.

También se dan problemas a nivel de fabricación, ya que llega un momento que ni vale el láser, que a ciertas escalas actúa como onda, de manera que es imposible de utilizar. Y después está lo de siempre, el Poderoso Caballero Don Dinero. La cosa tiene que ser rentable, si no, no hay nada que hacer. Una vez alcanzado cierto punto resulta excesivamente costoso superarlo, cosa que las compañías no se pueden permitir así como así.

Para solventar todos estos problemas se empezó a trabajar en un nuevo concepto de circuito integrado: el chip *sandwich*, que como su nombre indica, está compuesto de varios chips "tradicionales" superpuestos en diversas capas. A primera vista esto parece una solución fácil y sencilla. Nada más lejos de la realidad. Cuestión fundamental: ***¿Cómo unir e interconectar las capas?***

Supongamos la primera capa terminada. Hay que montar la siguiente. Para ello hay que cristalizar el silicio puro sobre el aislante (óxido de silicio), y es preciso que lo haga como un sólo cristal (monocristal). Pero sobre el aislante, empleando los métodos habituales lo único que se consigue es una gran cantidad de pequeños cristales desordenados, que no sirven.

Un método empleado en E.E.U.U. fue el de hacer las capas por separado y después juntarlas a modo de sandwich; pero para estructuras muy pequeñas es necesaria una gran precisión en el corte y el ensamblado de las capas, ya que los orificios de contacto tienen que encajar exactamente. Otra forma

consiste en aplicar silicio sobre la capa, y posteriormente inyectarle un núcleo. Para ello es preciso que el silicio éste en estado líquido (a más de 1400 grados centígrados), y a esta temperatura se daña el circuito entero. De momento se está experimentando en fundir el silicio de una forma rápida y en el punto preciso. Con láser se va recorriendo punto por punto la superficie del chip, pero es enormemente lento. Se consigue más rapidez con una varilla al rojo blanco que funde todo a la vez. Pero lo más problematico es estabilizar los múltiples cristales de silicio para que puedan actuar como uno solo.

Segundo problema gordo: interconexión de las capas.

Hay que "aplanarlas". Se sugirió recubrir la capa inferior con vidrio (óxido de silicio en bruto). Después hay que hacer unos pequenísimos y profundos agujeros verticales para introducir el equivalente a los cables de interconexión el un circuito integrado, pero la tecnología actual no está preparada para hacer algo tan preciso.

ALGORITMOS

Jesús Cea Algoritmos de compresión de textos

En el número 1 de nuestra revista publicamos un artículo sobre sistemas de compresión de "propósito general". Sin embargo, como ya dije en su momento, a veces precisamos algoritmos especiales para determinadas aplicaciones, como la compresión de gráficos o, como en el caso que nos ocupa este mes, la compresión de largos textos.

El hecho de utilizar rutinas compresoras específicas para un determinado tipo de ficheros conlleva, generalmente, una mayor velocidad y ahorro de memoria, pero tienen un gravísimo inconveniente; y es que sólo las pueden utilizar los propios programadores del fichero a comprimir (novatos, abstenerse). Ello es debido a que un descompresor específico debe saber en cada instante donde están los datos, su longitud, su formato, etc., y para ello no utiliza una cabecera de descompresión como los algoritmos descritos en el número 1 (si no lo tienes no sabes lo que te pierdes), lo cual implicaría perder parte de las ventajas inherentes a un sistema diseñado para una función tan concreta (si dispusiese de una cabecera que le diera la información, sería una rutina de propósito general), sino que se les debe comunicar desde el exterior. Obviamente, la única persona capacitada (en general) para proporcionar dicha información es su propio programador (en realidad esto no es un problema, ya que los que usan este tipo de compresores suelen ser los mismos que programan los ficheros que luego serán comprimidos...).

Probablemente una de las ventajas más decisivas respecto a los sistemas "más generales" sea el hecho de que se pueda usar la descompresión ¡¡EN TIEMPO DE EJECUCION!! (es decir, ir descomprimiendo SOLO lo que necesitamos en cada momento). Ello

posibilita tener un programa en memoria ¡¡MAS GRANDE QUE ESTA!!. Podemos introducir un texto de 100 Kbytes en una memoria de 64 Kbytes, por ejemplo, ¡¡y seguro que aún sobra sitio!!. Luego recurrimos a la descompresión fragmentada de aquellas frases que nos interesan en cada instante. ¡Algo impensable en un algoritmo de propósito general !. Evidentemente, esto ultimo implica que el programador tenga acceso directo a las rutinas necesarias, lo que no hacen más que confirmar lo que ya comenté anteriormente.

Esta y otras ventajas de los métodos de uso específico (utilizándolos adecuadamente, claro) son tan grandes comparadas con las de los algoritmos generales que los hacen preferibles a pesar de su mayor dificultad a la hora de usarlos, por no hablar de que sólo los pueden utilizar una minoría. Antes de proceder con nuestro estudio vamos a definir un par de conceptos que nos serán muy útiles más tarde. Consideraremos caracteres válidos a las letras, números, signos de puntuación y algunos, pocos, códigos de control (return, escape, fin de texto, etc.); unos 62 valores en total. Téngase muy en cuenta que no se incluyen caracteres mayúsculos y minúsculos sino sólo uno de los dos por razones obvias. Por lo tanto, es importante añadir un código de control que actúe a modo de "SHIFT", posibilitando la conmutación entre ambos juegos. También es muy útil la programación directa de algunas convenciones tales como, por ejemplo, el espacio después de los signos de puntuación, el paso automático a mayúsculas tras un punto o return, etc., que favorecerán la compresión propiamente dicha. También podemos asignar un código propio a los puntos suspensivos... El programa final se complica, pero los resultados son alentadores. Creo que ya estamos listos para empezar. Que os sea leve...

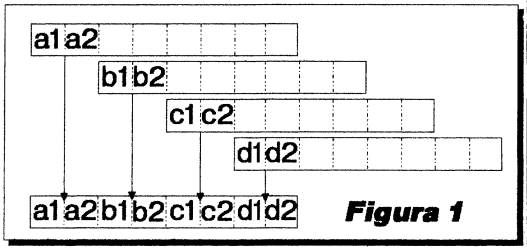


Figura 1

1. COMPRESION POR BITS NO UTILIZADOS

Dado que nuestro texto sólo está constituido por 62 códigos, está claro que se desperdicia buena parte de cada byte, dado que estos pueden almacenar hasta 256 valores diferentes.

Supongamos que todos los caracteres tienen la misma posibilidad de aparecer (hecho este muy alejado de la realidad). Ahora aplicamos:

LOG_2 62=5.92. Tomemos 6 para redondear. Así llegamos a la importante conclusión de que desperdiciamos 2 de los 8 bits que componen cada byte (bits que estarán siempre a 0 y que, por tanto, son redundantes). ¿Por qué no intentar aprovecharlos? La solución más que evidente consiste en la solapación de los bits no usados de un byte por parte de algunos de los bits sí utilizados del siguiente. Dada que comprimimos 8 bits en 6 ahorramos un 25%. En otras palabras, podemos reducir 4 caracteres a sólo 3 bytes. El diagrama de solapamiento (ver figura 1) es similar al descrito en el algoritmo de compresión de bits del número 1 (de hecho hace esto mismo, pero de una forma menos eficaz al tratarse de un sistema de ámbito más general). A pesar de que no resulta demasiado eficiente, este método es muy utilizado por su sencillez y velocidad. ¿Alguien da más por menos?

2. COMPRESION DE BITS NO UTILIZADOS Y CAMBIO DE

BANCO

En el ejemplo anterior presuponíamos, inocentemente, que los 62 códigos tienen las mismas posi-bilidades de salir. En realidad sabemos perfectamente que esto no es así. Las letras, el espacio y algún que otro signo más aparecen con mucha más frecuencia que los números, paréntesis, etc. Recordé mis viejos tiempos de radioadicto (radioaficionado, claro. No vayais a pensar mal) y, con ellos, mis programas para RTTY (radioteletipo). Internacionalmente se utiliza el sistema BAUDOT para RTTY, AMTOR, télex profesional de agencias de prensa, embajadas, etc. El sistema BAUDOT (el estandar, dado que hay muchas variantes, para el AMTOR por ejemplo) se basa en el código MURRAY (casi tan viejo como la misma radio). Ver figura 2. El código MURRAY utiliza 5 bits, 32 valores distintos, para transmitir los datos. ¿Cómo se consigue transmitir todos los códigos precisos para una comunicación normal con sólo 32 códigos diferentes? La respuesta es sencilla. Basta añadir un código de cambio de banco.

Supongamos que estamos imprimiendo letras y el siguiente caracter que sigue es un número. En ese instante enviamos el código de cambio de banco. Los caracteres que sigan serán impresos según "el banco de cifras". Cuando queramos volver a imprimir letras volvemos a poner un código de conmutación y lo que siga será tratado según "el banco de letras". El sistema es identico al del "SHIFT" de mayúsculas/minúsculas. Es conveniente la programación de cierta información usada frecuentemente, como el paso automático a letras después de un espacio, etc. Optimamente, este sistema consigue reducir el texto en un 37.5%. En realidad el porcentaje es algo más bajo debido a los cambios letras/cifras y viceversa, pero no suele bajar del 32% en el peor de los casos.

Es evidente que debemos utilizar una variante del código MURRAY modelada según nuestras necesidades propias, que incluya sólo los códigos que nos interesen y en el orden y banco más ventajoso que podamos. Ante todo, debemos eliminar la duplicación de los últimos 6 códigos, el LINEFEED-incluido por programación en el return-, un sólo código de cambio de banco en vez de dos, etc.

Las aparentes redundancias que presiden el código MURRAY son imprescindibles en su empleo original como protocolo de comunicación a distancia. Imaginaros que interceptamos una comunicación por la mitad (o que simplemente hemos perdido unos cuantos caracteres debido a las inevitables interferencias, QSB, etc.). En esos momentos críticos si estamos situados en un banco distinto del que está usando el emisor nos encontraremos con caracteres sin sentido en la salida. En el caso de utilizar un único código para el cambio de banco no tendremos más remedio que recurrir a un cambio "manual" debido a que si, por ejemplo, nosotros estábamos en cifras y el emisor en letras, al enviar el código de conmutación, nosotros pasaremos a estar en cifras y él en letras. La única solución es esperar que otra interferencia nos ponga en el modo correcto (por las leyes de MURPHY, seguro que perderemos mientras tanto lo más interesante) o podemos recurrir al cambio manual incluido en algunos programas (incluidos los mios).

En el caso de utilizar dos códigos distintos para cifras/letras y letras/cifras, el problema no es tal, dado que si estamos en modos distintos, al recibir un código de cambio nos situamos automaticamente en el banco correcto. Identica finalidad tiene la duplicación del espacio, así como el cambio automático a letras después de ellos. En estos

casos sólo perderemos 2 ó 3 caracteres hasta el siguiente espacio (a no ser que el error coincida en medio de una palabra muy larga, caso raro dado que en radio se utilizan casi exclusivamente abreviaturas para evitar esto mismo, además de crear así un lenguaje universal y reducir también el tiempo necesario para la emisión). Como en el caso que nos ocupa, las interferencias no nos preocupan, podemos redefinir los códigos redundantes, que ahora son superfluos, para la tarea que mejor nos convenga. Los detalles que los ponga cada uno a su gusto, aunque os aseguro que es muy fácil. Hasta es probable que os sobre algún código...

Alguien podría pensar en poner 4 bancos de 16 valores cada uno, para reducir a 4 los bits utilizados y conseguir de esta forma un ahorro del 50%. Ello es perfectamente factible aunque los casi constantes cambios de banco hacen bajar mucho el porcentaje. Así mismo, necesitamos ahora 3 códigos de cambio en vez de uno y para poder acomodarlos todos, deberemos eliminar algunos caracteres (seguro que los paréntesis, el apóstrofe, etc., serán los primeros en caer). El resultado final es similar o incluso algo más bajo que el sistema "normal", pero tiene la gran ventaja de que 2 códigos ocupan EXACTAMENTE un único byte, ahorrán-

Figura 2

CODIGO MURRAY ESTANDAR:

Códigos	Letras	Figuras
11000	A	- menos
10011	B	? interrogración
01110	C	: dos puntos
10010	D	N.D. asterisco
10000	E	3
10110	F	N.D. paralelas
01011	G	N.D. porcentaje
00101	H	N.D. dólar
01100	I	8
11010	J	BELL
11110	K	(
01001	L	)
00111	M	. punto
00110	N	, coma
00011	O	9
01101	P	0
11101	Q	1
01010	R	4
10100	S	' apóstrofe
00001	T	5
11100	U	7
01111	V	=
11001	W	2
10111	X	/ barra
10101	Y	6
10001	Z	+ más
00010	RETURN	
01000	LINEFEED	
11111	LETRAS	
11011	FIGURAS	
00100	ESPACIO	
00000	N.D.	

Los códigos se transmiten de izquierda a derecha (se leen al revés), representando el "1" la mayor frecuencia de la señal FSK.

N.D. significa que esos códigos no están definidos, constituyendo así reservas para futuras ampliaciones. De todos modos, el código AMTOR tampoco los utiliza, sino que aprovecha otros 6 códigos adicionales obtenidos debido a la mayor longitud disponible (7 bits en vez de 5).

Algunos de los códigos N.D. han sido utilizados en versiones modernizadas del código MURRAY para la transmisión de nuevos caracteres no contemplados en un principio. El 0 sigue igual debido a que posee una importante "PROCLIVIDAD" a dispararse en cuanto hay la más mínima interferencia. De esta forma, simplemente es ignorado, no reflejándose ningún error en la salida.

Los últimos 6 códigos son los mismos tanto en el banco de letras como en el de figuras (ver explicación en el texto).

donos así todo el trabajo de llevar un recuento de bits, con el problema adicional de que un caracter pueda estar repartido entre varios bytes. Es fácil de programar pero lento. Con el sistema "reducido" sabemos siempre donde empieza y termina cada caracter y no existe la posibilidad de que un dato quede "a caballo" de dos octetos, simplificando, reduciendo y acelerando así el descompresor (y la programación del mismo también). Esta ventaja quizás lo hace preferible a pesar de que los resultados a nivel de compresión no son, en general, tan buenos como el sistema de 5 bits. A todo esto hay que decir que los resultados a obtener dependen en gran medida de la organización correcta de cada uno de los 4 bancos. ATENCION, que a nadie se le ocurra siquiera construir cada banco según el orden alfabético, por ejemplo. Eso es catastrófico. La constitución de cada banco es tan crítica, que incluso puede variar de una frase a otra. Ni que decir que es una tarea ardua. Como pista os diré que es conveniente incluir en un mismo banco aquellos códigos que suelen aparecer asociados, como es el caso de la "Q" y la "U", la "C" y las vocales, etc.

3. COMPRESION DE BITS POR ARBOL BINARIO

Hasta ahora, todos los sistemas que hemos estudiado utilizan códigos cuya longitud era igual para todos los caracteres (exceptuando los cambio de banco). La idea fundamental de este método consiste en que los códigos más utilizados sean los más cortos y los menos frecuente, los más largos. Ello conlleva implicitamete que los diferentes caracteres posean (en general) diferentes longitudes, complicando un poco la programación. Sin embargo, los resultados son tan espectaculares que merece la pena sacrificar un par de días poniendo a punto el programa. Para la localización de los diferentes caracteres debemos diseñar un arbol binario en el que cada rama terminal represente un código distinto. Los códigos más frecuentemente utilizados deben estar más al principio, más cerca de la "RAIZ". Por ello, su expresión a nivel de bits es más corta que la de los códigos más "lejanos".

Obviamente, el diseño de un árbol binario eficaz no es trivial ni mucho menos. La construcción del árbol óptimo sería una tarea larga y difícil si no fuera por ciertos teoremas de la teoría de la información, enunciados por SHANNON y por el algoritmo generador de HUFFMANN, utilizado, además de para la compresión de textos, gráficos, etc., para la criptografía, análisis estadísticos, optimización de tiempos y de capacidad de almacenaje... No voy a extenderme sobre dicho algoritmo dado que ya hay un artículo sobre él en este mismo número.

En general, cada texto generará un árbol binario distinto, aunque a partir de cierta longitud crítica, las variaciones de uno a otro son practicamente despreciables. Ello permite crear un árbol que, aunque no sea el óptimo en todos los casos, es lo suficientemente general como para ser válido en casi todas las aplicaciones. Los perfeccionistas siempre pueden optar por diseñar un árbol distinto para cada texto...

Evidentemente, no todo son ventajas. Junto al inconveniente de la longitud variable para cada código (que en realidad es el corazón mismo de este procedimiento), el almacenamiento del árbol escogido también es un problema. Y grave, además. Una vez creado el árbol, debemos almacenarlo de la forma más compacta y práctica que sea posible (premisas normalmente opuestas por aquello de las leyes de ...). Debemos saber conjugar al máximo la ocupación de memoria y tiempo de acceso a cada elemento en particular.

Por las leyes de MURPHY, como siempre, sabemos que si deseamos un tiempo de acceso reducido debemos sacrificar memoria y viceversa (esto es una de los problemas históricos y universales de la informática).

Las técnicas utilizadas para ello son muy variables en cada caso y dependen en gran medida de la habilidad de cada uno, así como de lo que estemos dispuestos a pagar. Ni siquiera hay una solución general dado que ésta depende, incluso, del microprocesador utilizado. Sin embargo, en este caso, la solución más sencilla es, al mismo tiempo, la más rápida, aunque precisa unos 768 bytes para el árbol (rutina aparte). La estructura es la siguiente: (figura 3)

TIPO, NUM1, NUM2; donde TIPO indica si ambos son códigos finales, punteros o uno un código y el otro un puntero (en realidad sólo se precisan 2 de los 8 bits disponibles) y NUM1/2 representan el código final ó un puntero a la siguiente rama, de estructura idéntica a ésta misma. Con esta organización se nos permite almacenar hasta 256 códigos distintos. Como en nuestro caso sólo necesitamos 62, podemos rediseñar el sistema como sigue:

NUM1, NUM2; donde el bit 7 de NUM1/2 indica si se trata de un dato ó de un puntero, haciendo las veces de TIPO en el caso anterior. Con esta nueva organización, el árbol sólo ocupa 124 bytes... (tanto en este caso como en el anterior, NUM1/2 representan el offset hasta la siguiente rama a la que apuntan, indicando sólo cuantas ramas, y no cuantos bytes, han de saltarse. Para conocer el desplazamiento real en bytes, debemos multiplicar NUM1/2 por 3 ó 2 respectivamente.)

4. POR COMPRESION RELACIONAL

Los algoritmos vistos hasta ahora comprimen cada código individualmente, sin que exista relación alguna entre él mismo y los que le preceden y/o le siguen. El objetivo prioritario de los otros sistemas consiste en insertar el máximo de información en el mínimo espacio (para eso sirven los compresores...).

En este sentido, todos funcionaban de modo similar. Las diferencias con respecto al algoritmo que vamos a estudiar ahora son más que notables ya que son sistemas radicalmente distintos que cumplen los mismos objetivos recurriendo a consideraciones opuestas. Los sistemas anteriores intentaban acomodar 62 códigos desperdiciando lo menos posible. Este sistema utiliza los 256 códigos disponibles en un byte, facilitando además la programación a no tener datos esparcidos entre octetos adyacentes. Pero, ¿de dónde salen los restantes 194 códigos?.

Pensando un poco, nos damos cuenta de que tras ciertos códigos suelen aparecer otros determinados. Después de una "Q" suele (siempre) aparecer una "U", después de un punto suele aparecer un return (el espacio ya ha sido incluido por programación), después de la "Y" suele aparecer un espacio, etc.

Nuestro algoritmo se aprovecha de estas "anomalías" estadísticas de la siguiente forma: los primeros 62 códigos representan los caracteres normales "individuales". Los 194 códigos restantes son "abreviaturas" de aquellas "moléculas" formadas por más de una letra que se repitan a menudo. Podemos tener en cuenta 3 ó 4 letras además de 2, ahorrando todavía más (la secuencia "QUE" más espacio podría se comprimida a un sólo byte, etc.).

La construcción de una tabla es bastante general dado que en textos grandes la regularidad estadística nos garantiza tablas iguales en textos

distintos (siempre y cuando sean lo suficientemente largos). La tabla, en cambio, varía muchísimo de un idioma a otro dado que el vocabulario es totalmente distinto. Por ejemplo, en Inglés es muy común el "THE" mientras que en castellano no conozco ni una sola palabra que lo contenga. En los casos de acentos y la eñe (que hasta ahora no eran contemplados) se pueden utilizar códigos mayores que el 62 para expresarlos. Secuencias como "EL", "-CION", "-ANTE", "-MENTE", etc., pueden ser comprimidas a un sólo byte (o a dos como mucho). El nivel de compresión global depende, sobre todo, de la longitud escogido (2, 3, 4 letras...) para las secuencias. Utilizando 3 letras por ejemplo, como longitud, se llega facilmente a un nivel de compresión del 50% y con una programación extremadamente sencilla. Genial, ¿verdad?

5. POR COMPRESION DE PALABRAS:

Hasta ahora, todos los algoritmos anteriores realizaban la compresión tomando como unidad básica los caracteres individuales o, en el mejor de los casos, un pequeño átomo de 2, 3 ó 4 códigos. El sistema que nos ocupa utiliza las palabras como unidad mínima de compresión.

En el caso de que sólo tengamos 256 palabras distintas en todo el texto (un vocabulario un poquillo escueto), nos basta asignar un código 0-255 a cada palabra distinta. Debemos recordar que secuencias tales como la coma, punto más return, etc, cuentan como palabras.

Figura 3

TIPO	NUM1	NUM2
XXXXXXAB	UUUUUUUU	DDDDDDDD

Leyenda:

X. Irrelevante (generalmente este bit estará a 0)
A. Un cero significa que NUM1 representa un enlace a otra rama. Un uno quiere decir que nos encontramos en una rama terminal, por lo que NUM1 es ya el dato buscado.
B. Idem NUM2.
U. Dato contenido en NUM1 (un puntero a otra rama o un dato terminal).
D. Idem NUM2.

Este sistema permite representar hasta 256 códigos distintos. Dado que sólo necesitamos 62, podemos abreviarlo a :

NUM1	NUM2
AUUUUUUU	BDDDDDDD

De esta forma ahorramos, al menos, un 1/3 en comparación con el sistema anterior. El ahorro real es mayor dado que el número de códigos a representar es más pequeño. Para 62 códigos necesitamos unos 124 bytes, rutina aparte. Los significados de los símbolos anteriores son idénticos a los del primer caso.

Para conocer otros sistemas utilizados, recomiendo leer el artículo sobre el algoritmo generador de HUFFMANN.

En el caso más general de utilizar un vocabulario de N palabras, asignamos a las 255 palabras más utilizadas los códigos 1-255. El código 0 se identifica con el "cambio de banco" de anteriores sistemas. Después de un código 0, se repite el mismo proceso pero utilizando en esta ocasión el segundo banco de palabras, que serán las segundas más comunes. Si volvemos a encontrar otro 0 en lugar de un código válido (1-255), repetimos todo otra vez utilizando ahora el banco 3, y así ad infinitum.

Ni que decir tiene que reducir una palabras de 5 letras de media más un caracter para el espacio a un sólo código representa un nivel muy importante de compactación. De 6 a 1. No debemos preocuparnos por lo que ocupe el cambio de banco siempre y cuando las palabras hayan sido bien catalogadas en función de su frecuencia relativa de aparición. Generalmente basta con utilizar 2 ó 3 bancos, que representan 511 y 766 palabras distintas respectivamente (es importante recordar que los signos de puntuación, códigos de control, etc., también cuentan). Si precisamos un vocabulario mayor, para un diccionario, por ejemplo, siempre podemos optar por aumentar el número de bancos (el número total de palabras disponibles es 255*B+1, donde B es el número de bancos) o buscar alguna variante mejor, como elegir un código que permita expresar las palabras poco frecuentes como cadenas literales, o un código que permita especificar directamente el número de banco en vez de andar saltando de uno a otro secuencialmente. Como se suele decir en estos casos, todo depende de la habilidad y conocimientos (y ganas de trabajar) que posea cada uno. Valor, ¡y al toro!.

Cuando menos resulta paradójico (por no decir increible) que el algoritmo más sencillo de todos los que hemos tratado aquí sea el que mejores resultados proporciona a nivel de eficiencia (en ocupación de memoria y en velocidad). Por una vez se ha roto el maleficio...

Logicamente, los algoritmos estudiados este mes pueden combinarse para crear un "SUPER-COMPRESOR" de gran eficacia. Podemos, por ejemplo, expresar las palabras del algoritmo anterior a través de un árbol binario o, incluso, programar un algoritmo similar al de compresión relacional pero tomando como unidades a las palabras en vez de a las letras individuales (pudiendo, adicionalmente, expresarlas luego a través de un árbol...). Aunque no lo haya dicho, para la creacción del árbol óptimo o de unas tablas de calidad solemos dejar el trabajo pesado de cálculo al ordenador, que para eso está, utilizando un programa a tal efecto previamente escrito por nosotros mismos. No queremos que nadie muera de un infarto en medio de una inmensa serie de cálculos laboriosos, aburridos y, sobre todo, largos, muy laaaaaaargos...

Tampoco creais que estos sistemas son "todo lo que hay" en este campo, ni mucho menos. Sólo constituyen una pequeña muestra de lo que podeis encontraros si investigais un poco. Muchos de los algoritmos que circulan por el mundo adelante, más que para comprimir, tienen una función criptográfica. A los programadores no les suele gustar tener "ojos curiosos" hurgando por ahí sin control e intentan ponernos las cosas difíciles recurriendo a autenticas virguerías, algunas de las cuales nos hacen recordar aquello de que siempre hay alguien dispuesto a rizar el rizo. Y es que, aunque no lo confiesen, todos empezamos de la misma manera...

EL ORDENADOR Y SU ENTORNO

Pablo Lobariñas

Impresoras

En este artículo, como bien indica su título, vamos a tratar sobre las impresoras. ¿Qué son exactamente?, ¿en qué consisten?. Después de esta aproximación inicial, veremos que tipos pincipales hay y cuales son sus ventajas e inconvenientes.

Aunque este artículo no va a ser, logicamente, muy exaustivo, ya que nuestra revista pretende ir desde lo más simple, a lo más complicado, sí en cambio, se dará una visión general sobre este tema.

¿Qué es una impresora?

Lo primero que habremos de decir acerca de una impresora, es que se trata de lo que habitualmente se conoce como periférico. Es decir, se trata de un accesorio del ordenador que amplía y mejora sus posibilidades (aunque en multitud de ocasiones pasa a formar parte indispensable de éste). Este periférico llamado impresora, tiene como misión imprimir textos y gráficos sobre papel, de forma que actúa como una máquina de escribir aunque, claro está, con ciertas diferencias ya que, al ser parte integrante de un ordenador, posee una serie de características y posibilidades que no tiene una máquina de escribir. Así, permite imprimir gráficos sobre papel, que dependiendo del tipo de impresora puenden tener una calidad y definición increibles. Además poseen la ventaja de que pueden escribir en multitud de formas, y pasar de una a otra sin ninguna dificultad. Es decir, podemos hacer que nuestro texto aparezca escrito en caracteres góticos, romanos, o los que sean, pulsando una tecla, y en cuanto queramos. Claro que, debemos tener en cuenta que todas estas posibilidades dependen del tipo de impresora, ordenador, y sobre todo programa que estemos utilizando. Por tanto, y según hemos dicho, la función principal de una impresora es poner sobre papel todo aquello que nos interese, desde una carta, hasta una pantalla de ordenador. Podríamos decir, como comparación, que nuestro brazo derecho (o izquierdo) es nuestra impresora, que actúa según las órdenes de nuestro cerebro, que es como nuestro ordenador central. Para poder llevar a cabo todas estas operaciones, la impresora consta de una serie de partes que realizan una serie de determinadas funciones. Aunque según el tipo de impresora, estas partes varían, podemos decir en general que este dispositivo consta de:

- un aparato de impresión que, dependiendo del tipo que sea, puede consistir en una matriz de agujas, en un láser, un inyector de tinta, etc...

- un rodillo sobre el que se coloca el papel, y que se mueve automaticamente (en vertical) en el momento de la impresión.

- una memoria en la son introducidos los tipos de caracteres a imprimir.

- un dispositivo de entrada/salida a través del cual se conecta la impresora al ordenador.

Uno de los problemas que tenían las impresoras, y que en algunos casos siguen teniendo, es la fuerte tensión a que estaban sometidas, en cuanto a trabajo se refiere, de ahí que actualmente y desde hace algunos años, se estén introduciendo nuevas impresoras

mucho más fáciles de manejar, de cuidar, y mucho más duraderas y fiables.

Tipos de impresoras:

Hablaremos en este apartado de los tipos de impresora más usuales y los más conocidos.

a) La impresora de matriz de puntos por impacto:

Como en casi todas las impresoras actuales, el carro portapapel se mantiene quieto mientras la cabeza impresora se mueve a lo largo del mismo. La cabeza impresora está formada por una columna de agujas verticales. Cada aguja tiene un solenoide asociado (un solenoide es un circuito eléctrico con una serie de características especiales), cuando el solenoide se activa, la aguja correspondiente recibe un fuerte empujón, golpeando el papel a través de una cinta impregnada de tinta, poniendo un punto sobre el papel. Combinando las agujas de forma correcta, y moviendo la cabeza de impresión se formará el caracter correspondiente. Este tipo de impresoras suelen ser las más rápidas en cuanto a velocidad de impresion, variando ésta desde los 80 caracteres por segundo (c.p.s.) hasta lo 400. Actualmente, este tipo de impresoras emplean nueve agujas para producir una matriz de 7x9 o 9x9. Estas impresoras tienen el problema de que no sirven para el trabajo profesional, y de ahí que se haya intentado un nuevo método para este tipo llamado impresión multipaso, que consiste en que la cabeza de impresión pasa dos veces po la misma linea, una vez en una dirección, y la siguiente en la contraria, de forma que la linea queda más marcada y se notan menos los puntos, ya que variablemente, hay una pequeña falta de alineamiento entre los caracteres impresos y la linea de agujas, con lo que éstas tienden a imprimir en los espacios en blanco, claro que su velocidad queda reducida a la mitad.

b) Impresora de margarita o de pétalos:

Este tipo lleva un elemento impresor intercambiable, que se parece, precisamente, a una margarita con los pétalos surgiendo radialmente desde un núcleo central. Cada pétalo lleva un caracter en la punta, que es golpeando cuando llega al sitio correcto. Si queremos cambiar de tipo de caracteres sólo tendremos que cambiar de margarita, la cual es extraible facilmente. Aunque la calidad de impresión es igual o superior a la de una máquina de escribir, su velocidad es bastante inferior a la de una impresora matricial, variando entre 8 y 70 c.p.s.. Para ganar velocidad algunas de estas impresoras aprovechan el retorno de carro para imprimir otra linea, con lo que se gana un tiempo antes perdido. Este tipo de impresoras permite el subrayado, la negrita y se pueden combinar distintos tipos de letra cambiando la margarita entre una primera y una segunda pasada. Las modernas margaritas son de plástico, muy ligeras y sus pétalos son elásticos, lo que permite que vuelvan a su posición inicial después de cada golpe.

c) Impresoras láser y mediante chorro de tinta:

Este tipo de impresoras son mucho más costosas que las anteriores pero su calidad de impresión es inigualable (sobre en el caso de la láser), además su mantenimiento es barato y tienen una duración mayor. En el método de impresión mediante chorro de tinta, los puntos se obtienen disparando una gota de tinta sobre el papel. El punto se dispara mediante una chispa eléctrica que se produce en el interior de la cápsula de tinta. Este sistema es rápido, no es ruidoso y no presenta ningún elemento que pueda deteriorarse por el uso. Cuando la cápsula de tinta se vacía no hay más que sustituirla por una

nueva. El otro sistema, más caro, consiste en escribir un texto mediante un rayo láser dirigido e imprimir las marcas después mediante algún tipo de proceso xerográfico. Aunque, como se ha dicho resulta más caro, como no hay ningún dispositivo mecánico que toque el papel, puede ser extremadamente rápido y con una calidad realmente increible (y si no comprobadlo vosotros mismos leyendo esta revista).

En cuanto al papel para la impresión se utilizan de varios tipos, dependiendo en parte, del tipo de impresora y según lo que deseemos hacer. Normalmente todas las impresoras admiten cualquier tipo de papel (siempre que su anchura no rebase la de la máquina), y puede consistir en hojas sueltas, o en rollos plegados (como los que todos conocemos) que tienen agujeros en los lados, mediante los cuales, la impresora va hacer avanzar el papel. Este tipo es el que se suele emplear para imprimir listados y gráficos, mientras que las hojas sueltas, se suelen emplear para trabajos más profesionales como cartas, informes, artículos,etc... Por supuesto existen algunas impresoras (aunque las menos extendidas) que debido a su sistema de impresión necesitan de un tipo de papel especial, más costoso que el habitual, pero este tipo de impresoras va desapareciendo poco a poco, debido al engorro y coste que supone utilizar este tipo de papel especial (es el caso de las impresoras electrostáticas y térmicas).

Y hasta aquí lo que da de si este artículo. Esperando que os haya gustado me despido de vosotros hasta la proxima vez.

HISTORIA DE LA INFORMATICA (III)

Nacho Agulló

Los primeros ordenadores

En la parte anterior hemos visto como la automática alcanzó el suficiente grado de perfección como para ser usada en el tratamiento de información, y así llegó en ayuda de las matemáticas para, junto con ellas, crear la informática.

En esta parte veremos como, durante el siglo XIX, la revolución industrial surgida a raíz de la invención de la máquina a vapor, va a suponer avances en la automática que acabarán por repercutir en la informática. De momento, la humanidad disponía por vez primera de una máquina capaz de proporcionar una energía constante, lo que permitía automatizar infinidad de trabajos que antes tenía que realizar el ser humano.

Por otra parte, la mejora de la calculadoras mecánicas continuaba: en 1779 Mattieu Hahn diseñaba y construía la primera calculadora capaz de realizar las cuatro operaciones matemáticas. En 1807 Wahl, en Estados Unidos, realiza una máquina calculadora impresora capaz de escribir totalizadores, almacenar y registrar datos, partiendo de las ideas y realizaciones de Jacquard. Esta calculadora respondió a una gran necesidad del mundo contable de disponer de un dispositivo que automatizara los cálculos.

Más tarde, partiendo del trabajo de Wahl, Remington realizó su "máquina

contable".

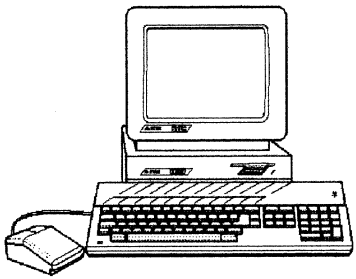
La fabricación en serie de máquinas calculadoras se inició por primera vez en 1810, en Alsacia (Francia), por parte de Charles Xavier Thomas de Cormal, director de una compañía de seguros.

Las máquinas de Babbage:

En este panorama de continuas mejoras y automatizaciones, pero sin ningún salto cualitativo real, va a aparecer un adelantado a su época, un pionero de la informática moderna: Charles Babbage.

En 1812 pensó por primera vez en la realización de un "ingenio diferencial", que terminó en 1822. Dicho ingenio podía calcular tablas de valores a partir de cualquier función matemática, siendo de especial aplicación para las tablas náuticas, usadas en navegación.

Tras la presentación de este ingenio a la Royal Society, le fué concedida una subvención de 1.500 libras por parte del Congreso. Babbage se aplicó entonces a realizar una nueva máquina, capaz de realizar tablas más complejas y precisas, tabulando polinomios de sexto grado con treinta posiciones decimales. El principal problema que Babbage tenía para desarrollar sus diseños era la falta de precisión de la ingeniería de la época, que no podía proporcionarle las piezas torneadas exactamente para que encajaran bien.



Gracias a la ayuda de su amigo al duque de Wellington, a la sazón primer ministro, pudo ir consiguiendo más dinero, hasta un total de 17.000 libras. Sin embargo, en 1834, finalmente el gobierno retiró su apoyo económico al proyecto. El ingeniero Joseph Clement, colaborador de Babbage en el proyecto, lo abandonó también poco tiempo después.

Entonces Babbage de dedicó a otro proyecto aún más ambicioso: el "ingenio analítico", que debía ser un computador universal. No sólo debía ser capaz de realizar los cálculos del "ingenio diferencial", sino cualquier otro tipo de cálculo.

Anteriormente, un alemán, J.H. Müller había planteado ideas similares, incluso dando soluciones a algunos aspectos mejores que las aportadas por Babbage; pero sus ideas tuvieron muy poca repercusión, además de no tener plasmación real.

Lady Ada Augusta, Condesa de Lovelace e hija de Lord Byron, se unió al proyecto. Tenía grandes dotes de matemática y comprendió plenamente el proyecto de Babbage, profundizando tanto en él como el propio autor, y escribiendo programas para el mismo. Describió el ingenio analítico diciendo: "La máquina analítica entrelaza modelos algebraicos del mismo modo que el telar de Jacquard lo hace con flores".

La memoria del ingenio almacenaba mil cifras decimales. Estructuralmente, se parecía mucho a los modernos ordenadores; contaba con un mecanismo similar a una máquina de escribir para imprimir los resultados; y para introducir los datos, aunque al principio contó con unos clavos largos, al estilo de un organillo, después Babbage adoptó el sistema de tarjetas

perforadas que inventara Basilio Bouchon en 1725, y que adoptara Joseph Jacquard para su telar, en 1802. El ingenio contaba también con un "molino aritmético", antecesor de los procesadores actuales, que operaba los datos. Una sofisticación importante era que a través de las tarjetas perforadas no sólo se indicaban los datos, sino también la operación a realizar. El ingenio también podía comparar los datos y actuar de acuerdo a la comparación, con lo que se producían bifurcaciones condicionadas: era, por tanto, autenticamente programable.

En 1852, cuando contaba con tan sólo treinta y seis años, murió Ada Lovelace. A partir de entonces Babbage tuvo que continuar su proyecto en solitario.

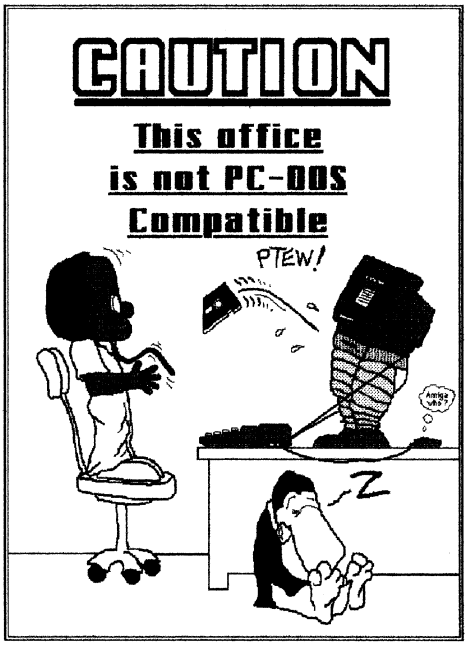
En 1862 el ingenio, parcialmente completo, fué exhibido en South Kensington (Londres). Sin embargo, Babbage no llegaría a verlo completo.

No obstante, su hijo Henry Babbage, consiguió terminarlo y presentarlo en 1910 a la "Astronomical Society".

Como puede verse, en el siglo XIX, asistimos a un continuo esfuerzo por mejorar las máquinas de cálculo automático. El mundo mercantil necesita de ayuda mecánica para una mejor administración y contabilidad, ante unas cifras cada vez más altas. Esta motivación para el avance de la informática no va a decaer, sino que se va a mantener, en continuo aumento, hasta nuestros días.

Por otra parte aparecen nuevos estudiosos relacionados con la informática que, al igual que Babbage, aportan de forma aislada nuevas ideas que más tarde llegarán a ser usadas para la producción de máquinas.

Avances del cálculo automático en el siglo XIX



En 1820 aparece el *Aritmómetro*, máquina capaz de sumar, restar y multiplicar. Disponía asimismo de un borrador de puesta a cero.

En 1841 Roth aportó a las máquinas de calcular un dispositivo para desembragar aportes sucesivos y así evitar el bloqueo o desbordamiento.

En 1849 Maurel y Fayet diseñaron el primer multiplicador automático sin intervención de operador, excepto para registrar los factores.

En 1871 Grant realiza un sistema de cremalleras en el cual una cifra a añadir determinaba el recorrido; mediante este sistema se resolvió el problema de provocar la rotación de ruedas.

En 1875 Barbour, en Boston, construyó la primera máquina de calcular capaz de imprimir números, tanto datos como resultados.

En el mismo año se patentó en

Estados Unidos la "rueda Odhner", sistema similar al cilindro de Leibniz, que significó el comienzo de la industrialización de la fabricación de calculadores.

En 1878 el español Ramón Verea García (1833-1899) inventó una calculadora que multiplicaba directamente y no por reiteración. Leon Bollée construiría otra máquina que resolvía el mismo problema en 1887, nueve años más tarde.

En 1884 aparece por vez primera el tablero de teclas cifradas para inscribir los números a tratar por la máquina de calcular, antecesor de nuestros actuales teclados. Hasta la fecha, se utilizaban estiletes, manivelas o cursores para inscribir los números.

Nuevas energías en la industria:

La aparición del vapor en el siglo XVIII revolucionó la industria. En el siglo XIX ésta verá la aparición de nuevas energías; aparecerán el motor eléctrico y el motor a explosión.

El primer motor eléctrico se construyó en 1834, y funcionaba a base de pilas, no resultando práctico ni económico.

Hasta que el belga Gramme, en 1872, inventó la dinamo perfeccionando la "máquina electromagnética reversible" del físico italiano Pacinotti, no se pudo contar con un suministro de electricidad utilizable. Gramme pronto comenzó la fabricación industrial de su invento, que sería fundamental para los posteriores progresos en la automatización.

Por otra parte, en 1877 aparece el primer motor de explosión a gas, construido por Nikolaus A. Otto, seguido en 1885 por el motor de Daimler Benz alimentado por derivados del petróleo, con el que enseguida comenzarían a rodar los primeros automóviles. En 1893 apareció el motor Diesel.

En un breve período de tiempo la humanidad había multiplicado sus formas de obtener energía, dando la industria un gran paso hacia delante; y con ella, la automatización.

Nuevos aportes matemáticos: Boole

Mientras la automatización avanzaba camino de proporcionar el soporte físico para nuestros modernos ordenadores, lo que hoy constituye el soporte lógico de los mismos cobraba forma a su vez. Un avance significativo lo constituyó la aparición del álgebra Booleana en 1847.

Boole se aplicó a la investigación de la lógica de las decisiones humanas, plasmando los razonamientos de las personas en terminos matemáticos. Aunque su lógica es demasiado compleja para ser expresada aquí, podemos destacar su aportación de los operadores lógicos AND (y) y OR (o),que han dado lugar a las puertas lógicas, imprescindibles en nuestros actuales ordenadores.

DATOS BIOGRAFICOS

George Boole (Lincoln, 1815-Cork, 1864).- Hijo de un zapatero remendón, aprendió matemáticas por sí mismo. tras la publicación de sus ideas, adquirió notoriedad y llegó a ser el primer profesor de matemáticas en la recién creada universidad de Cork. En 1854 estudió las ecuaciones diferenciales y el cálculo de las "diferencias finitas" en su obra "Estudio de las leyes del pensamiento".

CURIOSIDADES INFORMATICAS

Jesús Cea

Fractales (II)

TRANSFORMACIONES AFINES

En el número anterior explicábamos someramente los "fractales recursivos" y hacíamos una distinción estre éstos y los "generados por fórmula". Este mes vamos a hacer una introducción a éstos últimos, concretamente en su variedad lineal. Los fractales lineales, también llamados transformaciones afines, son el nexo de unión entre los fractales recursivos y los fractales no lineales. Son fractales de fórmula, pero producen resultados (en el límite) idénticos a los recursivos y viceversa, de ahí que constituyan un paso intermedio entre ambos extremos. La variedad lineal es, sin lugar a dudas, la más asequible y, al mismo tiempo, lo suficientemente potente como para servir de "entrenamiento" válido para estudios más avanzados tales como el conjunto de Mandelbrot y sus acólitos.

El porqué de su nombre es bien sencillo. En matemáticas, una variedad afín está constituida por un subespacio y un vector. En nuestro caso la transformación afín lleva el espacio total en un subespacio reducido, "girado" respecto del espacio canónico y desplazado según un vector. Así, las operaciones que puede realizar una transformación afín son una reducción de escala (imprescindible), un desplazamiento, un giro y, en algunos casos, una inversión. Debido a todo ello, una transformación afín presenta una propiedad bastante interesante: Si dos puntos equidistan de un tercero sus transformaciones equidistarán de la transformación del tercero también. Como corolarios se deduce que la transformación de una recta sigue siendo una recta, 2 rectas paralelas seguirán siendo paralelas, etc. Las distancias no se conservan, pero sí las relaciones primarias existentes en un principio. La transformación de un cuadrado puede nos ser otro cuadrado, pero la figura resultante también tendrá cuatro lados y sus 4 ángulos serán iguales a 90 , etc. Indudablemente, la transformación de una transformación sigue conservando estas propiedades aunque la figura se vuelva cada vez más irreconocible.

Probablemente una de las transformaciones afines más famosas y sencillas (y antiguas) fue definida por el matemático polaco WACLAW SIERPINSKI (1.882-1.969) en el año 1.916. Su nombre es "Triángulo de Sierpinski" por razones obvias, no hay más que ver la

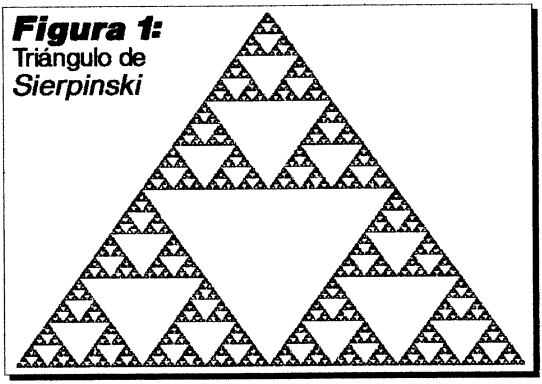


Figura 1: Triángulo de *Sierpinski*

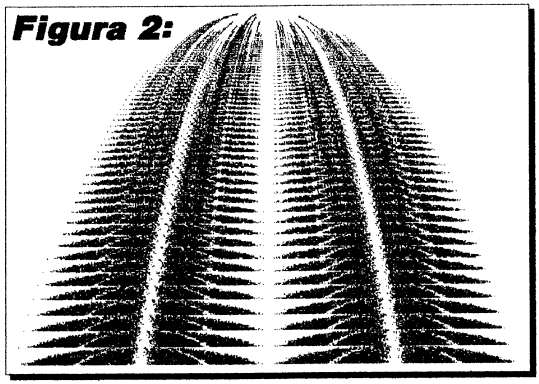
figura 1. El triángulo de Sierpinski se basa exclusivamente en una reducción y en un desplazamiento. La reducción consiste en reducir la figura original en 3 figuras de área 1/4. El desplazamiento consiste en colocar las 3 figuras obtenidas de tal forma que forman un triángulo isósceles. La figura límite, resultado de realizar unas cuantas transformaciones sucesivas, es el triángulo de Sierpinski.

Resulta un tanto paradójico que dicha imagen sea producto de iterar un cuadrilátero pero, experimentando un poco se comprueba que el resultado final es independiente de la figura original utilizada. Da lo mismo que usemos cuadrados, triángulos, círculos; el resultado es siempre el mismo (en el límite). La figura final sólo depende de las transformaciones empleadas y no de la posición inicial de los puntos a calcular. Ello es debido a la reducción de escala: La aplicación sucesiva de las transformaciones producen una "concentración" de los puntos originales, que en el límite convergerán a un sólo punto. Ni que decir tiene que esta otra propiedad simplifica enormemente los cálculos al poder utilizar siempre la misma figura inicial para todas las transformaciones afines que probemos. Generalmente usaremos el rectángulo de la pantalla.

¿Cual es la dimensión del triángulo de Sierpinski? Para que el área de cada triángulo sea 1/4, su lado debe reducirse a la mitad (debemos recordar que utilizamos como figura inicial el rectángulo de la pantalla). Dado que hay 3 triángulos y recordando lo explicado en el artículo anterior:

LOG3/LOG2=1.5849625. El área de la figura en el límite se calcula así: En cada iteración se resta $3n/4n+1$ de la figura iniciál. El área final será igual al área inicial menos el área que hayamos perdido en cada iteración. La suma de todas las áreas perdidas será:

Figura 2:



1/4(1-0.75)=1. El área final es 1-1=0. Es decir, el triángulo de Sierpinski, en el límite, "tiene área 0". Es una figura infinitamente tenue, formada por infinidad de puntos aislados que no encierran área alguna. Fantástico, ¿no?

La principal utilidad de este tipo de fractales es la reducción (compresión) de datos. Es mucho más rápida y sencilla la transmisión de una serie de "leyes" que describan una imagen que la transmisión de la imagen en sí misma. Además, dado que se trata de objetos fractales, nos está permitido ampliar cualquier zona que queramos, sin temer "pasarnos" de la resolución limitada de una imagen de televisión o de una fotografía. Otra cosa muy distinta es la velocidad, ya que no todos podemos permitirnos poseer un CRAY.

Desgraciadamente la codificación de una imagen en sus transformaciones afines equivalentes no es tarea sencilla. Aunque hay mucha gente trabajando en ello, no existe todavía un algoritmo general, a no ser para figuras muy sencillas en los que la codificación es casi trivial. De todos modos, parece que ya se han descubierto procedimientos bastante esperanzadores. Y mientras los investigadores se devanan los sesos intentando hallar la piedra filosofal, nosotros también podemos poner nuestro granito de arena jugando con

todo esto y, de paso, famirializarnos con sus inmensas posibilidades.

Veamos, por ejemplo, la construcción de nuestro triángulo de Sierpinski. Necesitamos tres transformaciones afines, una para cada subfigura. Dado que cada figura tiene 1/4 del área original, debemos multiplicar cada coordenada por 0.5 (reducir los lados a la mitad). Veamos ahora los desplaza-mientos: Si consideramos el origen en la esquina inferior izquierda, y la esquina opuesta como el punto (319,199), que es la resolución mínima de la mayoría de los sistemas informáticos disponibles "a nuestro nivel", una de las figuras tendrá desplazamiento (0,0), otra (160,0) y otra (80,100). Las tres transformaciones afines obtenidas finalmente son:

1. X=X*0.5; Y=Y*0.5
2. X=X*0.5+160; Y=Y*0.5
3. X=X*0.5+80; Y=Y*0.5+100

Con estas ecuaciones tenemos completamente definido nuestro fractal. Sólo queda dibujarlo. Para ello creamos un procedimiento que escoja aleatoriamente una de las transformaciones anteriores y se la aplique al punto actual. El nuevo punto así calculado debe ser reinsertado en la entrada de dicho procedimiento, y así sucesivamente creando un bucle infinito de realimentación. Al mismo tiempo cada nuevo punto será impreso en pantalla para deleite del programador, que verá pasar poco a poco de unos puntos dispersos aquí y allá a todo un señor fractal en cuestión de segundos. Bien, todo esto es muy bonito, pero, ¿Cómo crear nuestros propios fractales lineales? Tranquilos, convertirse en un Picasso informático no es tan difícil como parece.

Lo primero es la elección del "modelo" natural que nos gustaría imitar. Para empezar con algo sencillo, escojamos una cúpula, por ejemplo. El siguiente paso consiste en un análisis concienzudo para poder discernir inequivocamente aquellas propiedades que convierten una bóveda en lo que es. En nuestro caso, un objeto simétrico respecto al eje vertical (160,0-199), una base que ocupa la pantalla de parte a parte, una forma similar a una parábola con su vértice en (160,0), etc. Ahora debemos intentar plasmar estas características en un programa de ordenador. Empecemos por la base: Deseamos que ocupe totalmente la parte inferior de la pantalla, por lo que no le aplicamos reducción en el eje X. Como la base sólo ocupa la parte más baja de la pantalla asignamos una reducción en Y de, digamos, 20. De esta forma tenemos ya el estrecho rectángulo que será la base.

Ahora definamos la transformación

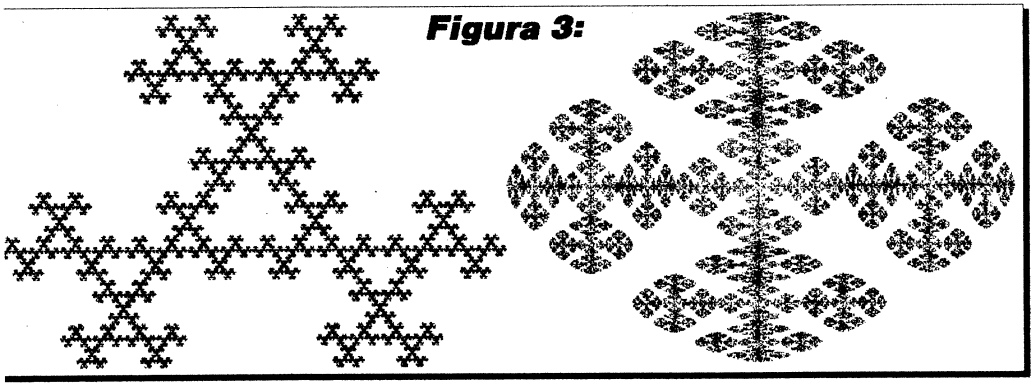


Figura 3:

correspondiente a la "altura" de la parábola: Es evidente que cuantos más "altos" se encuentren los puntos, más "concentrados" estarán en torno al eje de simetría. En otras palabras, que a mayor altura, menor "anchura". Debido a esto en este caso sí habrá reducción de escala en X. El valor concreto de dicha reducción depende de la forma final que queramos obtener. Un detalle a tener en cuenta es que si la figura es simétrica respecto al eje (160,0)-(160,199), la transformación TAMBIEN DEBE SERLO. En cuanto al eje Y, también debemos reducirlo para que el dibujo no se nos salga de la pantalla. Las transformaciones equivalentes a todo este rollo podrían muy bien ser:

1. X=X; Y=Y/20
2. X=X*49/50+3.2; Y=10+Y*19/20

Si introducimos estas fórmulas en nuestro programa comprobaremos en seguida que no funcionan. ¿Acaso nos hemos equivocado? En realidad no. Lo que ocurre es que entre nuestras fórmulas no existe ninguna que lleve la coordenada X hacia la esquina inferior derecha. Para obtener este efecto no tenemos más que descomponer una de nuestras transformaciones en dos, tal que una lo haga. De esta manera las tres trasformaciones obtenidas son:

1. X=X/2; Y=Y/20
2. X=X/2+160; Y=Y/20
3. X=X*49/50+3.2;Y=Y*19/20+10

En general, se precisan un *mínimo* de ***TRUNC(D)+2*** transformaciones afines para describir cualquier objeto fractal siempre, por simple que sea. (D es la dimensión)

Si ahora reintroducimos nuestro nuevo modelo matemático obtenemos "casi" la figura buscada. Sin embargo, podemos ver que los puntos se agrupan principalmente en la base de la cúpula, mientras que en su vértice apenas aparece algún que otro píxel despistado. Si este es el caso me imagino que en vuestro programa las tres transformaciones tienen la misma posibilidad de salir. Evidentemente ello produce que los puntos parezcan concentrarse en la base, dado que se imprimen estadisticamente el mismo número de puntos que en el cuerpo de la figura, pero parecen muchos más porque el área es mucho menor, por lo que la "densidad media" en más alta. La solución más lógica consiste en que cada transformación tenga una posibilidad de salir proporcional al área que posea. Si una transformación tiene un área doble que otra, debiera tener también el doble de posibilidades de salir. De esta forma se consigue la distribución uniforme generalmente buscada (figura 2). De todos modos, un recurso estético muy importante

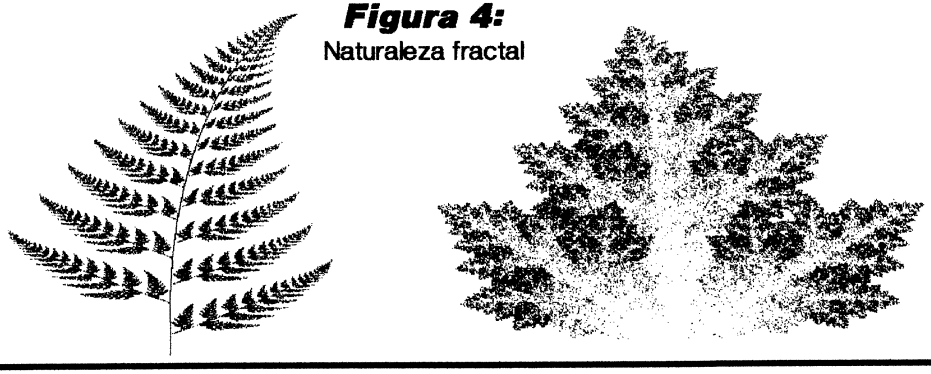


Figura 4:
Naturaleza fractal

consiste en variar artificialmente las proporciones sin obedecer a la "ley de las áreas". Los resultados pueden ser muy espectaculares, sobre todo con diseños complicados. El aspecto de una misma figura puede variar totalmente según las distintas posibilidades asignadas a cada trasformación.

¿Cual es la dimensión fractal de nuestra parábola? La verdad es que calcular la dimensión de un objeto cuyas transformaciones no son todas iguales, cuando existen giros y, sobre todo, cuando hay superposición de transformaciones es bastante complicado, por no decir imposible (al menos utilizando el sistema "tradicional"). En estos casos es preciso recurrir a las propiedades de "densidad fractal" del objeto.

El número de puntos "distinguibles" a un nivel de ampliación dado se calcula utilizando:

$S=C*R^D$, donde S es el número de puntos visibles, C es una constante propia de cada cuerpo, R es el radio del cuerpo en el que se inscribe nuestra figura y D es la dimensión del objeto. Aplicando logaritmos y teniendo en cuenta que en el caso de un cuadrado la constante C vale la unidad, queda:

$D=\log(S)/\log(R)$.

La forma de calcular la dimensión de una figura fractal consiste en inscribirla en el menor cuadrado posible, de radio R. Cuando se finaliza el dibujo basta contar el número de puntos impresos S, teniendo muy en cuenta de que los que se imprimieron encima de puntos ya calculados no se consideran. Luego se aplica la fórmula anterior y la dimensión de nuestra imagen fractal nos sale directamente.

En realidad las cosas no son tan fáciles. El hecho de tener una unidad de medida "discreta", el punto, implica grandes imprecisiones. La manera de mejorar el resultado consiste en reducir la escala o lo que es lo mismo, hacer los puntos "más pequeños". Para ello se aumenta el tamaño del cuadrado que circunscribe a nuestra figura. De todos modos, para obtener una precisión absoluta se requiere un cuadrado de lado infinito, algo bastante difícil de obtener. Debido a esto resulta evidente la necesidad de trabajar con el mayor cuadrado que nos permita la memoria de nuestro ordenador y aún así debemos tomar los resultados con ciertas reservas. Como ejemplo decir que calculando la dimensión del triángulo de Sierpinsky en una matriz de 2048*2048, matriz que ocupa unos 524Kbytes de memoria, sale un error por exceso de unas 3 centésimas...

Como podéis ver con los dibujos que acompañan al artículo, es posible definir fractales lineales muy bellos con sólo 4 transformaciones afines (figura 4), así que imaginaros lo que es posible crear con más. Por otro lado, siento no tener espacio para poder explicaros como se construyen los giros y las inversiones, así como la forma de diseñar transformaciones según modelos mucho más complicados -y más interesantes- que una vulgar parábola. De todos modos son técnicas muy sencillas que podréis descubrir por vosotros mismos si os pica también el gusanillo.

En el próximo número hablaremos del ***CONJUNTO DE MANDELBROT****, alias "El objeto más complicado de las matemáticas" (y con toda la razón del mundo). De esta forma ya podremos contar con una visión global de los diferentes tipos de fractales existentes. Existe una inmensa variedad de sistemas que si bien no tienen comportamiento fractal producen resultados fractaloides, como es el caso de los sistemas de difusión limitada, el efecto de percolación, etc. No vamos a estudiarlos sistematicamente pero es probable que salga un artículo sobre el tema en números próximos. De momento, tenemos proyectos más inmediatos.*

APLICACIONES PRACTICAS

Simón Gómez

El C.A.D. (I)

El CAD es una herramienta de trabajo relativamente moderna pensada para facilitar notablemente el trabajo a ingenieros, arquitectos, etc en sus proyectos y diseños. El acrónimo CAD viene del inglés Computer Aided Design (Diseño asistido por ordenador).

Podemos decir que el CAD nace ya con el primer transistor en 1947. Más tarde, el desarrollo de las pantallas gráficas y de los lápices ópticos le ayudan a su expansión y crecimiento. A mediados de los años cincuenta el sistema de defensa aerea americano fue el primero en usar y controlar consolas en las que los operadores identificaban los objetivos señalándolos con lápices ópticos.

Todas las tecnologías que usamos actualmente ya existían en 1965; los lápices ópticos se inventaron en 1958 y las mesas digitalizadoras estuvieron disponibles en 1964, pero alto coste de producción y, principalmente, la escasísima disponibilidad de un software adecuado fueron los responsables de su escasa difusion.

En 1962, Ivan Sutherland demostró en su tesis doctoral que los gráficos interactivos por ordenador eran factibles. De este trabajo se hizo una película sobre gráficos por ordenador que, gracias a una amplia difusión de la misma, generó un gran interés por el tema. Sutherland introdujo y profundizó en muchas de las técnicas que todavía se siguen utilizando hoy.

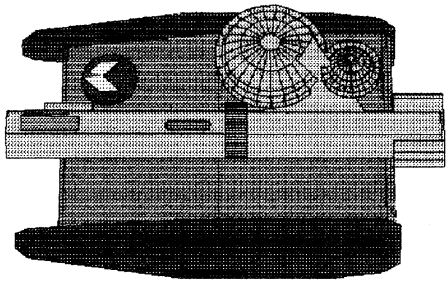
Por esa época, la General Motors y los laboratorios Lincoln demostraron por separado, aunque llegaron a la misma conclusión, que un tubo de rayos catódicos podía utilizarse como pantalla gráfica. En 1965 la General Motors, los Laboratorios Bell, Lockheed y la Macdonnell Douglas investigaban sobre gráficos con grandes ordenadores.

A principios de los sesenta, la General Motors inventa el DAC (Diseño Potenciado por Ordenador). Fue un auténtico sistema CAD, ya que los usuarios no tenían porque tener conocimientos de programación y, por primera vez, los diseñadores y toda aquella persona con un mínimo de conocimientos podía comunicarse con un ordenador graficamente. El DAC fue en un principio desarrollado para el IBM 7090, pero cuando empezó a funcionar correctamente, IBM saca la serie 360, incompatible con el programa del DAC. Sin embargo, gracias al DAC la General Motors pudo diseñar y producir importantes piezas cuando en 1964-65 cambiaron sus modelos de automóviles en el último momento.

Pero fue probablemente en 1973 cuando se produjo un salto cuántico en el CAD gracias al Programa de Análisis Estructural de Vehículos (VSAP) de la General Motors, permitiendo que la compañia ahorrara nueve meses en el diseño de los nuevos coches, la mayor parte gracias al diseño de prototipos mediante simulación. La fabricación se incorporó mucho más tarde a estas nuevas tecnologías, ya que fue preciso renovar toda la maquinaria de producción existente. Otro ejemplo de fabricación controlada por ordenador fue cuando en 1977 la compañía Minister Machine incrementó su producción en un 30% al acoplar un sistema de

CAD a su máquina de corte de piezas.

El CAD permitió (y permite) a las empresas obtener más beneficios y reducir el coste de fabricación y, por lo tanto, abaratar el producto final haciéndolo más competitivo, pues el CAD acorta el tiempo de producción, pudiendo hacer más piezas en menos tiempo.



CARACTERISTICAS DEL CAD

Si pudiéramos extender el concepto de "fábrica automatizada" hasta las últimas consecuencias, todo el proceso de fabricación estaría bajo el control del ordenador. Aún cuando los sistemas de CAD son distintos, tienen algunos denominadores comunes, unas características generales para todos ellos:

-Estación gráfica de ordenador

-Base de datos

-Control numérico de la máquina

-Manejo automático de los materiales

-Robótica

En este primer artículo vamos a tratar de explicar lo que es una estación gráfica y todo su entorno físico, funciones, etc.

ESTACION GRAFICA

Basicamente, una estación gráfica es una combinación de Hardware y Software (circuitos y programas) que proporcionan al usuario lo que necesita para interactuar con el ordenador y crear información gráfica (incluyendo entradas gráficas, visualización y procesamiento de gráficos). La mayor parte de estas estaciones han sido pensadas para trabajar conjuntamente con un macroordenador. Hay muchos elementos de entrada y salida que completan la estación gráfica: teclados, digitalizadores, plotters, etc. Para cumplir con las exigencias que demandan las aplicaciones de CAD las estaciones gráficas tienen que cumplir unos requisitos determinados, tales como pantallas de alta resolución que soporten resoluciones en pantalla que van desde 1024 por 1024 hasta 4096 por 4096, deberán ser capaces de manejar símbolos y formas en pantalla en tiempo real, representación de imagenes en tres dimensiones, etc. En las pantallas de tubos catódicos, se suelen emplear de tres tipos distintos de tecnología:

-Tubos de traza: Estos sistemas proporcionan enormes ventajas con su velocidad de respuesta para rápida actualización de la imagen y se acoplan perfectamente a los rápidos canales de entrada/salida. En contra tienen una capacidad limitada de reproducción de color y tienden a producir vibraciones cuando las figuras empiezan a ser cada vez más complicadas.

-Tubos de memoria: Su principal ventaja está en la ausencia de parpadeo. Una vez plasmada la imagen, permanece en la pantalla de fósforo sin necesidad de actualizarla. Sus inconvenientes más destacados están la baja velocidad de cambio de imagen y su limitada capacidad de color.

-Tubos de barrido: El barrido de pantalla resuelve los problemas anteriormente planteados por los otros dos tipos de pantalla. La imagen puede ser actualizada con velocidades constantes, mejorando así la calidad con

respecto a la televisión normal ofreciendo, además, una capacidad total de reproducción en color.

TECNOLOGIA DE UNA ESTACION GRAFICA:

Tres son los elementos principales de una estación gráfica: el procesador central(interno o externo), el controlador de pantalla y la pantalla. La capacidad de memoria de una estación nos viene dada por la fórmula $CM=2^N$, donde CM es la capacidad de memoria y N es el número de bits o palabras que puede direccionar el procesador. El tamaño de memoria es muy importante, ya que en todo momento debe contener las instrucciones del programa que se está ejecutando, aunque el programa se divida en varias partes y se vaya accediendo a ellas por medio de disco, disco duro, etc. El controlador de pantalla es el que permite la comunicación entre el ordenador y la pantalla. El controlador interpreta la información generada por el programa, la divide y la va enviando a la pantalla. A menudo este controlador también se encarga de llevar un control de los periféricos.

INTELIGENCIA Y FACTORES DE COSTE DE UNA ESTACION GRAFICA

Hay tres posibles niveles de configuración de una estación gráfica de CAD:

-Baja complejidad: Precisa de un ordenador central para poder trabajar con ella. Puede hacerse trabajo en dos dimensiones con ayuda de dispositivos complementarios.

-Media complejidad: Permite trabajar con ella durante largos periodos de tiempo sin la necesidad del soporte del ordenador central. Tiene posibilidad de soportar periféricos gráficos locales, como digitalizadores, etc. Suelen poseer poca memoria y tienen la posibilidad de trabajar con sólidos y otros gráficos en tres dimensiones.

-Alta complejidad: No precisa en absoluto de un ordenador central, pudiendo conectarse a una gran variedad de periféricos y dispositivos gráficos. Este tipo de sistemas dependen del software que lo acompaña. Se suelen emplear para dibujo de alta precisión.

DISPOSITIVOS DE ENTRADA

Definimos "Entrada" como el método de llevar información al ordenador. Existen numerosos dispositivos de entrada siendo el más universal de todos el teclado que llevan casi todos los sistemas informáticos. Existen también otros dispositivos que ayudan a que el operador haga su trabajo lo más comoda y rapidamente posible, como lápices ópticos, tabletas digitalizadoras, etc. Un sistema CAD tiene que estar preparado para recibir datos de distintos tipos y formatos. La información puede venir de fuentes tan diversas como de las ideas del propio programador o información acerca de objetos de otros sistemas CAD. Analizemos un poco más estos dispositivos:

-TECLADO: Es similar a una máquina de escribir pero emplea una pantalla en vez de papel. No todos los teclados son iguales ya que pueden tener teclas especiales, de función, para asignarles un cometido dentro del programa. También pueden tener distinta disposición del alfabeto; se denominan por las seis primeras letras de la parte superior izquierda. Así: el QWERTY, AZERTY, QWERTZ, etc.

-PANEL DE FUNCIONES PROGRAMABLES: Frecuentemente se suele emplear, junto con el teclado, un panel con funciones programables. Suelen tener entre dieciséis y treinta y dos teclas de función, generando cada una un código particular. Funcionan como

"macros" con las instrucciones más frecuentes y así no tener que acceder tanto al teclado, con el consiguiente ahorro de tiempo. Se suelen iluminar, con lo que el operador sabe lo que se está haciendo.

-TABLETA GRAFICA: Detectan la posición de un lápiz o un cursor sobre la superficie de digitalización. La tableta posee un entramado electromagnético con la misma resolución que la pantalla. El operario va dibujando sobre la tableta y aparece en la pantalla lo que está trazando.

-CURSORES: Pueden ser de uno de los tres tipos siguientes: Ratón, cursor o puntero. Los cursores digitalizadores tienen una mira desplazable sobre unos ejes rectangulares móviles y proporcionan el valor absoluto de las coordenadas x,y. Pueden disponer de varios botones para comunicarse con el programa. El lápiz es similar a un bolígrafo con un pulsador en la punta y sirven para dibujar a mano alzada y señalizar menús. El ratón sirve para obtener valores incrementales de las coordenadas y se pueden modificar para que funcionen como digitalizadores. Los cursores pueden ser restringidos o libres. Los primeros están fijados a un brazo mecánico y son usados para trabajos de alta precisión, como cartografía. Los segundos están unidos a través de un cable a la tableta.

-LAPIZ OPTICO: Sirve para comunicarse directamente con la pantalla. Sirve para marcar un punto, un carácter o un vector en la pantalla y decidir qué hacer con él a través de la selección de menús.

-SCANNERS: Convierten el contenido de una hoja de papel en información que pueda ser entendida por el ordenador y lo mete en pantalla para su posterior modificación.

DISPOSITIVOS DE SALIDA

Los dos dispositivos más comunes de salida son la impresora y el plotter. Cada uno tiene sus pros y sus contras que ahora vamos a analizar:

-IMPRESORA: No vamos a profundizar aquí sobre ellas porque ya hay un artículo sobre este tema en este mismo número.

-PLOTTER: Es el dispositivo de salida más usado en áreas de diseño asistido por ordenador. Al principio se usaban más bien poco ya que eran muy complejos y caros, pero a medida que se van sofisticando en sus posibilidades y simplificando en su manejo, más y más profesionales se pasan a él. Sus modelos van desde los más simples capaces de realizar simples dibujos lineales hasta los más complejos capaces de realizar planos, diseños y gráficos complicados. Frente a las impresoras, los plotters ofrecen una resolución mucho mayor y con colores. Son capaces de hacer reproducciones y ampliaciones de la misma calidad o detalle que el original y pueden trabajar en una gran diversidad de tamaños y calidades de papel, incluso en impresión fotográfica en "offset". La desventaja es que son caros aunque cada vez se van abaratando más. Los tipos de plotter usuales son:

-PLANO: Es el más sencillo. Se fija el papel sobre una superficie lisa y las plumillas se mueven sobre los ejes X-Y para dibujar la imagen. Se puede imprimir sobre casi todas las clases de papel.

-DE TAMBOR: La plumilla se mueve en horizontal y el papel se mueve en un tambor para conseguir la dimensión vertical. No tienen limitación en cuanto al tamaño del papel.

-DE RODILLO: Es un híbrido entre los dos anteriores. Posee un motor que desplaza el papel sobre el eje X y la plumilla se desplaza sobre el Y.