UNIVERSIDAD NACIONAL AUTONOMA
DE MEXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES
CAMPUS ACATLAN

FORMATOS GRAFICOS, DE SONIDO Y VIDEO

T E S I N A

QUE PARA OBTENER EL TITULO DE
LICENCIADO EN MATEMATICAS APLICADAS Y COMPUTACION
P R E S E N T A
ALEJANDRO ROBERTO RUBIO PEREZ

EDO DE MEXICO 1997

VNIVERSIDAD NACIONAL
AVENOMA DE
MEXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES "ACATLÁN"

COORDINACIÓN DEL PROGRAMA DE MATEMÁTICAS
APLICADAS Y COMPUTACIÓN

SR. ALEJANDRO ROBERTO RUBIO PÉREZ
Alumno de la carrera de Matemáticas Aplicadas y Computación
P r e s e n t e.

De acuerdo a su solicitud presentada con fecha 15 de febrero de 1996. me complace informarle que esta Coordinación tuvo a bien asignarle el siguiente tema de tesina: "Formatos gráficos. de sonido y video". el cual desarrollará como sigue:

Introducción
I. Formatos gráficos
II. Formatos de sonido
III. Formatos de video
Conclusiones

Asimismo fué designado como Asesor de la Tesina el Lic. Juan Carlos Rendón Aguilar, profesor de esta Escuela.

Ruego a usted tomar nota que en cumplimiento de lo especificado en la Ley de Profesiones. deberán prestar servicio social durante un tiempo mínimo de seis meses como requisito básico para sustentar el examen profesional. así como de la disposición de la Coordinación de la Administración Escolar en el sentido de que se imprime en lugar visible de los ejemplares de la tesina el título del trabajo realizado. Esta comunicación deberá imprimirse en el interior de la tesina.

A t e n t a m e n t e
"POR MI RAZA HABLARÁ EL ESPÍRITU"
Acatlán, Edo. de Méx., a 28 de abril de 1997.

LIC. BEATRIZ TRUEBA RIOS
Jefe del Programa de M.A.C.

A Dios
Por la oportunidad de vivir.

A mi Madre y a mi Padre
por darme su apoyo siempre desinteresado y comprensión

por darme la vida.

A mi hermano
Por su cariño, apoyo y amistad.

A mi asesor y sinodales
Por sus comentarios, criticas y sugerencias para la elaboración de este trabajo.

Al Personal del Centro de Cómputo y del D.S.C.
Por su apoyo desinteresado.

Introducción

Los formatos gráficos, de sonido y video, han resuelto las necesidades de los usuarios de computadoras y han generado a su vez nuevos retos, además las tendencias actuales que están modificando la visión de la computación, usan los formatos como su piedra angular y su impacto es tal, que prometen modificar la vida cotidiana.

Las ventajas de las *imágenes* en la computación son evidentes: La generación de animaciones, creación de planos para construcciones (*CAD*) y herramientas (*CAM*), composiciones para publicaciones, simulaciones matemáticas o físicas difíciles de comprender sin una ayuda gráfica, juegos de video y hasta como arte formal, han dado una nueva dimensión del uso de las computadoras.

Para hacer uso de tales beneficios se están utilizando formatos gráficos, los cuales permiten manipular y almacenar imágenes modificando la relación computadora - usuario; máquinas como la *Macintosh* de Apple, iniciaron el concepto de un entorno visual agradable al usuario. Pero no fue sino hasta el advenimiento de *Windows* de *Microsoft*, cuando la generalidad de los usuarios entraron de lleno a un ambiente operativo basado en imágenes; ahora, los nuevos sistemas operativos están usando los gráficos de manera cotidiana, las compañías especializadas en bases de datos como *Oracle, Sybase, Informix/Ilustra, etc.*, usan las imágenes como un elemento más de información y las soportan en sus productos, el *World Wide Web* usa cada vez más las imágenes en sus páginas, e inclusive actualmente se busca que sean vistas a tiempo real.

El concepto de *audio* nació casi a la par de la imagen pero su desarrollo ha sido muy lento, esto es debido en parte a que la mayoría de los equipos no eran lo suficientemente rápidos, además de que el *hardware* necesario para reproducir sonido es todavía considerado un lujo y la gran mayoría de los equipos vendidos no cuentan con él. La mayoría de los formatos de sonido están fuertemente orientados a una plataforma específica. Actualmente con el concepto de multimedia que engloba a las imágenes, sonidos, video, texto, la telefonía digital, además del abaratamiento del equipo de cómputo y las aplicaciones más intuitivas, estimulan a las empresas y los desarrolladores a generar formatos de sonido más generales, bajo estándares abiertos.

En el caso del *video*, que es el más reciente de los tipos de formatos, es todavía muy inmaduro - aún más que los formatos de sonido -, ya que el hardware incluido en la configuración básica no es lo suficientemente rápido para lograr una velocidad aceptable de desplegado, además es muy costoso el hardware especial para tal fin (cámaras de video, micrófonos, tarjetas de captura y conversión, entre otros), de hecho casi todos los formatos que están en existencia se basan muy fuertemente en un sistema determinado, del cual obtienen su máxima eficiencia. Apenas a finales de 1995 y principios de 1996 es cuando se ponen a la venta las primeras cámaras a bajo costo, principalmente para realizar videoconferencias; de hecho, el potencial de demanda del video hace que se modifique el mercado rápidamente.

Una cosa que se debe observar en cualquier tipo de formato es la extensión[1] que tiene cada formato, la cual es única y distintiva, su observancia es primordial debido a la existencia de muchos formatos de estructura similar.

[1] Se refiere por un lado a la cadena de tres caracteres complementarios al nombre utilizada por el sistema operativo MSDOS. Aunque también se refiere a las adiciones hechas a un formato o estándar.

Objetivo:

El objetivo de este trabajo es el de recopilar y dar a conocer los formatos más comunes de sonido, video, y gráficos; sus estructuras básicas y su orientación, los cuales son de gran importancia en la computación actual, para facilitar el desarrollo de aplicaciones y dar una referencia en español de estos formatos, los cuales son muy necesarios en las aplicaciones actuales.

Breve historia sobre los formatos de sonido, imágenes y video en las computadoras actuales

La computadora como herramienta ha mostrado que puede ser usada en una enorme variedad de aplicaciones, y ha sido hasta hoy, la más importante y flexible de todas las que posee el hombre.

En el inicio de la computación, no se concebía el hecho de que una computadora pudiese almacenar, manipular y reproducir un sonido, imagen o video, tal concepto fue cuestionado al empezar a usarse pantallas de *rayos catódicos* (monitores o *VTR*) como medio de recibir los resultados que producía la computadora; el primer monitor usado no estaba diseñado para desplegar gráficos, únicamente texto, sin embargo, el interés en usar gráficas queda de manifiesto desde ese momento en el arte ascii, consistente en ordenar una secuencia de caracteres de tal modo que se logra hacer un dibujo; de ahí es hasta las primeras PC (la original *Personal Computer* comenzó en 1976 con la Apple de Steve Jobs), cuando ha madurado la tecnología lo suficiente, para presentar gráficas a baja resolución y en dos colores, mientras que para las IBM PC el primer monitor en soportar gráficas fue el *Hercules*(de Hercules Technologies).

El primer formato gráfico de uso general en surgir fue el formato *PCX* de Zsoft en 1983, la primer implementación de este formato soportaba Blanco y Negro, 16 tonos de gris y 16 colores reales, en adelante las imágenes fueron ganado popularidad para el usuario común conforme fueron apareciendo tipos de monitores más refinados -*CGA, EGA, VGA, XGA*- para las IBM PC compatibles, y las *RGB* por parte de los sistemas Unix y Macintosh. Tal mejora constante en la calidad de los monitores obligó a los desarrolladores de software a crear formatos gráficos que hiciesen uso de tales ventajas.

A partir de este momento se inicia el mejoramiento substancial en la capacidad y en la calidad de almacenamiento de los formatos, esto se vio alentado con la introducción de los scanners, y el surgimiento del formato *TGA* -inicialmente concebido para almacenar imágenes capturadas de monitores, luego ajustadas para scanners-, estos formatos al principio eran exclusivos de una aplicación.

El impulso que permitió el licenciar (es decir, permitir su explotación por terceros) los formatos se debió a la existencia de programas y software de captura de imágenes (tal es el caso del Targa). Otro acontecimiento importante para los formatos gráficos ha sido el *Internet*, ya que muchas computadoras de distintas arquitecturas y capacidades pudieron compartir información (entre ello *imágenes*), por ende ahora la búsqueda se concentra en obtener formatos que sean compatibles con casi todos los equipos y que tengan un rendimiento aceptable en computadoras de bajo poder; sin lugar a dudas los formatos gráficos están impactando rápidamente la visión de muchas actividades cotidianas.

Con las mejores técnicas de captura y en un intento de estandarizar los datos, Organizaciones tales como la Junta de Expertos en Fotografía dependiente del ISO (*Organización de Estándares Internacionales*) crean un formato especialmente diseñado para imágenes de calidad fotográfica - el JPEG -, otros grupos como la CCITT (*Comité Consultivo de Telefonía y Telegrafía Internacional*), y dependencias

Norteamericanas Tales como la Marina, la Fuerza Área y el Ejército organizaron el formato CGM el cual es base para nuevos formatos tales como CALS, HOMCALS; actualmente hay una tendencia a fortalecer lenguajes como medio para estructurar formatos gráficos y se puede tener la seguridad que se verán formatos estándares para todas las plataformas.

Para los formatos de sonido, la tecnología que llevaría a su existencia, se inicia en los 70's gracias a los trabajos de varias compañías, desarrollándose la idea de la bobina electromagnética que fuese controlada por un procesador, y de esta manera poder reproducir sonidos previamente almacenados obteniéndose música.

En este desarrollo muchas empresas dedicadas a la computación y a la música se vieron involucradas, tales como Yamaha, IBM, Apple, Commmodore, etc., el uso inicial de esto fue enfocado a los juegos de video e instrumentos musicales electrónicos.

Para mediados de los 80, La Commodore junto con otras compañias y grupos organiza el primer formato de sonido: el MODule Player, formato que se origina en la familia Amiga, pudiendo almacenar hasta 4 canales de sonido con 64 instrumentos; existen varios subtipos que son soportados por distintas organizaciones y varios clones que son simples modificaciones en la cabecera, estructura, datos, etc.

A partir de 1989, las tarjetas dedicadas a sintetizar, almacenar y producir sonido para la familia de las PC's están a la venta, se organizan varios formatos para almacenar el sonido, pero todos eran especiales a cierta tarjeta e incompatibles con otras, hasta que Microsoft, lanza en 1991, Windows 3.0, y poco después, la extensión multimedia del MS-Windows (3.1), da a conocer su formato propietario: el formato WAV, contenido en el estándar MS-RIFF. De ahí, en la medida que ha soportado sonido de alta calidad y voz, motivó a otros sistemas a desarrollar formatos, tales como el AU de la SUN/NeXT Graphics derivado del estándar μ-Law., el AIFF-C de Apple, etc. Actualmente están apareciendo aplicaciones que soporten los formatos en todas las máquinas.

Para el video, la historia se inicia en 1984 con el formato FLI/FLC, (de Autodesk Inc., actualmente obsoleto para el video) donde retomando el clásico sistema cinematográfico, en cual la imagen se compone de varios cuadros (frames) en los cuales la imagen, se va modificando dando la impresión de movimiento, pero no es realmente un formato de video ya que no reproduce sonido, elemento indispensable en el video digital.

Es hasta 1991 cuando se puede transformar fácilmente la señal analógica de una cámara de video a un formato de video digital; por esas fechas son creados dos formatos que aún hoy están luchando por ser aceptados como estándares: El AVI (propuesto por *Microsoft e Intel*) , y el QuickTime (propuesto por *Apple*).

Además, cabe mencionar los esfuerzos por una estandarización de las señales de video electrónicas (que incluyen las de televisión) por parte de la CCITT, CIE (*Commission International de l'Eclairage*, Comisión Internacional de Iluminación), ISO, ANSI (*American National Standards Institute*, Instituto de Estándares Nacionales Americanos) entre otros, que han organizado el estándar *MPEG*, el cual con sus variantes abarca casi todos los métodos de transmisión digital de video, cabe

mencionar que no ha sido muy bien aceptado. Recientemente los ejércitos de los Estados Unidos, Australia y la OTAN están analizando un formato: El HOMCALS (derivado del formato *CALS*) que se espera sirva para almacenar video con un mejor factor de compresión, y una imagen más grande a mayor velocidad.

El desarrollo de los tres tipos de formatos guardan un esquema general: Al inicio se desarrollan formatos exclusivos para una determinada aplicación, de ahí, para algunos formatos surgen aplicaciones que los soportan, o son propuestos por compañías, organizaciones e individuos quedando el código del formato en dominio público. Recientemente la ISO y la ANSI han desarrollado propuestas de formatos estándares.

1. Gráficos

Los formatos gráficos son uno de los pilares de nuevos estándares, sistemas operativos, aplicaciones multimedia, bases de datos, etc., y es porque actualmente con la constante inclusión de nuevos usuarios que requieren un entorno más simple y más fácil de usar, han despertado un vivo interés por entrar a ambientes gráficos (*Ms Windows x, MAC OS, IRIX, Solaris*, etc.), donde los procesos y sus resultados son desplegados mediante atractivas imágenes. En vista de esto, los desarrolladores de software están creando formatos gráficos que sean más eficientes tanto para almacenar como para desplegar (*renderizar*) imágenes.

En el presente capítulo veremos los elementos básicos de una imagen, los tipos de imágenes que existen, las formas de estructurarlas y codificarlas; analizaremos algunos de los formatos más representativos, además veremos brevemente el porque son comprimidas y que compresores se usan más comúnmente con los formatos gráficos.

A quien desee estudiar y profundizar en el tema, es recomendable cuando menos, poseer conocimientos en estructura de datos, además de cierta experiencia en programación tanto para implementarlos, como para entender la organización de algunos formatos que están basados en lenguajes clásicos como el FORTRAN (FITS), C (XBM y XDW), FORTH (PostScrip), Pascal (GIF) y de algunos novedosos como el VRML.

Existen algoritmos que si bien no afectan la calidad de una imagen almacenada, si hacen posible su fácil almacenamiento y transportación; me refiero a los algoritmos de compresión, varios de los cuales hacen uso de la probabilidad y la estadística, que más adelante analizaremos.

Para los estudiantes de la carrera de Matemáticas Aplicadas y Computación, no debe ser difícil el comprender estos conocimientos, los cuales son una aplicación de sus cursos de programación, estructura de datos, métodos numéricos, probabilidad, estadística y álgebra lineal. El desconocimiento de algunos aspectos del tema no debe ser obstáculo para utilizar los formatos gráficos, ya que existen programas que se especializan en manipularlos (incluso convirtiendo entre formatos), y librerías que pueden ser adicionadas a los programas sin afectar los derechos de autor, ni incurrir en alguna infracción legal.

1.1 Teoría y nociones básicas

Pero, *¿Qué es un formato gráfico?*, en el sentido estricto del término, "Es aquel conjunto de reglas para el almacenamiento, mapulación y desplegado de una *imagen* " tales reglas son organizadas, dadas a conocer y revisadas por el "propietario" del formato. Ahora bien, el propietario puede ser alguna asociación, grupo o individuo, el cual otorga licencias para crear programas que usen el formato, es común que los formatos sean de dominio público, a veces los propietarios distribuyen el programa

codificador - decodificador como una librería ejecutable, la cual el desarrollador del software añade a su programa.

Un gráfico puede abarcar diversos tipos de imágenes que van desde dibujos simples como son los esquemas hasta fotografías de muy alta calidad, asimismo, el número de colores va desde el blanco y negro puro hasta millones de colores (matices); puede la imagen componerse por elementos claramente geométricos o por formas irregulares y difuminadas.

Sabemos también que una imagen se compone de una unión de "puntos". Estos puntos pueden ser haces luminosos en un medio electrónico, siendo el caso de los monitores y televisores, o bien, marcas de tinta minúsculas en papel, acetatos u otro medio físico; a estos puntos se les conoce como pixels, la cantidad de pixels que puedan ser renderizados (desplegados) en un área dada, marcara la calidad del medio., por ejemplo, un monitor VGA típico cuya calidad es aún aceptable contiene un arreglo de 640 x 480 pixels, es decir, agrupa a 307200 pixels en la pantalla, mientras que en un medio físico, digamos, una impresora láser, puede colocar en el área de una pulgada cuadrada de papel al menos 300 pixels, habiendo algunas que colocan 1500 pixels en esa misma área.

Un pixel se puede representar en un campo coordenado, donde los valores X y Y indican su posición., pero es necesario indicar que es necesario incluir un valor Z que correspondería al color que contiene el pixel, este valor se le conoce también como "profundidad".

Debido a que hay muchos factores que se combinan en una imagen, los cuales tienen amplios rangos de acción, no hay un método único que codifique a todos los tipos de imágenes posibles, por ello se usan varios métodos de codificación, además para optimizar el espacio físico necesario para almacenarlos se usan algoritmos de compresión, lo cual veremos a continuación.

1.2 Métodos de compresión

Los gráficos al ser convertidos en datos numéricos, son codificados, pero ello no es suficiente, ya que la cantidad de datos de una imagen típica es grande, sin contar los datos de control necesarios para el correcto desplegado de la información que siempre van en un formato; por ejemplo, una imagen que se componga de un arreglo bidimensional ***640 x 480 pixels***, es decir, 307200 pixels y cada pixel teniendo valiendo ***1 bit de profundidad*** (2 colores solamente) ocupa 37.5 Kb de espacio, Ese mismo archivo al ser comprimido usando un método de compresión como el LZW ocuparía solamente 4Kb., en el caso aplicaciones que demandan gran número de imágenes de alta definición con muchos colores como ocurre en la edición de publicaciones, la diferencia se puede medir en decenas en MegaBytes. La cantidad de métodos de compresión es muy grande aunque muchos son combinaciones o adaptaciones de los siguientes métodos vistos aquí.

1.2.1 Codificación Binaria y Simbólica

Un Gráfico puede ser codificado de tres maneras: Simbólica, Binaria, o bien, una unión de los dos anteriores. La codificación simbólica usa el código ASCII para representar una imagen, por medio de comandos que representan una estructura de pixels, en este caso es posible ver el código (y editarlo) con un procesador de textos, tal es el caso del formato PostScrip y el VRML. El siguiente tipo de codificación es la binaria, la cual se vale del uso del código binario para representar un conjunto de pixels.

En el caso de la Codificación Binaria, es ineludible comentar que a diferencia de la Codificación Simbólica, es necesario tomar en cuenta el tipo de procesador en que fue manipulada la imagen, ya que las computadoras usan un distinto *"orden de los bytes"* , esto significa que un pixel en una imagen se compone por valores de bits (profundidad de la imagen), estos dependiendo del procesador son almacenados en forma diferente, así, un pixel de 16 bits es almacenado en 2 bytes, - suponiendo que este valga 512 en decimal -, tenemos que un procesador lo almacena como 00000001 00000000 (01 00 hexadecimal), otro lo almacena como 00000000 00000001 (00 01 hex.)

Hay dos criterios usados para almacenar y leer los bytes, estos son el criterio del byte menos significativo (*Least-Significant-Byte*) y el criterio del byte más significativo (*Most-Significant-Byte*) - LSB y MSB en inglés -, por ejemplo, mientras LSB representa al número 512 como 01 00 hexadecimal y el MSB lo representa como 01 00 hex. El criterio LSB es usado por la familia de procesadores Intel y compatibles (PC's), mientras que el MSB es usado por Motorola (Macintosh) y en general por los procesadores RISC (Silicon Graphics, Sun Sparc, HP, etc.); como nota curiosa al LSB se le conoce como "Little endian", al MSB como "Big endian" y, en conjunto como "*endianess*" (estos nombres hacen referencia a "Los viajes de Gulliver").

A causa de tales diferencias, los formatos gráficos que operan en diferentes plataformas especifican el tipo de endianess de la computadora en el cual fueron almacenados, permitiendo que otra computadora con distinto orden de bytes pueda interpretar la imagen correctamente; en algunos formatos no especifica el endianess, por lo cual simplemente hay que revisar para que plataforma se creó el formato, por convención se considera que el endianess del formato es el mismo de la plataforma, y no se puede cambiar, no importa que plataforma se este usando; los programadores deben vigilar de cerca este aspecto.

Una vez codificada la imagen, normalmente se le aplica un método de compresión, colocándose en el archivo el resultado y los parámetros del método usado; los formatos pueden usar distintos tipos de compresión, dejando al usuario la opción de elegir el que le ofrezca los resultados más óptimos.

1.2.2 Compresión *Run - Length* [2]

Una de las maneras más simples para comprimir un archivo es la codificación Run-Length, este esquema se basa en simplificar la estructura de la imagen a base de dar un valor del pixel más un número que indica las veces en que se repite, así por ejemplo, se tiene la cadena aaaaaaaabbbbbbccccddeeeeeeffff, donde a, b, c, d, e, f son pixels, usando este esquema se obtiene a8b6c4d2e6f4. Este esquema es muy simple de implementar y trabaja muy bien en imágenes con grandes áreas de color constante.

Una implementación muy popular del Run-length es el basado en paquetes (PackBits ó Raw) usado por formato bitmap de Macintosh. Asumiendo datos de 8 bits se puede codificar valores repetidos en paquetes de dos bytes. El primer byte contiene un número n el cual va de 1 a 127, el segundo byte contiene el valor del pixel a representar; en el caso de cadenas de pixels que se repiten como cfeab, el primer byte da además de las veces en que se va repetir la longitud de la cadena, se da el número de pixels que componen la cadena.

1.2.3 Compresión *Huffman* [3]

La compresión Huffman es un esquema estadístico el cual reduce la longitud promedio del código usado para representar un grupo de pixels, el código Huffman es óptimo en el caso de que todas las probabilidades de los pixels sean potencias enteras de ½. El algoritmo se basa en las siguientes reglas:

1. *Ordenar y valorizar todos los símbolos según su probabilidad de ocurrencia..*
2. *Sucesivamente combinar los dos símbolos de más baja probabilidad para formar un nuevo símbolo; eventualmente construiremos un árbol binario donde cada nodo es la suma de las probabilidades de los nodos antecedentes.*
3. *Trazar un camino para cada hoja, tomando en cuenta la dirección de cada nodo.*

Así, existen muchos árboles y muchos tipos de códigos, pero la longitud final de cualquier árbol siempre será la misma; hay un tipo canónico donde cada nodo es representado por un bit siendo muy compacto y muy fácil de usar .

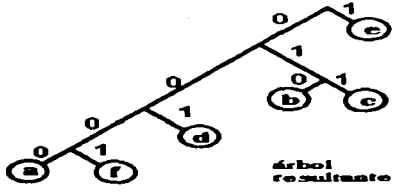
Por ejemplo, se tiene la cadena de pixels abbbccddeeeeeeeeef, las frecuencias de los valores son:

a: 1, b: 3, c: 3, d: 2, e: 9, f: 1

[2] Gailly, Jean-loup. Compression-FAQ part2.

[3] Ibidem.

Ahora se construye el árbol binario, poniendo en las hojas del mismo nivel los valores que tengan la misma frecuencia.



árbol resultante

Así, como a y f tienen la misma frecuencia quedan en el mismo nivel, la suma de la probabilidad de ambos es 2, así se va construyendo de abajo a arriba, hasta terminar.

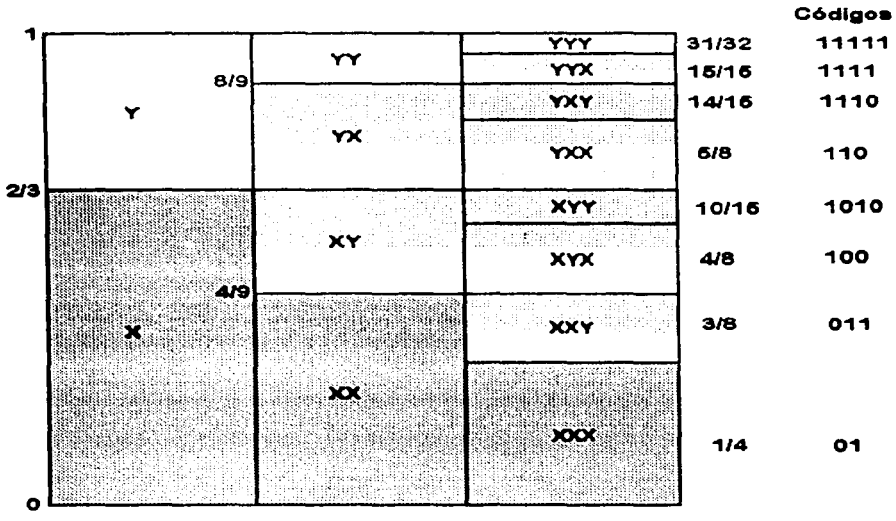
Este esquema es muy fácil de implementar y aplicar, pero sus beneficios máximos están condicionados a que la suma de las probabilidades sea una potencia entera de ½ .

1.2.4 Compresión *Aritmética* [4]

La compresión aritmética, al igual que el método Huffman, usa también la probabilidad de ocurrencia de los caracteres (pixels) del archivo (o cadena), sin embargo, los usa de manera totalmente diferente; este método representa la cadena por medio de un intervalo de números reales dentro de los límites [0,1], usando los valores de probabilidad de ocurrencia de los caracteres se modifica el valor que representa a la cadena: Si el valor probabilístico de ocurrencia es pequeño, el valor del intervalo para representar la cadena se vuelve más pequeño, pero el número de bits necesarios para representar tal intervalo se incrementa; en el caso contrario, un valor de ocurrencia grande, aumentará el intervalo, pero el número de bits necesarios será cada vez más reducido.

Por ejemplo se tienen dos caracteres X y Y, donde X tiene probabilidad de 2/3 y Y tiene probabilidad de 1/3, calculando las probabilidades posibles hasta una cadena de 3 caracteres.

[4] Loc. cit.



Se puede apreciar que al concatenarse caracteres repetitivos se necesita un código más reducido, esto es beneficioso en las imágenes con grandes áreas de un solo color, pero no impide que se obtengan compresiones apreciables con cualquier tipo de cadena.

Este método altamente eficiente, originalmente fue concebido usando la ocurrencia de los caracteres en el idioma inglés, sin embargo, pueden ser usadas las probabilidades de ocurrencia de los caracteres de cualquier otro idioma; pero en el caso de formatos gráficos no se puede usar una tabla fija de probabilidades, ya que tales probabilidades varían de imagen a imagen. Para resolver este problema se usa un *modelo adaptativo*, el cual ajusta la tabla de ocurrencias cada vez que el algoritmo lee un nuevo caracter, para realizar tal adaptación es usada la Codificación Dinámica de Markov (*Dynamic Markov Coding*, DCM) y la Predicción por Coincidencia Parcial (*Prediction by Partial Matching*, PPM).

El método tiene la ventaja de no depender de que la suma de las probabilidades sea una potencia entera de ½. Además en el caso de imágenes con grandes zonas de un solo color, la compresión se optmiza todavía más, ahora bien contrariamente a lo que podría pensarse, es poco usado por los formatos gráficos, ello se debe a que el algoritmo es muy reciente, está patentado, y lo que es más importante, este método usa grandes cantidades de recursos computacionales, tanto de memoria *RAM* como de procesamiento y al añadirse el *DCM* y el *PPM*, tal situación se hace más patente.

1.2.5 Compresión *LZ*

Los métodos de compresión *Lempel - Ziv* son conocidos también como métodos ***substitucionales***, los cuales básicamente sustituyen una frase o grupo de bytes con un valor que hace referencia a una previa ocurrencia de esta. Los métodos *LZ* se pueden agrupar en dos grandes familias, *la familia LZ77 y la familia LZ78*.

1.2.5.1 La familia LZ77

Esta familia fue creada en 1977 por ***Jakob Ziv y Abraham Lempel***, este es un esquema muy simple, el algoritmo mantiene la pista de los n bytes ya revisados, cuando una *frase* (cadena) ya revisada vuelve a presentarse se sustituye tal frase por dos valores, el primero indica la posición del primer caracter de la frase y el segundo la longitud de esta. Para descomprimir simplemente cuando un par ordenado (posición, longitud) es encontrado el algoritmo se va a la posición y copia la frase.

Los algoritmos más populares, han salido de la variación propuesta por *James Storer y Thomas Szymanski* en 1982; no es extraño que buscando mejorar la eficiencia del método *LZ77* se use combinaciones de este con algoritmos como el de *Huffman* (*LZH*) o con el mismo *LZ78* conocido como *LZFG*.

1.2.5.2 La familia LZ78 [5]

Diseñada en 1978, los esquemas *LZ78* van introduciendo frases en un arreglo o diccionario, y cuando una ocurrencia coincide con una frase de este, es sustituida por el índice correspondiente en el diccionario, las diferencias de los distintos esquemas en la familia son principalmente relacionados al manejo del arreglo, el cual comúnmente usa 4k del valor 0 al 255 para los caracteres simples, y del 256 al 4095 para las combinaciones de caracteres.

Para ejemplificar el funcionamiento del método básico de la familia LZ78, se usa la siguiente cadena : /WED/WE/WEE/WEB<fin>, cabe mencionar que <fin> es un caracter de terminación, para lo cual los formatos usan un <EoF> (End of File).

Fase de compresión:

[5] Ibidem.

Cadena de Entrada: /WED/WE/WEE/WEB<fin>

Caracter Introducido	Codigo Resultante	Nuevo codigo del diccionario y la cadena asociada
/W	/	256 = /W
E	W	257 =WE
D	E	258 =ED
/	D	259 =D/
WE	256	260 =/WEE
WEE	E	261 =E/
/W	260	262 =/WEE
EB	261	263 =E/W
<fin>	257	264 =WEB
	B	

Cadena Resultante: /WED<256>E<260><261><257>B<fin>

Fase de descompresión:

Cadena de Entrada: /WED<256>E<260><261><257>B<fin>

Caracter Introducido	Codigo Resultante	Nuevo codigo del diccionario y la cadena asociada
/W	/	256 =/W
E	W	257 =WE
D	E	258 =ED
256	D	259 =D/
E	/W	260 =/WEE
260	E	261 =E/
261	/WE	262 =/WEE
257	E/	263 =E/W
B	WE	264 =WEB
<fin>	B	

Cadena Resultante: /WED/WE/WEE/WEB<fin>

La gran ventaja del algoritmo es la capacidad de trasmitir el diccionario dentro de los datos comprimidos, es decir, no necesita añadir información acerca de la compresión. El algoritmo más conocido de esta familia es la mejora hecha por *Terry Welch* en 1984 (*LZW*) del algoritmo original, usada por los formatos gráficos *GIF y TIFF* - los cuales son muy populares-.

Una situación que ha provocado problemas es que muchos de los algoritmos derivados de las familias LZ - especialmente la familia ***LZ78*** -, son patentados; el caso que más discusión y controversia ha causado es el algoritmo *LZW*, cuya patente es propiedad de Unisys (la cual obtuvo al comprar la compañía estadounidense Sperry Co. en 1987) quien exige la compra de una licencia a todos aquellos creadores de programas que hagan uso del algoritmo, aplicándose a quienes creen programas para los formatos GIF y TIFF.

La compra de la licencia se hace extensiva también a quienes vendan programas que usen los anteriores formatos, y si esto fuera poco, el uso de la licencia exige que se otorgue a Unisys el 0.40% de las regalías netas por las ventas de tales programas. Por ello actualmente se busca sustituir este algoritmo (LZW) por algún otro de la familia *LZ77*.

De ahí la importancia que resulta la elección de un compresor para un formato gráfico.

1.2.6 Compresión *con pérdida*

El método de compresión con pérdida (*Lossy*) , se basa en descartar pixels los cuales no son críticos para la imagen y no afectan de manera notoria la calidad de la misma, siendo típicamente usadas en imágenes para vídeo digital e imágenes televisivas, el método toma en cuenta que el ojo humano tiene una gran capacidad de distinguir luminancia (*tonos de luz*), mientras que es muy pobre en distinguir la policromía, por ello es muy común desechar colores adyacentes dejando la armonía lumínica, logrando con ello una distorsión mínima; el formato gráfico que hace uso de esta compresión es el *jpeg*.

Existen Investigaciones para crear métodos de compresión que sean usables en los formatos gráficos, tales como la teoría de los fractales, wavelet, etc., por ahora y a mediano plazo es difícil su implementación.

1.3 Monocromía y Color

La *monocromía* y el *color* son conceptos muy relacionados con las imágenes, y por ende, con los formatos gráficos; las imágenes independientemente de su definición pueden estar en blanco y negro o en color. Las formas de tratar la *monocromía* y el *color* son variadas, pero todas parten de los siguientes conceptos.

1.3.1 Monocromía

En su forma más simple la monocromía se compone de regiones de un color *puro* (típicamente negro) y regiones de blanco, es común su uso en diagramas, impresoras, plotters y en general, cualquier dispositivo que opera en blanco y negro. Para poder crear sombras, texturas, difuminaciones, se usan patrones que son mezclas de pixels de ambos colores, dando la impresión de tonos de luz; tales imágenes son comúnmente llamadas "*escalas de gris*".

La ventaja de las imágenes monocromáticas, es su reducido tamaño, debido a que simplemente cada pixel vale un bit (1 para el blanco, 0 para el negro ó viceversa). Las desventajas que presenta se deben en primer lugar a que como el ojo humano tiene una gran capacidad para reconocer tonos de gris, se necesitan resoluciones muy altas para lograr un efecto agradable a la vista; en segundo lugar, los patrones que sirven de sombras son muy difíciles de editar.

1.3.1.1 Verdaderas Escalas de gris [6]

Una forma alternativa es almacenar la imagen usando verdaderos tonos de gris (Gray Scale), eliminando el uso de patrones, facilitando la edición. Para estas imágenes hay que hablar de su profundidad.

La profundidad en una imagen es el número máximo de bits usados para representar un tono de gris (o también *luminosidad*), y al mismo tiempo define cuantos tonos diferentes puede desplegar, así una imagen de 4 bits de profundidad puede desplegar hasta 16 tonos diferentes ($2^{\#\ bits}$), los pixels ahora además de tener un valor X y un valor Y (*monocromía simple*), tienen un valor para la profundidad, esta triada afecta el tamaño en bytes de la imagen, por ejemplo una imagen de 500 pixels de alto por 500 pixels de ancho con 16 tonos de gris sin comprimir, seria igual a 500 x 500 x ½ byte, el archivo resultante ocupa 125 000 bytes contra 31 250 bytes si sólo tuviera 1 bit de profundidad. Como el ojo humano fácilmente reconoce cerca de 64 tonos de luz en una imagen son necesarios 6 bits como mínimo para lograr imágenes realistas, pero las estaciones de trabajo actuales usan profundidades mayores en su salida gráfica.

1.3.1.2 Ventajas y problemas con las escalas de gris

La ventaja de las escalas de gris es su fácil manipulación, teniéndose algunos problemas: Primero, cuando una imagen es transferida de un medio físico a uno electrónico, tal como ocurre con el escaneo de fotografías, siempre hay variaciones en la imagen resultante con respecto del original; esto es debido a que los dispositivos manejan las variaciones de luminosidad de una manera lineal, mientras que el ojo humano responde a esas mismas variaciones de manera similar a una función de segundo orden.

Para poder enfrentar tales variaciones muchos formatos usan la técnica llamada *"corrección gama"*, tal técnica consiste en sumar todas las variaciones lumínicas en cada pixel, usándose la siguiente ecuación:

$$Salida = (Entrada)^{GAMA}$$

Donde *Entrada* y *Salida* son los valores de los pixels, el valor gama es especificado junto con el valor del pixel, si falta este valor se considera que la imagen ha sido corregida, o bien, se aplica un valor estándar de corrección.

El otro problema consiste en la situación inversa, es decir, cuando se imprime la imagen usando una impresora blanco y negro, la impresora tiene que usar alguna de las anteriores formas de *monocromía*, aún cuando la imagen este codificada en color, normalmente es traducida a escala de grises, teniendo el inconveniente en la definición, ya que esta tiene una relación directa con la calidad final, así, entre más definición se tendrá una mejor calidad.

[6] Poynton, A. Charles. ColorSpace-FAQ: "Frequently Asked Questions about Gamma and Colour"

1.3.2 Color

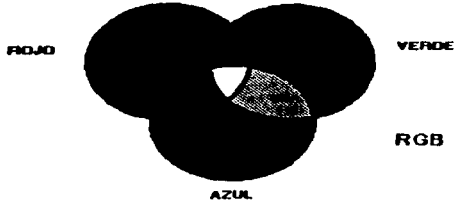
Las imágenes en color casi son la norma en presentaciones, documentos, sitios *internet, aplicaciones multimedia,* entre otros; todas las imágenes en color usan las siguientes estructuras:

1.3.2.1 Puntos de color

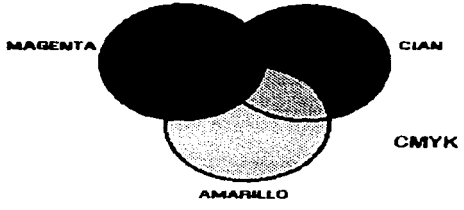
La forma más simple de las imágenes en color, es poner puntos de colores puros *(rojo, verde, azul)* únicamente; el resultado es una imagen a color, pero sin ninguna tonalidad, debido a ello, esta forma se aplica casi exclusivamente a aplicaciones vectoriales; para gráficas con sombras y amplias gamas tonales es usada la mezcla de colores.

1.3.2.2 Mezcla de colores [7]

Para imágenes que exigen mejor calidad, es decir, usan gran cantidad de colores, se utilizan los métodos *RGB, CMYK* y *HSI.*



RGB:
Este método se basa en usar los colores primarios *Rojo- Verde-Azul* - RGB en inglés -, siendo estos colores conocidos como *"aditivos"*, es decir que la suma de los tres da blanco en los medios electrónicos. La codificación de cada pixel se realiza agrupando en orden los valores de Rojo - Verde - Azul, tal orden puede ser en forma RGB o BGR (si el formato no lo especifica se considera RGB por defecto).



CMYK:
Este método usa los colores secundarios (o sustractivos) *Cyan - Magenta - Amarillo,* además se añade el *Negro* por dos razones: Primero, la mezcla de las tintas Cyan, Magenta y Amarillo da un negro muy pardo, por lo tanto se necesita un color negro puro; Segundo, el uso de la tinta negra es una forma barata de obtener colores más profundos al mezclar proporciones de esta y las tintas de

[7] Ibidem.

color. Es obvio mencionar que este es el método usado por las impresoras de color, normalmente se traduce de un formato RGB a uno CMYK antes de imprimir.

HSI:
Este método es relativamente reciente. Parte de los valores *HUE* que se refiere a un color específico como azul titanio, verde talo, etc. El siguiente valor en la composición es SATURATION, que se refiere a la cantidad de blanco que tiene el color, INTENSITY se refiere a la cantidad de luz que tiene el color, a veces se sustituye el término por *luminosity*.

Otra forma de almacenar el color es usando *"planos"* similares a los que usan en las imprentas, es decir, se codifica un renglón o la imagen completa en un solo color o plano, repitendose para los otros colores, pudiéndose usar RGB, CMYK, o HSI.

Existe una relación directa en los bits que almacena un pixel y la cantidad de colores usados, por ejemplo, se considera que una imagen de alta calidad necesita usar 24 bits de profundidad (o 3 bytes, 1 byte para el Rojo, 1 para el Verde y 1 para el Azul), o por decirlo de otra manera, son más de 16 millones de colores disponibles.

Pero es casi imposible que una imagen llegue a usar todos, en la mayoría de las imágenes no son necesarios tantos colores, y como resulta poco recomendable almacenar la información de todos los colores - aumenta el tamaño de un archivo enormemente -, se puede almacenar los datos referentes al color especifico en el pixel, o bien, se pueden usar "mapas de color" en los cuales se hace una tabla en la que cada entrada le corresponde un valor de un color, y simplemente cada pixel contiene el valor índice de la tabla, las tablas tienen un máximo de 256 entradas (colores), dando una reducción apreciable en el tamaño de los archivos.

1.3.2.3 Precisión en el color

Las imágenes a color sufren de los mismos inconvenientes que las imágenes blanco y negro, por ello es común añadir un valor de corrección gama; el uso del valor gama en una imagen evita que al pasar de un equipo a otro se vea demasiado "obscura" o demasiado clara; también permite evitar que un color sea más relevante que los otros, en otras palabras, que una imagen se vea "azulada", "verdosa" o muy roja, por ello la corrección se hace en base a un valor por convención denominado "punto blanco", que equivale al brillo que se obtiene al fundir hierro a 5600 °K, aunque por razones estéticas algunos fabricantes no siempre cumplen tal norma, por ello, hay organizaciones encargadas que crean y garantizan el cumplimiento de la normas en los equipos electrónicos, y entre ellas contamos al CIE, SMPTE, NTSC, ISO y ANSI, en base al trabajo de tales instituciones se estandarizan los formatos.

1.4 Aplicaciones de los formatos gráficos

Los formatos gráficos tienen una gran nivel de uso en todo tipo de aplicaciones en la actualidad. Pero la principal aplicación de los formatos gráficos está en las propias aplicaciones, pues con un estándar fijo, una imagen que para nosotros representa

información valiosa, puede ser codificada en diversos estándares sin perder sus cualidades y ser manipulada por los diversos sistemas operativos, plataformas, aplicaciones y finalmente por nosotros mismos.

En la propia industria de la computación, el uso de los formatos gráficos es muy importante, pues éstos son el soporte de los sistemas operativos basados en gráficos como *Microsoft Windows, Mac OS* ó *X-Windows* en *UNIX*, pues éstos necesitan de un formato nativo para guardar toda su información gráfica. La aparición de los formatos gráficos fijos y ampliamente aceptados por la industria de la computación, permite crear aplicaciones sólidas que pueden intercambiar información entre ellas.

En las Ciencias son usados en diagramas de flujo, gráficas - de pastel, de barra, poligonal-, organigramas, para representar resultados de análisis comparativos, en simulaciones de procesos que son imposibles entender sin una representación gráfica *(fractales, movimiento de estrellas, etc.)*. En la medicina y química, sirve para almacenar imágenes de *Tomografía axial*, mecanizar el conteo de glóbulos rojos, gráficar las características de nuevas estructuras moleculares, en el análisis de virus y bacterias etc.

En la industria, para el desarrollo, simulación, perfeccionamiento y control de procesos industriales, plantas químicas; almacenamiento y manipulación de fotografías de satélites, aéreas; visualización de procesos químicos e industriales para su estudio y perfeccionamiento, etc.

En el Diseño sus mejores aplicaciones son en la autopublicación, creación de publicidad y multimedia, que permiten abaratar los costos de las producciones gráficas y elevar dramáticamente su calidad y presentación en un menor tiempo en contra de los métodos tradicionales. El uso de los formatos gráficos permite que varios grupos de trabajo puedan crear separadamente la tipografía, retoque de imágenes y texto gráfico, e incluirlo todo en una sola creación final, sin necesidad de preocupase por hacer el trabajo en varias fases a la vez.

En la arquitectura e ingeniería, los formatos gráficos han marcado el estándar para hacer de manera eficiente trabajos en dibujo de planos, vistas isométricas y de perspectiva, e incluso con el auxilio de la realidad virtual, poder ver un proyecto sin necesidad de estar terminado y hacer las modificaciones con sólo unos cambios al programa, para modificar las texturas, tipos de materiales, orientación y luz del futuro edificio. Con el apoyo de los formatos gráficos los programas de CAD (*Diseño Asistido por Computadora*), pueden visualizar y determinar las características de una estructura antes de su construcción final.

En la educación son el auxiliar de los simuladores y tutoriales de entrenamiento, en las más diversas áreas. En el esparcimiento, es indudable su aplicación en juegos de vídeo, animaciones por computadora, arte por computadora o de ocio, etc.

1.5 Tipos de formatos de imagen

Los formatos gráficos pueden ser agrupados por muy diversas características, pero por razones más académicas estoy usando la clasificación en la cual se distingue por sus características de codificación.

Los formatos incluidos aquí no son los únicos, existe algo más de un centenar de formatos gráficos usados en las más diversas aplicaciones y fines. Elegí los formatos presentes en este trabajo, porque son los más ampliamente aceptados y usados en la comunidad dedicada a la computación, están suficientemente documentados y más importante aún, estos formatos ilustran claramente las formas más generales en que es almacenada una imagen.

Los tipos de formatos gráficos son el *Bitmap* (raster), *Vector, PDL* y *Metaarchivo*.

1.5.1 BitMaps

Los bitmaps son el tipo más simple y popular para almacenar imágenes, consiste básicamente en que una imagen es dividida a manera de cuadrícula en un plano cartesiano donde cada cuadro es un pixel, y a su vez, cada pixel contiene los valores de lumínosidad (Blanco y Negro) o de color (RGB, CMYK, HSI); para encontrar un determinado pixel se aplica el mismo principio de un mapa de localización, de ahí el nombre -bitmaps- ó mapas de bits, aunque también son conocidos como formatos RAW.

Casi siempre el punto de inicio se sitúa en la esquina superior izquierda; en esta área no existe puntos negativos y cada pixel es almacenado en una lista creándose un "mapa" de bits; en los formatos bitmaps siempre está asociado una serie de parámetros conocido como cabecera que describe a la imagen, la cual específica en que versión del formato fue guardado, además es común encontrar formatos que guardan en la cabecera información relativa a la corrección gama usada, comentarios del autor de la imagen, el método de compresión usado, el tipo de *Endianess* empleado, etc.

La cabecera puede estar compuesta por una sección como el MACpaint, estar dividida en varios bloques como el Targa, o bien, tener varias subcabeceras como en el BMP/DIB; la cabecera es un elemento primordial para poder decodificar correctamente una imagen, por ello los programas de edición codifican primero y verifican el correcto orden de la cabecera después.

Exite una gran cantidad de formatos gráficos bitmap en funcionamiento, tanto privados como públicos, a continuación se enumera los formatos mejor documentados, y que son en general, los más aceptados en la comunidad dedicada a la edición de imágenes.

1.5.1.1 PCX

Propietario: *Zsoft Corporation*, **actualmente del dominio público**
Extensión: *.pcx*

El formato PCX es uno de los más viejos, y por tanto, de los formatos más ampliamente soportados en el mercado. Existen hasta el momento 5 versiones (*véase*

la tabla 1.1), la versión más reciente soporta colores de 24 bits, ya sea usando una paleta de hasta 256 colores, o codificación RGB de 24 bits, no soporta la codificación *CMYK* ni *HSI* . Por ser un formato nativo de la familia de computadoras IBM PC's y compatibles, está fuertemente ligado al desarrollo de los monitores de tal familia, las versiones sucesivamente han ido soportando las características de los monitores *CGA, EGA, VGA, XGA*, actualmente la versión 5 del formato permite soportar futuras mejoras en los monitores, sin embargo, debido a lo anterior no cuenta con una bandera que indique el tipo de orden de byte.

Tabla 1.1: Versiones del formato PCX

Número de Versión	Producto / soft que lo presento	Tipo de Imagen que maneja
0	PC Paintbrush v. 2.5	Monocromático (2 colores) / 4 colores.
2	PC Paintbrush v. 2.8 con información de la paleta	Soporta todo lo anterior y además 16 colores.
3	PC Paintbrush v. 2.8 sin información de la paleta	
4	PC Paintbrush para Windows	
5	PC Paintbrush para Windows 3.x y mejor, MsDOS	Soporta todo lo anterior y además 256 colores de 24 bits de profundidad y colores RGB de 24 bits.

Estructura

La estructura del PCX comienza con una cabecera de 128 bytes de longitud, en la cual se anotan detalles como número de versión, la resolución de la imagen (en puntos por pulgada, dpi) y su dimensión (en pixels), el número de bytes por línea, y el número de planos. Para ver más en detalle la cabecera véase la tabla 1.2. En el caso de usarse paleta de colores existen tres tipos disponibles:

Paleta CGA: Actualmente obsoleta y en términos prácticos no utilizada desde la versión 5. Usa dos bytes: El byte 16 para definir una paleta de 4 combinaciones posibles y el Byte 19 que almacena tres variables C, P, y Y. La variable C define si la imagen es de *color* (0) o *monocromo* (1), P define la paleta Amarilla (0) o Blanca (1), ya que el monitor CGA tiene dos modalidades Amarilla (ámbar) o Blanca y Y que define la intensidad baja (0) o Alta (1).

Paleta EGA/VGA: La cabecera de la paleta se extiende desde el Byte 16 al 63, dividiéndose en 16 tripletas RGB. Cada color puede tomar solo un rango pequeño de 256 valores posibles, cada rango se denomina nivel y abarca 64 valores (o 4 niveles).

Paleta VGA: Esta paleta está colocada al final del bloque de imagen, teniendo una etiqueta de inicio 0C hexadecimal, a partir de ahí, de cada tres bytes se obtiene el valor de cada color, sin embargo, como solo puede ofrecer 64 niveles, la paleta debe ser dividida en cuatro al momento de ser desplegada.

El hecho que el VGA tenga dos posibles paletas se debe a que los primeros monitores solo manejaban 16 colores en modo gráfico. En el caso de colores RGB de 24 bytes simplemente se ignora cualquier valor en cabecera de la paleta y se lee el bloque de imagen directamente.

Tabla 1.2: Cabecera del formato PCX

Byte de inicio	longitud	Descripción
0	1	Bandera Zsoft en el pcx vale 10 (0A hex)
1	1	Número de Versión (véase tabla 1.1)
2	1	Codificación 1= Run Length.
3	1	Número de bits/pixel en cada plano (véase tabla 1.3)
4	8	Dimensiones de la imagen en pixels: X mín., Y mín., X máx, Y máx (el punto 0,0 está en la parte superior izquierda de la pantalla)
12	2	Resolución horizontal (pixel por pulg.) cuando es impresa.
14	2	Resolución vertical (pixel por pulg.) cuando es impresa.
16	48	Cabecera de la paleta CGA, EGA, VGA
64	1	Reservada por Zsoft siempre vale 0.
65	1	Número de planos de color.
66	2	Memoria necesitada para guardar un plano sin comprimir.
68	2	Interpretación de la cabecera 1 = color o B y N. 2 = escala de gris.
70	2	Tamaño de la pantalla de video horizontal (x-1).
72	2	Tamaño de la pantalla de video vertical (y-1).
74	54	Fin de la cabecera (caracteres en blanco).

El formato PCX usa la compresión Run-Length (RLE) para codificar la imagen, la codificación se realiza por medio de planos de color, en este caso, los colores de cada línea son codificados por separado, así, cada línea se almacena de la siguiente manera Línea 1: *RRRRRRR...RGGGGGGG...GBBBBBBBB...B*, y así sucesivamente hasta terminar con todas las líneas de la imagen, en el caso de existir una paleta se anota únicamente los índices del mapa, Línea 1: *IIIIIIIII...I*, el número de planos, el número de bytes usados para codificar la imagen definen su tipo, esto se aprecia más claramente en la *tabla 1.3.*, así el monocromático usa un plano de un bit, mientras que el *RGB* usa tres planos de ocho bits cada uno.

Tabla 1.3: Codificación de Colores

Numero de bits por pixel	Numero de planos	Descripcion
1	1	Monocromo
1	2	4 colores
1	3	8 colores
1	4	16 colores
2	1	Paleta CGA 4 colores
2	4	16 colores
4	1	Paleta CGA 16 colores
8	1	Paleta CGA 256 colores
8	3	16.7 Millones de colores

1.5.1.2 MacPaint

Propietario: ***Apple Computer Inc.***
Extensión: ***.pntg, .pnt***

El formato PNTG es nativo de los sistemas Macintosh, un formato que *únicamente maneja imágenes fijas de 720 x 576 pixels*, y solamente soporta imágenes monocromáticas de 1 bit de profundidad, no existe ninguna variación, pero a diferencia del PCX, contiene una bandera de orden de bytes, permitiendo que sea usado en distintas plataformas facílmente; a pesar de ser un formato rígido y limitado para editar por sus características, es práctico y el tamaño de sus imágenes es muy reducido, facilitando su transmisión.

Estructura [8]
La estructura de un formato MacPaint es muy simple, cuando es usado exclusivamente en sistemas Macintosh contiene una estructura propietaria conocida como *Macbinary header* - localizada al principio del archivo-, que usa estructuras *fork* que facilita su utilización por sistema operativo de Macintosh, existe una versión 2.0 en estado de prototipo en la cual las estructuras Macbinary se localizan al final del archivo.

Cuando el formato es usado en otros sistemas se sustituyen las estructuras fork por una cabecera (véase la Tabla 1.4) de 128 bytes, el PNTG usa la codificación RLE (Run Length Encoding) en una variante conocida como Packbits, también usado en el *PCX* y el *TIFF*. La versión 2.0 propone banderas adicionales para el manejo y edición del archivo.

Tabla 1.4:Macbinary Header

Offset	Longitud	Descripcion
00	1	Número de versión, siempre vale cero.
01	1	Longitud del nombre del archivo, hasta 63 caracteres son permitidos.
02	63	Nombre original del archivo en ASCII, según la longitud previamente dada.
65	4	Código tipo archivo Macintosh, PNTG, en ASCII.
69	4	Código denotando autoria en Macintosh, en ASCII; MPNT.
73	1	8 banderas de búsqueda, ignorar si se lee, se pone 0 si se escribe.
74	1	Siempre vale 0.
75	4	Posición X, Y en el archivo en la ventana Mac; ignora si se esta leyendo, ceros si esta escribiendo.
79	2	Ventana de archivo de búsqueda: ignora si se lee, ceros si se escribe.
81	1	Bandera "protegida" en LSB.
82	1	Siempre vale 0.
83	4	Longitud de datos fork para la reconstrucción del archivo en Macintosh.
87	4	Longitud de las fuentes fork (cero).

[8] Kay, David C. y Levine, John R. Graphics File Formats., pág. 33.

Tabla 1.4:Macbinary Header (Continuación)

Offset	Longitud	Descripcion
91	4	Fecha de creación en formato MSB.
95	4	Fecha de modificación en formato MSB.
		Valores en Macbinary II
Offset	**Longitud**	**Descripción**
99	2	Ignora si se lee, ceros si se escribe.
101	1	8 banderas de búsqueda adicionales; ignórese y/o vale cero.
116	4	Longitud el archivo desempacado; usados por programas de comunicación, ignórese y/o vale cero en cualquier otra situación.
120	2	Cero.
122	1	Versión mínima Macbinary II usado por los programas de escritura; vale 129.
123	1	Versión mínima Macbinary II usado por los programas de lectura.
124	2	CRC de los previos 124 bytes; ignórese y/o vale cero.

1.5.1.3 GEM

Propietario: *Novell*
Extensión: *.img*

El formato IMG fue desarrollado por *Digital Research* (hoy *Novell*),es un formato bien apoyado, es de estructura simple, las imágenes que maneja son del tipo monocromático, escala de gris y color aunque no apoya mapas de color.

Estructura
Los archivos GEM tienen una cabecera de ocho o nueve palabras de 16 bits, como se ve en la Tabla 1.5, los datos se codifican en forma de planos de color, y según el número de planos es la cantidad de color, tal como ocurre en el formato PCX, cada línea de la imagen es almacenada por separado.

Tabla 1.5:Cabecera GEM/IMG

Offset	Descripcion
0	Número de versión (vale 1)
2	Longitud de la cabecera en palabras
4	Profundidad total de la imagen
6	Longitud de las estructuras, típicamente vale 2
8	Ancho del pixel en micrones
10	Alto del pixel en micrones
12	Ancho de la imagen en pixels
14	Alto de la imagen en pixels
16	Bandera opcional

El método de compresión es el *Run Length*, para manejar este método de compresión, se tiene dos tipos de formas bloques de pixels y cadenas de un pixel, en el primer caso se representa primero el bloque y luego el número de repeticiones.

Para indicar una repetición de líneas, su usa la siguiente cadena (en hexadecimal):

00 00 FF NN

Donde *NN* es el número de repeticiones..Hay tres tipos de bloques posibles en una línea: Cadenas de bits, estructura de colores, colores puros.

Una cadena de bits, se indica con la bandera *80 NN*, para las estructuras de colores (cadenas de distintos pixels) se indica con *00 NN*, y para colores puros (lista de unos o ceros) se puede usar cualquiera de las anteriores indicaciones simplemente si se usa la etiqueta 80, se pondrán unos, en caso contrario serán ceros; en la práctica es muy común que sólo se utilice un tipo de bloque por línea dejando la imagen virtualmente sin comprimir. El formato GEM codifica su información en formato *MSB* (el byte más significativo va al principio) tanto para la cabecera como la imagen en sí.

1.5.1.4 TGA

Propietario: *Truevision Incorporated*
Extensión: *.tga*

El formato de *Truevision Inc.* es uno de los formatos más populares en sistemas PC, MAC, Amiga, entre otros. El formato "*Targa*" como popularmente se le conoce al *TGA* -Truevision Graphic Archive-, no es más que una confusión debido a que se le asocia con las tarjetas de captura de imágenes "Targa" de *AT&T* -división computadoras personales-, donde el formato fue presentado. Pero ya que se le conoce de ambas maneras en el medio, usaré ambos nombres indistintamente.

La intención de este formato es facilitar el almacenamiento de imágenes tipo bitmap que son directamente recibidas de escaners. Provee de una estructura flexible y fácil de usar que responde a las múltiples necesidades del escaneo de imágenes, Truevision agrupa dentro de la especificación TGA muchos tipos de estructuras, las cuales van desde el que no incluye imagen alguna, hasta el tipo que soporta imágenes con compresión de datos., además cabe mencionar que Truevision permite registrar subtipos para uso privado. Las diferencias entre la versión 1.0 y 2.0 se centran principalmente en añadir campos de identificación del propietario de la imagen, además de ampliar los tipos de compresión que inicialmente era el RLE (RunLength Encoding), se añadió el método Huffman y la compresión Delta (variante del RLE).

Los tipos y subtipos de formato que soporta la especificación TGA son muy amplios, provocando a menudo que no sea posible disponer de decodificadores para todos, por ello los tipos más "universales"- y mejor documentados- de la especificación TGA son:

Tabla 1.6: Tipos de imágenes TGA

Código Targa	Tipo de Imagen que Soporta
0	No soporta imágenes.
1	Imágenes en pseudo color sin compresión.
2	Imágenes en truecolor sin compresión.
3	Imágenes en Blanco y Negro (B&W) sin compresión.
9	Imágenes en pseudo color con compresión del tipo Runlength.
10	Imágenes en truecolor con compresión del tipo Runlength.
11	Imágenes en Blanco y Negro (B&W) con compresión del tipo Runlength.
32	Imágenes en pseudo color con compresión del tipo Huffman, Delta y Runlength.
33	Imágenes en pseudo color con compresión del tipo Huffman, Delta y Runlength a 4 pasadas (lecturas).

Sin embargo, aún estos tipos no son apoyados por todas las aplicaciones, para fines de uso y de facilidad, los tipos 1, 2, 9, y 10 que son ampliamente aceptados. En adelante me referiré al formato en términos de la *especificación 2.0.*

La ventaja de este formato consiste en ser simple, bien estructurado y fácil de implementar, además, actualmente soporta imágenes a color de 24 ó 32 bytes en truecolor (*color real en la imagen con 8 bytes por canal*) teniendo solamente en contra la variedad de subtipos.

Estructura del formato

El formato Targa inicia con una cabecera estructurada, seguido de un campo de identificación de longitud variable, un mapa de colores, la sección de la imagen propiamente dicha y un campo de terminación de archivo. La sección del campo de identificación empieza en el Offset[9] 18 inmediatamente después de la cabecera (*añadido en la versión 2.0*). Los bytes deben de ser leídos en el formato de Intel (*LSB: byte bajo - byte alto*).

Las imágenes en color pueden tener dos maneras de procesar el color, por un lado, los pixels llaman a un mapa de colores seleccionado sólo un valor (color), designándosele a esta manera como *PseudoColor*. La otra es conocida como *Truecolor* consiste en que cada pixel está conformado por la unión de tres colores rojo verde azul (*RGB*). Para aquellas imágenes en escala de gris se les denomina Black and White (*Blanco y Negro*).

Las imágenes son siempre guardadas en renglones, pero no hay un criterio definido en cuanto al orden en que deben ser guardados o leídos - de arriba a abajo o viceversa -, debido a la variedad de escaners y capturadores de pantalla; por ello se recomienda que se lea en la cabecera la secuencia de escaneo. Tampoco hay un criterio que norme la secuencia del renglón - der. a lzq. o viceversa -, pero es muy común tomar la lectura de izquierda a derecha, sin embargo, para evitar incompatibilidades la recomendación anterior debe seguirse. Debe tomarse en

[9] Offset se refiere a la posición del byte con respecto del inicio del archivo.

cuenta que un programa decoficador de este formato debe estar preparado para leer la información en cualquier secuencia valida.

Cabecera

La cabecera es muy importante en cualquier formato raster, y el formato TGA no es la excepción. Cada uno de los campos de la cabecera da las características de la imagen (*tamaño, profundidad, tipo de color, orden de presentación, etc.*), en el Targa hay campos obligados y campos opcionales, los campos obligatorios son de suma importancia al momento de codificar como al descodificar, ya que el éxito de almacenar y recuperar una imagen correctamente, consiste en respetar estos campos y verificar que los valores contenidos sean válidos. Acerca de los opcionales, pueden ser "ignorados" por el programa de recuperación, si no son del interés del creador de aplicación, pero, para evitar incompatibilidades, se deben formatear los campos no usados con los valores de anulación especificados.

Tabla 1.7: Cabecera del formato TGA

Offset	Longitud	Descripción
0	1	*Número de Caracteres en el Campo de Identificación.* Este campo es de un byte entero sin signo, el cual especifica la longitud de la cabecera de la imagen. tiene un rango de 0 a 255, si el valor es cero significa que no existe cabecera para la imagen (Código Targa 0).
1	1	*Tipo de Mapa de Color.* (Pseudo o True color). Este campo tiene un valor binario. (1 para Pseudo color, 0 para True color)
2	1	*Código del Tipo de Imagen.* Este campo siempre es un valor binario (según el tipo de imagen. 0, 1, 2, ... 33)
3	5	*Especificación del Mapa de Color.* El mapa de color existe si el tipo de mapa de color vale 1, en caso de valer 0, todos los valores del mapa de color se vuelven cero. El mapa de color nunca ha tenido gran aceptación, la gran mayoría de las implementaciones usan Truecolor (que no usa mapas de color)
3	2	*Origen del mapa de color.* El byte entero (en MSB), es la indicación de la primera entrada del color.
5	2	*Longitud del mapa de color.* El byte entero (en MSB), da la amplitud (número de colores) que tiene la imagen, el máximo valor del mapa es de 256., el origen y la longitud dan el número de colores existentes.
7	1	*Tamaño de la entrada del color.* Número de bits en cada entrada de color, este indica la profundidad del color, que es codificado según el estándar, habiendo 16 bits para el estándar Targa 16 en orden ARGB, o RGBA (queda a criterio del programador especificar el orden, pero la gran mayoría toma el orden ARGB) donde cada letra es un color al cual le corresponde un bit: R=rojo, G=verde, B=azul, A es un bit de información de la aplicación creadora, puede valer 0 sin ningún problema. 24 bits -o 3 bytes- para el estándar Targa 24, donde cada byte es un color en orden Rojo-Verde-Azul. 32 bits -o 4 bytes- para el estándar Targa 32. El cuarto byte es de información (A) dejado para uso de la aplicación, es comúnmente dejado en cero. Por razones de velocidad y facilidad es muy común el estándar Targa 24.

Tabla 1.7: Cabecera del formato TGA (Continuación)

Offset	Longitud	Descripción
8	10	*Especificación de la Imagen.* Define las características de la imagen según el valor del *Código del Tipo de Imagen.*
8	2	*Origen X de la imagen.* Entero (en MSB), es la coordenada X de la esquina inferior izquierda de la imagen.
10	2	*Origen Y de la imagen.* Entero (en MSB), es la coordenada Y de la esquina inferior izquierda de la imagen.
12	2	*Ancho de la imagen.* Entero (en MSB), es el ancho de la imagen en pixels, con base en los valores X y Y del origen - situado en la esquina inferior izquierda -.
14	2	*Alto de la imagen.* Entero (en MSB), es el alto de la imagen en pixels, con base en los valores X y Y del origen - situado en la esquina inferior izquierda -.
16	1	*Número de bits por Pixels.* Es el número de bits que tiene cada pixel. Pueden valer 8, 16, 24, o 32.
17	1	*Byte de Descripción de la Imagen* bits 0-3 Número de bits de atributo asociados con cada pixel. bit 4 Reservado debe ser cero. bit 5 Bit que marca el origen de la pantalla. 0= Origen en la esquina inferior izquierda. 1= Origen en la esquina superior izquierda. Este bit debe valer 0 para las imágenes de Truevision. bit 6-7 Bandera de entrelazado de almacenamiento de los datos. 00= sin entrelazado. 01= entrelazado de dos pasadas (par, impar). 10= entrelazado de cuatro pasadas. 11= reservado. (por los tipos que son usados este bit vale 0).

Identificador

Este campo fue añadido en la versión 2.0, con la intención de facilitar, y ayudar a documentar la imagen, aunque, actualmente son pocos los programas que toman en cuenta este campo.

Tabla 1.8: Adiciones al formato en la versión 2.0

Offset	Longitud	Descripción
18	varia	*Campo de Identificación de la Imagen.* Contiene un campo sin formato para almacenar información acerca de la imagen, el cual se especifica en 1 byte del registro de la imagen. Va de 0 a 255 caracteres, usualmente se da el valor de cero por razones de portabilidad, sin embargo, se deja este campo para facilitar al propietario de la imagen la documentación de la misma. Y si se necesita anotar más información acerca de la imagen, esta se guarda después del campo de datos de la imagen.

Imagen

Los anteriores bloques definen como el decodificador debe leer la imagen y son comunes a todos los tipos. En esta sección, sin embargo, las diferencias de cada tipo se hacen presentes, debiéndose explicar las variantes más populares.

Cada uno de los tipos enumerados se reconocen por el campo - el cual debe contener el número del tipo (especificación)-, este valor debe corresponder con el valor del campo *Tipo de Mapa de Color*, ya que es usado para verificar la integridad de los datos de la imagen.

Tabla 1.9: *Especificación Truevision 1*: imágenes en pseudo color sin compresión.

Offset	Longitud	Descripción
varia	varia	*Datos del Mapa de Color.* El valor del Tipo de Mapa de Color es 1. El offset o byte de inicio es determinado por el tamaño del campo de identificación de la imagen. La longitud es determinada por la especificación del mapa de color, el cual describe el tamaño de cada entrada y el número de entradas. Cada entrada del mapa de color es de 2, 3, o 4 bytes. los bits que no son usados se asumen para bits de especificación de atributos. La entrada de 4 bytes contiene 1 byte para el azul, 1 byte para el verde, 1 byte para el rojo y 1 byte de información de atributo de la aplicación (en raras ocasiones no vale 0, y aun así, no afecta la imagen). La entrada de 3 bytes contiene 1 byte para el azul, 1 byte para el verde, 1 byte para el rojo. La entrada de 2 bytes es fragmentada como sigue: ARRRRRGGGGGBBBBB, donde cada letra representa un bit., pero, por el orden de almacenamiento LSB, el primer byte será GGGBBBBB, y el segundo ARRRRRGG. La "A" representa un bit de atributo de corrección.
varia	varia	*Campo de Datos de la Imagen.* Este campo especifica (ancho) x (alto) los índices del mapa de color. Cada índice es almacenado como un número entero de bytes (típicamente 1 o 2). Todos los campos son sin signo. El byte de bajo orden es almacenado primero (según el formato de Intel LSB). al leer un valor de este campo se hace referencia al índice del mapa de color, y el valor (color) correspondiente del índice es desplegado.

Tabla 1.10: *Especificación Truevision 2*: imágenes en truecolor sin compresión.

Offset	Longitud	Descripción
varia	varia	*Datos del mapa de color.* El valor es 0, por lo tanto este campo no se usa.
varia	varia	*Campo de datos de la imagen.* Esta especifica (ancho) x (alto) de los pixels. Cada pixel especifica un valor de color RGB el cual es guardado como un número entero de bytes. La entrada de 2 bytes es fragmentada como sigue: RRRRRGGGGGBBBBB, donde cada letra representa un bit en formato LSB. Las imágenes Targa 24 son guardadas a veces como imágenes Targa 32.

Tabla 1.11:*Especificación Truevision 9*: imágenes en pseudo color con compresión del tipo Runlength.

Offset	Longitud	Descripción
varia	varia	*Datos del mapa de color.* El offset es determinado por el tamaño del campo de identificación de la imagen. La longitud es determinada por la especificación del mapa de color, el cual describe el tamaño de cada entrada y el número de entradas. Cada entrada del mapa de color es de 2, 3, o 4 bytes. La entrada de 4 bytes contiene 1 byte para el azul, 1 byte para el verde, 1 byte para el rojo y 1 byte de información de atributo. La entrada de 3 bytes contiene 1 byte para el azul, 1 byte para el verde, 1 byte para el rojo. La entrada de 2 bytes es fragmentada como sigue: AR...RG...GB...B, donde cada letra representa un bit., pero, por el orden de almacenamiento LSB, el primer byte será GGGBBBBB, y el segundo ARRRRRGG.
varia	varia	*Campo de datos de la imagen.* Esta especifica (ancho) x (alto) de los índices del mapa de colores. Cada renglón de los índices son comprimidos usando la compresión Run-length.

Tabla 1.12: *Especificación Truevision 10*: imágenes en truecolor con compresión del tipo Runlength.

Offset	Longitud	Descripción
varia	varia	*Datos del mapa de color.* Si la especificación del mapeo de color es 0, este campo no se aplica. De otro modo, debe leer (dar una pasada) para obtener la imagen. La especificación del mapeo de color describe el tamaño de cada entrada, y el número de entradas que va a tener que evitar. Cada entrada del mapa de color puede ser de 2, 3, o 4 bytes.
varia	varia	*Campo de datos de la imagen.* Esta especifica (ancho) x (alt) de los índices del mapa de colores. Cada renglón de los índices son comprimidos usando la compresión Run-length.Cada pixel especifica un valor de color RGB el cual es guardado como un número entero de bytes. La entrada de 2 bytes es fragmentada como sigue: ARRRRRGGGGGBBBBB, donde cada letra representa un bit., pero, por el orden de almacenamiento MSB, el primer byte será GGGBBBBB, y el segundo ARRRRRGG. La "A" representa un bit de atributo. La entrada de 3 bytes almacena un dato RGB. La entrada de 4 bytes contiene un valor RGBA. Para mayor rapidez, las imágenes Targa 24 son guardadas a veces como imágenes Targa 32.

Bloque de terminación

La versión 2.0 añade un nuevo campo: Footer o bloque de terminación, esta nueva sección consiste de cuatro campos: El puntero del archivo de extensiones, el puntero del archivo del directorio del desarrollador, la cadena de identificación "*TRUEVISION-TARGA*", y por último, la cadena de terminación.

Tabla 1.13: Terminador (Footer) del archivo TGA

Offset	Longitud	Descripcion
0	4	*Puntero del Archivo de Extensiones.* Si el valor del puntero es uno, se direcciona al archivo de Extensiones. Este archivo es una tabla con campos de características y extensión fija. por ser un archivo encadenado al archivo de imagen, normalmente es ignorado, y las aplicaciones pueden evitar leer este archivo sin perjuicio alguno en la imagen. Todos los campos en la tabla son opcionales, con caracter informativo, pero al ser estática la tabla, cada campo debe ser terminado con un caracter nulo, para los campos sin usar deben ser llenados con ceros si son numéricos, y con blancos si son de texto. Cabe aclarar que el campo Alfa o A en los bytes de colores, se definen aquí en tipo de atributo como sigue: 0 No hay datos alfa. 1 Indefinido, puede ser ignorado. 2 Indefinido, no puede ser ignorado. 3 Datos alfa regulares. 4 Datos alfa premultiplicados. Donde alfa es un multiplicador de los valores RGB. La tabla de corrección es un archivo auxiliar, que permite verificar si la tabla de colores fue bien leída. La imagen tamaño estampilla es una reproducción de la imagen original, de 64x64 pixels. La tabla Scan line contiene todas las posiciones (en pantalla), de cada renglón de la imagen.

Campos del archivo de Extensiones.

Offset	Longitud	Descripcion
0	2	Tamaño del área de extensiones (debe valer 495).
2	41	Nombre del autor, texto
43	81	Comentarios del Autor línea 1, texto
124	81	Comentarios del Autor línea 2, texto
205	81	Comentarios del Autor línea 3, texto
286	81	Comentarios del Autor línea 4, texto
367	2	Mes de creación del archivo, (1-12)
369	2	Día de creación del archivo, (1-31)
371	2	Año de creación del archivo, (en formato AAAA)
373	2	Hora de creación del archivo, (0-23)
375	2	Minuto de creación del archivo, (0-59)
377	2	Segundo de creación del archivo, (0-59)
379	41	Nombre del trabajo, texto
420	2	Horas empleadas en el trabajo (0 -XX)
422	2	Minutos empleados en el trabajo (0 -XX)
424	2	Segundos empleados en el trabajo (0 -XX)
426	41	Nombre del programa en que se creo la imagen.
467	2	Número de versión (con formato X.XX por cien)
469	1	Letra de versión (eg.: X.XXA, X.XXB, etc.)
470	1	Fondo color Azul.
471	1	Fondo color Verde.
472	1	Fondo color Rojo.
473	1	Fondo color Alfa.
474	2	Numerador del aspecto del pixel.
476	2	Denominador del aspecto del pixel.
478	2	Numerador del valor Gamma.
480	2	Denominador del valor Gamma.

		482 2 Offset del archivo de corrección del color. 486 4 Offset de la imagen en tamaño estampilla (64 x 64 pixels) 490 4 Offset del archivo de Line Scan. 494 4 Tipo de atributo.
4	4	*Puntero del Archivo del Directorio del Desarrollador.* Si el valor del puntero es uno, se direcciona al archivo del Directorio. Este Directorio tiene como finalidad, dar información acerca de la imagen, y varios aspectos en general. A diferencia del campo identificador, el directorio está estructurado, sin embargo, el diseñador puede elegir las características del campo de comentarios como son el tipo y la extensión; no obstante, este campo es poco usado. **Campos del Directorio del Desarrollador** Offset / Tamaño / Descripción 0 2 Número de entradas (campos) 2 2 Marca del campo 1 4 4 Posición inicial del campo 1 8 4 Tamaño del campo 1 12 2 Marca del campo 2 14 4 Posición inicial del campo 2 18 4 Tamaño del campo 2 ... ...
8	17	*Cadena de Identificación.* El texto de este campo consiste de la cadena "TRUEVISION-TARGA", la cual no se puede alterar, y va en mayúsculas.
25	1	*Cadena de Terminación.* Es un cero binario, al ir inmediatamente después de la cadena de identificación, se considera que el archivo fue terminado correctamente.

1.5.1.5 Sun Raster

Propietario: ***Sun Microsystems***
Extensión: ***.ras***

Sun creó el formato .ras como formato base para su sistema operativo "Sun OS" -que es una implementación del Unix-. El formato permite tratar imágenes monocromáticas, escala de gris, y en color (mapas de color o RGB). Las imágenes son almacenadas con una profundidad de 24 ó 32 bits por razones de compatibilidad, aunque el formato permite una profundidad libre.

Estructura [10]
El formato comienza con una cabecera de 32 bytes, dividido en ocho números en hexadecimal que son nombres simbólicos en lenguaje C, véase la tabla 1.14.

[10] Kay, David C. y Levine, John R. ob. cit., pág. 121.

Tabla 1.14: Cabecera del formato SUN Raster

Offset	Nombre en C	Descripción
0	magic	Número mágico 59a66a95h *(RAS_MAGIC)*
4	width	Ancho de la imagen en pixels
8	height	Alto de la imagen en pixels
12	depth	Número de bits por pixel
16	length	Tamaño de los datos de la imagen en bytes, o cero
20	type	Tipo de codificación: 0 (RT_OLD) Obsoleto 1 (RT_STANDARD) Archivos estándares 2 (RT_BYTE_ENCODED) Archivos comprimidos en rle 3 (RT_FORMAT RGB) Archivos en formato RGB sin comprimir 4 (RT_FORMAT_TIFF) Archivo tipo TIFF 5 (RT_FORMAT_IFF) Archivo tipo IFF 65535 (RT_EXPERIMENTAL) Varios tipos de codificación experimentales
24	maptype	Tipo de mapa (color o escala de gris)
28	maplength	Longitud del mapa en bytes

El "*número mágico*" como es conocido el campo de identificación del formato, también nos indica el tipo de orden en que fue almacenado el archivo, así el número mágico vale en *MSB:59a66a95h*, y vale *956aa659h* en *LSB*. El .ras maneja 6 diferentes tipos de codificación, el tipo 0 se considera obsoleto. En los tipos 4 y 5 se usa el mismo tipo de codificación que el formato TIFF, el tipo 6 está reservado y no se usa en aplicaciones comerciales.

La codificación de los datos en monocromático usa un bit por pixel, ordenado de izquierda a derecha cada línea, en el caso de escalas de gris se puede almacenar directamente los bytes como en el monocromático o usar un mapa de grises (cada pixel representa una entrada del mapa), en el caso del imágenes en color se usa la codificación RGB o mapas de color. El método RLE es usado cuando el formato comprime la imagen.

1.5.1.6 PBM (Portable Bitmap Utilites)

Propietario: *Jef Poskanzer, en la Universidad Carnige Mellon*
Extensión: *.pbm, .pgm, .ppm*

El formato PBM es parte de un grupo de utilerías gratuitas, para el manejo y conversión de imágenes creados en Unix por *Jef Poskanzer*, las utilerías permiten aplicar a una imagen corrección gama, difuminación, detectar líneas de color, recortar, etc., además permite convertir imágenes de un formato a otro, para realizar tales operaciones es necesario convertir la imagen del formato original (*GIF, TIFF, JPG, etc.*) al formato PBM necesario, y convertirlo a su vez al formato deseado. El formato PBM se subdivide en una serie de formatos especializados según el tipo de imagen que se trate, los cuales son: Para imágenes monocromáticas es el *Portable Bitmap for Monocrome bitmaps* (PBM), para imágenes en escalas de gris *Portable Grey Map* (PGM), y para imágenes en color está *el Portable Pixel Map*

(PPM), a los programas que pueden manejar los tres subformatos son referidos como programas PNM (*Portable aNy Map*), las utilerías aprovechan las "*Unix pipes*" - estructuras del sistema operativo especializadas en transferir datos- para lograr las conversiones directas sin necesidad de usar archivos temporales.

Estructura

Cada uno de los tres formatos inicia con una cabecera ASCII en la que se tiene un número "mágico" de identificación, además de unos parámetros en forma de cadena de dígitos enteros, cada uno de los parámetros está dividido por un caracter nulo (espacios, caracteres de tabulación, y de retorno de carro).

Para almacenar los datos se tienen dos variantes: pixels en ASCII y pixels en binario. Los pixels en ASCII son números ASCII separados por un espacio en blanco, para lograr la máxima portabilidad cada línea tiene como máximo 70 caracteres de largo, si los datos superan el limite se almacenan en la siguiente línea. Tanto la cabecera como los datos aceptan comentarios al final de la línea, los comentarios empiezan con caracter de número (#) de ahí el resto de la línea es ignorada.

Los pixels en binario varían según el tipo de imagen, que a continuación se mencionan.

Portable Bitmap for Monocrome

La cabecera se compone de tres campos:

Tabla 1.15: Cabecera del archivo PBM

Campo	Descripción
P1 / P4	Número Mágico en ASCII /Binario
width	Ancho de la imagen en pixels
height	Alto de la imagen en pixels

El valor del campo del número mágico indica que tipo de codificación es usada. Cuando se codifica valiéndose del ASCII el número mágico vale "P1"; inmediatamente después del campo Height se pone un caracter de retorno de carro, para representar cada pixel se tiene solo dos valores ASCII: "1" o "0" separados por un espacio en blanco, los pixels son almacenados y leídos de izquierda a derecha, y de arriba a abajo.

Cuando se usa el formato en binario, el número mágico vale "P4". Todos los datos se almacenan del mismo modo que en el ASCII, con la salvedad que los valores de la cabecera y los pixels son valores binarios y se almacenan en modo MSB.

Portable Grey Map

La cabecera se compone de los siguientes campos:

Tabla 1.16: Cabecera del archivo PGM

Campo	Descripción
P2 / P5	Número Mágico en ASCII /Binario
width	Ancho de la imagen en pixels
height	Alto de la imagen en pixels
max	Máximo valor de gris

En la versión ASCII del subformato, los pixels tienen un valor desde cero (negro) hasta un valor limite previamente establecido en la cabecera (blanco), la cabecera y los pixels son ordenados de la misma manera que el PBM, el número mágico vale "P2".

En la versión Binaria, donde todo se almacena en valores binarios, el número mágico vale "P4" y cada pixel vale un byte, por tal motivo el número de grises que alcanza la versión binaria se limita a 256, cosa que no ocurre con el ASCII, por lo demás, la imagen se codifica del mismo modo que en el PBM.

Portable Pixel Map

La cabecera se compone de los siguientes campos:

Tabla 1.17: Cabecera del archivo PPM

Campo	Descripción
P3 / P6	Número Mágico en ASCII /Binario
width	Ancho de la imagen en pixels
height	Alto de la imagen en pixels
max	Máximo valor del color componente

En la versión ASCII cada pixel se almacena en forma R,G,B, así el valor de negro es 0, 0, 0 hasta el valor de blanco máx, máx, máx; los valores de cada componente son separados por comas, y cada pixel se separa por medio de espacios en blanco. La cabecera se almacena de igual modo que los anteriores subformatos, los pixels son leídos de izquierda a derecha y de arriba a abajo.

La versión binaria sustituye cada valor de cada componente del pixel, por un byte, en este caso, el máximo valor de cada componente es de 255, limite que no tiene la versión ASCII, los pixels son guardados de izquierda a derecha y de arriba a abajo.

1.5.1.7 XBM (X Window bitmap)

Propietario: ***Consorcio X del Tecnológico de Massachusets*** **(MIT)**
extensión: ***.xbm***

El *consorcio X* es una organización que crea y propone normas, estándares para los ambientes gráficos en *Unix* (*X-window*). Para crear y manejar imágenes, iconos[11] y cursores (*punteros gráficos*).

Los X-Bitmaps son formatos muy poco comunes, ya que simultáneamente son bitmaps y código fuente de *lenguaje C*. Esto es a causa de que los bitmaps de iconos y cursores son compilados en Unix, estos bitmaps opcionalmente tiene un punto "caliente" (*hot spot*), es decir, un pixel de la imagen que es considerado la posición del cursor, por ejemplo, si el cursor tiene la forma de una flecha el punto "caliente" se sitúa comúnmente en la punta de la misma. Puede manejar colores (256 incluidos algunos grises) existen dos versiones: la X10 y la X11.

Estructura
El formato consiste en una serie de líneas de texto, las primeras definen el ancho y el alto de la imagen, y si ésta cuenta con un punto caliente; las siguientes dos líneas definen la posición X y Y respectivamente de tal punto, y son definidas como sigue:

```
#define nom_width ncols (ancho de la imagen)
#define nom_height nrows (alto de la imagen)
#define nom_x_hot xpos (Punto Caliente posición X)
#define nom_y_hot ypos (Punto Caliente posición Y)
```

Donde *nom* es el nombre de la imagen, *ncols* el número de columnas que tiene la imagen, *nrows* el número de renglones; *ncols, nrows, xpos, ypos* son números enteros; cabe mencionar que son definiciones de parámetros en C, además el formato no permite espacios en blanco fuera de los especificados, y se omite las líneas del punto caliente si no son usadas.

La siguiente línea define un arreglo estático en C que inicia el bloque de datos el cual es *static char nom_bits [ ] = {*, los datos se inician en la siguiente línea; cada renglón de datos no tiene limite de tamaño, cada byte del renglón es formateado como una constante hexadecimal *0xNN* de ocho bits (X11) o de 16 bits (X10) *0xNNNN*, siendo separado por comas, si el ancho de la imagen no es un factor de ocho habrá bits de desecho al final de cada renglón, en el último byte del renglón final, la coma se sustituye por la terminación "};". El formato es poco flexible, ya que no permite más espacios de los especificados, tampoco permite la existencia de renglones vacíos en el cuerpo del formato, ni acepta comentarios de ningún tipo (el nombre obviamente no acepta espacios en blanco). Por su estructura no tiene problemas con el orden de byte.

[11] Imagen de tamaño reducido que sirve como " *ancla* " para poder activar una aplicación, dentro de los diversos ambientes gráficos de los sistemas operativos X-Windows..

Un típico archivo XBM se ilustra con el siguiente ejemplo: Una imagen X11 de 16 x 16 pixels con un punto caliente en (4,4):

```
#define ejemplo_width 16
#define ejemplo_height 16
#define ejemplo_x_hot 4
#define ejemplo_y_hot 4
static char ejemplo_bits [ ] = {
   0x00,  0x00,  0x7c,  0x00,  0x82,  0x00,  0x82,  0x00,
   0x82,  0x00,  0x00,  0x00,  0x00,  0x00,  0x00,  0x00,
   0x00,  0x00,  0x00,  0x00,  0x00,  0x00,  0x00,  0x00,
   0x00,  0x00,  0x00,  0x00,  0x00,  0x00,  0x00,  0x00 };
```

En el caso de versión X10 del ejemplo anterior la única diferencia se aprecia en el formato de los datos, por ejemplo, la primer línea de datos del ejemplo anterior se vería así:

```
0x0000, 0x7c00, 0x8200, 0x8200,
```

1.5.1.8 XWD (X Window Dump)

Propietario: ***Consorcio X del Tecnológico de Massachusets*** **(MIT)**
extensión: ***.xwd***

El sistema *X-window* incluye las utilerías ***xwd*** que vacían el contenido gráfico de la pantalla o ventana a un archivo (xwd) y las utilerías ***xwud*** que despliegan lo capturado. Este formato consiste de una cabecera, un mapa de color opcional y los datos de la imagen. Existen dos variantes la X10 (actualmente en desuso) y la X11. Ya que la intención del consorcio X es hacer que sus recomendaciones sean lo más independientes del hardware como sea posible[12]. Existe un gran número de maneras de formatear los datos, según el número de tipos de pantalla. Dos maneras sobresalen: Las Clases Virtuales (***virtual class***) y el formato de imagen (***image format***).

Clases virtuales
Las clases virtuales están proyectadas para caracterizar varios tipos de controladores en los cuales el sistema X-Window debe correr. Las clases virtuales reflejan en donde la pantalla apoya el color, en donde la pantalla tiene un mapa de color fijo o variable, y donde los mapas de color se unen o se separan. Nótese que el número de colores no es parte de la clase.

StaticGray (código 0) - Para pantallas blanco y negro con un mapa de color fijo. De un bit por pixel es una pantalla monocromática, este es relativamente común en los monitores blanco y negro.

[12] La especificación X10 trabaja en 16 bits, en cambio la especificación X11 trabaja en 8 bits, por lo tanto abarca un mayor número de plataformas, siendo en este caso una ventaja para los objetivos del proyecto X-Window.

GreyScale (código 1) - Para pantallas blanco y negro con software que controla los tonos de gris de cada pixel, es poco común.

StaticColor (código 2) - Para pantallas de color con mapas fijas de color similares a las pantallas CGA de IBM, también poco usado.

PseudoColor (código 3) - Para pantallas de color con software que controla los tonos de color de los pixels, es similar al tipo VGA de IBM, la profundidad del pixel es de 4 o 8 bits.

TrueColor (código 4) - Para pantallas de color con un mapa fijo RGB, esta clase es de alta definición alcanzando 24 bits de profundidad.

DirectColor (código 5) - Para pantallas de color con software que mapean los colores Rojo, Verde, Azul directamente en la pantalla, y cada color es mapeado por separado.

Las clases por ser recomendaciones pueden no representar el hardware del monitor en que están trabajando, es decir, pueden presentar una imagen de una clase inferior con una clase superior manteniendo compatibilidad hacia atrás.

Formato de imagen

Cuando una imagen requiere más de un bit por pixel, hay múltiples maneras para guardar los pixels. Para esto, X-Window provee los formatos de imagen.

XYBitmap (código 0) - Una imagen monocromática con bits de cada renglón empacados en palabras del tipo de la maquina usada. Son parámetros el tamaño de palabra y el orden de byte de la misma (es conocido que Unix opera tanto en máquinas que usan MSB ó LSB).

XYPixmap (código 1) - Una imagen en escala de gris donde los pixels son almacenados en planos de bits, por ejemplo, si la imagen tiene cuatro bits por pixel, está es almacenada en cuatro distintas imágenes una por cada bit. Este método es muy ineficiente para manipularse, pero es altamente eficiente para guardar y desplegar imágenes en algunos monitores tales como el VGA de IBM.

Zpixmap (código 2) - Una imagen en escala de gris o color, donde los pixels de cada renglón son guardados en planos de bits.

Estructura

La cabecera de un formato *XWD* es como sigue:

Tabla 1.18: Cabecera de un archivo *XWD*

Offset	Nombre	Descripción
0	header_size	Tamaño de la cabecera
4	file_version	Versión "7"
8	pixmap_format	Formato de la imagen
12	pixmap_depth	Profundidad de la imagen

Tabla 1.18: Cabecera de un archivo *XWD (Continuación)*

Offset	Nombre	Descripcion
16	pixmap_width	Ancho de la imagen
20	pixmap_height	Alto de la imagen
24	xoffset	Imagen x Offset
28	byte_order	Orden del byte: MSB = 1, LSB = 0
32	bitmap_unit	Unidades del bitmap 8, 16, 32
36	bitmap_bit_order	Orden del byte de la imagen: MSB = 1, LSB = 0
40	bitmap_pad	Bloque de la imagen
44	bits_per_pixel	Bits por pixel
48	bytes_per_line	Bytes por línea
52	visual _class	Clase de mapa de color
56	red _mask	Mascara Roja
60	green_mask	Mascara Verde
64	blue_mask	Mascara Azul
68	bits_per_rgb	Bits por pixel lógico
72	colormap_entries	Número de entradas en mapa de color
76	ncolors	Número de estructuras de color
80	window_width	Ancho de la ventana
84	window_heigth	Alto de la ventana
88	window_x	Coordenada X superior izquierda de la ventana
92	window_y	Coordenada Y superior izquierda de la ventana
96	window_bdrwidth	Ancho del borde de la ventana
100	name	Nombre de la imagen, texto de terminación

Todos los campos excepto el nombre de la imagen son enteros de 4 bytes, almacenados en formato *MSB (Big-endian)*. El campo *header_size* indica el tamaño de la cabecera en bytes incluyendo el nombre. El campo *file_version* vale 7 para la versión X11 del formato. El *pixmap_format* es uno de los formatos de imagen descritos anteriormente. El *pixmap_format, byte_order, bitmap_unit, bitmap_bit_order, bitmap_pad,* y los campos *mask* describen la organización de los bits de la imagen, se verá más adelante esto.

El *xoffset* sirve para alinear la imagen a limites de direccionamiento, sin afectar a la ventana en uso. Los campos window documentan el tamaño de la imagen y su posición original en la pantalla, aunque la imagen no necesita ser desplegada en la misma posición, en el caso del ancho y alto de la ventana suele ser el mismo al de la imagen, con la excepción de imágenes que sean enmarcadas en ventanas. El campo de nombre de la imagen es una cadena ASCII opcional que documenta el nombre original de la ventana, para determinar su tamaño se debe restar 100 de la longitud total de la cabecera en *header_size*.

Mapa de color

El campo *ncolors* indica el número de entradas que tiene el mapa de color (*o gris*). Cada entrada de color tiene 12 bytes de largo (véase la tabla 1.19). Las imágenes con las clases virtuales *GreyScale, PseudoColor y DirectColor* deben forzosamente usar mapas de color, mientras que las restantes clases pueden usar los mapas de manera opcional *(generalmente no lo hacen).*

Tabla 1.19: Entradas de los mapas de color *XWD*

Offset	Tamaño	Nombre	Descripción
0	4	pixel	Número de entradas (cantidad de colores a tratar)
4	2	red	Valor de rojo
6	2	green	Valor de verde
8	2	blue	Valor de azul
10	1	flags	No usado
11	1	pad	Bloque

Los valores rojo, verde, azul son números de 16 bits sin signo, y los valores están normalizados a 65535 que representa la máxima brillantez. Cada entrada contiene un explícito número de entradas que identifica el valor del pixel (o los valores de los componentes del color para DirectColor).

Los datos de la imagen son almacenados en renglones - la imagen se almacena de izquierda a derecha y de arriba a abajo -. Cada renglón se compone de una secuencia de "unidades". El tamaño de cada unidad está documentado en el campo bitmap_unit de la cabecera, tal valor puede ser de 8, 16 ó 32 bits de largo, el tipo de orden en que se almacenan los bytes se determina según el valor (1 = MSB, 0 = LSB). El orden de los bits de los pixels se indica por separado, así, la imagen y el archivo pueden estar almacenados en orden de bytes diferentes. En la cabecera también se especifica el número de bits por pixel (la profundidad de la imagen).

El campo bitmap_pad indica el número de repeticiones de cada unidad, el valor de bitmap_pad es un múltiplo del valor en el campo bitmap_unit. Para las clases virtuales TrueColor y DirectColor los campos *mask* son bits que corresponden a cada uno de los colores RGB. Con tal cantidad de parámetros hay una enorme cantidad de variantes de imagen, aunque algunos tipos son más populares, es necesario revisar que variante soporta el sistema Unix dado.

1.5.1.9 FITS (Flexible Image Transport System)

Propietario: ***Grupo de trabajo FITS de la comisión 5 de la Unión Astronómica Internacional.***
Extensión: ***.fits, .fit***

El formato FITS (Flexible Image Transport System) es el formato universalmente usado por los astrónomos para almacenamiento y traslado de imágenes astronómicas. Originado a finales de los 70, rápidamente fue adoptado para todas las aplicaciones astronómicas. Diseñado originalmente para cintas magnéticas, ha sido extendido para soportar disquetes y CDROM's. Hay una gran diversidad de extensiones al formato FITS básico original, pero ninguna de ellas almacena imágenes.

Estructura
Un archivo FITS es una secuencia de registros de 2880 bytes de largo (el valor es el múltiplo común de los tamaños de palabra de las computadoras más populares). El

archivo inicia con una cabecera de registros, cada uno de los cuales contiene 36 etiquetas en ASCII de 80 bytes, y opcionalmente algún número de registros de datos. Los datos en el formato FITS básico es un arreglo n dimensional, en los archivos que contienen imágenes, el arreglo es de 2D con datos de imágenes en escalas de gris, o 3D con conjuntos de imágenes en escalas de gris.

Los registros de la cabecera son tratados como un conjunto de etiquetas en ASCII de 80 bytes en forma fija. Si el número de etiquetas no es un múltiplo de 36 la última etiqueta es rellenada hasta los 2880 bytes con caracteres en blanco.

Cada etiqueta contiene una palabra llave en la columna uno. La llave consiste de caracteres letras mayúsculas, dígitos, guiones, y el caracter de subrayado. Si la etiqueta asigna un valor a la llave, la columna nueve debe tener un signo de igual (=). Los valores deben estar encerrados entre comillas simples (*'*), las comillas se abren en la columna 11, el valor debe tener al menos ocho caracteres de largo, excluyendo las comillas, en caso de ser menos caracteres se añaden espacios en blanco. Los caracteres de lógica T (*verdadero)* y F *(falso)* deben estar en la columna 30. Los valores numéricos deben estar con alineación hacia la derecha y el último dígito de la cifra debe estar en la columna 30.

Para los valores complejos, la parte imaginaria debe estar alineada a la derecha también y el último dígito debe estar en la columna 50. Los valores enteros pueden tener cadenas de dígitos con signo. En valores de punto flotante, los valores deben tener un punto decimal y opcionalmente un exponente con signo siguiendo a una "D" o "E". El formateo de las cifras se debe a su compatibilidad con el lenguaje de programación FORTRAN 77. En el caso de los comentarios estos empiezan al final en las etiquetas con valores, o en la columna 10 en las etiquetas sin valores, el caracter slash (*/*) indica el inicio de los comentarios.

Etiquetas base

Cada archivo FITS debe contener las etiquetas base SIMPLE (etiqueta de inicio), BITPIX, NAXIS y END (etiqueta de terminación) en ese orden, aunque otros registros están mezclados entre ellos. Si el archivo contiene datos, la etiqueta NAXIS debe ser seguida por etiquetas NAXISi.

Las etiquetas base contiene los siguientes parámetros:

SIMPLE: Si el valor es 'T, el archivo es un FITS básico; si el valor es XTENSION, el archivo tiene extensiones además de arreglos de datos n dimensionales.

BITPIX: Los valores posibles en la etiqueta son 8, 16, 32, -32 o -64. Cuando los valores son 8, 16, 32, los datos son enteros con signo de uno, dos, o cuatro bytes respectivamente en formato MSB. En caso de valer -32 o -64, los datos son números de punto flotante en simple o doble precisión acorde con el IEEE, el primer bit es para el signo los siguientes 8 (simple) o 11 (doble) bits para el exponente, y el resto de los bits para la fracción. Los valores definidos por el IEEE pueden incluir valores infinitos.

NAXIS: El valor es el número de dimensiones en el arreglo de datos. Si vale cero, no hay datos que sigan.

NAXISi: Cuando NAXIS es mayor de cero, el valor i se sustituye por el número para formar las etiquetas NAXIS1, NAXIS2, NAXIS3, NAXIS4, y así sucesivamente-.

END: La etiqueta de terminación, no tiene valores.

ETIQUETAS OPCIONALES

Los archivos FITS pueden tener una serie de etiquetas opcionales, las cuales tienen la misma estructura que las etiquetas base, pueden ser ignoradas si el programa lector no los entiende, algunas de estas etiquetas están definidas por el estándar FITS. Las etiquetas opcionales relacionadas con el bitmap son:

Tabla 1.20: Etiquetas del formato FITS

Etiqueta	Explicación
BSCALE	Un factor que es usado en combinación de BZERO.
BZERO	Una constante que es añadida a los datos. La formula es: valor actual = *BZERO + BSCALE* * valor del archivo.
BUNIT	Una cadena que expresa la unidades en que está almacenada la imagen.
BLANK	Un valor numérico que representa valores desconocidos o en blanco. Es permitido únicamente valores enteros.
CTYPEi	Una cadena que nombra al eje dado i.
CRPIXi	Punto de referencia de un eje y, en unidades del eje i.
CRVALi	Valor de las unidades del eje y. Está relacionada con la etiqueta anterior.
CDELTi	Cantidad en la que las unidades se incrementan en el eje i.
CROTAi	Rotación en grados con respecto a un sistema coordinado estándar del eje i.
DATAMIN	El mínimo valor físico válido en el archivo.
DATAMAX	El máximo valor físico válido en el archivo.
EXTEND	Si el valor es T (verdadero). Las siguientes etiquetas son añadidas
DATE	Cadena alfanumérica que indica la fecha de creación del archivo, en formato 'DD/MM/YY'
ORIGIN	cadena que indica la organización que creo el archivo.
BLOCKED	Si vale T, es almacenado en un medio físico. (obsoleto)
DATE-OBS	Fecha de observación de la que se obtuvo la imagen, en igual formato que la etiqueta DATE.
TELESCOP	Identifica el telescopio usado.
INSTRUME	Identifica el instrumento usado con el telescopio (espectrógrafos, cámaras, etc.).
OBSERVER	Identifica al observador
OBJECT	Nombre del objeto observado.
EQUINOX	Número en punto flotante que expresa el equinoccio en años para el sistema coordinado
EPOCH	Un valor equivalente a EQUINOX (obsoleto).
AUTHOR	El nombre del autor.
REFERENC	Referencia bibliográfica
COMMENT	Comentario de libre extensión y formato.
HISTORY	Comentario acerca de la historia de la creación de la imagen.
BLANK KEYWORD	Si una etiqueta tiene los primeros 8 caracteres en blancos.

Datos de la imagen
Los valores inmediatamente después de la cabecera pertenecen a la imagen. Los datos están ordenados en forma de arreglo de acuerdo a la estructuración de los mismos en FORTRAN, con los subíndices variando rápidamente, los datos están empacados en registros de 2880 bytes sin importar la cantidad de estos. En caso de que el último registro no ocupe los 2880 bytes, estos son rellenados con ceros.

1.5.1.10 BMP/DIB

Propietario: *Microsoft Co.*
Extensión: *.bmp .dib*

Dentro de las actividades conjuntas que realizaron IBM y Microsoft en el desarrollo y mejoramiento de los sistemas operativos MS-Windows y OS/2, crearon el formato gráfico "Microsoft Device Independent Bitmap" (BMP/DIB). El formato BMP es ajustado para las necesidades de Ms-Windows, mientras el DIB hace lo propio en OS/2. Ambos formatos están codificados en LSB. El formato BMP/DIB abarca imágenes con una profundidad de 1, 4, 8, 24 bits (monocromáticas, escala de grises y color); las imágenes de hasta 8 bits usan mapas de colores, mientras que las imágenes de 24 bits son de color RGB puro.

Estructura
Cualquier variante del formato debe comenzar con una cabecera general, también llamada BITMAPFILEHEADER.

Tabla 1.21: Cabecera general del formato BMP/DIB

Offset	Tamaño en bytes	Nombre del campo	Descripción
0	2	bfType	Etiqueta ASCII de identificación. La etiqueta es "BM".
2	4	bfSize	Tamaño del archivo.
6	2	bfReserved1	Reservado a futuro, vale cero.
8	2	bfReserved2	Reservado a futuro, vale cero.
10	4	bfOffbits	Localización del byte (Offset) de inicio de los datos de la imagen.

El campo bfOffbits es de uso obligado debido a que las cabeceras de cada versión tienen diferentes longitudes. La cabecera para OS/2 tiene 12 bytes de longitud, mientras que la de MS-Windows cuenta con 40 bytes.

Cabecera para MS-Windows
La siguiente cabecera contiene los datos de la imagen, y opcionalmente, los del mapa de color. Conocida como BITMAPINFOHEADER en la documentación, se compone de los siguientes campos:

Tabla 1.22: Cabecera BMP/DIB para MS-Windows

Offset	Tamaño en bytes	Nombre del campo	Descripción
14	4	biSize	Tamaño de la cabecera: 40 bytes.
18	4	biWidth	Ancho de la imagen en pixels.
22	4	biHeight	Alto de la imagen en pixels.
26	2	biPlanes	Número de planos. Debe valer 1.
28	2	biBitCount	Bits por pixel. 1, 4, 8 o 24.
30	4	biCompression	Tipo de compresión. Si biBitCount =1 no hay compresión. En caso contrario se usa compresión RLE.
34	4	biSizeImage	Tamaño en bytes de la imagen comprimida.
38	4	biXPelsPerMeter	Resolución horizontal en pixels/metro.
42	4	biYPelsPerMeter	Resolución vertical en pixels/metro.
46	4	biClrUsed	Número de colores usados.
50	4	biClrImportant	Número de colores importantes.
54	4	N*bmiColors	Mapa de color

Mapa de color

Las imágenes de 1, 4, 8 bits por pixel deben usar un mapa de color. Los tamaños de tales mapas son normalmente 2, 16, 256 respectivamente,. Los mapas pueden ser más pequeños si no se usan todas las entradas del mapa. Si el campo *biClrUsed* es diferente de cero, contiene el número de colores usados, que es lo mismo al número de entradas del mapa de color. Si vale cero el mapa es usado plenamente. En imágenes de 24 bits la codificación se hace directamente.

A causa de que algunos dispositivos no pueden desplegar tantos colores como requiere la imagen, las entradas del mapa se ordenan en base a su importancia relativa en la imagen. El campo *biClrImportant* indica cuantos de esos colores son importantes para una buena reproducción de la imagen. Las entradas del mapa de color son de 4 bytes cada una y se componen de los siguientes campos:

Tabla 1.23: Entradas de color para MS-Windows

Offset	Nombre del campo	Descripción
0	rgbBlue	Valor del Azul.
1	rgbGreen	Valor del Verde.
2	rgbRed	Valor del Rojo.
3	rgbReserved	Reservado vale cero.

Datos del bitmap

Los datos de la imagen siguen inmediatamente al mapa de color. Los datos pueden estar sin comprimir, y para las imágenes de 2, 4, 8 bits de profundidad se puede usar también la compresión RLE. Los bits son guardados por renglón.

Bitmap de un bit por pixel: Cada pixel es un bit, entonces, hay ocho por byte.

Bitmap de cuatro bits por pixel: En imágenes sin comprimir los bytes llevan dos pixels por byte. Para imágenes comprimidas se usa compresión RLE (*véase compresión*). Una característica especial es el uso de una secuencia de terminación de renglón, esta secuencia es el valor hexadecimal "00 00".

Bitmap de ocho bits por pixel: Sin comprimir un pixel ocupa un byte. La compresión es la misma que en las imágenes de cuatro bits.

Bitmap de 24 bits por pixel: Cada pixel se compone de tres bytes; uno para el rojo, otro para el verde, y uno para el azul. Para indicar el fin de renglón se usan cuatro bytes en blanco.

Cabecera OS/2

La cabecera de la versión OS/2 es similar (y más simple) a la de MS-Windows. La cabecera es conocida también como BITMAPCOREINFO, y tiene los siguientes campos:

Tabla 1.24: Cabecera BMP/DIB para OS/2

Offset	Tamaño en bytes	Nombre del campo	Descripción
14	4	bcSize	Tamaño de la cabecera: 12 bytes.
18	2	bcWidth	Ancho de la imagen en pixels.
20	2	bcHeight	Alto de la imagen en pixels.
22	2	bcPlanes	Número de planos. Debe valer 1.
28	2	bcBitCount	Bits por pixel. 1, 4, 8 o 24.
54	3	N*bmcColors	Mapa de color

Las imágenes son similares al de la versión MS-Window. El mapa de color se compone de tres bytes cada entrada, Los datos de la entrada se llaman RGBTRIPLE y son:

Tabla 1.25: Entradas de color para OS/2

Offset	Nombre del campo	Descripción
0	rgbBlue	Valor del Azul.
1	rgbGreen	Valor del Verde.
2	rgbRed	Valor del Rojo.

Datos de la imagen OS/2

Los datos de la imagen siguen inmediatamente a la cabecera. Cada renglón es llenado con ceros en un múltiplo de 4 bytes. La codificación de los bits es idéntica a la versión de windows. La versión DIB no soporta compresión RLE.

1.5.1.11 GIF

Propietario: *Compuserve Inc.*
Extensión: *.gif*

El GIF es uno de los formatos más exitosos actualmente. Soportado por casi todas las aplicaciones gráficas de todas las plataformas, es además el formato base de los programas *World Wide Web* (WWW)[13], tales como Netscape, Microsoft Explorer, lenguajes como Java y Visual Basic. Actualmente, una característica de este formato que está revolucionando la manera de ver y usar al WWW es la capacidad del formato para almacenar y desplegar múltiples imágenes, en una secuencia animada, Esto hace más atractivo e interactivo el uso del internet. Sin embargo, este formato se encuentra en una fuerte discusión legal y ética, de la cual hablaré más adelante.

El formato GIF fue presentado originalmente en 1987, para 1989 se realizó la única revisión del formato hasta la fecha. El formato se compone de bloques y sub-bloques que indican una serie de parámetros y datos usados para reproducir una imagen. El archivo .gif está diseñado para almacenar más de una imagen. Las tablas de color son de dos tipos: Local y Global, la local se refiere a los colores usados por una imagen en particular, mientras que la global contiene colores asociados a toda la extensión de datos del archivo, es decir, todas las imágenes almacenadas. Ambas tablas son opcionales, y en el caso de una sola imagen es normal descartar la tabla global. Es necesario el uso de la tabla global en archivos que contengan múltiples imágenes debido a la limitada cantidad de colores que algunos monitores pueden desplegar, y además reduce el tamaño del archivo.

Bloques, Extensiones y Alcances
Los bloques pueden ser clasificados en tres grupos: Control, Desplegado gráfico y Propósito Especial. Los bloques de control contienen información para controlar el procesamiento de los datos o información ajustando los parámetros del hardware usado; ejemplo de estos son la cabecera, el descriptor de la ventana lógica, la tabla global de colores, la extensión de control gráfico y el terminador.

Los bloques de desplegado gráfico contienen la información para desplegar la(s) imagen(es) en cuestión. De estos podemos mencionar al descriptor de la imagen y la extensión de texto.

Los bloques de propósito especial conjuntan a todos aquellos bloques que no tienen nada que ver con el procesamiento y desplegado de la imagen. De ellos se destaca la extensión de comentarios y la extensión de aplicaciones. Con excepción del descriptor de la ventana lógica y la tabla global de colores, cuyos datos afectan a todas las imágenes incluidas en el archivo, los demás bloques de control tienen un alcance

[13] El World Wide Web es una serie de servicios computacionales por medio de los cuales, individuos u organizaciones pueden enviar y recibir información de toda clase sin importar la zona de origen o el momento en que se originó.

circunscrito a los bloques de desplegado que les siguen. Los bloques de propósito especial no intervienen de modo alguno en el proceso de decodificación.

Las etiquetas de los bloques se acomodan en los siguientes rangos: 00h a 7Fh es para bloques de desplegado gráfico con excepción del terminador (3Bh); 80h a F9h son bloques de control; de FAh a FFh lo ocupan los bloques de propósito especial. Estos rangos permiten identificar a que tipo pertenece un bloque que no pueda ser procesado. Los valores de los bloques y extensiones aceptados en versión 89a. son enumerados a continuación.

Tabla 1.26: Estructura del archivo GIF

Nombre del bloque	Etiqueta en hexadecimal
Cabecera	No tienen etiqueta
Descriptor de la ventana lógica	
Tabla Global de Colores	
Tabla Local de Colores	
Bloque de despliegado gráfico	
Extensión de texto	01
Descripción de la imagen	2C
Bloques de Control	
Extensión de Control gráfico	F9
Bloques de Propósito Especial	
Teminador	3B
Extensión de Comentarios	FE
Extensión de Aplicaciones	FF

Cada bloque tiene un campo que cuenta el número de bytes de que se compone un bloque excluyendo al terminador y el campo de conteo mismo. Los bloques de datos tienen tamaño variable que se ajusta para acomodar los datos, no así los demás bloques que son de longitud fija. Todos los bloques se introdujeron en la versión 87a. con excepción de los expresamente indicados.

Sub-bloques de datos

Contienen datos de la imagen. carecen de etiquetas, los sub-bloques son procesados en el contexto de los bloques de control. El primer byte indica el tamaño del sub-bloque que va de 0 a 255 bytes. Los datos son valores de 8 bits de extensión.

Cabecera

En este bloque se identifica al archivo GIF. Debe ser siempre el primer bloque en aparecer y consta de dos etiquetas de tres bytes cada uno. La primer etiqueta tiene el valor " *GIF* " necesariamente. El segundo contiene la versión usada, " 87a " (mayo 1987) o " 89a " (julio 1989).

Descriptor de la ventana lógica

Contiene los parámetros necesarios para definir el área del dispositivo de desplegado en la cual las imágenes serán mostradas. Las coordenadas están dadas desde el punto superior izquierdo de la ventana virtual, y no necesariamente se refieren a las coordenadas absolutas del dispositivo. Esto implica que las coordenadas deben ser referencias a coordenadas de ventana en un sistema gráfico, o coordenadas de

impresión cuando sea usada una impresora. El bloque es obligatorio y consta de los siguientes campos:

Tabla 1.27: Estructura de la ventana logica

Etiqueta	Tamaño en bytes	Explicación
Logical Screen Width	2	Ancho en pixels de la ventana lógica
Logical Screen Height	2	Alto en pixels de la ventana lógica
<Packed Fields>	1	Es una serie de campos y banderas que ocupan un byte. Contiene los siguientes campos: Global Color Table Flag (1 Bit): Indica si hay una tabla global de colores. 0= no hay tabla, 1= si hay tabla. Color Resolution (3 Bits): Número de bits disponibles por color primario en la imagen original menos uno. Sort Flag (1 Bit): Orden en el cual los colores son distribuidos, basado en las frecuencias de los mismos comenzando por el más usado en adelante. 0 = desordenado, 1= ordenado. Size of Global Color Table (3 Bits): Tamaño en bytes de los índices de la tabla global de colores.
Background Color Index	1	*Índice en la tabla para el color de fondo.* El color de fondo son aquellos colores no usados por una imagen. Es ignorado si vale 0. Cobra importancia si el bit de la Global Color Table Flag es uno.
Pixel Aspect Ratio	1	Factor usado para computar una aproximación del tamaño que tuvieron los pixels en la imagen original. Este valor va de 0 a 255. Si su valor es mayor a 1, se usa la formula: Tamaño original = (valor del campo Pixel Aspect Ratio +15)/64.

Tabla global de colores

Este bloque contiene una tabla de colores RGB. Esta tabla es usada tanto por imágenes que carecen de una tabla local de colores como por la extensión de texto. Este bloque es opcional, si está presente es indicado en la bandera correspondiente en el Descriptor de la ventana lógica, y se encuentra inmediatamente después del descriptor. El tamaño del bloque es calculado de la siguiente manera:

$$3 * 2^{(\text{Tamaño de la tabla global de color}+1)}.$$

Descripción de la imagen

Cada imagen en el bloque de datos se compone de una descripción de la imagen, una tabla local de colores y los datos de la imagen. La descripción de la imagen (image descriptor) es un bloque que está compuesto por los parámetros necesarios para procesar una imagen. Las coordenadas en este bloque están dadas en pixels e inscritas a la ventana lógica. Este bloque pertenece al grupo de desplegado gráfico, opcionalmente puede estar precedido por uno o varios bloques de control, y opcionalmente puede ser seguido de una tabla local de colores y siempre por los datos de una imagen. Este bloque se presenta sólo si hay una imagen, existiendo uno

por una imagen y es necesario para poder desplegarla. Puede haber tantos en el archivo como imágenes contenga este.

Este bloque se compone de los siguientes campos:

Tabla 1.28: Estructura del Descriptor de la Imagen

Etiqueta	Tamaño en bytes	Explicación
Image separator	1 byte	Indica el inicio del bloque. Contiene el valor fijo 2Ch
Image Left Position	2 bytes	La posición de la columna del limite izquierdo de la imagen, con respecto de la posición de la columna del limite izquierdo de la ventana lógica (que vale 0), en pixels.
Image Top Position	2 bytes	La posición del renglón del limite superior de la imagen, con respecto de la posición del renglón del limite superior de la ventana lógica (que vale 0), en pixels.
Image Width Position	2 bytes	Ancho de la imagen en pixels.
Image Height Position	2 bytes	Alto de la imagen en pixels.
<Packed Fields>	1 byte	Contiene las siguientes banderas y campos: Local Color Table Flag (1 Bit), Interlace Flag (1 Bit), Sort Flag (1 Bit), Reserved (2 Bits), Size of Local Color Table (3 Bits).

Los datos comprimidos (Packed Fields), son un grupo de campos que por su tamaño sólo necesitan un byte. En este caso los campos son:

Local Color Table Flag: Bandera que indica la presencia de una tabla local de colores de un bit de extensión. Si vale 1, la tabla está presente y se localiza inmediatamente después del bloque y antes de los datos.

Interlace Flag: Bandera de entrelazado, si vale 1 indica que la imagen está entrelazada. El entrelazado es una secuecia de guardado, la cual agrupa renglones separados, y conforme va siendo desplegada la imagen es posible verla, aunque con una resolución baja. Conforme son siendo trasmitidos y desplegados los grupos, va mejorando la calidad, hasta que la imagen es completada.

El entrelazado usado por el GIF es de 4 pasadas (grupos) ordenados de la siguiente manera:

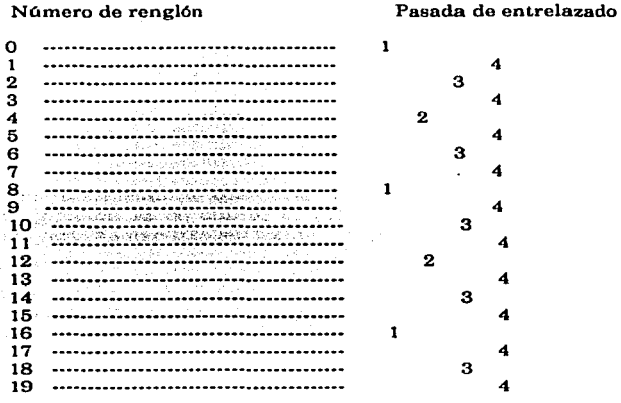
Grupo 1: Cada *octavo* renglón comenzando con el renglón 0.

Grupo 2: Cada *octavo* renglón comenzando con el renglón 4.

Grupo 3: Cada *cuarto* renglón comenzando con el renglón 2.

Grupo 4: Cada *segundo* renglón comenzando con el renglón 1.

Con el siguiente ejemplo se ilustra lo anterior:



Con este ejemplo se puede ver gráficamente el mejoramiento continuo de la calidad de la imagen. Casi siempre basta con dos pasadas para obtener una idea general de la imagen, esto es muy importante cuando la transmisión de la imagen es muy lenta.

Sort Flag: Esta bandera de un bit indica si la tabla local de colores está ordenada. Si está ordenada (vale 1 la bandera), lo está en orden decreciente comenzado por el color más usado en la imagen.

Reserved: Reservado para futuras revisiones al GIF. debe valer 0.

Size of Local Color Table: Si hay una tabla local de colores, este campo contiene el valor de dicha tabla. Para calcular su tamaño en bytes, simplemente se eleva al cuadrado el valor más uno.

Tabla local de colores

Es similar en todos lo aspectos a la tabla global de colores - incluyendo en la manera de calcular su tamaño- (véase la tabla global de colores). La diferencia radica en que la tabla local afecta a la imagen en que se inscribe, mientras que la tabla global afecta a más de una imagen. La tabla local tiene preferencia sobre la global. La tabla usada por la imagen se le denomina activa independientemente de que sea global o local.

Tabla de datos de la imagen
Los datos de la imagen están divididos en bloques de máximo 255 bytes, cada pixel es una entrada de la tabla activa (local o global) de colores. Los pixels están ordenados de izquierda a dererecha y de arriba a abajo, todos los bloques están comprimidos con el algoritmo LZW. Se reserva un byte al principio de la tabla que es conocido como "tamaño de código mínimo LZW" en el que se indica el número de bits usados como datos iniciales para el algoritmo (véase compresión LZ78).

Terminador
Indica el fin de la información del archivo GIF. Ocupa un byte con el valor fijo 3Bh.

Extensiones
En la versión 89a del GIF fueron introducidas varias extensiones. Estas extensiones son bloques opcionales que permiten modificar atributos de la imagen, introducir comentarios, introducir texto en la imagen, etc. Todos las extensiones son localizadas según se marque, siempre antes del terminador.

Extensión de control gráfico
Contiene parámetros usados cuando es procesado un bloque de desplegado gráfico. Sin limite en su número, este bloque debe preceder a un bloque de desplegado gráfico.

Los campos que contiene esta extensión son:

Tabla 1.29: Estructura de bloque de control gráfico

Etiqueta	Tamaño en bytes	Explicación
Extensión Introducer	1 byte	Indica el inicio de una extensión. tiene el valor 21h.
Graphic Control Label	1 byte	Etiqueta de identificación del bloque (F9h).
Block Size	1 byte	Número bytes del bloque debe valer 4 (no incluye cabecera, ni terminador).
<Packed Fields>	1 byte	*Reserved (3 Bits),* reservado para uso futuro. *Disposal Method (3 Bits)* Indica que debe hacer el decodificador una vez que desplegó la imagen. 0: No hace nada. 1: Deja la imagen sin alterar. 2: Restaura el color de fondo en el área de la imagen. 3: Restaurar el área de la imagen al estado en que estaba antes de ser desplegada. 4 al 7: Reservado a futuro. *User Input Flag (1 Bit)* Indica que debe esperar a una acción del usuario para continuar. La acción es definida por el decodificador (dar enter, teclear, dar clic con el mouse, etc.). *Transparent Color Flag (1 Bit)*Indica que hay un color transparente (de igual color del fondo).

Tabla 1.29: Estructura de bloque de control gráfico (Continuación)

Etiqueta	Tamaño en bytes	Explicación
Delay Time	2 bytes	Tiempo de espera que realiza la maquina antes de continuar con el proceso, su valor está dado en centésimas de segundo. Si la bandera user input flag está activa, el programa reanudara el proceso con lo que ocurra primero.
Transparent Color Index	1 byte	Valor de la entrada del color que será transparente. En caso de estar activa la bandera, el color de la imagen toma el color(es) del fondo. Esto permite a una imagen evitar la forma "cuadrada", dando un aspecto más natural.
Block Terminator	1 byte	Este bloque de valor cero indica el fin del bloque de control.

Extensión de comentarios

La extensión de comentarios contiene información de texto, como comentarios, créditos, descripciones, etc.; el cual no es parte de la(s) imagen(es). Los campos que tiene son:

Tabla 1.30: Extensión de Comentarios en GIF 89a

Etiqueta	Tamaño en bytes	Explicación
Extensión Introducer	1 byte	Indica el inicio de una extensión. tiene el valor 21h.
Comment Label	1 byte	Etiqueta de identificación del bloque vale Feh.
Comment Data	sub-bloques	Divididos en bloques de datos de texto ASCII de 7 bits, con tamaño de 1 a 255 bytes con un byte que indica su longitud al inicio de cada sub bloque.
Block Terminator	1 byte	Este bloque de valor cero indica el fin del bloque.

Extensión de texto

Contiene texto "plano" (sin características como **Bold**, *Itálico*, etc.) y los parámetros necesarios para desplegar ese texto como un gráfico. El texto se compone de caracteres ASCII de 7 bits imprimibles (no son de control). Los caracteres son desplegados en una red de celdas. El tamaño de las celdas está dado por los parámetros aquí contenidos. El número de celdas en la imagen es entera, eliminándose las fracciones de celda. Cada caracter ocupa una celda en forma monoespaciada. Necesita de la tabla global de colores para ser desplegado. Por ser un bloque de desplegado de imagen, puede ser modificado por una extensión de control gráfico. Los campos son:

Tabla 1.31: Formato de texto para GIF 89a

Etiqueta	Tamaño en bytes	Explicación
Extensión Introducer	1 byte	Indica el inicio de una extensión. tiene el valor 21h.
Plain Text Label	1 byte	Etiqueta de identificación del bloque (01h).
Block Size	1 byte	Número de bytes en la extensión. Tiene el valor fijo de 12.
Text Grid Left Position	2 bytes	Número de columnas (en pixels), al lado izquierdo de la red de celdas con relación de la imagen lógica.
Text Grid Top Position	2 bytes	Número de renglones (en pixels), al lado izquierdo de la red de celdas con relación de imagen lógica.

Tabla 1.31: Formato de texto para GIF 89a (Continuación)

Etiqueta	Tamaño en bytes	Explicación
Text Grid Width	2 bytes	Ancho de la red de celdas en pixels.
Text Grid Heigth	2 bytes	Alto de la red de celdas en pixels.
Caracter Cell Width	1 byte	Ancho de celda en pixels.
Caracter Cell Heigth	1 byte	Alto de celda en pixels.
Text Foreground Color Index	1 byte	Valor del color del texto.
Text Background Color Index	1 byte	Valor del color del fondo del área del texto.
Plain Text Data	sub-bloques	Divididos en bloques de datos de texto ASCII de 7 bits, con tamaño de 1 a 255 bytes con un byte que indica su longitud al inicio de cada sub-bloque.
Block Terminator	1 byte	Esta etiqueta de valor cero indica el fin del bloque.

Extensión de aplicaciones

Esta extensión contiene información específica a una aplicación. Contiene:

Tabla 1.32: Extensión de aplicaciones

Etiqueta	Tamaño en bytes	Explicación
Extensión Introducer	1 byte	Indica el inicio de una extensión. tiene el valor 21h.
Plain Text Label	1 byte	Etiqueta de identificación del bloque vale FFh.
Block Size	1 byte	Número de bytes en la extensión. Tiene el valor fijo de 11.
Application Identifier	8 bytes	Secuencia de 8 caracteres ASCII, usados para identificar la aplicación.
Application Authentification Code	3 bytes	Código binario que sirve para autentificar a una aplicación.
Application Text Data	sub-bloques	Divididos en bloques de datos contiene la información que la aplicación necesita.
Block Terminator	1 byte	Este bloque de valor cero indica el fin del bloque.

Situación actual de formato

El GIF a pesar de ser un formato gráfico popular y muy estimado en el mundo del internet, está inmerso en una discusión muy fuerte. Ello es debido a que el formato está en una disputa legal; no por el formato en si, sino por el método de compresión usado por esté, el Lempel-Ziv-Welch. El método de compresión fue desarrollado por Terry Welch en 1984, como una mejora al algoritmo LZ78 (véase compresión) y fue registrado a nombre de Sperry Corp. en 1985 con la patente 4,558,302 en Estados Unidos. En 1986 la Sperry Corp. es adquirida por la Unisys. En 1993 se llega a un acuerdo entre Compuserve y Unisys.

Este exige la compra de una licencia y el pago de regalías (el costo y los porcentajes pueden variar, dependiendo del producto y su costo de venta). Están obligados a adquirir tal licencia quienes desarrollen, o revendan un programa que haga uso del algoritmo LZW. Queda libre de cargo alguno la creación, manipulación, transmisión

o almacenamiento de un archivo GIF. Este asunto afecta también a los formatos TIFF, PostScrip, entre otros. Ante tal situación ha habido desde campañas de protesta, hasta la creación de formatos que sustituyan al GIF.

Ha habido la proposición de formatos sustitutos, entre ellos destacan: El *FGF* que aprovecha todo el código sin usar el algoritmo de compresión. El *GEF* que únicamente sustituye el LZW por una variante conocida como *LZUH*. Compuserve ha propuesto hacer cambios generales además de sustituir el compresor creando el formato *GIF24*. Existe un grupo de muy fuerte que sugiere usar como sustitutos el *PNG, TIFF, JPG*, entre otros. Todas estas propuestas aunque interesantes no han logrado ser aceptadas por la mayoría de la comunidad encargada de las aplicaciones. Por esto es muy posible ver cambios en el *GIF* a mediano plazo.

1.5.1.12 TIFF

Propietario: *Aldus Corporation.*
Extension: *.tif, .tiff*

El formato TIFF fue anunciado en 1986, mientras que la revisión 6.0 del formato fue liberada en junio de 1992. Este formato está pensado para el manejo, almacenamiento e intercambio de imágenes bitmaps independiente de cualquier plataforma, el TIFF soporta una serie de extensiones, es decir, un conjunto de etiquetas y codificaciones de uso privado, que son propuestas por empresas o individuos, los cuales deben registrar sus extensiones, de lo contrario Aldus no se hace responsable por las incompatibilidades que tengan al hacerse una nueva revisión del formato. Más adelante se verá con más detalle estas extensiones.

Estructura
El formato se compone de una cabecera y de uno o varios directorios de archivo de imagen (IFD en inglés). La cabecera es general a todo el archivo e identifica al mismo a continuación:

Cabecera

Tabla 1.33: Cabecera del formato TIFF

Offset	Longitud	Descripción
0	1	Indica el tipo de orden de byte usado, si es "II" (4949h), los datos están en formato LSB, si vale "MM" (4D4Dh), se está usando el formato MSB.
2	2	Etiqueta que indica que el archivo es TIFF, es el valor decimal entero 42.
4	3	Localización del directorio del archivo de imagen, generalmente los datos de la imagen siguen inmediatamente al directorio, aunque no necesariamente.

En la última etiqueta se indica donde está localizado el primer directorio de archivo de imagen (IFD) y éste al siguiente IFD, ocurriendo así sucesivamente hasta terminar el archivo, permitiendo el almacenamiento de varias imágenes con características distintas en un solo archivo.

Directorio de archivo de imagen (IFD)
Un directorio de archivo de imagen, es una sub-cabecera que contiene todos los parámetros requeridos para leer una imagen. Consiste de un valor numérico de 2 bytes el cual indica el número de parámetros o identificadores que requiere la imagen. Cada identificador tiene 12 bytes de longitud de la siguiente manera:

Tabla 1.34: Bloques que componen una Entrada de IFD

Byte	Explicación
0-1	Etiqueta que identifica el campo. Cada uno de los campos tiene un valor único, para las etiquetas propuestas se les asigna números superiores al 32768. Las IFD son ordenadas en forma ascendente para mayor facilidad.
2-3	Tipo de campo. Indica que tipo de valores (numéricos, texto, etc.) son permitidos por el IFD en particular.
4-7	El número de valores de acuerdo con el tipo de campo, conocido también como campo de conteo.
8-11	Localización del inicio de los datos de la etiqueta. Siempre se localiza después del área de la imagen.

El directorio termina con un valor que indica la localización (Offset) del siguiente IFD, si el archivo consta de sólo una imagen el valor se define en cero.

Los tipos de valores permitidos en cada identificador, están definidos en los bytes 2 y 3 del directorio de archivo de imagen, que consiste de un valor numérico el cual indica el tipo usado. Los tipos aceptados para el TIFF hasta la revisión 6.0 son:

Tabla 1.35: Tipos de datos en el TIFF

Tipo	Descripción
1 = BYTE	Entero de 8 bits sin signo.
2 = ASCII	Byte de 8 bits que contiene código ASCII de 7 bits, el último se deja en cero (nulo)
3 = SHORT	Entero de 16 bits sin signo.
4 = LONG	Entero de 32 bits sin signo.
5=RATIONAL	Equivale a dos valores LONG: El primero representa al numerador y el segundo al denominador.
6 = SBYTE	Entero de 8 bits con signo.
7=UNDEFINED	Un byte de 8 bits que puede contener cualquier cosa, dependiendo de la definición del campo.
8 = SSHORT	Entero de 16 bits con signo.
9 = SLONG	Entero de 32 bits con signo.
10=SRATIONAL	Equivale a dos valores SLONG: El primero representa al numerador y el segundo al denominador.
11 = FLOAT	Valor en formato IEEE de simple precisión(4-bytes)
12= DOUBLE	Valor en formato IEEE de doble precisión (8-bytes)

Todos estos valores están asociados con el campo de conteo, esto significa que sólo es colocado un solo valor por campo, si es necesario una estructura más compleja, se debe usar el tipo UNDEFINED e indicar la longitud de los valores necesarios.

Todos los identificadores permitidos por la especificación básica del TIFF, han sido agrupadas en los cuatro tipos básicos de imágenes TIFF: Imágenes binivel (monocromáticas), imágenes en escala de grises, imágenes a color con paleta e imágenes a color puro.

Imágenes de binivel

Las imágenes binivel es la definición que Aldus hace de las imágenes monocromáticas (veáse, tipos de imágenes). TIFF permite tanto el formato 0 es negro ó 0 es blanco. Los identificadores usados para este tipo de imagen son:

Tabla 1.36: Tipos de etiquetas para imágenes binivel

Nombre	Etiqueta (Hex/Dec)	Tipo	Valores que puede tomar	Explicación
Phometric Interpretation	262/106	SHORT	0= blanco es cero. 1= negro es cero. si se usa con el valor de compresión 2, la imagen se verá en video inverso.	TIFF indica en este campo el tipo de codificación del color en dos formatos: cero es negro o cero es blanco.
Compression	259/103	SHORT	1= sin compresión. 2=Codificación Run-Length Huffman modificado dimensional del CCITT. 3= Compresión Packbits, que es modificación del RLE.	Indica el tipo de compresión usada. La compresión solo afecta a los datos del área de la imagen.
ImageLength	257/101	SHORT o LONG		Indica el número de renglones en la imagen.
ImageWidth	256/100	SHORT o LONG		Indica el número de columnas en la imagen.
ResolutionUnit	296/128	SHORT	1= no hay medida, para imágenes que no necesitan este campo. 2= pulgadas 3= centímetros	
XResolution	282/11A	RATIONAL		Número de pixels por unidad de resolución en el ancho de la imagen (típicamente horizontal)
YResolution	283/11B	RATIONAL		Número de pixels por unidad de resolución en el alto de la imagen (típicamente vertical)

Tabla 1.36: Tipos de etiquetas para imágenes binivel (Continuación)

Nombre	Etiqueta (DEC/HEX)	Tipo	Valores aceptables	Explicación
RowsPerStrip	278/116	SHORT o LONG		Es el número de renglones almacenados en cada bloque. Ésta separación en bloques lo realiza el TIFF con el fin de facilitar la edición y eficientar los buffers.
StripOffsets	273/111	SHORT o LONG		Indica el inicio de ese bloque.
RowsPerStrip	278/116	SHORT o LONG		Es el número de bytes en el bloque una vez comprimidos con cualquier tipo.

Los bloques (strips) pueden estar en cualquier parte del archivo, Aldus define tres identificadores para su correcta localización y lectura.

Imágenes en escala de grises

Las imágenes en escalas de grises se diferencian de las monocromáticas, por la profundidad de sus pixels (véase tipos de imagen), los identificadores usados para las imágenes en escala de grises son las mismas que las binivel con una diferencia en la etiqueta de compresión: En las imágenes a color solo se toleran dos valores 1 = sin compresión y 32773 = compresión con PackBits, que generalmente es un tipo de compresión malo para imágenes con grandes áreas de un solo tono, por ello, Aldus recomienda no usar compresión .

Para las imágenes en escala de gris se añade el siguiente identificador:

Tabla 1.37: Identificador de escalas de grises

Nombre	Etiqueta (DEC/HEX)	Tipo	Valores aceptables	Explicación
BitsPer Sample	258/102	SHORT		Es el número de bits permitidos en cada pixel, habiendo 4 o 8, que permiten 16 o 256 tonos de gris.

Imágenes a color con mapa

Son aquellas imágenes que hacen uso de un mapa de color, y se representan por medio de los índices de dicho mapa. Para poder ser leído se adiciona en el identificador "Phometric Interpretation", el valor 3 que indica la paleta de colores. Además es añadido un identificador que define al mapa de color.

Tabla 1.38: Identificador de mapa de color

Nombre	Etiqueta (DEC / HEX)	Tipo	Valores aceptados	Explicación
ColorMap	320/140	SHORT	Máximo valor posible : $3*(2^{\text{bits por pixel}})$	Define un mapa de colores cuyas entradas son usadas por la imagen, la profundidad está limitada por el valor máximo antes descrito. Los colores se almacenan en tripletas RGB y van desde el negro (0,0,0), hasta el blanco (n,n,n).

Todos los demás identificadores son usados sin variación.

Imágenes de color puro (true color)
Son aquellas imágenes que almacenan cada color en su respectivo pixel (para más detalle véase tipos de imagen). Las modificaciones a los indicadores son:

Inclusión del valor 2 (RGB) en el identificador "Phometric Interpretation". Además se elimina el identificador ColorMap y se añade el identificador *SamplesPerPixels*.

Tabla 1.39: Identificador de imágenes True color

Nombre	Etiqueta (DEC / HEX)	Tipo	Valores aceptados	Explicación
SamplesPer Pixels	277/115	SHORT		Número de componentes por pixel. Este número vale 3 para imágenes RGB, 4 para imágenes CMYK, etc.

Los siguientes identificadores proveen datos adicionales acerca de la imagen. Son usados a discreción del desarrollador del programa y/o software cuando sean necesarios.

Tabla 1.40: Datos adicionales de la imagen TIFF

Nombre	Etiqueta (DEC / HEX)	Tipo	Valores aceptados	Explicación
CellLength	265/109	SHORT	1	Longitud de la matriz de difuminación usada para un archivo binivel (monocromático).
CellWidth	264/108	SHORT	1	Ancho de la matriz de difuminación usada para un archivo binivel (monocromático).
ExtraSamples	338/152	SHORT	m componentes	Indica el número de valores en cada pixel, se usa cuando es mayor al valor que el campo fotométrico sugiere. Los valores asociados son: 0 = datos no especificados, 1= Datos asociados con el canal alfa, 2= Datos no asociados con el canal alfa

Tabla 1.40: Datos adicionales de la imagen TIFF (Continuación)

Nombre	Etiqueta (DEC/HEX)	Tipo	Valores aceptables	Explicación
FillOrder	266/10A	SHORT	1 Byte	Marca el orden que tienen los bits dentro de un byte. 1= codificación MSB 2= codificación LSB. Por default vale 1
FreeByte Counts	289/121	LONG		Para cada cadena de bytes contiguos no usados en un archivo, número de bytes en la cadena.
FreeOffsets	288/120	LONG		Para cada cadena de bytes contiguos no usados en un archivo, localización de la cadena en el archivo.
Gray Response Curve	291/123	SHORT	N = $2^{BitsPerSample}$	Densidad óptica de cada valor de pixel posible en imágenes en escala de gris.
Gray Response Unit	290/122	SHORT	N = 1	Precisión que tiene el identificador GrayResponseCurve
MaxSample Value	281/119	SHORT	N= valores por pixel	Valor máximo de valores por pixel. El valor base es $2^{(BitsPerSample)}-1$.
MinSample Value	280/118	SHORT	N= valores por pixel	Valor mínimo de valores por pixel. El valor base es 0.
SubfileType	255/FF	SHORT	N=1	Una general indicación del tipo de datos contenidos en el sub archivo. 1= Completa resolución de los datos de la imagen. 2= Reducida resolución de los datos de la imagen. 3= Una página de una multimagen.
NewSubfile Type	254/FE	LONG	N=1	Una indicación general del tipo de datos contenidos en este sub archivo. Es un conjunto de 32 bits, son usados para esta revisión 3 bits, los demás deben valer 0. Bit 0: 1 versión reducida de una imagen, 0 en caso contrario. Bit 1: 1 es una imagen (página) parte de una imagen multipágina, 0 en caso contrario. Bit 2: Define máscara de transparencia de otra imagen, 0 en caso contrario.
Orientation	274/112	SHORT	N=1	Orientación de la imagen.
Planar Configuration	284/11C	SHORT	N=1	Cómo son almacenados los componentes de cada pixel. 1 formato chunky, es decir, los datos en secuencia RGB. 2 formato planar, es almacenamiento en planos de color.
Host Computer	316/13C	ASCII	texto	Computadora y/o sistema operativo en que se creó la imagen.

Tabla 1.40: Datos adicionales de la imagen TIFF (Continuación)

Nombre	Etiqueta (DEC / HEX)	Tipo	Valores aceptados	Explicación
Artist	315/13B	ASCII	texto	Indica quien creó la imagen.
Copyrigth	33432/8298	ASCII	texto	Aviso de derechos de autor.
DateTime	306/132	ASCII	N=20	El formato es "YYYY:MM:DDHH:MM:SS", usando reloj de 24 horas. La longitud de la cadena con el terminador es 20 caracteres.
Make	271/10F	ASCII	texto	Nombre del fabricante del escanner.
Model	272/110	ASCII	texto	Modelo o nombre del escanner.
Software	305/131	ASCII	texto	Nombre y versión del software usado para crear la imagen.

Los tipos de compresión soportados por el TIFF en el nivel básico son el esquema Packbits (similar al RLE), y la compresión grupo 3 de la CCITT (una mezcla de Run-Length y Huffman). Otras codificaciones y tipos son cubiertos en las extensiones a continuación mencionadas.

Extensiones
Las extensiones son características propuestas por terceros que pueden no ser soportadas por todos los decodificadores TIFF. Tales extensiones se realizan para introducir o usar técnicas o algoritmos no contemplados por Aldus. Las extensiones reconocidas por la revisión 6 del formato son como siguen:

Compresión
Son añadidos dos esquemas de compresión: El tipo 3, codificación CCITT.4 para datos en binivel. Tipo 5 compresión usando el algoritmo LZW, no se recomienda usar por problemas legales. (véase el formato GIF).Tipo 6 codificación CCITT.6 para datos en binivel.

Identificadores

DocumentName Nombre del documento del cual fue obtenida la imagen
PageName Nombre de la página del cual fue obtenida la imagen
PageNumber Número de página del documento del cual fue obtenida la imagen
Xposition Posición de la imagen respecto del inicio de la página en el eje X
Yposition Posición de la imagen respecto del inicio de la página en el eje Y

1.5.1.13 JPEG

Propietario: ***Joint Photographic Experts Group, dependiente de la ISO.***
Extensión: *.jpg*

El grupo de expertos fotógrafos, está encargado de crear normas de uso en transmisión, almacenamiento, codificación y descodificación gráfica. Hay varios grupos de trabajando en conjunto con gente del CCITT dedicados a poner estándares para radio, telefonía, televisión, etc. Uno de ellos desarrolló el estándar JPEG. El formato está diseñado para trabajar con imágenes fotográficas, o sea de alta calidad,

con un gran número de colores y tonos de luz, para ello se usa un tipo de compresión que crea una imagen semejante pero de menor tamaño que resultaría de la original.

Compresión sin pérdida y con pérdida

A diferencia de casi todos los formatos gráficos. El JPEG no necesariamente reconstruye la imagen original bit por bit. Pero muchas veces tal reconstrucción es de mejor calidad que usando un compresor sin pérdida (lossless). Veamos un ejemplo práctico:

Los digitalizadores típicamente producen una imagen de 24 bits de profundidad, con tres bits para el rojo, tres para el verde, y tres para el azul. Un formato que usa compresión, tal como el GIF, tiene un mapa de 256 entradas (colores) con una profundidad de 8 bits, en vez de los 24 originales; y 256 colores en vez de los 16 millones que dispone la imagen original. Como es de esperarse el tamaño es muy grande, aún usando el algoritmo de compresión LZW, y con una pérdida sensible de calidad en la imagen.

El JPEG por otro lado, guarda información de los cambios de color de la imagen, particularmente las variaciones en la brillantez, ya que el ojo humano es sensible a ellas. Al reconstruirse la imagen en el JPEG, no se obtiene la imagen original, pero ésta tiene variaciones de luminosidad similares a la imagen original, por tanto, para el ojo humano será casi igual al original. Aunque esta técnica es poco común en los formatos gráficos, es de amplio uso en los sistemas televisivos, de hecho, las mismas técnicas son usadas para los estándares de transmisión de video MPEG (que también es desarrollado por las mismas organizaciones que el JPEG).

El JPEG tiene por contra que la efectividad de su algoritmo y la calidad de sus resultados decaen cuando es usado en imágenes con pocos colores y muy anguladas; como pueden ser, caricaturas, diagramas, etc. Más sin embargo, el espacio en disco salvado por la compresión JPEG puede ser dramático. Por ejemplo, Una imagen de 727 x 525 pixels a 24 bits, ocupa cerca de 1145 Kb en su forma original. El archivo que resulta usando el formato GIF ocupa 240 Kb. Una versión de alta calidad en JPEG ocupa tan solo 115 Kb, y una versión comparable en calidad a la hecha por el GIF ocupa únicamente 58 Kb. La calidad y el espacio son factores decisivos que han hecho del JPEG un formato ampliamente aceptado.

Etapas de la compresión JPEG

Por ser un estándar tiene muchas variantes en la forma de codificarse, además de varias opciones de compresión. Por ello explicare el estándar JPEG base, que apoyado por las implementaciones presentes y futuras que del JPEG se hagan. Se inicia por convertir la imagen cidificado en RGB en un espacio de luminosidad / cromancia del color de codificación YCrCb, esto significa, una imagen base en escala de grises más dos canales de información de diferencias de color.

Los datos de la imagen pueden ser subdivididos, combinando pixels adyacentes en un solo valor. Entonces una transformación de coseno discreta (DCT) es aplicada para convertir los datos en información variable. La cuantificación truca los resultados del DCT en un pequeño rango de valores. Este es el paso que hace al JPEG un algoritmo con pérdida; los coeficientes de la cuantificación determinan

cuanta información se pierde y, por lo tanto la extensión de la compresión y la calidad de la imagen reconstruida. Finalmente el resultado de la cuantificación es comprimida usando ya sea un algoritmo de Huffman o aritmético para producir la salida final.

En la descompresión se siguen los pasos en orden inverso, se descomprimen los resultados de la cuantificación y usa una transformada discreta de coseno inversa para reconstruir la imagen. Los bits bajos perdidos en el proceso no son reconstruibles, así el descompresor usa ceros para ellos.

Esquemas de color y subdivisión

En una imagen, JPEG comprime cada canal por separado. Si bien es posible comprimir con los componentes RGB usuales. La compresión JPEG trabaja mejor con canales que son expresados como luminancia (brillantez) y cromancia (color), y los canales relacionados al color necesitan menos datos que los canales del brillo. En una imagen RGB, los tres canales llevan algo de información sobre el brillo, y los tres deben ser codificados con la misma calidad.

Número necesario de Bytes para Representar un Cuadro de 2x2 pixels

Un esquema (codificación) de color, conocido como CCIR 601, usa tres componentes, Y, Cb y Cr, los cuales son estrictamente brillantez (Y), intensidad de azul (Cb), intensidad de rojo (Cr).El canal Y puede ser usado como una versión blanco y negro de la imagen. (hay una variación que al los canales Cb y Cr les nombra U y V, llamándosele entonces YUV). Los componentes se definen como:

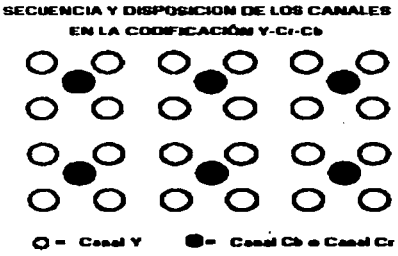
$$Y = 0.299R + 0.587G + 0.114B$$

$$Cb = 0.1687R - 0.3313G + 0.5B$$

$$Cr = 0.5R - 0.4187G - 0.0813B$$

Donde los valores R, G, B, son los componentes del pixel.

Nótese que ésto es en principio, una transformación sin pérdida, porque una tripleta YCbCr contiene la misma información que su contraparte RGB, con un cierto nivel de error debido a que los resultados son redondeados a valores enteros. Debido a las características ya mencionadas que tiene el ojo humano, una imagen aceptable puede ser reconstruida usando muchos menos componentes Cr y Cb que componentes Y.



JPEG permite que los diferentes componentes sean muestreados en diferentes cantidades. Una opción común es usar un elemento Cr y uno Cb por cuatro Y, así, por cada cuadro de 2x2 pixels hay 4 componentes (bytes) Y, un componente Cb, y un componente Cr o sea la mitad de los componentes (bytes) que en una codificación RGB sin prácticamente percibir pérdida alguna en la calidad de la imagen.

La técnica de tomar algunos componentes en cantidades menores que otros, es conocido como submuestreo (subsampling). El grado de submuestreo está estrechamente relacionado con los parámetros de cuantificación para determinar el grado de compresión y la calidad de la imagen reconstruida.

Codificación *DCT*

La *Transformación Discreta de Coseno* (DCT) transforma un arreglo de los datos de intensidad (Brillo) en un arreglo de datos de frecuencia, el cual indica que tan rápido varia la intensidad en la imagen. Para imágenes con múltiples componentes, como las imágenes YCbCr. el DCT es aplicado separadamente a bloques de datos de 8x8 para cada componente. Para datos submuestreados, habrá bloques que tengan más componentes YCbCr que otros.

Los puntos de datos (pixels) en cada bloque son numerados del (0,0) en el punto superior izquierdo hasta el punto inferior derecho (7,7). El DCT produce un nuevo bloque de 8x8 de los datos transformados usando la siguiente formula:

$$F(u,v) = \frac{1}{4}C(u)C(v)\left[\sum_{x=0}^{7}\sum_{y=0}^{7} f(x,y)\cos\frac{(2x+1)u\pi}{16}\cos\frac{(2x+1)v\pi}{16}\right]$$

$$Donde C(z) = \{1/\sqrt{2}, z = 0$$
$$C(z) = \{1, z \neq 0$$

El resultado del DCT es un conjunto de "frecuencias espaciales" F(u,v) (conocido como componente DC), el cual, estrictamente hablando, indica para cada uno de los conjuntos la velocidad (variación) de los colores en la imagen y en que grado cambian

los valores. así, F(0,0) es el valor que indica que área del cuadro de pixels no varía, F(1,0) indica el grado en el cual los valores cambian lentamente de manera horizontal (baja frecuencia) únicamente, y F(7,7) indica el grado al cual los valores cambian más rápidamente en ambas direcciones (alta frecuencia horizontal y verticalmente).

El DCT es en principio reversible usando la inversa de la transformada:

$$F(u,v)=\frac{1}{4}\left[\sum_{x=0}^{7}\sum_{Y=0}^{7}C(u)C(v)f(x,y)\cos\frac{(2x+1)u\pi}{16}\cos\frac{(2x+1)v\pi}{16}\right]$$

Los errores de redondeo hacen que los valores resultantes escasamente varíen de los originales.

Para los bloques de 8x8, el valor de salida de la transformación añade cerca de cuatro bits más que el valor de entrada. Por ejemplo, si la entrada es ocho bits, la salida será de doce. De este modo, vemos que nuestros datos han aumentado más que ser comprimidos. Lo importante del cambio de intensidad a frecuencias espaciales es que los cambios lentos son más notables que los cambios rápidos, por lo tanto, la baja frecuencia es más importante que la alta frecuencia para reconstruir la imagen. La cuantificación toma ventaja de esto.

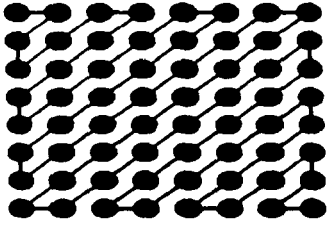
Cuantificación

La cuantificación define la precisión con la cual cada uno de los valores resultantes de la transformación son almacenados. JPEG usa cuantificación lineal, lo que significa simplemente, que cada uno de los valores de la transformación es dividida por un factor de cuantificación y redondeado a un entero para conseguir el valor que será guardado. Una tabla de factores de cuantificación de 8x8 es usada, un factor por cada término de salida. Esta tabla es almacenada en el archivo JPEG, de esta manera un codificador puede usar una tabla estándar o ajustar una para dar el grado de compresión deseado para una imagen en particular. Un archivo con múltiples componentes debe contener múltiples tablas, muy comúnmente, hay una

Una tabla típica tiene un factor bajo, cercano a 16, para cada componente DC F(0,0), mientras los componentes de más alta frecuencia tiene factores cercanos al 100. Dados los 12 valores DC del ejemplo anterior, este lo convierte en 8 bits para f(0,0) y cerca de 5.3 bits para F(7,7). Cuando se descodifican, los valores originales son (aproximadamente) recuperados multiplicando el factor de cuantificación. Por ejemplo, si el factor es 16, los valores DC del 120 al 135 serán convertidos a un valor de 8, los cuales al ser descomprimidos valdrán 128, una aproximación del valor de entrada.

Los componentes DC son tratados especialmente, porque los componentes DC adyacentes tienden a ser similares, cada componente DC es almacenado como la diferencia del componente DC anterior. Los componentes AC (que corresponde a las variaciones de alta velocidad) son lógicamente puestos en zigzag, (veáse la figura),

que pone la frecuencia más baja primero. Los componentes de cuantificación de alta velocidad seguramente serán cero, y la etapa de compresión final trabaja particularmente bien en grupos de ceros.



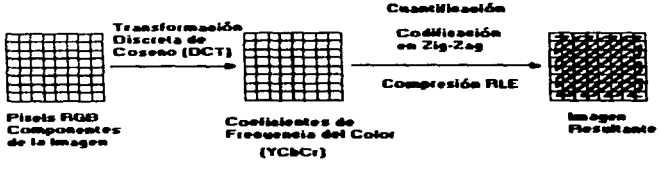
Codificación de los cuadros DCT

Compresión final

Los resultados de la cuantificación son comprimidos usando un tipo de compresión aritmética (véase compresión) o una versión modificada del algoritmo Huffman.

La variante de la codificación Huffman es conocida como codificación de longitud variable (Variable Length Code) o VLC, codifica los componentes DC como valores de cuatro bits, seguidos por un entero con signo que da la longitud del valor (veces en que se repite). Los componentes AC son almacenados como un valor de ocho bits seguida por un entero de longitud variable. Los cuatro bits superiores son el número de ceros que preceden este valor, y los cuatro bits inferiores son la longitud del entero.

Aunque el JPEG permite el uso de la compresión aritmética, que tiene un mejor desempeño que la Huffman, es poco usada por las restricciones debido a las patentes.



Proceso JPEG

Otras opciones JPEG

El estándar JPEG permite una gran variedad de modos de compresión y almacenamiento.

Modos progresivos
En una situación donde una imagen JPEG está siendo recibida de otro lugar (como ocurre en Internet), debería a veces, ser posible ver una versión en baja resolución de la imagen mientras llega. El modo progresivo mezcla la codificación de los datos para permitir esto. El modo de selección espectral envía los componentes de baja frecuencia primero. El modo de aproximación sucesiva envía los N bits superiores, y luego los inferiores, para cualquier N. Esos dos modos pueden ser combinados como sea necesario.No existe implementación alguna de los modos progresivos debido a la lentitud de la conexiones, y del proceso de codificación/descodificación de la imagen.

Modo jerárquico
En el modo jerárquico, una imagen es almacenada en varias resoluciones de calidad creciente. Por ejemplo, una imagen de 1000x1000 debe ser guardada en una versión reducida de 10x10, otra de 100x100, y por último la de 1000x1000. Cada imagen es almacenada como la diferencia de la versión más pequeña anterior. Tampoco ha sido implementado.

Modo sin pérdida (lossless)
Hay un modo de compresión sin pérdida especificado para usuarios que por alguna razón requieran una reconstrucción exacta bit por bit. El modo sin pérdida almacena cada pixel como la diferencia con el pixel inmediatamente anterior. Usa una compresión VLC para los datos, sin usar transformación alguna, ni cuantificación. De hecho, lo único que comparte con el JPEG con pérdida es el nombre. La calidad de la compresión sin pérdida es cercana a la de otros formatos como el GIF. Este modo se considera más un compromiso legal que técnico.

Formatos JPEG
A la fecha existen algunas implamentaciones del estándar JPEG, ya que en realidad no existe un formato JPEG básico, porque técnicamente se le considera un conjunto de propuestas para codificar una imagen fotográfica, pero si existen implementaciones hechas por terceros (muy pocas hasta la fecha). De todos ellos, *Adobe* y *C-Cube Microsystems* se distinguen.

JPEG-TIFF
Adobe introdujo en la especificación de su formato TIFF el uso de la codificación JPEG con pérdida. La revisión 6.0 del TIFF incluye recomendaciones para el uso de la codificación YCbCr, la cuantificación y los bloques necesarios para soportar la estructura JPEG. Debido a sus errores y ambigüedades. El JPEG no ha sido implementado en aplicaciones que hacen uso del TIFF. Muchos consideran que hasta que no se haga una revisión de la norma, es poco probable que sea implementado con éxito.

JPEG File Interchange Format
JFIF es una implementación del formato de intercambio de archivos JPEG (JFIF en inglés). Definida por Eric Hamilton de C-Cube Microsystems. El JFIF puede almacenar imágenes en escala de grises o la representación YCbCr de hasta 8 bits de profundidad. El JFIF se apega a la norma JPEG antes mencionada, en todas sus

características. Únicamente introduce un bloque propietario de aplicaciones (APP0) e información tanto del usuario, como de los programas que lo usen, el cual puede ser ignorado sin problema alguno por los descodificadores que no entiendan la extensión de C-Cube. Este es el formato desarrollado en base al JPEG más exitoso, casi desde su presentación ha sido el estándar en casi todas las plataformas. Su extensión es .JPG. El bloque APP0 (o el bloque de C-Cube) también contiene una versión reducida de la imagen original o thumbnail[14].

Estructura del formato JFIF

La sintáxis de un archivo JFIF sigue la misma sintáxis del formato de intercambio definido en el anexo B del documento ISO DIS 10918-1. C-Cube identifica a sus extensiones y marcadores con las cadenas X'FF', SOI, APP0, X'00'.

X'FF', SOI	Indicador del inicio del bloque APP0

La estructura del bloque APP0 es como sigue:

X'FF', APP0, length, identifier, version, units, Xdensity, Ydensity, Xthumbnail, Ythumbnail, (RGB)n.

X'FF'	APP0: Identificación del bloque

Length, identifier, version, units, Xdensity, Ydensity, Xthumbnail, Ythumbnail, (RGB)n: datos del bloque. se explica en la tabla 1.41.

Segmentos opcionales del marcador APP0

X'FF', SOFn, length, frame parameters: parámetros de información acerca de la imagen.

Number of components Nf = 1 or 3: número de componentes en los valores YCbCr. Esto es debido a que la imagen en JFIF, soporta sólo escalas de gris o 256 colores (8 bits).

Valor del primer componente, o sea el valor Y.
Valor del segundo componente, o sea el valor Cb.
Valor del tercer componente, o sea el valor Cr.

X'FF', EOI: etiqueta de terminación.

[14] La imagen thumbnail es una representación a pequeña escala de la imagen original, se utiliza en muchas aplicaciones como catálogos de imágenes para comodidad del usuario, al buscar una o varias imágenes específcas.

tabla 1.41: Valores de los campos del bloque APP0.

Campo	Extension en bytes	Explicacion
Length	2	Valor de la extensión total del bloque, incluyendo este campo, pero excluyendo el marcador de la extensión APP0.
Identifier	5	Identificador del bloque APP0, usa los valores X'4A', X'46', X'46', X'49', X'00'. La cadena "JFIF" únicamente identifica a este marcador APP0. La cadena debe tener paridad cero (bit 7=0).
version	2	El byte más significativo es usado para revisiones mayores, el menos significativo lo es para revisiones menores. La versión 1.02 es la actual en funciones.
units	1	Unidades para las densidades X y Y. Si vale 0= sin unidades, X y Y especifican el tamaño del pixel. Si vale 1= X y Y son puntos por pulgada. Si vale 1= X y Y son puntos por cm.
Xdensity	2	Densidad horizontal de los pixels.
Ydensity	2	Densidad vertical de los pixels.
Xthumbnail	1	Valor del largo de la versión reducida (thumbnail) de la imagen en pixels.
Ythumbnail	1	Valor del ancho de la versión reducida (thumbnail) de la imagen en pixels.
(RGB)n	3n	Valores RGB para la versión reducida de la imagen. El número de pixels para n se da con la siguiente formula: n = (X thumbnail)(Y thumbnail).

Extensiones APP0 JFIF

Inmediatamente después del bloque de datos puede haber una extensión marcador APP0. Este segmento de extensión sólo está presente en la versiones 1.02 ó superior del JFIF. La sintaxis de la extensión es:

X'FF', APP0, length, identifier, extension_code, extension_data

Tabla 1.42: Valores de los campos de las extensiones APP0 JFIF

Campo	Extension en bytes	Explicacion
length	2	Valor de la extensión total del bloque, incluyendo este campo, pero excluyendo el marcador de la extensión APP0.
identifier	5	Identificador de la extensión APP0, usa los valores X'4A', X'46', X'58', X'58', X'00'. La cadena "JFXX" únicamente identifica a este marcador APP0. La cadena debe tener paridad cero (bit 7=0).
extension_code	1	Código el cual identifica el tipo de extensión. Las siguentes extensiones son usadas por la versión 1.02: X'10': versión reducida de la imagen (Thumbnail) usando JPEG. X'11': Thumbnail usando almacenamiento sin compresión de 1 byte por pixel. X'13': Thumbnail usando almacenamiento sin compresión de 3 bytes por pixel.
extension_data	variable	Los segmentos de datos varían mucho según el tipo de extensión, véase los tipos de extensión para más detalles.

JFIF extensión X'10': Thumbnail usando compresión JPEG
Usa la técnica JPEG para tratar la imagen. Cada extensión es un bloque APP0 separado, con sólo los campos y datos referentes a la imagen. Igual que su similar en tamaño original los parámetros en la cabecera del cuadro son como siguen:

X'FF', SOI: identificador
X'FF', SOFn, length, frame parameters: parámetros de la imagen.
Number of components Nf = 1 or 3: número de componentes en los valores YCbCr.
Valor del primer componente, o sea el valor Y.
Valor del segundo componente, o sea el valor Cb.
Valor del tercer componente, o sea el valor Cr.

JFIF extensión X'11': Thumbnail usando un byte por pixel
Esta extensión soporta un byte por pixel y una paleta de colores en el campo *extension_data*. La extensión se compone de:

Tabla 1.42: Campos de la imagen Thumbnail

Campo	Extensión en bytes	Explicación
Xthumbnail	1	Valor del largo de la versión reducida (thumbnail) de la imagen en pixels.
Ythumbnail	1	Valor del ancho de la versión reducida (thumbnail) de la imagen en pixels.
paleta	768	Valores RGB de 24 bits (8 bits por componente) de los pixels para la paleta de (256) colores.
(pixel)n	n	Valores para los pixels del thumbnail. Se calcula n así: n =Xthumbnail x Ythumbnail

JFIF extensión X'13': Thumbnail usando un byte por pixel
Esta extensión soporta tres bytes por pixel y una paleta de colores en el campo extension_data. La extensión se compone de:

Tabla 1.43: Campos de la imagen Thumbnail byte por pixel

Campo	Extensión en bytes	Explicación
Xthumbnail	1	valor del largo de la versión reducida (thumbnail) de la imagen en pixels.
Ythumbnail	1	valor del ancho de la versión reducida (thumbnail) de la imagen en pixels.
(RGB)n	3n	Valores RGB de 24 bits para el thumbnail. Se calcula n así: n= (Xthumbnail)(Ythumbnail).

1.5.2 Vector

Las imágenes que se almacenan en esta categoría, se pueden describir como una unión de figuras geométricas simples, tales como líneas, círculos, elipses, polígonos, etc. Para representar esas figuras se usa una serie de comandos que contienen los datos necesarios de la figura, por ejemplo, si la figura a representar es una línea recta sólo necesitamos el punto de inicio y el de término.

Las imágenes vectoriales no soportan grecas complejas, sino en vez de ello es "rellenada" el área inscrita en la figura geométrica. El relleno puede ser un rayado, achurado, cuadriculado, etc., en una imagen monocromática; en el caso de imágenes a color se usan colores sólidos (un solo color por área), o bien, gradientes, los cuales son una secuencia de líneas las cuales van de un color sólido a otro, mediante el uso de tonos intermedios. Debido a que únicamente se necesitan los puntos básicos para construir una figura, es posible manipular su tamaño (o escalar una imagen) sin afectar la calidad de la imagen -cosa que es casi imposible en un bitmap-; además el orden de almacenamiento o desplegado de los elementos de la imagen es poco relevante., siendo innecesaria una estructura rígida en el archivo. Es común codificar por medio de ASCII las imágenes vectoriales, lo cual le da una gran independencia de plataforma.

Su mayor desventaja sin embargo, es que son formatos completamente ineficientes frente a imágenes muy complejas y difuminadas, como son las fotografías; los vectores tienden a usar mayor espacio que sus contra partes en bitmaps, exigen más memoria y capacidad de procesamiento.

Las imágenes vectoriales son usadas en diagramas, planos organigramas, etc. También son usadas para crear fuentes (letras) fáciles de usar.

1.5.2.1 DXF (Drawing Interchange Format)

Propietario: *Autodesk Inc.*
Extensión: *.dxf*

El formato DXF creado por *Autodesk* es uno de los formatos más apoyados y prácticamente es usado universalmente por las aplicaciones CAD/CAM, y por aplicaciones que no son propiamente CAD, pero que necesitan exportar / importar imágenes DXF. El DXF es el formato más cercano al estándar del tipo vector con todas las ventajas y desventajas que ello implica. Autodesk modifica algunos aspectos de su formato con cada nueva versión (revisión) que es liberada, aunque sin perder compatibilidad con las anteriores versiones, la revisión más popular es la 12.

Estructura
El formato se compone de 4 partes:

1. Sección de la cabecera:
La información general acerca de la imagen está localizada aquí. Cada parámetro tiene un nombre y un valor asociado. En la tabla I.1 (ver Apéndice I) se enlistan los parámetros y su significado. Las fechas son siempre escritas en el siguiente formato: <Fecha del calendario juliano>.<fracción> del mismo modo se aplica al tiempo transcurrido.

2. Sección tables
Esta sección contiene algunas tablas, cada una de las cuales contiene un número de variable de entradas. El orden de las tablas puede variar, pero sin embargo la tabla LTYPE siempre precederá a LAYER. Las tablas de esta sección son:

Tabla 1.44: Campos de la sección TABLES

Campo	Explicación
Linetype table (LTYPE)	Define todo lo relacionado con la posición del texto en el formato.
Layer table (LAYER)	Define las capas o niveles de la imagen.
Text Style table (STYLE)	Define al texto en si mismo, es decir, el tamaño del texto, la fuente usada, la familia de fuente usada, etc.
View table (VIEW)	Se fija el punto de vista de la imagen.
User Coordinate System table (UCS)	Define el usuario el sistema coordinado.
Viewport configuration table (VPORT)	Maneja los objetos y sus vistas
Dimensión Style table (DIMSTYLE)	
Application Identification table (APPID)	

Los nombres de cada tabla están en mayúsculas. cada elemento de cada tabla si no tiene valores, es simplemente eliminado para ahorrar espacio. Cada tabla inicia con el nombre de la misma y termina con el comando ENDTAB.

3. Sección blocks
Esta sección contiene todo el bloque de definiciones que son usadas en el dibujo. Inclusive bloques anónimos generados por el comando HATCH y por dimensionamiento asociativo. Todos los bloques están acotados entre los comandos BLOCK y ENDBLK. En caso de referencias externas, son escritas en el archivo DXF como cualquier bloque de definición, pero a diferencia de los bloques normales, incluyen el path y el nombre del archivo de la referencia externa.

4. Sección entities
Los objetos Entities son localizables tanto en la sección BLOCK como en la ENTITIES. Hay objetos que siempre deben aparecer, pero hay también opcionales, los cuales se indican con una N después de los parámetros.

Los programas que leen el DXF asumen que no hay orden en los grupos que describen a una entidad. Por ser el DXF una representación de una base de datos del dibujo; Autocad irá acomodando nuevos grupos conforme se necesiten, es posible una futura normalización.

Las entidades se dividen en grupos. El grupo 0 inicia la entidad. Los restantes grupos se componen de comandos ASCII que describen a la imagen, por ser una típica imagen vectorial cada comando cuenta con parámetros que facilitan el trazado de las líneas y arcos de las mismas, por razones de espacio, solo enumero los comandos existentes en la Revisión 12.

LINE	POINT	CIRCLE
ARC	TRACE	SOLID
TEXT	SHAPE	BLOCK
ENDBLK	INSERT	ATTDEF
ATTRIB	POLYLINE	VERTEX
SEQEND	3DFACE	VIEWPORT

DIMENSION	LINEAR	ANGULAR
DIAMETER	RADIUS	ORDINATE

Para reducir el tamaño del archivo, los valores de cada comando están dados en Sistema coordinado de entidades (SCE), la información adicional es la de los ejes Z y el valor de elevación, en imágenes de 3D. Desde la Revisión 10, opcionalmente se permite la codificación del archivo en binario, en vez del típico ASCII, esto reduce el tamaño del archivo tipo DXF cinco veces, y facilita el manejo del mismo.

1.5.2.2 UNIX plot format

Propietario: UNIX System Labs.
Extensión: .Plot

El formato fue creado a principios de los 70's y se ha mantenido prácticamente sin cambio desde entonces; gracias a su gran tiempo en la comunidad Unix, es ampliamente apoyado, pero por su alta simplicidad, está limitado en sus posibilidades.

El formato está basado en ASCII que describe imágenes para plotter. Este formato consiste de una secuencia de instrucciones de plotter. Cada instrucción es una letra minúscula, generalmente seguida de algunos parámetros. Las instrucciones terminan con un byte de nueva línea (0A h). Para los parámetros numéricos son usados enteros de dos bytes con signo, ordenándose los bytes según el tipo de máquina usado.

Las coordenadas son enteros de 16 bits, y la extensión del área dibujada, junto con los puntos de origen se definen en la etiqueta Space, es interesante mencionar que los valores de los ejes crecen en forma contraria al de la mayoría de los formatos (El eje x de izquierda a derecha, el eje y de abajo hacia arriba).

Estructura
El primer comando en el archivo debe ser Space. Los siguientes comandos pueden estar en cualquier orden con un comando Erase entre imágenes.

Space - El comando es la letra " s " seguida de cuatro parámetros numéricos. Los primeros dos parámetros son las coordenadas X y Y de la esquina izquierda inferior del área de dibujo. Los siguientes dos parámetros son las coordenadas X y Y de la esquina superior derecha.

Erase - El comando es la letra " e ". Blanquea la pantalla o inicia una nueva página en la impresora.

Move - El comando es la letra " m " seguida de dos parámetros numéricos, X y Y. Hacen a la coordenada el punto actual.

Line - El comando es la letra " l " seguida de cuatro parámetros numéricos, (X0, Y0) y (X1, Y1). El comando hace una línea de (X0, Y0) a (X1, Y1) y deja el cursor en (X1, Y1).

Continue - El comando es la letra " n " seguida de dos parámetros numéricos, X y Y. Dibuja una línea desde la posición anterior hasta el valor X, Y, y este se vuelve el nuevo punto actual.

Point - El comando es la letra " p " seguida de dos parámetros numéricos, X y Y. Dibuja un punto en el valor X, Y, y este se vuelve el nuevo punto actual.

Circle - El comando es la letra " c " seguida de tres parámetros numéricos, X, Y, y R. Dibuja un círculo con centro en (X, Y) y de radio R.

Arc - El comando es la letra " a " seguida de seis parámetros numéricos, X, Y, X0, Y0, X1, y Y1. Dibuja un arco con centro en (X, Y) desde el punto (X0, Y0), hasta el punto (X1, Y1).

Label - El comando es la letra " t " seguida de una cadena de texto. Dibuja la cadena.

Line type - El comando es la letra " f " seguida de una cadena de texto. la cadena define las subsecuentes líneas que son dibujadas (rellenar la imagen). Debe tener cualquiera de las siguientes cadenas: dotted, solid, longdashed, short-dashed, o dotdashed.

1.5.3 PDL

Los Lenguajes de Descripción de Página (*Page Description* Language) son la forma más novedosa de almacenamiento de imágenes: un formato PDL se caracteriza por componerse de un lenguaje que describe estructuras y objetos en un área (página) determinada. Los objetos son estructuras de lo más variado, ya que pueden almacenar texto, animaciones, imágenes 2D y 3D, etc., En el caso de objetos que contienen imágenes, estos usan estructuras particulares de los bitmaps y de los vectores ajustados a las necesidades del objeto, aún mezcladas si es necesario.

codificación definido como típico, los formatos PDL casi siempre usan la codificación de sus datos en ASCII, teniendo los problemas en los requerimientos de memoria y procesamiento que acarrea tal codificación. La gran versatilidad de un PDL consiste en que los objetos pueden representar cualquier cosa: figuras tridimensionales, textos a imprimir, perturbaciones atmosféricas, estructuras moleculares, estrellas, galaxias, etc., y desplegarlas en cualquier máquina sin importar su plataforma. Los tipos de PDL que ejemplifico son especialmente diseñados para tratar imágenes.

1.5.3.1 PostScrip

Propietario: ***Adobe Systems, Inc.***
Extensión: ***.ps***

El formato PostScrip es uno de los más populares en la creación de documentos independientes del tipo de máquina, es decir, los documentos conservan una presentación uniforme en cualquier plataforma que sean leídos.

El formato Postscrip tiene cuatro variantes: nivel 1, nivel 2, Encapsulado y Display. El nivel 1 maneja el lenguaje básico, abarcando imágenes monocromáticas, de color RBG, CMYK con una profundidad máxima de ocho bits. El nivel 2 además de lo anterior incluye el uso opcional de codificación binaria en vez de la codificación ASCII, uso de colormetría y espacios de color, además aumenta la profundidad hasta 32 bits. El formato base soporta además imágenes vectoriales.

El archivo Postscrip Encapsulado (EPS) describe una sola página, la cual se concatena sin modificación en un gran documento Postscrip. Las diferencias entre el EPS y las variantes regulares se centran en la inclusión de campos de comentarios, la anulación de algunos operadores, y la eliminación de acciones que pueden afectar a todo el documento. El EPS es la variante diseñada para el intercambio de gráficos.

El Postscrip Display es la variante diseñada para facilitar el uso en sistemas operativos gráficos, ya que contiene comandos que facilitan su uso por sistemas multitáreas, la creación de ventanas apropiadas, etc.

Estructura
El formato Postscrip es un lenguaje de programación parecido al FORTH, por ello, el formato no exige nada especial más allá del uso correcto de la sintáxis y semántica del formato, pero para poder mantener una estructura estable en los distintos sistemas, Adobe ha formalizado una serie de convenciones en el documento "Document Structuring Conventions" (DSC), que organiza una sección de prólogo, seguida de una sección script.

Las recomendaciones DSC facilitan a los manejadores de documentos (procesadores de documentos, programas de impresión, etc.) el procesar los archivos postscrip. Las recomendaciones DSC incluyen una serie de instrucciones llamadas comentarios que aunque son ignoradas por el editor, son usadas por los manejadores; su falta puede hacer ilegible el archivo, ello es especialmente cierto para las variedades como el EPS y el Display. La estructura recomendada por Adobe es:

Prólogo (Prolog)
  Cabecera (Header)
  Defaults (opcional)
  Procedimientos (Procedures)

Script
Configuración del Documento (Document setup)
Páginas de código Postscrip
.
.
.
Terminación del documento (Document trailer)

La estructura es organizada mediante comentarios DSC, los cuales comienzan con dos signos de porcentaje (%), el comentario no ocupa más de 255 caracteres terminando con el primer CR (retorno de carro) o CR/LF (línea de alimentación) que halle. La línea completa es sensible a las mayúsculas / minúsculas. Muchos comandos terminan con punto y coma, los argumentos son delimitados por espacios.

Un subconjunto de los comentarios DSC, se encarga de delimitar las secciones del archivo postscrip. Por ser el postscrip un formato con una especificación muy extensa (el documento abarca cerca de 700 páginas) solo me referiré a la secciones que tienen que ver con las imágenes.

La estructura del formato simplificada a imágenes sin texto es:

Prólogo (Prolog)
Cabecera (Header)
Procedimientos (Procedures)

Script
Código Postscrip
Terminación del documento (Document trailer)

Prólogo

El *prólogo* es la primer sección de un archivo postscrip, contiene una cabecera, una sección de definiciones opcionales -que no son usadas en imágenes-, y por último, una sección de procedimientos.

Sección *Cabecera*: Esta sección contiene una serie de comentarios que describen la variante usada, y algunos comentarios que describen el tipo de imagen. El comentario para describir al archivo es:

%!PS-Adobe-3.0 *parámetro*

Donde el parámetro es el valor que describe la variante PostScrip, en el caso específico de imágenes el tipo PostScrip especializado es el EPS, la etiqueta es "EPSF-3.0" . El "3.0" es la versión más reciente de la variante.

Los siguientes comentarios también están incluidos en la versión 2.0:

%%BoundingBox: *Xlowleft Ylowleft Xupright Yupright* Este comentario define el área rectangular en la que el texto o imágenes son colocados.

%%LanguageLevel: *level* Esta línea es necesaria si operadores Postscrip nivel 2 son usados, en ese caso level vale 2 sin signo.

%%Extensions: *CMYK* Esta línea es usada por archivos de nivel 1 que usan codificación CMYK.

%%DocumentData: *Binary* Esta línea se añade si datos de ocho bits son usados. Conjuntamente con este comentario se usan los comandos %%BeginData y %%EndData en la parte Script.

%%EndComments Terminador de la sección de cabecera. Opcional.

Sección de procedimientos: En esta sección el usuario define procedimientos propios. La sección se encierra entre los siguientes comentarios:

%%BeginProlog y %%EndProlog

Código PostScrip
Inmediatamente después del prólogo, es el cuerpo principal del archivo, es donde es definida la página (texto o imagen).

Terminador
Esta sección contiene una mezcla de código PostScrip y comentarios DSC, que realizan operaciones de limpieza de variables y reinicialización de los dispositivos usados por el programa lector. La sección se inicia con el comentario %%Trailer.

EPS (Encapsulated PostScrip)
El postscrip encapsulado es la forma principal usada para el manejo e intercambio de imágenes. El archivo EPS no define más de una página, además, la cabecera debe tener los siguientes comentarios:

%!PS-Adobe-3.0 EPSF-3.0 y,

%%BoundingBox: *Xlowleft Ylowleft Xupright Yupright*

En caso de usar comandos más allá del nivel 1 el resto de los comentarios anteriormente mencionados debe usarse.

Estructura
El PostScrip es un lenguaje de descripción de página, es decir, que hace uso de una gramática, así como de un compilador, todas la instrucciones son leídas por el intérprete Postscrip, por medio del parser son transformadas en objetos, hay dos tipos de objetos: Ejecutables y no ejecutables, los ejecutables son aquellos comandos que realizan una operación ó bien, llaman a un procedimiento previamente definido por el usuario, son también conocidos como *operadores*. Los objetos no ejecutables son datos o parámetros alfanuméricos y booleanos son también conocidos como *literales*. Los comentarios siempre inician con el caracter %.

El PostScrip opera usando pilas. Hay cuatro pilas en el PostScrip: Operand, Dictionary, Execution y Graphic. Cuando un objeto no ejecutable (Dato) es leído inmediatamente se añade en la pila Operand, en el caso de un operador sea encontrado es ejecutado inmediatamente después de ser leído por el parser; en caso de tener parámetros, estos son añadidos en la pila Operand, así como sus resultados. Ello permite tener a los operandos en secuencia pasando información de uno a otro. los datos de la pila son susceptibles de ser duplicados, borrados, pero nunca puede ser limpiada por completo la pila en el archivo EPS.

Cuando el intérprete encuentra una cadena de signos regulares, considera que no es un dato y lo define como objeto nombre. Tales objetos son etiquetas únicas de operadores o nombran algún procedimiento, para distinguirlos se antepone un caracter slash (/). A veces son parámetros de otros operadores (o sea datos) en tal caso, tales objetos son añadidos a la pila de datos.

Operadores

Los operadores son de dos tipos: Internos y definidos por el usuario, los comandos internos han sido previamente definidos por Adobe en la pila Dictionary. Los procedimientos se acomodan en la sección del usuario de la citada pila. Para definir un procedimiento se pone el nombre del mismo con un slash que lo anteceda, inmediatamente se pone los parámetros y operadores encerrados entre llaves { }, y al final se pone el comando def que abrirá una nueva entrada en la pila Dictionary.

El contenido de las llaves es colocado en la pila Operand y son realizadas las operaciones que contiene, así cada vez que el comando sea invocado, el intérprete llamará los datos de la pila y realizará la operación.

Los tipos de datos que maneja el PostScrip son:

Numérico: los datos numéricos están en decimal, y pueden ser notación decimal con signo y con tantos dígitos como sea necesario, o bien, en notación científica.

Texto: El texto es encerrado entre paréntesis en ASCII, o entre < > en forma hexadecimal. Para los caracteres especiales, es posible usar los caracteres ASCII o una serie de caracteres especialmente codificados.

Cada caracter especial comienza con un backslash (\) seguido del caracter a representar:

b = backspace	f = alimentación de página
n = nueva línea	r = retorno de carro
t = tabulación	nnn = caracter en octal
\ = backslah	

Las cadenas de caracteres pueden ser usadas como buffers en una pila. Para ello se usa el comando:

Stringsize Cadena

donde Stringsize es un número entero, así el intérprete empuja x bytes la pila, llenando el espacio con ceros. Los arreglos se componen de múltiples cadenas y/o números, encerrados entre paréntesis cuadrados [ ], todos los objetos ejecutables se ejecutan tal como se encuentran.

Gráficas

Cuando un intérprete Postscrip ejecuta una imagen, primero lee una serie de parámetros que definen a la imagen, la gran mayoría solo especifican datos para los dispositivos que usa el programa lector.

Sistemas de Coordenadas: El Postscrip considera a las gráficas como áreas de tinta oscura sobre un plano blanco virtual. Este plano virtual es también conocido como espacio del usuario. Un segundo sistema coordenado, conocido como espacio de dispositivo, está relacionado con el espacio del usuario por medio de matrices de transformación de coordenadas (CTM, en inglés).

Gracias al CTM es posible transformar una imagen - escalarla, rotarla, invertirla, etc.-, por medio del álgebra lineal, evitando en la medida de lo posible las deformaciones típicas de las imágenes bitmap, aunque también es aplicado en imágenes tipo vector.

Color : El Postscrip acepta el uso de los colores en formatos RGB, HSB (HSI), y CMYK (únicamente en el nivel 2), para manejar tanto el color como la escala de grises, define para cada modelo una serie de espacios, es decir, descompone cada color en una imagen de ese color, de manera similar a los planos de color referidos anteriormente. Para usarlos, es necesario usar el comando */Decive*, así, la escala de gris es */DeciveGray*, el RGB */DeciveRGB* y así sucesivamente. En el caso de las imágenes vectoriales es usado el parámetro CurrentColor que dependiendo del modelo de color o gris usado, afecta a la imagen y el relleno de la misma.

Imágenes Vectoriales

El EPS usa una serie de comandos que definen a la imagen, generalmente son una serie de comandos que hacen el trazado de las líneas y arcos de que se compone la imagen, e igualmente el relleno de áreas de la imagen. Algunos de los comandos usados para estas imágenes son:

x y moveto - El puntero se coloca en la posición x, y sin dejar trazada ninguna línea, los valores son absolutos a la imagen y están dados en números reales.

dx dy moveto - Realiza lo mismo que el anterior comando, pero en base a incrementos relativos de los puntos x, y.

x y lineto - Traza una línea del punto inicial a los valores x, y.

x y angle1 angle2Arc - Traza un arco con centro en x, y, partiendo del ángulo dado por angle1 hasta angle2. La curva es definida siguiendo en sentido contrario a las manecillas del reloj.

closepath - Traza una línea en el camino que va del punto de inicio hasta el valor actual x, y.

Newpath - Elimina el camino previamente trazado y deja el punto actual indefinido.

fill - Rellena con el color en uso el área rodeada por el camino, si este no esta cerrado, el comando automáticamente lo cierra y llena el área.

stroke - Hace la misma operación que Fill pero las uniones entre líneas y su terminación de las mismas puede ser ajustado.

Al ser impresa, la imagen terminada se puede ajustar usando el CTM previamente mencionado.

Imágenes Bitmap

Las imágenes bitmap cuentan con una serie de parámetros que identifican a la sección de la imagen, la profundidad de la misma, así como el tipo imagen que es, color o escala de grises.

Los datos son almacenados de dos maneras: En ASCII hexadecimal o binaria, para la primera es usado el comando readhexstring, la lectura se hace hasta que el intérprete localiza un fin de archivo o hasta que se alcance el límite de la cadena, cualquier caracter no hexadecimal es ignorado. La codificación binaria sólo se contempla en el nivel 2 del formato, para usarse es necesario poner el comando /Datasource binary, a partir de este comando el intérprete leerá los datos en 7 bits en ASCII para máxima portabilidad.

Color

En el nivel 2 del postscrip, permite que las imágenes bitmap cuenten con una cabecera que describe el tamaño de la imagen, el tipo de codificación, el tipo de esquema de color usado; todo encerrado entre símbolos "<< >>". Para definir el tipo de esquema de color se usa el comando Ncomponents que cuando vale 1 es para escala de gris, 3 para RGB/HSB, y 4 para CMYK. Hay la opción de codificar en forma simple (los tres valores en un pixel) o en planos de color.

En el nivel 2, otra innovación que hay en el postscrip es la compresión de la imagen, por medio de filtros (programas de conversión, generalmente externos al programa principal) lee y codifica en ASCII85, LZW, RLE, y CCITTFax.

Otra opción añadida es tener una imagen reducida de la original - también conocida como preview-, esta imagen está almacenada en cualquiera de los formatos Macpaint, Windows Meta File, TIFF y en el formato EPSI diseñado para tal tárea. El formato tiene una cabecera dedicada a definir que formato está usándose, así mismo cuenta con los filtros para poder leer la imagen.

1.5.3.2 PCL

Propietario: ***Hewlett Packard***
Extension: ***.pcl***

El formato PCL (Hewlett Packard Printer Control Language), es ampliamente apoyado por las impresoras láser HP y compatibles, pero únicamente soporta imágenes en monocromático, la actual versión es la número 5. Cada nueva versión simplemente añade nuevas cosas sin dejar de apoyar a las antiguas versiones.

Estructura
Los comandos del formato PCL son de dos tipos. El primero consiste de dos caracteres, uno de ellos es el caracter "ESC" más un una letra o número, por ejemplo "ESC E" restablece la impresora (reset). Para comandos más complejos la composición es: El caracter "ESC" primero seguido de un caracter "&", después sigue una letra minúscula que indica un conjunto de comandos, opcionalmente se le acompaña una cifra decimal, y para concluir se añade la letra mayúscula que especifica el comando en particular, todos ellos separados por espacios.

Por ejemplo: "ESC & a 13 R" indica a la impresora posicionarse en el renglón 13, y "ESC & a 20 C" posiciona la impresora en la columna 20. Cuando hay comandos consecutivos que pertenecen a un mismo grupo, puede simplicarse concatenándolos con el caracter "r", dejando sólo los parámetros, así, del ejemplo anterior obtenemos: "ESC & a 13 R r 20 C", no hay límite para las concatenaciones posibles.

Los comandos básicos para manejar la impresora:

Tabla 1.45: Códigos básicos de control PCL

Codigo	Significado
ESC E	Restablecer de la impresora.
ESC & l10	Posiciona la impresora en forma horizontal, por definición la página tiene orientación vertical.
ESC & r 0 F	Define la posición de la imagen *horizontal o vertical*, para conservar la máxima compatibilidad se pone por default en forma horizontal. (Disponible sólo en la versión PCL 5)
ESC & a N C	Coloca el cursor en la columna N, medido en puntos
ESC & a N H	Coloca el cursor en la columna N, medido en decipoints
ESC * p N X	Coloca el cursor en la columna N, medido en renglones
ESC & a N R	Coloca el cursor en el renglón N, medido en puntos
ESC & a N V	Coloca el cursor en el renglón N, medido en decipoints
ESC * p N Y	Coloca el cursor en el renglón N, medido en renglones

En donde *N* si es de signo negativo, la distancia se comienza a contar de la parte izquierda, si es positivo el conteo se inicia a la derecha, de no tener signo la cifra, por definición se ajusta a la izquierda.

El PCL define tres tipos medidas: puntos (dots), decipoints, renglones y columnas. Un punto es 1/300 de pulgada, que es la resolución básica de las impresoras láser. Un decipoint es un décimo de un punto de impresora, es decir, 1/720 . En el caso de los renglones y columnas depende del tipo de página, para el renglón se define como

tal un décimo de pulgada, un sexto de pulgada define a la columna. El punto 0,0 está localizado en la parte superior derecha de la página.

La impresora a veces crea páginas virtuales, dejando por definición ½ pulgada de margen, asi en una página tamaño carta de 8 ½ x 11 pulgadas se crea una página virtual de 8 x 10 pulgadas que es el máximo espacio en el que se puede transcribir texto o colocar una imagen. La página puede ser impresa en forma vertical u horizontal, para las imágenes que son impresas en forma vertical por definición, sólo hasta la versión 5 es posible imprimir en cualquier orientación.

Las imágenes son manejadas dentro del formato como un juego de instrucciones especiales. Las imágenes en PCL pueden ser impresas en 600, 300, 150, 100 o 75 puntos por pulgada (DPI en inglés) de resolución, obviamente que a mayor resolución se obtiene una imagen de mejor calidad. La velocidad de impresión depende mucho de la resolución. Por ejemplo, una imagen de 300 dpi ocupa alrededor de 900 k tomando varios minutos imprimir una imagen, mientras que una imagen de 75 dpi del mismo tamaño que la de 300 dpi ocupa 1/16 del espacio y tarda 16 veces menos en ser impresa.

Las instrucciones para manejar una imagen son:

ESC * t *N* R - Determina la resolución de la imagen, donde *N* vale 75, 100, 150, 300.

ESC * r *W* T - Alto de la imagen

ESC * r *H* S - Ancho de la Imagen

Por ser introducidos en la versión 5, los dos últimos comandos son ignorados por las anteriores versiones del formato, además, estos invariablemente se escriben en notación combinada. Por ejemplo, una imagen de 100 x 200 se escribirá:

ESC * r 100 t 200 S

Para los datos de la imagen se usa un comando de apertura, una serie de comandos relacionados con la imagen en sí misma, un comando de terminación. El comando de inicio es :

*ESC * r 1 A* - Si se sustituye el 1 con 0 la imagen se apega a la parte izquierda de la hoja

El comando de los datos de la imagen por renglón es:

*ESC * r N* W *datos de la imagen* - *N* indica el número de bytes que hay por renglón. Este comando no puede ser combinado.

El comando de terminación del área de la imagen es:

ESWC * r B

En el nivel 4 se incluye la modalidad de usar métodos de compresión, para el nivel 5 se añade la opción de simplificar la cantidad de líneas en blanco repetidas. Para el primer caso, cada renglón es comprimido individualmente, dándose la posibilidad de usar distintos métodos en una sola imagen. El comando es :

ESC * b *N M*

En donde M son los bytes de los cuales consta el renglón de la imagen, y N el tipo de compresión - 0: sin compresión, 1: Run Length, 2: Pack Bits, 3: Delta, 5: Adaptativo-, en el tipo 0 los datos son enviados sin compresión alguna. Para el tipo 1 y 2 veáse compresión Run Length, del cual previamente hemos hablado.

El tipo de compresión Delta consiste en enviar las partes que difieren del renglón anterior, el valor de control es binario dividéndose en dos partes. La estructura del byte es LLLPPPPP donde LLL es el número de repeticiones despues del bloque, el PPPPP indica la posición donde inicia la variación del renglón; así cada valor binario de control, es seguido de una serie de variaciones en hexadecimal. El último tipo usa los anteriores métodos de compresión pero puede abarcar varias líneas, tantas como lo permitan el límite de 32767 bytes para un bloque.

En el caso de tener una serie de *N* renglones en blanco, para simplificar el número de renglones se usa el comando (nivel 5):

ESC * b *N* Y

Finalmente el formato maneja instrucciones para crear cuadros y hacer rellenos, elementos apreciados en la construcción de páginas atractivas. Para hacer el cuadro en sus líneas horizontales:

ESC * c *N* V (en puntos)
ESC * c *N* B (en decipoints)

y para sus líneas horizontales:

ESC * c *N* H (en puntos)
ESC * c *N* A (en decipoints)

Recuérdese que pueden ser combinados, pero únicamente aquellos que tengan el mismo tipo de medida. Para rellenar este cuadro se puede hacer de dos maneras: Primero, se puede obscurecer el área dentro del cuadro, PCL ofrece 100 tonos que van del blanco puro (0) hasta el negro (100), el comando usado es:

ESC * *N* G

Donde *N* es el número de tono. El otro tipo de relleno es rayar el área del cuadro. Hay 6 tipos de relleno: 1 - líneas horizontales, 2 - líneas verticales, 3 - líneas inclinadas a la derecha, 4 - líneas inclinadas a la izquierda, 5 - cuadriculado vertical, 6 - cuadriculado en ángulo. El comando usa el valor N donde los anteriores seis tipos.

ESC * c *N* P

1.5.3.3 VRML

Propietario: *Silicon Graphics, Inc.*
Extensión: *.vml .vrml*

En la primera conferencia anual de *World Wide Web*, realizada en Génova, Suiza (1994). Un grupo de trabajo encabezado por Gavin Bell de *Silicon Graphics, Inc.*, Anthony Parisi de *Intervista Software.*, y moderado por Mark Pesce, comenzó los trabajos para el desarrollo de un formato que permita la transferencia, almacenamiento y tratamiento de información relacionada con mundos virtuales.

Tal información debe ser libre de usarse sin importar el software o la plataforma en que esté el usuario, y sea factible de ser usado en el WWW. Del trabajo desarrollado, se crea el Formato VRML (*Virtual Reality Modeling Language*; Lenguaje de modelado de realidad virtual), el VRML es un lenguaje para describir simulaciones interactivas con múltiples participantes: Mundos virtuales a través del internet. El cual tiene una estructura muy similar al HTML (*HyperText Markup Language*, Lenguaje de marcado de hipertexto) que es un lenguaje de descripción de página, estándar en la comunidad WWW.

El VRML al igual que el HTML, es un lenguaje de descripción de página, es decir, está conformado por una serie de parámetros, comandos e instrucciones, las cuales les permite describir un objeto (página de texto, mundo virtual, etc.). En el caso específico del VRML describe un mundo virtual, tal mundo se conforma de un área en tres dimensiones, en el cual existen objetos manipulables por el usuario.

Cada objeto tiene características propias, y puede realizar acciones. El formato VMRL es un formato gráfico ya que al crear mundos virtuales, es necesario crear objetos 3D y 2D usando imágenes bitmap y/o vector. El VRML a diferencia del HTML tiene parámetros, valores y estructuras para manejar directamente sus imágenes 3D que define como objetos, también cuenta con soporte para hacer uso de imágenes bitmap o vectoriales.

Especificación VRML
La especificación VRML fue dada a conocer en mayo de 1995 por Silicon Graphics siendo esté el propietario formal del formato. De manera muy abstracta, El VRML es un método por el cual los objetos pueden ser leídos y escritos. Teóricamente, los objetos pueden contener cualquier cosa - MIDI, Imágenes 2D y 3D, etc.-. El estándar define un conjunto de objetos usados para hacer gráficas 3D. Esos objetos son llamados nodos.

Castillo Medieval, por André Wezasak.[15]

Los nodos están arreglados en estructuras jerárquicas llamadas escenas gráficas. Las escenas gráficas son más que una mera colección de nodos, pues definen un orden para los nodos. La escena gráfica tiene una noción de estados, es decir, los primeros nodos en la escena pueden afectar a nodos que aparecen posteriormente en la escena. Por ejemplo un nodo de rotación o de materiales afectará a todos los nodos posteriores. Un mecanismo es definido para limitar los alcances de los efectos (nodos separadores), el cual permite que partes de la escena gráfica actúen de manera aislada en relación al todo.

Un nodo tiene las siguientes características:

- *Tipo de objeto.* Un nodo puede ser un cubo, una esfera, un plano, un mapa de texturas, una transformación, etc.
- *Los parámetros que distinguen este nodo de otros del mismo tipo.* Por ejemplo, cada nodo esfera tiene diferentes radios, y diferentes mapas de texturas que contendrán diferentes imágenes. Los parámetros son conocidos como campos. Un nodo puede tener muchos campos o ninguno.
- *Un nombre identifica al nodo.* Si bien debe haber un sólo nombre en cada nodo, el formato permite usar el mismo nombre para distintos nodos.
- *Nodos hijos.* Un objeto jerárquico está implementado para permitir que algunos tipos de nodos contengan a otros nodos. Los nodos padres recorren sus nodos hijos durante el despliegue del objeto. Los nodos que pueden tener nodos hijos son definidos como nodos de agrupación.

La sintaxis para representar a un nodo es:

nombreobjeto tipodeobjeto {campos nodos hijos}

Únicamente el tipo de objeto y las llaves son obligadas, todo lo demás es opcional. Los nombres de los nodos no deben comenzar con un dígito, tampoco deben tener espacios, caracteres especiales, comillas, ni ningún otro tipo de caracter que no sea parte del alfabeto usado en el inglés.

[15] LightScape Technologies.

Sintaxis general
Para facilitar la identificación de los archivos VRML, cada archivo debe iniciar con la cadena:

#VRML V1.0 ascii

El caracter "#" inicia un comentario, cualquier cosa en esa línea será ignorado, la línea termina con caracter de retorno de carro. Los caracteres de espacio, tabulación y retorno de carro son ignorados salvo donde se especifique. Después de la cabecera un archivo VRML contiene exactamente un nodo VRML.

Sistema de coordenadas
El VRML usa un sistema de coordenadas tridimensional con el criterio de la mano derecha. Por definición, los objetos son proyectados en un dispositivo bidimensional, los ejes positivos Z, X, Y en proyección, hacia la derecha y hacia arriba respectivamente.

Una cámara o una transformación de modelado pueden alterar la proyección. La unidad estándar para las longitudes y espacios es el metro. La medida estándar para los ángulos son los radianes.

Campos
Hay dos clases generales de campos; campos que contienen un solo valor (donde el valor puede ser un número, un vector, o una imagen), y los que pueden tener múltiples valores. Los nombres de los campos de un valor comienzan con "SF", los de valor múltiple inician con "MF". Cada campo define el tipo de formato para sus valores. En los campos de valores múltiples, estos son escritos como una serie de valores separados por comas -con excepción del ultimo valor-, y están encerrados entre paréntesis cuadrados "[ ]". En caso de no haber valores se ponen los paréntesis cuadrados juntos. Si hay únicamente un sólo valor pueden ser eliminados los paréntesis cuadrados sin ningún problema.

SFBitMask
SFbitmask es un campo de un sólo valor que contiene un conjunto de banderas de un sólo bit. El nodo que usa esta clase de campo define nombres nemotécnicos para las banderas. Las banderas son escritas en el campo en una lista de nombres nemónicos en este formato:

(bandera1 | bandera2 | ...)

SFBool
Es un tipo de campo que contiene un solo valor booleano. Por ende, éste campo acepta 1 y 0 (verdadero y falso respectivamente), además de los valores predefinidos TRUE y FALSE.

SFColor
Este campo contiene un color. Los campos SFColors son escritos como una tripleta RGB de números de punto flotante en notación científica estándar. El rango permitido de cada valor RGB es de 0.0 a 1.0.

SFEnum
Éste es un campo numérico. Los nodos que usan este campo definen número nemotécnico para los valores. Escrito del mismo modo que los valores del campo SFBitMask, su nombre difiere según debido al uso de este campo en varias clases de nodos.

SFFloat
Campo que contiene un número de simple precisión de punto flotante escrito en notación científica.

SFImage
Un campo que puede contener una imagen de color o en escala de grises bidimensional sin comprimir. Las imágenes son escritas en el archivo como tres enteros que representan el ancho, el alto y el número de componentes en la imagen, seguidas un número de valores hexadecimales que representan a cada uno de los pixels, separados por espacios. El número de pixels se calcula multiplicando alto*ancho de la imagen.

Los valores hexadecimales se escriben en notación 0xvalor_hexadcimal. Una imagen de un componente va a tener un valor hexadecimal que representa la intensidad de la imagen. Por ejemplo, 0xFF es máxima intensidad mientras que 0x00 es intensidad nula. Una imagen de dos componentes pone la intensidad en el primer byte y la transparencia en el segundo. Una imagen de tres componentes (RGB) tiene el componente rojo en el primer byte siguiéndole el componente verde y el componente azul (así 0xFF0000 es rojo).

Las imágenes de cuatro componentes ponen el byte de transparencia después de los componentes RGB. Un valor de 1.0 es completamente transparente (véase transparencia en GIF) y el 0.0 es completamente opaco. Los valores de los pixels son enteros sin signo en formato MSB. Los pixels están ordenados de izquierda a derecha y de abajo hacia arriba, así, el primer pixel es el pixel en la esquina inferior izquierda, mientras que el último ocupa la posición superior derecha.

SFLong
Es un campo que contiene un entero largo. Este valor se puede escribir en forma decimal, hexadecimal (iniciando con "0x") u octal (iniciando con "0").

SFMatrix
Este campo contiene una matriz de transformación. Las matrices son escritas en forma de lista de 16 números de punto flotante agrupados por renglones, comenzando por el renglón superior. Los valores están separados por espacios. Por ejemplo, una matriz que expresa la traslación de 7.3 unidades a lo largo del eje X es escrito como:

1 0 0 0 0 1 0 0 0 0 1 0 7.3 0 0 1

SFRotation
El campo contiene una rotación arbitraria. Las rotaciones están escritas como cuatro números de punto flotante separados por espacios. Los cuatro valores representan un eje de rotación seguida de la cantidad de rotación (según el criterio de la mano derecha) de los ejes en radianes. Por ejemplo, una rotación de 180 grados en el eje Y es:

0 1 0 3.14159265

SFString
Campo que guarda una cadena ASCII. La cadena está encerrada entre comillas (""). Cualquier caracter (incluyendo el de línea nueva) debe estar encerrado entre las comillas. Para incluir comillas en el texto es necesario anteponerles un caracter backslash (\).

SFVec2f
Campo que contiene un vector bidimensional compuesto de dos valores de punto flotante separados por espacios.

SFVec3f
Campo que contiene un vector bidimensional compuesto de tres valores de punto flotante separados por espacios.

MFColor
Un campo múltiple que contiene cualquier número de colores RGB. Los colores son escritos como una o más tripletas RGB de números de punto flotante en notación científica separados por comas.

MFLong
Es un campo múltiple que contiene cualquier número de enteros largos (de 32 bits). Los datos pueden escribirse en decimal, hexadecimal u octal.

MFVec2f
Un campo múltiple que contiene cualquier cantidad de vectores bidimensionales. Los valores están escritos como uno o más pares de valores de punto flotante separados por espacios.

MFVec3f
Un campo múltiple que contiene cualquier cantidad de vectores tridimensionales. Los valores están escritos como una o más triadas de valores de punto flotante separados por espacios.

Nodos

El formato VRML define varias clases de nodos. La mayoría de los nodos pueden ser clasificados dentro de una de las tres categorías: de forma (Shape), de propiedad o de agrupación. Los nodos de forma definen la geometría en la escena.

Los nodos de propiedad afectan la manera en que las formas son dibujadas., y los nodos de agrupación juntan los nodos unos con otros, permitiendo que colecciones de nodos sean tratadas como un solo objeto. Algunos nodos también controlan si sus hijos son o no dibujados.

Los nodos pueden contener muchos campos o ninguno. Cada nodo define el tipo, nombre y valor de defecto para cada uno de sus campos. El valor por definición es usado si el valor de un campo no está especificado en el archivo. Los campos de un nodo no necesitan estar en un orden estricto.

Hall por André Wezasak.[16]

Hay 36 nodos principales agrupados por tipo.

Primer grupo, Nodos de forma:
AsciiText, Cone, Cube, Cylinder, IndexedFaceSet, IndexedLineSet, PointSet, Sphere.

Segundo grupo, Nodos de propiedades:
Subgrupo de geometria y aparencia: Coordinate3, FontStyle, Info, LOD, Material, MaterialBinding, Normal, NormalBinding, Texture2, Texture2Transform, TextureCoordinate2, ShapeHints.

SubGrupo de Transformaciones: MatrixTransform, Rotation, Scale, Transform, Translation

Subgrupo de cámaras: OrthographicCamera, PerspectiveCamera

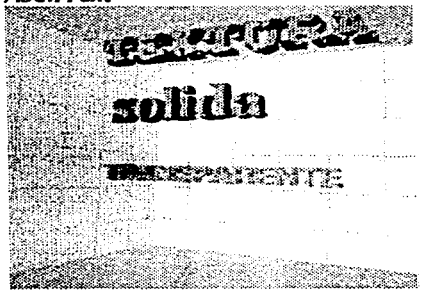
Subgrupo de Luces: DirectionalLight, PointLight, SpotLight-.

[16] LightScape Technologies.

Tercer grupo, nodos de agrupación: Separator, Switch, TransformSeparator, WWWAnchor.

Finalmente el nodo WWWInline no cae en ninguna categoría.

AsciiText

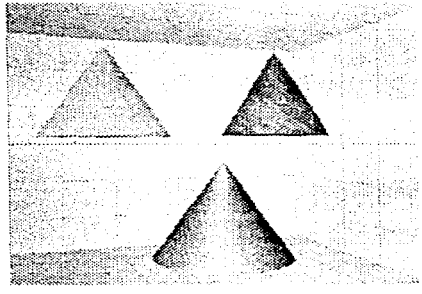


Este nodo representa cadenas de caracteres ASCII. La primer cadena es presentada con base en (0,0,0). Todas las cadenas subsecuentes se acomodan sucesivamente en el eje Y. (Véase el nodo FontStyle para una descripción del tamaño de los caracteres y su espaciado). El campo justificación determina la posición de la cadena en x dimensión. Left (valor por definición) pone la esquina izquierda de cada cadena en x = 0. CENTER pone el centro de la cadena en el valor x = 0. RIGHT pone la esquina derecha de la cadena en x = 0. El texto es desplegado de izquierda a derecha y de arriba a abajo, usando la fuente determinada en el nodo FontStyle.

Valores por definición del nodo.

```
AsciiText { string        ""      # MFString - Cadena de texto
        spacing       1       # SFFloat - Espaciado entre caracteres
        justification LEFT    # SFEnum - Justificación
        width         0 }     # MFFloat - Ancho del texto.
```

Cone



Este nodo representa un simple cono cuyos ejes centrales están alineados con el eje Y. Por definición el cono está en el centro (0,0,0) y tiene un tamaño de -1 a +1 en las tres direcciones. El cono tiene un radio de 1 en su base y un alto de 2, con su punta en 1 y su base en -1.

El cono es dibujado con la textura y el material indicado en ese momento. Si el material usado es PER_PART o PER_PART_INDEXED, el primer material es aplicado para los lados del cono, y el segundo material es usado para la base. De otro modo, el primer material será usado para todo el cono. Cuando una textura es aplicada a un cono, se aplica a los lados de manera diferente que a la base. En los

lados, la textura los envuelve en sentido contrario a las manecillas del reloj iniciando en el borde inferior. Para la base se "recorta" un cuadro de la textura y se aplica en la base del cono.

Partes del cono

SIDES	Lados del cono
BOTTOM	Base del cono
ALL	Todas las partes del cono.

Valores por definición del nodo.

```
Cone { parts        ALL   # SFBitMask - que parte del cono es señalado.
       bottomRadius  1    # SFFloat  - Radio de la base
       height        2 }  # SFFloat  - Altura del cono
```

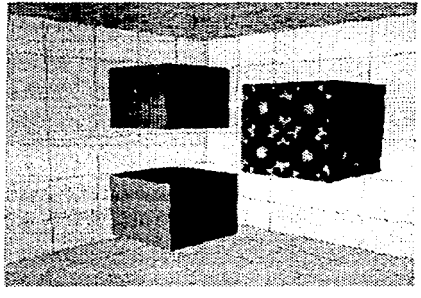
Coordinate3

Este nodo define un conjunto de coordenadas tridimensionales para ser usados por los nodos subsecuentes IndexedFaceSet, IndexedLineSet, PointSet. Este nodo no produce resultados visibles durante el desplegado, simplemente reemplaza las coordenadas para los subsecuentes nodos.

Valores por definición del nodo.

```
Coordinate3 { point  0 0 0 } # MFVec3f
```

Cube

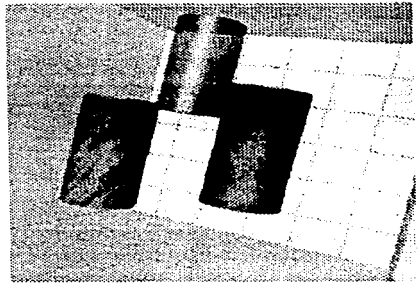


El nodo representa a un paralelepípedo cúbico alineado con los ejes coordinados. Por definición el cubo está en el punto inicial (0,0,0) y mide dos unidades en cada dimensión (o eje), de -1 a1. Al igual que otros objetos puede ser transformado y texturizado. Si el material es PER_PART, PER_PART_INDEXED, PER_FACE, o PER_FACE_INDEXED, serán colocados en las caras del cubo en este orden: frente (+Z), atrás (-Z), izquierda (-X), derecha (+X), arriba (+Y), abajo (-Y). Las texturas son aplicadas a cada cara del cubo.

Valores por definición del nodo.

```
Cube { width  2    # SFFloat / Ancho
height 2    # SFFloat / Alto
depth  2 } # SFFloat / profundidad
```

Cylinder



El nodo representa un cilindro centrado en el eje Y. Por definición, el cilindro está centrado en el origen y va de -1 a +1 en sus tres dimensiones. El cilindro tiene tres partes: Los lados, la parte de arriba y la base. Se pueden usar los campos de radio y alto para cambiar sus medidas.

Si el material usado es PER_PART o PER_PART_INDEXED, el primer material es aplicado para los lados del cilindro, el segundo material es usado para el tope y el tercer material es usado para la base. De otro modo, el primer material será usado para todo el cilindro. Cuando una textura es aplicada a un cilindro, esta es aplicada a los lados de manera diferente que a la base y la parte de arriba. En los lados, la textura los envuelve en sentido contrario a las manecillas del reloj iniciando en la base. Para la base y arriba se "recorta" un cuadro de la textura y se aplica.

Partes del cilindro

SIDES	La parte cilíndrica
TOP	La cara superior
BOTTOM	La cara inferior
ALL	El objeto completo

Valores por definición del nodo.

```
Cylinder { parts   ALL    # SFBitMask - parte del cilindro
           radius  1      # SFFloat - radio de la base
           height  2 }    # SFFloat - alto del cilindro
```

DirectionalLight

Este nodo define una fuente de iluminación que alumbra en rayos paralelos a un vector tridimensional dado. Un nodo de luz define una fuente de iluminación que puede afectar las formas subsecuentes en la escena gráfica, dependiendo del estilo de luz usado. Si un nodo de luz está bajo un separador, el nodo no puede afectar a ningún objeto fuera del separador.

Valores por definición del nodo.

```
DirectionalLight {on         TRUE      # SFBool  /activación del nodo
      intensity     1        # SFFloat /brillantez de la luz usada
      color         1 1 1    # SFColor /color de la luz
      direction     0 0 -1 } # SFVec3f /dirección de la fuente de luz (eje x)
```

FontStyle
Define el estilo de la fuente usada en nodos AsciiText subsecuentes. Únicamente define los atributos de la fuente. Se permite al programa asignar fuentes propias a varias combinaciones de atributos. El campo de tamaño especifica el alto de los caracteres y su espaciado entre ellos.

Familias definidas
SERIF Estilo Serif (Tal como la fuente TimesRoman).
SANS Estilo Sans Serif (Parecida a la Helvética).
TYPEWRITER Estilo de separación monoespaciada (Similar a la Courier).

Estilo de la fuente
NONE Tal como está definida la fuente.
BOLD **Enfatizado**
ITALIC *Itálicas*

Valores por definición del nodo.
```
FontStyle { size      10   # SFFloat  - Tamaño de la fuente en puntos por pixel
   family  SERIF           # SFEnum   - Tipo de fuente usada
   style   NONE }     # SFBitMask - Estilo de la fuente.
```

Group
Este nodo es la base de todos los nodos de agrupación. Un nodo de agrupación es aquel que contiene una lista ordenada de nodos hijos. Este nodo simplemente es un contenedor para los nodos hijos y no permite alterar cualquier estructura de ninguna forma. Durante la lectura, el estado (atributo) acumulado por un hijo es pasado a cada hijo sucesivo.

Valores por definición del nodo.
```
Group { }
```

IndexedFaceSet
Este nodo representa una figura 3D formada por caras (polígonos) desde vértices localizados en las coordenadas en curso. El nodo IndexedFaceSet usa los índices de su campo coordIndex para especificar las caras del polígono. Un índice de -1 indica que la cara en curso ha terminado y la siguiente comienza. El tratamiento del material (caras recubiertas por texturas) y la textura solida (caras con un color sólido, es decir sin patrón alguno) es como sigue: Los vínculos PER_PART y PER_FACE especifican un material o una textura solida para cada cara. PER_VERTEX especifica un material o textura solida para cada vértice. Los vínculos anteriores con terminación _INDEXED se comportan igual a los que no tiene la terminación, pero deben usar los índices correspondientes materialIndex o normalIndex. El vinculo usando material es igual a los valores de PER_VERTEX_INDEXED; si hay menos texturas solidas que los necesarios en el momento, los vértices normales serán generados automáticamente.

Las coordenadas de la textura (como está definido en TextureCoordinate2) puede ser vinculado a los vértices de una figura indexada usando los índices del campo

TextureCoordIndex. Como todas las figuras basadas en vértices, si hay una textura en proceso pero no tiene especificadas las coordenadas de textura, un mapa de coordenadas predefinido es calculado usando la "caja" que está vinculada a una figura (la caja es un prisma rectangular que encierra a la figura y sirve de referencia). La dimensión más grande de la caja define las coordenadas S, y la siguiente en tamaño define las coordenadas T. El valor de los rangos de la coordenada S van de 0 a 1 (de borde a borde). Los rangos de la coordenada T obviamente son menores que las coordenadas S.

Valores por definición del nodo.

```
IndexedFaceSet {coordIndex      0  # MFLong - Índice de coordenada
        materialIndex          -1  # MFLong - Índice de materiales
        normalIndex            -1  # MFLong - Índice de vínculos normales
        textureCoordIndex      -1 } # MFLong - Índice de coordenadas de la textura.
```

IndexedLineSet

Este nodo representa una figura 3D formada por líneas poligonales desde un vértice localizado en las coordenadas en curso. Este nodo usa los índices de su campo coordIndex para especificar las líneas poligonales. Un índice de -1 indica que la línea ha terminado y empieza la otra. Tanto las figuras a base de caras como de las figuras a base de líneas, son susceptibles de ser afectadas por una transformación.

El tratamiento del material en curso y de los vínculos normales es como sigue: El vinculo PERT_PART especifica un material o textura solida para cada línea polígonal. PER_VERTEX especifica el material o textura solida para cada vértice. Las versiones con terminación _INDEXED son similares a las que no los tienen, pero deben de usar los índices materialIndex o normalIndex. El material por definición es igual a OVERALL. La textura solida es por definición es igual a PER_VERTEX_INDEXED; si faltan vínculos normales en una etapa dada, las líneas se dibujarán sin iluminación. Las mismas reglas para la generación de texturas como IndexedFaceSet son usadas.

Valores por definición del nodo.

```
IndexedLineSet { coordIndex          0   # MFLong - Índice de coordenadas
      materialIndex          -1  # MFLong - Índice de materiales
      normalIndex            -1  # MFLong - Índice de vínculos normales
      textureCoordIndex      -1 } # MFLong - Índice de coordenadas de la textura
```

Info

Esta clase define un nodo de información en la escena gráfica. Este nodo este nodo no actúa en el desplegado del mundo virtual. Es usado para almacenar información de la escena gráfica, usado típicamente por aplicaciones específicas, guardan mensajes de derechos de copia u otras cadenas de texto:

Valores por definición del nodo.

```
      Info {string "<Undefined info>" }    # SFString - cadena de texto
```

LOD

Este nodo de agrupación es usado para permitir a las aplicaciones cambiar entre varias representaciones de los objetos automáticamente. Los hijos de este nodo típicamente representan el mismo objeto u objetos en diferentes niveles de detalle, de un alto nivel de detalle a un bajo nivel.

El punto central especificado del nodo LOD es modificado por la transformación en el espacio, y la distancia desde el centro transformado hasta el punto de vista es calculada. Si la distancia es menor al primer valor en el arreglo del rango, entonces el primer hijo del grupo LOD es dibujado. Si la distancia está entre el primer y segundo valor del arreglo, el segundo hijo es dibujado, y así sucesivamente.

Si hay N valores en los rangos del arreglo, el grupo LOD debe tener N+1 nodos hijos. Si hay menos nodos hijos que índices, el ultimo nodo hijo será repetido hasta el nivel más bajo de detalle (o sea , hasta terminar con los índices); si hay más nodos hijos que índices, los nodos extra serán ignorados., Cada valor en los rangos del arreglo debe ser menor que el valor previo, de lo contrario el resultado será imprevisible.

Valores por definición del nodo.

```
LOD { range    [ ]     # MFFloat - Arreglo de Rangos
      center  0 0 0 }  # SFVec3f - Punto central de la figura
```

Material

Este nodo define las propiedades de superficie del material para todas las subsecuentes figuras. El nodo Material define algunos de los componentes del material en uso durante el desplegado del mundo virtual. Diferentes figuras interpretan los materiales con múltiples valores de manera diferente. Para vincular los materiales a las figuras, se usa el nodo MaterialBinding.

Valores por definición del nodo.

```
Material { ambientColor  0.2 0.2 0.2    # MFColor - color de entorno
     diffuseColor    0.8 0.8 0.8    # MFColor - difusión del color
     specularColor     0 0 0        # MFColor - valor de especulación del color
     emissiveColor     0 0 0        # MFColor -cantidad de emisión del color
     shininess          0.2         # MFFloat -brillo
     transparency         0   }     # MFFloat -transparencia
```

MaterialBinding

El nodo define cómo es vinculado el material a las figuras que siguen en la escena gráfica. Cada figura interpreta la vinculación de manera diferente. Los materiales tienen un valor base, el cual es definido por el primer valor del campo de materiales. Como los campos de materiales tienen múltiples valores, la vinculación determina como esas figuras son distribuidas sobre la figura.

Los vínculos para las caras y los vértices son significativos únicamente a las figuras que hacen uso de ellos. Similarmente, los índices de vínculos son usados sólo por las figuras que los soportan. Cuando múltiples valores de un material son vinculados, los valores son reciclados basados en el periódo del componente del material con más valores. Por ejemplo, la siguiente tabla muestra los valores usados en el reciclado de

un material con dos colores ambientales, 3 colores de difusión, y uno o todos los otros componentes del material (el periódo de reciclado del material es 3):

Tabla 1.46: Valores típicos de un material

Material	Color de ambiente	Difusión del color	Otros campos
0	0	0	0
1	1	1	0
2	1	2	0
3	0	0	0

Tipos de vinculo

DEFAULT	Usa el vinculo por default.
OVERALL	El objeto completo tiene el mismo material
PER_PART	Un material para cada parte del objeto
PER_PART_INDEXED	Un material para cada parte del objeto, usando índices
PER_FACE	Un material para cada cara del objeto
PER_FACE_INDEXED	Un material para cada cara del objeto, usando índices
PER_VERTEX	Un material para cada vértice del objeto
PER_VERTEX_INDEXED	Un material para cada vértice del objeto, usando índices

Valores por definición del nodo.
MaterialBinding {value DEFAULT } *#SFEnum - Vinculo para enlazar un material con un objeto*

MatrixTransform

El nodo define una transformación geométrica en 3D usando una matriz de 4x4. Algunas matrices pueden llevar al programa a tener errores.

Valores por definición del nodo.

```
MatrixTransform {
        matrix   1 0 0 0      # SFMatrix
                 0 1 0 0
                 0 0 1 0
                 0 0 0 1   }
```

Normal

Este nodo define un conjunto de vectores tridimensionales de superficie normal, es decir, sin textura ni greca alguna, solamente colores puros. Los vectores son usados por los vértices de nodos de figuras (IndexedFaceSet, IndexedLineSet, PointSet) en la escena gráfica. Este nodo no presenta resultados visibles en la escena; simplemente reemplaza los valores de la normal para los nodos subsecuentes.

Valores por definición del nodo.

```
    Normal { vector  0 0 1 }      # MFVec3f - valor de la normal.
```

NormalBinding

Este nodo especifica como los vínculos normales se ligan a las figuras. Los vínculos para las caras y los vértices son significativos únicamente a las figuras que hacen uso de ellos. Similarmente, los índices de vínculos son usados sólo por las figuras que los soportan.

Tipos de Vinculo

DEFAULT	Usa el vinculo por default.
OVERALL	El objeto completo tiene la mismo textura solida
PER_PART	Una textura solida para cada parte del objeto
PER_PART_INDEXED	Una textura solida para cada parte del objeto, usando índices
PER_FACE	Una textura solida para cada cara del objeto
PER_FACE_INDEXED	Una textura solida para cada cara del objeto, usando índices
PER_VERTEX	Una textura solida para cada vértice del objeto
PER_VERTEX_INDEXED	Una textura solida para cada vértice del objeto, usando índices

Valores por definición del nodo.

```
NormalBinding { value  DEFAULT }   # SFEnum - Tipo de textura solida
```

OrthographicCamera

Una cámara ortográfica define una proyección paralela desde un punto de vista. Esta cámara no disminuye los objetos con la distancia, como lo haría el nodo PerspectiveCamera. El punto de vista del volumen para una cámara ortográfica es un paralepipedo rectangular. Por definición, la cámara está localizada en (0,0,1) y mira a través del eje Z negativo; los campos de posición y orientación pueden variar la posición de la cámara. El campo de altura define el alto total de la vista del volumen.

Una cámara puede ser puesta en un mundo virtual para especificar la localización en la cual el observador "entrará" al mundo virtual. Se permite a los programas modificar la cámara ayudando al usuario moverse a través del mundo virtual. Las cámaras también son afectadas por las transformaciones, pero puede colocarse el nodo de transformación después de la cámara.

Valores por definición del nodo.

```
OrthographicCamera { position          0 0 1    # SFVec3f - Posición de la cámara
      orientation      0 0 1  0  # SFRotation  - Orientación de la cámara
      focalDistance    5          # SFFloat   - Distancia focal
      height           2  }      # SFFloat   - Altura del área de visión de la cámara
```

PerspectiveCamera

Una cámara de perspectiva define una perspectiva desde un punto de vista. El punto de vista de esta cámara es una proyección igual a una pirámide truncada. Por definición la cámara está localizada en (0,0,1) y mira a lo largo del eje Z negativo. Al igual que en la cámara ortográfica, los campos de posición y altura pueden se usan para modificar las condiciones y posición de la cámara. Véase la cámara ortográfica.

Valores por definición del nodo.

```
PerspectiveCamera {position   0 0 1    # SFVec3f - Posición de la cámara
      orientation      0 0 1  0      # SFRotation - Orientación de la cámara
      focalDistance    5             # SFFloat  - Distancia focal
      heightAngle      0.785398 }    # SFFloat  - Altura del ángulo de la cámara
```

PointLight

Este nodo define una posición donde está una fuente de luz que ilumina un área tridimensional. La fuente de luz ilumina igualmente en todas direcciones; o sea, la fuente es omnidireccional. Un nodo de luz define una fuente de iluminación que puede afectar las subsecuentes figuras en la escena gráfica, dependiendo del estilo de luz usado. Las fuentes de luz son afectadas por las transformaciones. Un nodo de luz bajo un separador no afecta a nada que esté fuera del separador.

Valores por definición del nodo.

```
PointLight { on        TRUE    # SFBool  - Indica si está activa la fuente de luz
     intensity   1          # SFFloat - Intensidad de la luz
     color       1 1 1      # SFColor - Color de la luz
     location    0 0 1 }  # SFVec3f - Posición de la fuente de luz
```

PointSet

Este nodo representa un conjunto de puntos localizados en las coordenadas en ese momento usadas. El nodo PointSet usa las coordenadas en secuencia, iniciando el índice con el campo startIndex. El número de puntos en el conjunto es especificado por el campo numPoints. Un valor de -1 en éste campo indica que todos los valores restantes en la coordenadas serán usados como puntos. Los puntos son dibujados con el material y textura en uso. El tratamiento del material o vínculos normales es como sigue: PER_PART, PER_FACE, y PER_VERTEX son vinculados a cada punto. El valor del material es igual al campo OVERALL. El valor por definición de la textura normal es igual a PER_VERTEX. El campo startIndex es también usado por materiales o vínculos normales quienes indican el tipo de textura que debe ser usado por vértice.

Valores por definición del nodo.

```
PointSet { startIndex   0       # SFLong  - valor inicial del índice
  numPoints  -1 }     # SFLong  - número de puntos
```

Rotation

Este nodo define una rotación tridimensional con respecto a un eje arbitrario a través del origen. La rotación es acumulada en la transformación en curso, la cual es aplicada a las subsecuentes figuras.

Valores por definición del nodo.

```
Rotation { rotation     0 0 1  0 }  # SFRotation - Valor de rotación de las figuras.
```

Véase el campo de rotación.

Scale

Este nodo define una escala 3D con respecto al origen. Si los componentes del vector de escala no son los mismos, entonces produce una escala no uniforme.

Valores por definición del nodo.

```
Scale { scaleFactor  1 1 1 }  # SFVec3f - factor (vector) de escalamiento
```

Separator

Este nodo de agrupamiento realiza un "push" (salvado) del estado del proceso de recorrido antes de recorrer a sus nodos hijos y restaurarlos (hacer pop) después de recorrerlos. Esto aísla los hijos del separador del resto de la escena gráfica. Un separador puede incluir luces, cámaras, coordenadas, vínculos normales, vínculos y todas las demás propiedades. Los separadores también pueden seleccionar qué se despliega (render culling) en la escena gráfica. La selección de nodos permite ignorar a los hijos del separador si estos no son utilizados, basándose en la comparación de la caja (virtual) vinculada con la vista (en proceso) del volumen. La selección es controlada por el campo renderCulling. El campo anterior es puesto en AUTOmático por definición, permitiendo a la implementación (programa) decidir que seleccionar.

Tipos de selección

ON	Siempre trata de seleccionar el volumen de la vista.
OFF	Nunca trata de seleccionar el volumen de la vista.
AUTO	La implementación define el criterio de selección

Valores por definición del nodo.

```
Separator { renderCulling   AUTO }   # SFEnum - Tipo de selección a usar.
```

ShapeHints

El nodo indica qué caras del campo IndexedFaceSets son sólidas, contienen vértices ordenados, o caras convexas. Estos nodos permiten a los programas optimizar ciertas características del desplegado de los objetos. Lo optimizable incluye habilitar/deshabiltar la selección de nodos y desactivar la iluminación. Por ejemplo, si un objeto es sólido y tiene vértices ordenados, puede deshacer la selección y obscurecerlo. Si el objeto no es sólido pero tiene vértices ordenados, este nodo puede desactivar la selección e iluminar la figura. El nodo también afecta a los vínculos normales. Cuando tiene que generar vínculos normales, el nodo usa el campo crease Angle para determinar cuales ángulos deben ser suavemente sombreados y cuales deben tener afilados pliegues.

El plegado, es el ángulo entre superficie con vínculos normales en polígonos adyacentes. Por ejemplo un incremento de ángulo de 5 radianes significa que el borde entre dos caras poligonales debe ser sombreado finamente si las normales a las dos caras forman un ángulos de menos de 0.5 radianes (cerca de 30 grados). De otra manera estos serán facetados.

Ordenamiento de los vértices

UNKNOWN_ORDERING	El orden de los vértices es desconocido
CLOCKWISE	Los vértices de las caras están ordenadas a favor de las manecillas del reloj (visto desde afuera)
COUNTERCLOCKWISE	Los vértices de las caras están ordenadas en contra de las manecillas del reloj (visto desde afuera)

Tipo de Pliegue

UNKNOWN_SHAPE_TYPE	No se sabe nada acerca del plegado.
SOLID	El filo esta adjunto a un volumen.

Tipo de cara de la figura
UNKNOWN_FACE_TYPE No se sabe nada acerca de las caras de la figura
CONVEX Todas las caras son convexas

Valores por definición del nodo.
ShapeHints { vertexOrdering UNKNOWN_ORDERING # *SFEnum - Orden de los vértices*
shapeType UNKNOWN_SHAPE_TYPE # *SFEnum - Tipo de filo*
faceType CONVEX # *SFEnum - Tipo de las caras -si son convexas o no-*
creaseAngle 0.5 } # *SFFloat - ángulo de pliegue*

Sphere

Este nodo representa una esfera. Por definición, la esfera es centrada en el origen y tienen un radio de 1. La esfera puede ser transformada y ser dibujada con la textura o vínculos normales. Una esfera no tiene caras o partes. La esfera ignora materiales o vínculos normales, simplemente usa la primer textura especificada y tiene sus propios vínculos normales. Cuando una textura es aplicada a una esfera, la textura cubre la superficie entera, envolviendo la esfera en contra sentido de las manecillas del reloj.

Valores por definición del nodo.
Sphere { radius 1 } # *SFFloat - radio de la esfera*

SpotLight

Se usa este nodo para definir un punto de luz. Un punto de luz es colocado en un lugar fijo en el espacio e ilumina similarmente a un cono a lo largo de una dirección en particular. La intensidad de la iluminación diminuye exponencialmente tal como diverge un rayo de luz desde esta dirección hacia los bordes del cono. El grado de disminución y el ángulo del cono son controlados por los campos dropOffRate y cutOffAngle. Este nodo de luz es susceptible de ser modificado por las transformaciones que estén en proceso, además del tipo de luz que use la fuente.

Valores por definición del nodo.
SpotLight { on TRUE # *SFBool / Activación del nodo*
intensity 1 # *SFFloat / Intensidad de la luz*
color 1 1 1 # *SFVec3f / Color de la luz*
location 0 0 1 # *SFVec3f / Localización de la fuente de luz*
direction 0 0 -1 # *SFVec3f / Dirección de la fuente de luz*
dropOffRate 0 # *SFFloat / Grado de disminución*
cutOffAngle 0.785398 } # *SFFloat / Ángulo de apertura del cono de luz*

Switch

Este nodo de agrupación puede recorrer a uno, todos o ninguno de sus nodos hijos. Uno puede usar este nodo para activar o desactivar los efectos de algunas propiedades. El campo WhichChild especifica el valor del índice del hijo a recorrer, donde el valor del primer hijo es 0. Un valor de -1 significa que no recorre ningún nodo. Un valor de - 3 recorre a todos los nodos, haciendo exactamente igual un nodo de agrupación regular.

Valores por definición del nodo.
Switch { whichChild -1 } # *SFLong - Cual nodo recorrer.*

Texture2

Este nodo de textura define un mapa de textura y los parámetros para ese mapa. Es usado para aplicar textura a las subsecuentes figuras. La textura se puede leer desde una localización electrónica o URL (Localizador de fuentes uniforme, o sea, un sitio World Wide Web) especificada por el campo de nombre del archivo. Para desactivar esta textura, ponga el nombre del archivo como cadena vacía ("").

Las texturas pueden ser también especificadas en el mismo archivo poniendo en el campo de la imagen los datos de la textura.

Forma de recubrimiento
REPEAT Repite la textura externa 0-1 Rango de las coordenadas de la textura
CLAMP Fijar las coordenadas de la textura dentro del rango 0-1

Valores por definición del nodo.
Texture2 {filename "" # *SFString - Nombre del archivo. Local o foráneo (WWW)*
image 0 0 0 # *SFImage - Datos de la imagen*
wrapS REPEAT # *SFEnum - Recubrimiento valor S (coordenada horizontal)*
wrapT REPEAT } # *SFEnum - Recubrimiento valor T (coordenada vertical)*

Texture2Transform

El nodo define una transformación bidimensional aplicado a las coordenadas de la textura. Esto afecta la manera en que las texturas son aplicadas a las superficies de las figuras. La transformación consiste de un escalamiento no uniforme sobre un punto central arbitrario, una rotación sobre ese mismo punto, y una traslación. Esto permite al usuario cambiar el tamaño y la posición de las texturas en las figuras.

Valores por definición del nodo.
Texture2Transform { translation 0 0 # *SFVec2f - Vector de traslación*
rotation 0 # *SFFloat - Vector de rotación*
scaleFactor 1 1 # *SFVec2f - Factor de escalamiento*
center 0 0 } # *SFVec2f - Centro de la transformación*

TextureCoordinate2

El nodo define un conjunto de coordenadas bidimensionales para ser usadas en mapas de textura a los vértices de los objetos PointSet, IndexedLineSet, o IndexedFaceSet. Reemplaza las coordenadas de la textura en uso de la figura. El rango de coordenadas de la textura van de 0 a 1 a través de la textura. La coordenada horizontal llamada S, es especificada primero, seguido de la coordenada vertical T.

Valores por definición del nodo.
TextureCoordinate2 { point 0 0 } # *MFVec2f - Coordenadas bidimensionales*

Transform

Este nodo define una transformación geométrica 3D, consistente de un escalamiento no uniforme sobre un punto arbitrario, una rotación sobre un punto y eje arbitrario, y una traslación.

Valores por definición del nodo.

```
Transform { translation          0 0 0      # SFVec3f - Vector de translación
rotation              0 0 1 0    # SFRotation - Vector de rotación
scaleFactor           1 1 1      # SFVec3f - Factor de escalamiento
scaleOrientation      0 0 1 0    # SFRotation - Orientación del escalamiento
center                0 0 0 }    # SFVec3f - Valor del punto arbitrario
```

TransformSeparator

Este nodo de agrupación es similar al nodo separator en que este salva el estado en que están los nodos antes de recorrerlos. Sin embargo, Este salva únicamente la actual transformación; todos los otros estados eliminados. Este nodo puede ser usado para posicionar una cámara, como las transformaciones a la cámara no afectara al resto de la escena, porque la cámara verá la escena. En igual forma, este nodo puede ser usado para aislar las transformaciones de las fuentes de luz a otros objetos.

Valores por definición del nodo.

```
TransformSeparator { }
```

Translation

Este nodo describe una traslación por un vector tridimensional.

Valores por definición del nodo.

```
Translation { translation 0 0 0 }  # SFVec3f - Vector de traslación
```

WWWAnchor

El nodo WWWAnchor carga una nueva escena en un VRML browser[17], cuando uno de sus hijos es escogido. Exactamente el proceso de como el WWWAnchor cambia la escena de realidad virtual, esta reservado al browser; típicamente, el usuario da un clic con el mouse en uno de sus hijos y verá la nueva escena en la pantalla. Un WWWAnchor con un nombre vacío ("") no hace nada cuando sus hijos son escogidos. El nombre al que apunta este nodo, es un URL arbitrario.

El nodo tiende a ser como el nodo Separator, empujando el estado de recorrido antes de recorrer sus hijos y restaurarlo después. El campo de descripción en el nodo permite ser una hiperliga (liga que localiza otra escena VRML) amigable en vez de desplegar el URL.

El campo de mapa del nodo WWWAnchor es un valor que puede ser NONE (el valor por definición) o POINT. Si vale POINT entonces las coordenadas del punto en el

[17] Como su homólogo de HTML, un browser VRML lee mundos virtuales desde servidores WWW que contienen la información localmente o a grandes distancias.

objeto que el usuario escoja serán añadidas al URL en el campo de nombre con la sintáxis " ?x,y,z "[18].

Valores del mapa
NONE No añade información al URL
POINT Añade las coordenadas del objeto 3D al URL.

Valores por definición del nodo.
```
WWWAnchor { name ""          # SFString - Nombre del URL -nueva escena gráfica-
  description ""     # SFString - Descripción de la nueva escena gráfica.
  map    NONE }     # SFEnum - Valores del mapa
```

WWWInline

El nodo WWWInline lee sus hijos desde cualquier lugar en el World Wide Web. Exactamente cuando sus nodos hijos son leídos no está definido; la lectura de sus nodos hijos puede ser retardado hasta que el nodo este desplegado. Un nodo WWWInline con nombre vacío no hace nada. El nombre es un URL arbitrario.

Si el campo bboxSize especifica una caja circundante no vacía (una caja no esta vacía a menos que una de sus dimensiones sea cero), permitiendo a la implementación especificar dicha caja con los campos bboxSize y bboxCenter. Dando la posibilidad a la aplicación de saltar el nodo sin leer su contenido.

Valores por definición del nodo.
```
WWWInline { name ""          # SFString - URL del nodo. el valor "" no activa nada.
    boxSize 0 0 0       # SFVec3f - Tamaño de caja virtual que circunda a todo objeto.
    bboxCenter 0 0 0 } # SFVec3f - Punto central de la caja.
```

Instanciamiento

Un nodo puede ser el hijo de más de un grupo. Esto es llamado instanciamiento o instancing, (usando la misma instancia un nodo múltiples veces, es llamado "aliasing"), y es complementado usando el comando "USE".

El comando DEF define un nodo y crea una instancia simple del comando USE lo que indica que la instancia más reciente debe ser usado otra vez (es decir, se llama a otro nodo de manera similar a un subrutina en C). Si varios nodos tienen el mismo nombre, entonces la ultima DEF encontrada por el párser es usada. Los comandos DEF/USE están limitada a un solo archivo; no hay un mecanismo para el uso de nodos que están definidos en otros archivos.

El nombre de un nodo es usado tan pronto como es leído por el comando DEF por el browser y no es desechado hasta que es encontrado otro nodo con el mismo nombre o un fin de archivo. No puede ser usado un nodo con el comando USE en el nodo WWWInline.

[18] Esto es similar a los mapas de html en la que un imagen 2D sirve de ancla para enviar el browser a otra página, aquí simplemente se sustituye el mapa 2D por una figura 3D.

Extensibilidad
Las extensiones en el VRML son soportados por nodos autodescriptivos. Nodos que no son parte del estándar VRML deben escribir una descripción de sus campos primero, de lo contrario los browsers los ignoraran.

Esta descripción es puesta inmediatamente después de la llave de apertura del nodo, y consiste del comando "fields" seguido por una lista de tipos y nombres de campos usados por el nodo en cuestión, todo esta encerrado entre paréntesis cuadrados y separados por comas. Por ejemplo, si el nodo Cube no fuese parte del estándar se debería escribir de la siguiente forma:

```
Cube { fields [ SFFloat width, SFFloat height, SFFloat depth ]
    width 10 height 4 depth 3
```

Si es especificado campos para nodos que son parte del estándar VRML no produce error, simplemente se ignora el campo.

Relaciones
Un nodo nuevo puede ser un super conjunto de un nodo usado por el estándar. En este caso, si una implementación para el nuevo tipo de nodo no está disponible, el nuevo nodo puede ser tratado como el nodo en que esta basado (con algo de perdida de funcionalidad). Para soportar a un nuevo nodo, puede definir un campo "isA" del tipo MFString, que contiene los nombres de los tipos de los cuales es un super conjunto. Por ejemplo, un nuevo tipo de material llamado "ExtendedMaterial" que añade un índice de refracción como una propiedad del material puede ser escrito como:

```
ExtendedMaterial {  fields [ MFString isA, MFFloat indexOfRefraction,
                    MFColor ambientColor, MFColor diffuseColor,
                    MFColor specularColor, MFColor emissiveColor,
                    MFFloat shininess, MFFloat transparency ]
                     isA [ "Material" ]
  indexOfRefraction .34
                     diffuseColor .8 .54 1
```

Múltiples relaciones pueden ser especificadas en orden de preferencia; Las implementaciones esperan usar el primer nodo para el cual hay una implementación.

1.5.4 Metafile

Los metaarchivos (metafiles) son aquellos formatos que hace uso de distintas codificaciones, compresiones y estructuras, pudiendo usar los elementos combinados de los anteriores tipos. Casi todos son originados de las recomendaciones y propuestas de la ISO/ANSI en el CGM. Se ha permitido que terceros desarrollen formatos especializados para resolver sus necesidades muy particulares. Se han destacado en el desarrollo de formatos, los ejércitos australiano y de los EU, cuyo subtipo el CALS promete ser gran utilidad en el futuro.

1.5.4.1 CGM

Propietario: *American National Standars Institute*
Extensión: *.cgm*

El formato CGM es el primer formato diseñado para ser completamente independiente de software y hardware alguno. Propuesto por la *ANSI* y aceptado por la ISO. El CGM provee apoyo para líneas, círculos, elipses, polígonos, texto y bitmaps. Las propuestas acerca del formato están en la especificación ISO/IEC 8632:1992 que se divide en los siguientes documentos:

- ISO/IEC 8632.1:1992 ---- Parte 1: *Especificación funcional*
- ISO/IEC 8632.2:1992 ---- Parte 2: *Codificación en caracteres*
- ISO/IEC 8632.3:1992 ---- Parte 3: *Codificación binaria*
- ISO/IEC 8632.4:1992 ---- Parte 4: *Codificación en texto claro*

Además de los anteriores documentos existen un par de apéndices aprobados en 1992:

- ISO/IEC 8632:1992/Amd.1:[1994] - Apéndice 1: *Reglas para crear profiles*
- ISO/IEC 8632:1992/Amd.2 - Apéndice 2: *Extensiones de aplicaciones estructuradas*

A partir de tales documentos surge el CGM. El CGM es un metaarchivo (metafile), es decir, es una secuencia de elementos, en forma de comandos definidos en el lenguaje CGM. El metaarchivo puede contener una o varias imágenes, sin relación alguna entre sío en una secuencia animada. El estándar CGM sólo soporta imágenes en 2D. El archivo inicia con un elemento BEGIN METAFILE junto con una cabecera que incluye algunos parámetros explicativos de la imagen, cada imagen inicia con BEGIN PICTURE BODY y se cierra con un END PICTURE, repitiéndose con todas las imágenes usadas. El archivo termina con un comando END METAFILE.

Todas las imágenes están dadas en términos de Coordenadas de Dispositivo Virtual (VCD, en inglés). El VCD es una medida abstracta, y puede ser escalada a pixels o milímetros. Todos los valores que describen a la imagen bitmap, y los comandos que describen imágenes vectoriales están dadas en valores VCD, hay una gran cantidad de comandos propuestos y el revisarlos implica salir del objetivo de la tesina. Por ello a continuación mencionare los elementos restantes.

El formato acepta el uso de texto incluido en el archivo. Las familias de caracteres (Courier, USA Black, etc.) y el conjunto de caracteres (por ejemplo, US ASCII) no están definidos por el formato, por ello son usados muy poco en el CGM.

Los elementos del arreglo CELL ARRAY corresponde a un pixel que representa una imagen bitmap. Cada elemento del arreglo es un valor RGB o una entrada de un mapa de color. Como característica especial, el bitmap está arreglado en forma de paralelogramo, porque se considera que no existe dispositivo alguno que despliegue directamente un bitmap no rectangular.

El CGM ofrece dos tipos de implementaciones que permiten especificar extensiones. La primitiva de dibujo generalizada (GDP, en inglés), contiene un número de identificación, seguido de una lista de datos que describen a la imagen, y un bloque de datos. La otra implementación es la primitiva ESCAPE que tiene un número de identificación y un bloque de datos, todos ellos relacionados al dispositivo usado. La desventaja de estas implementaciones es que son recomendaciones, así, no hay un criterio que guíe a los creadores de aplicaciones que usen tales extensiones, presentándose obvias incompatibilidades.

Tipos de codificación

El CGM acepta tres tipos de codificaciones para el archivo CGM. La codificación en caracter es muy compacta y se usa para intercambio. La codificación binaria es la más comúnmente usada y pueden hacerse programas que manejen el formato fácilmente. El texto claro es muy usado en correcciones.

Codificación en caracteres

La codificación en caracteres es compatible con el estándar ISO 2022, la versión internacional que define la secuencias escape ANSI. Los datos son codificados en códigos escape, seguidos de caracteres que son comunes a todos los equipos y sofware existente (caracteres ASCII imprimibles). Hay dos variantes en esta codificación, ello se debe al manejo del octavo bit en cada equipo.

Un metaarchivo es iniciado por ESC % F y seguido de un ESC % @. Cada operador tiene asignado un código de uno o dos bytes, por ejemplo, POLYGON es 26h. Los números son codificados en hexadecimal, en el caso de los números reales se separa la fracción y el exponente. En elementos que usan listas de puntos, tal como en POLYGON, hay un modo opcional de incrementos que es bastante complejo que codifica puntos secuenciales usando una tabla Huffman de movimientos relativos.

Los datos CELL ARRAY (o bitmap) son codificadas las tripletas RGB en paquetes RLE de 5 o 6 bits, el bloque se inicia con un ESC X y termina con ESC \ , en caso de paquetes de 8 bits, los bloques inician con 98 y terminan con 9C.

Codificación Binaria

La codificación binaria es muy popular, debido a que es fácil de generar o manipular, produciendo un archivo razonablemente compacto. Un metaarchivo en binario es una secuencia de bytes con tamaño de palabra de 16 bits. Cada elemento ocupa una palabra, en formato MSB, y seguido por un conjunto de parámetros. La palabra se divide así, siendo cada letra un bit:

CCCC IIIIIII LLLL

CCCC identifica la clase del elemento, IIIIIII el identificador del elemento, por ejemplo POLYGON es clase 4, elemento 7. LLLL es el número de bytes del parámetro. Si hay más de 30 bytes de datos, LLLL son unos e indican que la siguiente palabra tiene la longitud del la lista de parámetros. Si el bit superior de la palabra tiene valor, otra palabra tiene más parámetros del elemento. cuando los parámetros son impares, se añade bytes hasta que hacer par el número de parámetros.

Los enteros son codificados típicamente en dos bytes, pero si las necesidades así lo requieren, hay enteros de 1, 2, 3, 4 bytes de longitud. Los reales pueden ser de 32 bits punto fijo (16 entero, 16 fracción), 64 bits punto fijo (32 entero, 32 fracción), 32 bits punto flotante o 64 bits doble según el estándar IEEE. El tipo de formato depende del valor VCD usado.

Los datos de CELL ARRAY alcanzan de uno hasta 32 bits de profundidad, y pueden estar en una lista simple o codificado en RLE. Las cadenas de datos son precedidas por un byte, si son igual o mayores a 255 bytes, son precedidas de un byte FFh.

Codificación en texto claro

El texto claro esta diseñado para ser escrito y leído por usuarios usando un simple editor de texto. Los elementos son identificados por sus nombres. Los enteros son escritos en decimal o con una base opcional. Los reales son escritos en decimal o forma exponencial. Los puntos son pares de números separados por comas, opcionalmente encerrados entre paréntesis.

Los elementos se delimitan con punto y coma. Las listas contienen valores absolutos o relativos, por ejemplo: POLYGON (10,10)(100,100)(90,-100). Los CELL ARRRAYS son listas pixels, con valores que opcionalmente son delimitados por paréntesis. Las cadenas son delimitadas por comillas simples o dobles. Los comentarios se encierran entre signos de porcentaje.

Apéndices

El formato fué definido por primera vez en 1986, la definición básica de los componentes tiene varios inconvenientes, primero, los elementos estaban limitados, y por ser recomendaciones la implementación del CGM ha sido difícil por no haber acuerdo para usar un conjunto de elementos y una codificación estable. Por ello se hacen en 1992 ajustes al formato. Se añaden 30 nuevos elementos, el desarrollo de librerías, mejor control de imágenes y son adjuntados dos apéndices que permiten el desarrollo de aplicaciones y formatos especializados. Cabe mencionar que al ser liberada la mejora de 1992, se invalidan completamente los documentos de1987.

Apéndice 1- Reglas para crear profiles

Este apéndice define las reglas para escribir "profiles", es decir, formatos que haciendo uso de partes del CGM y otras documentaciones realizan funciones óptimas en diseño, documentación, bases de datos, multimedia e hypermedia entre otras. El apéndice anota:

- Reglas para crear Profiles.
- Definición de (programas) generadores, intérpretes, y metaarchivos, en términos del profile.
- Un modelo de profile que sirve de guía para la creación de los mismos, conocido como PPF (Profile proforma). El apéndice 1 exige que todos los Profiles lo incluyan.

Apéndice 2: Extensiones de aplicaciones estructuradas

Define los requerimientos funcionales para el acceso a pequeñas secciones del metaarchivo o imagen. El uso de estas "imágenes inteligentes" incluyen manejo de

documentos electrónicos, bases de datos gráficas, aplicaciones gráficas en redes distribuidas, aplicaciones en multimedia e hypermedia.

Hay varias aplicaciones de los profiles. Las aplicaciones de Profile (AP) definen los elementos opciones y parámetros de la norma *ISO8632* (CGM) necesarios para realizar una determinada función y maximizar el intercambio entre quienes implementan el profile. Además de usar un subconjunto de elementos del CGM necesitan definir programas de codificación- decodificación. Algunos de los profiles más interesantes son:

Profile *Model*
El model profile es un formato que usa los tres tipos de codificación y es apropiado para gráficas científicas y técnicas (por ejemplo., Diseño asistido por computadora, cartografía, ciencias de la tierra, etc.), presentación y visualización de aplicaciones publicitarias.

Profile *ATA*
Este profile esta acorde con la especificación *ATA 2100*. Creado para el desarrollo de documentación técnica para la manufacturación y la operación de aviones comerciales. El profile soporta codificación binaria y en texto claro, y hace uso de las librerías. Es apropiado para intercambio de manuales, visualización y publicidad. El desarrollador y principal usuario es la ATA/AIA (Asociación de transportadores de líneas áreas/Asociación de la industria de líneas áreas).

Profile *CALS*
Desarrollado por el Departamento de Defensa de los E.U. en base a su especificación *MIL-D-28003A*. Soporta la codificación binaria, no sigue con la norma del apéndice 1 por haber sido desarrollado antes de la liberación del apéndice. El profile es usado para ilustraciones técnicas y publicaciones. Existe la propuesta para crear una versión que haga animaciones.

Profile *PIP*
Desarrollado para aplicaciones de información en la comunidad petrolera. El creador es Petrotechnical Open Software Corporation.

El CGM es usado como base por otras organizaciones para promover normas, un ejemplo de ello es El FIPS 128-1 (FIPS CGM) del gobierno de los E.U. El FIPS hace uso de los documentos básicos del CGM a la vez que se complementa con la especificación militar *MIL-D-28003A* y tiene como fines:

- Promover el intercambio de información gráfica.
- Reducir los excesos en el ciclo de vida de productos mediante el establecimiento de formatos para el almacenamiento y transferencia de información gráfica a través de organizaciones independientes de cualquier sistema.
- Promover el desarrollo y uso de aplicaciones de los profiles y coordinar los esfuerzos de tal desarrollo.

FORMATOS DE SONIDO

2. Sonido

El sonido añade una nueva dimensión a las aplicaciones computacionales, haciéndolas más atractivas al usuario común; una aplicación con efectos de sonido como puede ser una enciclopedia multimedia, tiene más impacto que el uso simple de texto e imágenes. La difusión de las aplicaciones que crean, almacenan y manipulan sonido ha sido lenta hasta hace poco, ello se debe principalmente al alto costo del hardware y que los dispositivos de sonido son todavía considerados accesorios de lujo. Esta situación está modificándose rápidamente con el abaratamiento de costos y la introducción de la multimedia.

En el presente capítulo mencionaré como se define que es sonido, y el caso especial de la música; veremos como son definidos, codifcados y cuáles formatos son los más usados.

Los técnicas usadas para codificar y almacenar formatos de sonido son similares a las usadas en formatos gráficos, algunos algoritmos de compresión son adaptaciones de los empleados en las imágenes.

Los formatos de sonido requieren conocimientos adicionales a los mencionados en los formatos gráficos (estructuras de datos, programación, probabilidad y estadística, etc.), sino además, es menester contar con algunos conocimientos de física, y especialmente, de música.

2.1 Nociones básicas del sonido

Cuando algo vibra en el aire crea ondas de presión. Tal movimiento es percibido por los tímpanos humanos como sonido. Las ondas de sonido varían en volumen (medido en decibeles, db), y en frecuencia (o tono, que son vibraciones por segundo, medidos en Hertz o Hz). Cabe recordar que la escala de decibeles es una escala logarítmica, es decir, al duplicar la potencia de un sonido sólo se aumentan 6 db en su volumen. El sonido al ser energía, es medido en vibraciones por segundo, estas vibraciones tienen una amplitud y frecuencia. De ello es posible definir sus características y ser representadas matemáticamente; por lo tanto, es posible grabar, almacenar, modificar y reproducir un sonido de una manera digital.

La grabación digital es factible de ser usada por equipo electrónico como son las computadoras. El proceso de conversión de un sonido analógico a uno digital, es explicado a continuación.

2.1.1 Sonido Analógico/Digital [19]

El sonido analógico se compone de una serie de ondas, las cuales son convertidas por un dispositivo Convertidor Audio/Digital (A/C Converter) en datos conformados por cadenas de bits.

[19] Rossum van, Guido.Sound File Formats, F.A.Q., part 1, (16)

La secuencia sonora se divide en segmentos, a su vez, de cada segmento se obtiene una muestra (sample) del sonido. La muestra se compone de una serie de bits, en donde cada bit guarda una parte de la amplitud de la frecuencia del sonido en el segmento. Lo que se está haciendo es hacer que una función continua como lo es el sonido, sea almacenada en forma discreta; debido a ello, entre más bits contenga una muestra, mejor será mejor la aproximación con respecto de la onda original, dando una mejor calidad de reproducción del sonido. El número de bits usados en una muestra es amplia, pero los valores más usados son 8 y 16 bits.

El número de muestras que son reproducidas en un segundo, se mide en Hertz. Los valores de reproducción más usados son presentados en la tabla 2.1.

Tabla 2.1: Velocidad de Reproducción

Muestras por segundo en kHz	Descripción
8	Estándar Telefónico, en cercana relación con los estándares μ-Law y A-Law.
11	Es la mitad de la velocidad típica de los sistemas Macintosh, muy popular.
18.9	Estándar CD-ROM/XA base.
22	Velocidad típica de los sistemas Macintosh, el valor preciso es 22254.545454545454.
32	Es usado en radio digital, en sistemas NICAM y HDTV de televisión.
37.8	Estándar CDROM/XA para alta calidad.
44056	Usado en audio profesional de video.
44100	Velocidad del CD de audio común.
48000	Velocidad para el DAT (Cinta de audio digital), aunque es común que se use el valor de 44.1 Mhz.

Una frecuencia analógica maneja valores fraccionarios, estos valores son redondeados (cuantificados) para ser almacenados, por ello es común los programas de reproducción hagan uso de filtros que eliminen los "ruidos", como se denomina a las aberraciones sonoras generadas a los valores obtenidos en el redondeo. Normalmente los formatos dividen el segmento sonoro en cuadros (o secciones) de 1 segundo, almacenando las muestras de ese segundo por separado.

Los bits que hay en un cuadro pueden a su vez agruparse en "canales", cada canal almacena una sección de sonido que es desplegada de manera individual; los canales existentes en un cuadro son reproducidos al mismo tiempo.

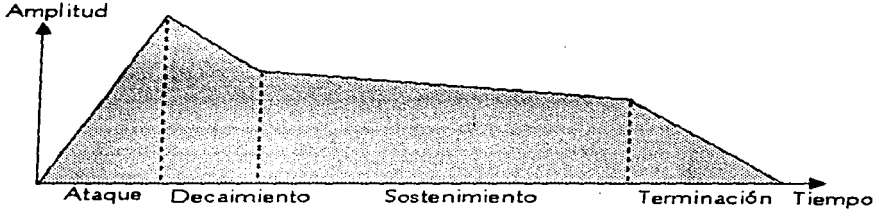
El número de canales en un cuadro determina la calidad del mismo, es decir, 1 canal el sonido es monoaural, 2 es estéreo, etc. (véase la figura). Cada canal de sonido usa una cadena propia, por ejemplo, en un cuadro de un segundo a 8 kHz estéreo, cuatro son para un canal y cuatro para el otro, es de práctica común almacenar los canales órdenandolos a partir del canal izquierdo, luego el derecho, el sourrund, etc.

	Canales 1	2	3	4	5	6
ESTÉREO	Izquierdo	Derecho				
3 CANALES	Izquierdo	Derecho	Central			
CUADRAFÓNICO	Izquierdo Frontal	Derecho Frontal	Izquierdo Anterior	Derecho Anterior		
4 CANALES	Izquierdo	Central	Derecho	Surround (ambiental)		
6 CANALES	Izquierdo	Izquierdo Central	Central	Derecho	Derecho Central	Surround (ambiental)

La distribución de los canales ilustrados en la imagen, se aplica a dispositivos estereofónicos. Si un dispositivo cuenta con menos canales de reproducción que los usados en un sonido, la reproducción se ajusta a las necesidades del dispositivo. El canal *surround* es usado para reproducir el sonido ambiental de fondo.

2.1.2 Música

Cuando el sonido a almacenar es música que se obtiene al tocar uno o varios instrumentos musicales (flautas, guitarras, mandolinas, trompetas, etc.), hay un método alternativo que consiste en guardar la información de la(s) "nota(s)" interpretada(s) en un instrumento musical. El estándar MIDI (Musical Instrument Digital Interface, Interface digital de instrumentos musicales) - desarrollado en 1984 -, proporciona un protocolo de transferencia entre instrumentos musicales y computadoras. Cada nota se compone de cuatro partes: ataque, decaimiento, sostenimiento y terminación. El ataque es el momento en que es iniciada la nota, el decaimiento es cuando es estabilizada, el sostenimiento indica la duración de la nota estable y la terminación se refiere al momento en que esta nota es terminada.



Al ser tocada una nota en un instrumento aprobado por la MMA (MIDI Manufacturers Association, Asociación de fabricantes de MIDI) como puede ser un sintetizador, mediante el convertidor A/D es transformado en datos, que a su vez se guardan en el archivo las características ya mencionadas de la nota, además del

instrumento del cual proviene el sonido de la nota, los efectos sonoros que modifican la nota, el ritmo, entre otras cosas.

Estos valores han sido previamente organizados por la MMA en tablas estandarizadas, que todos los instrumentos y dispositivos MIDI tienen integrados. Para reproducir el tema musical sólo basta con leer los datos del archivo y obtener de las tablas la información para reproducir la nota. Gracias a esa estandarización los archivos MIDI son de tamaño sensiblemente menor a los archivos de sonido. De la misma forma que ocurre en el sonido, las notas se dividen en secciones de un segundo y son agrupadas en canales, los cuales son reproducidos al mismo tiempo.

Una desventaja que aqueja la aceptación del estándar MIDI (y de formatos de sonido, pero en menor medida) es el diferente rendimiento del hardware básico con que cuentan los equipos de cómputo (véase la tabla 2.2), algunos modelos pueden contar con mejor hardware para reproducir música, lo cual da resultados variables y no permite asegurar un calidad absoluta y uniforme que se consigue con los formatos gráficos. Otra situación es que el estándar MIDI sólo contempla una gama pequeña de los instrumentos musicales existentes, algunos fabricantes como Roland han incluido grupos de instrumentos adicionales, pero si es usado un instrumento no contemplado en el estándar, al ser reproducido en un equipo que no disponga de la información del instrumento, sustituirá tal instrumento con alguno del estándar con detrimento de la armonía instrumental del tema musical. El MIDI sólo soporta instrumentos musicales, así las voces y ruidos que no pertenezcan a un instrumento musical, no pueden ser usados en la creación de una melodía. Más adelante se explica con detalle el estándar.

Tabla 2.2: Calidad del Hardware disponible

Sistema	Número de bits Velocidad de reproducción por seg	Número de canales disponible
Macintosh	8/22 kHz 16/64 kHz	1 / 4
Amiga	8/29 kHz	4 (cuatro voces en estéreo)
Sun	8,16/48 Mhz	1 (mono y estéreo)
NeXT	8,16/44.1 Mhz	1 (estéreo)
SGI	8,16/48 Mhz	4, 16 (estéreo)
Sony NWS	8,16/37.8 Mhz, 48 Mhz	1 (estéreo)
DEC	8/8 kHz	1
HP9000	8,16/48 Mhz	1 (estéreo)
Terminal NCD MCX	8,16/52 kHz	1 (estéreo)

Como se puede apreciar en la tabla 2.2, la calidad de los sistemas actuales, en lo que al sonido se refiere, es irregular, de hecho los valores aquí enumerados no son apoyados por todos lo modelos de los respectivos sistemas.

En el caso particular de las computadoras de la familia de las IBM PC compatibles, los dispositivos de almacenamiento y reproducción de sonido son accesorios opcionales, distribuidos por terceros. Para reproducir sonido digital y música MIDI en una PC, es necesario adicionar al hardware básico una tarjeta reproductora especializada. Existen varios fabricantes de tarjetas de sonido con renombre como Roland, AD-LIB, Creative Labs (productora de las tarjetas SoundBlaster), etc., que en general dan un rendimiento máximo de 44.1 Mhz, ya sea en monoaural o estéreo,

y a 8 ó 16 bits por muestra. También es posible conseguir una tarjeta que reproduzca y además grabe, sin embargo, ello no implica que las capacidades de velocidad y calidad de sonido sean las mismas para ambos modos.

Este tipo de tarjetas requieren actualmente que la computadora sea un procesador 80386, y posea al menos 8 Mb de memoria RAM con 50 Mb de espacio en disco duro. Estas características mínimas son necesarias para escuchar el sonido de modo continuo, sin sufrir molestos "cortes" o periodos de silencio, además de tener en la memoria el programa reproductor, el sonido a reproducir, y los controladores correspondientes de la tarjeta, hay que considerar que es rara la vez, en que sólo dedicamos todo el proceso de una computadora a escuchar una melodía o sonido, sino que está funcionando en conjunción con otra aplicación como puede ser un reproductor de video digital, un video juego, un programa multimedia, etc., a lo cual hay que añadir los requerimientos del programa huésped. El tamaño de típico de un archivo MIDI está en el orden de los 100 Kb, mientras que un archivo de sonido con la misma duración de tiempo puede llegar a valer 3 Mb

2.2 Métodos de Compresión

El sonido como cualquier tipo de dato es susceptible de ser comprimido, pero se ha encontrado que es difícil de lograr un nivel de compresión aceptable. El método de compresión Huffman (véase formatos gráficos) es el más popular en las cadenas (muestras) de 8 bits. Para valores superiores es común que se use métodos de compresión propietarios, como el PASC de Philips y el ACE/MACE de Macintosh.

2.2.1 μ-Law [20]

Una opción a esto es el uso de estándares públicos para compresión de voz. Los estándares CCITT G.721, G.723 y G.711 (que define los formatos A-Law y μ-Law) usan como compresión el ADPCM (Código de Modulación de pulso adaptativo). En Europa se maneja el estándar GSM 06.10.

El μ-Law es un desarrollo del CCITT , este método está especialmente diseñado para comprimir sonido; especificado en el documento CCITT G.723, este esquema de almacenamiento generalmente reduce a la mitad de su tamaño original un archivo de sonido. Es ampliamente usado en los formatos que codifican el sonido digital.

2.3 Aplicaciones de los formatos de sonido

Entre las aplicaciones más atractivas en los formatos de sonido, están las aplicaciones multimedia, ya que las presentaciones con sonido son más atractivas, ambientan las ideas expresadas y se pueden explicar mejor que simplemente un texto estático en la pantalla. La telefonía digital por su parte está extendiéndose a

[20] Rossum van, Guido.Sound File Formats, F.A.Q., part 2, (17)

todo el globo terráqueo sin importar país o cultura, con mejor una mayor facilidad y claridad de trasmisión que al emplear el sonido analógico.

El World Wide Web será beneficiado grandemente, ya que se están transmitiendo archivos de sonido para conversaciones a tiempo real (Voice chat), en foros internacionales interactivos; algunas estaciones radiofónicas ya están transmitiendo su programación en el Web (real audio) de manera cotidiana y estable. En las ciencias el sonido está siendo usado para analizar el rendimiento de equipos industriales y comerciales. En el campo de la salud, son usados programas activados por voz, para permitir a los discapacitados trabajar y desarrollarse en su entorno con una facilidad que no era posible previamente.

2.4 Formatos de sonido

Los formatos de sonido se pueden clasificar en dos grupos: Autodescriptivos y Raw. Los primeros se caracterizan por tener una cabecera bien definida y compleja, mientras que en los formatos de tipo *Raw*, su cabecera es muy simple, o bien, no disponen de ninguna. Los formatos autodescriptivos gozan de varios tipos posibles de esquemas de compresión, así como de escoger muchas velocidades de reproducción, se les puede encontrar en distintas arquitecturas. Los formatos raw sólo soportan un tipo de compresión y una velocidad de reproducción, están fuertemente ligados a su sistema nativo, y además son difíciles de convertir, ya que tienen que desechar la información que no puedan manejar, a veces incluyendo aspectos del sonido que influyen en la calidad del mismo. Los Formatos Autodescriptivos más sobresalientes se describen a continuación.

2.4.1 MIDI

Propietario: ***MIDI Manufacturers Association***
Extensión: ***.mid***

En 1984 es organizado bajo los auspicios de Apple, Commodore, entre otros; es un estándar para almacenar la música obtenida de instrumentos musicales, el estándar se denomina MIDI (Musical Instrument Digital Interface, Interface Digital de Instrumentos Musicales). Actualmente la MMA (*MIDI Manufacturers Association*, Asociación de Fabricantes de MIDI), es la propietaria y encargada de controlar las revisiones y actualizaciones al formato.

Estructura del estándar y formato
El estándar define una serie de especificaciones para los dispositivos de distintos fabricantes -de instrumentos musicales, tarjetas de sonido, computadoras, etc.- para lograr la generación y lectura de señales que permitan la codificación de una secuencia musical. Para codificar la secuencia musical, ésta es dividida en el número de canales que son tocados a la vez en un momento determinado.

Cada canal contiene un instrumento, el número máximo de instrumentos que pueden ser tocados simultáneamente es 16, es posible almacenar un máximo de 64 instrumentos diferentes en un archivo MIDI. Cada instrumento tiene definido un

número de notas que usará, el instrumento toca sólo una nota en un momento dado, o sea, hasta 16 notas de otros tantos instrumentos son escuchados a la vez (aunque se permite que un instrumento toque en más de un canal a la vez).

El estándar define los valores de 127 instrumentos, adicionalmente 46 instrumentos de percusión (para activarlos se usa un valor adicional), que son enlistados en la tabla 2.3, los valores de cada nota sin importar a que instrumento este ligado,se definen en la tabla 2.4; las notas estan agrupadas por octavas (es decir, en secuencia DO = C, RE = C# , ...). La octava -1 es la más baja (grave) en la escala; adicionalmente son almacenados efectos de sonido en la secuencia de cada nota. En los efectos que afectan a una nota se encuentran desde la variación del tiempo en la nota, hasta el uso de dispositivos como pedales y botones de ajuste.

Tabla 2.3: Lista de Instrumentos normales

INSTRUMENTOS MIDI	
0	Piano De Cola Acústica
1	Piano Acustico Vertical
2	Piano Electrico De Cola
3	Piano De Taberna
4	Piano De Rhodes
5	Piano De Coro
6	Clavicordio
7	Clarinete
8	Celesta
9	Glockenspeil, Glokenspiel, Carrillon U Organo De Campanas
10	Caja Musical
11	Vibrafono O Vibrarpa
12	Marimba
13	Xilofono
14	Campanas Tubulares
15	Dulcimer O Dlucemel
16	Órgano Hammond
17	Órgano De Percusion
18	Órgano De Rock
19	Órgano De Iglesia
20	Órgano De Reed
21	Acordeon
22	Armonica
23	Bandoleon (Órgano De Tango)
24	Guitarra Acustica (Nylon)
25	Guitarra Eléctrica (Metalicas)
26	Guitarra Eléctrica (Jazz)
27	Guitarra Eléctrica (Claro)
28	Guitarra Eléctrica (Sonido Apagado)
29	Guitarra Overdriven Tocada Aceleradamente
30	Guitarra Con Distorsión
31	Guitarra Armonica
32	Bajo Acústico
33	Bajo Eléctrico (Punteada)
34	Bajo Eléctrico (Con Plectro)
35	Bajo Sin Ceja
36	Slap Bass 1 (Bajo Tipo Jazz 1)
37	Slap Bass 2 (Bajo Tipo Jazz 2)
38	Synth Bass 1 (Bajo Sintetizado 1)
39	Synth Bass 2 (Bajo Sintetizado 2)
40	Violín
41	Viola
42	Violonchelo
43	Contrabajo
44	Cuerdas Tremolas
45	Cuerdas Punteadas
46	Arpa De Orquesta
47	Timbales Apagados
48	Conjunto De Cuerdas 1
49	Conjunto De Cuerdas 2
50	Cuerdas Sintetizadas 1
51	Cuerdas Sintetizadas 2
52	Coros De A's
53	Voces De O's
54	Voz Sintetizada
55	Entrada De Orquesta
56	Trompeta
57	Trombon
58	Tuba
59	Sordina De Trompeta
60	Corno Francés
61	Sección De Metales
62	Metal Sintetizado 1
63	Metal Sintetizado 2
64	Saxofon Soprano
65	Saxofon Alto
66	Saxofon Tenor
67	Saxofon Baritono

68	Oboe
60	Corno Inglés
70	Fagot
71	Clarinete
72	Picolo O Flautin
73	Flauta
74	Flauta De Pico
75	Flauta De Pan
76	Bottle Blow (Soplo De Botellas)
77	Shakulachi (Flauta Vertical Japonesa)
78	Silbato
79	Ocarina (Instrumento Musical En Forma De Huevo, Pájaro O Balata, Conocido Como Flauta Globular)
80	Lead 1 (Square)
81	Lead 2 (Sawtooth)
82	Lead 3(Caliope, Organo De Vapor)
83	Lead 4 (Chiff-Lead)
84	Lead 5 (Charanga)
85	Lead 6 (Voz)
86	Lead 7 (Fifths)
87	Lead 8 (Bajo+Lead)
88	Pad 1 (New Age)
89	Pad 2 (Warm)
90	Pad 3 (Polysynth)
91	Pad 4 (Coro)
92	Pad 5 (Bowed)
93	Pad 6 (Metallic)
94	Pad 7 (Halo)
95	Pad 8 (Sweep, Escobilla)
96	Efectos Especiales 1 (Lluvia)
97	Efectos Especiales 2 (Banda Sonora)

CLAVES DE PERCUSION	
35	Tambor Acústico Bajo
36	Tambor Bajo 1
37	Baquetas
38	Timbre Acústico de Tambor
39	Palmada
40	Tambor Electrico
41	Low-Floor (Tambor De Piso Bajo)
42	Closed High-Hat
43	High-Floor Tom
44	Pedal High-Hat
45	Low Tom
46	Open High-Hat
47	Low-Mid Tom
48	High-Mid Tom
49	Choque De Cimbalos 1

98	Efectos Especiales 3 (Cristal)
99	Efectos Especiales 4 (Atmosfera)
100	Efectos Especiales 5 (Esplendor)
101	Efectos Especiales 6 (Duendes)
102	Efectos Especiales 7 (Ecos)
103	Efectos Especiales 8(Ciencia-Ficción)
104	Citara
105	Banjo
106	Shamisen (Banjo Japonés)
107	Koto (Citara Japonesa)
108	Kalimba
109	Gaita
110	Violin
111	Shanai
112	Retintin De Campana
113	A Go Go
114	Tambores Metalicos
115	Caja Chima
116	Tambor Taiku
117	Tambor Melodico
118	Tambor Sintetizado
119	Cimbalos
120	Ruido Del Traste De La Guitarra
121	Ruido De Espiracion
122	Orilla Del Mar
123	Trinos De Un Pajaro
124	Timbre De Telefono
125	Helicóptero
126	Aplauso
127	Balazo

50	High Tom
51	Ride Cymbal 1
52	Cimbalos Chinos
53	Timbre De Bicicleta
54	Randereta
55	Ruido De Cimbalos
56	Cencerro
57	Choque De Cimbalos 2
58	Vibraslap
59	Ride Cymbal 2
60	Bongo Alto
61	Bongo Bajo
62	Conga Mute High (Conga Apagada Alta)
63	Conga Mute Open (Conga Apagada Alta)
64	Conga Baja
65	Timbal Alto

66	Timbal Bajo
67	A Go Go Alto
68	A Go Go Bajo
69	Cabasa
70	Maracas
71	Silbato Corto
72	Silbato Largo
73	Güiro Largo
74	Güiro Corto
75	Claves

76	Caja China Alta
77	Caja China Baja
78	Cuica o Zambomba Apagada
79	Cuica o Zambomba Sonora
80	Triángulo Pagado

Tabla 2.4: Lista de mensajes

Etiqueta del mensaje	Byte(s) de datos (en binario)	Descripción
	Mensajes de Canal de voz (Channel Voice Messages)	
1000cccc	0nnnnnnn 0vvvvvvv	*Evento de desactivación de la nota.* Este mensaje es enviado cuando una nota es soltada (terminada). El valor (nnnnnnn) es el número de la nota y el valor (vvvvvvv) es la velocidad.
1001cccc	0nnnnnnn 0vvvvvvv	*Evento de activación de la nota.* Este mensaje es enviado cuando una nota es oprimida (iniciada). El valor (nnnnnnn) es el número de la nota y el valor (vvvvvvv) es la velocidad.
1010cccc	0nnnnnnn 0vvvvvvv	*Presión de la tecla polifónica (durante la presión de la nota).* Este mensaje es enviado cuando la presión (velocidad) de una nota previamente en función cambia. El valor (nnnnnnn) es el número de la nota y el valor (vvvvvvv) es la nueva velocidad.
1011cccc	0ccccccc 0vvvvvvv	*Cambio de control.* El mensaje es enviado cuando el valor de un controlador cambia. Los controladores pueden ser pedales y sintetizadores de ajuste. Ciertos números de controladores son reservados para propósitos específicos - véase los mensajes de modo de canal-. El valor (ccccccc) es el número del controlador y (vvvvvvv) es el nuevo valor.
1100cccc	0ppppppp	*Cambio de programa.* El mensaje ocurre cuando el número parche (programa) cambia.
1101nnnn	0ccccccc	*Canal de presión.* Éste mensaje es enviado cuando el canal de presión cambia. La velocidad de respuesta de algunos teclados no soportan la tecla polifónica. Use este mensaje para enviar la velocidad más rápida (de todas las teclas presionadas en ese momento). (ccccccc) es el número de canal.
1110nnnn	0lllllll 0mmmmmmm	*Cambio en el giro de tono.* Este mensaje es enviado para indicar un cambio en el tono. El giro de tono es medido por el valor del bit 14.(lllllll) son los 7 bits menos significativos y (mmmmmmm) son los 7 bits más significativos.

Mensajes de modo de canal (Channel Mode Messages)		
1011nnnn	0ccccccc 0vvvvvvv	*Mensajes de modo de canal.* Usa la misma codificación como el mensaje de cambio de control, pero implementa el modo de control usando números de control reservados. Los números son: Control Local: Cuando el control local está desactivado, todos los dispositivos en un canal dado responderán únicamente a los datos recibidos acerca del MIDI. todos los demás datos son ignorados (datos tocados, pedales, etc.). El control local activo (on) restaura las funciones de los controladores normales. c = 122, v = 0: Control Local apagado (Off). c = 122, v = 127: Control Local encendido (On). **Todas las notas son canceladas.** Cuando el mensaje es enviado todos los osciladores serán apagados. c = 123, v = 0: Todas las notas son canceladas. c = 124, v = 0: Modo Omni apagado (Off). c = 125, v = 0: Modo Omni encendido (On). c = 126, v = M: Modo Mono Activo (On), donde M es el número de canales desactivados (solo un canal está activo). c = 127, v = 0: Modo Poly Activo (On), todos los canales están activos.
Mensajes Comunes del sistema		
11110000	0iiiiiii 0ddddddd 0ddddddd 11110111	*Sistema Exclusivo.* Diseñado para incluir mensajes que el estándar MIDI no soporta. (iiiiiii) es el código de identificación del fabricante de 7 bits. Si el sintetizador (o programa) reconoce el código de identificación como propio, leerá el resto del mensaje (ddddddd). De otra manera este será ignorado. para mensajes intercalados a tiempo real es necesario usar mensajes propietarios.
11110001		*Indefinido.*
11110010	0lllllll 0mmmmmmm	*Puntero de posición del tema musical.* Es un registro interno de 14 bits que contiene el número de pulsos MIDI (1 pulso = 6 ciclos de reloj MIDI) desde el inicio del tema., está en LSB, m está en MSB.
11110011	0sssssss	*Selección del tema.* La selección del tema específica cual secuencia musical o canción está siendo tocada.
11110100		*Indefinido.*
11110101		*Indefinido.*
11110110		*Petición de tono.* Cuando es recibida una petición de tono, todos los sintetizadores analógicos deben ajustar el tono en sus osciladores.
11110111		*Fin del sistema exclusivo.* indica el fin del bloque de datos exclusivos.
Mensajes de Tiempo real		
11111000		*Reloj de Compás.* Envía 24 señales por cuarto de nota cuando la sincronización es requerida.
11111001		*Indefinido.*

11111010		*Inicio.* Inicia la secuencia de tocado (el mensaje es acompañado con señales del reloj de compás).
11111011		*Continuación.* Continua en el punto en que la secuencia fue detenida.
11111100		*Paro.* Detiene la secuencia.
11111101		*Indefinido.*
11111110		*Detección de Actividad.* El uso de este mensaje fue opcional. Cuando es enviado inicialmente, el receptor esperara recibir otro mensaje de detección de actividad cada 300ms como máximo, ose asume que la conexión ha terminado. Una vez terminado, el receptor desactivara todos las voces y retornara a operación normal (no sensitivo).
11111111		*Reinicio.* Reinicia a todos los receptores en el sistema. Debe se usado esporádicamente, de preferencia bajo control manual. En particular, no debe se enviado en la activación.

El estándar tiene documentado los valores de los mensajes (señales) entre los dispositivos, tales señales como se ha explicado anteriormente definen las notas, su orden de aparición, los efectos usados, etc. Pero además tiene especificado valores de sincronización que son en multimedia y video.

Los controles de efectos pueden repetir un tema varias veces, variar el tono de una melodía, el uso de efectos de sonido, también hay muchos campos que están vacíos para permitir la ampliación del estándar.

La ultima revisión del estándar fue hecha a mediados de 1995. La MMA también define un formato de extensión .MID, para conseguir más información es necesario contactar a la organización, la cual vende copias del estándar completo en $49 U.S. dólares más cargo de envío (septiembre de 1996). Dentro de lo poco disponible acerca del formato distribuido por la MMA, está distribuido en forma de tablas, y la información es acerca de los valores de las notas y las etiquetas de aviso del formato, y la estructura del formato. (Véase el apéndice II).

2.4.2 NeXT/SUN

Propietarios: ***SUN Co. / NeXT Co.***
Extensión: *.snd*

Este formato tiene implementaciones tanto para computadoras NeXT como SUN, con sistema operativo Unix; explicaré la implementación en la paltaforma NeXT. El formato es una estructura del sistema operativo conocida como "SNDSoundStruct". Esta estructura está encapsulada por el kit de sonido del Sistema Operativo NeXT y por medio de éste es posible modificar los parámetros del archivo. El sistema Operativo Sun OS cuenta también con dicha estructura.

Estructura

El formato consta de una estructura en C que sirve de cabecera, los datos son referenciados por la cabecera ya que pueden estar en ese archivo o en otro distinto. La cabecera se describe a continuación y la explicación de los datos en la tabla 2.5.

```
typedef struct {
        int magic;
        int dataLocation;
        int dataSize;
        int dataFormat;
        int samplingRate;
        int channelCount;
        char info[4];
} SNDSoundStruct;
```

Tabla 2.5: Cabecera del formato NeXT SND

Valor	Explicación
magic	Es un número usado para identificar al archivo, por el kit de sonido. Este vale "0x2e736e64" que es el equivalente de ".snd"
dataLocation	Indica donde están los datos del formato, ya sea a continuación de la cabecera del archivo o en algún otro.
dataSize	Indica el tamaño de la sección de datos (este valor no incluye a la cabecera).
dataFormat	Contiene el valor del formato de codificación del sonido que tiene la sección de datos.
samplingRate	Contiene el valor de la constante entera de grabación o reproducción. Por ser estructura en C cada valor tiene un nombre. Los valores soportados por el formato son: Para grabación:SND_RATE_CODEC = 8 kHz. Para reproducción: SND_RATE_LOW = 22.05 Mhz (Salida de baja velocidad) SND_RATE_HIGH = 44.1 Mhz (Salida de alta velocidad)
channelCount	Es el número de canales.
info	Es una cadena de terminación que permite poner comentarios de al menos 4 bytes de longitud. El tamaño de la cadena se pone una sola vez, al ser creada la estructura después, no es posible ampliarla; es común dejar esta cadena en blanco.

Tipos de codificación

La codificación es representada con un número entero de 32 bits. NeXT se reserva los valores del 0 al 255. Los valores más usados son:

Tabla 2.6: Tipos de codificación NeXT SND

Valor	Codigo	Explicación
0	SND_FORMAT_UNSPECIFIED	Formato no especificado, reservado para codificaciónes desconocidas.
1	SND_FORMAT_MULAW_8	Muestras de 8 bits similar al μ-Law
2	SND_FORMAT_LINEAR_8	Muestras de 8 bits
3	SND_FORMAT_LINEAR_16	Muestras de 16 bits
4	SND_FORMAT_LINEAR_24	Muestras de 24 bits
5	SND_FORMAT_LINEAR_32	Muestras de 32 bits
6	SND_FORMAT_FLOAT	Muestras de punto flotante
7	SND_FORMAT_DOUBLE	Muestras de doble precisión

Tabla 2.6: Tipos de codificación NeXT SND (Cont.)

Valor	Código	Codificación
8	SND_FORMAT_INDIRECT	Muestras fragmentadas
9	SND_FORMAT_NESTED	Muestras Consecutivas
10	SND_FORMAT_DSP_CORE	Datos del core del programa DSP
11	SND_FORMAT_DSP_DATA_8	Muestras de 8 bits de punto fijo
12	SND_FORMAT_DSP_DATA_16	Muestras de 16 bits de punto fijo
13	SND_FORMAT_DSP_DATA_24	Muestras de 24 bits de punto fijo
14	SND_FORMAT_DSP_DATA_32	Muestras de 32 bits de punto fijo
15	Reservado	
16	SND_FORMAT_DISPLAY	No despliega datos de audio
17	SND_FORMAT_MULAW_SQUELCH	Muestras compatibles con el estándar μ -Law
18	SND_FORMAT_EMPHASIZED	Valor de 16 bits lineal con énfasis
19	SND_FORMAT_COMPRESSED	Valor de 16 bit lineal con compresión
20	SND_FORMAT_COMPRESSED_EMPHASIZED	Una combinación de los dos anteriores
21	SND_FORMAT_DSP_COMMANDS	Comandos del programa DSP
22	SND_FORMAT_DSP_COMMANDS_SAMPLES	Comandos del programa DSP relacionados con los comandos
23	SND_FORMAT_ADPCM_G721	Nueva codificación que usa los estándares CCITT, estos son apoyados por SUN
24	SND_FORMAT_ADPCM_G722	
25	SND_FORMAT_ADPCM_G723_3	
26	SND_FORMAT_ADPCM_G723_5	
27	SND_FORMAT_ALAW_8	

Todos los tipos que incluyen en la etiqueta "DSP", son utilizados por el programa de tratamiento de sonido de la plataforma NeXT: Soundkit, hay que mencionar que el tipo 10 especifica el core del programa, el 21 los comandos usados para un determinado archivo, etc.

Una característica especial de este formato es el hecho que el sonido puede ser fragmentado en varios archivos con su propia velocidad y codificación, no obstante, para el usuario la operación del programa es transparente (no percibe el proceso); esto tiene en pro la facilidad en la manipulación del sonido, en contra tiene el alto gasto de espacio y procesamiento que requiere. Para indicar la fragmentación del sonido se inicia con un archivo del tipo 8.

2.4.3 AIFF/AIFF-C

Propietario: *Apple Computer Inc.*
Extensión: *.AIF*

Apple en 1990 presenta su formato para manejo de archivos de sonido, el AIFF cuyas raíces se remontan a la especificación "EA IFF 85" de Electronic Arts. En 1991, libera la versión AIFF-C, que sustituirá al AIFF. La razón de ello es que el formato AIFF no puede comprimir la información, cosa que el AIFF-C si realiza. Las diferencias entre el AIFF y AIFF-C son además de la compresión, cambios menores en la etiquetas que indican el nuevo formato y coordinan la compresión de los datos.

Tipos de datos usados en el archivo AIFF

Los tipos de datos usados para almacenar la información de los archivos AIIF y AIIF-C, están codificados en MSB.

Tabla 2.7: Tipos de datos usados en el formato AIFF

Tipo	Explicación
char	8 bits con signo. Guarda caracteres ASCII y números del -128 al 128.
unsigned char	8 bits sin signo. Guarda números del 0 al 255.
short	16 bits con signo. Guarda números del -32768 al 32767.
unsigned short	16 bits sin signo. Guarda números del 0 al 65535.
long	32 bits con signo. Guarda números del -2147483648 al 2147483647.
unsigned long	32 bits sin signo. Guarda números del 0 al 4294967295.
extended	Número de punto flotante de 80 bits, acorde con el estándar 754 del IEEE.
pstring	Cadena de texto similar al usado en pascal. La cadena cuenta con un byte que indica la longitud, además de un byte de complemento. La cadena debe forzosamente ser par.
ID	Cadena de 32 bits de largo. Son cuatros caracteres que indican el tipo de sistema operativo

Algunos campos presentan valores no contemplados. Ello es que son estructuras definidas y están incluidas en el bloque, o bien, sirven para hacer referencia a otro bloque, el cual necesitan.

Estructura del archivo

Un archivo AIIF-C se compone de una serie de bloques de datos. Cada bloque se compone de una cabecera de 8 bytes seguida de una serie de datos de longitud variable. La cabecera contiene una etiqueta de identificación llamada ckID; una etiqueta que indica la longitud del bloque llamada ckDataSize, el valor no incluye a la cabecera. Los tipos de bloques usados por el formato AIIF-C son:

Form

Contiene la información de identificación del archivo, así como su longitud, y engloba a los bloques que contienen la información o parámetros relacionados con el sonido en cuestión. La etiqueta de identificación del bloque es "FORM". Los datos son divididos en la etiqueta "formType" y bloques locales (o sub bloques).

La etiqueta formType indica que formato es usado, debe ser "AIFC" en este caso, entonces se le llama bloque "FORM AIFFC". Los datos son otros bloques, los cuales cuentan con su propia cabecera y sus propios datos, también llamados bloques locales. en Apple II o PC el archivo se almacena en un archivo tipo DOS. En Mac debe ser almacenado en datos tipo fork. La extensión usada para estos archivos es ".AFC".

Tipos de bloques locales

Los bloques locales representan parámetros, datos de sonido y demás datos. Todos los bloques son opcionales, con excepción del bloque Common y el bloque Sound Data.

Format Version

Este bloque indica la versión de archivo AIFF-C usado. Una versión es las modificaciones que sufre el formato. Los valores del bloque son:

Tabla 2.8: Valores del bloque Format Version

Etiqueta	Descripción
ckID	"FVER"
ckDataSize	4
Data	Llamado también *timeStamp* es el número de segundos transcurridos desde el 1o. de enero de 1904 hasta la fecha de creación del archivo.

Common

Describe los parámetros fundamentales del sonido. Los datos que incluye son:

Tabla 2.9: Valores del bloque Common

Etiqueta	Tipo de dato	Explicación
ckID	ID	"COMM"
ckDataSize	long	El valor es de 22 bytes más el tamaño de la etiqueta pstring. Se debe complementar a un valor par con ceros, siempre que sea necesario.
		Datos del bloque
numChannels	short	Número de canales en el archivo: 1 mono, 2 estéreo, y así. Cualquier número de canales puede ser representado.
numSample Frames	Unsigned long	Contiene el número de muestras por cuadro de la información.
samplesize	short	Número de bits en la muestra sin haber sido comprimido.
samplerate	extended	Velocidad de reproducción del sonido en muestras/seg. (Hz)
compressionType	ID	Tipo de compresión utilizado, los valores son: "NONE" sin comprimir. "ACE2" IIGS ACE (Audio Compresión/Expansión) 2 a 1. "ACE8" IIGS ACE (Audio Compresión/Expansión) 8 a 3. "MACE3" Audio Compresión/Expansión de Macintosh 3 a 1. "MACE6" Audio Compresión/Expansión de Macintosh 6 a 1.
compressionName	ID	nombre largo del tipo de compresión: "NONE"= "not compresed". "ACE2" = "ACE 2-to-1". "ACE8"= "ACE 8-to-3". "MACE3"= "MACE 3-to-1". "MACE6"="MACE 6-to-1".

Los tipos de compresión usados en AIFF-C son propietarias de Apple. El tipo de compresión está en relación directa con el equipo usado; las compresiones ACE son para equipo no Macintosh, por que las codificaciones MACE logran un mejor factor a base de usar las facilidades que ofrece la arquitectura de la Mac. El valor que acompaña al tipo de compresión indica su factor y su efectividad. Por ejemplo: La compresión ACE 8 a 3, indica que una cadena de 8 bytes (8 bits = 1 byte) es reducida a solo tres bytes, y el MACE 6 a 1 hace lo propio a un factor de 6 bytes por uno.

Sound Data

Almacena los cuadros de muestras. se estructura de la siguiente manera:

Tabla 2.10: Valores del bloque Sound Data

Etiqueta	Tipo	Explicación
ckID	ID	Identificador "SSND"
ckDataSize	long	Indica el tamaño en bytes del bloque.
		Datos del bloque
offset	Unsigned long	Indica donde inician los datos, comúnmente vale 0.
blocksize	Unsigned long	En conjunción con el offset. Contiene el tamaño en bytes de los bloques de datos de sonido.
soundData	Unsigned long	Número de cuadros que componen el sonido.

Marker

Contiene los punteros que marcan las posiciones de los datos de sonido, éste es usado en conjunción con otros bloques, por ejemplo, el bloque Instrument lo usa para hacer cíclica una secuencia musical. El bloque se estructura de la siguiente manera:

Tabla 2.11: Valores del bloque Marker

Etiqueta	Tipo	Explicación
ckID	ID	Identificador "MARK"
ckDataSize	long	Indica el tamaño en bytes del bloque.
numMarkers	unsigned short	Número de puntos que contiene el bloque
Marker	markers[ ]	Secuencia de punteros.

Los punteros al ser usados por varias aplicaciones, tienen un nombre propio para su identificación. Cada puntero se compone de la siguiente manera:

Tabla 2.12: Valores del bloque Marker (ID)

Etiqueta	Tipo	Explicación
ID	short	Identificador. Este debe ser un número entero negativo.
Position	unsigned Long	El puntero indica el número de cuadro.
MarkerName	pstring	Nombre del puntero. Debe ser el número de bytes par.

Comments

Sirve para guardar comentarios acerca de los punteros. Solo se permite un bloque de comentarios en el archivo. El bloque se compone de los siguientes campos:

Tabla 2.13: Valores del bloque Comments

Etiqueta	Tipo	Explicación
ckID	ID	Identificador "COMT"
ckDataSize	long	Indica el tamaño en bytes del bloque.
numMarkers	unsigned short	Número de comentarios en el bloque
Marker	markers[]	Secuencia de comentarios.

Cada comentario se compone a su vez de las siguientes partes:

Tabla 2.14: Descripción del bloque Comments

Etiqueta	Tipo	Explicación
TimeStamp	unsigned long	Fecha de creación del comentario
MarkerId	Marker	Contiene el valor de identificación del puntero, para hacer referencia a esté. Debe valer cero si no hay un puntero ligado con el comentario.

Count	unsigned short	Longitud del texto.
Text	char	Texto del comentario.

Instrument

Define los parámetros básicos que un instrumento, puede usar para reproducir los datos de un sonido. Una porción del sonido puede ser repetida o alargada. Esta porción es conocida como segmento loop, y sólo es interrumpida por alguna señal como la presión de una tecla. Hay dos maneras de hacer un loop: loop hacia adelante y loop hacia adelante/hacia atrás (o loop ping pong). El loop hacia adelante repite varias veces el mismo segmento, mientras el loop ping pong hace una pasada hacia adelante e inmediatamente una hacia atrás y así sucesivamente.

La estructura que define un loop es como sigue:

Tabla 2.15: Valores del bloque Loop

Etiqueta	Tipo	Explicación
PlayMode	short	Define el tipo de loop a realizar: 0 ninguno. 1 loop hacia adelante. 2 loop ping pong.
BeginLoop	MarkerId	Indicador del inicio del loop. Usa un puntero del bloque Marker.
EndLoop	MarkerId	Indicador del fin del loop. Usa un puntero del bloque Marker.

El bloque Instrument se compone de los siguientes campos:

Tabla 2.16: Valores de bloque Instrument

Etiqueta	Tipo	Explicación
ckID	ID	Identificador "INST"
ckDataSize	long	Indica el tamaño en bytes del bloque, siempre vale 20.
baseNote	char	Nota del sonido original. Las unidades están en números MIDI del 0 al 127.
detune	char	Es el valor de ajuste de la nota. Las unidades están en céntimos (1/100 de un semitono) y de rango desde -50 a +50. Si es negativo el valor, la nota debe ser más baja; en caso contrario debe ser más alta.
lowNote	char	Es la nota más baja permitida que puede tocar el instrumento.
highNote	char	Es la nota más alta permitida que puede tocar el instrumento.
lowVelocity	char	Velocidad mínima de reproducción de la nota. Medido en unidades MIDI, 1 (la más lenta) a 127 (la más alta).
highVelocity	char	Velocidad máxima de reproducción de la nota.
gain	short	Es el volumen con el cual el sonido debe ser reproducido, con respecto del volumen base. Medido en decibeles, si es 6db (decibeles) se incrementara al doble el volumen; si es de -6 db el volumen es reducido a la mitad. Una vez tocada la nota el volumen debe ser ajustado a su valor normal.
sustainLoop	Loop	Especifica un loop cuando un instrumento está sosteniendo una nota.
releaseLoop	Loop	Especifica el fin de un loop, este usualmente ocurre por una entrada de teclado.

Se permite un solo bloque. Actualmente se discute incluir más de un instrumento en el archivo.

MIDI Data

Contiene datos MIDI estándares, conforme avanza el tiempo surgen nuevos instrumentos regulados por el estándar, estos instrumentos pueden no estar incluidos en la especificación AIFF-C (hasta que sea revisada formalmente). Usando la referencia que ofrece el MIDI es posible reproducir tales instrumentos o sustituirlos adecuadamente. Puede haber tantos bloques como datos MIDI sean necesarios. El bloque MIDI se compone de:

Tabla 2.17: Descripción del bloque MIDI Data

Etiqueta	Tipo	Explicacion
ckID	ID	Identificador "MIDI"
ckDataSize	long	Indica el tamaño en bytes del bloque.
MIDIdata	Unsined char	Datos MIDI.

Audio Recording

Guarda información pertinente a dispositivos de grabación de audio.

Tabla 2.18: Descripción del bloque Audio Recording

Etiqueta	Tipo	Explicacion
ckID	ID	Identificador "AESD"
ckDataSize	long	Indica el tamaño en bytes del bloque. Siempre vale 24.
AESChannelStatusData	Unsined char	Los 24 bytes están definidos en el documento AES Recommended Practice for Digital Audio Engineering - Serial Transmissio Format for Linearly Represented Digital Audio Data.

Application Specific

Sirve para diversos propósitos de los creadores de software. Contiene:

Tabla 2.19: Descripción del bloque Application Specific

Etiqueta	Tipo	Explicacion
ckID	ID	Identificador "APPL"
ckDataSize	long	Indica el tamaño en bytes del bloque.
OSType	ApplicationSignature	Identifica una particular aplicación. En aplicaciones Apple se usa la etiqueta "stoc"; si es apple II la etiqueta es "pdos".
data	char	Contiene los datos para la aplicación.

Bloques Text

Los bloques Text son una serie de comentarios simples en texto ASCII. Los bloques que pertenecen a tal categoría son Name: Nombre del sonido; Author: Nombre del autor; Copyright: Aviso de derechos reservados; Annotation: Comentarios del autor. No se permite más un bloque por archivo. Todos estos bloques se definen por igual.

Tabla 2.20: Descripción del bloque Text

Etiqueta	Tipo	Explicacion
ckID	ID	Identificador según sea el caso: "NAME", "AUTH", "©" O "ANNO"
ckDataSize	Es el número de caracteres usados en el texto.	
MIDIdata	Unsined char	El texto a mencionar. En el caso de copyright el texto escrito se le añade el prefijo "©".

2.4.4 Waveform Audio (WAV)

Propietarios: ***IBM corp. y Microsoft Co.***
Extensión: ***.wav***

El estándar MS RIFF es resultado del trabajo conjunto de IBM y Microsoft para desarrollar un estándar fácil de usar en aplicaciones de multimedia, diseñado para las necesidades de multimedia específicas de las IBM PC y compatibles, además de los sistemas operativos MS-Windows X y OS/2. Microsoft distribuye la especificación en los documentos "The multimedia Development Kit 1.0 Programmer's Reference" y "The Windows 3.1 Software Development Kit (SDK)'s Multimedia" de Microsoft Press.

El formato Waveform Audio File Format (WAVE) como se le conoce formalmente es una parte de este estándar. El formato es una serie bloques que describen un sonido digitalizado. El formato WAVE no tiene bloques especiales para almacenar música en concordancia con MIDI. El formato se almacena en codificación LSB, exceptuando donde se indique en el estándar presente o futuras revisiones. La extensión en cualquier sistema operativo es "*.wav*". En el estándar, el nombre simplificado de cada bloque se encierra entre los caracteres mayor/menor que < >. Cada bloque es descrito usando la estructura del lenguaje C++, y el tipo de datos usados por el formato corresponde a la capacidad del compilador C de microsoft. Los datos hexadecimales están en formato 0xXXXX.

Los programas que lean el formato deben ignorar los bloques desconocidos. El orden de los bloques es opcional con excepción del bloque <fmt-ck> que debe ir antes del bloque <wave-data>. Todo se engloba en el bloque general llamado <WAVE-form>. El símbolo "->" indica que el bloque se subdivide, sea en otros bloques o en campos. La estructura del formato es como sigue:

```
<WAVE-form> ->
        RIFF('WAVE'          Etiqueta de información
            <fmt-ck>         Formato de datos
            [<fact-ck>]      Bloque de información
            [<cue-ck>]       Puntos
            [<playlist-ck>]  Lista de sonidos
            [<assoc-data-list>] Lista de datos asociados
            <wave-data> )    Datos del sonido
```

La etiqueta "WAVE" es una simple cadena al inicio del archivo que indica al programa que está leyendo un archivo ".wav". Los bloques en que se compone el archivo se explican a continuación.

Bloque WAVE format
El bloque WAVE format <fmt-ck> especifica el formato del bloque <wave-data>. El bloque es definido como sigue:

<fmt-ck> -> fmt(<common-fields> <format-specific-fields>)

```
<common-fields> ->
struct
    {
    WORD  wFormatTag;          Categoría del formato
    WORD  wChannels;           Número de canales
    DWORD dwSamplesPerSec;     Velocidad
    DWORD dwAvgBytesPerSec;    Alineación de bloques en el buffer
    WORD  wBlockAlign;         Tamaño del bloque de datos
    }
```

Los campos en el sub-bloque <common-fields> son:

Tabla 2.21: Campos del sub-bloque Common-fields

Campo	Descripción
wFormatTag	Un número que indica la categoría de Wave format en el archivo. Los tipos creados por terceros deben ser registrados ante Microsoft. Más adelante explico las categorías aceptadas en el formato.
wChannels	Número de canales representados en los datos de sonido, 1 para monoaural o 2 para estéreo.
dwSamplesPerSec	Velocidad en muestras/seg. en el cual cada canal debe ser tocado.
dwAvgBytesPerSec	El número promedio de bytes por segundo al cual los datos del sonido debe ser transferidos.
wBlockAlign	Alineación de los bloques de los datos en el buffer. Indica el orden en que están acomodados los bloques de datos para su reproducción.

El bloque <format-specific-fields> consiste de parámetros de cero o más bytes, que pertenecen a una categoría dada. Las categorías son explicadas a continuación:

Categorías WAVE

La categoría de formato de un archivo WAVE, es especificado por el valor del campo wFormatTag del bloque <fmt-ck>. De la categoría usada en el archivo WAVE depende la representación de los datos y el contenido del sub-bloque <format-specific-fields>.

Tabla 2.22: Categorías WAVE

Valor de la etiqueta wFormatTag	Valor hexadecimal	Categoría
WAVE_FORMAT_PCM	(0x0001)	Formato Modulación de código de pulso (Pulse Code Modulation, PCM) de Microsoft.
IBM_FORMAT_MULAW	(0x0101)	Formato μ-Law de IBM
IBM_FORMAT_ALAW	(0x0102)	Formato A-Law de IBM
IBM_FORMAT_ADPCM	(0x0103)	Formato Modulación de código diferencial adaptativo (Adaptive Differential Pulse Code Modulation, AVC) de IBM.

Estándar A-Law y μ-Law

El formato MS-RIFF soporta la especificación CCITT G.711 de la CCITT para los estándares A-Law/ μ-Law, mono o estéreo, a velocidades de 8000, 11025, 22050, 44100 kHz. Los campos del formato son:

Tabla 2.23: Campos del estándar A-Law y μ-Law

Campo	Descripción
wFormatTag	Esta etiqueta puede ser "WAVE_FORMAT_ALAW" o "WAVE_FORMAT_MULAW"
nChannels	Número de canales en el sonido, 1 para mono, 2 estéreo.
nSamplesPerSec	Velocidad del archivo
nAvgBytesPerSec	Promedio de velocidad de los datos, con este campo el software puede estimar el tamaño del buffer a usar.
nBlockAlign	Tamaño de los bloques en bytes, el software procesa un número de bytes (el valor del campo) a la vez.
wBitsPerSample	Número de bits por muestra de datos
cbExtraSize	El tamaño en bytes de la información extra en la cabecera. En este caso vale cero.

Formato PCM

Los valores en el sub-bloque <format-specific-fields> en esta categoría se definen como sigue:

```
<PCM-format-specific> ->
struct
{
WORD wBitsPerSample;      Tamaño de muestra
}
```

El campo wBitsPerSample especifica el número de bits de datos usados para representar cada muestra de cada canal. Si hay varios canales, el tamaño de muestra es el mismo para todos. Para los datos PCM, el valor del campo wAvgBytesPerSec debe ser igual a la siguiente fórmula redondeada hacia el siguiente número entero:

$$\text{wChannels} \times \text{wBitsPerSecond} \times \frac{\text{wBitsPerSample}}{8}$$

El campo wBlockAlign debe ser igual al siguiente número entero:

$$\text{wChannels} \times \frac{\text{wBitsPerSample}}{8}$$

Empacado de datos para los archivos PCM

En un archivo de un solo canal, las muestras son guardadas consecutivamente. Para archivos estéreo el canal 0 representa el canal izquierdo y el canal 1 el derecho. Los siguientes diagramas ilustran el orden de los datos en archivos monoaural y estéreo de 8 y 16 bits:

Archivo monoaural de 8 bits PCM

Muestra 1	Muestra 2	Muestra 3	Muestra 4
Canal 0	Canal 0	Canal 0	Canal 0

Archivo estéreo de 8 bits PCM

Muestra 1		Muestra 2	
Canal 0	Canal 1	Canal 0	Canal 1
Izquierda	Derecha	Izquierda	Derecha

Archivo monoaural de 16 bits PCM

Muestra 1		Muestra 2	
Canal 0	Canal 0	Canal 0	Canal 0
Byte bajo	byte alto	Byte bajo	byte alto

Archivo estéreo de 8 bits PCM
Muestra 1

Canal 0	Canal 0	Canal 1	Canal 1
Byte bajo	byte alto	Byte bajo	byte alto
Izquierda	Izquierda	Derecha	Derecha

Formato de los datos de las muestras

Cada muestra es contenida en un número entero i. El tamaño de i es el número más pequeño de bytes requerido para contener el tamaño de la muestra especificada. El byte menos significativo es almacenado primero. Los bits que representan la amplitud de la muestra son colocados en los bits más significativos de i. y los bits restantes se ponen en cero.

Por ejemplo, si el tamaño de la muestra es de 12 bits, entonces cada muestra es almacenada en un entero de dos bytes. Los cuatro bits menos significativos del primer byte deben valer cero.

El formato de los datos y los valores máximos y mínimos para las muestras en waveform PCM de varios tamaños son como sigue:

Tabla 2.24: Tamaño de las muestras

Tamaño de la muestra	Formato de datos	Valor máximo	Valor mínimo
Uno a 8 bits	Entero sin signo	255 (FFh)	0
9 ó más bits	Entero con signo	valor positivo más grande de i	valor más negativo de i

Por ejemplo, los valores máximos, mínimos y medios para datos de 8 y 16 bits waveform PCM son como sigue:

Tabla 2.25: Valores del PCM

Formato	Valor máximo	Valor mínimo	Valor medio
PCM de 8 bits	255	0	128
PCM de 16 bits	32767	-32768	0

Almacenamiento de los datos WAVE

Los datos wave (información acerca de la onda que hace el sonido) es colocado en el bloque <wave-data>, y su estructura es como sigue:

<wave-data> -> { <data-ck> : <data-list> }

<data-ck> -> data(<wave-data>)
<wave-list> -> LIST('wavl' { <data-ck> : Muestras acerca de la onda.
<silence-ck> }...) Silencio.
<silence-ck> -> slnt(<dwSamples:DWORD>) Valor de conteo del número de muestras de silencio.

Nota: El bloque <slnt> representa silencio, no necesariamente secuencias de ceros. En un dato PCM de 16 bits, si el último valor de la muestra tocada antes del silencio es 10000, entonces los datos todavía están en el convertidor data/audio, este debe mantener el valor 10000. Si un cero es usado, un clic puede ser escuchado al inicio y al final de la sección de silencio. Si el sonido empieza con una sección de silencio. Entonces un valor cero debe ser usado hasta que termine dicha sección. Un clic puede ser escuchado si la siguiente sección de silencio empieza con un valor no cero.

Bloque FACT

El bloque fact <fact-ck> contiene información importante acerca del contenido del archivo WAVE. El bloque es estructurado así:

<fact-ck> -> fact(<dwFileSize:DWORD>) Número de muestras

Este bloque es requerido si los datos del sonido están guardados en un bloque LIST y para todas las categorías que usen compresión. No es usado en los archivos PCM que usan el formato del bloque data.

Este bloque se ampliará conforme vayan incluyéndose nuevos tipos de categorías. Los campos se añadirán a partir del campo <dwFileSize>.

Bloque Cue-Points

El bloque cue-points <cue-ck> identifica una serie de posiciones en los datos del sonido. El bloque tiene la siguiente forma:

<cue-ck> -> cue(<dwCuePoints:DWORD> Cantidad de puntos points
<cue-point>...) punto en el sonido table

```
<cue-point> -> struct {
DWORD  dwName;
DWORD  dwPosition;
FOURCC fccChunk;
DWORD  dwChunkStart;
DWORD  dwBlockStart;
DWORD  dwSampleOffset;
}
```

Los campos del bloque <cue-point> son:

Tabla 2.26: Campos del bloque Cue-point

Campo	Descripción
dwName	Especifica el nombre del punto. Cada registro <cue-point> tiene un nombre único.
dwPosition	Indica la posición de la muestra del punto seleccionado. Este es una muestra secuencial dentro del orden de reproducción del sonido.
fccChunk	Especifica el nombre o el bloque de identificación del bloque que contiene el punto.
dwChunkStart	Especifica la posición de inicio del bloque que contiene el punto. La posición es relativa al inicio de la sección de datos.
dwSampleOffset	Indica la posición del punto en relación con el inicio del bloque.

Bloque Playlist

El bloque playlist <playlist-ck> especifica el orden en que serán tocados una serie de puntos. El bloque es como sigue:

```
<playlist-ck> ->  plst(
        <dwSegments:DWORD>    Número total de los segmentos
        <play-segment>...)        Tabla de segmentos a tocar

        <play-segment> ->  struct {
                DWORD dwName;
                DWORD dwLength;
                DWORD dwLoops;
        }
```

Los campos <play-segment> son:

Tabla 2.27: Campos Play-segment

Campo	Descripción
dwName	Especifica el nombre del punto. Este valor debe coincidir con algunos de los nombres listados en la tabla de puntos.
dwLength	Especifica la longitud de la sección en muestras.
dwLoops	Indica el número de veces que será tocada la sección.

Bloque de datos asociados

La lista de los datos asociados <assoc-data-list> provee la habilidad de guardar la información como etiquetas de secciones de los datos. Es definido como sigue:

```
<assoc-data-list> ->  LIST('adtl'
        <labl-ck>           Etiqueta
        <note-ck>           Nota
        <ltxt-ck>           Texto con la longitud de los datos
        <file-ck>)          Datos del archivo

          <labl-ck> ->      labl(<dwName:DWORD>
                            <data:ZSTR>)

          <note-ck> ->      note(<dwName:DWORD>
                            <data:ZSTR>)
```

```
<ltxt-ck> ->    ltxt(<dwName:DWORD>
                <dwSampleLength:DWORD>
                <dwPurpose:DWORD>
                <wCountry:WORD>
                <wLanguage:WORD>
                <wDialect:WORD>
                <wCodePage:WORD>
                <data:BYTE>...)

<file-ck> ->    file(<dwName:DWORD>
                <dwMedType:DWORD>
                <fileData:BYTE>...)
```

Información de etiquetas y notas

Los bloque "labl" y "note" tienen campos similares. El bloque "Labl" contiene una etiqueta o titulo asociado a un punto. El bloque "note" guarda un texto de comentario para un punto. Sus campos son:

Tabla 2.28: Bloques de etiquetas Labl y Note

Campo	Descripcion
dwName	Especifica el nombre del punto. Este valor debe coincidir con uno de los nombres listados en la tabla de puntos.
data	Especifica una cadena de terminación nula que contiene una etiqueta (para el bloque "labl") o un comentario (para el bloque "note").

Texto con información de la longitud de los datos.

El bloque "ltxt" define texto que es asociado con una segmento de datos de longitud especifica. Los campos del bloque son:

Tabla 2.29: Bloques de etiquetas Ltxt

Campo	Descripcion
dwName	Especifica el nombre del punto. Este valor debe coincidir con uno de los nombres listados en la tabla de puntos.
dwSampleLength	Especifica el número de muestras en el segmento de los datos.
dwPurpose	Especifica el propósito o el tipo del texto.
wCountry	Especifica el código del país para el texto. (véase tabla 2.32)
wLanguage,wDialect	Define el código de leguaje y dialecto respectivamente. (véase tabla 2.33)
wCodePage	Especifica el código de pagina para el texto.

Tabla 2.30: Código de países

Codigo	Pais
000	Ninguno
001	USA
002	Canadá
003	Latinoamérica
030	Grecia
031	Países bajos
032	Bélgica
033	Francia
034	España
039	Italia
041	Suiza
043	Austria
044	Reino Unido
045	Dinamarca
046	Suecia
047	Noruega
049	Alemania
052	México
055	Brasil

061	Australia
064	Nueva Zelandia
081	Japón
082	Corea
086	República popular de China
088	Taiwan
090	Turquía
351	Portugal
352	Luxemburgo
354	Islandia
358	Finlandia

Tabla 2.31: Código de lengua y dialecto

Código de lenguaje	Código de dialecto	Lenguaje
0	0	Ninguno
1	1	Árabe
2	1	Búlgaro
3	1	Catalán
4	1	Chino Tradicional
4	2	Chino Simplificado
5	1	Checo
6	1	Danés
7	1	Alemán
7	2	Alemán suizo
8	1	Griego
9	1	Ingles USA
9	2	Ingles UK
10	1	Español
10	2	Español Mexicano
11	1	Finlandés
12	1	Francés
12	2	Francés Belga
12	3	Francés Canadiense
12	4	Francés Suizo
13	1	Hebreo
14	1	Húngaro
15	1	Islandés
16	1	Italiano
16	2	Suizo Italiano
17	1	Japonés
18	1	Coreano
19	1	Holandés
19	2	Holandés Belga
20	1	Noruego - Bokmal
20	2	Noruego - Nynorsk
21	1	Polaco
22	1	Portugués Brasileño
22	2	Portugués
23	1	Rhaeto-Romanico
24	1	Rumano
25	1	Ruso
26	1	Serbo-Croato (Latino)
26	2	Serbo-Croato (Cirilico)
27	1	Slovaco
28	1	Albanés
29	1	Sueco
30	1	Thai
31	1	Turco
32	1	Urdu
33	1	Bahasa

Información incrustada del archivo

El bloque "file" tiene información descrita en otros formatos (por ejemplo, un archivo "RDIB" o un texto ASCII). El bloque se compone de:

Tabla 2.32: Descripción del bloque File

Campo	Descripción
dwName	Especifica el nombre del punto. Este valor debe coincidir con uno de los nombres listados en la tabla de puntos.
dwMedType	Especifica el tipo de archivo contenido en el campo fileData. Si la sección fileData contiene un forma RIFF, el campo dwMedType es el mismo tipo que la forma RIFF.
filedata	Contiene el archivo de información.

2.4.5 MOD

Propietario: ***Varios***
Extensiones: ***varias***

Existe una familia de formatos de música, conocida como *modules*. Los modules son archivos de música digitalizada que no soportan almacenamiento de sonido, es decir, son formatos dedicados exclusivamente a reproducir música bajo el contexto de la notación musical estándar. Pero la diferencia fundamental entre los modules y otros formatos que manipulan música como el AIFF-C; es que no únicamente usan instrumentos definidos en el estándar MIDI, sino que también instrumentos definidos por el usuario como pueden ser ruidos, voces humanas, etc. El número de instrumentos se refiere la cantidad de instrumentos almacenados en un archivo; entiéndase por instrumento cualquier secuencia sonora que va desde un instrumento como la flauta hasta una voz repetitiva como en la música de estilo "rap".

Los miembros de la familias module son conocidos también como archivos "MOD". Todos los formatos MOD son propuestos por programas de edición de música conocidos como trackers, estos programas están orientados a facilitar la creación de música al dar más facilidades a los músicos en la creación, al permitirles usar sus propios instrumentos y/o efectos. Los formatos más conocidos son enumerados a continuación:

Tabla 2.33: Tipos de formatos MOD

Nombre	Extensión	No. de tracks	Instrumentos por track	No. de bits por muestra	Tracker
M.K.	MOD	4	31	8 bits	ProTracker
xCHN	MOD	6/8	31	8 bits	FastTracker 1
FLTx	MOD	4/8	31	8 bits	StarTrekker
NST	MOD	4	15	8 bits	Noise Tracker
669	669	8	64	8 bits	Composer 669
UNIS	669	8	64	8 bits	UNIS669
MTM	MTM	1-32	31	8 bits	MultiModuleEdit 1.01b
STM	STM	4	31	8 bits/variable	ScreamTracker 2.xx
S3M	S3M	16	99	8 bits/variable	ScreamTracker 3.2
ULT	ULT	1-32	64	8/16 bits/variable	UltraTracker 1.6
XM	XM	2-32	128	8/16 bits/variable	FastTracker II
FAR	FAR	16	64	8/16 bits	Farandole Composer 1.00
WOW	WOW	8	31	8 bits	Grave Composer
OKT	OKT	4-8	255	7/8 bits	Oktalyzer
MED	MED	4-8	32	8 bits	MED/OctaMED

La familia de formatos MOD es muy especial porque hay formatos que mantienen leves diferencias entre si siendo difícil de distinguirlas, a no ser las características en el número de tracks (pistas) usados. El campo del nombre del formato no tiene un criterio estable, ya que se puede encontrar al principio, en medio, o al final del archivo. Las extensiones de los formatos no son un método seguro ya que como es obvio, varios formatos comparten una misma extensión.

Los tracks son sinónimo del canal de sonido de otros formatos, es decir, cada track es una secuencia de notas (incluyendo silencios) que es reproducida a un tiempo. El

desplegar 32 tracks es libertad del programa reproductor dirigir la salida de tales tracks, pudiendo ir del monoaural hasta el cuadrafonico y aún más. El número de bits por muestra indica la calidad del sonido, entre más bits tenga una muestra mejor será la calidad del sonido. Una ventaja adicional es que un archivo MOD puede almacenar un tema muy largo en una pequeña fracción de espacio, mucho menor de lo que usan formatos como el AIFF-C, SND, y WAV.

Todos los formatos a pesar de sus diferencias manejan las siguientes estructuras y convenciones. La estructura más grande en los archivos MOD es el pattern, esté se compone de un conjunto de tracks, todo el conjunto de patterns deben estar ordenados en secuencia de reproducción. Cada track se divide en una matriz donde cada renglón es la unidad mínima de tiempo en la que una nota es almacenada; y cada columna es un elemento del track tal como volumen, efectos, notas, etc.

Una característica que reduce el tamaño del archivo en algunos formatos es que un pattern es susceptible de ser repetido cuantas veces sea necesario, inclusive si no está en orden estricto. De todos los formatos MOD la versión M.K. de protracker es la más ampliamente aceptada, además es la primera en ser desarrollada, y la comunidad dedicada a la creación de formatos MOD la considera el estándar base. Por las anteriores razones me ocuparé de este formato detalladamente.

M.K. Protracker

El primer formato MOD fue desarrollado por Karsten Obarski para la plataforma Commodre Amiga. El mod M.K. reproduce hasta cuatro canales a un tiempo con un tamaño de pattern de 64 renglones.

El número máximo de instrumentos que puede disponer el archivo es de 31; cada instrumento contiene a su vez un segmento de sonido, bytes que controlan su volumen, su duración, las veces que se repite y una etiqueta que lo identifica. Cada segmento o nota ocupa 8 bits. Como mencione anteriormente los mods no solo incluyen notas musicales sino otros tipos de secuencias sonoras. El archivo entero está codificado en MSB. El formato M.K. utiliza la siguiente estructura:

Tabla 2.34: Cabecera del archivo M.K.

Offset	Extensión en bytes	Descripción
0	20	Nombre del tema musical. El texto ASCII por definición debe ir en mayúsculas. En caso de sobrar espacio debe ser rellenado con caracteres nulos.

Bloques de instrumentos

Cada instrumento ocupa 28 bytes de extensión. La sección de instrumentos va del Offset 20 al 948. Cada instrumento se compone de los siguientes campos:

Tabla 2.35: Bloque de instrumentos

Extensión en bytes	Descripción
22	Nombre del instrumento. Usando el caracter "#" como prefijo de la cadena sirve también para guardar comentarios del autor.
2	Longitud del instrumento en bytes.
1	Los cuatro bits menos importantes del byte son usados para ajustar la frecuencia del sonido. Los demás valen cero.
1	Volumen del sonido. El rango de sonido va de 0 a 64 (64 es el valor máximo y cada valor es un -0.1 de decibel).
2	Punto en el instrumento en que este es repetido. Si no se repite se deja el valor en cero.
2	Longitud en bytes de la sección que es repetida. Vale cero si no se usa.

Sección de parámetros

En esta sección se indican valores relacionados con la reproducción musical. Además de una etiqueta de identificación.

Tabla 2.36: Sección de parámetros

Offset	Extensión en bytes	Descripción
950	1	Longitud del tema. El rango va de 1 a 128. Indica cuantos patterns existen en el archivo.
951	1	Este byte debe valer 127.
952	128	Es una lista de los patterns que componen el tema musical. Se permite repetir los patterns tantas veces como sea necesario.
1080	4	Etiqueta de identificación. Usualmente es la cadena "M.K.", aunque algunos trackers la sustituyen por las cadenas "FLT4", "FLT8" y "M!K!".

Sección de datos

A partir del byte 1084 comienza la sección de datos del archivo. La longitud total está en relación con el número de patterns usados. Cada pattern ocupa 1024 bytes de extensión. Las cuatro notas de que se compone cada uno de los 64 renglones, están ordenadas de izquierda a derecha. Cada nota tiene 4 bytes de extensión. La estructura de los cuatro bytes es como sigue:

El número de muestras se refiere al valor de la cadena de sonido. El periodo de la nota indica el momento en que debe ser tocado el segmento de sonido, es decir un instrumento tiene un segmento de sonido, al darse el periódo se puede ajustar para escuchar el segmento en la nota correspondiente. Simplemente se compara el valor con tablas de sonido MIDI. Los efectos como su nombre lo indica, dan efectos especiales a un sonido (o nota). Los efectos aceptados en el formato M.K. se indican en la tabla 2.39.

Tabla 2.37: Efectos de sonido.

Efectos	parámetros
0 - Arpeggio	0xy : x- Añade le primer medio tono, y-el 2o.
1 - Aumentar tiempo	1xx : subir velocidad
2 - Disminuir tiempo	2xx : bajar velocidad
3 - Portamento	3xx : subir/bajar velocidad

4 - Vibrato	4xy : x-velocidad, y-profundidad
5 - Portamento + Variación de tiempo	5xy : x-subir velocidad, y-bajar velocidad
6 - Vibrato + Variación de tiempo	6xy : x-subir velocidad, y-bajar velocidad
7 - Tremolo	7xy : x-velocidad, y-profundidad
8 - No usado	
9 -	9xx : offset (23 -> 2300)
A - Ajuste de volumen	Axy : x-subir la velocidad, y-downspeed
B - Salto de posición	Bxx : posición del tema
C - Ajuste de volumen	Cxx : volumen, 00-40
D - Salto	Dxx : posición de salto en el sig. pattern
E - Filtros del grupo E	Exy : véase abajo..
F - Ajuste de tiempo	Fxx : velocidad(00-1F) / tiempo (20-FF)
E0- Ajuste de filtro	E0x : 0-filtro activo, 1-
E1- Aumento fino de tiempo	E1x : valor
E2- Reducción fina de tiempo	E2x : valor
E3- Control Ghisenndo	E3x : 0-apagado, 1-encendido
E4- Set Vibrato Waveform (tipo de onda)	E4x : 0-seno, 1-sierra, 2-cuadrado
E5- Creación de un loop	E5x : punto de inicio del loop
E6- Salto hacia un loop	E6x : salto al loop, tócalo x veces
E7- Set Tremolo Waveform	E7x : 0-seno, 1-sierra, 2-cuadrado
E8- No usado	
E9- Repasar la Nota (retrig)	E9x : repaso de la nota c/ x espacios
EA- Aumento fino de volumen	EAx : subir x al volumen
EB- Reducción fina de volumen	EBx : reducir x al volumen
EC- Corte de la nota	ECx : corte de la nota + x espacios
ED- Retraso de la Nota	EDx : retardo de la nota x
EE- Retraso del Pattern	EEx : retardo del pattern x
EF- Invertir un Loop	EFx : velocidad

Debido a su estructura similar al estándar MIDI, el usuario puede sustituir los instrumentos a su gusto; en el caso de los elementos no MIDI, solamente se debe guardar la cadena correspondiente a la nota presente en el tema musical. De hecho hay muchas más variantes que las mencionadas en este trabajo, pero muchos propietarios no permiten su uso o distribución gratuita.

Formatos Raw

Los formatos Raw se caracterizan por su falta de flexibilidad, esta es debida a que sólo manejan una velocidad definida, un sólo tipo de compresión y están limitados a un software determinado. Algunos formatos tiene versiones para sonido y versiones para MIDI.

Este tipo de formatos son principalmente usados por las tarjetas de sonido de la familia IBM - PC, a raíz de ello, los desarrolladores de software deben incluir en sus programas archivos con los formatos Raw de las tarjetas más populares. Como es obvio suponer, este tipo de formatos se encuentra en franca decadencia, y mediano plazo, es probable su sustitución total por alguno formato más independiente del hardware.

2.4.6 Tandy Deskmate

Propietario: *Tandy Co.*
extensión: *.snd*

Desarrollado por Tandy para su línea de computadoras, el formato ha alcanzado cierta aceptación en el mundo de las computadoras IBM PC, especialmente en programas que hacen uso de la bocina integrada que es suministrada en cualquier computadora.

El formato Tandy puede almacenar secuencias de sonido o musicales. En el caso de almacenar un segmento de sonido, lo puede hacer usando velocidades de 5500 Hz, 11 kHz o 22 kHz., para la pista musical sólo es posible grabarla o reproducirla a una velocidad fija de 11 kHz con un máximo de 16 notas diferentes por instrumento.

Estructura
El formato se compone de un archivo que contiene una cabecera fija y el segmento de sonido, si se desea almacenar sonido el segmento de sonido se sustituye por una lista de las notas a tocar, además se indexa un segundo archivo el cual contiene las muestras de bits de las cuales se compone la nota a tocar. El archivo de música no se puede leer si falta el de instrumentos.

El formato puede usar compresión de datos si es necesario; se usan diferentes tipos según sea lo que se va comprimir: sonido o música. Los campos de la cabecera que no sean usados son invalidados mediante el uso de los valores especificados.

Tabla 2.38: Cabecera del formato Tandy

Offset	Longitud en Bytes	Descripción
0	1	Byte de identificación del formato vale 1Ah.
1h	1	Código de Compresión: 0 = sin compresión. 1 = compresión de música. 2 = compresión de sonido.
2h	1	Número de notas en el archivo de instrumentos. Vale 1 si es archivo de sonido.
3h	1	Número de Instrumentos. 0 si es archivo de sonido. 0FFh si el archivo de instrumentos no tiene valor definido. El rango aceptado de instrumentos va de 1 a 32.
4h	10	Nombre del sonido o instrumento. Se llena a la derecha con valores nulos si es de menos de 10 caracteres.
0Eh	8	Velocidad en muestras por segundo. Nótese que puede ser definido cualquier valor además de los 5.5 kHz, 11 kHZ y 22 kHZ; pero no será leído por los programas de Tandy.
10h	variable	Empieza la sección de notas, cada una de 28 bytes. El número de notas está dado en el offset 2h.

Tabla 2.39: Archivo de Música

Offset	Longitud	Descripción
0h	byte	Define el tono de la nota: 1 = A1 tono estándar americano; 2 = A#1 nivel más alto; etc. A1 es la nota más baja permitida, la más alta es B6 (3Fh).
1h	byte	Archivos de sonido y de instrumentos. si no hay definición del valor de notas, vale 0. En caso contrario vale 0FFh.
2h	2 bytes	Rango de la nota, el primer byte es el limite inferior, el segundo es el limite superior.
4h	dword	Byte donde las muestras para esta nota, no están comprimidos.
8h	dword	Lista la longitud de los datos, Si estos están comprimidos, de no estarlo se pone 0.
0Ch	4 bytes	No usado, se pone en cero.
10h	dword	Número de muestras en cada nota.
14h	dword	Número de muestra que marca el inicio de la región de sostenimiento de la nota. vale cero si no está definido la región.
18h	dword	Número de muestra que marca el fin de la región de sostenimiento de la nota. Vale cero si no está definido la región.

FORMATOS DE VIDEO

3. Video

Uno de los inventos que más ha afectado a la sociedad del siglo XX es el **video**. El video tiene sus raíces en el cine y en la televisión. Actualmente ha venido a reforzar las aplicaciones computacionales, entre ellas la tan novedosa "multimedia". El video consiste de dos elementos independientes, pero sincronizados: audio e imagen.

El video por componerse de imagen y sonido, dos elementos diferentes e independientes pero susceptibles de ser codificados, almacenados y manipulados. Como se verá a lo largo del capitulo, no existe un formato que presente a los elementos del video como una unidad, sino que se compone de una unión de un formato gráfico y uno de sonido, los cuales he explicado en los anteriores capítulos. El tipo de formato a usar tanto para el sonido como para la imagen, dependerá de las características de cada formato, en algunos casos es posible mezclar los formatos de tal modo se obtengan los mejores beneficios en espacio y en velocidad de reproducción.

Los formatos de video se pueden considerar como una aplicación practica de los formatos de gráficos y de sonido con únicamente unos parámetros de sincronización.

Para quien se interese en trabajar con formatos de video, debe conocer acerca de los formatos gráficos y de sonido que va a usar, adicionalmente de los elementos que el propietario del formato de video adicione, como son , orden de almacenamiento de las imágenes y del sonido, la sincronización de la imagen y el sonido, etc.

3.1 Nociones básicas del video

El video digital se compone de dos elementos básicos: la secuencia animada y la pista sonora.

La secuencia animada es una sucesión de "cuadros". Cada cuadro se constituye de una imagen fija que más o menos se diferencia de la que le precede. Cada imagen es comprimida y desplegada por separado. Las imágenes o cuadros al ser desplegados en secuencia y rápidamente (al menos 24 imágenes por segundo) da la idea de movimiento al espectador. Esto es posible ya que una imagen permanece cierto tiempo en la retina del ojo, y como cada cuadro varia poco del anterior el efecto de animación se asume que es real.

La pista sonora lo constituye un segmento de sonido o una partitura musical, pudiéndose codificar y almacenar con algunas de las técnicas previamente mencionadas en este trabajo (véase formatos de sonido). La secuencia sonora (o musical) es sincronizada con la secuencia animada en progreso; el sonido puede ser almacenado de manera intercalada con la secuencia animada o en un archivo separado.

Hay varios autores que consideran como video únicamente la secuencia animada, si bien ésta visión es correcta para la televisión y el cine. Los estándares que describen al video digital procesado en computadora, deben considerar tanto la secuencia animada como el sonido que le puede acompañar, y se les considera indivisibles. Los estándares sin embargo aceptan el uso de video sin sonido, como una opción.

3.1.1 Captura de video

El proceso de conversión de la señal de video analógico a digital es el siguiente. Cuando la luz reflejada por un objeto (figuras, terrenos, personas, películas, etc.) pasa a través de la lente de la cámara de video se convierte en señal electrónica mediante el dispositivo acoplado por carga (*Charge-Couple Device*, CCD) el cual convierte la imagen en una señal televisiva de acuerdo a los estándares NTSC, PAL/SECAM, HDTV, cuyo dispositivo de almacenamiento típico es la cinta de video.

La señal televisiva de la videocámara o videograbadora es transformada en señal digital por medio de una tarjeta digitalizadora de video; para la codificación de la señal puede usarse los esquemas de color RGB o YCbCr según el tipo de la tarjeta y la calidad de esta. La resolución de la tarjeta varia según la marca y el tipo de tarjeta, típicamente las medidas en pixel del cuadro van desde 32 x 32 hasta 640 x 480. Aunque las nuevas tarjetas alcanzan resoluciones de 1024 x 780 pixels y mucho más resolución en modelos a futuro. Algunos tipos de tarjetas vienen con una cámara de video la cual transfiere directamente la señal analógica de la videocámara en video digital.

El sonido es almacenado al mismo tiempo por un dispositivo de audio (micrófonos, reproductoras de CD's, etc.) en una sección especial junto con la secuencia animada, o en un archivo diferente. Este proceso es comúnmente hecho a una velocidad de 22 Mhz en estéreo, tanto los datos de la secuencia animada como el sonido (o música) son sincronizados, es decir, la música y la secuencia animada son ajustados para que coincidan en su reproducción. El control de la sincronización en la secuencia animada se usa la codificación del SMPTE (*Society of Motion Picture and Televisión Engineers*, Sociedad de fotos en movimiento e ingenieros de televisión) que consiste de una cadena de identificación única en cada cuadro, la cadena es HH:MM:SS:CC (Horas:Minutos:Segundos:Cuadros), es normal que solo se usen hasta los segundos. Esta cadena permite identificar un cuadro en especial para vincularlo con un cuadro de sonido específico,véase la obra de Tay Vaughan (5) para conocer más acerca del video analógico y como es tratado.

3.2 Estándares de video[21]

Ahora bien existen criterios diferentes para la codificación de video en televisión. Estos criterios afectan a los dispositivos de captura y las características del video obtenido en una computadora.

3.2.1 NTSC

Las especificaciones creadas en 1952 por la NTSC (*National Televisión Standars Committe*, Comité Nacional de estándares de televisión), definen que cada cuadro de vidéo se compone de 525 líneas de barrido entrelazado (primero son mostradas las

[21] Vaughan, Tay. Todo El poder de multimedia.

líneas nones y luego las pares) a una velocidad de refresco[22] de 60 Hz. La relación del cuadro es 4:3 que es la relación vertical: horizontal (número de líneas : columnas de la imagen). Es el estándar usado en Norteamérica y Japón.

3.2.2 PAL

El sistema PAL (*Phase Alternate Line*, Línea de fase alterna) usa 625 líneas de barrido entrelazado a 50 Hz de velocidad. Extendido en el Reino Unido, casi toda Europa, Australia y Sudáfrica.

3.2.3 SECAM

Usado en Rusia, Francia entre otros, El sistema SECAM (*Sequential Color And Memory*, sistema secuencial de color y memoria) es una variedad que tiene las mismas características de barrido y velocidad que el PAL aunque con criterios distintos.

3.2.4 HDTV

De acuerdo a las conceptos que actualmente se manejan, el sistema de televisión de alta definición (*High Definition Televisión)* deberá de tener 1200 líneas de resolución, velocidad de refresco de al menos 60 Hz y relación de cuadro 16:9. Los cuadros de video comúnmente alcanzan la resolución de 640 x 480 sin entrelazado a una velocidad de 66.1 Hz, las imágenes más grandes son reducidas o recortadas para ajustarse a estos tamaños.

Otro punto que se debe considerar son los colores, en los estándares hay colores que por sus características son prohibidos. Por ejemplo, un rojo puro de una computadora puede "manchar" a otros colores (teñir sus bordes cono rojo) al ser presentados en una televisión. debido a esto tanto para la captura como para el desplegado de video es necesario usar las normas del estándar de video en el cual se desea reproducir el video.

3.3 Compresión del video

El video por definición ocupa una cantidad datos demasiado grande, consideremos que un cuadro es una imagen, si esta es de, por ejemplo de 640 x 480 a 24 bits de profundidad, normalmente ocupa más de 1 Mb, además a esto hay que añadir otras 23 para obtener una secuencia de un segundo, la cantidad de datos requeridos para una secuencia de unos minutos llegara a ocupar cientos de megabytes, todo esto sin considerar el espacio usado por el sonido. Por ello, los formatos de video se sirven de varios métodos de compresión para sus datos; tomando en cuenta que existen dos partes del video - el sonido y la secuencia animada- estas son comprimidas por separado.

[22] Es el tiempo que tarda en ser redibujada una imágen en la pantalla de video.

La compresión para el audio se basa en los métodos de compresión utilizados en los formatos de sonido (véase formatos de sonido), en la codificación de los datos se añade valores de sincronización con la secuencia animada.

Una situación totalmente diferente es la secuencia animada. Sabiendo que la secuencia animada consiste de una sucesión de imágenes, es susceptible dicha sucesión de ser comprimida en dos formas.

La primer forma es comprimir cada cuadro independientemente a este método se le conoce como *compresión espacial*, es decir, cada cuadro es una imagen estática y como tal es susceptible a ser comprimida por los métodos usados para los formatos gráficos (véase compresión en los formatos gráficos), el método de compresión es sensible a las características de las imágenes que componen la secuencia animada. Por ejemplo el formato QuickTime usa el método Animation para video constituidos por imágenes creadas por computadora, mientras que el método Cinepack es usado para imágenes de calidad fotográfica.

La siguiente forma de comprimir una secuencia animada es la llamada *compresión temporal*, esta se basa en el hecho que un cuadro en general tiende a ser muy similar al cuadro que le precede y al que le sucede. El método simplifica el cuadro indicando únicamente las diferencias que este tiene con su antecesor; para ello es definido un cuadro base, los siguientes cuadros se hacen de las diferencias con el cuadro original. Los criterios para introducir otro cuadro base son variados, sin embargo, si una imagen cambia es tan difirente de antecesora de tal modo que supere un límite, el siguiente cuadro será un cuadro base.

La compresión espacial como la compresión temporal no son excluyentes, sino complementarias, o sea, pueden ser usadas ambas para comprimir la secuencia animada.

Un aspecto que toma relevancia en las técnicas de compresión de la secuencia animada, es el hecho que las técnicas de compresión son asimétricas. En otras palabras, en los métodos de compresión de imágenes fijas y sonido la duración de la etapa de compresión y la etapa de descompresión es casi igual, mientras que en una secuencia animada tal relación es asimétrica de manera notoria. Por ejemplo, el método de compresión Cinepack tarda de 10 a 12 horas para comprimir tan solo diez minutos de video, pero es capaz de reproducirlo a tiempo real sin deformación alguna.

También en los métodos de compresión existe la capacidad de múltiples recompresiones, es decir, un formato es susceptible de ser descomprimido, editado y vuelto a comprimir sin afectar la calidad de la imagen. No todos los métodos pueden realizar esto. Los métodos usados en la secuencia animada son descritos con detalle en cada formato.

3.3.1 Hardware para video

Las aplicaciones actuales de video requieren de procesadores muy veloces que sean capaces de tratar la enorme cantidad de información, gráfica y de sonido, evitando

hasta donde sea posible que el usuario tenga que sufrir de "congelamiento" (paro involuntario) en la reproducción del video, o que la secuencia animada se realice primero y posteriormente el escuche el sonido, el cual debía se reproducido al mismo tiempo que la imagen.

En las estaciones de trabajo con equipo multimedia, como pueden ser las O², Indy's, Indigos de Silicon Graphics; SUN Ultra de SUN Microsystems, etc., son ideales para grabar, editar y reproducir video a tiempo real, por sus sistemas de multiproceso, la gran cantidad de recursos disponibles en almacenamiento. Sin embargo están limitadas a tratar dos o tres videos a la vez, ya que el rendimiento del sistema baja de manera notable.

En el caso de la familia IBM PC la reproducción del video a tiempo real, se ha visto limitado por el bajo rendimiento de los procesadores de la familia 80x86, ya que la secuencia animada absorbe del 60% al 80% del tiempo de procesamiento disponible. Sin embargo, con el lanzamiento del procesador Pentium de Intel se ha logrado mejorar la velocidad de la reproducción, pero sin llegar a un nivel realmente aceptable, ya que son necesarios el 50% de los ciclos del reloj para la secuencia animada. Debido a ello se han propuesto mejoras al hardware, entre ellas la introducción de un juego de instrucciones suplementario en la estructura del procesador para aplicaciones de multimedia conocido como MMX. Todos los fabricantes importantes de procesadores (Intel, AMD, Cyrix, Texas Instruments), harán uso de la tecnología MMX en sus próximas versiones de chips.

Aunque limitadamente es posible reproducir video; los requerimientos mínimos se basan en usar un procesador 80486 DX a 33 Mhz, con 4 Mb en memoria RAM. A condición de no poder ejecutar ningun otro proceso.

3.4 Aplicaciones del video

El video digital está siendo usado en multimedia para el entretenimiento y la enseñanza. Los estándares de video digital serán la base de las transmisiones televisivas del siguiente siglo en todo el mundo, la tan anunciada Televisión de Alta Definición se servirá de los formatos de video para transmitir, no solamente por los canales convencionales, sino también por el Internet. Se planea sustituir los videocasetes grabados en diferentes señales con discos ópticos y CD-ROMs que contengan hasta 4 horas de video, a esta tecnología se le conoce como DVD (Digital Versatile Disk, Disco Versátil Digital). La tecnología telefónica más reciente introduce el concepto de "conferencias multimedia", consistente en enviar además del sonido tradicional, la imagen de la persona con la cual se está comunicando.

3.5 Formatos de video

Los formatos de video están vinculados a los programas de uso del propietario, y en el estado en que están apenas están extendiéndose, la situación es similar a la vivida por la imágenes hace 13 o 15 años, es seguro que veremos surgir dentro de poco muchos formatos de video.

Ya que los formatos de video afectan a varias industrias de comunicaciones (incluso fuera del mundo de la computación), algunas organizaciones de estándares han generado esquemas de codificación de colores, de compresión, entre otros, para crear normas estables a todos los equipos que se vean involucrados con el video. La ISO ha organizado el estándar MPEG, si bien es usado en todas las plataformas y apoyado por los creadores de reproductores de video; debido sus características técnicas y mercadologicas, sin embargo, ha tenido una mediana aceptación, y sirve como referencia de las necesidades que hay en lo que al video se refiere.

3.5.1 MPEG

Propietarios: ***ISO/MPEG group.***
Extensión: ***.mpg/.mp2***

MPEG (*Moving Pictures Experts Group*, Grupo de Expertos de Fotografías en Movimiento) es un grupo de tarabajo que bajo los auspicios de la ISO, genera estándares para video digital y audio compresión. Para el video digital definen una cadena comprimida, lo cual implica el uso de un compresor/descompresor. Los creadores de software al usar los estándares MPEG tienen la libertad de usar la ventajas particulares que ofrece la plataforma en que están trabajando.

Estándares MPEG
MPEG tiene 4 estándares para el tratamiento de video - y el audio asociado-; actualmente está en uso e implementado el estándar MPEG-1. Los estándares creados para el video son:

MPEG-1:"Codificación de imágenes en movimiento y audio asociado para almacenamiento digital de medios (con velocidad) de hasta 1.5 Mb/seg." (Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to about 1.5 Mb/s).

Estado del estándar: Estándar descrito en el documento ISO-11172, completado en octubre de 1992. El estándar (y el documento) se divide en las siguientes partes:

1. System - *Describe la sincronización y el multiplexado de video y audio.* Documento ISO-11172.1
2. Video - *Describe la compresión de señales de video.* Documento ISO-11172.2
3. Audio - *Describe la compresión de señales de audio.* Documento ISO-11172.3
4. Pruebas de conformidad *que describen procedimientos para determinar las características* Documento ISO-11172.4

MPEG-2: "Codificación Genérica de imágenes en movimiento y audio asociado" (Generic Coding of Moving Pictures and Associated Audio)

Estado del estándar: Estándar descrito en el documento ISO-13818, completado en noviembre de 1994. El estándar (y el documento) se divide en las siguientes partes:

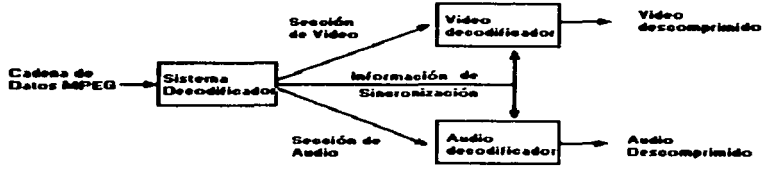
1. System - *Describe la sincronización y el multiplexado de video y audio.* Documento IS-13818.1
2. Video - *Describe la compresión de señales de video.* Documento ISO-13818.2
3. Audio - *Describe la compresión de señales de audio.* Documento ISO-13818.3
4. Pruebas de conformidad *que describen procedimientos para determinar las características.* Documento ISO-13818.4

MPEG-3: No existe, fué fusionado con el estándar MPEG-2

MPEG-4: "Codificación Audio-Visual de baja velocidad" (Very Low Bitrate Audio-Visual Coding)
Estado del estándar: Petición de propuestas en su primer plazo.

Estructura del MPEG

La codificación MPEG inicia con una secuencia de video de baja resolución con una velocidad típica de 352, 240, 30 cuadros/seg., pero con una alta calidad de audio (44.1 Mhz). Las imágenes están en color, en base al esquema YUV el cual es similar al esquema YCbCr del JPEG (es menester mencionar que ambos estándares son creados por la misma organización). La secuencia animada y de audio son separadas y tratadas de manera diferente, el resultado se conjunta en un solo archivo. Para reproducir se realiza el proceso inverso como se puede apreciar en la figura. Primeramente hablare acerca de la secuencia animada.



ESTRUCTURA DEL MPEG

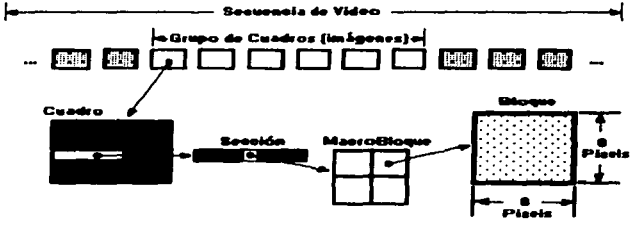
Secuencia Animada

El esquema básico de la codificación y compresión de la secuencia animada consiste en predecir el movimiento (variación) de un cuadro a otro en la secuencia animada (compresión temporal), y entonces son usadas transformadas de coseno discreto (véase JPEG) para organizar y comprimir la redundancia de las imágenes (compresión espacial).

Para comprimir un cuadro (imágen) este es dividido en bloques de 8x8 pixels, la predicción del movimiento está dada en el canal de luminancia Y en bloques de 16x16. En otras palabras, dado el bloque de 16x16 en el cuadro actual que está tratando de codificar, se debe buscar una coincidencia de ese bloque con otro de un

cuadro previo o posterior (hay modos de predicción hacia atrás donde los últimos cuadros son puestos al principio para permitir la interpolación entre cuadros). Los coeficientes DCT (ya sea del bloque, o de las diferencias de este con otro bloque cercano) son calculados y cuantificados de igual modo al JPEG. Así mismo, los resultados del proceso - coeficientes, vectores de movimiento, parámetros de cuantificación, entre otros- son comprimidos usando la codificación VCD, variante del huffman.

El tamaño de los cuadros (en pixels) viene del estándar de video digital CCIR-601. La fuente de entrada de video del formato MPEG usa una variante del estándar llamada SIF , los cuadros de gran tamaño son recortados para que sean divisibles entre 8 o 16, para evitar la existencia de bloques truncos o incompletos. Aún cuando EU y Europa se manejan estándares distintos (EU NTSC, Europa PAL/SECAM), el MPEG usa una velocidad de cuadro uniforme para ambos sistemas.



Esquema de Compresión Espacial Usando Compresión JPEG

Al ser similar la compresión espacial del MPEG al método de compresión JPEG es obvio que cuenta con sus desventajas: la calidad de la compresión y su resultado depende de calidad de la imágenes a comprimir. Así, el video MPEG trabaja bien con video tomado con cámara, mientras que entrega resultados mediocres con secuencias de animación por computadora (imágenes anguladas y muy uniformes).

Predicción de cuadros

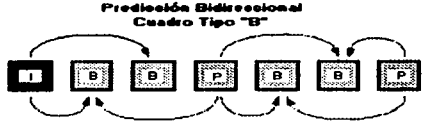
Además de la codificación de cada cuadro, es posible reducir los datos de los cuadros al mínimo. Esto se hace presentado únicamente en los cuadros las diferencias con un cuadro base (compresión temporal).

Hay tres tipos de cuadros. Los cuadros del tipo "I" (Intra frames) son simplemente codificados como una imagen sola, sin necesidad de información previa de los anteriores o posteriores cuadros, también son llamados cuadros base.

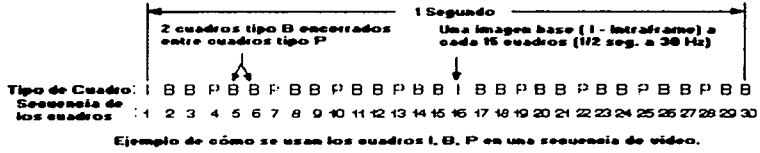
El siguiente tipo de cuadro es el "P", este cuadro es resultado de la predicción de un cuadro I o P anterior, típicamente este cuadro contiene las diferencias con respecto del anterior cuadro, sin embargo, algunos cuadros por la gran cantidad de diferencias terminan por ser similares a los del tipo I.

Por ultimo los cuadros tipo B, estos son bidireccionales, es decir, usan dos cuadros: Uno adelante y uno atrás para predecir al cuadro B, los cuadros usados para la predicción son los cuadros más cercanos del tipo P o I.

El criterio de uso depende exclusivamente de usar un algoritmo hacia adelante o uno hacia atrás, y si la variación entre cuadros es demasiada, simplemente se almacena el cuadro como de tipo I. Los cuadros I son base para el calculo de los siguientes cuadro sean P o B.

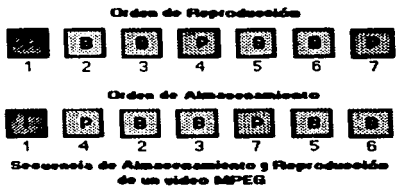


La secuencia de los cuadros descodificados usualmente es:



Ejemplo de cómo se usan los cuadros I, B, P en una secuencia de video.

Donde generalmente hay 15 cuadros intermedios desde un cuadro I a otro I (sin importar el sistema que se trate). Esto es basado en requerimientos de accesos aleatorios, calculándose que se necesita un punto inicial cada 0.4 segundos en promedio. Cuando es descodificado el video se envían los cuadros P, I y se procesan antes de los B, una vez procesados se reproducen en el orden en que deben aparecer en el video.



Secuencia de Almacenamiento y Reproducción de un video MPEG

Compresión del audio

El oído a diferencia del ojo, capta un gran rango dinámico y una mayor resolución de audio, aunque es más lento que el ojo. MPEG recomienda tres métodos de compresión MPEG-1 -el estándar para audio-, es la compresión de señales de audio que usa un esquema de codificación de alto rendimiento. ISO/MPEG describe en su documento ISO 11172-3 (MPEG - Audio) una familia de tres esquemas de compresión, llamados Audio Layer-1, Layer-2, Layer-3 con incremento de complejidad y rendimiento. Los tipos de compresión son jerárquicos, es decir, el uso de un tipo de compresión superior puede leer un archivo comprimido con esquemas inferiores. Por ejemplo, el Layer-3 está capacitado para leer Layer-1 y Layer-2, pero el Layer-2 sólo puede leer al Layer-1 y su propia información.

El estándar sólo define el formato de la cadena comprimida y del descompresor. Esto es para permitir futuras adiciones o implementaciones propuestas al estándar. Todos los niveles Layer usan la misma estructura básica. El esquema de codificación puede ser descrito como "Adaptador de percepción del ruido" (perceptual noise shaping).

El codificador analiza los componentes espectrales de la señal de audio calculando un banco de filtros y aplica un modelo psicoacústico para estimar el nivel de ruido audible almacenando tales componentes. En su etapa de cuantificación y codificación, el codificador trata de asignar el número disponibles de bits de manera que abarque tanto la cantidad necesaria en bits como disimular el ruido.

El descodificador es mucho menos complejo, únicamente sintetiza una señal de audio de los componentes espectrales. Todos los niveles usan el mismo sistema de análisis de banco de filtros (polifase con 32 sub-bandas). El nivel Layer-3 añade una transformación MDCT (Transformación discreta de coseno modificada) para incrementar la resolución de las frecuencias. Todas las capas usan la misma cabecera de información en sus datos, para soportar la estructura jerárquica.

Todos los niveles usan una cadena de datos que contiene partes que son sensibles a los errores (cabecera, localización de los bits, factores de escalamiento, información alterna) y partes que son menos sensibles (datos de los componentes espectrales). Todos los niveles usan 32, 44.1 o 48 kHz de muestreo. Asi mismo, los niveles trabajan con similares velocidades:

Layer-1: de 32 kbps a 448 kbps

Layer-2: de 32 kbps a 384 kbps

Layer-3: de 32 kbps a 320 kbps

Para elegir nivel de compresión de audio, depende las necesidades del sistema y la velocidad de reproducción. Los niveles de compresión tienen un óptimo rendimiento a ciertas velocidades de reproducción. Así el Layer-1 tiene una velocidad de 192 kbps[23] por canal audio y Layer-2 tiene una velocidad de 128 kbps por canal de audio.

Layer-3 soporta 64 kbps e inferiores. Las pruebas realizadas por la CCITT-CCIR a los niveles del MPEG-Audio donde la prueba califica la calidad del sonido que va de 1 (muy ruidoso) a 5 (transparente, sonido exactamente igual al original). Los resultados indican que una menor velocidad produce una mejor calidad en el sonido, por ejemplo, mientras que Layer-2 promedia entre 2.1 a 2.6, Layer-3 alcanza entre 3.6 y 3.8. MPEG permite enmascarar el sonido, es decir usar filtros para suavizar el ruido y aclarar el sonido deseado. El audio puede ser estéreo o monoaural.

3.5.2 QuickTime

Propietario: *Apple Co.*
extensión: *.qt*

QuickTime es un sistema de extensiones (formatos) desarrollado por Apple, que permite a los usuarios ver y editar video, animación, música, texto entre otra información dinámica en los sistemas IBM PC compatibles y Macintosh de multimedia. Este sistema de extensiones incluye estándares de interfaces y formatos.

QuickTime en su formato de multimedia soporta cinco diferentes tipos de datos:

- Video
- Animación
- Sonido
- Música (*pistas de música digitalizada estilo MID I*)
- Texto

QuickTime puede contener cualquiera de esos tipos de datos, o todos; y lo que es más importante, todos los datos sin importar su tipo son sincronizados. Además, QuickTime permite al usuario crear sus propios tipos de datos. Ya que solo atañe a la tesina el video, sonido y música solamente hablare de estos. QuickTime considera como video a la secuencia animada únicamente, la banda sonora o musical que

[23] KiloBytes por segundo. (Kbps)

puede acompañar al video es separada en un formato de sonido o musical respectivamente.

Video

QuickTime para sincronizar la información utiliza el estándar de codificación de tiempo del SMPTE. También puede duplicar el área del video para lograr el efecto de pantalla completa, además soporta el estándar MPEG-1 (aunque es necesario incluir tarjetas de video que soporten el estándar y la compresión MPEG para poder ser usado).

Para almacenar el video, QuickTime utiliza varios estándares de compresión propietarios, muchos de los cuales no permiten la distribución de su tecnología sin contratar una licencia de uso. Varios de estos estándares son similares en estructura al MPEG, es decir comprimen cada cuadro base o de inicio -compresión espacial- y comprimen en relación a otros cuadros - compresión temporal-.

La compresión espacial, como antes mencione, comprime cada cuadro por separado, esto se hace mediante un método de compresión usado en imágenes (véase formatos gráficos).

La compresión temporal es aplicado a secuencias de video. tomando ventaja del hecho de que un cuadro dado tiene algo en común con el cuadro previo. QuickTime define *Key frames* como sus cuadros de base iniciales desde donde se mide la diferencia entre los cuadros.

Se recomienda colocar un cuadro base a cada determinado número de cuadros consecutivos, la distancia recomendada entre cuadros base en video normal (obtenido de una cámara de video) es de 4 a 15, en animación por computadora va de 30 a 150.

Es común en los procesos introducir cuadros base artificialmente, es decir dentro del espacio recomendado en antes de añadir otro cuadro *Key frame* de la imagen puede haber cuadros que al tener una diferencia del 90% o superior se debe poner un nuevo cuadro. La velocidad de aparición de los cuadros base debe ser un submultiplo de la velocidad de los cuadros originales, así, en el estándar NTSC a 30 fps (cuadros por segundo) debe haber un cuadro base cada 30, 15 o 10 cuadros; en PAL de 25 fps a cada 25, 12.5,6.25, y así sucesivamente.

Una situación que generan los estándares de codificación de video, es el hecho que manejan diversos tipos de imagen tanto en resolución, profundidad de colores, como número de pixels que despliegan en un cuadro.

En los estándares de compresión de video, existe una diferencia adicional con los métodos de compresión usados anteriormente, el asunto versa en la relación de compresión / descompresión de la secuencia animada. Mientras en los métodos de compresión de imágenes fijas y sonido el tiempo de descompresión es casi igual al tiempo de compresión en una maquina de características similares, los métodos de compresión de video tardan más en comprimir que en descomprimir (no todos

obviamente), por ello Apple hace recomendaciones y da criterios de decisión para elegir un método de compresión, recuérdese que por ejemplo Cinepack tarda de 10 a 12 hrs en comprimir 10 minutos. La codificación de 10 minutos de video del Cinepack se puede considerar excesivo, pero están rápido en la reproducción que es adecuado para ser usado en aplicaciones de multimedia basados en CD-ROM's 2X (doble velocidad), que es una de las metas del QuickTime como formato de video.

El documento FAQ[24] de QuickTime, simplifica la información acerca de los formatos en tablas. Los campos de la tabla son:

Material Ideal: Tipo de imagen que mejor comprime el método, las imágenes que conforma una secuencia de video puede ser de calidad fotográfica -obtenida de video-, o de imágenes creadas por computadora. Las imágenes por computadora cuentan con grandes áreas de color constante y/o grandes cantidades de ángulos.

Profundidad Soportada: Es el número de bits de profundidad que alcanza una imagen, o sea, la nitidez máxima de la imagen. Algunos métodos solamente soportan determinadas profundidades, y si se especifica, imágenes en color o blanco y negro solamente, en caso contrario soporta ambas.

Tamaño mínimo / máximo: Es el tamaño de cuadro que puede desplegar como mínimo y máximo respectivamente. En caso de que el tamaño de los cuadros del video original no "caben" en el tamaño máximo de cuadro que pueden ser desplegado, el cuadro se recorta o reduce, según se necesite.

Estilo de compresión: Es el tipo de compresión que puede usar el método:.Espacial o temporal. se reserva al usuario como parámetro del formato el usar o no cualquiera de las compresiones.

Tiempo de compresión: Es la relación del tiempo de compresión con el de descompresión. La relación consiste de dos valores (X:Y). Donde X es el tiempo de compresión y, Y es el tiempo de descompresión, medido en segundos.

Soporta límite del tamaño de datos: Varios métodos necesitan separar los datos en bloques de un cierto límite para optimizar el proceso de compresión.

Soporta compresión sin perdida: Indica si el método usa un algoritmo de compresión sin perdida, en caso contrario se le aplica un algoritmo con perdida (Lossy).

Soporta recompresión sin perdida: Indica la posibilidad que un cuadro que ya haya sido comprimido con el algoritmo pueda ser vuelto a comprimir otra vez sin que sus datos se deformen. Esto es de vital importancia para quienes se dedican a editar un videos.

[24] FAQ, Frecuently Answered Questions , Respuestas a las preguntas más frecuentes sobre un tema en especifico.

None (Ninguno)

Material Ideal	Cualquiera
Profundidad soportada	No hay límite
Tamaño mínimo / máximo	1x1 / 320x240
Estilo de compresión	Ninguno
Tiempo de compresión	Simétrico ≡ (1:1)
Soporta límite del tamaño de datos	No
Soporta compresión sin perdida	No se aplica
Soporta recompresión sin perdida	Si

Esta opción de compresión (como lo define Apple) al simplemente almacenar el segmento de video sin ninguna compresión, permitiendo que el material pueda ser editado cuantas veces sea necesario.

Animation

Material Ideal	Animación generada por computadora
Profundidad soportada	Sin límite
Tamaño mínimo / máximo	1x1/320x240
Estilo de compresión	Espacial + Temporal
Tiempo de compresión	Asimétrico ≡(3:1)
Soporta límite del tamaño de datos	No
Soporta compresión sin perdida	Si
Soporta recompresión sin perdida	Si

Como dentro del video (secuencia animada) hay un gran genero de tipos de imágenes de distinta complejidad. Para su almacenamiento digital hay varios métodos que presentan mejores resultados con un tipo de imagen que con otro. En el caso Animation es aplicable a imágenes que cuentan con pocos colores, objetos muy angulados, con grandes áreas de un solo tono; las imágenes creadas por computadora (estadísticas, diagramas, comics, etc.) se ajustan mejor a estas características, el método usa el algoritmo de compresión RLE para almacenar la secuencia animada.

Graphics

Material Ideal	Animación generada por computadora
Profundidad soportada	Color de 8 bits/escala de gris
Tamaño mínimo / máximo	1x1/320x240
Estilo de compresión	Espacial + Temporal
Tiempo de compresión	Asimétrico ≡ (16:1)
Soporta límite del tamaño de datos	No
Soporta compresión sin perdida	Si
Soporta recompresión sin perdida	Si

De manera similar al método Animation, el método Graphics usa el método de compresión RLE, y ambos son muy parecidos en su tamaño de cuadro, entre otras características. Sin embargo al limitarse a una profundidad fija sus archivos son la mitad del tamaño que los producidos usando Animation, pero su velocidad de compresión y de descompresión es mas lenta que éste (en la descompresión casi se

tarda el doble de tiempo), Apple recomienda fuertemente usar este método en videos con tamaño de cuadro pequeño (imágenes pequeñas).

Los siguientes métodos trabajan mejor con secuencias de video "real" (imaénes de calidad fotgráfica) obtenidas de una cámara de video.

Component Video

Material Ideal	Video
Profundidad soportada	24 bits en color
Tamaño mínimo / máximo	2x1/320x240
Estilo de compresión	Espacial
Tiempo de compresión	Simétrico ≅ (1:1)
Soporta límite del tamaño de datos	No
Soporta compresión sin perdida	No
Soporta recompresión sin perdida	No

Comprime los datos sin pérdida y cuenta con la capacidad de compresión / descompresión ilimitada, o sea, no se degrada si se descomprime y recomprime seguidamente. diseñado para la facilitar la edición de los segmentos del video - incluido retoque en los cuadros-; tiene por desventaja la pobre relación de compresión de casi 2:1 (únicamente reduce casi a la mitad los datos del video). Usa la paleta YUV para simplificar su conversión con la señal analógica de las cámaras de video.

Video

Material Ideal	Video
Profundidad soportada	16 bits color
Tamaño mínimo / máximo	4x4/320x240
Estilo de compresión	Espacial + Temporal
Tiempo de compresión	Asimétrico ≅ (7:1)
Soporta límite del tamaño de datos	No
Soporta compresión sin perdida	No
Soporta descompresión sin perdida	Si

Apple Video es un método de compresión de alta velocidad manteniendo un nivel de calidad y compresión aceptable. Soporta compresión espacial y temporal, si es usada la primera, la relación del grado de compresión va de 5:1 a 8:1, y si son usados ambos la compresión alcanza niveles de hasta 25:1. Es usado para transferir secuencias de video entre distintas plataformas - Macintosh y PC por ejemplo-.

Cinepack

Material Ideal	Video
Profundidad soportada	24 bits de color, escala de gris
Tamaño mínimo / máximo	1x1/320x240
Estilo de compresión	Espacial + Temporal
Tiempo de compresión	Asimétrico ≅ (192:1)
Soporta límite del tamaño de datos	Si
Soporta compresión sin perdida	Si
Soporta recompresión sin perdida	No

Es el método de compresión más popular. Diseñado para comprimir secuencias de video de 16 y 24 bits de profundidad. Las medidas de la imagen deben ser un múltiplo de 4 de lo contrario se tiene perdida de eficiencia. Es demasiado lento al comprimir pero muy rápido al descomprimir. Otra condicionante es que la secuencia de video debe ser muy nítida, y si la secuencia animada ya ha sido previamente comprimida con otro método, los resultados serán pobres.

Indeo Video

Material Ideal	Video
Profundidad soportada	24 bits color
Tamaño mínimo / máximo	8x4/320x240 Característica especial que las imágenes deben ser más anchas que altas.
Estilo de compresión	Espacial + Temporal
Tiempo de compresión	Asimétrico ≡ (2:1)
Soporta límite del tamaño de datos	Si
Soporta compresión sin perdida	No
Soporta descompresión sin perdida	No

Dentro de los formatos aceptados por QuickTime el método Indeo Video es similar a Cinepack. El método fue creado por Intel, éste tarda menos de la tercera parte en comprimir que Cinepack, pero usa más tiempo de procesamiento que Cinepack en reproducir, y es inferior a éste en la calidad del video resultante, además trabaja mal con procesadores de bajo rendimiento. Tiene un excelente rendimiento en imágenes "Talking head", es decir, donde el fondo es estático y los objetos en primer plano están en movimiento. Una recomendación adicional se refiere a que el cuadro base debe estar espaciado a una distancia promedio de 4 cuadros.

Radius Studio

Material Ideal	Video
Profundidad soportada	24 bits a color
Tamaño mínimo / máximo	16x8/320x240
Estilo de compresión	Espacial
Tiempo de compresión	Asimétrico ≡ (3.5:1)
Soporta límite del tamaño de datos	si
Soporta compresión sin perdida	no
Soporta descompresión sin perdida	si

Es una compresión propietaria que trabaja con su propio hardware (Video visión Studio y Video visión Telecast), el algoritmo determina si está presente la tarjeta Telecast. De no ser así, el programa simplemente no puede decodificar/codificar el video.

TrueMotion-S

Este algoritmo será incluido en la lista de métodos de compresión en la siguientes versiones, de momento, por el momento sólo está disponible el descompresor, para aquellos que desean usar este método, apple sugiere comprar el paquete a su propietario Horizons Technology.

Sonido
El estándar QuickTime distingue la música y el sonido digitalizado como entes separados, para ello se sirve de los siguientes formatos:

En la música además del uso del estándar MIDI, incluye sus propias extensiones y tablas de instrumentos más allá del estándar conocidos como "Music Track", para ello usa los instrumentos y las librerías de sonido sintetizado de Roland (marca de digitalizadores de audio); con este sistema se puede convertir un sonido de 16 bits en 8 bits, para una reproducción más simple usando un espacio reducido.

En el caso del sonido digitalizado, QuickTime usa los formatos definidos por Apple (AIFF-C), Microsoft (Wav) y el Estándar μ-Law de la CCITT (véase formatos de sonido). Apple introduce el siguiente método de compresión de sonido:

IMA 4:1: QuickTime usa el formato de compresión de audio de la Asociación de Multimedia interactiva (Interactive Multimedia Association, IMA) en formato 4:1, -reduce el espacio ocupado por cuatro muestras de sonido digitalizado a sólo una-. El IMA 4:1 es un esquema de compresión con perdida el cual provee 16 bits por muestra a 44kHz de velocidad de reproducción, al ser introducido este método se planea sustituir al MACE 3:1(véase formato AIFF-C).

El formato de los datos está siendo manejado muy estrictamente por Apple, al grado de no revelar las técnicas que usa a menos que el interesado firme una licencia de uso con ésta firma. Apple misma distribuye el editor para el QuickTime en las plataformas Macintosh system 7 y Ms Windows x. En el caso de Unix coproduce el paquete Xanim, diseñado para los sistemas X-window. Solamente Netscape Communications, Inc, ha obtenido hasta hoy, la licencia de uso para producir un programa Plug-in [25] para su browser Netscape Gold 3.0.

3.5.3 Audio Video Interleaved (AVI)

Propietarios: *Microsoft Co., Intel Co.*
Extensión: *.avi*

Microsoft trabajando conjuntamente con Intel desarrollo un formato de audio y video entrelazado conocido como AVI (Audio Video Interleaved), con este formato se decide unir los segmentos animados y de sonido en un solo archivo sincronizado de manera similar al MPEG.

El AVI funciona primordialmente con imágenes de 24 bits a color. El esquema usado de codificación del color es el YUV, el cual trabaja de igual manera que el formato YCbCr de JPEG, esto es para facilitar el trabajo de conversión con el video

[25] Proviene del término Plug-in, *conectelo y úselo inmediatamente*. Una forma fácil y barata de añadir actualizaciones a los programas de software en forma de "parches" o nuevos bloques del programa que actualizan al mismo con solo una simple instalación menor.

analógico. Como el QuickTime y el MPEG, AVI usa compresión espacial y temporal, su método de compresión está basado en el software Indeo Video de intel, el cual en su versión 3.0 cuenta con un rendimiento cercano al estándar MPEG-1.

Para el almacenamiento de sonido Microsoft recurre al formato WAVE (véase formatos de sonido), para almacenar el sonido en el archivo además de usar los bloques del formato WAV previamente explicados, simplemente se pone énfasis en la sincronización con el video, la sincronización está controlada por el programa compresor.

El formato está diseñado para dar un rendimiento óptimo en los procesadores intel 486 o superiores, no necesariamente se necesitan estos procesadores para ver un video AVI, pero en los sistemas IBM PC es indispensable contar con este tipo de procesador para poder manejar el proceso de lectura del video AVI, manteniendo la velocidad de la secuencia de sonido y video a un nivel mínimamente aceptable.

Existen varios programas que editan y leen el formato AVI. En la plataforma PC está Video for Windows; en Macintosh QuickTime 2.x y en Unix Xanim. Por ser el método de compresión del formato un software y no un algoritmo éste formato es modificado constantemente, esto implica que los programas que funcionen con un viejo software de compresión no podrán leer un video almacenado en una versión actualizada del software.

Microsoft no revela detalles de su formato debido a que el método de compresión es un programa registrado por Intel. Para más detalles es necesario firmar una licencia con Microsoft/Intel.

Indeo Video

El software usado para comprimir archivos AVI es Indeo Video de Intel. Este software está diseñado para trabajar con videos almacenados en CD-ROMs. Indeo define la velocidad de transferencia del cuadro de imagen, sus medidas y la relación con los cuadros de sonido.

Las versiones anteriores de Indeo estaban pensadas para trabajar con un CD-ROM de velocidad simple (1x) con tamaño de cuadro de 240 x 180 pixeles y con una velocidad de 10 cuadros por segundo, opera principalmente en procesadores intel inferiores al modelo 486, la transferencia máxima de datos de video está limitada a 150 Kbps. Con el uso de CD-ROMs de doble velocidad (2x) la transferencia de datos sube a 300 Kbps con un tamaño de cuadro de 320x240 pixeles a 15 cuadros por segundo de alta calidad, pero para reproducir los videos es necesario contar con al menos un procesador 486 SX a 33 Mhz de velocidad.

Para los CD-ROM de 3x a 4x, la transferencia de datos llega de los 400 Kbps a 450 Kbps con el mismo tamaño de datos que en CD-ROM de doble velocidad, pero sólo necesita hasta el 50% de los ciclos de reloj que el cpu tiene disponible, de hecho, Intel recomienda usar un CPU Pentium a 133 Mhz, para tratar sin problemas el video.

Las características de procesador recomendadas por intel aseguran un que es posible leer archivo AVI con una calidad cercana a un archivo MPEG-1 de similares opciones.

El proceso de compresión intercala cuadros de video con cuadros de sonido en un archivo. Intel permite manejar varias relaciones de intercalación; la relación de intercalación recomendada para video obtenido de una cámara de video es de 1:1 (1 cuadro de video por 1 segmento de sonido), en algunas ocasiones puede ser ajustada la relación, por ejemplo, en secuencias de sonido con gran cantidad de silencios.

Al igual que los otros formatos de video, Indeo puede usar cuadros base (key frames), de los cuales parten los cuadros que almacenan la diferencia con los cuadros base o cuadros de predicción (delta frames), los cuadros de predicción pueden ser obtenidos del cuadro base anterior, o del cuadro posterior al cuadro de predicción, cuando se usan ambos el cuadro se le conoce como cuadro de predicción bidireccional.

La cantidad de cuadros base usados incide directamente tanto en la calidad con el espacio del video, es decir, a mayor número de cuadros base será mayor la calidad del video, pero al mismo tiempo aumentará fuertemente el tamaño del archivo; por ello se recomienda elegir el espaciamiento de cuadros que permita tener un archivo reducido con una calidad aceptable.

El software para AVI permite la edición de los cuadros base por separado, los cuadros de predicción son automáticamente corregidos.

Una característica distintiva del AVI es la posibilidad de transparentar objetos dentro de cada cuadro. Es decir, a los objetos principales en la imagen se puede "eliminar" el color circundante (véase transparencia en GIF), de ahí el fondo se puede remplazar con una imagen cualesquiera. A los objetos no transparentados se les conoce como "sprites".

Indeo Video 3.2 es la versión más reciente del método de compresión. Por ser un software que está constantemente bajo modificaciones es común que los programas que lean AVI bajo un compresor anterior no podrán trabajar con nuevas estructuras de la versión 3.2. Para salvar el problema Intel sugiere la constante actualización del software.

Conclusiones

En este trabajo he investigado, compilado y analizado información dispersa acerca de los formatos gráficos, de sonido y video. Aunque los diferentes formatos están pensados para almacenar y manipular determinados tipos de datos, se ha podido apreciar que no existen respuestas únicas para procesar un tipo de información, sino que cada método aprovecha alguna característica de la información a codificar y la explota; es posible ver soluciones similares en distintos tipos de formatos.

Los formatos gráficos cuentan con las técnicas más desarrolladas y refinadas para el almacenamiento, transmisión y depuración de imágenes estáticas. Actualmente el cambio en los formatos gráficos es la introducción de nuevas técnicas de compresión en estudio. Entre ellas podemos mencionar al Wavelet, HARC-C y la compresión Fractal, la cuales son compresiones asimétricas. Hay un interés creciente en crear, manejar y almacenar imágenes tridimensionales de una manera simple y eficaz. Otro aspecto que influirá en los formato gráficos es la capacidad de los diseñadores de los formatos y de aplicaciones de sortear la gran cantidad de derechos de propiedad que hay en las técnicas para almacenar formatos gráficos.

Los formatos de sonido se encuentran en una etapa de desarrollo más temprana. Los esfuerzos están concentrados en homologar el hardware tan dispar que usan las plataformas, y cuando se acepte un hardware estándar de sonido que este dentro de la configuración mínima de un equipo de computación, veremos un desarrollo más vigoroso de los estándares y la aparición de formatos especializados a una tárea definida.

Los formatos de video son los de más reciente aparición en la computación, actualmente se les puede considerar en una etapa similar a los formatos gráficos hace 13 o 15 años, en donde los formatos están unidos a una aplicación específica y orientados a una plataforma específica. Es de esperarse que en los próximos años comience haber una pleyade de formatos nuevos con características que faciliten su uso en tal o cual aplicación, y es probable que siga la evolución marcada por los formatos de video.

Los esfuerzos se están concentrando en producir video a una velocidad suficiente que permita ser transmitido y recibido conforme es recibido (es decir, a tiempo real) en una transmisión en Internet. La idea de Televisión Interactiva será realidad muy probablemente con el uso del Internet y los formatos de video.

El presente trabajo me ha sido grato investigar, analizar y conjuntar algo que todos usamos pero desconocemos realmente como son los formatos gráficos, de sonido y video.

Al investigar este tema descubrí con más detalle que los formatos gráficos, de sonido y video tienen temas relacionados que por nuestra falta de visión no consideramos, como es la compresión, el tratamiento de imágenes y sonido, etc. Decidí hacer una tesina debido a que hay demasiados aspectos a desarrollar, que no habían sido mencionados siquiera anteriormente. El presente trabajo toca ligeramente los tópicos más generales del tema.

El tema de los formatos me ha obligado a utilizar todo le que he visto y aprendido en la carrera de MAC. Materias que un momento dado no consideraba importantes, son ahora de gran importancia para entender los formatos, por ejemplo la materia de programación FORTRAN, el cual consideraba obsoleto, se usa al menos, en el formato gráfico FITS, y está siendo actualizado. Si hay un tema que se identifique completamente con el perfil del estudiante de la carrera de MAC es el tema de los formatos gráficos, de sonido y video.

Toda la información en general está muy dispersa y fragmentada, en algunos casos es muy difícil localizar, o hay intereses económicos y tecnológicos que entorpecen la recopilación de la información.

Mi investigación se basó principalmente en la información disponible en Internet, ello se debió por una parte, a que la información tecnológica en México se menosprecia, y si no es así llega con atraso, que en el caso de la computación siendo un área tan cambiante es demasiado. Por otro lado, algunas fuentes hacen llegar su información por medio de la distribución de documentos vía Internet; esto lo explican ellos, porque es más fácil y barato que hacer los documentos en papel, además se benefician más personas que de otro modo no podrían encontrar la información.

El estudiante de M.A.C. debe estar consciente acerca de estos temas y concentrase en aspectos que en este momento son muy atractivos, los cuales modificarán la visión del mundo de manera similar a la imprenta, radio y televisión. Las técnicas, estructuras y codificaciones usadas por los formatos gráficos, de sonido y video, no requieren de conocimientos más profundos que aquellos que el alumno obtiene a lo largo de su carrera. Adicionalmente son necesarios conocimientos generales acerca del campo al que es aplicado el formato para lograr los máximos beneficios.

Considero que el estudiante de M.A.C. sin importar el perfil que tome encontrará interesante analizar a los formatos desde todos los puntos de vista, ya que abarcan cantidades de aspectos que no son parte de los propósitos de la presente tesina, pero que pueden dar grandes cantidades de investigaciones, ya que como se ve un formato necesita conocimientos de métodos numéricos, análisis de algoritmos, estadística, álgebra lineal, ecuaciones diferenciales, programación FORTRAN, estructuras de datos, teoría de la computación, teleproceso por tan sólo mencionar algunas materias.

Sugiero que se de un fuerte impulso al uso del internet como una fuente de información, ya que constantemente se está ampliando, además dará la oportunidad de entregar trabajos con temas muy actuales y de interés general.

En México, el interés por los formatos ha sido sumamente reducido, así lo dan fe los pocos documentos que al respecto hay de estos temas. Una recomendación que considero importante hacer, es que los formatos gráficos, de sonido y video tienen un caudal de conocimientos que no podemos pasar por alto, además por la importancia que reviste para la sociedad en general, y por su impacto que se comienza a sentir en gran escala.

Otro aspecto que quiero resaltar, es el hecho que por más trivial parezca estudiar los formatos, hay de por medio la obtención de ganancias monetarias, como ocurre con Unisys y el formato GIF, hasta ventajas tácticas y tecnológicas; como ejemplo puedo

señalar el trabajo de Armada de los E.U. en los formatos CALS, la NASA en el formato FITS, Vicar, etc.

Glosario

ANSI
American National Standards Institute. Instituto nacional americano de estándares. Es una institución no gubernamental que desarrolla y publica estándares para uso voluntario en los Estados Unidos.

ASCII
American Standard Code for Information Interchange. Código estándar americano para el intercambio de información. El método más popular de codificación usado para convertir letras, números puntuación y códigos de control en formato digital.

Audio.
Todo aquello que un humano percibe a través del oído. Las frecuencias de audio ocupan el rango de 15Hz a 20,000Hz.

BPS
*Bits Per Second.*Bits por segundo. El número de bits transferidos en sistema de comunicación de datos.

CAD
Computer Aided Design. Diseño asistido por computadora. Se refiere al uso de la computadora en el diseño de un producto.

CAM
Computer Aided Manufacturing. Manufacturación asistida por computadora. Uso de computadoras en los procesos de fabricación de productos.

CD-ROM
Compact Disc-Read Only Memory. Memoria de sólo lectura en disco compacto. Es un medio de almacenamiento basado en un disco óptico de 4.75 pulgadas de radio (desarrollado por NV Philips y Sony Corporation). De características similares a los discos compactos de audio y algunos videodiscos; puede almacenar alrededor de 550Mb.

CCITT
Consultative Committee for International Telephone and Telegraph. Comité consultivo para la telefonía internacional y la telegrafía. Una organización internacional dedicada a crear protocolos que permitan la compatibilidad global para la transmisión de voz, datos y video a través de todos los equipos de telecomunicaciones y cómputo.

CGA
Color Graphics Adapter. Adaptador gráfico de color. Es un estándar para desplegado de video a baja resolución, inventado para la primera IBM-PC. Tiene una resolución de 300 x 200 pixels.

CIE
Commission International de l'Eclairage. Comisión Internacional en iluminación. Desarrolla sistemas de conversión de color.

CYMK
Cyan, Magenta, Yellow, and blacK. Cían, Magenta, Amarillo, y Negro. Los cuatro colores secundarios usados en la impresión de imágenes a color.

Compresión
La traducción de datos a una forma más compacta para almacenamiento y transmisión.

Cuadro
Se refiere a una sola imagen que puede ser desplegada en secuencia con otras imágenes levemente diferentes para crear animaciones.

DAT
Digital Audio Tape. Cinta de Audio Digital. Medio de almacenamiento de audio de alta calidad.

DPI
Dots Per Inch. Puntos por pulgada. Medida usada para calificar la calidad de una impresora.

Digitalización
Proceso de transformar la señal analógica de video en señal digital.

EGA
Enhanced Graphics Adapter. Adaptador gráfico mejorado. Tecnología para desplegado en monitores de las IBM-PC, sustituido por el VGA; la resolución del EGA es de 640 x 350 pixels.

Gama
Variación luminica generada al convertir una imagen de un medio físico a uno digital.

HTML
Hyper Text Markup Language. Lenguaje de marcado de Hipertexto, usado para crea paginas web, es similar en características al VRML.

Hertz
Medida basada en calcular los ciclos por segundo que tiene una onda.

HSB
Hue Saturation Brightness. Con el modelo HSB (HSL), todos los colores son definidos mediante sus niveles de Hue (pigmento), Saturación (cantidad de pigmento) y Brightness (cantidad de blanco incluido).

HDTV
High Definition TV. Televisión de Alta definición. Un estándar propuesto el cual recomienda doblar el numero actual de 525 a 1050 líneas por imagen, obteniendo un a relación 16:9, dando a la televisión una forma mas cercana a la de la pantalla de cine.

ISO
International Standards Organization. Organización internacional de estándares. Grupo Internacional responsable de establecer y manejar varios comités de estándares y grupos de expertos.

JPEG
Joint Photographic Experts Group. Grupo de expertos fotógrafos. Comité de trabajo que está bajo la guía de la ISO, el cual intenta definir un estándar universal de compresión de imágenes almacenadas en computadoras.

MPEG
Motion Pictures Experts Group. Grupo de Expertos de imágenes en movimiento. Comité de la ISO, dedicado a proponer estándares de compresión de video.

LSB
Least Significat Byte. Byte menos significativo. Es el método de almacenamiento de los bits de un byte, en cual se almacena primero la parte inferior (o menos importante) del byte.

MSB
Most Significat Byte. Byte más significativo. Es el método de almacenamiento de los bits de un byte, en cual se almacena primero la parte superior (o más importante) del byte.

Multimedia
Refiere a la conjunción de información de diferente formato (video, audio, gráficos, animación, texto, y demás).

MIDI
Musical Instrument Digital Interface. Interfase digital para instrumentos musicales. Estándar para el control por medio de una computadora de instrumentos musicales.

NTSC
National Television Systems Committee. Comité de sistemas nacionales de televisión. Depende de la Asociación de industrias electrónicas (Electronics Industries Association, EIA) que preparó en diciembre de 1953, el estándar para la producción de televisión comercial en color.

PAL
Phase Alternation Line. Línea de alternación de fase. El estándar Europeo para el video excepto en Francia.

Offset
Nombre que indica la posición de un byte con respecto del inicio de archivo. Indica la posición absoluta del byte.

RGB
Red, Green and Blue. Esquema de color que usando los colores básicos (rojo, verde, azul) codifica una imagen o video.

RLE
Run-Length Encoding. Codificación Run-length. Método de compresión

SECAM
Sequential Couleur A Memoire. Color secuencial con memoria. Estándar televisivo usado en la Europa del este y Francia.

SMPTE
Society of Motion Picture and Television Engineers. Sociedad de ingenieros de fotos en movimiento y televisión. Grupo encargado de normar las transmisiones de televisión.

Video digital
Representa una señal de video compuesta por números binarios que describen un conjunto finito de colores y niveles de luminosidad.

Video analógico
Representa una señal de video compuesta por un infinito numero de variaciones de tonos.

VGA
Video Graphics Array. Arreglo gráfico de video. Estándar de IBM para el desplegado de video. Provee resolución media de texto y gráficas. Tiene una resolución de 640x480 pixels.

URL
Uniform Resources Locator. Localizador uniforme de fuentes. Dirección en internet que nos permite localizar, un archivo o documento.

XGA
Extended Graphics Adapter. Adaptador gráfico mejorado. El mas reciente estándar de IBM que engloba al VGA y soporta altas resoluciones (hasta 1024 pixels por 768 líneas).

YCbCr
Esquema de codificación para fotos realistas en la cual son separados la luminosidad y la cromancia en canales separados, como el ojo humano es menos sensible a las variaciones del color que a las variaciones en la iluminación, YCbCr permite usar la mitad de los bytes en los canales CbCr que los necesarios en el canal Y.

Apéndice I

Tabla I.1: Cabecera del formato DXF

Variable	Descripción
$ACADVER	El número de versión: AC1006 = R10, AC1009=R11 y R12
$ANGBASE	Dirección del Ángulo 0
$ANGDIR	Sentido que siguen los ángulos: 1 = A favor de las manecillas del reloj, 0 = En contra de las manecillas del reloj
$ATTDIA	Atributo de entradas, 1 = Activado, 0 = Desactivado
$ATTMODE	Atributo de visibilidad: 0 = Ninguno, 1 = Normal, 2= Total
$ATTREQ	Atributo de petición durante la edición: 1= Activado, 0= Desactivado
$AUNITS	Formato de las Unidades para los ángulos
$AUPREC	Precisión de la unidades para los ángulos
$AXISMODE	Si no es cero, ejes activos (no usado en R12)
$AXISUNIT	Espaciado en los ejes X y Y (no usado en la revisión 12)
$BLIPMODE	Modo activo si no vale cero
$CECOLOR	Número de Entidad de color: 0 = BYBLOCK, 256 = BYLAYER
$CELTYPE	Entidad de tipo de línea o BYBLOCK , o BYLAYER
$CHAMFERA	Distancia primera del chamfer
$CHAMFERB	Distancia Segunda del chamfer
$CLAYER	Nombre actual de la capa
$COORDS	0 = Despliegue de coordenadas estáticas, 1 = Continua actualización, 2 = Formato "d<a"
$DIMALT	Dimensionamiento en unidades alternativo, se realiza la variable es no cero.
$DIMALTD	Lugares de unidades decimales alternativo
$DIMALTF	Factor de escala de unidades Alternativo
$DIMAPOST	Sufijo de dimensionamiento Alternativo
$DIMASO	1 = Crear dimensionamiento asociativo, 0 = Dibujo de entidades individuales
$DIMASZ	Tamaño de la dimensión de las flechas
$DIMBLK	Nombre del bloque flecha
$DIMBLK1	Primer nombre del bloque flecha
$DIMBLK2	Segundo nombre del bloque flecha
$DIMCEN	Tamaño del centro de las líneas / marcas
$DIMCLRD	Dimensión de la línea de color, El rango es de 0 = BYBLOCK hasta 256 = BYLAYER
$DIMCLRE	Dimensión de la extensión de la línea de color, El rango es de 0 = BYBLOCK hasta 256 = BYLAYER
$DIMCLRT	Dimensión del color del texto, El rango es de 0 = BYBLOCK hasta 256 = BYLAYER
$DIMDLE	Dimensión de la extensión de línea
$DIMDLI	Dimensión del incremento de línea
$DIMEXE	Extensión de la línea
$DIMEXO	Extensión del offset de la línea
$DIMGAP	Dimensión de la línea

Tabla I.1: Cabecera del formato DXF

Variable	Descripcion
$DIMLFAC	Factor de la escala en medición lineal
$DIMLIM	Limites de la dimensión, es generado si no vale cero
$DIMPOST	Sufijo de dimensionamiento general
$DIMRND	Redondeo de valores para la distancia de la dimensión
$DIMSAH	Usar bloques separados de flecha, si no vale cero
$DIMSCALE	Factor de escalamiento de dimensionamiento general
$DIMSE1	Primera línea de extensión suprimida, si vale distinto a cero
$DIMSE2	Segunda línea de extensión suprimida, si vale distinto a cero
$DIMSHO	1 = Recomputa las dimensiones mientras es arrastrado, 0 = arrastrado de la imagen original
$DIMSOXD	Suprime la extensiones externas
$DIMSTYLE	Dimensión del estilo del nombre
$DIMTAD	Texto acerca de la dimensión de línea
$DIMTFAC	Factor de la escala de desplegado
$DIMTIH	Texto en la horizontal
$DIMTIX	Fuerza al texto dentro de las extensiones
$DIMTM	Tolerancia mínima
$DIMTOFL	Si el texto está fuera de las extensiones, fuerza las extensiones de la línea entre extensiones
$DIMTOH	Texto externo horizontal
$DIMTOL	Tolerancias de la dimensión
$DIMTP	Tolerancia superior
$DIMTSZ	Tamaño de dimensionamiento: 0 = No se usa
$DIMTVP	Posición del texto vertical
$DIMTXT	Altura del texto
$DIMZIN	Supresión de dimensiones "feet & inch"
WGCODEPAGE	Código de dibujo de pagina. Se hace cuando se crea un nuevo dibujo es creado.
$DRAGMODE	0 = Off, 1 = On, 2 = Auto
$ELEVATION	Conjunto actual de elevación
$EXTMAX	Valores X, Y, y Z del dibujo de la esquina superior derecha
$EXTMIN	Valores X, Y, y Z del dibujo de la esquina inferior izquierda
$FILLETRAD	Radio del Fillet
$FILLMODE	Modo de llenado
$HANDLING	Manejo activado si es mayor que cero
$HANDSEED	Siguiente manejo disponible
$INSBASE	Conjunto de inserción base definido por el comando BASE
$LIMCHECK	No cero si la marca del limite está activado
$LIMMAX	Limites X, Y de la esquina superior derecha
$LIMMIN	Limites X, Y de la esquina inferior izquierda
$LTSCALE	Escala global del tipo de línea
$LUNITS	Formato de las unidades de las coordenadas y distancias
$LUPREC	Unidades de precisión para coordenadas y distancias
$MAXACTVP	Número de vistas máxima a ser generada
$MENU	Nombre del menú del archivo
$MIRRTEXT	Texto espejo
$ORTHOMODE	Modo ortonormal si no vale cero
$OSMODE	Modos de objetos

Tabla I.1: Cabecera del formato DXF

Variable	Descripción
$PDMODE	Modo de desplegado del punto
$PDSIZE	Tamaño del punto de despliegue
$PELEVATION	Actual elevación del espacio del papel
$PEXTMAX	Extensiones máximas X, Y, y Z para el espacio del papel
$PEXTMIN	Extensiones mínimas X, Y, y Z para el espacio del papel
$PLIMCHECK	Checa los Límites del papel
$PLIMMAX	Límites máximos X y Y del espacio del papel
$PLIMMIN	Límites mínimos X y Y del espacio del papel
$PLINEGEN	Convenciones de generación del tipo de patrón de línea de los vértices de una polilínea 2D. 1 = Un patrón continuo es generado sobre los vértices de la polilínea. 0 = Cada segmento de la polilínea inicia y termina con una raya.
$PLINEWID	Ancho de polilínea por defecto
$PSLTSCALE	Controles del tipo de línea del papel 1 = Escala no especial 0 = Convenciones de escalamiento
$PUCSNAME	Nombre actual del espacio del papel en UCS
$PUCSORG	Origen actual del espacio del papel en UCS
$PUCSXDIR	Eje X actual del espacio del papel en UCS
$PUCSYDIR	Eje Y actual del espacio del papel en UCS
$QTEXTMODE	Modo texto rápido
$REGENMODE	Modo REGENAUTO si vale distinto a cero.
$SHADEDGE	0 = Caras sombreadas, bordes no remarcados 1 = Caras sombreadas, bordes remarcados 2 = Caras no sombreadas, bordes en entidad de color 3 = Caras en entidad de color, bordes en negro
$SHADEDIF	Porcentaje de luz ambiental / difusa, rango de 1 a 100, por defecto 70
$SKETCHINC	Incremento en el registro del croquis
$SKPOLY	0 = Líneas de croquis 1 = Croquis de polilíneas
$SPLFRAME	Desplegado del control del polígono, 1= on, 0 = off
$SPLINESEGS	Número de segmentos de línea por spline
$SPLINETYPE	Tipo de curva para PEDIT Spline
$SURFTAB1	Número de medidas de tabulaciones en al primera dirección
$SURFTAB2	Número de medidas de tabulaciones en al primera dirección
$SURFTYPE	Tipo de superficie para PEDIT Smooth
$SURFU	Densidad de la superficie (para PEDIT Smooth) en la dirección M
$SURFV	Densidad de la superficie (para PEDIT Smooth) en la dirección N
$TDCREATE	Fecha / hora de creación del dibujo
$TDINDWG	Tiempo acumulativo para este dibujo
$TDUPDATE	Fecha / hora de la ultima modificación del dibujo
$TDUSRTIMER	Tiempo transcurrido por el usuario
$TEXTSIZE	Altura del texto base
$TEXTSTYLE	Nombre del tipo de texto
$THICKNESS	Actual espesor puesto por el comando ELEV
$TILEMODE	1 = Modo de compatibilidad para previa versión 0 = De otro modo
$TRACEWID	Ancho de trazo por defecto
$UCSNAME	Nombre del actual UCS

Tabla I.1: Cabecera del formato DXF

Variable	Descripción
$UCSNAME	Nombre del actual UCS
$UCSORG	Origen del actual UCS (en WCS)
$UCSXDIR	Dirección del actual UCS en ejes X (en coordenadas mundiales)
$UCSYDIR	Dirección del actual UCS en ejes Y (en coordenadas mundiales)
$UNITMODE	Bit menos significativo = Despliega fracciones, en pulgadas y pies
$USERI15	Cinco variables enteras diseñadas para ser usados por terceros
$USERR15	Cinco variables enteras diseñadas para ser usados por terceros
$USRTIMER	0 = Timer apagado, 1 = Timer encendido
$VISRETAIN	0 = No retiene los valores Xref, 1 = Retiene los valores Xref
$WORLDVIEW	1 = Convierte UCS a WCS durante Dview / Vpoint 0 = No cambia

Apéndice II

Las siguientes tablas son distribuidas de manera gratuita por la MMA, bajo condición de no sufrir modificación en su estructura.

Tabla II.1: Códigos de las notas

Numero de octava	C	C#	D	D#	E	F	F#	G	G#	A	A#	B
-1	0	1	2	3	4	5	6	7	8	9	10	11
0	12	13	14	15	16	17	18	19	20	21	22	23
1	24	25	26	27	28	29	30	31	32	33	34	35
2	36	37	38	39	40	41	42	43	44	45	46	47
3	48	49	50	51	52	53	54	55	56	57	58	59
4	60	61	62	63	64	65	66	67	68	69	70	71
5	72	73	74	75	76	77	78	79	80	81	82	83
6	84	85	86	87	88	89	90	91	92	93	94	95
7	96	97	98	99	100	101	102	103	104	105	106	107
8	108	109	110	111	112	113	114	115	116	117	118	119
9	120	121	122	123	124	125	126	127				

Nota: los valores están en decimal.

Tabla II.2: Sumario de etiquetas MIDI y Datos de bytes

Etiqueta				Bytes de datos	
Valor del primer Byte			Función	Segundo Byte	Tercer Byte
Binario	Hex	Decimal			
10000000	80	128	Canal 1 NotaApagada	Número de la nota	Velocidad de la nota
10000001	81	129	Canal 2 "	(0-127)	(0-127)
10000010	82	130	Canal 3 "	"	"
10000011	83	131	Canal 4 "	"	"
10000100	84	132	Canal 5 "	"	"
10000101	85	133	Canal 6 "	"	"
10000110	86	134	Canal 7 "	"	"
10000111	87	135	Canal 8 "	"	"
10001000	88	136	Canal 9 "	"	"
10001001	89	137	Canal 10 "	"	"
10001010	8A	138	Canal 11 "	"	"
10001011	8B	139	Canal 12 "	"	"
10001100	8C	140	Canal 13 "	"	"
10001101	8D	141	Canal 14 "	"	"
10001110	8E	142	Canal 15 "	"	"
10001111	8F	143	Canal 16 "	"	"
10010000	90	144	Canal 1 Nota Activa	"	"
10010001	91	145	Canal 2 "	"	"
10010010	92	146	Canal 3 "	"	"
10001100	8C	140	Canal 13 "	"	"
10010000	90	144	Canal 1 Nota Activa	"	"
10010001	91	145	Canal 2 "	"	"
10010010	92	146	Canal 3 "	"	"

Tabla 11.2: Sumario de etiquetas MIDI y Datos de bytes (Continuación)

Etiqueta				Bytes de datos	
Valor del primer Byte			Función	Segundo Byte	Tercer Byte
Binario	Hex	Decimal			
10001100	8C	140	Canal 13 "	"	"
10001101	8D	141	Canal 14 "	"	"
10001110	8E	142	Canal 15 "	"	"
10001111	8F	143	Canal 16 "	"	"
10010000	90	144	Canal 1 Nota Activa	"	"
10010001	91	145	Canal 2 "	"	"
10010010	92	146	Canal 3 "	"	"
10011010	9A	154	Canal 11 "	"	"
10011011	9B	155	Canal 12 "	"	"
10011100	9C	156	Canal 13 "	"	"
10011101	9D	157	Canal 14 "	"	"
10011110	9E	158	Canal 15 "	"	"
10011111	9F	159	Canal 16 "	"	"
10100000	A0	160	Canal 1 Tecla Polifónica	"	Cantidad de
10100001	A1	161	Canal 2 "	"	Presión
10100010	A2	162	Canal 3 "	"	(0-127)
10100011	A3	163	Canal 4 "	"	"
10100100	A4	164	Canal 5 "	"	"
10100101	A5	165	Canal 6 "	"	"
10100110	A6	166	Canal 7 "	"	"
10100111	A7	167	Canal 8 "	"	"
10101000	A8	168	Canal 9 "	"	"
10101001	A9	169	Canal 10 "	"	"
10101010	AA	170	Canal 11 "	"	"
10101011	AB	171	Canal 12 "	"	"
10101100	AC	172	Canal 13 "	"	"
10101101	AD	173	Canal 14 "	"	"
10101110	AE	174	Canal 15 "	"	"
10101111	AF	175	Canal 16 "	"	"
10110000	B0	176	Canal 1 Control de	Véase	Véase
10110001	B1	177	Canal 2 Cambio	la tabla 2.7	la tabla 2.7
10110010	B2	178	Canal 3 de modo	"	"
10110011	B3	179	Canal 4 "	"	"
10110100	B4	180	Canal 5 "	"	"
10110101	B5	181	Canal 6 "	"	"
10110110	B6	182	Canal 7 "	"	"
10110111	B7	183	Canal 8 "	"	"
10111000	B8	184	Canal 9 "	"	"
10111001	B9	185	Canal 10 "	"	"
10111010	BA	186	Canal 11 "	"	"
10111011	BB	187	Canal 12 "	"	"
10111100	BC	188	Canal 13 "	"	"
10111101	BD	189	Canal 14 "	"	"
10111110	BE	190	Canal 15 "	"	"
10111111	BF	191	Canal 16 "	"	"

Tabla 11.2: Sumario de etiquetas MIDI y Datos de bytes (Continuación)

Etiqueta					Bytes de datos	
Valor del primer Byte			Función		Segundo Byte	Tercer Byte
Binario	Hex	Decimal				
11000000	C0	192	Canal 1	Cambio de	Número de	Sin datos
11000001	C1	193	Canal 2	Programa	Programa (0-127)	"
11000010	C2	194	Canal 3	"	"	"
11000011	C3	195	Canal 4	"	"	"
11000100	C4	196	Canal 5	"	"	"
11000101	C5	197	Canal 6	"	"	"
11000110	C6	198	Canal 7	"	"	"
11000111	C7	199	Canal 8	"	"	"
11001000	C8	200	Canal 9	"	"	"
11001001	C9	201	Canal 10	"	"	"
11001010	CA	202	Canal 11	"	"	"
11001011	CB	203	Canal 12	"	"	"
11001100	CC	204	Canal 13	"	"	"
11001101	CD	205	Canal 14	"	"	"
11001110	CE	206	Canal 15	"	"	"
11001111	CF	207	Canal 16	"	"	"
11010000	D0	208	Canal 1	Canal	Cantidad de	"
11010001	D1	209	Canal 2	Presión	Presión	"
11010010	D2	210	Canal 3	"	(0-127)	"
11010011	D3	211	Canal 4	"	"	"
11010100	D4	212	Canal 5	"	"	"
11010101	D5	213	Canal 6	"	"	"
11010110	D6	214	Canal 7	"	"	"
11010111	D7	215	Canal 8	"	"	"
11011000	D8	216	Canal 9	"	"	"
11011001	D9	217	Canal 10	"	"	"
11011010	DA	218	Canal 11	"	"	"
11011011	DB	219	Canal 12	"	"	"
11011100	DC	220	Canal 13	"	"	"
11011101	DD	221	Canal 14	"	"	"
11011110	DE	222	Canal 15	"	"	"
11011111	DF	223	Canal 16	"	"	"
11100000	E0	224	Canal 1	Control	Control de giro	Control de giro
11100001	E1	225	Canal 2	de giro	de tono en	de tono en
11100010	E2	226	Canal 3	tono	LSB	MSB
11100011	E3	227	Canal 4	"	(0-127)	(0-127)
11100100	E4	228	Canal 5	"	"	"
11100101	E5	229	Canal 6	"	"	"
11100110	E6	230	Canal 7	"	"	"
11100111	E7	231	Canal 8	"	"	"
11101000	E8	232	Canal 9	"	"	"
11101001	E9	233	Canal 10	"	"	"
11101010	EA	234	Canal 11	"	"	"
11101011	EB	235	Canal 12	"	"	"
11101100	EC	236	Canal 13	"	"	"

Tabla 11.2: Sumario de etiquetas MIDI y Datos de bytes (Cont.)

Etiqueta				Bytes de datos	
Valor del primer Byte			Funcion	Segundo	Tercer
Binario	Hex	Decimal		Byte	Byte
11101101	ED	237	Canal 14 "	"	"
11101110	EE	238	Canal 15 "	"	"
11101111	EF	239	Canal 16 "	"	"
11110000	F0	240	Sistema Exclusivo	**	**
11110001	F1	241	Código de Tiempo	mas detalle en la especificación	mas detalle en la especificación
11110010	F2	242	Puntero de Posición del tema	LSB	MSB
11110011	F3	243	Selección del tema (#)	(0-127)	SIN VALOR
11110100	F4	244	Indefinido	?	?
11110101	F5	245	Indefinido	?	?
11110110	F6	246	Petición de tono	SIN VALOR	SIN VALOR
11110111	F7	247	Fin del sistema (EOX)	"	"
11111000	F8	248	Reloj de compás	"	"
11111001	F9	249	Indefinido	"	"
11111010	FA	250	Inicio	"	"
11111011	FB	251	Continuación	"	"
11111100	FC	252	Paro	"	"
11111101	FD	253	Indefinido	"	"
11111110	FE	254	Detección de Actividad	"	"
11111111	FF	255	Reinicio del sistema	"	"

Tabla 11.3 Byte de estado del 176 al 191

Valor del segundo Byte			Funcion	Valor del tercer byte	
Binario	Hex	Dec		Valor	Orden de byte
00000000	00	0	Banco de selección	0-127	MSB
00000001	01	1	Ajuste de modulación	0-127	MSB
00000010	02	2	Control de soplido	0-127	MSB
00000011	03	3	Indefinido	0-127	MSB
00000100	04	4	Control del pedal	0-127	MSB
00000101	05	5	Tiempo del Portamento	0-127	MSB
00000110	06	6	Entrada de Datos	0-127	MSB
00000111	07	7	Canal de Volumen	0-127	MSB
00001000	08	8	Balance	0-127	MSB
00001001	09	9	Indefinido	0-127	MSB
00001010	0A	10	Pan	0-127	MSB
00001011	0B	11	Controlador de expresión	0-127	MSB
00001100	0C	12	Control de efectos 1	0-127	MSB
00001101	0D	13	Control de efectos 2	0-127	MSB
00001110	0E	14	Indefinido	0-127	MSB
00001111	0F	15	Indefinido	0-127	MSB
00010000	10	16	Controlador de Propósito Gen. #1	0-127	MSB
00010001	11	17	Controlador de Propósito Gen. #2	0-127	MSB
00010010	12	18	Controlador de Propósito Gen. #3	0-127	MSB
00010011	13	19	Controlador de Propósito Gen. #4	0-127	MSB

Tabla 11.3 Byte de estado del 176 al 191

Valor del segundo Byte			Función	Valor del tercer byte	
Binario	Hex	Dec		Valor	Orden de byte
00010100	14	20	Indefinido	0-127	MSB
00010101	15	21	Indefinido	0-127	MSB
00010110	16	22	Indefinido	0-127	MSB
00010111	17	23	Indefinido	0-127	MSB
00011000	18	24	Indefinido	0-127	MSB
00011001	19	25	Indefinido	0-127	MSB
00011010	1A	26	Indefinido	0-127	MSB
00011011	1B	27	Indefinido	0-127	MSB
00011100	1C	28	Indefinido	0-127	MSB
00011101	1D	29	Indefinido	0-127	MSB
00011110	1E	30	Indefinido	0-127	MSB
00011111	1F	31	Indefinido	0-127	MSB
00100000	20	32	Banco de Selección	0-127	LSB
00100001	21	33	Ajuste de modulación	0-127	LSB
00100010	22	34	Control de soplido	0-127	LSB
00100011	23	35	Indefinido	0-127	LSB
00100100	24	36	Control del pedal	0-127	LSB
00100101	25	37	Tiempo del Portamento	0-127	LSB
00100110	26	38	Entrada de Datos	0-127	LSB
00100111	27	39	Canal de Volumen	0-127	LSB
00101000	28	40	Balance	0-127	LSB
00101001	29	41	Indefinido	0-127	LSB
00101010	2A	42	Pan	0-127	LSB
00101011	2B	43	Controlador de expresión	0-127	LSB
00101100	2C	44	Control de efectos 1	0-127	LSB
00101101	2D	45	Control de efectos 2	0-127	LSB
00101110	2E	46	Indefinido	0-127	LSB
00101111	2F	47	Indefinido	0-127	LSB
00110000	30	48	Controlador de Propósito Gen. #1	0-127	LSB
00110001	31	49	Controlador de Propósito Gen. #2	0-127	LSB
00110010	32	50	Controlador de Propósito Gen. #3	0-127	LSB
00110011	33	51	Controlador de Propósito Gen. #4	0-127	LSB
00110100	34	52	Indefinido	0-127	LSB
00110101	35	53	Indefinido	0-127	LSB
00110110	36	54	Indefinido	0-127	LSB
00110111	37	55	Indefinido	0-127	LSB
00111000	38	56	Indefinido	0-127	LSB
00111001	39	57	Indefinido	0-127	LSB
00111010	3A	58	Indefinido	0-127	LSB
00111011	3B	59	Indefinido	0-127	LSB
00111100	3C	60	Indefinido	0-127	LSB
00111101	3D	61	Indefinido	0-127	LSB
00111110	3E	62	Indefinido	0-127	LSB
00111111	3F	63	Indefinido	0-127	LSB
01000000	40	64	Pedal de sordina	<63=off/ 64=on	- -
01000001	41	65	Portamento	<63=off/64=on	- -
01000010	42	66	Sostenimiento	<63=off/64=on	- -

Tabla 11.3 Byte de estado del 176 al 191 (Continuación)

Valor del segundo Byte			Función	Valor del tercer byte	
Binario	Hex	Dec		Valor	Orden de byte
01000011	43	67	Pedal Suave	<63=off/64=on	- -
01000100	44	68	Pedal de pie	<63=off/64=on	- -
01000101	45	69	Sostenimiento 2	<63=off/64=on	- -
01000110	46	70	Control de Sonido 1	0-127	LSB
01000111	47	71	Control de Sonido 2	0-127	LSB
01001000	48	72	Control de Sonido 3	0-127	LSB
01001001	49	73	Control de Sonido 4	0-127	LSB
01001010	4A	74	Control de Sonido 5	0-127	LSB
01001011	4B	75	Control de Sonido 6	0-127	LSB
01001100	4C	76	Control de Sonido 7	0-127	LSB
01001101	4D	77	Control de Sonido 8	0-127	LSB
01001110	4E	78	Control de Sonido 9	0-127	LSB
01001111	4F	79	Control de Sonido 10	0-127	LSB
01010000	50	80	Controlador de Propósito Gen. #1	0-127	LSB
01010001	51	81	Controlador de Propósito Gen. #2	0-127	LSB
01010010	52	82	Controlador de Propósito Gen. #3	0-127	LSB
01010011	53	83	Controlador de Propósito Gen. #4	0-127	LSB
01010100	54	84	Control de Portamento	0-127	Nota fuente
01010101	55	85	Indefinido	0-127	LSB
01010110	56	86	Indefinido	0-127	LSB
01010111	57	87	Indefinido	0-127	LSB
01011000	58	88	Indefinido	0-127	LSB
01011001	59	89	Indefinido	0-127	LSB
01011010	5A	90	Indefinido	0-127	LSB
01011011	5B	91	Profundidad de los efectos 1	0-127	LSB
01011100	5C	92	Profundidad de los efectos 2	0-127	LSB
01011101	5D	93	Profundidad de los efectos 3	0-127	LSB
01011110	5E	94	Profundidad de los efectos 4	0-127	LSB
01011111	5F	95	Profundidad de los efectos 5	0-127	LSB
01100000	60	96	Entrada de Datos +1	N/A	
01100001	61	97	Entrada de Datos -1	N/A	
01100010	62	98	Número de parámetro no Registrado en LSB	0-127	LSB
01100011	63	99	Número de parámetro no Registrado en LSB	0-127	MSB
01100100	64	100	Número de parámetro Registrado en LSB	0-127	LSB
01100101	65	101	Número de parámetro Registrado en MSB	0-127	MSB
01100110	66	102	Indefinido	?	
01100111	67	103	Indefinido	?	
01101000	68	104	Indefinido	?	
01101001	69	105	Indefinido	?	
01101010	6A	106	Indefinido	?	
01101011	6B	107	Indefinido	?	
01101100	6C	108	Indefinido	?	
01101101	6D	109	Indefinido	?	

Tabla 11.3 Byte de estado del 176 al 191 (Cont.)

Valor del segundo Byte			Función	Valor del tercer byte	
Binario	Hex	Dec		Valor	Orden de byte
01101110	6E	110	Indefinido	?	
01101111	6F	111	Indefinido	?	
01110000	70	112	Indefinido	?	
01110001	71	113	Indefinido	?	
01110010	72	114	Indefinido	?	
01110011	73	115	Indefinido	?	
01110100	74	116	Indefinido	?	
01110101	75	117	Indefinido	?	
01110110	76	118	Indefinido	?	
01110111	77	119	Indefinido	?	
01111000	78	120	Todo el sonido apagado	0	
01111001	79	121	Reinicia todos los controladores	0	
01111010	7A	122	Control local	0 on /127 off	
01111011	7B	123	Todas notas apagadas	0	
01111100	7C	124	Modo Omni apagado	0	
01111101	7D	125	Modo Omni activo	0	
01111110	7E	126	Modo Poly apagado	**	
01111111	7F	127	Modo Poly encendido	0	

Tabla II.4: Sumario de etiquetas MIDI y Datos de bytes

Etiqueta				Bytes de datos	
Valor del primer Byte			Función	Segundo	Tercer
Binario	Hex	Decimal		Byte	Byte
10000000	80	128	Canal 1 Nota Apagada	Número de la nota	Velocidad de la nota
10000001	81	129	Canal 2 "	(0-127)	(0-127)
10000010	82	130	Canal 3 "	"	"
10000011	83	131	Canal 4 "	"	"
10000100	84	132	Canal 5 "	"	"
10000101	85	133	Canal 6 "	"	"
10000110	86	134	Canal 7 "	"	"
10000111	87	135	Canal 8 "	"	"
10001000	88	136	Canal 9 "	"	"
10001001	89	137	Canal 10 "	"	"
10001010	8A	138	Canal 11 "	"	"
10001011	8B	139	Canal 12 "	"	"
10001100	8C	140	Canal 13 "	"	"
10001101	8D	141	Canal 14 "	"	"
10001110	8E	142	Canal 15 "	"	"
10001111	8F	143	Canal 16 "	"	"
10010000	90	144	Canal 1 Nota Activa	"	"
10010001	91	145	Canal 2 "	"	"
10010010	92	146	Canal 3 "	"	"
10010011	93	147	Canal 4 "	"	"
10010100	94	148	Canal 5 "	"	"
10010101	95	149	Canal 6 "	"	"
10010110	96	150	Canal 7 "	"	"
10010111	97	151	Canal 8 "	"	"

Tabla II.4: Sumario de etiquetas MIDI y Datos de bytes (Continuación)

Etiqueta				Bytes de datos	
Valor del primer Byte			Función	Segundo	Tercer
Binario	Hex	Decimal		Byte	Byte
10011000	98	152	Canal 9 "	"	"
10011001	99	153	Canal 10 "	"	"
10011010	9A	154	Canal 11 "	"	"
10011011	9B	155	Canal 12 "	"	"
10011100	9C	156	Canal 13 "	"	"
10011101	9D	157	Canal 14 "	"	"
10011110	9E	158	Canal 15 "	"	"
10011111	9F	159	Canal 16 "	"	"
10100000	A0	160	Canal 1 Tecla Polifónica	"	Cantidad de
10100001	A1	161	Canal 2 "	"	Presión
10100010	A2	162	Canal 3 "	"	(0-127)
10100011	A3	163	Canal 4 "	"	"
10100100	A4	164	Canal 5 "	"	"
10100101	A5	165	Canal 6 "	"	"
10100110	A6	166	Canal 7 "	"	"
10100111	A7	167	Canal 8 "	"	"
10101000	A8	168	Canal 9 "	"	"
10101001	A9	169	Canal 10 "	"	"
10101010	AA	170	Canal 11 "	"	"
10101011	AB	171	Canal 12 "	"	"
10101100	AC	172	Canal 13 "	"	"
10101101	AD	173	Canal 14 "	"	"
10101110	AE	174	Canal 15 "	"	"
10101111	AF	175	Canal 16 "	"	"
10110000	B0	176	Canal 1 Control de	Véase	Véase
10110001	B1	177	Canal 2 Cambio	la tabla 2.7	la tabla 2.7
10110010	B2	178	Canal 3 de modo	"	"
10110011	B3	179	Canal 4 "	"	"
10110100	B4	180	Canal 5 "	"	"
10110101	B5	181	Canal 6 "	"	"
10110110	B6	182	Canal 7 "	"	"
10110111	B7	183	Canal 8 "	"	"
10111000	B8	184	Canal 9 "	"	"
10111001	B9	185	Canal 10 "	"	"
10111010	BA	186	Canal 11 "	"	"
10111011	BB	187	Canal 12 "	"	"
10111100	BC	188	Canal 13 "	"	"
10111101	BD	189	Canal 14 "	"	"
10111110	BE	190	Canal 15 "	"	"
10111111	BF	191	Canal 16 "	"	"
11000000	C0	192	Canal 1 Cambio de	Número de	Sin datos
11000001	C1	193	Canal 2 Programa	Programa (0-127)	"
11000010	C2	194	Canal 3 "	"	"
11000011	C3	195	Canal 4 "	"	"
11000100	C4	196	Canal 5 "	"	"
11000101	C5	197	Canal 6 "	"	"
11000110	C6	198	Canal 7 "	"	"

Tabla II.4: Sumario de etiquetas MIDI y Datos de bytes(Cont.)

Etiqueta					Bytes de datos	
Valor del primer Byte			Función		Segundo Byte	Tercer Byte
Binario	Hex	Decimal				
11000111	C7	199	Canal 8	"	"	"
11001000	C8	200	Canal 9	"	"	"
11001001	C9	201	Canal 10	"	"	"
11001010	CA	202	Canal 11	"	"	"
11001011	CB	203	Canal 12	"	"	"
11001100	CC	204	Canal 13	"	"	"
11001101	CD	205	Canal 14	"	"	"
11001110	CE	206	Canal 15	"	"	"
11001111	CF	207	Canal 16	"	"	"
11010000	D0	208	Canal 1	**Canal**	**Cantidad de**	"
11010001	D1	209	Canal 2	**Presión**	**Presión**	"
11010010	D2	210	Canal 3	"	**(0-127)**	"
11010011	D3	211	Canal 4	"	"	"
11010100	D4	212	Canal 5	"	"	"
11010101	D5	213	Canal 6	"	"	"
11010110	D6	214	Canal 7	"	"	"
11010111	D7	215	Canal 8	"	"	"
11011000	D8	216	Canal 9	"	"	"
11011001	D9	217	Canal 10	"	"	"
11011010	DA	218	Canal 11	"	"	"
11011011	DB	219	Canal 12	"	"	"
11011100	DC	220	Canal 13	"	"	"
11011101	DD	221	Canal 14	"	"	"
11011110	DE	222	Canal 15	"	"	"
11011111	DF	223	Canal 16	"	"	"
11100000	E0	224	Canal 1	**Control**	**Control de giro**	**Control de giro**
11100001	E1	225	Canal 2	**de giro**	**de tono en**	**de tono en**
11100010	E2	226	Canal 3	**tono**	**LSB**	**MSB**
11100011	E3	227	Canal 4	"	**(0-127)**	**(0-127)**
11100100	E4	228	Canal 5	"	"	"
11100101	E5	229	Canal 6	"	"	"
11100110	E6	230	Canal 7	"	"	"
11100111	E7	231	Canal 8	"	"	"
11101000	E8	232	Canal 9	"	"	"
11101001	E9	233	Canal 10	"	"	"
11101010	EA	234	Canal 11	"	"	"
11101011	EB	235	Canal 12	"	"	"
11101100	EC	236	Canal 13	"	"	"
11101101	ED	237	Canal 14	"	"	"
11101110	EE	238	Canal 15	"	"	"
11101111	EF	239	Canal 16	"	"	"
11110000	F0	240	Sistema Exclusivo		**	**
11110001	F1	241	Código de Tiempo		**mas detalle en la especificación**	**mas detalle en la especificación**
11110010	F2	242	Puntero de Posición del tema		**LSB**	**MSB**

Tabla II.4: Sumario de etiquetas MIDI y Datos de bytes (Cont.)

Valor del primer Byte			Etiqueta Función	Bytes de datos Segundo Byte	Tercer Byte
Binario	Hex	Decimal			
11110011	F3	243	Selección del tema (#)	(0-127)	SIN VALOR
11110100	F4	244	Indefinido	?	?
11110101	F5	245	Indefinido	?	?
11110110	F6	246	Petición de tono	SIN VALOR	SIN VALOR
11110111	F7	247	Fin del sistema (EOX)	"	"
11111000	F8	248	Reloj de compás	"	"
11111001	F9	249	Indefinido	"	"
11111010	FA	250	Inicio	"	"
11111011	FB	251	Continuación	"	"
11111100	FC	252	Paro	"	"
11111101	FD	253	Indefinido	"	"
1111110	FE	254	Detección de Actividad	"	"
1111111	FF	255	Reinicio del sistema	"	"

Tabla II.5 Byte de estado del 176 al 191

Valor del segundo Byte			Función	Valor del tercer byte	
Binario	Hex	Dec		Valor	Orden de byte
00000000	00	0	Banco de selección	0-127	MSB
00000001	01	1	Ajuste de modulación	0-127	MSB
00000010	02	2	Control de soplido	0-127	MSB
00000011	03	3	Indefinido	0-127	MSB
00000100	04	4	Control del pedal	0-127	MSB
00000101	05	5	Tiempo del Portamento	0-127	MSB
00000110	06	6	Entrada de Datos	0-127	MSB
00000111	07	7	Canal de Volumen	0-127	MSB
00001000	08	8	Balance	0-127	MSB
00001001	09	9	Indefinido	0-127	MSB
00001010	0A	10	Pan	0-127	MSB
00001011	0B	11	Controlador de expresión	0-127	MSB
00001100	0C	12	Control de efectos 1	0-127	MSB
00001101	0D	13	Control de efectos 2	0-127	MSB
00001110	0E	14	Indefinido	0-127	MSB
00001111	0F	15	Indefinido	0-127	MSB
00010000	10	16	Controlador de Propósito Gen. #1	0-127	MSB
00010001	11	17	Controlador de Propósito Gen. #2	0-127	MSB
00010010	12	18	Controlador de Propósito Gen. #3	0-127	MSB
00010011	13	19	Controlador de Propósito Gen. #4	0-127	MSB
00010100	14	20	Indefinido	0-127	MSB
00010101	15	21	Indefinido	0-127	MSB
00010110	16	22	Indefinido	0-127	MSB
00010111	17	23	Indefinido	0-127	MSB
00011000	18	24	Indefinido	0-127	MSB
00011001	19	25	Indefinido	0-127	MSB
00011010	1A	26	Indefinido	0-127	MSB
00011011	1B	27	Indefinido	0-127	MSB
00011100	1C	28	Indefinido	0-127	MSB

Tabla II.5 Byte de estado del 176 al 191 (Cont.)

Valor del segundo Byte			Función	Valor de Datos en el byte	
Binario	Hex	Dec		Valor	Orden de byte
00011101	1D	29	Indefinido	0-127	MSB
00011110	1E	30	Indefinido	0-127	MSB
00011111	1F	31	Indefinido	0-127	MSB
00100000	20	32	Banco de Selección	0-127	LSB
00100001	21	33	Ajuste de modulación	0-127	LSB
00100010	22	34	Control de soplido	0-127	LSB
00100011	23	35	Indefinido	0-127	LSB
00100100	24	36	Control del pedal	0-127	LSB
00100101	25	37	Tiempo del Portamento	0-127	LSB
00100110	26	38	Entrada de Datos	0-127	LSB
00100111	27	39	Canal de Volumen	0-127	LSB
00101000	28	40	Balance	0-127	LSB
00101001	29	41	Indefinido	0-127	LSB
00101010	2A	42	Pan	0-127	LSB
00101011	2B	43	Controlador de expresión	0-127	LSB
00101100	2C	44	Control de efectos 1	0-127	LSB
00101101	2D	45	Control de efectos 2	0-127	LSB
00101110	2E	46	Indefinido	0-127	LSB
00101111	2F	47	Indefinido	0-127	LSB
00110000	30	48	Controlador de Propósito Gen. #1	0-127	LSB
00110001	31	49	Controlador de Propósito Gen. #2	0-127	LSB
00110010	32	50	Controlador de Propósito Gen. #3	0-127	LSB
00110011	33	51	Controlador de Propósito Gen. #4	0-127	LSB
00110100	34	52	Indefinido	0-127	LSB
00110101	35	53	Indefinido	0-127	LSB
00110110	36	54	Indefinido	0-127	LSB
00110111	37	55	Indefinido	0-127	LSB
00111000	38	56	Indefinido	0-127	LSB
00111001	39	57	Indefinido	0-127	LSB
00111010	3A	58	Indefinido	0-127	LSB
00111011	3B	59	Indefinido	0-127	LSB
00111100	3C	60	Indefinido	0-127	LSB
00111101	3D	61	Indefinido	0-127	LSB
00111110	3E	62	Indefinido	0-127	LSB
00111111	3F	63	Indefinido	0-127	LSB
01000000	40	64	Pedal de sordina	<63=off/ 64=on	- -
01000001	41	65	Portamento	<63=off/64=on	- -
01000010	42	66	Sostenimiento	<63=off/64=on	- -
01000011	43	67	Pedal Suave	<63=off/64=on	- -
01000100	44	68	Pedal de pie	<63=off/64=on	- -
01000101	45	69	Sostenimiento 2	<63=off/64=on	- -
01000110	46	70	Control de Sonido 1	0-127	LSB
01000111	47	71	Control de Sonido 2	0-127	LSB
01001000	48	72	Control de Sonido 3	0-127	LSB
01001001	49	73	Control de Sonido 4	0-127	LSB
01001010	4A	74	Control de Sonido 5	0-127	LSB
01001011	4B	75	Control de Sonido 6	0-127	LSB

Tabla II.5 Byte de estado del 176 al 191 (Cont.)

Valor del segundo Byte			Función	Valor del tercer byte	
Binario	Hex	Dec		Valor	Orden de byte
01001100	4C	76	Control de Sonido 7	0-127	LSB
01001101	4D	77	Control de Sonido 8	0-127	LSB
01001110	4E	78	Control de Sonido 9	0-127	LSB
01001111	4F	79	Control de Sonido 10	0-127	LSB
01010000	50	80	Controlador de Propósito Gen. #1	0-127	LSB
01010001	51	81	Controlador de Propósito Gen. #2	0-127	LSB
01010010	52	82	Controlador de Propósito Gen. #3	0-127	LSB
01010011	53	83	Controlador de Propósito Gen. #4	0-127	LSB
01010100	54	84	Control de Portamento	0-127	Nota fuente
01010101	55	85	Indefinido	0-127	LSB
01010110	56	86	Indefinido	0-127	LSB
01010111	57	87	Indefinido	0-127	LSB
01011000	58	88	Indefinido	0-127	LSB
01011001	59	89	Indefinido	0-127	LSB
01011010	5A	90	Indefinido	0-127	LSB
01011011	5B	91	Profundidad de los efectos 1	0-127	LSB
01011100	5C	92	Profundidad de los efectos 2	0-127	LSB
01011101	5D	93	Profundidad de los efectos 3	0-127	LSB
01011110	5E	94	Profundidad de los efectos 4	0-127	LSB
01011111	5F	95	Profundidad de los efectos 5	0-127	LSB
01100000	60	96	Entrada de Datos +1	N/A	
01100001	61	97	Entrada de Datos -1	N/A	
01100010	62	98	Número de parámetro no Registrado en LSB	0-127	LSB
01100011	63	99	Número de parámetro no Registrado en LSB	0-127	MSB
01100100	64	100	Número de parámetro Registrado en LSB	0-127	LSB
01100101	65	101	Número de parámetro Registrado en MSB	0-127	MSB
01100110	66	102	Indefinido	?	
01100111	67	103	Indefinido	?	
01101000	68	104	Indefinido	?	
01101001	69	105	Indefinido	?	
01101010	6A	106	Indefinido	?	
01101011	6B	107	Indefinido	?	
01101100	6C	108	Indefinido	?	
01101101	6D	109	Indefinido	?	
01101110	6E	110	Indefinido	?	
01101111	6F	111	Indefinido	?	
01110000	70	112	Indefinido	?	
01110001	71	113	Indefinido	?	
01110010	72	114	Indefinido	?	
01110011	73	115	Indefinido	?	
01110100	74	116	Indefinido	?	
01110101	75	117	Indefinido	?	
01110110	76	118	Indefinido	?	

Tabla II.5 Byte de estado del 176 al 191 (Cont.)

Valor de Encendido Byte			Función	Valor de Encendido Byte	
Binario	Hex	Dec		Valor	Encendido Byte
01110111	77	119	Indefinido	?	
01111000	78	120	Todo el sonido apagado	0	
01111001	79	121	Reinicia todos los controladores	0	
01111010	7A	122	Control local	0 on /127 off	
01111011	7B	123	Todas notas apagadas	0	
01111100	7C	124	Modo Omni apagado	0	
01111101	7D	125	Modo Omni activo	0	
01111110	7E	126	Modo Poly apagado	**	
01111111	7F	127	Modo Poly encendido	0	

Bibliografía

[1] **Rimmer**, Steve. Bit-mapped graphics. Windcrest / Mc Graw-Hill, 1993.

[2] **Kay**, David C. y **Levine**, John R. Graphics File Formats, Windcrest /McGraw-Hill 1991.

[3] **Walden**, Jeff. File Formats for Popular PC Software: A Programmer's reference. John Wiley & Sons, Inc. 1986.

[4] **Aho**, Alfred. Estructura de datos y algoritmos. . Addisson Wesley, Willmingthon, 1988.

[5] **Vaughan**, Tay. Todo El poder de multimedia. Mcgraw-Hill, Distrito Federal, Mexico, 1995. 2a Edicion.

[6] **Gailly**, Jean-loup. Compression-FAQ:part2. Newsgroup, paris, Francia, 1996.

ftp://rtfm.mit.edu/pub/usenet/news.answers/compression-faq/part2

[7] **Poynton**, A. Charles. ColorSpace-FAQ: "Frequently Asked Questions about Gamma and Colour" Newsgroup, NY.New York, 1995.

http://www.inforamp.net/~poynton/

[8] **Murray**, James D. Graphics File Formats FAQ (Part 1 of 4): General Graphics Format Questions. Newsgroup, Orange County, California, 1996.

ftp://rtfm.mit.edu/pub/usenet/news.answers/graphics/fileformats-faq1

[9] **Murray**, James D. Graphics File Formats FAQ (Part 3 of 4): Where to Get File Format Specifications. Newsgroup, Orange County, California, 1996.

ftp://rtfm.mit.edu/pub/usenet/news.answers/graphics/fileformats-faq3

[10] Compuserve Incorporated. Graphics Interchage Format. Compuserve, Columbus,1990.

http://www.dcs.ed.ac.uk/~mxr/gfx/2d/GIF89a.txt

[11] Aldus Corporation. Tag Image File Format: Revision 6.0.Aldus, Seattle,1992.

http://www.dcs.ed.ac.uk/~mxr/gfx/2d/TIFF.txt

[12] Autodesk, Incorporated. Drawing Interchange and File Formats Release 12 . Autodesk, Sausalito, California, E.U.,1992.

http://www.dcs.ed.ac.uk/~mxr/gfx/2d/DXF12.txt

[13] Hamilton, Eric.JPEG File Interchange Format Version 1.02.C-Cube Microsystems, Milpitas, California. 1992.

http://www.dcs.ed.ac.uk/~mxr/gfx/2d/JPEG.txt

[14] Truevision, Incorporated. Truevision Graphics Archive .Shadeland Station, Indianapolis, 1992.

http://www.dcs.ed.ac.uk/~mxr/gfx/2d/TGA.txt

[15] Pesce Mark, Parsi Anthony **y Bell** Gavin.Virtual Reality Modeling Language. Silicon Graphics Computer Systems, 1995.

http://www.dcs.ed.ac.uk/~mxr/gfx/3d/vrml.txt

[16] Rossum van, Guido. Sound File Formats, F.A.Q., part 1, Newsgroup, Amsterdam,agosto, 1996.

ftp://rtfm.mit.edu/pub/usenet/news.answers/audio-fmts/part1

[17] Rossum van, Guido. Sound File Formats, F.A.Q., part 2, Newsgroup, Amsterdam,agosto, 1996.

ftp://rtfm.mit.edu/pub/usenet/news.answers/audio-fmts/part2

[18] Apple Computer. AIFF-C. Apple Computer, 1991.

[19] Reckhard, Tobias. MOD FAQ. Newsgroup, Massachussets, Agosto, 1996
http://www.armory.com/~greebo/faq1.html

[20] Varios. MPEG fAQ, MPEG/ISO, 1995.
http://www.mpeg.org/faq/

[21] Varios. MPEG Overview, C-Cube Systems, 1995
http://www.c-cube.com:80../tecno/mpeg.html

[22] Varios. QuickTime fAQ, Apple Computer, 1996.
http://www.QuickTimeFAQ.org

[23] Varios. Indeo Video FAQ,, Intel, 1995.
http://www.intel.com/pc-supp/multimed/indeo/