UNIVERSITAT POLITÈCNICA DE CATALUNYA
ESCOLA TÈCNICA SUPERIOR D'ENGINYERS INDUSTRIALS DE TERRASSA
DEPARTAMENT D'ENGINYERIA TÈXTIL I PAPERERA

TESIS DOCTORAL

INCORPORACIÓN DE INTERFACES GRÁFICAS A LA SIMULACIÓN DE TEJIDOS JACQUARD MEDIANTE HARDWARE ESTÁNDAR.

por

MANUEL OCHOA VIVES

Ingeniero Industrial de la E.T.S.E.I. de Terrassa

para la obtención del Titulo de

DOCTOR INGENIERO INDUSTRIAL

Director de tesis: Dr. Ing. Ind. FRANCISCO HERNANDEZ ABAD

Codirector de tesis: Dr. Ing. Ind. JOSEP MUMBRU LAPORTA

e-mail: mochoa@ege.upc.es

Terrassa, Octubre de 1997

INDICE DE CONTENIDOS

CAPITULO 1

ANTECEDENTES, JUSTIFICACION Y OBJETIVOS

1.1. INTRODUCCION

El año 1832 se considera como el inicio de la Revolución Industrial en Cataluña y España, con la puesta en marcha de la fábrica BONAPLATA, RULL, VILAREGUT Y CIA. accionada por la energía del vapor. La historia nos enseña que las industrias van a ir incorporando diversos equipos auxiliares necesarios par el proceso de fabricación y manipulación.

La industrialización del sector textil catalán, tuvo su continuidad a partir de esa fecha y así, a mediados del siglo pasado surgieron dos industrias textiles bien significativas como la ESPAÑA INDUSTRIAL y el VAPOR VELL DE SANTS, al tiempo que se creaban industrias auxiliares, sobre todo metalúrgicas, como NUEVO VULCANO y la MAQUINISTA TERRESTRE Y MARITIMA.

La necesidad de energía primaria en el sector textil ha sido fundamental para su desarrollo y de todos es bien conocido el establecimiento de industrias fabriles en las cuencas de los ríos, aprovechando la energía del agua para producir el movimiento necesario en el proceso industrial.

La dependencia entre las diversas tecnologías de cada momento histórico y su aplicación en diversos sectores industriales, ha quedado patente en el proceso seguido por la Industria Textil desde sus comienzos, donde se aprecia la inevitable vinculación a las industrias metalúrgicas para la fabricación de piezas para telares y de maquinas de vapor.

Haciendo un salto en el tiempo hasta nuestros días, esta interdependencia es aún mucho más palpable, por cuanto los progresos técnicos necesarios en la Industria Textil, no pueden proceder directamente de dicho sector, sino de otros sectores industriales cuyos progresos y mejoras se pueden aprovechar directa o indirectamente en el proceso de

fabricación o en el diseño de la propia maquinaria.

Del mismo modo, las inquietudes del sector textil, en cuanto a su mejora y nueva maquinaria fuerzan a otros industriales a desarrollarse y sacar nuevos componentes y equipos que, a parte de ser útiles a dicho sector textil, lo sean para otros sectores industriales en otras aplicaciones de fabricación. Un ejemplo significativo es precisamente el desarrollo de la informática para el diseño textil, que puede usarse para otros tipos de diseño en otro sector de la industria.

La tesis doctoral que se presenta entra de lleno en este último aspecto, ya que, siendo el diseño de motivos (dibujos, colores, composición...) un factor importante en la aceptación de la "moda", no está pensado exclusivamente para su uso en este sector, sino que puede acomodarse a cualquier tipo de diseño gráfico en otros sectores.

Es por ello que, no se puede independizar ni aislar un sector de fabricación como es el Textil para que cree su propia tecnología, debiéndose apoyar en el resto de sectores industriales y sobre todo en el desarrollo de equipos auxiliares.

Siendo la Industria Textil uno de los cuatro pilares de la economía catalana, y a pesar de la crisis que ha padecido y sigue padeciendo dicho sector junto con otros sectores, continúa desarrollándose por el esfuerzo de unos industriales luchadores que incorporan constantemente a sus procesos, las nuevas tecnologías, como son: la informática, la robótica y otra importante como es el diseño.

1.2. ANTECEDENTES Y AGRADECIMIENTOS

Sin menospreciar los esfuerzos de quienes han desarrollado paquetes informáticos de diseño textil, tanto en el aspecto comercial para su beneficio como los que han tenido inquietudes investigadoras en este aspecto, agradezco públicamente a las personas que me han introducido en esta materia y me han hecho sentir esta inquietud de investigar en este tema y en particular.

Al Doctor Francisco Hernández Abad, Catedrático de Universidad en el Area de Expresión Gráfica en la Ingeniería de la ETSEIT, su paciencia y buen criterio para conseguir unas rutinas informáticas que produzcan un entorno amigable y sus conocimientos para abordar esta tarea ya que desde el año 1985 se introdujo en el tema de diseño textil siendo diversos los artículos publicados tanto en revistas como en congresos, así como un programa de diseño textil en BASIC, que se puede considerar como el punto de partida para el desarrollo de esta tesis doctoral.

Al Doctor José Mumbrú Laporta, Catedrático de Universidad en el Area Textil de Diseño, Análisis y Acabado de Tejidos de la ETSEIT, su experiencia en el diseño de tejidos y sus buenos consejos para orientar sobre las necesidades y posibilidades de las rutinas que se iban creando, sintiendo una gran inquietud en el diseño de tejidos por ordenador, como muestran sus publicaciones y trabajos realizados.

1.3. JUSTIFICACION Y OBJETIVOS

Uno de los motivos que promueven la realización de esta tesis doctoral es la de impulsar el desarrollo de la investigación acerca del diseño textil mediante ordenadores estándares. A pesar de la existencia de paquetes de diseño comerciales, la mayoría de ellos están desarrollados sobre equipos muy potentes y nada compatibles, quedando los usuarios a merced del distribuidor para cuantas cuestiones, mejoras y adaptaciones se requieran.

Con el desarrollo de la informática, han aumentado las posibilidades de las demás áreas industriales, beneficiándose de sus ventajas y posibilidades, no concibiéndose en la actualidad por ejemplo un gabinete de arquitectura sin un equipo gráfico de diseño. Así pues, creemos que nuestra Industria Textil puede adaptarse a las nuevas tecnologías informáticas, facilitándoles paquetes de diseño de tejidos, fáciles de manipular y que a su vez puedan gobernar la maquinaria textil reproduciendo los diseños pensados con ordenador sin tener que hacer la puesta en carta de forma manual.

Antes de iniciar este trabajo de investigación se ha procurado conocer la realidad en ese momento, no deseando crear un programa informático que dependiera de un sistema operativo que nos procurara unas interfaces gráficas tipo Windows dependientes de un software determinado o competir con programas muy específicos y de alto coste en el ramo textil.

Con las interfaces que se han desarrollado es muy fácil conseguir un diseño de tejido, y manipularlo con la base de datos lo suficientemente amplia para poder incorporar modificaciones y composiciones. Del mismo modo podremos manipular y mezclar colores a nuestra voluntad, viendo casi instantáneamente las distintas variedades de colorido que un mismo diseño puede aportarnos.

Una cuestión muy importante será la de considerar los periféricos tanto de entrada como de salida, mereciendo especial atención los de entrada, como son: el teclado, el dispositivo señalador (ratón), escáner y vídeo, de forma que podamos reproducir en pantalla tejidos existentes y por tanto introducirlos en nuestra base de datos para su posible manipulación. Respecto los periféricos de salida habrá que tener en cuenta la resolución de la pantalla, impresora y salida para máquinas de control numérico.

Para que todo lo anterior sea posible se ha procurado una buena gestión de almacenamiento de datos en disco duro ó en disquete, de forma que se pueda guardar cualquier diseño y se pueda recuperar fácilmente en el momento deseado.

Las interfaces gráficas desarrolladas e implementadas como subrutinas de programación se integrarán en un programa principal como módulos secundarios independientes y estructurados de forma que se puedan comunicar con los periféricos tanto de entrada como de salida, a veces directamente y otras mediante programas interfaces.

Con los conceptos implicados en la tecnología textil debidamente asimilados, se creará una estructura de datos que pueda ser manipulada sin restricciones, interconectada con un programa principal a base de menús de selección, cuyo acceso se producirá mediante la creación de un sistema inteligente basado en la programación orientada a objetos.

Una vez terminado el paquete de diseño, se efectuarán pruebas para explorar los posibles fallos o fugas no detectados inicialmente, primero a nivel interno y luego a nivel externo, procurando que el primer acceso externo se produzca en empresas o instituciones que hayan mostrado su interés en el desarrollo del producto.

CAPITULO 2

GRAFICOS POR COMPUTADORA

2.1. INTRODUCCION

La historia de los gráficos por computadora es relativamente corta en el espacio de tiempo pero de gran contenido y con grandes avances en el sector de la informática. Esta disciplina ha conseguido introducirse en el mundo del arte y del diseño de tal forma que los artistas mas creativos tienen en sus manos una poderosa herramienta que permite adaptar sus ideas de forma mas rápida y con resultados de presentación difíciles de lograr sin el uso de estas técnicas.

El desarrollo de los gráficos por computadora ha sido posible porque simultáneamente han ido evolucionando el hardware y el software informatico, si bien el primero es el que ha tenido mas influencia en el aumento de las prestaciones gráficas ya que las ideas de los diseñadores de software se encuentran con las limitaciones del hardware disponible en el momento.

Con los primeros pasos de la era informática, o como se denominó inicialmente, era de los computadores ya se realizaron gráficos en tubos de rayos catódicos (CRT), si bien el nacimiento oficial de los gráficos por computadora se debe a la tesis doctoral de Ivan Sutherland que desarrolló un paquete de dibujo SKETCHPAD en el que mediante la interacción del teclado y de un lápiz óptico podía seleccionar una serie de símbolos construidos y almacenados previamente en una estructura de datos, siendo el precursor del Sistema Gráfico Interactivo Jerárquico para Programadores (PHIGS).

Esta metodología inicial requería de equipos muy potentes y sofisticados que se fueron desarrollando y aplicando en industrias que podían costearse estos equipos como por ejemplo la General Motors, en el diseño de automóviles y otras industrias que requerían muchas horas de dibujo, apareciendo los conceptos de CAD (Diseño Asistido por Computador) y de CAM (Fabricación Asistida por Computador).
Durante unos años esta tecnología quedó restringida a unos privilegiados, y no fue hasta

el año 1981 en el que IBM lanzó al mercado el primer microordenador PC con una pantalla monocromo (MDA) que era capaz de crear figuras gráficas con la combinación de los 256 caracteres previamente desarrollados por IBM. Podemos considerar este momento como la rampa de lanzamiento de los gráficos por computadora para la propia IBM, APPLE y otros, que de forma imparable sigue evolucionando a pasos agigantados, no pudiendo predecir las nuevas tecnologías que tendremos el día de mañana.

Casi inmediatamente del lanzamiento del primer PC apareció el adaptador de gráficos en color (CGA) desarrollado también por IBM en el que se podían controlar ya no los caracteres de texto sino los pixels obteniendo imágenes a color que si bien no podían competir con las imágenes generadas por otros ordenadores si fue un lanzamiento millonario de microordenadores. En 1984 salió al mercado un adaptador de gráficos mas perfeccionado (EGA) que no tuvo una gran acogida a nivel de programadores gráficos que seguían prefiriendo el ordenador al microordenador (PC). Sin embargo los precios de los PCs con EGA fueron bajando y el interés de los programadores subiendo, empezándose a programar juegos gráficos para el consumo de ordenadores personales.

Sin desmerecer todo lo aportado hasta el momento un paso de gigante fue el lanzamiento en 1987 del adaptador VGA (Video Graphics Array) que permite una generación de imágenes de alta calidad poniendo al alcance del pequeño bolsillo este "electrodoméstico" que es el ordenador personal, más conocido popularmente como PC.

Desde los inicios de los gráficos por computadora, ya se vio la necesidad de estandarizar unas especificaciones que fueran independientes de los dispositivos de entrada o salida, por lo que se creó una especificación 3D Core Graphics System, más conocida como CORE. Esta primera especificación tuvo una gran implantación, siendo aceptada por ANSI (American National Standards Institute) e ISO (International Standards Organitation), siendo la base para crear la primera especificación gráfica estandarizada como fue la GKS (Graphical Kernel System) que era una versión

depurada de CORE aplicada a 2 dimensiones, con posterioridad también se creo la versión 3D de GKS que está siendo sustituida por un sistema gráfico más complejo denominado PHIGS (Programmer's Hierarchical Interactive Graphics System), teniendo un sucesor más completo denominado PHIGS PLUS.

Estas especificaciones son unos estándares pensados por y para instituciones oficiales o profesionales, resultando de una gran complejidad de implantación y diseñados para ordenadores muy potentes, lejos del alcance del usuario de PCs. Por ello se han desarrollado otras especificaciones estándares de uso mas popular, por compañías comerciales y universidades cuyo objetivo es el de dar mas difusión entre el publico en general. Los mas conocidos industrialmente son el PostScript, el Open GL, el X Window System, etc. Hay otros estándares de software gráfico para uso comercial como son el SRGP (Simple Raster Graphics Package) y el SPHIGS (Simple PHIGS).

En realidad no deben importarnos estas especificaciones internas de los ordenadores que solucionan los sistemas operativos, los lenguajes de programación y los paquetes gráficos en general, ya que están lo suficientemente implantadas y verificadas; el problema nos aparece al crear un programa gráfico y pretender guardar la imagen diseñada para poderla recuperar y manipular posteriormente.

En este capítulo se expone la problemática del almacenamiento de datos gráficos y el formato gráfico de almacenamiento que se ha diseñado específicamente para el tratamiento textil del programa que nos ocupa.

2.2. BASES DE DATOS GRAFICOS

Uno de los problemas que se encuentra cualquier programa informático que manipule gráficos es el de disponer adecuadamente de una serie de rutinas para leer, y escribir imágenes lo más rápido posible soportando distintos formatos gráficos procedentes de un escáner, vídeo, de otro programa gráfico o del propio programa que genera dichas imágenes. Será preciso poder acceder a imágenes almacenadas en formatos gráficos que normalmente manejan en nuestros ordenadores.

Se pueden distinguir dos tipos de almacenamiento de gráfico:

- El tipo bitmap que es el más común y el más sencillo de implementar para almacenar una imagen como una malla o matriz de puntos. Los valores de cada pixel y los nodos de la malla, se representan y almacenan de forma independiente. El nombre de bitmap viene de que cada dato o bit es mapeado en la imagen haciendo que tenga un valor determinado en ese punto. Los formatos GIF, TIFF, JPEG, etc. están creados con este tipo de almacenamiento.

- El tipo vectorial es el más común en programas de CAD, definiéndose las imágenes como una serie de líneas o curvas, con regiones llenas de colores. o bien como formas simples regulares.

Existe un caso especial, como el formato PostScript que puede considerarse una mezcla de los dos, siendo en realidad un lenguaje de programación solo para impresoras, permitiendo el acceso a gráficos vectoriales y al de bitmaps.

En lo que sigue nos vamos a referir a los formatos gráficos del tipo bitmap para manejar ficheros de imágenes.
A la hora de elegir un tipo de formato para un fichero, lo importante es definir su

utilidad posterior, si solo deseamos que se lea en un programa o deseamos que se pueda acceder a través de correo electrónico o deseamos que su utilización sea lo mas rápidamente posible sin importar una cierta pérdida de calidad. Aquí es donde tiene importancia conocer el tipo de compresión que utiliza el formato elegido.

No existe todavía el formato universal que resuelva todos los problemas y dependerá de nuestras posibilidades y necesidades elegir el formato adecuado.

Una recomendación ampliamente aceptada es el de usar formatos del tipo JPEG cuando no importa una cierta pérdida de información no perceptible como por ejemplo al visualizar una fotografía; y el formato TIFF cuando se desea una mejor calidad de imagen sin perdida de información. El formato TIFF mejorado tiene grandes posibilidades de convertirse en el formato del futuro.

Antes de hacer una descripción de los algunos de los formatos gráficos mas usuales, es conveniente conocer los distintos métodos de compresión que usan los mismos.

2.2.1. Métodos de Compresión

Ante la gran cantidad de datos que un formato gráfico maneja, se hace absolutamente necesario emplear métodos de almacenamiento de compresión que a su vez descompriman cuando se requiera volver a usarlos. Esta metodología está tomando cada vez más importancia con las nuevas tecnologías de transmisión y de comunicación telefónica.

Los tres métodos mas importantes de compresión son: el JPEG (Joint Photographic Experts Group), el LZW (Lempel-Ziv & Welch) y el RLE (Run Length Encoded), de los cuales se va a hacer una breve descripción

Para el tratamiento de imágenes el método elegido debe intentar cubrir los siguientes objetivos:

- Poder procesar imágenes de cualquier tipo.
- Alcanzar un mínimo de compresión de 2:1 ó no aplicar la compresión.
- No depender de pequeñas variaciones en los pixels, haciendo uso de las técnicas de reducción de colores.
- Ser lo más rápido posible, tanto al comprimir como al descomprimir.
- El código implementado de compilación para compresión y descompresión debe ser lo más pequeño posible aconsejándose no superar los 10 Kb.
- Procurar no usar operaciones en coma flotante.

Los métodos tradicionales LZW y RLE cumplen correctamente todos estos objetivos pudiendo comprimir cualquier tipo de datos, y en cambio el JPEG deja de cumplir los dos últimos, lo que es bueno para la compresión de imágenes reales aunque con perdida de información a una velocidad razonable que prime sobre la calidad de la imagen.

No hay que olvidar que la calidad tiene un precio y a veces será preferible menor calidad en menos tiempo y además mas barato de transmisión y otras veces lo que primará será la calidad del imagen, no importando ni el tiempo ni el dinero.

Sin lugar a dudas, estos métodos de compresión quedarán mejorados en un futuro por nuevos métodos mas sencillos, más rápidos, de mayor compresión y de mejor calidad.

Método JPEG

JPEG son las siglas de Joint Photography Experts Group pudiéndose disponer de una librería de Thomas G. Lane y del Independent JPEG Group's software en ftp://ftp.uu.net/graphics/jpeg/.

La compresión JPEG se está convirtiendo en el estándar para el almacenamiento de imágenes, entendiendo por tales aquellas que no son generadas por el ordenador sino obtenidas por otros medios como escáner, cámara o vídeo. Este método ha sido concebido por un grupo de expertos en fotografía en colaboración con los comités del CCITT y la ISO.

La explicación y comprensión del método es realmente complejo, interesándonos mas qué hace, que cómo lo hace. Anteriormente ya se ha mencionado que al contrario de los otros métodos, el JPEG no reconstruye la imagen original perfectamente, a menos que recortemos sensiblemente su velocidad.

JPEG almacena, de forma prioritaria, los cambios de color, a los que el ojo humano es especialmente sensible, logrando que nuestra percepción visual no detecte diferencias entre las imágenes antes y después del proceso.

El nacimiento de JPEG pretendía cubrir los siguientes objetivos:

- Conseguir la mejor relación entre ratio de compresión y fidelidad a la imagen.
- No quedar restringido al tamaño de la imagen y su número de colores.
- Conseguir la mayor compatibilidad de implementación.
- Diversas posibilidades de codificación:
    - Codificación secuencial de izquierda a derecha y de arriba a abajo;
    - Codificación progresiva a base de múltiples pasadas.
    - Codificación con reconstrucción perfecta.

Codificación jerárquica con posibilidad independiente a varias resoluciones.

JPEG tiene muy en cuenta los cambios de brillo en la imagen usando espacios de color del mundo de la televisión como es el YC_bC_r., de acuerdo con la siguiente conversión del espacio RGB.

$$Y = 0{,}0299\ R + 0{,}587\ G + 0{,}114\ B$$

$$C_b = -0{,}1687\ R - 0.3313\ G + 0{,}5\ B$$

$$C_r = 0{,}5\ R - 0{,}4187\ G - 0{,}0813\ B$$

El método JPEG está articulado en el uso de la transformada del coseno (DCT) para convertir una matriz de datos en otro de frecuencias que nos indique como varían las intensidades.

En la etapa de codificación, se pretende conocer cómo y cuando cambian los valores, los cuales variarán muy poco en las frecuencias más bajas para imágenes normales. En esta etapa de cuantificación se procurará no almacenar más precisión que la deseada para la imagen requerida mediante una tabla de cuantificación

En la etapa de descodificación, los valores originales se aproximan multiplicando por un factor de cuantificación que es el que produce pérdida de información.

Este método está específicamente diseñado para tratar imágenes y en el que se puede indicar el grado de compresión en función de la calidad deseada. Esto es muy importante pues por una pérdida de información, que prácticamente no es visible, se consiguen unos ratios de compresión impresionantes.

Método LZW

LZW son las siglas de sus desarrolladores Lempel, Ziv y Welch, pensado inicialmente para su implementación sobre el hardware, pero ha tenido gran difusión sobre el software como el compresor de UNIX.

De su importancia cabe resaltar que uno de los formatos gráficos mas usados como el GIF, usa el método LZW.

Sus principales características son:

- Posibilidad de manejar cualquier tipo de imagen.
- Estar basado en la repetición de patrones.
- Rapidez de compresión y de descompresión.
- No usar operaciones en coma flotante.
- No perder información de la imagen.

La característica fundamental de LZW es que tiene en cuenta el hecho de que en una imagen haya partes que se repitan para de esta forma simplificar su tratamiento, manipulando tres tipos de objetos: una serie de caracteres de entrada, un conjunto de códigos de salida y una tabla de patrones, usando un algoritmo para la compresión y otro para la descompresión .

El método LZW es muy potente, teniendo una gran difusión en los formatos GIF y TIFF

Método RLE

RLE, cuyas siglas proceden de Run Length Encoding es un método genérico muy utilizado por que proporciona buenos ratios de compresión con una metodología bastante sencilla.

Sus características principales son parecidas al método anterior:

- Posibilidad de manejar cualquier tipo de imagen.
- Rapidez de compresión y mayor rapidez de descompresión.
- No usar operaciones en coma flotante.
- No perder información de la imagen.
- Implementación sencilla.

La metodología que usa es la de almacenar bloques iguales de una imagen en un solo valor, usando un contador que indique el número de repeticiones, en el caso de bloques distintos almacena una copia del mismo y un contador que indique su tamaño. En ambos casos utiliza un contador para indicar el tamaño del buffer en la imagen de salida. Por lo tanto, debe distinguirse cuando de trata de bloques de datos iguales y cuando son distintos.

Los bloques se distinguen por su signo, de forma que cuando es positivo indica un tipo de bloque y cuando es negativo el otro.

Siendo un método bastante sencillo de implementar con ratios de compresión mas que aceptables, la tendencia actual es la de combinar este método con el LZW, indicado anteriormente.

2.2.2. Formatos gráficos bitmap

GIF:

GIF son las siglas de Graphics Interchange Format, propiedad de Compuserve.

Este formato es el mas estándar en el mundo del PC debido a su buen método de compresión LZW , soportando hasta un máximo de 256 colores.

Este formato incluye una estructura interna en el fichero con las siguientes partes o bloques:

- Descriptor GIF
- Descriptor Pantalla
- Mapa Color Global
- Descriptor Imagen
- Mapa Color Local
- IMAGEN
- Terminador GIF

El Descriptor GIF identifica el tipo de fichero como GIF e informa sobre la versión, ocupando 6 caracteres: los tres primeros contienen la palabra GIF' y los otros tres identifican la versión.

El Descriptor Pantalla describe los parámetros que afectaran a todas las imágenes contenidas el fichero GIF. Como el bloque de la imagen es independiente del resto, un fichero puede contener más de una imagen.

El Mapa Color Global contiene la descripción del mapa de colores que será usado por

defecto en las imágenes contenidas en el fichero. La existencia y longitud de este bloque depende del bloque anterior. Su estructura es sencillamente una matriz de bytes de dimensión 3, de la forma RGBRGB...

El Descriptor Imagen acota en pixels donde empieza y termina la imagen tanto en la parte superior, inferior derecha e izquierda y en que posición o separación de los márgenes. Además indica si el almacenamiento de la imagen se hace de forma entrelazada o no , y si el mapa de color está ordenado o no . El estar ordenado significa que el primer color de la tabla debe ser considerado como el más importante.

El Mapa Color Local contiene la descripción del mapa de colores que será usado en la imagen que sigue. La existencia y longitud de este bloque depende del bloque anterior, siendo su estructura como la del mapa de color global y almacenado de la forma RGBRGB...

La IMAGEN es la parte que contiene la imagen en sí, consistiendo en una secuencia de bloques, que contienen un índice al mapa de color activo para cada píxel de la imagen. Los índices están almacenados de izquierda a derecha y de arriba a abajo, y deben estar dentro del tamaño del mapa activo.

Aunque se ha dicho que el método de compresión usado era el LZW, en realidad conlleva una modificación del mismo, denominándose método LZW de longitud de código variable. Esta modificación del LZW tiene en cuenta la variación de longitud que sufren los códigos que representan las ristras ya reconocidas en la imagen original. Por ello lo primero que se especifica en el bloque de la imagen es el número de bits del código inicial.

La compresión se realiza en varias etapas:

- Establecer el tamaño del código definiendo el número de bits necesarios para

representar los datos en curso. El primer byte del bloque IMAGEN almacena el número mínimo de bits necesarios para representar el conjunto de píxels en curso. El tamaño de código inicial también indica que debe empezarse a comprimir con un bit más del que representa.

- Comprimir la secuencia de píxels a series de códigos comprimidos. El algoritmo LZW convierte una serie de datos en una serie de códigos, existiendo un código especial de limpieza que reinicializa todos los parámetros y tablas, tanto en la compresión como en la descompresión. Su aparición puede darse en cualquier momento y el descompresor debe estar preparado para ello. Existe otro código que indica el final de la información marcando el fin de la compresión de la imagen.
- Construir una serie de bytes tomando el conjunto de códigos comprimidos y convertirlos a series estándar de 8 bits.
- Empaquetar los bytes en bloques precedidos por un contador.

El Terminador GIF indica el fin del fichero GIF. Contiene el valor 0x3b.

Además de los bloques mencionados existen otros bloques no menos importantes como:

- Bloque de Control Gráfico: se identifica con 0x21 seguido de 0xf9.
- Bloque de Comentario: se identifica con 0x21 seguido de 0xfe.
- Bloque de Texto: se identifica con 0x21 seguido de 0x21.
- Bloque de Aplicación: se identifica con 0x21 seguido de 0xff.

Como resumen del formato GIF, la descripción del fichero es muy minuciosa, aportando importantes modificaciones al método LZW original, siendo ampliamente aceptado que hasta el formato TIFF codifica de forma similar.

HEIGHTFIELD:

Este formato no se puede considerar un formato gráficos como tal, sino una descripción de un objeto en 3D que sirva para hacer el render en el programa Rayshade, que es un programa de ray tracing de Craig Kolb.

Normalmente se parte de imágenes en planta expresadas como DEMs (Digital Elevation Models). Como su nombre indica heightfileds significa campos de altura, y las imágenes que emplea utilizan píxels de mayor intensidad para expresar mayor altura

Es un formato cuyo único elemento en la cabecera expresa el tamaño en píxels del heightfield.

Las alturas de cada punto se almacenan del tipo float, orientadas de izquierda a derecha y de arriba a abajo. Como puede verse volvemos a fijarnos en la implementación que tenga la máquina para expresar un real .

Resumiendo, se trata de un formato particular de un programa que almacena alturas y no imágenes.

JPEG / JFIF

El JPEG FIF (File Interchange Format) o JFIF es un formato de fichero con una estructura que permite almacenar e intercambiar las compresiones JPEG, que como se ha dicho antes dispone de un buen ratio de compresión pero con cierta perdida de información en la codificación.

Este formato no incluye ninguna de la características encontradas en la especificación del TIFF JPEG, teniendo como misión el poder intercambiar imágenes comprimidas mediante JPEG. La recomendación es usar la compresión JPEG base para que cualquier lector que soporte JFIF lo pueda leer.

El uso de este formato en programas informáticos requiere el uso de la librería externa libjpeg de Thomas G. Lane y la Independent JPEG Group's software.

El espacio de color usado es el YC_bC_r con los componentes RGB calculados a partir de YC_bC_r .

La marca JPEG FIF APP0 es obligatoria al comienzo del fichero. La marca JFIF se identifica por la cadena JFIF seguida de un cero.

Al tratarse de un formato que se beneficia del método JPEG, tiene una compresión impresionante con respecto a sus competidores. JFIF es uno de los formatos más importantes hoy en día porque TIFF no ha cuajado completamente la incorporación del JPEG.

PBM / PGM / PPM:

Este es el formato Portable BitMap más sencillo y más extendido en el mundo del UNIX a través de cualquier workstation, debiendo su origen PBM Plus de Jef Poskanzer no usando ningún tipo de compresión y siendo el elegido por aquellos programas que quieran poder escribir un fichero estándar sin trabajarse mucho los interfaces con otros formatos o lo que es lo mismo sin necesidad de utilizar ninguna librería externa.

Existen dos formas principales de almacenaje, ya sea en ASCII o en binario. En la primera, todo se almacena de forma que sea legible por cualquier persona pero ocupando mucho espacio, y en la segunda forma se almacena con un byte con ese valor, siendo interpretado directamente por el ordenador, ocupando menos espacio que el anterior pero mucho más en comparación con los formatos que usan compresión.

La cabecera almacena el tipo de fichero ASCII o binario existiendo tres profundidades soportadas: de 1, 8 y 24 planos, dando lugar a tres tipos de fichero:

- Portable Bit Map (PBM):
- Portable Grey Map (PGM):
- Portable Pixel Map (PPM):

Este es un formato extremadamente sencillo de usar y leer, sobre todo en sus formatos ASCII, y que deberíamos usar si trabajamos en UNIX y queremos asegurarnos que todo el mundo lo pueda leer.

Los formatos ASCII son los únicos que se pueden transmitir directamente, sin ningún tipo de codificación, a través de correo electrónico.

PCX

El formato PCX es propiedad de Zsoft Corporation y está ligado a su programa Paintbrush, utilizando un método de compresión RLE simplificado. Su utilización está muy extendido en el mundo del PC, sobre todo bajo Windows, que junto con el BMP, es el formato estándar.

Este formato y su evolución está ligado a las propias evoluciones de los PCs. Inicialmente sólo existían tarjetas de vídeo de 16 colores fijos, este formato solo soportaba esos 16 colores. A continuación las tarjeta mejoraron a 16 colores cualesquiera y el formato PCX también. Cuando aparecieron las tarjetas VGA de 256 colores, este formato también se amplió. Finalmente con las tarjetas de 24 bits, la última versión también soporta imágenes de este tipo.

Para los expertos en programación este es un formato que da la apariencia de improvisación en el momento de la creación ya que cada nueva revisión es un parche de la anterior haciendo que a veces cada versión sea incompatibles con la anterior porque los desarrolladores de este formato solo pensaron en 16 colores creyendo que no se iban a necesitar más.

El método de compresión es un RLE muy simplificado, con la siguientes características:

- Los dos primeros bits indican si los seis siguientes son un contador o un dato.
- La compresión se hace a lo largo de líneas.
- Si el formato está separado en planos, la compresión se aplica por separado a cada uno de ellos. Por tanto primero debe descomprimirse y luego unirlos.

En resumen, se considera un mal diseño de este formato, pero nadie le niega su amplia implantación en el mundo de los PCs.

PIX:

Este formato es el utilizado por los programas de Alias Research, utilizando un método de compresión RLE simplificado, siendo uno de los formatos mas sencillos y con poco ratio de compresión.

Al soportar solo 8 bits las imágenes se consideran en tono de gris, pues no tiene mapa de definición de color.

La estructura de fichero solo consta de dos partes:

- Cabecera.
- Imagen.

La cabecera contiene las dimensiones de pantalla, los márgenes en pixels y la profundidad de la imagen.

Para el almacenamiento de la imagen utiliza el método RLE pero con la diferencia de no considerar algunos bloques de pixels para simplificar el fichero.

Este tipo es adecuado para la creación de máscaras en 8 bits, en las que no tiene sentido el color, ya que la intensidad especifica el grado de transparencia de la máscara.

En resumen este formato es muy sencillo de implementación, pero el ratio de compresión es inferior al que se obtiene con el método RLE tradicional.

Su uso queda restringido para imágenes generadas por ordenador pues los colores son siempre puros no diferenciando negros entre si (R=G=B=0).

RAW y RGB

Estos dos formatos se agrupan en un solo bloque por su parecido aunque realmente no son unos formatos gráficos completos, ya que no usan ningún método de compresión, sino almacenando directamente a un fichero. Estos dos formatos tienen su gran uso en los centros de investigación donde no importa mucho la compresión, que como se ha dicho es necesaria para la transmisión o la comercialización de programas.

La estructura del formato RAW:

- De 8 bits: la estructura interna es la de una matriz de bytes con un orden de arriba a abajo de la imagen y de izquierda a derecha siendo apto para imágenes de grises, que se pueden usar como máscara.

- De 24 bits: la estructura es la misma que para el caso anterior, pero existen 3 bloques, que representan las tres componentes básicas R G B.

La estructura del formato RGB

- De 24 bits: un solo bloque con una estructura de matriz de tres dimensiones, para cada componente RGB, siendo el orden de arriba a abajo y de izquierda a derecha.

Es bastante normal utilizar este formato y comprimirlo posteriormente con otro comando consiguiendo un formato extremadamente sencillo y una compresión equivalente a la que nos podría dar el GIF.

Estos dos formatos, junto con el PBM, son las formas más sencilla de almacenar las imágenes en un fichero directamente desde la memoria del ordenador.

RLA

Este formato RLA (Run Length encoded, versión A) es propiedad de Wavefront Technologies y utilizado por sus productos usando un tipo de compresión RLE con canales separados.

La principal desventaja de este formato es la dependencia del tipo de máquina, aunque varias máquinas usen el mismo programa

Otro factor negativo es que los lectores que se incluyen en el programa no respetan las especificaciones que se dan, incluso su propio lector da errores de lectura en el ejemplo de escritor de RLA que trae el manual de instrucciones.

Los lectores del programa no respetan la información almacenada en la cabecera. que a pesar de ser muy amplia, tiene estos defectos enunciados.

La imagen, está codificada según el método RLE de forma que compresión en sí sea igual a la del método original:

- Cada canal se codifica por separado y línea por línea.
- Al principio de cada línea se incluye un campo, que indica el tamaño, en bytes, del bloque codificado que le sigue.
- Para bloques iguales, el contador es de tipo positivo y para bloques distintos, el contador es negativo.

A pesar de una espectacular cabecera, la información proporcionada es la de siempre y la matriz de offsets ocupa un espacio innecesario muchas veces. El único método de compresión soportado es el RLE, lo que suele hacer que los ficheros sean de gran tamaño, sobre todo para programas de Ray tracing.

RLE

A pesar de su nombre genérico, y para no confundirlo con el método de compresión RLE que usa, aquí nos referimos al formato RLE desarrollado y usado por la Librería Utah Raster Toolkit de Spencer W. Thomas de Universidad de Utah. Es un formato bien pensado para su utilización con Rayshade, siendo su uso bastante frecuente en los centros que se dedican a los gráficos por computadora.

Esta librería se puede encontrar en ftp://ftp.princeton.edu/Graphics/urt/. y es una gran ventaja disponer de ella para el uso del formato RLE.

Las características del formato son:

- Pude soportar cualquier tipo de profundidad de imágenes excepto las binarias.
- La base del formato son los canales de color (rojo, verde, azul, etc...) o la referencia a un mapa de color.
- Soporta hasta 255 canales por imagen.
- Los canales soportan cualquier profundidad, aunque están restringidos a 8 bits .
- La imagen se almacena por líneas comprimidas mediante el método RLE puro.

Al igual que la mayoría de los formatos dispone de una buena cabecera en la que se almacena el tamaño de la imagen, numero de colores, bits por pixel, numero de mapas de color, color del fondo y otros.

A continuación de la cabecera le sigue la imagen en sí. Está almacenada con el origen en la esquina inferior izquierda y se describe con una serie de operaciones que la define y que son: SkipLines, SetColor, SkipPixels, PixelData, RunData y EOF como fin de la imagen RLE. De esta forma se conseguirán almacenar más de una imagen en un mismo fichero.

TARGA /TGA

El TARGA o TGA como se le conoce comúnmente es uno de los formatos más extendidos que hay en el mercado, estando considerado hasta hace poco, como el formato estándar de los formatos gráficos de alta calidad (16.4 millones de colores).

Fue introducido en el mercado por AT&T, para ser el formato de almacenamiento de las tarjetas de digitalización ATVista de TrueVision siendo ampliamente aceptado pues es raro encontrar un programa de imágenes que no incorpore un filtro para soportar TGA.

Su versatilidad de compresión es tal, que permite almacenar sin comprimir, o usar algún método de compresión, tal como se indica a continuación:

- Imagen mapeada sin comprimir
- Imagen RGB sin comprimir
- Imagen en blanco y negro sin comprimir
- Imagen mapeada, compresión RLE
- Imagen RGB, compresión RLE
- Imagen en blanco y negro, compresión RLE
- Imagen mapeada, compresión de tipo Huffman, Delta y RLE

El formato más extendido es el de usar una imagen RGB sin comprimir pues el primer objetivo era grabar lo digitalizado siendo el más sencillo de almacenar y de recuperar. El de más uso es el de almacenar la imagen RGB con el método de compresión RLE.

La estructura general del fichero es:

-Cabecera
- Mapa de Color
- Imagen

El bloque del Mapa de Color es opcional su inclusión pues una imagen true color o

color verdadero no lo necesita.

El primer byte de la cabecera indica el tamaño en bytes que tendrá el último campo de la misma. Este es un string que suele ser utilizado para describir la imagen, dar el nombre del creador, etc.

El segundo bit de la cabecera nos informa sobre la existencia o no del mapa de color en el cuerpo del fichero. El siguiente indica el tipo de compresión o no.

Con respecto al Mapa de color, el primer y segundo campo indican el índice del primer valor almacenado en el mapa de color y el número de entradas totales del mimo, respectivamente.

El tercer, y último campo de este bloque, indica el número de bits que ocupa cada entrada. Los valores posibles son 16, 24 y 32.

Finalmente respecto de la Imagen, los cuatro primeros campos de este bloque nos sitúan la imagen y nos dan el tamaño de la misma. Las dos primeras definiciones corresponden a la posición del punto inferior más a la izquierda de la imagen y las dos siguientes al número, nos dan el número de píxels de la anchura y de la altura.

El formato TARGA es uno de los más utilizados hoy en día en todo aquello que tiene que ver con el tratamiento digital de imágenes. También, es el formato preferido a la hora de almacenar información de forma simple sin comprimir, pues nos asegura la facilidad en su tratamiento y la tansportabilidad a otros programas, algo cada vez más importante.

TIFF

TIFF son las siglas de Tag-based Interchange File Format, propiedad de Aldus. Es un formato muy potente, bien pensado y puede convertirse en el estándar definitivo. Está definido de tal forma que su ampliación no modifica para nada lo definido hasta ese momento, lo que evita problemas de incompatibilidad. Lo incluye y lo permite todo: cualquier tipo de imágenes y cualquier tipo de compresión. Tiene en cuenta la ordenación de bytes de cada arquitectura, estando preparado para que el usuario introduzca información propia sin afectar a la imagen.

Así pues sus características principales se resumen diciendo que incluye:

-Cualquier formato de compresión (Pack Bits, Fax, LZW, JPEG, ...).

-Cualquier tipo de información (quien hizo la imagen, resolución en cualquier formato, reglas de ajuste para el color, etc...).

- Cualquier espacio de color (RGB, CYMK, etc...).
- Campos adicionales para lo que el usuario quiera, etc.
- Posibilidad de ampliación dado su formato interno en TAGs.

Existe una librería de Sam Leffler para el manejo completo de imágenes TIFF y que se ha convertido en otro verdadero estándar. Es muy difícil encontrar un programa, no comercial, que use este formato gráfico y no emplee la libtiff. Se encuentra disponible en ftp://ftp.sgi.com/sgi/src/tiff

Sus gran flexibilidad hace que sea un formato muy complejo, de forma que incluso los algoritmos de lectura/escritura del formato completo son enormes y por si solo ya se trata de un proyecto.

La complejidad del proyecto para crear una librería, que maneje el formato TIFF puede ser enorme, ya que los fuentes de libtiff ocupan casi el medio mega y esto no incluye el

manejo del JPEG (que son otros 600 Kb.).

La primera versión de TIFF fue introducida por Aldus Corporation en 1986. y la que se va a resumir a continuación es la última revisión, 6.0 de junio del 92.

Para obtener mas información se puede consultar a través de Internet en ftp://ftp.sgi.com/graphics/tiff/libtiff.tar.Z

Entre las ventajas principales del formato TIFF hay que destacar:

- Es capaz de describir imágenes bitonales, tonos de gris, mapeadas y true color.
- El usuario puede elegir el método de compresión que mejor le convenga.
- No está sujeto a ningún tipo de hardware, ni sistema operativo, ni sistema de ficheros, ni compilador ni procesador.
- Está diseñado para ser ampliado según las necesidades que aparezcan.
- Permite la inclusión de información privada al usuario que crea el fichero.

La última revisión 6.0 incorpora además:

- Definición de la imagen en formato CMYK y en CIE L*a*b.
- Una sección del colorimetría RGB mejorada.
- Definición de la imagen apilada (tiled).
- Compresión JPEG.

La cabecera no contiene nada de información sobre la imagen o imágenes que el fichero puede almacenar. Todo esto es independiente dentro de TIF y se especifica mediante entradas de los IFD (Image File Directory). Serán estas últimas las que se encarguen de definir todo aquello que concierne a una imagen.

La cabecera del formato tiene la siguiente estructura:

Byte Inicio	Descripción

0-1	Tipo de ordenamiento del fichero (tanto para 16 como para 32 bits)
2-3	Identifica el fichero como de tipo TIFF, la ordenación de su valor depende del campo anterior
4-7	Offset al primer IFD. El directorio puede estar en cualquier posición del fichero tras la cabecera.

La organización de los IFDs es la de una matriz de TAGs pero que utiliza el último de sus campos para enlazarse con el siguiente. Esto es, el último campo de un IFD es un puntero, un offset, al siguiente, de forma que el último de los IFDs contiene 0 en ese offset. El primer campo de los IFDs indica el número de TAGs que éste contiene.

La estructura de un TAG es la siguiente:

Byte Inicio	Descripción
0-1	Magic que identifica el campo (llamado TAG).
2-3	Tipo del campo.
4-7	Número de valores, contador del Tipo indicado.
8-11	El valor o offset al Valor para el campo. Debe comenzar alienado con una palabra, por tanto debe ser un valor par. Puede apuntar a cualquier lugar del fichero, incluso tras la imagen.

El único lugar en el que se puede almacenar información de la imagen es en los TAGs, de los que podemos tener tantos como queramos, pero 255 por IFD. La explicación de los campos es la siguiente:

- Valor o offset: Contiene el valor si el tamaño de la palabra es de 4 bytes. Si es de menos, está justificado a la izquierda con el valor del offset.
- Contador: en versiones anteriores recibía el nombre de longitud. Es el número de valores, no el de bytes.

- Tipos:

Byte	Descripción
1 Byte	8 bits, sin signo.
2 ASCII	8 bits, contiene códigos ASCII de 7 bits y el último debe ser 0
3 Short	16 bits, sin signo.
4 Long	32 bits, sin signo.
5 Rational	2 longs que equivalen a un cociente, el primero es el numerador y el segundo el denominador.
6 Byte	8 bits, con signo
7	8 bits, puede contener cualquier cosa.
8 Short	16 bits, con signo
9 Long	32 bits, con signo.
10 Rational	2 longs que equivalen a un cociente, el primero es el numerador y el segundo el denominador.
11 Float	4 bytes, precisión normal (IEEE)
12 Double	8 bytes, precisión doble (IEEE)

El hecho de que el número de IFDs sea ilimitado, y por tanto, el de TAGs también, permite que el número de imágenes almacenadas en un mismo fichero puede ser de más de una.

Existen distintos tipos de imagen, dándose a continuación las descripciones de cada uno de los TAGs necesarios.

Imágenes Bitonales

Color

Una imagen bitonal contiene dos colores, blanco y negro.

PhotometricInterpretation

Tag = 262 (0x106)

Tipo = short

Valor	Descripción
0	blanco es 0.
1	negro es 0.

Compresión

Método de compresión utilizado en el almacenamiento de la imagen:

Compression

Tag = 259 (0x103)

Tipo = short

Valor	Descripción
1	no comprimido.
2	RLE Huffman modificado CCITT Group 1.
32773	packbits

Tamaño

La imagen está organizada como una matriz rectangular de píxels.

ImageLength

Tag = 257 (0x101)

Tipo = short o long (número de filas de la imagen)

ImageWidth

Tag = 256 (0x100)

Tipo = short o long (número de columnas de la imagen)

Dimensiones Físicas

Junto con los TAGs anteriores, informan del tamaño real de la imagen.

ResolutionUnit

Tag = 296 (0x128)

Tipo = short

Valor	Descripción
1	sin unidad de medida. Para imágenes que no tienen un aspecto ratio cuadrado.
2	pulgadas (por defecto)
3	centímetros

XResolution

Tag = 282 (0x11a)

Tipo = Rational (Número de píxels por unidad en anchura)

YResolution

Tag = 282 (0x11a)

Tipo = Rational (Número de píxels por unidad en altura).

Colocación de la información

La información de las imágenes se puede encontrar en cualquier lugar del fichero TIFF. También se permite el almacenamiento por trozos o tiras (strips) para incrementar la flexibilidad de edición y la eficiencia de I/O.

RowsPerStrip

Tag = 278 (0x116)

Tipo = short o long (Número de filas por strip).

StripOffsets

Tag = 273 (0x111)

Tipo = short o long (Para cada strip, el offset de colocación).

StripByteCounts
Tag = 279 (0x117)
Tipo = short o long (número de bytes después de la compresión).

Campos obligatorios en una imagen bitonal

Con estas listas trataremos de dar una visión global de las similitudes y diferencias entre los distintos tipos de imágenes. Aquí se listarán los TAGs que deben estar definidos, de forma obligatoria, para cada formato. En este caso para las imágenes bitonales:

Nombre	Hex	Tipo	Valor
ImageWidth	100	Short/Long	
ImageLength	101	Short/Long	
Compression	103	Short	1, 2 ó 32773
PhotometricInterpr...	106	Short	0 ó 1
StripOffsets	111	Short/Long	
RowsPerStrip	116	Short/Long	
StripByteCounts	117	Short/Long	
Xresolution	11a	Rational	
YResolution	11b	Rational	
ResolutionUnit	128	Short	1, 2 ó 3

Las imágenes bitonales recibieron el nombre de TIFF Clase B en la descripción original.

Imágenes de Escalas de Grises

Es una generalización del formato bitonal en el que se permiten las

tonalidades de grises.

Diferencias con las imágenes bitonales

En la descripción original, este formato puede estar almacenado sin comprimir o comprimido según el método PackBits.

BitsPerSample

Tag = 258 (0x102)

Tipo = Short (Es el número de bits por componente.

La descripción original sólo permite valores de 4 y 8 bits.

Campos obligatorios

Nombre	Hex	Tipo	Valor
ImageWidth	100	Short/Long	
ImageLength	101	Short/Long	
BitsPerSample	102	Short	4 ó 8
Compression	103	Short	1 ó 32773
PhotometricInterpr...	106	Short	0 ó 1
StripOffsets	111	Short/Long	
RowsPerStrip	116	Short/Long	
StripByteCounts	117	Short/Long	
Xresolution	11a	Rational	
YResolution	11b	Rational	
ResolutionUnit	128	Short	1, 2 ó 3

Su nombre ha sido, hasta ahora, el de TIFF Clase G.

Imágenes Mapeadas

Este formato y el anterior son iguales en cuanto a almacenamiento, sólo cambia el significado de los índices. En este caso, cada uno de ellos indica un mapa de color de una paleta de 16.7 millones de colores.

ColorMap

Tag = 320 (0x140)

Tipo = Short

N = 3 * (2 ^ BitsPerSample) Define el mapa de color que referencia cada valor almacenado en la imagen. El orden es RGB y los valores van del 0 al 65535 .

Campos obligatorios

Nombre	Hex	Tipo	Valor
ImageWidth	100	Short/Long	
ImageLength	101	Short/Long	
BitsPerSample	102	Short	4 ó 8
Compression	103	Short	1 ó 32773
PhotometricInterpr...	106	Short	3
StripOffsets	111	Short/Long	
RowsPerStrip	116	Short/Long	
StripByteCounts	117	Short/Long	
Xresolution	11a	Rational	
YResolution	11b	Rational	
ResolutionUnit	128	Short	1, 2 ó 3
ColorMap	140	Short	

Este tipo de imágenes es conocido como TIFF de Clase P.

Imágenes True Color (Color verdadero)

Para cada píxel se almacenan tres valores, uno para cada componente RGB, y no existe un mapa de color.

Las diferencias con las mapeadas son :

BitsPerSample = 8, 8, 8

En el TIFF Base, cada componente tiene profundidad 8.

PhotometricInterpretation = 2 (RGB)

No hay mapa de color

SamplesPerPixel

Tag = 277 (0x115)

Tipo = Short (Número de componentes por píxel)

Campos obligatorios

Nombre	Hex	Tipo	Valor
ImageWidth	100	Short/Long	
ImageLength	101	Short/Long	
BitsPerSample	102	Short	8, 8, 8
Compression	103	Short	1 ó 32773
PhotometricInterpr...	106	Short	2
StripOffsets	111	Short/Long	
SamplesPerPixel	115	Short	
RowsPerStrip	116	Short/Long	
StripByteCounts	117	Short/Long	
Xresolution	11a	Rational	
YResolution	11b	Rational	
ResolutionUnit	128	Short	1, 2 ó 3

El nombre de este formato es TIFF Clase R.

Hasta ahora se han indicado aquellos TAGs que influyen directamente en la descripción de la imagen. Sin embargo, ya en la especificación base existen muchos más:

Artist: Persona que creó la imagen.

Tag = 315 (0x13b)

Tipo = ASCII

CellLength La altura de la matriz que debe usarse para crear una imagen.

Tag = 265 (0x109)

Tipo = Short

N = 1 (Debería estar sólo presente si Threshholding = 2).

CellWidth La anchura de la matriz que debe usarse para crear una imagen.

Tag = 264 (0x108)

Tipo = Short

N = 1

Copyright Mensaje de copyright.

Tag = 33432 (0x8298)

Tipo = ASCII

DateTime Día y hora de la creación de la imagen.

Tag = 306 (0x132)

Tipo = ASCII

N = 20 El formato es el japonés, YYYY:MM:DD HH:MM:SS.

ExtraSamples Descripción de componentes extra.

Tag = 338 (0x152)

Tipo = Short

N = m (Especifica que cada píxel tiene m componentes extra.)

Los posibles valores son:

Valor	Descripción
0	Información sin especificar
1	Canal alfa (opacidad)
2	Canal alfa no asociado (transparencia)

FillOrder: La ordenación lógica de los bits en un byte.

Tag = 266 (0x10a)

Tipo = Short

N = 1

FreeByteCounts: Indica el número de bytes sin uso para cada string.

Tag = 289 (0x121)

Tipo = Long

FreeOffsets Indica la posición de comienzo de los bytes sin uso en el string.

Tag = 288 (0x120)

Tipo = Short

GrayResponseCurve: Para imágenes de tonos de grises, la densidad óptica de cada posible valor del píxel.

Tag = 291 (0x123)

Tipo = Short

N = 2 ^ BitsPerSample

GrayResponseUnit: Indica la información contenida en la GrayResponseCurve.

Tag = 290 (0x122)

Tipo = Short

N = 1

Las densiometrías ópticas escalan su valores, normalmente, entre 0 y 2.

Valor	Descripción
1	Cada número representa décimas.
2	Cada número representa centésimas.
3	Cada número representa milésimas.
4	Cada número representa diez milésimas.
5	Cada número representa cien milésimas.

HostComputer: Computadora y S. O. utilizado cuando se creó la imagen.

Tag = 316 (0x13c)

Tipo = ASCII

ImageDescription: Descripción del motivo de la imagen.

Tag = 270 (0x10e)

Tipo = ASCII

Make: Escáner, u otro equipo usado para la captura de la imagen.

Tag = 271 (0x10f)

Tipo = ASCII

MaxSampleValue: Valor del máximo componente usado.

Tag = 281 (0x119)

Tipo = Short

N = SamplesPerPixel Por defecto es 2^(BitsPerSample)+1.

MinSampleValue: Valor del mínimo componente usado.

Tag = 280 (0x118)

Tipo = Short

N = SamplesPerPixel Por defecto es 0.

Model: Modelo o número del escáner, digitalizador u otro equipo utilizado para la captura de la imagen.

Tag = 272 (0x110)

Tipo = ASCII

NewSubfileType: Indica el tipo de información contenida en el subfichero.

Tag = 254 (0xfe)

Tipo = Long

N = 1

Orientation: Orientación de la imagen con respecto a las filas y las columnas.

Tag = 274 (0x112)

Tipo = Short

N = 1

Valor	Descripción
1	la fila 0 representa el margen superior la columna 0 representa el margen izquierdo
2	la fila 0 representa el margen superior la columna 0 representa el margen derecho
3	la fila 0 representa el margen inferior la columna 0 representa el margen derecho
4	la fila 0 representa el margen inferior la columna 0 representa el margen izquierdo
5	la fila 0 representa el margen izquierdo la columna 0 representa el margen superior

6	la fila 0 representa el margen derecho la columna 0 representa el margen superior
7	la fila 0 representa el margen derecho la columna 0 representa el margen inferior
8	la fila 0 representa el margen izquierdo la columna 0 representa el margen inferior

PlanarConfiguration: Indica como son almacenados los píxels.

Tag = 284 (0x11c)

Tipo = Short

N = 1

Valor	Descripción
1	Formato chunky. Los valores de cada píxel se almacenan de forma contigua. El orden da cada componente se indica mediante PhotometricInterpretation. Para imágenes de 24 bits sería RGBRGBRGB...
2	Formato planar. Los componentes son almacenados en planos. Los valores de los conjuntos StripOffets y de StripByteCounts se ordenan como una matriz bidimensional, SamplesPerPixel como filas y StripPerImage como columnas. Con PhotometricInterpretation se describe el tipo de información almacenada en cada componente del plano.

Software: Nombre y versión del programa utilizado para la creación de la imagen.

Tag = 305 (0x131)

Tipo = ASCII

SubfileType: Tipo de información que contiene en subfichero.

Tag = 255 (0xff)

Tipo = Short

N = 1

Valor	Descripción
1	información de pantalla completa
2	información reducida de tamaño
3	una hoja parte de un informe de más

Threshholding: Para imágenes en blanco y negro que representan tonos de gris, la técnica usada para convertir desde gris a blanco y negro.

Tag = 263 (0x107)

Tipo = Short

N = 1

Valor	Descripción
1	ni dither, ni haltone ha sido aplicado a la imagen.
2	un dither ordenado o halftone ha sido utilizado
3	Se ha utilizado un proceso aleatorio.

Esta lista adicional que se ha detallado no incluía los tags ya indicados en la explicación de cada uno de los formatos.

Todo lo visto hasta ahora pertenecía a la descripción base del formato, y a partir de ahora, se describirán extensiones TIFF que no tiene porque estar soportadas por los lectores.

Se mencionarán todas y cada una de las extensiones con un breve descripción:

Compression

Tag = 259 (0x103)

Tipo = Short

N = 1

Valor	Descripción
3	codificación de imágenes bitonales CCITT T4 [CCITT]
4	codificación de imágenes bitonales CCITT T6 [CCITT]
5	compresión de tipo LZW, y del formato GIF.
6	compresión de tipo JPEG, (Esta extensión, no está aún utilizada porque aun existen problemas en su puesta en práctica.

T4Options

Tag = 292 (0x124)

Tipo = Long

N = 1

Ver Compression=3.

Consta de 32 bits que simulan flags, el bit 0 es el de menor orden:

Num bit	Descripción
bit 0 es 1	para codificación en 2 dimensiones.
bit 1 es 1	si está en modo descomprimido
bit 2 es 1	si se han añadido bits hasta completar el fin de línea.

T6Options

Tag = 293 (0x125)

Tipo = Long

N = 1

Ver Compression = 4.

Al igual que el anterior, consta de 32 bits a modo de flags:

Valor	Descripción
bit 0	sin uso, siempre 0

bit 1es 1 1 si el modo descomprimido está permitido.

DocumentName: El nombre del documento del que se capturó la imagen.

Tag = 269 (0x10d)

Tipo = ASCII

PageName: El nombre de la página de la que capturó la imagen.

Tag = 285 (0x11d)

Tipo = ASCII

PageNumber: El número de la página de la que se capturó la imagen.

Tag = 297 (0x129)

Tipo = Short

N = 2

PageNumber[0] es el número de la página actual.

PageNumber[1] es el total de paginas

Xposition: Posición X de la imagen.

Tag = 286 (0x11e)

Tipo = Rational

N = 1

El offset X en unidades de resolución (ResolutionUnits) del margen izquierdo de la página.

Yposition: Posición Y de la imagen.

Tag = 286 (0x11e)

Tipo = Rational

N = 1

El offset Y en unidades de resolución (ResolutionUnits) del margen superior de la página.

Predictor

Tag = 317 (0x13d)

Tipo = Short

N = 1

Sólo se utiliza cuando el método de compresión es LZW.

TileWidth

Tag = 322 (0x142)

Tipo = Short/Long

N = 1

La anchura de las pilas en píxels, esto es el número de columnas.

TileLength

Tag = 323 (0x143)

Tipo = Short/Long

N = 1

La altura de las pilas en píxels, esto es el número de filas.

TileOffsets

Tag = 324 (0x144)

Tipo = Long

N = TilesPerImage para PlanarConfiguration = 1

N = SamplesPerPixel * TilesPerImage para PlanaConfiguration = 2

TileByteCounts

Tag = 315 (0x145)

Tipo = Short/Long

N = TilesPerImage para PlanarConfiguration = 1

N = SamplesPerPixel * TilesPerImage para PlanaConfiguration = 2

InkSet

Tag = 332 (0x14c)

Tipo = Short

N = 1

El conjunto de tintas usadas en la separación de la imagen.

Valor	Descripción
1	CMYK. El orden en cian, magenta, amarillo y negro. Normalmente, 0 representa el 0% de tinta y 255 el 100%.
2	no CMYK. (ver InkNames).

NumberOflnks: Número de tintas.

Tag = 334 (0x14e)

Tipo = Short

N = 1

InkNames

Tag = 333 (0x14d)

Tipo = ASCII

N = número total de caracteres en los nombres de los colores.

DotRange

Tag = 336 (0x150)

Tipo = Byte/Short

N = 2 ó 2*SamplesPerPixel

Los valores de los componentes corresponden del 0 al 100% del punteado. DotRange[0] es el 0% y DotRange[1] el 100%.

TargetPrinter: Descripción de lo que se pretende a la hora de la impresión.

Tag = 337 (0x151)

Tipo = ASCII

N = cualquiera

HalftoneHints

Tag = 321 (0x321)

Tipo = Short

N = 2

Define un función acromática. Contiene dos valores de 16 bits cada uno. El primero especifica el tono de gris más claro que debería ser convertida a la tinta más clara. El segundo lo hace al contrario, trata la más oscura.

ExtraSamples

Tag = 338 (0x152)

Tipo = Short

N = 1

El valor que cada píxel contiene sobre la opacidad ha de ser multiplicado por el color correspondiente.

SampleFormat: Especifica como se debe interpretar la información de los píxels:

Tag = 339 (0x153)

Tipo = Short

N = SamplesPerPixel

Valor	Descripción
1	datos enteros sin signo
2	datos enteros con signo (complemento a dos)
3	datos reales (formato IEEE)
4	datos sin formato

SMinSampleValue: Mínimo número de valores de muestreo

Tag = 340 (0x154)

Tipo = el campo tipo que mejor se ajusta a los datos

N = SamplesPerPixel

SMaxSampleValue: número Máximo número de valores de muestreo

Tag = 341 (0x155)

Tipo = el campo tipo que mejor se ajusta a los datos

N = SamplesPerPixel

WhitePoint: Cromaticidad del punto blanco

Tag = 318 (0x13e)

Tipo = Rational

N = 2

Es el valor utilizando el diagrama de cromaticidad CIE 1931.

PrimaryChromaticities: Cromaticidad de cada color primario.

Tag = 319 (0x13f)

Tipo = Rational

N = 6

El orden es rojo[x], rojo[y], verde[x], verde[y], azul[x] y azul[y].

TransferFunction

Tag = 301 (0x12d)

Tipo = Short

N = {1 ó 3} * (1 << BitsPerSample)

Describe la función de transferencia de los datos de la imagen.

Valor	Descripción
1 << BitsPerSample	La función de transferencia es igual para cada canal
3 << BitsPerSample	Hay tres tablas, la ordenación es la misma que la señalada por el campo PhotometricInterpretation

TransferRange: Expande el rango de TransferFunction

Tag = 342 (0x156)

Tipo = Short

N = 6

El primer valor del par está asociado con TransferBlack y el segundo a TransferWhite. El orden es el especificado por PhotometricInterpretation.

ReferenceBlackWhite

Tag = 532 (0x214)

Tipo = Rational

N = 6

Especifica un par de valores. El primero vuelve a tener relación con TransferBlack y el segundo con TransferWhite.

PhotometricInterpretation (extensión)

Tag = 262 (0x106)

Tipo = Short

N = 1

YCbCrCoefficients

Tag = 529 (0x211)

Tipo = Rational

N = 0

La transformación de RGB a YCbCr se hace a través de tres valores que representan los componentes para calcular la luminancia Y.

Y = (LumaRed * R + LumaGreen * G + LumaBlue * B)

Cb = (B -Y) / (2 - 2 * LumaBlue)

Cr = (R -Y) / (2 - 2 * LumaRed)

R = Cr * (2 - 2 * LumaRed) + Y

G = (Y - LumaBlue * B - LumaRed * R) / LumaGreen

B = Cb * (2 - 2 * LumaBlue) + Y

YCbCrSubSampling

Tag = 530 (0x212)

Tipo = Short

N = 2

Especifica los factores de submuestreo usados para los componentes de la cromaticidad de la imagen YCbCr. Esta compuesto de dos campos, YCbCrSampleHoriz y YCbCrSubxampleVert, que especifican los factores horizontales y verticales, respectivamente.

Valor	Descripción
1	La anchura (altura) de esta imagen de chroma es igual a la anchura (altura) de la imagen asociada de luminancia.
2	La anchura (altura) de esta imagen de chroma es la mitad de la anchura (altura) de la imagen asociada de luminancia.
4	La anchura (altura) de esta imagen de chroma es un cuarto de la anchura (altura) de la imagen asociada de luminancia.

YCbCrPositioning

Tag = 531 (0x213)

Tipo = Short

N = 1

Especifica la posición del muestreo de la cromaticidad respecto a las componentes de la luminancia.

Hasta aquí se ha expuesto de forma más amplia que en los otros formatos la descripción del formato TIFF, por la importancia que tiene, siendo el más utilizado en el campo de la autoedición y en opinión de los expertos probablemente consiga imponerse en todos los demás .

Como se ha podido ver este formato TIFF es el más completo de todos los existentes actualmente en el mercado. La flexibilidad de su definición, los TAGs, le dan una versatilidad que ningún otro formato alcanza.

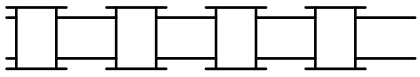
2.3. GRAFICOS TEXTILES POR COMPUTADORA

Es de justicia considerar a Joseph Marie Jacquard como el precursor de los gráficos textiles por computadora cuando todavía no se habían inventado los computadores, habiendo inventado un telar en el año 1805, que mediante unas fichas perforadas, almacenaba la información del gráfico que se había pensado sobre papel, de forma que el telar leía estas fichas tejiendo flores y otros elementos que no se podían tejer con el telar convencional de lizos.

En el año 1832 Charles Babbage inició su proyecto de diseñar una máquina analítica basada en el telar de Jacquard para realizar cálculos algebraicos, finalizando esta era preinformática en el año 1944 cuando Howard Aiken construyo el primer computador para IBM conocido como Mark 1.

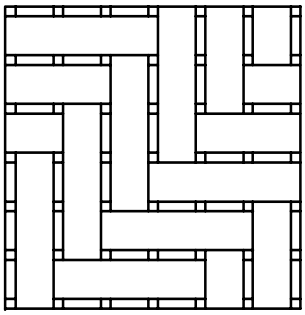
No fue hasta el año 1964 cuando Janice R. Lourie de IBM se interesó en utilizar los ordenadores para realizar el diseño textil, al asociar la manipulación de hilos en un telar, mas bien conocido como acción de tejer, con el tratamiento de los números en un ordenador, estableciendo unas analogías entre los tejidos y la tecnología de los ordenadores según el siguiente criterio:

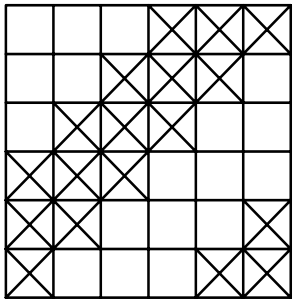
- La evolución de un hilo o una pasada en un tejido por sus “tomo o dejo” se pueden representar por un numero binario en un ordenador por “unos o ceros”, que a su vez equivaldrá a un solo número decimal.



equivale a binario 10101010 y decimal 170

- Un ligamento constituido por las evoluciones de hilos y pasadas se puede asociar a una matriz de números en un ordenador.



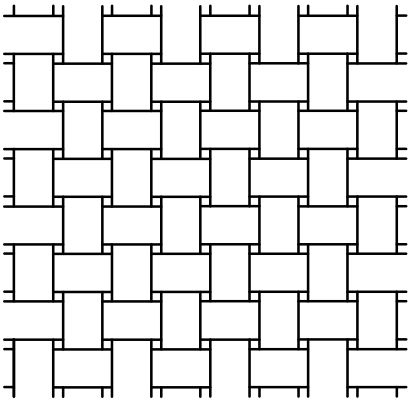


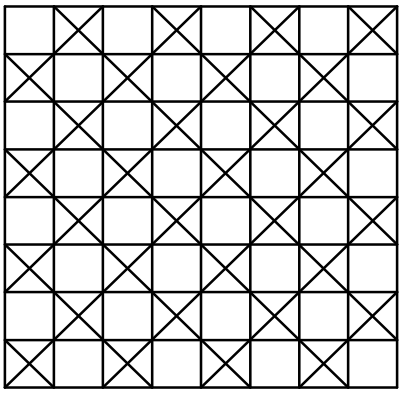
0	0	0	1	1	1
0	0	1	1	1	0
0	1	1	1	0	0
1	1	1	0	0	0
1	1	0	0	0	1
1	0	0	1	1	1

- A partir de los datos del diagrama del ligamento, se pueden obtener los diagramas del remetido y del picado por adición binaria de los lizos y la organización de dichos datos se puede tratar de forma análoga en las direcciones de memoria de un ordenador con instrucciones de manipulación de datos, con solo indicar su posición de fila y columna.

- El dibujo de un ligamento es análogo a un diagrama lógico de instrucciones de un algoritmo de ordenador, pudiéndose imponer cuantas restricciones deseemos dar a nuestro diseño.

Como resumen de las premisas anteriores cualquier diseño textil se puede asociar a un grupo de números binarios, transformándose fácilmente en una matriz, pudiéndose tratar su representación y manipulación de forma informática.





0	1	0	1	0	1	0	1
1	0	1	0	1	0	1	0
0	1	0	1	0	1	0	1
1	0	1	0	1	0	1	0
0	1	0	1	0	1	0	1
1	0	1	0	1	0	1	0
0	1	0	1	0	1	0	1
1	0	1	0	1	0	1	0

$$\begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

El problema que se presenta con los diseños textiles es que con los “unos y ceros” únicamente se consideran los “tomos y los dejos” sin contemplar el color asociado a cada hilo. La forma de resolverlo es asociando a cada cruce o pixel de pantalla, un elemento de la matriz que lleve asociado el color RGB en forma de porcentaje de rojo, de azul y de verde.

El color del elemento se identificará con un número hexadecimal de 6 cifras de forma

que las dos primeras cifras contando por la izquierda corresponden al azul abarcando 64 tonos de azul, las dos cifras intermedias son para el verde y las dos de la derecha son para el rojo.

Así pues los 16 colores fundamentales que se van a utilizar por defecto serán los siguientes:

Negro	00 00 00
Azul	2a 00 00
Verde	00 2a 00
Cian	2a 2a 00
Rojo	00 00 2a
Magenta	2a 00 2a
Marrón	00 15 2a
Blanco	2a 2a 2a
Gris	15 15 15
Azul brillo	3f 15 15
Verde brillo	15 3f 15
Cian brillo	3f 3f 15
Rojo brillo	15 15 3f
Magenta brillo	3f 15 3f
Amarillo	15 3f 3f
Blanco brillo	3f 3f 3f

2.4. BASE DE DATOS ASOCIADA A EGITEX

El programa informático de diseño textil que se ha elaborado como trabajo de investigación en esta tesis doctoral se denomina EGITEX, mostrándose su descripción y desarrollo en el CAPITULO 6, debiendo afrontar el problema de disponer de una serie de rutinas para escribir y leer las imágenes que el propio programa genera.

A la vista de los formatos gráficos mas generalizados expuestos anteriormente se ha optado por crear un sistema propio de almacenamiento de datos que permita identificar cada elemento de la matriz de diseño por la fila, la columna y el color asociado a dicha posición. De esta forma se han conseguido una base de datos sencilla, completa y de poca extensión, resultando muy apropiada para el manejo de dicho programa.

Para los motivos diseñados dentro de una retícula se ha creado una rutina que grabe dicha información en un fichero con extensión .TEX, y posteriormente existe otra rutina capaz de leer ficheros con esta extensión y formato capaz de mostrarla en pantalla para su posterior edición, transformación o manipulación.

Igualmente se han creado dos rutinas capaces de escribir y leer ficheros con la extensión .PAL que se refiere a distintas paletas de colores, modificando el aspecto del diseño en función de la paleta de colores elegida y permitiendo modificarla o crear nuevas paletas.

Como se podrá comprobar en los gráficos que aparecen en el CAPITULO 6, las retículas de diseño y las simulaciones de tejido que se muestran, se pueden insertar en este documento una vez capturadas por cualquier programa comercial que incorpore los filtros gráficos de extensión .BMP, .TIF JPG etc, y almacenarlas en alguno de estos formatos, de acuerdo con la descripción que de ellos, se ha hecho previamente en el presente capítulo.

2.4.1. Archivos .TEX

El paquete de utilidades informáticas que se presenta en este trabajo, dispone de una matriz bidimensional mat2[i][j], que guarda el valor del color que corresponde a cada fila y columna, grabándose en un fichero de extensión .TEX, mostrándose a continuación parte del código en lenguaje C/C++ que realiza la misión de escribir un fichero con este formato:

```
pf=fopen(nombrefich, "w+");                    //puntero al fichero
for(i=1;i<=nf;i++)                             //contador de filas
{       for(j=1;j<=nc;j++)                     // contador de columnas
        {       _itoa(mat2[i][j],buffer,10);   //asignar valor a un buffer
                fputs(buffer,pf);              //pasar del buffer al puntero
                fputc('\n',pf);   }       }    // fin de fichero
 fclose(pf);                                   //cerrar fichero
```

A continuación se muestra el contenido de un fichero .TEX en el que las dos primeras líneas indican las filas y columnas y el resto de líneas el color en cada uno de los elementos de la matriz. En este caso, para no ocupar mucho espacio se muestra una retícula de 4 filas y 3 columnas.

```
4
3
1
0
0
1
2
3
3
3
3
4
4
5
```

2.4.1. Archivos . PAL

Del mismo modo que en el caso anterior se han preparado unas rutinas para escribir y leer paletas de colores, que a pesar de tener la posibilidad de 64 x 64 x 64 = 262.144 colores se ha optado por simplificar la paleta a 16 colores simultáneos con el consiguiente ahorro de memoria, mejores prestaciones y mayor compatibilidad con las tarjetas de vídeo disponibles por el usuario.

A continuación se lista parte del código que escribe en un fichero los 16 colores que se tengan seleccionados en ese momento.

```
pf=fopen(nombrefich, "w+");                    //puntero al fichero
for(i=1;i<=15;i++)                             //contador de filas
        {       _ltoa(color[i],buffer,10);     //asignar valor a un buffer
                fputs(buffer,pf);              //pasar del buffer al puntero
                fputc('\n',pf);   }      }     // fin de fichero
 fclose(pf);                                   //cerrar fichero
```

Comparando los códigos de programación con el mostrado anteriormente para la retícula se observa el uso de la función _ltoa( ) en lugar de la función _itoa() por cuanto en el caso de la retícula almacenamos números enteros “integer” y para la paleta almacenamos enteros largos “long integer” con ocho cifras para el color como se ha indicado en el apartado 2.3 anterior.

CAPITULO 3

TECNOLOGIA DEL COLOR

3.1. INTRODUCCION

El Color es un elemento fundamental en los gráficos por computadora, por cuanto tiene una gran importancia en la calidad de visualización de una imagen en pantalla, ya que de él depende el resultado final de cualquier programa gráfico. Por ello se ha considerado que merece un capitulo aparte el estudio de dicho componente para comprender el tratamiento informático del color y su aplicación en el paquete de diseño textil que se presenta.

La percepción visual de un objeto ya sea en la propia realidad o a través de una pantalla de ordenador, es una percepción subjetiva, pues depende del propio ojo humano como órgano fisiológico (del que ya se pueden medir algunos de sus parámetros de sensibilidad) y del cerebro como órgano psico-fisiológico (del que aún no se pueden parametrizar sus respuestas con exactitud).

El fenómeno completo de la percepción visual implica una extensión, una duración y un color, que es el que nos definirá básicamente el objeto a partir de una luminosidad o claridad, un tono y una saturación o nivel de intensidad.

Así pues el concepto de Color puede considerarse como una percepción subjetiva de visualización, pues aparte de las características intrínsecas del objeto y del tipo de iluminación a que esté sometido, depende fundamentalmente de las características fisiológicas de la persona que observa.

Aunque no se puede considerar al Color como un parámetro estándar y medible objetivamente, desde siempre se ha pretendido teorizar sus características para establecer unos criterios de medición, que han dado como resultado la aceptación de unas recomendaciones CIE (Commission Internationale de L'Eclairage), que unifican criterios y establecen unos parámetros básicos para su medición y tabulación.

En el estudio del Color se han definido unos cuantos atributos que son los que se intentarán cuantificar para una identificación normalizada del mismo.

- Energía luminosa o luz.- Es la parte de energía visible que incide directamente en la retina del ojo.

- Color.- Entendido como color percibido es la diferenciación entre dos objetos en función del espectro de la energía radiante.

- Tono.- Es la propia percepción del color tal como se le conoce coloquialmente (verde, rojo, etc...).

- Brillo.- Sensación que se produce de colores más o menos intensos según se emita mayor o menor cantidad de luz (las variaciones de brillo van de brillante a tenue).

- Luminosidad o claridad.- Sensación de mayor o menor luz emitida con relación a un estímulo blanco (sus variaciones van de luminoso a oscuro).

- Cromaticidad o croma.- Sensación que diferencia un estímulo cromático, de otro acromático distinto, pero del mismo brillo, perceptible por la conjunción del tono y la saturación.

- Saturación.- Sensación que diferencia un estímulo cromático de uno acromático independientemente de su brillo.

3.2. ENERGIA LUMINOSA - LUZ

La luz es la fuente productora del color ya que al incidir en un objeto, éste devuelve parte de la luz incidente, transformando esta energía en forma de color, pudiéndose afirmar que el color que es una porción de la luz que ha llegado a este objeto.

Es interesante conocer los aspectos físicos de la luz para llegar a estudiar el Color y parametrizar esos atributos que repercuten en la percepción visual.

Bajo este punto de vista, sólo nos interesará conocer la luz como un fenómeno de propagación de ondas electromagnéticas dentro del rango de longitudes de ondas que podemos percibir.

Las magnitudes físicas que intervienen en la propagación electromagnética de la luz se pueden descomponer en dos tipos básicos: magnitudes radiométricas y magnitudes fotométricas:

Como magnitudes radiométricas se entienden las que rigen el fenómeno de energía electromagnética y que son medibles según recomendaciones CIE; algunas de estas magnitudes son las que se exponen a continuación pudiéndose destacar entre ellas:

- Energía radiante o radiación Q: Cantidad de energía emitida que en forma de electrones y partículas en movimiento denominados fotones, propagan la energía (J=julio).

- Flujo radiante o Potencia radiante P= dQ/dt: Cantidad de energía radiante por unidad de tiempo (W= watio= julio/ seg).

- Excitancia radiante o Emitancia radiante M= dP/dA_1: Cantidad de flujo radiante

emitido por unidad de superficie de la fuente emisora. (W/m^2.).

- Irradiancia o Absorbancia $E=dP/dA_2$: Cantidad de flujo radiante absorbido por unidad de superficie de la fuente receptora. (W/m^2.).

- Intensidad radiante $I= dP/\omega_1$: Cantidad de flujo radiante emitido en una determinada dirección por unidad de ángulo sólido de la fuente emisora. (watio/esterorradián =W/sr).

- Radiancia = $L =dI/dS. \cos \alpha$: En una determinada dirección es la cantidad de flujo radiante por unidad de ángulo sólido (Intensidad radiante) y unidad de superficie que forma un ángulo α con la normal a esa dirección. (W/m^2.sr).

- La concentración espectral de cualquiera de estas magnitudes anteriores en una determinada longitud de onda dada λ se encuentra como la cantidad de esta magnitud en un intervalo de infinitesimal que contenga esta λ, dividida por ese intervalo. Así por ejemplo la concentración espectral de la potencia radiante P se encuentra como:

$$P_\lambda = dP / d\lambda.$$

Las otras magnitudes que nos interesan son las fotométricas que se refieren a la parte de radiación visible que llamaremos energía luminosa, siendo las que nos interesan desde el punto de vista del Color, por cuanto es la porción de energía radiante que podemos percibir. Estas magnitudes fotométricas o luminosas tienen su equivalencia y correspondencia con las anteriores.

La energía luminosa es función de la energía radiante afectada por un rendimiento luminoso k_λ, para cada longitud de onda, definiéndose las magnitudes fotométricas como:

- Flujo luminoso monocromático $F_\lambda = k_\lambda \cdot P_\lambda$: función del flujo radiante

- Flujo luminoso total F: Es la integral del flujo luminoso monocromático para todas las longitudes de onda visibles, cuya unidad de medida es el lumen (lm).

$$F = K \int_\lambda P_\lambda \cdot k_\lambda \cdot dk$$

- Excitancia luminosa o Emitancia luminosa $M = dF/dA_1$.: Cantidad de flujo luminoso emitido por unidad de superficie del foco luminoso. (lm/m^2.).

- Iluminancia $E = dF/dA_2$.: Cantidad de flujo luminoso absorbido por unidad de superficie de la fuente receptora. (lm/m^2.).

- Intensidad luminosa $I = dP/\omega_1$.: Cantidad de flujo luminoso emitido en una determinada dirección por unidad de ángulo sólido del foco emisor.
  (lm/sr= candela=cd).

- Luminancia = $L = dI/dS \cdot \cos\alpha$: En una determinada dirección es la cantidad de flujo luminoso por unidad de ángulo sólido (Intensidad luminosa) y unidad de superficie que forma un ángulo α con la normal a esa dirección.
  ($cd/m^2 = lm/ m^2 .sr$).

Para poder medir estas unidades fotométricas y poderlas cuantificar en el laboratorio se necesita conocer primero las condiciones de iluminación mas habituales como son la luz del día y la luz artificial. Estos dos emisores físicos de luz son los mas cotidianos y se conocen como fuentes de energía radiante.
La Colorimetría pretende establecer unas condiciones normalizadas de simulación de estas fuentes de luz mediante la definición de iluminantes.

Las fuentes de energía radiante o fuentes de luz están constituidos por emisores físicos de luz como son una bombilla o el propio sol (Luz diurna), de modo que sea cual sea influyen en la sensación visual del color y dependerá del espectro de dicha fuente la calidad de observación del color de un objeto. Cambiando la fuente de luz cambia la sensación de color de un mismo objeto para un mismo observador.

La CIE ha definido y normalizado distintas fuentes de energía radiante y de las más importantes se presenta un breve resumen.

- Fuentes de luz natural.-

Esta fuente de energía se refiere a la luz solar diurna, que no se ha podido conseguir completamente de forma artificial, ya que la influencia de los cambios atmosféricos y la posición del sol hacen muy difícil conseguir unas condiciones naturales normalizadas y por ello no se ha reconocido todavía ninguna fuente artificial que simule completamente la luz del día.

Para conseguir si cabe simular lo mejor posible esta luz natural que es la que tenemos diariamente en la visualización del color es por lo que se ha normalizado unas condiciones de iluminación llamadas Iluminante D y que veremos más adelante.

- Fuentes de luz artificiales.-

Estas fuentes de energía están constituidas por los denominados radiadores térmicos que no son más que bombillas incandescentes que simulan precisamente la luz domestica que es a la que estamos sometidos cotidianamente en ausencia de sol dentro de nuestras casas.

En la definición de los radiadores térmicos intervienen unos factores de

comparación con respecto al radiador perfecto o radiador Plankiano desarrollado a base de fórmulas teóricas y experimentales.

El factor mas importante a considerar de un radiador térmico es la temperatura de incandescencia que se mide en grados Kelvin.

Dentro de los radiadores térmicos artificiales destacan las lámparas de tungsteno, lámparas de tungsteno-halógeno y las lámparas de filamento de tungsteno filtrado, siendo estas últimas las que más se aproximan a la simulación de la luz del día.

Otro tipo de fuentes de luz artificial son las lámparas de descarga eléctrica en las que la energía radiante se produce por el paso de la corriente eléctrica a través de un gas. Este tipo de luz es la que tenemos cuando nuestra iluminación procede del típico fluorescente.

Los Iluminantes son las condiciones teóricas de una distribución espectral determinada para simular precisamente las condiciones de luz cotidiana.

Esta condición teórica del iluminante normalizado puede no ser cumplida por ninguna fuente de luz en su totalidad pero si han conseguido fuentes de luz que se aproximen mucho al iluminante que simulan.

En la técnica de la Colorimetría la CIE ha especificado el uso de dos Iluminantes normalizados.

- Iluminante normalizado A CIE.-

Corresponde a la iluminación de una lámpara de tungsteno a una temperatura de 2856 K.

- Iluminante normalizado D65 CIE.-

Corresponde a la iluminación de la luz del día y que como se ha comentado anteriormente no se ha conseguido encontrar una fuente de luz artificial que cumpla perfectamente la distribución espectral propuesta.

En la actualidad la CIE ha establecido unos métodos y criterios que posibilitan la utilización de fuentes de luz artificiales en colorimetría con resultados satisfactorios.

3.3. TRANSFORMACIONES DE LA LUZ EN COLOR

Independientemente de las características fisiológicas del observador, la luz que incide en un objeto sufre una serie de transformaciones físicas que darán lugar a un determinado aspecto visual del mismo, produciéndose fenómenos de Absorción, Transmisión, Dispersión, Refracción y Reflexión.

Al incidir la luz sobre un objeto, parte de la radiación se reflejará, parte se absorberá y otra parte se transmitirá, dependiendo de las propias características fotométricas que produce el tipo de material, dando lugar a lo que se denomina como sensibilidad espectral del receptor, que será distinta para cada longitud de onda y que será función, entre otros, del espesor, temperatura, estado superficial, ángulos de incidencia y de observación y también será función de la propia composición espectral de la radiación.

Entre los fenómenos físicos mencionados anteriormente, el de reflexión es el que más nos interesa destacar desde el punto de vista de color del objeto, pues es el que nos determina que parte de la energía incidente se devuelve sin cambio de longitud de onda y por tanto es la parte de luz podemos percibir.

La transmisión de la luz en una superficie dependerá de varios factores dando lugar a diferentes tipos de reflexión y refracción en función del tipo de luz, del tipo de superficie y del ángulo de incidencia.

Los distintos tipos de superficie también tendrán su influencia, debiendo considerar si son transparentes, traslúcidas u opacas, a través de sus coeficientes de reflexión y de refracción.

Se pueden destacar dos tipos fundamentales de reflexión: reflexión especular y reflexión difusa.

La reflexión especular es aquella producida por un punto de luz que incide en una superficie y a modo de espejo refleja dicho punto de luz en forma de brillo que dependerá del ángulo de inclinación, como por ejemplo la reflexión directa de la luz procedente de una bombilla o del sol.

La reflexión difusa es aquella producida por la luz ambiental o luz indirecta, transmitiéndose en todas las direcciones sin producir sombras e iluminando el objeto de forma suave, como por ejemplo los objetos de una habitación con la ventana abierta orientada al norte.

Tampoco deberemos olvidar la reflexión ambiental que es la producida por una fuente de luz distribuida generando un determinado sombreado, como por ejemplo la iluminación de un fluorescente o la iluminación producida en un día nublado.

Para cada tipo de superficie tendremos un factor de reflectancia k que es el cociente entre el flujo luminoso radiante reflejado y el flujo luminoso radiante incidente en una unidad de superficie reflectante perfecta. Este factor es importante en el estudio de color pues es el que delimitará la parte de luz visible que llega al observador y producirá la sensación de color del objeto y que será el que nos permite identificar al mismo.

Dentro del campo de la Informática Gráfica, los primeros estudios de iluminación consideraron un modelo simple de iluminación para calcular la cantidad de luz absorbida, reflejada o transmitida (refractada), considerando únicamente la influencia de los focos de luz sobre los objetos.

El concepto de partida para calcular la iluminación de un objeto, está basado en la *ley del coseno de Lambert*, de modo que la intensidad reflejada en una superficie considerada perfectamente difusa se expresa como:

$$I = I_L . k_d . \cos\theta$$

siendo k_d el coeficiente de reflexión difusa, $0 \leq k_d \leq 1$ (Fig. 3.1).

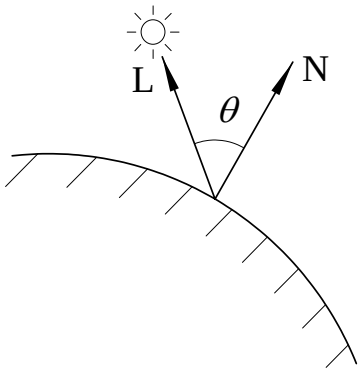


Fig. 3.1 - Intensidad reflejada en una superficie difusa perfecta.

Considerando la luz ambiental como una fuente de luz distribuida, la ecuación anterior se convierte en:

$$I = I_a . k_a + I_L . k_d . \cos\theta$$

Añadiendo la atenuación de la luz en función de la distancia del objeto al foco tendremos:

$$I = I_a . k_a + \frac{I_L . k_d . \cos\theta}{d + K}$$

donde K es una constante.

Considerando la luz reflejada especularmente (Fig. 3.2) cuya expresión es:

$I_s = I_L . w(i,\lambda) \cos^n\alpha$ es decir: $I_s = I_L . k_s . \cos^n\alpha$

siendo k_s el coeficiente de reflexión especular $0 \leq k_s \leq 1$.

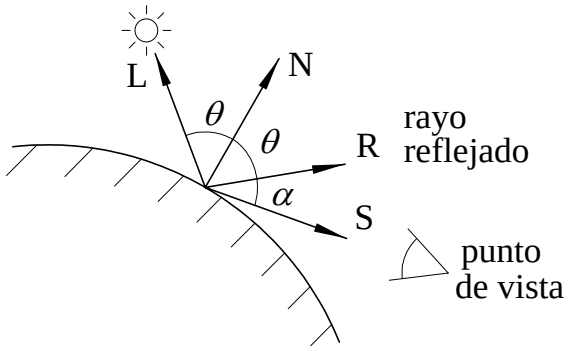


Fig. 3.2 - Intensidad reflejada especularmente.

La ecuación de iluminación, denominada también función *shading*, queda:

$$I = I_a . k_a + \frac{I_L}{d + K}(k_d . \cos\theta + k_s . \cos^n\alpha)$$

que puede extenderse para múltiples puntos de luz.

$$I = I_a . k_a + \sum_{j=1}^{m} \frac{I_{Lj}}{d + K}(k_d . \cos\theta_j + k_s . \cos^n\alpha_j)$$

Considerando los vectores unitarios $\overline{R}, \overline{S}, \overline{N}, \overline{L}$, podemos calcular el coseno de un ángulo como el producto escalar de esos vectores unitarios que lo forman:

$$\cos\theta = \overline{N} * \overline{L}$$
$$\cos\alpha = \overline{R} * \overline{S}$$

Así pues la formula de la iluminación global para una sola fuente de luz se transforma en:

$$I = I_a . k_a + \frac{I_L}{d + K}\left[k_d (\overline{N} * \overline{L}) + k_s (\overline{R} * \overline{S})^n\right]$$

siendo pues necesario el cálculo de dichos vectores unitarios.

3.4. VISION DEL COLOR

La visualización real de una imagen no es más que una percepción sensorial de la realidad, tratando de emular lo que nuestros ojos ven en la realidad. Así pues haremos una breve reseña de la complejidad del sistema visual humano (observador), en contraste con la aparente sencillez que se nos presenta. La implementación de un sistema de tratamiento de imágenes por computadora implica la comprensión y el uso de muchas de las características y parámetros de nuestro sistema receptor de imágenes. Así pues, describiremos algunos de los conceptos fundamentales del sistema de visión humana.

Como es sabido, el ojo humano es el sistema óptico más complejo y perfecto conocido hasta ahora y que cualquier sistema artificial trata de emular. Aunque para nuestros propósitos no tenga mucho interés conocer con detalle la anatomía del globo ocular, sí que resulta práctico conocer sus características principales para entender su complejo funcionamiento (Fig. 3.3).

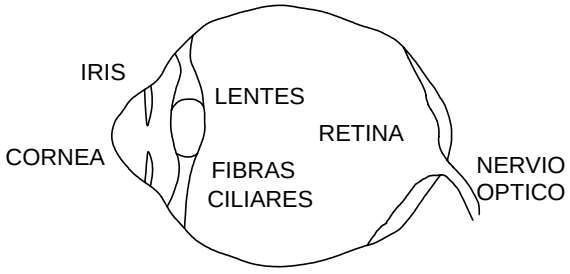


Fig. 3.3- Representación esquemática del ojo humano.

Fisiológicamente el ojo humano es como una esfera ligeramente ovalada, recubierta por fuera por la córnea y la esclerótica y por dentro por la coroides y la retina. La coroides acaba en el cuerpo ciliar y en el iris. Este último funciona como un diafragma, limitando la cantidad de luz que entra en el globo ocular a través de un orificio central,

denominado pupila. Detrás del iris se encuentra el cristalino, formado por agua, grasas y proteínas, que funciona como una lente convergente modificable gracias a las tensiones de las fibras ciliares, enfocando sobre la retina la luz procedente del exterior en la parte posterior del globo. En realidad, la córnea presenta un primer efecto de convergencia a la radiación, actuando como una superficie óptica activa.

En la retina coexisten dos tipos de receptores: los conos y los bastones. Los primeros, en cantidad de unos seis a siete millones por globo ocular, son sensibles a las diferentes longitudes de onda del espectro visible, es decir, a los diferentes colores. Los bastones, en cantidad mucho mayor, unos cien millones por globo, son sensibles a diferentes niveles de luminosidad y su sensibilidad es mucho mayor que la de los conos. Por este motivo, podemos distinguir formas y colores en condiciones adversas de luz.

Otro de los factores que intervienen en el tratamiento de imágenes es el conocimiento de la luz propiamente dicha que es una radiación electromagnética cuya longitud de onda se expresa en nanómetros (nm) o micrómetros (μm). La longitud de onda λ, del espectro visible toma valores aproximadamente entre:

$$360 \text{ nm} \leq \lambda \leq 780 \ \mu\text{m}$$

Para cada longitud de onda corresponde un color distinto. De todos modos el ojo percibe cualquier objeto gracias a la luz reflejada respecto de la que ha incidido. En general, la luz reflejada para cada longitud de onda viene dada por:

$$I(\lambda) = \rho(\lambda).L(\lambda)$$

Donde $L(\lambda)$ y $\rho(\lambda)$ son la radiación y el coeficiente de reflectividad del objeto respectivamente, para esta radiación.

La luminosidad o energía emitida por unidad de área en una unidad de ángulo sólido es la magnitud fotométrica que corresponde a la sensación de claridad o brillantez. Esta

energía es la percibida por el ojo y depende de su respuesta para cada longitud de onda. Esta respuesta no es lineal para distintas intensidades luminosas sino que para cambios de intensidad corresponden alteraciones casi logarítmicas en la respuesta del ojo, cuya gráfica se observa en Fig. 3.4.

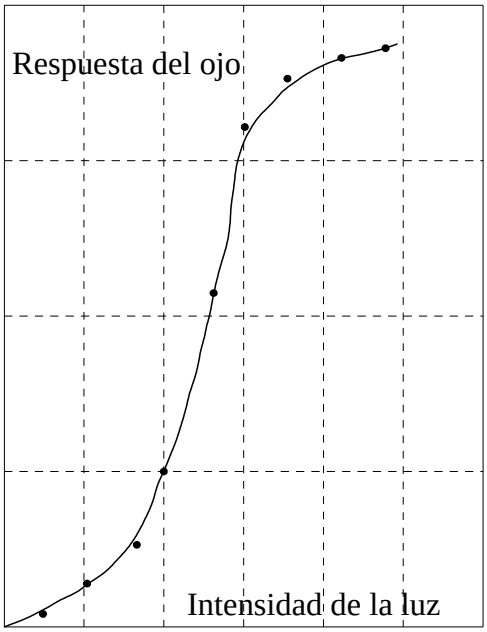


Fig. 3.4 - Curva de respuesta del ojo humano a variaciones de la intensidad de la luz.

El concepto de contraste nace de la diferencia entre los valores de luminosidad de dos objetos contiguos; los brillos aparentes (luminosidad subjetiva) pueden ser considerablemente distintos a variaciones importantes de la luminosidad del entorno.

Conscientes de la complejidad del fenómeno humano y de la transmisión de la luz como una radiación electromagnética, la transmisión de la luz en una superficie dependerá de varios factores dando lugar a diferentes tipos de reflexión y refracción en función del tipo de luz, del tipo de superficie y también del ángulo de incidencia.

Desde siempre se han desarrollado diversas teorías sobre el comportamiento del ojo humano con respecto a la visión del color siendo la mas ampliamente aceptada la teoría

tricromática o de los tres colores primarios en la que se indica que es posible obtener cualquier color a partir de la mezcla aditiva de tres luces monocromáticas adecuadas, con la única condición de que cualquiera de estos tres colores primarios no pueda obtenerse por la mezcla de los otros dos.

De entre las diversas combinaciones posibles de tres colores básicos se han elegido los tres colores cuyas longitudes de onda están mas separadas siendo el Rojo, el Verde y el Azul los tres colores primarios aceptados constituyendo el modelo RGB (Red, Green, Blue).

3.5. COLORIMETRIA

La Colorimetría es la rama de la ciencia que estudia numéricamente la sensación humana que se produce bajo la influencia del color.

Esta especificación del color no puede considerarse como una ciencia exacta a pesar de que se han establecido unos observadores tipo y unas fuentes de luz estándar que permiten normalizar unos procedimientos y unos resultados, de forma que puedan ser aceptados genéricamente.

Por ello se han propuesto diversas teorías o modelos de color normalizados. Con el paso de los años la CIE ha ido revisando y modificando sus recomendaciones adaptándose a los nuevos avances tecnológicos que se han ido produciendo de acuerdo con modelos matemáticos que reproduzcan lo mas fielmente posible los resultados experimentales.

3.5.1. Modelos de color normalizados

MODELO TRIESTIMULO (CIE 1931)

En este modelo se definen tres valores XYZ triestimulo como una cuantificación de la respuesta del ojo humano en el rango de longitudes de onda desde 360 nm hasta 780 nm con intervalos de 5 nm. si bien en la práctica se acostumbran a tomar intervalos de 20 nm. de acuerdo con las siguientes fórmulas:

$$X = k \sum_{360}^{780} P_\lambda \, . \, \bar{x}(\lambda) \, d\lambda$$

$$Y = k \sum_{360}^{780} P_\lambda . \bar{y}(\lambda)\, d\lambda$$

$$Z = k \sum_{360}^{780} P_\lambda . \bar{z}(\lambda)\, d\lambda$$

- P_λ es función de la energía radiante del objeto que es: $P_\lambda = R(\lambda) . S(\lambda)$.

- $R(\lambda)$ funcióm de las propiedades de reflectancia espectral del objeto según sus características geométricas.

- $S(\lambda)$ es la sensibilidad espectral o concentración de potencia radiante sobre el objeto.

- $\bar{x}(\lambda)$, $\bar{y}(\lambda)$, $\bar{z}(\lambda)$ son las funciones del observador normalizado o función de igualación del color.

Estos valores triestímulo X Y Z identifican numéricamente al estímulo que produce un color en un observador. Dado que la representación gráfica de estos tres valores sería una gráfica tridimensional de más difícil representación que una gráfica bidimensional se han establecido otras coordenadas de cromaticidad de acuerdo con:

$$x = \frac{X}{X + Y + Z}$$

$$y = \frac{Y}{X + Y + Z}$$

$$z = \frac{Z}{X + Y + Z}$$

Para construir una gráfica bidimensional se adopta Y =cte. y se consideran únicamente

las coordenadas (x,y) dado que x+y+z = 1.

El resultado de aplicar todos los colores cabe en un área limitada por una curva en forma de herradura para cada valor de Y, siendo éste el principal inconveniente de este modelo, ya que para cada valor de Y tenemos una gráfica distinta.

El inconveniente de esta gráfica es que no es un modelo euclidiano y para paliar este inconveniente se han desarrollado otros modelos matemáticos mas uniformes, de modo que las diferencias numéricas de color correspondan diferencias sensoriales de color.

MODELO CIELuv (CIE 1976)

Este modelo sustituye al que se especificó en CIE 1960 y CIE 1964 (modelo U V W) en el que se intenta perfeccionar el modelo triestimulo construyendo una nueva gráfica bidimensional a partir de los tres valores XYZ pero con otra definición de coordenadas.

Inicialmente se definieron tres variables U V W que intervenían en la formula ΔE de diferenciación de color de dos materiales y que posteriormente fueron sustituidas (CIE 1976) por $L^* u^* v^*$ quedando del siguiente modo:

$$L^* = 25\left(100\frac{Y}{Y_0}\right)^{1/3} - 16$$

$$u^* = 13\,L^*\left(u' - u'_0\right)$$

$$v^* = 13\,L^*\left(v' - v'_0\right)$$

siendo:

$$u' = \frac{4\,X}{X + 15\,Y + 3\,Z} \qquad v' = \frac{9\,Y}{X + 15\,Y + 3\,Z}$$

$$u'_0 = \frac{4\,Y_0}{X_0 + 15\,Y_0 + 3\,Z_0} \qquad v'_0 = \frac{9\,Y_0}{X_0 + 15\,Y_0 + 3\,Z_0}$$

Los valores X_0 , Y_0 , Z_0 corresponden a los valores triestímulo del blanco perfecto correspondiente al iluminante normalizado utilizado.

La formula de diferencia de color entre dos estímulos distintos se calcula según:

$$\Delta E^* = \sqrt{\left(\Delta L^*\right)^2 + \left(\Delta u^*\right)^2 + \left(\Delta v^*\right)^2}$$

Con este modelo de color la dimensión L^* especifica la luminosidad y para cada L^* constante tenemos una gráfica de cromaticidad (u',v') distinta, siendo una transformación proyectiva de la gráfica anterior según (x,y).

Como se dijo al final del apartado anterior el diagrama (x,y) no es un modelo euclidiano porque las distancias numéricas son distintas para sensaciones visuales de diferencias iguales de color. El motivo de la nueva gráfica (u',v') es precisamente el conseguir esta diferenciación de colores.

Este modelo CIELuv está considerado útil por los expertos para la visualización de mezclas de fuentes de luz aditivas.

MODELO CIELab (CIE 1976)

Este modelo está considerado por algunos autores como una modificación del CIELuv

expuesto anteriormente y su principal utilidad se encuentra en la visualización de mezclas substrativas.

El modelo CIELab se ha implantado en el campo textil que nos ocupa, quedando contemplada su aplicación en las Normas UNE 40-081 y UNE 40-435 que corresponden a las especificaciones dadas por la "COMMISSION INTERNATIONALE DE L'ECLAIRAGE" (CIE).

Como podremos contemplar en la formulación de este modelo, la función L^* es idéntica al modelo anterior, y la diferencia estriba en una nueva definición de coordenadas (a^*,b^*) que sustituyen a las coordenadas (u',v') del modelo anterior.

Igualmente este modelo contempla la función ΔE de diferenciación de color de dos materiales, como veremos a continuación.

La formulación de este modelo CIELab está basado en un modelo propuesto por Hunter (HUNTERLab) que no se aceptó en su totalidad pero del que se aprovechó sus fundamentos, adoptándose las fórmulas siguientes:

$$L^* = 25\left(100\frac{Y}{Y_0}\right)^{1/3} - 16 = 116\left(\frac{Y}{Y_0}\right)^{1/3} - 16$$

$$a^* = 500\left[\left(\frac{X}{X_0}\right)^{1/3} - \left(\frac{Y}{Y_0}\right)^{1/3}\right]$$

$$b^* = 200\left[\left(\frac{Y}{Y_0}\right)^{1/3} - \left(\frac{Z}{Z_0}\right)^{1/3}\right]$$

Como antes, los valores X_0 , Y_0 , Z_0 corresponden a los valores triestímulo del blanco perfecto correspondiente al iluminante normalizado utilizado.

La formula de diferencia de color entre dos estímulos distintos se calcula según:

$$\Delta E^* = \sqrt{\left(\Delta L^*\right)^2 + \left(\Delta a^*\right)^2 + \left(\Delta b^*\right)^2}$$

La dimensión L^* especifica la luminosidad y para cada L^* constante tenemos una gráfica de cromaticidad (a^*,b^*) distinta.

Además de los valores L^*, a^*, b^*, también se usan las coordenadas polares: C^* (Cromaticidad) y H^* (Tono) como vemos en la gráfica siguiente:

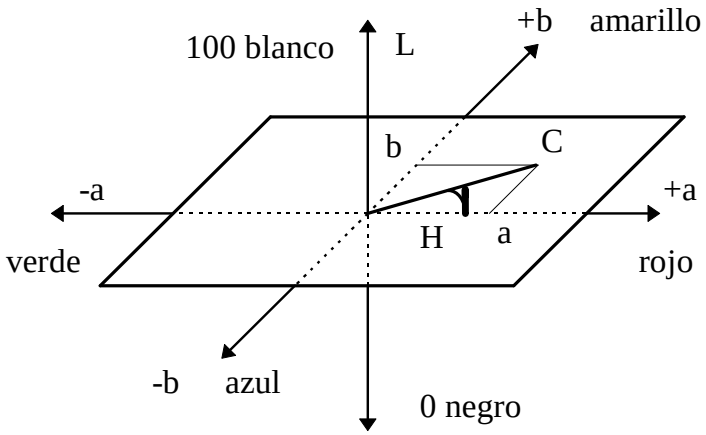


Fig. 3.5

Lo normal es dar las coordenadas L a b pero es mas intuitivo dar los valores L C H que son: - Luminosidad L

- Cromaticidad: $C = \sqrt{a^2 + b^2}$

- Tono: $H = \text{arcotan}\ \frac{b}{a}$

Las normas UNE mencionadas anteriormente se refieren a la determinación de las diferencias de color en el campo textil, pues la medición de un color siempre se realiza por diferencia de color con respecto a un color patrón, por lo que es mas importante calcular las diferencias entre variables que la propia variable en sí.

Además de usar la diferencia de Luminosidad (ΔL^*), tanto para el modelo CIELuv como para el modelo CIELab se usan otras variables de correlación CIE 1976 como son:

- Diferencia de Cromaticidad (ΔC^*).
- Diferencia de Saturación (ΔS^*).
- Diferencia de Tono (ΔT^*).

Estas diferencias se definen del siguiente modo en el modelo CIELab, considerando el subíndice “s” a la muestra estándar o patrón y “p” a la muestra particular a contemplar:

$$\Delta L^* = L_p^* - L_s^*$$

$$\Delta C^* = \sqrt{\left(a_p^* - a_s^*\right)^2 + \left(b_p^* - b_s^*\right)^2}$$

$$\Delta S^* = \sqrt{\left(a_p^*\right)^2 + \left(b_p^*\right)^2} - \sqrt{\left(a_s^*\right)^2 + \left(b_s^*\right)^2}$$

$$\Delta H^* = \sqrt{\left(\Delta C^*\right)^2 - \left(\Delta S^*\right)^2} == \left[\left(\Delta E^*\right)^2 - \left(\Delta L^*\right)^2 - \left(\Delta C^*\right)^2\right]$$

La CIE utiliza el símbolo ΔH^* en lugar de ΔT^* que usa la UNE al usar la acepción inglesa de “hue” para el tono.

Hemos de destacar que CIE recomienda usar cualquiera de estos modelos CIELuv o CIELab pero no especifica claramente cual es la aplicación mas apropiada a cada uno de ellos.

Intuitivamente podemos constatar que cuando hemos de cuantificar un color es mas fácil asociar las características sensitivas de luminosidad, cromaticidad y tono de un color, que interpretar unas variables abstractas X Y Z que se refieren al contenido en rojo, verde y azul de un determinado color.

Estas recomendaciones o modelos de color no son las únicas formas de identificar un color, sino que diversos autores que han experimentado la colorimetría, han establecido unos sistemas de ordenación de colores a base de muestras físicas de color ordenadas que sean más fáciles de manipular que estas recomendaciones

De entre los sistemas de ordenación de colores podemos destacar como mas importantes por la repercusión de su uso industrial a los siguientes:

- Sistema de Ostwald.
- Sistema de Munsell
- Sistema DIN
- Sistema Plaza
- Sistema Sueco
- Sistema Osa

3.5.2 Modelos de color simplificados

Los modelos de color normalizados expuestos anteriormente son recomendaciones teóricas para la medición del color, basados en funciones matemáticas, que han sido simplificados para su implantación en los gráficos por computadora y que son de uso general en esta rama de la ciencia.

La elección adecuada de un espacio de color condiciona la representación cromática de los resultados. Para la mayoría de las aplicaciones es más que suficiente utilizar el modelo simplificado RGB, aunque hemos de indicar que los modelos de color simplificados se pueden clasificar en dos tipos:

- Los que están basados en atributos sensoriales como son el tono, matiz, saturación, etc. habiéndose desarrollado los modelos HSV, HLS,...

- Y los que se basan mas en las características de los dispositivos reproductores del color azul, verde, etc., (como son las pantallas y las impresoras) escogiendo los modelos RGB, CMY,...

MODELO R G B

Este modelo se puede considerar como una transformación lineal del modelo triestimulo XYZ, simplificando dichas funciones a tres variables mas sensitivas como son los porcentajes de rojo, verde y azul.

Estas tres variables RGB (rojo, verde, azul), forman un modelo tridimensional en forma de cubo basado en la teoría tricromatica de la visión de color.

Este modelo está basado en la mezcla aditiva de los tres colores primarios para obtener cualquier color; es decir para producir un determinado color se suman las contribuciones individuales de cada primario.

Para convertir los colores especificados en un determinado tipo de monitor se emplean las conversiones de un espacio de color RGB al espacio normal de colores X Y Z.

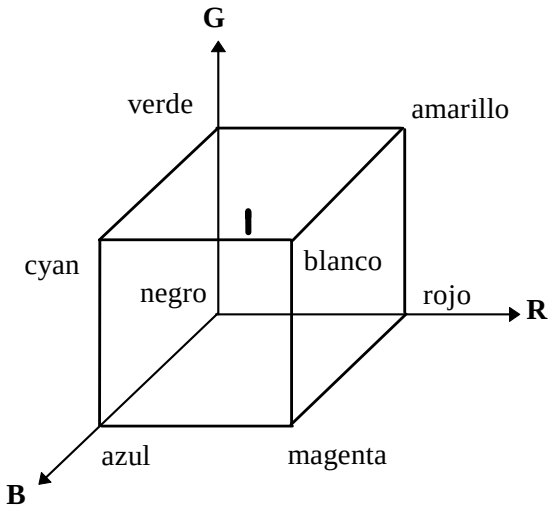


Fig. 3.6

MODELO C M Y (C M Y K)

Este modelo es el complementario del anterior RGB considerando tres primarios sustractivos Cyan Magenta y Amarillo en cuya ausencia se obtiene el color blanco tal como se aprecia en la gráfica siguiente:

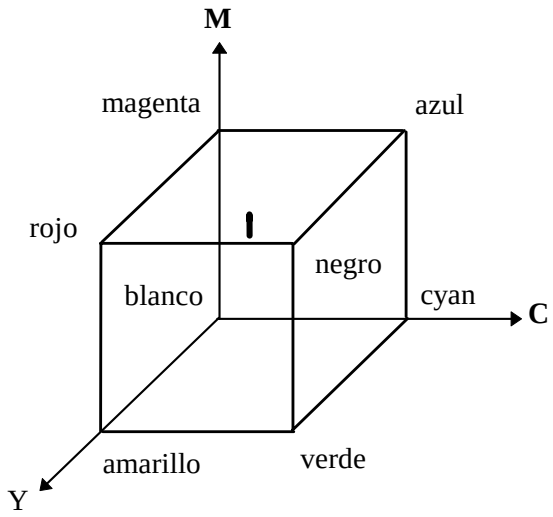


Fig. 3.7

Este modelo adquiere su importancia en los dispositivos de impresión de colores.

Otro modelo de color complementario de este es el CMYK en el que se usa el negro como cuarto color K de acuerdo con la siguiente relación:

$K = \min(C,M,Y)$

$C = C - K$

$M = M - K$

$Y = Y - K$

MODELO H S V

Este modelo aplica la descripción del color en los términos de Tono (H), Saturación (S) y valor (V) , también conocido como modelo HSB siendo B la brillantez. En este modelo el sistema de coordenadas es cilíndrico y el espacio de color queda determinado por una pirámide hexagonal tal como se ve en la figura siguiente:

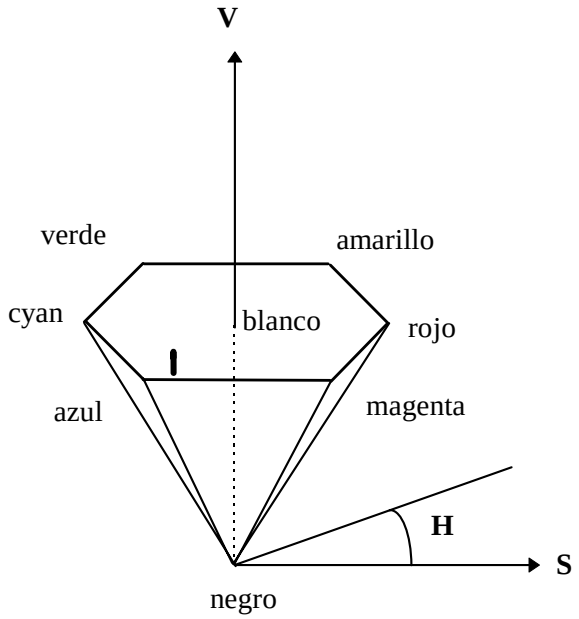


Fig. 3.8

Este modelo también tiene una analogía con el modelo, RGB ya que al observar el cubo según una proyección que se transforme en hexágono, este coincide con la base de la pirámide hexagonal. La parte superior de la pirámide hexagonal corresponde a la proyección que se puede ver a lo largo de la diagonal principal del cubo RGB.

Este modelo se adapta perfectamente al trabajo de tintas, sombras y tonos.

MODELO H L S

Este modelo similar al anterior utilizando las variables de Tono (H), Luminosidad (L) y Saturación (S), en el que se ha sustituido la V por L.

En este modelo se percibe la sensación de más oscuro o mas claro en lugar del blanco, dando lugar a una doble pirámide hexagonal.

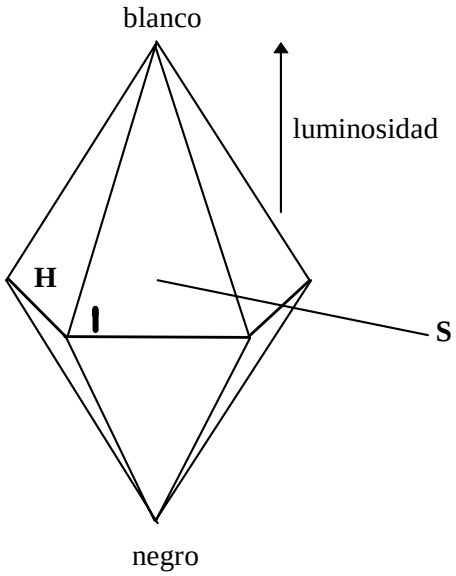


Fig. 3.9

3.6. INFORMATICA DEL COLOR

La mayoría de los monitores están basados en un diseño estándar del tubo de rayos catódicos (TRC), en el que los electrones denominados también rayos catódicos son emitidos desde un dispositivo hacia la pantalla atravesando diversos sistemas más o menos sofisticados para su enfoque y deflexión del rayo hacia un punto determinado de la pantalla que está recubierta de una sustancia fosfórica que iluminará el pixel o punto de pantalla. Los monitores a color utilizan una combinación de sustancias fosfóricas que emiten luz en diferentes colores siendo el método mas utilizado el de usar el modelo de color RGB.

La imagen final que se obtiene en un programa gráfico está referenciada a los pixels del monitor, y es el ordenador el que debe facilitar la información necesaria del color asignado a cada pixel, codificando dicha información de acuerdo a las características del monitor que interpreta normalmente el espacio RGB.

Fundamentalmente existen dos sistemas de codificar el color refiriéndose a sistemas de color indexado o sistemas de color verdadero.

El sistema de color indexado se basa en codificar en cada pixel una tabla de color llamada LUT (look up table), de forma que codifica en dicha tabla la información RGB que se quiere representar. Dependiendo del tipo de tarjeta gráfica que dispongamos podremos codificar 8, 16 o 24 bits por pixel, lo que nos dará la posibilidad de simultanear más o menos colores a la vez.

El sistema de color verdadero codifica directamente el valor RGB deseado. Este sistema dispone de una paleta de colores más pequeña que el anterior par el mismo numero de bits de la tarjeta gráfica.

3.6.1. Tratamiento del color

Toda imagen que aparece en el monitor del ordenador está almacenada en algún lugar de la memoria en forma de unos y ceros; cada cierto tiempo (veces por segundo) se va regenerando y mostrando en dicho monitor o pantalla.

Para poder almacenar una imagen en la memoria del ordenador, la pantalla se divide en tiras horizontales de un pixel de alto, empezando normalmente por la esquina superior izquierda, recorriendo esta primera fila hasta el extremo derecho, saltando a continuación a la segunda fila, repitiendo el proceso y así sucesivamente hasta la ultima fila; de esta forma se almacena una pantalla que consideramos de dos dimensiones en una secuencia lineal que podemos almacenar en memoria.

Este conjunto de bits se denomina mapa de la pantalla y es la que incluye toda la información de la imagen que tenemos en pantalla.

Con esta explicación de asignar unos y ceros a cada pixel de pantalla se entiende fácilmente la manipulación de una imagen monocromática, pues los pixels estarán simplemente encendidos o apagados.

El tratamiento informático del color tiene el mismo fundamento, pero no es suficiente indicar si el pixel o punto de pantalla está encendido o apagado sino que además necesita más información acerca del color que debe mostrar.

Para el caso monocromático un punto no necesita tener asociado mas que 1 bit (0 o 1), Si nuestra tarjeta gráfica permite asociar 2 bits por pixel podremos codificar y asignar a cada punto hasta 4 colores. Con 3 bits por pixel tendremos la posibilidad de 8 colores, con 4 bits aumenta a 16 colores, con 8 bits tendremos 256 colores y así sucesivamente. Esto nos indica la importancia que tiene el hardware disponible para tener una mayor

posibilidad de numero de colores dependiendo la cantidad de colores simultáneos a las prestaciones de la tarjeta que tengamos asociada a nuestro monitor.

Como se ha mencionado anteriormente el tratamiento habitual del color en informática gráfica se basa en el sistema RGB identificando un color por su componentes de rojo, verde y azul que se manipulan por separado.

3.6.2. Conversión del color

En este apartado se van a exponer las fórmulas para conversión entre diferentes espacios de color:

A partir de las coordenadas triestímulo XYZ (CIE 1931) podemos obtener los valores RGB de la pantalla a partir de un matriz T_{RGB} de transformación, basada en las coordenadas cromáticas de los fósforos del monitor y de su punto blanco:

$$(R, G, B) = \left[T_{RGB}\right](X, Y, Z)$$

En esta transformación nos encontramos con el problema de que no todos los colores CIE pueden ser representados en el espacio RGB del monitor, por lo que será necesario encontrar el vecino más próximo.

Esta formula de conversión se puede expresar mas completamente del siguiente modo:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} T_{RX} & T_{RY} & T_{RZ} \\ T_{GX} & T_{GY} & T_{GZ} \\ T_{BX} & T_{BY} & T_{BZ} \end{bmatrix} \times \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

Los elementos de la matriz T_{ij} son experimentales y todavía que se sepa no tenemos unos valores aceptados por todos.

El mismo problema nos aparece cuando queremos hacer la conversión de CMY a RGB en el que la transformación de forma simple sería a partir de la fórmula siguiente, pero la dificultad está en considerar adecuadamente los valores de la matriz de transformación:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} T_{RC} & T_{RM} & T_{RY} \\ T_{GC} & T_{GM} & T_{GY} \\ T_{BC} & T_{BM} & T_{BY} \end{bmatrix} \times \begin{bmatrix} C \\ M \\ Y \end{bmatrix}$$

En este caso podemos hacer una rápida aproximación si tenemos en cuenta la propia definición de los espacios RGB y CMY, ya que cada valor primario puede sustraerse de la luz blanca de coordenadas (R,G,B)=(1,1,1) dando la correspondiente coordenada CMY o lo que es lo mismo CMY = (1,1,1) - RGB.

Otras conversiones interesantes son las aproximaciones de los espacios de color HSV y HLS a partir de las coordenadas en el espacio RGB:

Espacio HSV:

$$H(r,g,b) = \begin{cases} \pi / 3(g - b) / \Delta(r,g,b) & \text{si } r = \max\{r,g,b\} \\ \pi / 3(2 + (b - r) / \Delta(r,g,b)) & \text{si } g = \max\{r,g,b\} \\ \pi / 3(4 + (r - g) / \Delta(r,g,b)) & \text{si } b = \max\{r,g,b\} \end{cases}$$

$$\Delta(r,g,b) = \max\{r,g,b\} - \min\{r,g,b\}$$

$$S(r,g,b) = \frac{\max\{r,g,b\} - \min\{r,g,b\}}{\max\{r,g,b\}}$$

$$V(r,g,b) = \max\{r,g,b\}$$

Espacio HSL: El cálculo de H es el mismo que el anterior

$$H(r,g,b) = \begin{cases} \pi / 3(g - b) / \Delta(r,g,b) & \text{si } r = \max\{r,g,b\} \\ \pi / 3(2 + (b - r) / \Delta(r,g,b)) & \text{si } g = \max\{r,g,b\} \\ \pi / 3(4 + (r - g) / \Delta(r,g,b)) & \text{si } b = \max\{r,g,b\} \end{cases}$$

$$\Delta(r,g,b) = \max\{r,g,b\} - \min\{r,g,b\}$$

$$S(r,g,b) = \begin{cases} \dfrac{\max\{r,g,b\} - \min\{r,g,b\}}{\max\{r,g,b\} + \min\{r,g,b\}} & \text{si } L \le 0.5 \\ \dfrac{\max\{r,g,b\} - \min\{r,g,b\}}{2 - (\max\{r,g,b\} + \min\{r,g,b\})} & \text{si } L > 0.5 \end{cases}$$

$$L(r,g,b) = \frac{\max\{r,g,b\} + \min\{r,g,b\}}{2}$$

CAPITULO 4

TRANSFORMACIONES GEOMETRICAS

4.1. INTRODUCCION

Las rutinas gráficas que debe incorporar cualquier programa informático de diseño han de estar implementadas de forma que permitan mover los objetos de sitio, orientarlos en una determinada dirección, cambiarlos de tamaño, etc. Estos movimientos se realizan a través de las denominadas transformaciones geométricas y que básicamente se refieren a los conceptos de traslación, giro, simetría y escalado.

Para el estudio de estas transformaciones geométricas y su posterior desarrollo en forma de algoritmos deberemos buscar un modelo matemático adecuado que esté lo suficientemente unificado y resuelto algebraicamente.

Entre los diversos modelos matemáticos que podían solucionar estas cuestiones, la experiencia ha demostrado que el modelo mas adecuado para manejar los objetos gráficos es el modelo matricial por cuanto es el de mejor implantación algoritmica y que además una sola matriz de cálculo permite obtener las coordenadas finales de cualquier sucesión de transformaciones evitando múltiples cálculos intermedios.

En este capítulo se presenta un breve estudio matemático de los conceptos de vectores y matrices con las operaciones fundamentales que se van a necesitar para su aplicación en un programa informatico de diseño; a continuación se desarrollarán matricialmente los movimientos básicos de traslación, giro, simetría y escalado de forma unitaria, para obtener finalmente matrices compuestas que incluyan todos estos movimientos en una sola matriz de transformación.

4.2. VECTORES

En cualquier espacio de n dimensiones los parámetros iniciales necesarios para trabajar en el mismo están constituidos: por un origen de coordenadas y por los vectores unitarios en la dirección de los ejes principales, lo que constituirá el sistema básico de referencia.

Para explicar los conceptos fundamentales nos vamos a referir a un espacio bidimensional y de ejes principales perpendiculares entre sí, este sistema ortogonal constituye el estudio del plano euclídeo al que aplicaremos los principales movimientos que se utilizan en los gráficos por computadora.

La posición de un punto cualquiera P_1 queda determinada por sus coordenadas x_1 y_1 referidas a los vectores unitarios de los dos ejes principales del sistema de coordenadas o sistema de referencia O X Y, constituyendo el vector de posición OP_1 referido al origen O.

$$\overline{OP_1} = (\overline{x_1}, \overline{y_1}) = (x.\bar{u}, y.\bar{v})$$

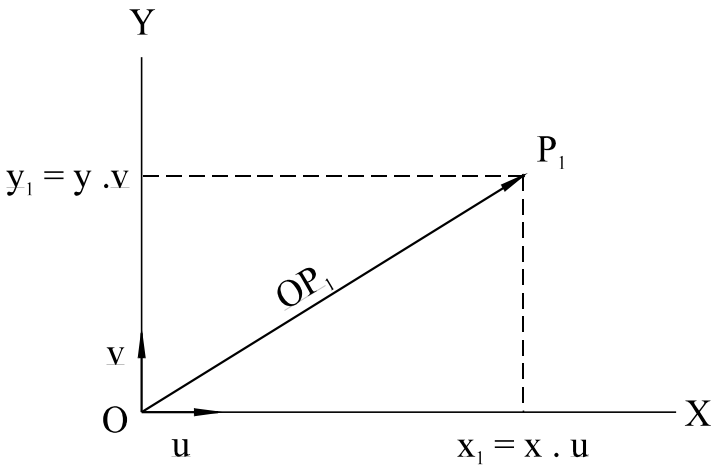


Fig. 4.1

Este punto P_1 se identifica únicamente por sus coordenadas cartesianas (x, y) que identifican al vector de posición $\overline{OP_1}$ pudiéndose escribir de forma matricial con una columna o con una fila.

$$P_1 = \begin{bmatrix} x \\ y \end{bmatrix} \qquad P_1 = \begin{bmatrix} x & y \end{bmatrix}$$

Con este tipo de vectores se pueden realizar algunas operaciones matemáticas elementales:

- La suma de dos vectores (regla del paralelogramo) que se realiza componente a componente dando lugar a un nuevo vector:

$$\overline{m} + \overline{n} = \overline{p} = \begin{bmatrix} 2 \\ 3 \end{bmatrix} + \begin{bmatrix} 4 \\ 5 \end{bmatrix} = \begin{bmatrix} 6 \\ 8 \end{bmatrix}$$

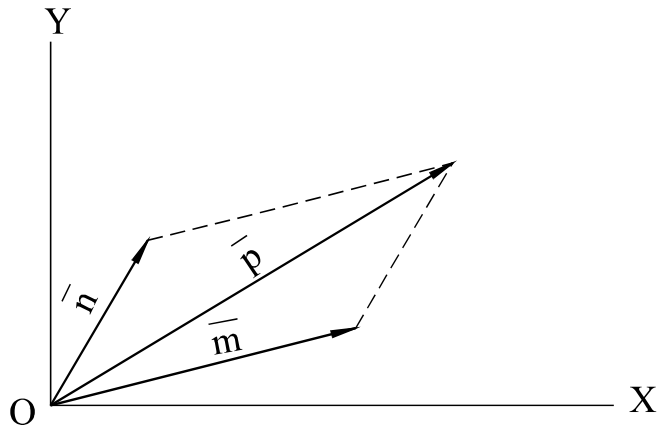


Fig. 4.2

- El producto escalar de un vector por un numero real da lugar a otro vector múltiplo del primero

$$\overline{m} \times R = \overline{p} = \begin{bmatrix} 2 \\ 3 \end{bmatrix} \times 4 = \begin{bmatrix} 8 \\ 12 \end{bmatrix}$$

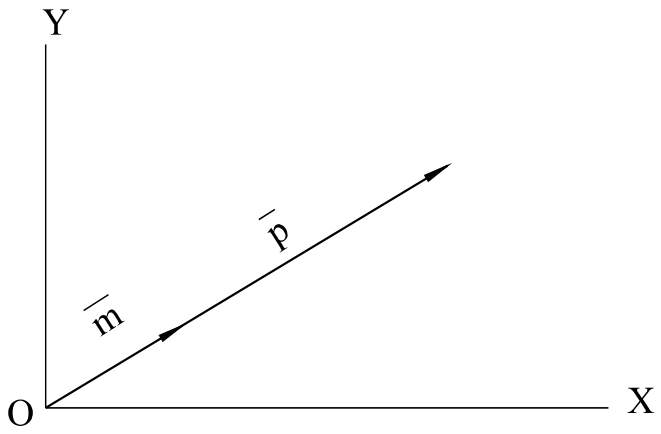


Fig. 4.3

- El producto interior o producto punto de dos vectores da lugar a un numero real.

$$\overline{m} \times \overline{n} = \begin{bmatrix} 2 \\ 3 \end{bmatrix} \times \begin{bmatrix} 4 \\ 5 \end{bmatrix} = 2 \times 4 + 3 \times 5 = 23$$

Esta operación de producto interior toma su importancia cuando realizamos la operación $\sqrt{\overline{m} \times \overline{m}}$ que no es mas que la longitud del vector o dicho de otro modo es la distancia de este punto al origen de coordenadas (módulo).

$$\sqrt{\overline{m} \times \overline{m}} = \sqrt{\begin{bmatrix} 3 \\ 4 \end{bmatrix} \times \begin{bmatrix} 3 \\ 4 \end{bmatrix}} = \sqrt{3 \times 3 + 4 \times 4} = 5$$

Las transformaciones geométricas que se van a desarrollar serian de difícil interpretación y manejo si usáramos únicamente el concepto de vectores, dadas las limitaciones que nos imponen sus operaciones y por la dependencia de estar utilizando siempre el origen de coordenadas en la manipulación del vector; por ello usaremos el concepto de matriz que es mas amplio para manipular dichos vectores ya que nos permite resolver los sistemas de ecuaciones.

4..3. MATRICES

Básicamente una matriz es un concepto más amplio de vector pues se trata de una sucesión de números dispuestos de forma vertical y horizontal formando filas y columnas a las que les podremos identificar como matrices asociadas a un movimiento y poder realizar las transformaciones geométricas que nos interese y que se desarrollaran a continuación.

Antes de considerar las operaciones básicas que podemos realizar con las matrices se entiende que un vector no es mas que una matriz de una sola columna.

La convención que se va a seguir para identificar los elementos de una matriz serán los siguientes: el primer subíndice se referirá a la fila empezando en 1 y el segundo subíndice se referirá a la columna que también empezara en 1.

Así pues una disposición típica de una matriz cualquiera es:

$$[A] = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & a_{ij} & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix}$$

[A] es una matriz de m filas y n columnas, siendo a_{ij} un elemento de la matriz en el que i indica la fila que va desde 1 hasta m y j la columna que va desde 1 hasta n

Cuando m = n se trata de una matriz cuadrada

Si m = 1 se trata de una matriz fila

Si n = 1 es una matriz columna.

Los elementos a_{11}, a_{22} , $a_{33,}$ ... a_{ij} son los elementos de la diagonal, de forma que si solo

existen elementos de este tipo iguales a 1 y el resto de elementos son 0 esta matriz se denomina matriz unidad [U].

Dada una matriz [A] (m,n) podemos construir su matriz simétrica (n,m) en la que los elementos $a_{ij} = a_{ji}$

Dos matrices [A] y [B] son iguales cuando $a_{ij} = b_{ij}$.

De las múltiples propiedades matemáticas que se desarrollan en los textos especializados de cálculo de matrices, únicamente nos van a interesar dos operaciones que necesitaremos utilizar en los movimientos fundamentales y que se refieren a la suma y al producto de matrices.

La suma de dos matrices [A] y [B] sólo podrá realizarse cuando ambas tengan el mismo numero de filas y de columnas dando lugar a una nueva matriz [C] cuyos elementos se encuentran a partir de $c_{ij} = a_{ij} + b_{ij}$.

sea $[A] = \begin{bmatrix} 2 & 4 \\ 3 & 5 \end{bmatrix}$ y $[B] = \begin{bmatrix} 5 & 7 \\ 6 & 8 \end{bmatrix}$

la matriz suma será $[C] = \begin{bmatrix} 2+5 & 4+7 \\ 3+6 & 5+8 \end{bmatrix} = \begin{bmatrix} 7 & 11 \\ 9 & 13 \end{bmatrix}$

El producto de dos matrices solo es posible si el numero de filas de la primera coincide con el numero de columnas de la segunda [A](m,n) x [B](n,p) = [C] (m,p) obteniéndose los elementos del siguiente modo:

$$c_{ij} = a_{i1} \cdot b_{1j} + a_{i2} \cdot b_{2j} + a_{i3} \cdot b_{3j} + \ldots + a_{in} \cdot b_{nj} = \sum_{k=1}^{n} a_{ik} \cdot b_{kj}$$

De una forma mas gráfica vamos a ver el calculo de un elemento de la matriz [C] producto de dos matrices [A] x [B] y que dispondremos del siguiente modo

$$\begin{bmatrix} a_{11} & a_{12} & a_{31} \\ \hline a_{21} & a_{22} & a_{23} \\ \hline a_{31} & a_{32} & a_{33} \end{bmatrix} \quad = \quad \left[\begin{array}{c|cc} b_{11} & b_{12} & b_{31} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{array}\right]$$

$$\begin{bmatrix} a_{11} & a_{12} & a_{31} \\ \hline a_{21} & a_{22} & a_{23} \\ \hline a_{31} & a_{32} & a_{33} \end{bmatrix} \quad = \quad \begin{bmatrix} \dots & \dots & \dots \\ c_{21} & \dots & \dots \\ \dots & \dots & \dots \end{bmatrix}$$

para encontrar el elemento c con índice i=2 j=1 su calculo será

$$c_{21} = a_{21} \cdot b_{11} + a_{22} \cdot b_{21} + a_{23} \cdot b_{31} = \sum_{k=1}^{3} a_{2k} \cdot b_{k1}$$

En este caso el cálculo completo de todos los elementos resultantes se realizará informáticamente mediante tres bucles contadores:

```
for ( i=1; i≤ 3; i++)
        for ( j=1; j≤ 3; j++)
                for ( k=1; k≤ 3; k++)
                        cij + = aik + ckj ;
```

Otra característica de una matriz es su determinante, que se calcula de un modo recursivo, existiendo varios métodos de cálculo como por ejemplo el método de Sarrus cuya resolución se muestra en el siguiente ejemplo:

$$\begin{vmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{vmatrix} = 1.5.9 + 2.6.7 + 4.8.3 - 7.5.3 - 4.2.9 - 8.6.1 = 0$$

4.4. COORDENADAS HOMOGENEAS

Como se ha dicho anteriormente un punto cualquiera P en el plano bidimensional queda determinado por sus coordenadas x_1 y_1 referidas al sistema de coordenadas OXY; sin embargo para los cálculos matriciales de transformaciones se usarán las llamadas coordenadas homogéneas que no son más que las coordenadas cartesianas anteriores multiplicadas por un factor t distinto de 0 de forma que si hacemos:

$\frac{x_1 . t}{t}$ y $\frac{y_1 . t}{t}$ y asignamos $x = x_1 . t$, $y = y_1 . t$;

a estos tres números (x, y, t) se les denominan coordenadas homogéneas del punto P.

A modo de ejemplo; un punto A del plano con coordenadas $x_1 = 4$, $y_1 = 5$ referidas al sistema de coordenadas OXY, se representa como:
$A = (x_1,y_1) = (4,5)$ en coordenadas cartesianas
y también en coordenadas homogéneas como:
$A = (x,y,t) = (4,5,1) = (8,10,2) = (12,15,3) = ...=(x_1 . t, y_1 . t, t)$.

Normalmente y de forma usual se usa t = 1 con lo que las coordenadas homogéneas son las mismas coordenadas cartesianas añadiéndoles el 1. $A = (x,y,t) = (4,5,1)$

La ecuación de cualquier recta del plano puede escribirse como:
$a.x + b.y + c . t = 0$

Si consideramos dos puntos cualquiera A y B pertenecientes a una recta y cuyas coordenadas homogéneas son $A = (x_1, y_1, t_1)$ $B=(x_2, y_2, t_2)$ se cumple:

$$a.x_1 + b.y_1 + c . t_1 = 0$$
$$a.x_2 + b.y_2 + c . t_2 = 0$$

La resolución de la ecuación de una recta se puede escribir de forma matricial en función de las coordenadas de dos de sus puntos haciendo nulo su determinante.

$$\begin{vmatrix} x & y & t \\ x_1 & y_1 & t_1 \\ x_2 & y_2 & t_2 \end{vmatrix} = 0$$

Cualquier movimiento o transformación de un punto P del plano de coordenadas homogéneas (x,y,1) en un nuevo punto P' de coordenadas (x',y',1) cumplirá la siguiente ecuación:

$$x' = a_{11}.x + a_{21}.y + a_{31}$$
$$y' = a_{12}.x + a_{22}.y + a_{32}$$

que expresado en forma matricial tiene la forma:

$$\begin{bmatrix} x' & y' & 1 \end{bmatrix} = \begin{bmatrix} x & y & 1 \end{bmatrix} . \begin{bmatrix} a_{11} & a_{12} & 0 \\ a_{21} & a_{22} & 0 \\ a_{31} & a_{32} & 1 \end{bmatrix}$$

Siendo [T] la matriz asociada a dicha transformación

$$\begin{bmatrix} T \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & 0 \\ a_{21} & a_{22} & 0 \\ a_{31} & a_{32} & 1 \end{bmatrix}$$

4.5. TRANSFORMACIONES EN EL PLANO

Estas transformaciones se pueden contemplar desde dos aspectos distintos: el primero será realizando movimientos del sistema de coordenadas y encontrar las nuevas coordenadas del objeto o punto determinado y el otro seria cambiar el objeto de posición manteniendo el origen de coordenadas y encontrar las coordenadas finales.

4.5.1 Movimientos del sistema de coordenadas

Estos movimientos tendrán su aplicación cuando definamos varias ventanas de pantalla y deseemos que cada una de ellas tenga su propio sistema de coordenadas.

A continuación se establece el convenio de indicar el sistema de referencia inicial como OXY y el sistema de referencia final como O'X'Y' calculando las nuevas coordenadas (x' y') de un punto con respecto al sistema de referencia final, a partir de sus coordenadas iniciales.

Traslación de los ejes principales

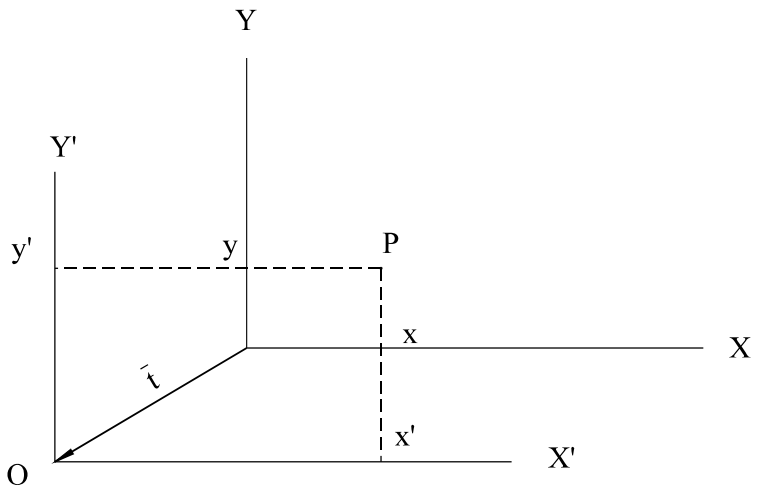


Fig. 4.4

La ecuación que nos facilita las nuevas coordenadas x' y' se resuelve del siguiente modo.

$$x' = x + t_x \qquad y' = y + t_y$$

Se observa que las nuevas coordenadas respecto de las antiguas solo están afectadas del vector de traslación $\overline{OO'} = \bar{t} = (t_x, t_y)$ de los orígenes de los centros de coordenadas.

Estas dos ecuaciones pueden expresarse matricialmente como

$$\begin{bmatrix} x' & y' & 1 \end{bmatrix} = \begin{bmatrix} x & y & 1 \end{bmatrix} . \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{bmatrix}$$

siendo $[T] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{bmatrix}$ la matriz asociada a este movimiento de traslación de ejes.

Giro de los ejes principales

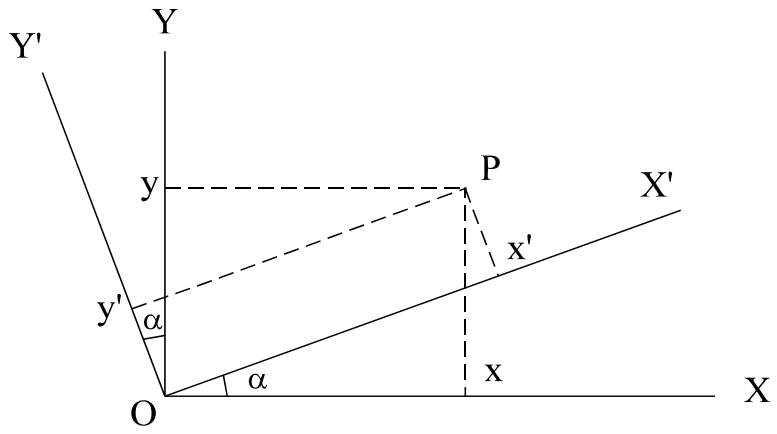


Fig. 4.5

Para encontrar analíticamente las nuevas coordenadas x' y' del punto P nos ayudaremos gráficamente de otra figura auxiliar (fig. 4.6) en la que se han construido los paralelogramos ADPB y HIPE de forma que la coordenada inicial x = OD = EP =HI y la coordenada y=OE =DP =AB por lo que las nuevas coordenadas se resuelven:

x' = OC + CB y' = OG -GI

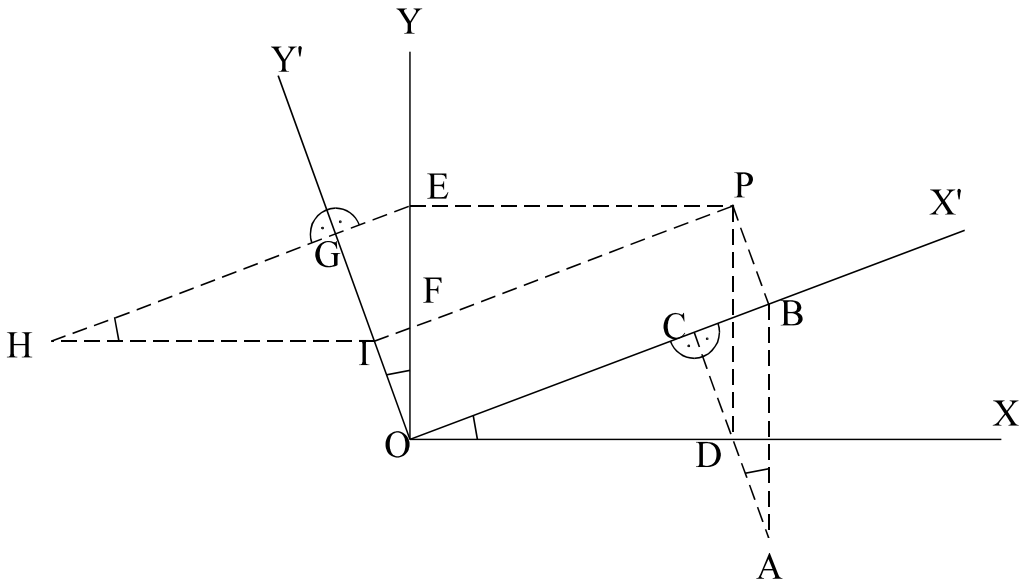


Fig. 4.6

como:

$OC = OD \cdot \cos\alpha = x \cdot \cos\alpha$

$CB = AB \cdot \text{sen}\,\alpha = y \cdot \text{sen}\,\alpha$

$OG = OE \cdot \cos\alpha = y \cdot \cos\alpha$

$GI = HI \cdot \text{sen}\,\alpha = x \cdot \text{sen}\,\alpha$

Así pues las ecuaciones que resuelven un giro de ángulo α de los ejes principales son

$x' = x \cdot \cos\alpha + y\,\text{sen}\,\alpha$

$y' = -x \cdot \text{sen}\,\alpha + y\cos\alpha$

que expresado matricialmente

$$\begin{bmatrix} x' & y' & 1 \end{bmatrix} = \begin{bmatrix} x & y & 1 \end{bmatrix} . \begin{bmatrix} \cos\alpha & -\operatorname{sen}\alpha & 0 \\ \operatorname{sen}\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

y $\begin{bmatrix} G \end{bmatrix} = \begin{bmatrix} \cos\alpha & -\operatorname{sen}\alpha & 0 \\ \operatorname{sen}\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$ la matriz asociada a este movimiento de giro de ejes.

Simetría del eje Y respecto del eje principal X

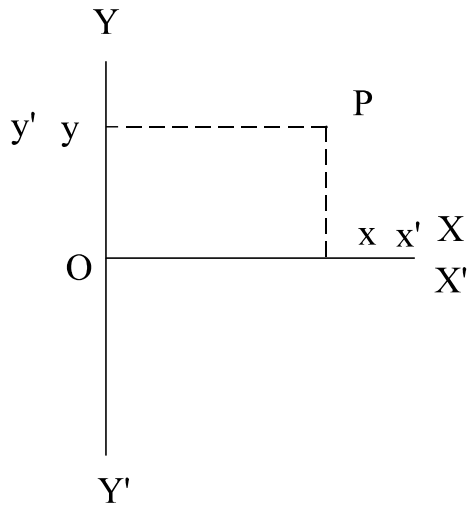


Fig. 4.7

Este movimiento también se puede considerar como un giro de 180º del eje Y respecto del eje X; resultando las ecuaciones de transformación bien simples:

$$x' = x \qquad y' = -y$$

que expresado matricialmente

$$\begin{bmatrix} x' & y' & 1 \end{bmatrix} = \begin{bmatrix} x & y & 1 \end{bmatrix} . \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

siendo $\begin{bmatrix} S_x \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

la matriz asociada a ésta simetría del eje Y respecto del eje X.

Simetría del eje X respecto del eje principal Y

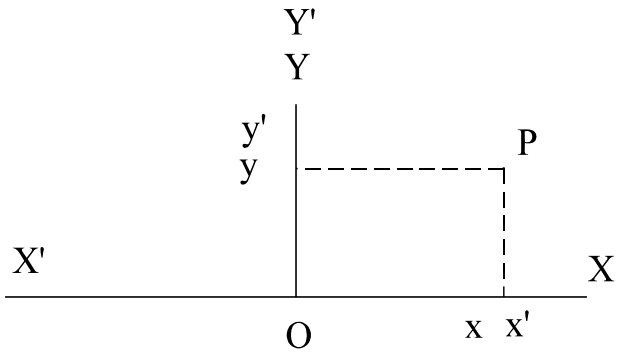


Fig. 4.8

Como en el caso anterior las ecuaciones de este movimiento son evidentes:

x’ = - x y’ = y que expresado matricialmente

$$\begin{bmatrix} x' & y' & 1 \end{bmatrix} = \begin{bmatrix} x & y & 1 \end{bmatrix} . \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

siendo $\left[S_y\right] = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

la matriz asociada a esta simetría del eje X respecto del eje Y.

Composición de movimientos

Mediante la aplicación de dos movimientos en distinto orden de aplicación se va a demostrar la propiedad no conmutativa del cambio de coordenadas compuesto por varios movimientos

Traslación y simetría sobre el eje X

La aplicación de estos dos movimientos se realizará aplicando primero una traslación y a continuación la simetría, resultando una matriz de transformación como el producto de las dos matrices asociadas a los movimientos unitarios.

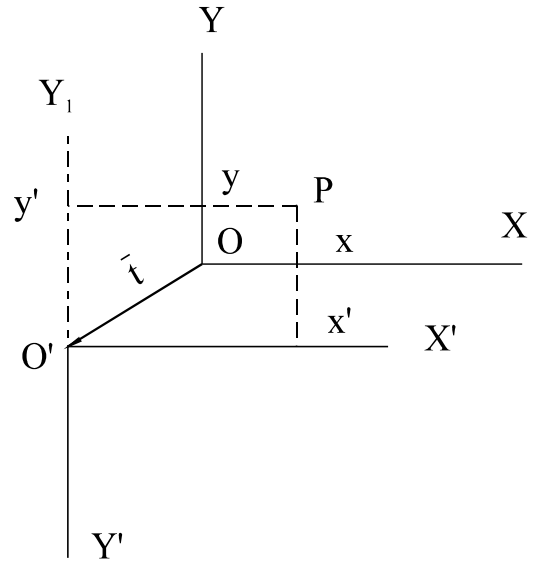


Fig. 4.9

$$\left[TS_x\right] = \left[T\right] . \left[S_x\right]$$

$$[TS_X] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{bmatrix} . \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ t_x & -t_y & 1 \end{bmatrix}$$

$$[x' \quad y' \quad 1] = [x \quad y \quad 1] . \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ t_x & -t_y & 1 \end{bmatrix}$$

resultando las ecuaciones:

$$x' = x + t_x \qquad y' = -y - t_y = -(y + t_y)$$

Simetría sobre el eje X y traslación

En este caso aplicamos primero la simetría y luego la traslación, resultando un cambio de coordenadas distinto del anterior, lo que nos indica que la composición de cambios de coordenadas no es conmutativa. El resultado del producto de las dos matrices asociadas nos demuestra esta no conmutatividad.

Si aplicamos el mismo vector de traslación t con coordenadas t_x, t_y la posición final del sistema de coordenadas será distinta por cuanto este vector esta referido al sistema de coordenadas intermedio una vez aplicada la simetría.

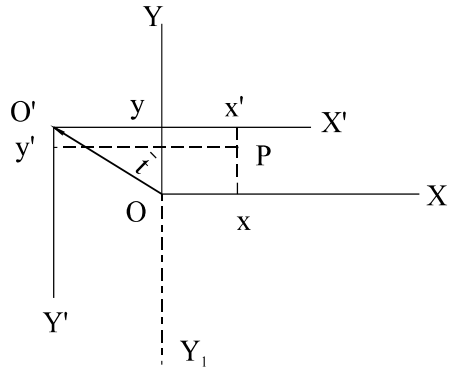


Fig. 4.10

$[S_X T] = [S_X] . [T]$ siendo $[TS_X] \neq [S_X T]$

$$[S_x T] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ t_x & t_y & 1 \end{bmatrix}$$

$$[x' \quad y' \quad 1] = [x \quad y \quad 1] . \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ t_x & t_y & 1 \end{bmatrix}$$

resultando las ecuaciones distintas del caso anterior:

$$x' = x + t_x \qquad y' = -y + t_y$$

Caso general

Un cambio general del sistema de coordenadas se puede realizar con la combinación de aplicar sucesivamente los casos simples vistos anteriormente de traslación giro y simetría.

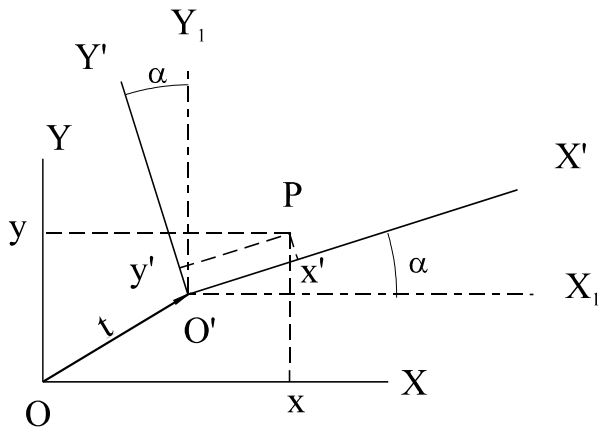


Fig. 4.11

Como cada movimiento simple tiene su propia matriz asociada que lo identifica, bastará hacer una composición de matrices para encontrar la matriz única que corresponda a ese cambio general del sistema..

Tal como se aprecia en la figura 4.11 el caso mas general de cambio de coordenadas se

puede contemplar por la aplicación de una traslación de vector t = ($-t_x$, $-t_y$) en primer sistema de coordenadas para luego aplicarle un giro y obtener el sistema final.

$$\begin{bmatrix} x' & y' & 1 \end{bmatrix} = \begin{bmatrix} x & y & 1 \end{bmatrix} . \begin{bmatrix} a_{11} & a_{12} & 0 \\ a_{21} & a_{22} & 0 \\ a_{31} & a_{32} & 1 \end{bmatrix}$$

$$\begin{bmatrix} TG \end{bmatrix} = \begin{bmatrix} T \end{bmatrix} . \begin{bmatrix} G \end{bmatrix}$$

$$\begin{bmatrix} TG \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -t_x & -t_y & 1 \end{bmatrix} . \begin{bmatrix} \cos\alpha & -\operatorname{sen}\alpha & 0 \\ \operatorname{sen}\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} =$$

$$\begin{bmatrix} 1.\cos\alpha + 0.\operatorname{sen}\alpha + 0.0 & 1.-\operatorname{sen}\alpha + 0.\cos\alpha + 0.0 & 1.0 + 0.0 + 0.1 \\ 0.\cos\alpha + 1.\operatorname{sen}\alpha + 0.0 & 0.-\operatorname{sen}\alpha + 1.\cos\alpha + 0.0 & 0.0 + 1.0 + 0.0 \\ -t_x.\cos\alpha - t_y.\operatorname{sen}\alpha + 1.0 & -t_x.-\operatorname{sen}\alpha - t_y.\cos\alpha + 1.0 & -t_x.0 - t_y.0 + 1.1 \end{bmatrix} =$$

$$\begin{bmatrix} TG \end{bmatrix} = \begin{bmatrix} \cos\alpha & -\operatorname{sen}\alpha & 0 \\ \operatorname{sen}\alpha & \cos\alpha & .0 \\ -t_x.\cos\alpha - t_y.\operatorname{sen}\alpha & -t_x.-\operatorname{sen}\alpha - t_y.\cos\alpha & 1 \end{bmatrix}$$

Resultando la ecuación general del siguiente modo:

$$x' = x.\cos\alpha + y.\operatorname{sen}\alpha - t_x.\cos\alpha - t_y.\operatorname{sen}\alpha$$

$$y' = -x.\operatorname{sen}\alpha + y.\cos\alpha + t_x.\operatorname{sen}\alpha - t_y.\cos\alpha$$

o escritas de otro modo sacando factor común

$$x' = \cos\alpha\,(x - t_x) + \operatorname{sen}\alpha\,(y - t_y)$$

$$y' = -\operatorname{sen}\alpha\,(x - t_x) + \cos\alpha\,(y - t_y)$$

El caso anterior se ha resuelto considerando que los ejes principales tenían la misma orientación, pero si los ejes tienen distinta orientación tal como se muestra en la figura

4.12, será preciso añadirle un movimiento de simetría y aplicarle la matriz asociada a ella.

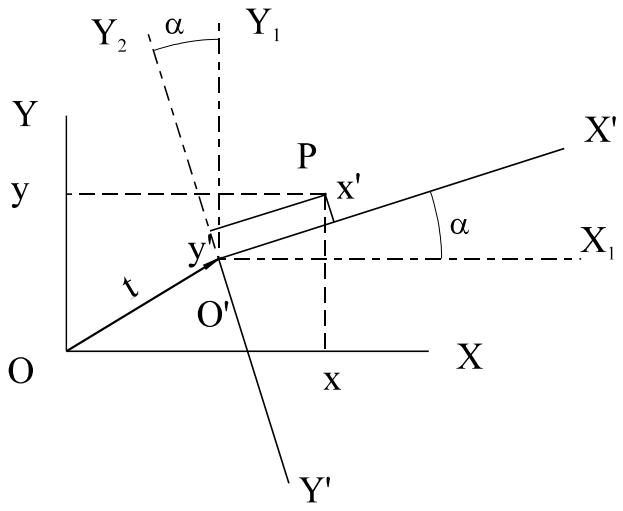


Fig. 4.12

$$[TGS] = [T].[G].[S] = [TG].[S]$$

$$[TGS] = \begin{bmatrix} \cos\alpha & -\text{sen}\,\alpha & 0 \\ \text{sen}\,\alpha & \cos\alpha & .0 \\ -t_x.\cos\alpha - t_y.\text{sen}\,\alpha & -t_x.-\text{sen}\,\alpha - t_y.\cos\alpha & 1 \end{bmatrix} . \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} =$$

$$[TGS] = \begin{bmatrix} \cos\alpha & \text{sen}\,\alpha & 0 \\ \text{sen}\,\alpha & -\cos\alpha & .0 \\ -t_x.\cos\alpha - t_y.\text{sen}\,\alpha & -t_x.\text{sen}\,\alpha + t_y.\cos\alpha & 1 \end{bmatrix}$$

resultando las ecuaciones

$$x' = \cos\alpha\,(x - t_x) + \text{sen}\,\alpha\,(y - t_y)$$

$$y' = \text{sen}\,\alpha\,(x - t_x) - \cos\alpha\,(y - t_y)$$

4.5.2 Movimientos del objeto en el plano

Cualquier punto P perteneciente a un objeto queda identificado por sus coordenadas cartesianas (x, y), que como se ha dicho anteriormente se transformarán en homogéneas (x,y,1), pues ello permitirá un tratamiento matricial uniforme, de los distintos movimientos a desarrollar.

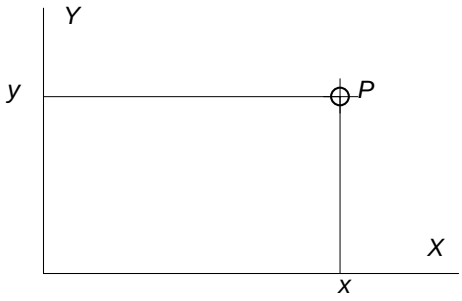


Fig. 4.13

Cualquier movimiento o transformación del punto P en un nuevo punto P' de coordenadas (x',y',1) cumplirá la siguiente ecuación:

$$x' = a_{11}.x + a_{21}.y + a_{31}$$

$$y' = a_{12}.x + a_{22}.y + a_{32}$$

que expresado en forma matricial sería:

$$\begin{bmatrix} x' & y' & 1 \end{bmatrix} = \begin{bmatrix} x & y & 1 \end{bmatrix} . \begin{bmatrix} a_{11} & a_{12} & 0 \\ a_{21} & a_{22} & 0 \\ a_{31} & a_{32} & 1 \end{bmatrix}$$

Siendo [T] la matriz asociada a dicha transformación

$$[T] = \begin{bmatrix} a_{11} & a_{12} & 0 \\ a_{21} & a_{22} & 0 \\ a_{31} & a_{32} & 1 \end{bmatrix}$$

Traslación

Es la forma geométrica fundamental de generación de diseños, siendo el movimiento a través del cual una figura es sometida a una repetición vertical, horizontal o diagonal, todo ello dentro de intervalos regulares.

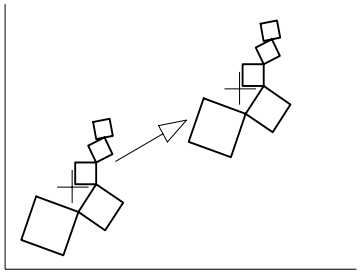


Fig. 4.14

Una traslación viene definida por un vector $\bar{t} = (t_x , t_y)$, obteniéndose el nuevo punto con coordenadas x', y', según la siguiente relación:

$$x' = x + t_x$$

$$y' = y + t_y$$

Las nuevas coordenadas respecto de las antiguas solo están afectadas por las coordenadas del vector de traslación, que puede expresarse en forma matricial, siento [T] la matriz asociada a dicha traslación:

$$\begin{bmatrix} x' & y' & 1 \end{bmatrix} = \begin{bmatrix} x & y & 1 \end{bmatrix} . \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{bmatrix}$$

$$\begin{bmatrix} T \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{bmatrix}$$

Esta matriz de transformación es la misma que se indicó en el caso anterior de traslación de los ejes principales, si bien, en el caso de cambio del sistema de coordenadas la dirección del vector de traslación era opuesto al que ahora se considera.

Esta observación nos indica que un movimiento de traslación (o de cualquier otro tipo como veremos mas adelante) conlleva las mismas ecuaciones matriciales de transformación pero considerando vectores de sentido contrario. Si se consideran las mismas coordenadas y el mismo sentido del vector las ecuaciones serán las que correspondan a la matriz inversa de la matriz de transformación.

Giro

Es el movimiento que permite repetir una figura alrededor de un punto fijo e imaginario (conocido como centro de giro) mediante un determinado ángulo α, pudiéndose repetir a intervalos regulares.

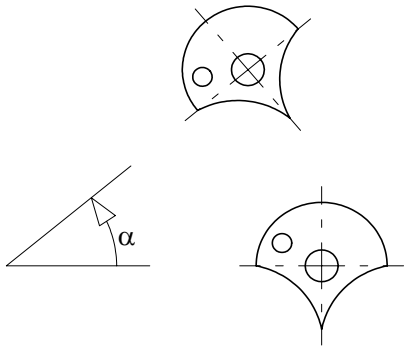


Fig. 4.15

En primer lugar consideramos las ecuaciones que corresponden al giro de un punto respecto del origen de coordenadas considerado como centro de giro:

La ecuación de dicha transformación es:

$$x' = x \,.\, \cos\alpha - y \,.\, \text{sen}\,\alpha$$
$$y' = x \,.\, \text{sen}\,\alpha + y \,.\, \cos\alpha$$

correspondiendo la siguiente expresión matricial y siendo [G] la matriz asociada a dicho giro:

$$\begin{bmatrix} x' & y' & 1 \end{bmatrix} = \begin{bmatrix} x & y & 1 \end{bmatrix} . \begin{bmatrix} \cos\alpha & \text{sen}\,\alpha & 0 \\ -\text{sen}\,\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$[G] = \begin{bmatrix} \cos\alpha & \text{sen}\,\alpha & 0 \\ -\text{sen}\,\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Podemos observar que esta matriz es la inversa a la que se ha indicado en el caso del giro de los ejes principales, por cuanto el sentido del ángulo de giro considerado ha sido el mismo que se consideró en el caso anterior de giro del sistema de coordenadas.

$$\begin{bmatrix} \cos\alpha & -\text{sen}\,\alpha & 0 \\ \text{sen}\,\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} \cos\alpha & \text{sen}\,\alpha & 0 \\ -\text{sen}\,\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = 1$$

Si el centro de giro es un punto cualquiera de coordenadas (m_1, m_2) la matriz resultante puede encontrarse a partir de la composición de varios movimientos y se obtendrá a partir de la multiplicación de las matrices que corresponden a los movimientos siguientes:

1º Trasladar el objeto y el centro de giro al origen de coordenadas

2º Realizar un giro respecto del origen de coordenadas.

3 Retrasladar el objeto con un vector inverso al anterior.

$$\left[G_m\right] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -m_1 & -m_2 & 1 \end{bmatrix} . \begin{bmatrix} \cos\alpha & \text{sen}\,\alpha & 0 \\ -\text{sen}\,\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ m_1 & m_2 & 1 \end{bmatrix} =$$

$$\left[G_m\right] = \begin{bmatrix} \cos\alpha & \text{sen}\,\alpha & 0 \\ -\text{sen}\,\alpha & \cos\alpha & 0 \\ -m_1.\cos\alpha + m_2.\text{sen}\,\alpha + m_1 & -m_1.\text{sen}\,\alpha - m_2.\cos\alpha + m_2 & 1 \end{bmatrix}$$

Simetría

Este movimiento permite duplicar la figura a través de una línea imaginaria, conocida como eje de simetría o de reflexión, produciendo de esta manera una imagen especular.

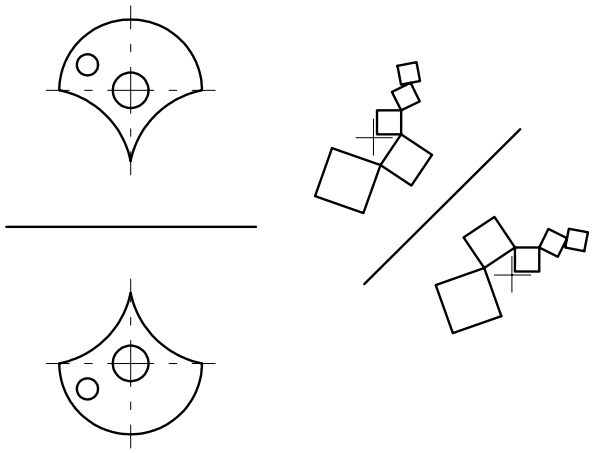


Fig. 4.16

En este movimiento de simetría se considerarán varios casos, según sea la posición de este eje de simetría, por lo que cambiará la matriz asociada en cada caso:

- Simetría en el eje X:

$$x' = x \qquad y' = -y \qquad \left[S_x\right] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Simetría en el eje Y:

$$x' = -x \qquad y' = y \qquad \left[S_y\right] = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Simetría central respecto del origen:

$$x' = -x \qquad y' = -y \qquad \left[S_0\right] = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Simetría central respecto de un punto cualquiera:

Para este movimiento, primero nos trasladaremos al origen, luego realizaremos una simetría central respecto del origen, para finalmente volver a trasladar a la posición anterior:

$$\left[S_m\right] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -m_1 & -m_2 & 1 \end{bmatrix} . \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ m_1 & m_2 & 1 \end{bmatrix} =$$

$$\left[S_m\right] = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 2m_1 & 2m_2 & 1 \end{bmatrix}$$

- Simetría respecto una recta que pasa por el origen formando un ángulo α con el eje X:

Para encontrar la matriz asociada, este caso se compone de:

- un giro.
- una simetría respecto a un eje (el X por ejemplo)
- un giro en sentido contrario.

por lo que la matriz asociada será:

$$\left[S_r\right] = \begin{bmatrix} \cos\alpha & \text{sen}\,\alpha & 0 \\ -\text{sen}\,\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} \cos\alpha & \text{sen}\,\alpha & 0 \\ -\text{sen}\,\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} =$$

$$\left[S_r\right] = \begin{bmatrix} \cos 2\alpha & \text{sen}\,2\alpha & 0 \\ \text{sen}\,2\alpha & -\cos 2\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

La ecuación general de dicha transformación:

$$x' = x.\cos 2\alpha + y.\text{sen}\,2\alpha$$
$$y' = x.\text{sen}\,2\alpha - y.\cos 2\alpha$$

Si la recta eje de simetría es la bisectriz ($\alpha = 45^o$) la ecuación será:

$$x' = y$$
$$y' = x$$

por lo que la matriz del movimiento es: $\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

- Simetría respecto una recta cualquiera que no pasa por el origen de coordenadas.

Para encontrar la matriz asociada, este caso se compone de:

1º -Traslación paralela que pase por el origen de coordenadas

2º - Simetría anterior según:

- un giro.
- una simetría respecto a un eje (el X por ejemplo)
- un giro en sentido contrario.

3º - Traslación a la posición inicial.

El resultado de aplicar estos movimientos da la siguiente matriz de transformación:

$$\left[S_r'\right] = \begin{bmatrix} \cos 2\alpha & \text{sen } 2\alpha & 0 \\ \text{sen } 2\alpha & -\cos 2\alpha & 0 \\ -m_1.\cos 2\alpha - m_2.\text{sen } 2\alpha + m_1 & -m_1.\text{sen } 2\alpha + m_2.\cos 2\alpha + m_2 & 1 \end{bmatrix}$$

Como una variante del movimiento general de simetría se incluye el caso de:

- Simetría Deslizada: en donde una figura es sometida a dos movimientos combinados, siendo uno de ellos una simetría en el eje X (o eje Y) y el segundo una traslación; ambos en relación con el mismo eje de simetría deslizada establecido, es decir dicha traslación debe ser paralela a la línea de simetría escogida.

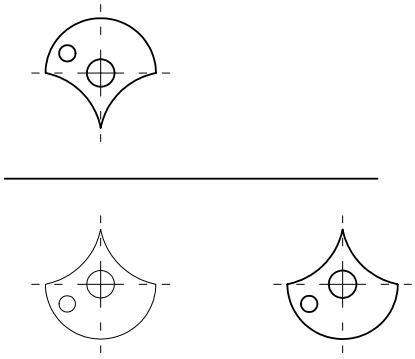


Fig. 4.17

Así pues también se presentarán dos casos más:

- Simetría deslizada respecto del eje X, que expresado matricialmente es:

$$\left[SD_x\right] = \left[S_x\right].\left[T_x\right] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ t_x & 0 & 1 \end{bmatrix}$$

siendo:
$$x' = x + t_x$$
$$y' = -y$$

- Simetría deslizada respecto del eje Y:

$$\left[SD_y\right] = \left[S_y\right].\left[T_y\right] = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & t_y & 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & t_y & 1 \end{bmatrix}$$

cuya ecuación es:
$$x' = -x$$
$$y' = y + t_y$$

Escalado

Es la transformación por un cambio de escala uniforme afectando con el mismo coeficiente de proporcionalidad a todas las dimensiones.

También se presentarán dos casos, según sea la posición del punto respecto del que se hace el escalado:

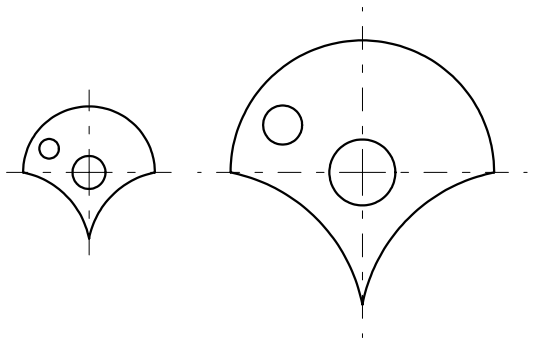


Fig. 4.18

- Respecto del origen:

La ecuación de dicha transformación es:

$$x' = k \cdot x$$
$$y' = k \cdot y$$

y su matriz asociada:

$$\left[H_0\right] = \begin{bmatrix} k & 0 & 0 \\ 0 & k & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Respecto de un punto m de coordenadas (m_1, m_2).

Dicho movimiento se compone de una traslación en el origen, una cambio de escala respecto del origen y una traslación inversa a la anterior. Así pues:

$$\left[H_m\right] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -m_1 & -m_2 & 1 \end{bmatrix} \cdot \begin{bmatrix} k & 0 & 0 \\ 0 & k & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ m_1 & m_2 & 1 \end{bmatrix} =$$

$$\left[H_m\right] = \begin{bmatrix} k & 0 & 0 \\ 0 & k & 0 \\ -km_1 + m_1 & -km_2 + m_2 & 1 \end{bmatrix}$$

siendo la ecuación general de la transformación:

$$x' = k \,.\, x + (1-k)\, m_1$$

$$y' = k \,.\, y + (1-k)\, m_2$$

4.6. TRANSFORMACIONES TRIDIMENSIONALES

El estudio del espacio de tres dimensiones se puede considerar como una ampliación del plano o espacio bidimensional, al que se le añade una coordenada z.

Este apartado tiene una analogía con el anterior, pudiéndose estudiar las matrices de transformación a partir de los movimientos del sistema de coordenadas o bien del objeto en el espacio, teniendo en cuenta esta nueva coordenada.

Un punto cualquiera P del espacio tridimensional queda determinado por sus tres coordenadas x_1 y_1 z_1 referidas al sistema de coordenadas OXYZ; del mismo modo que en el plano para los cálculos matriciales de transformaciones se usarán las coordenadas homogéneas:

$$\frac{x_1 \cdot t}{t} \quad , \quad \frac{y_1 \cdot t}{t} \quad y \quad \frac{z_1 \cdot t}{t}$$

asignando $x = x_1 . t$, $y = y_1 . t$, $z = z_1 . t$,

Así pues las nuevas coordenadas homogéneas de este punto P son (x, y, z, t).

Cualquier movimiento o transformación de un punto P del espacio con coordenadas homogéneas (x,y,z,1) en un nuevo punto P’ cuyas nuevas coordenadas sean (x’,y’,z’,1) cumplirá la siguiente ecuación:

$$x' = a_{11}.x + a_{21} . y + a_{31} . z + a_{41}$$
$$y' = a_{12}.x + a_{22} . y + a_{32} . z + a_{42}$$
$$z' = a_{13}.x + a_{23} . y + a_{33} . z + a_{43}$$

que expresado en forma matricial tiene la forma:

$$\begin{bmatrix} x' & y' & z' & 1 \end{bmatrix} = \begin{bmatrix} x & y & z & 1 \end{bmatrix} \cdot \begin{bmatrix} a_{11} & a_{12} & a_{13} & 0 \\ a_{21} & a_{22} & a_{23} & 0 \\ a_{31} & a_{32} & a_{33} & 0 \\ a_{41} & a_{42} & a_{43} & 1 \end{bmatrix}$$

Siendo [T] la matriz asociada a dicha transformación

$$[T] = \begin{bmatrix} a_{11} & a_{12} & a_{13} & 0 \\ a_{21} & a_{22} & a_{23} & 0 \\ a_{31} & a_{32} & a_{33} & 0 \\ a_{41} & a_{42} & a_{43} & 1 \end{bmatrix}$$

Como el programa informático que se presenta no desarrolla la simulación de tejidos en tres dimensiones, no se va a hacer la exposición completa de las transformaciones tridimensionales, pero queda abierta la puerta de su aplicación en un futuro ya que el método de tratamiento matricial de los movimientos es valido para la 3ª dimensión.

A modo de ejemplo de cualquier transformación tridimensional se presenta la ecuación para una transformación de traslación que resultaría del siguiente modo

La traslación definida por un vector $\bar{t} = (t_x, t_y, t_z)$, transforma el nuevo punto según la siguiente relación:

$$x' = x + t_x$$

$$y' = y + t_y$$

$$z' = z + t_z$$

De forma análoga a lo expuesto para el plano, las nuevas coordenadas respecto de las antiguas solo están afectadas por las coordenadas del vector de traslación, pudiéndose

expresar en forma matricial, siento [T] la matriz asociada a dicha traslación:

$$\begin{bmatrix} x' & y' & z' & 1 \end{bmatrix} = \begin{bmatrix} x & y & z & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ t_x & t_y & t_z & 1 \end{bmatrix}$$

$$[T] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ t_x & t_y & t_z & 1 \end{bmatrix}$$

CAPITULO 5

DISEÑO DE MOTIVOS

5.1. INTRODUCCION

Desde siempre el hombre ha diseñado a partir de simples formas geométricas. La naturaleza exhibe numerosos ejemplos de regularidad y simplicidad, desde las estrellas en sus trayectos, la maravilla de la cristalización y sus cristales, la rica ordenación de las flores, etc. La capacidad de modificar conscientemente los entes naturales para lograr un objeto final con un orden y un propósito significativo es inherente a la naturaleza humana, a su capacidad de diseñar, y es aquí donde el dibujo, bajo el punto de vista de creación de motivos para su aplicación en el campo de la ornamentación se ha tratado convencionalmente desde el punto de vista artístico.

El diseñador maneja muchas veces estos elementos para obtener sus creaciones, otras veces mezcla "formas" grandes con pequeñas, buscando un equilibrio en su composición final, introduce también a menudo, de forma consciente o inconsciente, el contraste para lograr un énfasis visual, con ello aporta otras composiciones informales que no dependen de conceptos matemáticos, sino de criterios más o menos subjetivos.

Los principios de diseño abordados en este capítulo, se concretan en composiciones simples a base de giros y simetrías, por otra parte ya conocidos y utilizados únicamente en su forma más elemental en el diseño y en el "software" de los equipos de CAD.

Se van a concretar unas estructuras básicas, formadas por series de puntos unidos, que dan lugar a líneas y espacios situados en un plano, que conservando sus atributos y aplicando los criterios antes citados, pueden generar el diseño final.

Los mismos elementos decorativos presentan diversas alternativas, cuando se eligen diferentes estructuras básicas y se ordenan en repeticiones según formas, tamaños, posiciones y colores.

La clasificación de los diseños es una difícil tarea si no se establecen unas normas claramente definidas, por lo que se va a crear una notación con una simbología adecuada que facilite la descripción del camino a seguir para la obtención del patrón final, base para la realización de diseños. Se entra en la creación de motivos exponiendo como se realiza su formación, su dependencia, y se les asigna una nomenclatura para su designación.

Elegido un motivo, es en general, adaptable a los variados movimientos expuestos, y con ellos se consiguen una amplia gama de diseños originales sumamente interesantes, no utilizados anteriormente, a lo sumo sólo algunos de ellos, por no existir una clasificación adecuada que permitiera de una forma racional, posibilitar un camino para desarrollar todo el conjunto de variantes que pueden darse.

El dibujo, desde el concepto de creación de motivos, para su aplicación en el campo de la ornamentación de tejidos, se ha tratado convencionalmente desde el punto de vista artístico. Muy poco o escasamente, se han aplicado conceptos matemáticos basados en un orden y unas leyes que permitan un tratamiento científico que posibilite su representación y movimientos para crear el diseño. Se establecen unos principios para intentarlo, aprovechando la potencia que proporciona la informática y la visión inmediata de los motivos en las pantallas de los ordenadores.

Todo este estudio busca proporcionar herramientas y medios que faciliten en gran manera el trabajo de los artistas y diseñadores, permitiendo ampliar poderosamente su actividad creativa, por ello se ha establecido como objetivo principal desarrollar de manera sencilla el proceso de obtención de los patrones de diseño más importantes.

Esta notación y clasificación se ha desarrollado para su aplicación al diseño textil, pero sus principios son válidos y pueden generalizarse a cualquier tipo de diseño.

5.2. ESTRUCTURAS BÁSICAS

Después de haber definido los movimientos fundamentales principales y antes de someterlos a una serie de combinaciones para la formación de otras, se debe señalar que existe un elemento ligado a todas que llamaremos Estructura Básica, el cual no es más que una serie de puntos unidos pertenecientes a una matriz. Dichos puntos de un modelo dado pueden ser trasladados en direcciones diferentes a través del plano, conservando el mismo tamaño, contenido y forma, generando así a partir de una estructura básica el total del diseño.

De las múltiples figuras geométricas que se pueden construir en el plano, uniendo puntos de una determinada matriz, solo se utilizarán cinco de ellas (Fig 5.1), que constituirán las estructuras básicas siguientes: cuadrada, rectangular, paralelogramo, hexagonal y rómbica. La estructura hexagonal toma el nombre de la composición de dos triángulos equiláteros uniendo dos vértices.

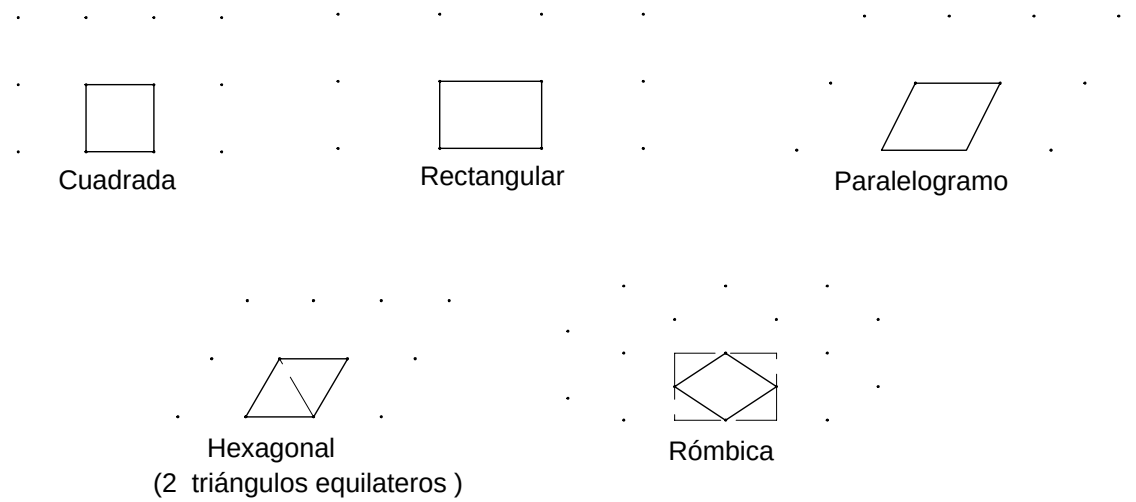


Fig. 5.1

Estas cinco estructuras básicas son conocidas también como "celdas de Bravais", en honor de dicho autor, que fue quien propuso la utilización de estos cinco tipos en el diseño artístico. En la figura siguiente se muestran estas cinco estructuras básicas con

un motivo inscrito a reproducir.

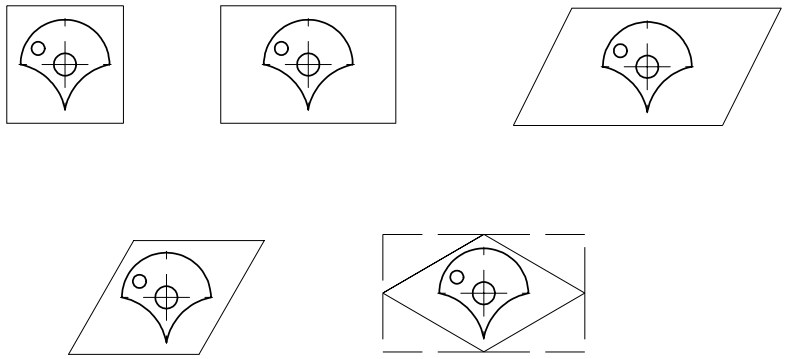


Fig. 5.2

Al variar la estructura básica, cuadrado, rectángulo, etc., los mismos elementos decorativos presentan diversas soluciones, proporcionando alternativas de decoración que permitirán la mejor adecuación en la ornamentación buscada..

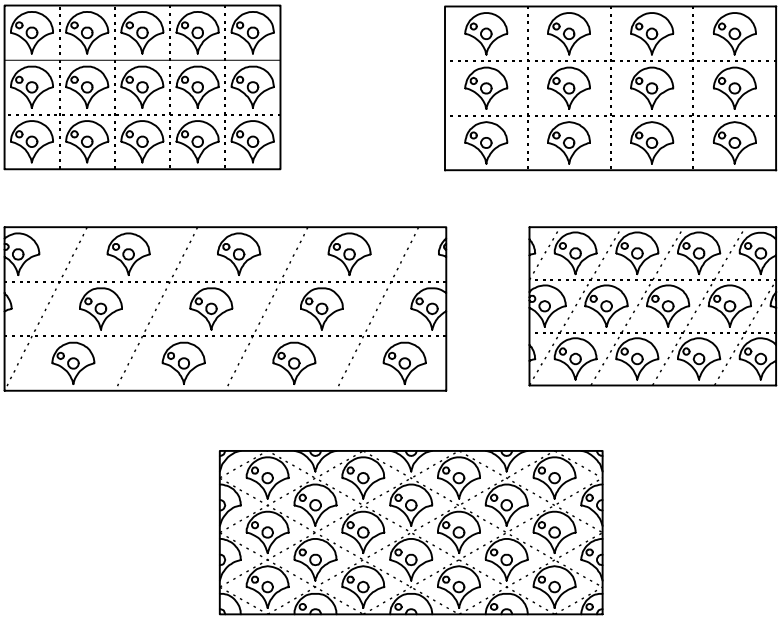


Fig. 5.3

Al señalar cinco tipos de estructuras básicas dentro del plano, se puede establecer a través de ellas una forma de repetición regular de un diseño cualquiera, definiendo esta

repetición como la reproducción por transferencia de un motivo a través de una distancia dada desde un lugar a otro y todo ello dentro del mismo plano, permitiendo a la vez la permanencia del motivo en su posición inicial.

Refiriéndose estrictamente al motivo, éste se puede someter a los diferentes movimientos fundamentales o a la combinación de ellos, inscribiendo el motivo final en una estructura básica. A modo de ejemplo, en la figura siguiente se muestra el motivo inicial, los movimientos a los cuales ha sido sometido, el motivo resultante y su inclusión en una estructura básica.

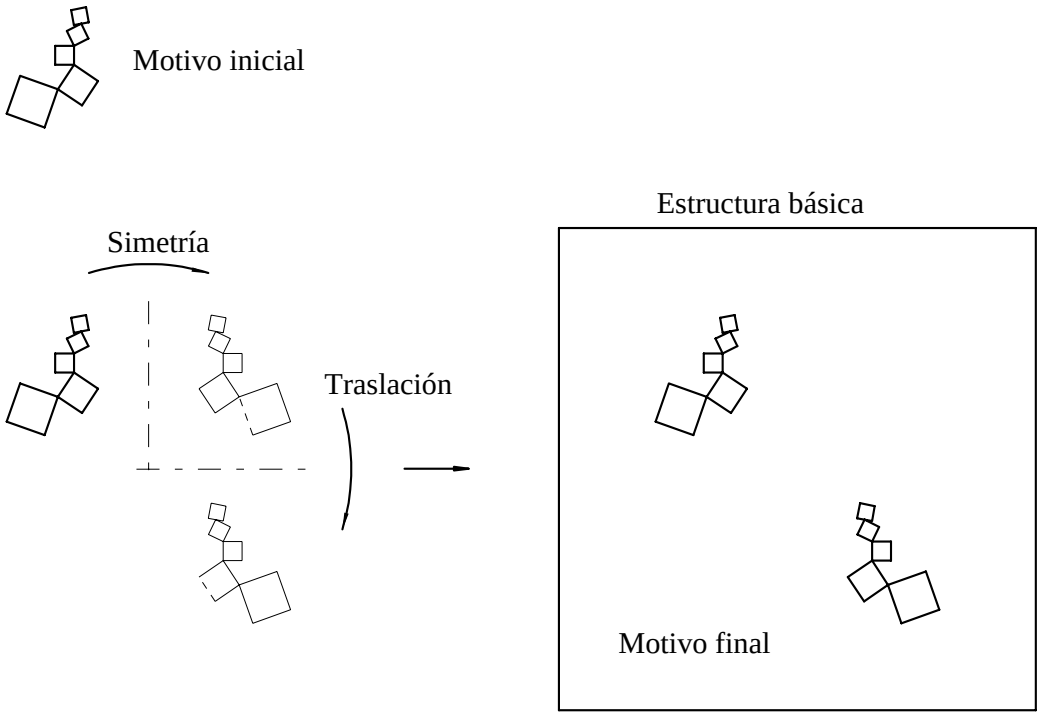


Fig. 5.4

5.3. CREACION DE MOTIVOS

Dependiendo de las operaciones o movimientos a realizar para la creación de un motivo, éste se puede clasificar inicialmente de dos maneras:

1º).- Los motivos que están formados sólo por movimientos a base de giros. La nomenclatura que los identificará será: g_n.

siendo: g = giro

n = número de grados del ángulo de giro (de 0 a 360º).

La figura 5.5 muestra un ejemplo de un motivo en base a giros de 120º.

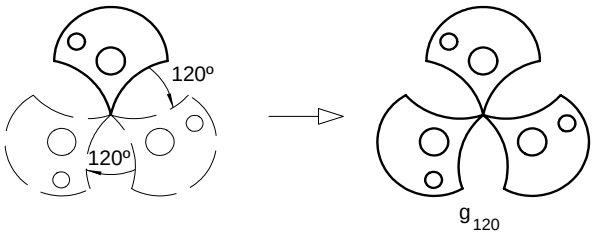


Fig. 5.5

1º).- Los motivos que están formados por simetrías. En este caso la nomenclatura adoptada es : s_m.

siendo: s = simetría

m = número de ejes de simetría (de 1 a m).

En la figura 5.6 se puede observar el ejemplo de un motivo con un eje de simetría.

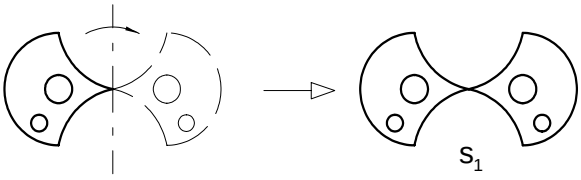


Fig. 5.6

Dependiendo del motivo, también cabe considerar que puede ser asimétrico o simétrico.

Los motivos asimétricos son entidades independientes, mientras que los simétricos han sido originados a través de simetrías o giros de otros más simples. Un motivo asimétrico adopta la nomenclatura inicial de g_0 y los simétricos de s_m , siendo el s_1, la mínima expresión de esta familia.(Fig. 5.7).

Fig.5.7

De acuerdo con la nomenclatura antes establecida, los motivos se pueden dividir en dos grandes grupos:

Primer Grupo :

Lo constituyen 3 casos particulares cuyas características son las siguientes:

- g_0: Se trata de un motivo asimétrico, sin movimiento fundamental asociado según Fig. 5.7 a, que se denomina como unidad fundamental.
- s_1: Es el motivo g_0 con un solo eje de simetría, lo cual produce un motivo final bilateralmente simétrico, según Fig. 5.6 y 5.7 b, a su vez compuesto por dos unidades fundamentales.
- g_{180}: Motivo con un ángulo de giro de 180º. En este caso se deberá elegir el centro de giro adecuado de tal forma que el motivo final ofrezca el mismo aspecto desde diferentes puntos de observación, (arriba-abajo, derecha-izquierda). Un ejemplo de ello se muestra en la figura 5.8.

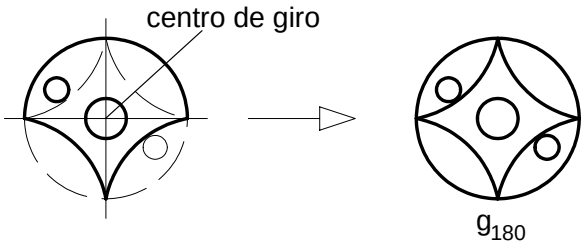


Fig.5.8

Segundo Grupo:

Se refiere al resto de motivos que podrán obtenerse dando valores a los subíndices n y m. Con valores de n, distintos de 0º y 180º (ya contemplados en el primer grupo), se dan los siguientes casos de giros:

g_{120}: Se realiza mediante 3 giros de 120º, 240º, y 360º. La última rotación coincide consigo misma.(Fig. 5.9)

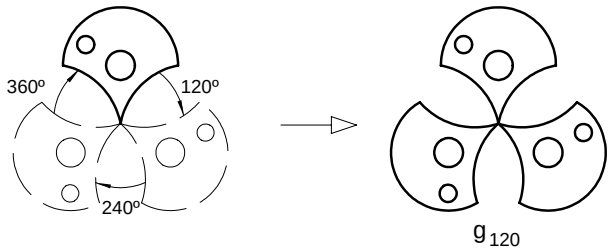


Fig. 5.9

g_{90}: Se realizan 4 giros de 90º, 180º, 270º, y 360º. (Fig. 5.10)

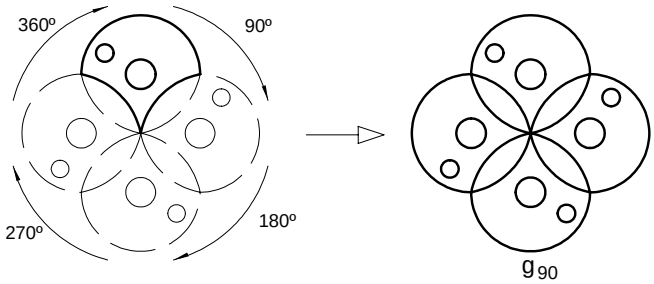


Fig. 5.10

g_{72}: Cinco movimientos rotacionales caracterizan a este motivo, realizándose

mediante giros de 72º, 144º, 216º, 288º y 360º.(Fig. 5.11)

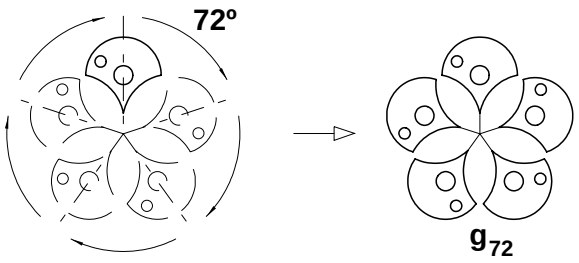


Fig. 5.11

g_{60}: Los giros con ángulos de rotación de 60º forman esta clase.(Fig. 5.12)

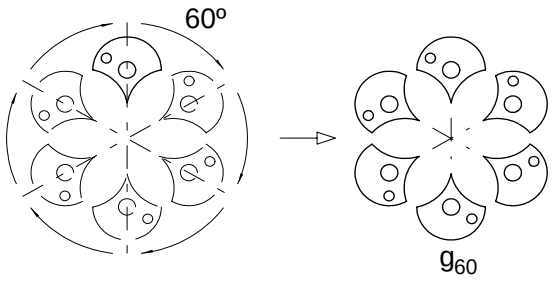


Fig. 5.12

Del mismo modo con valores de m, distintos de 1 (ya contemplado en el primer grupo), se dan los siguientes casos de simetrías:

s_2: Exhibe una simetría bilateral a través de dos ejes, horizontal y vertical, que se intersectan a 90º. (Fig. 5.13)

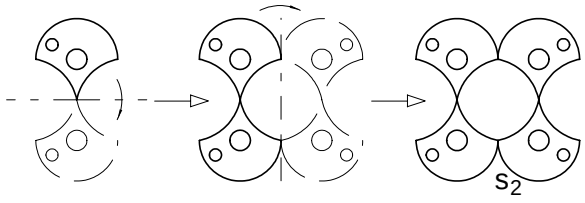


Fig. 5.13

s_3: Este movimiento compuesto posee tres ejes de simetría formando ángulos de 60º entre sí, originando un motivo final que consta de 6

unidades fundamentales. (Fig. 5.14).

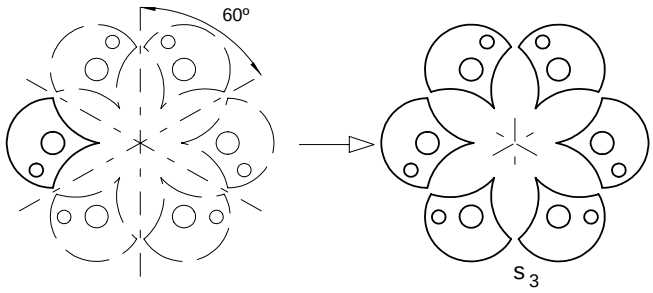


Fig. 5.14

s_4: La intersección de cuatro ejes de simetría con ángulos de 45º producen motivos con 8 unidades fundamentales. (Fig. 5.15)

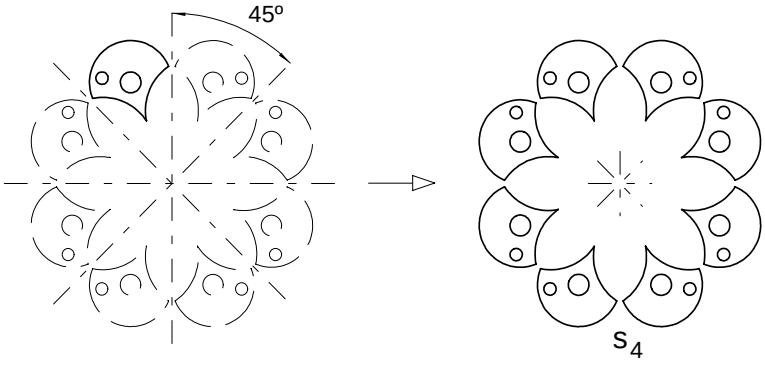


Fig. 5.15

s_5: Se produce la intersección de cinco ejes de simetría, con un ángulo de 36º, empleando para su formación 10 unidades fundamentales (Fig. 5.16).

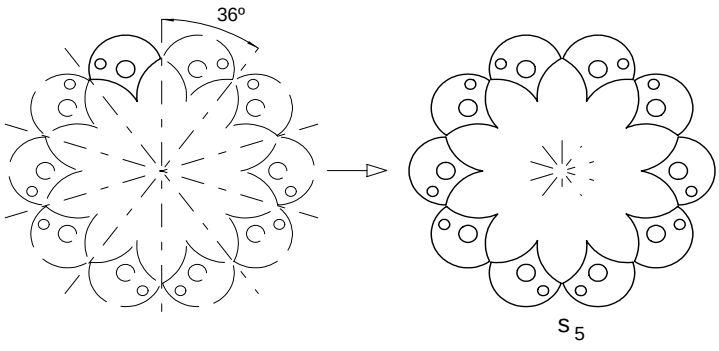


Fig. 5.16

s_6: De forma semejante a las anteriores se emplean 6 ejes de simetría con

ángulos de 30º entre sí, constituyendo un motivo final con 12 unidades fundamentales. (Fig. 5.17)

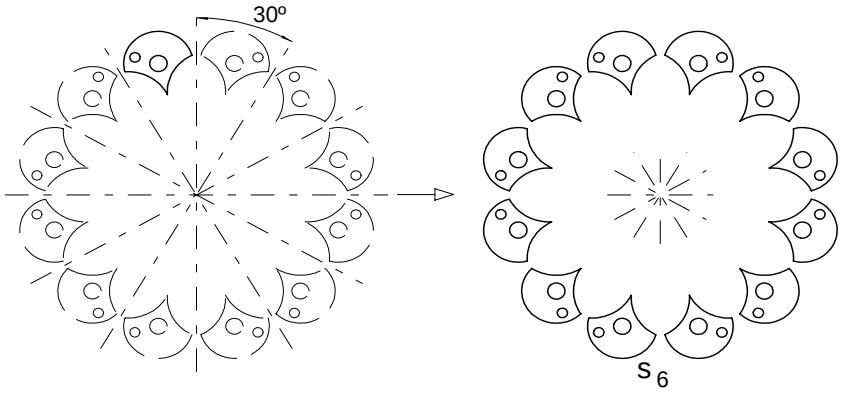


Fig. 5.17

Se ha de indicar que los motivos de la familia s_m, con m>1 también pueden ser obtenidos por movimientos giratorios de "α" grados de un motivo bilateralmente simétrico (s_1). La Fig. 5.18 muestra la reproducción de un motivo de clase s_2 a partir de un motivo fundamental (a), al que se convierte en una unidad bilateralmente simétrica (b) y posteriormente se aplica un giro de 180º para llegar a obtener el motivo final (c).

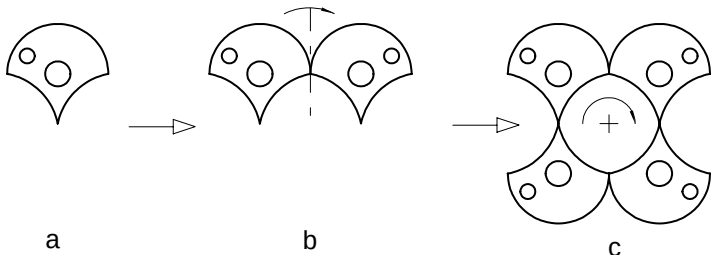


Fig. 5.18

Se puede comprobar que el resultado es un s_2 idéntico al de la figura 5.13.

Con respecto a los otros casos de simetría pertenecientes a esta familia, los ángulos de giro correspondientes son:

s_3: α = 120º, 240º y 360º.

s_4: α = 90º, 180º, 270º y 360º.

s_5: α = 72º, 144º, 216º, 288º y 360º.

s_6: α = 60º, 120º, 180º, 240º, 300º y 360º.

Como ejemplo de lo anterior, la figura 5.19 muestra un motivo del tipo s_4 obtenido a partir de giros, coincidiendo este resultado con el motivo final presentado anteriormente en la figura 5.15.

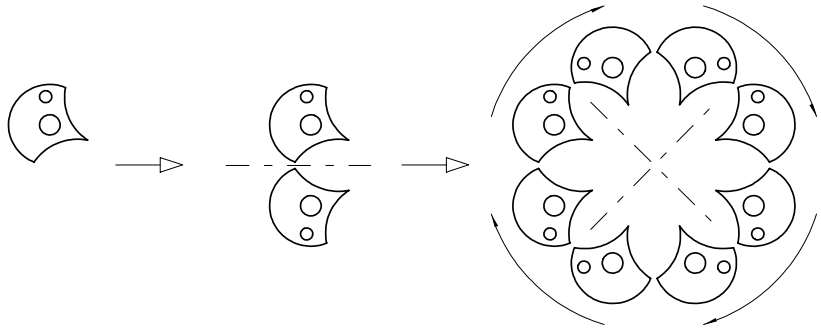


Fig. 5.19

Una vez visto este caso particular, se podrían conseguir motivos de grado superior para las clase g y s, siendo la condición para cada caso, la construcción de un círculo que tenga infinitos ejes de simetría e infinitos ángulos de giro.

La figura 5.20 proporciona ejemplos de motivos con órdenes elevados (grado 10) para las clases g y s, respectivamente.

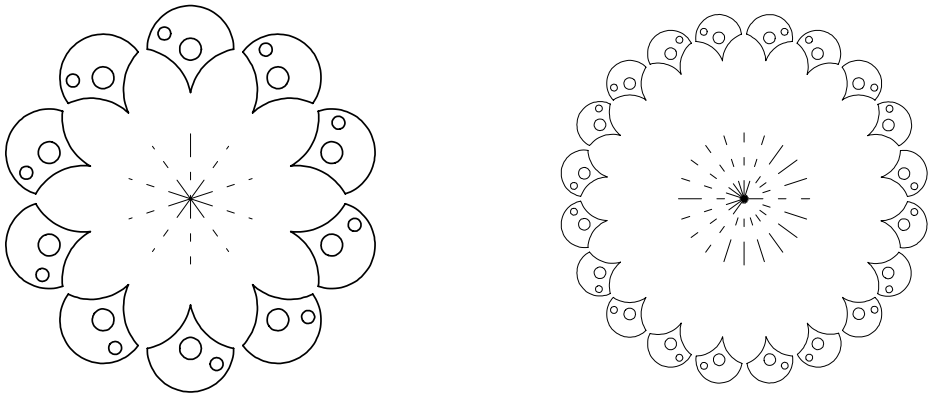


Fig.-5.20

5.4. CLASIFICACION DE LOS DISEÑOS

5.4.1 Notación a. b c d

Como se ha indicado anteriormente, todas las estructuras básicas que se usan para realizar diseños, pueden ser trasladadas en 2 direcciones independientes a través del plano. Ahora bien, si al motivo que se sitúa dentro de esta estructura, se le aplican los movimientos fundamentales, entonces se podrán obtener diferentes clases de "PATRONES" para la creación de diseños.

Una forma de relacionar el motivo elaborado a través de los movimientos fundamentales aplicados a éste, con la estructura básica empleada, es la creación de una NOTACION, la cual, mediante una simbología apropiada, facilitará una descripción detallada de las acciones a realizar para la obtención de un patrón de diseño final.

Esta notación consta de 4 índices o símbolos, teniendo cada símbolo a su cargo, la descripción de una característica particular. NOTACION : **a. b c d**

El primer símbolo **a.** denota el tipo de estructura básica asociada al patrón de diseño. De los múltiples elementos geométricos posibles, sólo se consideran 5 estructuras diferentes, las cuales se codifican del modo que indica la tabla siguiente.

Tabla 5.I

Estructura Básica	Cuadrada	Rectangular	Paralelogramo	Hexagonal	Rómbica
a	10	20	30	40	50

La última estructura rómbica (a=50), se considera inscrita en un rectángulo como se aprecia en la Fig. 5.21, de tal forma que los ejes de reflexión a utilizar, sean perpendiculares a los lados de esta celda rectangular.

El segundo símbolo **b**, significa el orden de rotación dentro del patrón elegido, de acuerdo con la tabla 5.II siguiente:

Tabla 5.II

Orden de rotación	**b**
0º ó No existe	1
180º	2
120º	3
90º	4
60º	6

Un número de 5 giros, se considera suficiente para obtener la mayoría de los patrones necesarios en el campo del diseño textil, sin embargo no se excluyen otras posibilidades que se puedan obtener dando valores diferentes al ángulo de giro. Con esta restricción se coincide con los criterios que también se discuten en otros campos, como por ejemplo en el de cristalografía.

La figura 5.21 muestra todos los casos utilizando **b** = 1.

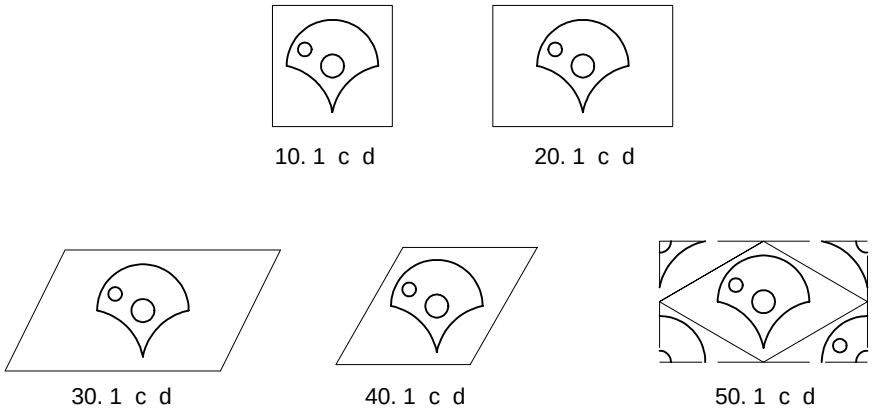


Fig. 5.21

La próxima figura 5.22 muestra algunos ejemplos para distintos valores de **a** y de **b,** aplicando las rotaciones correspondientes al motivo elegido.

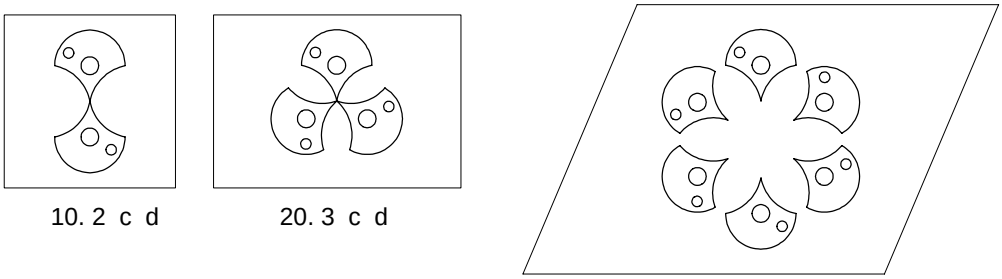


Fig. 5.22

El tercer símbolo **c**, denota la presencia de un eje de simetría, perpendicular al eje X de la estructura básica y el tipo de simetría realizada se especifica en la siguiente Tabla 5.III.

Tabla 5.III

Simetría	**c**
No existe	1
S	2
SD	3

Donde: S = Simetría y SD = Simetría deslizada.

La figura 5.23 muestra algunos ejemplos para distintos valores de **a , b** y **c**

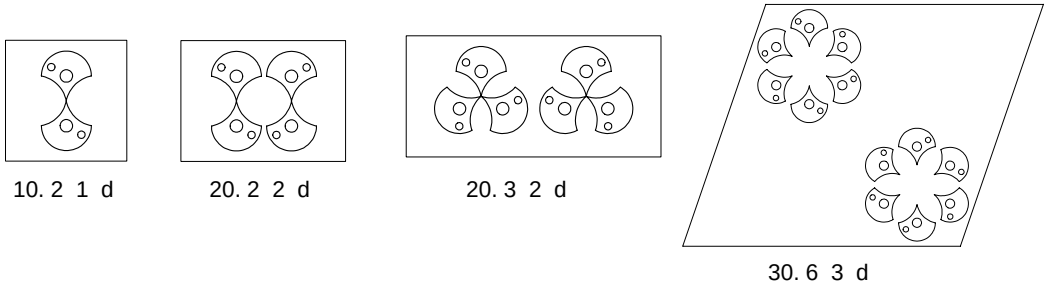


Fig. 5.23

El cuarto y último símbolo **d**, denota la presencia de un eje de simetría con un ángulo α, con respecto al eje X de la estructura básica (Fig. 5.24):

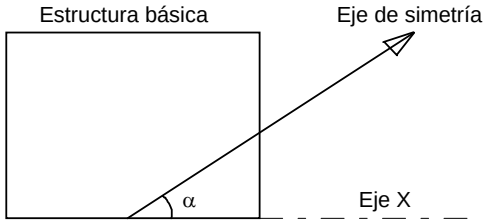


Fig. 5.24

Se ha establecido una relación entre el valor que toma este ángulo y el orden de rotación mostrado anteriormente por el segundo símbolo "**b**", de acuerdo con la tabla 5.IV siguiente:

Tabla 5.IV

b	**Orden de rotación**	**α**
1	0º	180º
2	180º	180º
3	120º	60º
4	90º	45º
6	60º	60º

Con respecto a los valores que puede tomar el símbolo **d**, en la tabla 5.V se indican los tres posibles índices a considerar, teniendo en cuenta que la relación expuesta anteriormente con el orden de rotación, sólo se aplicará cuando **d** sea distinto de la unidad.

Tabla 5.V

Simetría	**d**
No existe	1
S	2
SD	3

Particularidades

Cuando **c** y **d** toman valores iguales a 1, se está señalando que en el patrón de diseño no existe ningún tipo de simetría, y por lo tanto no será necesario representarlos en la notación.

Así por ejemplo:

a b c d	
20. 1 1 1	== 20. 1
10. 2 2 1	== 10. 2 2
30. 1 1 2	== 30. 1 1 2

Si el símbolo **d** es igual a 1, se está indicando que no existen simetrías con eje de reflexión inclinado, por lo cual no es necesario anotarlo: 10. 2 2 1 == 10. 2 2

Si el símbolo **c** es igual a 1, pero el **d** es distinto de 1, no se puede omitir la representación del símbolo **c**, ya que ello podría dar lugar a una interpretación equivocada de la notación, pues 30.1 1 2 ≠ 30. 1 2.

La notación 30. 1 1 2 significa:

- a = 30 Estructura de paralelogramo
- b = 1 No existe ningún tipo de giro
- c = 1 No existen simetrías respecto al eje Y
- d = 2 Indica la presencia de un eje de simetría con un ángulo de 180º respecto al eje X de la estructura y en este caso, la base de dicha estructura coincide con el eje de simetría.

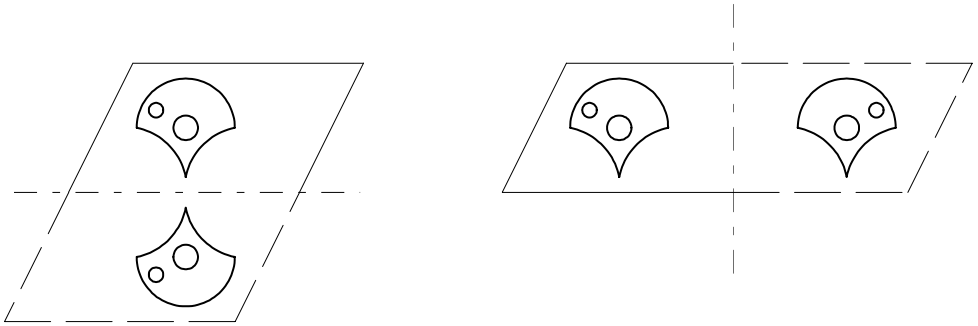
Mientras que la notación 30. 1 2 indicaría:

a = 30 Estructura de paralelogramo

b = 1 No existe ningún tipo de giro

c = 2 Indica la presencia de un eje de simetría perpendicular al eje X de la estructura.

La figura 5.25 muestra la diferencia entre estas dos notaciones, apreciándose el distinto resultado obtenido, confirmando la no posible simplificación de **c**, si el símbolo **d** es distinto de 1.



30. 1 1 2 30. 1 2

Fig. 5.25

5.4.2. Representación

Una manera de representar las combinaciones que se pueden establecer entre los símbolos de esta notación es a través de 3 ejes ortogonales, denominándose cada uno: **b**, **c** y **d**, respectivamente. El primer símbolo **a**, no se representa, ya que para cada uno de sus valores (10, 20, 30, 40 ó 50) se cumplen las relaciones que se dan a través de la representación tridimensional de los otros símbolos.

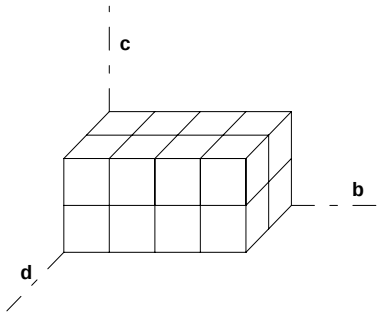


Fig. 5.26

Al eje **b** le corresponden 5 valores y a los ejes **c** y **d** 3 valores, señalándose que el punto de intersección de estos tres ejes (origen) toma un valor igual a 1. La figura 5.27 muestra diferentes casos para un valor variable de **b**.

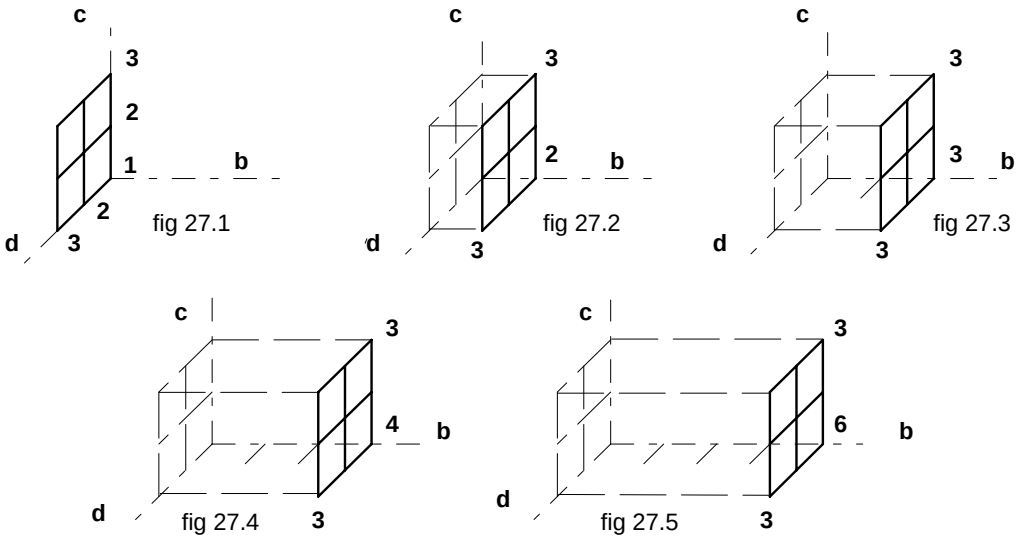


Fig. 5.27

Se observa que en la figura 5.27.1, donde **b** = 1 (recordando que este valor indica que

no existe ningún movimiento giratorio) se presentan 9 combinaciones dentro del plano **cd**.

a . b c d

1 1 1 No existen movimientos giratorios ni simétricos.

1 1 2 Existe una simetría con respecto al eje X de la estructura básica.

1 1 3 Simetría deslizada con respecto al eje X de la estructura básica.

1 2 1 Simetría con respecto al eje Y (perpendicular al eje X).

1 2 2 Simetría con respecto al eje Y y otra simetría tomando como eje de reflexión al propio eje X.

1 2 3 Simetría con respecto al eje Y y de otra simetría deslizada con respecto al mismo eje X.

1 3 1 Simetría deslizada con respecto al eje Y.

1 3 2 Simetría deslizada con respecto al eje Y y posteriormente una simetría con respecto al eje X.

1 3 3 Dos simetrías deslizadas, una con respecto al eje Y y la otra con respecto al eje X.

Descritos estos 9 casos, se puede aplicar el mismo criterio para analizar cada punto cuando **b** toma otros valores, considerando que para **b** = 2 (fig. 5.27.2) el ángulo α del eje inclinado descrito por el símbolo **d** es igual a 180º, o sea que coincide con el eje X de la estructura básica (de la misma manera que para **b** = 1).

Para **b** = 3 y **b** =6 según fig. 5.27.3 y fig. 5.27.5, respectivamente, cuando se analiza el símbolo **d** se considera que el ángulo α será igual a 60º y para **b** = 4 según fig. 5.27.4, este ángulo es igual a 45º (estos valores de α se han descrito en la Tabla 5.IV).

La Fig. 5.28, establece las relaciones para un valor de **c** variable, indicando el tipo de simetría respecto al eje Y que es perpendicular al eje X de la estructura básica.

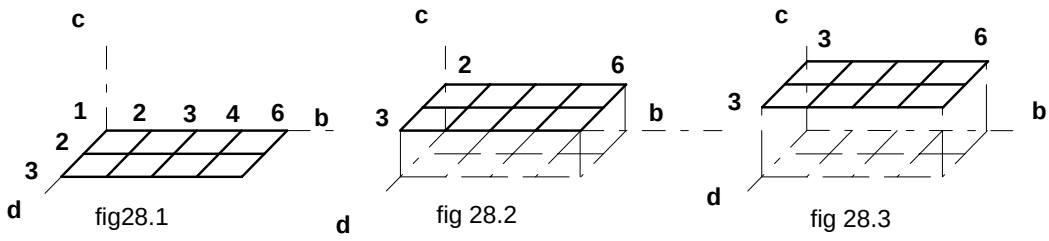


Fig. 5.28

Como ejemplo de análisis, en la fig. 5.28.1, cuando **c** =1, se producen 15 relaciones:

a . b c d

1 1 1 No existen movimientos giratorios ni simétricos.

1 1 2 Existe una simetría con respecto al eje X de la estructura básica.

1 1 3 Simetría deslizada con respecto al eje X de la estructura básica.

2 1 1 Existe una rotación de 180º.

2 1 2 Giro de 180º y luego una simetría con respecto al eje X.

2 1 3 Giro de 180º y luego una simetría deslizada con respecto al eje X.

3 1 1 Giros de 120º del motivo inicial.

3 1 2 Giros de 120º y luego una simetría a través de un eje inclinado que forma un ángulo de 60º con respecto al eje X.

3 1 3 Giros de 120º y luego una simetría deslizada a través de un eje inclinado que forma un ángulo de 60º con respecto al eje X.

4 1 1 Giros de 90º.

4 1 2 Giros de 90º y luego una simetría con respecto a un eje inclinado que forma un ángulo de 45º con respecto al eje X.

4 1 3 Giros de 90º y luego una simetría deslizada con respecto a un eje inclinado que forma un ángulo de 45º con respecto al eje X.

6 1 1 Giros de 60º.

6 1 2 Giros de 60º y una simetría con respecto a un eje inclinado que forma 60º con el eje X.

6 1 3 Giros de 60º y una simetría deslizada con respecto a un eje inclinado que forma 60º con el eje X.

Señalar que los 3 primeros casos de estas 15 relaciones coinciden también con las 3 primeras combinaciones analizadas en el estudio del plano **cd,** para **b** = 1.

Para **c** = 2 según fig. 5.28.2 y **c** = 3 según fig. 5.28.3, se aplicará el mismo criterio aunque esta vez se considerará la presencia de una simetría (**c** = 2) o de una simetría deslizada (**c** = 3) según sea el caso.

Por último, la Fig. 5.29 varía el valor de **d** , que es el símbolo responsable de indicar la presencia de algún tipo de simetría a través de un eje inclinado.

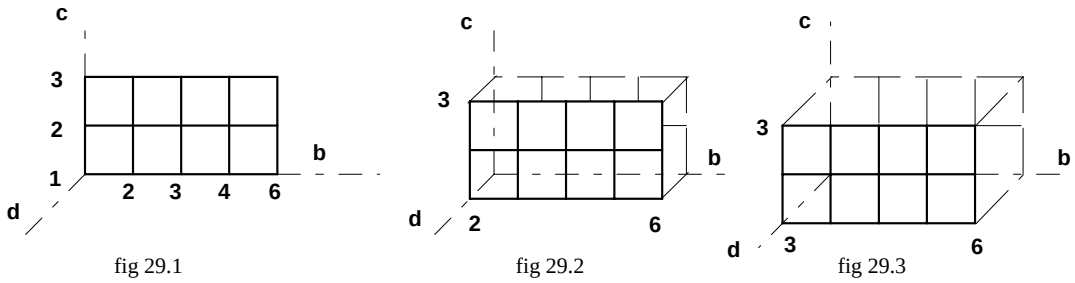


fig 29.1 fig 29.2 fig 29.3

Fig. 5.29

Para el caso indicado en la fig. 5.29.1, cuando **d** =1, se producen 15 casos también:

a . b c d

1 1 1 No existen movimientos giratorios ni simétricos.

1 2 1 Simetría con respecto al eje Y.

1 3 1 Simetría deslizada con respecto al eje Y.

2 1 1 Giro de 180º.

2 2 1 Giro de 180º y luego una simetría respecto al eje Y.

2 3 1 Giro de 180º y luego una simetría deslizada respecto al eje Y.

3 1 1 Giros de 120º del motivo inicial.

3 2 1 Giros de 120º y luego una simetría a través del eje Y.

3 3 1 Giros de 120º y luego una simetría deslizada a través del eje Y.

4 1 1 Giros de 90º.

4 2 1 Giros de 90º y luego una simetría con respecto al eje Y.

4 3 1 Giros de 90º y luego una simetría deslizada con respecto al eje Y.

6 1 1 Giros de 60º.

6 2 1 Giros de 60º y una simetría con respecto al eje Y.

6 1 3 Giros de 60º y una simetría deslizada con respecto al eje Y.

Aquí son 7 las combinaciones ya estudiadas en los análisis de los planos **cd** y **bd**, que coinciden los resultados (111, 121, 131 para **b** = 1 y 211, 311, 411, 611 para **c** = 1).

Para **d** = 2 según fig. 5.29.2 y **d** = 3 según fig. 5.29.3, se debe tener en cuenta la presencia de diferentes simetrías para cada combinación y el análisis a realizar seguirá las mismas pautas dadas para **d** = 1.

Para cada valor de **a** (responsable de la estructura básica a utilizar) se establecen todas las situaciones antes analizadas, en la Fig.5.30, tratando de esta forma de reunir y dar una idea gráfica de las interrelaciones que se producen entre los 4 símbolos.

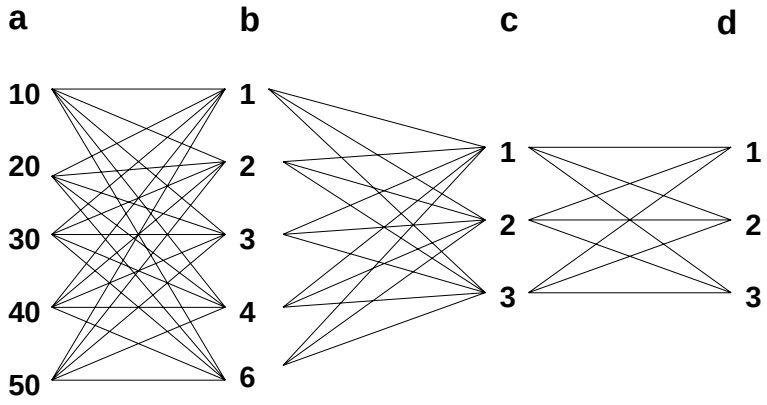


Fig. 5.30

De esta forma, a través de una manera simbólica y gráfica se establecen todas las combinaciones que se producen entre los parámetros que rigen la elaboración de los diseños, proporcionando así un amplio rango de posibilidades de creación, y si a ello se añade la elección de un motivo adecuado, susceptible de someterse a diferentes movimientos ya estudiados, se lograrán resultados sumamente interesantes y originales.

5.5. CREACION DE DISEÑOS

Como continuación de la notación propuesta anteriormente, se exponen algunos casos de las múltiples combinaciones que ofrece la notación **a. b c d** indicada.

5.5.1 Notación 10. 1 c d

El primer símbolo **a** se refiere al tipo de estructura básica, siendo para estos casos del tipo cuadrado **(a = 10**). Del mismo modo, al considerar **b =1**, se indica que no existen giros para el patrón inicial elegido.

Las posibles combinaciones de la notación **10. 1 c d**, en donde **c** y **d** adoptarán todos los valores posibles se muestra en la figura 5.31 señalando 9 combinaciones diferentes para valores de **a** y **b** únicos.

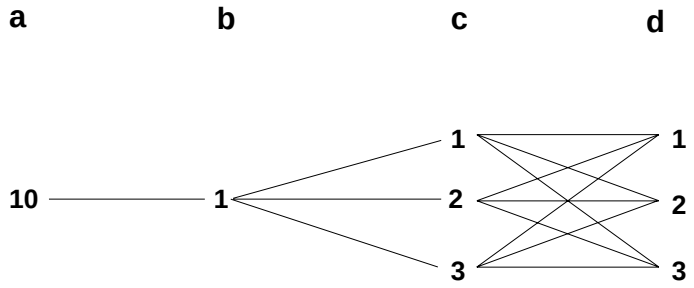


Fig. 5.31

El motivo inicial elegido es del tipo asimétrico s_0, facilitando de esta manera la mejor visualización de los cambios ocasionados por los diferentes movimientos a aplicar.

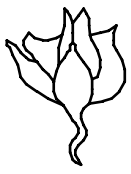


Fig. 5.32

Estableciendo como condiciones de partida la estructura cuadrada y la ausencia de giros para los 9 casos, al motivo inicial se le aplican los movimientos que indican las

notaciones respectivas, siendo el resultado final de cada uno de ellos el motivo base que se someterá a traslación.

Con cada notación se realizará una simulación de su reproducción en una área mayor que permitirá observar los diferentes efectos que ofrecen las distintas distribuciones.

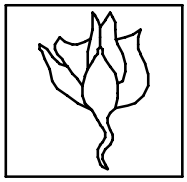
Notación 10. 1 1 1

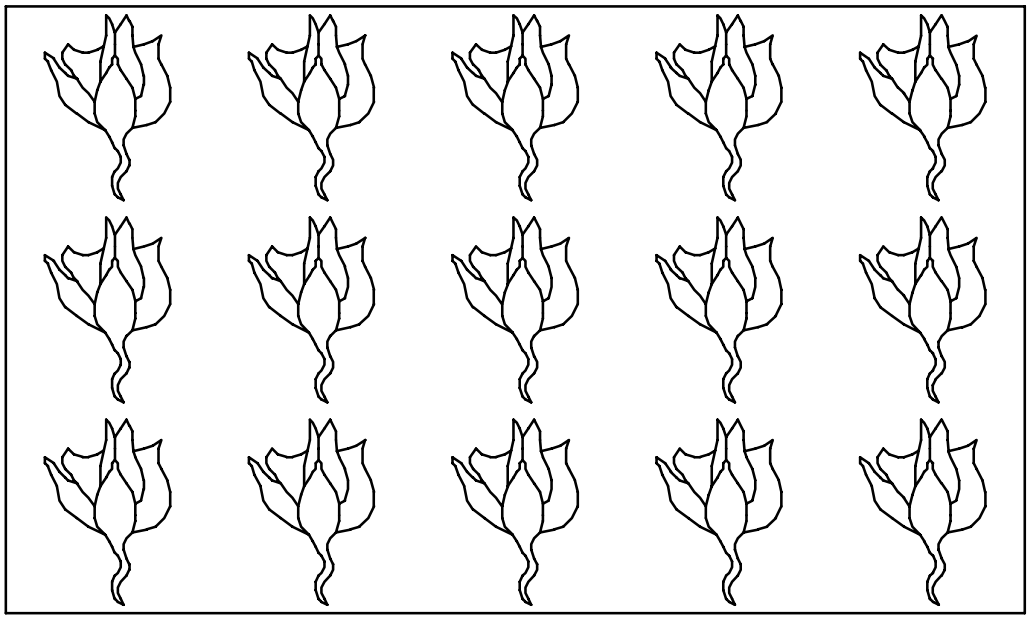
a = 10 Estructura básica de tipo cuadrangular.

b = 1 No existen movimientos giratorios.

c = 1 No existen simetrías con respecto al eje Y.

d = 1 No existen simetrías con respecto a un eje inclinado.





Se trata del caso más elemental dentro de todas las combinaciones posibles (independientemente de la estructura básica **a** utilizada) ya que únicamente consta de un simple movimiento de traslación del motivo inicial.

En este primer ejemplo y en los siguientes vemos la necesidad de disponer de un programa informático de diseño textil para las simulaciones respectivas, ya que los motivos adquieren otras características, en donde el delineado del mismo y el juego de colores, tanto del motivo como del fondo, le otorgarán efectos agradables a la vista.

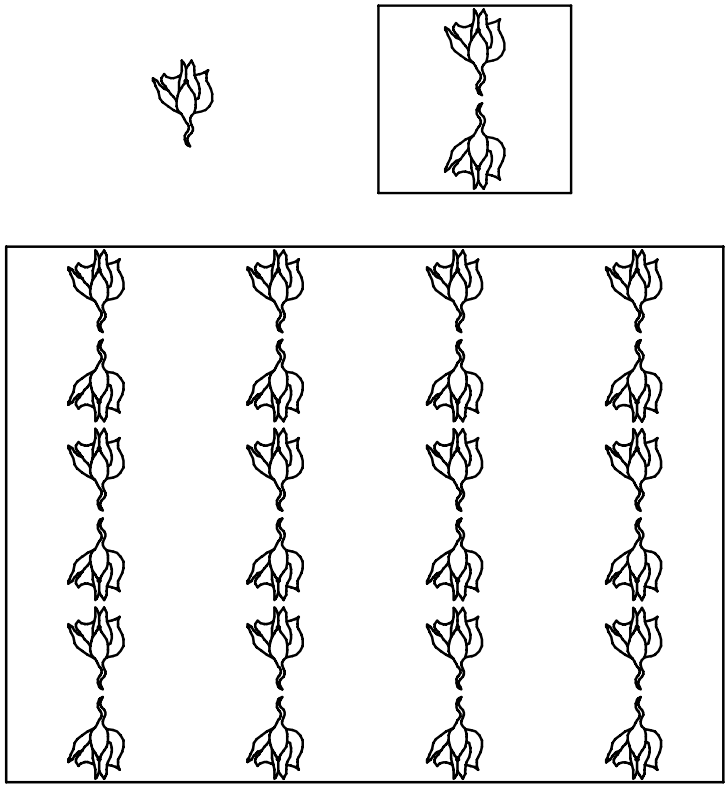
Notación 10. 1 1 2

a = 10 Estructura básica de tipo cuadrangular.

b = 1 No existen movimientos giratorios.

c = 1 No existen simetrías con respecto al eje Y.

d = 2 Simetría con respecto al eje X.



Notación 10. 1 1 3

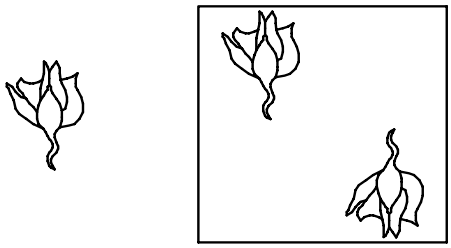
a = 10 Estructura básica de tipo cuadrangular.

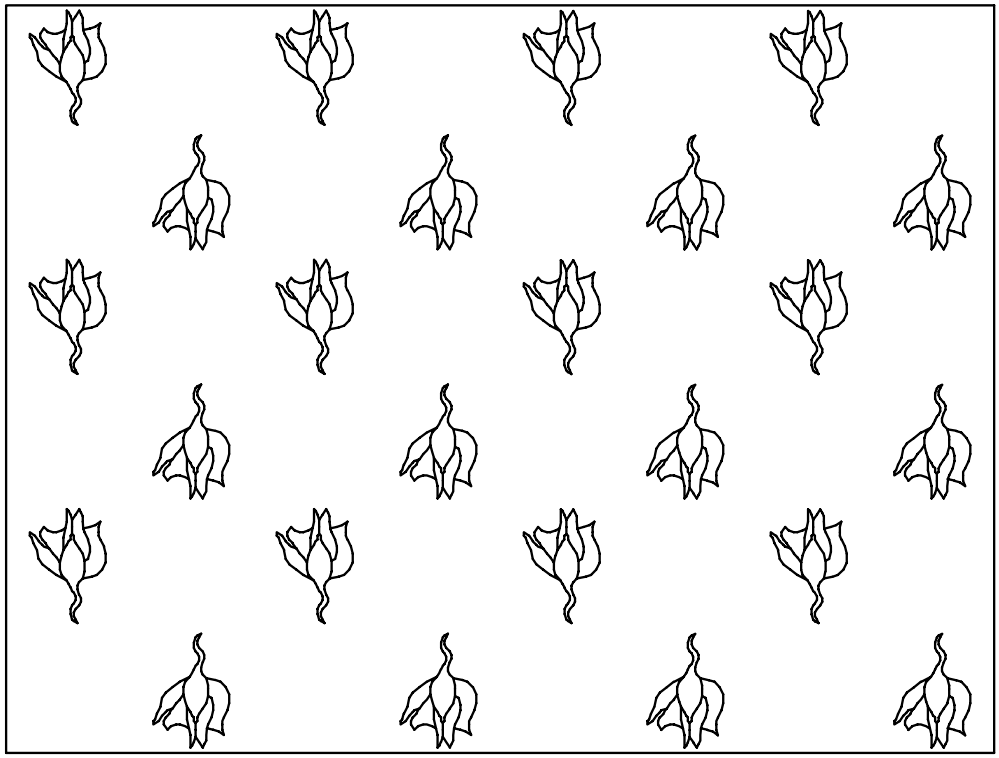
b = 1 No existen movimientos giratorios.

c = 1 No existen simetrías con respecto al eje Y.

d = 3 Simetría deslizada con respecto al eje X.

Para una distribución alternada, esta notación es una de las más indicadas.





Notación 10. 1 2 1

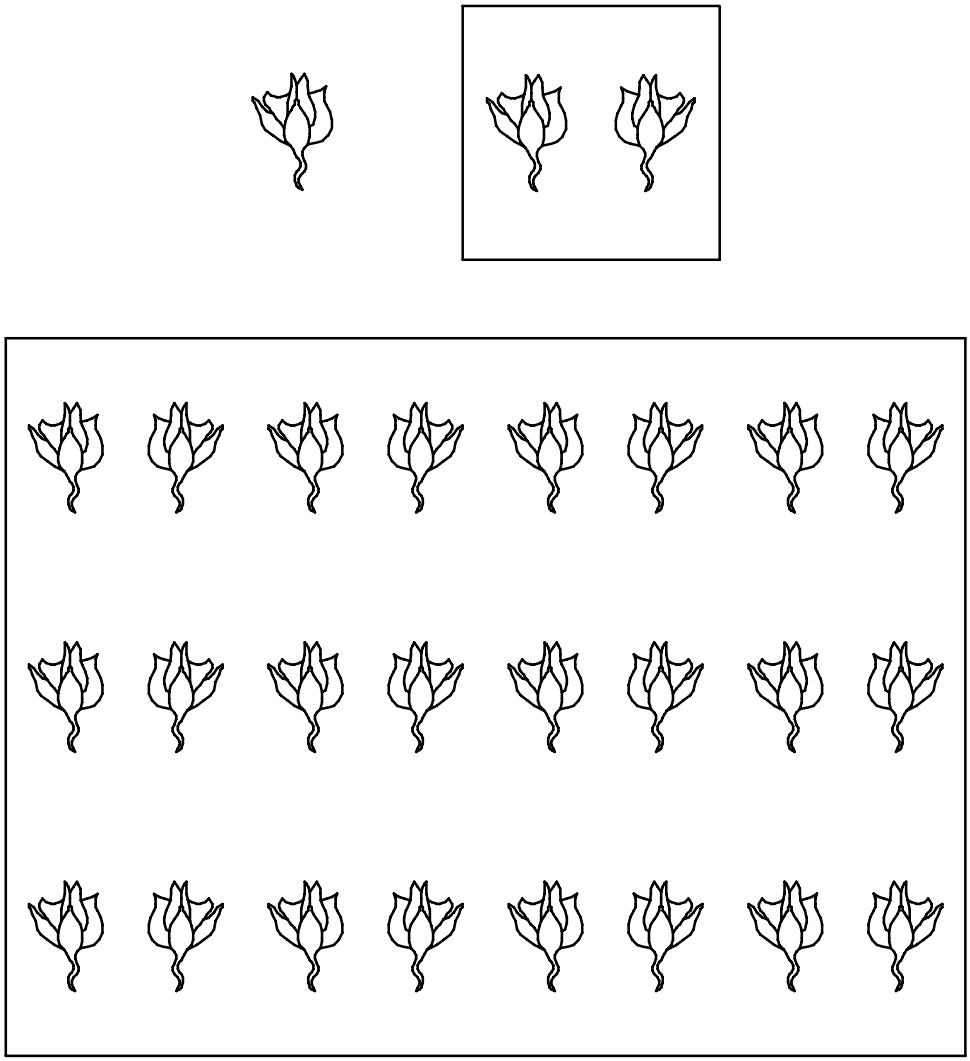
a = 10 Estructura básica de tipo cuadrangular.

b = 1 No existen movimientos giratorios.

c = 2 Simetría con respecto al eje Y.

d = 1 No existen simetrías con respecto a un eje inclinado.

Para un cambio de **c**, se nota la diferencia cuanto sea más asimétrico sea motivo inicial.



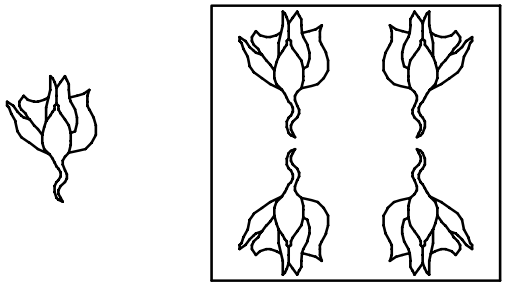
Notación 10. 1 2 2

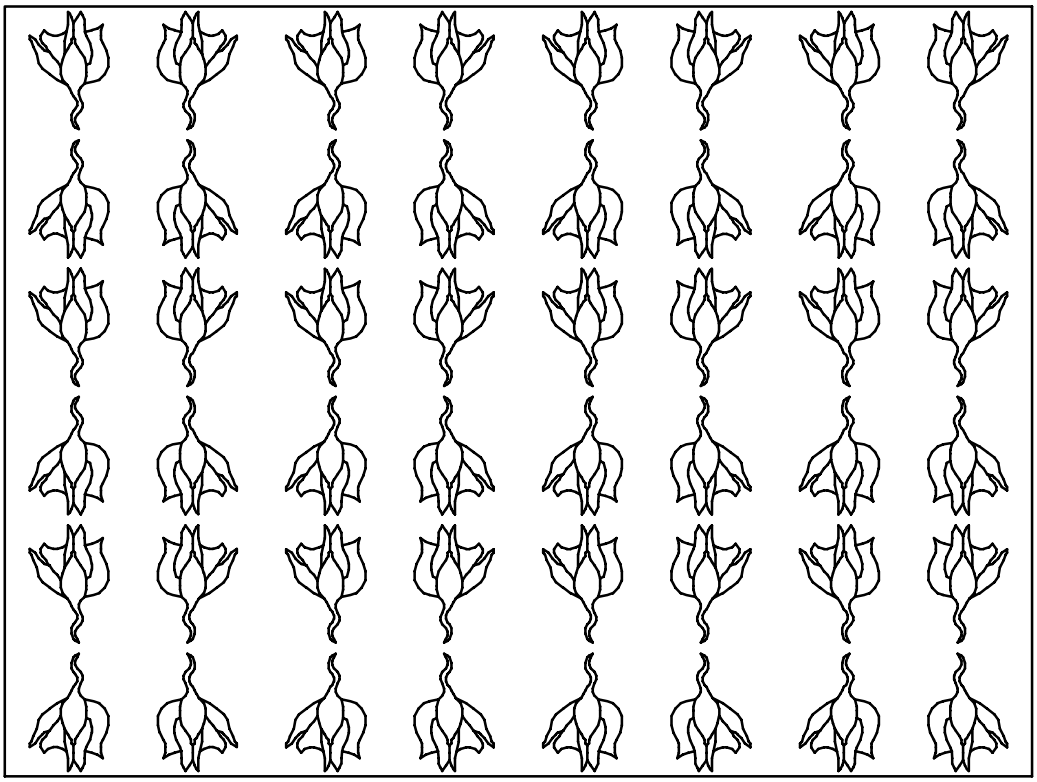
a = 10 Estructura básica de tipo cuadrangular.

b = 1 No existen movimientos giratorios.

c = 2 Simetría con respecto al eje Y.

d = 2 Simetría con respecto al eje X.





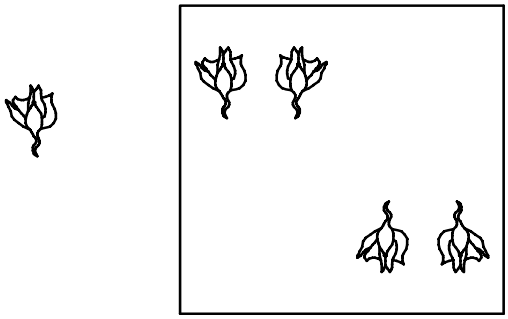
Notación 10. 1 2 3

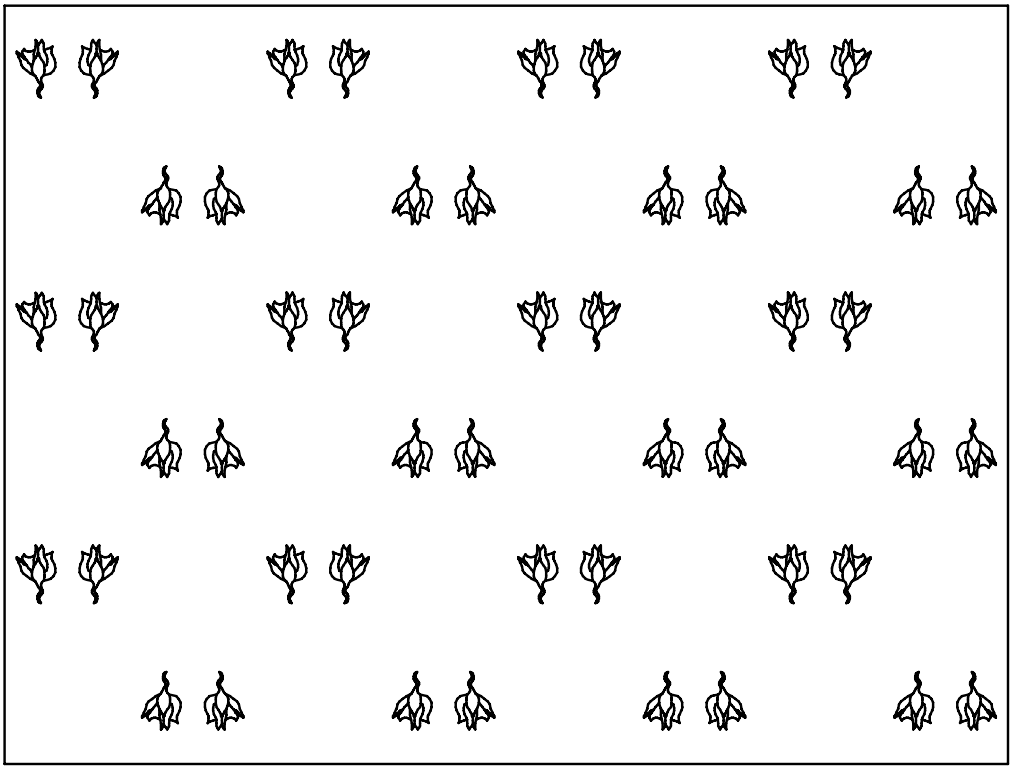
a = 10 Estructura básica de tipo cuadrangular.

b = 1 No existen movimientos giratorios.

c = 2 Simetría con respecto al eje Y.

d = 3 Simetría deslizada con respecto al eje X.





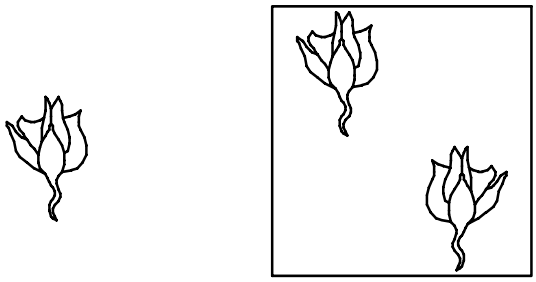
Notación 10. 1 3 1

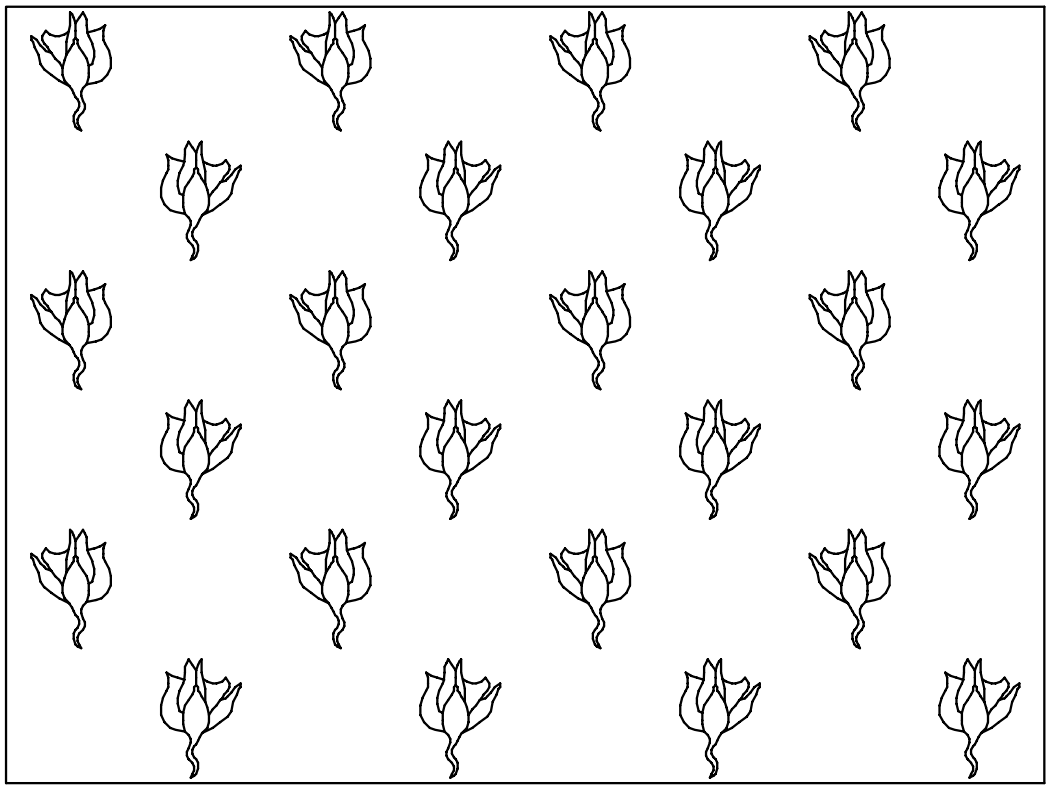
a = 10 Estructura básica de tipo cuadrangular.

b = 1 No existen movimientos giratorios.

c = 3 Simetría deslizada con respecto al eje Y.

d = 1 No existen simetrías con respecto a un eje inclinado.





Notación 10. 1 3 2

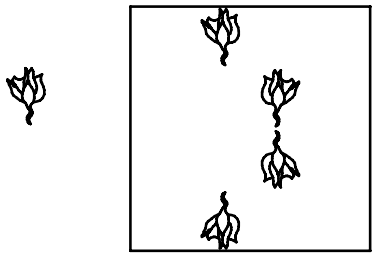
a = 10 Estructura básica de tipo cuadrangular.

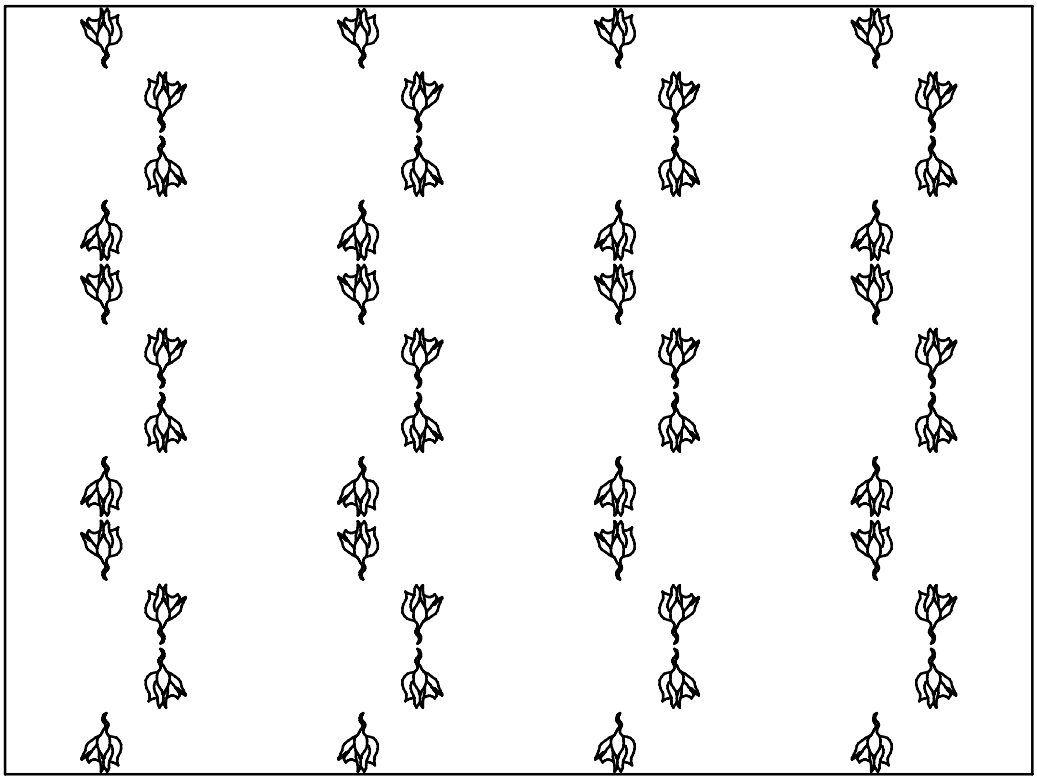
b = 1 No existen movimientos giratorios.

c = 3 Simetría deslizada respecto al eje Y.

d = 2 Simetría con respecto al eje X.

Al utilizar valores distintos de 1 para los símbolos **c** y **d**, se multiplica el número de unidades básicas dentro del área de diseño. Las simetrías con respecto a dos ejes opuestos permiten trasladar al motivo inicial en sentidos diferentes.





Notación 10. 1 3 3

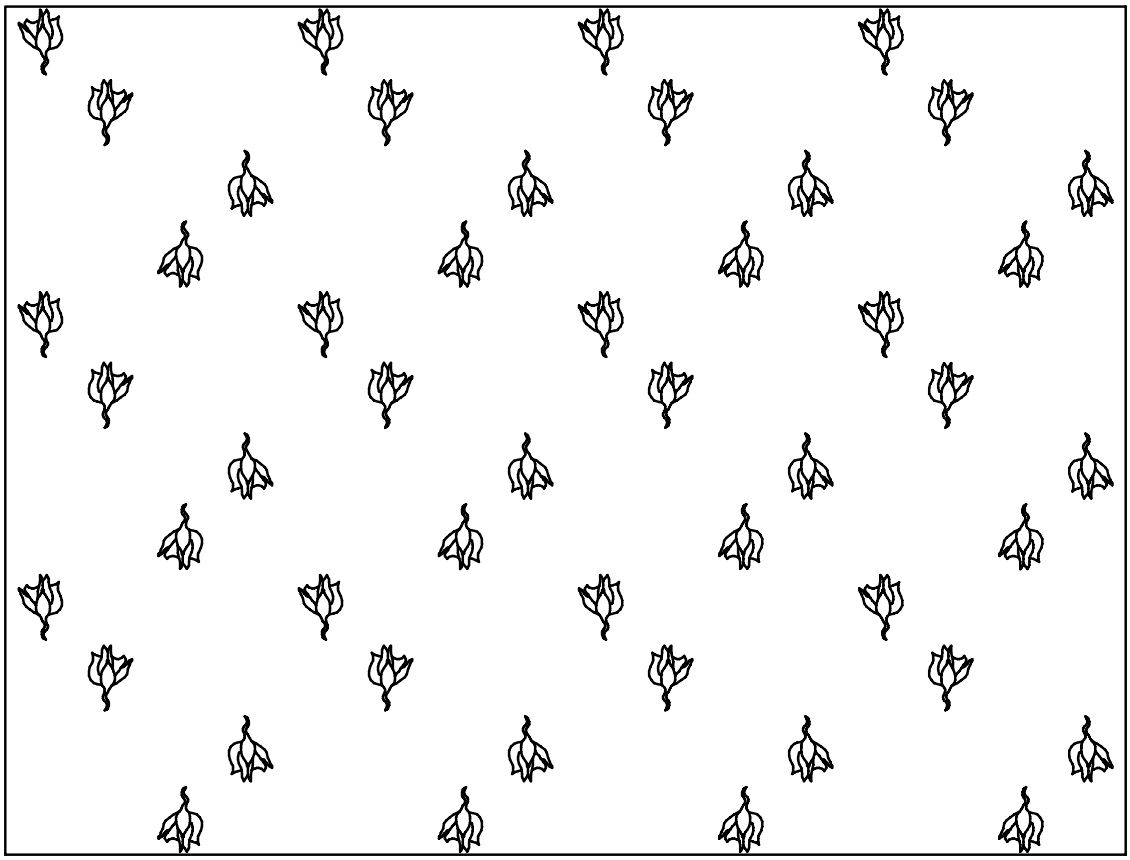
a = 10 Estructura básica de tipo cuadrangular.

b = 1 No existen movimientos giratorios.

c = 3 Simetría deslizada con respecto al eje Y.

d = 3 Simetría deslizada con respecto al eje X.

Si se compara con la notación 10. 1 3 2, se observa una cierta similitud, si bien la diferencia radica en la traslación efectuada con la segunda simetría.



5.5.2. Notación 20. b 1 d

En este apartado se desarrollará la notación **20. b 1 d**, en donde **b** y **d** pueden adoptar todos los valores posibles según se aprecia en la Fig 5.33.

Señalaremos que los 3 primeros casos de estas 15 representaciones coinciden también con las 3 primeras combinaciones analizadas en el estudio del plano **cd,** para **b** = 1, si bien en este caso se ha utilizado una estructura básica rectangular en lugar de cuadrada.

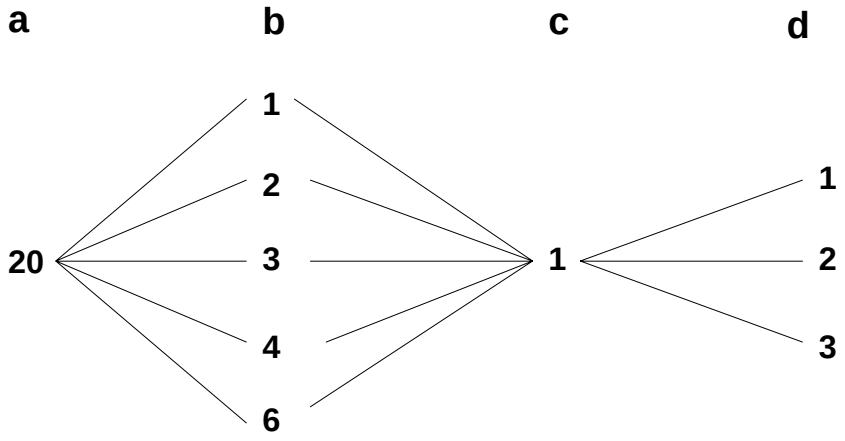


Fig. 5.33

Podemos apreciar en estas 15 posibilidades descritas, los distintos valores de notación que se obtienen, manteniendo constantes los valores de **a** y **c** únicos.

El primer símbolo **a** se refiere al tipo de estructura básica, siendo para todos los casos del tipo rectangular **(a = 20**). Al considerar **c=1**, se indica que no existen simetrías respecto a un eje vertical Y (perpendicular a X) para el patrón inicial elegido.

Solo se van a exponer los casos en que b≠1 por cuanto lo único en que varia es la estructura rectangular en lugar de la cuadrada.

Notación 20. 2 1 1

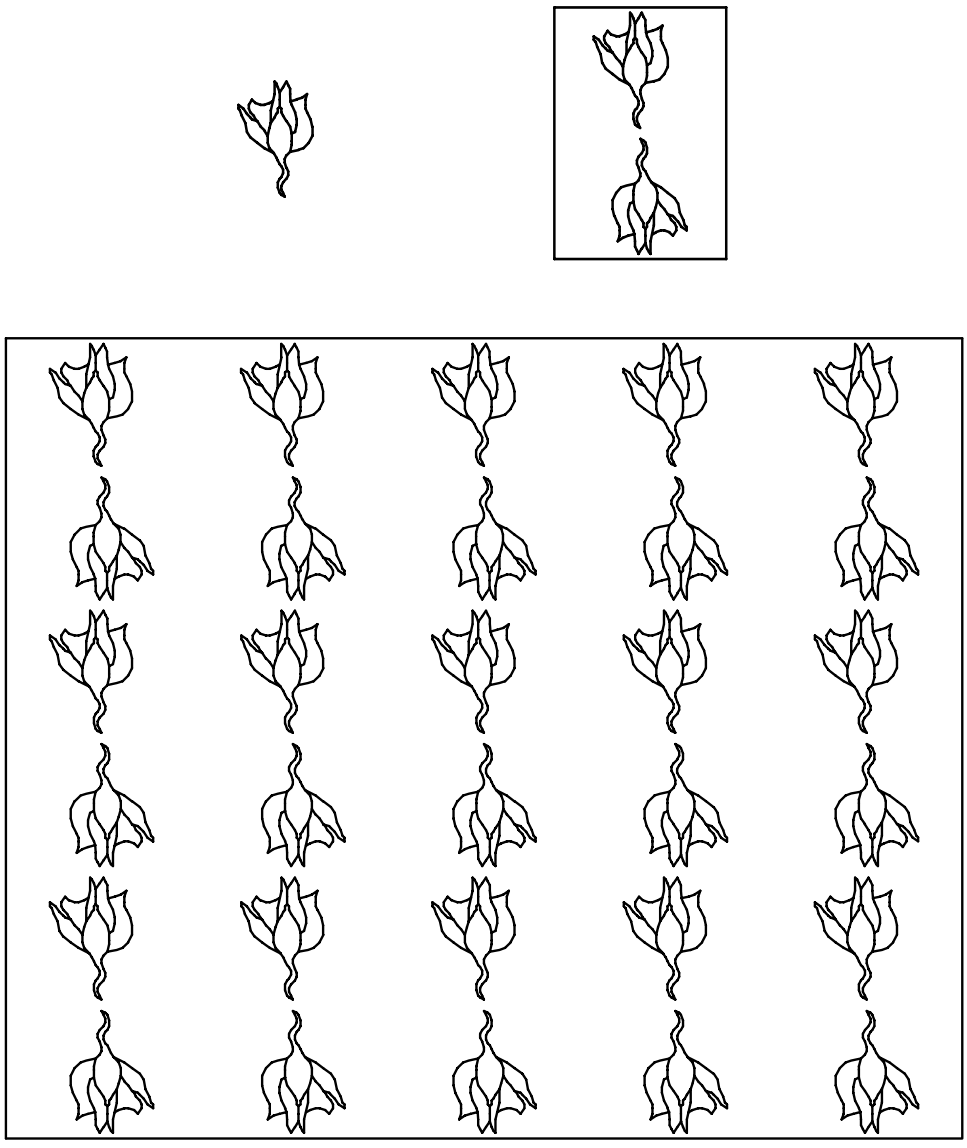
a = 20 Estructura básica de tipo rectangular.

b = 2 Se produce un giro de 180º.

c = 1 No existen simetrías con respecto al eje Y.

d = 1 No existen simetrías con respecto a un eje inclinado.

Al tratarse de un motivo asimétrico el giro se diferencia de la simetría, pero si el motivo fuera simétrico, el resultado coincidiría con la Notación 20.112.



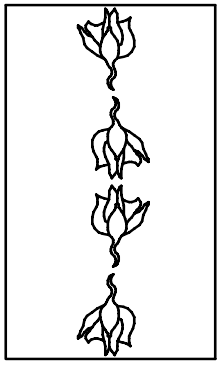
Notación 20. 2 1 2

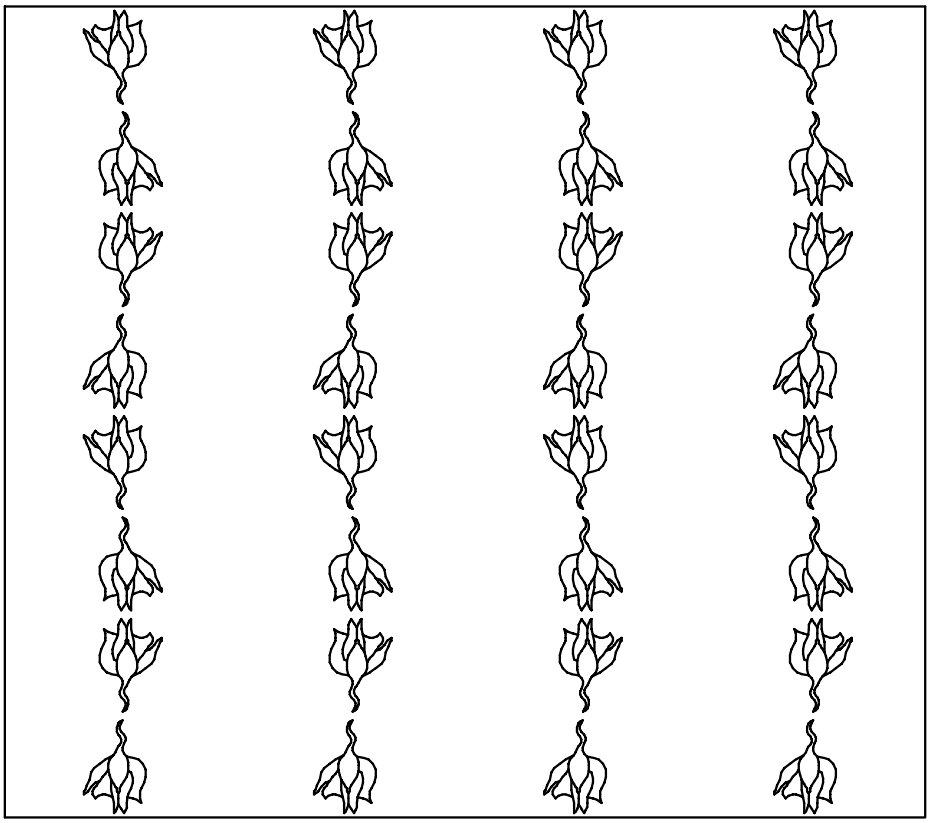
a = 20 Estructura básica de tipo rectangular.

b = 2 Se produce un giro de 180º.

c = 1 No existen simetrías con respecto al eje Y.

d = 2 Simetría con respecto al eje X.





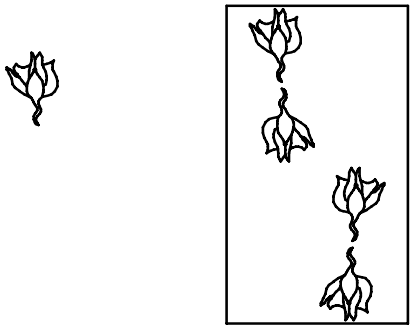
Notación 20. 2 1 3

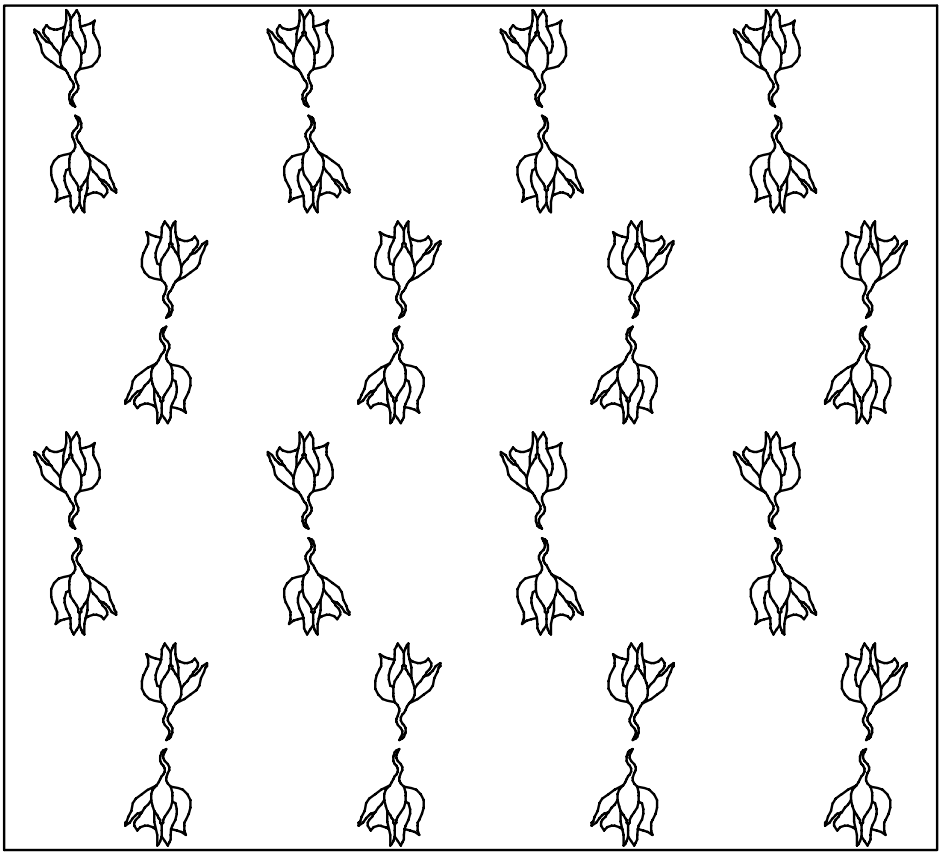
a = 20 Estructura básica de tipo rectangular.

b = 2 Se produce un giro de 180º.

c = 1 No existen simetrías con respecto al eje Y.

d = 3 Simetría con respecto al eje X .





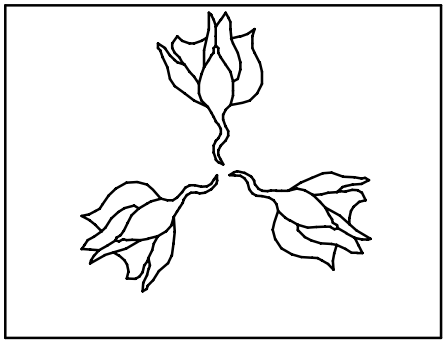
Notación 20. 3 1 1

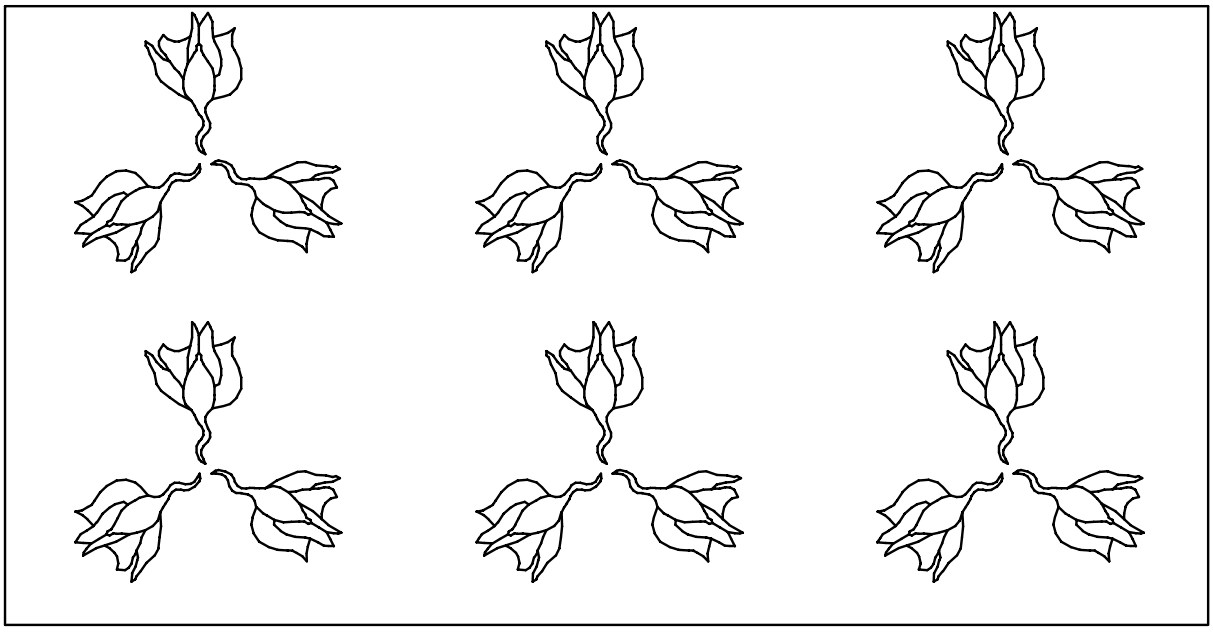
a = 20 Estructura básica de tipo rectangular.

b = 3 Giros de 120º del motivo inicial.

c = 1 No existen simetrías con respecto al eje Y.

d = 1 No existen simetrías con respecto a un eje inclinado.





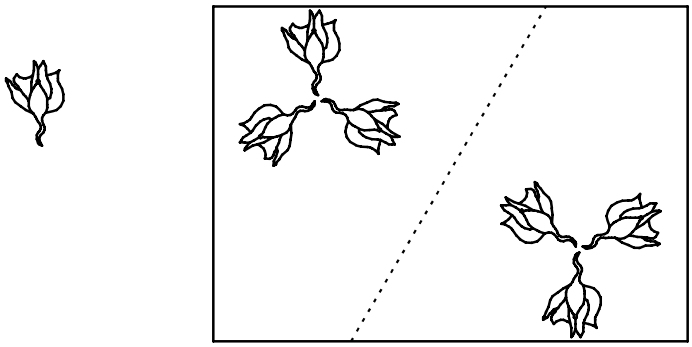
Notación 20. 3 1 2

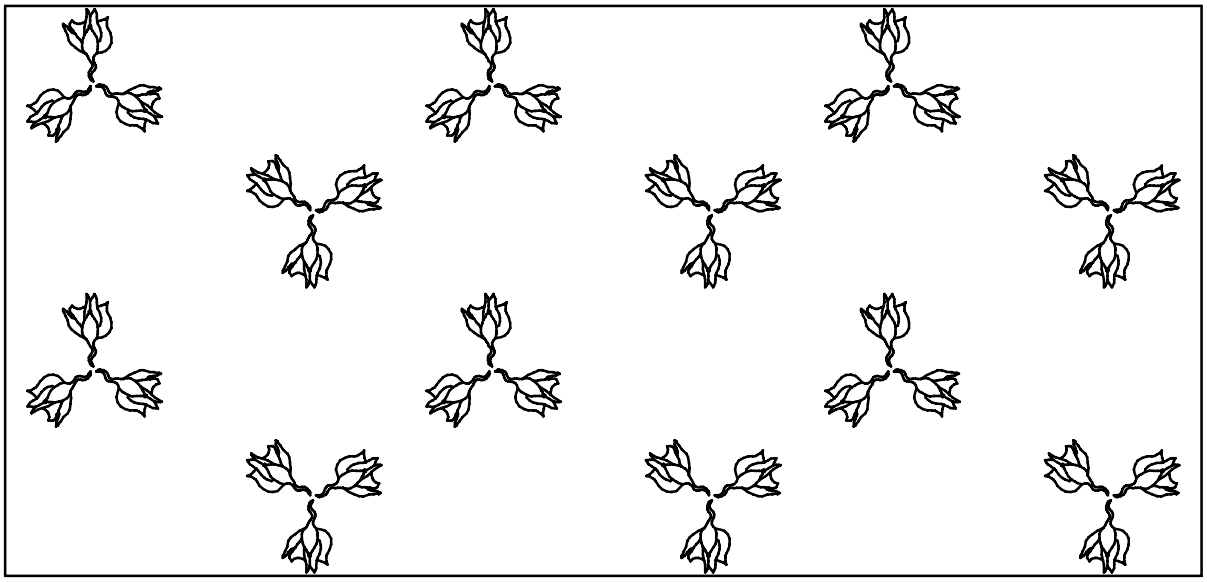
a = 20 Estructura básica de tipo rectangular.

b = 3 Giros de 120º del motivo inicial.

c = 1 No existen simetrías con respecto al eje Y.

d = 2 Simetría deslizada con respecto a un eje inclinado que forma 60º con respecto al eje X.





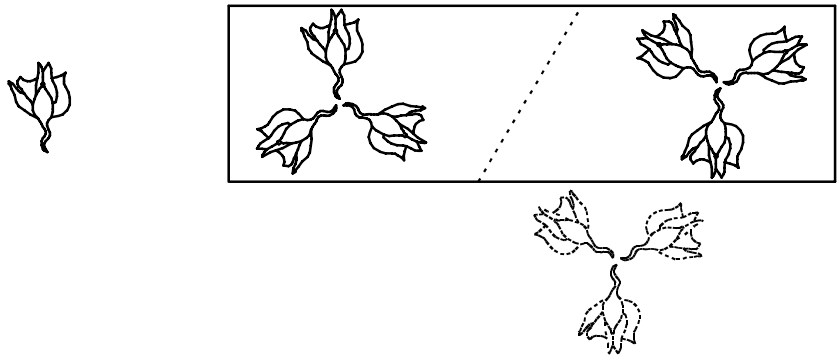
Notación 20. 3 1 3

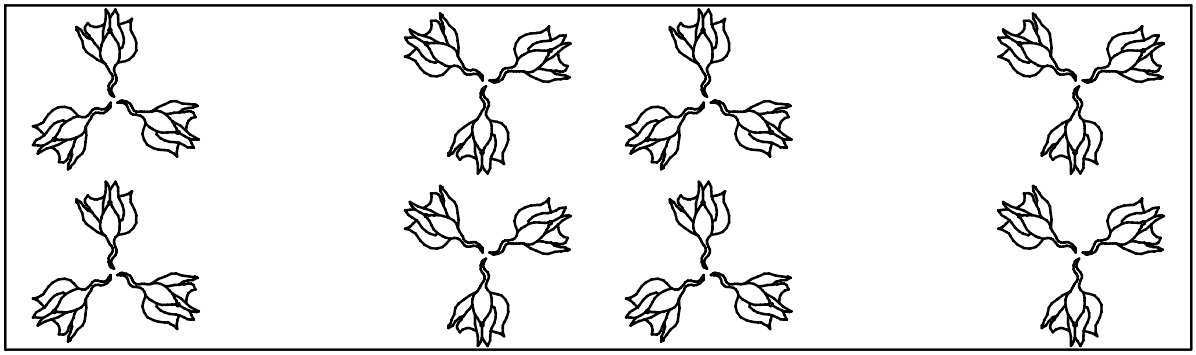
a = 20 Estructura básica de tipo rectangular.

b = 3 Giros de 120º del motivo inicial.

c = 1 No existen simetrías con respecto al eje Y.

d = 3 Simetría deslizada con respecto a un eje inclinado que forma 60º con respecto al eje X.





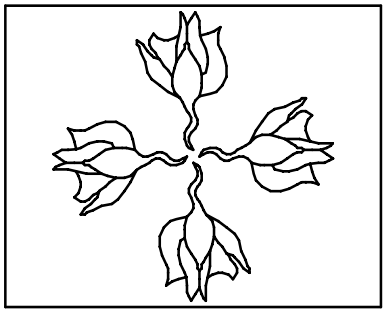
Notación 20. 4 1 1

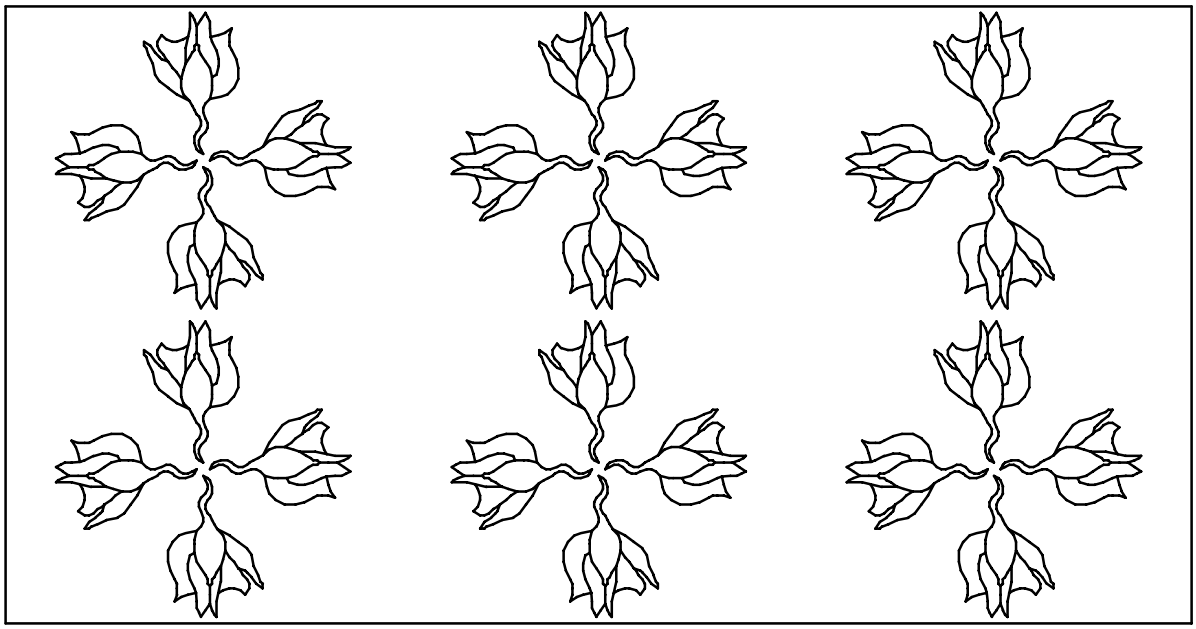
a = 20 Estructura básica de tipo rectangular.

b = 4 Se producen giros de 90º del motivo inicial.

c = 1 No existen simetrías con respecto al eje Y.

d = 1 No existen simetrías con respecto a un eje inclinado.





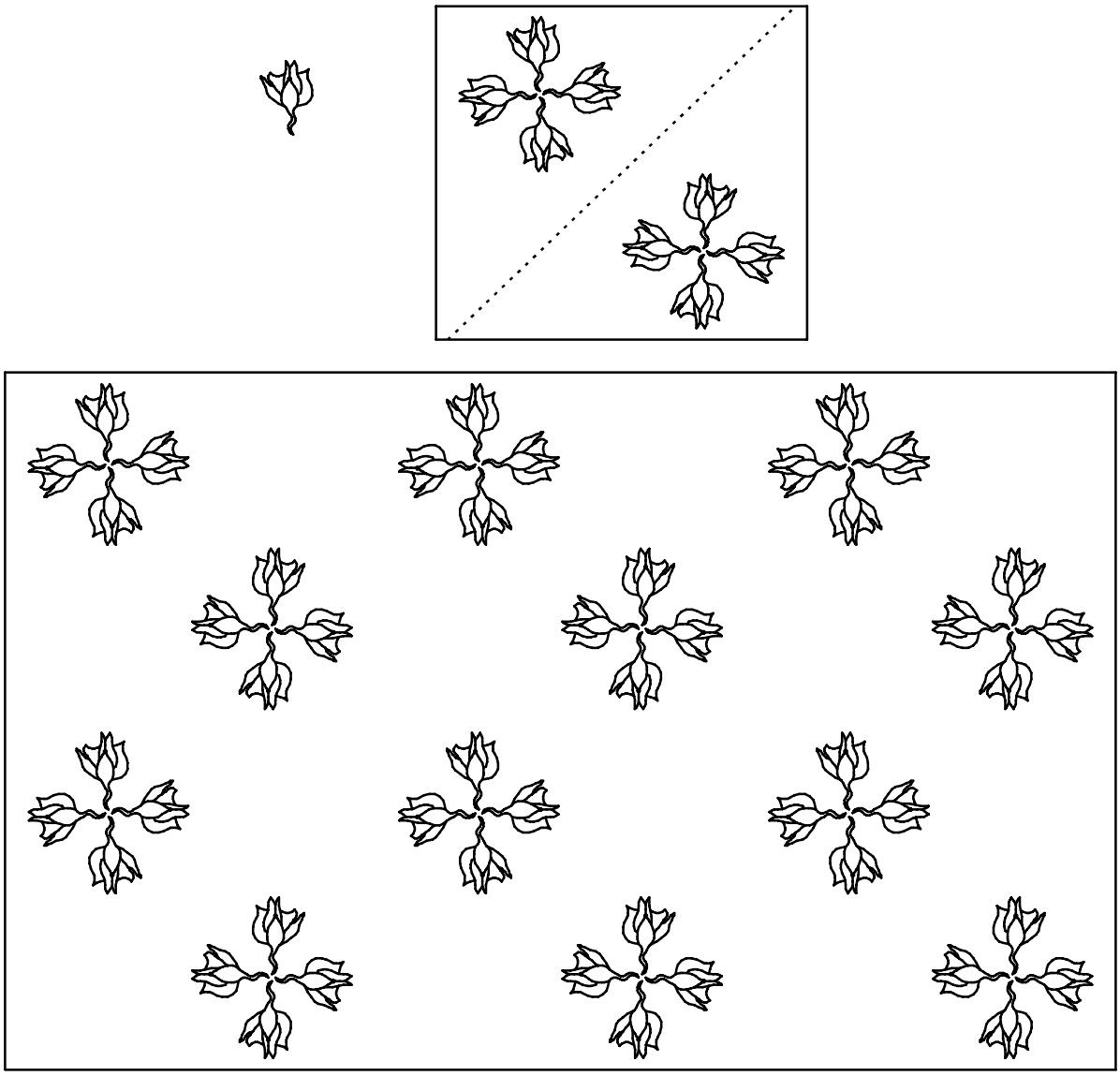
Notación 20. 4 1 2

a = 20 Estructura básica de tipo rectangular.

b = 4 Se producen giros de 90º del motivo inicial.

c = 1 No existen simetrías con respecto al eje Y.

d = 2 Simetría deslizada con respecto a un eje inclinado que forma 45º con respecto al eje X.



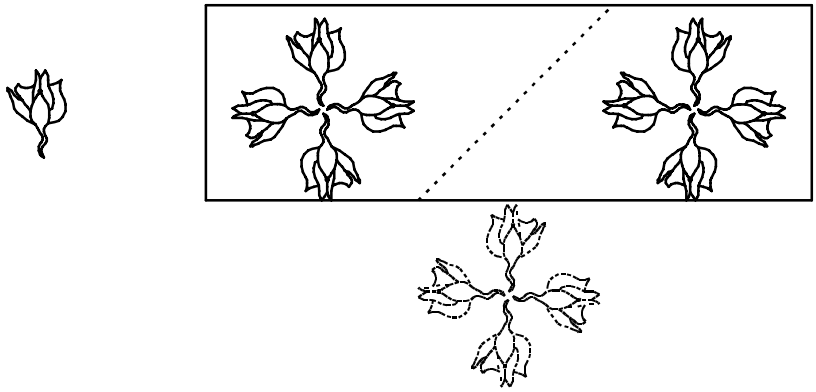
Notación 20. 4 1 3

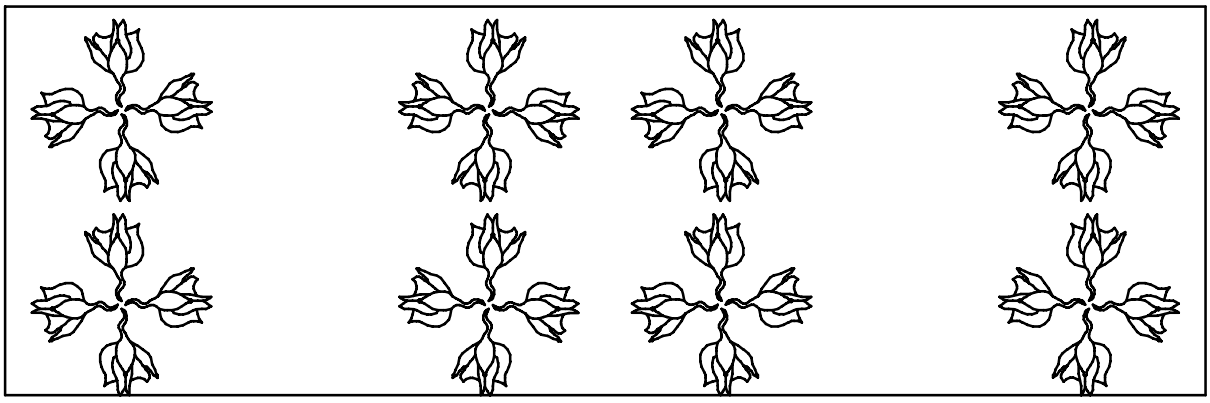
a = 20 Estructura básica de tipo rectangular.

b = 4 Se producen giros de 90º del motivo inicial.

c = 1 No existen simetrías con respecto al eje Y.

d = 3 Simetría deslizada con respecto a un eje inclinado que forma 45º con respecto al eje X.





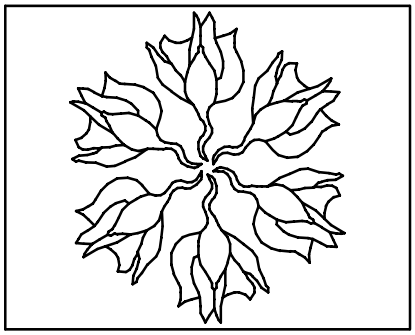
Notación 20. 6 1 1

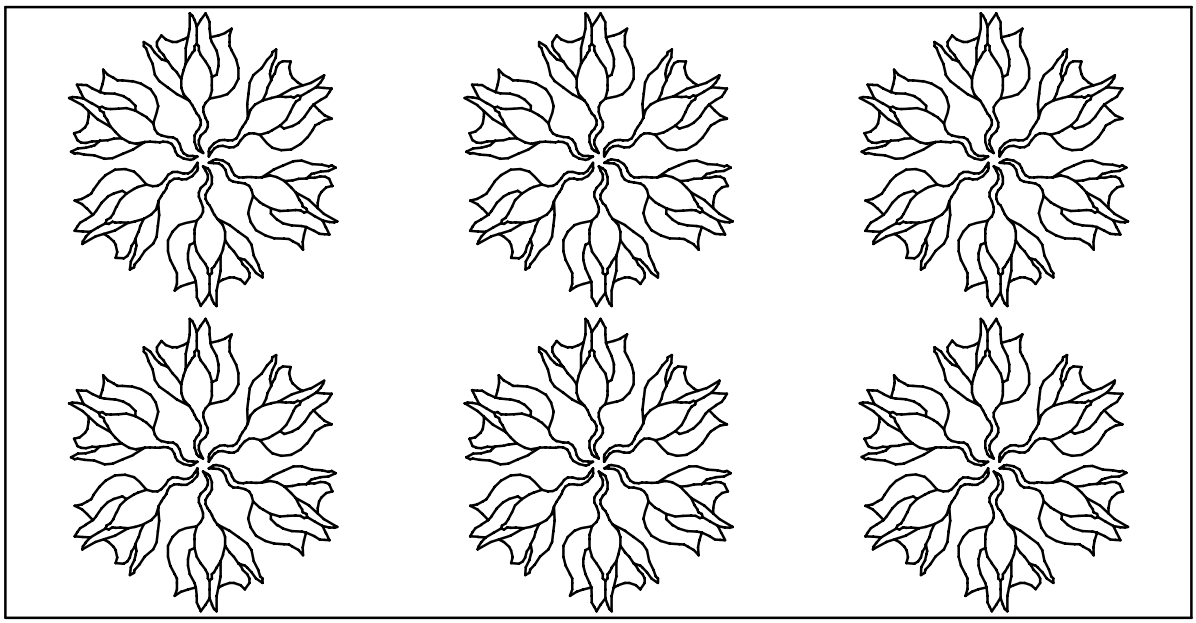
a = 20 Estructura básica de tipo rectangular.

b = 6 Se producen giros de 60º del motivo inicial.

c = 1 No existen simetrías con respecto al eje Y.

d = 1 No existen simetrías con respecto a un eje inclinado.





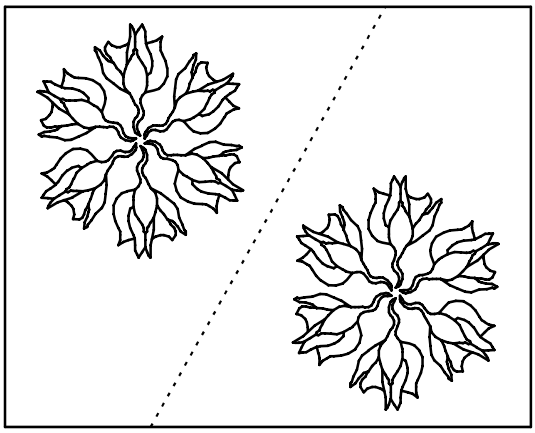
Notación 20. 6 1 2

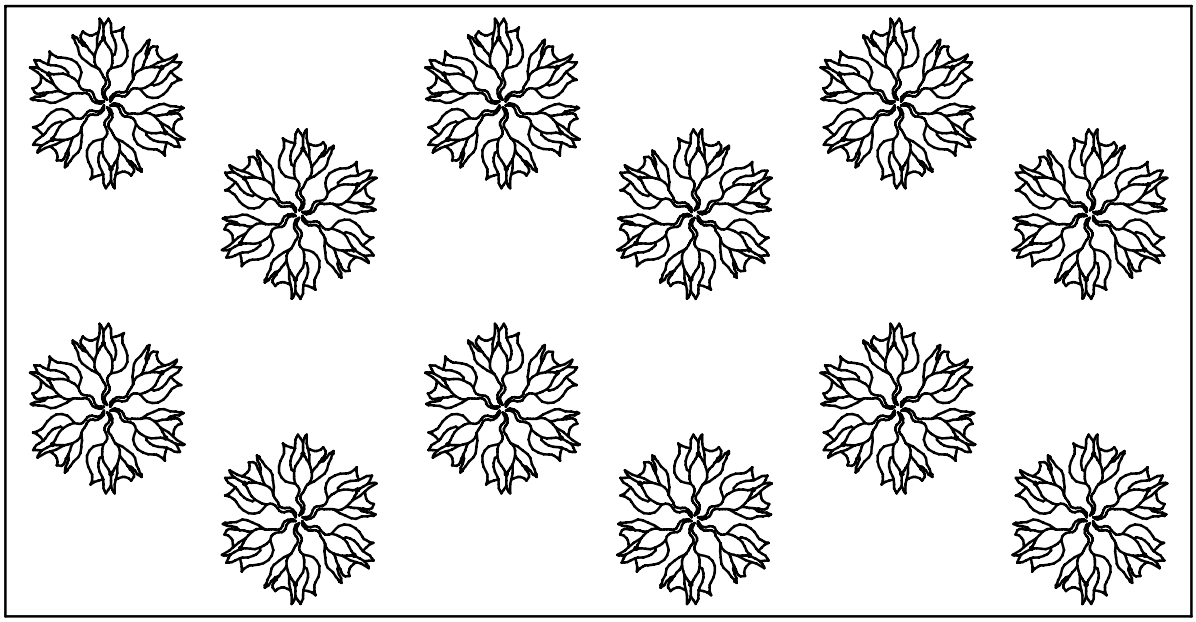
a = 20 Estructura básica de tipo rectangular.

b = 6 Se producen giros de 60º del motivo inicial.

c = 1 No existen simetrías con respecto al eje Y.

d = 2 Simetría deslizada con respecto a un eje inclinado que forma 45º con respecto al eje X.





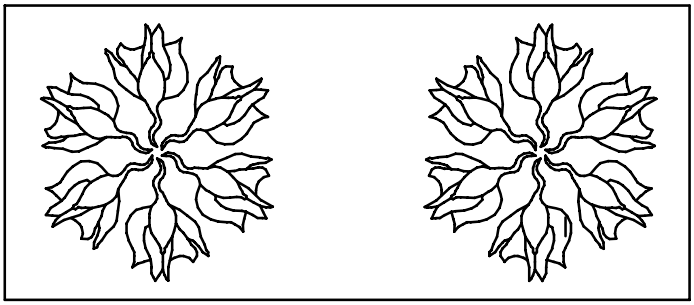
Notación 20. 6 1 3

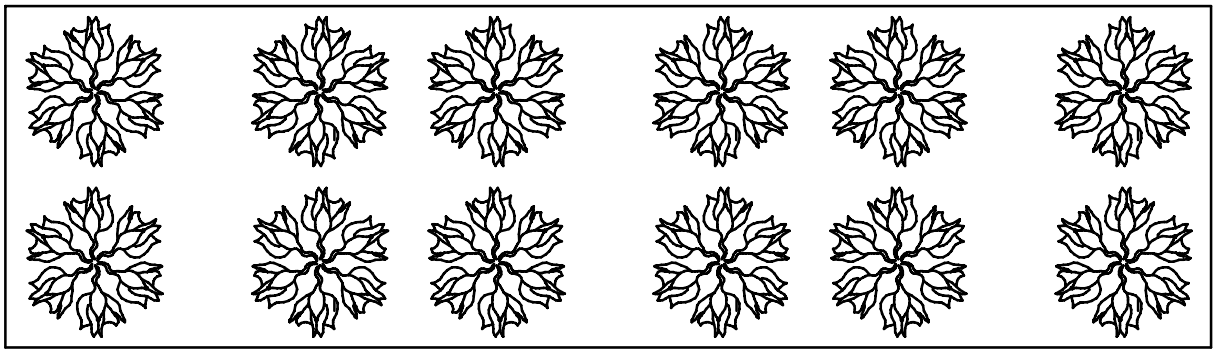
a = 20 Estructura básica de tipo rectangular.

b = 6 Se producen giros de 60º del motivo inicial.

c = 1 No existen simetrías con respecto al eje Y.

d = 3 Simetría deslizada con respecto a un eje inclinado que forma 60º con respecto al eje X.





5.5.3. Notación 10. c b d

Este apartado aumenta las posibilidades de obtención de diseños a partir de un motivo, alterando las posiciones de la notación propuesta **a. b c d** permutando el orden de **c** y **b**, respectivamente, de forma que el esquema de todas las combinaciones posibles considerando una estructura básica cuadrangular se muestra en la fig. 5.34.

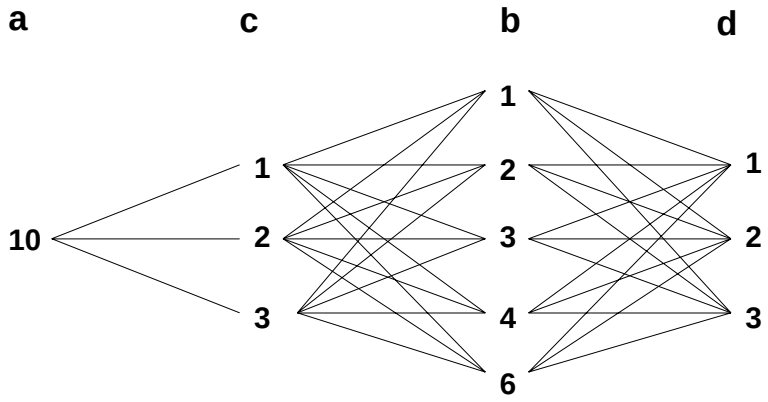


Fig. 5.34

a = Tipo de estructura básica..

c = Simetrías respecto eje Y.

b = Orden de rotaciones.

d = Simetrías respecto un eje con ángulo variable con relación a X

Cuando **b** y **c** son iguales a 1, no influye la permutación de ambos y los resultados obtenidos serían los indicados anteriormente, dado que el resultado de aplicar la nueva notación daría lugar a figuras equivalentes.

a. b c d		a. c b d
10. 1 1 1	=	10. 1 1 1

En el esquema anterior se aprecia que con esta permutación de **b** y **c** existen 45 combinaciones en total, de las cuales no vamos a considerar el caso **c** = 1 por lo anteriormente expuesto. El esquema con las nuevas combinaciones es el que se muestra

en la Fig. 5.35 y sobre dichas posibilidades se van a desarrollar algunos diseños que se expondrán a continuación.

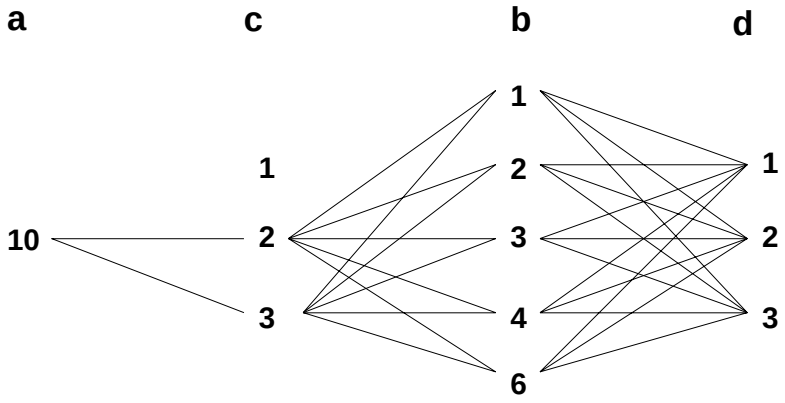


Fig. 5.35

Para no confundir la notación de los valores que se pueden adoptar según el esquema anterior, con la notación **a. b c d** que se ha desarrollado hasta el momento, (Por Ejemplo 10.2 2 3), se adopta el criterio de incluir las letras **c b d** en los distintos valores que se consideren.(Ejemplo 10.c2 b2 d3).

Al motivo inicial asimétrico distinto del anterior según Fig. 5.36 se incluirá en una estructura básica del tipo cuadrangular **a = 10**, y se le practicarán algunos de los movimientos indicados en el esquema anterior.

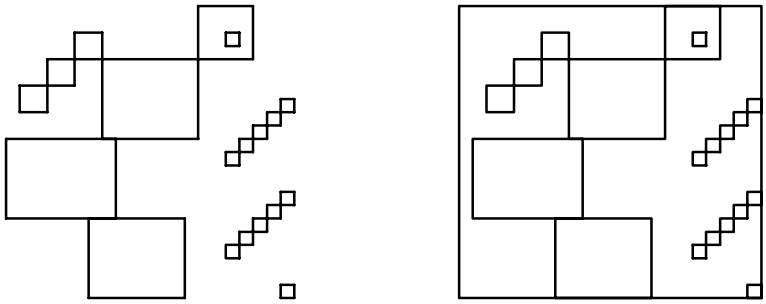


Fig. 5.36

Notación 10. c2 b2 d1

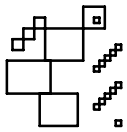
a = 10 Estructura básica cuadrada.

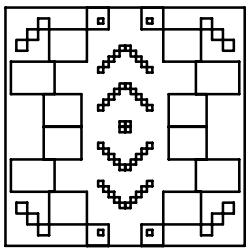
c = 2 Simetría respecto el eje Y.

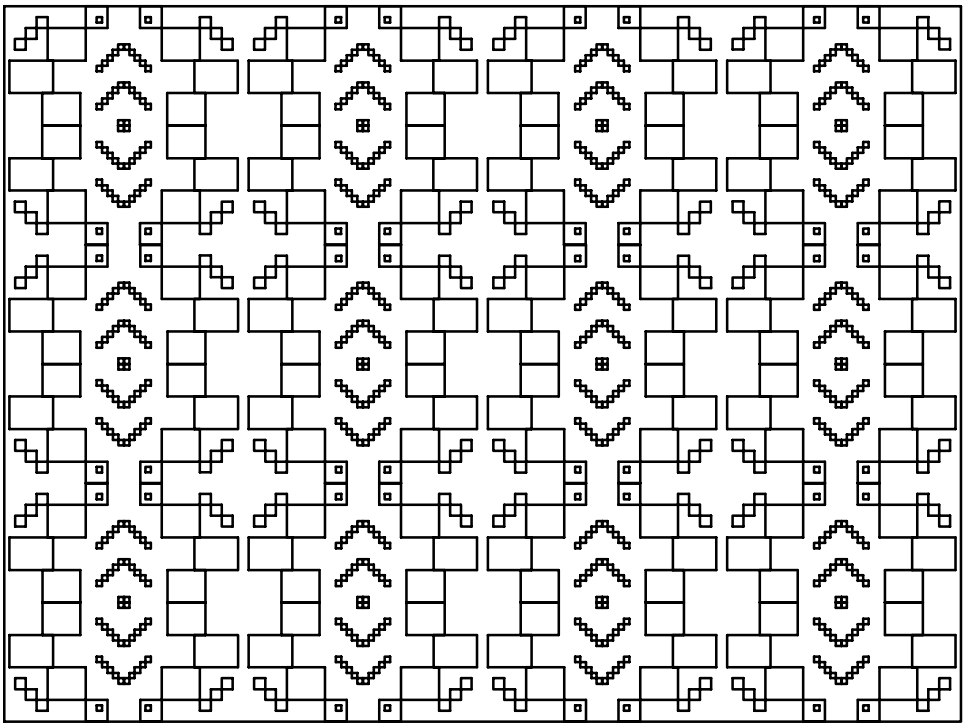
b = 2 Se aplica un giro de 180º.

d = 1 No existen simetrías con respecto a un eje inclinado.

Aunque el motivo inicial es asimétrico, con el primer movimiento se ha transformado en uno simétrico; al aplicarle el movimiento de rotación b=2 se obtiene el mismo resultado al de aplicar una simetría en X. Así pues este 10. c2 b2 d1 equivale a 10. c2 b1 d2.







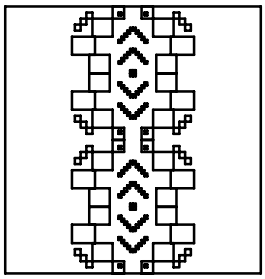
Notación 10. c2 b2 d2

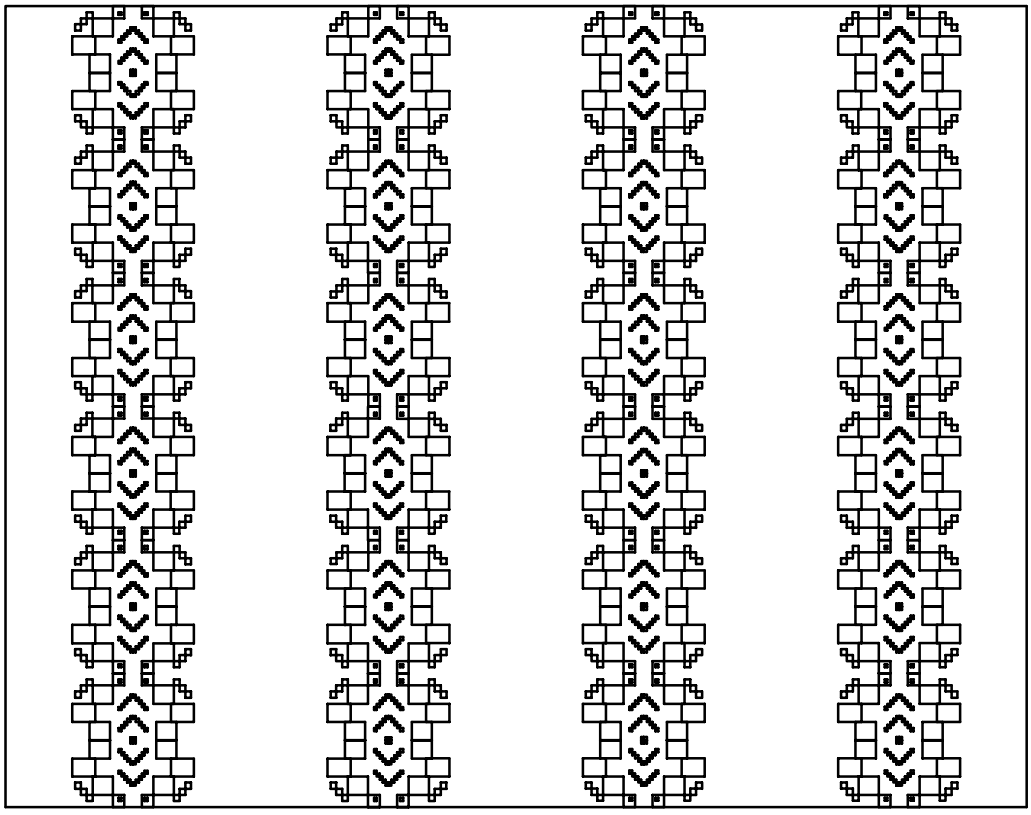
a = 10 Estructura básica cuadrada.

c = 2 Simetría respecto el eje Y.

b = 2 Se aplica un giro de 180º.

d = 2 Simetría respecto eje X por cuanto al ser b=2 el ángulo que corresponde al eje inclinado es de 180º y coincide con el eje X.





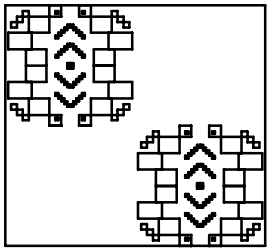
Notación 10. c2 b2 d3

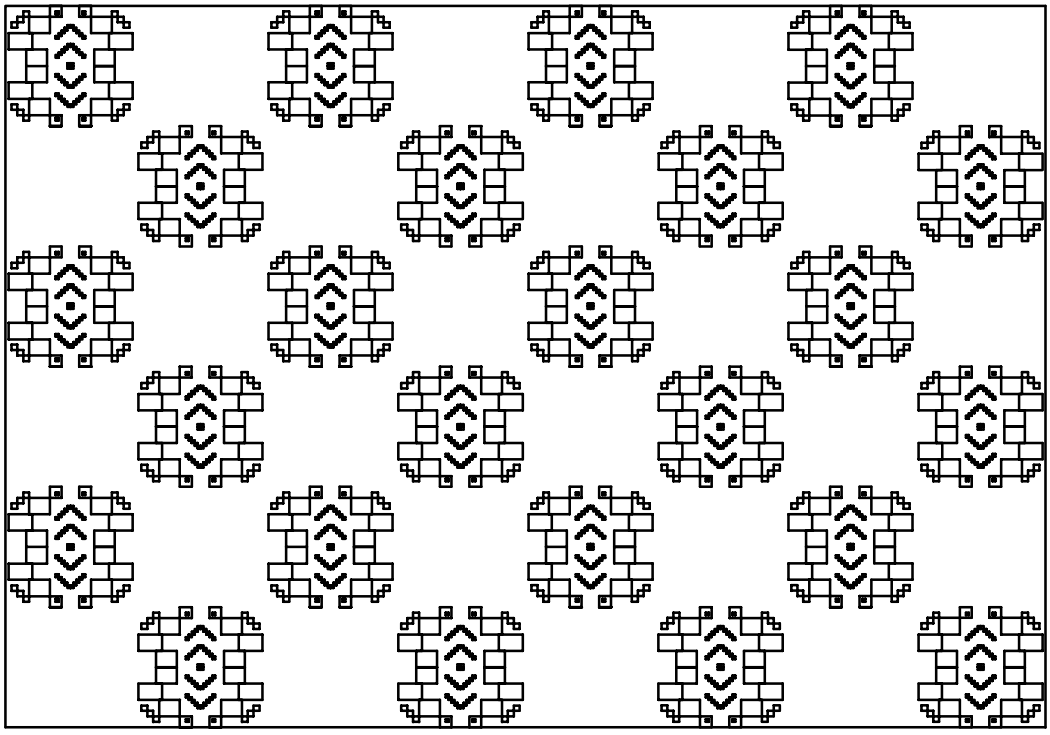
a = 10 Estructura básica cuadrada.

c = 2 Simetría respecto el eje Y.

b = 2 Se aplica un giro de 180º.

d = 3 En este caso la simetría con respecto el eje Y es deslizada.





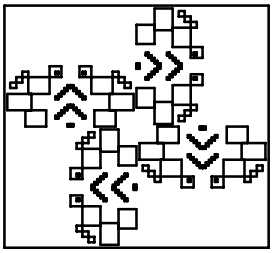
Notación 10. c2 b4 d1

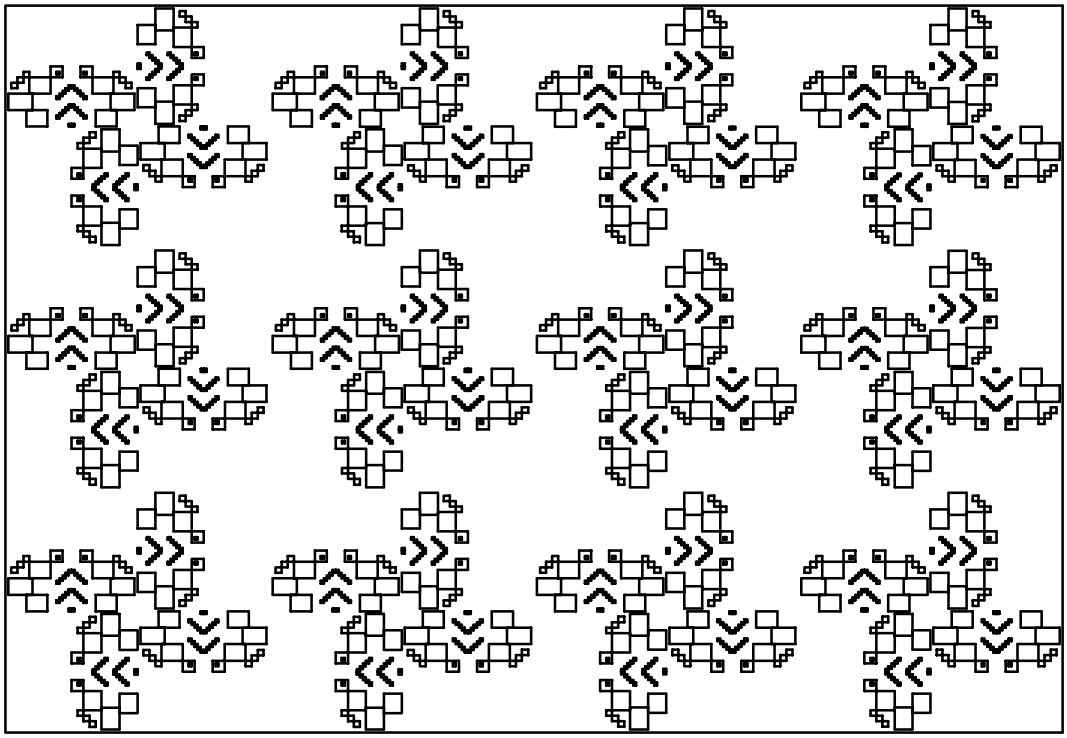
a = 10 Estructura básica cuadrada.

c = 2 Simetría respecto el eje Y.

b = 4 Se aplican giros de 90º.

d = 1 No existen simetrías con respecto a un eje inclinado.





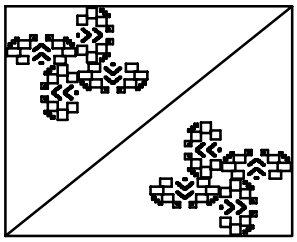
Notación 10. c2 b4 d2

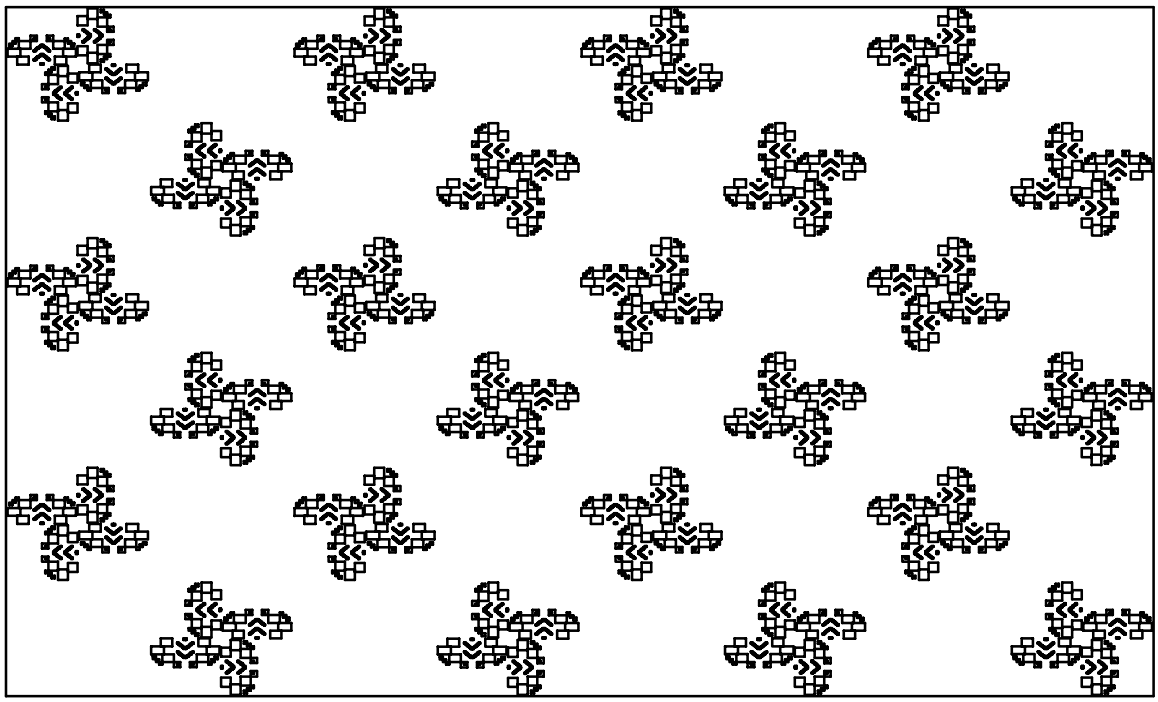
a = 10 Estructura básica cuadrada.

c = 2 Simetría respecto el eje Y.

b = 4 Se aplican giros de 90º.

d = 2 Simetría respecto un eje inclinado a 45º.





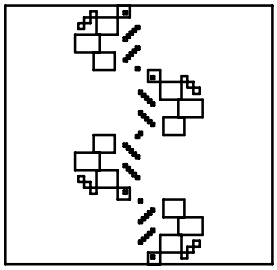
Notación 10. c3 b2 d1

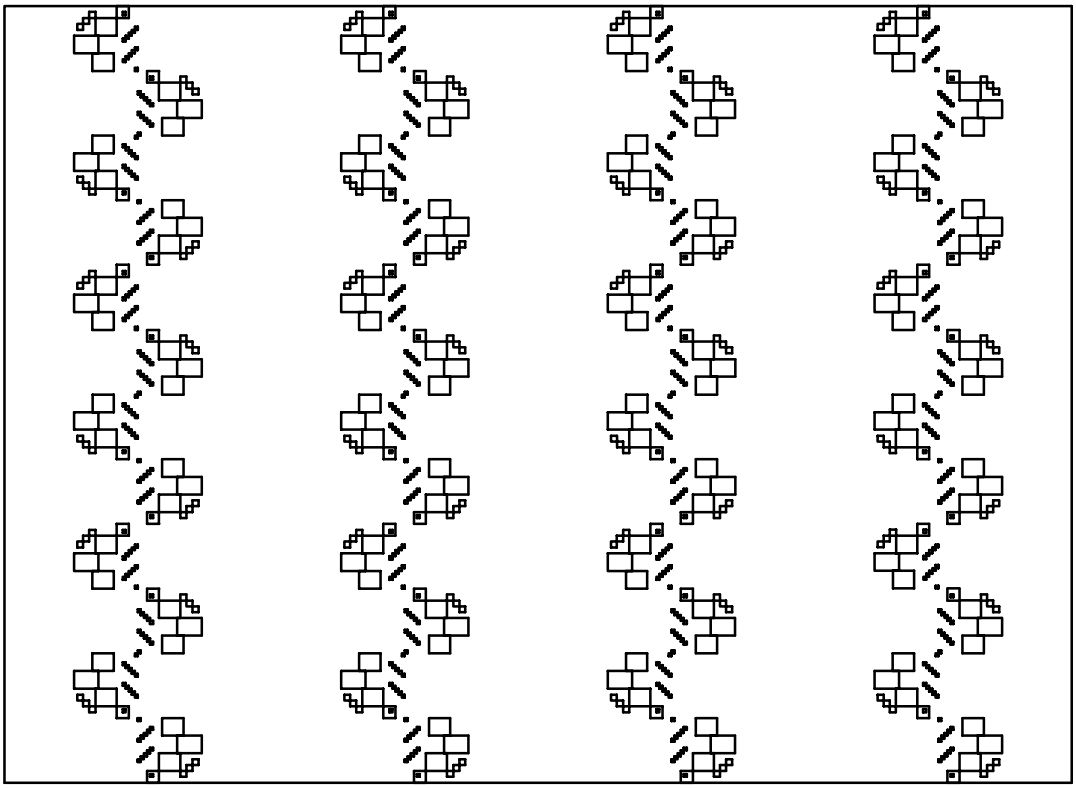
a = 10 Estructura básica cuadrada.

c = 3 Simetría deslizada respecto el eje Y.

b = 2 Se aplica un giro de 180º.

d = 1 No existen simetrías con respecto a un eje inclinado.





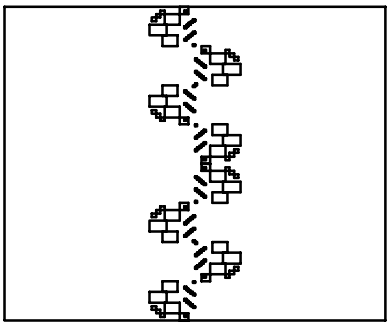
Notación 10. c3 b2 d2

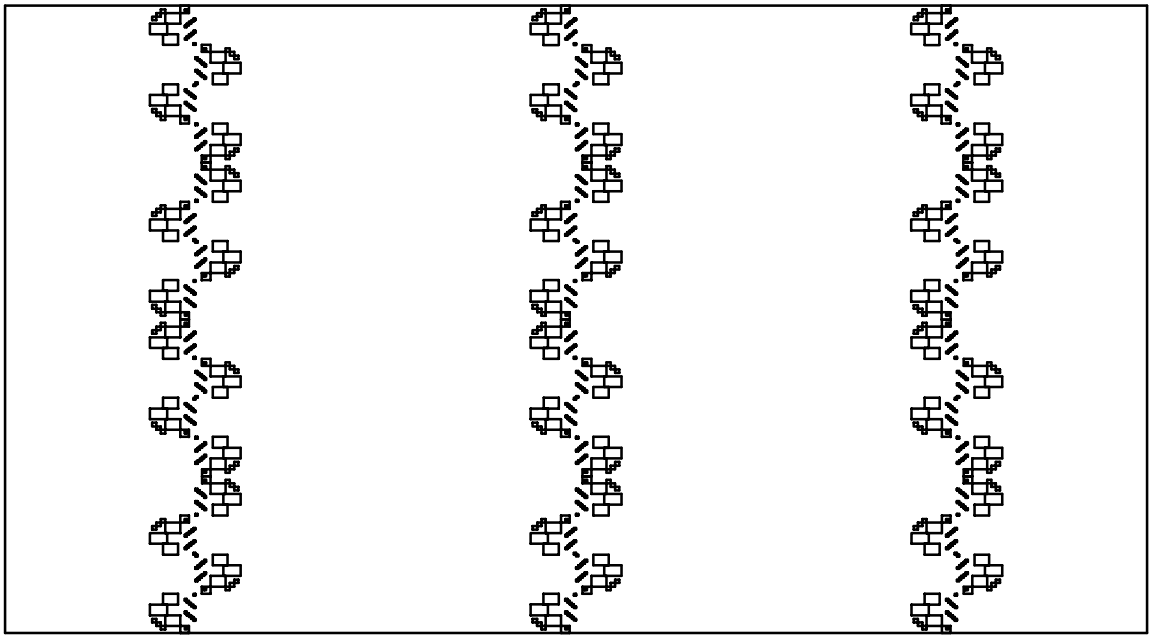
a = 10 Estructura básica cuadrada.

c = 3 Simetría deslizada respecto el eje Y.

b = 2 Se aplica un giro de 180º.

d = 2 Simetría respecto eje X. (ángulo 180º)





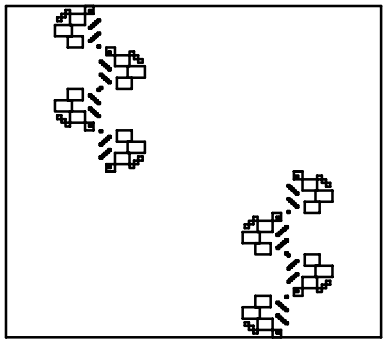
Notación 10. c3 b2 d3

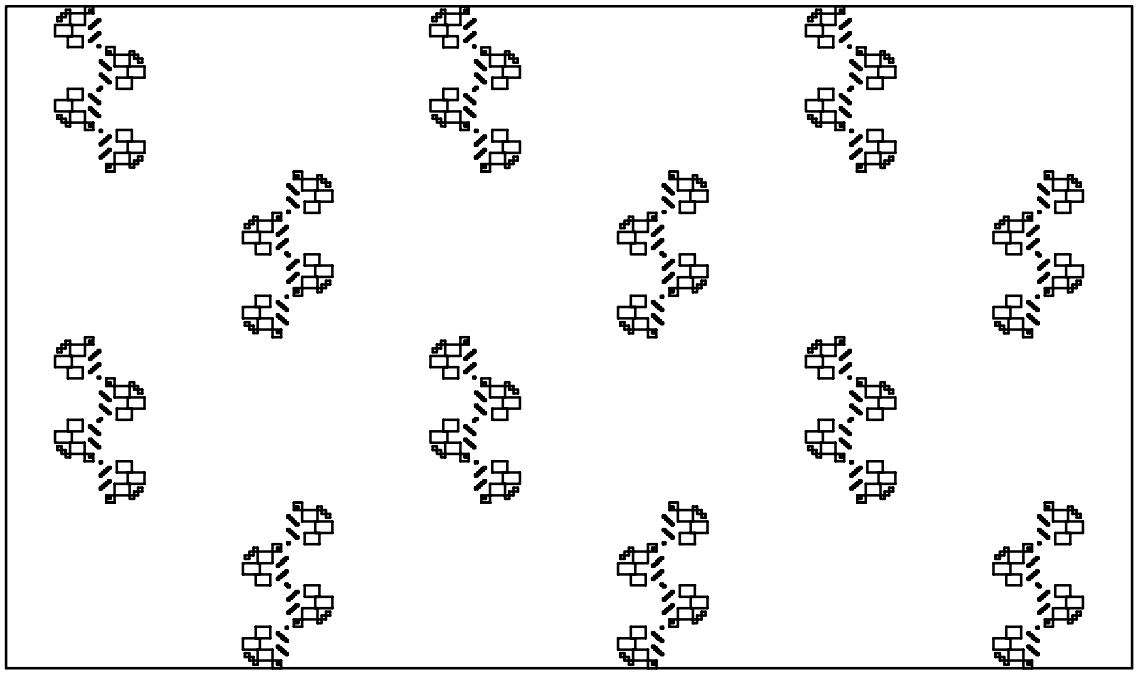
a = 10 Estructura básica cuadrada.

c = 3 Simetría deslizada respecto el eje Y.

b = 2 Se aplica un giro de 180º.

d = 3 Simetría deslizada respecto eje X (ángulo de 180º).





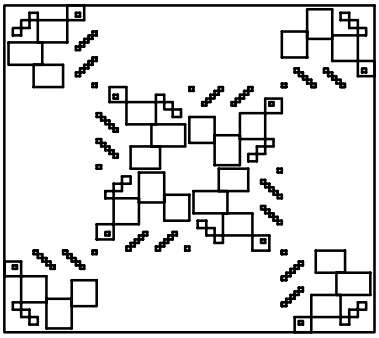
Notación 10. c3 b4 d1

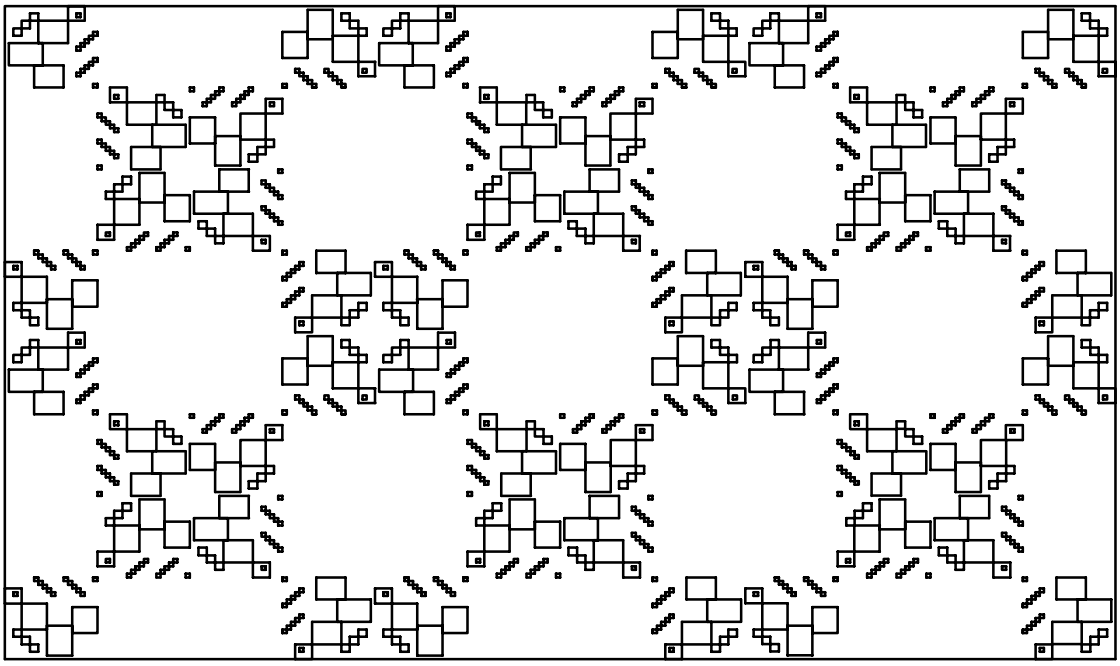
a = 10 Estructura básica cuadrada.

c = 3 Simetría deslizada respecto el eje Y.

b = 4 Se aplican giros de 90º.

d = 1 No existen simetrías con respecto a un eje inclinado.





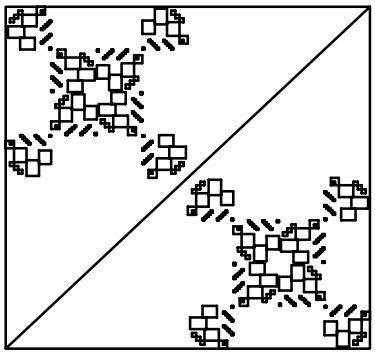
Notación 10. c3 b4 d2

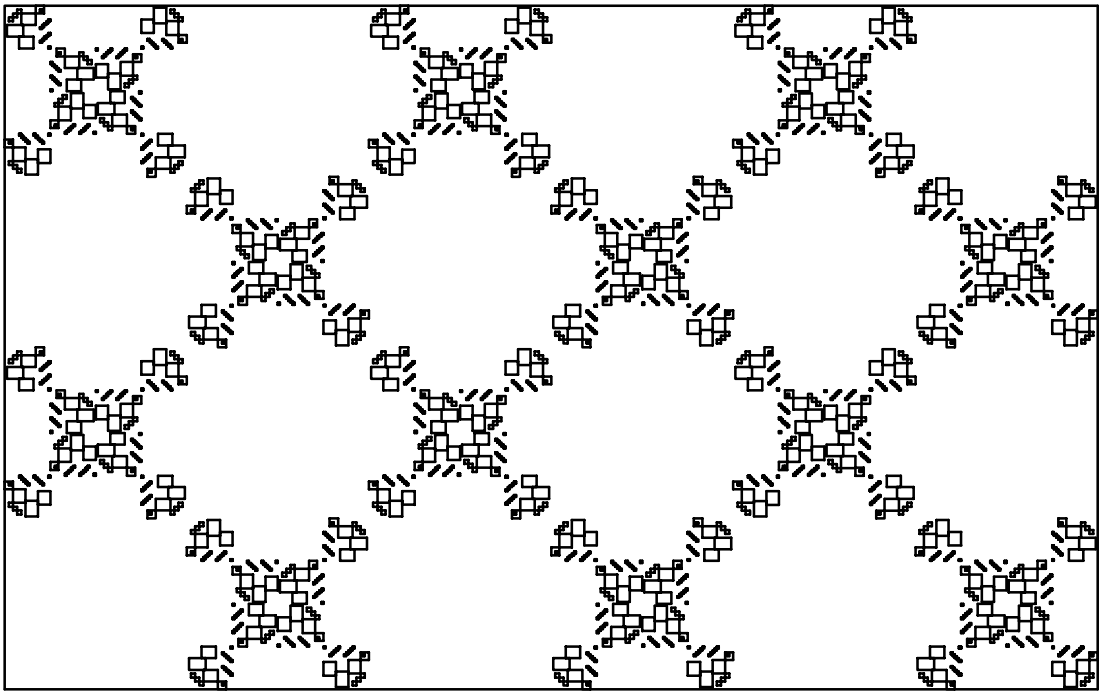
a = 10 Estructura básica cuadrada.

c = 3 Simetría deslizada respecto el eje Y.

b = 4 Se aplican giros de 90º.

d = 2 Simetría respecto un eje inclinado a 45º.





5.5.4. Notación 20. c d b

Una nueva posibilidad de permutación de los índices de la notación seguida es la que se indica a continuación cambiando los índices **b** y **d** por **d** y **b** del apartado anterior según **a. c d b** de forma que adoptando **a=20** se van a exponer algunos de los casos que nos ofrece el diagrama esquemático en la fig. 5.37 con un nuevo motivo inicial asimétrico como el de la fig. 5.38 eliminando los que se pueden obtener por notaciones vistas anteriormente.

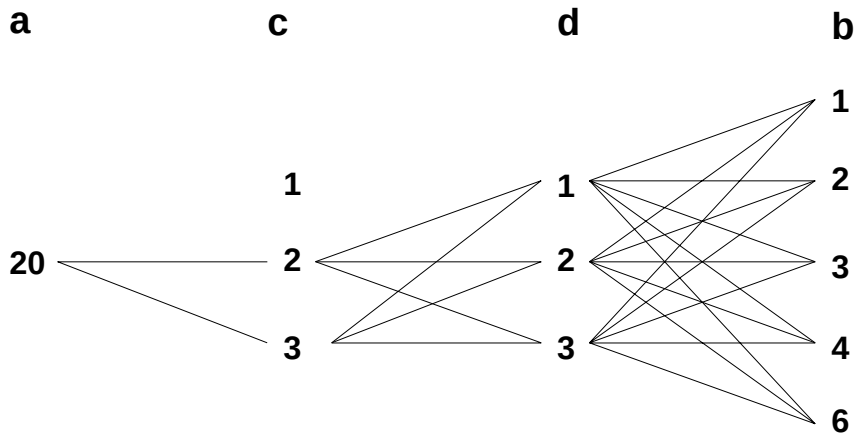


Fig. 5.37

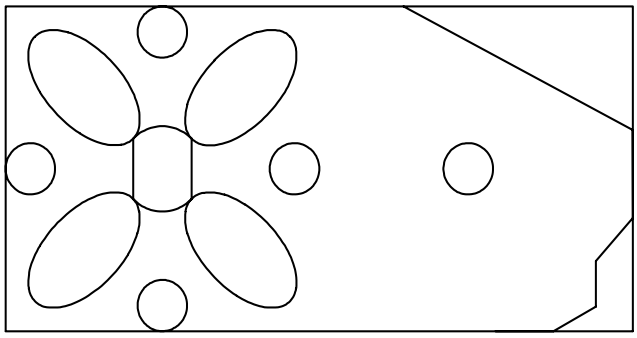


Fig. 5.38

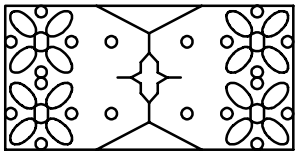
Notación 20. c2 d1 b2

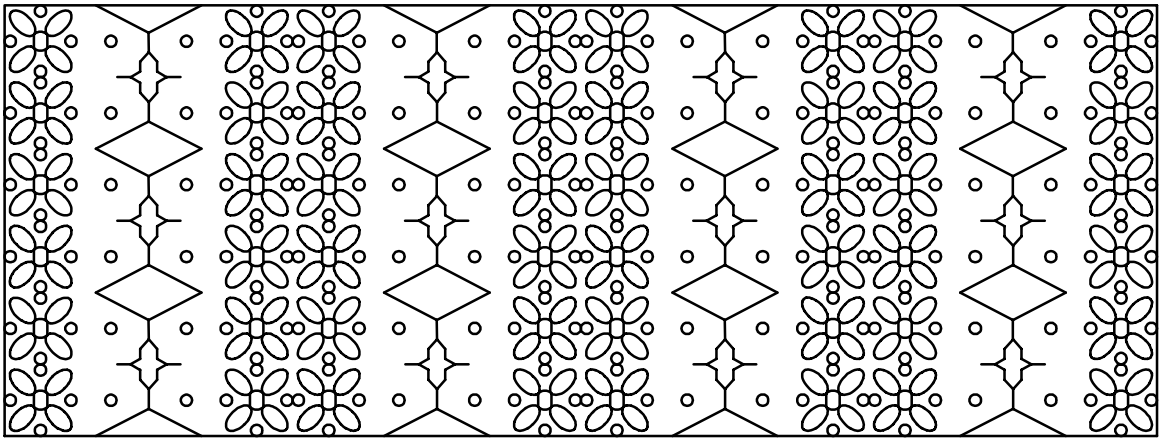
a = 20 Estructura básica rectangular.

c = 2 Simetría respecto el eje Y.

d = 1 No existen simetrías con respecto a un eje inclinado.

b = 2 Se aplica un giro de 180º.





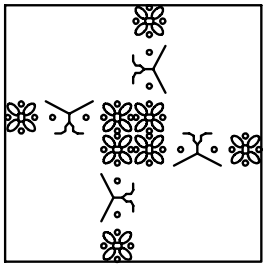
Notación 20. c2 d1 b4

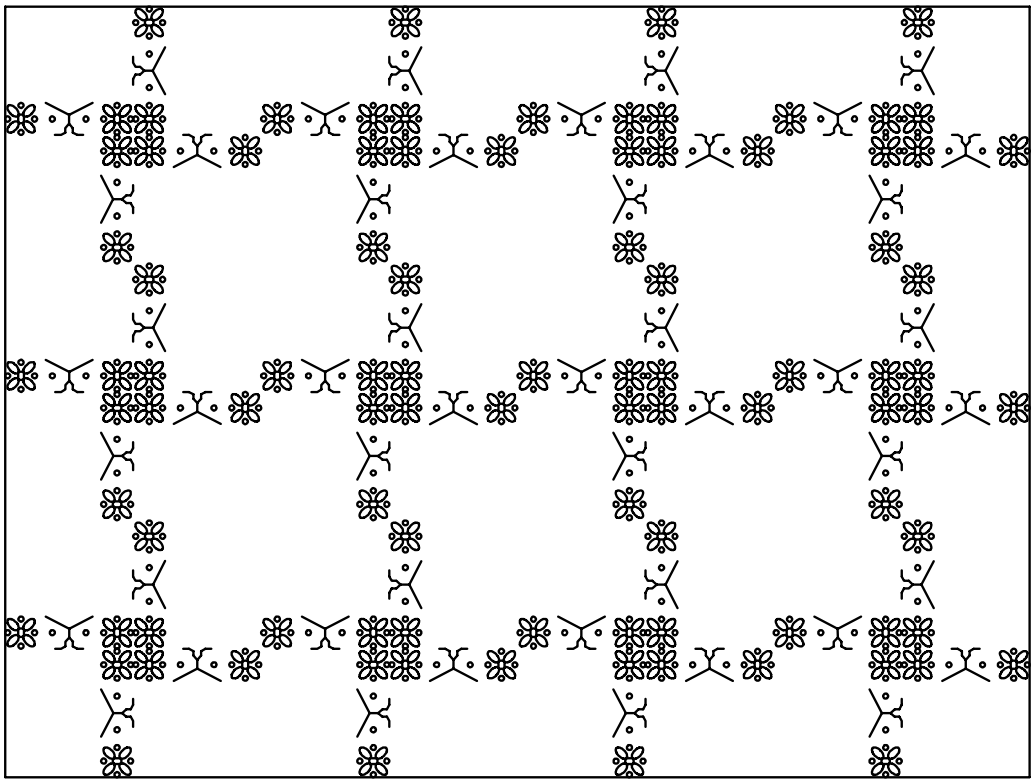
a = 20 Estructura básica rectangular.

c = 2 Simetría respecto el eje Y.

d = 1 No existen simetrías con respecto a un eje inclinado.

b = 4 Se aplican cuatro giros de 90º.





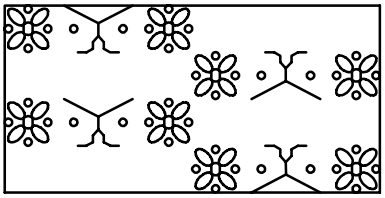
Notación 20. c2 d3 b2

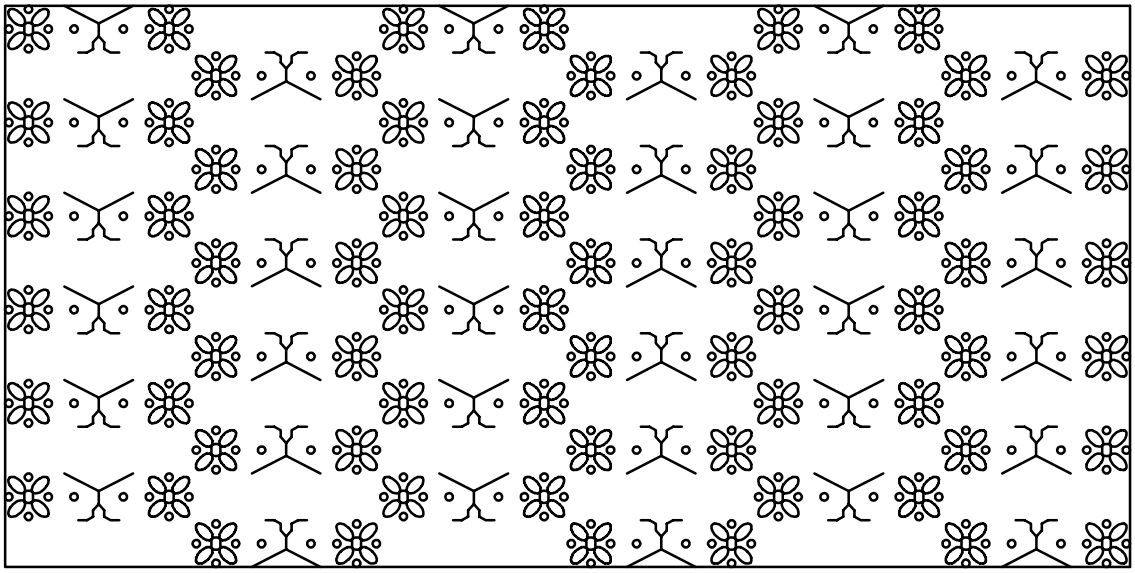
a = 20 Estructura básica rectangular.

c = 2 Simetría respecto el eje Y.

d = 3 Simetría deslizada respecto eje Y.

b = 2 Se aplica un giro de 180º.





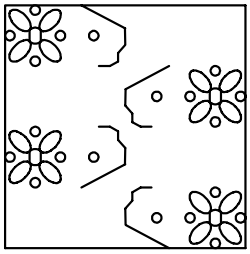
Notación 20. c3 d1 b2

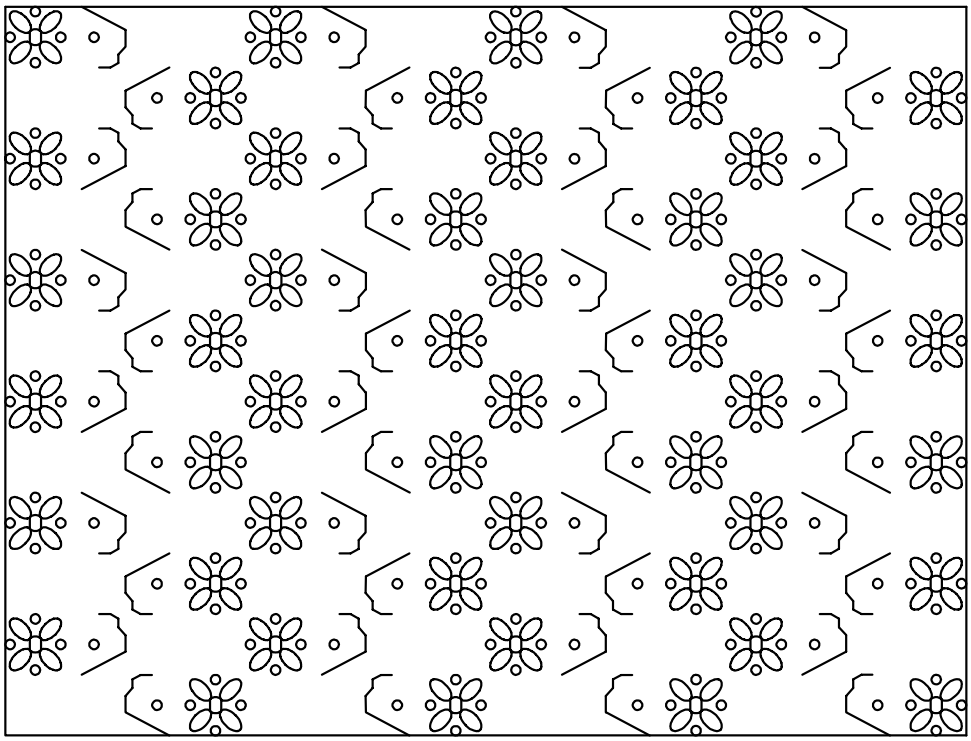
a = 20 Estructura básica rectangular.

c = 3 Simetría deslizada respecto el eje Y.

d = 1 No existen simetrías con respecto a un eje inclinado.

b = 2 Se aplica un giro de 180º.





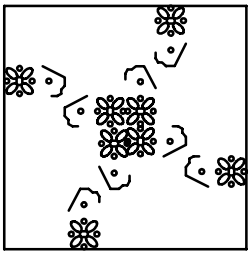
Notación 20. c3 d1 b4

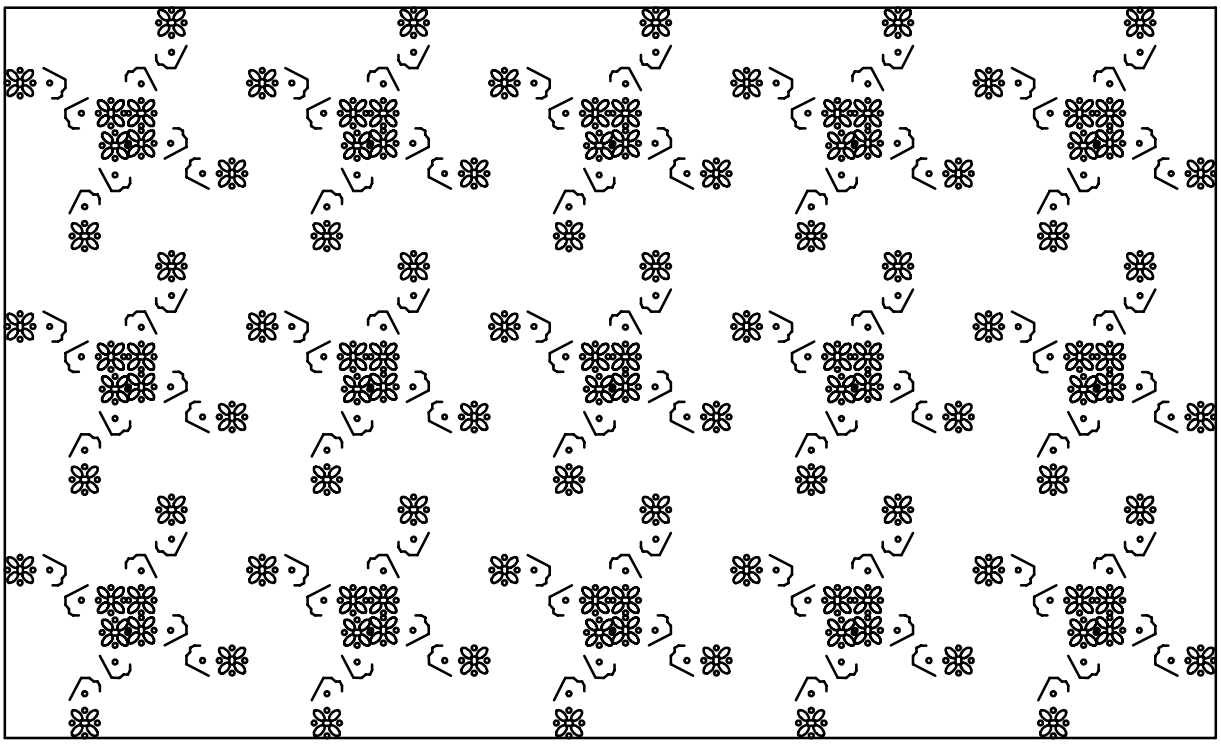
a = 20 Estructura básica rectangular.

c = 3 Simetría deslizada respecto el eje Y.

d = 1 No existen simetrías con respecto a un eje inclinado.

b = 4 Se aplican cuatro giros de 90º.





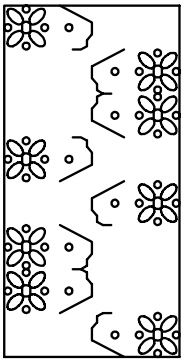
Notación 20. c3 d2 b2

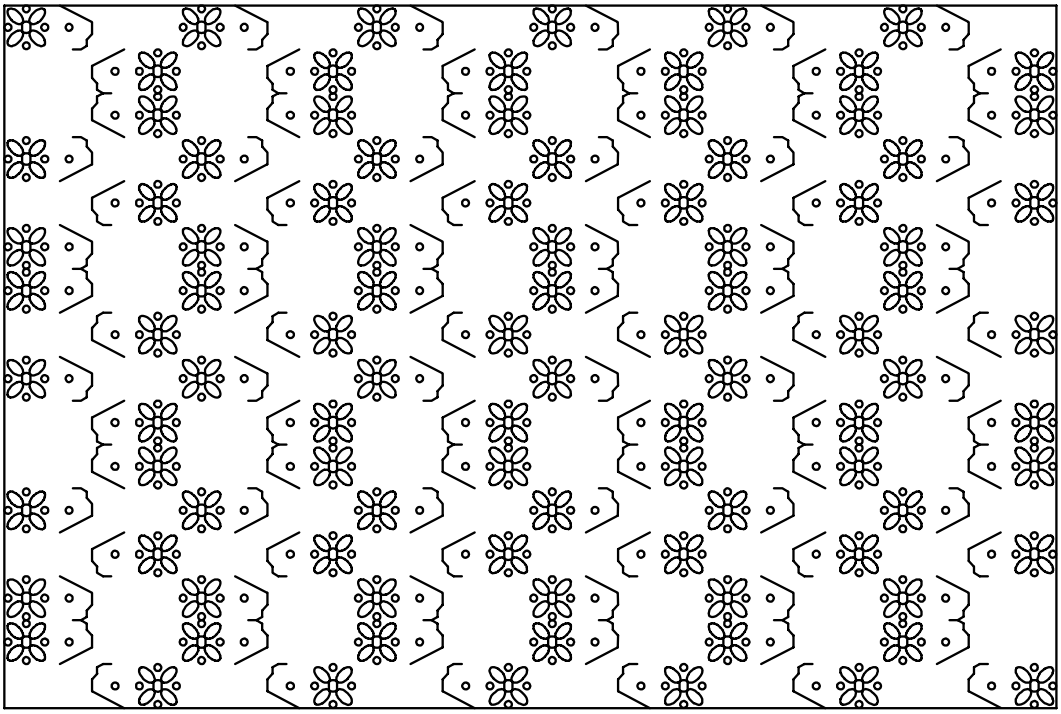
a = 20 Estructura básica rectangular.

c = 3 Simetría deslizada respecto el eje Y.

d = 2 Simetría respecto eje X. (ángulo 180º)

b = 2 Se aplica un giro de 180º.





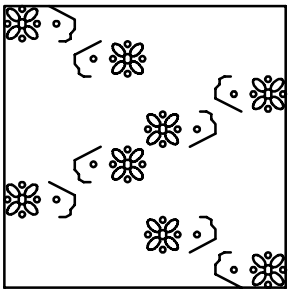
Notación 20. c3 d3 b2

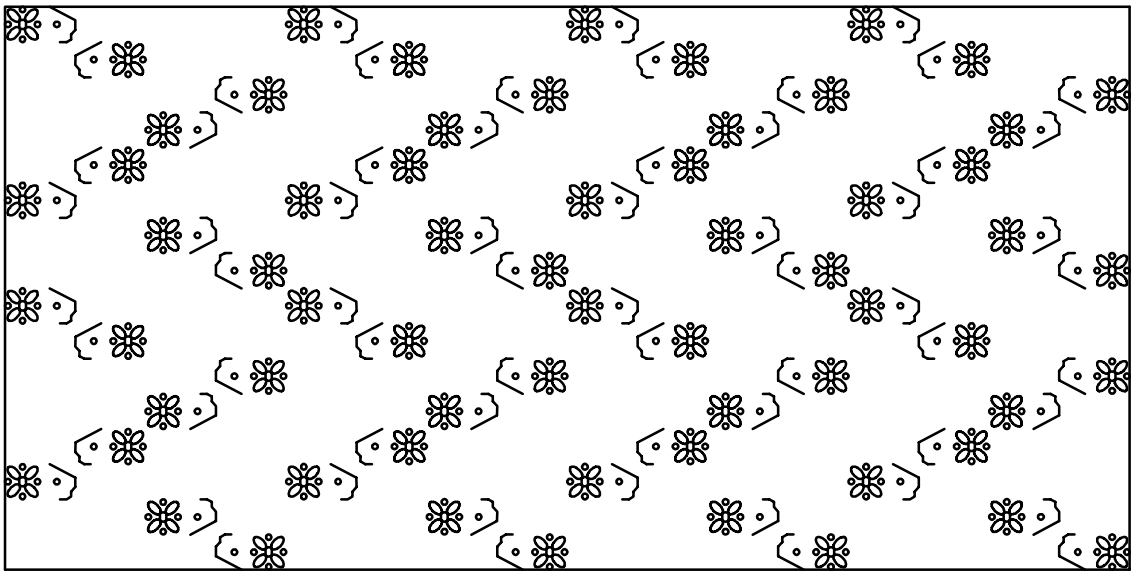
a = 20 Estructura básica rectangular.

c = 3 Simetría deslizada respecto el eje Y.

d = 3 Simetría deslizada respecto eje X (ángulo de 180º).

b = 2 Se aplica un giro de 180º.





5.5.5. Notaciones compuestas

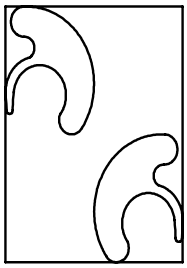
Ejemplo 1:

En este ejemplo vamos a realizar primero un motivo según 20. 1 3

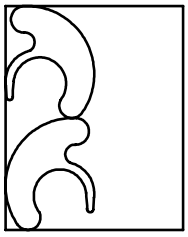
a = 20 Estructura básica rectangular.

b = 1 No existen movimientos giratorios.

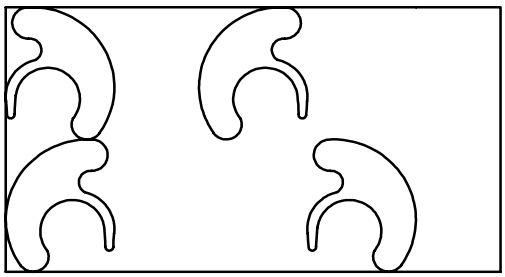
c = 3 Existe una simetría deslizada respecto al eje Y.



Otra posibilidad de simetría deslizada sería un 20. 1 3 especial como variante del anterior, en el que la simetría deslizada respecto del eje Y es especial, ya que la posición final de dicha simetría deslizada se encuentra debajo del propio motivo inicial.



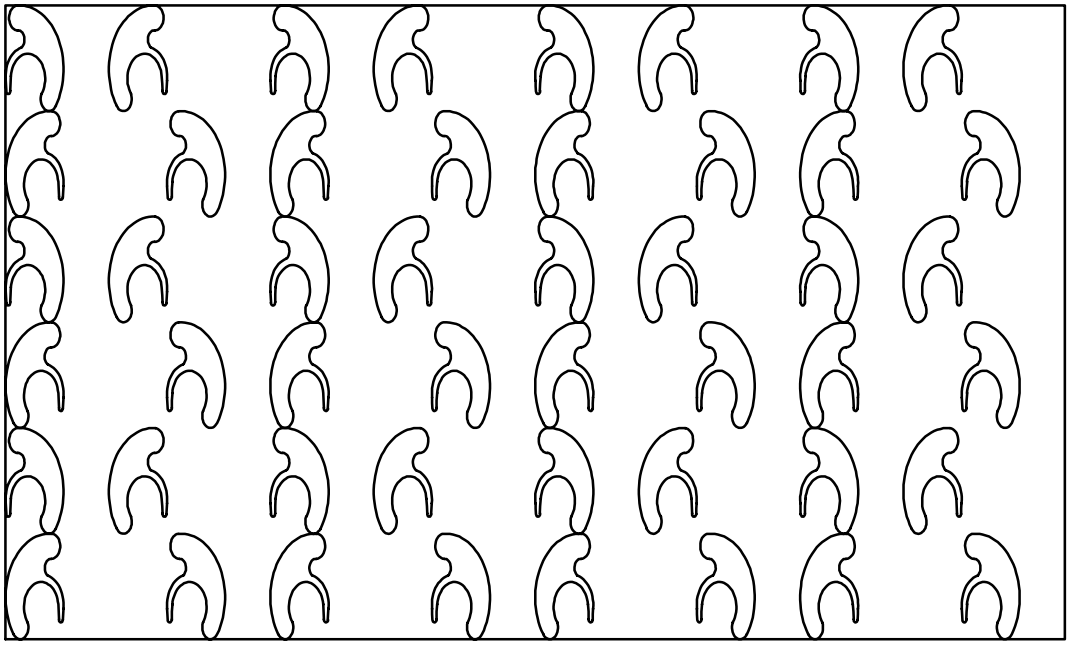
Realizando una composición de ambos y enmarcándolos en una estructura básica rectangular, da como resultado el siguiente motivo:



El diseño obtenido no dispone de una notación específica, ya que la intención de esta representación es la de dar a entender que se pueden combinar los motivos creados individualmente, para obtener de esta manera múltiples familias o combinaciones.

Para este tipo de patrones, el factor más importante es el buen criterio del diseñador al saber elegir los motivos adecuados para elaborar la composición final, ya que individualmente cada uno de ellos puede dar excelentes resultados, pero lo que hay que considerar principalmente es su capacidad de combinación con otros patrones.

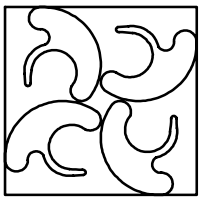
Una simulación general de esta reunión es la que se muestra en la siguiente figura, en donde la composición final distribuye al motivo en diferentes posiciones del plano creando un orden que es susceptible de ser modificado solo con variar las notaciones de los patrones bases utilizados inicialmente.



Ejemplo 2:

En este ejemplo se utilizarán 2 tipos de notaciones, una para el tratamiento del motivo y otra para elaborar el patrón de diseño.

El motivo inicial es un motivo asimétrico g_0, que se somete a 4 movimientos giratorios de 90º cada uno, transformándose en un g_4, que corresponde a una notación 10. 4 :



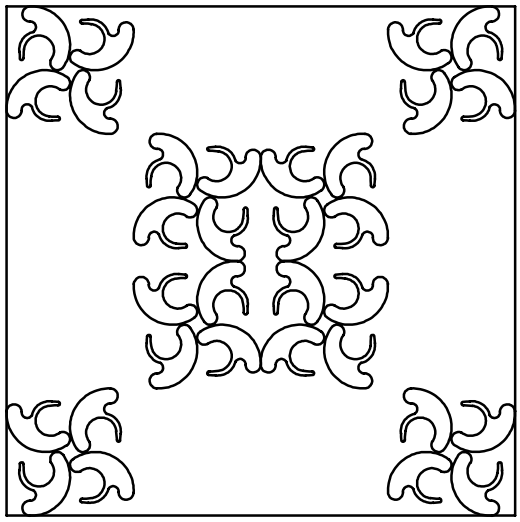
Si a este motivo le aplicamos una notación 10. 2 2 2

a = 10 Estructura básica cuadrangular.

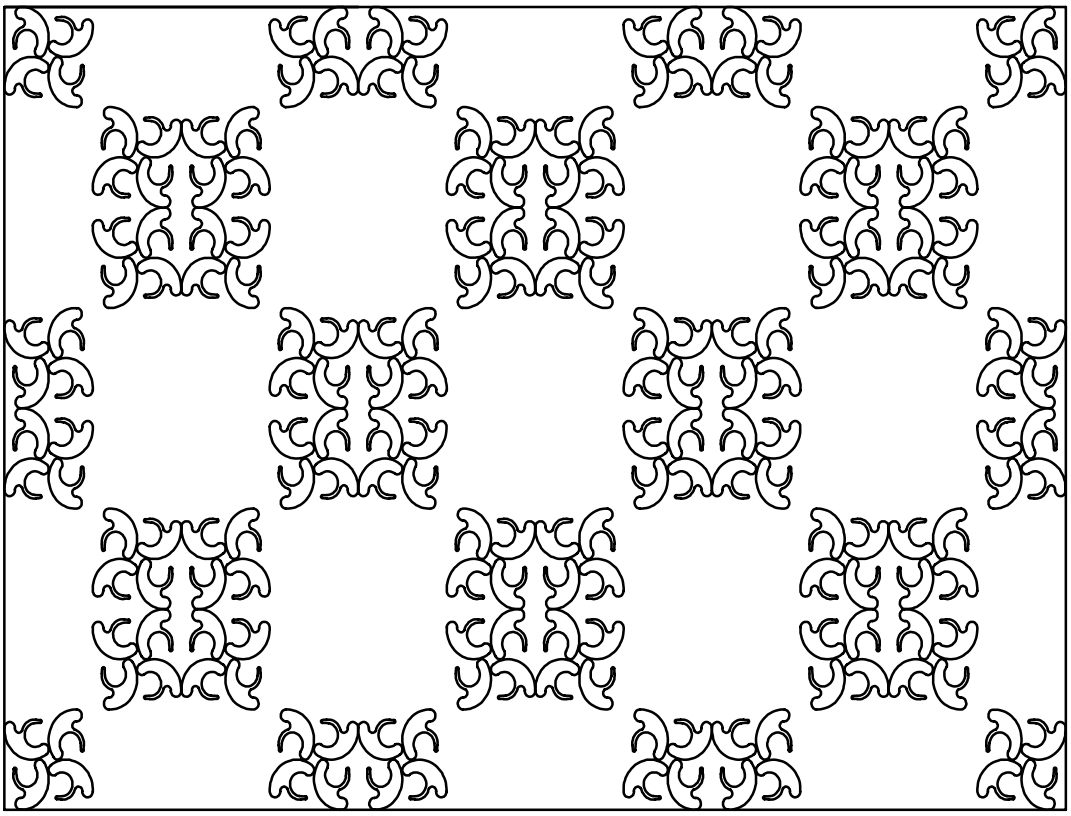
b = 2 Giro de 180º del motivo g_4.

c = 2 Simetría con respecto al eje Y.

d = 2 Simetría con respecto al eje X.



La figura siguiente muestra la simulación total del motivo final generado a través de las dos notaciones conjuntas permitiendo darle mayor complejidad en cuanto a la distribución del motivo base dentro del plano de trabajo.

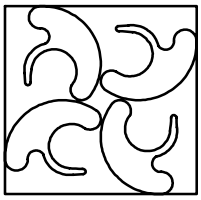


Como se observa, al emplear los recursos dados para el tratamiento del motivo y posteriormente del patrón de diseño, el resultado final varía notablemente con respecto al motivo base. Se debe tener en cuenta que la asimetría del motivo contribuye a obtener nuevas formas finales .

Ejemplo 3:

Igual que en el ejemplo anterior también se utilizarán los 2 tipos de notaciones: una para el motivo y otra para el patrón de diseño:

El motivo inicial será el mismo g_4 anterior



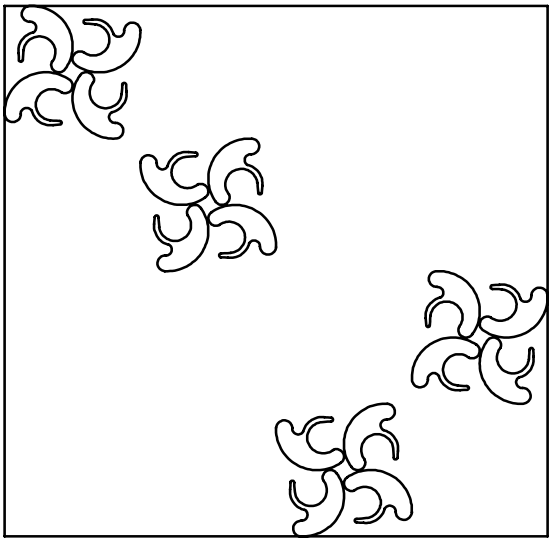
al que se le aplica la notación 10. 1 3 3

a = 10 Estructura básica cuadrangular.

b = 1 No existen giros.

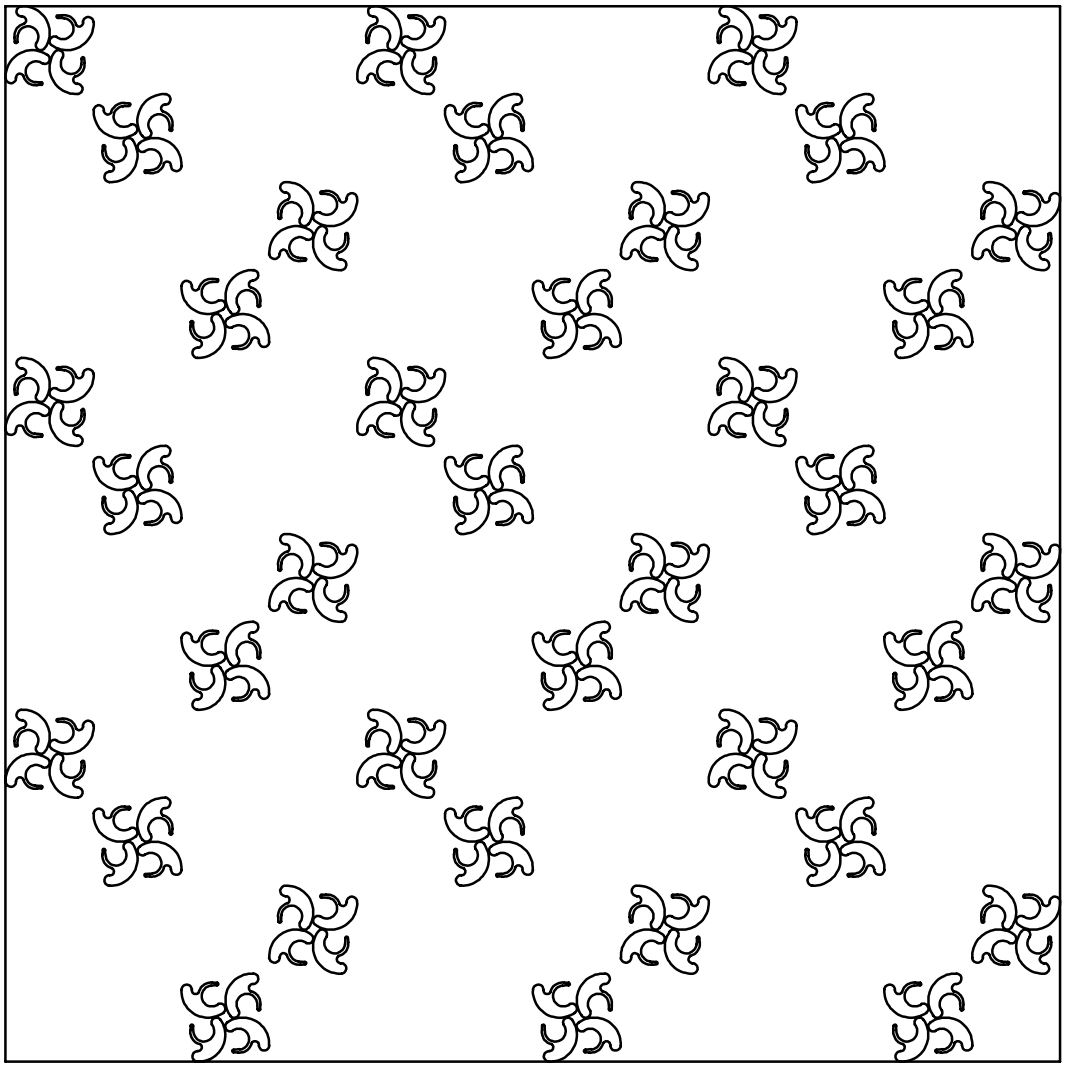
c = 3 Presencia de una simetría deslizada con respecto al eje Y.

d = 3 Presencia de una simetría deslizada con respecto al eje X.



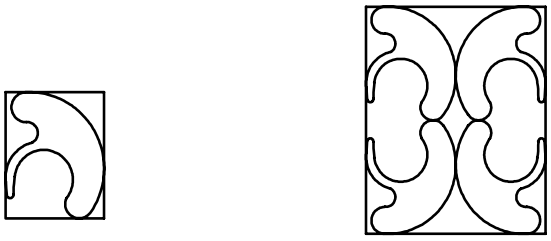
Cuando se requiere efectos de movimiento y a la vez de una distribución del motivo en dos direcciones ortogonales es muy útil emplear este tipo de notación.

La simulación de este patrón de diseño se muestra en la figura siguiente:



Ejemplo 4:

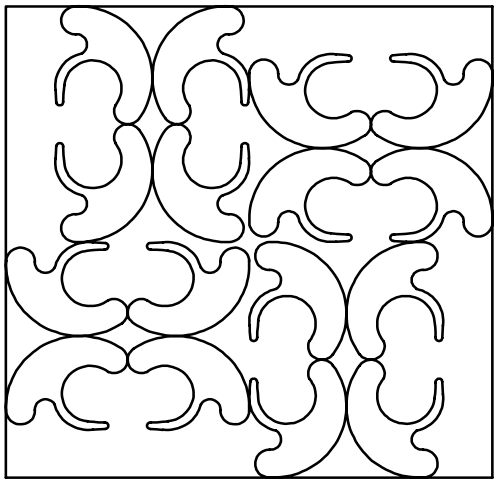
El motivo inicial g_0 se transforma en un motivo base mediante la notación 20. 1 2 2.



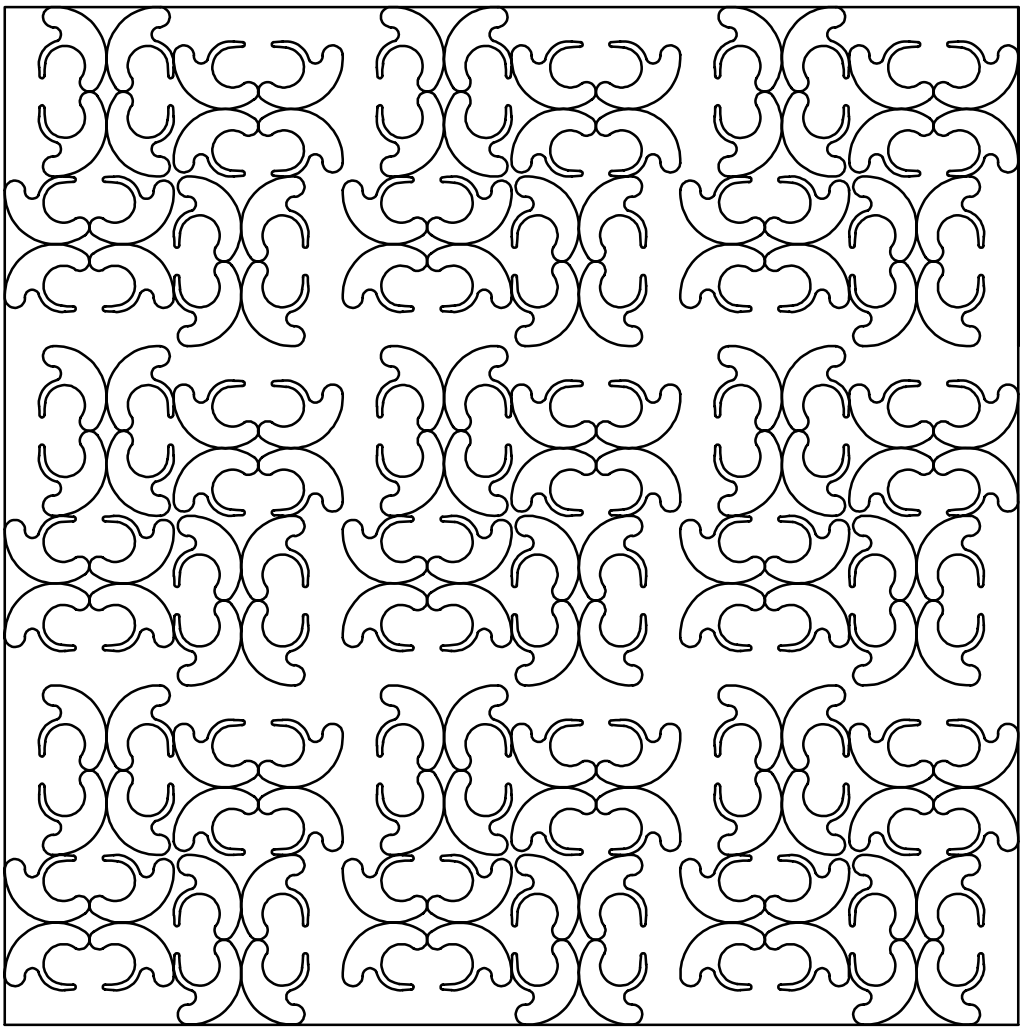
al que se le aplica otra notación 10. 4

a = 10 Estructura básica cuadrangular.

b = 4 Presencia de movimientos giratorios, en donde se realizan cuatro giros de 90º cada uno.



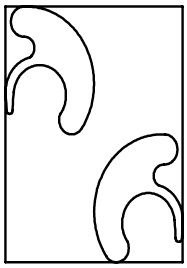
La simulación de esta composición se muestra a continuación:



Ejemplo 5:

En este caso se ha tomado como motivo base el elaborado anteriormente en el ejemplo 1, para posteriormente aplicarle los movimientos que lo convertirán en el patrón de diseño final.

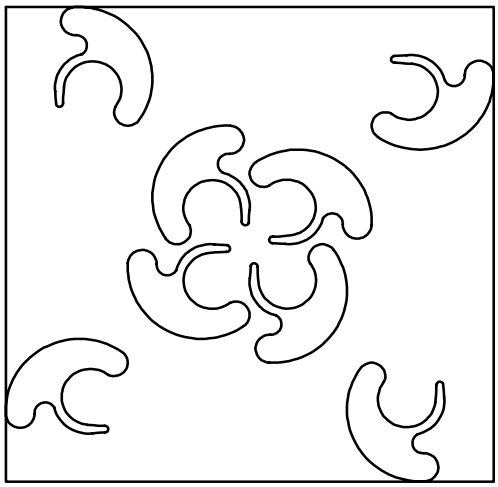
Motivo base según Notación 20. 1 3.



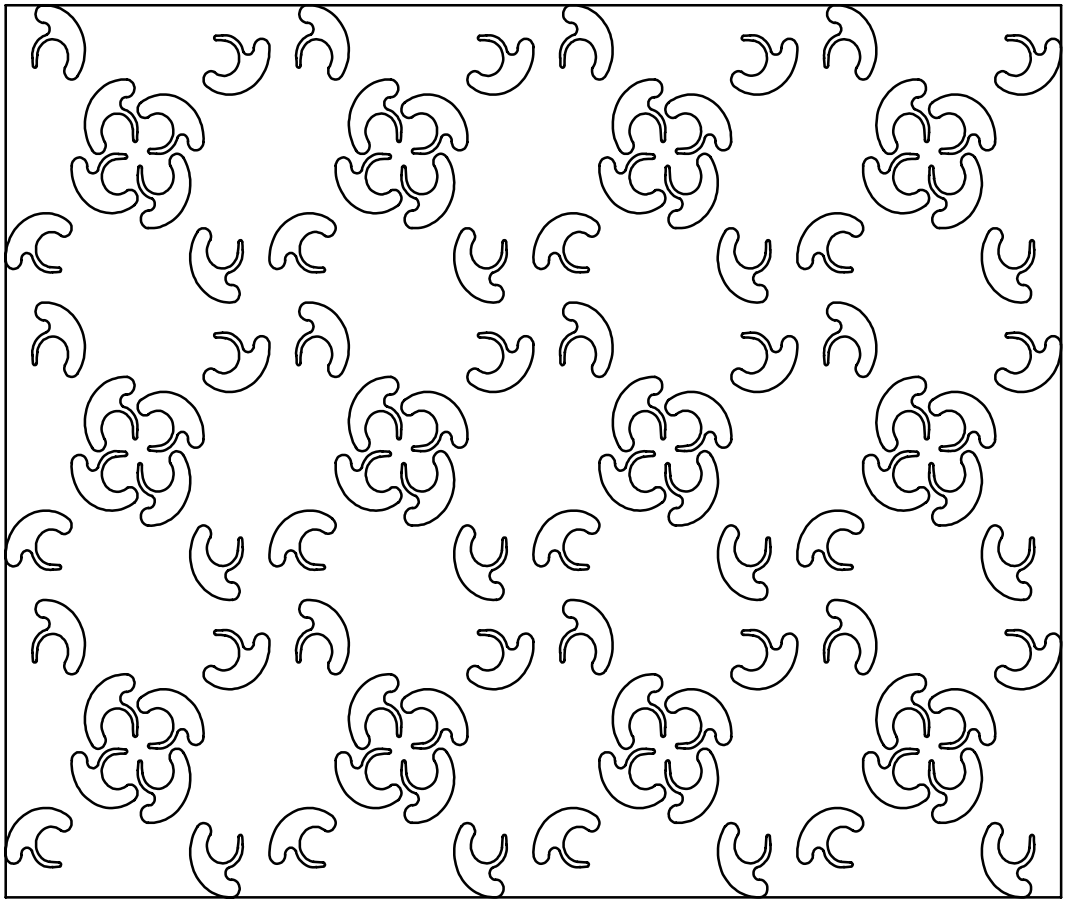
Al que le aplicamos la notación 10. 4

a = 10 Estructura básica cuadrangular.

b = 4 Se realizan cuatro giros de 90º al motivo base.



Una simulación del patrón generado a partir del motivo base indicado se muestra en la siguiente figura:



Con estos ejemplos se pretende demostrar que se puede utilizar como motivo base cualquiera de los patrones obtenidos en otros casos y someterlos a los movimientos conocidos y codificarlos de acuerdo a la notación que se ha desarrollado.

CAPITULO 6

INVESTIGACION Y DESARROLLO DE LA INTERFACE GRAFICA

6.1. INTRODUCCION

Este capítulo se refiere a la propia investigación desarrollada en esta tesis doctoral y cuyo resultado final es un paquete informático de diseño aplicado al textil, habiéndose creado unos procedimientos o rutinas informáticas cuya explicación y listado se detallan más adelante para que las puedan aprovechar quienes las consideren interesantes en el desarrollo de programas informáticos de diseño en cualquier disciplina.

Estas rutinas serán de gran utilidad para las personas no especialistas en programación que se decidan a programar alguna cosa de tipo gráfico utilizando su propio ordenador personal compatibles (PCs), dado que existe muy poca información en el mercado de estas utilidades, tan elementales como necesarias.

Uno de los aspectos que se ha tenido en cuenta, ha sido el de crear un sistema de comunicación amigable entre el usuario y el ordenador, a través de teclado y/o ratón, que incorpora las utilidades más comunes, como por ejemplo el movimiento interactivo de este ultimo.

La estructura modular de estas rutinas permitirá al especialista no informático despreocuparse de estos aspectos tan elementales que no llevan incorporados los lenguajes de programación.

Los sistemas de entrada de datos con desplazamiento del cursor, en los lenguajes de programación son muy elementales y no inmediatos, requiriendo de rutinas no facilitadas por los textos de aprendizaje y de desarrollo del lenguaje en cuestión. Por este motivo incluso un profesor universitario no informático que quiera hacer un pequeño programa para su especialidad se ve coartado de entrada a la hora de introducir los datos con un determinado formato, o al pretender corregir el más pequeño error durante su introducción.

Los programas comerciales de calidad han resuelto de manera más o menos satisfactoria estas rutinas, pero no se facilitan más detalles para su aprovechamiento en otros desarrollos.

Cualquiera que haya hecho sus pinitos en el lenguaje mas extendido como es la programación en BASIC, se habrá dado cuenta de la poca operatividad que ofrecen las instrucciones primarias (Por ejemplo cuando se introduce una serie de datos no es posible rectificar un dato ya introducido).

Las rutinas se han programado en lenguaje C/C++ de Microsoft teniendo la particularidad de ser transportables a otros programas de diseño en forma de objeto dentro de una librería.

El paquete final de diseño textil que se presenta no pretende competir con los programas existentes en el mercado ni con las interfaces altamente sofisticadas, dada nuestra limitación de recursos humanos y técnicos, pero si aportar un granito de arena al personal universitario en sus labores docentes y de investigación, tanto en la vertiente de diseño de tejidos Jacquard como en el aprovechamiento de las rutinas para otro tipo de aplicación.

6.2. PLANIFICACION

6.2.1. Consideración de los conceptos implicados.

Antes de pensar en el desarrollo de cualquier programa informatico, es de vital importancia la búsqueda metódica y organizada de la bibliografía existente acerca de los conceptos implicados y que en el caso que nos ocupa, se pueden resumir en los temas genéricos siguientes:

Teoría de tejidos y de Jacquard
Lenguajes de programación y rutinas estándares.
Bases de datos gráficas
Simulación de procesos
Normativa española y comunitaria sobre los temas implicados.

El conocimiento de dichos conceptos en fuentes de solvencia ahorra muchos esfuerzos y tiempo a la hora de desarrollar el trabajo de forma correcta, sin problemas de estancamiento por falta de información.

Independientemente de la bibliografía existente en el propio departamento, dado que la UPC dispone de un sistema informatizado para búsqueda de información por títulos, materias, autores, etc. se podrán conseguir fotocopias de los artículos deseados o solicitar una consulta o préstamo del libro deseado en cualquier centro de la UPC.

Otra fuente de información posible será la biblioteca del Colegio de Ingenieros Industriales y el servicio de información de que dispone el ICT que está conectada informáticamente con otras bases de datos.

Como fuente mas especializada en la tecnología textil se solicita información propia o

internacional mediante las bases de datos internacionales de información textil.

La bibliografía necesaria referente a la tecnología informática, se consultará específicamente en caso necesario al Departamento de Lenguajes y Sistemas Informáticos de la propia UPC con sede en la Facultad de Informática de Barcelona.

A continuación se exponen algunos de los conceptos importantes de partida, que inicialmente son los que se deberán dominar antes de iniciar el resto del trabajo y sin cuyos conocimientos sería difícil emprender un proyecto con garantías de éxito.

Hilos, pasadas, trama, urdimbre, densidad de trama, densidad de urdimbre, diámetro hilo, título, torsión, color...

Ligamentos: tipos, generación, base de evoluciones, escalonados, motivos, nomenclatura, condiciones y restricciones...

Telares de calada:

- Montura de lizos:
  - lizos
  - remetido
  - picado
- Montura Jacquard:
  - arcadas
  - ordenes de pasada
  - puesta en carta

Tratamiento de color:

- Paletas: por defecto
  - selección
  - edición
  - variación:
    - continua
    - discreta
    - composición
  - color hilo y color pasada

Edición de retículas:

- generar: nueva retícula
  - captura:
    - disco
    - escáner
    - vídeo
- movimientos: traslación
  - giros
  - simetrías: totales
    - parciales
  - escalado
  - ampliación filas y columnas
  - reducción filas y columnas

Estructura de la base de datos:

- matriz base ligamento
- matriz de atributos
- características de las matrices:

flexibilidad
fácil manipulación
intercambiabilidad

Sistemas de almacenamiento:
- gestión de ficheros
- compatibilidad datos diferentes soportes
- adaptación futuras versiones

Sistemas de visualización:
- tarjetas gráficas
- pantallas
- resolución

Periféricos:
- entradas:
  - teclado
  - ratón
  - disco duro
  - disco flexible
  - cd rom
  - escáner
  - vídeo
- salidas:
  - disco duro y disco flexible
  - impresora
  - máquinas de control numérico.

6.2.2. Técnicas a emplear (software)

Para desarrollar el software adecuado que permita conseguir nuestro objetivo se ha decidido emplear el lenguaje de programación C. Dicho lenguaje C se ha convertido en el primer lenguaje para profesionales de programación, tanto en ordenadores personales como en grandes maquinas que utilizan sistemas operativos UNIX.

Una de las razones mas importantes de la elección del C, es su compatibilidad y portabilidad, ya que un mismo programa puede ejecutarse tanto en un PC como en una workstation; esta estandarización no es común en otros lenguajes de programación, como el PASCAL, BASIC, etc. que son más dependientes del entorno y de la maquina (portabilidad).

A estas ventajas de compatibilidad, podemos añadirles otras importantes como son: programación modular, potencia, flexibilidad, estructuración, concisión, etc., pero el C estándar tiene una gran desventaja como es, la compilación y la depuración que otros lenguajes tenían superadas, por lo que se han desarrollado entornos amigables de programación en C que son sistemas integrados de programación, compilación y depuración.

Las primeras rutinas se programaron y compilaron en el entorno de Microsoft Qick C 2.0, pero a medida que fueron aumentando las necesidades de manipulación de datos y de memoria se cambió al entorno de programación Microsoft C/C++ versión 7.0 que ofrece las posibilidades de programación orientada a objetos para aplicaciones Windows y MS-DOS.

El lenguaje C se creó en 1972 por Dennis Ritchie en AT&T Bell Laboratories basado en un lenguaje precursor B de Ken Thomson en 1970 con la intención de recodificar el sistema operativo UNIX para su transportabilidad e independencia de las máquinas.

Estas especificaciones fueron recopiladas y ampliamente aceptadas a través del libro "The C Programming Languaje" de Kernigan y Ritchie, estando considerado como el libro blanco del lenguaje C.

Precisamente esta transportabilidad hace que el lenguaje C sea de tan "bajo nivel" que incluso carece de instrucciones de entrada y salida para manejo de cadenas, debiéndose recurrir a las funciones de biblioteca que facilita el propio fabricante o creándolas expresamente el usuario.

El lenguaje C se puede considerar a la vez como un lenguaje de bajo y alto nivel, pues está lo suficientemente próximo al lenguaje máquina para proporcionar un gran control sobre la implementación de un programa, pero lo suficientemente alejado como para no depender de las particularidades del hardware.

El lenguaje de programación C++ incluye al propio lenguaje C como un subconjunto y fue creado como una ampliación de C pero respetando lo que ya estaba creado. Este lenguaje C++ fue creado por Bjarne Stroustrup también en AT&T Bell Laboratories.

La mayoría de compiladores usan la terminología C/C++ por cuanto se le considera un lenguaje híbrido que permite programar y compilar el C simple o bien usar las posibilidades de programación orientada a objetos (OOP) que incorpora el C++.

El compilador C/C++ de Microsoft dispone de una serie de funciones de biblioteca que no será necesario volver a programar y se hará uso de ellas en los archivos de cabecera siguientes:

```
#include "io.h"          // Manipulación de archivos y E/S de bajo nivel
```

```
#include "conio.h"      // E/S de consola y puerto
#include "graph.h"      // Estructuras de datos gráficos y funciones de primitivas gráficas
#include "malloc.h"     // Asignación dinámica de memoria
#include "stdlib.h"     // Rutinas de biblioteca estándar
#include "string.h"     // Funciones de cadena
#include "time.h"       // Utilidades de fecha y hora

#include "0includ.h"    // Declaración de funciones particulares del programa.
```

El programa informático a desarrollar se estructurará en varios módulos o ficheros que será necesario compilar en conjunto para formar un solo ejecutable, siendo ésta una de las posibilidades que nos ofrece el compilador elegido.

Para las rutinas del manejo y control del ratón se requiere una alta velocidad de computación, de forma que podamos conocer las coordenadas del movimiento del ratón en tiempo real sin cálculos intermedios; esta circunstancia se ha resuelto programando algunas instrucciones en lenguaje ensamblador por ser el lenguaje natural de un ordenador, ya que utiliza directamente las posibilidades de hardware del mismo, este lenguaje ensamblador se ha integrado en las propias rutinas de C/C++ de forma embebido con la instrucción _asm; de esta forma el control del ratón se ejecuta más rápidamente, ya que se actúa directamente sobre la interrupción del ratón asignando los valores al registro determinado.

Esto supone un cierto inconveniente, como es el hecho de que para otras plataformas deberán sustituirse las instrucciones básicas por sus equivalentes. (Esto no sucede con el resto de funciones programadas en lenguaje C/C++ exclusivamente).

6.2.3. Equipo empleado (hardware)

El material que se disponía al iniciar este trabajo fue de un ordenador personal con procesador 386 a 33 Mhz y que se ha ido cambiando en el curso del tiempo de acuerdo con las posibilidades, siendo los requisitos mínimos recomendables del sistema, el disponer de un ordenador tipo 486 con 8 Mb de memoria RAM y monitor VGA de 640 x 480 pixels.

En la actualidad el equipo empleado y sobre el que se ha desarrollado la última versión y que en principio podemos considerar adecuado es el siguiente:

- Ordenador personal compatible PENTIUM con procesador INTEL a 133 Mhz. con 3

- Monitor CRT super VGA de 14" con una resolución máxima de 1024 x 768 pixels de 0,28 mm. y 256 colores.

- Disco duro de 1,6 Gb para almacenamiento masivo de datos

- Disquetera de 3 1/2 ".

- Unidad CD ROM

- Dispositivo señalador y de entrada tipo ratón compatible Microsoft de tres botones con su driver correspondiente.

- Impresora láser como dispositivo de salida de texto o de gráficos en blanco y negro Hewlett Packard LaserJet 5L.
- Impresora de color como dispositivo de salida del tejido simulado del tipo de inyección de tinta marca Hewlett Packard DeskJet 690.

- Sistema operativo MSDOS 6.22 bajo Windows 95.

- Escáner Hewlett Packard HP ScanJet IIcx de millones de colores a 24 bits, y resolución óptica de 400 dpi (puntos por pulgada).

Con el material descrito anteriormente se pueden conseguir los objetivos fijados, a una velocidad informática aceptable, pudiéndose cambiar alguno de ellos en función de las necesidades y de los nuevos productos que salgan al mercado. En un futuro está previsto incorporar las tarjetas controladoras de vídeo con entrada y salida para vídeo VHS y la tarjeta necesaria para gobierno de máquinas de control numérico.

6.2.4. Planificación experimental

Una vez elegido el equipo básico y teniendo en mente las distintas prestaciones que deseamos y que podremos conseguir, lo primero será preparar unas herramientas de software (Interfaces) que permitan un manejo cómodo e interactivo de los comandos de utilización. Estas primeras herramientas a preparar y que no serán las únicas en función de como vaya desarrollándose el trabajo serán como mínimo:

Sistemas de menús

Sistemas de selección

Sistemas de entrada de datos

Sistemas de control:

Creación de drivers e interfaces para ratón, escáner, impresora...

Detección y prevención de errores

Sistemas de ayuda interactiva

Se creará un programa principal del cual dependerán unos módulos secundarios independientes y estructurados de forma que se puedan comunicar con los periféricos tanto de entrada como de salida, a veces directamente y otras mediante programas interface.

Con los conceptos implicados en la tecnología textil debidamente asimilados, se creará una estructura de datos que pueda ser manipulada sin restricciones, interconectada con un programa principal a base de menús de selección, cuyo acceso se producirá mediante la creación de un sistema inteligente basado en la programación orientada a objeto.

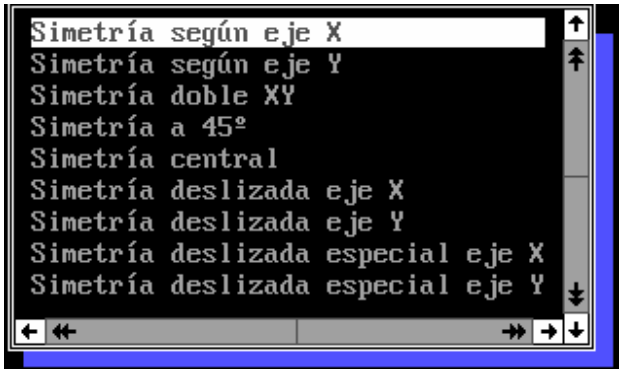
A modo de ejemplo a continuación se muestran alguno de los objetos que se han creado y que aparecen en el transcurso del desarrollo del programa.

En la parte superior de la pantalla se incorporará un menú desplegable de forma

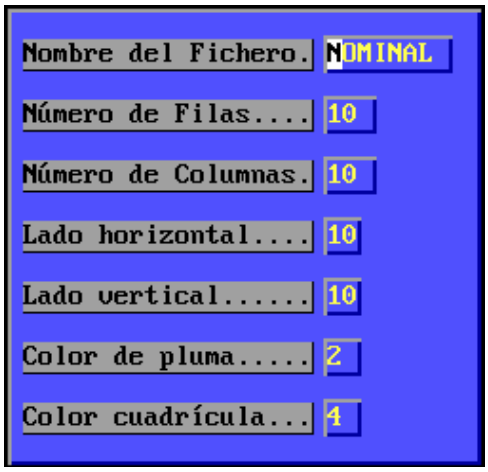
interactiva por teclado o ratón que irá desplegándose en función de la opción por la que nos vayamos moviendo, y hasta que no se pulse el ratón o la tecla no queda seleccionada para ejecutar la acción que corresponda.

Los submenús que incorporen puntos suspensivos no serán de acción inmediata sino que darán lugar a otras opciones, como por ejemplo **Abrir...** un fichero del directorio, que apareciendo en orden alfabético se posibilitará su elección. (Ver figura siguiente)

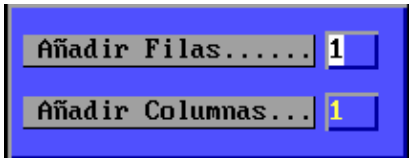
Esta ventana anterior y la siguiente corresponden a la misma rutina a la que se le han cambiado las instrucciones de entrada.



Tal y como se ha mencionado anteriormente, los lenguajes de programación tienen el inconveniente de no incluir un entorno amigable para la entrada de datos, por lo que se deberá crear un objeto que nos posibilite ejecutar esta función, tal como nos muestra la ventana siguiente



Al igual que en el caso anterior este objeto se podrá modificar en función de nuestras necesidades de entrada y salida de datos, pero aprovechando la misma rutina de definición.



6.3. CREACION DE LAS PRINCIPALES RUTINAS PARA LA INTERFACE.

El paquete completo de diseño textil que se presenta, incluye el fichero denominado 0PRINC.CPP que desarrolla el programa principal de ejecución, expresamente preparado para la especialidad de diseño textil y que es susceptible de cambiar los comandos para su uso en otra especialidad de diseño ya que no contiene ninguna función en particular, sino únicamente llamadas a las distintas funciones o rutinas que se han elaborado de forma modular y estructurada.

Todas las rutinas que se han ido creando y que se van a necesitar en el programa de diseño, se han ido definiendo incorporándolas en diversos ficheros en función de su aplicación de afinidad, estando declaradas en un fichero de inclusión denominado 0INCLUD.H indicando a modo de comentario en que módulo o fichero se encuentran definidas para mayor comodidad del programador.

Aparte de estos dos ficheros el programa necesita de los demás ficheros que contienen este tipo de rutinas y que se indican a continuación:

0RATON.CPP
0FICHER.CPP
0RUTINA.CPP
0MENUS.CPP
0MOVIM.CPP
0PALETA.CPP
0TEXTIL.CPP

Cada una de las funciones definidas en cualquier módulo o fichero pueden ser ejecutadas desde cualquier otro modulo sin tener que volverlas a escribir nuevamente.

6.3.1. Declaración de variables y funciones

En este apartado se lista el fichero 0INCLUD.H que incluye la declaración de una estructura para entrada de datos, la asignación de unas variables, y la declaración de todas las funciones que se van a definir en los respectivos módulos

```
struct entradas{int vi,hi,li;char *ni;int ti;char *var;};
#define FILASMAX 300
#define COLUMMAX 300
#define NADA 0
#define ENT 1
#define DER 2
#define IZQ 3
#define ABA 4
#define ARR 5
#define PGU 6
#define PGD 7
#define ESC 8
#define INI 9
#define FIN 10
#define RET 12
#define TAB 13
#define ESP 14
#define INS 15
#define DEL 16
#define SAL 1000
#define TRUE 1
#define FALSE 0
#define SI 1
```

```
#define NO 0
#define C 1
#define ND 2
#define NI 3

// en el fichero 0RATON.CPP se definirán:
void cabecera(void);
void InitMouse(void);
void ShowCursor(void);
void HideCursor(void);
void GetCursor(void);
void XYraton(int *x_dev,int *y_dev);
void Pulsacion(void);
void Raton1o2(void);
void AbsorbePulsacion(void);
void PuntoRaton(void);
void SetCursor(void);
void Get5(void);
void Get6(void);
void Rangohor(void);
void Rangover(void);
void GrafCursor(void);
void FlechaCursor(void);
void TextCursor(void);
void Motion(void);
void GetMotio(void);
void ZonaRaton(void);
int DoblePul(void);
```

```
// en el fichero 0FICHER.CPP se encuentran:
        int nombrevalido(char *nombre);
        int grabareticula(char *nombrefich);
        int cargareticula(char *nombrefich);
        int grabapaleta(char *nombrefich);
        int cargapaleta(char *nombrefich);
        int ficheros(char *ext);
        int cuenta_lineas(FILE *fich);
        void corta(char *tira,int largo);
        void ordenatira(char *tiras[], int num);
        void NOHAYFICHEROS(void);

// en el fichero 0MENUS.CPP:
        void selecmenu(char *amenu[]);
        void LISTABLOQUE1(char *matriz[]);
        void VIDEONORMAL1(void);
        void RESALTA1(void);
        void VACIAZONAS(void);
        void ZONAGLOBAL(void);
        void ZONAMENU(void);
        void ZONARETICULA(void);
        void ZONAPALETA(void);
        void ZONAAJUSTE(void);
        void RESALTAMENU(void);
        int HALLASUBMENU(void);
        void SALIR(void);
```

```
// en el fichero 0RUTINA.CPP:
        void RUTICAMPO (int BOR_RUTI, int NOCONF_RUTI,
                                int EDI_RUTI, int NUM,struct entradas *campo);
        void CONFIRMAR(void);
        void selecmatriz(int *indice,char *matriz[]);
        void DELIMITA(void);
        void LISTABLOQUE(char *matriz[]);
        void VIDEONORMAL(void);
        void RESALTA(void);
        int MULTIPLENTRADA(void);
        void Selrectan(void);
        void ORDVERT(int x1,int y1,int x2,int y2);
        void AGET(int x1,int y1,int x2,int y2);
        void APUT(int x1,int y1,int x2,int y2);
        void APUTR(int x1,int y1,int x2,int y2);
        void APUTXOR(int x1,int y1,int x2,int y2);
        void Flecha_arriba(int a,int b);
        void Flecha_abajo(int a,int b);
        void Flecha_izquierda(int a,int b);
        void Flecha_derecha(int a,int b);
        void RESALTAC(int v,int h,char c,int color,int fondo);
        void RESALTAF(int v,int h,char *f,int color,int fondo);
        void BOTON(int x1,int y1,int x2,int y2);
        char *AlineaIzq(char *f);
        char *AlineaDer(char *f);
        char *Abreesp(char *f,int POSI);
        char *Cierraesp(char *f,int POSI);
```

```
// En el fichero 0PALETA.CPP
        void paleta(float a,float b,float lhp,float lvp,int nfp,int ncp,short cp);
        void calculacolor(void);
        void cajaajustes(void);
        void ajusterojo(void);
        void ajusteverde(void);
        void ajusteazul(void);

// En el fichero 0MOVIM.CPP
        void TRASLADAR(void);
        void SIMETRIA(void);
        void GIROS(void);
        void COPIAR(void);
        void RELLENAR(void);
        void IGUALAMATSEG(void);
        void RECUPERAMAT(void);
        void COGEZONA(void);

// en el fichero 0TEXTIL.CPP
        void reticular(float a,float b);
        void rectangulor(float izqx,float izqy,float lh,float lv,short c,short f );
        void rellenareti(int x1,int y1,int x2,int y2,float lh,float lv,int nf,int nc);
        void retipunto(float a,float b,float lh,float lv,int nf,int nc);
        void BORRARETICULA(void);
        void PINTAR(int a);
        void retiparcial(float a,float b,int filai,int columi,int filaf,int columf);
        void SUSTITUYECOLOR(void);
        void MEZCLACOLOR(void);
```

6.3.2. Funciones principales

En este apartado se van a describir las principales rutinas que se han creado y que se fueron probando en pequeños programas creados a medida que se iban formando para depurar posibles errores intentando comprobar todas las variantes y posibilidades.

La primera cuestión que se abordó fue la de crear unas rutinas para el manejo del ratón que el lenguaje de programación no lleva incorporado. Estas rutinas se han desarrollado en Asembler para una mayor velocidad de acción entre el periférico (ratón) y la unidad central.

De entre las funciones previstas de ratón más o menos necesarias quisiéramos destacar la función ZonaRaton() para detectar en que zona de la pantalla se mueve el ratón convirtiendo este movimiento en una espera inteligente para la ejecución de un comando o de otro, que por supuesto tiene que haberse previsto con antelación, delimitando correctamente las zonas posibles de maniobra.

```
//************************************
void InitMouse(void)  //Inicializa el ratón si se ha leído el driver
{
   // Si no se ha cargado el driver devuelve ra_ton=0
   // Si se ha cargado el driver devuelve ra_ton = -1
   _asm
        {
        Mov  AX,0            ;Pone 0 en el registro AX
        Int  51              ;Ejecuta la función 0 de la interrupción 51
        Mov  ra_ton,AX       ;Pone en la variable ra_ton el valor del registro AX
        Mov  cboton,BX       ;Numero de botones reconocidos
        }
```

```
   if (ra_ton != -1)
          {      _setvideomode(_DEFAULTMODE);
                 printf("\n OJO! Cargar driver del ratón y volver a ejecutar");getch();
                 exit (1);
          }
}

//************************************
void ShowCursor(void)          //Mostrar Cursor del ratón
{  _asm
    {
        Mov  AX,1
        Int 51
    }
}

//************************************
void HideCursor(void)  //Esconder Cursor del rato
{       _asm
        {
        Mov  AX,2
        Int 51
        }
}




//************************************
void GetCursor(void)   //Obtener coordenadas y botón pulsado del ratón
```

```
{
int x;
float a;
   _asm
        {
        Mov AX,3
        Int 51
        Mov x,CX
        Mov y_rat,DX
        Mov b_rat,BX
        }
   a=float (cv.numxpixels/640.0);
   x_rat=x*int(a);
}

//************************************
void XYraton(int *x_dev,int *y_dev)  //Devolver coordenadas y botón pulsado
{   _asm
        {
        Mov AX,3
        Int 51
        Mov x_rat,CX
        Mov y_rat,DX
        Mov b_rat,BX
        }
   *x_dev = x_rat;
   *y_dev = y_rat;
}
```

```
//************************************
void Pulsacion(void)   //esperar pulsación ratón
{   b_rat=0;
   while (!b_rat)
         {  _asm
            {   Mov AX,3
                Int 51
                Mov b_rat,BX
            }
         }
}

//************************************
void AbsorbePulsacion(void)   //espera hasta haber soltado el botón
{
    int a;
    do
         {  _asm
            {   Mov AX,3
                Int 51
                Mov b_rat,BX
            }
           if (b_rat>0) a=b_rat;
         } while (b_rat>0);
   b_rat=a;
}


//************************************
```

```
void Raton1o2(void)    //obliga a pulsar botón 1 o 2
{
    b_rat=0;
    do
        {   _asm
            {   Mov AX,3
                Int 51
                Mov b_rat,BX
            }
        } while (b_rat < 1 || b_rat > 2 );
     AbsorbePulsacion();
}

//************************************
void PuntoRaton(void)          //tomar coordenadas tras pulsar y soltar
{
   Pulsacion();
   GetCursor();
   AbsorbePulsacion();
}

//************************************
void SetCursor(void)  //Posicionar el ratón
{
   float a;
   int x;
   a=((float)cv.numxpixels/640);
```

```
   x=(int)(x_rat/a);
   _asm
        {
        Mov AX,4
        Mov CX,x
        Mov DX,y_rat
        Int 51
        }
}

//************************************
void Get5(void)       //obtención datos botón en pulsación
{   _asm
        {
        Mov AX,5              ;          / 0 botón izq.
        Mov BX,bot_rat                ;bot_rat= - 1 botón der.
        Int 51                ;          \ 2 botón centro
        Mov estado_boton,AX           ;estado (0,1,2,3,4)
        Mov num_botpulsa,BX           ;nº de botones pulsados
        Mov ult_xratpul,CX    ;última coord x pulsada
        Mov ult_yratpul,DX    ;última coord y pulsada
        }
}

//************************************
void Get6(void)  //Obtención datos botón suelto
{   _asm
        {
```

```
        Mov AX,6                ;           / 0 botón izq.
        Mov BX,bot_rat          ;bot_rat= - 1 botón der.
        Int 51                  ;     \ 2 botón centro
        Mov estado_boton,AX   ;estado (0,1,2,3,4)
        Mov num_botpulsa,BX   ;nº de botones sueltos
        Mov ult_xratpul,CX      ;última coord x soltada
        Mov ult_yratpul,DX      ;última coord y soltada
        }
}

//************************************
void Rangohor(void)  //Limitar recorrido horizontal del ratón
{
   float a;
   int x;
   a=((float)cv.numxpixels/640);
   x=(int)(xmax_rat/a);
   if (xmax_rat > cv.numxpixels)
     {x=(int)(cv.numxpixels/a);
        xmax_rat=(int)(cv.numxpixels);}
   _asm
        {
        Mov AX,7
        Mov CX,xmin_rat
        Mov DX,x
        Int 51
        }
}
//************************************
```

```
void Rangover(void)   //Limitar recorrido vertical
{
   if (ymax_rat > cv.numypixels) ymax_rat=cv.numypixels;
   _asm
         {
         Mov AX,8
         Mov CX,ymin_rat
         Mov DX,ymax_rat
         Int 51
         }
}

//************************************
void GrafCursor(void)          //Cursor tipo gráfico
{
   int cursor[16][2]={
         65535,65535,65535,65535,65535,65535,65535,65535,
         65535,65535,65535,65535,65535,65535,65535,65535,
         128,128,128,128,128,128,128,128,
         65535,128,128,128,128,128,128,128
         };                    // matriz diseño cursor gráfico
   int *pa =&cursor[0][0];    // puntero almacena dirección memoria del
                              // primer elemento de la matriz
   h_xrat=8; h_yrat=8;
   _asm
         {
         Mov AX,9
         Mov BX,h_xrat
         Mov CX,h_yrat
```

```
        Mov DX,WORD PTR pa
        Int 51
        }
}

//************************************
void FlechaCursor(void) //Cursor gráfico tipo flecha
{
    int flecha[16][2]={
        65535,65535,65535,65535,65535,65535,65535,65535,
        65535,65535,65535,65535,65535,65535,65535,65535,
        32768,24576,30720,15872,16256,8160,8184,4088,
        4080,2016,2032,952,796,14,7
        };
   int *pb =&flecha[0][0];
   h_xrat=0; h_yrat=0;
   _asm
        {
        Mov AX,9
        Mov BX,h_xrat
        Mov CX,h_yrat
        Mov DX,WORD PTR pb
        Int 51
        }
}

//************************************
void TextCursor(void)  //Cursor de texto
{   _asm
```

```
        {
        Mov  AX,10
        Mov  BX,text_rat
        Mov  CX,1
        Mov  DX,5
        Int  51
        }
}

//************************************
void GetMotio(void)   //Obtención sentido movimiento ratón
{   _asm
        {
        Mov  AX,11
        Int  51
        Mov  der_rat,CX
        Mov  aba_rat,DX
        }
}

//************************************
void Motion(void)   //sensibilidad RATÓN 0-255.....0 muy sensible
{    _asm
        {
        Mov  AX,15
        Mov  CX,sen_xrat
        Mov  DX,sen_yrat
        Int  51
        }
```

```
}

//************************************
int DoblePul(void)    //Detecta pulsación simple o doble del ratón
{
   time_t seg,tanteri;
   PuntoRaton();tanteri=time(&seg);PuntoRaton();
   if (time(&seg)-tanteri<1) return(1);
   else return(0);
}

//************************************
void ZonaRaton(void) //Detecta zonas movimiento del ratón
{   int n;
   numero_zona=0;
   for(n=1;n<=cantidad_zonas;n++)
         {
         xy1=_getphyscoord(short(dzona[n][0]),short(dzona[n][1]));
         xy2=_getphyscoord(short(dzona[n][2]),short(dzona[n][3]));
         ORDVERT(xy1.xcoord,xy1.ycoord,xy2.xcoord,xy2.ycoord);
         if ((x_rat<=xder)&&(x_rat>=xizq) && (y_rat<=yinf)&&(y_rat>=ysup))
            {numero_zona=n;break;}
         }
}
```

```
//************************************
```

Las funciones que se detallan a continuación son precisamente para controlar la zona de pantalla en que nos movemos con el ratón y decidir en cada momento que acción se producirá en función del botón que pulsemos o la tecla que presionemos.

```
void VACIAZONAS(void)       //Vacía todas las zonas
{  int i;
   for (i=0;i<100;i++)
    {      dzona[i][0]=0;
           dzona[i][1]=0;
           dzona[i][2]=0;
           dzona[i][3]=0;
     }
}

//************************************
void ZONAGLOBAL(void)   // Zona 0 por defecto
{          dzona[0][0]=0;
           dzona[0][1]=0;
           dzona[0][2]=639;
           dzona[0][3]=479;
}

//************************************
void ZONAMENU(void)       // Zona 1 menú principal
{          dzona[1][0]=0;
           dzona[1][1]=0;
           dzona[1][2]=640;
           dzona[1][3] = AC;
```

```
}

//************************************
void ZONARETICULA(void) // Zona 2 retícula
{       dzona[2][0]=izx;
        dzona[2][1]=izy;
        dzona[2][2]=izx+nc*lh;
        dzona[2][3]=izy+nf*lv;
}

//************************************
void ZONAPALETA(void)   // Zona 3 Paleta
{       dzona[3][0]=615;
        dzona[3][1]=20;
        dzona[3][2]=615+20;
        dzona[3][3]=20+16*20;
}

//************************************
void ZONAAJUSTE(void)   // Zona 4 ajuste color
{       dzona[4][0]=560;
        dzona[4][1]=345;
        dzona[4][2]=635;
        dzona[4][3]=479;
}
```

```
//**********************************
```

La función Selrectan() siguiente, permite seleccionar una zona de pantalla que se usa para delimitar datos que luego podrán ser copiados, trasladados, borrados, etc,. Estos datos pueden ser de la base de datos gráfica (retícula), de puntos de pantalla o simplemente límites para otras funciones. Lleva asociada otra función que ordena los vértices para que puedan ser utilizados los datos siempre en el mismo orden

```
void Selrectan(void)  //incluir la función ORDVERT() a continuación.
{
  b_rat=0;
  xmovila=avent;
  ymovila=bvent;
  do
        {
          if ((x_rat == xmovila) && (y_rat == ymovila)) {}
          else if ((x_rat == avent) && (y_rat == bvent)) {}
          else if (y_rat==bvent)
                {       HideCursor();
                        AGET(avent,bvent,x_rat,bvent);
                        APUTR(avent,bvent,x_rat,bvent);ShowCursor();}
          else if (x_rat==avent)
                {       HideCursor();
                        AGET(avent,bvent,avent,y_rat);
                        APUTR(avent,bvent,avent,y_rat);ShowCursor();}
          else
                {       HideCursor();
                        AGET(avent,bvent,x_rat,bvent);
                        lineabuff1=lineabuff;
                        AGET(x_rat,bvent,x_rat,y_rat);l
```

```
                ineabuff2=lineabuff;
                AGET(x_rat,y_rat,avent,y_rat);
                lineabuff3=lineabuff;
                AGET(avent,y_rat,avent,bvent);
                lineabuff4=lineabuff;
                lineabuff=lineabuff1;
                APUTR(avent,bvent,x_rat,bvent);
                lineabuff=lineabuff2;
                APUTR(x_rat,bvent,x_rat,y_rat);
                lineabuff=lineabuff3;
                APUTR(x_rat,y_rat,avent,y_rat);
                lineabuff=lineabuff4;
                APUTR(avent,y_rat,avent,bvent);
           ShowCursor();
        }
xmovila=x_rat;ymovila=y_rat;
GetCursor();
if ((x_rat == xmovila) && (y_rat == ymovila)) {}
else if ((xmovila == avent) && (ymovila == bvent)) {}
else if (ymovila==bvent)
        {       HideCursor();
                APUT(avent,bvent,xmovila,bvent);
                ShowCursor();_ffree(lineabuff);}
else if (xmovila==avent)
        {       HideCursor();
                APUT(avent,bvent,avent,ymovila);
                ShowCursor();_ffree(lineabuff);}
else
        {       HideCursor();
```

```
                    lineabuff=lineabuff1;
                    APUT(avent,bvent,xmovila,bvent);
                    lineabuff=lineabuff2;
                    APUT(xmovila,bvent,xmovila,ymovila);
                    lineabuff=lineabuff3;
                    APUT(xmovila,ymovila,avent,ymovila);
                    lineabuff=lineabuff4;
                    APUT(avent,ymovila,avent,bvent);ShowCursor();
                    _ffree(lineabuff);_ffree(lineabuff1);_ffree(lineabuff2);
                    _ffree(lineabuff3);_ffree(lineabuff4);
                }
        } while (!b_rat);
    AbsorbePulsacion();
    HideCursor();
    lineabuff=lineabuff1;APUT(avent,bvent,xmovila,bvent);
    lineabuff=lineabuff2;APUT(xmovila,bvent,xmovila,ymovila);
    lineabuff=lineabuff3;APUT(xmovila,ymovila,avent,ymovila);
    lineabuff=lineabuff4;APUT(avent,ymovila,avent,bvent);
    ShowCursor();
    _ffree(lineabuff);_ffree(lineabuff1);_ffree(lineabuff2);
    _ffree(lineabuff3);_ffree(lineabuff4);
}
```

```
//************************************
```

Esta función sirve para identificar una zona por sus filas y columnas dentro de una retícula textil, haciendo uso previo de la función anterior selrectan().

```
void COGEZONA(void)
{       _settextposition(30,1);
        printf("Elegir extremo superior izquierdo Zona");
        PuntoRaton();
        _settextposition(30,1);
        printf("                                ");
        //calculo el vértice superior izquierdo del cuadro elegido
        x_rat=izx+(int((x_rat-izx)/lh))*lh;
        y_rat=izy+(int((y_rat-izy)/lv))*lv;
        avent=x_rat;bvent=y_rat;
        _settextposition(30,1);
        printf("Completar con el extremo inferior derecho ");
        Selrectan();
        _settextposition(30,1);
        printf("                                ");
        x_rat=izx+(int((x_rat-izx)/lh))*lh+lh;
        y_rat=izy+(int((y_rat-izy)/lv))*lv+lv;
        cvent=x_rat;dvent=y_rat;
        ORDVERT(avent,bvent,cvent,dvent);
        avent=xizq;bvent=ysup;cvent=xder;dvent=yinf;
        fili=int((ysup-izy)/lv)+1;
        coli=int((xizq-izx)/lh)+1;
        filf=int((yinf-izy)/lv);
        colf=int((xder-izx)/lh);
}
```

```
//************************************
```

Las funciones que se muestran a continuación: grabareticula(), cargareticula(), grabapaleta(), cargapaleta(), ficheros(), acceden directamente al disco duro para explorar la base de datos gráfica que se ha creado *.TEX o *.PAL para leer o escribir en la misma.

```
int grabareticula(char *nombrefich)
{   int i,j;
    char buffer[60];
    FILE *pf;
    pf=fopen(nombrefich, "w+");
    _itoa(nf, buffer, 10 );fputs(buffer,pf);fputc('\n',pf);
    _itoa(nc, buffer, 10 );fputs(buffer,pf);fputc('\n',pf);
    for(i=1;i<=nf;i++)
     {     for(j=1;j<=nc;j++)
            {  mat2[i][j]=seg2[i][j];
               _itoa(mat2[i][j],buffer,10);fputs(buffer,pf);fputc('\n',pf);
            }
     }     fclose(pf);return(SI);
}

//************************************
int cargareticula(char *nombrefich)
{ FILE *pf;
  char buffer[5];
  int i,j;
  if((pf=fopen(nombrefich, "r")) == NULL)
     { printf("\a");_settextposition(28,1);
       perror("Error al abrir fichero    Pulse una tecla");getch();
```

```
      _settextposition(28,1);
      printf("                                ");return(NO);
     }
  fgets(buffer, 5, pf);nf=short(atoi(buffer));     //---numero de filas
  fgets(buffer, 5, pf);nc=short(atoi(buffer));     //---numero de columnas
  mat2[1][0]=nf;mat2[0][1]=nc;
  seg2[1][0]=nf;seg2[0][1]=nc;
  for(i=1;i<=nf;i++)                               //---color fila/columna
  { for(j=1;j<=nc;j++)
    {     if(!ferror(pf) && !feof(pf))
           {  fgets(buffer, 5, pf);
              mat2[i][j]=short(atoi(buffer));
              seg2[i][j]=short(atoi(buffer));
           }
    }
  }
  fclose(pf);
  if (ferror(pf))
    { printf("\a");_settextposition(28,1);
      perror("Error durante la lectura  Pulse una tecla");getch();return(NO);
      _settextposition(28,1);
      printf("                                ");return(NO);
    }
   else     {return(SI);}
}
```

```
//************************************
int grabapaleta(char *nombrefich)
{   int i;
    char buffer[60];
    FILE *pf;
    pf=fopen(nombrefich, "w+");
    for(i=0;i<=15;i++)
     {
          _ltoa(color[i],buffer,10);fputs(buffer,pf);fputc('\n',pf);
     }
     fclose(pf);
     return(SI);
}

//************************************
```

Tal como se ha dicho anteriormente, se puede comprobar que se utiliza la misma función de selecmatriz() para crear la ventana de dialogo; en este caso su función es elegir distintos tipos de paleta almacenadas en el disco duro con la función cargapaleta().


```
int cargapaleta(char *nombrefich)
{ FILE *pf;
  char buffer[10];
  int i;
```

```
 if((pf=fopen(nombrefich, "r")) == NULL)
   {printf("\a");_settextposition(28,1);
    perror("Error al abrir fichero    pulse una tecla");getch();
    _settextposition(28,1);
    printf("                                  ");return(NO);}
    for(i=0;i<=15;i++)                                    //---color fila/columna
         {
           {    if(!ferror(pf) && !feof(pf))
                 {  fgets(buffer, 10, pf);
                    color[i]=atol(buffer);
                 }
           }
         }
 fclose(pf);
 if (ferror(pf))
   {    printf("\a");_settextposition(28,1);
        perror("Error durante la lectura...Pulse una tecla");getch();return(NO);
        _settextposition(28,1);
        printf("                                  ");return(NO);
   }
  else
   {return(SI);}
}
```

//**********

Para elegir algún fichero o retícula gráfica, en este caso se utiliza el mismo procedimiento de selecmatriz() con la nueva función ficheros:


```
int ficheros(char *ext) // almacena ficheros directorio en una matriz
{
   #define N 80          // longitud línea 80
   #define M 100         // máximo 100 ficheros
   FILE *in;             // declara punteros FILE
   static char cadena[N];
   static char matri[M][N];
   extern char *dir[M];
   static char temporal[N];
   static char ext_interno[N];
   char *compara[15];
   int i,lineastot;
   int lineas=1;
   int ct = 0;
   int distintaext;
   int nombrevalido;
   static char falsonom[15];
   FILE *nulo;           // declara punteros FILE
   static char fichnulo[15];
   int final=0;
```

```
        // se crea un fichero como minimo para el dir
        strcpy(fichnulo,"ZZZZZZZZ.");strcat(fichnulo,ext);
        nulo = fopen(fichnulo, "w"); fclose(nulo);

strcpy(temporal,"dir *.");
strcat(temporal,ext);
strcat(temporal," >DATOS.TMP");
system(temporal);    remove(fichnulo);
strcpy(ext_interno," ");
strcat(ext_interno,ext);   //a¤adir espacio a la extension

if((in = fopen("DATOS.TMP", "r")) == NULL)    // abre fichero lectura
          {    fprintf(stderr, "Error de fichero directorio"); exit(2); }
lineastot= (cuenta_lineas(in));
for(i=0;i<lineastot;i++)
      {   fgets(cadena,N,in); // lee fichero por cadenas
         while (strlen(cadena)>14) // eliminar lineas cortas
               {  corta(cadena,12);
                  sprintf(&matri[i][N],cadena); //almacena cadenas en matriz
                  lineas++;
               }
         if(lineas>100)
               { _settextposition(30,1);
                  printf(" MAS DE 100 ficheros... Eliminar algunos");
                        break;
               }
      }
if (fclose(in) != 0 ) fprintf(stderr, "Error al cerrar ficheros\n");
strcpy(falsonom,"ZZZZZZZZ ");strcat(falsonom,ext);
```

```
   while(ct<lineas)
    {   compara[15]=matri[ct]+8;
        distintaext =strncmp(compara[15],ext_interno,4);
        compara[15]=matri[ct];
        nombrevalido =strcmp(compara[15],falsonom);
        if (distintaext)
           {dir[final]="VACIO";}
        else
           { final++;
                dir[final]=matri[ct];
                if (!nombrevalido)
                        {final--;}
           }
        ct++;
    }

   ordenatira(dir,final);          // clasificador de tiras
   return(final);
}
```

```
//************************************
```

Con la siguiente función de selecmenu() se presenta en pantalla una línea superior con el menú principal, desplegable a su vez en submenús, que ofrecen opciones de acción directa o indirecta; el ejemplo muestra el submenú que se descuelga de la opción **ARCHIVO** del menú principal.


```
void selecmenu(char *amenu[]) // devuelve submenu y opción
{
   char cadena[150], cadenatemp[150];
   char *titulos[15], *colgantes[15];
   char far *buffmenu;
   char a;
   int i=0,k=0,ncar=0;
   int posicion[15];
   int longopcion=0,longtemp=0;
   int j,xg1,yg1,xg2,yg2,largo,TECLA;
   cantsubmenus=0; //externa
//  Inicialización de variables
   for (i=0;i<100;i++)
     {    dzona[i][0] = 0; dzona[i][1] = 0;dzona[i][2] = 0;dzona[i][3] = 0;}
   AGET(0,0,639,16);buffmenu=lineabuff; // almaceno pantalla menu
   strcpy(cadenatemp,amenu[0]);
   a='x';
```

```
   while (a != '\0')       //Posicion de inicio de cada submenu
    {    a=cadenatemp[ncar++];
         if (a == ' ') posicion[++k]=ncar+1;
    }
   i=1;titulos[i]= strtok(cadenatemp," ");
   while(titulos[i]!=NULL){titulos[++i]= strtok(NULL," ");}
   cantsubmenus=i-1;

//  Las zonas 21,22,23... se utilizan para seleccionar el submenu 1,2,3...
   for (i = 21; i<= (20+cantsubmenus);i++)
    {    dzona[i][0] = posicion[i-20]*HC-HC;
         dzona[i][1] = 0;
         dzona[i][2] = dzona[i][0]+strlen(titulos[i-20])*HC;
         dzona[i][3] = AC;
    }

   opcion=1;
   TECLA=NADA;
   while ((TECLA != ENT) & (TECLA != ESC))
    { if(submenu==0) submenu =cantsubmenus;
         if(submenu>cantsubmenus) submenu =1;
         i=1;longopcion=0;
         strcpy(cadena,amenu[submenu]);
         colgantes[i]= strtok(cadena,",");
         while(colgantes[i]!=NULL)
          { colgantes[++i]= strtok(NULL,",");
            longtemp=strlen(colgantes[i-1]);
            longopcion=(longopcion>longtemp)?longopcion:longtemp;
          }
```

```
NFIL=i-1;
HideCursor(); //BORRA ZONA MENU Y LO IMPRIME DE NUEVO
_setcolor(0);_rectangle (_GFILLINTERIOR,0,0,639,short(AC));
RESALTAF(1,1,amenu[0],15,0);
RESALTAF(1,posicion[submenu],titulos[submenu],15,4);
ShowCursor();
 //      Obtencion de NTOTAL,lonuni,u,nsimult,nz,cantidad_zonas
NCOL=1;X1=posicion[submenu];Y1=2;BORRAR=SI;CF=0;
lonuni=longopcion;NTOTAL=NFIL;
ShowCursor();
NCOL =1;
separa=longopcion+1;              //Separación entre columnas
nsimult=NCOL*NFIL;                //Número de elementos simultáneos
p=1;                              //Primer elemento a visualizar
u=NFIL;                           //Ultimo elemento a visualizar
xf=X1+NCOL*separa;                //Pos X último caracter en el rectangulo
yf=Y1+NFIL;                       //Pos Y último caracter en el rectangulo
largo=xf-X1+2;                    //Long horiz en caracteres del rectangulo
nz=NFIL*NCOL;                     //Nº zonas interiores para ZonaRaton
cantidad_zonas=20+cantsubmenus;   //Nº zonas totales para ZonaRaton

// MEMORIZACION DE ZONAS PARA SELECCION OPCION (ZONA 1,2,...19,20)
for (i=1;i<=nz;i++)     //zonas de opciones de un submenu
 {  j=i/NFIL;k=(i%NFIL);
   if(!k)
        {k = NFIL; j = j - 1;}
        dzona[i][0] = (X1-1+separa*j)*HC;
        dzona[i][1] = (Y1-2+k)*AC;
        dzona[i][2] = dzona[i][0]+separa*HC;
```

```
            dzona[i][3] = dzona[i][1]+AC;
        }

// CALCULO DEL RECUADRO BLANCO EXTERNO (REF. xizq,ysup,xder,yinf)
        ORDVERT((HC*(X1-2)),(AC*(Y1-2)),(HC*(xf+3)),(AC*(yf)));
        xg1=xizq;yg1=ysup+AC/2;xg2=xder+HC;yg2=yinf+AC;
        if ((xg1<0)||(xg2>639)||(yg1<0)||(yg2>479))
          {   printf("\nSobrepasa LA PANTALLA  Pulse una tecla");getch();exit(1);}
        HideCursor();
        // MEMORIZACION DE LA ZONA PARA RESTITUIR AL FINALIZAR
        if (BORRAR==SI)
          {  AGET(xg1,yg1,xg2,yg2);lineabuff1=lineabuff;}
        // DIBUJO DE LOS RECUADROS DE LA ZONA GLOBAL
        ORDVERT((HC*(X1-2)),(AC*Y1-2),(HC*(xf-1)),(AC*yf-AC));
        _setcolor(0);
        _rectangle (_GBORDER,short(xizq+1), short(ysup-AC+1), short(xder-HC/2+1),
                                short(ysup-AC+1)); // Recuadro titulo
        _setcolor(9);
        _rectangle (_GFILLINTERIOR, short(xizq+HC), short(ysup-AC+2),
                                short(xder+HC/2),short(yinf+AC/2-1));// Sombra
        _setcolor(15);
        _rectangle (_GBORDER,short(xizq+1),short(ysup-AC),short(xder-HC/2+1),
                                short(yinf));    // Recuadro blanco

        // ZONAS PARA MOVIMIENTO POR RATON
        nselec=p;                       // selecciona LA OPCION 1
        DELIMITA();
        LISTABLOQUE1(colgantes);
        RESALTA1(); // PASA A VIDEO INVERSO LA OPCION 1
```

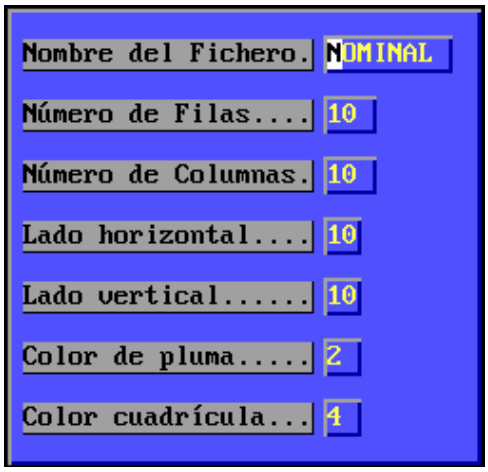
```
TECLA=NADA;
while ((TECLA!=ENT)&(TECLA!=ESC)&(TECLA!=IZQ)&(TECLA!=DER))
 {
  TECLA=MULTIPLENTRADA(); // devuelve TECLA
  if (TECLA<=FIN) VIDEONORMAL1();
                  // VUELVE A VIDEO NORMAL LO RESALTADO
  if ((numero_zona) && (TECLA==ENT))
   {      switch (numero_zona - nz)
          {      case 1:TECLA = ARR;break;
                 case 2:TECLA = ABA;break;
                 case 3:TECLA = PGD;break;
                 case 4:TECLA = PGU;break;
                 case 5:TECLA = IZQ;break;
                 case 6:TECLA = DER;break;
                 case 7:TECLA = PGU;break;
                 case 8:TECLA = PGD;break;
                 default:
                   if(b_rat) nselec=p+numero_zona-1;
                   TECLA = ENT;break;
          }
   }
  switch (TECLA)
   {
        case IZQ:
          nselec=1;   submenu=submenu-1;  break;
        case DER:
          nselec=u;   submenu=submenu+1;  break;
        case ABA:
          nselec=nselec+1;
```

```
            if (nselec > u)
              { nselec=1;
                  DELIMITA();LISTABLOQUE1(colgantes);
               }
            RESALTA1();break;
          case ARR:
            nselec=nselec-1;
            if (nselec < p)
              { nselec=u;
                  DELIMITA();LISTABLOQUE1(colgantes);
              }
            RESALTA1();break;
          case ESC:
            opcion=0;RESALTA1();break;
          case ENT:
            opcion=nselec;break;
          default:;   // PULSACIONES DE TECLAS NO CONSIDERADAS
      }       // fin del switch (TECLA)
   }       // fin de while (TECLA) excepto DER IZQ y cambio de submenu
          //TRAS PULSAR ENTER O ESCAPE
if (BORRAR == 1)
  {  HideCursor();
    lineabuff=lineabuff1;
    APUT(xg1,yg1,xg2,yg2);
    _ffree(lineabuff);_ffree(lineabuff1);
    ShowCursor();
  }
// CONTROL DE PULSACION EXTERIOR CON RATON (salto de submenu)
if ((numero_zona - 20)>0)
```

```
            {   HideCursor();
                lineabuff=lineabuff1;
                APUT(xg1,yg1,xg2,yg2);
                _ffree(lineabuff);_ffree(lineabuff1);
                ShowCursor();
                submenu=numero_zona - 20;nselec=1;TECLA=NADA;
                numero_zona=0;
            }
     }    // fin de while ((TECLA != ENT) & (TECLA != ESC))
   HideCursor();         //BORRA ZONA MENU Y LO IMPRIME DE NUEVO
   lineabuff=buffmenu;
   APUT(0,0,639,16);_ffree(lineabuff);_ffree(buffmenu);
   ShowCursor();
} //fin de rutina selecmenu
```

//**********

La función que se presenta a continuación es la que nos posibilita la entrada de datos en el sistema de forma interactiva con posibilidad de acceso aleatorio a los datos, navegación, corrección de errores, incorporando llamadas a otras funciones auxiliares. (como por ejemplo confirmar la aceptación de los datos presentados en pantalla).




```
void RUTICAMPO(int BOR_RUTI,int CONFIR_RUTI,int EDI_RUTI,
                              int NUM,struct entradas *campo)
                    // entrada-salida de datos
{
   int i;
   int CF,TECLA,TECLA1;    //color final,tecla,tecla de confirmacion
   int HORIZ,VERT;      // POSICION CARACTER ACTIVO
   int HMIN,HMAX,VMIN,VMAX;//Valores extremos de posición de campos
   int NUMC;              //nº caracter activo en el campo
   int INSER=0;           //Modo de inserción desactivado

// Inicializacion de variables
   for (i=0;i<100;i++)
    {     dzona[i][0] = 0;
```

```
        dzona[i][1] = 0;
        dzona[i][2] = 0;
        dzona[i][3] = 0;
     }
   HMIN=100;VMIN=100;VMAX=0;HMAX=0;
   cantidad_zonas=NUMAX+1; //Nº zonas para ZonaRaton
   for(i=NUMIN;i<=NUMAX;i++)
        {     //------CALCULO DE LA ZONA A ALMACENAR------
           HMIN=(HMIN> int((campo[i].hi-strlen(campo[i].ni))))
                                ? (campo[i].hi-strlen(campo[i].ni)) : HMIN;
           VMIN=(VMIN>campo[i].vi) ? campo[i].vi :VMIN;
           HMAX=(HMAX<(campo[i].hi+campo[i].li))
                                ? (campo[i].hi+campo[i].li) : HMAX;
           VMAX=(VMAX<campo[i].vi) ? campo[i].vi :VMAX;
        }
// ALMACENAMIENTO PANTALLA ZONA CAMPOS A EDITAR
   HMIN=HMIN-3;
   VMIN=VMIN-2;
   HMAX=HMAX+1;
   VMAX=VMAX+1;
   HideCursor();
   AGET(HMIN*HC,VMIN*AC,HMAX*HC,VMAX*AC);
   lineabuff1=lineabuff;
   BOTON(xizq+2,ysup+2,xder-2,yinf-2);

//  DEFINICION DE ZONAS DE RATON E IMPRESION DE CAMPOS
// Zona global de control(toda la pantalla)
     //NUMAX+1 ZONA FUERA DE CAMPOS
     dzona[NUMAX+1][0] = 0;
```

```
    dzona[NUMAX+1][1] = 0;
    dzona[NUMAX+1][2] = 639;
    dzona[NUMAX+1][3] = 479;

// Zonas para control de campos
   for(i=NUMIN;i<=NUMAX;i++)
       {   //------IMPRESION DE MENSAJES DE CAMPOS------
          HideCursor();_setcolor(0);
  _rectangle (_GFILLINTERIOR,short(int(campo[i].hi-strlen(campo[i].ni)-2)*HC),
       short((campo[i].vi-1)*AC),short((campo[i].hi-2)*HC),short((campo[i].vi)*AC));
          temp1=campo[i].ni;
       RESALTAF(campo[i].vi,(campo[i].hi-strlen(campo[i].ni)-1),temp1,0,7);
          //IMPRESION DE TODOS LOS CAMPOS Y LIMITACION DE ZONAS
          dzona[i][0] = (campo[i].hi-1)*HC; dzona[i][1] = (campo[i].vi-1)*AC;
          dzona[i][2] = dzona[i][0]+campo[i].li*HC-1;dzona[i][3] = dzona[i][1]+AC-1;
          BOTON(dzona[i][0],dzona[i][1],dzona[i][2],dzona[i][3]);
          switch (campo[i].ti)
          {
                case ND: //alinea a la derecha
                      if (i==NUMIN)
                      {temp1=campo[i].var;campo[i].var=AlineaIzq(temp1);break;}
                  temp1=campo[i].var;
                  campo[i].var=AlineaDer(temp1);
                  break;
                case NI: //alinea a la izquierda
                  temp1=campo[i].var;campo[i].var=AlineaIzq(temp1);
                  break;
                default: //no alinea
                  break;
```

```
            }
            temp1=campo[i].var;RESALTAF(campo[i].vi,campo[i].hi,temp1,14,9);
        }          //------FIN IMPRESION DE MENSAJES DE CAMPOS------

if(EDI_RUTI)
{       //---EDI_RUTI=1 PARA editar CAMPOS
        // ALMACENAMIENTO PANTALLA ZONA TECLA Ins
 HideCursor();
 AGET(600,460,639,479);lineabuff3=lineabuff;
 temp1="Ins";RESALTAF(30,77,temp1,15,0);
 // Edición de campos
 // Posibilita entrar un carácter, tecla de control o pulsación ratón
 ShowCursor();ShowCursor();
 HORIZ=campo[NUM].hi;VERT=campo[NUM].vi;
 TECLA=NADA;TECLA1=NADA;CF=4;numero_zona=0;
if(!CONFIR_RUTI) TECLA=ENT;          //no confirma
while ((TECLA != ENT) & (TECLA != ESC))
  {
     RESALTAC(VERT,HORIZ,campo[NUM].var[HORIZ-campo[NUM].hi],0,15);
   numero_zona=0;                //Inicializa Zona de Rat¢n
   TECLA=MULTIPLENTRADA();            //Devuelve TECLA
   RESALTAC(VERT,HORIZ,campo[NUM].var[HORIZ-campo[NUM].hi],14,9);
   switch (TECLA)
     {
     case DER:
       HORIZ=HORIZ+1;
       if (HORIZ==(campo[NUM].hi+campo[NUM].li))
             HORIZ=campo[NUM].hi;
       break;
```

```
case IZQ:
   HORIZ=HORIZ-1;
   if (HORIZ<campo[NUM].hi)
         HORIZ=campo[NUM].hi+campo[NUM].li-1;
   break;
case ABA:
case TAB:
   if ((campo[NUM].ti==ND)&&(campo[NUM].var[campo[NUM].li-1]==' '))
     {  temp1=campo[NUM].var;
        campo[NUM].var=AlineaDer(temp1);
        temp1=campo[NUM].var;
        RESALTAF(campo[NUM].vi,campo[NUM].hi,temp1,14,9);
     }
   else if ((campo[NUM].ti==NI)&&(campo[NUM].var[0]==' '))
     {  temp1=campo[NUM].var;
        campo[NUM].var=AlineaIzq(temp1);
        temp1=campo[NUM].var;
        RESALTAF(campo[NUM].vi,campo[NUM].hi,temp1,14,9);
     }
   NUM++;
   if(NUM>NUMAX) NUM=NUMIN;
     HORIZ=campo[NUM].hi;VERT=campo[NUM].vi;
        if (((campo[NUM].ti==ND) || (campo[NUM].ti==NI)) &&
              (campo[NUM].var[0]==' '))
        {     temp1=campo[NUM].var;
              campo[NUM].var=AlineaIzq(temp1);
              temp1=campo[NUM].var;
              RESALTAF(campo[NUM].vi,campo[NUM].hi,temp1,14,9);
        }
```

```
    break;
  case ARR:
    if ((campo[NUM].ti==ND)&&(campo[NUM].var[campo[NUM].li-1]==' '))
      {   temp1=campo[NUM].var;
          campo[NUM].var=AlineaDer(temp1);
          temp1=campo[NUM].var;
          RESALTAF(campo[NUM].vi,campo[NUM].hi,temp1,14,9);
      }
    else if ((campo[NUM].ti==NI)&&(campo[NUM].var[0]==' '))
      {  temp1=campo[NUM].var;
          campo[NUM].var=AlineaIzq(temp1);
          temp1=campo[NUM].var;
          RESALTAF(campo[NUM].vi,campo[NUM].hi,temp1,14,9);
      }
    NUM--;
    if(NUM<NUMIN) NUM=NUMAX;
     HORIZ=campo[NUM].hi;VERT=campo[NUM].vi;
           if (((campo[NUM].ti==ND) || (campo[NUM].ti==NI)) &&
                      (campo[NUM].var[0]==' '))
                {         temp1=campo[NUM].var;
                          campo[NUM].var=AlineaIzq(temp1);
                          temp1=campo[NUM].var;
                       RESALTAF(campo[NUM].vi,campo[NUM].hi,temp1,14,9);
                }
    break;
  case DEL:
    NUMC=HORIZ-campo[NUM].hi;
    temp1=campo[NUM].var;
    campo[NUM].var=Cierraesp(temp1,NUMC);
```

```
   campo[NUM].var=temp1;
   RESALTAF(campo[NUM].vi,campo[NUM].hi,temp1,14,9);
   break;
case RET:
   if (HORIZ==campo[NUM].hi) {printf("\a");break;}
   NUMC=HORIZ-campo[NUM].hi-1;
   temp1=campo[NUM].var;
   campo[NUM].var=Cierraesp(temp1,NUMC);
   campo[NUM].var=temp1;
   RESALTAF(campo[NUM].vi,campo[NUM].hi,temp1,14,9);
   HORIZ=HORIZ-1;
   break;
case INS:
   if (INSER)
        {INSER=0;temp1="Ins";RESALTAF(30,77,temp1,15,0);}
   else  {INSER=1;temp1="Ins";RESALTAF(30,77,temp1,0,15);}
   break;
case ESP:
   NUMC=HORIZ-campo[NUM].hi;
   if (INSER)
        {temp1=campo[NUM].var;
         campo[NUM].var=Abreesp(temp1,NUMC);
         temp1=campo[NUM].var;
        RESALTAF(campo[NUM].vi,campo[NUM].hi,temp1,14,9);
        }
   else
        {campo[NUM].var[NUMC]=' ';
        RESALTAC(VERT,HORIZ,campo[NUM].var[HORIZ,
                        campo[NUM].hi],14,9);
```

```
            }
        HORIZ=HORIZ+1;
        if (HORIZ==(campo[NUM].hi+campo[NUM].li))
             HORIZ=campo[NUM].hi;
        break;
    // Opciones de salida y ratón
    case ESC:
       break;
    case ENT:
       if(campo[NUM].ti==ND) //alinea derecha?
            {       temp1=campo[NUM].var;
                    campo[NUM].var=AlineaDer(temp1);
                    temp1=campo[NUM].var;
                    RESALTAF(campo[NUM].vi,campo[NUM].hi,temp1,14,9);
            }
             // Zonas de campos válidos( solo al pulsar raton)
       if (!numero_zona)     // tecla ENTER
            {
             numero_zona=0;break;
            }
       else if(numero_zona>NUMAX) //RATON FUERA ZONAS
            { TECLA=NADA;numero_zona=0;printf("\a");break;}
       else    //CAMPOS VALIDOS
            { NUM=numero_zona;
              VERT=campo[NUM].vi;HORIZ = x_rat / HC+1;
              TECLA=NADA;numero_zona=0;break;
            }

    default:    // Resto de teclas
```

```
NUMC=HORIZ-campo[NUM].hi;
if((TECLA>57 & TECLA<169)|(TECLA>32 & TECLA<43)|(TECLA==44))
                //caracteres imprimibles (excepto cifras)
  {  if ((campo[NUM].ti !=ND)&&(campo[NUM].ti !=NI))
        { if (INSER)
             {temp1=campo[NUM].var;
              campo[NUM].var=Abreesp(temp1,NUMC);
              temp1=campo[NUM].var;
             RESALTAF(campo[NUM].vi,campo[NUM].hi,temp1,14,9);
             }
          campo[NUM].var[NUMC]=char(TECLA);
          RESALTAC(VERT,HORIZ,campo[NUM].var[HORIZ,
                    campo[NUM].hi],14,9);//modo normal
          HORIZ=HORIZ+1;
             if (HORIZ==(campo[NUM].hi+campo[NUM].li))
                    HORIZ=campo[NUM].hi;
          break;
         }
       else {printf("\a");break;}
  }
if((TECLA>47 & TECLA<58)|(TECLA==45)|(TECLA==46))
  {
     if (INSER)
       {temp1=campo[NUM].var;
        campo[NUM].var=Abreesp(temp1,NUMC);
        temp1=campo[NUM].var;
        RESALTAF(campo[NUM].vi,campo[NUM].hi,temp1,14,9);
       }
     campo[NUM].var[NUMC]=char(TECLA);
```

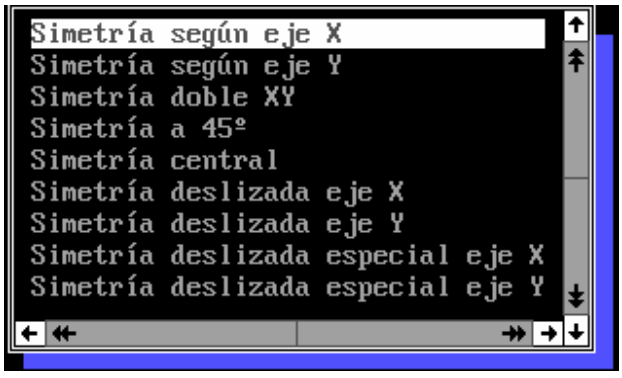
```
                RESALTAC(VERT,HORIZ,campo[NUM].var[HORIZ,
                        campo[NUM].hi],14,9);//modo normal
                HORIZ=HORIZ+1;
                if (HORIZ==(campo[NUM].hi+campo[NUM].li))
                    HORIZ=campo[NUM].hi;
                  break;
              }
            TECLA=NADA;
            break;

        } //fin del switch (TECLA)
    }    // fin while ((TECLA != ENT) & (TECLA != ESC))

  //RESTAURACION DE LA PANTALLA (zona TECLA Ins)
     lineabuff=lineabuff3;
     HideCursor();
     APUT(600,460,639,479);
    _ffree(lineabuff);_ffree(lineabuff3);
  }  //  FIN if(EDI_RUTI)---con EDI_RUTI=0 no edita CAMPOS

  if (CONFIR_RUTI) CONFIRMAR(); // Confirmar aqui

//RESTAURACION DE LA PANTALLA (zona campos)
   if(BOR_RUTI)
     { lineabuff=lineabuff1;
        HideCursor();
        APUT(HMIN*HC,VMIN*AC,HMAX*HC,VMAX*AC);
     }
    _ffree(lineabuff);_ffree(lineabuff1);
```

```
    ShowCursor();ShowCursor();
    TECLA=NADA;TECLA1=NADA;numero_zona=0;
    EDI_RUTI=1;//---Para que edite por defecto
} //  fin de RUTICAMPO
```

//**********

A continuación se presenta la función que permite mostrar diferentes opciones de decisión, facilitando escoger cualquiera de ellas de forma interactiva; esta función, denominada selecmatriz(), puede mostrar cualquier sucesión de opciones si previamente se le adjudica una matriz de elementos. De este modo se presentan los distintos archivos que se pretenden abrir, guardar, etc. así como las distintas opciones de movimientos en simetrías, giros, etc.

La navegación en matrices con un número elevado de elementos, se consigue incorporando a la misma un control de visualización basado en el número de filas y de columnas para delimitar la zona de pantalla utilizada.




```
void selecmatriz(int *indice,char *matriz[])
{
   int i,j,k,xg1,yg1,xg2,yg2,largo,TECLA;
   char *car;
```

```
//      Inicialización de variables
   for (i=0;i<100;i++)
        {       dzona[i][0] = 0;
                dzona[i][1] = 0;
                dzona[i][2] = 0;
                dzona[i][3] = 0;
        }
   //   Obtencion de NTOTAL,lonuni,u,nsimult,nz,cantidad_zonas
   ShowCursor();lonuni=0;u=0;
   NTOTAL=0;car="nada";   // NTOTAL = nº de elementos de la matriz
   while (car != NULL)
        {  if(int(lonuni) < int(strlen(matriz[NTOTAL])))
           lonuni=strlen(matriz[NTOTAL]);
           NTOTAL++;
           car=matriz[NTOTAL];
        }
   NTOTAL=NTOTAL-1;
   if (u>NTOTAL)
        u=NTOTAL;
   if (NTOTAL < NFIL)
        NFIL = NTOTAL;
   if (NFIL*NCOL > NTOTAL)
       NCOL = (NTOTAL - 1) / NFIL + 1;
   separa=lonuni+1;
   nsimult=NCOL*NFIL;
   p=1;                         //Primer elemento a visualizar
   u=p+nsimult-1;               //Ultimo elemento a visualizar
   xf=X1+NCOL*separa;           //Posicion X del último caracter en el rectangulo
   yf=Y1+NFIL;                  //Posicion Y del último caracter en el rectangulo
```

```
   largo=xf-X1+2;              //Longitud horizontal en caracteres del rectangulo
   nz=NFIL*NCOL;               //Numero de zonas interiores para ZonaRaton
   cantidad_zonas=nz+8;        //Numero de zonas totales para ZonaRaton

   //     MEMORIZACION DE ZONAS PARA SELECCIONAR OPCION
   for (i = 1; i<= nz;i++)
         {   j = i / NFIL;
             k = (i % NFIL);
            if (!k)
                  {k = NFIL; j = j - 1;}
            dzona[i][0] = (X1-1+separa*j)*HC;
            dzona[i][1] = (Y1-2+k)*AC;
            dzona[i][2] = dzona[i][0]+separa*HC;
            dzona[i][3] = dzona[i][1]+AC;
         }
   // RECUADRO BLANCO EXTERNO (REF. xizq,ysup,xder,yinf)
ORDVERT((HC*(X1-2)-HC/2),(AC*(Y1-2)),(HC*(xf+3)),(AC*(yf)));
xg1=xizq;
yg1=ysup+AC/2;
xg2=xder+HC;
yg2=yinf+AC;
if ((xg1 < 0) || (xg2 > 639) || (yg1 < 0) || (yg2 > 479))
         {printf("SOBREPASA LA PANTALLA    Pulse una tecla");
           getch();exit(1);
         }
HideCursor();
   //        MEMORIZACION DE LA ZONA PARA RESTITUIR AL FINALIZAR
if (BORRAR==SI)
{        AGET(xg1,yg1,xg2,yg2);
```

```
        lineabuff1=lineabuff;
}
   //        DIBUJO DE LOS RECUADROS DE LA ZONA GLOBAL
ORDVERT((HC*(X1-2)-HC/2),(AC*Y1-(3*AC)/2),(HC*(xf+1)),(AC*yf+(AC/2)));
_setcolor(9); //sombreado y recuadro externo e interno
_rectangle (_GFILLINTERIOR,short(xizq+HC),short(ysup+AC),short(xder+HC+2),
                                    short(yinf+AC/2));
_setcolor(0);
_rectangle (_GFILLINTERIOR,short(xizq),short(ysup),short(xder+1),short(yinf));
_setcolor(15);
_rectangle (_GBORDER,short(xizq),short(ysup),short(xder+1),short(yinf));
_rectangle (_GBORDER,short(xizq+HC/4),short(ysup+HC/4),short(xder-AC/8+1),
                                    short(yinf-AC/8));

   //        ZONAS PARA MOVIMIENTO POR RATON
i=nz+1;  //1ª zona para seleccionar opción anterior
xzonarat=(xf-1)*HC+1;
yzonarat=Y1*AC-AC/3;
dzona[i][0]=xzonarat;
dzona[i][2]=xzonarat+HC+5;
dzona[i][1]=yzonarat;
dzona[i][3]=yzonarat-AC;
_setcolor(15);
_rectangle (_GFILLINTERIOR,short(dzona[i][0]),short(dzona[i][1]),short(dzona[i][2]),
                                    short(dzona[i][3]));
_setcolor(0);
_rectangle (_GBORDER,short(dzona[i][0]),short(dzona[i][1]),short(dzona[i][2]),
                                    short(dzona[i][3]));
_setcolor(0);
```

```
Flecha_arriba(xzonarat+HC/2+2,yzonarat-AC+AC/4);
_setcolor(8);
i=nz+2;  // 2ª zona para seleccionar opción siguiente
yzonarat=(Y1+NFIL)*AC+AC/3;
dzona[i][0]=xzonarat;
dzona[i][2]=xzonarat+HC+5;
dzona[i][1]=yzonarat;
dzona[i][3]=yzonarat-AC;
 _setcolor(15);
_rectangle (_GFILLINTERIOR,short(dzona[i][0]),short(dzona[i][1]),short(dzona[i][2]),
                                        short(dzona[i][3]));
_setcolor(0);
_rectangle (_GBORDER,short(dzona[i][0]),short(dzona[i][1]),short(dzona[i][2]),
                                        short(dzona[i][3]));
_setcolor(0);Flecha_abajo((xzonarat+HC/2+2),(yzonarat-AC/4));
_setcolor(8);
i=nz+3;  // 3ª zona para seleccionar página anterior
dzona[i][0]=xzonarat;
dzona[i][2]=xzonarat+HC+5;
dzona[i][1]=yzonarat-AC;
dzona[i][3]=(ysup+yinf)/2;
_setcolor(7);
_rectangle (_GFILLINTERIOR,short(dzona[i][0]),short(dzona[i][1]),short(dzona[i][2]),
                                    short(dzona[i][3]));
_setcolor(8);
_rectangle (_GBORDER,short(dzona[i][0]),short(dzona[i][1]),short(dzona[i][2]),
                                    short(dzona[i][3]));
   // DOBLE FLECHA ABAJO
_setcolor(0);
```

```
Flecha_abajo((xzonarat+HC/2+2),(yzonarat-AC-AC/4));
Flecha_abajo((xzonarat+HC/2+2),(yzonarat-AC-AC/2));
_setcolor(8);
i=nz+4;  //4ª zona para seleccionar página siguiente
yzonarat=Y1*AC-AC/3;
dzona[i][0]=xzonarat;
dzona[i][2]=xzonarat+HC+5;
dzona[i][1]=yzonarat;
dzona[i][3]=(ysup+yinf)/2;
_setcolor(7);
_rectangle (_GFILLINTERIOR,short(dzona[i][0]),short(dzona[i][1]),short(dzona[i][2]),
                                    short(dzona[i][3]));
_setcolor(8);
_rectangle (_GBORDER,short(dzona[i][0]),short(dzona[i][1]),short(dzona[i][2]),
                                    short(dzona[i][3]));
   // DOBLE FLECHA ARRIBA
_setcolor(0);
Flecha_arriba(xzonarat+HC/2+2,yzonarat+AC/4);
Flecha_arriba(xzonarat+HC/2+2,yzonarat+AC/2);
i=nz+5;  // 5ª zona para seleccionar opción a la izquierda
xzonarat=(X1-2)*HC-1;
yzonarat=(Y1+NFIL)*AC+AC/3;
dzona[i][0]=xzonarat;
dzona[i][2]=xzonarat+2*HC;
dzona[i][1]=yzonarat;
dzona[i][3]=yzonarat-(AC*3)/4-1;
_setcolor(15);
_rectangle (_GFILLINTERIOR,short(dzona[i][0]),short(dzona[i][1]),short(dzona[i][2]),
                                    short(dzona[i][3]));
```

```
_setcolor(0);
_rectangle (_GBORDER,short(dzona[i][0]),short(dzona[i][1]),short(dzona[i][2]),
                                    short(dzona[i][3]));
_setcolor(0);
Flecha_izquierda(xzonarat+HC/3+2,yzonarat-AC/4-2);
 _setcolor(8);
i=nz+6;  //6ª zona para seleccionar opción a la derecha
xzonarat=(xf-1)*HC+1;
dzona[i][0]=xzonarat;
dzona[i][2]=xzonarat-2*HC;
dzona[i][1]=yzonarat;
dzona[i][3]=yzonarat-(AC*3)/4-1;
_setcolor(15);
_rectangle (_GFILLINTERIOR,short(dzona[i][0]),short(dzona[i][1]),short(dzona[i][2]),
                                    short(dzona[i][3]));
_setcolor(0);
_rectangle (_GBORDER,short(dzona[i][0]),short(dzona[i][1]),short(dzona[i][2]),
                                    short(dzona[i][3]));
_setcolor(0);
Flecha_derecha((xzonarat-HC/2),(yzonarat-AC/4-2));
_setcolor(8);
i=nz+7;  //7ª zona para seleccionar salto grupo izquierda
xzonarat=X1*HC-1;
dzona[i][0]=xzonarat;
dzona[i][2]=(X1+xf-2)*HC/2;
dzona[i][1]=yzonarat;
dzona[i][3]=yzonarat-(AC*3)/4-1;
_setcolor(7);
_rectangle (_GFILLINTERIOR,short(dzona[i][0]),short(dzona[i][1]),short(dzona[i][2]),
```

```
                                        short(dzona[i][3]));
_setcolor(8);
_rectangle (_GBORDER,short(dzona[i][0]),short(dzona[i][1]),short(dzona[i][2]),
                                        short(dzona[i][3]));
_setcolor(0);
Flecha_izquierda(xzonarat+HC/3+2,yzonarat-AC/4-2);
Flecha_izquierda(xzonarat+HC,yzonarat-AC/4-2);
 _setcolor(8);
i=nz+8;  //8ª zona para seleccionar salto grupo derecha
xzonarat=(xf-3)*HC+1;
dzona[i][0]=xzonarat;
dzona[i][2]=(X1+xf-2)*HC/2;
dzona[i][1]=yzonarat;
dzona[i][3]=yzonarat-(AC*3)/4-1;
_setcolor(7);
_rectangle (_GFILLINTERIOR,short(dzona[i][0]),short(dzona[i][1]),short(dzona[i][2]),
                                        short(dzona[i][3]));
_setcolor(8);
_rectangle (_GBORDER,short(dzona[i][0]),short(dzona[i][1]),short(dzona[i][2]),
                                        short(dzona[i][3]));
_setcolor(0);Flecha_derecha(xzonarat-HC,yzonarat-AC/4-2);
Flecha_derecha(xzonarat-HC/3-2,yzonarat-AC/4-2);
nselec=p;
DELIMITA(); // selecciona LA OPCION 1ª
LISTABLOQUE(matriz);
RESALTA(); // PASA A VIDEO INVERSO LA OPCION 1ª
TECLA=NADA;
while ((TECLA != ENT) & (TECLA != ESC))
        {  TECLA=MULTIPLENTRADA(); // devuelve TECLA
```

```
if (TECLA<=FIN)
        VIDEONORMAL(); //vuelve a viedo norml
if ((numero_zona) && (TECLA==ENT))
  {     switch (numero_zona - nz)
        {   case 1:TECLA = ARR;
                  break;
            case 2:TECLA = ABA;
                  break;
            case 3:TECLA = PGD;
                  break;
            case 4:TECLA = PGU;
                  break;
            case 5:TECLA = IZQ;
                  break;
            case 6:TECLA = DER;
                  break;
            case 7:TECLA = PGU;
                  break;
            case 8:TECLA = PGD;
                  break;
            default:
        if(b_rat) nselec = p + numero_zona - 1;
        TECLA = ENT;break;
        }
  }

switch (TECLA)
  {
        case INI:
```

```
            p = 1; u = p + nsimult - 1; nselec = p;
            DELIMITA();LISTABLOQUE(matriz);
            RESALTA();
            break;
    case FIN:
            p=NTOTAL-nsimult+1;u=NTOTAL;nselec=NTOTAL;
            DELIMITA();LISTABLOQUE(matriz);
            RESALTA();
            break;
    case DER:
            nselec=nselec+NFIL;
            if ((nselec > u) && (nselec <= NTOTAL))
                    {p=p+NFIL;u=u+NFIL;}
            DELIMITA();LISTABLOQUE(matriz);
            RESALTA();
            break;
    case IZQ:
            nselec=nselec-NFIL;
            if ((nselec < p) && (nselec > 0))
                    {p=p-NFIL;u=u-NFIL;}
            DELIMITA();LISTABLOQUE(matriz);
            RESALTA();
            break;
    case ABA:
            nselec=nselec+1;
            if (nselec > u)
                    {p=p+NFIL;u=u+NFIL;}
            DELIMITA();LISTABLOQUE(matriz);
            RESALTA();
```

```
                break;
        case ARR:
                nselec=nselec-1;
                if (nselec < p)
                        {       p=p-NFIL;u=u-NFIL;
                                DELIMITA();LISTABLOQUE(matriz);
                        }
                 RESALTA();
                break;
        case PGU:
                p=p-nsimult;u=p+nsimult-1;nselec=p;
                DELIMITA();LISTABLOQUE(matriz);
                RESALTA();
                break;
        case PGD:
                p=u+1;u=p+nsimult-1;nselec=p;
                DELIMITA();LISTABLOQUE(matriz);
                RESALTA();
                break;
        case ESC:
                indice=0;
                RESALTA();
                break;
        case ENT:
                RESALTA();
                *indice=nselec;break;
        default:;   // Teclas no consideradas
    } //fin del switch (TECLA)
}   // fin while ((TECLA != ENT) & (TECLA != ESC))
```

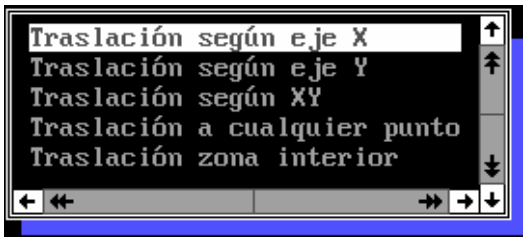
```
                    //TRAS PULSAR ENTER O ESCAPE

if (BORRAR == 1)
          { HideCursor();
            lineabuff=lineabuff1;
            APUT(xg1,yg1,xg2,yg2);_ffree(lineabuff);_ffree(lineabuff1);
            ShowCursor();
          }
}    // fin de rutina selecmatriz
```

A continuación se listan las funciones TRASLADAR(), SIMETRIA(), GIROS(), COPIAR(), RELLENAR(), que van a realizar los movimientos previstos anteriormente en el Capitulo 4 de "TRANSFORMACIONES GEOMETRICAS" a partir de la base de datos gráfica previamente almacenada como retícula o a partir de una nueva retícula que se está creando. Para estos movimientos se usarán dos matrices bidimensionales mat2[i][j] y seg2[i][j], para poder guardar una como seguridad y poder deshacer los cambios en cualquier momento.

En el apartado siguiente 6.3.3 de Funciones auxiliares se describirán las funciones IGUALAMATSEG() y RECUPERAMAT() que ejecutan esta posibilidad de volver atrás.

La primera función de TRASLADAR() nos ofrece a través de selecmatriz(), distintas opciones de traslación, que por su reducido número de posibilidades se presenta una única columna con los items correspondientes asignados a la matriz trasl[i].

Cada uno de los items realiza los cálculos pertinentes para ejecutar el movimiento previsto y llamar finalmente a la función de presentación de la nueva retícula en pantalla que se verá posteriormente reticular(izx,izy).



El resto de funciones que se refieren a los movimientos básicos se comportan del mismo modo, siendo únicamente necesario elegir la opción por el teclado o por el ratón.

```
//***********************************
void TRASLADAR(void)
{
   trasl[1]="Traslación según eje X";
   trasl[2]="Traslación según eje Y";
   trasl[3]="Traslación según XY";
   trasl[4]="Traslación a cualquier punto";
   trasl[5]="Traslación zona interior";
   NFIL=5;NCOL=1;X1=35;Y1=3;BORRAR=SI;CF=0;indice=0;
   selecmatriz(&indice,trasl);
   switch(indice)
         {
            case 1:        // traslada segun eje X
                  IGUALAMATSEG();
                  for(j=1;j<=nc;j++)
                    {  for(i=1;i<=nf;i++)
                       {  seg2[i][j]=f;
                            seg2[i+nf][j]=mat2[i][j];
                       }
                  }
                  nf=2*nf;
                  seg2[1][0]=nf;
                  seg2[0][1]=nc;
                  ZONARETICULA();
                  reticular(izx,izy);
                  break;
            case 2:        // traslada segun eje Y
                  IGUALAMATSEG();
                  for(i=1;i<=nf;i++)
```

```
        {  for(j=1;j<=nc;j++)
           {  seg2[i][j]=f;
                 seg2[i][j+nc]=mat2[i][j];
           }
        }
      nc=2*nc;
      seg2[1][0]=nf;
      seg2[0][1]=nc;
      ZONARETICULA();
      reticular(izx,izy);
      break;
case 3:      // traslada segun XY
      IGUALAMATSEG();
      for(i=1;i<=nf;i++)
        {  for(j=1;j<=nc;j++)
           {  seg2[i][j]=f;
                 seg2[i+nf][j]=f;
                 seg2[i][j+nc]=f;
                 seg2[i+nf][j+nc]=mat2[i][j];
           }
        }
      nf=2*nf; nc=2*nc;
      seg2[1][0]=nf;
      seg2[0][1]=nc;
      ZONARETICULA();
      reticular(izx,izy);
      break;
case 4:      // traslada a cualquier punto
      IGUALAMATSEG();
```

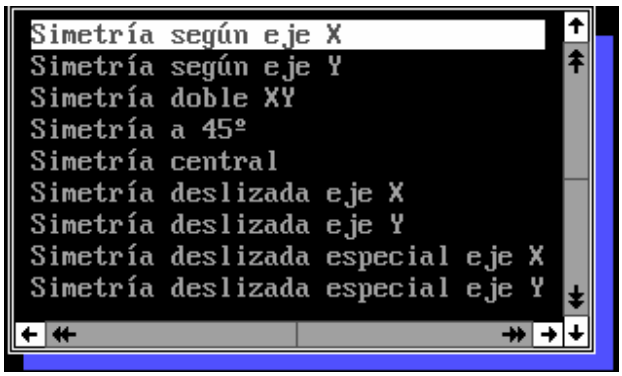
```
        PuntoRaton();
        filref=int((y_rat-izy)/lv);
        colref=int((x_rat-izx)/lh);
        x_rat=izx+colref*lh;
        y_rat=izy+filref*lv;
        colf=abs(colref)+nc;
        filf=abs(filref)+nf;
        for(i=1;i<=filf;i++)
         {  for(j=1;j<=colf;j++)
                 seg2[i][j]=f;
         }
        fili=(filref<1) ? 0 : abs(filref);
        coli=(colref<1) ? 0 : abs(colref);
        for(i=1;i<=nf;i++)
          {  filn=fili+i;
            for(j=1;j<=nc;j++)
                 { coln=coli+j;
                   seg2[filn][coln]=mat2[i][j];
                 }
          }
        nf=filf; nc=colf;
        seg2[1][0]=nf;
        seg2[0][1]=nc;
        ZONARETICULA();
        izx=(izx<x_rat) ? izx : x_rat;
        izy=(izy<y_rat) ? izy : y_rat;
        reticular(izx,izy);
        break;
case 5:         // traslada zona interior
```

```
IGUALAMATSEG();
ZONARETICULA();
xmin_rat=dzona[2][0];
ymin_rat=dzona[2][1];
xmax_rat=dzona[2][2];
ymax_rat=dzona[2][3]-1;
Rangohor();Rangover();
COGEZONA();
_setcolor(15);HideCursor();
_rectangle (_GBORDER, short(avent), short(bvent), short(cvent),
                    short(dvent));
_settextposition(30,1);
printf("Indicar nueva posición vértice superior izquierdo");
ShowCursor(); PuntoRaton();
_settextposition(30,1);printf("                                        ");
for(i=fili;i<=filf;i++)
  {  for(j=coli;j<=colf;j++)
         seg2[i][j]=f; //borra zona
  }
filref=int((y_rat-izy)/lv)+1;
colref=int((x_rat-izx)/lh)+1;
for(i=fili;i<=filf;i++)
  {  filn=i+filref-fili;
    for(j=coli;j<=colf;j++)
         {  coln=j+colref-coli;
           if ((filn<=nf)&&(coln<=nc))
              seg2[filn][coln]=mat2[i][j]; //nueva zona
         }
  }
```

```
                ZONARETICULA();
                retiparcial(izx,izy,fili,coli,filf,colf);
                retiparcial(izx,izy,filref,colref,filref+(filf-fili),colref+(colf-coli));
                ZONAGLOBAL();
                xmin_rat=dzona[0][0];
                ymin_rat=dzona[0][1];
                xmax_rat=dzona[0][2];
                ymax_rat=dzona[0][3];
                Rangohor();Rangover();
                break;
            default:
                 {}
        }//fin switch
}
```

//**********

Aspecto que ofrece el submenu de simetrias con la funcion SIMETRIA() previa llamada a selecmatriz().




```
void SIMETRIA(void)
{
   simet[1]="Simetría según eje X";
   simet[2]="Simetría según eje Y";
   simet[3]="Simetría doble XY";
   simet[4]="Simetría a 45º";
   simet[5]="Simetría central";
   simet[6]="Simetría deslizada eje X";
   simet[7]="Simetría deslizada eje Y";
   simet[8]="Simetría deslizada especial eje X";
   simet[9]="Simetría deslizada especial eje Y";
   NFIL=9;NCOL=1;X1=35;Y1=4;BORRAR=SI;CF=0;indice=0;
   selecmatriz(&indice,simet);
   switch(indice)
         {
            case 1:        // simetria eje X
                  IGUALAMATSEG();
                  for(j=1;j<=nc;j++)
```

```
        {  for(i=1;i<=nf;i++)
             seg2[2*nf-i+1][j]=mat2[i][j];
        }
      fili=nf;nf=2*nf;
      seg2[1][0]=nf;
      seg2[0][1]=nc;
      ZONARETICULA();
      retiparcial(izx,izy,fili,1,nf,nc);
      break;
case 2:       // simetria eje Y
      IGUALAMATSEG();
      for(i=1;i<=nf;i++)
        {  for(j=1;j<=nc;j++)
             seg2[i][2*nc-j+1]=mat2[i][j];
        }
      coli=nc;nc=2*nc;
      seg2[1][0]=nf;
      seg2[0][1]=nc;
      ZONARETICULA();
      retiparcial(izx,izy,1,coli,nf,nc);
      break;
case 3:       // simetria doble XY
      IGUALAMATSEG();
      for(i=1;i<=nf;i++)      // sobre X
        {  for(j=1;j<=nc;j++)
             seg2[2*nf-i+1][j]=mat2[i][j];
        }
      nf=2*nf;          //sobre Y
      for(i=1;i<=nf;i++)
```

```
  {  for(j=1;j<=nc;j++)
            seg2[i][2*nc-j+1]=seg2[i][j];
  }
 nc=2*nc;
 seg2[1][0]=nf; seg2[0][1]=nc;
 ZONARETICULA();
 reticular(izx,izy);
 break;
case 4:   // Simetria 45º
 IGUALAMATSEG();
 for(i=1;i<=nc;i++)
   {  for(j=1;j<=nc;j++)
         seg2[nf+i][j]=f;
    }
 for(i=1;i<=nf;i++)
    {     for(j=1;j<=nf;j++)
           seg2[i][nc+j]=f;
    }
 for(i=1;i<=nf;i++)
  {   for(j=1;j<=nc;j++)
         seg2[nc+nf-j+1][nc+nf-i+1]=mat2[i][j];
  }
 fili=nf;
 nf=nf+nc;  nc=nc+fili;
 seg2[1][0]=nf;
 seg2[0][1]=nc;
 ZONARETICULA();
 reticular(izx,izy);
 break;
```

```
case 5:        // Simetria central
       IGUALAMATSEG();
       for(i=1;i<=nf;i++)
        {   for(j=1;j<=nc;j++)
                {  seg2[nf+i][j]=f;
                   seg2[i][nc+j]=f;
                   seg2[2*nf-i+1][2*nc-j+1]=mat2[i][j];
                 }
        }
       nf=2*nf; nc=2*nc;
       seg2[1][0]=nf;
       seg2[0][1]=nc;
       ZONARETICULA();
       reticular(izx,izy);
       break;
case 6:        // simetria deslizada eje X
       IGUALAMATSEG();
       for(i=1;i<=nf;i++)
        {   for(j=1;j<=nc;j++)
            {  seg2[nf+i][j]=f;
                  seg2[i][nc+j]=f;
                  seg2[2*nf-i+1][j+nc]=mat2[i][j];
            }
        }
       nf=2*nf;  nc=2*nc;
       seg2[1][0]=nf;
       seg2[0][1]=nc;
       ZONARETICULA();
       reticular(izx,izy);
```

```
        break;
case 7:         // simetria deslizada eje Y
        IGUALAMATSEG();
        for(i=1;i<=nf;i++)
         {   for(j=1;j<=nc;j++)
             {  seg2[nf+i][j]=f;
                  seg2[i][nc+j]=f;
                  seg2[i+nf][2*nc-j+1]=mat2[i][j];
             }
         }
        nf=2*nf;  nc=2*nc;
        seg2[1][0]=nf;
        seg2[0][1]=nc;
        ZONARETICULA();
        reticular(izx,izy);
        break;
case 8:         // simetria deslizada especial eje X
        IGUALAMATSEG();
        for(i=1;i<=nf;i++)
         {   for(j=1;j<=nc;j++)
                  seg2[nf-i+1][nc+j]=mat2[i][j];
         }
        coli=nc;nc=2*nc;
        seg2[1][0]=nf;
        seg2[0][1]=nc;
        ZONARETICULA();
        retiparcial(izx,izy,1,coli,nf,nc);
        break;
case 9:         // simetria deslizada especial eje Y
```

```
                IGUALAMATSEG();
                for(i=1;i<=nf;i++)
                  {  for(j=1;j<=nc;j++)
                       seg2[nf+i][nc-j+1]=mat2[i][j];
                  }
                fili=nf;nf=2*nf;
                seg2[1][0]=nf;
                seg2[0][1]=nc;
                ZONARETICULA();
                retiparcial(izx,izy,fili,1,nf,nc);
                break;
           default:
                 {}
        }//fin switch
}

//************************************
void GIROS(void)
{
   giro[1]="1 Giro de 90º horario";
   giro[2]="1 Giro de 180º";
   giro[3]="1 Giro de 270º";
   giro[4]="3 Giros de 90º";
   giro[5]="1 Giro de 180º abatido";
   NFIL=5;NCOL=1;X1=40;Y1=5;BORRAR=SI;CF=0;indice=0;
   selecmatriz(&indice,giro);
   switch(indice)
        {
          case 1: //1 giro 90 horario
```

```
IGUALAMATSEG();
if(nc==nf)
  { for(i=1;i<=nf;i++)
    {   for(j=1;j<=nc;j++)
         seg2[j][2*nc-i+1]=mat2[i][j]; //1 giro derecha
    }
   nc=2*nc;
  }
else if(nc>nf)
  { for(i=1;i<=nf;i++)
    {   for(j=1;j<=nc;j++)
        {  seg2[nc-nf+i][j]=mat2[i][j];
           seg2[j][nc+nf-i+1]=mat2[i][j];
        }
    }
   for(i=1;i<=nc-nf;i++)
    {   for(j=1;j<=nc;j++)
        seg2[i][j]=f;
    }
   fili=nf;
   nf=nc; nc=fili+nc;
  }
else if(nc<nf)
  { for(i=1;i<=nc;i++)
    {   for(j=nc+1;j<=nf+nc;j++)
        seg2[i][j]=f;
    }
   for(i=1;i<=nf;i++)
    {   for(j=1;j<=nc;j++)
```

```
                seg2[nf-nc+j][nc+nf-i+1]=mat2[i][j];
              }
             nc=nc+nf;
           }
        seg2[1][0]=nf; seg2[0][1]=nc;
        ZONARETICULA();
        reticular(izx,izy);
        break;
case 2:        // 1 giro de 180º
        IGUALAMATSEG();
        for(i=1;i<=nf;i++)
         {   for(j=1;j<=nc;j++)
                {  seg2[nf+i][j]=f;
                   seg2[i][nc+j]=f;
                   seg2[2*nf-i+1][2*nc-j+1]=mat2[i][j];
                 }
         }
        nf=2*nf;  nc=2*nc;
        seg2[1][0]=nf; seg2[0][1]=nc;
        ZONARETICULA();
        reticular(izx,izy);
        break;
case 3:        // 1 giro 270
        IGUALAMATSEG();
        if(nc==nf)
          {  for(i=1;i<=nf;i++)
             { for(j=1;j<=nc;j++)
                   seg2[2*nc-j+1][i]=mat2[i][j];
             }
```

```
      nf=2*nf;
    }
  else if(nc>nf)
    { for(i=nf+1;i<=nf+nc;i++)
      {    for(j=1;j<=nc-nf;j++)
           seg2[i][j]=f;
      }
     for(i=1;i<=nf;i++)
      {    for(j=1;j<=nc;j++)
           seg2[nf+nc-j+1][nc-nf+i]=mat2[i][j];
      }
     nf=nc+nf;
    }
  else if(nc<nf)
    { for(i=1;i<=nf;i++)
      {    for(j=1;j<=nc;j++)
            {   seg2[i][nf-nc+j]=mat2[i][j];
               seg2[nc+nf-j+1][i]=mat2[i][j];
            }
      }
     for(i=1;i<=nf;i++)
      {    for(j=1;j<=nf-nc;j++)
           seg2[i][j]=f;
      }
     fili=nf;
     nf=nc+nf; nc=fili;
    }
  seg2[1][0]=nf; seg2[0][1]=nc;
  ZONARETICULA();
```

```
        reticular(izx,izy);
        break;

case 4:   // 3 giros
        IGUALAMATSEG();
        if(nc==nf)
          {  for(i=1;i<=nf;i++)
             { for(j=1;j<=nc;j++)
                 { seg2[j][2*nc-i+1]=mat2[i][j]; //1 giro horario
                   seg2[2*nc-j+1][i]=mat2[i][j]; //1 giro antihorario
                   seg2[2*nf-i+1][2*nc-j+1]=mat2[i][j]; //1 giro 180º
                 }
             }
            nf=2*nf; nc=2*nc;
          }
        else if(nc>nf)
          { for(i=1;i<=nf;i++)
             {    for(j=1;j<=nc;j++)
                  {   seg2[nc-nf+i][j]=mat2[i][j];    //1 horario
                      seg2[j][nc+nf-i+1]=mat2[i][j];  // 1horario
                      seg2[nf+nc-i+1][2*nc-j+1]=mat2[i][j];// 180º
                      seg2[nc+nc-j+1][nc-nf+i]=mat2[i][j];  // antihor
                   }
             }
            for(i=1;i<=nc-nf;i++) //rellenar huecos
             {    for(j=1;j<=nc;j++)
                  { seg2[i][j]=f;
                    seg2[j][nc+nf+i]=f;
                    seg2[nc+nf+i][nc+j]=f;
```

```
            seg2[nc+j][i]=f;
           }
      }
     nf=2*nc; nc=2*nc;
   }
else if(nc<nf)
   { for(i=1;i<=nf;i++)
     {    for(j=1;j<=nc;j++)
          {  seg2[nf-nc+j][nf+nf-i+1]=mat2[i][j];   // 1 horario
            seg2[nf+nf-i+1][nf+nc-j+1]=mat2[i][j]; // 180º
            seg2[i][nf-nc+j]=mat2[i][j];            // antihor
            seg2[nf+nc-j+1][i]=mat2[i][j];          //antihor
          }
      }
     for(i=1;i<=nf;i++) //rellenar huecos
      { for(j=1;j<=nf-nc;j++)
          { seg2[i][j]=f;
            seg2[j][nf+i]=f;
            seg2[nf+i][nf+nc+j]=f;
            seg2[nc+nf+j][i]=f;
          }
      }
     nc=2*nf; nf=2*nf;
   }
seg2[1][0]=nf;
seg2[0][1]=nc;
ZONARETICULA();
 reticular(izx,izy);;
break;
```

```
        case 5:     // 1 giro de 180º abatido
                IGUALAMATSEG();
                for(i=1;i<=nf;i++)
                 {  for(j=1;j<=nc;j++)
                        seg2[2*nf-i+1][nc-j+1]=mat2[i][j];
                 }
                nf=2*nf;
                seg2[1][0]=nf; seg2[0][1]=nc;
                ZONARETICULA();
                reticular(izx,izy);
                break;

          default:
                 {}
       }//fin switch
}
```

```
//************************************
void COPIAR(void)
{
   copia[1]="Copia interior";
   copia[2]="Copia zona";
   copia[3]="Copia en cualquier punto";
   NFIL=3;NCOL=1;X1=40;Y1=6;BORRAR=SI;CF=0;indice=0;
   selecmatriz(&indice,copia);
   switch(indice)
         {
            case 1: //copia interior
                  IGUALAMATSEG();
                  ZONARETICULA();
                  xmin_rat=dzona[2][0];
                  ymin_rat=dzona[2][1];
                  xmax_rat=dzona[2][2];
                  ymax_rat=dzona[2][3];
                  Rangohor();Rangover();
                  PuntoRaton();
                  fili=1;coli=1;
                  filref=int((y_rat-izy)/lv)+1;
                  colref=int((x_rat-izx)/lh)+1;
                  for(i=fili;i<=nf-filref+1;i++)
                    {  filn=i+filref-fili;
                       for(j=coli;j<=nc-colref+1;j++)
                            {  coln=j+colref-coli;
                               seg2[filn][coln]=mat2[i][j];
                            }
                    }
```

```
        ZONARETICULA();
        retiparcial(izx,izy,filref,colref,filref+(nf-fili),colref+(nc-coli));
        ZONAGLOBAL();
        xmin_rat=dzona[0][0];
        ymin_rat=dzona[0][1];
        xmax_rat=dzona[0][2];
        ymax_rat=dzona[0][3];
        Rangohor();Rangover();
        break;
    case 2:   //copia zona
        IGUALAMATSEG();
        ZONARETICULA();
        xmin_rat=dzona[2][0];
        ymin_rat=dzona[2][1];
        xmax_rat=dzona[2][2];
        ymax_rat=dzona[2][3]-1;
        Rangohor();Rangover();
        COGEZONA();
        _setcolor(15);HideCursor();
        _rectangle (_GBORDER, short(avent), short(bvent), short(cvent),
                                    short(dvent));
        _settextposition(30,1);
        printf("Indicar nueva posición vértice superior izquierdo");
        ShowCursor();
        PuntoRaton();
        _settextposition(30,1);
        printf("                                        ");
        filref=int((y_rat-izy)/lv)+1;
        colref=int((x_rat-izx)/lh)+1;
```

```
        for(i=fili;i<=filf;i++)
         {  filn=i+filref-fili;
           for(j=coli;j<=colf;j++)
                {  coln=j+colref-coli;
                  if ((filn<=nf)&&(coln<=nc))
                     seg2[filn][coln]=mat2[i][j];
                 }
         }
         ZONARETICULA();
         retiparcial(izx,izy,fili,coli,filf,colf);
         retiparcial(izx,izy,filref,colref,filref+(filf-fili),colref+(colf-coli));
         ZONAGLOBAL();
         xmin_rat=dzona[0][0];
         ymin_rat=dzona[0][1];
         xmax_rat=dzona[0][2];
         ymax_rat=dzona[0][3];
         Rangohor();Rangover();
         break;
    case 3://copia en cualquier punto
        IGUALAMATSEG();
        PuntoRaton();
        filref=int((y_rat-izy)/lv);
        colref=int((x_rat-izx)/lh);
        x_rat=izx+colref*lh;
        y_rat=izy+filref*lv;
        colf=abs(colref)+nc;
        filf=abs(filref)+nf;
        for(i=1;i<=filf;i++)
         {  for(j=1;j<=colf;j++)
```

```
            seg2[i][j]=f;
  }
fili=(filref<1) ? abs(filref) : 0;
coli=(colref<1) ? abs(colref) : 0;
for(i=1;i<=nf;i++)
  {  filn=fili+i;
     for(j=1;j<=nc;j++)
          { coln=coli+j;
            seg2[filn][coln]=mat2[i][j];
          }
  }
fili=(filref<1) ? 0 : abs(filref);
coli=(colref<1) ? 0 : abs(colref);
for(i=1;i<=nf;i++)
   {  filn=fili+i;
      for(j=1;j<=nc;j++)
          { coln=coli+j;
            seg2[filn][coln]=mat2[i][j];
          }
   }
nf=filf;
nc=colf;
seg2[1][0]=nf;
seg2[0][1]=nc;
ZONARETICULA();
izx=(izx<x_rat) ? izx : x_rat;
izy=(izy<y_rat) ? izy : y_rat;
reticular(izx,izy);
break;
```

```
            default:
                    { }
          }//fin switch
}

//************************************
void RELLENAR(void)
{  rell[1]="Relleno de una Zona";
   rell[2]="Relleno General";
   NFIL=2;NCOL=1;X1=40;Y1=7;BORRAR=SI;CF=0;indice=0;
   selecmatriz(&indice,rell);
   switch(indice)
         {
            case 1:         // relleno de zona
                  IGUALAMATSEG();
                  ZONARETICULA();
                  xmin_rat=dzona[2][0]; ymin_rat=dzona[2][1];
                  xmax_rat=dzona[2][2]; ymax_rat=dzona[2][3];
                  Rangohor();Rangover();
                  COGEZONA();
                  for(i=fili;i<=filf;i++)
                    {  for(j=coli;j<=colf;j++)
                           seg2[i][j]=f;
                    }
                  ZONARETICULA();
                  retiparcial(izx,izy,fili,coli,filf,colf);
                  ZONAGLOBAL();
                  xmin_rat=dzona[0][0];
                  ymin_rat=dzona[0][1];
```

```
                xmax_rat=dzona[0][2];
                ymax_rat=dzona[0][3];
                Rangohor();Rangover();
                break;
          case 2:       // relleno total
                IGUALAMATSEG();
                for(i=1;i<=nf;i++)
                  {  for(j=1;j<=nc;j++)
                        seg2[i][j]=f;
                  }
                reticular(izx,izy);
                break;
          default:
                 {}
        }//fin switch
}
```

```
//************************************
```

A continuación se listan las funciones: paleta(), calculacolor(), cajaajustes(), ajusterojo(), ajusteverde(), ajusteazul(), que hacen referencia a la manipulación del color en términos RGB, posibilitando el ajuste de 64 tonos de color verde, rojo o azul para cada uno de los 16 colores iniciales

```
void paleta(float a,float b,float lhp,float lvp,int nfp,int ncp,short cp)
{     //dibujar paleta
   int contfil,contcol;
   float izqx,izqy;
   short fp=0;
   HideCursor();
   for (contfil=1;contfil<=nfp;contfil++)
         {  for (contcol=1;contcol<=ncp;contcol++)
            {      izqx=a+lhp*(contcol-1);
                   izqy=b+lvp*(contfil-1);
                    _remappalette(fp, color[fp]);
                   rectangulor(izqx,izqy,lhp,lvp,cp,fp);
                   fp++;
            }
         }
   ShowCursor();
}

//************************************
void calculacolor(void)
{   // calcula valor rgb del color y porcentaje
   azulhexa  = (color[f] >> 16) & 0x3F;
   verdehexa = (color[f] >> 8) & 0x3F;
```

```
   rojohexa  = color[f] & 0x3F;
   azul = int(azulhexa*100/63);
   verde =int(verdehexa*100/63);
   rojo =int(rojohexa*100/63);
}

//************************************
void cajaajustes(void)
{   //dibujar caja ajuste color y texto grafico
   int posih,posiv,redcol,grecol,blucol;
   short cc;
   char rinforma[10],ginforma[10],binforma[10];
   char *fuentes = "t'tms rmn'h10w5b";
   if ((_registerfonts("TMSRB.FON")) < 0)  //en directorio
         {_outtext("No se encuentra el fichero TMSRB.FON");getch(); exit(1);}
   redcol=int(455-(75./100.)*rojo);
   grecol=int(455-(75./100.)*verde);
   blucol=int(455-(75./100.)*azul);
   cc=1;if (f==1) cc=15;
   HideCursor();
   rectangulor(560,345,75,120,8,8);
   rectangulor(560,465,75,14,8,8);
   rectangulor(583,350,30,30,c,f);posiv=385;
   if (_setfont(fuentes) >= 0)
    {
         posih=570;
         rectangulor(posih,posiv,4,75,cc,0);
         posih=posih-6;
         rectangulor(posih,redcol,18,10,4,4);
```

```
        _setcolor(15);_moveto_w(posih+4,redcol); _outgtext("R");
         _setcolor(8);  _moveto_w(posih+2,467);_outgtext("888");
        _setcolor(15);  _moveto_w(posih+2,467); _itoa(rojo,rinforma,10 );
        _outgtext(rinforma);
        posih=595;
        rectangulor(posih,posiv,4,75,cc,0);
        posih=posih-6;
        rectangulor(posih,grecol,18,10,2,2);
        _setcolor(15); _moveto_w(posih+4,grecol); _outgtext("G");
         _setcolor(8);  _moveto_w(posih+2,467);_outgtext("888");
        _setcolor(15);  _moveto_w(posih+2,467); _itoa(verde,ginforma,10 );
        _outgtext(ginforma);
        posih=620;
        rectangulor(posih,posiv,4,75,cc,0);
        posih=posih-6;
        rectangulor(posih,blucol,18,10,1,1);
        _setcolor(15); _moveto_w(posih+4,blucol); _outgtext("B");
        _setcolor(8);   _moveto_w(posih+2,467);_outgtext("888");
        _setcolor(15);  _moveto_w(posih+2,467); _itoa(azul,binforma,10 );
        _outgtext(binforma);
    }
   _unregisterfonts();
   ShowCursor();
}


//************************************
void ajusterojo(void)
{   rojo=460-y_rat;     //entre 0 y 75
```

```
   rojohexa=int((rojo/75.)*63); //entre 0 y 3f
   color[f] = azulhexa << 16 | verdehexa << 8 | rojohexa;
   _remappalette(f, color[f]);
}

//************************************
void ajusteverde(void)
{   verde=460-y_rat;
   verdehexa=int((verde/75.)*63);
   color[f] = azulhexa << 16 | verdehexa << 8 | rojohexa;
   _remappalette(f, color[f]);
}

//************************************
void ajusteazul(void)
{   azul=460-y_rat;
   azulhexa=int((azul/75.)*63);
   color[f] = azulhexa << 16 | verdehexa << 8 | rojohexa;
   _remappalette(f, color[f]);
}
```

Finalmente se listan las funciones particulares que nos permiten plantear un dibujo textil en la pantalla en forma de reticula y la simulación parcial o total del tejido, con las posibilidades de mezclar, sustituir, alternar, etc. un determinado color.

```
//**********************************
void reticular(float a,float b)
{   //rellenar reticula con cuadricula (cuadro)
   int contfil,contcol;
   float izqx,izqy;
   short f1;
   HideCursor();
   if (cuadro) //pinta cuadro
    { for (contfil=1;contfil<=nf;contfil++)
        { for (contcol=1;contcol<=nc;contcol++)
           {    izqx=a+lh*(contcol-1);
                izqy=b+lv*(contfil-1);
                f1=short(seg2[contfil][contcol]);
                rectangulor(izqx,izqy,lh,lv,c,f1);
           }
        }
    }
   else // pinta cuadro del mismo color
   { for (contfil=1;contfil<=nf;contfil++)
        { for (contcol=1;contcol<=nc;contcol++)
           {    izqx=a+lh*(contcol-1);
                izqy=b+lv*(contfil-1);
                f1=short(seg2[contfil][contcol]);
                rectangulor(izqx,izqy,lh,lv,f1,f1);
           }
```

```
        }
    }
   ShowCursor();
}

//************************************
void rectangulor(float izqx,float izqy,float lh,float lv,short c,short f )
{       //dibujar rectangulo relleno con borde
   _setcolor(f);
   _rectangle_w(_GFILLINTERIOR,izqx,izqy,izqx+lh,izqy+lv);
   _setcolor(c);
   _rectangle_w(_GBORDER,izqx,izqy,izqx+lh,izqy+lv);
}

//************************************
void rellenareti(int x1,int y1,int x2,int y2,float lh,float lv,int nf,int nc)
{  int contfil,contcol,i,posix,posiy;
   contcol=int((x2-x1)/nc);
   contfil=int((y2-y1)/nf);
   HideCursor();
   retipunto(x1+1,y1+1,lh,lv,nf,nc);
   AGET(x1+1,y1+1,x1+int(nc),y1+int(nf));
   for (i=1;i<=(contcol-1);i++)
        { posix=int((x1+1)+nc*i);posiy=int(y1+1);
          APUT (posix,posiy,posix+int(nc),posiy+int(nf));
          _ffree(lineabuff); //video reverse
        }
   AGET(x1+1,y1+1,x2-nc*contcol-1,y1+int(nf));
   posix=int((x1+1)+nc*contcol);posiy=int(y1+1);
```

```
   APUT (posix,posiy,x2-1,posiy+int(nf));
   _ffree(lineabuff); //video reverse
   AGET(x1+1,y1+1,x2-1,y1+int(nf));
   for (i=1;i<=(contfil-1);i++)
         { posix=(x1+1);posiy=(y1+1)+int(nf*i);
           APUT (posix,posiy,x2-1,posiy+int(nf));
           _ffree(lineabuff); //video reverse
         }
   AGET(x1+1,y1+1,x2-1,y2-nf*contfil-1);
   posix=(x1+1);posiy=(y1+1)+int(nf*contfil);
   APUT(posix,posiy,x2-1,y2-1);
   _ffree(lineabuff); //video reverse
   ShowCursor();ShowCursor();
}

//************************************
void retipunto(float a,float b,float lh,float lv,int nf,int nc)
{   int contfil,contcol;
   float izqx,izqy;
   short f;
   lh=1.0;lv=1.0;
   for (contfil=1;contfil<=nf;contfil++)
         {   for (contcol=1;contcol<=nc;contcol++)
                  { izqx=a+lh*(contcol-1);
                    izqy=b+lv*(contfil-1);
                    f=short(seg2[contfil][contcol]);
                    _setcolor(f);
                    _setpixel_w(izqx,izqy);
                  }
```

```
        }
}

//************************************
void BORRARETICULA(void)
{   int i,j;
   HideCursor(); //limpiar actual
   _setviewport(short(izx),short(izy),short(izx+(nc*lh)),short(izy+(nf*lv)));
   _clearscreen(_GVIEWPORT);
   _setviewport(0,0,cv.numxpixels,cv.numypixels);
        //vaciar matriz actual
        for (i=1;i<=nf;i++)
          { for (j=1;j<=nc;j++)
              seg2[i][j]=0; //color negro
          }
}

//************************************
void PINTAR(int a)
{   int dib;
   int fil_a=0;
   int col_a=0;
   b_rat=0;
   ZONARETICULA();
   xmin_rat=dzona[2][0];
   ymin_rat=dzona[2][1];
   xmax_rat=dzona[2][2];
   ymax_rat=dzona[2][3];
   Rangohor();Rangover();
```

```
while (b_rat!=2)
{     GetCursor();
      dib=1;
      if (b_rat==1)
      {
            fil=int((y_rat-izy)*(float)nf/(dzona[2][3]-dzona[2][1]))+1;
            col=int((x_rat-izx)*(float)nc/(dzona[2][2]-dzona[2][0]))+1;
            if((fil>nf)||(fil==0))
                  dib=0;
            if((col>nc)||(col==0))
                  dib=0;
            if(dib)
            {
                  if ((fil!=fil_a)||(col!=col_a))
                  {  fil_a=fil;
                     col_a=col; HideCursor();
                     seg2[fil][col]=f; // a=1 simple
                  if (cuadro)
                  rectangulor((izx+lh*(col-1)),(izy+lv*(fil-1)),lh,lv,c,f);
                  else
                  rectangulor((izx+lh*(col-1)),(izy+lv*(fil-1)),lh,lv,f,f);
                  if (a>1)  //a=2  doble
                  {        seg2[nf-fil+1][nc-col+1]=f;
                           if (cuadro)
                           rectangulor((izx+lh*(nc-col)),(izy+lv*(nf-fil)),lh,lv,c,f);
                           else
                           rectangulor((izx+lh*(nc-col)),(izy+lv*(nf-fil)),lh,lv,f,f);
                  }
                  if (a>2)  //a=4 cuadruple
```

```
                    {       seg2[fil][nc-col+1]=f; seg2[nf-fil+1][col]=f;
                            if (cuadro)
                            { rectangulor((izx+lh*(col-1)),(izy+lv*(nf-fil)),lh,lv,c,f);
                            rectangulor((izx+lh*(nc-col)),(izy+lv*(fil-1)),lh,lv,c,f);
                            }
                            else
                            { rectangulor((izx+lh*(col-1)),(izy+lv*(nf-fil)),lh,lv,f,f);
                            rectangulor((izx+lh*(nc-col)),(izy+lv*(fil-1)),lh,lv,f,f);
                            }
                    }
                            ShowCursor();
                            _settextposition(30,1);
                            printf("Fila=%3d  Columna=%3d",fil,col);
                    }
                }
            }
        }
    ZONAGLOBAL();
    xmin_rat=dzona[0][0];
    ymin_rat=dzona[0][1];
    xmax_rat=dzona[0][2];
    ymax_rat=dzona[0][3];
    Rangohor();Rangover();
}

//************************************
void retiparcial(float a,float b,int filai,int columi,int filaf,int columf)
{   //rellenar trozo reticula
    int contfil,contcol;
```

```
   float izqx,izqy;
   short f1;
   HideCursor();
   if (cuadro) //pinta cuadro
    { for (contfil=filai;contfil<=filaf;contfil++)
        { for (contcol=columi;contcol<=columf;contcol++)
           {    izqx=a+lh*(contcol-1);
                izqy=b+lv*(contfil-1);
                f1=short(seg2[contfil][contcol]);
                if ((contfil<=nf)&&(contcol<=nc))
                        rectangulor(izqx,izqy,lh,lv,c,f1);
           }
        }
    }
   else // pinta cuadro del mismo color
   { for (contfil=filai;contfil<=filaf;contfil++)
        { for (contcol=columi;contcol<=columf;contcol++)
           {    izqx=a+lh*(contcol-1);
                izqy=b+lv*(contfil-1);
                f1=short(seg2[contfil][contcol]);
                if ((contfil<=nf)&&(contcol<=nc))
                        rectangulor(izqx,izqy,lh,lv,f1,f1);
           }
        }
    }
   ShowCursor();
}

//************************************
```

```
void  SUSTITUYECOLOR(void)
{
   int i,j;
   short fpa,fpn;
   _settextposition(30,1);printf("Elige color a sustituir");
   PuntoRaton();
   fpa=short(15-_getpixel_w(x_rat,y_rat));
   _settextposition(30,1);printf("Elige un nuevo color   ");
   PuntoRaton();
   fpn=short(15-_getpixel_w(x_rat,y_rat));
   for(i=1;i<=nf;i++)
    {     for(j=1;j<=nc;j++)
           {
             if(seg2[i][j]==fpa)      seg2[i][j]=fpn;
           }
    }
   reticular(izx,izy);
   _settextposition(30,1);printf("                    ");
}
```

```
//************************************
void  MEZCLACOLOR(void)
{
   int filapaleta;
   short fpa,fpn;
   unsigned long azul1,azul2,verd1,verd2,rojo1,rojo2,rojohexa,verdehexa,azulhexa;
   _settextposition(30,1);
   printf("Elige primer color a mezclar ");
   PuntoRaton();
   fpa=short(15-_getpixel_w(x_rat,y_rat));
   _settextposition(30,1);
   printf("Elige segundo color a mezclar");
   PuntoRaton();
   fpn=short(15-_getpixel_w(x_rat,y_rat));
   _settextposition(30,1);
   printf("Elige color a sustituir en paleta");
   ZONAPALETA();
   PuntoRaton();
    filapaleta=int ((y_rat-20)*16/(dzona[3][3]-dzona[3][1]))+1;
    if(filapaleta>16)
         filapaleta=16;
   azul1  = (color[fpa] >> 16) & 0x3F;
   verd1 = (color[fpa] >> 8) & 0x3F;
   rojo1  = color[fpa] & 0x3F;
   azul2  = (color[fpn] >> 16) & 0x3F;
   verd2 = (color[fpn] >> 8) & 0x3F;
   rojo2  = color[fpn] & 0x3F;
   azulhexa=long((azul1+azul2)/2);
   verdehexa=long((verd1+verd2)/2);
```

```
    rojohexa=long((rojo1+rojo2)/2);
    color[filapaleta-1]=azulhexa << 16 | verdehexa << 8 | rojohexa;
    _remappalette(short(filapaleta-1), color[short(filapaleta-1)]);
    _settextposition(30,1);printf("                                    ");
}
```

6.3.3. Funciones auxiliares

En este apartado se adjunta una lista de funciones auxiliares que son requeridas por las funciones consideradas como mas importantes, y que, al tratarse de llamadas repetitivas por varias de ellas, se ha considerado separarlas de las mismas para una mayor comodidad y rapidez de ejecución:

```
//************************************
void cabecera()// para la definición del modo de pantalla y dimensiones de la misma
{
    system ("cls");
  _setvideomode(_VRES16COLOR);
  _getvideoconfig(&cv);
  AC = cv.numypixels/cv.numtextrows;
  HC = cv.numxpixels/cv.numtextcols;
}

//************************************
```


```
void CONFIRMAR(void)      //Acción de confirmar datos o de cancelar
{
  int i,TECLA1;
                    //-----ALMACENA ZONA CONFIMACION/CANCELACION
  HideCursor();
```

```
AGET(320-90,430,320+90,450);lineabuff2=lineabuff;
for (i=0;i<100;i++)     //inicializa zonas
  { dzona[i][0] = 0; dzona[i][1] = 0;dzona[i][2] = 0;dzona[i][3] = 0;}

//     --IMPRESION MENSAJES DE CONFIMACION/CANCELACION
// Zona 1 de confirmación
i=1;dzona[i][0] = (31-1)*HC; dzona[i][1] = (28-1)*AC;
dzona[i][2] = dzona[i][0]+(HC*8-1);dzona[i][3] = dzona[i][1]+AC-1;
HideCursor();
BOTON(dzona[i][0],dzona[i][1],dzona[i][2],dzona[i][3]);
temp1="Aceptar ";RESALTAF(28,31,temp1,14,9);
// Zona 2 de cancelaci¢n
i=2;dzona[i][0] = (41-1)*HC; dzona[i][1] = (28-1)*AC;
dzona[i][2] = dzona[i][0]+(HC*8-1);dzona[i][3] = dzona[i][1]+AC-1;
HideCursor();
BOTON(dzona[i][0],dzona[i][1],dzona[i][2],dzona[i][3]);
temp1="Cancelar";RESALTAF(28,41,temp1,14,9);
ShowCursor();ShowCursor();
// -----FIN IMPRESION MENSAJES CONFIRM/CANCELACION
// Accion de Confirmar/Cancelar
numero_zona=0;TECLA1=NADA;
while ((TECLA1!=ENT)&(TECLA1!=ESC))
  {  TECLA1=MULTIPLENTRADA();
        switch (numero_zona)
           {case 0:
                if (TECLA1==ENT) {CANC_RUTI=0;break;}
                if (TECLA1==ESC) {CANC_RUTI=1;break;}
                printf("\a");TECLA1=NADA;break;
            case 1:CANC_RUTI=0;break;
```

```
                case 2:CANC_RUTI=1;break;
                default:printf("\a");TECLA1=NADA;break;
               }
             numero_zona=0;
      }//fin while (TECLA1!=ENT o ESC)

  //  /* ------RESTAURA ZONA CONFIMACION/CANCELACION
    lineabuff=lineabuff2;
    HideCursor();
    APUT(320-90,430,320+90,450);
    ShowCursor();ShowCursor();
    _ffree(lineabuff);_ffree(lineabuff2);
}
```

```
//************************************
```

Esta rutina de combrobación se utiliza cada vez que se introduce un nuevo nombre de fichero, comprobando que los caracteres sean validos y aceptados por el sistema operativo.


```
int nombrevalido(char *nombre)
{
   int cont;
    for(cont=0;cont<8;cont++)
     { char a;a=nombre[cont]; //caracteres validos NOMBRE FICH.
          if(!((a>47 & a<58)|(a>64 & a<91)|(a>96 & a<123)|(a==32)|(a==45)))
           {printf("\a");
           _settextposition(30,1);
           printf("Nombre de Fichero NO VALIDO       Pulse una tecla");getch();
           _settextposition(30,1);
            printf("                                        ");
           return(NO);
          }
     }
    return(SI);
}
//************************************
int cuenta_lineas(FILE *fich)
{   char linea_buf[81];
```

```
   int c=0;
   while(!feof(fich))
         { fgets(linea_buf,81,fich);
           ++c;
         }
   fseek(fich,0L,SEEK_SET);  //principio fichero
   return c;
}

//************************************
void corta(char *tira,int largo)
{   if(int(strlen(tira))>largo)
         *(tira+largo)='\0';
}

//************************************
void ordenatira(char *tiras[], int num)
{   char *temp;
   int tope, busca;
   for (tope = 1; tope < num; tope++)
         for (busca = tope; busca < num+1; busca++)
            if (strcmp(tiras[tope],tiras[busca]) > 0)
                  {   temp = tiras[tope];
                     tiras[tope] = tiras[busca];
                     tiras[busca] = temp;
                  }
   tiras[num+1] = '\0'; //para asegurar el £ltimo campo
}
```

```
//************************************
void NOHAYFICHEROS(void)
{
  AGET(6*HC,5*AC,50*HC,11*AC);buftemp1=lineabuff;
  NUMIN=1;NUMAX=1;
  RUTICAMPO(1,1,0,1,campo2);
 lineabuff=buftemp1;APUT(6*HC,5*AC,50*HC,11*AC);
_ffree(buftemp1);_ffree(lineabuff);
 }

//************************************
void LISTABLOQUE1(char *matriz[])  //para selecmenu
{   char bufertitulo[80];
   int i,j,px,py;
   HideCursor();_setcolor(0);
   _rectangle (_GFILLINTERIOR,short((X1-2)*HC+2),short((Y1-1)*AC),
                          short((xf-1)*HC-HC/2),short((yf-1)*AC-1));
   for (i=p;i<=u;i++)
         { j=((i+1-p) % nsimult)+((i+1-p)/nsimult)*nsimult;
           py=Y1-1+(j-((j-1)/NFIL)*NFIL);     //linea para imprimir
           px=X1+((j-1)/NFIL)*separa;          //columna para imprimir
           _settextposition(short(py),short(px));_settextcolor(14 );
           sprintf(bufertitulo, "%s", matriz[i]);_outtext( bufertitulo );
         }
   ShowCursor();
}

//************************************
void VIDEONORMAL1(void)
```

```
{   int j,sx,sy,P1,P2,P3,P4;
    j=((nselec_ant+1-p) % nsimult)+((nselec_ant+1-p)/nsimult)*nsimult;
    sx=X1+((j-1)/NFIL)*separa;          //posicion x primer caracter opcion
    sy=Y1-1+(j-((j-1)/NFIL)*NFIL);     //posicion y primer caracter opcion
    P1=sx*HC-HC-5;        P2=(sy-1)*AC-1;
    P3=(sx+lonuni)*HC-5;      P4=(sy*AC)-2;
    HideCursor();
    AGET(P1,P2,P3,P4);APUTR(P1,P2,P3,P4);_ffree(lineabuff); //video reverse
    ShowCursor();ShowCursor();
}

//************************************
void RESALTA1(void)
{   int j,sx,sy,P1,P2,P3,P4;
    j=((nselec+1-p) % nsimult)+((nselec+1-p)/nsimult)*nsimult;
    sx=X1+((j-1)/NFIL)*separa;                  //posicion x primer caracter opcion
    sy=Y1-1+(j-((j-1)/NFIL)*NFIL);              //posicion y primer caracter opcion
    P1=sx*HC-HC-5;      P2=(sy-1)*AC-1;
    P3=(sx+lonuni)*HC-5;   P4=(sy*AC)-2;
    HideCursor();
    AGET(P1,P2,P3,P4);APUTR(P1,P2,P3,P4);_ffree(lineabuff); //video reverse
    ShowCursor();ShowCursor();
}
```

```
//************************************
void RESALTAMENU(void)// localiza submenu y lo resalta
{   int menuant,nummenup;
    char titmenuant[15];
    nummenup=HALLASUBMENU();
    RESALTAF(1,colmenueleg,menueleg,15,4);
    menuant=nummenup;
    strcpy(titmenuant,menueleg);
    while ((numero_zona==1)&&(!b_rat))
     {    GetCursor();ZonaRaton();
          nummenup=HALLASUBMENU();
          if (nummenup!=menuant)
             { RESALTAF(1,colmenupri[menuant],titmenuant,15,0);
               RESALTAF(1,colmenueleg,menueleg,15,4);
               menuant=nummenup;
               strcpy(titmenuant,menueleg);
             }
     }
    RESALTAF(1,colmenupri[menuant],titmenuant,15,0);
}

//************************************
int HALLASUBMENU(void)
 {
    int nummenup, colraton;
    char cadenatemp[150];
    char *menuprin[15];
    char a;
    int i=0,k=0,ncar=0;
```

```
   cantsubmenus=0; //externa
   colraton=int((x_rat+HC/2)/HC);
   if (colraton<=1) colraton=2;
   strcpy(cadenatemp,menusub);
   a='x';
   while (a != '\0')       //Posicion de inicio de cada submenu
         {  a=cadenatemp[ncar++];
           if (a == ' ') colmenupri[++k]=ncar+1;
         }
   colmenupri[k+1]=81;
   i=1;menuprin[i]= strtok(cadenatemp," ");
   while(menuprin[i]!=NULL)
     {  menuprin[++i]= strtok(NULL," ");  }
   cantsubmenus=i-1;
   for(i=1;i<=cantsubmenus;i++)
    {   if ((colraton<colmenupri[i+1])&&(colraton>=colmenupri[i]))
             {     nummenup=i;colmenueleg=colmenupri[i];}
    }
   strcpy(menueleg,menuprin[nummenup]);
   return(nummenup);
}

//************************************
void SALIR(void)
{
   int TECLA1;
   HideCursor();
   // Zona 1 de confirmación
   dzona[1][0] = 13*HC;
```

```
dzona[1][1] = 17*AC;
dzona[1][2] = dzona[1][0]+(HC*17);
 dzona[1][3] = dzona[1][1]+3*AC;
BOTON(dzona[1][0],dzona[1][1],dzona[1][2],dzona[1][3]);
 temp1="GUARDAR CAMBIOS";RESALTAF(19,15,temp1,14,9);
// Zona 2 de cancelación
dzona[2][0] = 40*HC;
dzona[2][1] = 17*AC;
dzona[2][2] = dzona[2][0]+(HC*19);
dzona[2][3] = dzona[2][1]+3*AC;
HideCursor();
BOTON(dzona[2][0],dzona[2][1],dzona[2][2],dzona[2][3]);
temp1="SALIR SIN GUARDAR";RESALTAF(19,42,temp1,14,9);
ShowCursor();ShowCursor();
// Accion de Confirmar/Cancelar
numero_zona=0;TECLA1=NADA;
while ((TECLA1!=ENT)&(TECLA1!=ESC))
  {  TECLA1=MULTIPLENTRADA();
       switch (numero_zona)
          {case 0:
               if (TECLA1==ENT) {CANC_RUTI=0;break;}
               if (TECLA1==ESC) {CANC_RUTI=1;break;}
               printf("\a");TECLA1=NADA;break;
           case 1:CANC_RUTI=0;break;
           case 2:CANC_RUTI=1;break;
           default:printf("\a");TECLA1=NADA;break;
          }
       numero_zona=0;
  }//fin while (TECLA1!=ENT o ESC)
```

```
}

//************************************
void DELIMITA(void)
{   if (p < 1)
        {       p=1;
                u=p+nsimult-1;
        }
   if (u > NTOTAL)
        {       u=NTOTAL;
                p=u-nsimult+1;
                if (p<1)
                        p = 1;
        }
   if (nselec < p)
        nselec = p;
   if (nselec > u)
         nselec = u;
}

//************************************
void LISTABLOQUE(char *matriz[]) // para selecmatriz
{
 int i,j,px,py;
 HideCursor();_setcolor(0);
 _rectangle          (_GFILLINTERIOR,short((X1-2)*HC),short((Y1-1)*AC),short((xf-
1)*HC),
                                 short((yf-1)*AC));
```

```
 for (i=p;i<=u;i++)
        {   j=((i+1-p) % nsimult)+((i+1-p)/nsimult)*nsimult;
            py=Y1-1+(j-((j-1)/NFIL)*NFIL);   //linea para imprimir
            px=X1+((j-1)/NFIL)*separa;        //columna para imprimir
            _settextposition(short(py),short(px));
            printf("%s",matriz[i]);
        }
 ShowCursor();
}

//************************************
void VIDEONORMAL(void)
{
        int j,sx,sy,P1,P2,P3,P4;
        j=((nselec_ant+1-p) % nsimult)+((nselec_ant+1-p)/nsimult)*nsimult;
        sx=X1+((j-1)/NFIL)*separa;    //posicion x primer caracter opcion
        sy=Y1-1+(j-((j-1)/NFIL)*NFIL);        //posicion y primer caracter opcion
        P1=sx*HC-HC;         P2=(sy-1)*AC;
        P3=(sx+lonuni)*HC-9;  P4=(sy*AC)-2;
        HideCursor();
        AGET(P1,P2,P3,P4);
        APUTR(P1,P2,P3,P4);
        _ffree(lineabuff); //video reverse
        ShowCursor();ShowCursor();
}
```

```
//************************************
void RESALTA(void)
{       int j,sx,sy,P1,P2,P3,P4;
        j=((nselec+1-p) % nsimult)+((nselec+1-p)/nsimult)*nsimult;
        sx=X1+((j-1)/NFIL)*separa;    //posicion x primer caracter opcion
        sy=Y1-1+(j-((j-1)/NFIL)*NFIL);        //posicion y primer caracter opcion
        P1=sx*HC-HC;         P2=(sy-1)*AC;
        P3=(sx+lonuni)*HC-9;  P4=(sy*AC)-2;
        HideCursor();
        AGET(P1,P2,P3,P4);
        APUTR(P1,P2,P3,P4);
        _ffree(lineabuff); //video reverse
        ShowCursor();ShowCursor();

}

//************************************
int MULTIPLENTRADA(void)      //devuelve TECLA
{   int tec,car,car1,TECLA;
   AbsorbePulsacion();
   b_rat=0;tec=0;nselec_ant=nselec;
   while ((!tec) & (!b_rat))
        {GetCursor();tec=kbhit();kbhit();}
   if (!b_rat)
        {
          car=getch();car1=kbhit();
               // comprobacion tecla doble
          if (car1==255)
               {  while ((!tec))
```

```
                {tec=kbhit();}
            car=getch();kbhit();
             car=100*car;
        }
        // asignacion de valor a las diferentes teclas
    if ((car>=33) && (car<=168))
        return car;
        switch (car)
          {
                case 8:TECLA=RET;break;
                case 9:TECLA=TAB;break;
                case 13:TECLA=ENT;break;
                case 27:TECLA=ESC;break;
                case 32:TECLA=ESP;break;
                case 7100:TECLA=INI;break;
                case 7200:TECLA=ARR;break;
                case 7300:TECLA=PGU;break;
                case 7500:TECLA=IZQ;break;
                case 7700:TECLA=DER;break;
                case 7900:TECLA=FIN;break;
                case 8000:TECLA=ABA;break;
                case 8100:TECLA=PGD;break;
                case 8200:TECLA=INS;break;
                case 8300:TECLA=DEL;break;
                default:TECLA=NADA;break;
          }
     return TECLA;
    }    // fin de if (!b_rat)
// Entrada por raton (botones 1,2,3)
```

```
    ZonaRaton();
    AbsorbePulsacion();
  if ((!numero_zona) || (b_rat==2))
    { TECLA = ESC;
        return TECLA;
    }
   TECLA = ENT;
   return TECLA;
}


//************************************
void ORDVERT(int x1,int y1,int x2,int y2) //Devuelve xizq,ysup,xder,yinf
{
   xy1=_getphyscoord(short(x1),short(y1));
   xy2=_getphyscoord(short(x2),short(y2));
   xizq=(xy1.xcoord>xy2.xcoord) ? xy2.xcoord : xy1.xcoord;
   xder=(xy1.xcoord>xy2.xcoord) ? xy1.xcoord : xy2.xcoord;
   ysup=(xy1.ycoord>xy2.ycoord) ? xy2.ycoord : xy1.ycoord;
   yinf=(xy1.ycoord>xy2.ycoord) ? xy1.ycoord : xy2.ycoord;
}

//************************************
void AGET(int x1,int y1,int x2,int y2)
{
   ORDVERT(x1,y1,x2,y2);
   HideCursor();
   t_imagen = (size_t)_imagesize(short(xizq),short(ysup),short(xder),short(yinf));
   lineabuff= (char far *)_fmalloc(t_imagen );
```

```
   if (lineabuff ==(char far *)NULL)
         exit(!_setvideomode(_DEFAULTMODE));
   _getimage(short(xizq),short(ysup),short(xder),short(yinf),lineabuff);
   ShowCursor();
}

//************************************
void APUT(int x1,int y1,int x2,int y2)
{
   ORDVERT(x1,y1,x2,y2); HideCursor();
   _putimage(short(xizq),short(ysup),lineabuff,_GPSET);
   ShowCursor();
}
//************************************
void APUTR(int x1,int y1,int x2,int y2)
{
   ORDVERT(x1,y1,x2,y2); HideCursor();
   _putimage(short(xizq),short(ysup),
  lineabuff,_GPSET);
   _putimage(short(xizq),short(ysup),
   lineabuff,_GPRESET);
   ShowCursor();
}

//************************************
void APUTXOR(int x1,int y1,int x2,int y2)
{
   ORDVERT(x1,y1,x2,y2); HideCursor();
   _putimage(short(xizq),short(ysup),lineabuff,_GXOR);
```

```
   ShowCursor();
}

//************************************
void Flecha_abajo(int a,int b)
{   _rectangle (_GFILLINTERIOR,short(a),short(b),short(a+1),short(b-7));
   _rectangle (_GFILLINTERIOR,short(a-1),short(b-1),short(a+2),short(b-1));
   _rectangle (_GFILLINTERIOR,short(a-2),short(b-2),short(a+3),short(b-2));
   _rectangle (_GFILLINTERIOR,short(a-3),short(b-3),short(a+4),short(b-3));
}

//************************************
void Flecha_arriba(int a,int b)
{   _rectangle (_GFILLINTERIOR,short(a),short(b),short(a+1),short(b+7));
   _rectangle (_GFILLINTERIOR,short(a-1),short(b+1),short(a+2),short(b+1));
   _rectangle (_GFILLINTERIOR,short(a-2),short(b+2),short(a+3),short(b+2));
   _rectangle (_GFILLINTERIOR,short(a-3),short(b+3),short(a+4),short(b+3));
}

//************************************
void Flecha_izquierda(int a,int b)
{   _rectangle (_GFILLINTERIOR,short(a),short(b),short(a+7),short(b-1));
   _rectangle (_GFILLINTERIOR,short(a+1),short(b+1),short(a+1),short(b-2));
   _rectangle (_GFILLINTERIOR,short(a+2),short(b+2),short(a+2),short(b-3));
   _rectangle (_GFILLINTERIOR,short(a+3),short(b+3),short(a+3),short(b-4));
}

//************************************
void Flecha_derecha(int a,int b)
```

```
{   _rectangle (_GFILLINTERIOR,short(a),short(b),short(a-7),short(b-1));
    _rectangle (_GFILLINTERIOR,short(a-1),short(b+1),short(a-1),short(b-2));
    _rectangle (_GFILLINTERIOR,short(a-2),short(b+2),short(a-2),short(b-3));
    _rectangle (_GFILLINTERIOR,short(a-3),short(b+3),short(a-3),short(b-4));
}

//************************************
// resalta caracter situado en la posicion
void RESALTAC(int v,int h,char c,int color,int fondo)
{   int P1,P2,P3,P4;
    char bufer[80];
    P1=(h-1)*HC;P2=(v-1)*AC;
    P3=P1+(HC)-1;P4=P2+AC-1;
    HideCursor();
    color=(color+fondo)%16;
    _settextposition(short(v),short(h));
    _settextcolor(short(color));
    sprintf(bufer, "%c", c );_outtext( bufer );
    AGET(P1,P2,P3,P4);
    _setcolor(short(fondo));
    _rectangle (_GFILLINTERIOR,short(P1),short(P2),short(P3),short(P4));
    APUTXOR(P1,P2,P3,P4);
    _ffree(lineabuff); //video reverse
    ShowCursor();ShowCursor();
}

//************************************
// resalta frase situada en la posicion
void RESALTAF(int v,int h,char *f,int color,int fondo)
```

```
{   int P1,P2,P3,P4;
    char bufer[80];
    P1=(h-1)*HC;P2=(v-1)*AC;
     P3=P1+(HC*strlen(f))-1;P4=P2+AC-1;
    HideCursor();
     color=(color+fondo)%16;
    _settextposition(short(v),short(h));
    _settextcolor(short(color));
    sprintf(bufer, "%s", f );_outtext( bufer );
    AGET(P1,P2,P3,P4);
    _setcolor(short(fondo));
    _rectangle (_GFILLINTERIOR,short(P1),short(P2),short(P3),short(P4));
    APUTXOR(P1,P2,P3,P4);
    _ffree(lineabuff);
    ShowCursor();ShowCursor();
}

//************************************
void BOTON(int x1,int y1,int x2,int y2)
{   ORDVERT(x1,y1,x2,y2);
 _setcolor(7);
_rectangle (_GFILLINTERIOR,short(xizq-2),short(ysup-2),short(xder+2),
                                    short(yinf+2));
 _setcolor(1);
_rectangle (_GFILLINTERIOR,short(xizq),short(ysup),short(xder+2),short(yinf+2));
_setcolor(9);
_rectangle (_GFILLINTERIOR,short(xizq),short(ysup),short(xder),short(yinf));
}
```

```
//************************************
char *AlineaDer(char *f)       // ALINEA A LA DERECHA
{   int POSI,i,j,longi;
    char *b;
    b=f;
    longi=int(strlen(f));
    for (POSI=longi;POSI>=1;POSI--)          //calcula POSI
          {if (f[POSI-1]!=' ') break;}
    for (i=POSI;i>=0;i--)
          { j=i+longi-POSI;b[j]=f[i];}
    for (i=(longi-POSI-1);i>=0;i--)
          { b[i]=' ';}
    b[longi]='\0';
    return (b);
}

//************************************
char *AlineaIzq(char *f)       // ALINEA A LA IZQUIERDA
{   int POSI,i,longi;
    char *b;b=f;longi=int(strlen(f));
    for (POSI=0;POSI<=(longi-1);POSI++)      //calcula POSI
          {if (f[POSI]!=' ') break;}
    for (i=POSI;i<=(longi-1);i++)
          {b[i-POSI]=char(f[i]);}
    for (i=(longi-POSI);i<=longi-1;i++)
          { b[i]=char(' ');}
    return (b);
}
```

```
//************************************
char *Abreesp(char *f,int POSI) // Abre un espacio para insertar un caracter
{   int longi,i;
   longi=int(strlen(f));
   if (longi==POSI){return (f);}
   for (i=(longi-1);i>=POSI;i--)
         {f[i]=f[i-1];}
   f[POSI]=' ';
   return (f);
}

//************************************
char *Cierraesp(char *f,int POSI) //Anula un caracter
{   int longi,i;
   longi=int(strlen(f));
   if (longi==POSI){f[longi-1]=' ';return (f);}
   for (i=(POSI);i<=(longi-2);i++)
         {f[i]=f[i+1];}
   f[longi-1]=' ';
   return (f);
}

//************************************
void IGUALAMATSEG(void)   // iguala matrices
{   int i,j;
   mat2[1][0]=seg2[1][0];
   mat2[0][1]=seg2[0][1];
   for (i=1;i<=nf;i++)
    {     for (j=1;j<=nc;j++)
```

```
            mat2[i][j]=seg2[i][j];
    }
}

//************************************
void RECUPERAMAT(void) // recupera matriz seguridad mat2
{  int i,j;
   nf=mat2[1][0];
   nc=mat2[0][1];
   seg2[1][0]=mat2[1][0];
   seg2[0][1]=mat2[0][1];
   for (i=1;i<=nf;i++)
    {     for (j=1;j<=nc;j++)
            seg2[i][j]=mat2[i][j];
    }
}
```

6.3.4. Programa principal

El programa principal de diseño es el que se lista completo a continuación, contiene comentarios de interés en algun apartado y algunas de las pantallas decisorias que aparecen en el transcurso del programa.

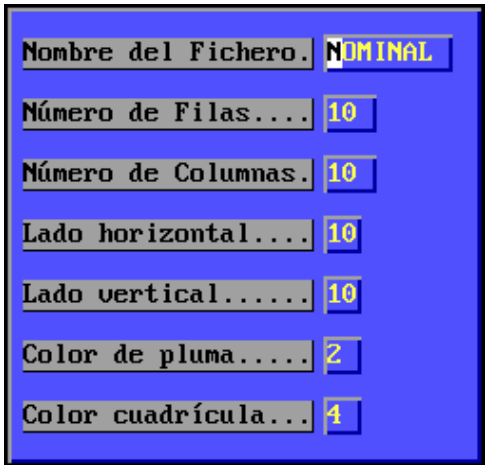
En este texto no creo de interes mostrar todas las pantallas que están previstas, pues dependerá de la opción elegida, pero sirven como muestra de la interface creada y las posibilidades de ampliación a nuevos comandos.

```
// 0PRINC.CPP  fichero principal
// juntar con (0ficher, 0menus, 0movim, 0paleta, 0raton, 0rutina y 0textil)

#include "io.h"
#include "conio.h"
#include "graph.h"
#include "malloc.h"
#include "stdio.h"
#include "stdlib.h"
#include "string.h"
#include "0includ.h"

struct xycoord xy1,xy2; // de graph.h
struct videoconfig cv; // de graph.h
char *temp1; //para string temporales
char far *lineabuff; // para almazenar zona pantalla
char far *lineabuff1;   //                  "
char far *buftemp1;
int AC,HC;                          //alto y ancho caracter en esta resolucion
```

```
int xizq,xder,ysup,yinf;        // valores devueltos por la funcion ORDVERT
int izx,izy,lh,lv,nc,nf;        // valores para reticula
int cuadro;
int ra_ton;                     // raton cargado (-1) o no cargado (0)
int x_rat, y_rat, b_rat;        // coordenadas raton y boton pulasdo
int x_dev,y_dev;                // coordenadas devueltas del raton
int xmin_rat, ymin_rat;         // limites coordenadas horizontales
int xmax_rat, ymax_rat;         // limites coordenadas verticales
int cboton;                     // reconocer cuantos botones
int bot_rat;                    // boton 0,1,2
int estado_boton;               // estado boton 0,1,2,3,4
int num_botpulsa;               // numero veces pulsado
int ult_xratpul, ult_yratpul;   // ultimas coordenadas pulsadas
int h_xrat, h_yrat;             // punto caliente matriz del raton
int sen_xrat, sen_yrat;         // sensibilidad raton 0-255 0 muy sensible
int text_rat;                   // cursor texto: 1= hardware; 0= software
int der_rat,aba_rat;            // sentido ultimo movimiento raton
int cantidad_zonas;             // cantidad de zonas de movimiento raton
int numero_zona;                // Indice numeracion de zonas
int dzona[100][4];              // coordenadas limites de zona
int avent,bvent,cvent,dvent;    // coordenadas ventana seleccionada
int numenu;
int submenu, opcion,cantsubmenus;
char *menu[1][10];
char *menusub;
char *dir[100];
char *menurat[74];
int X1,Y1,NFIL,NCOL,BORRAR,CF,indice;
int lonuni,nz,nsimult,separa,p,u,xf,yf;
```

```
int nselec,nselec_ant;
int NUMIN,NUMAX,CANC_RUTI;
int mat2[FILASMAX][COLUMMAX];
int seg2[FILASMAX][COLUMMAX];
int fil,col,fil_a,col_a,filpal;
short c,f;          //color cuadricula y fondo reticula
int NTOTAL,BOTRATON,ZONAANT;
int cantficheros;
long color[16]={_BLACK, _BLUE, _GREEN, _CYAN, _RED, _MAGENTA,
                _BROWN, _WHITE, _GRAY,   _LIGHTBLUE, _LIGHTGREEN,
                _LIGHTCYAN, _LIGHTRED, _LIGHTMAGENTA,
                _LIGHTYELLOW, _BRIGHTWHITE };
        // entradas{int vi,hi,li;char *ni;int ti;char *var;}
        //ti => C=1,ND=2(alinea derecha),NI=3(alinea izquierda)
extern struct entradas campo1[]=
        { 0,0,0," ",C," ",
          4 ,30,8,"Nombre del Fichero.",C,"NOMINAL ",
          6,30,3, "Número de Filas....",NI,"10 ",
          8,30,3, "Número de Columnas.",NI,"10 ",
          10,30,2,"Lado horizontal....",NI,"10",
          12,30,2,"Lado vertical.....",NI,"10",
          14,30,2,"Color de pluma.....",NI,"2 ",
          16,30,2,"Color cuadricula...",NI,"4 ",
        };
```

// Con la introducción de estos parámetros en la estructura **entradas campo1[]** vemos el aspecto que presenta cuando llamamos a la funcion RUTICAMPO (). Los valores pasados en la rutina podrán modificarse de forma interactiva y ser almacenados de nuevo, afectando a la visualización de la retícula y a la base de datos que la define.



// En el programa se han asignado dos campos más a la estructura **entradas** que tomará otro aspecto cuando llamemos a la misma función anterior de RUTICAMPO.

```
extern struct entradas campo2[]=
        { 0,0,0," ",C," ",
          8,30,11,"RETICULA",C,"Inexistente",
          8,30,8, "Nombre Actual.....",C,"SINTITUL",
          11,30,8,"Nuevo Nombre.....",C,"       ",
          8,30,8, "Se borrará la Ret¡cula ",C,"       ",
          8,30,9, "RETICULA",C,"Existente",
          14,30,3," Añadir Filas.....",NI,"1  ",
          16,30,3," Añadir Columnas...",NI,"1  ",
        };
```

```
extern struct entradas campo3[]=
        { 0,0,0," ",C," ",
          8,30,11,"PALETA",C,"Inexistente",
          8,30,8, "Nombre Actual.....",C,"SINTITUL",
          11,30,8,"Nuevo Nombre.....",C,"       ",
          8,30,8, "Se borrará la Paleta ",C,"       ",
          8,30,9, "PALETA",C,"Existente",
        };

//********AQUI SE INICIA EL PROGRAMA PRINCIPAL
int main (void)
{
char *nuevaret, *nuevapal;
int i=0,j=0,k=0;
cabecera();
InitMouse();
GrafCursor();
 //------------valores por defecto en la reticula
 nf=10;nc=10;lh=10;lv=10;f=2;c=4;cuadro=1;
 nuevaret="SINTITUL.TEX";
 nuevapal="SINTITUL.PAL";
 mat2[1][0]=nf;
 mat2[0][1]=nc;
 seg2[1][0]=nf;
seg2[0][1]=nc;
 izx=2;izy=17;//posicion vertice superior izquierda reticula
 paleta(615,20,20,20,16,1,1);
 calculacolor();
 cajaajustes();
```

```
 numenu=1; // Primer menu y unico en este caso
 submenu=0;// Inicialización
 menu[1][0]= " ARCHIVO RETICULA PALETA COLORES PANTALLA
              TRANSFORMAR UTILES AYUDA";
 HideCursor();
_setcolor(0);_rectangle (_GFILLINTERIOR,0,0,639,short(AC));
 RESALTAF(1,1,menu[1][0],15,0);   //---imprime menú en blanco intenso
 ShowCursor();
 strcpy(menusub,menu[1][0]);//nuevo

INICIO:
menu[1][0]= " ARCHIVO RETICULA PALETA COLORES PANTALLA
              TRANSFORMAR UTILES AYUDA";
menu[1][1]= "Nuevo...,Abrir...,Guardar,Guardar Como...,Renombrar...,
              Borrar...,SALIR";
menu[1][2]= "Deshacer,Nueva...,Editar...,Ampliar...,
              Mover,Mostrar,Limpiar,Ocultar,Borrar";
menu[1][3]= "Cargar...,Grabar,Grabar Como...,Renombrar,Borrar...,
              Mostrar,Ocultar";
menu[1][4]= "Pintar,Pintar doble,Pintar Cuadruple,Sustituir,Mezclar";
menu[1][5]= "Simular Tejido,Simular Curso,Simulación Completa,
              Limpia Tejido,Limpia Zona,Limpia Pantalla";
menu[1][6]= "Traslación...,Simetrias...,Giros...,Copias...,Rellenos...";
menu[1][7]= "Oculta Menu,Sin Cuadricula,Con Cuadricula,
              Mostrar Ajuste,Ocultar Ajuste,Utiles Raton...";
menu[1][8]= "Archivo,Retícula,Paleta,Colores,Pantalla,Transformar,Utiles,Ratón";
```

// A continuación se muestra el aspecto de la pantalla en las diferentes posibilidades que nos ofrece el menú principal y los submenús desplegables que cuelgan de cada uno de ellos cuando llamemos a la función selecmenu().


```
//  DEFINICION DE ZONAS DE RATON
VACIAZONAS();      // 100 zonas definidas para control del raton
ZONAGLOBAL();      // Zona por defecto     (0) (toda la pantalla)
ZONAMENU();        // Zona menú principal        (1)
ZONARETICULA(); // Zona cuadro reticula        (2)
ZONAPALETA();      // Zona paleta          (3)
ZONAAJUSTE();      // Zona ajuste color    (4)
```

```
cantidad_zonas=4;
ZONAANT=0;b_rat=0;
ShowCursor();
while (!b_rat)   //exploración de pantalla con el ratón
   {    GetCursor();ZonaRaton();
        if(!(numero_zona==ZONAANT))
         { switch (numero_zona)
          {
          case 1:         //ZONA MENUS
                RESALTAMENU();  numero_zona=1;
                break;
          case 2:          //ZONA RETICULA
                fil_a=0;col_a=0;
                while ((numero_zona==2)&&(!b_rat))
                 { GetCursor();ZonaRaton();
                   fil=int((y_rat-izy)*(float)nf/(dzona[2][3]-dzona[2][1]))+1;
                   if(fil>nf)
                         fil=nf;
                   col=int((x_rat-izx)*(float)nc/(dzona[2][2]-dzona[2][0]))+1;
                   if(col>nc)
                          col=nc;
                   if ((fil!=fil_a)||(col!=col_a))
                    {     _settextposition(30,1);
                          printf("Fila=%3d  Columna=%3d",fil,col);
                          fil_a=fil;col_a=col;
                    }
                 }
                _settextposition(30,1);printf("                    ");
                numero_zona=2;
```

```
        break;
case 3:         //ZONA PALETA
        fil_a=0;
        while ((numero_zona==3)&&(!b_rat))
         { GetCursor();ZonaRaton();
           filpal=int ((y_rat-20)*16/(dzona[3][3]-dzona[3][1]))+1;
           if(filpal>16)
                 filpal=16;
           if ((filpal!=fil_a))
            {     _settextposition(30,30);
                  printf(" Color=%d ",filpal-1);
                  fil_a=filpal;
             }
         }
        _settextposition(30,30);
        printf("               ");
        numero_zona=3;
        break;
case 4:         //ZONA AJUSTE COLOR
        fil_a=0;col_a=0;
        while ((numero_zona==4)&&(!b_rat))
         { GetCursor();ZonaRaton();
           if ((y_rat!=fil_a)||(x_rat!=col_a))
                 fil_a=y_rat;col_a=x_rat;
         }
        numero_zona=4;
        break;
default:
        numero_zona=0;
```

```
                break;
        }   // fin switch (numero_zona)
        ZONAANT=numero_zona;
        }    //fin de if(!(numero_zona==ZONAANT))
    }        //fin while (!b_rat)

//*****DECISIONES EN ZONA DE RETICULA*****
if((numero_zona==2)&&(seg2[fil][col]!=f))
  {     seg2[fil][col]=f;
        mat2[fil][col]=f;
        HideCursor();
        if (cuadro)
                rectangulor((izx+lh*(col-1)),(izy+lv*(fil-1)),lh,lv,c,f);
        else
                rectangulor((izx+lh*(col-1)),(izy+lv*(fil-1)),lh,lv,f,f);
        ShowCursor();
  }

//*****DECISIONES EN ZONA DE PALETA*****
if((numero_zona==3)&&(f!=filpal-1))
  {     f=(short)(filpal-1);
        calculacolor();
        cajaajustes();
  }
```

```
//*****DECISIONES EN ZONA DE AJUSTE*****
if(numero_zona==4)       // para ajustar color
{        calculacolor();
         if ((y_rat>=385)&&(y_rat<=460))
         {
                  if ((561<x_rat)&&(x_rat<586))
                           ajusterojo();
                  if ((586<x_rat)&&(x_rat<611))
                           ajusteverde();
                  if ((611<x_rat)&&(x_rat<635))
                           ajusteazul();
                  calculacolor(); cajaajustes();
         }
 }

//*****DECISIONES A TRAVES DE MENU PRINCIPAL*****
if(numero_zona==1)
  { submenu=HALLASUBMENU();
    opcion=1;
    selecmenu(menu[numenu]); //devuelve submenu y opcion
    opcion=submenu*10+opcion;
    switch (submenu)
    {
    case 1:       //*****FICHEROS*****
                  // "Nuevo...,Abrir...,Guardar,Guardar Como...,Renombrar...,
                  //  Borrar...,SALIR";
     switch (opcion)
         {
         case 11:         // Nuevo...
```

```
BORRARETICULA();
NUMIN=1;NUMAX=7;
RUTICAMPO(1,1,1,1,campo1);
if((!CANC_RUTI)&(nombrevalido(campo1[1].var)))
{     sprintf(nuevaret,campo1[1].var);
      campo2[2].var=campo1[1].var;
      strcat(nuevaret,".TEX");
      nf=atoi(campo1[2].var);
      nc=atoi(campo1[3].var);
      seg2[1][0]=nf;
      seg2[0][1]=nc;
      lh=atoi(campo1[4].var);
      lv=atoi(campo1[5].var);
      f=short(atoi(campo1[6].var));
      c=short(atoi(campo1[7].var));
      int i,j;  //asignar valores a la matriz
      for(i=1;i<=nf;i++)
       { for(j=1;j<=nc;j++)
           {    mat2[i][j]=f;
                seg2[i][j]=f;
           }
       }
      izx=2;izy=17;  //posicion vertice superior izquierda
      reticular(izx,izy);
      if(!_access(nuevaret,0))        //si existe el fichero...
       { printf("\a");NUMIN=5;NUMAX=5;
         RUTICAMPO(1,1,0,5,campo2);
         if(!CANC_RUTI)
               grabareticula(nuevaret);
```

```
            }
           else
            grabareticula(nuevaret);
    }
    break;
case 12:          //Abrir...
    BORRARETICULA();
    cantficheros=ficheros("TEX");
    if (!cantficheros)
          {NOHAYFICHEROS();break;}
    NFIL=4;NCOL=5;X1=3;Y1=20;BORRAR=SI;CF=0;indice=0;
    selecmatriz(&indice,dir);
    if ((indice!=0)&&(indice<=cantficheros))
     {     corta(dir[indice],8);
           sprintf(campo2[2].var,dir[indice]);
           strcat(dir[indice],".TEX");
           nuevaret=dir[indice];
           if(cargareticula(nuevaret))
                  reticular(izx,izy);
     }
    break;
case 13:   //Guardar con el mismo nombre
    grabareticula(nuevaret);
    break;
case 14:   //Guardar como...
    //almacenar zona
    AGET(6*HC,5*AC,50*HC,11*AC);
    buftemp1=lineabuff;
    NUMIN=2;NUMAX=2;
```

```
    campo2[3].var=campo2[2].var;
    RUTICAMPO(0,0,0,2,campo2);
    NUMIN=3;NUMAX=3;
    RUTICAMPO(1,1,1,3,campo2);
    lineabuff=buftemp1;APUT(6*HC,5*AC,50*HC,11*AC);
    _ffree(buftemp1);_ffree(lineabuff);
    if((!CANC_RUTI)&(nombrevalido(campo2[3].var)))
     {    sprintf(nuevaret,campo2[3].var);
          strcat(nuevaret,".TEX");
          if( !_access( nuevaret, 0 ) ) //comprueba nombre existente
             {     AGET(6*HC,5*AC,50*HC,11*AC);buftemp1=lineabuff;
                   NUMIN=5;NUMAX=5;
                   RUTICAMPO(1,1,0,5,campo2);
                   lineabuff=buftemp1;APUT(6*HC,5*AC,50*HC,11*AC);
                   _ffree(buftemp1);_ffree(lineabuff);
                    if(!CANC_RUTI)
                            grabareticula(nuevaret);
                    break;
             }
          grabareticula(nuevaret);
     }
    break;
case 15:          //Renombrar...
    cantficheros=ficheros("TEX");
    if (!cantficheros)
          {NOHAYFICHEROS();break;}
    NFIL=4;NCOL=5;X1=3;Y1=20;BORRAR=SI;CF=0;indice=0;
    selecmatriz(&indice,dir);
    if ((indice!=0)&&(indice<=cantficheros))
```

```
{     corta(dir[indice],8);
      sprintf(campo2[2].var,dir[indice]);
      strcat(dir[indice],".TEX");
      // ALMACENAMIENTO ZONA
      AGET(6*HC,5*AC,50*HC,11*AC);buftemp1=lineabuff;
      NUMIN=2;NUMAX=2;
      campo2[3].var=campo2[2].var;
      RUTICAMPO(0,0,0,2,campo2);
      NUMIN=3;NUMAX=3;
      RUTICAMPO(1,1,1,3,campo2);
      lineabuff=buftemp1;
      APUT(6*HC,5*AC,50*HC,11*AC);
      _ffree(buftemp1);_ffree(lineabuff);
      if((!CANC_RUTI)&(nombrevalido(campo2[3].var)))
        {   sprintf(nuevaret,campo2[3].var);strcat(nuevaret,".TEX");
            if( !_access( nuevaret, 0 ) )
              {     AGET(6*HC,5*AC,50*HC,11*AC);buftemp1=lineabuff;
                    NUMIN=5;NUMAX=5;
                    RUTICAMPO(1,1,0,5,campo2);
                    lineabuff=buftemp1;APUT(6*HC,5*AC,50*HC,11*AC);
                    _ffree(buftemp1);_ffree(lineabuff);
               }
            else
              { rename(dir[indice],nuevaret); }
        }
    }
  break;
case 16:  //Borrar...
  cantficheros=ficheros("TEX");
```

```
        if (!cantficheros)
              {NOHAYFICHEROS();break;}
        NFIL=4;NCOL=5;X1=3;Y1=20;BORRAR=SI;CF=0;indice=0;
        selecmatriz(&indice,dir);
        if ((indice!=0)&&(indice<=cantficheros))
          {   corta(dir[indice],8);
              sprintf(campo2[4].var,dir[indice]);
              strcat(dir[indice],".TEX");
               // ALMACENAMIENTO ZONA
               AGET(6*HC,5*AC,50*HC,11*AC);buftemp1=lineabuff;
               NUMIN=4;NUMAX=4;
               RUTICAMPO(1,1,0,4,campo2);
               lineabuff=buftemp1;APUT(6*HC,5*AC,50*HC,11*AC);
               _ffree(buftemp1);_ffree(lineabuff);
               if(!CANC_RUTI)
                     {remove(dir[indice]);}
          }
        break;
      case 17:   //SALIR
        SALIR();
        if(!CANC_RUTI)
              grabareticula(nuevaret);
        _setvideomode(_DEFAULTMODE);
        exit(1); //Salir del programa
      default:
        {}
      }
  break; // fin case 1 (ARCHIVOS)
```

```
   case 2:        //*****RETICULA*****
                  //"Deshacer,Nueva...,Editar...,Ampliar...,
                  //Mover,Mostrar,Limpiar,Ocultar,Borrar";
switch (opcion)
        {
        case 21: // Deshacer y ver reticula anterior
           HideCursor();
           _setviewport(short(izx),short(izy),short(izx+(nc*lh)),short(izy+(nf*lv)));
           _clearscreen(_GVIEWPORT);
           _setviewport(0,0,cv.numxpixels,cv.numypixels);
           RECUPERAMAT();
           reticular(izx,izy);
           break;
        case 22: // Nueva...
           NUMIN=2;NUMAX=7;
           RUTICAMPO(1,1,1,2,campo1);
           if(!CANC_RUTI)
           {      BORRARETICULA();
                  nf=atoi(campo1[2].var);
                  nc=atoi(campo1[3].var);
                  seg2[1][0]=nf;
                  seg2[0][1]=nc;
                  lh=atoi(campo1[4].var);
                  lv=atoi(campo1[5].var);
                  f=short(atoi(campo1[6].var));
                  c=short(atoi(campo1[7].var));
                  for (i=1;i<=nf;i++) // llenar matriz actual
                  { for (j=1;j<=nc;j++)
                      {     seg2[i][j]=f;
```

```
                    mat2[i][j]=f;
                }
            }
        reticular(izx,izy);
    }
    break;
case 23: // editar...
    NUMIN=2;NUMAX=7;
    _itoa(nf, campo1[2].var, 10 );
    _itoa(nc, campo1[3].var, 10 );
    _itoa(lh, campo1[4].var, 10 );
    _itoa(lv, campo1[5].var, 10 );
    _itoa(f, campo1[6].var, 10 );
    _itoa(c, campo1[7].var, 10 );
    RUTICAMPO(1,1,1,2,campo1);
    if(!CANC_RUTI)
    {   nf=atoi(campo1[2].var);
        nc=atoi(campo1[3].var);
        lh=atoi(campo1[4].var);
        lv=atoi(campo1[5].var);
        f=short(atoi(campo1[6].var));
        c=short(atoi(campo1[7].var));
        reticular(izx,izy);
    }
    break;
case 24:   // Ampliar...
```

// Aqui se muestra el aspecto de la pantalla cuando queremos modificar el numero de filas o de columnas de nuestra reticula.


```
    NUMIN=6;NUMAX=7;
    RUTICAMPO(1,1,1,6,campo2);
    if(!CANC_RUTI)
    {     nf=nf+(atoi(campo2[6].var));
          nc=nc+(atoi(campo2[7].var));
          reticular(izx,izy);
    }
    break;
case 25:  // Mover reticula a nueva posicion
    _settextposition(30,1);
    printf("Indicar nuevo punto con el rat¢n");
    PuntoRaton();
    _settextposition(30,1);
    printf("                              ");
    izx=x_rat;   izy=y_rat;
    ZONARETICULA();
    reticular(izx,izy);
    break;
case 26: // Mostrar
    reticular(izx,izy);
    break;
case 27:    //limpia reticula
```

```
            BORRARETICULA();
            reticular(izx,izy);
            break;
         case 28:    //Ocultar reticula
            HideCursor();
            _setviewport(short(izx),short(izy),short(izx+(nc*lh)),short(izy+(nf*lv)));
            _clearscreen(_GVIEWPORT);
            _setviewport(0,0,cv.numxpixels,cv.numypixels);
            break;
         case 29:   // borrar reticula
            BORRARETICULA();
            break;
         default:
            {}
         }
    break; // fin case 2 (RETICULA)

case 3:          //*****PALETA*****
                 //"Cargar...,Grabar,Grabar Como...,Renombrar,Borrar...,
                 // Mostrar,Ocultar";
         switch (opcion)
         {
         case 31: // Cargar  paleta...
            cantficheros=ficheros("PAL");
```

// Las diversas paletas se almacenan en el directorio a modo de ficheros normales con la extensión .PAL mostrándose en la llamada con el siguiente aspecto.


```
    if (!cantficheros)
          {NOHAYFICHEROS();break;}
    NFIL=4;NCOL=5;X1=3;Y1=20;BORRAR=SI;CF=0;indice=0;
    selecmatriz(&indice,dir);
    if ((indice!=0)&&(indice<=cantficheros))
      {     corta(dir[indice],8);
            sprintf(campo3[2].var,dir[indice]);
            strcat(dir[indice],".PAL");
            nuevapal=dir[indice];
            if(cargapaleta(nuevapal))
              {     paleta(615,20,20,20,16,1,1);
                    calculacolor();
                    cajaajustes();
              }
      }
    break;
  case 32: // grabar paleta
    grabapaleta(nuevapal);
    break;
```

```
case 33: // grabar como...
   //almacenar zona
   AGET(6*HC,5*AC,50*HC,11*AC);buftemp1=lineabuff;
   NUMIN=2;NUMAX=2;
   campo3[3].var=campo3[2].var;
   RUTICAMPO(0,0,0,2,campo3);
   NUMIN=3;NUMAX=3;
   RUTICAMPO(1,1,1,3,campo3);
   lineabuff=buftemp1;APUT(6*HC,5*AC,50*HC,11*AC);
  _ffree(buftemp1);_ffree(lineabuff);
   if((!CANC_RUTI)&(nombrevalido(campo3[3].var)))
    {    sprintf(nuevapal,campo3[3].var);
         strcat(nuevapal,".PAL");
         if( !_access( nuevapal, 0 ) ) //comprueba nombre existente
            {     AGET(6*HC,5*AC,50*HC,11*AC);buftemp1=lineabuff;
                  NUMIN=5;NUMAX=5;
                  RUTICAMPO(1,1,0,5,campo3);
                  lineabuff=buftemp1;APUT(6*HC,5*AC,50*HC,11*AC);
                  _ffree(buftemp1);_ffree(lineabuff);
                   if(!CANC_RUTI)
                           grabapaleta(nuevapal);
                   break;
            }
         grabapaleta(nuevapal);
    }
   break;
case 34: // Renombrar
   cantficheros=ficheros("PAL");
   if (!cantficheros)
```

```
        {NOHAYFICHEROS();break;}
NFIL=4;NCOL=5;X1=3;Y1=20;BORRAR=SI;CF=0;indice=0;
selecmatriz(&indice,dir);
if ((indice!=0)&&(indice<=cantficheros))
  {  corta(dir[indice],8);
     sprintf(campo3[2].var,dir[indice]);
     strcat(dir[indice],".PAL");
       // ALMACENAMIENTO ZONA
     AGET(6*HC,5*AC,50*HC,11*AC);buftemp1=lineabuff;
     NUMIN=2;NUMAX=2;
     campo3[3].var=campo3[2].var;
     RUTICAMPO(0,0,0,2,campo3);
     NUMIN=3;NUMAX=3;
     RUTICAMPO(1,1,1,3,campo3);
     lineabuff=buftemp1;APUT(6*HC,5*AC,50*HC,11*AC);
     _ffree(buftemp1);_ffree(lineabuff);
     if((!CANC_RUTI)&(nombrevalido(campo3[3].var)))
        {   sprintf(nuevapal,campo3[3].var);strcat(nuevapal,".PAL");
            if( !_access( nuevapal, 0 ) )
              {     AGET(6*HC,5*AC,50*HC,11*AC);
                    buftemp1=lineabuff;
                    NUMIN=5;NUMAX=5;
                    RUTICAMPO(1,1,0,5,campo3);
                    lineabuff=buftemp1;APUT(6*HC,5*AC,50*HC,11*AC);
                    _ffree(buftemp1);_ffree(lineabuff);
               }
            else
              { rename(dir[indice],nuevapal); }
        }
```

```
      }
    break;
  case 35: // Borrar
    cantficheros=ficheros("PAL");
    if (!cantficheros)
          {NOHAYFICHEROS();break;}
    NFIL=4;NCOL=5;X1=3;Y1=20;BORRAR=SI;CF=0;indice=0;
    selecmatriz(&indice,dir);
    if ((indice!=0)&&(indice<=cantficheros))
      {   corta(dir[indice],8);
          sprintf(campo3[4].var,dir[indice]);
          strcat(dir[indice],".PAL");
           // ALMACENAMIENTO ZONA
           AGET(6*HC,5*AC,50*HC,11*AC);buftemp1=lineabuff;
           NUMIN=4;NUMAX=4;
           RUTICAMPO(1,1,0,4,campo3);
           lineabuff=buftemp1;APUT(6*HC,5*AC,50*HC,11*AC);
           _ffree(buftemp1);_ffree(lineabuff);
           if(!CANC_RUTI)    {remove(dir[indice]);}
      }
    break;
  case 36: // Mostrar paleta
    paleta(615,20,20,20,16,1,1);
    break;
  case 37: // ocultar  paleta
    ZONAPALETA();
    _setviewport(short(dzona[3][0]), short(dzona[3][1]), short(dzona[3][2]),
                    short(dzona[3][3]));
    _clearscreen(_GVIEWPORT);
```

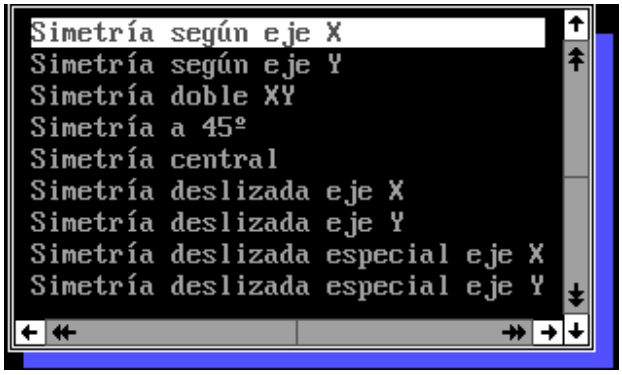
```
            _setviewport(0,0,cv.numxpixels,cv.numypixels);
            break;
         default:
            {}
         }
    break; // fin case 3 (PALETA)

case 4:          //*****COLORES*****
                 //"Pintar,Pintar doble,Pintar Cuadruple,Sustituir,Mezclar";
          switch (opcion)
         {
         case 41: //Pintar
            PINTAR(1);
            break;
         case 42: //Pintar doble
            PINTAR(2);
            break;
         case 43: //Pintar cuadruple
            PINTAR(4);
            break;
         case 44: //Sustituir color
            SUSTITUYECOLOR();
            break;
         case 45: //Mezclar
            MEZCLACOLOR();
            break;
         default:
            {}
         } // fin case 4 (COLORES)
```

```
        break;

case 5:         //*****PANTALLA*****
                //"Simular Tejido,Simular Curso,Simulación Completa,
                //Limpia Tejido,Limpia Zona,Limpia Pantalla";
   switch (opcion)
        {
        case 51: // Simular Tejido (simular en cuadro)
           _settextposition(30,1);
           printf("Indicar punto inicio con el ratón");
           PuntoRaton();
           _settextposition(30,1);printf("                          ");
           avent=x_rat;bvent=y_rat;
           _settextposition(30,1);
          printf("Indicar punto final");
           Selrectan();
           _settextposition(30,1);printf("                ");
           cvent=x_rat;dvent=y_rat;
           ORDVERT(avent,bvent,cvent,dvent);
           avent=xizq;bvent=ysup;cvent=xder;dvent=yinf;
           _setcliprgn(short(avent+1),short(bvent+1),short(cvent),short(dvent));
           rellenareti(avent,bvent,cvent,dvent,lh,lv,nf,nc);
           _setcliprgn(0,0,cv.numxpixels,cv.numypixels);
           _setcolor(15);
           HideCursor();
           _rectangle (_GBORDER,short(avent),short(bvent),short(cvent),short(dvent));
           ShowCursor();
           break;
        case 52: //Simular tejido ajustado al curso
```

```
    _settextposition(30,1);printf("Indicar punto inicio con el rat¢n");
    PuntoRaton();
    _settextposition(30,1);printf("                              ");
    avent=x_rat;bvent=y_rat;
    _settextposition(30,1);printf("Indicar punto final");
    Selrectan();
    _settextposition(30,1);printf("                    ");
    cvent=x_rat;dvent=y_rat;
    ORDVERT(avent,bvent,cvent,dvent);
    avent=xizq;bvent=ysup;cvent=xder;dvent=yinf;
    // calculo para ajustar ventana y respetar ligamento
    cvent=avent + (int)((cvent-avent)/nc)*nc;
    dvent=bvent + (int)((dvent-bvent)/nf)*nf;
    _setcliprgn(short(avent+1),short(bvent+1),short(cvent),short(dvent));
    rellenareti(avent,bvent,cvent,dvent,lh,lv,nf,nc);
    _setcliprgn(0,0,cv.numxpixels,cv.numypixels);
    _setcolor(15);
    HideCursor();
    _rectangle (_GBORDER,short(avent),short(bvent),short(cvent),short(dvent));
    ShowCursor();
    break;
  case 53: //Simular tejido a pantalla completa)
    _clearscreen(_GCLEARSCREEN);
    HideCursor(); HideCursor();
    // calculo para ajustar ventana y respetar ligamento
    avent=0;bvent=0;
    cvent=cv.numxpixels-1;dvent=cv.numypixels-1;
    rellenareti(avent,bvent,cvent,dvent,lh,lv,nf,nc);
     _setcolor(15);
```

```
    _rectangle (_GBORDER,0,0,short(cvent),short(dvent));
    MULTIPLENTRADA();
    _setcolor(0);
    _rectangle (_GFILLINTERIOR,0,0,639,short(AC));
    RESALTAF(1,1,menu[1][0],15,0); //imprime menú en blanco intenso
    ShowCursor();
    break;
case 54:  //limpia tejido
    HideCursor();
    _setviewport(short(avent),short(bvent),short(cvent),short(dvent));
    _clearscreen(_GVIEWPORT);
    _setviewport(0,0,cv.numxpixels,cv.numypixels);
    break;
case 55:   //limpia zona
    PuntoRaton();
    avent=x_rat;bvent=y_rat;
    Selrectan();
    cvent=x_rat;dvent=y_rat;
    HideCursor();
    _setviewport(short(avent),short(bvent),short(cvent),short(dvent));
    _clearscreen(_GVIEWPORT);
    _setviewport(0,0,cv.numxpixels,cv.numypixels);
    break;
case 56:   //limpia pantalla
    HideCursor();
    _clearscreen(_GCLEARSCREEN);
    _setcolor(0);_rectangle (_GFILLINTERIOR,0,0,639,short(AC));
    RESALTAF(1,1,menu[1][0],15,0);//imprime menú en blanco intenso
    break;
```

```
        default:
           {}
        }
    break; // fin case 5 (PANTALLA)


case 6:         //*****TRANSFORMAR*****
                //"Traslación...,Simetrias...,Giros...,Copias...,Rellenos...";
                // La pantalla que nos servirá para tomar decisiones en cualquiera de los
                casos anteriores de traslación, simetría, etc... será como la que se indica a
                continuación (caso particular para los movimientos de simetría).
```




```
    switch (opcion)
        {
        case 61:        //  trasladar zona
           TRASLADAR();
                break;
        case 62:        //  simetrias
           SIMETRIA();
                break;
        case 63:        //  giros
           GIROS();
```

```
                break;
        case 64:            //    copias
           COPIAR();
                break;
        case 65:            //    rellenos
           RELLENAR();
                break;
        default:
           {}
        }
    break; // fin case 6 (TRANSORMAR)


case 7:         //*****UTILES*****
                //"Oculta Menu,Sin Cuadricula,Con Cuadricula,
                //Mostrar Ajuste,Ocultar Ajuste,Utiles Raton...";
switch (opcion)
        {
        case 71: // oculta menu
           ZONAMENU();
           _setviewport(short(dzona[1][0]), short(dzona[1][1]), short(dzona[1][2]),
                        short(dzona[1][3]));
           HideCursor();
           _clearscreen(_GVIEWPORT);
           _setviewport(0,0,cv.numxpixels,cv.numypixels);
           MULTIPLENTRADA();
           _setcolor(0);
           _rectangle (_GFILLINTERIOR,0,0,639,short(AC));
           RESALTAF(1,1,menu[1][0],15,0); //imprime menú en blanco intenso
```

```
    ShowCursor();
    break;
case 72:   //sin  cuadricula
    cuadro=0;
    reticular(izx,izy);
    break;
case 73:   //con cuadricula
    cuadro=1;
    reticular(izx,izy);
    break;
case 74: // Mostrar caja de ajuste color
    cajaajustes();
    break;
case 75: // ocultar caja de ajustes
    ZONAAJUSTE();
    _setviewport(short(dzona[4][0]), short(dzona[4][1]), short(dzona[4][2]),
                    short(dzona[4][3]));
    _clearscreen(_GVIEWPORT);
    _setviewport(0,0,cv.numxpixels,cv.numypixels);
    break;
case 76:// utiles de raton
      menurat[1]= "Normal ";
      menurat[2]= "Gráfico ";
      menurat[3]= "Flecha ";
      menurat[4]= "Mostrar ";
      menurat[5]= "Ocultar ";
      NFIL=3;NCOL=2;X1=50;Y1=8;BORRAR=SI;CF=0;indice=0;
      selecmatriz(&indice,menurat);
      switch(indice)
```

```
                {
                 case 1:
                    InitMouse();
                        break;
                 case 2:
                    GrafCursor();
                        break;
                 case 3:
                    FlechaCursor();
                        break;
                 case 4:
                    ShowCursor();
                        break;
                 case 5:
                    HideCursor();
                    MULTIPLENTRADA();ShowCursor();
                         break;
                 default:
                          {}
                }
        default:
           {}
        } // fin case 7 (UTILES)
        break;

//************************************
   default:       // del case principal
     {}
   }// fin de switch (submenu)
```

```
  }  //fin if de menus principales
goto INICIO;
return (0);            // fin de main
}
```

6.4. PROGRAMA DE DISEÑO TEXTIL EGITEX.EXE

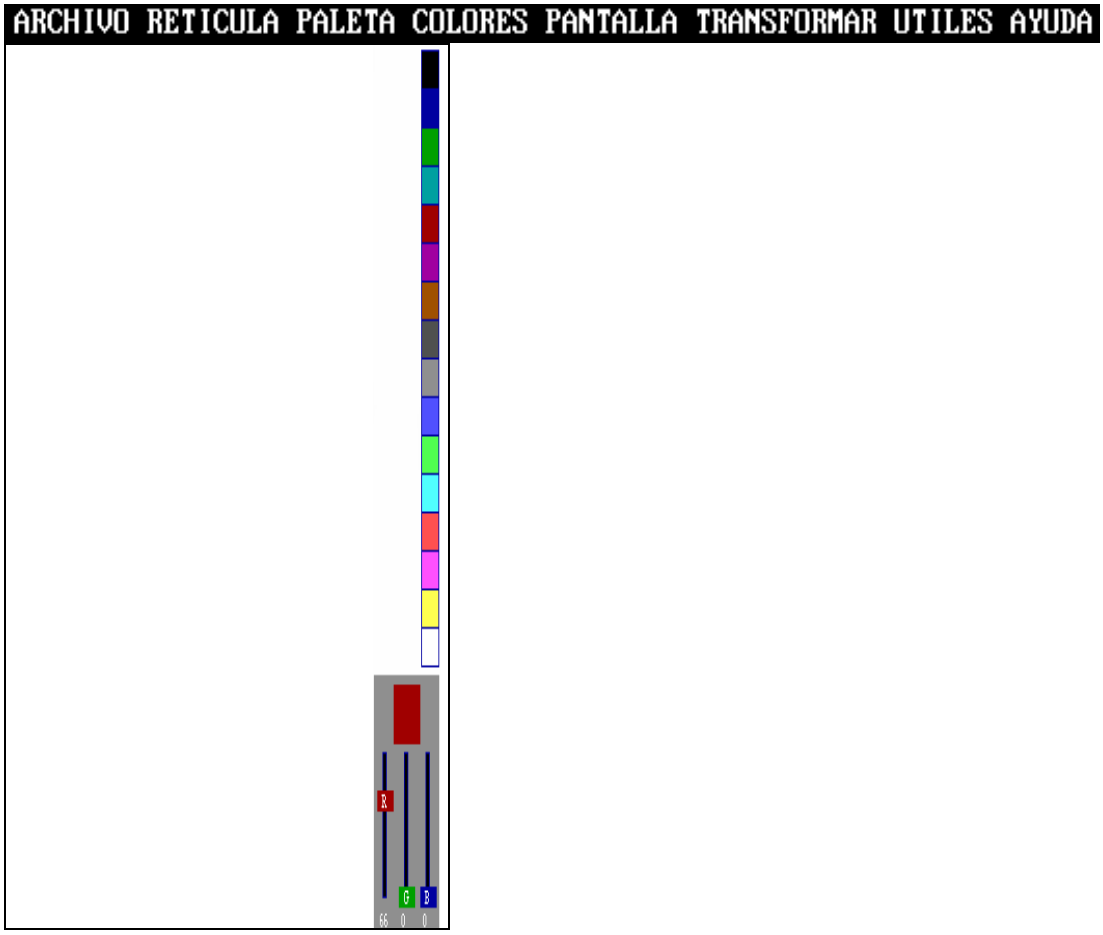
Para fabricar el programa ejecutable desde sistema operativo MSDOS o Windows se han enlazado los ocho ficheros que se han descrito anteriormente:

0PRINCIP.CPP
0INCLUD.H
0RATON.CPP
0FICHER.CPP
0RUTINA.CPP
0MENUS.CPP
0MOVIM.CPP
0PALETA.CPP
0TEXTIL.CPP

en un solo proyecto, denominado EGITEX.MAK el cual se ha compilado y linkado en C/C++, resultando un solo fichero ejecutable que se denomina EGITEX.EXE que al ocupar un espacio de 126.066 Kb, es fácilmente transportable en un solo disquete

Al ejecutar el programa aparecerá una pantalla como la que se muestra a continuación, donde el ratón explora las diferentes zonas y decide actuar de forma inteligente en función de la misma, para resaltar los menús, seleccionar color para dibujar, cambiar el color mediante la caja de ajustes RGB, o la zona de retícula en que puede pintar con el color seleccionado.

Mientras que el ratón se mueve por la zona superior del menú principal se resalta la opción por la que pasa y sólo cuando se pulsa, se despliega el submenú correspondiente a la acción deseada; en este momento tenemos la posibilidad de elegir un comando de ese submenú, o bien, pasar a los otros submenús colgantes mediante ratón o teclas de cursor del teclado.

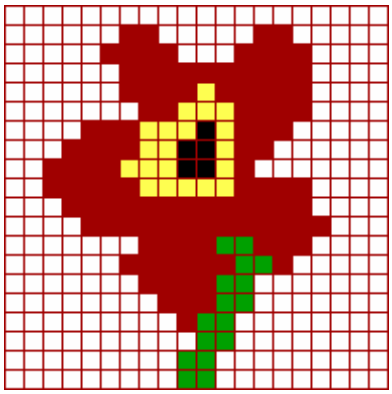


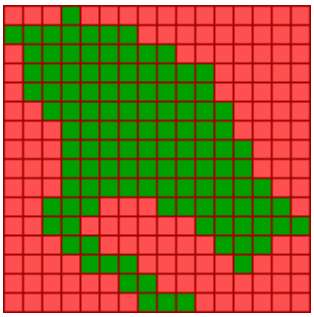
A modo de ejemplo se muestra en la pantalla siguiente el submenú que aparece con la opción del menú principal ARCHIVO.

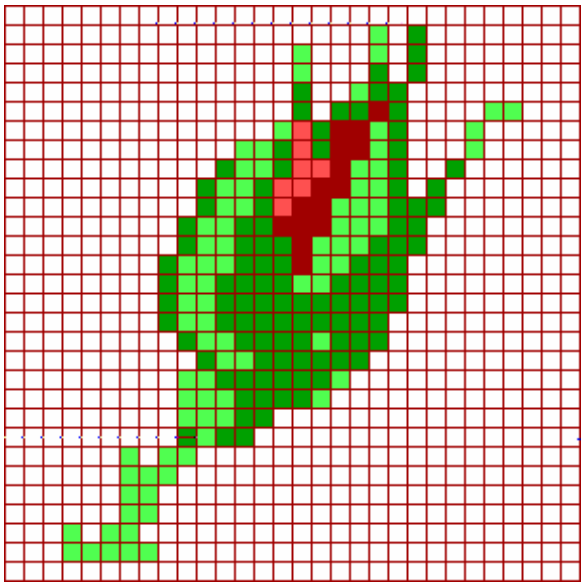
No siendo el objetivo de este apartado el de realizar un manual de instrucciones de este programa, se va a prescindir de mostrar todas y cada una de las funciones que realiza cada submenú, exponiéndose a continuación algunos de los diseños que se pueden realizar con cambio de color y aplicando algunos movimientos de los explicados en capítulos anteriores.

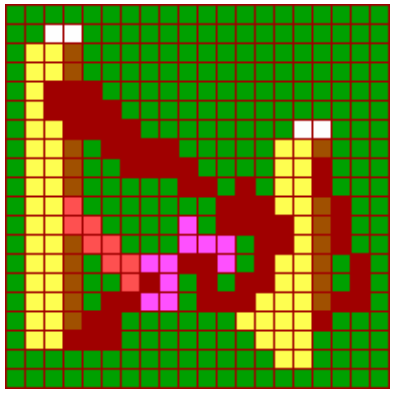
Aquí se van a mostrar las retículas independientes y se dejan para el próximo apartado algunas simulaciones de tejido que las mismas proporcionan.

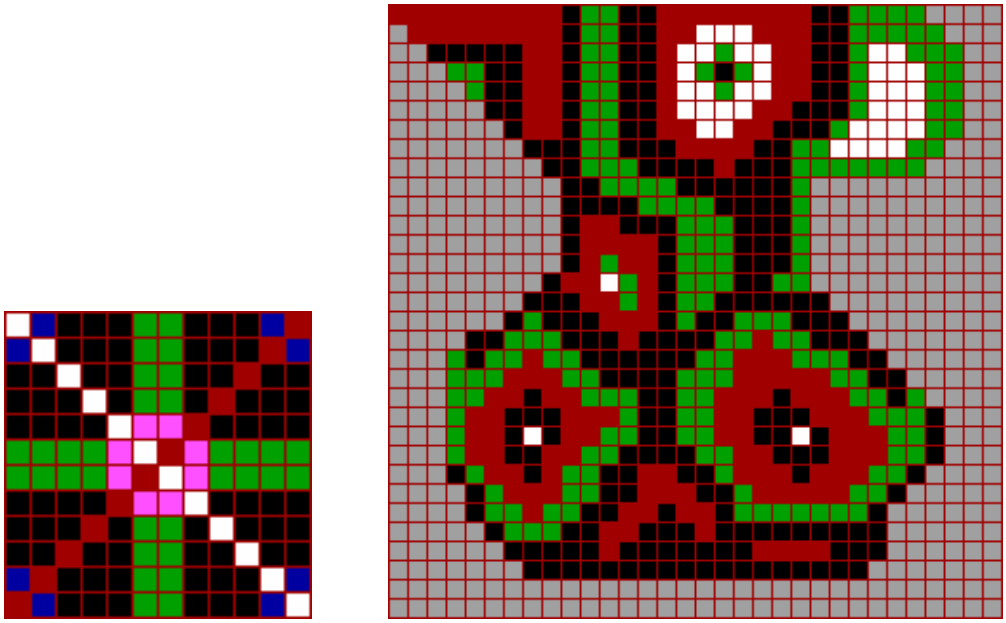
- Ejemplos de motivos obtenidos con la opción PINTAR del menú colgante COLORES:



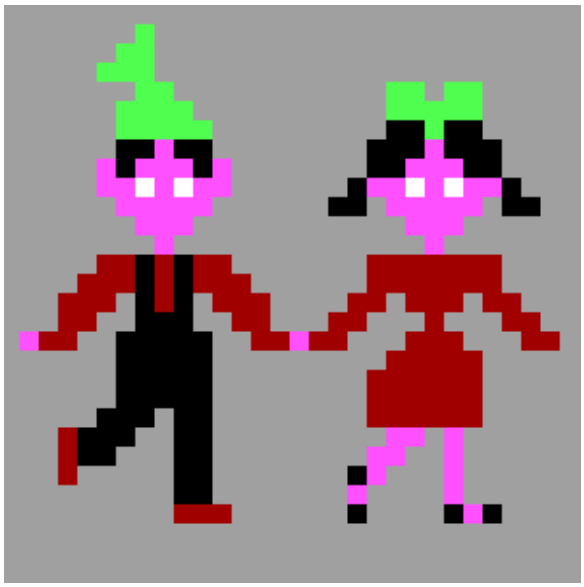




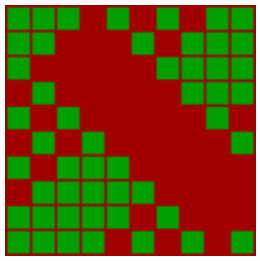
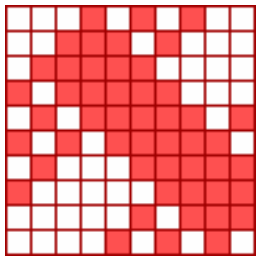
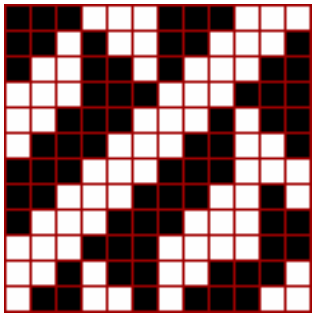
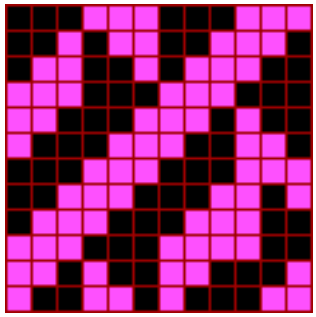




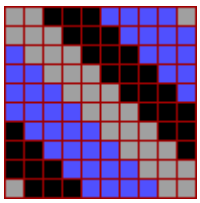
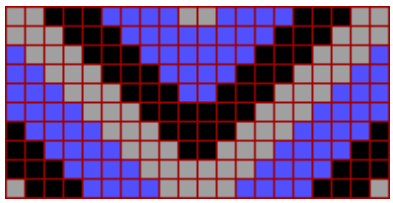
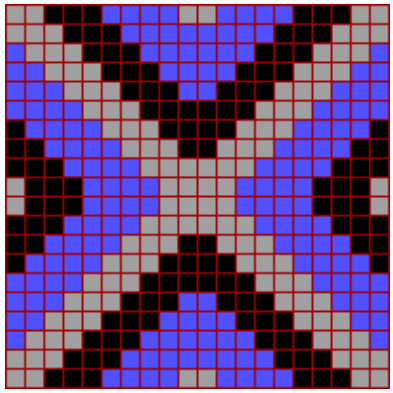
- Posibilidad de desactivar la cuadrícula con la opción SIN CUADRICULA en el menú colgante de UTILES:



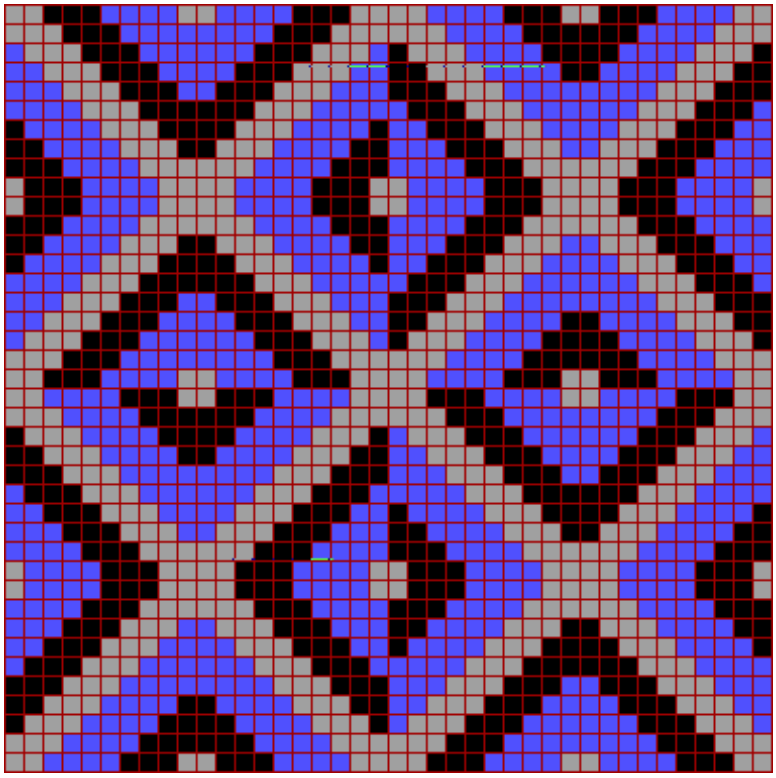
- Cambio de color en un mismo motivo con la opción SUSTITUIR del menú colgante de COLORES, disponiendo además de la opción de mezclar dos colores para obtener un tercero.

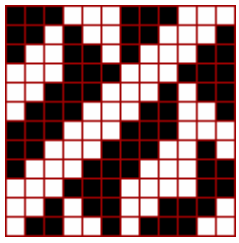
- Obtención de diseños compuestos aplicando los movimientos descritos en el Capítulo anterior como por ejemplo traslaciones, simetrías, giros...:

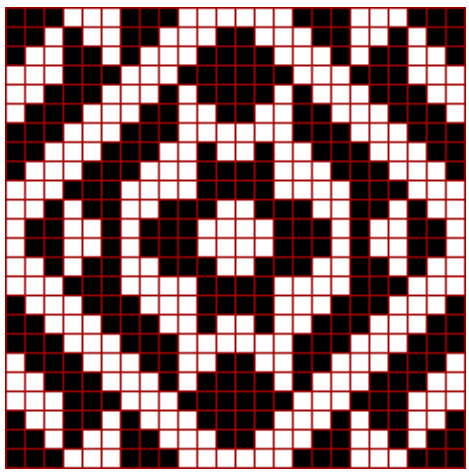
Motivo inicial Simetría eje Y Simetría eje X



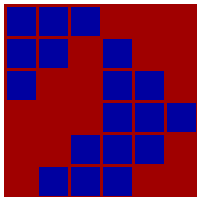
Tres giros de 90º (Motivo final)



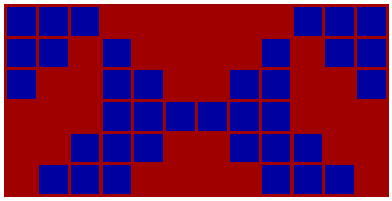
Motivo inicial



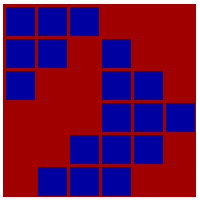
Simetría doble (eje X, eje Y)



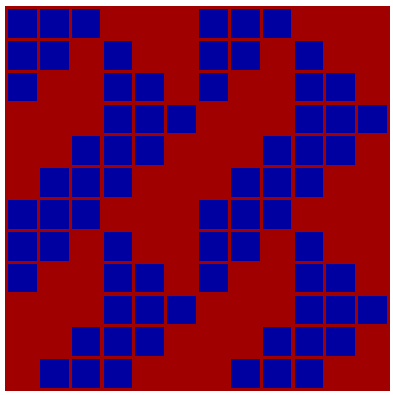
Motivo inicial



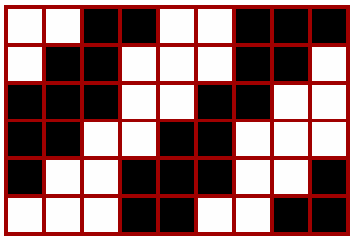
Simetría eje Y



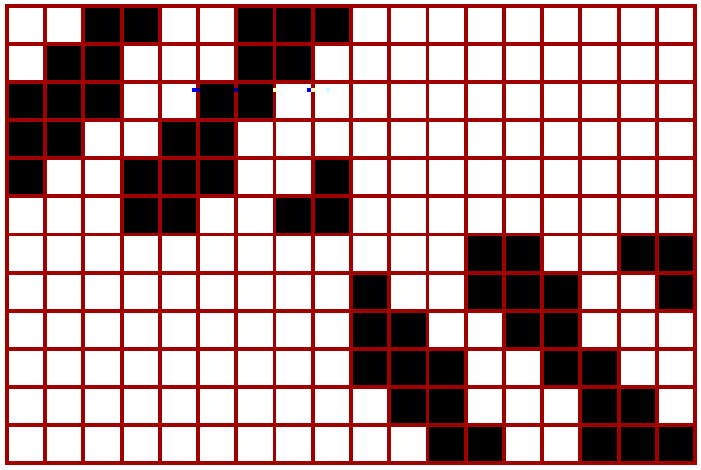
Motivo inicial



Traslaciones

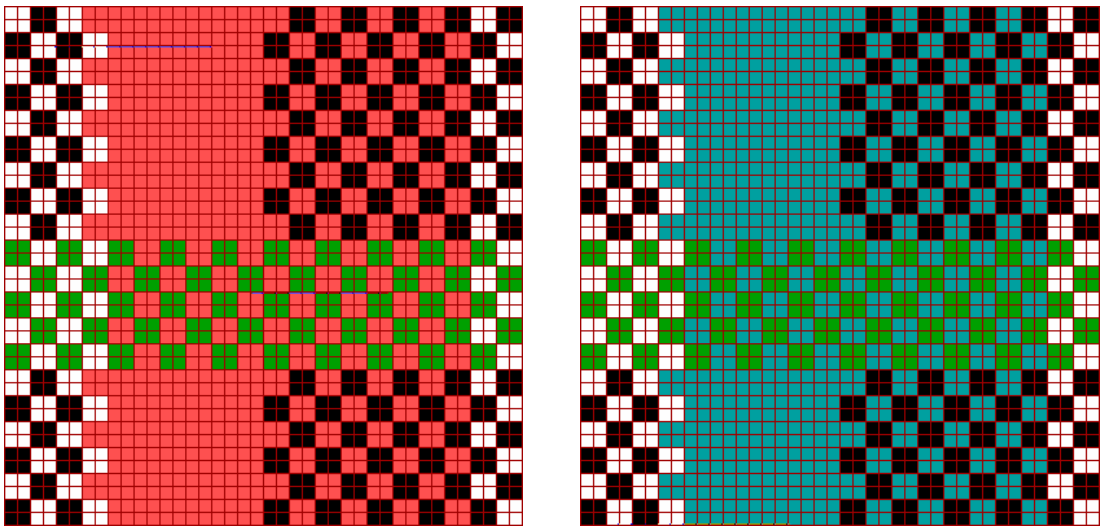


Motivo inicial

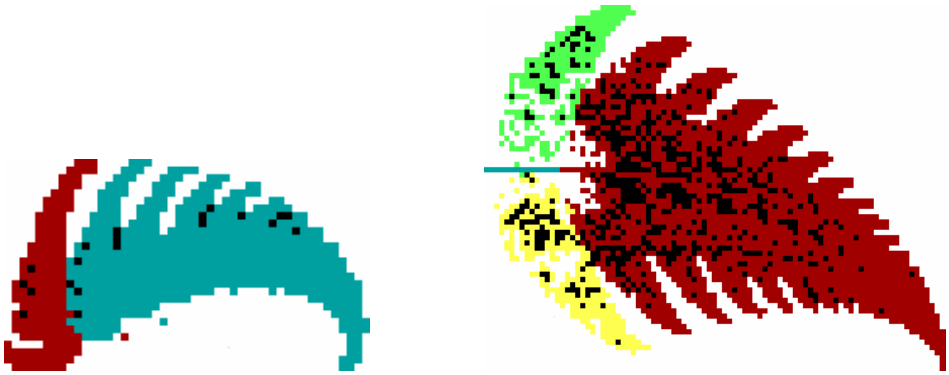


Simetría deslizada

- Ejemplos de diseño copiando zonas y con sustitución de colores

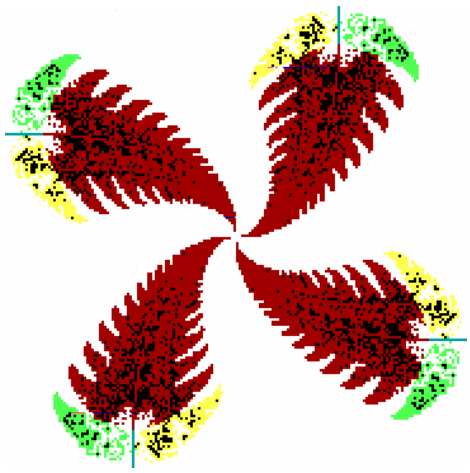


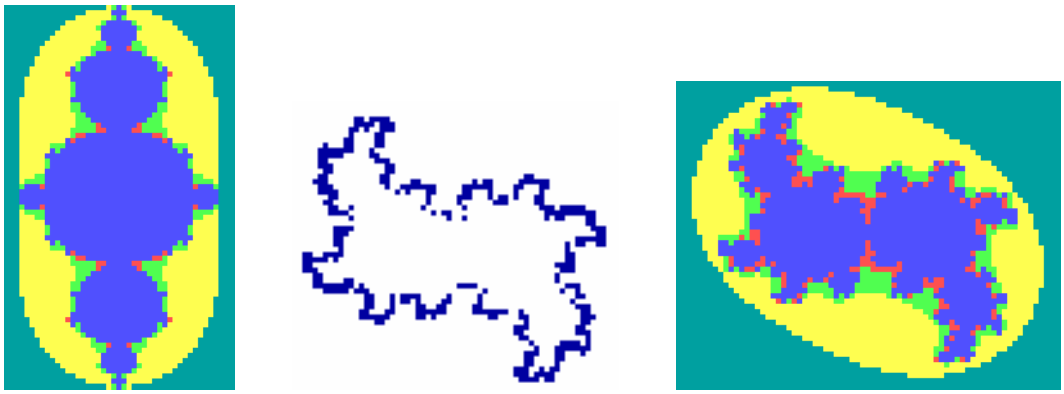
- Ejemplo de nuevas posibilidades de diseño incorporando las rutinas de cálculo de fractales del tipo de sistemas de función iterada o de conjuntos de Julia.



Sistemas de Función Iterada (SFI)

Una vez generado el fractal y asociado a una retícula, se le puede someter a cualquier movimiento, como por ejemplo 3 giros de 90º.





Conjuntos de Julia

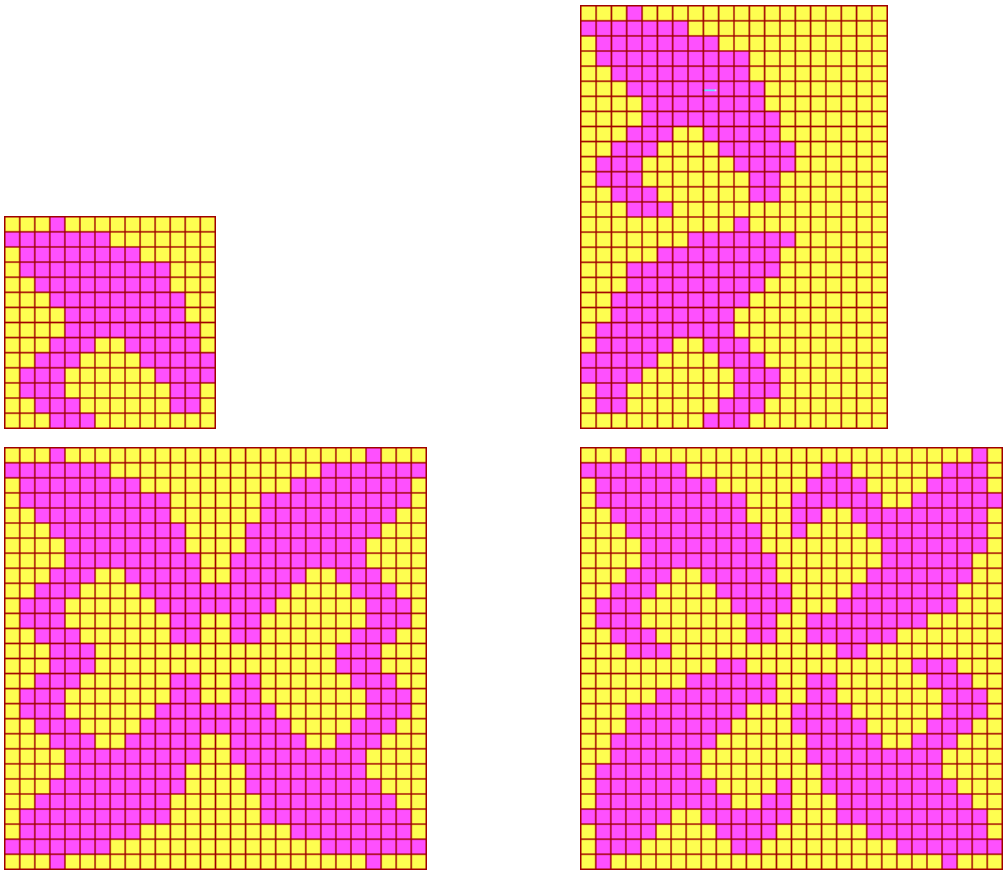
Como puede comprobarse por las muestras aquí presentadas, las posibilidades de obtención de diseños son infinitas, siendo la capacidad del diseñador la que da lugar a efectos más o menos acertados y/o espectaculares.

6.5. MUESTRAS DE DISEÑO DE TEJIDOS MEDIANTE EGITEX

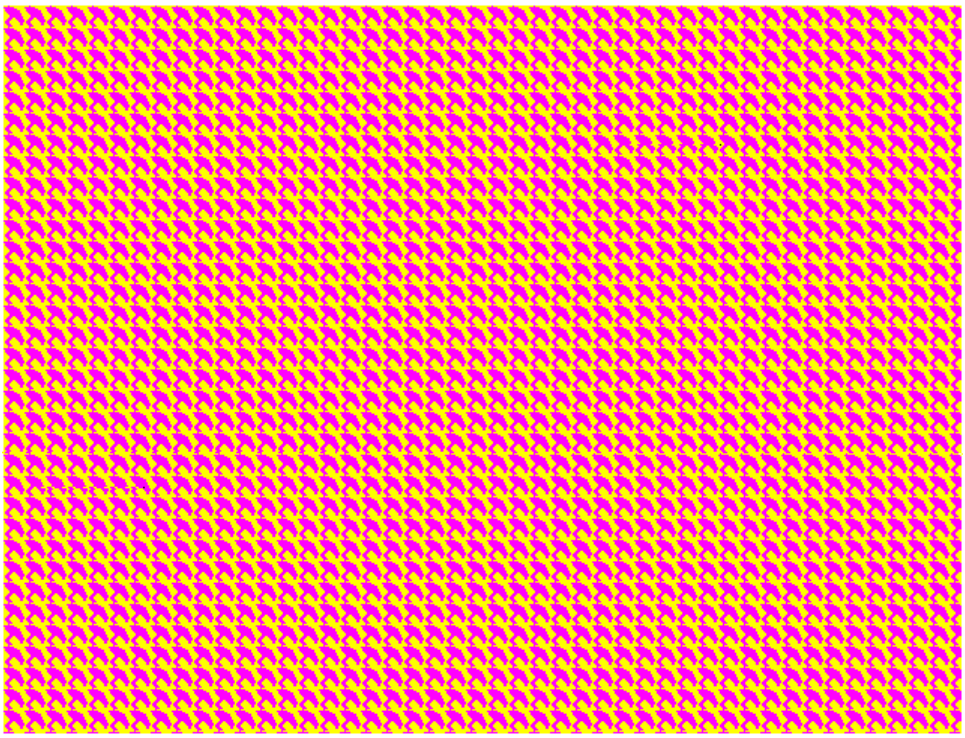
Siguiendo la notación propuesta en el Capitulo 5 anterior se muestran a continuación algunas simulaciones de tejido que el programa EGITEX realiza con solo acceder al menú desplegable de PANTALLA.

Las primeras simulaciones que se presentan corresponden precisamente a los ejemplos de diseño de motivos que se han realizado en el Apartado 5.5.5 **Notaciones Compuestas**, a los que se les ha aplicando la sustitución de colores, para ver distintos aspectos en las composiciones que se han realizado.

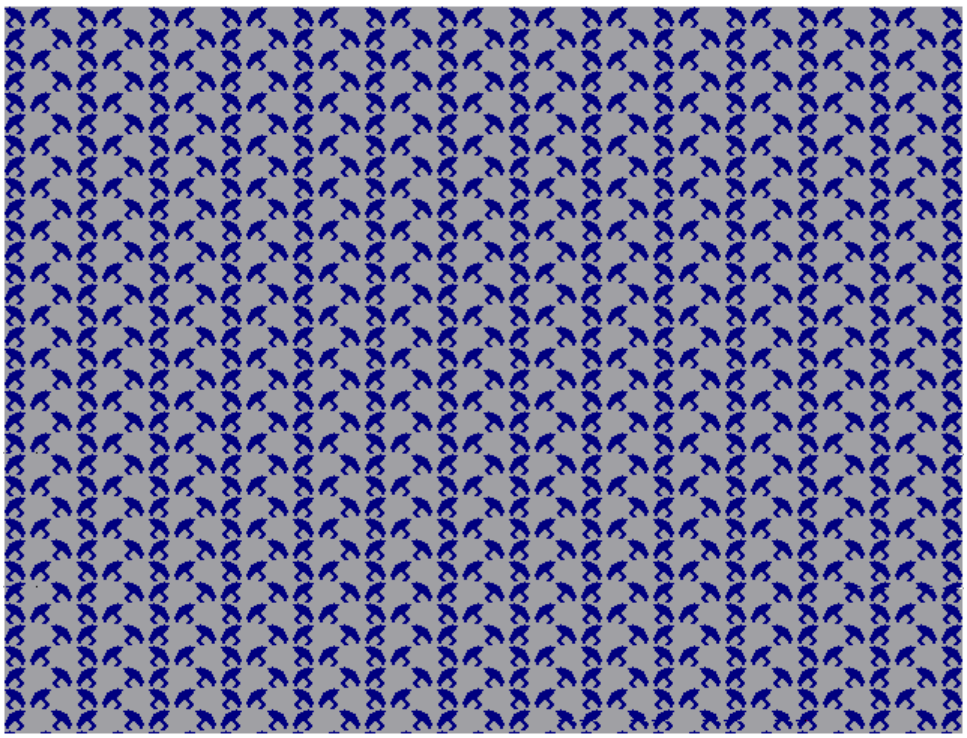
Partiendo de un motivo inicial, se consiguen nuevas retículas con la aplicación de movimientos de simetría, simetría deslizada o giros.



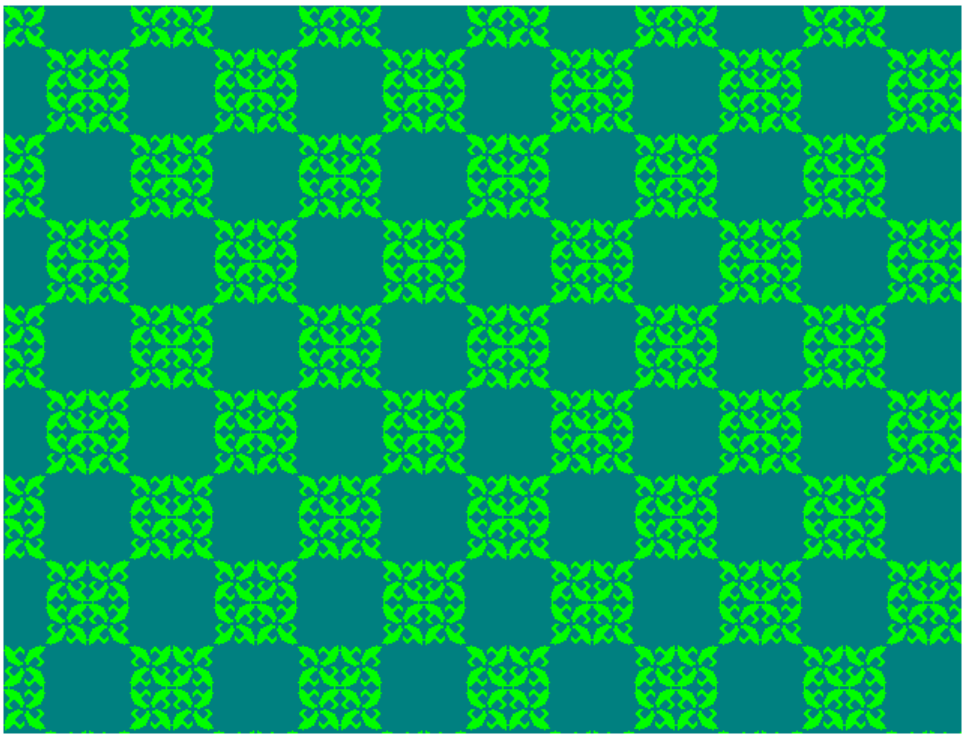
Esta simulación corresponde al motivo inicial sin ninguna transformación.



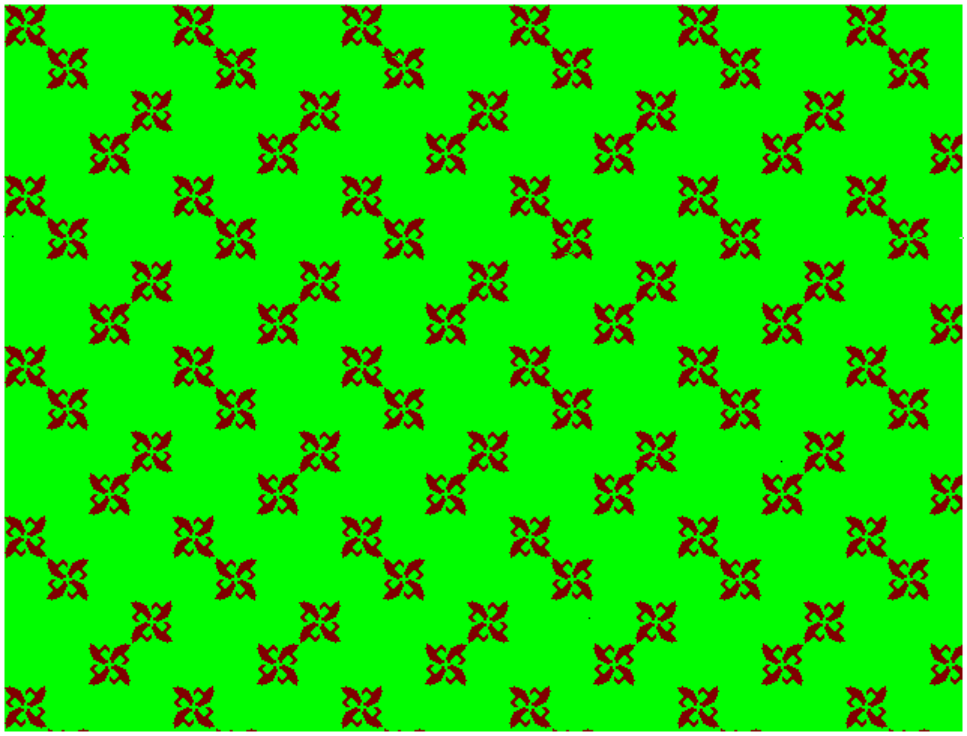
Ejemplo 1 del Apartado 5.5.5.



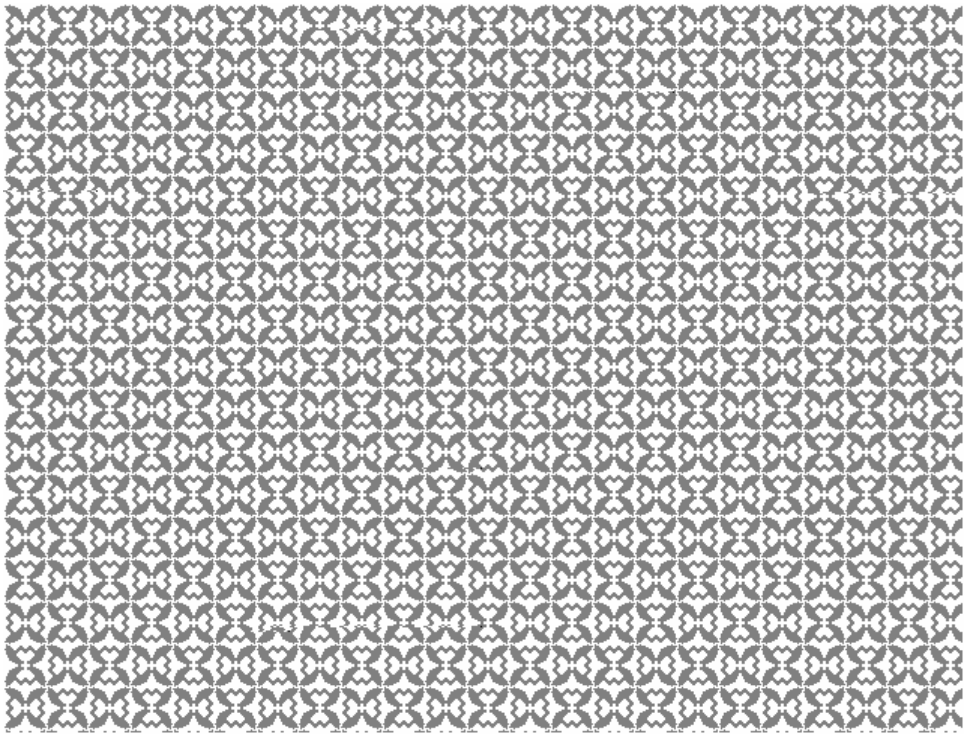
Ejemplo 2 del Apartado 5.5.5.



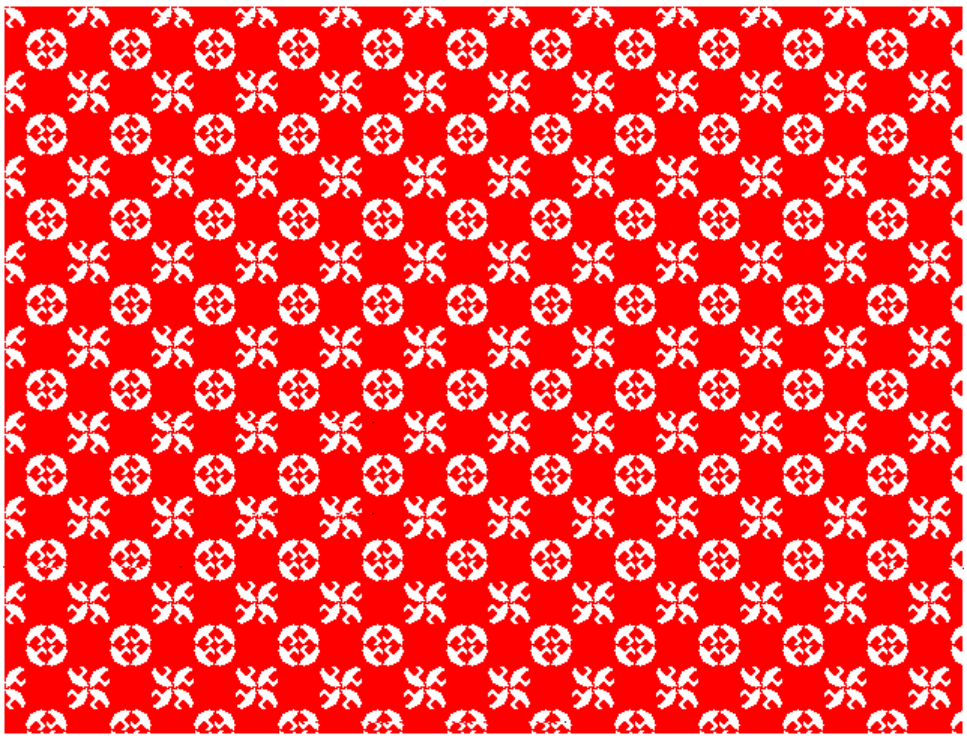
Ejemplo 3 del Apartado 5.5.5.



Ejemplo 4 del Apartado 5.5.5.

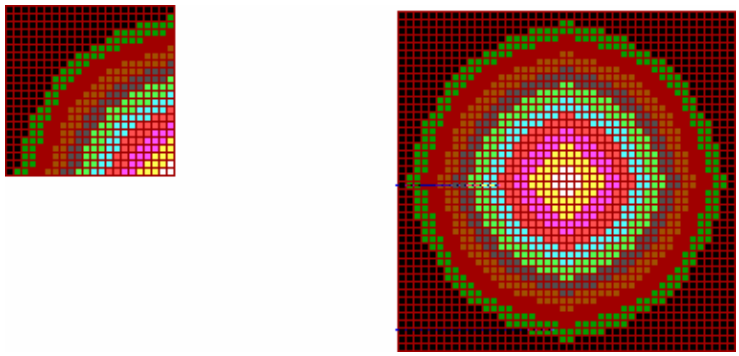


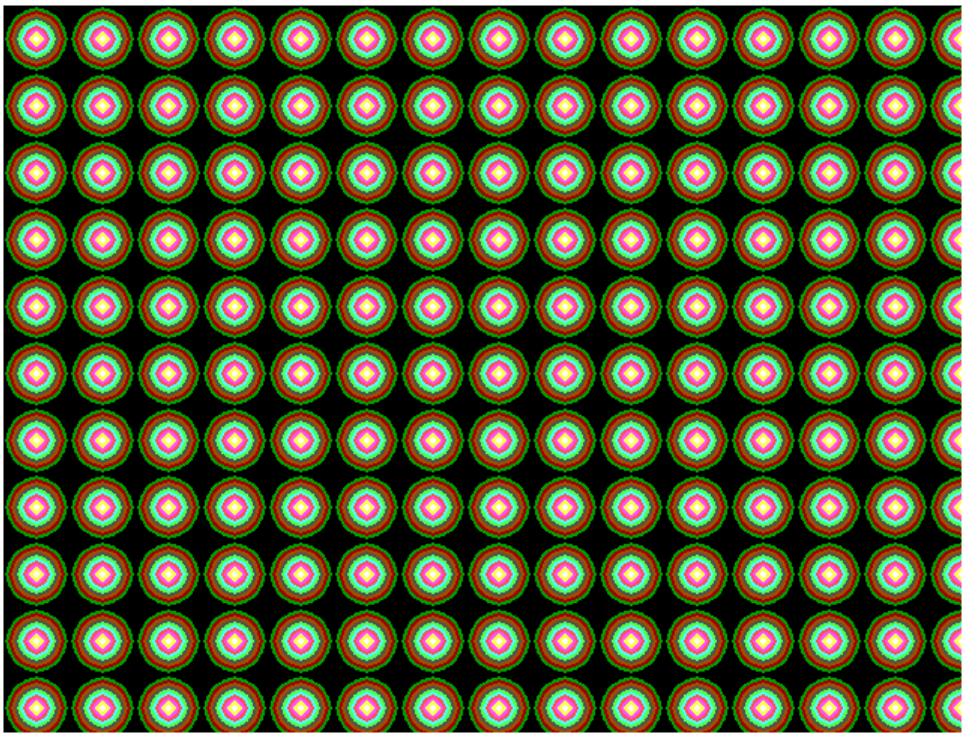
Ejemplo 5 del Apartado 5.5.5.



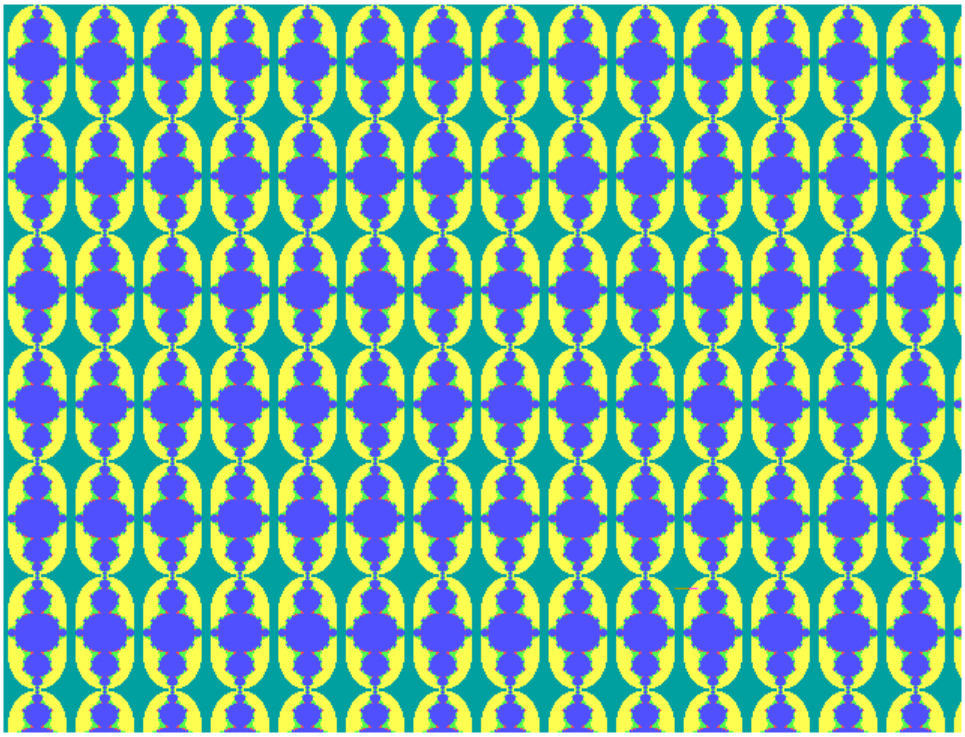
A continuación se muestran algunos otros ejemplos de simulación del tejido, en los que el resultado final dependerá de la imaginación de cada uno.

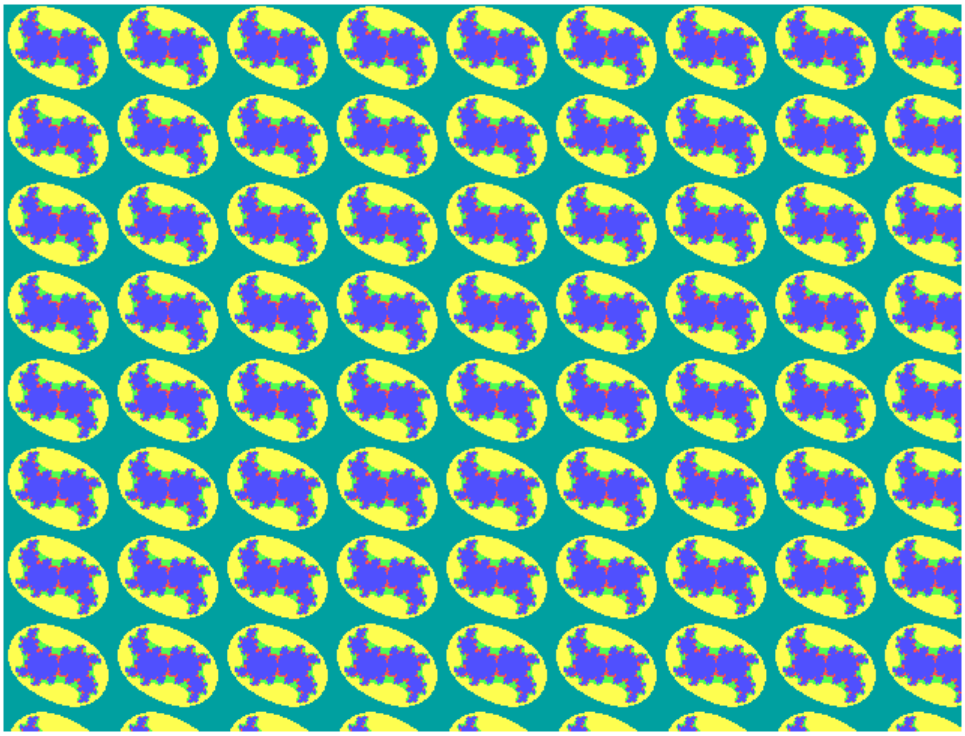
El diseño siguiente muestra la obtención de un circulo de colores a partir de un cuadrante del mismo al que se le aplica una doble simetría.



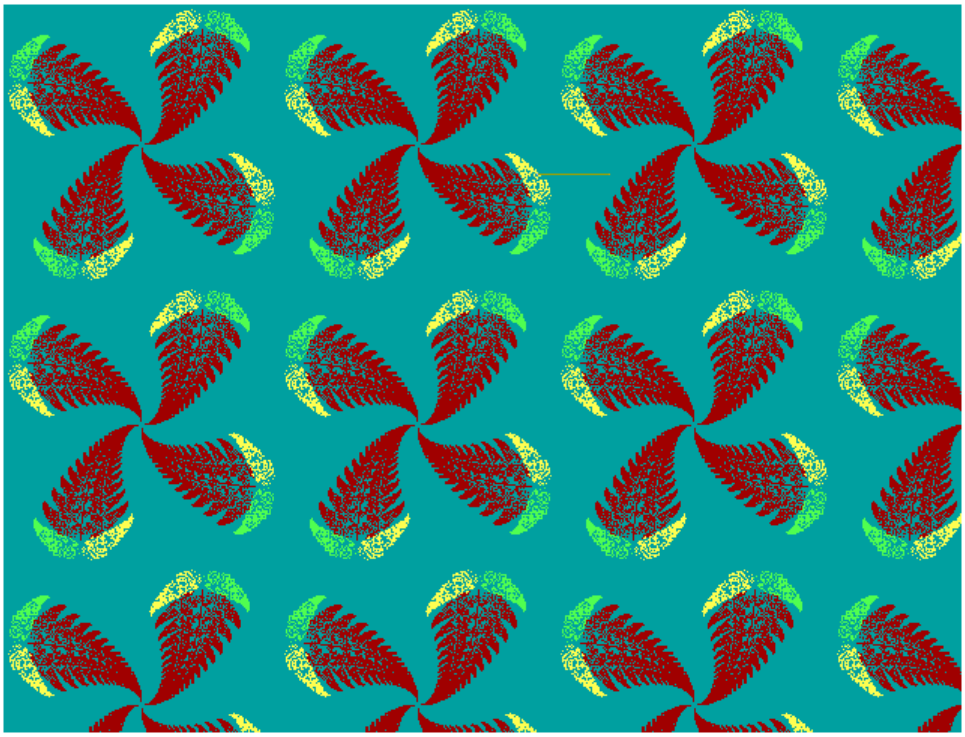


Las dos próximas simulaciones corresponden a un diseño de tejido realizado en base a la aplicación de la geometría fractal de los conjuntos de Julia.

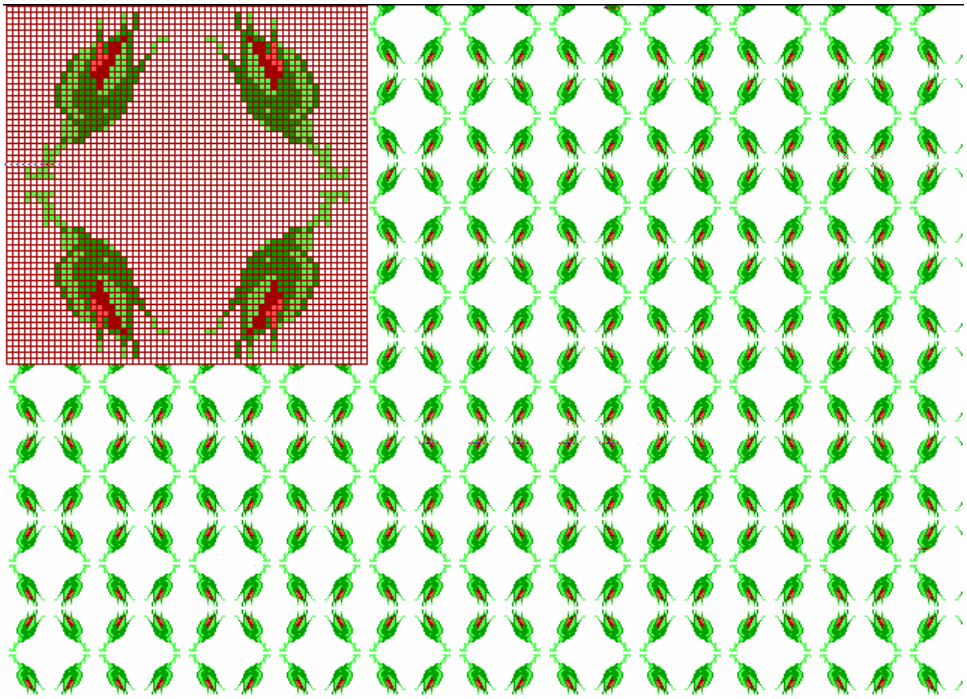




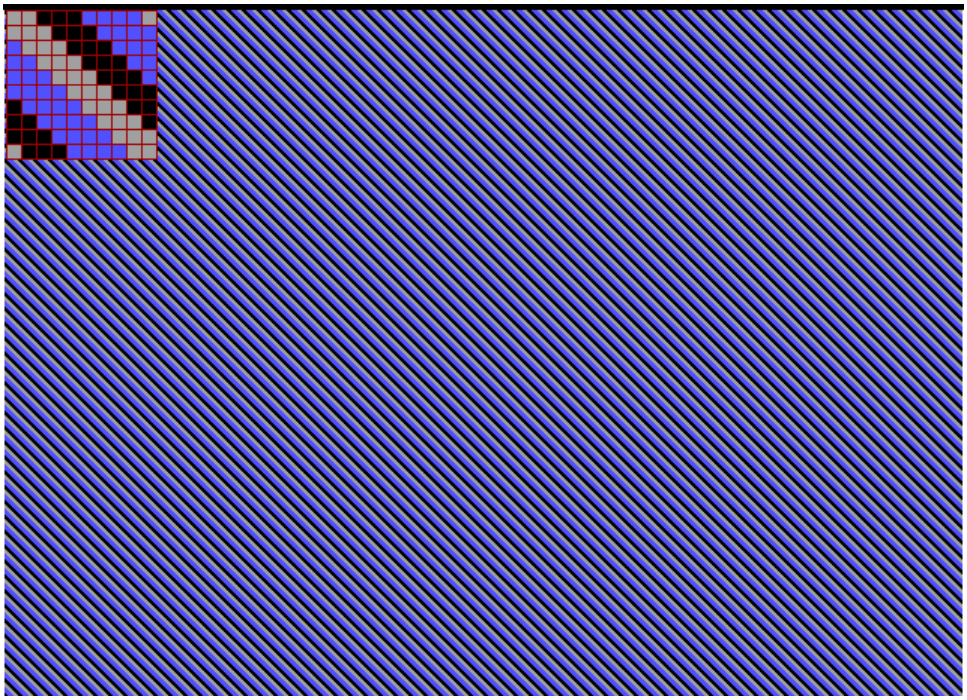
Simulación corresponde a un fractal de sistemas de función iterada (SFI), al que se le han aplicado 3 giros de 90º.



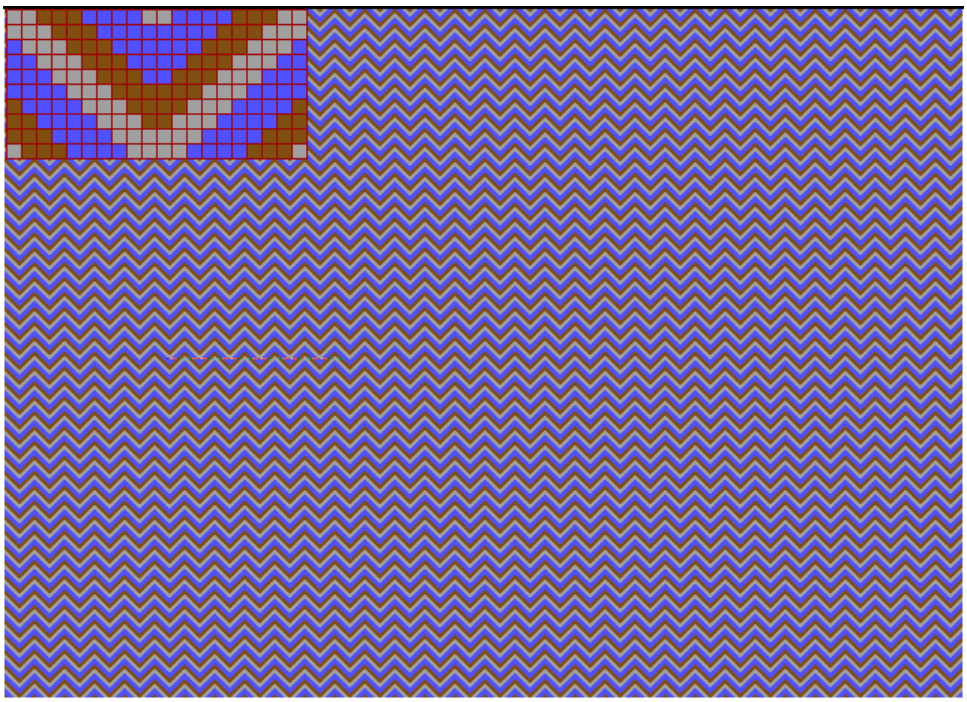
En la simulación siguiente se muestra un motivo inicial con una doble simetría



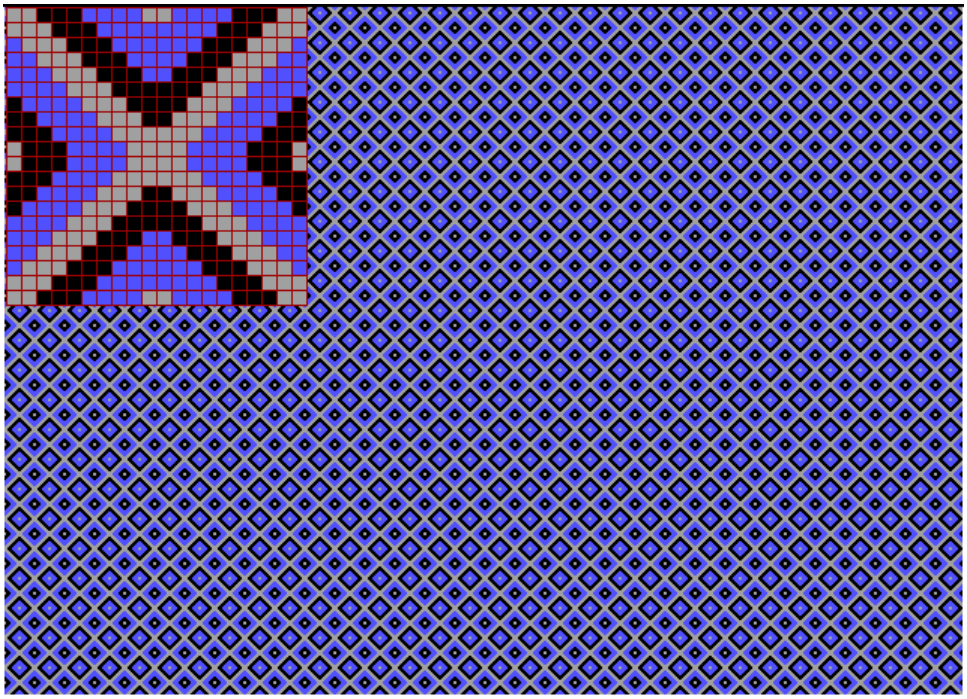
Simulación de un ligamento.



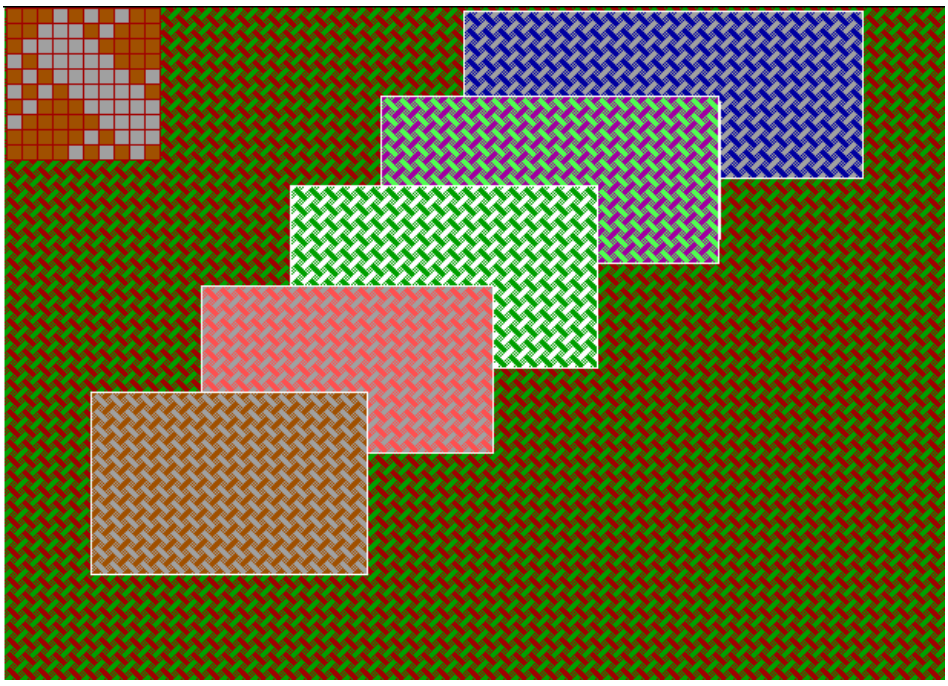
El ligamento anterior con una simetría respecto del eje Y y cambio de un color.



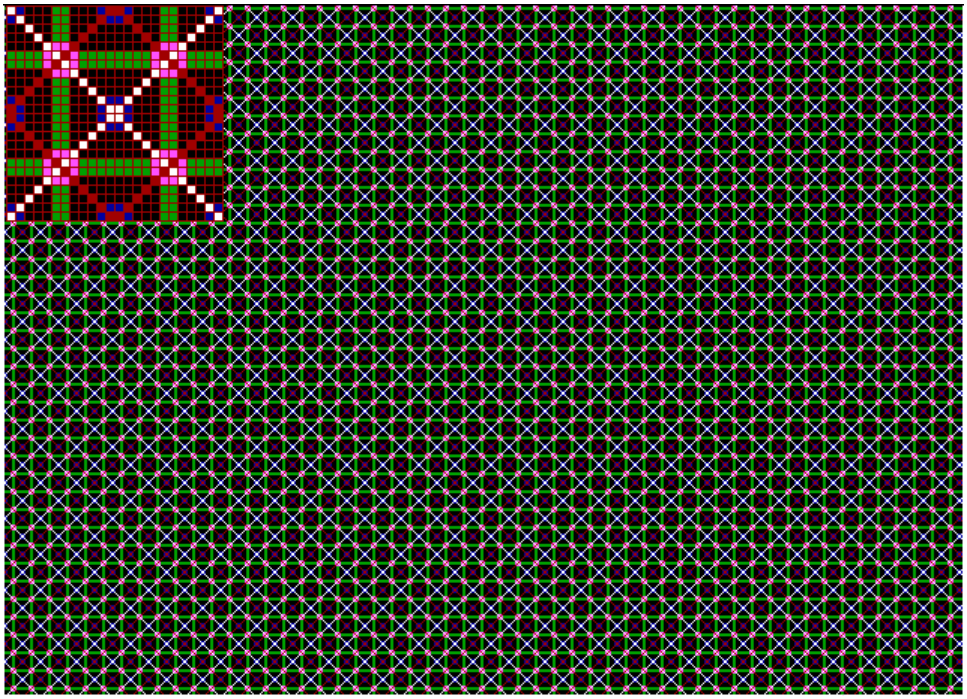
Ligamento realizado con otra simetría sobre el eje X.



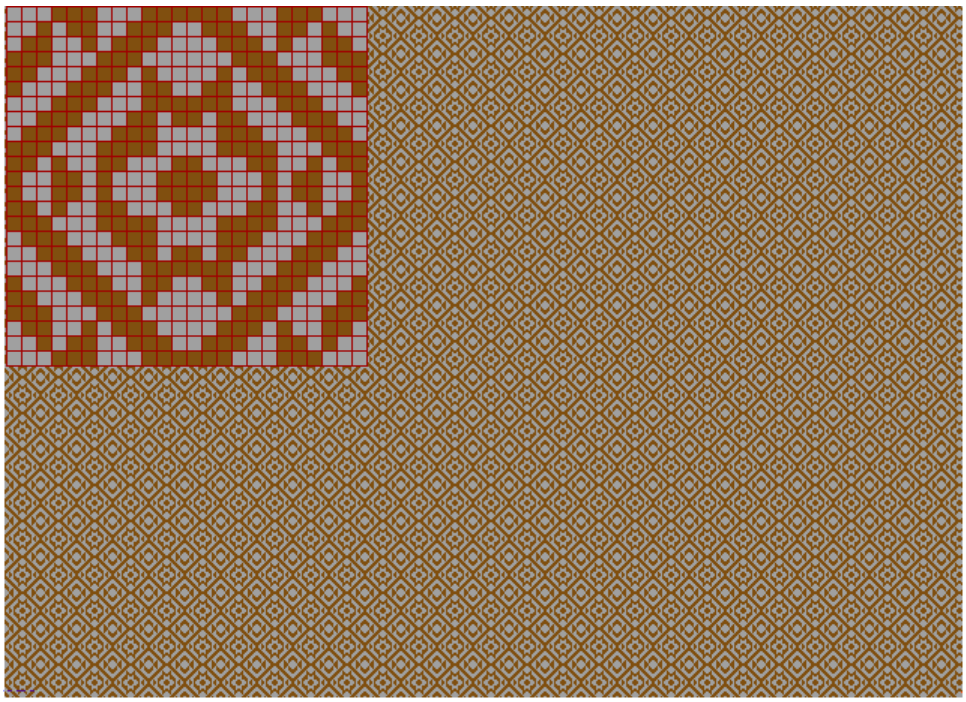
Posibilidad de realizar un muestrario de colores.



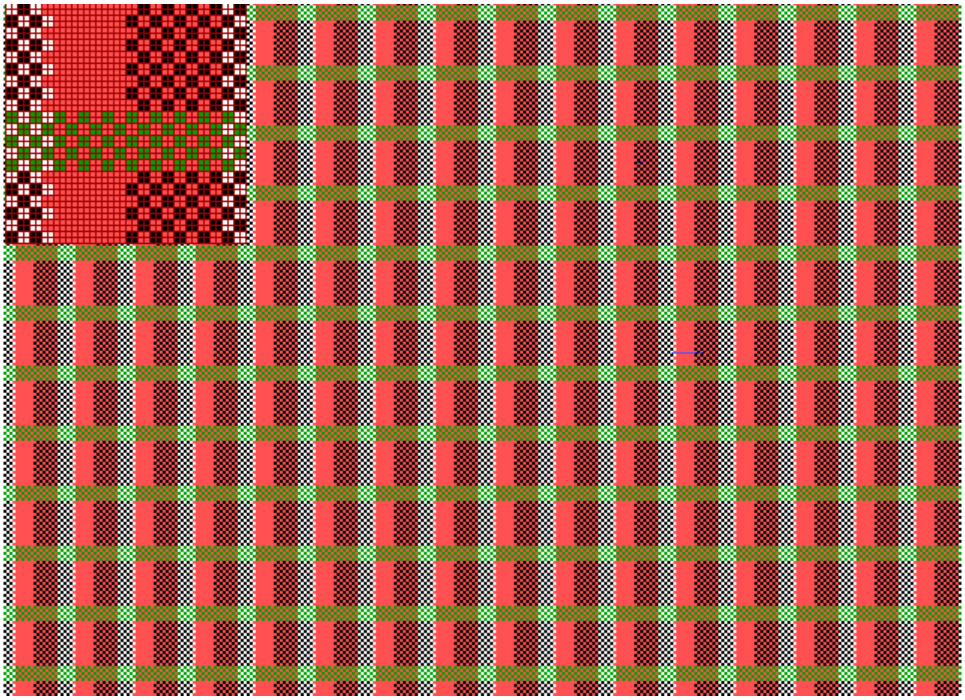
La retícula final se obtiene por simetrías de un diseño simple.



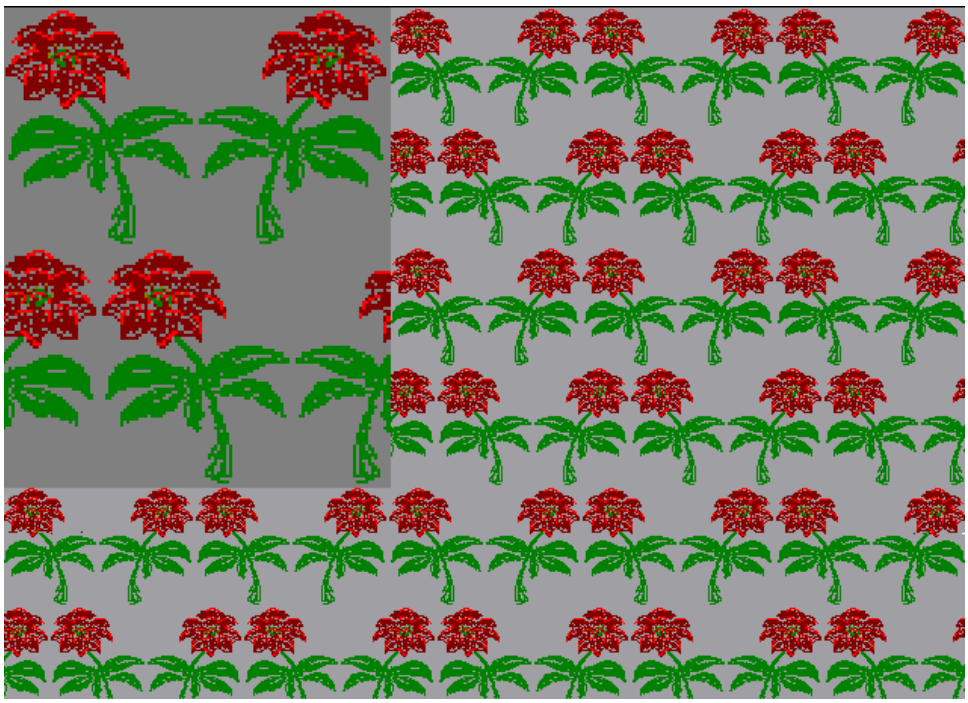
Igual que la figura anterior.



Simulación de una retícula mantel.



Motivo final obtenido mediante una simetría y traslaciones interiores.



6.6. PREVISIÓN DE FUTURO

Indudablemente las rutinas incorporadas en este trabajo no contemplan todos los aspectos imaginables de creación de una interface inteligente, ni todos los aspectos que se refieren a la tecnología del tejido, siendo muy amplio el campo que se puede incorporar a este programa, y que de forma más o menos inmediata se irán incorporando y mejorando ciertos detalles que aumenten aún más las posibilidades de diseño.

Entre las mejoras y ampliaciones posibles cabría destacar:

- La generación automática de un ligamento en base a su ley de evolución.
- Incorporación de las telas a dos caras, dobles telas y/o telas múltiples, generando los ligamentos deseados en zonas cerradas.
- Consideraciones en la simulación del tejido de la densidad de trama y densidad de urdimbre.
- Incorporación de hilos de diferentes diámetros en la retícula.
- Incorporación de dibujos capturados por escáner o procedentes de otro formato en una retícula, para posibilitar su edición, cambio de color y aplicación de movimientos.
- Insertar retículas almacenadas en disco, en otras retículas existentes.
- Adaptar los datos de la retícula al control numérico de un telar.
- En función del aumento de potencia y prestaciones de los ordenadores considerar las nuevas tarjetas de vídeo que se vayan implantando para mejorar la posibilidad de usar más colores simultáneos sin perdida de rapidez y calidad de visualización.

CAPITULO 7

CONCLUSIONES

7.1. CONCLUSIONES

Este paquete de diseño, se ha ido perfeccionando a lo largo del último año, probándose las rutinas en pequeños programas independientes. Posteriormente el paquete completo se ha facilitado a diseñadores textiles, que han mostrado su interés y han ofrecido su colaboración para detectar errores y sugerir mejoras que se han incorporado.

Tras esta primera fase interactiva de perfeccionamiento se considera esta 1ª versión como definitiva, procediéndose a partir de este momento a comprobar la bondad de dicha versión, ofreciéndose a empresas textiles que deseen colaborar en su implantación y a quienes deseen aprovechar el trabajo realizado para nuevos desarrollos de programas informáticos de diseño gráfico en cualquier disciplina.

Esta primera versión de diseño textil, por su estructura adaptable, posibilitará la creación de nuevas versiones que mejoren y sustituyan a la anterior.

Las interfaces creadas pueden acomodarse a cualquier tipo de diseño gráfico de otros sectores de la industria, al no estar pensadas exclusivamente para su uso en el diseño textil.

Estas rutinas pueden ser de gran utilidad para no especialistas informáticos que requieren programar aspectos gráficos de diseño, utilizando el ordenador personal compatible, ya que no existe información asequible y fácil de comprender que traten estas utilidades, tan elementales como necesarias, pudiéndose despreocupar de estos aspectos que no llevan incorporados los lenguajes de programación.

Se ha creado un sistema de comunicación amigable entre el usuario y el ordenador, a través de teclado y/o ratón, permitiendo la entrada de datos con desplazamiento del cursor y corrección de errores en su introducción antes de su aceptación.

La programación en lenguaje C/C++ permite la transportabilidad a otros programas de diseño en forma de objeto dentro de una librería.

Este programa informático no depende de interfaces comerciales, creyendo muy humildemente que se ha hecho un esfuerzo considerable para no caer en la tentación de aprovechar los entornos gráficos más populares pero que al final condicionan el futuro del trabajo realizado.

A partir de un motivo textil simple, con este paquete informático, se consiguen fácilmente diseños complejos de tejido, aprovechando las herramientas de transformación y composición. Igualmente se pueden manipular y mezclar colores a voluntad, viendo casi instantáneamente las distintas variedades de colorido que un mismo diseño puede aportar.

Conscientes de la existencia de paquetes de diseño textil comerciales, con este trabajo de investigación se ha abordado el diseño textil mediante ordenadores estándares, por cuanto la mayoría de esos paquetes comerciales están desarrollados sobre equipos muy potentes y con un software que no facilitan al usuario, quedando a merced del distribuidor para cuantas cuestiones, mejoras y adaptaciones se requieran.

Finalmente, la interface propuesta no pretende competir con las interfaces comerciales más sofisticadas, sino ofrecer una nueva posibilidad al personal universitario en sus labores docentes y de investigación, tanto en la vertiente de diseño de tejidos Jacquard como en el aprovechamiento de las rutinas para otro tipo de aplicación.

CAPITULO 8

BIBLIOGRAFIA

8.1. LIBROS

Tecnología del Tejido: Tomo Primero: Teoría de Tejidos.

Vicente Galcerán Escober

Ed. gráficas Costa (Terrassa), 1960.

Depósito Legal: B-9891-1960

Watson's Textile Design and Colour. Elementary Weaves and Figured Fabrics

Z. Grosicki

Edit. Newnes-Butterweords(London-Boston)- 1975

ISBN 0-408-70515-9

Watson's Advanced Textile Desing. Compound Woven Structures

Z.J.Grosicki.

Edit. Newnes-Butterweords(London-Boston)- 1977

ISBN 0-408-00250-6

Techniques du Tissage de Rubans

Hans Walter Kipp

Publié par : Fondation JTM, Frick, Suisse

Edit. Sauerländer

ISBN 3-7491-2997-0 Suisse.

Textile Graphics/ Computer Aided.

Janice R. Lourie

Fairchild Publications, Inc., New York, 1973.

Standard Book Number: 87005-108-3

Procedural Elements for Computer Graphics.

Rogers D. F.

McGraw-Hill, 1985

ISBN-0-07-Y66503-6

Gráficas por computadora.

Donald Hearn y M. Pauline Baker.

Prentice-Hall Hispanoamericana, S.A. 1989

ISBN-968-880-122-4

Traducción de ISBN 0-13-165382-2

Ilumination and Color in Computer Generated Images.

Hall R.

Springer-Verlag, 1989

Introducción a la graficación por computador.

Foley, Van Dam, Feiner, Hughes y Phillips.

Addison-Wesley Iberoamericana, 1996

ISBN-0-201-62599-7

El Lenguaje de programación C.

Brian W. Kernighan y Dennis M. Ritchie.

Prentice-Hall Hispanoamericana, S.A. 1989

ISBN-968-880-024-4

Traducción de ISBN 0-13-110163-3

Programación en Microsoft Quick C.

Waite M., Prata S., Costales B. y Henderson H.

Ediciones Anaya Multimedia S.A.

ISBN 84-7614-350-8 Depósito Legal: M-37.575-1991

Biblioteca de programas en C. Técnicas, rutina y funciones

Purdum J.J., Leslie T.C., Stegemoller A.L.

Ediciones Anaya Multimedia S.A.

ISBN 84-7614-116-5 Depósito Legal: M-39.017-1986

Técnicas avanzadas en C. Desarrollo de aplicaciones

Gerald E. Sobelmab, David E. Krekelberg.

Ediciones Anaya Multimedia S.A.

ISBN 84-7614-117-3 Depósito Legal: M-39.014-1986

Programación en C/C++.

E. Hernández Orallo y J. Hernández Orallo

Editorial Paraninfo S.A.

ISBN 84-283-2021-7 Depósito Legal: M-18.252-1993.

Curso de programación C/C++.

Fco. Javier Ceballos Sierra

RA-MA Editorial

ISBN 84-7897-200-5 Depósito Legal: M-27061-1995.

Microsoft C/C++7.

William H. Murray y Chris H. Pappas

McGraw-Hill /Interamericana de España, S.A.

ISBN 84-481-1924-X Depósito Legal: M-12.972-1994.

CAD aplicado al diseño ornamental.

J. Mumbrú Laporta, M. Ochoa Vives y M. Orellana Almendras

Ediciones UPC Colección AULA TEORICA nº 40

ISBN 84-7653-552-X

Depósito Legal: B-25.584-95.

Fundamentos de los gráficos con ordenador.

J. A. Sellarés i Chiva

Ediciones y Distribuciones Universitarias S.A. (EDUNSA)

Colección INFORMATICA BASICA nº 13

ISBN 84-7747-012-X Depósito Legal: B-851-1988.

Curvas planas y dibujo por ordenador.

Victor Villoria San Miguel

Editorial DOSSAT, S.A.

ISBN 84-237-0808-X Depósito Legal: M-27234-1992.

La sintaxis de la imagen.

D. A. Dondis

Editorial Gustavo Gili, S.A.

ISBN 84-252-0609-X Depósito Legal: B-26.445-1994.

Diseño. Historia, teoría y práctica del diseño industrial.

Bernhard E. Bürdek

Editorial Gustavo Gili, S.A.

ISBN 84-252-1619-2 Depósito Legal: B-34.477-1994.

Fundamentos del diseño.

Wucius Wong

Editorial Gustavo Gili, S.A.

ISBN 84-252-1643-5 Depósito Legal: B-1.835-1995.

Las Formas del Color.

Karl Gerstner

Ed. Hermann Blume

ISBN 84-7214-396-1 Depósito Legal: M-3.129-1988.

8.2. ARTICULOS EN REVISTAS

La simulación de tejidos por ordenador

Mumbrú J., Veciana F.

Revista de Quimica Textil, Abril-Junio 1986

Principios en que se basa el diseño de tejidos por ordenador.

J. Mumbrú Laporta.

Técnica Textil Internacional vol 32 nº 2 1988.

ISSN: 0040-1900 Depósito Legal: B-14.077-1966

Los sistemas CAD como herramientas de apoyo al diseño y fabricación de tejidos Jacquard.

J. Mumbrú Laporta.

Revista de la Industria Textil, Número 268, 1989

ISSN: 0210-0800 Depósito Legal: B-6624/1959

Introducción al diseño de tejidos por ordenador.

Hernández Abad F.; Capdevila Juan X.

Novática vol XV, nº 83, año 1989

ISSN: 0211-2124 Depósito Legal: B-15.154-1975

Representación por ordenador de los ligamentos I.

Hernández Abad F.; Capdevila Juan X.

Técnica Textil Internacional vol 34 nº 1 1990.

ISSN: 0040-1900 Depósito Legal: B-14.077-1966

Representación por ordenador de los ligamentos II.

Hernández Abad F.; Capdevila Juan X.

Técnica Textil Internacional vol 35 nº 5 1991.

ISSN: 0040-1900 Depósito Legal: B-14.077-1966

Representación por ordenador de los ligamentos III.

Hernández Abad F.; Capdevila Juan X.

Técnica Textil Internacional vol 36 nº 6 1992.

ISSN: 0040-1900 Depósito Legal: B-14.077-1966

The Geometry of Regular Repeating Patterns.

Hann M.A., Thomson G.M

Textile Progress, vol 22, nº 1, año 1992

Colorimetría: El ojo y la visión del color (I).

Valldeperas J.

Técnica Textil Internacional vol 39 nº 3 1995.

ISSN: 0040-1900 Depósito Legal: B-14.077-1966

Colorimetría: El ojo y la visión del color (II).

Valldeperas J.

Técnica Textil Internacional vol 39 nº 4 1995.

ISSN: 0040-1900 Depósito Legal: B-14.077-1966

Colorimetría: El ojo y la visión del color (y III).

Valldeperas J.

Técnica Textil Internacional vol 39 nº 5 1995.

ISSN: 0040-1900 Depósito Legal: B-14.077-1966

Clasificación y Tratamiento Matricial en el Diseño de Tejidos. Su aplicación al CAD Textil. 1ª Parte.

Mumbrú J., Ochoa M., Orellana M.

Revista de la Industria Textil, Número 321, Octubre 1994

ISSN: 0210-0800 Depósito Legal: B-6624/1959

Clasificación y Tratamiento Matricial en el Diseño de Tejidos. Su aplicación al CAD Textil. 2ª Parte.

Mumbrú J., Ochoa M., Orellana M.

Revista de la Industria Textil, Número 322, Noviembre 1994

ISSN: 0210-0800 Depósito Legal: B-6624/1959

Clasificación y Tratamiento Matricial en el Diseño de Tejidos. Su aplicación al CAD Textil. 3ª Parte.

Mumbrú J., Ochoa M., Orellana M.

Revista de la Industria Textil, Número 323, Diciembre 1994

ISSN: 0210-0800 Depósito Legal: B-6624/1959

Clasificación y Tratamiento Matricial en el Diseño de Tejidos. Su aplicación al CAD Textil. 4ª Parte.

Mumbrú J., Ochoa M., Orellana M.

Revista de la Industria Textil, Número 335, Febrero 1996

ISSN: 0210-0800 Depósito Legal: B-6624/1959

Clasificación y Tratamiento Matricial en el Diseño de Tejidos. Su aplicación al CAD Textil. 5ª Parte.

Ochoa M., Mumbrú J., Cot M.

Revista de la Industria Textil, Número 344, Enero 1997

ISSN: 0210-0800 Depósito Legal: B-6624/1959

Aplicación de la geometría fractal en el diseño de estampados textiles.

Hernández F., Palominos P., Orellana M., Ochoa M.

Técnica Textil Internacional, Vol 39 Núm 5, Sep-Oct 1995

ISSN: 0040-1900 Depósito Legal: IB-14.077-1966

Los fractales y su contribución al diseño de estampados textiles".

Hernández F., Palominos P., Orellana M., Ochoa M.

Boletín INTEXTER, Número 109, Enero-Julio 1996

ISSN: 01131-6756 Depósito Legal: B-27042-1990

La utilización del "Plotter" en el diseño textil.

Mumbrú J., Cot M.

Revista de la Industria Textil, Número 342, Noviembre 1996

ISSN: 0210-0800 Depósito Legal: B-6624/1959

8.3. ACTAS DE CONGRESOS

LIBRO DE ACTAS DEL II CONGRESO DE EXPRESION GRAFICA EN LA INGENIERIA.

La Rabida (Huelva) 1,2 y 3 Junio de 1990.

LIBRO DE ACTAS DEL III CONGRESO DE EXPRESION GRAFICA EN LA INGENIERIA.

Las Palmas de Gran Canaria 4, 5, 6 y 7 Junio de 1991.

ISBN 84-7806-049-9

LIBRO DE ACTAS DEL IV CONGRESO DE EXPRESION GRAFICA EN LA INGENIERIA.

Madrid 3 ,4, 5 y 6 Junio de 1992.

LIBRO DE ACTAS DEL V CONGRESO INTERNACIONAL DE EXPRESION GRAFICA EN LA INGENIERIA.

Gijón 2, 3 y 4 Junio de 1993.

ISBN 84-88034-22-9

LIBRO DE ACTAS DEL VI CONGRESO INTERNACIONAL DE EXPRESION GRAFICA EN LA INGENIERIA.

Toledo 26, 27 y 28 de Mayo de 1994.

ISBN 84-88248-28-8 Depósito Legal: CR-175-94.

LIBRO DE ACTAS DEL VII CONGRESO INTERNACIONAL DE EXPRESION GRAFICA EN LA INGENIERIA.

Vigo 31 Mayo, 1 y 2 Junio de 1995.

ISBN 84-88363-41-9 Depósito Legal: VG-385-95.

LIBRO DE ACTAS DEL VIII CONGRESO INTERNACIONAL DE INGENIERIA GRAFICA.

Jaén 5,6 y 7 de Junio de 1996.

ISBN 84-88924-71-0 Depósito Legal: J-341-95.

LIBRO DE ACTAS DEL IX CONGRESO INTERNACIONAL DE INGENIERIA GRAFICA.

Bilbao 4,5 y 6 de Julio de 1997.

Depósito Legal: BI-788-97.

LIBRO DE ACTAS DEL I CONGRESO ESPAÑOL DE IFORMATICA GRAFICA. CEIG '91.

Madrid, 11, 12, 13 y 14 de Junio de 1991.

LIBRO DE ACTAS DEL II CONGRESO ESPAÑOL DE IFORMATICA GRAFICA. CEIG '92.

San Sebastian, 3, 4 y 5 de Junio de 1992.

LIBRO DE ACTAS DEL III CONGRESO ESPAÑOL DE IFORMATICA GRAFICA. CEIG '93.

Granada, 2, 3 y 4 de Junio de 1993.

Depósito Legal: PM 538-1995

LIBRO DE ACTAS DEL IV CONGRESO ESPAÑOL DE IFORMATICA GRAFICA. CEIG '94.

Zaragoza, Junio de 1994.

LIBRO DE ACTAS DEL V CONGRESO ESPAÑOL DE IFORMATICA GRAFICA. CEIG '95.

Palma de Mallorca, 28, 29 y 30 de Junio de 1995.

Depósito Legal: PM 538-1995

LIBRO DE ACTAS DEL VI CONGRESO ESPAÑOL DE IFORMATICA GRAFICA. CEIG '96.

Valencia, 26, 27 y 28 de Junio de 1996.

LIBRO DE ACTAS DEL VII CONGRESO ESPAÑOL DE IFORMATICA GRAFICA. CEIG '97.

Barcelona, 25, 26 y 27 de Junio de 1997.

8.4. VARIOS

Tesis doctoral: **Consideraciones sobre Modelado y Caracterización Colorimétrica de Imágenes Digitalizadas.**

Elena Gonzalez Rodriguez

E.T.S.de I.I. de Vigo 1994

Proyecto docente e investigador: **Desarrollo del software para la generación de rutinas gráficas y su aplicación al diseño textil.**

Francisco Hernández Abad

E.T.S.I.I. de Terrassa 1992

Proyecto docente e investigador: **Desarrollo, implementación y prueba de una librería gráfica de alto nivel**

Pedro Company Calleja

Universitat Jaume I de Castellón 1996

Conferencia: **El CAD/CAM en la estampación textil "III Curso estampación"**

Josep Mumbrú Laporta

EUITIT 1986

Conferencia: **Los sistema CAD/CAM como herramienta de apoyo al diseño y fabricación de tejidos Jacquard.**

Josep Mumbrú Laporta

Jornadas sobre técnicas CAD/CAM organizadas por el Ministerio de Industria y Energia.

AITEX de Alcoy Mayo de 1989

Curso: **Aplicaçoes de sitemas CAD/CAM ao textil.**

Josep Mumbrú Laporta

Servicio de Publicaciones de la Universidad de Beira Interior.

Diciembre de 1989

Internet: **Librería de Utilidades Gráficas.** de Raul Rivero Uría

http://www3.uniovi.es/nosotros/rivero/

UNE 40-080-84

Determinación de coordenadas cromáticas “CIE”.

UNE 40-081-84

Determinación de las diferencias de color según el sistema ANLAB.

UNE 40-435-84

Determinación de diferencias de color según el sistema CIELAB.