ESCUELA SUPERIOR **POLITECNICA** DEL LITORAL

INSTITUTO DE CIENCIAS MATEMÁTICAS

INGENIERIA EN ESTADÍSTICA INFORMÁTICA

" DISEÑO **DE UN PAQUETE** GRÁFICO **COMPUTACIONAL** "

TESIS DE GRADO

Previo a la obtención del título de

INGENIERO EN ESTADÍSTICA INFORMÁTICA

PRESENTADA POR:

KARLA IVETH DONOSO CONTRERAS

Guayaquil - Ecuador
1.999

AGRADECIMIENTO

Desde lo más profundo de mi persona van mis agradecimientos ante todo a Dios por no abandonarme en ningún momento dándome siempre la fuerza y la confianza para seguir adelante. Luego a mi madre y a mi padre que no descansaron hasta verme en el sitial donde me encuentro, apunto de culminar mi carrera.

¡ Gracias por su lucha !

DEDICATORIA

Con todo mi amor dedico esta tesis a mi madre Raquel a mi padre Vicente y a mis dos hermanos Alex y Vanesa.

TRIBUNAL DE GRADUACION

Ing. **Félix** Ramírez
DIRECTOR DEL **I.C.M.**

Ing. Luis Rodríguez
DIRECTOR DE TESIS

Ing. Guillermo Baquerizo
VOCAL

Ing. Albert Espinel
VOCAL

DECLARACION EXPRESA

Autor con su firma original:

"La responsabilidad del contenido de ésta tesis de grado me corresponde exclusivamente, y el patrimonio intelectual de la misma de la Escuela Superior Politécnica del Litoral".

Presentado por:

Karla Iveth Don-oso Contreras

Guayaquil – Ecuador

1.999

RESUMEN

La programación es una salida a la búsqueda de soluciones a problemas, con la única finalidad de prestar una ayuda inmediata a usuarios que necesitan de rápidos resultados. Una de las necesidades que se íe presentan a todo usuario en su interacción con la máquina es el de contar con un sistema que brinde las características de una disciplina gráfica, y esta es precisamente la finalidad del proyecto a realizar constituído por el ***Diseño de un Paquete Gráfico Computacional*** que a pesar de la existencia de otros paquetes se caracterizará por el uso de fundamentos empleados en la programación de gráficos y la intervención de herramientas matemáticas.

INDICE GENERAL

INDICE DE FIGURAS

INDICE DE TABLAS

INTRODUCCION

El trabajo a presentarse contiene el desarrollo de un programa de dibujo que proporciona un ambiente gráfico, donde el usuario pueda interactuar. En otras palabras no es mas sino el diseño de un Paquete **Grafico,** llamado **KYC_DIBUJA (KYC_DRAW)**.

El objetivo principal en el **diseño** de este paquete es darle al usuario toda la facilidad requerida para mejorar un paquete gráfico, con no solamente usos **trradicionales** sino tambiéen con la **preseentación** de **gráficas** estadísticas de presentación.

Durante el desarrollo de los capítulos, se presentará toda la información que se necesita para llevar a cabo este objetivo. Cada capítulo contendrá sus metas específicas orientadas a crear el programa de dibujo, incluyendo así el material que tenemos a nuestro alcance proporcionado por el lenguaje TURBO **C++,** gracias a el mismo se podrá construir herramientas de dibujo derivadas de distintas funciones que poco a poco serán analizadas. Uno de las metas principales de este paquete es la de presentar una relación usuario **-** computadora con un desarrollo de aplicaciones **gráficas** interactivas para realizar líneas y dibujos. Además es posible la **creación** de **gráficas** de presentación como barras en dos y tres dimensiones y pasteles.

Este paquete constituye la primera versión, pero es posible hacer muchas mejoras, que se establecerán en las recomendaciones.

1. OBJETIVO Y ALCANCE DEL PROYECTO

El Paquete Gráfico Computacional representa una estrecha relación usuario-computadora que incluye el desarrollo de aplicaciones gráficas interactivas, por esta razón su contenido implica el desarrollo de algoritmos de graficación que proporcionen un funcionamiento completo de líneas, curvas y dibujos ya sean regulares o irregulares, además la creación de botones programados para la selección de una alternativa de acción.

Con la ayuda de un soporte matemático manejar gráficas de dos dimensiones, trasladarlas, moverlas por medio de cambios de escala y la oportunidad de rotarlas. Para el diseño de este paquete será indispensable la intervención del soporte gráfico y de gráficos orientados a objetos de Turbo C++.

Entre las **aplicaciones del paquete** está un manejo de estadística descriptiva, con la creación de gráficas como barras, gráficas de pastel, y barras tridimensionales. Además la creación de figuras con la oportunidad de pintarlas y moverlas.

1.1 Referencias Generales sobre el contenido para el diseño del paquete gráfico

Para lograr el diseño de un paquete gráfico es necesario tomar en cuenta temas básicos de programación, comenzando con una introdución a la programación de gráficos que cubre funciones básicas que Turbo C++ suministra para el tratamiento de éstos. Donde la utilización de Turbo C++, cubre lo que se refiere a funciones de biblioteca y sus extensiones.

Las aplicaciones gráficas varían en materia y tipo, siendo de gran relevancia en nuestro proyecto los gráficos estadísticos.

Una parte también necesaria para el éxito del paquete son los gráficos orientados a objetos en Turbo C++, pues presenta la técnica de la programación orientada a objetos gráficos. Además de la presencia de diversos objetos de control como botones que ayudarán a activar funciones gracias a la interacción del usuario con la máquina por medio del ratón, de esta manera, el ratón se convierte en un dispositivo de entrada de gran uso con la única finalidad de prestar al usuario mayor facilidad.

2. INTRODUCCION A LA PROGRAMACION GRAFICA

2.1 Antecedentes de la programación gráfica

Al darse inicio a la era de los computadores existieron grandes dificultades por la necesidad de pantallas gráficas, pues la capacidad y características de cada computadora no eran suficientes, los conjuntos de caracteres gráficos basados en memoria ROM eran la única alternativa de juego de caracteres ASCII, existía poca oferta de acceso directo a la memoria de vídeo y las limitaciones del CPU y memoria del sistema hacían complicada su utilización.

Luego con la llegada de las CPUs más potentes, los sistemas operativos, las altas velocidades y con monitores en color de alta resolución se iba solucionando poco a

poco el problema. Al principio la pantalla para el texto contaba de 25 líneas de 80 caracteres y no había diferencia entre este modo y el modo gráfico, estas pantallas tenían un ancho de 8 pixeles y 9 pixeles de alto y se crearon como un conjunto de caracteres de memoria ROM la cual era descrita sobre la memoria RAM Era posible que los pixeles accedieran sobre la memoria RAM del video e intercalarse con los caracteres estándar sin necesidad de cambiar el modo de representación y sin rechazar un tipo de pantalla en función de otra.

Se inició con la creación de computadoras con un sistema de vídeo realmente bajo, lo que conllevó a la creación de tarjetas de vídeo, las primeros fueron Hércules, MGCA y EGA. Pero el problema no se solucionaba allí, mas bien éste se incrementaba pues era necesario para el programador poder identificar con que adaptador estaba trabajando, pues cada uno presentaba diferentes propiedades ya sea de memoria o de pantalla. Por lo tanto se concluyó que para evitar este inconveniente todo software deberá preguntar al hardware que configuración tiene, aunque es posible que esto no se dé, por ello la creación de lenguajes como Turbo Pascal 5.5 y Turbo C++ han solucionado en gran manera el problema.

Pero esto no quiere decir que se ha descuidado a los adaptadores de vídeo sino más bien se ha tratado de conseguir un mejor rendimiento, ofreciendo nuevas posibilidades pero provocando una confusión en el momento de usar estas

herramientas. Este es efectivamente el objetivo de Programación de Gráficos en Turbo C++, el de evitar estas confusiones.

2.2 Desarrollo actual de la Programación Gráfica

PAQUETES GRAFICOS

El desarrollo actual de la computación incluye también el software de soporte gráfico. Existen paquetes que tienen módulos gráficos para el tratamiento de los datos de salida generados en ellos y que no se incluyen en esta sección, ya que se detallan dentro del software al que pertenecen.

GPR100 es una biblioteca de rutinas gráficas para dibujar con impresoras a gran escala (plotters) de la marca BENSON que entiendan este lenguaje.

Para su uso deberá crearse un programa fuente en cualquier lenguaje de programación, dicho programa deberá ser compilado y una vez obtenido el programa objeto se realizará la conexión con la biblioteca. En este software están incluídas la ayuda necesaria para la depuración de los programas de usuario que usan la librería, además de ficheros de comandos de ejemplo. Las rutinas se pueden dividir en cuatro subconjuntos lógicos.

Se denominan rutinas de usuario a aquellas que pueden ser invocadas por los mismos a la hora de escribir el programa en el lenguaje de alto nivel que las va a usar. 0 en lo contrario, existen otras rutinas, llamadas internas, que son invocadas por las rutinas que usan los usuarios, no pudiendo usarse directamente por éstos.

Con este software pueden obtenerse gráficas simples sobre varios tipos de dispositivos de salida, con la ventaja de que el programador puede realizar el programa para un dispositivo de salida arbitrario (de ahí, la denominación independiente del dispositivo), es decir, dejar éste sin definir, y que sea el usuario final el que le indique al programa en el momento de la ejecución qué tipo de dispositivo de salida va a usar, eligiéndolo de una lista.

Concretamente, el programa (PGPLOT) tiene a disposición del usuario rutinas para: el dibujo de líneas, polígonos, relleno de estos, realización de histogramas, mapas, etc.. Mover el cursor o pluma, según el dispositivo que se esté usando, obtener la posición del cursor, etc. Escribir textos para los ejes y, en general, en la posición deseada. Cambiar atributos del dibujo, como grosor de líneas, color de relleno, tipo de letra en los textos incluidos, etc.. Realizar operaciones en dispositivos de salida interactivos para manejar el ratón, cursor gráfico, palanca de mando, etc.

Para utilizarla es necesario hacer un programa fuente (en cualquier lenguaje de programación). Dicho programa deberá ser compilado y una vez obtenido el programa objeto se realizará el conexión con las rutinas de la biblioteca usadas de la

forma siguiente: programa-objeto ($ LINK), PGPLOT/LIBRERÍA (LIBRARY). Para ejecutar el programa bastará teclear: correr programa ($ RUN). Este software está disponible en un programa de aplicación basado en la biblioteca gráfica, (SECLUO).

PGPLOT con él se pueden dibujar tablas de datos y operar con ellas de forma interactiva. Su uso es muy cómodo, siendo su funcionamiento a base de comandos dentro de la aplicación. Esta aplicación ha sido desarrollada en el Instituto de Astrofísica de Andalucía (IAA). Está disponible en el programa de aplicación (SECLUO)

En este centro hay a disposición de los usuarios un sistema gráfico muy potente compuesto por varios productos de este fabricante. Concretamente, una biblioteca de rutinas gráficas, un editor gráfico, una aplicación para realizar interactivamente superficies de contorno y curvas de nivel, otra aplicación para realizar interactivamente diagramas de líneas, barras, etcétera, otra aplicación para manejar ficheros de segmentos y una biblioteca gráfica basada en un estándar.

Carta de Señal **(CUECHART)** .- Permite, mediante una serie de preguntas-respuestas, realizar una serie de diagramas estándar con formato. Está disponible en un programa de aplicación (SECLUO).

WINDOWS DRAW 4.0 .- Uno de los paquetes gráficos más sencillo y a la vez potentes del mercado. Incluye tres aplicaciones:

Windows Draw,

Foto Mágica (Photo Magic) y

ABC Manejador Medio (Media Manager).

Con *Windows Dibujar* (Draw) podrá dibujar cualquier cosa gracias a las 30 formas y líneas geométricas de las que dispone; además podrá simular el trazado a mano alzada con la edición de las curvas de Bézier. Igualmente podrá modificar dibujos, crear fácilmente diagramas, personalizar los atributos del texto, utilizar patrones de relleno, aplicar efectos especiales tales como mezclas y texto a lo largo de una curva, etc.

Con *Fofo Mágica (Photo Magic)* podrá escanear fotografías y capturar pantallas, para posteriormente corregirlas o editarlas, o incluso aplicarles hasta 40 efectos diferentes.

Con *ABC Manejador Medio (Manager)* podrá organizar todos los elementos gráficos que posee visualizando miniaturas de los mismo, las cuales podrá cortar, copiar, pegar,y arrastrar entre cualquiera de las aplicaciones del Windows 95. En definitiva, una interesante herramienta para todos aquellos que deseen adentrarse en el mundo de la imagen y diseño digital con sencillez y a bajo costo.

DbCAD .- Es la herramienta que le permitirá unir imágenes (vectoriales) con información almacenada en bases de datos para crear aplicaciones de CAD o gestión

de mapas desde su lenguaje favorito de programación Windows o WindowsNT. Entre sus características están el satisfacer la amplia demanda de aplicaciones que integren gestión de bases de datos con tratamiento de imágenes vectoriales y mapas de bits, DbCAD ofrece una librería completa para Windows 3.x y Windows 95.

Se trata de un conjunto de funciones diseñadas pensando en el desarrollador, que brinda las ventajas de una alta integrabilidad en aplicaciones, por una parte, y la posibilidad de gestionar toda la interfaz de usuario sin necesidad de ocuparse de detalles y, por tanto, la flexibilidad para desarrollar aplicaciones más rápida y cómodamente. DbCAD es el resultado de la amplia experiencia de la empresa ABACO como integrador de sistemas, y suministra interfaces para utilizar los recursos de la librería desde múltiples lenguajes de programación: Visual C++, Visual Basic, CA-Visual Objects, PowerBuilder, SQL Windows, Access, Foxpro, Paradox y otros.

DbCAD es compatible con la mayoría de los formatos gráficos y vectoriales del mercado (que es capaz de leer y visualizar directamente), y añade varias funciones sofisticadas que no se encuentran en la mayoría de los paquetes gráficos, tales como la posibilidad de solapar gráficos vectoriales y de mapas de bits. Además, le permite almacenar, gestionar y describir toda la información vectorial en una base de datos estándar, importándolas de otros archivos o creándolas directamente. Gracias a estas características, el desarrollador puede poner en práctica muchas ideas,

particularmente en las áreas en que es necesario enlazar con bases de datos información gráfica: Documentación Técnica, Sistemas de Información Geográfica, Diseño Paramétrico, etc.

Impresión, se ofrece un conjunto de funciones para la impresión de imágenes y texto. Estas funciones se apoyan en los controladores y tipos de letra disponibles a Windows.

Existen además otros paquetes, que han aparecido en el mercado hace muy poco tiempo que recogen versiones anteiores y están revolucionando el mercado, pues se han convertido en los más vendidos y utilizados, con un número de ususario cada vez mayor. Estos son:

. AutoCAD
- CorelDRAW

Autocad Versión 13 para Windows 95.-

Es un sucesor de alto rendimiento del sistema operativo Windows de Microsoft. Es un producto perfectamente integrado como aplicación de 32 bits para Windows 95, lo que implica una mejora del rendimiento bruto de un 20 por 100. Permite el trabajo con múltiples sesiones; los nombres de los dibujos, los archivos de símbolos de fuente o de menú, pueden contener hasta un máximo de 256 caracteres.

Además se ha mejorado la introducción de sistemas de coordenadas, de modo de selección y de referencia, se han incluido mejoras en el dibujo de sombreados, en la gestión de estilos de texto, en la acotación, y en los sólidos con nuevas órdenes para visualización y tratamiento.

CorelDRAW 7.0.-

Es la nueva versión de CorelDRAW con nuevo diseño para responder hasta al más exigente usuario. Es un paquete de dibujo muy abierto, le proporciona al usuario variedad de opciones para el dibujo y pintado, para crear todo tipo de esquemas desde los más simples hasta los que cuentan con características muy sofisticadas.
Muchos de los menús son presentados ahora en nuevas ventanas que son activadas según lo pida el usuario.

Cuenta con una ventan que sirve de tutor, dando todos las instrucciones, paso a paso de cómo realizar una tarea. Presentado un pintado de degraadado muy sofisticado. El alcance de colores sea incrementado. Se ha intereesado por aumentar una variedad de ventanas panorámicas que ayudarán al usuario a realizar su tarea con mayor efectividad.

2.3 Bibliotecas Gráficas

Se puede tener acceso a una biblioteca de funciones gráficas mediante Turbo C++ con GRAFICOS.H (GRAPHICS.H), como parte de las bibliotecas de los modelos de memoria estándar, añadiendo bibliotecas que son específicas del modelo de memoria.

Cuando utilizamos el entorno de desarrollo integrado de Turbo C, se produce la carga automática del modelo de memoria adecuado, que coincide con el modelo seleccionado. Pero la biblioteca gráfica está separada y no queda incluída automáticamente en tiempo de compilación es por esta razón que existen otras dos opciones de utilización de las funciones gráficas, la utilización de los archivos. PRJ y la incorporación de GRAFICOS.LIB (GRAPHICS.LIB) en una o más bibliotecas estándar.

Siempre se debe definir cada uno de los códigos fuente antes de utilizar cualquier función gráfica, escribiendo al inicio de cada programa (*#include < graphics.h>*).

De los dos métodos el más sencillo para incluir la biblioteca gráfica en el momento de compilar es el de archivos.PRJ, que no es más sino un programa establecido como C creare un nuevo archivo de proyecto con .PRJ, por lo que incluirá la línea (*prognomb graphics.lib)*, de donde para compilarse se seleccionará el elemento proyecto del menú desplegable y se introducirá el nombre respectivo.

En Turbo C++ en la versión de línea de órdenes (TCC.EXE),la biblioteca gráfica debe referenciarse en cuanto se ejecute dicha orden, (*tcc prognon graphics. lib).*
La biblioteca gráfica puede incorporarse a una o más bibliotecas estándar a través de la biblioteca de códigos objeto Turbo (TLIB.EXE).

La biblioteca gráfica se la puede incorporar a todas las bibliotecas de los modelos de memoria creándose y ejecutándose el archivo.BAT.

Es posible enlazar los controladores gráficos y las fuentes gráficas antes de incorporar la biblioteca gráfica.

2.4 Funciones de Pantalla

Los modos de las pantallas gráficas gozan de las mismas características de control que los de las pantallas de texto, como son la creación de ventanas y las rutinas de tratamiento en la pantalla. No todas las funciones tienen un equivalente en el modo de texto. Por otra parte, no todas las funciones del modo de texto tienen la función equivalente en el modo gráfico. Puede suceder que algunas de las funciones parezcan similares a sus equivalentes en el modo de texto, pero pueda que no funcionen de la misma manera.

Existen funciones que afectan tanto a la pantalla como a la ventana gráfica las siguientes son las más conocidas:

Cleardevice ( ).-

Esta función borra la totalidad de la pantalla gráfica, independientemente de los parámetros de la ventana gráfica, desplazando la posición vigente, a la posición (0,0). Esto no afecta a la ventana gráfica activa. Los parámetros de la ventana gráfica no sufren ninguna modificación. Así mismo la función no devuelve ningún valor ni genera algún tipo de error. El comportamiento de esta función es muy parecida a la función de texto, permite el borrado exclusivo (clrscr) de la ventana activa, provocando la restructuración de la totalidad de la pantalla gráfica activa, sin afectar el resto de las pantallas gráficas. Las funciones de texto, como borrado exclusivo, no funcionan en modo gráfico y viceversa.

Clearviewport ( ).-

Borra la ventana gráfica vigente y define como nueva posición vigente la posición (0.0) de la ventana gráfica. A diferencia de la función cleardevice, clearviewport se limita a una zona específica de la pantalla y actúa de forma similar a la orden de borrado (clrscr) sobre la ventana activa.

Setviewport ( ).-

Esta función permite marcar la ventana gráfica activa, siendo su actividad equivalente a la de la función de la ventana de texto. Las coordenadas (izquierda, arriba derecha, abajo) son absolutas en el entorno de la pantalla y solo afectan a la página gráfica activa.

El quinto argumento especificado en la llamada a poner el viewport (setviewport) es dentrolimite, si dentrolimite tiene un valor distinto de cero (es cierto), todos los dibujos quedan restringidos al ámbito de la ventana gráfica activa. Si dentrolimite vale cero, los dibujos pueden extenderse más allá del perímetro de la ventana gráfica, sin ningún tipo de límite.

Los límites de una ventana gráfica no afectan a las órdenes de obtener la imagen (getimage) y poner la imagen (putimage). Las imágenes que vayan a representarse en pantalla a nivel de pixel no quedarán truncadas en el perímetro de la ventana gráfica, independientemente de los valores de dentrolimite.

Si las coordenadas que se pasan a la función de setviewport no son válidos, el resultado gráfico devolverá un valor de error, lo que permitirá que los parámetros utilizados anteriormente se activen en la ventana gráfica. Para inicializar la ventana gráfica vigente con los valores en la pantalla usamos inicializar gráficos (initgraph) o poner modo gráfico (setgraphmode).

Getviewsettings ( ).-

La variable estructurada ventana gráfica devuelve las coordenadas de la ventana gráfica vigente y los valores de dentrolimite. Las coordenadas así especificadas corresponden a valores absolutos de pantalla.

Para valores de dentrolimite distintos de cero, los dibujos resultan truncados en los márgenes de la ventana gráfica vigente.

2.5 Trabajo con coordenadas

Para desarrollar elementos gráficos, debemos tomar en cuenta su posición para ello usamos las coordenadas de pixel. Los pixeles son los pequeños puntos que se ven en la pantalla que pueden ser direccionados mediante el uso de sistemas de coordenadas similar al que se utiliza cuando se acceden los renglones y columnas en una tabla.

El sistema de coordenadas es llamado ***sistema de coordenadas cartesianas.*** Estas coordenadas son inherentemente dimensionales y por lo tanto pueden definirsse objetos en término de unidades que son naturales para las aplicaciones y para el usuario. Estas coordenadas orientadas a aplicaciones o usuario son llamadas coordenadas usuario y se conoce como un mundo de coordenadas al intervenir

objetos en dos dimensiones y tres dimensiones La amplitud de un sistema de coordenadas depende del adaptador de video que se utilice.

Para el desarrollo de aplicaciones gráficas las líneas, rectas, y curvas son parte fundamental, sin embargo, determinadas imágenes solo pueden tener su origen en la manipulación de pixeles de forma individual.

Las funciones de manipulación de líneas rectas y curvas difícilmente pueden trabajar si no es con un procedimiento de estructura de pixels. El uso de funciones de tratamientos de pixeles a una escala superior permite salvar, reescribir y borrar imágenes completas, o combinar imágenes que ya existen en la pantalla.

La mayoría de los modos gráficos inician sus coordenadas en la esquina superior de la pantalla, con la coordenada (0,0). La coordenada de pixel con el mayor valor se encuentran en la esquina inferior derecha de la pantalla. Su valor depende del modo gráfico, pero existen funciones del Turbo C++ que se pueden llamar para pedir esos valores como: máximo valor de x (getmaxx( )), máximo valor de y (getmaxy( )) . Es posible trabajar con estas funciones y crear programas que pueden trabajar bien en varios adaptadores gráficos y diferentes modos.

Las siguientes funciones pueden accesar solamente un pixel de la pantalla a la vez.

Putpixel.-

Esta función activa el pixel especificado, lo pone en una posición determinada a un color particular

Getpixel.-

Se usa para determinar el color actual de un pixel para cualquier lugar de la pantalla, en otras palabras devuelve el índice del elemento de la paleta que corresponde o no al valor real del color.

3. SOPORTE MATEMATICO

3.1 Transformación geométrica

Este capítulo es una referencia de cómo transformar los objetos representados en coordenadas hacia la pantalla. Lo que se busca es manipular objetos para poder ser dibujados en diferentes ubicaciones, orientaciones y escalas, efectos que serán producidos gracias a las transformaciones geométricas.

Las transformaciones usadas son operaciones matemáticas básicas para el uso en la programación gráfica por lo que formará parte preponderante en el diseño del paquete. Una transformación es una función o ecuación que define como se debe cambiar o transformar un conjunto de datos en otro. De esta manera podemos provocar por medio de la transformación que una imagen sea afectada ya sea por:

- Traslación
- Rotación
- Cambio de Escala

Analizaremos cada uno de estos efectos en la imagen, la manera de cómo ejecutar estas operaciones serán narradas en capítulos posteriores.

Podremos hacer uso de estas transformaciones en dos dimensiones y en tres dimensiones para generar vista en perspectiva de objetos tridimensionales.

Traslación.-

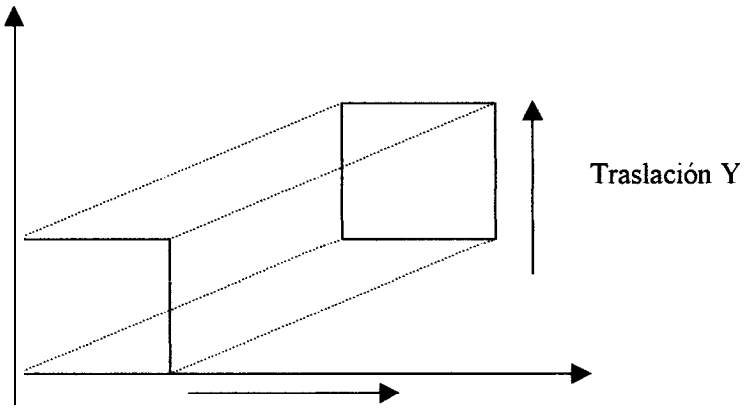
Para llevarla a cabo se necesita de una transformación sencilla. Básicamente una traslación define cómo se supone que debe moverse un punto de un lugar en el espacio a otro. Se especifica como podemos mover cada punto para que un objeto sea trasladado. La traslación de un pixel en la pantalla se puede hacer sumando un valor apropiado a cada una de las coordenadas x y y para que se mueva un punto.

$$x' = x + Dx$$

$$y' = y + Dy$$

Figura 3.1

Traslación de un objeto



Cambio de Escala.-

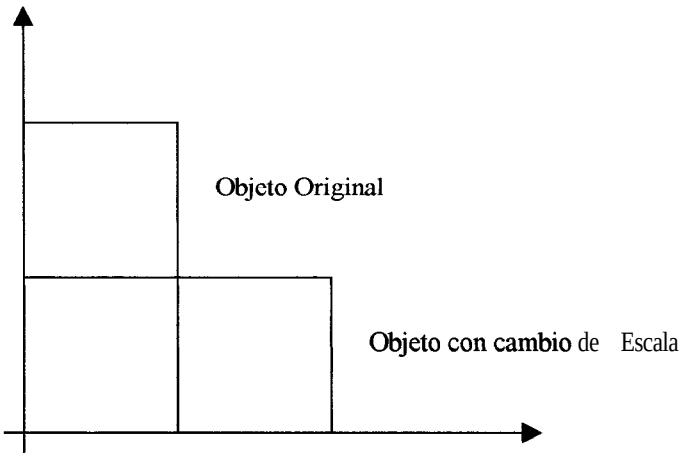
Para el cambio de escala de un polígono de dos dimensiones es necesario multiplicar cada una de las coordenadas del polígono original por un factor de escala. Si quisiéramos cambiar la escala del cuadro de la figura 3.1 al doble en la dirección x y a la mitad en la dirección y. Hacemos uso de ecuaciones como:

$$x' = x * factorx$$

$$y' = y * factory$$

Figura 3.2

Cambio de escala de un objeto al doble en X y a la mitad de Y



Es factible mover el objeto al mismo tiempo que se cambia la escala, a menos que se trate de (0.0), ya que esta coordenada no cambia. Por lo tanto, si quiere cambiar la escala de un objeto y al mismo tiempo mantener uno de sus puntos en la misma posición de tal manera que el objeto no se mueva, se deben dar dos pasos adicionales. En primer lugar, el objeto se debe trasladar a fin de que el punto que usted quiere mantener fijo se mueva al origen. En segundo lugar, después de aplicar las transformaciones de escala el objeto debe ser regresado a su lugar por la misma cantidad que he movido al inicio. El resultado final es que el objeto cambie de escala, pero el punto que se mueve al origen y se regresa permanece en el mismo lugar. Esta manera de cambiar de escala es más útil que la otra.

Rotación.-

Esta es otra de las transformaciones que representa girar un objeto alrededor de un punto. Las ecuaciones para ejecutar la rotación son:

$$x' = x * \cos(\text{ángulo}) - y * \sin(\text{ángulo})$$

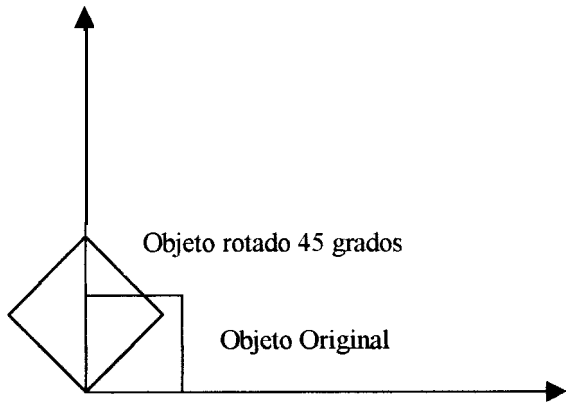
$$y' = x * \sin(\text{ángulo}) + y * \cos(\text{ángulo})$$

En vez de irnos a la geometría de donde se derivan estas ecuaciones, veámoslas desde el punto de vista del usuario. Las variables x y y del lado derecho de la ecuación representa el punto que está siendo rotado y la variable ángulo en que especifica el ángulo en que se rota el punto.

Para rotar con éxito un objeto necesitamos tomar un punto sobre el cual sea factible girar el objeto, trasladarse al origen, rotar el polígono y luego regresar a donde estaba.

Fig 3.3

Rotación de un polígono



Transformación deformante.-

Existen transformaciones que provocan efectos, una de ellas es la deformación en los objetos en los cuales esta transformación es aplicada. Donde se multiplica las coordenadas del objeto por un factor de escala y la adición de un desplazamiento.

Las ecuaciones que producen este efecto son:

$$x' = x + c * y$$

$$y' = d * x + y$$

Como ocurre en las rotaciones, lo que queremos es deformar un polígono alrededor de un punto, entonces debemos trasladar el polígono al origen, aplicar la transformación deformante y después regresarlo. Además si transformamos pixeles en la pantalla necesitamos compensar por la relación de aspecto del monitor

3.2 Representación Matricial

Los puntos en el plano xy pueden ser trasladados a nuevas posiciones añadiendo una cierta cantidad de puntos a las coordenadas de inicio. Para cada punto **P(x,y)** que ha sido movido Dx unidades paralelas a el eje x y Dy unidades paralelas al eje y para el nuevo punto **P'(x',y')** se puede escribir:

x'= x + Dx

y' = y + Dy

Por lo tanto podemos definir como vectores

P = [x Y 1

P' = [x' y']

T = [Dx Dy]

Entonces, P' = P + T

Un objeto puede ser trasladado aplicando a cada punto del objeto las ecuaciones: $x' = x + Dx$, $y' = y + Dy$. Porque cada línea en un objeto es hecha por un número infinito de puntos sobre una línea que puede ser trasladada por traslación únicamente de los puntos finales y dibujando una nueva línea con el traslado final de puntos.

Los puntos pueden ser escalados con Sx a lo largo del eje x y con Sy a lo largo del eje y introduciendo nuevos puntos por las multiplicaciones:

$x' = x \cdot Sx$

$y' = y \cdot Sy$

Definimos a S en forma matricial como $\begin{bmatrix} Sx & 0 \\ 0 & Sy \end{bmatrix}$, por lo que podemos escribir estas operaciones en forma de matriz,

$$[x' \quad y'] = [x \quad y] \cdot \begin{bmatrix} Sx & 0 \\ 0 & Sy \end{bmatrix}$$

o también,

$P' = P \cdot S$

La escala de un punto es a partir del origen. Los puntos pueden ser rotados a través de un ángulo θ sobre el origen. Podemos definir la notación matemáticamente como:

$x' = x.\cos\theta - y.sin\theta$

$y' = x. .sin\theta + y.\cos\theta$

En forma de matriz, tenemos

$$[x' \quad y'] = [x \quad y]\begin{bmatrix} \cos\theta & sin\theta \\ -sin\theta & \cos\theta \end{bmatrix}$$

0 en forma

$P' = P . R$

Primero analizaremos la escala y rotación sobre el origen, luego nos referiremos a rotación sobre un punto arbitrario.

Los ángulos positivos son medidos contrarios a la manecilla del reloj desde x hacia y, en cambio en el caso de ángulos negativos son con la trayectoria igual al de las manecillas del reloj. En este caso nos van a hacer de mucha ayuda las identidades

$$\cos(-\theta) = \cos\theta \; y \; \text{sen}(-0) = -sin\theta$$

Una rotación de θ transformaciones $P(x,y)$ dentro de $P'(x',y')$. Porque la rotación es sobre el origen, las distancias de el origen a P y a P' son iguales y son etiquetados con r en la figura. Por simple trigonometría, podemos notar que:

$$x = r\cos\phi$$

$$y = r\,\text{sen}\,\phi$$

Y

$$x' = r\cos(\theta + \phi) = r\cos\phi\cos\theta - r\,\text{sen}\,\phi\,\text{sen}\,\theta,$$

$$y' = r\,\text{sen}(\theta + \phi) = r\cos\phi\,\text{sen}\,\theta + r\,\text{sen}\,\phi\cos\theta.$$

3.2.1 Coordenadas Homogéneas y Representación Matricial de Transformaciones en 2 dimensiones

Las matrices que representa como hacer una traslación, un cambio de escala, y una rotación son las siguientes:

$P' = P + T$

$P' = P \cdot S$

$P' = P \cdot R$

Desafortunadamente, la traslación se resuelve como una suma, a diferencia que la rotación y el cambio de escala que es resuelta por medio de una multiplicación. Pero es posible que combinemos todas estas transformaciones.

Si nosotros expresamos puntos en coordenadas homogéneas, todas las transformaciones pueden ser tratadas como multiplicaciones. Las coordenadas homogéneas fueron desarrolladas en geometría y tienen consecuentemente gran aplicación en gráficos. Numerosas subrutinas de paquetes gráficos y algunos procesos muestran el trabajo con coordenadas homogéneas y transformaciones. En algunos casos el programa de aplicación las usa directamente para paso de parámetros hacia el paquete gráfico, mientras en otros casos son aplicados únicamente sin el paquete y no son visibles para el programador.

En coordenadas homogéneas, el punto P(x,y) es representado como P(W . x , W . y, W) para algún factor de escala $w \neq 0$. Entonces se puede obtener una representación de coordenadas homogéneas para cada punto P(X, Y, W), podemos encontrar la representación de coordenadas cartesianas en 2D para el punto $x = X / Y$ y $y = Y / W$. Podemos denotar a W como 1, por lo tanto la división no será requerida. Se puede pensar como encajar las coordenadas homogéneas en un plano de dos dimensiones , escalarlo por W, en el plano z = W (en este caso z = 1)

En este caso los puntos son representados ahora por 3 elementos del vector fila, así la transformación de matrices, que multiplica un punto de un vector produce otro punto en el vector, debe ser de 3 x 3. La forma de una matriz para coordenadas homogéneas de 3 x 3, está representada como:

$$[x' \quad y' \quad 1] = [x \quad y \quad 1] \begin{bmatrix} 1 & 0 & 0 \\ Dx & Dy & 1 \end{bmatrix}$$

y también puede ser expresado como,

P' = P T(Dx , Dy),

Donde

$$T(\text{Dx, Dy}) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ Dx & Dy & 1 \end{bmatrix}$$

Si un punto P es trasladado $(Dx_1$, Dy,)a P' y estos son trasladados por $(Dx_2,$ Dy,) a P, el resultado será una traslación de $(Dx_1 + Dx_2, Dy_1$ + Dy,) siguiendo el siguiente procedimiento:

$$P' = P.T(Dx_1, Dy_1),$$

$$P'' = P'.T(Dx_2, Dy_2).$$

Sustituyendo la primera dentro de la segunda, entonces se obtiene:

$$P'' = (P \,.\, T(Dx_1, Dy_1) \cdot T(Dx_2, Dy_2)) = P \,.\, (T(Dx_1, Dy_1) \cdot T(Dx_2, Dy_2))$$

El producto matriz de $T(Dx_1,\ Dy_1) \cdot\ T(Dx_2,\ Dy,)\ es,$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ Dx_1 & Dy_1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ Dx_2 & Dy_3 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ Dx_1 + Dx_2 & Dy_1 + Dy_2 & 1 \end{bmatrix}$$

Entonces la traslación $(Dx_1 + Dx_2, Dy_1 + \text{Dy,})$. El producto matriz se refiere básicamente a como es comprendido, catenado, concatenado o compuesto de $T(Dx_1, Dy_1) \cdot T(Dx_2, Dy_2)$.

Las ecuaciones de escala explicadas anteriormente como :

x' = x . Sx

y' = y . Sy

Son representadas en forma de matriz de la siguiente manera,

$$[x' \quad y' \quad 1] = [x \quad y \quad 1] \cdot \begin{bmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \end{bmatrix}$$

Definiendo,

$$S(Sx, Sy) = \begin{bmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \end{bmatrix}$$

De lo cual podemos obtener,

$$P' = P \;.\; S(Sx, Sy)$$

Como únicamente la traslación es aditiva, cuando se realiza un cambio de escala se debería multiplicar. Dando como resultado,

$$P' = P \;.\; S(Sx_1, Sy_1),$$

$$P'' = P'.S(Sx_2, Sy_2),$$

Substituyendo la primera dentro de la segunda, se obtiene

$$P' = (P \cdot S(Sx_1, Sy_1) \cdot S(Sx_2, Sy_2)) = P \cdot (S(Sx_1, Sy_1) \cdot S(Sx_2, Sy_2))$$

El producto matriz de $S(Sx_1, Sy_1) \cdot S(Sx_2, Sy_2)$ es,

$$\begin{bmatrix} Sx_1 & 0 & 0 \\ 0 & Sy_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} Sx_2 & 0 & 0 \\ 0 & Sy_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} Sx_1 \cdot Sx_2 & 0 & 0 \\ 0 & Sy_1 \cdot Sy_2 & 0 \cdot \\ 0 & 0 & 1 \end{bmatrix}$$

Por lo tanto las escalas son inicialmente multiplicativas.

Finalmente, refiriéndonos a rotación haciendo uso de la ecuación establecida,

$$x' = x.\cos\theta \quad y.sin\theta$$

$$y' = x.\ .sin\theta + y.\cos\theta$$

Podemos representarla como,

$$[x' \quad y' \quad 1] = [x \quad y \quad 1] \cdot \begin{bmatrix} \cos e & sen\ e & 0 \\ - sen8 & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Tomando

$$R(\theta) = \begin{bmatrix} \cos\theta & sen\,\theta & 0 \\ -sen\,\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

Obtenemos,

$P' = P\ .\ R(0).$

Si quisiéramos lograr dos rotaciones sucesivas, serían por medio de una adición. Considerando una submatriz de 2 x 2 que contiene senos y cosenos, tomamos cada una de las dos columnas como vectores. El vector deberá seguir tres propiedades:

1. Cada una es un vector unitario
2. Cada una es perpendicular a la otra (su producto punto es cero)
3. La dirección especificada de los vectores son rotadas por R(8) así como para estar sobre el eje positivo x y y .

Las primeras dos propiedades son también verdaderas para las filas de la submatriz de 2 x 2. Las dos direcciones son para que los vectores sean rotados a lo largo del eje x y del eje y.

3.2.2 Composición de transformaciones en 2D

Mostraremos cómo la composición puede ser usada para combinar la matriz fundamental de R, S y T para producir así los resultados deseados. El propósito básico de la composición de transformaciones es que representa el método más eficiente para aplicarlo a una transformación compuesta simple de un punto que para aplicarlo a una serie de transformaciones, una después de la otra.

Si consideramos la rotación de un objeto sobre algún punto arbitrario Pr ya que solo conocemos como rotar sobre el origen, convertimos el problema original en tres distintos problemas.

Así para rotar sobre Pr, una secuencia de tres transformaciones fundamentales es necesario:

1. Trasladar de tal manera que Pr este en el origen
2. Rotar
3. Trasladar de tal manera que el punto de origen regrese a Pr

La transformación lograda es:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -x_1 & -y_1 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos\theta & \operatorname{sen}\theta & 0 \\ -\operatorname{sen}\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ x_1 & y_1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos\theta & \operatorname{sen}\theta & 0 \\ x_1(1-\cos\theta) + y_1 \operatorname{sen}\theta & y_1(1-\cos\theta) - x_1 \operatorname{sen}\theta & 1 \end{bmatrix}$$

Esta composición de transformaciones por la multiplicación de matrices es un ejemplo de cómo coordenadas homogéneas pierden simplicidad.

Algo similar es realizado para escalar un objeto sobre un punto arbitrario $\mathbf{P_1}$; trasladar $\mathbf{P_1}$ al origen, escalar, regresar a $\mathbf{P_1}$. En este caso la transformación requerida es:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -x_1 & -y_1 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos\theta & \text{sen}\,\theta & 0 \\ -\text{sen}\,\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ x_1 & y_1 & 1 \end{bmatrix} =$$

$$\begin{bmatrix} & & \\ & & \\ x_1(Sx\theta - Sx) & y_1(Sy\theta - Sy) & 1 \end{bmatrix}$$

Supongamos que nosotros deseamos escalar, rotar y posicionar un objeto, siendo $\mathbf{P_1}$ el centro para la rotación y el cambio de escala. La secuencia debería ser trasladar $\mathbf{P_1}$ a el origen, ejecutando el cambio de escala y la rotación, entonces se traslada del origen a la nueva posición $\mathbf{P_2}$. Una estructura de datos que registra esta transformación debe contener los factores de escala, el ángulo de rotación, y que traslación se desea realizar, o debe simplemente registrar la matriz de la transformación compuesta:

$$T(-x_1, y_1) \,.\, S(Sx, Sy) \cdot R(\theta) \cdot T(x_2, y_2)$$

Dado que M_1 y M_2 cada una representan una traslación fundamental, un cambio de escala, o una rotación, cuando se produce la multiplicación de matrices $M_1 . M_2 = M_2 . M_1$ no conmutativas. Sin embargo, puede ocurrir que la conmutatibilidad se mantenga en casos especiales, como:

M_1	M_2
Traslación	Traslación
Cambio de Escala	Cambio de escala
Rotación	Rotación
Escala con Sx = Sy	Rotación

En estos casos no debemos interesarnos del orden para la multiplicación de las matrices.

La composición más general de R, S, y operaciones de T producirán una matriz de la forma:

$$M = \begin{bmatrix} r_{11} & r_{12} & 0 \\ r_{21} & r_{22} & 0 \\ t_x & t_y & 1 \end{bmatrix}$$

La parte superior izquierda de 2 x 2 es una rotación compuesta y una matriz de escala, mientras el t_x y t_y son traslaciones compuestas. Calculando P . M como un vector una matriz de 3 x 3 toma nueve múltiplos y seis sumandos. La estructura fija de la última columna de la matriz M simplifica la operación a:

$$x' = x \cdot r_{11} + y \cdot r_{21} + t_x$$

$$y' = x \cdot r_{12} + y \cdot r_{22} + t_y$$

Lo que provoca la reducción del proceso a cuatro múltiplos y cuatro sumandos, una significante aceleración, especialmente porque la operación debe ser aplicada a cientos o exactamente doscientos de puntos por pintura. Así mientras matrices de 3 x 3 son convenientes y útiles para transformaciones compuestas de 2D, la matriz final puede ser aplicada más eficientemente para reconocer esta estructura especial.

Existen demás áreas donde es importante la vista de un objeto, ya sea como una molécula 0 como un avión. Si cada vista puede ser creada y mostrada cuidadosamente, entonces se logrará que el objeto rote de manera dinámica. Para llevar a cabo ésta rapidez, cada punto individual y línea del objeto debe ser transformada lo más cuidadosamente posible.

La rotación definida por las ecuaciones:

$$x' = x.\cos\theta - y.sin\theta$$

$$y' = x..sin\theta + y.\cos\theta$$

Requieren de cuatro multiplicadores y dos sumandos. Las cuales se perfeccionarán si reconocemos a causa de que θ es pequeño, entonces el cos θ es más aproximado hacia 1. Por lo que las ecuaciones de rotación pueden ser:

$$x' = x - y \operatorname{sen} \theta$$

$$y' = x \operatorname{sen} \theta + y$$

Donde se van a requerir tan solo de dos multiplicadores y dos sumandos. Pero para lograr mayor aproximación podría ser mejor remplazar x en la segunda ecuación, introduciendo la primera.

$$x' = x \quad y \operatorname{sen} \theta$$

$$y' = x' \operatorname{sen} \theta + y$$

El determinante de la matriz correspondiente de 2 x 2 es ahora 1, como son los determinantes de todas las matrices de rotación.

Las fórmulas representan una aproximación hacia el valor correcto de x' y y' , solo existirá un mínimo error. En cada momento las fórmulas son aplicadas a los nuevos valores de x y , lo que provocará que el error aumente. Repitiendo indefinidamente,

la rotación mostrada de la imagen comenzará a parecerse a un conjunto de líneas dibujadas aleatoriamente.

Pero este problema puede ser eliminado completamente, manteniendo la lista original de coordenadas (x, y). Después de cada rotación de $360°$, los datos rotados serán descartados, y los datos originales usados una vez más para restaurar la rotación. Esto significa que será necesario un espacio para la lista extra de coordenadas.

Es necesario recalcar que todas estas transformaciones geométricas son realizadas cuando trabajamos con matrices y conocer de su procedimiento es parte en el disegño de un paquete gráfico. Pero durante el diseño de KYC_DIBUJA se presentaron problemas de memoria ya que aplicar una matriz del tamaño de la pantalla ocuparía demasiada memoria de la disponible para el modo gráfico de Turbo C++.

Pero con esta base para lograr la rotación, la traslación y el cambio de escala, se emplearon operaciones muy similares para el desarrollo de ésta herramienta gráfica.

4. ALGORITMOS DE GRAFICACIÓN

Para comenzar con la programación de éste paquete gráfico iniciaremos con mencionar algoritmos de graficación, muy necesarios para lograr el diseño del paquete.

Notación algorítmica:

- Primero se deescribirán Las variables definieeendo su tipo.
- Las flechaas hacia la derecha (hacia afuera), representan las variables que se les ingresa un valor.
- Las flechas hacia la izquierda (hacia adentro), son Las que van aa tomar el valor dado en el algoritmo

- ***Para dibujar una línea:***

Se definen las variables y su tipo.

Variables

x1, x2, yl, y2 : enteros

INICIO

Los puntos iniciales son ingresados

→ x1

→ yl

Los puntos finales son ingresados

→ x2

→ y2

para i ← 0, x2 – x1, 1

x ← x1 +i

y ← ((y2-y1)/(x2-x1)) * (x-x1) + yl

FIN

Podemos entonces llamar a la función putpixel especificando los parámetros obtenidos X y Y.

- ***Dibujo de un rectángulo***

Se definen las variables y su tipo.

Variables

x1, x2, yl, y2 : enteros

INICIO

Los puntos iniciales son ingresados

⟶ x1

⟶ y1

Los puntos fínales son ingresados

⟶ x2

⟶ y2

hacer

para i ← 0, x2 – x1, 1

x ← x1 +i

y ← ((y1-y1)/(x2-x1)) * (x-x1) + yl

fin

para i ← 0, x2 , 1

x ← i

```
    y ← (y2-y1) * (x2-x) + yl
  fin

  para i ← 0, x2 – x1, 1
    x ← x1 + i
    y ← ((y2-y2)/(x2-x1)) * (x-x1) + y2
  fin

  para i ← 0, x1 , 1
    x t i
    y ← (y2-y1) * (x1-x) + yl
  fin
```

FIN

- ***Dibujo de una elipse***

Variables

x, y, ang_ini, ang_fin, a, b: enteros

INICIO

Determinar punto en el eje x primera posición del ratón

→ x1

Determinar punto en el eje y primera posición del ratón

→ y1

Determinar punto en el eje x segunda posición del ratón

→ x2

Determinar punto en el eje y segunda posición del ratón

→ y2

Valor de los ángulos inicial y final

→ ang_ini

→ ang_fin

Valor de los radios del eje x,y

a ← (x2-x1)/2

b ← (y2-y1)/2

Puntos centrales fínales

x0 ← xl+a

y0 ← yl+b

Grafique la elipse, llamando a la función

FIN

- ***Dibuja una barra en 2 dimensiones***

Se ingresan los valores de cada esquina de la pantalla de dibujo: el valor izquierdo y superior(arriba), y el valor derecho con el inferior(abajo) y se llamaran izquierdag, **derecha_p, arriba_p,** abajog

Variables

izquierdag, derechag, arribag, abajog, $X_{1..n}$,: enteros

i, n, alto, ancho, eje-x, eje-y: enteros

$Y_{1..n}$: caracteres

INICIO

→ Llamar a la función da-datos ( )

→ **n,** i, abajog, arribag, **derecha_p,** izquierdag

alto ← abajog –**arriba_p**

ancho tderechag –**izquierda_p**

eje-x ← ancho -2 / n

eje_y ← alto -2 / n+l

j ← ancho – **eje_y**

para i ← 0, n

ubicar X_i entre j ∧ alto

f ← número de puntos de la pantalla gráfica / valor máximo de Y_i

eje_x ← f * Y_i

derecha → j + eje-x

abajo → alto – c

arriba → alto

izquierda → j

llamar a la función barra a donde se ingresarán todos los valores abajo, abajo, izquierda, derecha.

fin

FIN

- ***Dibujo de una barra 3D***

  Se ingresan los valores de cada esquina de la pantalla de dibujo: el valor izquierdo y superior(arriba), y el valor derecho con el inferior(abajo) y se llamarán izquierdaq, derecha_p,arriba_p, abajoq

Variables

izquierda>, derecha_p, arriba_p, abajo-p, $X_{1..n}$, profundidad, tapa: enteros

i, n, alto, ancho, eje-x, eje_y, f, c: enteros

$Y_{1..n}$: caracteres

INICIO

→ Llamar a la función da-datos ( )

→ n, i, abajop, arribap, dereechap, izquierdap, profundidad, tapa

alto ← abajo_p –arriba_p

ancho ← derecha_p –izquierda_p

eje-x ← ancho -2 / n

eje_y ← alto -2 / n+l

j ←ancho – eje_y

para i → 0, n

ubicar X_i entre j ∧ alto

f → número de puntos de la pantalla gráfica / valor máximo de Y_i

eje-x ← f * Y_i

derecha ← j + eje-x

abajo ← alto – c

arriba ←alto

izquierda ← j

llamar a la función barra3D

fin

FIN

- ***Dibujar un pastel***

Variables

Porcentaje1 . . . . $_{n}$, $Grados_{1...n}$: enteros

$angi_{1...n}$,$angf_{1...n}$: enteros

$datos_{i...n}$: real

INICIO

→ llamar a la funcion da_datos( )

→ n, i, r, xl,yl

para i → 0, n-l

Sum → sumar los porcentajesi

$Porcentaje_i \rightarrow datos_i$ * 1 OO / Sum

$Grados_i \rightarrow Porcentaje_i$ * 3.6

$ang_0 \rightarrow 0$

$ang_0 \rightarrow Grados_0$

$angi_i$ → angi $_{i-1}$ + Grados $_{i-1}$

$angf_i$ → angf $_{i-1}$ $Grados_i$

radio → r

x-centro → xl

y-centro → y 1

Grafique el pastel, con los valores de x ,y ,radio inicial y radio final obtenidos

fin

FIN

- ***Para trasladar figuras***

Variables

x1, yl, x2, y2: enteros

Dx, Dy, xn, yn: enteros

INICIO

→ llamar a la función ejecutar( )

Coordenadas inicales de selección

→ x1

→ y1

Definir la nueva coordenada con el ratón

→ x2

→ y2

En estas coordenadas es donde se posicionará la figura

D x 3 x2-x1

x' → x1 + Dx

Dy → y2-y1

y' → y1 + Dy

Traslade la figura a las nuevas coordenadas

FIN

- *Cambio de escala*

Variables

x1, yl, x2, y2: enteros

sx, sy: enteros

INICIO

Coordenadas inicales

→ x1

→ y1

Definir la nueva coordenada con el ratón

→ x2

→ y2

→ s x

→ sy

En estas coordenadas es donde se posicionará la figura

xn → x1 *sx

yn → yl * sy

Figura cambia a las nuevas coordenadas

FIN

- ***rotar***

Variables

x1, yl, angulo: enteros

INICIO

Coordenadas inicales

→ x1

→ y1

Definir el ángulo

→ ángulo

Transformar el ángulo

radianes → angulo * 3.14159 / 180

cosang → cos (radianes)

sinang → sin (radianes)

Nuevas coordenadas rotadas

x → x1 * cosang - yl * sinang

y → x1 * sinang + y * cosang

Entonces la figura es rotada

FIN

Estos algoritmos constituyen parte esencial en el desarrollo de futuros capítulos, para el diseño de KYC_DIBUJA (KYC_DRAW).

5. SOPORTE GRAFICO DE TURBO C++

El presente capítulo lista algunas de las funciones incluidas en TURBO C++, expresa una breve descripción de la función así como su prototipo y una explicación de sus parámetros y el valor de retorno en caso de que exista.

Nota: Es importante recalcar que el presente capítulo se da el caso mencionado en el resumen, se usan varios términos en inglés que debido a que son comandos usados para la programación, no tienen traducción. Además al incluir ciertos parámetros también para la programación dentro de estas funciones, en muchos casos no han sido tildados porque en el lenguaje utilizado de programación, los parámetros no pueden ser tildados.

arc( ) .-

Dibuja un arco circular en un punto inicial especificado con un radio fijo. El arco se dibuja desde un ángulo inicial hasta un ángulo final.

arc(x, y, ai, af, radio);

x,y	El centro del arco especificado en coordenadas absolutas
ai	ángulo inicial especificado en grados
af	ángulo final especificado en grados
radio	radio especificado en pixeles

bar( ).-

Dibuja una barra rectangular rellena con el color de dibujo y el patrón de relleno actuales. La función bar( ) no dibuja un marco para la barra.

bar(izquierda, arriba, derecha, abajo);

izquierda, arriba	La esquina superior izquierda a la barra.
derecha, abajo	La esquina inferior derecha

bar3d( ).-

Dibuja una barra tridimensional rellena con el color de dibujo y patrón de relleno actuales.

bar3d(izquierda, arriba, derecha, abajo, profundidad, tapa);

izquierda, arriba	La esquina superior izquierda de la barra
derecha, abajo	La esquina inferior derecha
profundidad	La profundidad de la tercera dimensión de la barra especificada en pixeles.
tapa	Indica si se pone una tapa a la barra. Cualquier valor diferente de cero especifica que se trace la tapa.

circle ( ).-

Dibuja un círculo con un radio fijo en un punto inicial especificado. El círculo se dibuja en el color actual de dibujo los círculos sólo se pueden dibujar con el estilo de línea predeterminado. La anchura de la línea se puede poner antes de dibujar el círculo usando la función setlinestyle( ).

circle (x, y, radio);

x,y	El centro del círculo especificado en coordenadas absolutas.
radio	El radio especificado en pixeles

cleardevice ( ).-

Borra toda la pantalla gráfica y mueve la posición actuaal a (0,0) en el ventana activa. La rutina cleardevice( ) se debe usar cuando necesita borrar la pantalla,

aunque la ventana activa esté puesta a un tamaño menor que la pantalla completa. Sí solo necesita borrar la ventana actual debe utilizar la función clearviewport.

```
cleardevice ( void );
```

clearviewport( ).-

Borra ventana activa y mueve la posición actual a (0,0) en el viewport activo. Solo se borra el viewport activo. Si la ventana activa es más chico que la pantalla entera, el área fuera de la ventana no se altera.

```
clearviewport ( void );
```

closegraph.-

Termina el sistema gráfico de la BGI y libera la memoria utilizada por los manejadores gráficos, tipos de letras y buffers. La pantalla se restaura al modo en que se encontraba antes de que la BGI fuera inicializada. Usted siempre debe llamar esta rutina cuando su programa ya no necesite las rutinas de la BGI, a fin de que se pueda liberar memoria.

```
closegraph ( void );
```

detectgraph ( ).-

Verifica el hardware para determinar qué manejador gráfico y modo debe usar, Esta rutina siempre selecciona el modo que dé la más alta resolución posible para el adaptador de video instalado.

dertectgraph (*graphdriver, *graphmode);

graphdriver Regresa un código entero para indicar cual manejador gráfico se puede usar.

graphmode Regresa un código entero del modo de la más alta resolución que se puede utilizar.

drawpoly ( ).-

Dibuja un polígono empleando el color de dibujo y el estilo de línea actuales.

drawpoly(numpuntos, *polypuntos);

numpuntos El número de puntos(pares de coordenadas) usadas para dibujar el polígono.

polypuntos Un arreglo de ellos puntos para el polígono. Cada punto está representado por un par de coordenadas X y Y. Los puntos inicial y final deben ser el mismo para que el polígono se dibuje como una figura cerrada.

ellipse ( ).-

Dibuja un arco elíptico en el punto inicial especificado (centro) con radios horizontal y vertical fijos. La ellipse se dibuja desde el ángulo inicial hasta el ángulo final. Puede dibujar una elipse completa (con los extremos juntos) poniendo el ángulo inicial a 0 y el ángulo final a 360. La elipse se dibuja en sentido contrario a las manecillas del reloj, donde 0 grados está en la posición de las 3 y 90 grados está en las 12 horas. Usted no puede cambiar el estilo de línea de la elipse, pero puede modificar el espesor de la línea usando **setlinestyle( )**.

ellipse (x, y, **ai, af,** radiox, **radioy);**

x, y	El centro del arco especificado en coordenadas absolutas.
ai, af	Los ángulos inicial y final especificado en grados.
radiox, **radioy**	Los radios horizontal y vertical en pixeles

fillellipse ( ).-

Dibuja una ellipse con el patrón de relleno y color actuales. Esta función dibuja una elipse completa. Si desea dibujar una elipse parcial use la rutina ellipse( ).

fíllellipse (**x, y,** radiox, **radioy);**

x, Y	El centro del arco especificado en coordenadas absoltutas.

radiox, radioy Los radios horizontal y vertical en pixeles

fillpoly( ).-

Dibuja y rellena un polígono usando los estilos de línea y de relleno así como los colores de dibujo y de relleno actuales. Si quiere dibujar un polígono sin rellernarlo use drawpoly( ).

fillpoly(numpuntos, *polypuntos);

numpuntos El número de puntos(pares de coordenadas) usadas para dibujar el polígono.

polypuntos Un arrreglo de puntos para el polígono. Cada punto está representado por un par de coordenadas X y Y.

floodfill( ).-

Inunda una región cerrada con el color y patrón de relleno actuales. Si el punto especificado para la inundación se encuentra fuea de una región cerrada, todo, excepto de la región cerrada, se rellena.

floodfill (x, y, borde);

x, y La ubicación de la región a rellenar. Si ésta posición se encuentra dentro de una área cerrada, la región se rellena. Si la posición se

localiza hera del área cerrada, todo se rellena a excepción del área cerrada.

borde El código de color de la orilla.

getaspectratio( ).-

Determinación de aspecto actual del manejador gráfico y el modo.

getaspectratio(*xasp, *yasp);

xasp, yasp Variables usadas para regresar los valores de X y Y de la relación de aspecto.

getbkcolor( ).-

Regresa el color de fondo. El color de fondo se regresa como un código entero que puede ir de 0 a 15.

getbkcolor (void);

getcolor( ).-

Regresa el color de dibujo actual. El color de dibujo se regresa como un código entero que puede ir de 0 a 15

getcolor(void);

getdrivername( ).-

Regresa el nombre del manejador gráfico instalado actualmente.

getdrivername (void);

getgraphmode( ).-

Regresa el modo gráfico actual para el manejador gráfico instalado.

getgraphmode (void);

getimage( ).-

Copia una imagen rectangular de bits de la pantalla a memoria. La imagen se puede restaurar posteriormente llamando a putimage( ).

getimage(izquierda, arrriba, derecha, abajo, void *bitmap);

izquierda, arriba	La esquina superior izquierda del rectángulo de la pantalla.
derecha, abajo	La esquina inferior derecha del rectángulo de ella pantalla.
bitmap	Hace referencia a la posición de memoria donde se almacena la imagen de bits.

getmaxcolor( ).-

Regresa el valor máximo de color en la paleta de color activa.

getmaxcolor(void);

getmaxmode( ).-

Regresa el código entero para el modo de más alta resolución para el manejador actual

getmaxmode(void);

getmaxx( ).-

Regresa la coordenada horizontal máxima paaara el adaptador gráfico y modo activos.

getmaxx(void);

getmaxy( ).-

Regresa la coordenada vertical máxima para el adaptador gráfico y modo activos.

getmaxy(void);

getmodename( ).-

Regresa el nombre del modo gráfico activo. El nombre se regresa como un texto.

getmodename(modenumero);

modenumero Un código entero que especifica el modo gráfico.

getmoderange( ).-

Regresa el rango de modos gráficos activo para un manejador dado. El nombre se regresa como un texto.

getmoderange(graphdriver, *modoal, *modoba);

graphdriver	Código entero que hace referencia a un manejador.
modoal, modoba	Los modos alto y bajo

getpixel( ).-

Obtiene el color de un pixel en un posición especificada.

getpixel(x, y);

x, y	La posición vertical y horizontal del pixel.

getx( ).-

Obtiene la coordenada horizontal actual.

getx(void);

gety( ).-

Obtiene la coordenada vertical actual.

gety(void);

graphdefaults( ).-

Pone los atributos gráficos a sus valores predeterminados.

graphdefaults(void);

grapherrormsg( ).-

Regresa un texto de mensaje de error para el código de error regresado por graphresult( ).

grapherrormsg(errorcodigo);

errorcodigo El código de entero de error regresado por graphresult( ).

graphresult( ).-

Regresa el código de **eerror** de la última llamada gráfica inválida.

graphresult(void);

imagesize( ).-

Obtiene el número de bytes necesarios para guardar una imagen rectangular de pantalla. Si el tamaño de la imagen es mayor de 64 **K** se regresa 1.

imagesize(izquierda, arriba, derecha, abajo)

izquierda, arriba	La esquina superior izquierda a la barra.
derecha, abajo	La esquina inferior derecha

initgraph( ).-

Inicializa el sistema gráfico cargando un manejador gráfico especificado y ajustando el modo gráfico. La función initgraph( ) se debe llamar antes de usar cualquiera de las rutinas de dibujo.

initgraph (*graphdriver, *graphmode, *pathtodriver);

graphdriver	Proporciona un código entero para especificar cuál manejador gráfico se debe usar.
graphmode	Proporciona un código entero para especificar el modo gráfico.
pathtodriver	El nombre de ruta de directorio donde se encuentran guardados los archivos de los manejadores gráficos.

line( ).-

Dibuja una línea entre dos puntos usando coordenadas de pantalla absolutas. La línea se dibuja con el color y estilos de línea actuales. Para cambiar los parámetros de la línea se llama a la función **setlinestyle(**) antes de dibujar la línea.

line(x0, y0, x1, yl);

x0, y0	El extremo inicial.
x1, y1	El extremo final

linerel( ).-

Dibuja una línea desde la posición actual (CP) hacia una nueva posición usando coordenadas relativas. Después de que la línea se dibuja, el nuevo CP se determina sumando el desplazamiento (dx,dy) al CP anterior.

linerel (int dx, int dy);

dx,dy	Las distancias relativas horizontal y vertical usadas para dibujar la línea. Cualquiera de estos valores se puede especificar como un número positivo 0 negativo.

lineto( ).-

Dibuja una linea desde la posición actuaal hacia una nueva posición que está especificada en coordenadaas absolutas.

lineto(x, y);

x, Y	El punto inicial de la linea. Esta posición se convierte en la nueva posición actual después de que la linea se dibuja.

moverel( ).-

Mueve la posición actual a una distancia relativa desde el CP en el ventana activa.

moverel(dx, dy);

dx,dy	El desplazamiento para mover el CP

moveto( ).-

Mueve la posición actual a una nueva posición.

moveto(x ,y);

x, Y	La nueva posición actual.

outtext( ).-

Despliega un texto en la posición actual en la ventana activa. El texto se despliega con los parámetros actuales de tamaño, tipo de letra, dirrección y justificación.

outtext(*texto);

texto — El texto a desplegar en la ventana activa.

outtextxy( ).-

Despliega un texto en la posición especificada.

outtextxy(x, y, *texto);

x, Y — La posición en coordenadas absolutas donde se desplegará el testo.

texto — El texto a desplegar en la ventana activa.

pieslice( ).-

Dibuja y rellenaa una rebanada de pastel con el color de dibujo y patrón de relleno actuales.

pieslice(x, y, ai, af, radio);

x,y	El centro de la rebanada.
ai, af	Los ángulos inicial y final en grados.
radio	El radio en pixeles.

putimage( ).-

Copia una imagen de bits dentro de la pantaalla. Esta función junto con getimage( ) es útil para efectuar animación.

putimage(izquierda, arrriba, *bitmap, op);

putpixel( ).-

Despliega un pixel en el color especificado.

putpixel(x, y, pixelcolor);

x,y	Posición del pixel.
pixelcolor	El código de color especificado como un valor, entero.

rectangle( ).-

Dibuja un rectángulo en el estilo de línea y colores actuales.

rectangel(izquierda, arriba, derecha, abajo);

izquierda., arriba	La esquina superior izquierda.
derecha, abajo	La esquina inferior derecha.

registerbgidriver( ).-

Registra un manejador gráfico de la BGI. Un manejador se puede cargar de disco o se puede encadenar como un archivo .OBJ.

registerbgidriver(void(*manejador(void));

manejador	El nombre del manejador registrado.

registerbgifont( ).-

Registra un manejador de tipo de letra.

registerbgifont(void(*fuente)(void));

fuentee	El nombre del tipo de letra registrado.

restorecrtmode( ).-

Restaura la pantalla al modo en que se encontraba antes de que se seleccionará un modo gráfico.

restorecrtmode(void);

sector( ).-

Dibuja y rellena una rebanada de pastel elíptica con el color y patrón de relleno actuales.

sector(x, y, ai, af, radiox, radioy);

x,y	El punto central de la rebanada de pastel.
ai, af	Los ángulos inicial y final en grados.
radiox, radioy	Los radios horizontal y vertical en pixeles.

setactivepage( ).-

Asigna la página activa para el despliegue gráfico.

setactivepage(numpagina);

numpagina	El número de página que se activa.

setallpalette( ).-

Cambia todos los colores de la paleta.

setallpalette(*paleta);

paleta — Contiene la información de color de la paleta. Esta estructura de datos se presenta en la descripción de getpalette( ).

setaspectratio( ).-

Asigna los valores X y Y de la relación de aspecto para el hardware gráfico y el modo activos.

setaspectratio(xasp, yasp);

xasp, yasp — Los nuevos valores horizontal y vertical de la relación de aspecto.

setbkcolor( ).-

Asigna el color de fondo.

setbkcolor(color);

color — Un código entero que especifica el nuevo color de fondo.

setcolor( ).-

Asigna el color de dibujo activo. El número máximo de colores que están disponibles para un adaptador gráfico y modo determinados se pueden obtener mediante una llamada a getmaxcolor().

setcolor(color);

color	Un código entero (0- 15) que especifica el nuevo color de dibujo.

setfillpattern( ).-

Selecciona un patrón y color de relleno definidos por el usuario. El patrón de relleno definido por el usuario se crea utilizando máscaras de bit.

setfillpattern(*patron, color);

patron	Especifica el patrón definido por el usuario.
color	Especifica el color de relleno.

setfillstyle( ).-

Selecciona el patrón y color de relleno activos. El patrón de relleno especificado por esta rutina se emplea para dibujar elementos gráficos con las siguientes funciones: bar( ), bar3d( ), fillpoly( ), floodfill( ), pieslice( ) y sector( ).

setfillstyle(patron, color);

patron	Especifica uno de los doce patrones de relleno soportados por la BGI.
color	Especifica el color de relleno.

setgraphbufsize( ).-

Asigna el tamaño del buffer gráfico interno usado por las rutinas de la BGI tales como floodfill( ). El tamaño predeterminado es 4 K. En caso de que se utilice esta función se la debe llamar antes de que se llame a initgraph. Regresa el número de bytes del buffer definido anteriormente.

setgraphbufsize(unsigned bufsize);

bufsize	El número de bytes que serán usados por el buffer gráfico.

setgraphmode( ).-

Cambia el sistema a modo gráfico y borra la pantalla. A fin de usar esta función se debe llamar antes a initgraph( ).

setgraphmode(modo);

modo	El modo gráfico que se selecciona.

setlinestyle( ).-

Asigna la anchura y el estilo de línea actual.

setlinestyle(linestyle, unsigned upattern, thickness);

linestyle	El estilo de línea especificado como un código entero. (Tabla II)
upattern	El estilo de línea de usuario definido como una máscara de 16 bits.
thickness	La anchura de la línea. Se permiten dos estilos; NORM_WIDTH que es 1 pixel de ancho y THICK_WIDTH que equivalen a 3 pixeles de ancho.

setpalette( ).-

Cambia un color en la paleta de color activa.

setpalette(indice, color);

indice	El índice de la paleta (0- 15) (Tabla III)
color	El color que se asigna

setrgbpalette( ).-

Cambia un color en la paleta de color activa.

setrgbpalette(numcolor, rojo, verde, azul);

numcolor	La posición de la paleta a ser cambiada.
rojo, verde, azul	Los componentes del color.

settextjustify( ).-

Asigna el estilo de amplificación del texto.

settextjustify (horiz, vert);

horiz	El código para la amplificación horizontal.
vert	El código para la amplificación vertical.

settextstyle( ).-

Define los parámetros **usaddos** por la BGI.

settextstyle (fuente, dirección, tamaño);

fuente	El tipo de letra a utilizar puede ser cualquiera de los valores que se **enlistan** en la tabla III,
dirección	Dirección de las letras que se usaran cuando se trace el texto. Puede ser **HORIZ_DIR** (valor =0) para texto horizontal, o **VERT_DIR (valor=1)** para texto vertical.
tamaño	Cantidad en que se amplían los caracteres desplegados. Puede estar entre 0 y 10. Un valor de 0 sólo puede usarse con letras por trazos y le indica a la BGI que utilice el tamaño de **caracter** especificado.

setusercharsize( ).-

Define el tamaño para el despliegue de los caracteres de tipo de letra por trazos. El **tamaño** se especifica como un factor de amplificación.

setusercharsize (multx, divx, multy, divy);

multx, multy	Especifican los factores de multiplicación tanto de la altura como de la anchura.
divx, divy	Especifican los factores de división de altura y anchura.

setviewport( ).-

Asigna la ventana activa (viewport) actual.

setviewport (izqierda, arriba, derecha, abajo, clipflag);

izquierda, arriba	La esquina superior izquierda del ventana(viewport).
derecha, abajo	La esquina inferior derecha del ventana(viewport).
clipflag	Determina si los elementos gráficos se recortan en el viewport activo. Si este parámetro es diferente de cero se recortan los objetos gráficos.

setvisualpage( ).-

Asigna el número de página gráfica visual.

setvisualpage(numpag);

numpag Un código entero que especifica la página visual activa.

setwritemode( ).-

Asigna el modo de escritura para las rutinas gráficas de dibujo de líneas. El modo de escritura asignado por esta rutina afecta a las funciones de dibujo

setwritemode (modescritura);

modescritura Especifica el modo de escritura. Los modos soportados son: COPY_PUT(valor=0) y XORT_PUT(valor =1). El modo COPY_PUT se usa para sobreescribir lo que se encuentre en pantalla con la nueva línea que se dibuja. El modo XORTPUT se utiliza para efectuar una operación de OR exclusivo.

textheigh ( ).-

Regresa la altura de un texto en pixeles. El tamaño de letra actual (activa) y el factor de amplificación se usan para calcular la altura del texto.

textheigh (*texto);

texto El texto al que se calcula el tamaño.

textwidth ( ).-

Regresa la anchura de un texto en pixeles. El tamaño de letra actual (activa) y el factor de amplificación se usan para calcular la anchura del texto.

textwidth (*texto);

texto

TABLAS

Tabla I .-

Parámetros gráficos predeterminados

Atributo Gráfico	Valor predeterminado
Posición Actual (CP)	Esquina superior izquierda (0,0).
Viewport	Dimensioneas de la pantalla entera (0, 0, ancho-l, alto-1).
Paleta	La que está predeterminada para el adaptador actual.
Color de frente	El color máximo.
Color de fondo	El color 0 en la paleta.
Estilo de línea	Línea sólida.
Color de relleno	El color máximo.
Estilo de relleno	Relleno sólido.
Tamaño de la letra	1.
Dirección de la letra	Horizontal.
Justificación de la letra	Izquierda y arriba.

Tabla II.-

Códigos de estilo de línea

Valor	Nombre
0	Línea sólida (SOLID_LINE)
1	Línea Punteada (DOTTEEDLINE)
2	Líneea centro (CENTER_LINE)
3	Líneas con rayas (DASHED_LINE)

Tabla III.-

Códigos de estilo de línea

Valor	Color	Valor	Color	Valor	Color
0	Negro	5	Magenda	*10*	Verde Claro
1	Azul	6	Café	II	Gris Claro
2	Verde	*7*	Gris Claro	*12*	Rojo Claro
3	Gris	*8*	Gris Oscuro	*13*	Magenda Claro
4	Rojo	*9*	Azul Claro	*14*	Amarillo
				15	Blanco

Tabla IV.-

Códigos de justificación de texto.

Nombres macro	Valor	Descripción
Izquierda (LEFT_TEXT)	0	Texto justificado a la izquierda.
Centro (CENTER_TEXT)	1	Texto centrado (tanto para texto horizontal
Derecha (RIGHT_TEXT)	2	como vertical)
Inferior (BOTTOM_TEXT)	3	Texto justificado a la derecha.
Superior (TOP_TEXT)	4	Justificar el texto por abajo.
		Justificar el texto por arriba.

Tabla V.-

Estilo de letras

Nombre	Valor	Descripción
Bits(DEFAULT_FONT)	0	Letra por mapa de bits de 8 x 8.
(TRIPLEX_FONT)	1	Letra por trazos triplex.
Pequeña(SMALL_FONT)	2	Letra por trazos pequeña.
(SANS_SERIF_FONT)	3	Letra por trazos sans serif(estilo de letra).
Gótica(GOTHIC_FONT)	4	Letra por trazos gótica.

6. Gráficos orientados a obietos en TURBO C++

TURBO C++ retiene características como la potencia y la flexibilidad en el trato con una interfaz, con su programación del sistema a bajo nivel y con su eficiencia, economía y potentes expresiones. Lo más interesante es que se introduce en el dinámico mundo de la programación orientada a objetos y la convierte en una plataforma de abstracción del problema a alto nivel. El modelo, es posible explotarlo completamente para producir una solución orientada a objetos aplicando mucho el pensamiento para resolver problemas.

De manera que los objetos y sus operaciones asociadas deben ser identificados y todas las estructuras necesarias deben ser construidas.

Por este motivo los gráficos orientados a objetos en TURBO C++, son primordiales para lograr el diseño del paquete gráfico.

En KYC_DIBUJA se usa una estructura para trabajar con un objeto, el objeto ratón, de manera que se incluyan en ésta muchas de funciones que serán aplicadas para poder trabajar con el ratón. Con la ayuda de estos objetos podemos desechar la programación convencional para la construcción de gráficas poderosas.

TURBO C++, nos proporciona facilidad para trabajar en la construcción de gráficos, con la ayuda de objetos, tales como:

- El objeto ratón
- Objetos gráficos de control:
  - Tipo botón

6.1 El objeto ratón.-

Es necesario que para trabajar en una pantalla gráfica se haga uso de un dispositivo de entrada de datos, para ello está a nuestro alcance el objeto ratón considerado como un mecanismo principal para la selección. Para trabajar con este objeto es posible hacerlo de dos maneras: leyendo los códigos de salida del ratón e identificarlos con

las teclas del cursor, o incorporando directamente un interfaz de ratón en el programa e interpretar los sucesos provocados por el mismo.

6.1.1 Tipos de ratones

Existen dos tipos básicos de ratones:

El ratón de Microsoft (de dos botones).

El ratón de Logitech (de tres botones).

Aunque haya la presencia de diversas marcas en el mercado, gran cantidad de ellos se ajustan a estos dos estándares.

El de dos botones admite 16 funciones, con capacidad cada uno de ellos, para dar una respuesta de creación e interrupción, o bien para ser leídos cuando estén pulsados o libres de pulsación.

El de tres botones admite todas las características del ratón de dos botones, e incorpora dos funciones del ratón nueva la 16 y 19 y un tercer botón en medio.

6.1.2 Leyendo los códigos de salida del ratón

Para la realización de este método se hace uso de una interfaz por omisión del ratón que es suministrado con el mismo hardware, por lo tanto el programador puede despreocuparse de este aspecto.

Los efectos que causa el mover el ratón hacia una cierta dirección son los mismos que los producidos por las teclas direccionales, de esa manera si nosotros movemos el ratón hacia la derecha, el sistema recibe los códigos, de la misma forma que si pulsáramos la flecha direccional hacia la derecha.

De forma similar, el botón izquierdo del ratón devuelve el mismo código que se genera al pulsar la tecla enter.

6.1.3 Un interfaz del ratón

Para trabajar con el ratón en un paquete gráfico lo más conveniente es crear un interfaz para el mismo que permita al programa cargar sus parámetros y leer directamente los sucesos y posición de éste, puesto que se presentan diversas dificultades en la interpretación indirecta de la entrada generada por un ratón, como por ejemplo el desplazamiento del ratón y las respuestas de las teclas puede que no estén asignados correctamente, y la información devuelta se hace menos completa que la leída directamente por el ratón.

Al crear un interfaz para el ratón, se busca lograr un control directo del usuario con el objeto de control, que proporcione los procedimientos para:

⇨ Restringir el desplazamiento del ratón a una zona determinada de la pantalla.

⇨ Leer directamente la posición en la pantalla en la que se encuentra el ratón.

⇨ Cambiar los factores de respuesta del ratón y ajustar el desplazamiento vertical y horizontal de éste a los factores de desplazamiento de la pantalla.

⇨ Seleccionar los cursores del ratón gráfico o crear cursores nuevos.

⇨ Ocultar o mostrar de forma selectiva el cursor del ratón.

Para comenzar a trabajar con el ratón solo se requieren de unas cuantas funciones básicas para inicializarlo, obtener la posición de su cursor y mover el cursor.

Un sistema de ratón consiste de dos elementos importantes:

- El mecanismo del ratón
- Un programa residente en memoria, conocido como manejador de ratón, el cual proporciona el soporte de bajo nivel que se necesita para comunicarse con el ratón, además es responsable de mantener automáticamente la posición del cursor del ratón e identificar la opresión de cualquier botón.

El manejador del ratón se carga en memoria cuando se enciende la computadora mediante una instrucción en el archivo autoexec.bat, una vez que este objeto de control se carga., el manejador queda disponible para cualquier programa que ejecute después.

La manera de acceder al manejador del ratón se realiza mediante una interrupción de software número 33 h de la PC.

Un manejador de ratón da servicio a las llamadas a esta interrupción redirigiéndola a sus funciones de bajo nivel. La función específica que se selecciona depende del valor del registro AX en el momento de la interrupción, otros registros como : BX, CX, DX, se utilizan para pasar parámetros de rutina del ratón. Así mismo las funciones del ratón emplean estos cuatro registros para regresar a la función que los llama, acciones como: la ubicación del ratón y el estado de los botones.

La interrupción 33h se puede llamar por medio de la función **int86( )**, puesta a nuestro alcance gracias a Turbo C++, la cual proporciona acceso directo a la capacidad de interrupción del procesador y su forma es:

int 86(interrup_número, &inregs, &outregs);

inregs.- registro de entrada

outregs.- registro de salida

El primer argumento es el vector de interrupción que será usado, y los otros dos argumentos son apuntadores a una estructura de valores de registros.

En el diseño del paquete se empleará el programa del ratón con referencia a los cuatro argumentos del ratón, los cuales corresponden a los registros AX, BX, CX y DX, respectivamente.

6.2 Funciones del ratón

El manejador del ratón incluye más de 20 funciones, para el desarrollo del paquete gráfico no se emplearán todas, pero se hará una revisión de todas las que podríamos implementar.

Tabla VI.

Funciones del manejador del ratón

Número de función	Descripción
0	Reinicia el ratón y regresa a su estado.
1	Muestra el cursor del ratón en la pantalla.
2	Quita el cursor del ratón en la pantalla.
3	Regresa la posición del ratón y el estado de los botones.
4	Mueve el cursor del ratón a la posición virtual (x,y)
5	Regresa el número de veces que un botón ha sido oprimido desde la llamada anterior
6	Regresa el número de veces que el botón ha sido soltado desde la llamada anterior.
7	Define los límites horizontales del cursor.
8	Define los límites verticales del cursor.
9	Define el cursor usado en modo gráfico.
10	Define el cursor usado en modo texto.
11	Lee los contadores de movimiento del ratón.
12	Define una rutina de interrupción.
13	Activa la emulación de pluma luminosa.
14	Desactiva la emulación de pluma luminosa.

15	Define la relación entre los movimientos del ratón y del cursor.
16	Oculta el cursor si se encuentra dentro de una región.
19	Define los parámetros para definir movimientos del ratón más rígidos.
20	Intercambia rutinas de interupción.
21	Regresa el estado del manejador del ratón.
22	Guarda el estado del manejador del ratón.
23	Restaura el estado del manejador del ratón.
29	Pone el número de página usado por el cursor del ratón.
30	Regresa el número de página usado por el cursor del ratón.

Para emplear el ratón en KYC_DIBUJA se declara un tipo de dato que define una variable ratón-t como una estructura que incluye los campos que nos dan la información si el botón del ratón se encuentra presionado y en que coordenadas se encuentra.

La inicialización del ratón se logra llamando la función número 0 en este caso nuestra función se llamará poner-ratón. Entre las funciones empleadas están:

ver_raton(int visible).-

Con esta función podemos hacer visible o no al ratón.

ratonxy(int x, int y).-

Mueve el cursor-r del ratón a la posición virtual (x,y).

raton_oprimido( ).-

Verifica si los botones del ratón han sido o no oprimidos.

raton_posicion( ).-

Devuelve la posición del ratón.

raton_limitex(int min_x, int max_x).-

Donde se limita el movimiento del ratón en la pantalla, determinando la máxima y mínima posición en x.

raton_limitey((int min_y, int max_y).-

Donde se limita el movimiento del ratón en la pantalla, determinando la máxima y mínima posición en y.

liberar-raton( ).-

Especifica cuando el ratón no se encuentra presionado.

Demás funciones empleadas para implementar el ratón son detalladas en el capítulo 8. Gracias al manejador del ratón que viene incluido en el hardware podemos activar varias acciones para el ratón.

Por lo tanto, utilizando la función de reiniciar, función 0, tendremos la oportunidad de decir cuándo está o no presente el ratón.

En modo gráfico, el cursor predeterminado del ratón se despliega como un símbolo de flecha. A pesar de que es posible cambiar el tipo de cursor desplegado, emplearemos el tradicional.

La responsabilidad del manejador del ratón es mantener la posición del cursor del ratón, llamando a la función 3 podemos pedirle al manejador del ratón que nos regrese las coordenadas del ratón. La función 3 no espera que se le dé un argumento, y regresa las coordenadas del ratón.

Cuando la pantalla está en un modo con 320 columnas, las coordenadas virtuales en la dirección x no son idénticas, Pero el ajuste es simple, las coordenadas de pantalla en este caso siempre están a la mitad de las coordenadas virtuales, por lo tanto las

coordenadas del manejador del ratón solo necesitan ser divididas entre dos cuando el modo gráfico actual tiene 320 columnas.

Los *botones del ratón,* pueden ser accesados mediante el manejador del ratón, pero trabajamos en este caso con dos botones, en casos donde se cuenten con tres botones en el ratón, se los puede manejar como una extensión al ratón de dos botones. La función 5 del ratón se utiliza para determinar el número de veces que alguno de los botones del ratón ha sido oprimido desde la última ves que se revisó o desde la inicialización.. El número de acciones del botón, incluye el número de veces que se oprime para la función 5.

6.3 Objetos gráficos de control.-

Las extensiones proporcionadas por TURBO C++ a la programación orientada a objetos ofrecen diversas ventajas al programador, una de las áreas en las que la ventaja es más vistosa, es la de la creación de objetos gráficos de control.

En las aplicaciones gráficas, un objeto de control tiene que asumir cinco criterios principales, a saber:

⇨ Crear y mantener su propia imagen.

⇨ Cambiar su tamaño y su posición, si fuera necesario.

⇨ Cambiar su apariencia de acuerdo con su función.

⇨ Responder a un suceso provocado por el ratón directa o indirectamente

En el entorno de los objetos gráficos, es necesario prestar mucha atención a una única ocurrencia de un tipo de objeto específico, cuando ya se ha creado y depurado la primera ocurrencia, pueden hacerse un sin número de duplicaciones asignando nuevas acciones y variaciones.

La creación de un nuevo tipo de objeto derivado de uno en funcionamiento requiere mucho menos tiempo y trabajo que el desarrollo de una propiedad nueva, ya que gran parte se deriva del objeto progenitor y no necesita una repetición laboriosa.

Durante el desarrollo del diseño del paquete, se ha utilizado como objeto de control para el mismo a: botones, cuyas acciones parten de funciones que son detalladas en capítulos siguientes para crear los botones se utiliza la función poner-boton( ), donde se establecen las coordenadas y a partir de ellas surgen las variaciones para cada botón. Los íconos fueron diseñados a mano, es decir para la representación de una imagen en cada uno de los botones que demuestre la acción del mismo, se dibujó sobre cada botón con solo el uso de pixeles, gracias a ellos se fue dando forma al dibujo. Para la ejecución del ícono se llama al ratón y se oprime en el botón deseado, obteniendo como resultado el gráfico que el usuario desea hacer. Todo esto es

posible por funciones que fueron incluidas para cada botón, Las mismas que se recalcarán en el capítulo 8.

7. DISEÑO DE UN PAQUETE GRÁFICO INTERACTIVO

7.1 Criterios

Dibujar figuras en un programa usando TURBO C++ es bastante simple, pero requiere conocer el lenguaje. Sin embargo, en un programa de dibujo o una aplicación, donde existe la creación de gráficos pero con la ayuda del ratón, es cuando el usuario dibuja en forma interactiva utilizando los objetos de control que el paquete le proporciona. Pero la BGI no incluye rutinas de dibujo interactivo, por lo que en el desarrollo del programa se crean las rutinas necesarias para lograr la interacción.

KYC_DIBUJA, cumple con características primordiales para un usuario que no requiere conocimiento de programación, es realmente fácil de usar, tiene una ventana de ayuda, donde usted encuentra notas principales que debería conocer, con tan solo

presionar el botón AYUDA (Determinado por un signo de interrogación). Una vez que usted presiona el botón de ayuda con el signo de interrogación, luego ubíquese sobre el botón que desea conocer más a fondo, presiónelo y aparecerá una ventana de ayuda para ese botón específico.

Este paquete adquiere una flexibilidad en la creación de figuras gracias al uso de 13 herramientas, donde cada una de ellas está implementada dentro de una función para lograr un medio poderoso de creación. Además incluye también, otra herramienta que nos ayuda a dibujar gráficas estadísticas, lo único que el usuario debe hacer es ingresar los datos correspondientes.

Todas las herramientas tienen un objetivo, el de proporcionar al usuario facilidad en la construcción de figuras. Las herramientas que incluye KYC-DIBUJA pueden ejecutarse con la ayuda del ratón, objeto de control que ocupa un papel importante en la obtención de resultados en el diseño de gráficas de este paquete interactivo.

7.2 Ciclo de desarrollo del paquete gráfico

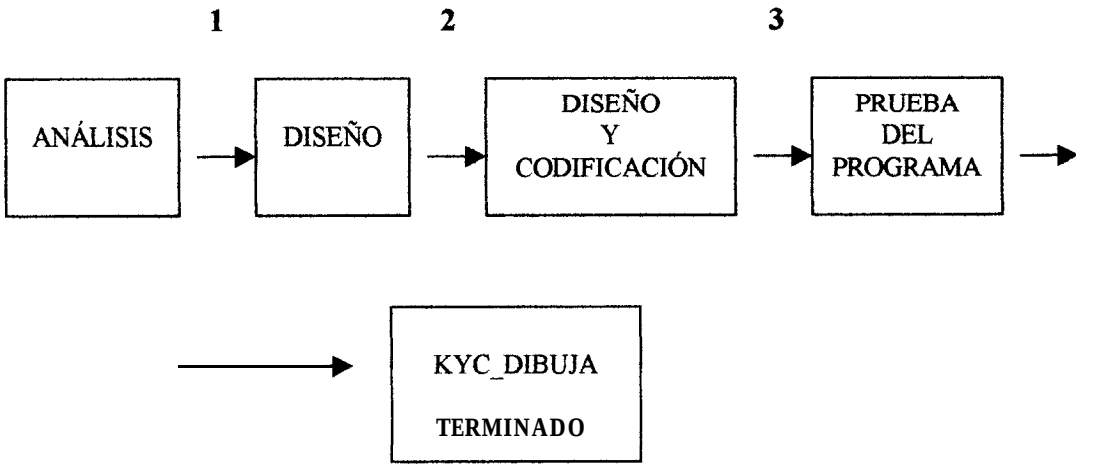
En el diseño de este paquete gráfico es necesario recalcar las fases o metodología que se utilizó para crear KYC-DIBUJA la cantidad de tiempo empleado en cada una de las tareas que se llevaron a cabo y la importancia y dependencia demostrada

mediante una gráfica con la finalidad de indicar la jerarquía de los módulos del programa.

El ciclo de desarrollo de KYC-DIBUJA fue el siguiente:

Fig 7.1

Fases de desarrollo de KYC-DIBUJA



La ventaja de usar de manera estructurada un ciclo de desarrollo del paquete es reducir el tiempo requerido, ya que se lleva una organización de Las tareas a realizar. Los números que están en la parte superior fase a fase representan:

1. Es posible que ocurran preguntas de diseño, que serán contestadas en el análisis del paquete donde se establecen los requerimientos.

2. Preguntas de especificaciones del programa serán propuestas durante el desarrollo de la codificación, y en la fase de diseño es donde Las especificaciones son establecidas es por ello que puede retornar a la segunda fase buscando respuesta.

3. Una vez que se ha realizado la prueba del programa pueden presentarse errores y fallas en el mismo, por lo tanto de la cuarta fase necesita regresar a la codificación, revisando de esta manera si el código fuente está fallando.

Una vez que todos estos problemas han sido superados podemos decir que KYC-DIBUJA ha sido terminado. Para programar las actividades empleadas para el desarrollo de KYC-DIBUJA, y determinar de esa manera las actividades que son interdependientes, y el tiempo establecido para la ejecución de cada una de ellas, una gráfica de Gantt es una forma fácil para calendarizar tareas. Como es esencialmente una gráfica en donde las barras representan cada tarea o actividad a realizarse, va a ser muy útil para indicar la realización de actividades para el diseño de KYC-DIBUJA.

Tabla VII

Listado de las actividades realizadas para el desarrollo de KYC_DIBUJA

EVENTO	NOMBBRE DE LA TAREA	DURACIÓN EN DIAS	PRECESOR
1	Aceptación del tema propuesto	5	
2	Identificar los objetivos del tema	7	1
3	Determinación de los reque-rimientos del tema	7	2
4	Análisis del sumario	5	3
5	Presentación del sumario y aceptación	12	4
6	Búsqueda de herramientas de ayuda	15	1
7	Búsqueda de información	25	1
8	Análisis de los temas	170	7,6
9	Establecer criterios	10	5
10	Diseño del paquete	35	9
11	Codificación del programa	175	9
12	Prueba del programa	2	11
13	Documentación final de la tesis	7	8.11
14	Presentación final	13	13

El inicio de las actividades fue a partir del mes de junio, en la tabla donde se describen las actividades se mencionan también el tiempo en días que duró cada actividad concluyendo así todas las actividades a principios del mes de febrero aproximadamente. La representación de la gráfica de Gantt de la tabla VII. es presentado en el anexo 1.

7.3 Interacción

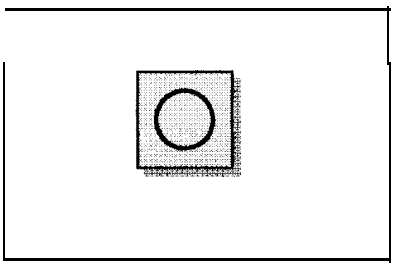
Para conocer más de cerca como es la relación usuario computadora será detallado como es el uso de algunas de las herramientas que se presentan en KYC_DIBUJA.

Una vez que se presiona el botón donde se encuentra dibujado el icono deseado para trabajar, se activa automáticamente la orden en ejecución.

Dibujo de círculos. -

Fig7.2

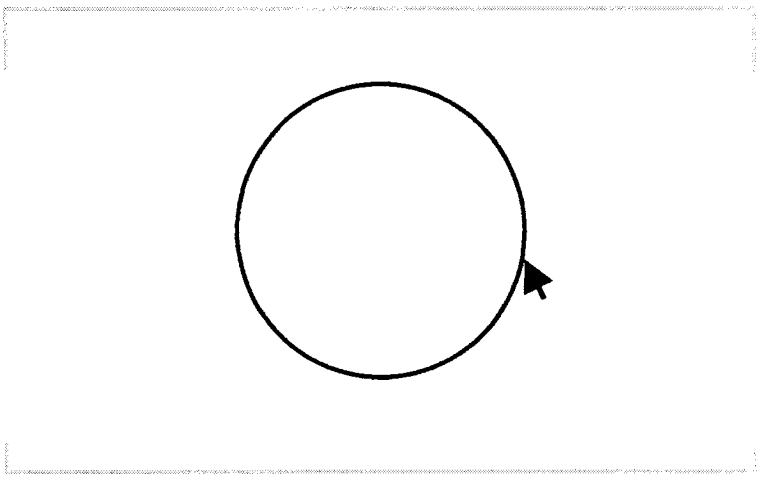
Icono que activa la función para obtener un círculo.



Permite dibujar interactivamente un círculo, tal como muestra la figura. El dibujo de un círculo se va iniciar al especificar el punto central del mismo. Esto se hace moviendo el ratón a la posición deseada y soltando el botón izquierdo del ratón que nos encontramos oprimiendo. Además mientras se mantiene presionado el botón se puede arrastrar el ratón a la derecha o a la izquierda para cambiar el tamaño del círculo, y así desplazarnos por todo el pantalla de dibujo (viewport), y leer las coordenadas exactas donde nos encontramos para luego activarse, según la acción desarrollada por el ratón. Conforme se mueve el ratón el radio del círculo se calcula continuamente, tomando como punto central la diferencia de las coordenadas (x0,y0) y las (xl,yl), que no son más sino la lectura de la primera posición del ratón al activar presionando (click), hasta la última posición del ratón al desactivar el arrastre de la figura soltando el botón.

Fig7.3

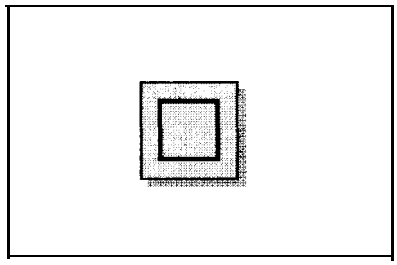
. Uso del ícono para dibujar círculos



Dibujo de cuadrados. -

Fig7.4

Icono que activa la función para obtener un cuadrado.



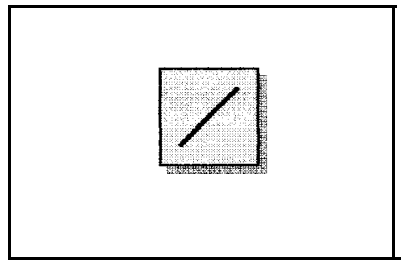
Al igual que antes, el dibujo no se inicia sino hasta que se presione el botón izquierdo del ratón, pero eso solamente si antes ha activado el ícono de la ejecución del cuadrado y la única manera de salir de la función de dibujo es soltando el botón.

La obtención del rectángulo es posible tan solo depende del usuario, si él lo desea lo único que debe hacer es arrastrar el ratón a la posición donde desee el rectángulo con un valor de x o de y mayor al de la posición inicial.

Dibujo de lineas. -

Fig 7.5

Icono que activa la función para obtener una línea.

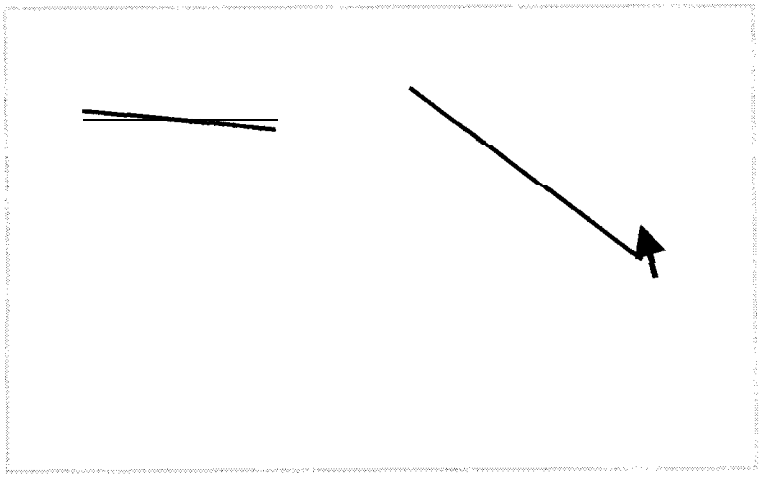


Se usa para dibujar segmentos de línea permite dibujar una sola línea apuntando al inicio de la línea, oprimiendo y manteniendo oprimido el botón izquierdo del ratón y luego arrastrando el ratón al punto final que se desea. Cuando se suelta el botón, la línea se fija. Sin embargo, mientras el botón está presionado, se continúa dibujando la línea desde la posición inicial del ratón donde se oprimió el botón, hasta la posición actual del ratón. Por lo tanto al tiempo que se mueve el ratón por la pantalla, la línea se alarga y se reduce tanto como se requiere.

Este procedimiento para dibujar líneas se muestra en la figura, este tipo de línea se llama de hule, ya que la línea parece que fuera flexible como una liga. Para dibujar más líneas se repite el mismo proceso simple efectuado anteriormente accionando el ícono

Fig 7.6

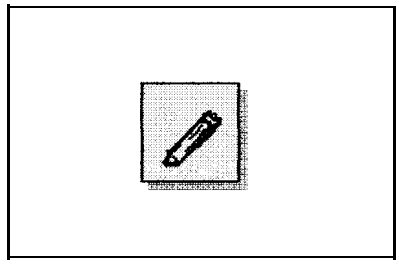
Uso del botón para dibujar líneas.



Dibujar con un lápiz.-

Fig 7.7

Icono que activa la función para dibujar a mano.



Se utiliza para dibujar figuras a mano libre.

Activando este ícono usted puede dibujar curvas continuas mientras mantenga oprimido el botón izquierdo del ratón, en el momento que se suelta el botón se detiene el dibujo. Para continuar dibujando, solamente vuelve a activar el lápiz.

La acción de dibujo comienza cuando usted lo decida y después en la posición donde se encuentre con el ratón empieza a marcarse cada pixel mientras mantenga presionado el botón indicado.

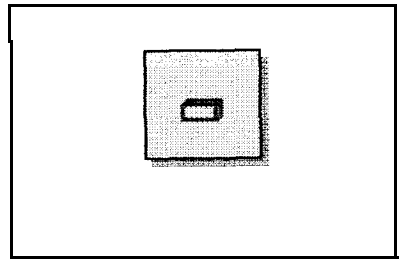
Fig. 7.8

Dibujo realizado presionando el botón de lápiz

Borrudo. ▪

Fig. 7.9

Botón que activa el borrador



Una vez que que se realiza alguna acción para un dibujo y se obtienen los resultados, es posible borrarlos sino son los deseados. La acción de borrado, se logra poniendo

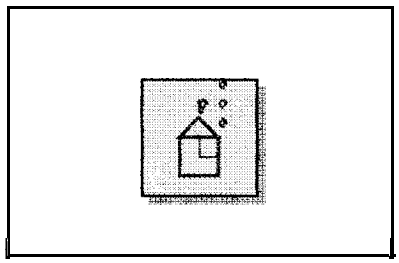
una región rectangular, debajo del cursor del ratón, al color de fondo de la pantalla de dibujo. Oprimiendo el botón izquierdo del ratón y moviendo éste, es posible borrar cualquier área que se desee.

Se dibujan unas barras pequeñas rellenas con el color de fondo de la pantalla de dibujo en donde se localice el ratón mientras se oprima el botón del mismo. Para generar estos pequeños bloques borradores, los colores de relleno y de dibujo se ponen a color de fondo.

Pintura en aerosol. -

Fig 7.10

Icono que activa el efecto de aerosol.



Un efecto simple de pintura en aerosol es el que aquí se obtiene, poniendo el color de dibujo especificado a los pixeles seleccionados aleatoriamente dentro de una región circular.

Mientras se mantenga oprimido el botón izquierdo del ratón, la acción de pintar con aerosol se efectúa, entre más tiempo mantenga el ratón en una posición, más se pintará esa posición. Para detener momentáneamente la pintura en aerosol, suelte el botón izquierdo del ratón. Es posible escoger el color para el pintado en aerosol ya que KYC_DIBUJA cuenta además con una paleta de colores.

Estos botones presentados son algunos de los empleados en el paquete, se dibujaron para que el usuario tenga idea de cuál es la acción que desea realizar

8. PROGRAMACION DE UN PAQUETE GRAFICO

8.1 Creación de herramientas gráficas

En capítulos anteriores se hace referencia de las funciones gráficas que proporciona TURBO C++, para ayuda del programador. En el presente capítulo se hará uso de estas funciones para la creación de herramientas gráficas necesarias en el diseño del paquete gráfico.

Se han implementado varias funciones utilizadas como herramientas para el diseño de un ambiente gráfico, para dibujar figuras y formas. Cada una de ellas a la vez puede utilizar otras subfunciones de manera de contribuir para el desarrollo de atributos que provoquen la acción deseada.

Entre ellas:

- Inicializar ( )
- poner-set ( )
- marco (x1, yl, x2, y2, color, aspecto)
- texto-set (fuente, tamaño, ampliación)
- gprintf (x, y, color, mensaje, tipo)
- poner-boton (x, y, aspecto, texto, colorTexto, , aspecto del texto)
- poner-mensaje (mensaje, color, posición)
- da_boton (x, y)
- ver_boton (#botón, aspecto)
- da-datos ( )
- ejecutar ( )
- dibujar-figu (x1, yl, x2, y2, tipo)
- paste ( )
- barra ( )
- barra3D ( )
- rotar (figura, grados)
- escala-rotar (escala-tipo)
- poner ayuda (tópico)
- lee archivo (tópico)

inicializar ( ).-

Al principio de todo programa gráfico, es necesario poner la tarjeta del adaptador de video (hardware) del monitor en modo gráfico, a fin de desplegar elementos gráficos. La función de la BGI que se emplea para configurar la computadora a modo gráfico es llamada initgraph ( ), que contiene tres argumentos, los dos primeros son apuntadores a enteros. Estos contienen los valores lógicos que especifican el adaptador de video y el modo a usar. El tercer argumento especifica el nombre del archivo donde se encuentran almacenados los archivos de manejadores gráficos de dispositivos de TURBO C++.

poner-set ( ).-

Esta función como su nombre lo dice pone la presentación que el usuario visualiza como pantalla gráfica donde puede interactuar. Aquí se especifican los marcos de dibujo, de estado y textos de presentación.

marco ( ).-

Presenta un marco tridimensional, muchas veces es empleada dentro de otras funciones. Esta herramienta emplea seis argumentos donde se especifica la dimensión del marco. En el paquete esta función se emplea para crear el marco de dibujo de la pantalla, los marcos de estado y para la creación del marco para los botones.

texto-set ( ).-

Función que pone el modo del texto, el tamaño del mismo y la posición.

gprint ( ).-

Imprime el modo gráfico de un texto, donde sean especificadas las coordenadas que este se posicionará.

poner-boton ( ).-

Función encargada de poner los botones en la pantalla gráfica, entre los argumentos de esta función se encuentra la inclusión de un texto que estará ubicado en el interior del botón.

poner-mensaje ( ).-

Función encargada de poner mensajes con el color especificado. La posición del mensaje no está incluida entre los argumentos de esta función pero para darle ubicación se hacen uso de otras funciones.

da_boton ( ).-

Con ayuda de esta función es posible conocer que botón es el que fue presionado., de todos los que se presentan en pantalla de dibujo. Información necesaria para ejecutar alguna acción deseada.

ve_boton ( ).-

Pone los botones en la pantalla del paquete gráfico. A diferencia de la función poner-boton ( ), no solo incluye la ubicación de algún texto dentro del botón sino, también figuras necesarias para la visualización del usuario, con la finalidad que el mismo identifique la acción del botón con el solo hecho de observar que figura esta representada en cada botón.

da-datos ( ).-

Esta función nos permite ingresar datos, tanto para el eje **x,** como para el eje y. Para luego ser llamados en las gráficas de presentación como las barras, las barras en tres dimensiones y el pastel. Recibe datos numéricos y de texto, especificados para cada eje. Cuando se la llama aparece un marco en el centro de la pantalla de dibujo y se ingresa los datos.

ejecutar ( ).-

Nos permite ejecutar las acciones especificadas para cada botón de la pantalla gráfica, en donde se especifican las coordenadas que van a ser dibujadas dependiendo de la posición del ratón.

dibujar-figura ( ).-

Cada botón presentado en la pantalla gráfica tiene sus acciones especificas, las cuales son desarrolladas en sta función. Se la llama dentro de la función ejecutar ( ), para lograr el objetivo de dibujo deseado.

pastel ( ).-

Es una función para obtener una gráfica de presentación, como lo es el pastel. Pide los datos de la función da-datos ( ), y los gráfica en el centro de la pantalla, distribuyendo el porcentaje para cada texto determinado.

barra ( ).-

Es una función para obtener una gráfica de presentación, como lo es la barra en dos dimensiones. Pide los datos de la función da-datos ( ), y los grafica en la pantalla de dibujo, distribuyendo el valor específico para cada texto del eje x.

barra3d ( ).-

Es una función para obtener una gráfica de presentación, como lo es la barra en tres dimensiones. Pide los datos de la función da-datos ( ), y los grafica en la pantalla de dibujo, distribuyendo el valor especifico para cada texto del eje x.

rotar( ).-

Con esta función podemos rotar una figura, es posible rotarla hacia arriba y hacia abajo. La rotación es cada 45 grados.

escala-rotar ( ).-

En ésta función decidimos si lo que queremos es cambiar la escala al objeto haciendo variar el tamaño de una figura o si lamamos a la función para rotarlo.

lee-archivo ( ).-

Función encargada de leer el archivo de ayuda para el usuario. El archivo se realizó el mismo programa (TURBO C++).

poner-ayuda ( ).-

Esta función fue implementada para presentar una ayuda al usuario. Ya que se realizó también una función que pide un archivo donde se editó la ayuda para cada

botón, era necesario crear una función que llame al archivo y así se presente en una ventana de ayuda en la pantalla de KYC_DIBUJA.

8.2 Análisis y diseño de algoritmos

Para la ejecución de muchas de las tareas del paquete gráfico KYC_DIBUJA, se han llevado a cabo un sin número de funciones, ya sea de ejecución de cualquier acción o de gráficos. En el programa principal es en donde son llamadas la mayoría de ellas. Los algoritmos presentados en el desarrollo de este subcapítulo son de las funciones principales , las que no son aquí recalcadas y Las llamadas fueron desarrolladas en el capítulo 4 de algoritmos de graficación, en este capítulo solo se encuentran los algoritmos de ejecución de los gráficos antes desarrollados.

Para ejecutar las acciones de los botones:

ejecutar ( )

Las variables x1,x2,y1,y2 son introducidas cuando lee las posiciones del ratón.

x1, yl, x2, y2 : enteros

El número de botón seleccionado se especifica en la variable:

n_ botones: enteros

INICIO

Ingreso del número de botón seleccionado, son 12 botones, excluyendo al de datos, que se lo llama aparte.

≡n_botones

si 1 > n-botones < 12

hacer

presionar raton

mientras esté presionado capturar las coordenadas del ratón

si x1 → coordenada x del ratón

y 1 → coordenada y del ratón

hacer

tomar la nueva posición del ratón

x2 → coordenada x nueva del ratón

y2 → coordenada y nueva del ratón

entonces

llamar a la función dibujar-figura

FIN

Para realizar las acciones de cada botón:

dibujar-figura ( )

Ingresamos los valores de las coordenadas especificadas por la posición del ratón

x1, yl, x2, y2 : enteros

Número del botón seleccionaado

n_botón : entero

INICIO

Ingresan los valores de las coordenadas

≡ x1, yl, x2, y2

En caso de que se seleccione cierto número de 2 a ll se realiza la acción específica para ese botón.

si n-botón → 2

dibujar rectángulo para seleccionar la imagen

fin

si n_botón → 3

dibujar rectángulo

fin

si n_botón → 4

dibujar barra para borrar

fin

si n_botón → 5

efecto aerosol

fin

si n botón → 6

rotar 45 grados hacia la izquierda

fin

si n_botón → 7

```
    dibujar  línea
fin
si n-botón → 8
    dibujar  elipse
fin
si n-botón → 9
    dibujar  a  mano  libre
fin
si n-botón → 10
    cambio  de  escala
fin
si n-botón → ll
    rotar  45  grados  a  la  derecha
fin

FIN
```

8.3 Módulos en orden de desarrollo

Para impíementar KYC-DIBUJA los módulos para desarrollar el programa llevaron un cierto orden y a medida que se realizaban se iban ejecutando, para probar la efectividad de la programación.

En la siguiente tabla se especifica el orden desarrollo de cada módulo para obtener la implementación total de KYC-DIBUJA.

Tabla 8.1

Orden de módulos

ORDEN	MODULO	TIEMPO	ESTRATEGIA
1.	Se desarrollo para comenzar, funciones para la presentación de KYC-DIBUJA. - Botones - Marco de dibujo - Marco de estado	2 semanas	Observar otros paquetes gráficos para diseñar presentación.
2.	Presentación para botones: - Gráfica para cada botón según su acción específica - Desarrollo de las funciones que activan la acción.	3 semanas	Gráficos observados en el paquete PINTAR(PAINT), para hacer los mismos en cada botón.
3.	Botón de Datos: - Activar botones para gráficas de presentación	5 semanas	Una vez determinado el objetivo de botón de datos haciendo uso de conocimien-

	- Crear las funciones para: - Barra - Barra 3D - Pastel		tos en gráficas estadísticas, las funciones para cada gráfica se desarrollaron, tomando en cuenta esos conocimientos.
4.	Crear la estructura para el ratón: - Funciones que ayudan a manejar el ratón todas ellas están incluidas en una librería creada. - Las funciones son las ya referidas anteriormente.	3 semanas	Haciendo uso de herramientas de investigación se crearon las funciones para activar el ratón gracias al manejador del software.
5.	Módulo principal para la ejecución de las acciones de cada botón, con la ayuda del ratón. Este módulo principal llama a as funciones ya creadas realizadas en otras librerías.	2 semanas	Como ya la mayoría de las funciones habían sido creadas, ya solo tenía que llamarlas según eran requeridas.
6.	Creación de la ventana de ayuda y del botón de salida de KYC-DIBUJA	½ semana	Es necesario que el usuario tenga una ayuda, a pesar de la facilidad de uso de KYC-DIBUJA.

9.APLICACION

9.1 Manejo gráfico de estadística descriptiva.

Como se especificó al inicio del trabajo, una aplicación del paquete gráfico a más de proporcionar al usuario la capacidad de dibujar, le da la facilidad de graficar datos ingresados por él mismo.

Para una demostración de lo que es capaz de hacer el paquete de dibujo, se propondrán tres ejemplos donde utilicemos :

- Barras
- Barras en tres dimensiones

- Pastel

Después de los resultados que proporciones KYC_DIBUJA, se hará una breve descripción del mismo, con la finalidad de exponer las conclusiones que se pueden discernir de la observación gráfica presentada en la pantalla de dibujo.

Ejemplo 1:

La Bolsa de Valores de Guayaquil determinó en el último mes la presencia bursátil de los Bancos emisores conforme al volumen de acciones ocurridas en la última semana del mes.

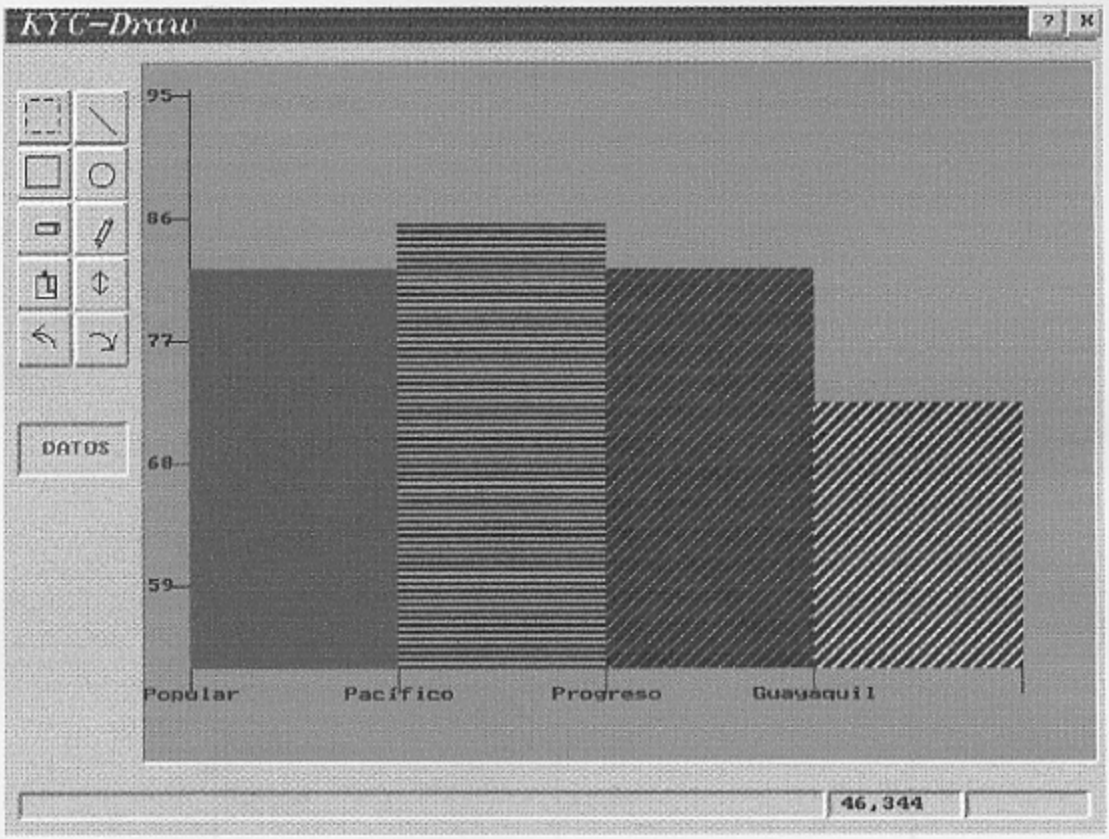
Tabla 9.1

Datos para usar la función graficar barras

BANCOS	PRESENCIA BURSÁTIL %
Popular	82
Pacífico	85
Progreso	82
Guayaquil	73

Figura 9.1

Pantalla de dibujo aplicando barras



En la gráfica presentada se observa que el Banco del Pacífico alcanzó la mayor presencia bursátil frente a los demás bancos en este último mes.

Ejemplo 2:

Los Bancos privados especificaron el interés interbancario que cerró la semana anterior. Cada interés está determinado por día, con la finalidad de verificar la variación que se ha presentado.

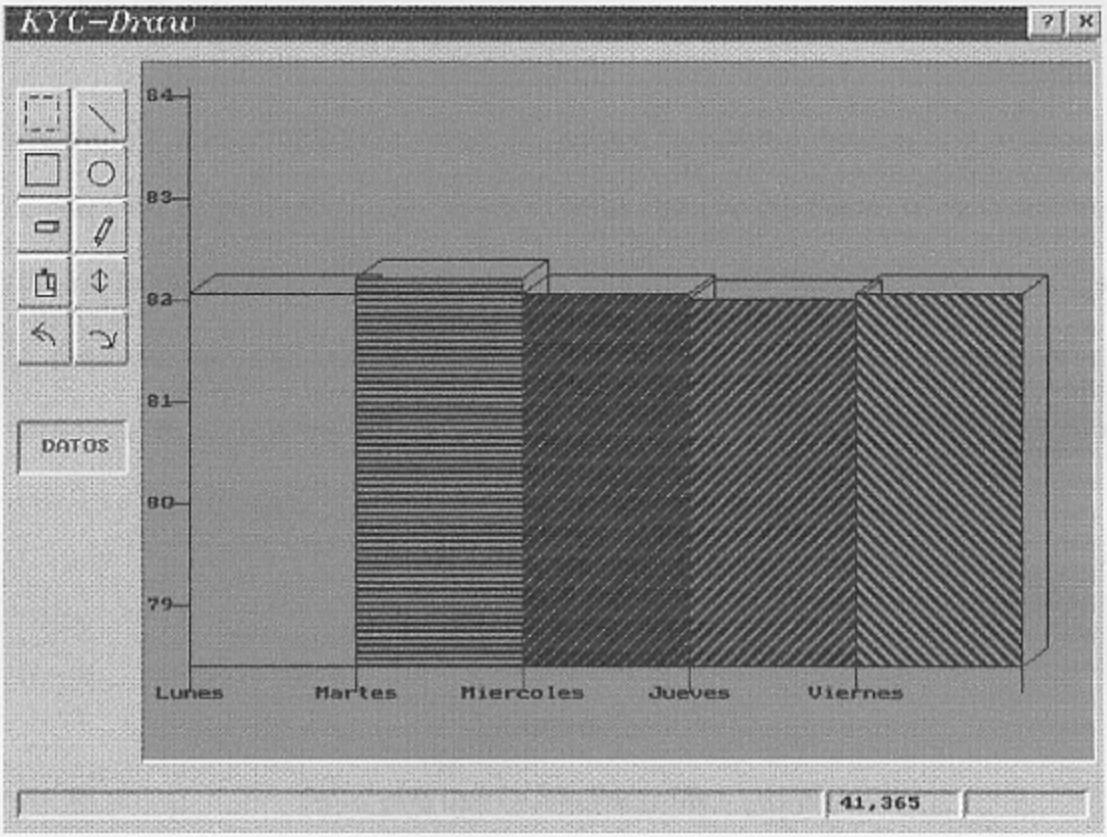
Tabla 9.2

Datos para ejemplo para Barras Tridimensionales

DÍAS	INTERÉS INTERBANCARIO
Lunes	82.10
Martes	82.75
Miércoles	82.02
Jueves	81.91
Viernes	82.16

Figura 9.2

Pantalla de dibujo aplicando barras



Por los resultados del gráfico se puede observar la pequeña variación que ha existido con respecto a las tasas de interés interbancario. Es posible ver que esta tasa cerró casi con un valor muy aproximado al del inicio de semana.

Ejemplo 3:

Ecuador ha sido considerado como el gigante bananero, pero con la continua competencia de otros países que también cuenta con este privilegio, se estableció un sistema de cupos, que representa una cuota para cada país. La repartición de cuotas en toneladas métricas está representada en la siguiente tabla. Se desea saber el porcentaje de repartición de cuotas de cada uno de los países que intervienen con relación a sus toneladas.

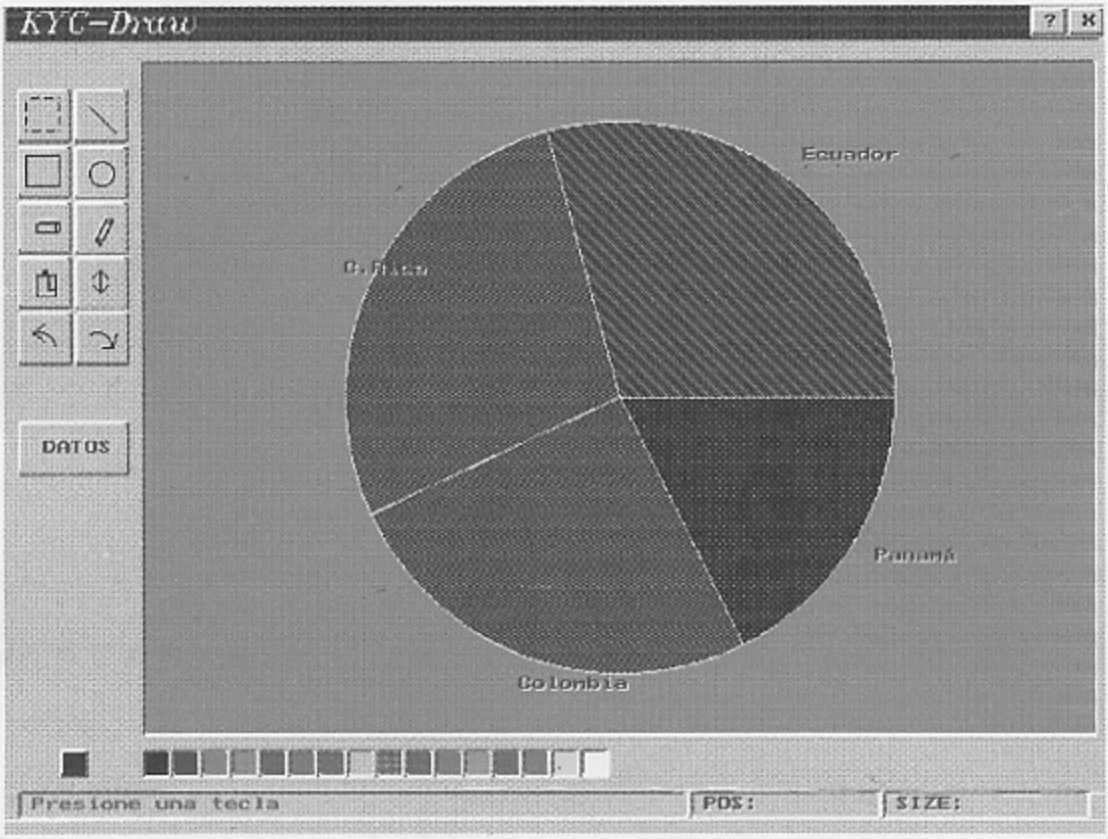
Tabla 9.3

Datos para usar la función **graficar** pastel

PAÍSES	TONELADAS MÉTRICA
Ecuador	668.217
Costa Rica	634.609
Colombia	583.344
Panamá	399.200

Figura 9.3

Pantalla de dibujo aplicando gráfica de pastel



Observamos que el porcentaje en la repartición de cuotas es mayor en Ecuador un porcentaje aproximado del 30%, a diferencia de los demás, a pesar de que Costa Rica y Colombia está muy cercanos.

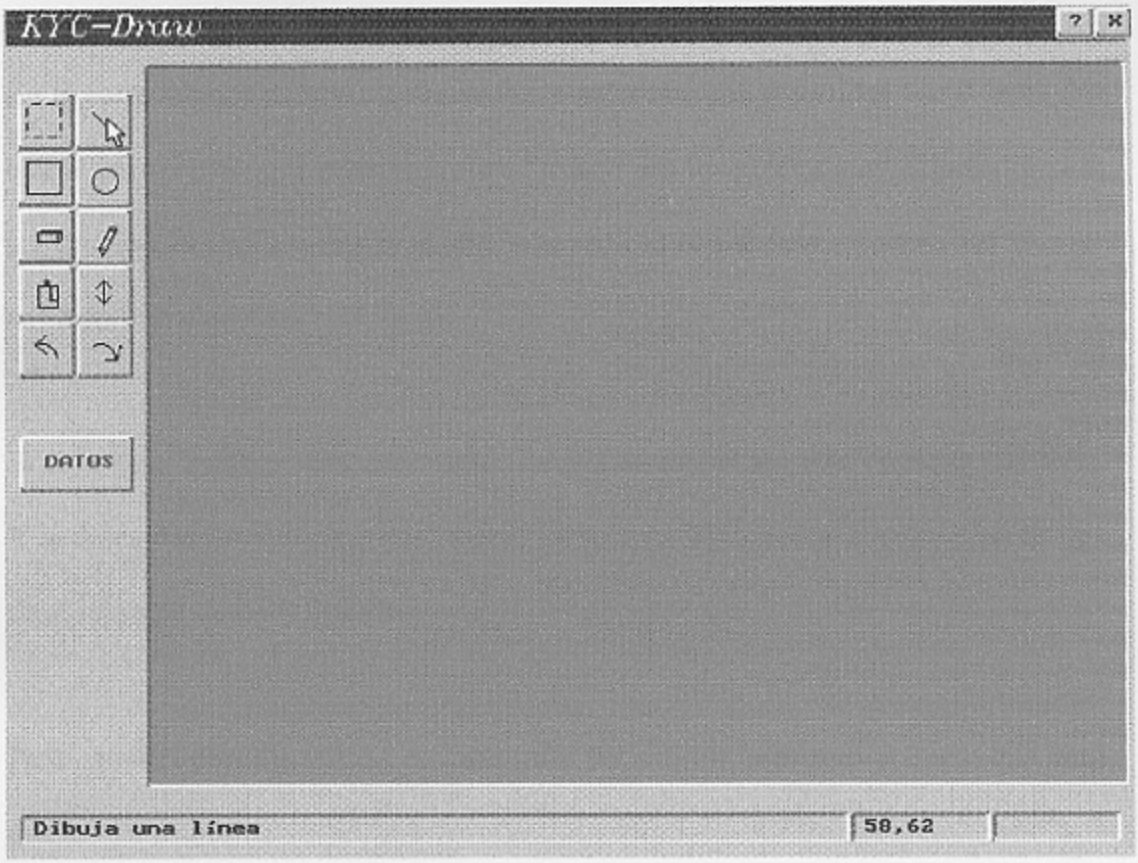
10. Documentación y Prueba

En el presente capítulo se conocerá más a fondo el paquete, con sus respectivos botones. Se presentará la pantalla de inicio que el usuario visualiza al entrar en KYC_DIBUJA.

Pantalla de inicio, es posible observar que el ratón se encuentra ubicado en el botón de línea, y podemos ver en la parte inferior el mensaje “Dibuja una línea”.

Fig 10.1

Pantalla de KYC_DIBUJA



Así como se presenta el mensaje de dibujar línea, cada vez que el ratón sea presionado, la acción del mismo será indicada como mensaje en la parte inferior.

En la siguiente demostración se puede apreciar como el usuario puede hacer uso de los botones, en este caso se ha utilizado el dibujo libre, acción proporcionada por el lápiz, aparece también un cuadrado en la pantalla de dibujo, e incluso para dibujar la flor se realizó un círculo en la mitad de la misma. En el momento que se imprimió la

pantalla el botón de selección se encontraba activado porque se seleccionó la computadora dibujada, con la finalidad de ser trasladada hacia un nuevo punto según lo requiera el usuario.

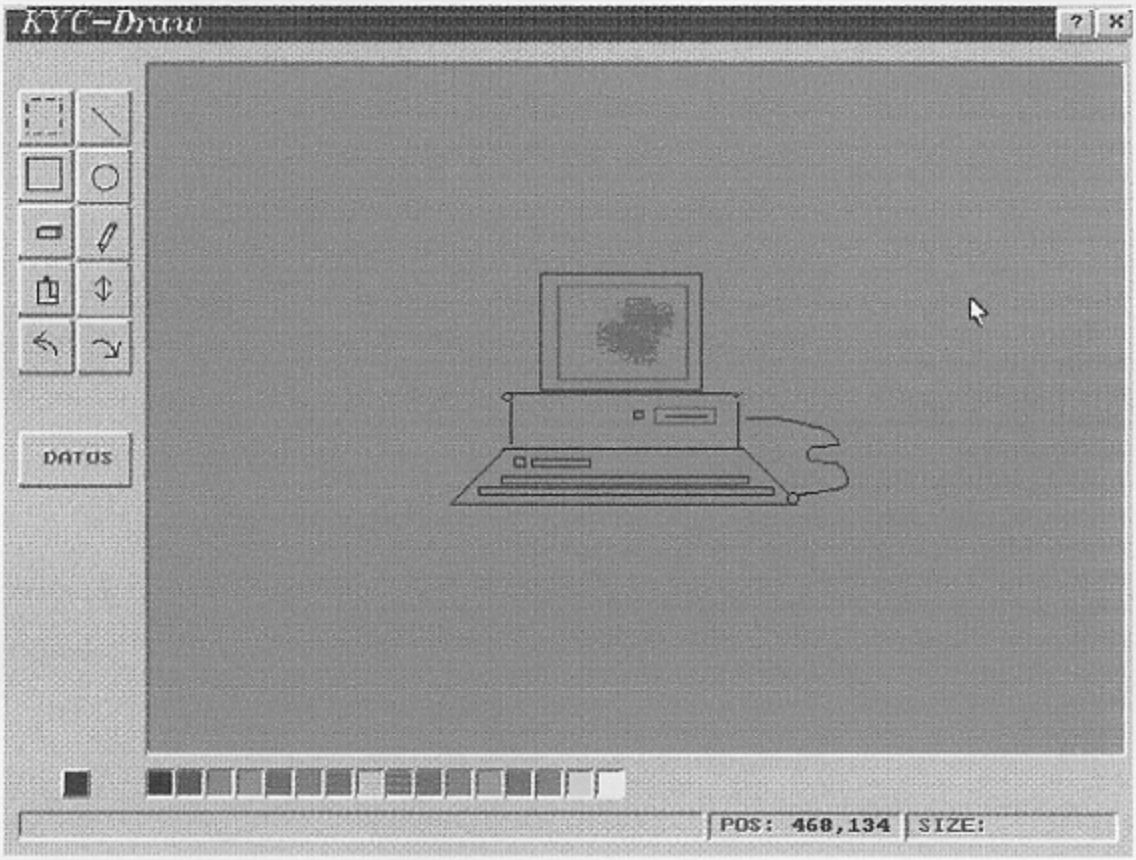
En la parte inferior derecha, aparecen los pixeles que son recorridos por el ratón, los que se encuentran en la primera barra de estado son los pixeles a nivel pantalla toda, en cambio Las coordenadas de la otra barra indican la distancia recorrida una vez que se ha presionado el botón deseado.

Si el usuario quisiera abrir una nueva pantalla de dibujo tan solo presiona una tecla. Y si desea ir a la ayuda que proporciona el paquete basta con presionar el botón que tiene un signo de interrogación.

Para cerrar la pantalla gráfica es decir, salir del paquete presionar el botón que tiene una x.

Fig. 10.2

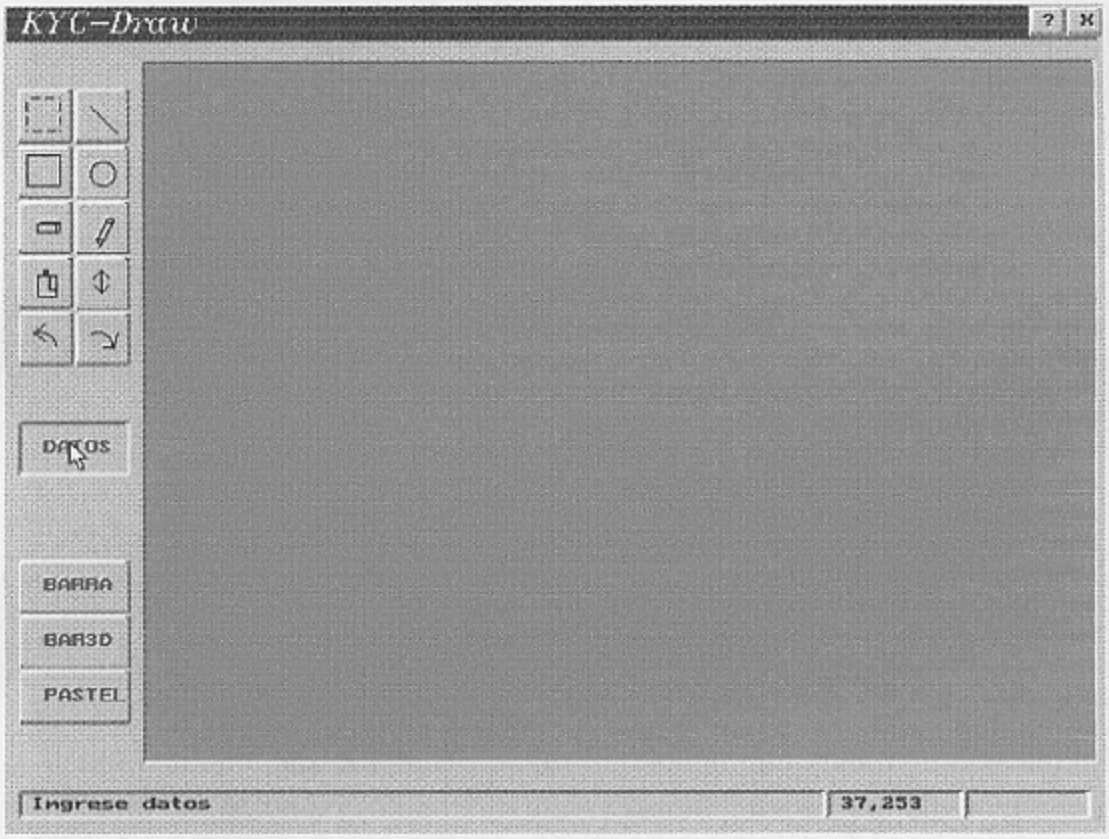
Usando la pantalla de dibujo



El botón que dice datos es el que nos permitirá escoger el tipo de gráfica que queramos activar, o bien barras o pasteles. Una vez que seleccionemos el tipo de gráfica, se activará una pequeña ventana donde ingresaremos la cantidad de datos, los valores del eje x, que pueden incluir texto o cantidades, y los valores de y que son los numéricos.

Fig. 10.3

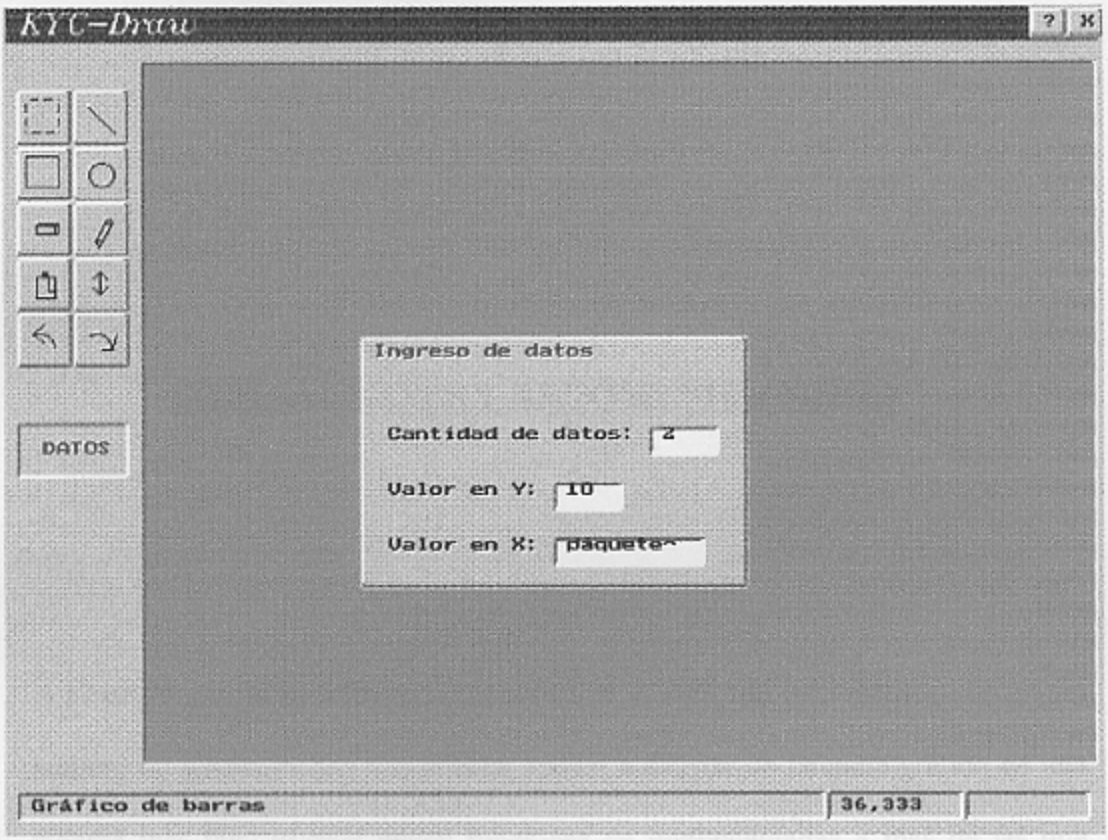
Pantalla con el botón DATOS activado



Los botones del tipo de gráfica han aparecido cuando presionamos datos, ahora veamos la ventana que aparecerá si escoge una. Donde ingresara el número de datos que va a introducir y cada uno de los ejes tato x como y, indicando los datos correspondientes.

Fig.10.4

Ventana para ingresar los datos



Se a abierto la ventana de almacenamiento de datos, que contiene todos los valores ingresados por es usuario especificando primero, la cantidad de valores a introducir.

Estas demostraciones de pantalla son un breve ejemplo de la capacidad de KYC_DIBUJA (KYC_DRAW), en el capítulo posterior, se seguirán analizando estas pantallas, pero con usos más estadísticos.

CONCLUSIONES Y RECOMENDACIONES

1. A pesar de que en la actualidad contamos en el mercado informático, con un sin número de lenguajes que nos pueden ayudar a diseñar un paquete gráfico de una manera más fácil porque tienen funciones ya implementadas a más de las que tiene TURBO C++. KYC_DIBUJA es una clara demostración de que localmente puede ser construida una herramienta usando un lenguaje de programación convencional.

2. Es por ello que para que la tecnología se convierta en una gran herramienta es necesario que sea explotada de manera que no se desaprovechen ninguna de las

facilidades que podría presentar. Este paquete brinda un servicio totalmente flexible, es fácil de acceder a él y el usuario no encontrará ninguna dificultad para interactuar con él.

3. KYC-DIBUJA puede alcanzar mayor capacidad para brindar al usuario un buen servicio, estas innovaciones serán colocadas en nuevas versiones, donde se incluirá el pintado de relleno de las figuras pero con diferentes colores no como en el actual que solo cuenta con pintar bordes, el nuevo se llamaría KYCPINTAR (KYC_PAINT)

4. Es posible también que en nuevas versiones, los datos que son ingresados sean almacenados, al igual que la figura en un archivo. Con la finalidad de dar una nueva facilidad al usuario.

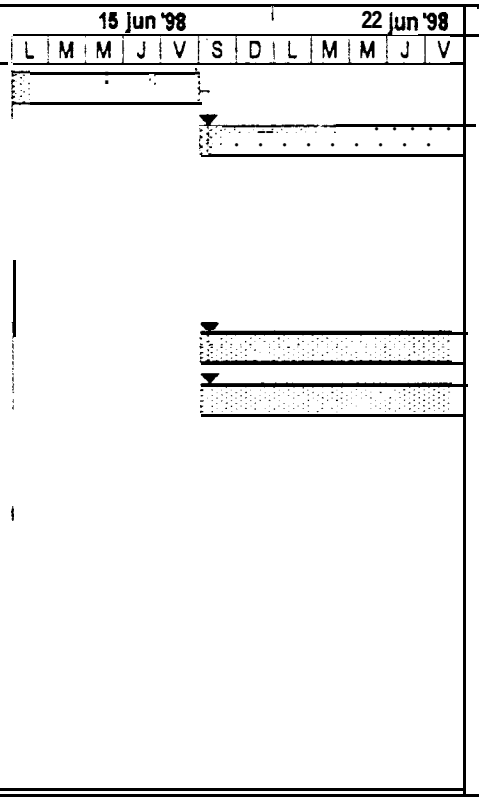
5. El sistema KYC-DIBUJA quedará disponible como herramienta de uso en el Instituto de Ciencias Matemáticas.

6. El conocimiento y la búsqueda de información continua provocarán que para años futuros el interactuar con una máquina sea casi hacerlo con un ser humano.

ANEXO 1

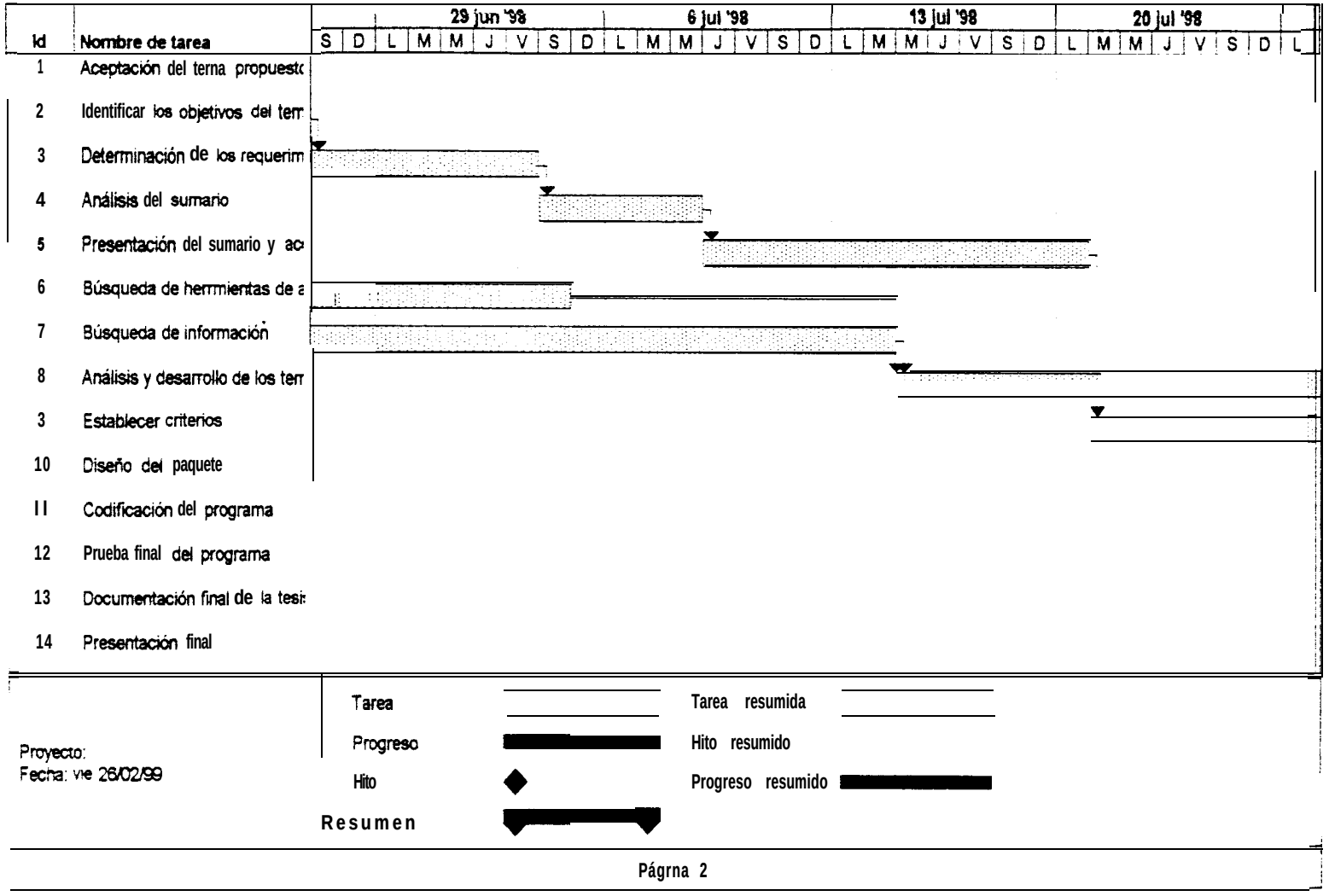
El contenido de este anexo es la presentación de la Gráfica de Gantt, cuya tabla de actividades está en la Tabla VII.

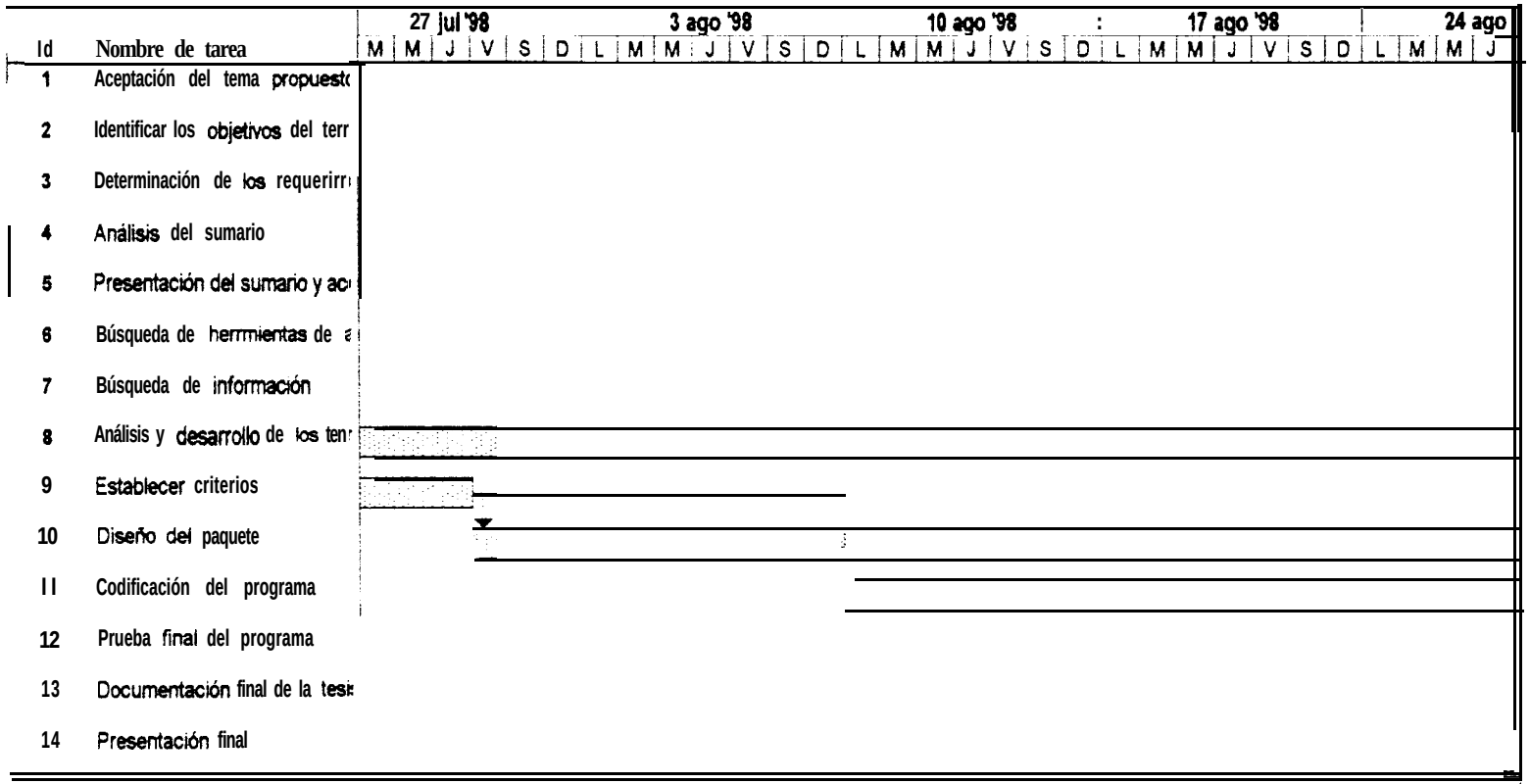
Id	Nombra de tarea	Duración	Comienzo	Fin	Predecesoras
1	Aceptación del tema propuesto	5d	lun 15/06/98	vie 19/06/98	
2	Identificar los objetivos del tem	7d	sáb 20/06/98	vie 26/06/98	1
3	Determinación de los requerim	7d	sáb 27/06/98	vie 3/07/98	2
4	Análisis del sumario	5d	sáb 4/07/98	mié 8/07/98	3
5	Presentación del sumario y ac	12d	jue 9/07/98	lun 20/07/98	4
6	Búsqueda de herrmientas de a	15d	sáb 20/06/98	sáb 4/07/98	1
7	Búsqueda de información	25d	sáb 20/06/98	mar 14/07/98	1
8	Análisis y desarrollo de los ten	170d	mié 15/07/98	jue 31/12/98	7;6
9	Establecer criterios	10d	mar 21/07/98	jue 30/07/98	5
10	Diseño del paquete	35d	vie 31/07/98	jue 3/09/98	9
11	Codificación del programa	175d	lun 10/08/98	dom 31/01/99	9
12	Prueba final del programa	2d	lun 1/02/99	mar 2/02/99	11
13	Documentación final de la tesis	7d	lun 1/02/99	mar -	8:11
14	Presentación final	13d	mié 10/02/99	vie 26/02/99	13

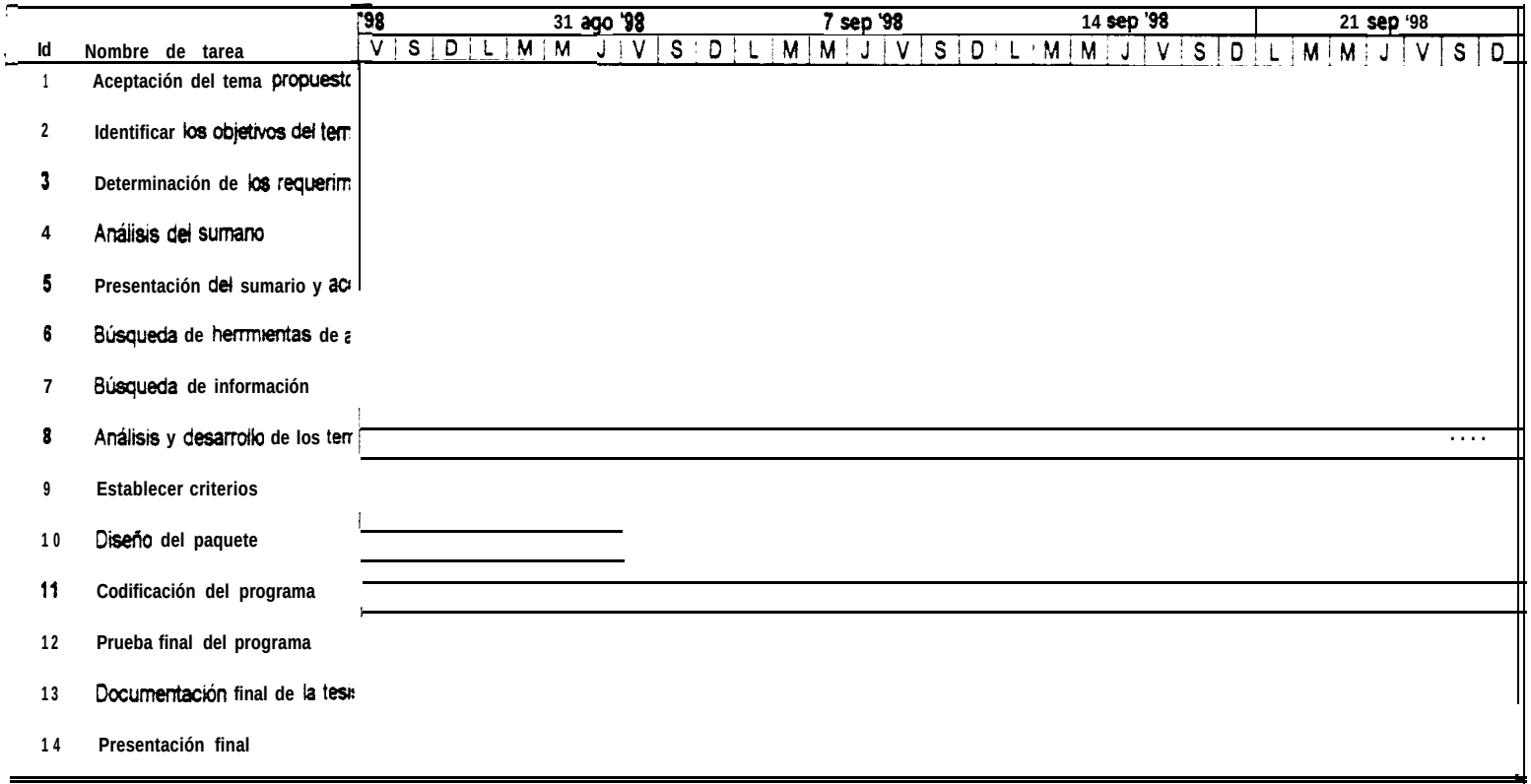


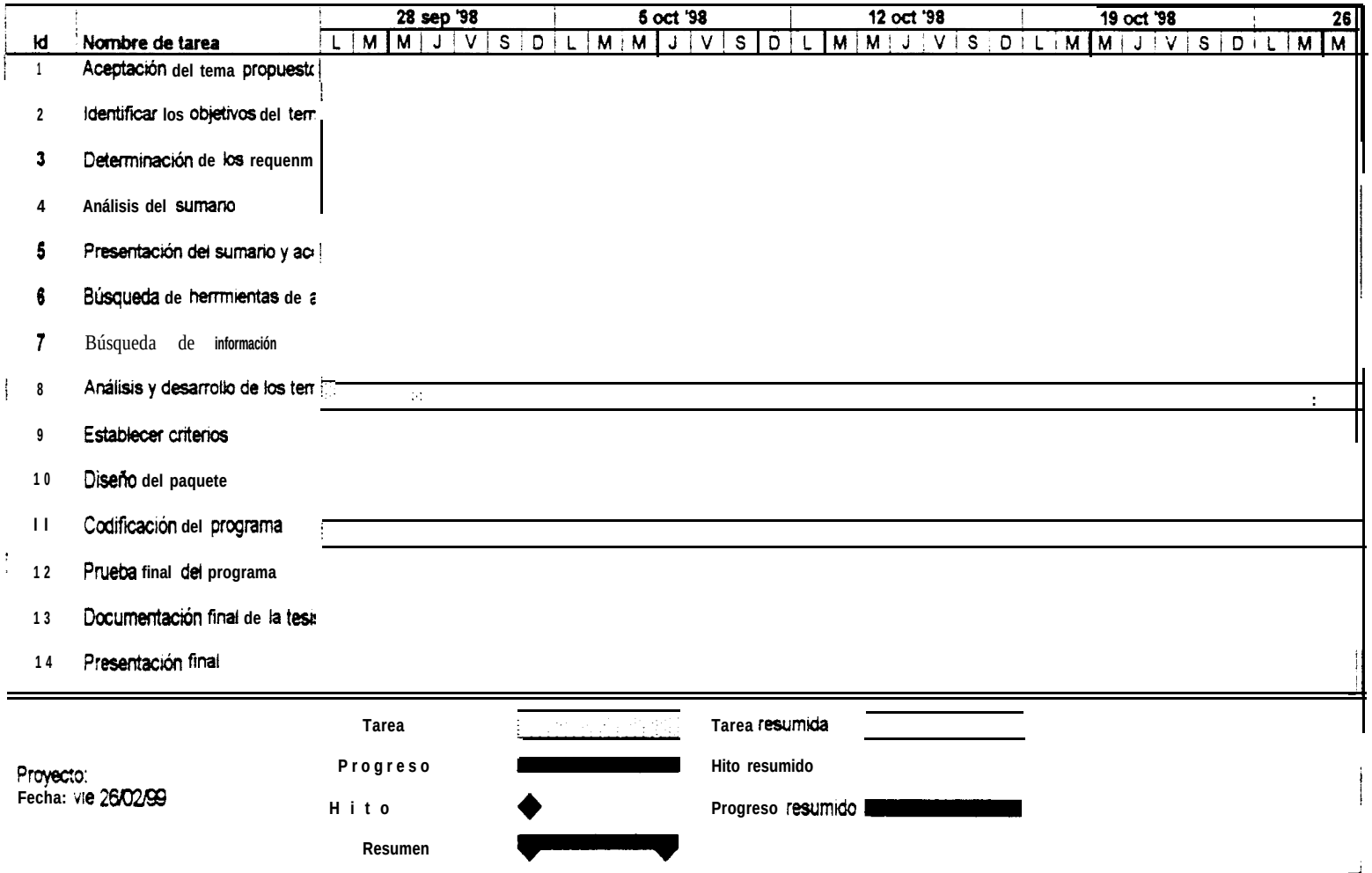
Tarea
Progreso
Hito
Resumen
Tarea resumida
Hito resumido
Progreso resumido

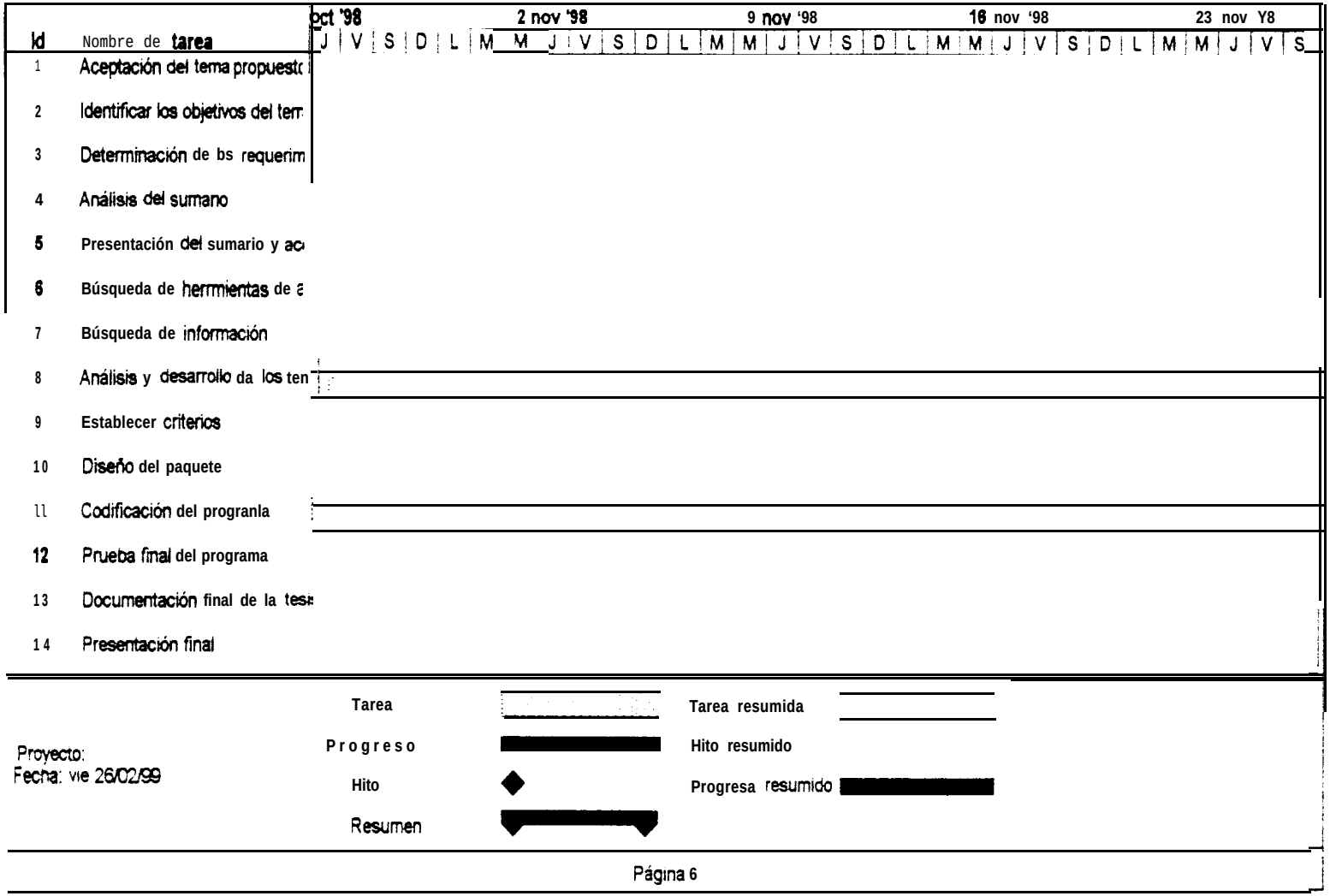
Proyecto:
Fecha: vie 26/02/99


29 jun '98
6 jul '98
13 jul '98
20 jul '98
S D L M M J V S D L M M J V S D L M M J V S D L M M J V S D L
Id
Nombre de tarea
1 Aceptación del terna propuesto
2 Identificar los objetivos del tem
3 Determinación de los requerim
4 Análisis del sumario
5 Presentación del sumario y ac
6 Búsqueda de herrmientas de a
7 Búsqueda de información
8 Análisis y desarrollo de los tem
3 Establecer criterios
10 Diseño del paquete
11 Codificación del programa
12 Prueba final del programa
13 Documentación final de la tesi
14 Presentación final
Tarea
Progreso
Hito
Resumen
Tarea resumida
Hito resumido
Progreso resumido
Proyecto:
Fecha: vie 26/02/99

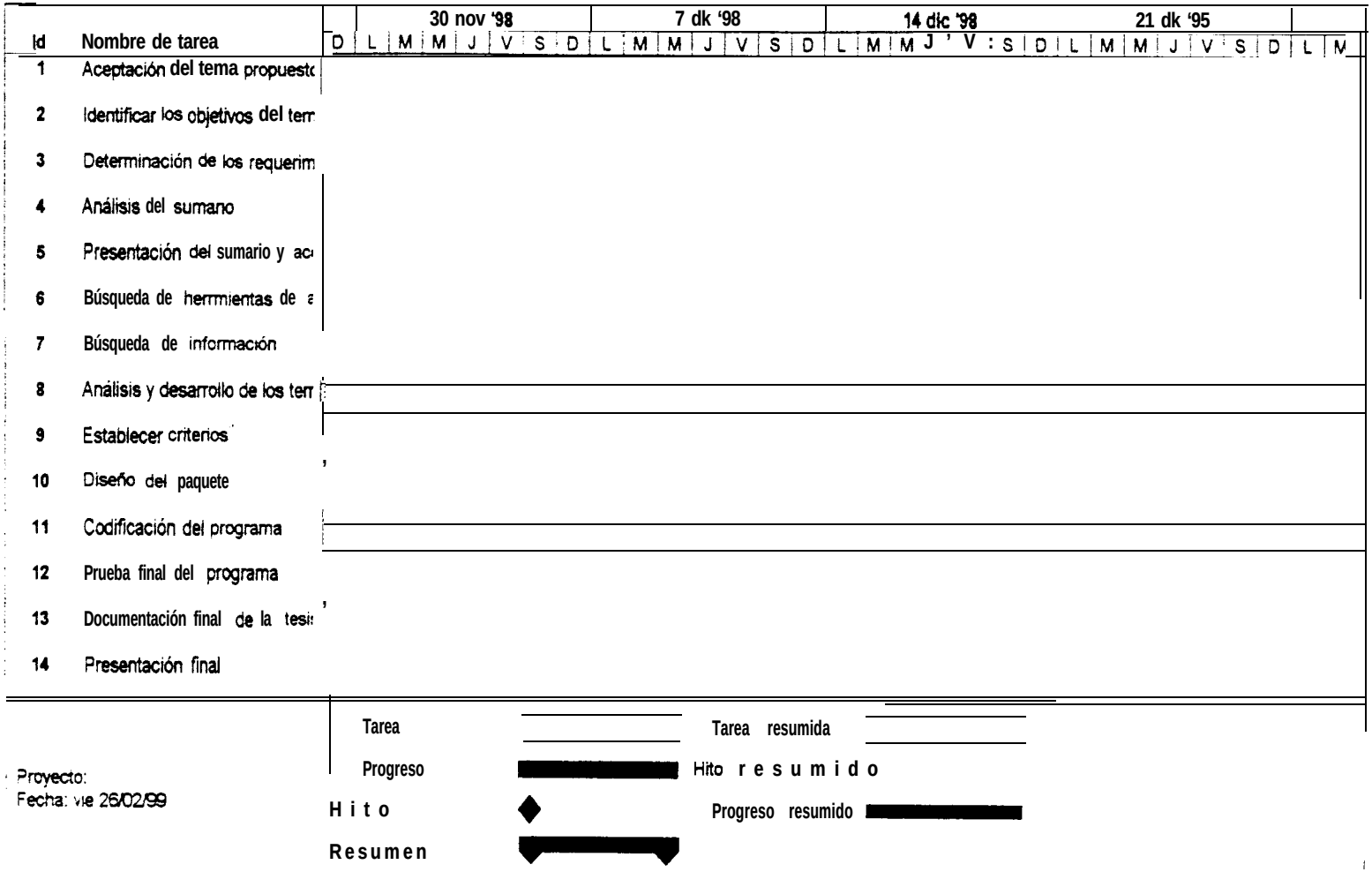

27 jul '98
3 ago '98
10 ago '98
17 ago '98
24 ago
M M J V S D L M M J V S D L M M J V S D L M M J V S D L M M J
Id
Nombre de tarea
1 Aceptación del tema propuest
2 Identificar los objetivos del terr
3 Determinación de los requerirr
4 Análisis del sumario
5 Presentación del sumario y ac
6 Búsqueda de herrmientas de a
7 Búsqueda de información
8 Análisis y desarrollo de los ten
9 Establecer criterios
10 Diseño del paquete
11 Codificación del programa
12 Prueba final del programa
13 Documentación final de la tesi
14 Presentación final
Tarea
Progreso
Hito
Resumen
Tarea resumida
Hito resumido
Progresa resumido
Proyecto:
Fecha: vie 26/02/99


'98
31 ago '98
7 sep '98
14 sep '98
21 sep '98
V S D L M M J V S D L M M J V S D L M M J V S D L M M J V S D
Id
Nombre de tarea
1 Aceptación del tema propuesto
2 Identificar los objetivos del tem
3 Determinación de los requerim
4 Análisis del sumario
5 Presentación del sumario y ac
6 Búsqueda de herrmientas de a
7 Búsqueda de información
8 Análisis y desarrollo de los tem
9 Establecer criterios
10 Diseño del paquete
11 Codificación del programa
12 Prueba final del programa
13 Documentación final de la tesi
14 Presentación final
Tarea
Progreso
Hdo
Resumen
Tarea resumida
Hito resumido
Progreso resumido
Proyecto:
Fecha: vie 26/02/99

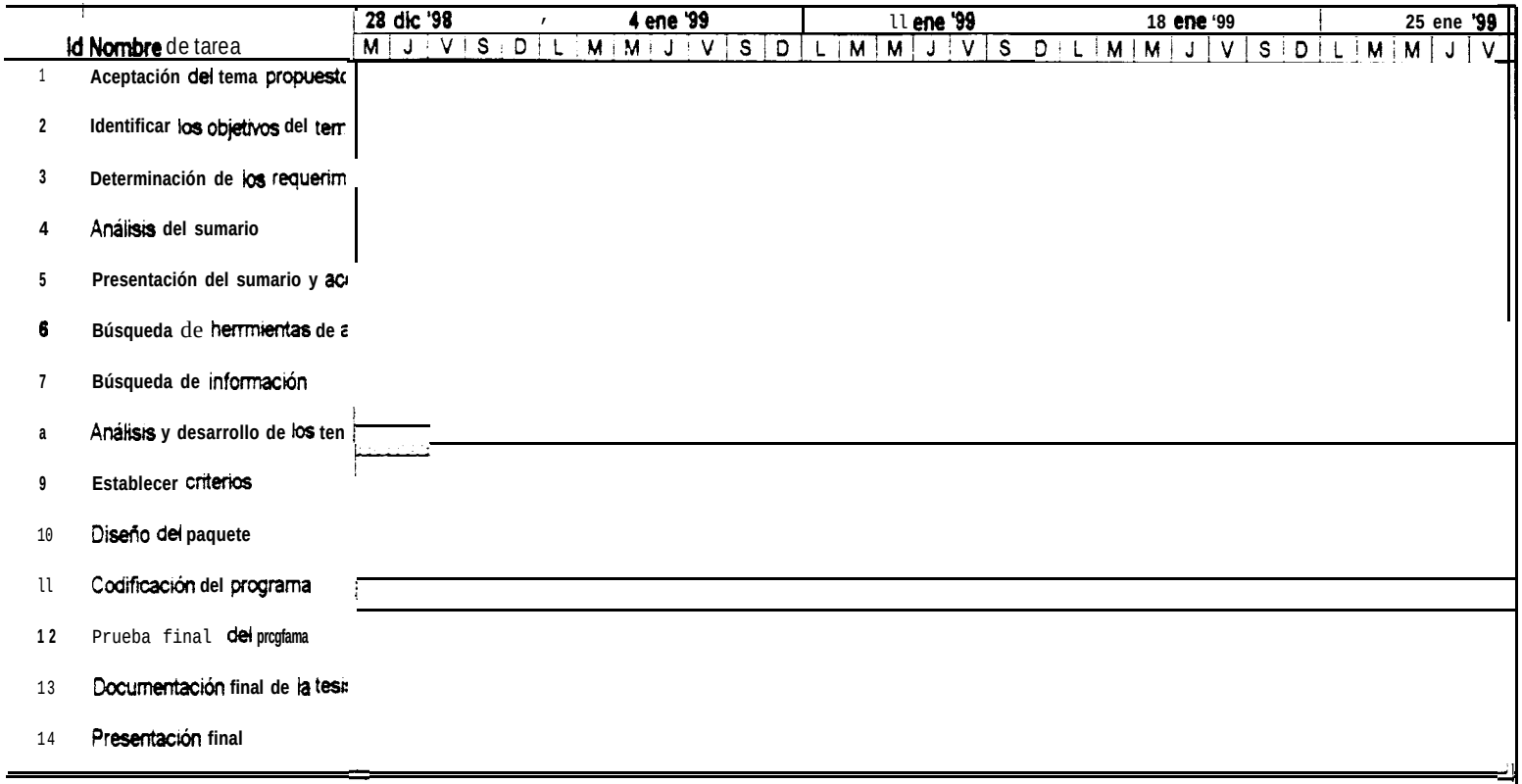

28 sep '98
5 oct '98
12 oct '98
19 oct '98
26
Id
Nombre de tarea
L M M J V S D L M M J V S D L M M J V S D L M M J V S D L M M
1 Aceptación del tema propuest
2 Identificar los objetivos del ter
3 Determinación de los requenm
4 Análisis del sumario
5 Presentación del sumario y ac
6 Búsqueda de herrmientas de a
7 Búsqueda de información
8 Análisis y desarrollo de los ter
9 Establecer criterios
10 Diseño del paquete
11 Codificación del programa
12 Prueba final del programa
13 Documentación final de la tes
14 Presentación final
Tarea
Progreso
Hito
Resumen
Tarea resumida
Hito resumido
Progreso resumido
Proyecto:
Fecha: vie 26/02/99


Id
Nombre de tarea
oct '98
2 nov '98
9 nov '98
16 nov '98
23 nov Y8
J V S D L M M J V S D L M M J V S D L M M J V S D L M M J V S
1 Aceptación del tema propuest
2 Identificar los objetivos del ter
3 Determinación de bs requerim
4 Análisis del sumario
5 Presentación del sumario y ac
6 Búsqueda de herrmientas de
7 Búsqueda de información
8 Análisis y desarrollo da los ten
9 Establecer criterios
10 Diseño del paquete
11 Codificación del programla
12 Prueba final del programa
13 Documentación final de la tes
14 Presentación final
Tarea
Progreso
Hito
Resumen
Tarea resumida
Hito resumido
Progresa resumido
Proyecto:
Fecha: vie 26/02/99

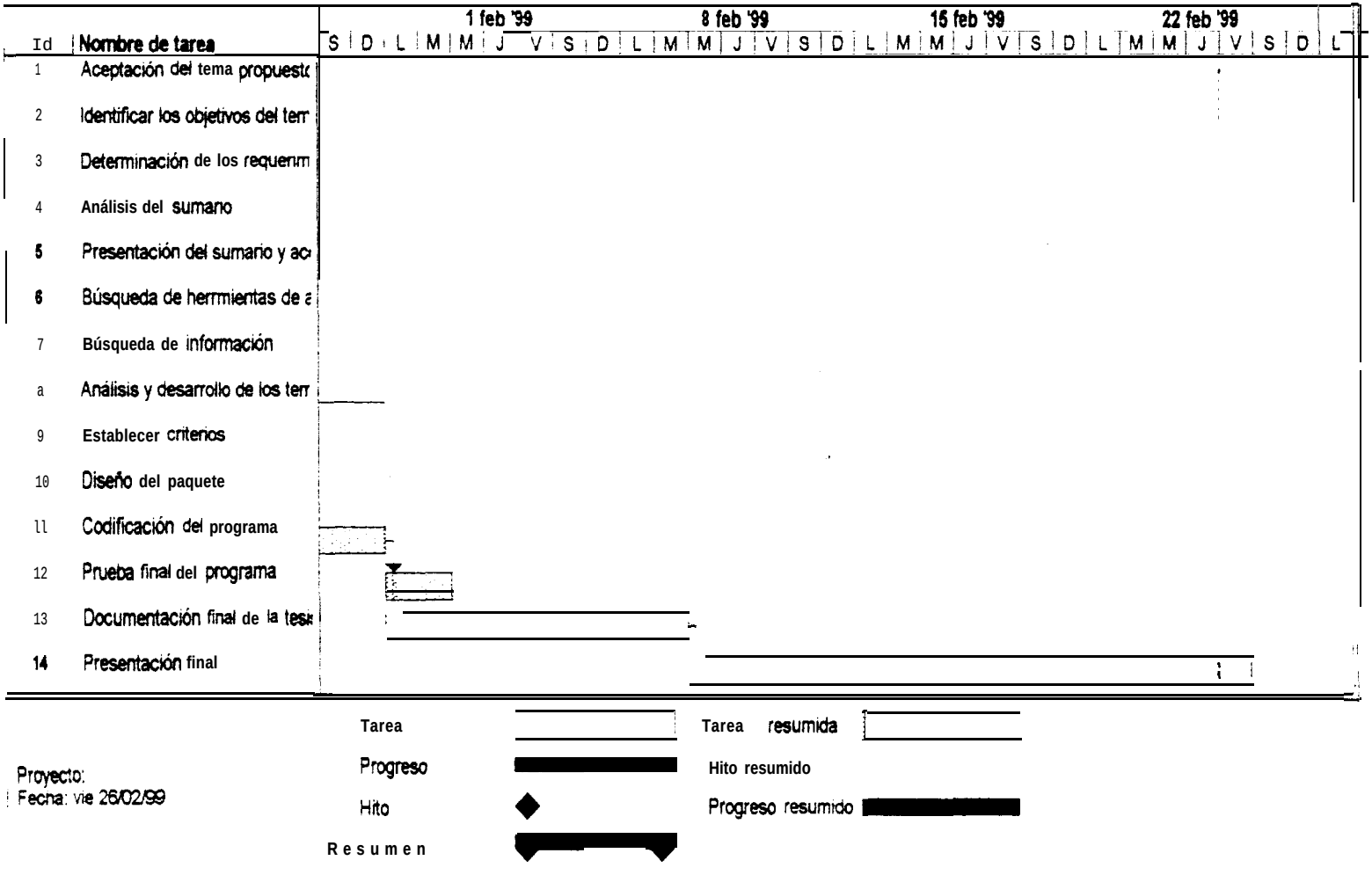
Id	Nombre de tarea
1	Aceptación del tema propuesto
2	Identificar los objetivos del tem
3	Determinación de los requerim
4	Análisis del sumario
5	Presentación del sumario y ac
6	Búsqueda de herrmientas de a
7	Búsqueda de información
8	Análisis y desarrollo de los tem
9	Establecer criterios
10	Diseño del paquete
11	Codificación del programa
12	Prueba final del programa
13	Documentación final de la tesi
14	Presentación final



Proyecto:
Fecha: vie 26/02/99



Proyecto:
Fecha: vie 26/02/99


Id
Nombre de tarea
1 feb '99
8 feb '99
15 feb '99
22 feb '99
S D L M M J V S D L M M J V S D L M M J V S D L M M J V S D L
1 Aceptación del tema propuesto
2 Identificar los objetivos del tem
3 Determinación de los requerim
4 Análisis del sumario
5 Presentación del sumario y ac
6 Búsqueda de herrmientas de a
7 Búsqueda de información
a Análisis y desarrollo de los tem
9 Establecer criterios
10 Diseño del paquete
11 Codificación del programa
12 Prueba final del programa
13 Documentación final de la tesi
14 Presentación final
Tarea
Tarea resumida
Progreso
Hito resumido
Hito
Progreso resumido
Resumen
Proyecto:
Fecha: vie 26/02/99

BIBLIOGRAFÍA

1. Ceballos Francisco. **ENCICLOPEDIA DEL LENGUAJE C.** Editorial RA-MA. Madrid, España. 1.99 1

2. Willey John. **GRÁFICAS PODEROSAS** CON **TURBO C++**. Editorial Limusa,S.A. México, D.F.,1.994

3. Wesley Addison. **PROGRAMACION DE GRÁFICOS EN TURBO C++.** Editorial Ben Ezzell. Massachusett, 1.990

4. Kendall Kenneth. **ANÁLISIS Y DISEÑO DE SISTEMAS.** Editorial Prentice Hall Hispanoamericana,S.A. México. 1.997

5. Internet. **BIBLIOTECAS.** http\\www.español.yahoo.com\Materiales_de_consulta