void far arc(int x, int y, int comienzo_angulo, int final_angulo, int radio);
Esta función creará un arco circular. El arco tiene como centro el punto especificado por los argumentos **x** e **y**, y es dibujado con el radio especificado: **radio**. El arco no está rellanado, pero es dibujado usando el color actual. El arco comienza al ángulo especificado por el argumento **comienzo_angulo** y es dibujado en la dirección contraria al de las agujas del reloj hasta llegar al ángulo especificado por el argumento **final_angulo**. La función arc usa el este (extendiéndose hacia la derecha del centro del arco en la dirección horizontal) como su punto de 0 grados. La función setlinestyle puede usarse para establecer el grosor del arco. La función arc, sin embargo, ignorará el argumento **trama** de la función setlinestyle.

void far bar(int izquierda, int superior, int derecha, int inferior);
Esta función dibujará una barra rectangular y rellenada de dos dimensiones. La esquina superior izquierda de la barra rectangular está definida por los argumentos **izquierda** y **superior**. Estos argumentos corresponden a los valores x e y de la esquina superior izquierda. Similarmente, los argumentos **derecha** e **inferior** definen la esquina inferior derecha de la barra. La barra no tiene borde, pero es rellenada con la trama de relleno actual y el color de relleno como es establecido por la función setlinestyle.

void far bar3d(int izquierda, int superior,int derecha, int inferior, int profundidad, int banderin_tapa);
Esta función creará una barra rectangular y rellenada de tres dimensiones. La esquina superior izquierda de la barra rectangular más frontal está definida por los argumentos **izquierda** y **superior**. Estos argumentos corresponden a los valores x e y de la esquina superior izquierda del rectángulo más frontal. Similarmente, los argumentos **derecha** e **inferior** definen la esquina inferior derecha del rectángulo más frontal. La barra tiene borde, en todas las tres dimensiones, rellenada con el color y estilo de línea actuales. El rectángulo más frontal es rellenado usando la trama de relleno actual y el color de relleno como es establecido por la función setlinestyle. El argumento **banderin_tapa** es usado para especificar si es o no es posible apilar varias barras encima de cada una. Si **banderin_tapa** tiene un valor distinto a cero, entonces la barra está "tapada". Si **banderin_tapa** tiene un valor de cero, entonces la barra no está "tapada", permitiendo otras barras ser apiladas encima de ésta.

void far circle(int x, int y, int radio);
Esta función se usas para dibujar un círculo. Los argumentos **x** e **y** definen el centro del círculo, mientras que el argumento **radio** define el radio del círculo. El círculo no es rellenado pero es dibujado usando el color actual. El grosor de la circunferencia puede ser establecido por la función setlinestyle; sin embargo, el estilo de la línea es ignorado por la función circle. La proporción anchura-altura para el modo actual es considerado cuando se calcula el círculo. Por esta razón, alterando los valores de defecto x e y de los factores de anchura-altura afectará el círculo (ya no seguirá siendo redondo).

void far cleardevice(void);
Esta función es usada para despejar una pantalla gráfica. La función cleardevice usa el color de fondo actual, como es establecido por la función setbkcolor, para rellenar la pantalla. La posición del cursor gráfico es la esquina superior izquierda de la pantalla - posición (0,0) - después de que la pantalla haya sido borrado.

void far clearviewport(void);
Esta función es usada para rellenar la pantalla actual del usuario con el color de fondo actual. El color de fondo puede ser establecido con la función setbkcolor. La posición del cursor gráfico es la esquina superior izquierda de la pantalla actual del usuario. Esta posición es (0,0) según la pantalla actual del usuario.

void far closegraph(void);
Esta función es usada para cerrar el sistema gráfico como es iniciada por la función initgraph. La función closegraph libera toda la memoria usada por el sistema gráfico y luego restaura el modo de vídeo al modo de texto que estaba en uso anteriormente a la llamada a la función initgraph.

void far detectgraph(int far *driver, int far *modo);
Esta función es usada para detectar el adaptador gráfico y el modo óptimo para usar con el sistema en uso. Si la función detectgraph no puede detectar ningún dispositivo gráfico, el argumento *driver es asignado grNotDetected (-2). Una llamada a graphresult resultará en un valor de retorno de -2, o grNotDetected.

Existen varios valores que indican los diferentes dispositivos gráficos que pueden ser usados por el argumento *driver. Un valor de 0, o DETECT, inicia la funcionalidad de autodetección, el cual determina el driver óptimo a usar.

Para cada dispositivo existen varios valores que indican los diferentes modos gráficos que pueden ser usados por el argumento *modo. Sin embargo, si el argumento *driver es asignado el valor de 0, o DETECT, el argumento *modo es automáticamente establecido al modo de resolución mas alto para el driver.

void far drawpoly(int numpuntos, int far *puntos);
Esta función es usada para crear un polígono con un número especificado de puntos. El argumento numpuntos es usado para definir el número de puntos en el polígono. Para la función drawpoly, el número de puntos debe ser el número actual de puntos más 1 para poder crear un polígono cerrado. En otras palabras, el primer punto debe ser igual al último punto. El argumento *puntos apunta a un array de números de longitud numpuntos multiplicado por 2. Los dos primeros miembros del array identifica las coordenadas x e y del primer punto, respectivamente, mientras que los dos siguientes especifican el siguiente punto, y así sucesivamente. La función drawpoly dibuja el perímetro del polígono con el estilo de línea y color actuales, pero no rellena el polígono.

void far ellipse(int x, int y, int comienzo_angulo,int final_angulo, int x_radio, int y_radio);
Esta función es usada para dibujar un arco elíptico en el color actual. El arco elíptico está centrado en el punto especificado por los argumentos x e y. Ya que el arco es eilptíco el argumento x_radio especifica el radio horizontal y el argumento y_radio especifica el radio vertical. El arco elíptico comienza con el ángulo especificado por el argumento comienzo_angulo y se extiende en un sentido contrario a las agujas del reloj al ángulo especificado por el argumento final_angulo. La función ellipse considera este - el eje horizontal a la derecha del centro del elipse - ser 0 grados. El arco elíptico es dibujado con el grosor de línea actual como es establecido por la función setlinestyle. Sin embargo, el estilo de línea es ignorado por la función ellipse.

void far fillellipse(int x, int y, int x_radio, int y_radio);
Esta función es usada para dibujar y rellenar una elipse. El centro de la elipse es especificado por los argumentos x e y. El argumento x_radio especifica el radio horizontal y el argumento y_radio especifica el radio vertical de la elipse. La elipse es dibujado con el perímetro en el color actual y rellenada con el color de relleno y la trama de relleno actuales.

void far fillpoly(int numpuntos, int far *puntos);
Esta función es usada para crear un polígono rellenado. El argumento **numpuntos** es usado para definir el número de puntos en el polígono. Al contrario que la función drawpoly, la función automáticamente cierra el polígono. El argumento ***puntos** apunta a un array de números de longitud **numpuntos** multiplicado por 2. Los dos primeros miembros del array identifica las coordenadas x e y del primer punto, respectivamente, mientras que los dos siguientes especifican el siguiente punto, y así sucesivamente. La función fillpoly dibuja el perímetro del polígono con el estilo de línea y color actuales. Luego, el polígono es rellenado con la trama de relleno y color de relleno actuales.

int far getbkcolor(void);
Esta función es usada para obtener el valor del color de fondo actual. El color de fondo, por defecto, es el color 0. Sin embargo, este valor puede cambiar con una llamada a la función setbkcolor.

Existen varios valores para ciertos colores de fondo.

La función getbkcolor retorna el valor del color de fondo actual.

Constante	Valor	Significado
BLACK	0	Negro
BLUE	1	Azul
GREEN	2	Verde
CYAN	3	Cían
RED	4	Rojo
MAGENTA	5	Magenta
BROWN	6	Marrón
LIGHTGRAY	7	Gris Claro
DARKGRAY	8	Gris Oscuro
LIGHTBLUE	9	Azul Claro
LIGHTGREEN	10	Verde Claro
LIGHTCYAN	11	Cían Claro
LIGHTRED	12	Rojo Claro
LIGHTMAGENTA	13	Magenta Claro
YELLOW	14	Amarillo
WHITE	15	Blanco

int far getcolor(void);
Esta función obtiene el valor del color actual. El color actual es el color usado para dibujar líneas, arcos, etc.. Este color no es el mismo que el color de relleno. El valor del color obtenido es interpretado según el modo que esté en uso.

Existen varios valores para ciertos colores de fondo.

Valor de retorno:

La función getcolor retorna el valor del color actual.

int far getmaxx(void);
Esta función es usada para obtener la coordenada máxima de la pantalla en la dirección horizontal. Este valor suele ser la resolución horizontal máxima menos 1.

Valor de retorno:

La función getmaxx retorna la coordenada máxima de la pantalla en la dirección horizontal.

int far getmaxy(void);
Esta función es usada para obtener la coordenada máxima de la pantalla en la dirección vertictal. Este valor suele ser la resolución vertictal máxima menos 1.

Valor de retorno:

La función getmaxy retorna la coordenada máxima de la pantalla en la dirección vertictal.

unsigned far getpixel(int x, int y);
Esta función es usada para obtener el valor del color del píxel especificado por los argumentos x e y. Estos argumentos especifican las coordenadas de la pantalla del píxel a ser evaluado. Cuando se evalúa el valor del color retornado, el modo gráfico en uso debe ser considerado.

Existen varios valores para describir colores.

int far getx(void);
Esta función es usada para obtener la posición, en la dirección horizontal, del cursor gráfico. El valor retornado especifica el lugar del píxel horizontal del cursor gráfico (la coordenada x), relativo a la pantalla del usuario actual.

Valor de retorno:

La función getx retorna la coordenada x del cursor gráfico.

int far gety(void);
Esta función es usada para obtener la posición, en la dirección vertical, del cursor gráfico. El valor retornado especifica el lugar del píxel vertical del cursor gráfico (la coordenada y), relativo a la pantalla del usuario actual.

Valor de retorno:

La función gety retorna la coordenada y del cursor gráfico.

void far initgraph(int far *driver, int far *modo, int far *path);
Esta función es usada para cargar o validar un dispositivo gráfico y cambiar el sistema de vídeo a modo gráfico. La función initgraph debe ser llamada antes de cualesquier funciones que generan una salida gráfica sean usadas.

Existen varios valores a ser usados para el argumento *driver. Si *driver es asignado a DETECT, ó 0, la función detectgraph es llamada, y un dispositivo y modo gráfico apropiados son seleccionados. Asignando a *driver cualquier otro valor predefinido inicia la carga del dispositivo gráfico correspondiente.

Existen varios valores a ser usados para el argumento *modo. Estos valores deberían corresponder al dispositivo especificado en el argumento *driver.

El argumento *path especificad el directorio donde los dispositivos gráficos están localizados. La función initgraph buscará el dispositivo primeramente en este directorio. Si no es encontrado, la función buscará en el directorio de inicio. Cuando el argumento *path es NULL, solamente el directorio de inicio es buscado.

Otra forma para evitar cargando el dispositivo desde el disco cada vez que el programa es ejecutado es ligarlo o enlazarlo al dispositivo apropiado en un programa ejecutable.

Valor de retorno:

La función initgraph no retorna ningún valor. Sin embargo, cuando la función initgraph es llamada, el código de error interno es activado. Si la función initgraph termina con éxito, el código es asignado un 0. Si no, el código es asignado así:

-2	grNotDetected	La tarjeta gráfica no se encontró
-3	grFileNotFound	El fichero del dispositivo no se encontró
-4	grInvalidDriver	El fichero del dispositivo es inválido
-5	grNoLoadMem	No hay suficiente memoria para cargar el dispositivo

void far line(int x1, int y1, int x2, int y2);
Esta función es usada para conectar dos puntos con una línea recta. El primer punto es especificado por los argumentos x1 e y1. El segundo punto es especificado por los argumentos x2 e y2. La línea se dibuja usando el estilo de línea actual, el grosor, y el color actual. La posición del cursor gráfico no es afectado por la función line.

void far linerel(int dx, int dy);
Esta función es usada para dibujar una línea recta a una distancia y dirección predeterminadas desde la posición actual del cursor gráfico. El argumento dx especifica el número relativo de píxels para atravesar en la dirección horizontal. El argumento dy especifica el número relativo de píxels para atravesar en la dirección vertical. Estos argumentos pueden ser tanto valores positivos como negativos. La línea se dibuja usando el estilo de línea actual, el grosor, y el color actual desde la posición actual del cursor gráfico a través de la distancia relativa especificada. Cuando la línea esté terminada, la posición del cursor gráfico es actualizado al último punto de la línea.

void far lineto(int x, int y);
Esta función es usada para dibujar una línea recta desde la posición actual del cursor gráfico hasta el punto especificado por los argumentos x e y. La línea se dibuja usando el estilo de línea actual, el grosor, y el color actual. Después de que la línea recta haya sido dibujado, la posición del cursor gráfico es actualizado a la posición especificado por los argumentos x e y (el punto final de la línea).

void far moveto(int x, int y);
Esta función es usada para colocar el cursor gráfico al punto especificado por los argumentos x e y. Ya que el cursor es movido desde su posición anterior al punto especificado por los argumentos x e y, no hay dibujo alguno.

Esta función es usada para mostar una cadena de caracteres. El argumento *cadena_texto define la cadena de texto a ser mostrado. La cadena es mostrado donde está el cursor gráfico actualmente usando el color actual y fuente, dirección, valores, y justificaciones de texto. La posición del cursor permanece sin ser cambiado al menos que la justificación horizontal actual es LEFT_TEXT y la orientación del texto es HORIZ_DIR. Cuando esto sea el caso, la posición del cursor es colocada horizontalmente a la anchura del píxel de la cadena de texto. Además, cuando se use la fuente por defecto, cualquier texto que se extiende a fuera del área gráfica actual es truncado.

Aunque la función outtext está diseñada para texto sin formato, texto con formato puede ser mostrada a través del uso de un búfer de caracteres y la función sprintf.

void far outtextxy(int x, int y, char far *cadena_texto);
Esta función es usada para mostar una cadena de caracteres. El argumento *cadena_texto define la cadena de texto a ser mostrado. La cadena es mostrada en la posición descrita por los argumentos x e y usando el color actual y fuente, dirección, valores, y justificaciones de texto. Cuando se use la fuente por defecto, cualquier texto que se extiende fuera del área gráfica actual es truncado.

Aunque la función outtextxy está diseñada para texto sin formato, texto con formato puede ser mostrada a través del uso de un búfer de caracteres y la función sprintf.

void far putpixel(int x, int y, int color);
Esta función es usada para asignar el valor del color a un píxel en particular. La posición del píxel en cuestión está especificado por los argumentos x e y. El argumento color especifica el valor del color del píxel.

Existen varios valores para describir colores.

void far rectangle(int izquierda, int superior, int derecha, int inferior);
Esta función dibujará un rectángulo sin rellenar su interior usando el color actual. La esquina superior izquierda del rectángulo está definida por los argumentos izquierda y superior. Estos argumentos corresponden a los valores x e y de la esquina superior izquierda. Similarmente, los argumentos derecha e inferior definen la esquina inferior derecha del rectángulo. El perímetro del rectángulo es dibujado usando el estilo y grosor de línea actuales.

void far setbkcolor(int color);
Esta función es usada para asignar el color de fondo al valor del color de fondo especificado por el argumento color.

Existen varios valores para ciertos colores de fondo.

void far setfillpattern(char far *trama, int color);
Esta función es usada para seleccionar una trama de relleno definido por el usuario. El argumento **trama** apunta a una serie de ocho bytes que representa una trama de relleno de bits de 8 x 8. Cada byte representa una fila de ocho bits, donde cada bit está encendido o no (1 ó 0). Un bit de 0 indica que el píxel correspondiente será asignado el color de relleno actual. Un bit de 0 indica que el píxel correspondiente no será alterado. El argumento **color** especifica el color de relleno que será usado para la trama.

void far setfillstyle(int trama, int color);
Esta función es usada para seleccionar una trama predefinida y un color de relleno. El argumento **trama** especifica la trama predefinida, mientras que el argumento **color** especifica el color de relleno.

Existen trece valores ya definidos para tramas. Sin embargo, la trama USER_FILL (valor 12) no debería usarse para asignar unla trama definida por el usuario. En su lugar, se debería usar la función setfillpattern.

void far setlinestyle(int estilo, unsigned trama, int grosor);
Esta función es usada para definir las características de líneas para líneas rectas.

El argumento **estilo** especifica la trama de línea predefinida para su uso. El argumento **trama** es una trama de 16 bits que describe el estilo de línea cuando el argumento **estilo** es USERBIT_LINE, ó 4. Un bit 1 en esta trama indica que el píxel correspondiente será asignado el color actual. Un bit 0 indica que el píxel correspondiente no será alterado. El argumento **grosor** define el grosor de la línea.

Existen varios valores para los diferentes estilos y grosores de líneas rectas.

Estilos de Líneas

Constante	Valor	Significado
SOLID_LINE	0	Línea continua
DOTTED_LINE	1	Línea hecha con puntos
CENTER_LINE	2	Línea centrada
DASHED_LINE	3	Línea discontinua
USERBIT_LINE	4	Línea definida por el usuario

Grosores para Líneas

NORM_THICK	1	Grosor es de 1 píxel
THICK_WIDTH	3	Grosor es de 3 píxels

void far settextstyle(int fuente, int orientacion, int tam_caracter);
Esta función es usada para especificar las características para la salida de texto con fuente. El argumento **fuente** especifica la fuente registrada a usar. La fuente ha de estar registrada para resultados predecibles; es decir, usa registerbgifont antes de usar esta función. El argumento **orientacion** especifica la orientación en que el texto ha de ser mostrado. La orientación por defecto es HORIZ_DIR. El argumento **tam_caracter** define el factor por el cual la fuente actual será multiplicada. Un valor distinto a 0 para el argumento **tam_caracter** puede ser usado con fuentes escalables o de bitmap. Sin embargo, un valor distinto a 0 para el argumento **tam_caracter**, el cual selecciona el tamaño del carácter definido por el usuario usando

la función setusercharsize, solamente funciona con fuentes escalables. El argumento **tam_caracter** puede agrandar el tamaño de la fuente hasta 10 veces su tamaño normal.

Existen varios valores y constantes para las justificaciones.

Fuentes para Texto

Constante	Valor	Significado
DEFAULT_FONT	0	Fuente bitmap de 8x8
TRIPLEX_FONT	1	Fuente escalable de tipo triple
SMALL_FONT	2	Fuente escalable pequeña
SANS_SERIF_FONT	3	Fuente escalable de tipo sans serif
GOTHIC_FONT	4	Fuente escalable de tipo gótico
SCRIPT_FONT	5	Fuente escalable de tipo manuscrito
SIMPLEX_FONT	6	Fuente escalable de tipo manuscrito simple
TRIPLEX_SCR_FONT	7	Fuente escalable de tipo manuscrito triple
COMPLEX_FONT	8	Fuente escalable de tipo complejo
EUROPEAN_FONT	9	Fuente escalable de tipo europeo
BOLD_FONT	10	Fuente escalable en negrita

Orientaciones para Texto

Constante	Valor	Significado
HORIZ_DIR	0	Texto horizontal
VERT_DIR	1	Texto vertical

int far textheight(char far *texto);
Esta función es usada para determinar la altura, en píxels, de la cadena de texto especificada por el argumento ***texto**. La altura del texto se determina usando la fuente actual y el tamaño del carácter.

int far textwidth(char far *texto);
Esta función es usada para determinar la anchura, en píxels, de la cadena de texto especificada por el argumento ***texto**. La anchura del texto se determina usando la fuente actual y el tamaño del carácter.

Valor de retorno:

La función textwidth retorna la anchura, en píxels, del texto especificado por el argumento.

Referencia:
http://c.conclase.net/Borland/index.php