CONCEPTOS BÁSICOS DE GRAFICACIÓN EN C++

Bruno López Takeyas
Instituto Tecnológico de Nuevo Laredo
Reforma Sur 2007, C.P. 88250, Nuevo Laredo, Tamps. México
http://www.itnuevolaredo.edu.mx/takeyas
E-mail: takeyas@itnuevolaredo.edu.mx

Resumen: *En numerosas ocasiones los estudiantes requieren representar gráficamente el comportamiento de modelos matemáticos, estadísticos, de investigación de operaciones, etc. mediante programas computacionales diseñados "a la medida" sin necesidad de apoyarse en paquetes de cómputo comerciales. Por esto, se presentan las siguientes consideraciones básicas de graficación en lenguaje C++,*

Palabras claves: *Graficación, resolución, píxel, lenguaje C++, monitor.*

1 INTRODUCCIÓN

Tal como un artista selecciona diversos medios para representar sus pinturas, los programadores, escogen un modo y formato especial para habilitar el monitor para graficar. Cada modo proporciona ciertas características como la resolución, número posible de colores, modo texto o modo gráfico y otros elementos donde cada modo requiere de cierto equipo (hardware).

1.1 Resolución

Las imágenes gráficas mostradas en un monitor de computadora se componen de pequeños puntos llamados píxeles, los cuales están distribuidos en la pantalla en filas; existe una cantidad específica de filas y cada fila tiene una cantidad específica de píxeles. La cantidad de píxeles usada en la pantalla se conoce como resolución. Cada modo gráfico tiene una resolución particular.

1.2 Inicializar el monitor en modo gráfico

Para habilitar el monitor en modo gráfico y utilizar sus píxeles y funciones de gráficos, es necesario incluir el encabezado **`#include <graphics.h>`** que contiene las declaraciones y funciones relacionadas con graficación e inicializar el monitor en modo gráfico y utilizar sus píxeles con la función ***`initgraph()`***. Dicha función requiere las siguientes declaraciones:

```
int monitor=DETECT;
// Variable para detectar el tipo
// de monitor
int modo;
// Modo de operación del monitor
```

también se puede declarar e inicializar con un tipo de monitor específico como:

```
int monitor=VGA;
// Variable para usar el monitor
// tipo VGA
int modo=VGAHI;
//  Usar el monitor VGA a su
//maxima resolución
```

Fig. 1. Declaración de variables para habilitar el monitor en modo gráfico

Para terminar de usar el monitor en modo gráfico y devolverlo a su modo de texto normal se usa la función ***`closegraph()`***.

1.2.1 La función initgraph()

Una vez declaradas las variables monitor y modo que controlarán la resolución identificando el tipo de pantalla o monitor y su modo de operación respectivamente, se utiliza la función `initgraph()` para habilitar el monitor seleccionado en modo gráfico. La función `initgraph()` tiene 3 parámetros o argumentos:

1) La variable que identifica el monitor.
2) El modo de operación gráfico.
3) Subdirectorio que contiene los controladores de los monitores (archivos con extensión BGI) y los archivos con los tipos de letra (extensión CHR) como lo muestra la Fig.2.

```
int monitor=DETECT, modo;

initgraph(&monitor,&modo,"\\tc\\bgi");
```

Fig. 2. La función initgraph().

Si se desea usar el directorio actual por defecto, se utiliza la función `initgraph()` como lo indica la Fig. 3.

```
int monitor=DETECT, modo;

initgraph(&monitor,&modo,"");
```

Fig. 3. La función initgraph() usando el subdirectorio actual por defecto.

1.3 Uso de coordenadas

Una vez que se inicializa el monitor en modo gráfico, las coordenadas tienen al píxel como unidad de medida. La función ***`getmaxx()`*** calcula la cantidad de píxeles por renglón y la función ***`getmaxy()`*** calcula la cantidad de renglones de la pantalla.

Las funciones de gráficos tienen como estándar el orden de manejo de coordenadas como columna, renglón; es decir, primero se anota la columna y después el renglón para posicionarse en dicha coordenada.

Cabe destacar que el conteo de columnas y renglones inicia partiendo de la esquina superior izquierda del monitor.

2 LÍNEAS, FIGURAS GEOMÉTRICAS, COLORES Y RELLENOS

Sería muy difícil considerar todas las opciones posibles de todas las funciones de graficación; sin embargo, en este artículo se tratan los temas fundamentales para implementarlas. Básicamente se mostrará que antes de utilizar un color, un tipo de línea, de relleno, o cualquier función de definición del tipo de trazo, etc. es necesario definirlo. A continuación se muestran las funciones básicas de graficación.

2.1 La función line()

Esta función se utiliza para dibujar una línea entre 2 puntos. Para ello, la función requiere 4 parámetros que representan las coordenadas (en pixeles) de los dos puntos que se desea unir mediante una línea recta. La Fig. 4 muestra un ejemplo que une los puntos 50,100 y 300,200 (columna, renglón respectivamente).

```
line(50,100,300,200);
```

Fig. 4. La función line().

2.2 La función setlinestyle()

Esta función se utiliza para determinar el tipo de línea o trazo que se desea. Se pueden utilizar trazos con línea continua, línea punteada, línea interrumpida, o un patrón de línea definido por el usuario. Esta función requiere 3 argumentos:

1) Tipo de línea: Puede ser SOLID_LINE, DOTTED_LINE, CENTER_LINE, DASHED_LINE o USERBIT_LINE.
2) Patrón: Este argumento regularmente es ignorado (excepto cuando se trata de un tipo de línea definido por el usuario).
3) Ancho de línea: Define la amplitud del trazo.

La Fig. 5 muestra un ejemplo que une los puntos 50,100 y 300,200 con una línea punteada.

```
setlinestyle(DOTTED_LINE,0, NORM_WIDTH);
line(50,100,300,200);
```

Fig. 5. La función setlinestyle().

2.3 La función circle()

Esta función dibuja un círculo y requiere 3 argumentos:

1) Coordenada de la columna del centro (en pixeles).
2) Coordenada del renglón del centro (en pixeles).
3) Radio del círculo (en pixeles).

La Fig. 6 dibuja un círculo cuyo centro se encuentra en el punto 300,150 y su radio es de 27 pixeles.

```
circle(300,150,27);
```

Fig. 6. La función circle().

2.4 La función rectangle()

Esta función dibuja un rectángulo indicando las coordenadas de las esquinas superior izquierda e inferior derecha respectivamente. La Fig. 7 muestra un ejemplo de una función que dibuja un rectángulo desde el punto 50,100 hasta el punto 400,250.

```
rectangle(50,100,400,250);
```

Fig. 7. La función rectangle().

2.5 La función setcolor()

Se utiliza esta función para definir el color de los trazos siguientes; es decir, antes de dibujar un trazo de un color específico, éste debe definirse. Esta función sólo tiene un argumento que representa el código del color deseado. P. ejem. BLACK, RED, BLUE, GREEN, YELLOW, etc. o bien su número entero correspondiente. La Fig. 8 muestra la tabla de colores y sus respectivos valores.

BLACK	0
BLUE	1
GREEN	2
CYAN	3
RED	4
MAGENTA	5
BROWN	6
LIGHTGRAY	7
DARKGRAY	8
LIGHTBLUE	9
LIGHTGREEN	10
LIGHTCYAN	11
LIGHTRED	12
LIGHTMAGENTA	13
YELLOW	14
WHITE	15

Fig. 8. Tabla de colores y sus valores.

La Fig. 9 muestra un ejemplo del uso de la función setcolor() donde se dibuja un circulo de color rojo y después un rectangulo de color azul.

```
setcolor(RED);
circle(300,150,27);

setcolor(BLUE);
rectangle(50,100,400,250);
```

Fig. 9. Uso de la función setcolor().

2.6 Definiendo el tipo de relleno de una figura con la función setfillstyle()

Si se desea rellenar una figura, es necesario definir previamente el patrón y color del relleno. La Fig. 10 muestra los patrones de relleno disponibles.

PATRÓN		DESCRIPCIÓN
EMPTY_FILL	0	Color del fondo
SOLID_FILL	1	Relleno sólido con el color determinado
LINE_FILL	2	Relleno con línea (----)
LTSLASH_FILL	3	Relleno con /// líneas de ancho normal
SLASH_FILL	4	Relleno con /// líneas
BKSLASH_FILL	5	Relleno con \\\ líneas
LTBKSLASH_FILL	6	Relleno con \\\ líneas de ancho normal
HATCH_FILL	7	Relleno de líneas

		cruzadas ligeras
XHATCH_FILL	8	Relleno de líneas cruzadas gruesas
INTERLEAVE_FILL	9	Relleno de líneas
WIDE_DOT_FILL	10	Relleno de puntos espaciados
CLOSE_DOT_FILL	11	Relleno de puntos cercanos
USER_FILL	12	Relleno definido por el usuario

Fig. 10. Patrones de relleno de la función setfillstyle()

Por ejemplo, si se desea definir el patrón de relleno de puntos cercanos de color rojo, se usa la función `setfillstyle()` como lo muestra la Fig. 11.

```
setfillstyle(CLOSE_DOT_FILL, RED);
```

Fig. 11. Selección del patrón de relleno CLOSE_DOT_FILL de color RED

2.7 La función floodfill()

Una vez seleccionado el patrón de relleno mediante la función `setfillstyle()`, se procede a rellenar una figura usando la función `floodfill()`. Es muy importante resaltar que la figura que se desea rellenar esté completamente cerrada, ya que esta función contiene un algoritmo que busca el contorno de la figura y, en caso de encontrar una apertura, la función extralimitará la figura y también rellenará la parte externa de la misma. La función `floodfill()` requiere identificar un punto que se encuentre dentro del contorno de la figura y necesita 3 argumentos:

1) Coordenada de la columna del punto interno de la figura.
2) Coordenada del renglón del punto interno de la figura.
3) Color del borde de la figura.

El ejemplo mostrado en la Fig. 12 dibuja un circulo de color ROJO y lo rellena de color AZUL sólido.

```
setcolor(RED);
circle(300,150,27);

setfillstyle(SOLID_FILL,BLUE);
floodfill(300,150,RED);
```

Fig. 12. Uso de la función floodfill().

3 CÓMO MOSTRAR MENSAJES EN MODO GRÁFICO

Aunque el monitor se encuentre habilitado en modo gráfico, se puede combinar la colocación de texto y gráficas en la pantalla; sin embargo, en esta sección se hace énfasis en la colocación de mensajes en formato gráfico, definiendo el tipo de letra, dirección y tamaño del mensaje deseado.

3.1 La función settextstyle()

Antes de mostrar un mensaje, debe seleccionarse el tipo de letra, dirección y tamaño del mensaje mediante la función `setfillstyle()` la cual requiere 3 argumentos:

1) El tipo de letra. (Ver Fig. 13).
2) La dirección del mensaje (horizontal o vertical).
3) Tamaño.

Tipo letra	Valor	Archivo
DEFAULT_FONT	0	-
TRIPLEX_FONT	1	TRIP.CHR
SMALL_FONT	2	LITT.CHR
SANS_SERIF_FONT	3	SANS.CHR
GOTHIC_FONT	4	GOTH.CHR

Fig. 13. Tipos de letra.

3.2 La función outtextxy()

Una vez definido el tipo de letra, dirección y tamaño correspondiente, se usa la función outtextxy() para desplegar un mensaje gráfico en la pantalla. Esta función requiere 3 argumentos:

1) Coordenada de la columna donde se desea mostrar el mensaje.
2) Coordenada del renglón.
3) Mensaje a mostrar.

La Fig. 14 muestra un ejemplo.

```
settextstyle(GOTHIC_FONT, HORIZ_DIR, 5);
outtextxy(100,200,”Tec Laredo”);
```

Fig. 14. Mostrar un mensaje gráfico en la pantalla.

4 APLICACIÓN TÍPICA

El programa de la Fig. 15 es un claro ejemplo del uso de líneas, figuras geométricas elementales, colores y rellenos

```
/*
Programa para graficar figuras
geometricas, lineas, texto, colores y
rellenos

 MiniTaller: Tecnicas avanzadas de
programacion en Lenguaje C++
 Instructor: M.C. Bruno Lopez Takeyas
*/

#include <graphics.h>  /* Encabezado con
declaraciones de graficos*/
#include <conio.h>
#include <stdio.h>

void main(void)
{
  int monitor=DETECT, modo; /*
Declaracion de tipo de monitor y modo*/
/* Automaticamente detecta el tipo de
monitor*/

  initgraph(&monitor,&modo,"\\tc\\bgi");
/* Inicializa el modo grafico indicando
el monitor y modo utilizado*/
/* El subdirectorio \\tc\\bgi indica la
ruta de localizacion de los archivos
*.BGI (monitores) y *.CHR (tipos de
letras)*/


gotoxy(1,23);printf("getmaxx()=%d",getma
xx());

gotoxy(1,24);printf("getmaxy()=%d",getma
xy());

  setcolor(YELLOW); /* Establece el
color amarillo (de aqui en adelante los
trazos aparecen de este color*/
  line(0,0,50,50); /* Dibuja una linea
desde 0,0 hasta 50,50*/

  setcolor(WHITE);  /*Establece el color
blanco*/
  circle(100,200,30); /* Dibuja un
circulo cuyo centro esta en 100,200 y de
radio=30 pixeles*/

  setfillstyle(LINE_FILL,RED); /*
Establece el relleno de lineas rojas*/
  floodfill(100,200,WHITE); /*Rellena el
contorno desde 100,200 hasta encontrar
un trazo blanco*/

  rectangle(200,100,300,200); /* Dibuja
un rectangulo desde 200,100 hasta
300,200*/
  setfillstyle(HATCH_FILL,BLUE); /*
Establece el relleno como cuadricula*/
  floodfill(250,150,WHITE); /* Rellena
el contorno desde 100,200 hasta
encontrar un trazo blanco*/

  setcolor(GREEN); /*Establece el color
verde*/
  settextstyle(GOTHIC_FONT,HORIZ_DIR,5);
/* Establece el font como Gotico
en posicion Horizontal de tamaño 5*/
  outtextxy(330,100,"Gothic");
/*Despliega el mensaje "Gothic" en
330,100*/

  setcolor(CYAN); /*Establece el color
celeste*/

settextstyle(SANS_SERIF_FONT,VERT_DIR,7)
;
/* Establece el font como Sanserif en
posicion Vertical de tamaño 7*/
  outtextxy(330,200,"Sanserif");/*
Despliega el mensaje "Sanserif" en
330,200*/

  getch();
  closegraph(); /* Termina el modo
grafico (vuelve a su modo de texto
normal)*/
  return;
}
```

Fig. 15.- Aplicación típica.

5 CONCLUSIONES

Aunque existen muchas otras funciones de graficación, este artículo presenta los conceptos y funciones básicas para iniciar la codificación de

programas en C++ que permitan utilizar el monitor en modo gráfico. Aquí se muestran las operaciones fundamentales de graficación y se presentan ejemplos representativos, los cuales pueden obtenerse en el sitio http://www.itnuevolaredo.edu.mx/takeyas o bien solicitarse al autor escribiendo un correo electrónico a takeyas@itnuevolaredo.edu.mx.

6 BIBLIOGRAFÍA

• Barkakati Nabajyoti. **"The Waite Group´s. Turbo C Bible"**. Howard W. Sams & Company. Estados Unidos. 1990.

• Deitel y Deitel. **"C++ Cómo programar"**. Segunda edición. Pearson-Prentice Hall. Estados Unidos. 1999.

• Lafore, Robert. **"The Waite Group´s. Turbo C. Programming for the PC"**. Revised Edition. Howard W. Sams & Company. Estados Unidos. 1990.

• López Takeyas, Bruno. **"Minitaller: Técnicas avanzadas de programación en lenguaje C++".** Instituto Tecnológico de Nuevo Laredo, Tam. México. 2003.

• Schildt, Herbert. **"Turbo C. Programación avanzada"**. Segunda edición, McGraw Hill. Estados Unidos. 1990.