LINUX • JAVA • UNIX • DEMOSCENE • ENSAMBLADOR • PROGRAMACIÓN 3D • C

PROGRAMACIÓN ACTUAL

AÑO 1 • NÚMERO 4

ARGENTINA 10$ • CHILE 3000$ • PORTUGAL 1500$

WATCOM C 11.0
POR SIEMPRE C++

SÓLO 995 ptas

SECCIÓN NIVEL BÁSICO
Arrays en el **CURSO DE PROGRAMACIÓN**, Punteros en el **CURSO DE C**, funcionamiento de la pila en **ENSAMBLADOR**

SECCIÓN LAS EMPRESAS DEMANDAN
Análisis del nuevo **VISUAL BASIC 5.0**, empezamos a estudiar en profundidad el lenguaje SQL en **BASES DE DATOS**

SECCIÓN BAJO NIVEL
Pintando polígonos en **ENTORNOS 3D**, el efecto fuego en **DEMOSCENE**, programación de la paleta en **PROGRAMACIÓN GRÁFICA**

SECCIÓN CAMPUS
Control de Frames y ventanas en **JAVASCRIPT**, GNU/make en **LINUX**, concentradores en **REDES LOCALES**, primeros programas en **ADA**

PROGRAMACIÓN 3D: PINTANDO POLÍGONOS

A FONDO: VISUAL BASIC 5.0

Prensa Técnica

DELPHI 3.0, BORLAND A LA CARGA

GRATIS EN NUESTRO **CD ROM**

Netscape Communicator (versión final)
Microsoft Internet Explorer 4.0 (versión beta)
Linux RED HAT 4.2

Estamos en plena "Linuxmania" el sistema operativo de moda, última versión en exclusiva

Programación de la paleta

En el anterior artículo vimos cómo hacer dibujos de primitivas para hacer presentaciones, pero utilizábamos los colores por defecto que nos ofrece el ordenador. Ahora aprenderemos a cambiar los colores a nuestro antojo para dar a nuestros trabajos más vistosidad.

En esta nueva entrega se va a tratar la paleta de colores del ordenador y los posibles efectos que se pueden realizar gracias a su modificación, como el fundido de una imagen a negro, la rotación de los colores o el fundido desde el negro a una imagen.
En las demos e intros se utilizan algunos de estos efectos debido a la facilidad de programación.
Para representar los colores el PC se sirve de una paleta que utiliza de forma parecida a la de los pintores. En ella se realizan mezclas para conseguir unas nuevas tonalidades o nuevos colores que se puedan necesitar.
Cada color está constituido por una mezcla tres colores básicos : el rojo, el verde y el azul. A estos se les llama componentes porque la union de estas tres dan como resultado un color. Por lo tanto, una paleta estará constituida por tres componentes para cada color.
Para realizar estas mezclas se utiliza un sistema llamado RGB (del inglés red-green-blue) que combina estos tres colores básicos. El porqué de la utilización de este sistema es porque el monitor genera el color centrando los tres haces de luz en un solo punto. Cada haz es de un color básico, es decir, un haz es rojo, otro verde y el otro azul, por lo que al unirlos se verá en la pantalla el color que corresponda a la mezcla de las tres intensidades de cada haz. Basta con acercarse a una televisión para comprobar que cada punto está compuesto por tres pequeños puntos, cada uno activado con una intensidad.
De esta forma, para conseguir el color blanco las tres componentes deberán estar al máximo, y para conseguir el color negro las tres componentes deben estar a cero. Siguiendo la misma tónica, el color rojo más puro la tarjeta lo genera con R=63, G=0 y B=0. Podemos generar así hasta 262.144 colores, como se verá en el apartado "El DAC de la Tarjeta gráfica".
De esta manera, la paleta de colores tomando como referencia el modo 320x200 a 256 colores, es simplemente un array de 768 elementos (256 colores * 3 componentes cada uno = 768) conteniendo las componentes de estos 256 colores que la VGA guarda internamente para que, al redibujar la pantalla (puede consultarse el proceso de retrazado en la entrega anterior), cuando la tarjeta encuentra en la VideoRAM el numero 0, por ejemplo, va a esta tabla y activa los 3 haces para que generen esa tonalidad RGB.
La paleta gráfica por defecto tiene unas componentes determinadas de manera que siempre el color 0 es el negro, el 1 el azul, etc... Si miramos la paleta por defecto de la VGA, estos 768 elementos que contiene almacenados la tarjeta tendría un aspecto como:

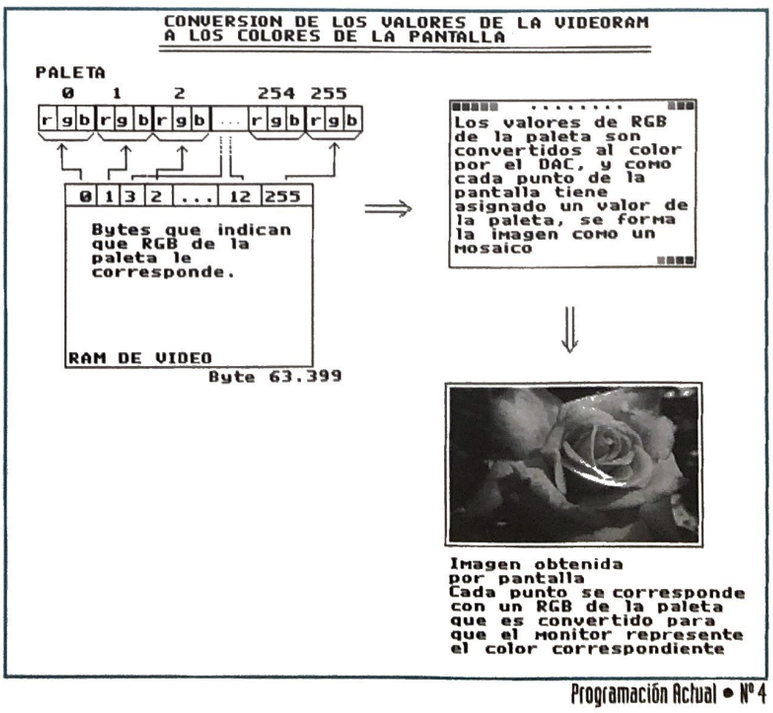
```
Paleta_Defecto[768] =
 { 0,0,0, 0,0,17, ... etc ... };
```

El color 0 es negro porque sus componentes son (0,0,0). De la misma manera, el color 1 es azul por defecto porque sus componentes son (0,0,17). El color 15 es blanco porque sus componentes son (63,63,63).
Si queremos que el color 0 en vez de ser negro (0,0,0) sea un amarillo pálido, podemos cambiar dentro de este array interno los valores (0,0,0) por, por ejemplo, (23,44,0).

```
Paleta_Defecto[0] = 23;
Paleta_Defecto[1] = 44;
Paleta_Defecto[2] = 0;
```

De esta misma manera, podemos cambiar el color 1, el 15, el 255... en resumen: podemos cambiar cualquier color a la tonalidad que deseemos. De ahí la necesidad de cambiar la paleta: para cada dibujo necesitamos unos colores en concreto (más verdes y menos rojos, todo grises, etc...).
En los ejemplos del CD que acompaña a la revista tenemos la imagen gráfica de una rosa. Este dibujo necesita muchos tonos de rosas que no hay en la paleta por defecto, y por eso necesitamos cambiar la paleta por defecto y elegir unas componentes que nos permitan disponer de estos colores. Por lo tanto cargamos la paleta original de la imagen (generada por el programa de dibujo durante su escaneado o creación), y que consiste en un array de 768 elementos con las componentes de los 256 colores. Ahora sólo necesitamos cambiar las componentes de la paleta por defecto por estas para que la tarjeta represente correctamente este dibujo.
El programa EJEMPLO3.EXE hace esto: primero carga en la VideoMemoria en 320x200 el dibujo de una rosa sin modificar la paleta (y por lo tanto ésta aparecerá con la paleta por defecto), y tras pulsar una tecla cambia esta paleta por la paleta correspondiente a la imagen.

FIGURA 1. RELACION ENTRE LOS VALORES DE LA VRAM Y LA PALETA.



EL DAC DE LA TARJETA GRAFICA

La paleta por definición nos delimita un número de colores máximo con el que se va a trabajar. Éste dependerá del modo de video en el que nos encontremos. Este límite viene dado por un "chip" de la tarjeta VGA llamado DAC. Por ejemplo, en el modo 13h como ya sabemos disponemos de 256 colores.
El DAC (digital-analog converter) se utiliza para convertir las señales digitales que genera el ordenador a señales analógicas comprensibles por el monitor.
De esta forma, y si el monitor esta capacitado para ello, con sólo cambiar nuestra tarjeta de vídeo por otra con un DAC mejor se conseguirían más colores representables por la pantalla. El DAC de una tarjeta VGA es de 6 bits con lo que se consiguen 64 combinaciones diferentes para cada componente de color. Si recordamos, cada color tiene tres componentes diferentes(RGB), por lo que se puede calcular el numero de colores diferentes representable por una tarjeta VGA:

64 Rojos * 64 Verdes * 64 Azules = 64 ^ 3 = 262.144

Este número es la cantidad de colores diferentes capaz de generar la VGA, pero no se pueden mostrar todos a la vez por la limitación de la paleta. Dependiendo del modo de vídeo podremos disponer de un número mayor o menor de colores; de esta forma de entre los 262.144 colores la paleta nos fuerza a coger un número determinado de ellos. Por ejemplo, un modo de 256 colores indica que la paleta tendrá 256 colores de los posibles 262.144, y modificando las componentes de cualquiera de estos colores podemos tener acceso a otras tonalidades de entre el total generable.

LA PANTALLA Y LOS COLORES

Como hemos visto, el ordenador trabaja con una paleta donde almacena los colores y una pantalla donde se indica qué color de la paleta es el que corresponde en esa posición de la pantalla. Por tanto, en la pantalla se pondrá un byte para indicar qué color de la paleta debe colocarse en este lugar, ya que en un byte se puede almacenar 256 números diferentes y la paleta está compuesta por 256 colores. Así, si en una posición de la pantalla se encuentra el valor 15, se buscará el color número 15 de la paleta. Por tanto se puede decir que los valores que se almacenan en la memoria de vídeo son 'punteros' que apuntan a un valor de la paleta. De esta forma, si se cambia la tonalidad de un color, la imagen no se ve afectada, sólamente que los puntos de la pantalla que tengan ese color cambiarán al nuevo tono. Es decir, si en un byte de la VideoMemoria se tiene el número 10, significa que en ese lugar de la pantalla se debe poner el color número 10. Cuando el DAC retrazando la pantalla lee este valor, lee de la paleta (guardada por la tarjeta) las componentes del color 10 (r=Paleta[10*3], g=Paleta[10*3+1], b=Paleta[10*3+2]) y programa los 3 haces para que dibujen en pantalla esa tonalidad. Cambiando estas 3 componentes de la paleta, cuando el DAC lea otra vez este número, cogerá de este array interno las nuevas componentes y el color resultante en pantalla será el que nosotros deseábamos, sin necesidad de cambiar ningún dato del dibujo original. Por lo tanto la imagen no cambia, sólo cambia la tonalidad de todos los puntos de la pantalla que tengan dicho color.
Esto se puede ver ilustrado en la figura 1.

LISTADO 1

```
void SetRGBColor ( int color, char R, char
G, char B);
{
        asm {
                mov ax, 1015h
                mov bx, [color]
                int 10h
                mov dh, [R]
                mov ch, [G]
                mov cl, [B]
        }
}
```

COMO CAMBIAR LA PALETA

Después de iniciar cualquier modo gráfico de

FIGURA 2. IMAGEN DE UNA ROSA CON SU PALETA CORRESPONDIENTE.

TABLA 1.

Registro DAC	Puerto	Función
PEL Address Read Mode Register	03c7h	Expresa dentro de la tabla interna de 256 colores (PEL) el color al que apunta el puntero interno del DAC cuando leemos la paleta del puerto 0c39h (PEL DATA REGISTER). Cada 3 lecturas del puerto 3c9h este registro se incrementa en 1 (cada 3 componentes leidas se incrementa el puntero para que se refiera al siguiente color).
PEL Address Write Mode Register	03c8h	Apunta dentro de la tabla interna al color que se modificará al realizar 3 escrituras en el PEL DATA REGISTER. Tras estas 3 escrituras en el puerto 03c9h el puntero se incrementa en 1 para hacer referencia al siguiente color dentro de la tabla PEL.
PEL Data Register	03c9h	Mediante este puerto podemos modificar o leer la paleta interna de la tarjeta gráfica. Al escribir un valor en este puerto el DAC lo almacena dentro de la paleta actual en la dirección apuntada por el puntero de escritura, e igual para la lectura. Este puerto constituye para el programador la ventana a través de la cual modificar la paleta de la tarjeta, conjuntamente con los registros PEL Address Read y PEL Address Write, que nos permiten seleccionar el color a leer o escribir. Tras esta selección los 3 valores leidos/escritos en este puerto constituirán las componentes RGB del color.

LISTADO 2

```
Void SetPaletteBIOS (char far *paleta)
{
asm {
        les dx, paleta
        mov ax, 1012h
        mov bx, 0
        mov cx, 256
        int 10h
  }
}
```

256 (o menos) colores, el ordenador nos ofrece una paleta por defecto, que contiene unos colores para uso general. Esto puede ser útil en determinados casos por ser una paleta variada, pero también puede ser que no interese. Supongamos que se quiere hacer el dibujo de una selva. Para este dibujo se necesita una gama de verdes amplia, mientras que probablemente otros colores como rojos y azules no se utilicen, por lo que si se utiliza la paleta por defecto probablemente al dibujo le falten verdes. En cambio, si se cambia la paleta por otra donde el color predominante sea el verde, el dibujo podrá utilizar unos tonos de verdes específicos para las hojas, otros tonos para los matorrales, etc. con lo que se consigue más realismo en la imagen.

Para cambiar un color sólamente hay que cambiar sus tres componentes. Hay varias formas de hacerlo : utilizando las órdenes que el BGI pone a nuestra disposición, por medio de interrupciones o por medio de puertos. Para empezar veremos cómo se realiza por medio del lenguaje C.
Para poder utilizar la librería gráfica hace falta incluir en la cabecera del programa la librería <graphics.h> en la que se incluyen todas las funciones del BGI, pero nos centraremos en la función para cambiar la paleta : 'setrgbpalette()'. Esta función cambia el color por su RGB, por lo que tendrá como parámetros el número de color a cambiar y las tres componentes del nuevo color :

- setrgbpalette (numcolor, rojo, verde, azul);

donde 'numcolor' es el número de la paleta que deseamos cambiar, y 'rojo', 'verde' y 'azul' son las tres componentes del color que queremos obtener.
Si se inicia el modo gráfico con un bloque ensamblador, no se podrán utilizar las funciones BGI porque requieren que se inicie el modo con la función initgtraph(), que inicia unas variables como el ancho o el alto de la pantalla para que el BGI pueda trabajar.

USANDO LA BIOS

Se puede cambiar la paleta con los servicios del MS-DOS, por medio de la interrupción 10h. Esta interrupción gestiona todo lo relacionado con los gráficos del ordenador, pero de momento veremos el cambio de la paleta y en la lectura de un color.
Se puede cambiar un sólo color o un bloque de colores de la paleta. Primero veremos cómo cambiar cualquier color con los servicios de la BIOS.
Hay una interrupción que nos hace este trabajo, y es la subfunción 1010h de la interrupción 10h. Para utilizar esta interrupción se deben pasar los parámetros como se muestra en la tabla 1. En CH, CL y DH se envían las tres componentes RGB (en este orden) que deben estar comprendidas entre 0 y 63. Cuando se hayan asignado los parámatros de entrada se llama a la interrupción 10h, y la BIOS se encarga de cambiar el color.
También se puede cambiar un bloque de colores de una sola vez, que podemos utilizar para cambiar la paleta completa con una sola llamada. El servicio a utilizar es el 1012h de la interrupción 10h, y la llamada a la interrupción también está explicada en la tabla 1.
Veamos un bloque de codigo que cambie toda la paleta entera, suponiendo que la nueva paleta está almacenada en un array de datos llamado 'paleta' y que se este trabajando con un modo de vídeo de 256 colores:

```
asm {
        mov ax, seg paleta
        mov es, ax
        mov dx, offset paleta
        mov ax, 1012h
        mov bx, 0
        mov cx, 256
        int 10h
}
```

En el listado 2 encontramos la función SetPaletteBIOS() que realiza este mismo trabajo. La instrucción assembler 'LES DX, paleta' carga en ES:DX el segmento y el offset

TABLA 2.

Servicio	Parámetros de llamada	Descripción
AX = 1010h	BX = número de color a cambiar DH = rojo CH = verde CL = azul	Set one color register - Este servicio cambia el color indicado por BX. Las componentes RGB se deben indicar en los registros DH, CH y CL respectivamente antes de llamar a la interrupción.
AX = 1012h	ES:DX - Tabla donde se indican las componentes BX = primer color a cambiar CX = número de colores que se van a cambiar.	Set block of colors registers. Cambia un bloque de colores. En la tabla se indican las componentes de cada color de forma sucesiva (R,G,B, R,G, B,...) por lo que la tabla debe tener un tamaño de CX*3. Hay que informar a la interrupción cuál va a ser el primer color a cambiar y cuántos.
AX = 1015h	BX = número de color a leer Devuelve: DH = rojo CH = verde CL = azul	Read one color register - Devuelve las tres componentes del color indicado en BX. Las tres componentes las devuelve en DH, CH y CL.
AX = 1017h	ES:DX - Tabla donde serán guardados los colores. BX = Primer color a leer CX = Número de colores que se van a leer.	Read block of color registers - Como en la función 1012h, se utiliza una tabla para guardar los valores leídos. La dirección de esta tabla se pasa en los registros ES:DX. En BX se guarda el primer color que se debe leer y en CX cuantos colores se leerán.

de un puntero far en C. Hemos de hacerlo así debido a que a una función C no se le puede pasar un array completo, sino que hemos de indicarle (mediante un puntero char far *) la dirección en memoria donde está el array que le pasamos como parámetro. Podemos llamar a esta función con una línea como:

```
SetPaletteBIOS( MiPaleta );
```

siendo MiPaleta un array de 768 elementos conteniendo las componentes de los 256 colores, una tras otra.

LECTURA DE LA PALETA

La BIOS también nos proporciona servicios para la lectura de las componentes de cualquier color. El proceso de lectura de la paleta es sencillo y nos proporcionará las componentes de la paleta que está en la memoria para poder almacenarla en un array, y posteriormente utilizarlo por ejemplo para hacer un fundido.
El servicio de la BIOS a utilizar es el 1015h de la interrupción 10h, que lee un sólo registro de la paleta y nos devuelve su correspondiente RGB. En la tabla 1 se muestran los parámetros necesarios para su llamada, y el listado 1 es un ejemplo del empleo de este servicio.
Este bloque de código en C utiliza la función de leer un sólo color de la paleta. Esta función se puede utilizar para leer toda la paleta entera si la usamos dentro de un bucle; sin embargo, ya hay un servicio que nos devuelve las componentes de un bloque de colores. El servicio es el 1017h, como puede verse en la tabla 1.
Ahora ya se puede modificar cualquier color o bloque de colores para obtener presentaciones más profesionales, pero como se comentó en la entrega anterior, las interrupciones software son demasiado lentas. Para conseguir mayor velocidad se debe acceder más directamente a la tarjeta de video, a través del DAC. El acceso al DAC consiste en el acceso directo a los registros de la tarjeta gráfica "conectados" al controlador del DAC, de manera que pasamos directamente a controlar el hardware de la tarjeta de una forma más profesional.

ACCESO DIRECTO AL DAC

Para cambiar un color de la paleta mediante el acceso al DAC se utilizan unos puertos específicos. En total hay cinco puertos, representados con detalle en la tabla 2.
Los puertos 0x3C7 y 0x3C8 actúan como punteros hacia la paleta, es decir, el valor enviado a estos puertos indica el color dentro de la paleta de 256 colores que debe ser leido o escrito mediante el acceso al puerto 0x3C9.
Por ejemplo, en el caso de que se utilice el puerto 0x3C9 para lectura (se lee un valor de él), nos devolverá las componentes correspondientes al color indicado en 0x3C7.
Al utilizar el puerto 0x3C9 automáticamente se incrementa el puntero (al realizar 3 lecturas o 3 escrituras), por lo que al volver a utilizar el puerto nos devolverá las componentes del siguiente color. Así, se pueden leer o escribir más de un color de forma consecutiva sin utilizar cada vez el puerto 0x3C7 ó el 0x3C8, sólamente llamando al puerto 0x3C9 para cada componente que se quiera leer o escribir.
Suponiendo que se quiera modificar las componentes de un color de la paleta por otras diferentes, los pasos a seguir serian los siguientes:

1.- Enviar al puerto 0x3c8 el numero de color a cambiar (dirección para escritura)
2.-Enviar al puerto 0x3c9 la primera componente del nuevo color (rojo)
3.-Enviar al puerto 0x3c9 la segunda componente del nuevo color (verde)
4.- Enviar al puerto 0x3c9 la tercera componente del nuevo color (azul)

De forma práctica, un programa en C sería:

```
outportb (0x3C8, num_color);
outportb (0x3C9, rojo);
outportb (0x3C9, verde);
outportb (0x3C9, azul);
```

También se puede hacer con un bloque en ensamblador dentro del C para mayor rapidez, usando las instrucciones assembler 'IN' y 'OUT'. La instrucción 'IN AL, DX' lee del puerto DX el valor que contiene y lo almacena en AL. De la misma manera, 'OUT DX, AL' escribe en el puerto DX el valor contenido en el registro AL. Tan sólo hemos de cargar los registros con los valores correctos para evitar las llamadas a funciones C (como outportb() e inportb(), que además de ser más lentas que el bloque assembler, requieren la inclusión del fichero DOS.H).

```
asm {
        mov dx, 0x3C8
        mov al, [numcolor]
        out dx, al
        mov dx, 0x3C9
        mov al, [rojo]
        out dx, al
        mov al, [verde]
        out dx, al
        mov al, [azul]
        out dx, al
        }
```

LECTURA/MODIFICACIÓN DE LA PALETA COMPLETA

Como puede verse en la tabla 2, al realizar en el ejemplo anterior las 3 lecturas, el puntero interno del DAC se incrementa automáticamente, por lo que podemos leer las componentes del siguiente color sin especificarlo en el puerto 3c7h (igual ocurre en la escritura).
Para acelerar el proceso se puede trabajar sobre un array que contenga los valores de la paleta para luego poner toda la paleta entera de una sola vez, utilizando la instrucción de ensamblador 'rep outsb' que es más rápido que poner cada valor de la paleta de forma individual. La instrucción outsb (instrucción utilizable a partir del procesador 80188), escribe en el puerto DX el valor contenido en DS:[SI], e incrementa SI. El prefijo REP repite esta instrucción el número de veces especificado en el registro CX, por lo que podemos enviar la paleta completa (el array de 768 elementos) al DAC con el siguiente pseudocódigo:

```
DS:SI = Array[0]
DX = 3C8h
Puerto[3C8h] = 0
DX = 3C9h
CX = 256*3 = 768
REP OUTSB
```

Este bloque de pseudocódigo selecciona (mediante la escritura en el puerto 03C8h) la modificación del color 0.
A continuación el REP OUTSB manda CX bytes desde DS:SI (nuestro array conteniendo los 768 elementos RGB) al puerto 3C9h. Los primeros 3 bytes del array son las componentes del color 0, los 3 siguientes del color 1, etc. De esta manera, como cada 3 escrituras al puerto 3C9h se incrementa el color que estamos modificando, al enviar 256*3=768 bytes estamos cambiando la paleta completa.
Como ejemplo más concreto del cambio de los 256 colores de la paleta puede examinarse el listado 5, que contiene la función SetPalette(). El proceso de lectura es similar al de escritura, pero usando la función INSB, que lee del puerto especificado en DX y lo almacena en ES:[DI], incrementando DI en 1. De esta manera podemos leer también un bloque de colores a un array apuntado por ES:DI.
Ahora que ya sabemos leer y escribir los colores de la paleta (tanto un sólo color como un bloque completo de ellos), podemos hacer varios efectos con la paleta. El primero que se va a ver es el fundido a negro, en el que se tiene una pantalla y la hacemos desaparecer de la pantalla de forma suave.