Programación en Ambiente Windows Utilizando las Bibliotecas ObjectWindows de Borland

José Anibal Arias Aguilar
Alumno de la Maestría del CINTEC-IPN.
Osvaldo Espinosa Sosa
Alumno de la Maestría del CINTEC-IPN.

Es evidente que el uso del ambiente Windows se ha convertido en un estándar para el usuario de computadoras personales. Es por esto que se hace cada vez más necesario aprender a programar aplicaciones que funcionen bajo este ambiente. Existen dos tipos de programación para Windows: programación estándar y programación en base a objetos. Cuando se programa en base a objetos, la mejor alternativa consiste en utilizar las bibliotecas OWL ("ObjectWindows Library") que proporciona el compilador de Borland. En el presente artículo se pretende mostrar el mecanismo básico de una aplicación de este tipo implementando el popular juego Tetris ™.

La Programación Orientada a Objetos (POO)

La POO es una nueva forma de enfocar el trabajo de la programación. Toma las mejores ideas de la programación estructurada y las combina con nuevos conceptos que alientan una visión diferente de la tarea de la programación. La POO permite descomponer un problema en subgrupos de partes relacionadas. Así, se pueden traducir estos subgrupos en unidades autocontenidas llamadas objetos.

C++ es una versión expandida de C. Mantiene su eficiencia, flexibilidad y filosofía añadiendo al mismo tiempo soporte para la POO: objetos, polimorfismo y herencia.

Objetos. *Una clase es una entidad lógica que contiene datos y un código que manipula esos datos. Dentro de una clase, parte de ese código o datos pueden ser exclusivos de la clase e inaccesibles fuera de ella; a esto se le denomina encapsulación de datos. Los objetos se obtienen a partir de la definición de una variable del tipo clase, y esto se conoce como instanciación de clases.*

Polimorfismo. *El polimorfismo describe la capacidad del código para comportarse de diferentes maneras dependiendo de los tipos de datos que se estén tratando.*

Herencia. *La herencia es el proceso por el cual una clase puede adquirir las propiedades de otra. Esto es importante porque soporta el concepto de clasificación. Si se considera, la mayoría del conocimiento se hace manejable por medio de clasificaciones jerárquicas.*

Windows

Para el usuario corriente, Microsoft Windows es una extensión gráfica del sistema operativo MS-DOS. Windows extiende al DOS de diversas maneras: puede soportar múltiples programas concurrentes, maneja gráficas de alto nivel y proporciona una interface estándar de comunicación con los programas. Los programas que corren bajo esta plataforma son manejados por eventos, es decir, la estructura y operación de estos programas se centra alrededor de eventos generados por el usuario (como "clicks" del ratón o presión de teclas). Tradicionalmente, los programas dictan la secuencia que el usuario debe seguir para lograr los propósitos del programa, sin embargo en Windows (y en otras interfaces gráficas de usuario, como Apple Macintosh u OS/2 Presentation Manager) los programas le permiten al usuario decidir los pasos requeridos para completar una tarea, ver **figura 1**.

Windows es un sistema operativo multitarea en el que los programas no son interrumpidos para ejecutar otros, sino que ellos mismos se interrumpen para permitir que otros programas se puedan ejecutar. El «distribuidor de tareas» de Windows se integra por un sistema de transmisión de mensajes y es gracias a los mensajes que los programas pueden recibir información proveniente de los usuarios y de las funciones de la interface de usuario. Para el programador, un mensaje es la notificación de la ocurrencia de un evento que puede necesitar de alguna acción específica.

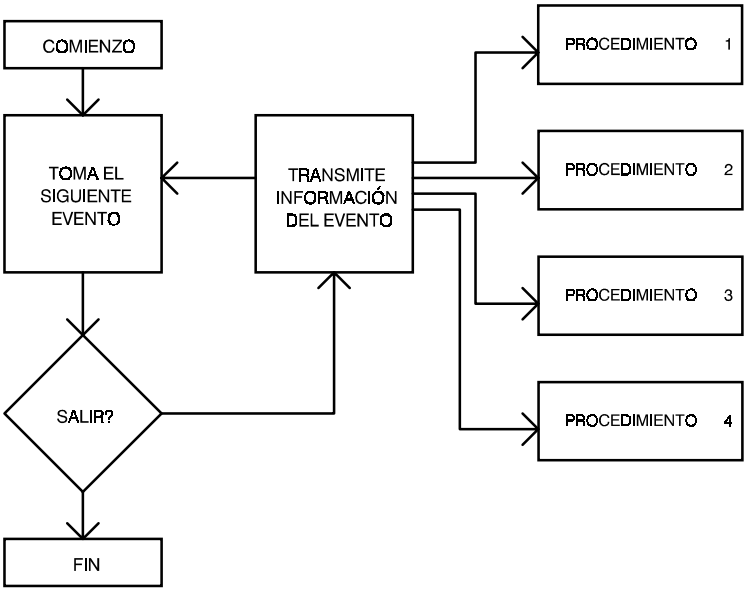

Figura 1. Programa manejado por eventos.

Manejo de los Recursos de Windows

Algunos de los recursos de Windows que pueden utilizar sus aplicaciones son:

- Contextos de dispositivo
- Pinceles
- Plumas
- Fuentes o tipos
- Cajas de diálogo
- Iconos

Windows maneja una interface muy compleja. La interacción entre las aplicaciones del usuario y Windows se efectúa mediante un grupo de más de 500 llamadas a función que en conjunto se denomina API ("Application Program Interface", interface de programas de aplicación). Al plantear el uso del lenguaje C, se producen programas Windows ilegibles, de difícil mantenimiento y depuración. Hay otras áreas que ofrecen dificultades especiales a los diseñadores de aplicaciones, como son el uso de la memoria, las bibliotecas de enlace dinámico (DLL), programas de interface con múltiples documentos (MDI) así como fuentes e impresoras.

Borland vino en ayuda del programador Windows C++ mediante un nuevo marco de aplicaciones denominado Biblioteca ObjectWindows (OWL). OWL sería a una aplicación Windows, lo que TurboVision es a una aplicación DOS. Con ayuda de la OWL se facilita considerablemente el proceso de escribir los programas Windows y se evita que aparezcan diversos detalles de bajo nivel en el código de aplicaciones.

Anatomía de una Aplicación OWL

Es necesario que todo programa escrito con ayuda de OWL soporte al menos dos clases de objetos: un objeto de aplicación y un objeto de ventana. La aplicación es un marco de trabajo complejo encargado de administrar todos los objetos del programa, entre otros: menús, ventanas descendientes y cajas de diálogo. La parte medular de toda aplicación OWL es su programa principal. El siguiente listado es el programa principal de la aplicación descrita en este artículo.

```
# include <owl.h>

int OwlMain(int,char **)
{
  TetrisApp  App;
  return  App.Run();
}
```

TetrisApp la principal clase de aplicación que tiene a su cargo el control de todas las ventanas, las cajas de diálogo y otros objetos de la aplicación. Todo programa OWL debe construirse con base en una clase de aplicación derivada de la clase *TApplication*, que es utilizada por todas las aplicaciones Windows para inicializar, crear la ventana principal y pedir al sistema que le envíe mensajes. El programa de aplicación interactúa con el proceso de inicialización a través de la función

TetrisApp::InitMainWindow()

que a su vez crea la instancia de una ventana de usuario asignada dinámicamente.

Con la ayuda de *TWindow*, el programa soporta las características básicas de ventana: registro, desplazamiento y modificaciones de tamaño. Los «procedimientos de ventana» son funciones contenidas en los objetos de la ventana de base TetrisWindow. Tales procedimientos son utilizados como respuesta a los mensajes de Windows.

El manejo de las cajas de diálogo es un procedimiento que consta de dos pasos: primero se usa el programa *Resource Workshop* para crear la ventana de diálogo y todos sus controles; después se escribe una clase derivada de *TDialog* para manejar los comandos de diálogo y los controles.

Descripción de la forma en que la Aplicación Opera Bajo Windows

Como ya se mencionó anteriormente, la apariencia de la ventana está determinada por las funciones miembro de la clase derivada de la clase *Tapplication*, y el manejo e interpretación de los recursos y el procesamiento de las funciones que entregan el resultado de la aplicación en sí están determinadas por las funciones miembro de la clase derivada de la clase *Twindow* (o cualquiera de la opciones disponibles).

En esta sección del artículo se describirá la forma de utilizar a la clase derivada de *TWindow* para implementar un juego, el ya conocido Tetris. Comenzaremos por mencionar que las funciones propias del juego se incluyen como funciones publicas de *TetrisWindow*, clase derivada de *TWindow*; esta clase contiene además funciones para el procesamiento de eventos como lo son el temporizador de Windows, el presionar una tecla, el utilizar opciones de los menús, etc. y que a su vez está relacionada con una tabla de definición de respuestas que ligará los comandos con las definiciones de función del archivo de recursos.

Cabe mencionar que este tipo de aplicación debe hacer uso del temporizador de windows, por lo que está declarado su uso tanto en la clase *TetrisWindow* como en la tabla de respuestas; la razón de hacerlo obedece a que el juego debe tener la misma velocidad aparente al ejecutar el programa en computadoras con diferente frecuencia de operación del microprocesador. Cuando se hace esto en ambiente DOS, simplemente se usan funciones de retardo en un bucle de procesamiento. En windows esto no es posible porque al generarse un bucle se interrumpe la búsqueda de los mensajes propios del ambiente Windows, y si el bucle es infinito, se pierde totalmente el control del sistema hasta que se destruye el bucle y se finaliza la función que se estaba procesado. Es por lo anterior que el proceso de mover las piezas en forma descendente se hace a través de una función que se llama constantemente a intervalos definidos de tiempo. Para lograr el propósito, se define una función *EvTimer* que es la encargada de controlar la velocidad de procesamiento del juego y que tiene acceso directo al temporizador.

También se ha definido en las clases una función *EvKeydown* que detecta el mensaje generado por la opresión de una tecla y que permitirá detectar la función que el usuario desea que se realice en el momento, a saber permitirá rotar, desplazar a la izquierda o derecha una figura o bien suspender momentáneamente el juego.

El juego Tetris se basa en ordenar figuras que van descendiendo en un área determinada y completar líneas para desaparecer y permitir seguir jugando, en caso de llenar el área al tope el juego finaliza. Es por esto que en la clase *TetrisWindow* se incluyen funciones para las siguientes actividades: dibujar la figura actual, dibujar el ambiente de juego, establecer la puntuación, desplazar la figura actual, comprobar el momento en que una pieza ya no pude descender, posicionamiento de la pieza, y detección de línea completa principalmente. El juego basa su operación en simular en pantalla lo que ocurre en una matriz m x n, en donde una localidad vacía contiene un cero y una llena contiene un valor de uno, lo que facilita el hecho de revisión del estado del juego, y permite realizar el proceso de detección de límites y topes para los desplazamientos de las figuras.

El proceso de dibujar una nueva figura consiste en copiar una imagen que se ha representado en forma de una matriz, y que se selecciona aleatoriamente en base a la función rand() del lenguaje C. El número de figuras es fácilmente ampliado con mínimas modificaciones al listado del programa.

Para simular el movimiento de una figura en el ambiente gráfico utilizamos las funciones getimage y putimage. En ambiente Windows se utiliza la función *BitBlt* para lograr lo mismo que las dos anteriores; para mostrar su uso en el programa, después del uso de *BitBlt* se pone como comentario la forma en que se utilizaría getimage y putimage ilustrando la similitud. El uso de esta función solo requiere entender lo que se denomina un «contexto» en la programación para ambiente Windows, podemos decir simplemente que es el medio en el cual y para el cual se realiza un proceso, en este caso tenemos los dos principales que son la memoria de trabajo y el despliegue.

Por último, mencionaremos que la compilación del programa se realiza tomando en cuenta al menos tres archivos: el archivo fuente con extensión CPP, el archivo de recursos con extensión RC y un archivo de definiciones con extensión DEF, los cuales se engloban en el archivo de proyecto con extensión IDE

La **figura 2** muestra el aspecto visual del programa.

Conclusiones

Uno de los aspectos más importantes que debe tener en cuenta el programador al utilizar programación orientada a objetos usando OWL, es que se requieren mayores recursos de cómputo para compilar los programas que con la programación tradicional, esto debido a que se busca

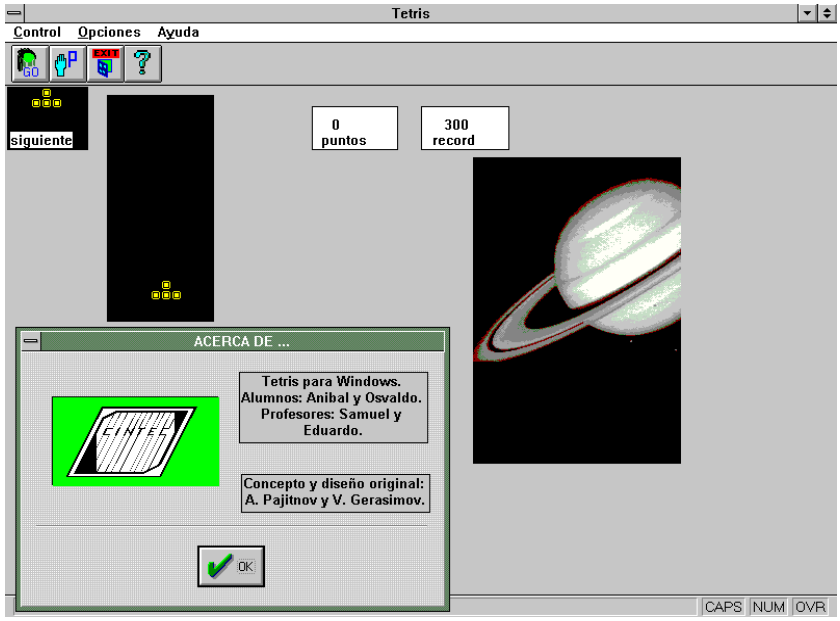

Figura 2. Aspecto visual del programa

información en muchas bibliotecas, y el trabajo de cómputo es intenso, aunque al programador le resulte más fácil escribir programas bajo este esquema, ya que con pocas líneas de código fuente es posible crear ventanas vistosas.

Un hecho indiscutible es el siguiente: el programa ejecutable producido por compilar usando OWL es bastante más grande en tamaño que el que provee la programación tradicional; es fácil deducir que se incluyen componentes muchas veces redundantes de código no optimizado. Es responsabilidad del programador elegir la ruta más conveniente para crear sus aplicaciones teniendo como base el panorama anterior.

Finalmente, el código correspondiente de este ejemplo se encuentra disponible en el CINTEC, para cualquier persona que lo requiera.

Bibliografía

[1] Nabajyoti Barkakati. *"C Programmer's guide with Borland C++ 4.0"*. Editorial SAMS

[2] Herbert Schildt. *"Turbo C ++: Manual del usuario"*. Editorial McGraw Hill.

[3] Borland. *"Object Windows Library Reference"*. Editorial Borland.