Ayuda GPong

Volver

Descripción funcional

GPong es otra versión más del clásico juego pong, la idea original fue de Ralph Baer en 1951 pero no pudo implementarla porque su jefe rechazó la idea. Hubo que esperar a 1958, año en el que Willy Higginbotham hizo la primera implementación, llamada "Tenis para dos".

No hay mucho más que decir del juego, simplemente tenemos dos jugadores intentando devolver la pelota, gana el que haga más puntos al otro. El punto se hace cuando no se es capaz de devolverla.

Las características principales de esta versión son:

- Música y sonido.
- Personalización de teclas.
- Animaciones y transparencias.
- Interfaz amigable: barra de herramientas con consejos (tooltips) y dibujos descriptivos.
- Posibilidad de jugar contra la máquina.
- Pausa en cada momento del juego.
- Multiidioma, actualmente solo inglés y español, se agradecen otras traduciones...
- Introduce una variación frente al juego original, y es que los palos se pueden mover tambien en la coordenada x, no solo en la y, esto da mas maniobrabilidad.
- Varios niveles de dificultad de juego.
- Posibilidad de jugar cero, uno o dos jugadores.

Volver

Volver

Manual de usuario

Interfaz.

Como se puede ver en la captura, los elementos básicos son:

- **Barra de menú:** Zona desde donde se manejan todas las opciones, se puede acceder a los elementos con el teclado pulsando alt y sin soltar la tecla pulsando el la letra subrayada, en algunas versiones de windows no

aparecerán subrayadas hasta que no se puse la tecla alt. Los elementos son:
    - Fichero: Contiene elementos comunes, se escogió este nombre para relacionarlo con otros programas y hacerlo más intuitivo.
        - Nuevo: Empieza una nueva partida, se puede acceder directamente con la tecla F2.
        - Salir: Termina el programa y la partida. Tecla rápida: Alt + F4 (nota: el signo '+' significa que se pulsa la primera tecla y sin soltarla la segunda).
    - Ver: Elementos que afectan la jugabilidad, en algunas versiones algunos elementos pueden estar inhabilitados.
        - Barra de herramientas: Permite mostrar u ocultar la barra de herramientas.
        - Barra de estado: Permite mostrar u ocultar la barra de estado.
        - Pantalla completa: Cambia entre el modo de pantalla normal o completa. Tecla de acceso rápido: F11.
    - Juego: Opciones de la partida.
        - Preferencias (F5): Permite configurar varios parámetros del juego.
        - Personalizar (F7): Permite configurar los controles del juego.
        - Pausa (P): Para el juego en cualquier momento.
    - Ayuda:
        - Índice: Presenta esta ayuda.
        - Acerca de: Muestra un pequeño diálogo con los créditos.
- **Barra de herramientas:** Permite acceder a las partes más usadas de la barra de menú. Contiene:
    - Nuevo: Empieza un nuevo juego.
    - Preferencias: Accede al diálogo de preferencias.
    - Controles: Accede al diálogo de configuración de controles.
    - Pausa: Pausa el juego.
    - Ayuda: Accede a la ayuda en linea.
    - Acerca de: Créditos.
- **Ventana de juego:** Es la ventata donde se desarrolla el juego. Más información en la zona de juego.

Preferencias.

Este es el diálogo principal con las preferencias del juego.
Se divide en cuatro categorías:

- Velocidad: Indica la velocidad de la pelota.
- Multimedia: Características de sonido del juego:
  - Sonido: Activa o desactiva el sonido.
  - Música: Activa o desactiva la música, puede dar problemas con algunas tarjetas de sonido.
- Nivel: Capacidad de reacción de la máquina.
- Número de jugadores:
  - Ninguno: Modo demostración, máquina vs. máquina.
  - Un Jugador: Máquina vs. Humano.
  - Dos jugadores: Para jugar con otro jugador humano.

Los botones son standar en windows:

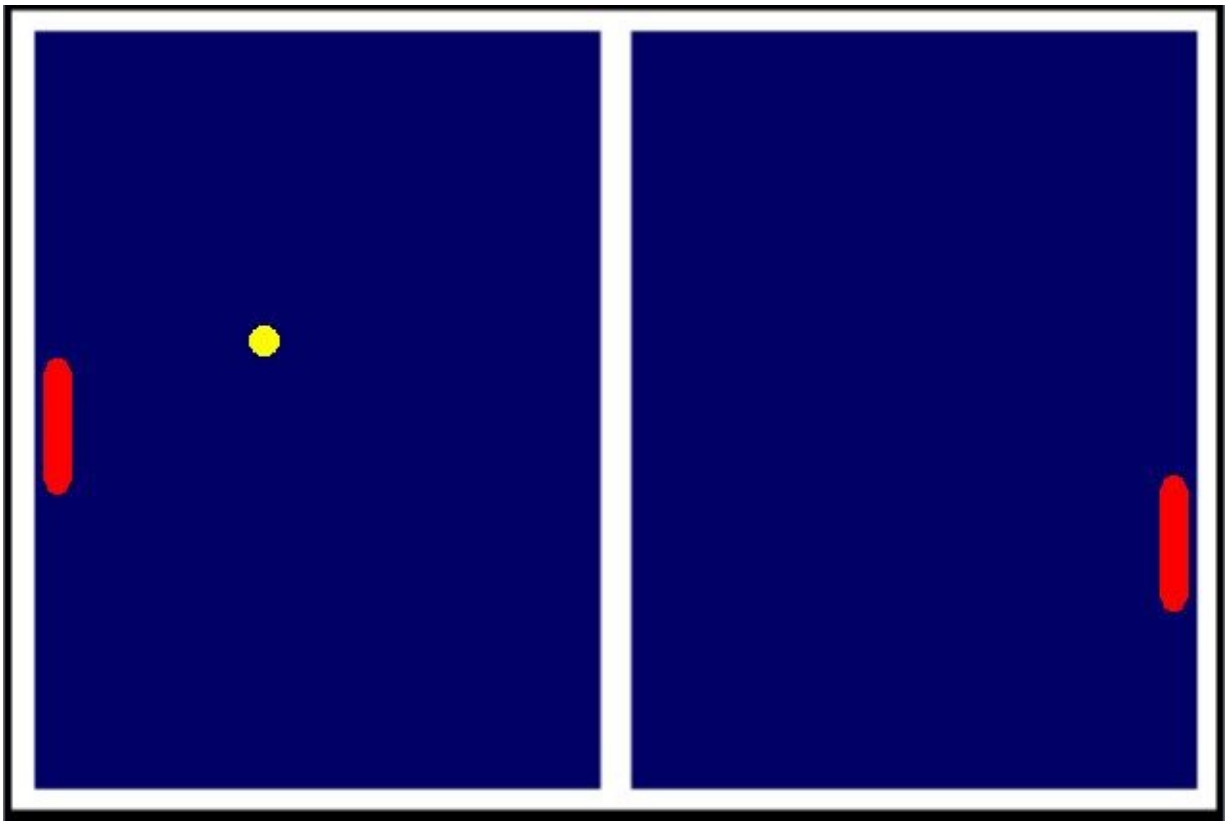
- Aceptar: Guarda la selección actual y la aplica.
- Cancerlar: Descarta los cambios.
- Aplicar: Guarda la configuración actual pero no cierra el diálogo.

Las configuraciones se guardan en un archivo ini en disco para que cada vez que carguemos el juego empiece con la configuración anterior.

Personalizar.

Permite configurar los controles del juego. El funcionamiento es sencillo, basta con seleccionar la tecla para cada movimiento y para cada jugador del rango propuesto.
El botón "Originales" restaura las teclas por defecto del juego, estas son las que están en la foto.

Juego.



Esta es la ventana principal del juego, y donde se desarrolla toda la partida, se puede ver como el jugador de la izquierda se propone golpear la pelota sin que le traspase por el fondo. El resto es habilidad.

Requerimientos:

- **Software:**
  - Cualquier versión de windows en procesadores compatibles intel, que corra en modo protegido de 32 bits, es decir: windows 95 y posteriores, por ejemplo: windows 98, 98se, me, nt3.50, nt4, 2000, XP.
- **Hardware:**
  - Cualquier procesador capaz de ejecutar los requerimientos de software debería valer, el juego no usa direct3d ni opengl asi que no es necesario un adaptador con aceleración 3d.

Volver

olver

Manual del programador

Introducción:

El lenguaje de programación del juego es c, y se ha usado la api básica de windows, no hay mfc en el código.

El juego se divide en dos archivos de código fuente, game.c y gpong.c. La funcion principal reside en gpong.c; es en este archivo donde se incializan las estructuras y donde se crea y maneja el interfaz y se crea la ventana hija donde se desarrolará el juego. Dicha ventana hija reside en game.c.

Para facilitar la labor de creación de diálogos, aceleradores de teclado y para evitar que los usuarios modifiquen algunos archivos, se ha usado un archivo de recursos. Este archivo tiene la peculiaridad de que provee recursos en tres idiomas: Español, Inglés y Neutral. Los dos primeros se proporcionan para recursos dependientes del idioma, como cadenas de texto, menú, diálogos y algunos bitmaps. El idioma neutral, se proporciona para recursos independientes del lenguaje: sonidos, iconos aceleradores y otros bitmaps.

Internamente en todo el código se ha tendido a mantener los nombres de variables, de estructuras y de comentarios en inglés, puesto que es el idioma en el que se escriben las estructuras de control de flujo y los nombres de funciones de la biblioteca de windows y de c.

Estructuras

Las estructuras globales del juego son estas:

Preferencias:

La estructura básica para la configuración del programa es esta:

```
typedef struct _PREFERENCES {
        char speed;
        char level;
        union {
                char back_color[8];
                char back_bitmap[FILELEN];
        };
        union {
                char plyr_color[8];
                char plyr_bitmap[FILELEN];
        };
        union {
                char ball_color[8];
                char ball_bitmap[FILELEN];
        };
        BOOL sound;
        BOOL music;
        int nplayers;
} PREFERENCES;
```

Define todas las preferencias configurables desde el diálogo de preferencias, la configuración de teclas se ha cambiado a esta otra estructura:

```
typedef struct _KEYS {
        int left;
        int right;
        int up;
        int down;
} KEYS;
```

simplemente para destacar que se configura en otro diálogo, y para permitir en un futuro ampliar el número de jugadores facilmente, ya que cada uno tiene sus

teclas propias.

Juego:

Cada jugador tiene una posición en el campo, esta se maneja con esta estructura:

```
typedef struct _PLAYER {
        int left;
        int right;
        int top;
        int bottom;
        BOOL t_left;
        BOOL t_right;
        BOOL t_top;
        BOOL t_bottom;
} PLAYER;
```

Los campos "booleanos" indican para que lado fue el último movimiento. La utilidad de esto es imprimir o quitar velocidad a la bola si estamos golpeandola hacia adelante (más fuerza) o hacia atrás (menos fuerza).

La pelota tambien tiene su posición, en esta estructura tambien se guardan otras propiedades de la pelota, como por ejemplo ancho, alto y los dx y dy que indican el incremento de la pelota (estos "diferenciales" pueden ser positivos para un movimiento positivo en la coordenada x o negativos para un movimiento negativo):

```
typedef struct _BALLINFO {
        int width;
        int height;
        int x;
        int y;
        int dx;
        int dy;
} BALLINFO;
```

Esta estructura guarda la puntuación del juego en cada momento y para cada jugador:

```
typedef struct _SCORE {
        int player1;
        int player2;
} SCORE;
```

Interfaz

La función principal crea una ventana e incia el bucle de mensajes:

```
while (GetMessage (&msg, NULL, 0, 0))
        if (!TranslateAccelerator (msg.hwnd, hAccelTable, &msg)) {
                TranslateMessage (&msg);
                DispatchMessage (&msg);
        }
```

El bucle de mensajes se mantiene en la función DlgProc, la cual al crearse inicializa todos sus elementos:
Menú, barra de herramientas, etc. Procesa los mensajes de pulsaciónes de teclas de sus ventanas (WM_COMMAND), y los trata según su significado, modificando variables o abriendo diálogos; cabe destacar esto:

```
case WM_KEYDOWN:
        SendMessage (hGame, WM_KEYDOWN, wParam, lParam);
        break;
```

Lo que hace es enviar el mensaje de pulsación de teclado no recogido por sus ventanas hijas (barra de herramientas) y que por tanto no ha sido enviado en forma de wm_command, al procedimiento de ventana de la ventana del juego.

El resto de este procedimiento es trivial, a destacar: WM_NOTIFY que se usa para los "tooltips".

Juego

El juego se desarrolla en una ventana hija de la principal, el motivo para tal separación es porque con este sistema se puede tratar al juego como una entidad separada, por ejemplo para esconder las barras de herramientas o pasar a ventana completa.

El procedimiento principal de esta parte es GameProc(), el cual carga todos los archivos necesarios: sonidos, bitmaps de fondo, jugadores y pelota, pone el marcador a cero, y carga el timer principal del juego.

El timer es la parte más conflictiva. como función de callback se especifica NULL asi que simplemente se recibe un mensaje wm_timer para ese mismo procedimiento de ventana. En este mensaje se procesa la nueva posición de la pelota, de los jugadores y se pinta todo. La manera de hacer esto se explica más abajo.

El mensaje wm_paint de este diálogo es algo atípico:

```
if (is_paused || just_scored) {
        hdc = GetDC (hWnd);
        Draw (hdc, cClient);
        ReleaseDC (hWnd, hdc);
} else
        return DefWindowProc (hWnd, message, wParam, lParam);
```

Esto redibuja la pantalla cuando está parado el juego o se acaba de marcar (y por tanto la animación está parando el juego), en caso contrario se llama a DefWindowProc para que pinte las modificaciones hechas en el mensaje wm_timer.

En el mensaje WM_COMMAND de este procedimiento, se maneja la pausa simplemente quitando el timer cuando se para el juego y activandolo de nuevo cuando de quita la pausa, redibuja tambien la pantalla para que se pinte el cartel de pausa.

El mensaje WM_USER, queda reservado para que el procedimiento padre le avise de que tiene que releer las configuraciones.

Dibujo

El dibujo es la parte más optimizada. Para evitar que de barridos de pantalla, se ha usado un doble buffer (función Draw()) que vuelca todo sobre memoria y cuando lo tiene sobre pantalla.

Esta función se ayuda de CreateBitmapMask() para crear la máscara a partir del color especificado como argumento, que despues será usada en bitblt() haciendo un SRCAND para producir las transparencias.

Puede que alguna de estas funciones pierda velocidad pero de esta forma se evita usar las primivitas de las DirectX (direct draw) para este trabajo y por tanto la necesidad de una tarjeta aceleradora.

En UpdateBall() es donde se maneja el movimiento de la pelota, controla cuando la pelota choca (frente a pared o frente a los palos), y controla cuando se ha marcado gol. En este último caso, se para el juego, se lanza la animación de gol y se pone un timer para volver a quitar la animación (en realidad la animación se dibuja en Draw() y esta funcion y la de callback del timer se encargan de cambiar una variable: just_scored a verdadero o falso).

Teclado

El teclado se controla síncronamente (polling), esto ayuda a que no haya problemas de colisión al ser la pelota síncrona y los palos asíncronos, aún perdiendo velocidad (no necesaria en este juego). Otro motivo más sutil, es que haciendolo asincrono, windows nos oculta la primera pulsación hasta el tiempo especificado en la configuración de teclado de windows, lo cual mata la interactividad del juego. La manera de evitar esto no es tan obvia, y no se otra que no sea usando las DirectX (direct input en este caso).

En sincronismo del teclado se llama a la función GetKeyboardState() del api de windows, que nos devuelve el estado de cada tecla en una matriz de 256 teclas. Despues haciendo un and con 0x80 se comprueba si está pulsada en ese momento: if (keys[player1kbd.left] & 0x80... El movimiento de las raquetas se limita a su campo, y aunque esté pulsada no se actua si nos encontramos en el límite de nuestro campo. La linea entera sería parecida a esta:

```
if (keys[playerXkbd.posicion] & 0x80 && jugadorX_en_limite_de_campo)
```

```
        mover_la_posicion_de_la_raqueta_que_draw_la_pintara_cuando_s
}
```

En este punto, puede retornar esta función dependiendo del número de jugadores, y por tanto comprobar las teclas para ninguno, para uno o para los dos jugadores.

I.A.

La "inteligencia artificial" del juego se encuentra en la función computer(), que es llamada cuando hay menos de dos jugadores.
Esta función es llamada para cada jugador, y se encarga a partir de unas condiciones:
si (la_pelota_está_en_nuestro_campo y la_pelota_no_está_a_nuestra_altura) .... de mover la posición de la raqueta de la máquina hacia arriba o hacia abajo.

Otras

La función playmidi() carga un midi desde disco a memoria para ser reproducido despues (por ahora la propia función lo pone en marcha), devuelve un handler a dicho midi. Por motivos desconocidos la llamada a MCIWndCreate() se retrasa en devolver en función del tamaño del midi.

Recursos

Se han incluido recursos de todo tipo: aceleradores de teclado, bitmaps, etc..
El tamaño de los bitmaps, generalmente está especificado en el archivo common.h de cabecera. El motivo para hacerlo aquí en vez de conseguir las dimensiones del bitmap en tiempo de carga, es que son valores poco propensos a cambiar y evitamos entrar en trabajo con bitmaps que el usuario puede cambiar (no es probable que un usuario cambie los bitmaps dentro del ejecutable desensamblando el binario, pero este método se usa tambien para archivos fuera de los recursos y se ha mantenido uniforme) y desbordarnos.

Los recursos dependientes del idioma, por ejemplo el bitmap de pausa, cuentan con una version para cada idioma soportado. Esto permite una aplicación adaptada al idioma del usuario en tiempo de carga (sin recompilar), como se ha expuesto más arriba.

Todos los recursos han sido creados con las herramientas del visual studio, por tanto puede ser que no sean compatibles hacia atrás con otras versiones anteriores del compilador de recursos.

Hay recursos de más tipos (vease el archivo de recursos), destacamos el recurso personalizado "WAVE" para el audio de choque y de celebración y la tabla de caracteres que contiene mensajes de error, los tooltips y otros textos en varios idiomas.

Macros

Se ha intentado en todo momento posible usar macros en lugar de funciones puesto que son más rápidas y evitan duplicar código, hay que tener cuidado especial en estos casos, especialmente cuando modifican variables. En algunos casos es trivial (player?kbd son variables globales):

```
#define DEFAULT_KEYS() {                \
        player2kbd.left = VK_LEFT;      \
        player2kbd.right = VK_RIGHT;    \
        player2kbd.up = VK_UP;          \
        player2kbd.down = VK_DOWN;      \
        player1kbd.left = 'A';          \
        player1kbd.right = 'D';         \
        player1kbd.up = 'W';            \
        player1kbd.down = 'S';          \
}
```

Pero en otros casos no es tan aparente (en este caso estamos cargando las cadenas para los tooltips, proceso que se realiza para cada botón de la barra de herramientas):

```
#define BUTTONTEXT(res, val) \
        LoadString (hInst, res, btntext, MAX_RC_STRING_LEN - 1); \
        btntext [lstrlen (btntext) + 1] = '\0'; /* Double-null termi
        val = (int) SendMessage (hWndToolbar, TB_ADDSTRING, 0, (LPAR
```

Esta sección ha sido puesta para avisar al futuro programador de efectos secundarios con el uso de estas estructuras, tan potentes y delicadas.

Preferencias

El modelo de preferencias (lectura, carga y escritura) tiene su propia sección ya que no es trivial y se ha hecho usando el c standar en lugar del api de windows para E/S de archivos.

La función que lee las preferencias es read_cfg() que primero carga unas preferencias por defecto que irá sobreescribiendo con lo que lea del archivo ini. Despues abre el archivo y para cada linea que no esté vacia ni empiece por '#' busca el valor hasta el caracter '=', lo compara frente a distintas cadenas para comprobar si es una variable válida, y si lo es se traduce a una variable dentro de la estructura de preferencias o de teclado y se almacena.

Al iniciar el diálogo de preferencias, se le mandan los mensajes para selecionar los "botones de radio" adecuados en función de lo que se ha leido anteriormente. Cada mensaje de pulsación generado por los controles hijo de este diálogo hace que se mande su correspondiente mensaje de selección al control y de deselección a los de su grupo.

En caso de que el usuario acepte o aplique se graba la configuración y se restaura el estado anterior de pausa.

La escritura del archivo se hace en write_cfg() con las clásicas fopen() y fprintf() por medio de macros. El resultado es una lista de pares variable=valor,
este sencillo sistema tiene el efecto secundario de que el usuario no puede editar a mano el archivo ini puesto que se borrarian sus comentarios en la escritura del archivo, pero tampoco lo necesita puesto que para estas modificaciones se proporcionan los diálogos del juego.

Configuración del teclado

Esta parte puede resultar algo más complicada y requiere su sección aparte.
Al abrir el diálogo de configuración de teclado, (WM_INITDIALOG) se manda un mensaje para cada posible tecla y para cada control, de esta forma

```
for (i = 0; i < 8; i++) {
        control = GetDlgItem (hDlg, controls[i]);
        for (j = NUM - 1; j >= 0; j--)
                SendMessage (control, CB_INSERTSTRING, 0L, (LPARAM)
}
```

Esto rellena los controles de selección. Despues manda otro mensaje a cada control para que seleccione la tecla que se haye escogida por el usuario en preferencias, y copia en variables temporales la selección de teclas para trabajar hasta que se pulse aceptar o cancelar.
La macro que hace este trabajo es esta:

```
#define SELECT_CB_TEXT(_control, _key)          \
        SendMessage (GetDlgItem (hDlg, _control), CB_SELECTSTRING, 0
                (LPARAM) strings[getkey (_key)])
```

La función getkey merece ser explicada aparte. Lo que hace es sencillo, puesto que solo traduce de ascii a offset de la tabla de caracteres permitidos (ver mas abajo). Pero como lo hace puede parecer confuso.
Algunas teclas tienen tradución directa: case 'caracter': return codigo_ascii; pero las letras, eran demasiadas para este sistema asi que se dejaron en el "default" del "switch" una serie de condiciones que restan a la tecla en cuestión una constante relativa dentro de la tabla ascii. Esto funciona porque en la tabla de caracteres las letras están ordenadas. Si la letra fuera misuscula se resta respecto a la 'a' y si fuera mayúscula respecto a la 'A', lo cual hace que el "offset" de la matriz de tradución de teclas sea igual para unas que para otras (creo que esta parte está mas clara en el código que aquí, la función es: customize() dentro de gpong.c).

Desde WM_COMMAND se hace el trabajo con los controles del diálogo, en el caso de que se acepte se cargan en las variables de teclado globales las teclas selecionadas por el usuario y se escribe el archivo .ini, despues se cancela (lo

cual solo cierra el diálogo), por eso no hay break en esa selección del switch, tambien se restaura el estado de pausa anterior a la carda del diálogo.
El control de botón "originales" solo manda un mensaje a cada control para que seleccione las teclas codificadas en el código, no se selecciona esta configuración hasta que no se acepta el diálogo.

La tabla de caracteres permitidos referenciada arriba es esta::

```
static char strings_r[] = { 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H',
        'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W',
        'Y', 'Z', '0', '1', '2', '3', '4', '5', '6', '7', '8', '9',
        '.', '-', VK_LEFT, VK_RIGHT, VK_UP, VK_DOWN };
```

y la selección de una nueva tecla se hace así:

```
if(HIWORD(wParam) == CBN_SELCHANGE)
        player1tmp.left = (int) strings_r[(char) SendDlgItemMessage
                IDC_PLY1_LEFT, CB_GETCURSEL, 0L, 0L)];
```

Es decir: si hubo un cambio de selección en el control, entonces la nueva tecla (temporal) para el jugador 1 movimiento izquierda (en este caso), es el resultado de interrogar al control del jugador 1 izquierda por la selección actual traducido mediante la tabla de caracteres permitidos.

Ayuda

Se proporciona ayuda lista para ser compilada e integrada en el nuevo sistema de ayuda de microsoft(tm), tipo msdn. Pero como es poco común ver este tipo de ayudas, no se utiliza. Asi que se proporcionan ficheros de proyecto para el "HTML Workshow de microsoft", que permiten generar un archivo chm a partir de páginas web.

Este program se puede descargar gratuitamente de microsoft(tm).

Requisitos

Supongo que el código puede compilarse en cualquier versión de visual studio, pero los ficheros de proyecto requieren la versión .Net o posterior.
Para pasarlo a VS6 bastaria con crear un fichero de proyecto, añadir los fuentes y el fichero de recursos y el bmp usado de fondo.
El motivo de mantener estos archivos a parte, es para que el usuario pueda cambiarlos y personalizar el juego.
Tambien se ha intentado mantener la compatibilidad con el compilador gratuito cygwin http://cygwin.com . Pero esta se perdió en las primeras versiones de la aplicación, concretamente al usar controles comunes de windows, una de las preferencias es hacer que compile de nuevo con este compilador.

Volver