74. Horizontal Text Scroller Demo with Music Player using GDI and FMOD

The Icebreaker, Fri 31 Aug 2007

Horizontal Text Scroller Demo with Music Player using GDI and FMOD

Written By The Icebreaker

Quote: "Programming is like sex: One mistake and you have to support for a lifetime."

Introduction

This is my first tutorial written for Adok/Hugi's Diskmag, I have decided to write it after I came across on a request for a Text Scroller using GDI and a request for a Win32 "XM" Music player. So here we are. I hope that you will enjoy the tutorial, and you will be able to code your own text scroller after you read it, or even better a "vertical text scroller" or a scroller with images. I must state that while this tutorial is for complete newbies, a basic knowledge of C is needed, because with the "copy-paste" stuff, you won't learn anything. The MingwC and the Dev-Cpp IDE from bloodshed.net is used to compile this crap, but I think that you shouldn't encounter problems with other compilers. Actually, I am very attached to this IDE and Compiler because you know: "The software is like sex: it's better when it's free." This intro went too long, so let's rock (:

The Story of a simple text scroller

The concept behind the text scroller is very simple, but it may seem difficult first. In Windows we can achieve this by using the TextOut() procedure and by writing the text on the screen in different locations, and this way we can create a scroller effect. Now let's see the actual code for a simple window first, don't be intimidated if you don't understand the code 100%, this is normal. (an API reference should help)

```
#include /*  Declare Windows procedure  */
LRESULT CALLBACK WindowProcedure (HWND, UINT, WPARAM, LPARAM);

/*  Make the class name into a global variable  */
char szClassName[ ] = "WindowsApp";

int WINAPI WinMain (HINSTANCE hThisInstance,
                    HINSTANCE hPrevInstance,
                    LPSTR lpszArgument,
```

```
                int nFunsterStil)

{
    HWND hwnd;               /* This is the handle for our window */
    MSG messages;            /* Here messages to the application are saved */
    WNDCLASSEX wincl;        /* Data structure for the windowclass */

    /* The Window structure */
    wincl.hInstance = hThisInstance;
    wincl.lpszClassName = szClassName;
    wincl.lpfnWndProc = WindowProcedure;      /* This function is called by windows */
    wincl.style = CS_DBLCLKS;                 /* Catch double-clicks */
    wincl.cbSize = sizeof (WNDCLASSEX);

    /* Use default icon and mouse-pointer */
    wincl.hIcon = LoadIcon (NULL, IDI_APPLICATION);
    wincl.hIconSm = LoadIcon (NULL, IDI_APPLICATION);
    wincl.hCursor = LoadCursor (NULL, IDC_ARROW);
    wincl.lpszMenuName = NULL;                /* No menu */
    wincl.cbClsExtra = 0;                     /* No extra bytes after the window class */
    wincl.cbWndExtra = 0;                     /* structure or the window instance */
    /* Use Windows's default color as the background of the window */
    wincl.hbrBackground = (HBRUSH) COLOR_BACKGROUND;

    /* Register the window class, and if it fails quit the program */
    if (!RegisterClassEx (&wincl))
        return 0;

    /* The class is registered, let's create the program*/
    hwnd = CreateWindowEx (
           0,                   /* Extended possibilites for variation */
           szClassName,         /* Classname */
           "Windows App",       /* Title Text */
           WS_OVERLAPPEDWINDOW, /* default window */
           CW_USEDEFAULT,       /* Windows decides the position */
           CW_USEDEFAULT,       /* where the window ends up on the screen */
           544,                 /* The programs width */
           375,                 /* and height in pixels */
```

```
        HWND_DESKTOP,       /* The window is a child-window to desktop */
        NULL,               /* No menu */
        hThisInstance,      /* Program Instance handler */
        NULL                /* No Window Creation data */
        );

    /* Make the window visible on the screen */
    ShowWindow (hwnd, nFunsterStil);

    /* Run the message loop. It will run until GetMessage() returns 0 */
    while (GetMessage (&messages, NULL, 0, 0))
    {
        /* Translate virtual-key messages into character messages */
        TranslateMessage(&messages);
        /* Send message to WindowProcedure */
        DispatchMessage(&messages);
    }

    /* The program return-value is 0 - The value that PostQuitMessage() gave */
    return messages.wParam;
}


/*  This function is called by the Windows function DispatchMessage()  */

LRESULT CALLBACK WindowProcedure (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    switch (message)                /* handle the messages */
    {
        case WM_DESTROY:
            PostQuitMessage (0);       /* send a WM_QUIT to the message queue */
            break;
        default:                      /* for messages that we don't deal with */
            return DefWindowProc (hwnd, message, wParam, lParam);
    }

    return 0;
}
```

This is the standard code for creating/maintaining a window using "pure" C and the Win32API. If you want to get deeper inside Windows Programming then you can start with "theForger's" tutorials. Now we are going to change a few things in this code, all the modified lines are highlighted.

#include #define WND_WIDTH 500 /* stores the width of our window */
#define WND_HEIGHT 100 /* stores the height of our window */

#define ID_TIMER1   1000 /* ID for our timer */
#define TIMER1_INTERVAL   10   /* our timer interval in MS */

#define MAGIC_NUMBER1 550 /* magic number #1 */
#define MAGIC_NUMBER2 650 /* magic number #2 */

/* Define some colors using RGB values */
#define black RGB(0,0,0)
#define silver RGB(192,192,192)
#define red   RGB(255,0,0)
#define lime   RGB(0,255,0)
#define blue   RGB(0,0,255)
#define yellow RGB(255,255,0)
#define white RGB(255,255,255)

/* Declare Windows procedure */
LRESULT CALLBACK WindowProcedure (HWND, UINT, WPARAM, LPARAM);

/* This is the actual scroller procedure */
void ScrollDemo(HDC hdc, RECT rect);

/* Make the class name into a global variable */
char szClassName[ ] = "ScrollerDemo"; /* modified */

/* Texts to be scrolled (: */
char   text1[]   = "Horizontal Text Scroller Demo - Coded by The Icebreaker";
char   text2[]   = "| e-mail: -email- | web: www.icebreaker.3x.ro |";

/* Loop Variables used to scroll the texts */

```
int     loop1       = MAGIC_NUMBER1;    /* magic number #1 */
int     loop2       = MAGIC_NUMBER2;    /* magic number #2 */

/* the window background color */
COLORREF bgcolor      = black;
/* choose the text colors */
COLORREF text1_color   = red;
COLORREF text2_color   = lime;

/* Handles for the fonts */
HFONT hFont1,hFont2;


int WINAPI WinMain (HINSTANCE hThisInstance,
                    HINSTANCE hPrevInstance,
                    LPSTR lpszArgument,
                    int nFunsterStil)

{
    HWND hwnd;               /* This is the handle for our window */
    MSG messages;            /* Here messages to the application are saved */
    WNDCLASSEX wincl;        /* Data structure for the windowclass */

    /* The Window structure */
    wincl.hInstance = hThisInstance;
    wincl.lpszClassName = szClassName;
    wincl.lpfnWndProc = WindowProcedure;      /* This function is called by windows */
    wincl.style = CS_DBLCLKS;                 /* Catch double-clicks */
    wincl.cbSize = sizeof (WNDCLASSEX);

    /* Use default icon and mouse-pointer */
    wincl.hIcon = LoadIcon (NULL, IDI_APPLICATION);
    wincl.hIconSm = LoadIcon (NULL, IDI_APPLICATION);
    wincl.hCursor = LoadCursor (NULL, IDC_ARROW);
    wincl.lpszMenuName = NULL;                /* No menu */
    wincl.cbClsExtra = 0;                     /* No extra bytes after the window class */
    wincl.cbWndExtra = 0;                     /* structure or the window instance */
    /* Use Windows's default color as the background of the window */
```

```
    wincl.hbrBackground = (HBRUSH) COLOR_BACKGROUND;

    /* Register the window class, and if it fails quit the program */
    if (!RegisterClassEx (&wincl))
        return 0;

    /* The class is registered, let's create the program*/
    hwnd = CreateWindowEx (
           0,                   /* Extended possibilites for variation */
           szClassName,         /* Classname */
           "Horizontal Text Scroller Demo",       /* modified -  Title Text */
           WS_SYSMENU | WS_MINIMIZEBOX | WS_BORDER, /* modified - no resize,no maximizebox */
           CW_USEDEFAULT,       /* Windows decides the position */
           CW_USEDEFAULT,       /* where the window ends up on the screen */
           WND_WIDTH,           /* The programs width */
           WND_HEIGHT,          /* and height in pixels */
           HWND_DESKTOP,        /* The window is a child-window to desktop */
           NULL,                /* No menu */
           hThisInstance,       /* Program Instance handler */
           NULL                 /* No Window Creation data */
           );

    /* Make the window visible on the screen */
    ShowWindow (hwnd, nFunsterStil);

    /* Run the message loop. It will run until GetMessage() returns 0 */
    while (GetMessage (&messages, NULL, 0, 0))
    {
        /* Translate virtual-key messages into character messages */
        TranslateMessage(&messages);
        /* Send message to WindowProcedure */
        DispatchMessage(&messages);
    }

    /* The program return-value is 0 - The value that PostQuitMessage() gave */
    return messages.wParam;
}
```

```
/*  This function is called by the Windows function DispatchMessage()  */

LRESULT CALLBACK WindowProcedure (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
   switch (message)                  /* handle the messages */
   {
      case WM_CREATE:
         {
          /* we create our two fonts */
          hFont1 = CreateFont(-16, 0, 0, 0, 400, 0, 0, 0, DEFAULT_CHARSET, OUT_DEFAULT_PRECIS,
                CLIP_DEFAULT_PRECIS, DEFAULT_QUALITY, DEFAULT_PITCH | FF_DONTCARE, "Arial Black");
          hFont2 = CreateFont(-12, 0, 0, 0, 400, 0, 0, 0, DEFAULT_CHARSET, OUT_DEFAULT_PRECIS,
                CLIP_DEFAULT_PRECIS, DEFAULT_QUALITY, DEFAULT_PITCH | FF_DONTCARE, "Arial Black");

          /* we setup our timer */
          SetTimer(hwnd,ID_TIMER1,TIMER1_INTERVAL,NULL);

          break;
         }
      case WM_TIMER:
         {
          /* it's our timer1? */
          if (wParam == ID_TIMER1) {

             RECT rect;
             HDC hdc = GetDC(hwnd);

             GetClientRect(hwnd,&rect);   /* get da window client rect (: */

             ScrollDemo(hdc,rect);        /* draw da stuff */

             ReleaseDC(hwnd,hdc);

          }
          break;
         }
      case WM_KEYDOWN:
         {
```

```
            /* user pressed escape? if yes then exit! */
            if (wParam == VK_ESCAPE) PostQuitMessage(0);
            break;
           }
        case WM_CLOSE:
           {
            /* stop the timer, and clean-up this mess (: */
            KillTimer(hwnd,ID_TIMER1);
            DeleteObject(hFont1);
            DeleteObject(hFont2);
            DestroyWindow(hwnd); /* finally destroy the window */
            break;
           }

        case WM_DESTROY:
           PostQuitMessage (0);       /* send a WM_QUIT to the message queue */
           break;
        default:                      /* for messages that we don't deal with */
           return DefWindowProc (hwnd, message, wParam, lParam);
     }

     return 0;
}


/*
  This is the actual scroller procedure.
  We use Double-Buffering to avoid flickering, and create a smooth animation. (:
*/
void ScrollDemo(HDC hdc, RECT rect)
{
  LOGBRUSH logBrush;
  HBRUSH   hBrush;

  /* Create our "fake" canvas */
  HDC     hdcBuffer    = CreateCompatibleDC(hdc);
  HBITMAP hbmBuffer    = CreateCompatibleBitmap(hdc, rect.right, rect.bottom);
  HDC     hbmOldBuffer = SelectObject(hdcBuffer, hbmBuffer);
```

```
  logBrush.lbStyle = BS_SOLID;
  logBrush.lbColor = bgcolor;
  logBrush.lbHatch  = 0;

  hBrush = CreateBrushIndirect(&logBrush);

  /* Clear our "fake" canvas */
  FillRect(hdcBuffer, &rect, hBrush);
  SetBkMode(hdcBuffer, TRANSPARENT);

  /* Write The First Text */
  SelectObject(hdcBuffer,hFont1);

  SetTextColor(hdcBuffer,text1_color);

  if (loop1>-MAGIC_NUMBER1) { loop1--; } else { loop1=MAGIC_NUMBER1; } /* where to draw the text? */

  TextOut(hdcBuffer,loop1,10,text1,strlen(text1));  /* draw to the "fake" canvas */

  /* Write The Second Text */
  SelectObject(hdcBuffer,hFont2);

  SetTextColor(hdcBuffer,text2_color);

  if (loop2>-MAGIC_NUMBER2) { loop2--; } else { loop2=MAGIC_NUMBER2; } /* where to draw the text? */

  TextOut(hdcBuffer,loop2,40,text2,strlen(text2)); /* draw to the "fake" canvas */

  BitBlt(hdc, 0, 0, rect.right, rect.bottom, hdcBuffer, 0, 0, SRCCOPY); /* BufferSwap?!! */

  /* clean-up this mess (: */
  DeleteObject(hBrush);

  SelectObject(hdcBuffer, hbmOldBuffer);
  DeleteDC(hdcBuffer); /* delete our "fake" canvas */
  DeleteObject(hbmBuffer);
}
```

Duhhh.... here is a screenshot ... hmmm I think that lots of things have to explained here (:

First of all we have defined the WIDTH and the HEIGHT of the window, so we can change them in a professional way. Then we define other two constants used by the TIMER, which is used to draw the whole thing on the screen. The TIMER1_INTERVAL represents how often we update the screen. In this case it is 10 ms, so the animation will appear quite smooth. Another interesting thing that you may have noticed while reading the source code is that we define two magic numbers. Well, you have to figure out these ones for yourself if you change the texts, (and you will change the texts!!!!), however it is possible to write some kind of procedure to figure out those numbers for you automatically, I wanted to keep this tutorial as simple as possible. (KISS = Keep It Simple Stupid)

Next we define some colors using RGB (Red, Green, Blue) values for easier management. The other variables are self explanatory. We also changed the Window Look in the CreateWindowEx(), but major changes are in WindowProcedure, so let's do a quick review of what we added here. In WM_CREATE section which is executed once the application is started, we create the two Fonts (using Arial Black, but you can change them to any available font on your system), and we setup the timer. In the WM_TIMER section first we check if it's our timer by comparing wParam with ID_TIMER1. If yes then we get the window canvas using HDC hdc = GetDC(hwnd); then we get the actual window width and height minus the caption bar, border and stuff. Then we call our ScrollDemo() procedure with the two values we got before. Finally we release the window canvas by calling ReleaseDC(hwnd,hdc);. The WM_KEYDOWN section is self explanatory. In the WM_CLOSE section we stop the timer, delete the two fonts created earlier and we destroy the window. Finally we have arrived at our procedure ScrollDemo(). First I think that we need describe what Double-Buffering is. If we just draw on the screen at very short intervals, our animation will start to flicker, and it will look like a mess);, and we don't want to this happen, so we setup another "fake" virtual screen and we draw all the stuff on it, and finally we draw the whole fake screen over the visible screen. This way we avoid flickering, because the screen is updated only once, but we can perform as many operations we want on the "fake" screen. Now let's take a look at our procedure. First we setup our fake screen, the we clear it to the background color we want, then we select our font and text color, and we draw our text on the "fake" screen. We do the same thing with the second text, and finally we draw the whole thing over the real screen. After this we can clear/delete the used variables. The two loops are used the find out where to draw our texts at a specified time,and here we make use of those magic numbers defined before.

Now I think that you should compile, if you haven't done it before. Version 1.0 was made, because there will be a Version 2.0. I think that many of you have seen the "About" box of the "Serials 2k", if not then I will tell you now that there the background is fading from black to silver and from silver to black. How we can achieve this?!! Here is the code:

```
#include #define WND_WIDTH  500  /* stores the width of our window  */
#define WND_HEIGHT 100  /* stores the height of our window */

#define ID_TIMER1          1000 /* ID for our timer */
#define TIMER1_INTERVAL    10   /* our timer interval in MS */
```

```
#define ID_TIMER2          1001 /* ID for our second timer */
#define TIMER2_INTERVAL    5    /* our timer (2) interval in MS */

#define MAGIC_NUMBER1 550  /* magic number #1 */
#define MAGIC_NUMBER2 650  /* magic number #2 */

/* Define some colors using RGB values */
#define black  RGB(0,0,0)
#define silver RGB(192,192,192)
#define red    RGB(255,0,0)
#define lime   RGB(0,255,0)
#define blue   RGB(0,0,255)
#define yellow RGB(255,255,0)
#define white  RGB(255,255,255)

/* Declare Windows procedure  */
LRESULT CALLBACK WindowProcedure (HWND, UINT, WPARAM, LPARAM);

/* This is the actual scroller procedure */
void ScrollDemo(HDC hdc, RECT rect);

/* This the background changer stuff */
void ChangeBackground(void);

/* Mess with the colors (: */
COLORREF FadeIn(COLORREF old_color);
COLORREF FadeOut(COLORREF old_color);

/* Make the class name into a global variable  */
char szClassName[ ] = "ScrollerDemo";  /* modified */

/* Texts to be scrolled (: */
char    text1[]      = "Horizontal Text Scroller Demo - Coded by The Icebreaker";
char    text2[]      = "| e-mail: -email- | web: www.icebreaker.3x.ro |";

/* Loop Variables used to scroll the texts */
int     loop1        = MAGIC_NUMBER1;    /* magic number #1 */
int     loop2        = MAGIC_NUMBER2;    /* magic number #2 */
```

```
/* the window background color */
COLORREF bgcolor       = silver;
/* choose the text colors */
COLORREF text1_color   = red;
COLORREF text2_color   = black;

/* Handles for the fonts */
HFONT hFont1,hFont2;

int WINAPI WinMain (HINSTANCE hThisInstance,
                    HINSTANCE hPrevInstance,
                    LPSTR lpszArgument,
                    int nFunsterStil)

{
    HWND hwnd;               /* This is the handle for our window */
    MSG messages;            /* Here messages to the application are saved */
    WNDCLASSEX wincl;        /* Data structure for the windowclass */

    /* The Window structure */
    wincl.hInstance = hThisInstance;
    wincl.lpszClassName = szClassName;
    wincl.lpfnWndProc = WindowProcedure;      /* This function is called by windows */
    wincl.style = CS_DBLCLKS;                 /* Catch double-clicks */
    wincl.cbSize = sizeof (WNDCLASSEX);

    /* Use default icon and mouse-pointer */
    wincl.hIcon = LoadIcon (NULL, IDI_APPLICATION);
    wincl.hIconSm = LoadIcon (NULL, IDI_APPLICATION);
    wincl.hCursor = LoadCursor (NULL, IDC_ARROW);
    wincl.lpszMenuName = NULL;                /* No menu */
    wincl.cbClsExtra = 0;                     /* No extra bytes after the window class */
    wincl.cbWndExtra = 0;                     /* structure or the window instance */
    /* Use Windows's default color as the background of the window */
    wincl.hbrBackground = (HBRUSH) COLOR_BACKGROUND;

    /* Register the window class, and if it fails quit the program */
    if (!RegisterClassEx (&wincl))
```

```
    return 0;

  /* The class is registered, let's create the program*/
  hwnd = CreateWindowEx (
      0,                    /* Extended possibilites for variation */
      szClassName,          /* Classname */
      "Horizontal Text Scroller Demo",  /* modified -  Title Text */
      WS_SYSMENU | WS_MINIMIZEBOX | WS_BORDER, /* modified - no resize,no maximizebox */
      CW_USEDEFAULT,        /* Windows decides the position */
      CW_USEDEFAULT,        /* where the window ends up on the screen */
      WND_WIDTH,            /* The programs width */
      WND_HEIGHT,           /* and height in pixels */
      HWND_DESKTOP,         /* The window is a child-window to desktop */
      NULL,                 /* No menu */
      hThisInstance,        /* Program Instance handler */
      NULL                  /* No Window Creation data */
      );

  /* Make the window visible on the screen */
  ShowWindow (hwnd, nFunsterStil);

  /* Run the message loop. It will run until GetMessage() returns 0 */
  while (GetMessage (&messages, NULL, 0, 0))
  {
    /* Translate virtual-key messages into character messages */
    TranslateMessage(&messages);
    /* Send message to WindowProcedure */
    DispatchMessage(&messages);
  }

  /* The program return-value is 0 - The value that PostQuitMessage() gave */
  return messages.wParam;
}


/*  This function is called by the Windows function DispatchMessage()  */

LRESULT CALLBACK WindowProcedure (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
```

```
{
   switch (message)                /* handle the messages */
   {
      case WM_CREATE:
         {
          /* we create our two fonts */
          hFont1 = CreateFont(-16, 0, 0, 0, 400, 0, 0, 0, DEFAULT_CHARSET, OUT_DEFAULT_PRECIS,
                CLIP_DEFAULT_PRECIS, DEFAULT_QUALITY, DEFAULT_PITCH | FF_DONTCARE, "Arial Black");
          hFont2 = CreateFont(-12, 0, 0, 0, 400, 0, 0, 0, DEFAULT_CHARSET, OUT_DEFAULT_PRECIS,
                CLIP_DEFAULT_PRECIS, DEFAULT_QUALITY, DEFAULT_PITCH | FF_DONTCARE, "Arial Black");

          /* we setup our timer */
          SetTimer(hwnd,ID_TIMER1,TIMER1_INTERVAL,NULL);

          SetTimer(hwnd,ID_TIMER2,TIMER2_INTERVAL,NULL);

          break;
         }
      case WM_TIMER:
         {
          /* it's our timer1? */
          if (wParam == ID_TIMER1) {

             RECT rect;
             HDC hdc = GetDC(hwnd);

             GetClientRect(hwnd,&rect);   /* get da window client rect (: */

             ScrollDemo(hdc,rect);        /* draw da stuff */

             ReleaseDC(hwnd,hdc);

          }
          if (wParam == ID_TIMER2) ChangeBackground();
          break;
         }
      case WM_KEYDOWN:
         {
```

```
            /* user pressed escape? if yes then exit! */
            if (wParam == VK_ESCAPE) PostQuitMessage(0);
            break;
           }
        case WM_CLOSE:
           {
            /* stop the timer, and clean-up this mess (: */
            KillTimer(hwnd,ID_TIMER1);
            KillTimer(hwnd,ID_TIMER2);
            DeleteObject(hFont1);
            DeleteObject(hFont2);
            DestroyWindow(hwnd); /* finally destroy the window */
            break;
           }
        case WM_DESTROY:
           PostQuitMessage (0);       /* send a WM_QUIT to the message queue */
           break;
        default:                      /* for messages that we don't deal with */
           return DefWindowProc (hwnd, message, wParam, lParam);
    }

    return 0;
}

/*
  This is the actual scroller procedure.
  We use Double-Buffering to avoid flickering, and create a smooth animation. (:
*/
void ScrollDemo(HDC hdc, RECT rect)
{
  LOGBRUSH logBrush;
  HBRUSH   hBrush;

  /* Create our "fake" canvas */
  HDC     hdcBuffer    = CreateCompatibleDC(hdc);
  HBITMAP hbmBuffer    = CreateCompatibleBitmap(hdc, rect.right, rect.bottom);
  HDC     hbmOldBuffer = SelectObject(hdcBuffer, hbmBuffer);
```

```
  logBrush.lbStyle = BS_SOLID;
  logBrush.lbColor = bgcolor;
  logBrush.lbHatch  = 0;

  hBrush = CreateBrushIndirect(&logBrush);

  /* Clear our "fake" canvas */
  FillRect(hdcBuffer, &rect, hBrush);
  SetBkMode(hdcBuffer, TRANSPARENT);

  /* Write The First Text */
  SelectObject(hdcBuffer,hFont1);

  SetTextColor(hdcBuffer,text1_color);

  if (loop1>-MAGIC_NUMBER1) { loop1--; } else { loop1=MAGIC_NUMBER1; } /* where to draw the text? */

  TextOut(hdcBuffer,loop1,10,text1,strlen(text1));  /* draw to the "fake" canvas */

  /* Write The Second Text */
  SelectObject(hdcBuffer,hFont2);

  SetTextColor(hdcBuffer,text2_color);

  if (loop2>-MAGIC_NUMBER2) { loop2--; } else { loop2=MAGIC_NUMBER2; } /* where to draw the text? */

  TextOut(hdcBuffer,loop2,40,text2,strlen(text2)); /* draw to the "fake" canvas */

  BitBlt(hdc, 0, 0, rect.right, rect.bottom, hdcBuffer, 0, 0, SRCCOPY); /* BufferSwap?!! */

  /* clean-up this mess (: */
  DeleteObject(hBrush);

  SelectObject(hdcBuffer, hbmOldBuffer);
  DeleteDC(hdcBuffer); /* delete our "fake" canvas */
  DeleteObject(hbmBuffer);
}
```

```
COLORREF FadeIn(COLORREF old_color)
{
  unsigned int r,g,b;

  r=GetRValue(old_color); r++;
  g=GetGValue(old_color); g++;
  b=GetBValue(old_color); b++;

  return RGB(r,g,b);
}

COLORREF FadeOut(COLORREF old_color)
{
  unsigned int r,g,b;

  r=GetRValue(old_color); r--;
  g=GetGValue(old_color); g--;
  b=GetBValue(old_color); b--;

  return RGB(r,g,b);
}

void ChangeBackground(void)
{
 static unsigned int dir = 0;
 if (bgcolor == black ) dir=0; /* set the direction { FadeOut || FadeIn } */
 if (bgcolor == silver) dir=1;
 if (dir) { bgcolor = FadeOut(bgcolor); } else { bgcolor = FadeIn(bgcolor); }
}
```

We have defined another timer, and we have added three new procedures. The FadeIn increments the color's RGB value by one, the FadeOut decrements the color's RGB value by one. The ChangeBackground procedure just switches between the two color changer functions, achieving a nice fading effect. This procedure is called by our second timer, and two other variables are modified; the bgcolor is defined by default as silver and the text2_color is black by default. All the other stuff is the same. Enjoy (:

There is no demo without music ...

Finally we will add a simple "mod" music player to our demo. If you "get" the concept, it shouldn't be hard to implement a simple

"stream" player if you need it. For music we use the FMOD sound system from fmod.org. To compile this version you will need to Install FMOD for your compiler, and you will need to link the library for this exe to work. If the space is a problem (for example when you are coding a 64KB demo) then use MiniFMOD from the same website. The code is well commented and pretty much self explanatory,so I won't give any explanations here:

```
#include #include #include #define WND_WIDTH  500  /* stores the width of our window  */
#define WND_HEIGHT 100  /* stores the height of our window */

#define ID_TIMER1          1000 /* ID for our timer */
#define TIMER1_INTERVAL    10   /* our timer interval in MS */

#define ID_TIMER2          1001 /* ID for our second timer */
#define TIMER2_INTERVAL    5    /* our timer (2) interval in MS */

#define MAGIC_NUMBER1 550  /* magic number #1 */
#define MAGIC_NUMBER2 650  /* magic number #2 */

/* Define some colors using RGB values */
#define black  RGB(0,0,0)
#define silver RGB(192,192,192)
#define red    RGB(255,0,0)
#define lime   RGB(0,255,0)
#define blue   RGB(0,0,255)
#define yellow RGB(255,255,0)
#define white  RGB(255,255,255)

#define MUSIC "music.xm" /* the music file name */

/* Declare Windows procedure  */
LRESULT CALLBACK WindowProcedure (HWND, UINT, WPARAM, LPARAM);

/* This is the actual scroller procedure */
void ScrollDemo(HDC hdc, RECT rect);

/* This the background changer stuff */
void ChangeBackground(void);
```

```
/* Mess with the colors (: */
COLORREF FadeIn(COLORREF old_color);
COLORREF FadeOut(COLORREF old_color);

/* Make the class name into a global variable  */
char szClassName[ ] = "ScrollerDemo";  /* modified */

/* Texts to be scrolled (: */
char    text1[]      = "Horizontal Text Scroller Demo - Coded by The Icebreaker";
char    text2[]      = "| e-mail: -email- | web: www.icebreaker.3x.ro |";

/* Loop Variables used to scroll the texts */
int     loop1        = MAGIC_NUMBER1;    /* magic number #1 */
int     loop2        = MAGIC_NUMBER2;    /* magic number #2 */

/* the window background color */
COLORREF bgcolor      = silver;
/* choose the text colors */
COLORREF text1_color   = red;
COLORREF text2_color   = black;

/* Handles for the fonts */
HFONT hFont1,hFont2;


/* Music related (: */
FMUSIC_MODULE *mod = 0;


int WINAPI WinMain (HINSTANCE hThisInstance,
                    HINSTANCE hPrevInstance,
                    LPSTR lpszArgument,
                    int nFunsterStil)

{
   HWND hwnd;               /* This is the handle for our window */
   MSG messages;            /* Here messages to the application are saved */
   WNDCLASSEX wincl;        /* Data structure for the windowclass */
```

```
/* We check out the FMOD version and we Initialize the sound system */

if (FSOUND_GetVersion()     {
    MessageBox(0,"FSOUND_GetVersion() - You are using the wrong DLL version!","Fatal Error",MB_ICONERROR);
    return 1;
}

/* Init the sound system with default values */
if (!FSOUND_Init(44100, 32, FSOUND_INIT_USEDEFAULTMIDISYNTH))
{
    MessageBox(0,"FSOUND_Init() failed!","Fatal Error",MB_ICONERROR);
    return 1;
}


/* The Window structure */
wincl.hInstance = hThisInstance;
wincl.lpszClassName = szClassName;
wincl.lpfnWndProc = WindowProcedure;      /* This function is called by windows */
wincl.style = CS_DBLCLKS;                 /* Catch double-clicks */
wincl.cbSize = sizeof (WNDCLASSEX);

/* Use default icon and mouse-pointer */
wincl.hIcon = LoadIcon (NULL, IDI_APPLICATION);
wincl.hIconSm = LoadIcon (NULL, IDI_APPLICATION);
wincl.hCursor = LoadCursor (NULL, IDC_ARROW);
wincl.lpszMenuName = NULL;                /* No menu */
wincl.cbClsExtra = 0;                     /* No extra bytes after the window class */
wincl.cbWndExtra = 0;                     /* structure or the window instance */
/* Use Windows's default color as the background of the window */
wincl.hbrBackground = (HBRUSH) COLOR_BACKGROUND;

/* Register the window class, and if it fails quit the program */
if (!RegisterClassEx (&wincl))
    return 0;

/* The class is registered, let's create the program*/
```

```
   hwnd = CreateWindowEx (
          0,                    /* Extended possibilites for variation */
          szClassName,          /* Classname */
          "Horizontal Text Scroller Demo",       /* modified -  Title Text */
          WS_SYSMENU | WS_MINIMIZEBOX | WS_BORDER, /* modified - no resize,no maximizebox */
          CW_USEDEFAULT,        /* Windows decides the position */
          CW_USEDEFAULT,        /* where the window ends up on the screen */
          WND_WIDTH,            /* The programs width */
          WND_HEIGHT,           /* and height in pixels */
          HWND_DESKTOP,         /* The window is a child-window to desktop */
          NULL,                 /* No menu */
          hThisInstance,        /* Program Instance handler */
          NULL                  /* No Window Creation data */
          );

   /* Make the window visible on the screen */
   ShowWindow (hwnd, nFunsterStil);

   /* Run the message loop. It will run until GetMessage() returns 0 */
   while (GetMessage (&messages, NULL, 0, 0))
   {
      /* Translate virtual-key messages into character messages */
      TranslateMessage(&messages);
      /* Send message to WindowProcedure */
      DispatchMessage(&messages);
   }

   /* The program return-value is 0 - The value that PostQuitMessage() gave */
   return messages.wParam;
}


/*  This function is called by the Windows function DispatchMessage()  */

LRESULT CALLBACK WindowProcedure (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
   switch (message)              /* handle the messages */
   {
```

```
case WM_CREATE:
    {
     /* we create our two fonts */
     hFont1 = CreateFont(-16, 0, 0, 0, 400, 0, 0, 0, DEFAULT_CHARSET, OUT_DEFAULT_PRECIS,
            CLIP_DEFAULT_PRECIS, DEFAULT_QUALITY, DEFAULT_PITCH | FF_DONTCARE, "Arial Black");
     hFont2 = CreateFont(-12, 0, 0, 0, 400, 0, 0, 0, DEFAULT_CHARSET, OUT_DEFAULT_PRECIS,
            CLIP_DEFAULT_PRECIS, DEFAULT_QUALITY, DEFAULT_PITCH | FF_DONTCARE, "Arial Black");

     /* we setup our timer */
     SetTimer(hwnd,ID_TIMER1,TIMER1_INTERVAL,NULL);

     SetTimer(hwnd,ID_TIMER2,TIMER2_INTERVAL,NULL);



     /* Load The Song */
     mod = FMUSIC_LoadSong(MUSIC);
     /* Set Looping to TRUE or FALSE */
     FMUSIC_SetLooping(mod, TRUE);
     /* Play the song ... */
     FMUSIC_PlaySong(mod);


     break;
    }
case WM_TIMER:
    {
     /* it's our timer1? */
     if (wParam == ID_TIMER1) {

        RECT rect;
        HDC hdc = GetDC(hwnd);

        GetClientRect(hwnd,&rect);   /* get da window client rect (: */

        ScrollDemo(hdc,rect);        /* draw da stuff */

        ReleaseDC(hwnd,hdc);
```

```
            }
            if (wParam == ID_TIMER2) ChangeBackground();
            break;
           }
       case WM_KEYDOWN:
           {
            /* user pressed escape? if yes then exit! */
            if (wParam == VK_ESCAPE) PostQuitMessage(0);
            break;
           }
       case WM_CLOSE:
           {

             /* Stop the song */
            FMUSIC_StopSong(mod);
            /* Free the memory used by the sound */
            FMUSIC_FreeSong(mod);
            /* shutdown the sound system */
            FSOUND_Close();

            /* stop the timer, and clean-up this mess (: */
            KillTimer(hwnd,ID_TIMER1);
            KillTimer(hwnd,ID_TIMER2);
            DeleteObject(hFont1);
            DeleteObject(hFont2);
            DestroyWindow(hwnd); /* finally destroy the window */
            break;
           }
       case WM_DESTROY:
           PostQuitMessage (0);       /* send a WM_QUIT to the message queue */
           break;
       default:                   /* for messages that we don't deal with */
           return DefWindowProc (hwnd, message, wParam, lParam);
   }

   return 0;
}
```

```
/*
  This is the actual scroller procedure.
  We use Double-Buffering to avoid flickering, and create a smooth animation. (:
*/
void ScrollDemo(HDC hdc, RECT rect)
{
  LOGBRUSH logBrush;
  HBRUSH   hBrush;

  /* Create our "fake" canvas */
  HDC     hdcBuffer    = CreateCompatibleDC(hdc);
  HBITMAP hbmBuffer    = CreateCompatibleBitmap(hdc, rect.right, rect.bottom);
  HDC     hbmOldBuffer = SelectObject(hdcBuffer, hbmBuffer);

  logBrush.lbStyle = BS_SOLID;
  logBrush.lbColor = bgcolor;
  logBrush.lbHatch  = 0;

  hBrush = CreateBrushIndirect(&logBrush);

  /* Clear our "fake" canvas */
  FillRect(hdcBuffer, &rect, hBrush);
  SetBkMode(hdcBuffer, TRANSPARENT);

  /* Write The First Text */
  SelectObject(hdcBuffer,hFont1);

  SetTextColor(hdcBuffer,text1_color);

  if (loop1>-MAGIC_NUMBER1) { loop1--; } else { loop1=MAGIC_NUMBER1; } /* where to draw the text? */

  TextOut(hdcBuffer,loop1,10,text1,strlen(text1));  /* draw to the "fake" canvas */

  /* Write The Second Text */
  SelectObject(hdcBuffer,hFont2);

  SetTextColor(hdcBuffer,text2_color);
```

```
  if (loop2>-MAGIC_NUMBER2) { loop2--; } else { loop2=MAGIC_NUMBER2; } /* where to draw the text? */

  TextOut(hdcBuffer,loop2,40,text2,strlen(text2)); /* draw to the "fake" canvas */

  BitBlt(hdc, 0, 0, rect.right, rect.bottom, hdcBuffer, 0, 0, SRCCOPY); /* BufferSwap?!! */

  /* clean-up this mess (: */
  DeleteObject(hBrush);

  SelectObject(hdcBuffer, hbmOldBuffer);
  DeleteDC(hdcBuffer); /* delete our "fake" canvas */
  DeleteObject(hbmBuffer);
}
COLORREF FadeIn(COLORREF old_color)
{
  unsigned int r,g,b;

  r=GetRValue(old_color); r++;
  g=GetGValue(old_color); g++;
  b=GetBValue(old_color); b++;

  return RGB(r,g,b);
}

COLORREF FadeOut(COLORREF old_color)
{
  unsigned int r,g,b;

  r=GetRValue(old_color); r--;
  g=GetGValue(old_color); g--;
  b=GetBValue(old_color); b--;

  return RGB(r,g,b);
}

void ChangeBackground(void)
{
 static unsigned int dir = 0;
```

```
 if (bgcolor == black ) dir=0; /* set the direction { FadeOut || FadeIn } */
 if (bgcolor == silver) dir=1;
 if (dir) { bgcolor = FadeOut(bgcolor); } else { bgcolor = FadeIn(bgcolor); }
}
```

Outroduction

I hope that you have enjoyed the tutorial, and you have learned something from this. If you have any comments,questions,suggestions, or if you have found any errors please drop me a mail at: -email-. I also have a website somewhere, hummm .... mrrr... here it is: www.icebreaker.3x.ro. Don't forget that Without knowledge, no power!, and that the code is public domain, you can do whatever you want with it, but give me credits if necessary (: If Adok/Hugi publishes this crap (hehe) then I have a few ideas for future tutorials.

Ps: Don't bother to send mails to blame me, and to say that this tutorial is a crap. If you do so then pay attention to this: Mess with the best, die like the rest!

The source be with you!

The Icebreaker