

sebastianfernandes.com

[Capturing the screen (Win32/C++)]

Because of the functions provided with the Windows API, capturing the screen to the Clipboard becomes a relatively simple task. This tutorial assumes that you have a basic knowledge of C++ and some understanding of the Windows environment.

In the sample code that complements this tutorial, the following calls have been added to the WinMain procedure. However, your application does much more, and you will have identified a more suitable place to add these.

1. Include `stdafx.h`. The functions and constants we will require will be declared/defined in `winuser.h` and `wingdi.h`, which are included in `windows.h` – in turn included in `stdafx.h`.

```
#include "stdafx.h"
```

2. Get the device context of the Desktop by calling the `GetDC` API and passing it the Desktop handle (`HWND_DESKTOP`).

```
hdc = GetDC( HWND_DESKTOP );
```

3. We can now determine the screen dimensions.

```
scrWidth = GetDeviceCaps( hdc, HORZRES );  
scrHeight = GetDeviceCaps( hdc, VERTRES );
```

4. Next, we create a compatible memory context and bitmap.

```
hdcMem = CreateCompatibleDC( hdc );  
hBitmap = CreateCompatibleBitmap( hdc, scrWidth, scrHeight );
```

If this was successful, `hBitmap` should not be `NULL`, i.e. “`if (hBitmap)`” should evaluate to true.

5. Assuming step 4 was successful, we can select the bitmap into the memory context we just created.

```
SelectObject(hdcMem, hBitmap);
```

6. Next, we copy the Desktop to the memory device context.

```
BitBlt( hdcMem, 0, 0, scrWidth, scrHeight,  
        hdc, 0, 0, SRCCOPY );
```

7. The clipboard – open, clear, copy, and close.

```
OpenClipboard( NULL );  
EmptyClipboard();  
SetClipboardData( CF_BITMAP, hBitmap );  
CloseClipboard();
```

OK, not so fast. We opened the clipboard providing the handle to our application window. In the sample code, there is no application window, so we passed NULL. This associated the Clipboard with the current task.

When we emptied the Clipboard, we both cleared it's current contents and temporarily took ownership of the Clipboard. We placed the bitmap data onto the Clipboard by specifying the data type with CF_BITMAP and providing a handle to our data. Then, we released ownership of the Clipboard.

8. Finally, we cleanup by deleting the memory context and releasing the device context.

```
DeleteDC( hdcMem );  
ReleaseDC( HWND_DESKTOP, hdc );
```

The completed procedure is as follows:

```
/* sfcapscr.cpp : Capture screen to clipboard

This application simply copies the screen to the system
clipboard - nothing more than you would get by pressing
Print Screen on your keyboard!  It's only value is as a
demonstration of how you can do the same thing
programmatically....

Copyright (c) 2002, sebastianfernandes.com

*/

#include "stdafx.h"

int APIENTRY WinMain(HINSTANCE hInstance,
                    HINSTANCE hPrevInstance,
                    LPSTR      lpCmdLine,
                    int        nCmdShow)
{
    HDC          hdc, hdcMem;
    HBITMAP      hBitmap;
    static int    scrWidth, scrHeight;

    hdc = GetDC( HWND_DESKTOP );
    scrWidth = GetDeviceCaps( hdc, HORZRES );
    scrHeight = GetDeviceCaps( hdc, VERTRES );
    hdcMem = CreateCompatibleDC( hdc );
    hBitmap = CreateCompatibleBitmap( hdc, scrWidth, scrHeight );
    if( hBitmap )
    {
        SelectObject(hdcMem, hBitmap);
        BitBlt( hdcMem, 0, 0, scrWidth, scrHeight,
                hdc, 0, 0, SRCCOPY );
        OpenClipboard( NULL );
        EmptyClipboard();
        SetClipboardData( CF_BITMAP, hBitmap );
        CloseClipboard();
    }
    DeleteDC( hdcMem );
    ReleaseDC( HWND_DESKTOP, hdc );

    return 0;
}
```