Display a custom thumbnail for your application (and while you're at it, a custom live preview)

devblogs.microsoft.com/oldnewthing/20130225-00

February 25, 2013

Raymond Chen

By default, when the taskbar or any other application wants to display a thumbnail for a window, the result is a copy of the window contents shrunk down to the requested size. Today we're going to override that behavior and display a custom thumbnail.

Take the program from last week and make these changes:

```
#include <dwmapi.h>


BOOL
OnCreate(HWND hwnd, LPCREATESTRUCT lpcs)
{
  g_hicoAlert = LoadIcon(nullptr, IDI_EXCLAMATION);
  g_wmTaskbarButtonCreated = RegisterWindowMessage(
                              TEXT("TaskbarButtonCreated"));
  BOOL fTrue = TRUE;
  DwmSetWindowAttribute(hwnd, DWMWA_FORCE_ICONIC_REPRESENTATION,
                        &fTrue, sizeof(fTrue));
  DwmSetWindowAttribute(hwnd, DWMWA_HAS_ICONIC_BITMAP,
                        &fTrue, sizeof(fTrue));
  return TRUE;
}
```

We start by enabling custom thumbnails by setting the `DWMWA_HAS_ICONIC_BITMAP` attribute to `TRUE`. This overrides the default thumbnail generator and allows us to provide a custom one.

Next is a helper function that I broke out from this program because it's useful on its own. It simply creates a 32bpp bitmap of the desired size and optionally returns a pointer to the resulting bits.

```
HBITMAP Create32bppBitmap(HDC hdc, int cx, int cy,
                          RGBQUAD **pprgbBits = nullptr)
{
 BITMAPINFO bmi = { 0 };
 bmi.bmiHeader.biSize = sizeof(bmi.bmiHeader);
 bmi.bmiHeader.biWidth = cx;
 bmi.bmiHeader.biHeight = cy;
 bmi.bmiHeader.biPlanes = 1;
 bmi.bmiHeader.biBitCount = 32;
 bmi.bmiHeader.biCompression = BI_RGB;
 void *pvBits;
 HBITMAP hbm = CreateDIBSection(hdc, &bmi,
           DIB_RGB_COLORS, &pvBits, NULL, 0);
 if (pprgbBits) *pprgbBits = static_cast<RGBQUAD*>(pvBits);
 return hbm;
}
```

Next, we take our `PaintContent` function and make it render into a DC instead:

```
void
RenderContent(HDC hdc, LPCRECT prc)
{
 LOGFONTW lf = { 0 };
 lf.lfHeight = prc->bottom – prc->top;
 wcscpy_s(lf.lfFaceName, L”Verdana”);
 HFONT hf = CreateFontIndirectW(&lf);
 HFONT hfPrev = SelectFont(hdc, hf);
 wchar_t wszCount[80];
 swprintf_s(wszCount, L”%d”, g_iCounter);
 FillRect(hdc, prc, GetStockBrush(WHITE_BRUSH));
 DrawTextW(hdc, wszCount, -1, const_cast<LPRECT>(prc),
          DT_CENTER | DT_VCENTER | DT_SINGLELINE);
 SelectFont(hdc, hfPrev);
 DeleteObject(hf);
}
```

In our case, we will want to render into a bitmap:

```
HBITMAP
GenerateContentBitmap(HWND hwnd, int cx, int cy)
{
 HDC hdc = GetDC(hwnd);
 HDC hdcMem = CreateCompatibleDC(hdc);
 HBITMAP hbm = Create32bppBitmap(hdcMem, cx,cy);
 HBITMAP hbmPrev = SelectBitmap(hdcMem, hbm);
 RECT rc = { 0, 0, cx, cy };
 RenderContent(hdcMem, &rc);
 SelectBitmap(hdcMem, hbmPrev);
 DeleteDC(hdcMem);
 ReleaseDC(hwnd, hdc);
 return hbm;
}
```

We can use this function when DWM asks us to generate a custom thumbnail or a custom live preview bitmap.

```
void
UpdateThumbnailBitmap(HWND hwnd, int cx, int cy)
{
 HBITMAP hbm = GenerateContentBitmap(hwnd, cx, cy);
 DwmSetIconicThumbnail(hwnd, hbm, 0);
 DeleteObject(hbm);
}


void
UpdateLivePreviewBitmap(HWND hwnd)
{
 RECT rc;
 GetClientRect(hwnd, &rc);
 HBITMAP hbm = GenerateContentBitmap(hwnd, rc.right – rc.left,
                                     rc.bottom – rc.top);
 DwmSetIconicLivePreviewBitmap(hwnd, hbm, nullptr, 0);
 DeleteObject(hbm);
}


 // WndProc
 case WM_DWMSENDICONICTHUMBNAIL:
  UpdateThumbnailBitmap(hwnd, HIWORD(lParam), LOWORD(lParam));
  break;
 case WM_DWMSENDICONICLIVEPREVIEWBITMAP:
  UpdateLivePreviewBitmap(hwnd);
  break;
```

One of the quirks of the `WM_DWMSENDICONICTHUMBNAIL` message is that it passes the x- and y-coordinates backwards. Most window messages put the x-coordinate in the low word and the y-coordinate in the high word, but `WM_DWMSENDICONICTHUMBNAIL` does it the other way

around.

Since we're generating a custom thumbnail and live preview bitmap, we need to let the window manager know that the custom rendering is out of date and needs to be re-rendered: Invalidate the custom bitmaps when the counter changes.

```
void OnCommand(HWND hwnd, int id, HWND hwndCtl, UINT codeNotify)
{
  switch (id) {
  case IDC_INCREMENT:
    ++g_iCounter;
    InvalidateRect(hwnd, nullptr, TRUE);
    DwmInvalidateIconicBitmaps(hwnd);
    break;
  }
}
```

And finally, just to be interesting, we'll also stop rendering content into our main window.

```
void
PaintContent(HWND hwnd, PAINTSTRUCT *pps)
{
    // do nothing
}
```

Run this program and observe that the window comes up blank. Ah, but if you hover over the taskbar button, the custom thumbnail will appear, and that custom thumbnail has the number 0 in it. Click on the button in the thumbnail, and the number in the custom thumbnail increments.

As a bonus, move the mouse over the thumbnail to trigger Aero Peek. The live preview bitmap contains the magic number! Move the mouse away, and the magic number vanishes.

Now, this was an artificial example, so the effect is kind of weird. However, you can imagine using this in less artificial cases where the result is useful. You application might be a game, and instead of using the default thumbnail which shows a miniature copy of the game window, you can have your thumbnail be a tiny scoreboard or focus on a section of the board. For example, if you are a card game, the thumbnail might show just the cards in your hand.

I can't think of a useful case for showing a live preview bitmap different from the actual window. The intended use for a custom live preview bitmap is for applications like Web browsers which want to minimize a tab's memory usage when it is not active. When a tab becomes inactive, the browser can destroy all graphics resources except for a bitmap containing the last-known-valid contents of the window, and use that bitmap for the thumbnail and live preview.

Raymond Chen

Follow