How can I convert a WIC bitmap to a Windows Runtime SoftwareBitmap? part 3: Filling the SoftwareBitmap directly

devblogs.microsoft.com/oldnewthing/20230413-00

April 13, 2023

Raymond Chen

Last time, we converted a WIC bitmap to a Windows Runtime SoftwareBitmap by copying the pixels of the WIC bitmap to a buffer, and then creating the SoftwareBitmap from that same buffer But you don't have to pass the pixels through a buffer. The `SoftwareBitmap` lets you access its pixel buffer directly.

```
winrt::SoftwareBitmap ToSoftwareBitmap(IWICBitmapSource* wicBitmap)
{
    // Look up the Windows Runtime pixel format and alpha mode.
    WICPixelFormatGUID format;
    winrt::check_hresult(wicBitmap->GetPixelFormat(&format));

    static struct Mapping
    {
        WICPixelFormatGUID const& format;
        int bytesPerPixel;
        winrt::BitmapPixelFormat pixelFormat;
        winrt::BitmapAlphaMode alphaMode;
    } const mappings[] = {
        {
            GUID_WICPixelFormat32bppPRGBA,
            4,
            winrt::BitmapPixelFormat::Rgba8,
            winrt::BitmapAlphaMode::Premultiplied
        },
        { ... etc ... },
    };

    auto it = std::find_if(std::begin(mappings),
    std::end(mappings), [&](auto&& mapping)
        { return mapping.format == format; });
    if (it == std::end(mappings)) {
        throw winrt::hresult_error(
            WINCODEC_ERR_UNSUPPORTEDPIXELFORMAT);
    }

    // Create a SoftwareBitmap of the same size and format.
    UINT width, height;
    winrt::check_hresult(wicBitmap->GetSize(&width, &height));
    // Avoid zero-sized or oversized bitmaps (integer overflow)
    if (width == 0 || height == 0 ||
        width > ~0U / it->bytesPerPixel / height) {
        throw winrt::hresult_error(
            WINCODEC_ERR_IMAGESIZEOUTOFRANGE);
    }
    winrt::SoftwareBitmap bitmap(
        pixelFormat, width, height, alphaMode);

    // Copy the pixels into the SoftwareBitmap.
    auto buffer = bitmap.LockBuffer(winrt::BitmapBufferAccessMode::Write);
    if (buffer.GetPlaneCount() != 1) {
        throw winrt::hresult_error(
            WINCODEC_ERR_UNSUPPORTEDPIXELFORMAT);
    }
    auto planeDescription = buffer.GetPlaneDescription(0);
    auto reference = buffer.CreateReference();
    winrt::check_hresult(wicBitmap->CopyPixels(
        nullptr, planeDescription.Stride,
```

```
        reference.Capacity(), reference.data()));

    // "buffer" destructor releases the lock

    return bitmap;
}
```

This avoids the intermediate `Buffer`, which is definitely a good thing for large bitmaps, so we're down to just two copies of the pixels, one in the original `wicBitmap` and one in the converted `SoftwareBitmap`.

However, there is still a lot of hassle with having to parse out the original bitmap format and convert it to a Windows Runtime bitmap format.

If we would rather coerce the bitmap to a specific format, we can hard-code that into the function and avoid having to do format sniffing:

```
winrt::SoftwareBitmap ToSoftwareBitmap(IWICBitmapSource* wicBitmap)
{
    // Coerce into premultiplied ARGB
    winrt::com_ptr<IWICBitmapSource> prgbaBitmap;
    winrt::check_hresult(WICConvertBitmapSource(
        GUID_WICPixelFormat32bppPRGBA,
        wicBitmap,
        prgbaBitmap.put()));

    // Create a SoftwareBitmap of the same size and format.
    UINT width, height;
    winrt::check_hresult(prgbaBitmap->GetSize(&width, &height));
    // Avoid zero-sized or oversized bitmaps (integer overflow)
    if (width == 0 || height == 0 ||
        width > ~0U / 4 / height) {
        throw winrt::hresult_error(
            WINCODEC_ERR_IMAGESIZEOUTOFRANGE);
    }
    winrt::SoftwareBitmap bitmap(
        winrt::BitmapPixelFormat::Rgba8, width, height
        winrt::BitmapAlphaMode::Premultiplied);

    // Copy the pixels into the SoftwareBitmap.
    auto buffer = bitmap.LockBuffer(winrt::BitmapBufferAccessMode::Write);
    if (buffer.GetPlaneCount() != 1) {
        throw winrt::hresult_error(
            WINCODEC_ERR_UNSUPPORTEDPIXELFORMAT);
    }
    auto planeDescription = buffer.GetPlaneDescription(0);
    auto reference = buffer.CreateReference();
    winrt::check_hresult(prgbaBitmap->CopyPixels(
        nullptr, planeDescription.Stride,
        reference.Capacity(), reference.data()));

    // "buffer" destructor releases the lock

    return bitmap;
}
```

Copying the pixels directly from the `IWICBitmapSource` to the `SoftwareBitmap` removes an intermediate copy.

This is great, but it turns out we can do even better. We'll look some more next time.