How can I convert a WIC bitmap to a Windows Runtime SoftwareBitmap? part 2: Via a buffer

devblogs.microsoft.com/oldnewthing/20230412-00

April 12, 2023

Raymond Chen

Last time, we converted a WIC bitmap to a Windows Runtime SoftwareBitmap by encoding the WIC bitmap to a stream, then decoding it back into a SoftwareBitmap. But you don't have to pass the pixels through a stream. The `SoftwareBitmap` lets you pass the pixels directly in the form of an `IBuffer`.

```
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 buffer that can hold the pixels.
    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);
    }
    auto size = width * height * it->bytesPerPixel;
    winrt::Buffer buffer(size);
    buffer.Length(size);

    // Copy the pixels into the buffer.
    winrt::check_hresult(wicBitmap->CopyPixels(
        nullptr, width * it->bytesPerPixel, size,
        buffer.data()));
    winrt::SoftwareBitmap softwareBitmap(it->pixelFormat,
        width, height, it->alphaMode);

    // Create a SoftwareBitmap from the buffer.
    return winrt::SoftwareBitmap::CreateCopyFromBuffer(
```

```
        buffer, pixelFormat, width, height, alphaMode);
}
```

We no longer use the Windows Runtime `BitmapDecoder`, which means that we can do all of our work synchronously and return a `SoftwareBitmap`.

The idea here is that we peek at the `IWICBitmap` to see what its pixel format is, copy the pixels to a buffer, and then create a `SoftwareBitmap` of a matching format from that buffer. Unfortunately, there doesn't appear to be an easy way to convert between WIC pixel formats and Windows Runtime pixel formats, so we had to create a huge lookup table.

If you would rather force the `SoftwareBitmap` into a specific pixel format, then you can get rid of the format-sniffing code and instead use `WICConvertBitmapSource` to convert the `wicBitmap` to a matching source, and then just hard-code all the pixel format nonsense.

It turns out that even this is working too hard. The pixels for the bitmap get copied twice: Once from the `wicBitmap` to the buffer, and then again from the buffer to the final `SoftwareBitmap`. Next time, we'll reduce it to just one copy.