How can I convert a WIC bitmap to a Windows Runtime SoftwareBitmap? part 4: Handing it over

devblogs.microsoft.com/oldnewthing/20230414-00

April 14, 2023

Raymond Chen

Last time, we converted a WIC bitmap to a Windows Runtime SoftwareBitmap by copying the pixels of the WIC bitmap directly into the SoftwareBitmap. But you don't even have to copy pixels at all!

The `ISoftwareBitmapNativeFactory::CreateFromWICBitmap` method gives you a direct transfer of a `IWICBitmap` into a `SoftwareBitmap`.

```
#include <windows.graphics.imaging.interop.h>

winrt::SoftwareBitmap ToSoftwareBitmap(IWICBitmap* wicBitmap)
{
    winrt::SoftwareBitmap bitmap{ nullptr };

    auto native = winrt::create_instance<
        ISoftwareBitmapNativeFactory>(
        CLSID_SoftwareBitmapNativeFactory);

    winrt::check_hresult(native->CreateFromWICBitmap(
        wicBitmap, true, winrt::guid_of<winrt::SoftwareBitmap>(),
        winrt::put_abi(bitmap)));

    return bitmap;
}
```

First, we create a null `SoftwareBitmap` that will hold the result.

Next, we ask for the `ISoftwareBitmapNativeFactory` interface from a `SoftwareBitmap-NativeFactory`.

Finally, we call the `CreateFromWICBitmap` method to transmogrify the `wicBitmap` into a `SoftwareBitmap`, saying that the resulting bitmap is read-only (`true`).

And then we return the resulting bitmap.

The `SoftwareBitmap` doesn't make a copy of the `IWICBitmap`. It just copies the reference. As a result, no pixels are copied at all!

Since the `SoftwareBitmap` has a reference to the original `IWICBitmap`, you have two usage patterns.

If you are "giving away" the `IWICBitmap`, then you can pass `forceReadOnly = false` to make the resulting `SoftwareBitmap` read-write. The `SoftwareBitmap` now owns the `IWICBitmap` and may choose to modify it.

If you are sharing the `IWICBitmap`, then pass `forceReadOnly = true` to make the resulting `SoftwareBitmap` read-only. That way, the `SoftwareBitmap` won't make changes to the `IWICBitmap`.

If your `IWICBitmap` is a no-cache bitmap created from a `IWICBitmapSource`, then you need to pass `forceReadOnly = true` because the pixels are being generated on the fly and there is no buffer to modify.

I wrote out the above sequence in multiple steps, but you can collapse it into a one-liner:

```
winrt::SoftwareBitmap ToSoftwareBitmap(IWICBitmap* wicBitmap)
{
    return winrt::capture(
        winrt::create_instance
            <ISoftwareBitmapNativeFactory>
            (CLSID_SoftwareBitmapNativeFactory),
        &ISoftwareBitmapNativeFactory::CreateFromWICBitmap,
        wicBitmap, false);
}
```

Note that this call will fail if the `IWICBitmap` is in a format not supported by `SoftwareBitmap`.

Bonus chatter: A `SoftwareBitmap` can have an `IWICBitmap` inside it, or it can have a `IMF2DBuffer` inside (for video formats like NV12). If you have a `SoftwareBitmap`, you can reach inside and access the inner bitmap buffer by using `ISoftwareBitmapNative::GetData`.