BC6HBC7EncoderCS

<https://github.com/walbourn/directx-sdk-samples>

This is the DirectX SDK's Direct3D 11 sample updated to use the Windows 10 SDK without any dependencies on legacy DirectX SDK content. This sample is a Win32 desktop DirectX 11.0 application for Windows 10, Windows 8.1, Windows 8, and Windows 7.

**This is based on the legacy DirectX SDK (June 2010) Win32 desktop sample. This is not intended for use with Windows Store apps, Windows RT, or universal Windows apps.**

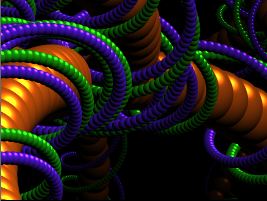
# Description

This sample implements BC6H/BC7 encoding using DirectCompute 4.0 acceleration. BC6H and BC7 are two new block-compressed texture formats introduced in Direct3D 11 that target much better visual quality than previous BC texture formats. On systems that have a graphics card with Compute Shader 4.0 support, this sample utilizes DirectCompute 4.0 to accelerate encoding.

Note that this sample utilizes DirectCompute 4.0 to accelerate BC6H/BC7 encoding, because DirectCompute 4.0 is available on most Direct3D 10 devices available today (proper driver update necessary), you can encode images to BC6H/BC7 format very fast on a system with such a D3D10+ card installed. However, to be able to sample from BC6H/BC7 textures directly in a shader, a Direct3D 11 display card is required. In other words, you can produce BC6H/BC7 textures fast on a D3D10 CS40 capable system using this sample utility, but your application still needs to run on a real D3D11 graphics card.

With the full source available, you can easily integrate this utility into your content creation pipeline to make your title benefit from these new BC formats.

Note that this functionality has been integrated into the [*DirectXTex*](http://go.microsoft.com/fwlink/?LinkId=248926) library.



# Command-line usage

BC6HBC7EncoderCS.exe (options) (filter) Filename0 Filename1 Filename2...

* /bc6hs: Encode to BC6H\_SF16 and save the encoded texture.
* /bc6hu: Encode to BC6H\_UF16 and save the encoded texture.
* /bc7: Encode to BC7 and save the encoded texture.
* /nomips: Disables the automatic generation or use of mipmaps
* /srgb: causes the /bc7 case to use BC7\_UNORM\_SRGB rather than BC7\_UNORM in the output file
* /aw <float>: Sets the weight for BC7 encoding of alpha. For images with alpha, you can increase this past 1.0 to give the alpha quality more weight than the color information. For images without alpha, you can set this to 0 to increase the color information encoding quality particularly if the alpha channel is noisy.
* /point, /linear, /cubic, /fant, /point\_dither, /cubic\_dither, /fant\_dither: Controls the DirectXTex filter settings when generating mipmaps

**Note:** mipmapping in this tool is only supported for power-of-2 textures. All source images must be multiples of 4 in width and height.

The input filename can be any WIC-supported bitmap format (.BMP, .PNG, HD Photo, etc.), a .DDS file, or a .TGA file.

The output filename is generated from the input filename as a .DDS file with \_BC6 or \_BC7 added.

# More Information

[Texture Block Compression in Direct3D 11](https://walbourn.github.io/direct3d-11-textures-and-block-compression/)

[BC6HBC7EncoderDecoder Sample Update](https://walbourn.github.io/bc6hbc7encoderdecoder-sample-update/)

[Where is the DirectX SDK (2021 Edition)?](https://aka.ms/dxsdk)

[Games for Windows and DirectX SDK blog](https://walbourn.github.io/)