Contents

[SurfaceLevel 2.0 1](#_Toc174207145)

[Overview 1](#_Toc174207146)

[Control Flow 1](#_Toc174207147)

# SurfaceLevel 2.0

## Overview

SurfaceLevel 2.0 is a texture converter and basic image editor. It loads many image file formats, converts to and from almost every video-game API texture (OpenGL, Vulkan, Metal, and Direct3D 12), offers a fast and high-quality resampler, and saves to many file formats. It is useful for performing nearly all video-game–related texture operations, such as compression, pre-multiplying alpha, generating normal maps, resampling/generating crisp mipmaps, etc., and for general-purpose image resampling, colorspace conversion, format conversion, etc. SurfaceLevel 2.0 is meant to gather a large quantity of features together and to provide more options for those features than you will find in any other tool. For example, when resampling, it is possible to specify different samplers for your image width, height, and depth, and different samplers for the RGB colors and the alpha channel—the sharpest resamplers often create ringing, which is particularly bad for the alpha channel; here you can using a ringing filter for the colors but a non-ringing filter for alpha (while specifying the same or different filters for both the color and alpha widths, heights, and depths).

SurfaceLevel 2.0 seeks to be useful in the level of detail for each feature, the number of features, and the performance of each feature. Supports volume (3-D) textures, texture arrays, cube maps, and mipmaps.

## Control Flow

SurfaceLevel 2.0 doesn’t try to be overly smart, so as long as you understand the basic internal workflow it should be easy to predict what it will do in any edge cases. Internally it performs the following operations in order:

1. Loads the image file.
2. Performs a conversion to the desired target format.
   1. Converts to RGBA64F.
   2. Converts to linear, applying any applicable colorspace conversions and gamma corrections necessary.
   3. Applies requested transforms, such as swapping, swizzling, flipping, pre-multiplying, etc.
   4. Generates mipmaps.
   5. Converts to the requested texture format, applying gamma-correction as necessary, etc.
3. Saves to the desired file.
   1. If the current format is not directly supported by the file, it is converted to the nearest format that is supported by the file.

The ways it tries to be smart:

1. By default, it will try to ensure only 1 gamma transform occurs from source to linear and from linear to destination. If images do not contain gamma data, they are assumed to be sRGB (-g, -gamma, -srgb, -rgbe, and -linear to change this assumption). If they contain gamma data, whether implicit by the format (VK\_FORMAT\_B8G8R8\_SRGB, for example) or from an embedded ICC profile, that is used instead. However, if you manually supply a source gamma value with -g, -gamma, -srgb, -rgbe, or -linear, your supplied gamma will be stacked with any contained/embedded gamma data. This can allow you to correct images that may have been saved with incorrect gamma. To specify your own gamma curve to be used *in-place* of any embedded or selected colorspace profiles, use -g, -gamma, -srgb, -rgbe, or -linear to define your own gamma curve and -ignore\_input\_colorspace\_gamma to ignore the gamma curve in any colorspace profiles being used.
2. To simplify the process, it is not necessary to specify the export format. If your conversion format is not supported, it will find the closest match that is supported. It will try to ensure a lossless conversion, but the option to specify the export format is always available if needed.