Technical Brief & User Guide

NVIDIA Texturing Scripts for Adobe Photoshop

Javascript and Photoshop

Since version 7.0, Adobe Photoshop has been scriptable. The scripting can use either Applescript® on the Macintosh®, Visual BASIC on Windows®, or Javascript® on either platform. NVIDIA® scripting tools are written in Javascript for maximum portability and usefulness.

Photoshop® scripts are much like Photoshop Actions, but they can alter their behavior on the fly— they are essentially *smart actions*.

The standard location for Javascript tools in Photoshop is in the \Presets\Script\ sub-directory within your Photoshop installation directory (in Applications on Mac, or Program Files on Windows). If scripts are placed there and Photoshop is started/restarted, those scripts display in Photoshop's File➢Scripts... menu. If the scripts are in any other disk location, they can be accessed using the File➢Scripts➢Browse... function.

For more specific information on the power of Photoshop scripting, refer to the "Scripting Guide" folder in Photoshop's installation folder, and Adobe's web site: http://partners.Adobe.com/public/developer/photoshop/devcenter.html

Mipster

Mipster.js is a mip-map generation tool that leverages the power of Photoshop CS2's own image filtering engine along with NVIDIA's DDS exporter and Normal-map-creation plugin filter (for Windows). While Mipster will function without either of these NVIDIA plugins (e.g., on Macintosh), it works best in tandem with them. They can both be obtained at http://developer.nvidia.com/

What is a mip-map?

When a texture image is mip-mapped, it contains multiple copies of the *root* image—each smaller image, or *mip-map*, represents the texture as it should appear at each of those smaller sizes. Figure 1 shows a simple example:

Figure 1. MIP-MAP Example

The *root* image is a 256x256 painting of the skin textures from the NVIDIA Dusk demo. The smaller versions of this image are mip-maps for smaller *harmonic* sizes: 128x128, 64x64, 32x32… etc down to a single 1x1 texel.

Providing these pre-sized versions of the root texture allows game engines texture with high quality and high speed. By default, the mip-maps (or mips) are created by simply resizing the image repeatedly. This process is often invisible to the texture artist. While this is adequate in many cases, often small or important details are lost in the translation from large to small (important details like eyes may be lost, or text rendered unreadable by naïve resizing). Using Mipster lets you see the mip results as you work, and makes it easy to alter, sharpen, or re-paint any arbitrary mip level.

Mipster is a tool meant to enhance Photoshop workflow. Unlike other tools, Mipster uses standard Photoshop methods to achieve its results, including the creation of Photoshop layers for each different mip, so that subsequent editing can be done easily and safely within Photoshop. Mipster is designed to let you see and control what you get, visually.

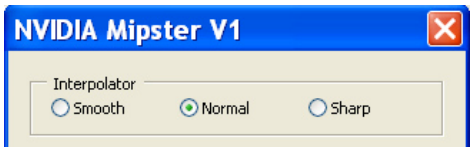
Running Mipster

Mipster runs on any open square texture, and non-destructively creates a mipped copy of that texture. If you have installed Mipster.js in the standard location (inside Photoshop's Presets\Scripts folder), then "Mipster" appears under the File➢Scripts... menu. Just select it while your source image is open, and Mipster begins by presenting the dialogue box shown here. The dialog box has three major sections.

Interpolator

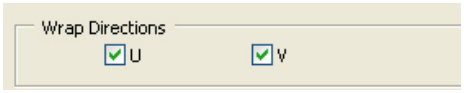
The Interpolator frame provides three options, which correspond exactly to Photoshop's own resize filters: **Bicubic_Smoother**, **Bicubic**, and **Bicubic_Sharper**. Photoshop will use the selected filter when creating each mip level.

Mipster always resizes from the original-sized image for each mip. For example, it will resize the 256x256 image to 8x8 to create an 8x8 mip, rather than progressively resizing downward from the next-highest mip (in this case 16x16). This preserves useful information from the root image for each mip level.



Wrap Directions

By default, Mipster assumes that this texture is intended to wrap, or repeat, along both the **U** and **V** directions in the texture (also known as **X** and **Y** directions). Mipster adjusts its down-sampling accordingly, so that repeat-edge transitions are smooth. If the texture is not intended to repeat in either direction, un-check the appropriate box to get the best results and also avoid pollution along edges from the non-repeating opposite side of the texture.

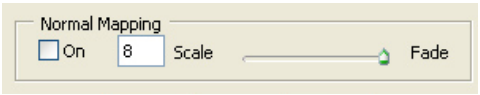


Normal Mapping

If you have installed the NVIDIA **NormalMapFilter** plug-in (visible under the Photoshop Filters menu as nvTools➢NormalMapFilter....), then you can use Mipster to operate this filter for all the mip levels in your texture. The Normal Map filter creates tangent-space normal maps from a grayscale bump map – details, along with the filter itself, can be found at http://developer.nvidia.com/object/photoshop_dds_plugins.html

When the checkbox is set, normal maps are generated for each individual mip level. Currently the default setting is to use 4-sample normal approximation (future versions of Mipster may present more choices).

The Scale factor matches the scale set in the NormalMapFilter plug-in. Note that higher Scale values show a more-pronounced normal-mapping effect.



The Fade slider ranges from 0 (no fade) to 1 (full fade). The slider controls how much to fade the Scale value for each successive mip level, so that bumps will not appear to get bigger as a 3D object moves further-away from the camera.

Figure 2. Grayscale bumpmap image and mip levels

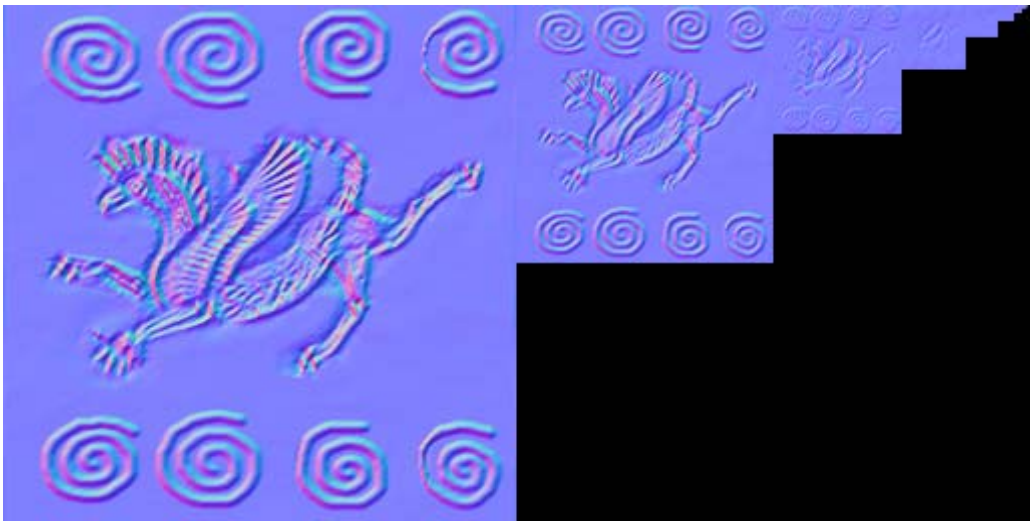


Figure 3. Normal map with mips
Note fading of lower mip levels

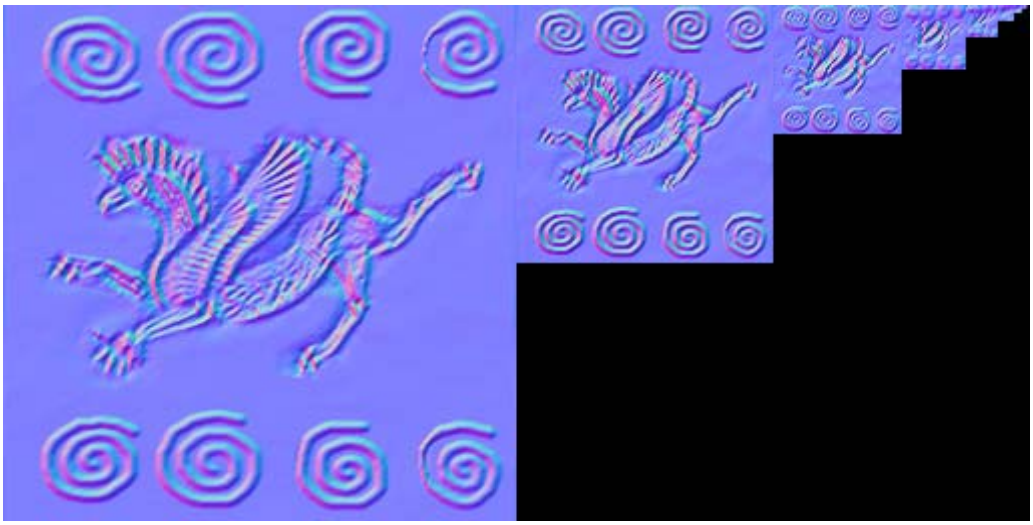


Figure 4. Normal map without fading between mips (bumps may appear too large/noisy on low levels)

When normal mapping is enabled, Mipster generates twice as many Photoshop layers. Each mip gets two different layers; one a mip of the normal map (which will be tagged as hidden) and one for the normal-map of that mip layer. If you want to make adjustments to the bump map of any particular layer (sharpening or blurring are the most-common), it's easy to re-apply the normal map filter by hand.

Why Not Just Make Normal Maps for the Root Image?

Normal map generation is a kind of image filtering, intended to look good in screen space for the final rendered 3D model. If we scale the filtered normal map, the various mip levels will have bogus normal filters (the bump maps will reduce to random noise at the lower sizes). Figure 5 shows a bad example and Figure 6 shows a correct example.

As you can see from the examples, the amount of bad bump fading looks similar to the good example, but the width of the bump filter reduced at each mip level – the result when rendered on a 3D model will be undesirable sparkling at small sizes and around contour edges. Thus you can see random stray values from small details at lower mip levels in the bad example, small details that have been properly handled in the correctly-faded image. This disparity will be even stronger if you use sharpening between mip levels (often advisable for normal maps).

1.

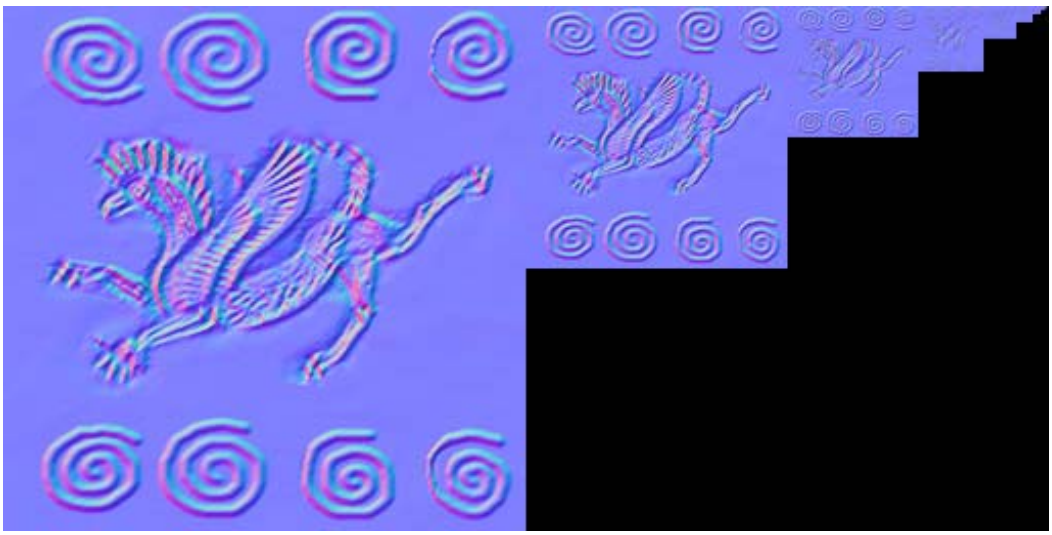


Figure 5. Wrong: Mip-mapping naively applied to a pre-normal-mapped image

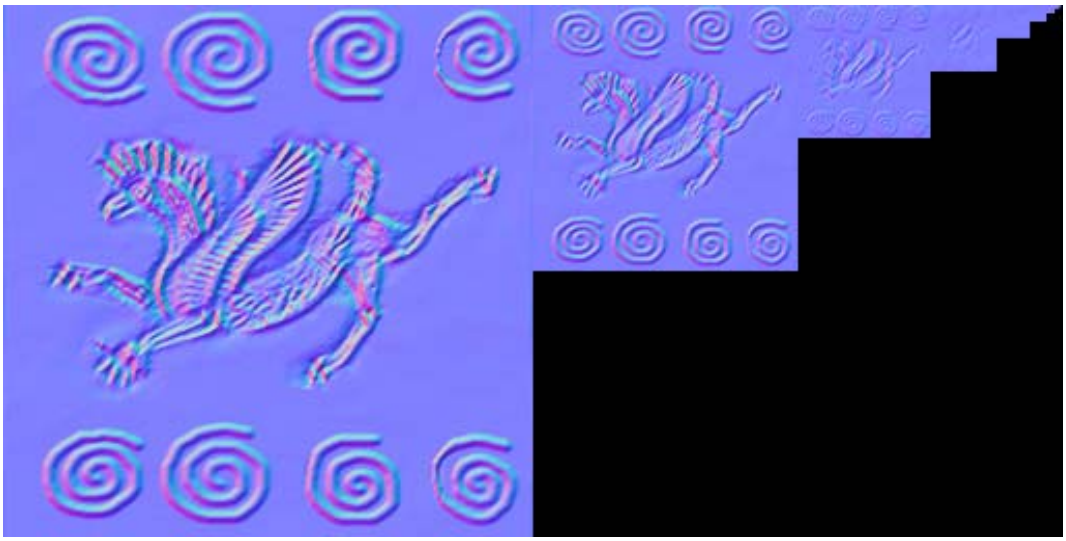
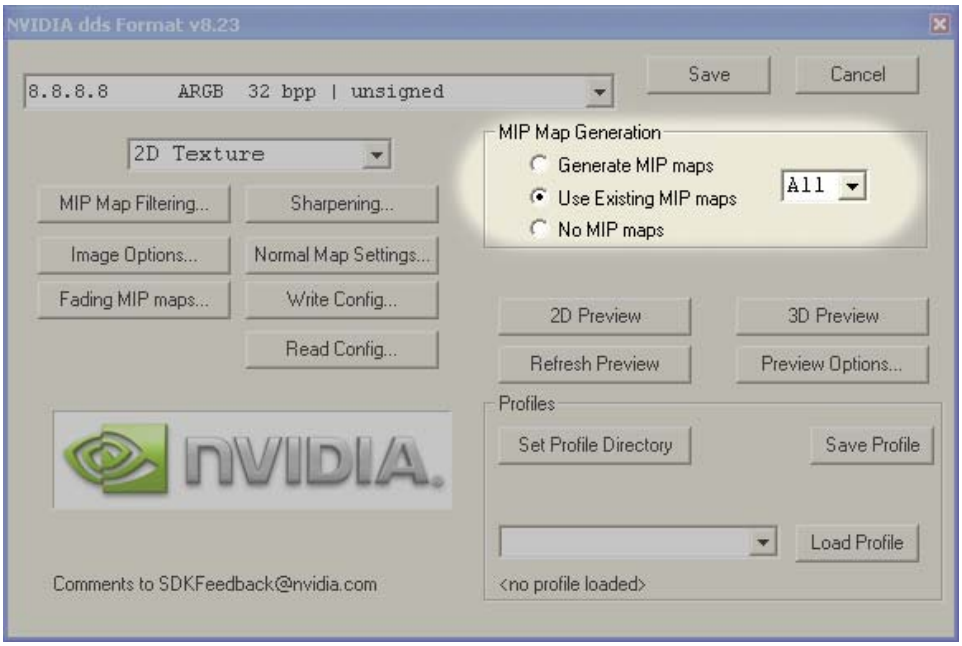


Figure 6. Right: Correctly-faded image based on grayscale bump map (same as Figure 3)

Exporting Your Mipster Results

Once you have generated mip maps using Mipster, save them as a Photoshop files (.psd). You will also need to export them as a .dds file for use in most 3D game engines. The NVIDIA DDS export plugin, also available at http://developer.nvidia.com/object/photoshop_dds_plugins.html, will handle the job perfectly. Follow these simple steps:

2. Save your mipped texture as a . psd to preserve all layering information.
3. Flatten all layers using the **`Flatten Image`** command (available either in the Layers menu or Layers-palette side menu)
4. Select File➢Save As...
5. In the file dialog, select DDS as the format (if DDS is not available, you may not have installed the plugin – check the URL mentioned above for the latest version)
6. Press the **Save** button, and the DDS compression dialog (below) will display. In the DDS dialog, be sure to check **Use Existing MIP maps.** Choose whatever compression scheme suits your application (generally we recommend DXT1).

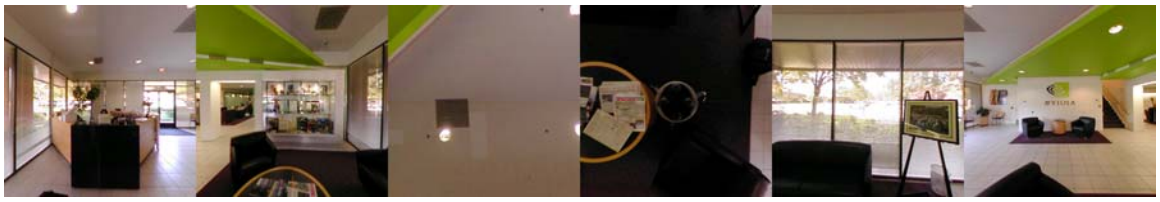


7. Press **Save** and you are done!

NVIDIA Cube Map Shuffler

The cube map shuffler script rearranges cube-map-face elements for use with different export and editing tools. It can quickly convert and copy between different cubemap layout formats, a task that has previously been annoyingly slow and error-prone. If you are used to editing cube map faces, you'll find this script makes format changes a lot easier.

The Shuffler's two format destination targets are shown in Figure 7:



6×1: The NVIDIA "default_reflection.dds" mapped horizontally

Inverse T (a.k.a. **Inverse Cross**): NVIDIA **default_reflection** remapped

Figure 7. Shuffler's Formats

The shuffler accepts as input formats cube maps in either of these formats and also 1×6 (vertical, created by some DDS tools) or 1×1 formats. For 1×1 textures, the original image will be replicated on all faces of the cube.

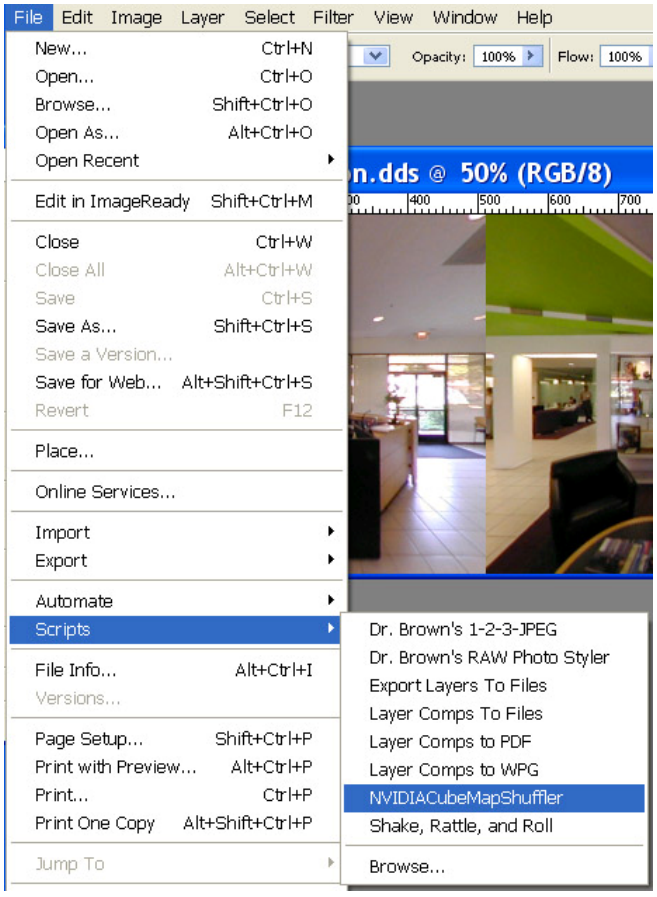
Images not in any recognized cubemap format will not be processed.

Using the Cube Map Shuffler

The cube map shuffler works on the currently-open document. Before you begin, has a cube map in one of the standard input formats, already open. Then run the script to create a new differently-formatted copy of that map.

The shuffler is non-destructive—your original document will not be changed.

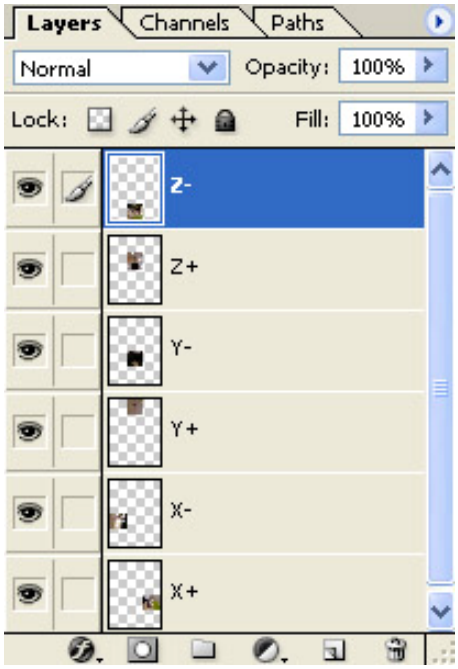
If you have installed **NVIDIACubeMapShuffler.js** in the standard location, you will find it in the Photoshop File➢Automate... menu (shown below). If not, use the Browse... option to find NVIDIACubeMapShuffler.js and open it. The script starts automatically.



The shuffler quickly creates a new copy of the existing document. The destination format will be either 6×1 format or inverse-t, depending upon the format of the input image.

The new copy will be in RGB mode, regardless of the mode of the input image. It also keeps the bit depth settings for 16-bit integer and 32-bit floating-point textures. 1-bit bitmaps will be promoted to 8-bit RGB. The new copy will also be assigned the metadata category cubemap.

Finally, the new copy will have the various cube faces split into individual layers, so that they are easier to manage using Photoshop tools like the paint brush. The new layers are labeled according to the direction they face:



You can use the shuffler as often as you like; ping-ponging between layouts. For final export, set the format to 6×1, flatten the layers and export using the NVIDIA DDS plugin.

Web Resources:

Adobe's Scripting-Guide web site:
http://partners.Adobe.com/public/developer/photoshop/devcenter.html

An explanation of normal mapping can be found at:
http://en.wikipedia.org/wiki/Normal_mapping

NVIDIA Corporation
2701 San Tomas Expressway
Santa Clara, CA 95050
www.nvidia.com