Info: Extended info on Bright 1.83.

CLOSE WINDOW

Info

BRIGHT 1.83 (c) 2000 by Erik de Neve, erik@epicgames.com

"Best palette Reduction for Industrial Grade High quality Textures"

- The BRIGHT manual -

Contents

Contents

Introduction

BRIGHT is a 32-bit windows-95 command-line utility written for single-sweep conversion of multiple textures to 8-bit palettized format using either unique, or a specifiable number of shared palettes, retaining the optimal color range; there is also a special option to provide a good color representation for mipmapped versions of the texture.

Distribution of a limited set of optimal palettes between similar textures is fully automatic.

The Unreal Editor recognizes textures with matching palettes when they are first imported, and makes sure only one copy of the palette exists in the UTX file. This can benefit hardware rendering somewhat since some 3D graphics hardware cards (notably VOODOO/3dfx) cannot cache multiple-palette data internally and need to have it uploaded every single frame.

It may take some experimenting, but in all cases BRIGHT should be able to provide the best possible custom palette, far better than the conversion algorithms in most graphics packages currently on the market. Results depend on color sampling depth (with which you can trade off quality for speed) and the number of unique palettes allowed (see options below) when repalettizing groups of textures together.

File formats and requirements

Input files are either 24-bit, or automatically promoted to be 24-bit when loading. Output files are in the (Unreal/Editor-readable) palettized PCX(default) or BMP formats, or the Targa format.

Bright reads the following file formats:

8/16/24/32 bit	TGA	Targa format

24-bit truecolor	BMP	Windows Bitmap format
8-bit palettized	BMP (*)	
8-bit palettized	PCX	ZSoft PCX format

(*) Warning: some programs write 8-bit BMP files using an archaic compressed format, for example Paintshop Pro; the solution is to either promote these to 24-bits before saving, or to save them as PCX or TGA files.

UNREAL SPECIFICS:

Note: Using shared palettes has become less of an urgent issue since the release of Unreal 1. It is still recommended for textures that obviously share the same hues and are used often, or each frame (HUD textures !)

When converting texture sets for Unreal, you'll want to use the shared-palette options (see below) to limit the number of unique palettes, which helps to speed up rendering on hardware 3d cards because on most hardware, palettes cannot be buffered in on-card memory like textures, and have to be uploaded every frame.

The recommended number of palettes is less than 1/4 the number of textures: which means, for a 40-texture set, try to see if using a 10 palette limit gives good results. If so, you may want to push it down to 8 or even 5 palettes per set, depending on the number and diversity of the textures.

Alternatively, this can be done automatically with the -error option (see below.)

Quick start examples

Once all 24-bit textures for a texture set are collected in a single directory, you can palettize them in one sweep.

To convert an entire directory of 24-bit BMP files to palettized PCXes, using N shared palettes for the whole texture set:

call BRIGHT with the following parameters:

BRIGHT *.bmp -auto N -o -8 -mips -mask 0.0

To convert a single image to an 8-bit image with it's own unique palette, do this:

BRIGHT naliqueen.bmp naliqueen.pcx

Note that for single-picture conversion, the output filename is required; for batch conversion, the second filename is optional and will be interpreted as either a destination directory or the first part of the output filename.
For example

BRIGHT *.bmp c:\destination\

will convert the .bmp files to .pcx files and put these in the c:\destination directory.

BRIGHT *.bmp NEW8BIT

converts the .bmp files to .pcx files with "NEW8BIT" prepended to all
the filenames.

Source art delivered in non-BMP 24-bit format (e.g. TIFs) can be converted
to 24-bit BMPs using standard software like Photoshop.

Masking and transparency tricks

To indicate holes in masked textures in Unreal, make sure to have your 24-bit source art use an uniform full-brightness pink, orange, or green color. When masking is enabled during palette quantization using the -mask X switch, any such areas will be detected and mapped exclusively to color 0 in the resulting palette.

Mask outlines may still need some manual adjustment in the final palettized output. The easiest way to do this is with an image editor that allows modification of palettized images, e.g. Paint Shop Pro (available as shareware on the web at www.jasc.com) or Adobe Photoshop. In PSP, type shift-P to see the palette. For making final adjustments to the mask, click on the 0th color entry of the palette and temporarily set it to a bright color; after editing, set that color back to black (0,0,0) before writing the final image to disk.

Most options map the special masking palette entry 0 to black because other colors might cause artefacts in bilinear interpolation on e.g. creature skins in the game; -pinkmask however makes sure that this color remains pink, useful for manual editing of the texture after palettization.

For whatever purpose, when you need too make sure there is an unused black color (palette entry 0) in the palette, you can use the -black options as explained in the command line parameters section below. Alternatively, just specify a smaller target number of colors.

Transparent textures usually have a black base color. This base black can be forced to palette color 0 similar to masked textures, which provides a speedup in Unreal software rendering (which then knows what texels to skip.) See also the -trans option described below.

Improving quality in the output images

Banding in pictures is a result of the limited number of colors available in a palette. Bright tries to design its palette smartly to minimize banding, but it has to work within the 256 (or smaller) color limit.

When the resulting picture seems to lose some color saturation, you can specify the -RGB option to use different interal scaling mechanism, which reproduces colors more accurately (though sometimes at the cost of slightly more banding.)

You may lose small patches of unique colors or gradations, specially if they take up relatively small amount of space in a large texture. To force the palette to include this color or do a better job for a certain hue, you can make a version of your texture where you clone this part many times, thereby forcing the palettization process to assign more importance to these colors. Two strategies can be used for this, either painting in unused areas of a texture, enlarging a texture and cutting out the result after palettization, or explicitly by making the final texture use the palette of the temporaty version with the -palette FILE command.

Command line parameters

The command syntax is:

BRIGHT [switches] file1 [file2] [switches]

General options:

-sizebias — with common-palette creation, this compensates for image size bias. By default smaller pictures are less important than bigger pictures when creating a common palette, because all individual pixels have the same weight regardless of what image they are in. When this switch is specified, the weight is distributed equally per-picture instead of per-pixel.

-mips — predict and take into account mipmap colors.

-8 — allow 8-bit .bmp image input as if it is 24-bit.

-colors N — target number of colors to quantize to, N = [1..256] default is 256.

-o, -overwrite — Overwrite existing files without warning

-bmp — Write output in bmp format. (default is pcx)

-tga — Write output in tga format. (default is pcx)

-common — Create a single common palette for all images, e.g. for animation sequences or small texture sets that look similar enough to need only one palette.

-dither N — Dither strength: 0 = no dither, 100 = max default is NO dither. This is only necessary for textures with smooth ramps, in cases where there are not enough palette colors allocated to smoothly represent those ranges. [Note: this is mostly broken. Bright is designed to make dithering unnecessary, since such checkerboard artefacts always look ugly in 3D engines.]

-bump X — Output a bumpmap for use with Unreal's 'IceTextures'. This simply maps the intensity gradient of the input image to a horizontal displacement value in the range (-128, 127) which is useful to create subtle glass-like refraction effects when used as the 'GlassTexture' in an IceTexture. X is the strength multiplier, with a useful range of about 1.0 to 8.0.

Key color masking & transparency stuff:

-pinkmask — Precise masking out of all pure pink (255,0,255) colors. The masked pixels will be mapped to palette color [0], which will remain pink in the output file. If used on multiple images, all output images will have color [0] set to pink regardless of the presence of pink masked

areas in the input images.

-graymask Precise masking out of all pure gray (128,128,128) colors. The masked pixels will be mapped to palette color [0], which will remain gray n the output file.

-colmask R G B Precise masking of pixels that have the specified color (R,G,B) The masked pixels will be mapped to palette color [0], which will get the specified color in the output file.

-pinkscreen Force pink colors to be masked out, with a default mask selection range tolerance (see below) of 0.7 If you do need a different tolerance you can combine this with the -mask X option.

-black Force absolute black included as palette color 0, independent of the presence of a possible other black color in the palette; useful for manual masking.

-mask X Automatic 'magic wand' masking, tolerance X (0.0..1.0) Similar to the magic wand selection tool in Fractal Painter, this will interpret as 'masked' a range of colors that is similar to the main masking-indication color. Will only take effect on images that have a clearly marked (pink or light-green) masked-out area. It is recommended to use only -mask 0.0 during batch conversion, with pre-cleaned masked textures (using -outmask)

-outmask [Single-file usage only:] Outputs a 24-bit bmp image with the mask cleaned up as specified by -mask and -fatmask, but filled in with the uniform masking color as specified in the input image. Very useful for automatic cleaning when it has ambiguous fuzzy pixels around mask edges; for example, masked textures made with Fractal Painter's floaters tend to have such edges.

-trans X [Single-file usage only:] Outputs a 24-bit bmp image with all near-black colors remapped to black. Useful with some transparent and other textures, to get rid of near-black pixels that would be indistinguishable but potentially slow down the game (Unreal-specific, software masking and translucency) Small values here are the most useful; 1.0 would map any image completely to black.

Color sampling options:

-ycc use Ycbcr color space (default) which samples colors somewhat less accurately than RGB (since there are conversion steps involved) but usually results in the best subjective image quality, especially for difficult pictures with wildly varying colors.

-rgb Use RGB colors space. Try this if you are converting single, limited- hue textures or small batches of images, and want to obtain the absolute highest quality. In most cases though, Ycc will provide the best subjective color mappings.

-depth NNN try sampling color bit depths, range: 5 to 8 (R,G,B or Y Cr Br) eg. -depth 888 gives the highest quality(default)

-mipmaps Take into account downscaled versions (mipmaps) when creating the palette. Example: a sharp black-and-white checkerboard pattern may need gray mipmaps to look good at a large distance, and thus needs gray palette entries.

-bins N force upper limit to # of distinct input colors; an internal measure of accuracy for the palette generation. Warning: large numbers greatly increase processing time.

Clustering palettes:

-auto N Automatic palette combining, allow a total of N unique palettes for the total batch set.

"-auto 1" is equivalent with "-common".

-predepth NNN Color depths for pre-sampling palette combining; this determines the accuracy at which temporary palettes are generated to determine which textures should share their palettes; does not affect the final color depth (which is determined by -depth)

Default setting: 655

-error X Automatic palette combining, with max allowed error X; this is a very subjective measure though; a useful value is about 200, which in moderately large texture sets (40 - 80 textures) causes the palettizer to generate about 1 palette for every 7 textures, on average.

While this option can be very useful since it provides an automatic limit to the error, it is usually easier to use the "-auto N" parameter, since adjusting the number of desired palettes directly is a bit more intuitive than fiddling with an error value.

Special palette stuff:

-palette FILE Force use of palette from bitmap FILE; if FILE is not an 8-bit palettized image, a suitable palette will be created from it on the fly.

-writepal FILE Outputs all palettes generated in as sample color-ramp pictures (numbers are prefixed to create unique names, if necessary.)

Presets: (not recommended)

-fast faster but uses less accurate sampling
-best try more accurate sampling (can be *slow*)
-perfect force losless RGB (limited-hue pictures only!)