SSX and Tricky texture lightmaps information

SSX and Tricky uses pre-calculated baked lightmaps for texture (then has dynamic lighting which only adjusts the rider as the rider moves under the light source)

In SSX and Tricky the texture lightmap uses a performance feature of the PS2, which is:

- (Cd - Cs) * As
- Cd is destination texture
- Cs is the lighting converted to a form to subtract from the texture
- As is the lighting converted to a form to multiply by the texture

The lightmaps are stored in the (map name)_L.ssh files. E.g alaska_L.ssh

The lightmaps are type 5 RGBA 32 bit bitmap images. Each lighting image in Tricky is 128x128 contains many lightmaps, which for curved patches are 8x8 (might be bigger for models). So for each Bezier curved patch there are UV lighting co-ordinate in the patch description x,y, w, h. Multiple these numbers by the lightmap bitmap width to get the area of the bitmap to use for the lightmap. E.g if

X = 0.125 y = 0.125 w = 0.0625 h = 0.0625

And lightmap width is 128, then we get:

- slice_x1 = 0.125*128 = 16
- slice_y1 = 0.125*128 = 16
- slice_x2 = slice_x1 + (w* image_width) = 16 + 0.0625*128 = 16 + 8 = 24
- slice_y2 = slice_y1 + (h* image_height) = 16 + 0.0625*128 = 16 + 8 = 24

So take an 8x8 lightmap from 16 to 24 in the x and y directions and then apply to the texture.

The lightmap bitmap contains 2 parts inside the same bitmap:

- An RGB lightmap
- An Alpha channel lightmap

To apply the lightmap to the texture:

- Rotate extracted lightmap 90 degrees counter clockwise (I don't know why, found I had to do this).
- Resize lightmap to the size of the texture (after texture has been mapped to surface, which means it might be flipped or tiled into multiple copies depending on the UV texture co-ordinates)
- Subtract the RGB lightmap part from the texture image and divide by 255 -> *subtracted normalised RGB image*
- Take the Alpha channel lightmap part and turn it into an RGB image with red, green and blue equal to the alpha value.
- Divide the RGB alpha image by 255 -> *normalised alpha image*
- Multiple the *subtracted normalised RGB image* by the *normalised alpha image*
- Multiple the result by 255 -> final lightmap adjusted texture

To apply this in Unity I assume a custom shader could be made. Or use this program to calculate a different pre-baked lightmap image to apply that fits a built-in shader.

This utility manually calculates the texture mapped to UV co-ordinates, applies the lightmap, then saves the result as the final texture lightmap adjusted for each Bezier curved patch, since Wavefront obj files do not support a separate pre-baked lightmap with separate UV co-ordinates.