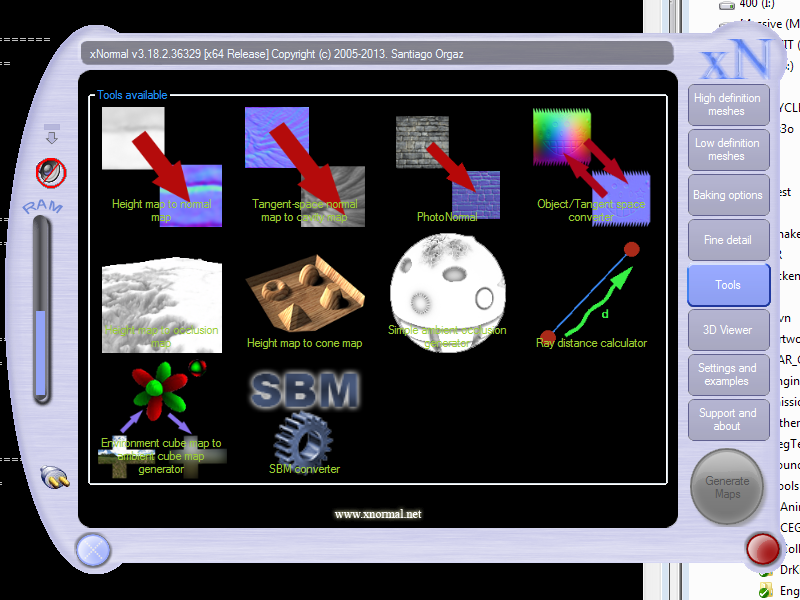
Per-vertex ambient occlusion generation tutorial/manual

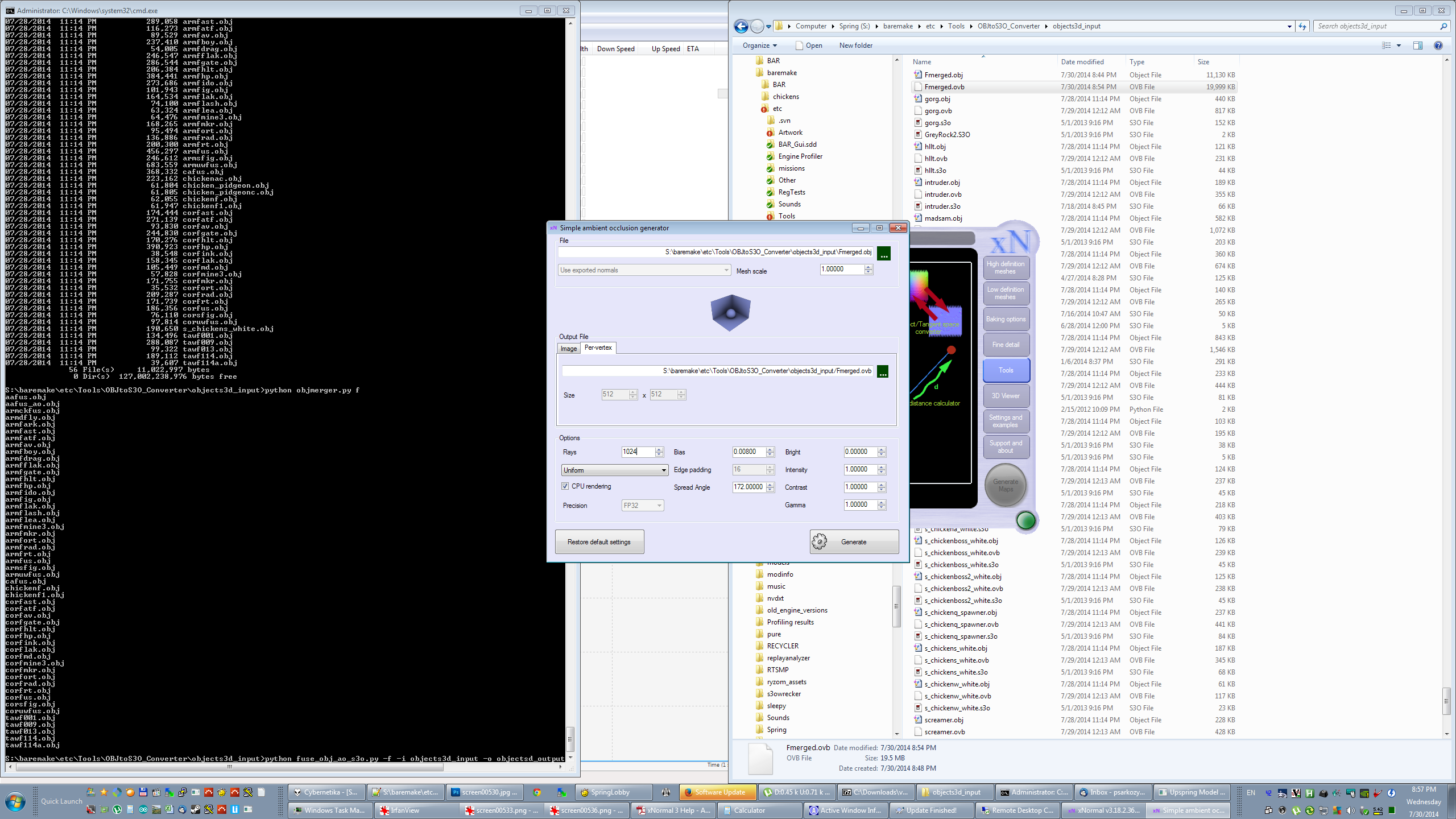
How it works:

S3O models are converted to .obj, and optionally, a ground plane is added beneath them, to simulate ao from the ground. These models are then loaded into xNormal’s Simple ambient occlusion tool:



which is pretty fast, and is thus driven by an autohotkey script (because who wants to click 10000 times, and because the ‘tools’ bits of xNormal cannot be scripted, unlike all the other parts).

Reasonably good settings for xNormal:



xNormal generates the per-vertex ambient occlusion data, and exports it to a .ovb format. The .ovb format is text based, and specifies the ambient occlusion of each vertex with a VCol=RGB row.

Next, the python script is run again, this time taking the original s3o models, and the .ovb files and encoding the ao term into the U channel of the UV coordinates. This is possible, because the U channel has been previously clamped to [0.0, 1.0], and thus we have 23 bits of precision available. Only 14 bits of precision are actually used by BAR (note that the largest texture is the arm one, and it is 4096 wide. With 14 bits of space to address it, we get 16K levels, so we can address it with up to quarter-texel precision). The 7 least significant bits are set to the AO term. These 7 least significant bits will range from [5,250] to mitigate any rounding errors, and the default value for it is 200 (this is useful for pieces that rotate, which is parsed from the unit’s script).

Once the models are copied back into the mod, only the custom vertex and fragment shaders need to be adjusted. A varying type parameter needs to be added to both, because the vertex shader will calculate from the U coordinate and normalize/clamp our AO value, and the fragment shader will interpolate the AO term over each fragment in each triangle.

Notes/requirements

-This only works on games that use customunitshaders for every unit. If a customunitshader is not used, then this method will have no effect (even though the AO term will be encoded into the model)

-Needs xNormal

-Needs Autohotkey

-Needs python

-Needs a willingness to do shit.

Steps:

1. Copy all of the .s3o files you wish to process to a custom directory (objects3d\_input).

Copy units and scripts dir too (./units/ and ./scripts/)

Note that if an s3o has a different filename than the unitdef that uses it, you can pass the –forceall option to fuse\_obj\_ao\_s3o.py option to force the AO baking for that unit with the assumption that it is not a building and cant fly.

1. Launch python script with: python fuse\_obj\_ao\_s3o.py --groundplate -i objects3d\_input -u units --scriptdir scripts
2. Open xNormal, click on tools, click on simple ambient occlusion, adjust AO settings as shown above in ‘reasonably good settings for xnormal’
3. Launch autohotkey script by double clicking it
4. Run python with : python fuse\_obj\_ao\_s3o.py -f -i objects3d\_input -o objects3d\_output --scriptdir scripts > fuselog3.txt

Optimal settings are the ones above

BAR units requiring piecewise baking are included in the fuse\_obj\_ao\_s3o.py script