Installation

Simply import the Unity package in your project. All the materials can be found in "**Turbulence Library/Materials"**.

Typical Use

If you are on Windows, use the materials found in "**Turbulence Library/Materials/CG Unlit**". On other platforms, use the GLSL variant of the materials in "**Turbulence Library/Materials/GLSL Unlit**".

Drop any material on a plane on the *xz* axis to see the noise material in action. If you want to enable the *Displace* option on the material, you should use a custom plane mesh as the standard Unity plane has a very small amount of vertices. Note that the base axis can be easily modified in the shader. See **Advanced Use** section for more information.

Each shader is completely open and can be modified for maximum customization. They can be found in the "**Turbulence Library/Shaders**" folder. Like materials, you should use the CG version on Windows and GLSL on other platforms.

If you need to sample the height field (grayscale values) of the material as a *Texture2D*, use the *GetHeightMap()* function found in the "**Turbulence Library/Scripts/NoiseManager.cs**" script. The script takes the material to be sampled (Material property) and the output texture's resolution (Width and Height properties) as input. The generated *Texture2D* will be mapped to the *NoiseTex* property. Note that this feature requires the use of a *RenderTexture*, only available with a Unity Pro license.

For a complete example scene containing all materials and their properties, open the "Workshop" scene in the "**Turbulence Library/Example**" folder.

Advanced Use

If you want to change the base axis of the noise functions, open the noise shader and go to the vertex and fragment programs ("vert" in CG and "#ifdef VERTEX" in GLSL). Find the fbm() function and change the height value and input position to the desired axis. For example, in CG, you could change "v.vertex.y = fBM(float3(v.vertex.xz, ..." to "v.vertex.z = fBM(float3(v.vertex.xy, ..." in the vertex program and "float h = fBM(float3(i.worldPos.xz, ..." to "float h = fBM(float3(i.worldPos.xy, ..." in the fragment program.

Some shaders can use different noise algorithms that are implemented but not applied by default. This is the case with the Perlin noise function, which uses the "Classic" implementation by default but can be switched to the "Improved" algorithm. Find the "Perlin3D" function in the relevant shaders and change the compilation conditionals to the desired values.

GLSL shaders cannot use octaves as an iterator, so you must specify the *OCTAVES* constant at the top of each GLSL shader. Default value is 12.

The *Displace* property enables vertex displacement in the material. Note that this does **not** update the mesh's vertices, so lighting and shading will not be applied correctly (the wireframe of the displaced mesh

will be the same as a plane). If you want to update the mesh's vertices, the basic approach would be to sample the height field with the *GetHeightMap()* method mentioned in **Typical Use** , read pixels (height data) on the texture and update the vertices with these values.

Note that each shader maps the height value to the *alpha* channel of the fragment output. The *r*, *g*, and *b* channels are used for coloring and texturing, which means you could add transparency to the materials.

Examples of the above scenarios will be provided soon.

Questions/comments/suggestions

Feel free to contact me at jeremie.stamand@gmail.com.