Using Noise with *glman*

The *glman* tool automatically creates a 3D noise texture and places it into Texture Unit **3**. Your shaders can access it through the pre-created uniform variable `Noise3`. You can reference it in your shader as

```
uniform sampler3D Noise3;
. . .
vec3 stp = uNoiseFreq * vMCposition;
vec4 nv  = texture( Noise3, stp );
```

The "noise vector" texture *nv* is a vec4 whose components have separate meanings. The .r component is the low frequency noise. The .g component is twice the frequency and half the amplitude of the .r component, and so on for the .b and .a components. Each component is centered around the middle value of .5

Component	Term	Term Range	Term Limits
0	**nv.r**	**0.5 ± .5000**	**0.0000 → 1.0000**
1	**nv.g**	**0.5 ± .2500**	**0.2500 → 0.7500**
2	**nv.b**	**0.5 ± .1250**	**0.3750 → 0.6250**
3	**nv.a**	**0.5 ± .0625**	**0.4375→ 0.5625**
	sum	**2.0 ± ~ 1.0**	**~1.0 → 3.0**
	sum – 1	**1.0 ± ~ 1.0**	**~0.0 → 2.0**
	(sum – 1) / 2	**0.5 ± ~ 0.5**	**~0.0 → 1.0**
	(sum – 2)	**0.0 ± ~ 1.0**	**~1.0 → 1.0**

So, if you would like to have a four-octave noise function that ranges from 0. to 1, then do this:

```
float  sum = nv.r + nv.g + nv.b + nv.a;    // range is 1. -> 3.
       sum = ( sum - 1. ) / 2.;            // range is now 0. -> 1.
```

If you would like to have a four-octave noise function that ranges from -1 to 1, then do this instead:

```
float sum = nv.r + nv.g + nv.b + nv.a;     // range is 1. -> 3.
      sum = ( sum - 2. );                  // range is now -1. -> 1.
```

By default, the *glman* 3D noise texture has dimensions 64 × 64 × 64. You can change this by putting a command in your GLIB file of the form

```
Noise3D 128
```

to get dimension 128 × 128 × 128, or choose whatever resolution you want (up to around 400 × 400 × 400). Remember that for the most general use, the resolution should be a power of two

The first time *glman* creates a 3D noise texture for you, it will take a few seconds. But, *glman* then writes it to a local file, and the next time this 3D texture is needed, it is read from the file, which is a lot faster.

A 2D noise texture works the same way, except you get at it with

```
    uniform sampler2D Noise2;
    ...
    vec2 st = uNoiseFreq * vST;
    vec4 nv = texture( Noise2, st );
```

The only difference is that a 2D noise texture is indexed by a **`vec2`** while the 3D noise texture is indexed by a **`vec3`**, but both return a **`vec4`**.