

GEOMORPH MACRO version 1.0

Geomorph_mcr.pov

**A POV-Ray utility to make isosurface landscapes with a touch of
geology and geomorphology**

By
Thomas A. M. de Groot
e-mail: t.degroot@inter.nl.net

Introduction

Isosurfaces are exciting features for simulating the surface of the Earth. While heightfields are easy to generate and generally fast to render, isosurfaces are much more complex, allowing – among other things – the generation of overhanging rocks. Isosurfaces are also much slower to render, but the complexity and the beauty of the results are a fair trade-off.

Generating landscapes with isosurfaces may prove a slow and arduous process, especially for the less mathematically skilled. It is for them that I started work on this macro so that all kinds of landscapes can be obtained with a minimum of efforts and knowledge. This does not mean of course, that the more mathematically skilled will not enjoy the macro.

The second reason why I started work on this macro, is because I wanted to be able to generate landscapes that included geological or geomorphological elements like folds and faults. I am not sure if I was entirely successful. Further work on the macro is needed.

Needless to say that this macro is one of the many possible variants that you can imagine. In that respect, consider the work of Jaime Vives Piqueres, Christoph Hormann, Zeger Knaepen, Nathan O'Brien, and many others.

Usage

Put the macro in the most convenient folder for you. Standard, that should be the include folder of POV-Ray.

The macro can be used both in stand alone mode, or called by another script. The stand alone mode is useful for testing different settings before using them in a particular scene. As the standalone switch is true by default, it has to be set when the macro is called from outside, before including the macro file:

```
#declare Standalone = false;  
#include "Geomorph_mcr.pov"
```

You can then declare the parameters, succinctly described in the next section. As all parameters have a default value, you may only have to change a few of them. The macro comes with an example settings series that you can run in stand alone mode.

Example:

```
#declare Sphered = false;  
#declare Cracksize = 0.5;  
#declare UpperBound = 1;  
#declare Xsize = 100;  
#declare Zsize = 100;
```

```
#declare Isoscale = <2, 0.8, 2>;
#declare Isotrans = <0, -0.5, 90>;
#declare MaxGrad = 15;
```

Then, you call the macro which generates an isosurface called Landscape:

```
Geomorph()
object {Landscape}
```

The object {Landscape} can, in turn, be used for placing objects like trees or houses. If you have several landscapes in your scene, you may want to declare it under another name before calling the macro again:

```
#declare MyLandscape = object {Landscape}
```

The macro elements

The Geomorph Macro contains 50 different parameters that you can change and combine individually and in all possible manners. All have a default value called by the macro when not declared from the outside. Although perhaps all combinations may not prove to be possible, and although some combinations have been fixed by me, the number of different possible landscapes is huge and may answer to all your wishes.

By choice, the basis for all the landscapes is the `f_hetero_mf` function. Its x-, y-, and z-settings are randomised and so indirectly controllable, but the H, lacunarity, octaves, offset, gain, and noise generator, are fixed in this version of the macro.

The parameters are grouped into the following elements:

Patterns:

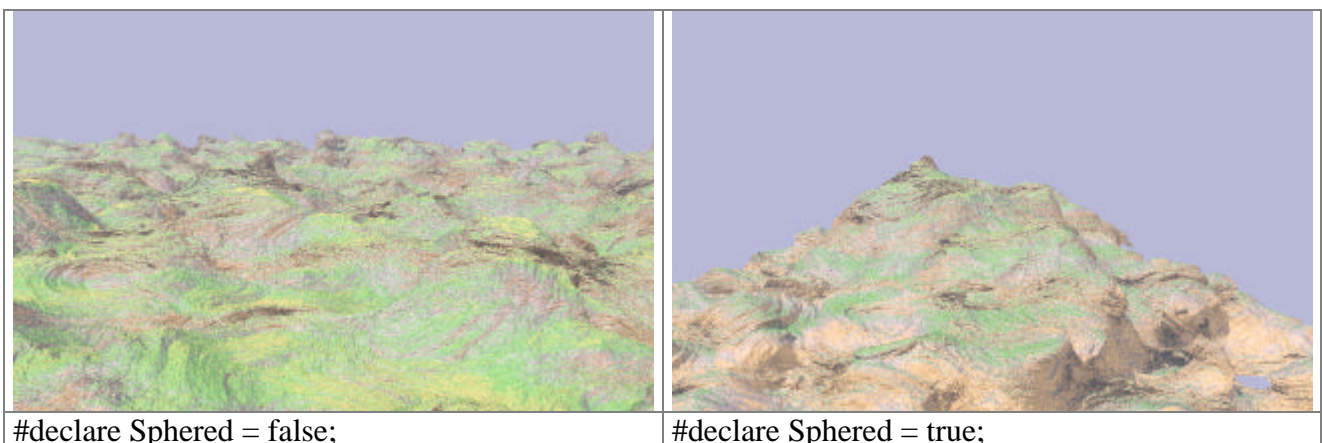
[parameters: Noised; Ridged; Wrinkled; Rippled; Waved; Dented; Sphered; Crackled]

Patterns groups the available pattern functions used by the macro. A Boolean switch turns them on or off. When turned off, corresponding parameters in other elements are not used.

In particular, Sphered is an important parameter. When true, it allows for the generation of an island or a single rock (see example).

Like the `f_hetero_mf` function, the `f_ridged_mf` function (`Ridged = true`) has fixed H, lacunarity, octaves, offset, gain, and noise generator values in this version of the macro.

Example: `#declare Wrinkled = true;`



Warp:

[parameters: Wrinkleswarp; Rippleswarp; Waveswarp; Dentswarp; Cracklewarp]

A Boolean switch allows you to use warp in the selected pattern functions. This switch does not exist for the noise, the ridge, and the sphere functions.

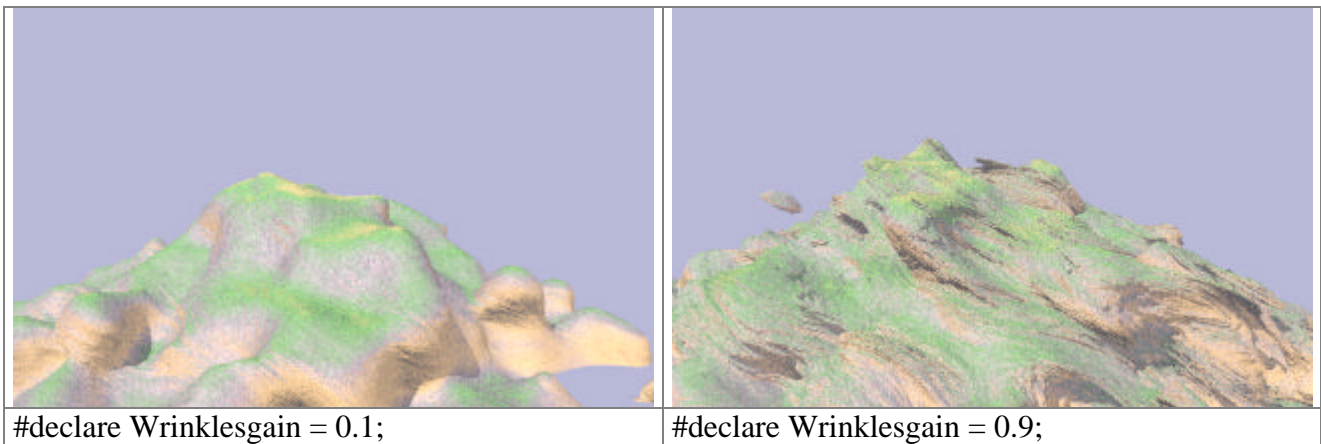
Example: #declare Wrinkleswarp = true;

Gain:

[parameters: Wrinklesgain; Ripplesgain; Wavesgain; Dentsgain; Spheregain; Cracklegain]

The force of each pattern function can be controlled by increasing or decreasing its gain. This is particularly useful if you want to avoid isosurface break-ups and floating pieces in space (see example).

Example: #declare Wrinklesgain = 0.5;



Warp turbulence:

[parameters: Wrinklesturb; Ripplesturb; Wavesturb; Dentsturb; Crackleturb]

If warp has been selected for a pattern function, the turbulence of the warp can be controlled by a vector.

Example: #declare Wrinklesturb = <0.5, 0.9, 0.9>;

Scale:

[parameters: Wrinklescale; Ripplescale; Wavescale; Dentscale; Cracklescale; Cracksize]

The selected pattern functions can be scaled by adding a scaling vector.

Example: #declare Wrinklescale = <1, 0.5, 1>;

Angle:

[parameters: Wrinklesangle; Ripplesangle; Wavesangle; Dentsangle; Crackleangle]

In the same way, the inclination of the pattern can be controlled. This is particularly useful to simulate inclined geological strata.

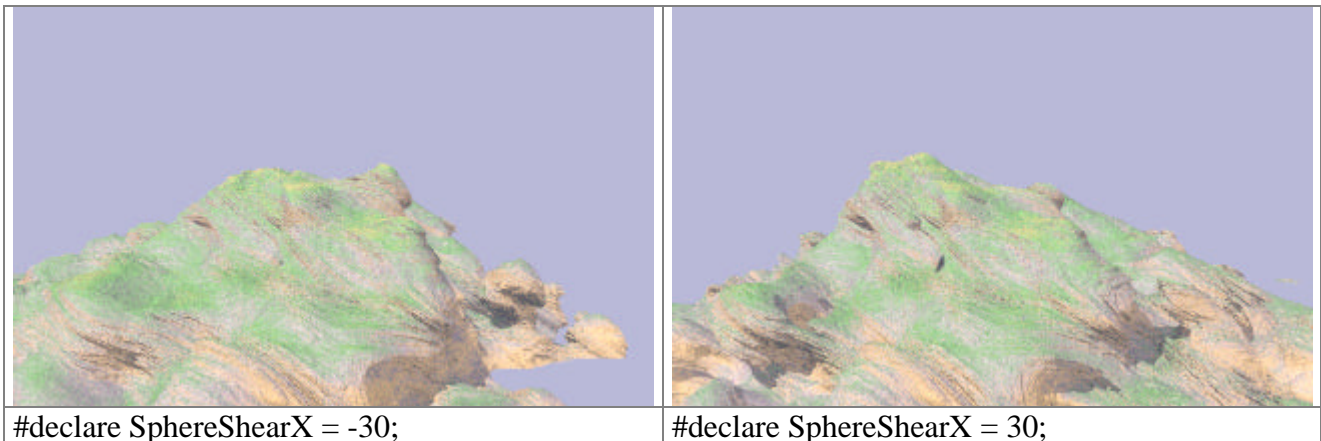
Example: #declare Wrinklesangle = <0, 0, -30>;

Sphere:

[parameters: SphereShearX; SphereShearZ; Diameter]

The sphere function that can be used to generate islands or single rocks, has a few additional parameters. It can be sheared in the x- or the z-direction, so that the rock or island looks asymmetrical (see example). Think of Gibraltar. Diameter, is obviously the diameter of the sphere.

Example: `#declare SphereShearX = -10;`



Noise and randomness:

[parameters: Noise; Stream]

With Noise, you have control over the `f_noise3d` function, if Noised has been declared true. Stream is an integer used by the random number generator.

Example: `#declare Noise = 0.3;`

Isosurface:

[parameters: UpperBound; LowerBound; MaxGrad; Xsize; Zsize; Isoscale; Isotrans]

The final isosurface can also be controlled. UpperBound and LowerBound determine the limits of the bounding box along the y-axis; Xsize and Zsize determine the limits of the bounding box along the x- and z-axis respectively. MaxGrad enables you to control the maximum gradient to be used. Start low when testing, as this renders faster, then adjust following the directives given to you in the message window. Isoscale and Isotrans can additionally be used to scale and/or translate the complete isosurface according to your own needs.

Example: `#declare UpperBound = 2;`

Texture:

[parameters: Landtex; Simpletex; Slope; SlopeValue]

You can, of course, use your own textures by declaring Landtex. Default, there is a pair of simple textures provided by Christoph Hormann (landscape.pov, June 2001). Simpletex is a boolean switch set to false by default (thus using the more interesting of the two); Slope is a boolean switch that enables you to use slope (and altitude) or a simple gradient texture; SlopeValue controls the ratio between slope and altitude. It should always be set between 0 and 1.

Example: `#declare Landtex = T_Stone1;`

Copyright

The macro and this documentation, Copyright Thomas de Groot, 2005.

This is freeware. You are free to use and change the macro or the documentation for your own needs. You are not allowed to sell them. Please, acknowledge its use in some way, especially if used for an entry in any competition, like IRTC.