

|  |                    |
|--|--------------------|
| <a href="#">Introduction.....</a>  | <a href="#">1</a>  |
| <a href="#">Usage.....</a>   | <a href="#">2</a>  |
| <a href="#">The macro elements.....</a>  | <a href="#">3</a>  |
| <a href="#">Element 1 - Substrate:.....</a>                                      | <a href="#">3</a>  |
| <a href="#">Element 2 - Noise:.....</a>  | <a href="#">4</a>  |
| <a href="#">Element 3 - Patterns:.....</a>                                       | <a href="#">5</a>  |
| <a href="#">Element 4 – Pattern Warp:.....</a>                                   | <a href="#">6</a>  |
| <a href="#">Element 5 – Pattern Gain:.....</a>                                   | <a href="#">6</a>  |
| <a href="#">Element 6 – Pattern Warp Turbulence:.....</a>                        | <a href="#">7</a>  |
| <a href="#">Element 7 – Substrate/Pattern Scale:.....</a>                        | <a href="#">7</a>  |
| <a href="#">Element 8 – Substrate/Pattern Angle:.....</a>                        | <a href="#">7</a>  |
| <a href="#">Element 9 – Fractal Translation:.....</a>                            | <a href="#">7</a>  |
| <a href="#">Element 10 - Island:.....</a>  | <a href="#">7</a>  |
| <a href="#">Element 11 - Isosurface:.....</a>                                    | <a href="#">8</a>  |
| <a href="#">Element 12 - Texture:.....</a>                                       | <a href="#">8</a>  |
| <a href="#">Update history.....</a>  | <a href="#">9</a>  |
| <a href="#">Copyright.....</a>   | <a href="#">9</a>  |
| <a href="#">Appendix 1 – Parameters and defaults for the two versions.....</a>   | <a href="#">10</a> |
| <a href="#">Appendix 2 – Parameter settings for Islands in the examples.....</a> | <a href="#">13</a> |
| <a href="#">Appendix 3 - The sea in the examples.....</a>                        | <a href="#">15</a> |

## 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 these macros 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 macros.

The second reason why I started work on these macros is that I wanted to be able to generate landscapes that included geological or geomorphological elements like folds and faults. Further work on the macros will continue.

Obviously, these macros are two of 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. And finally, the very comprehensive isosurface tutorial by Mike Williams – who is also at the origin of a very early stage of these macros – should be acknowledged.

Why two macros? Initially, there were some fundamental differences in the way the final isosurface function was calculated in version 2.1 or version 2.2. It rapidly appeared to me that both generated interesting landscapes of their own. I decided then to keep them as two main branches of the Geomorph tree. Defaults are identical in both versions (see Appendix 1) and, with the same parameter settings, you can switch from one to the other simply by calling a different macro. A few parameters are resident to one or the other macro exclusively. They will not be activated in the other one (see Appendix 1).

## Usage

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

These versions of the macros are backward compatible with the settings from version 1 to 2.0, but the results will not be identical as a few fundamental changes have been made. So, if you really love a landscape made with an older version, do not delete that version of the macro!

The macros 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 to false when the macros are called from outside, and before including the macro file. Example:

```
#declare Standalone = false;  
#include "Geomorph_mcr2B.pov"
```

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

Example:

```
#declare Isolated = 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 that generates an isosurface called Landscape:

---

Geomorph()  
object {Landscape}

The object {Landscape} can 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 Macros contain more than 80 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 (see Appendix 1). Although perhaps not all combinations may 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. You are advised to start with a simple landscape, with few parameters, and slowly increase the complexity. In many cases, a simple landscape will already appeal to you.

The parameters are grouped into the following elements:

### Element 1 - Substrate:

[parameters: HeteroInf; HetH; HetLac; HetOct; HetOff; HetT; HetNG; Noised; Ridged; RidgedInf; RidH; RidLac; RidOct; RidOff; RidG; RidNG; Fractal; Mandelbrot; Complex; Iterations; Exponent; Exterior; ExFactor; Interior; InFactor]

The substrate for all the landscapes is the `f_hetero_mf` function which can be used either with `y` scaled infinitely (`HeteroInf = true`) or not. `H`, `Lacunarity`, `Octaves`, `Offset`, `Gain`, and `Noise Generator`, can now be controlled in this version of the macro (`HetH`, `HetLac`, `HetOct`, `HetOff`, `HetT`, `HetNG`). From the POV-Ray documentation:

- **H** is the negative of the exponent of the basis noise frequencies used in building these functions (each frequency `f`'s amplitude is weighted by the factor `f - H`). In landscapes, and many natural forms, the amplitude of high frequency contributions are usually less than the lower frequencies. When `H` is 1, the fractalization is relatively smooth ("1/f noise"). As `H` nears 0, the high frequencies contribute equally with low frequencies as in "white noise".
- **Lacunarity** is the multiplier used to get from one 'octave' to the next. This parameter affects the size of the frequency gaps in the pattern. Make this greater than 1.0
- **Octaves** is the number of different frequencies added to the fractal. Each 'Octave' frequency is the previous one multiplied by 'Lacunarity', so that using a large number of octaves can get into very high frequencies very quickly.
- **Offset** is the 'base altitude' (sea level) used for the heterogeneous scaling
- **T** scales the 'heterogeneity' of the fractal. `T=0` gives 'straight 1/f' (no heterogeneous scaling). `T=1` suppresses higher frequencies at lower altitudes
- **Generator type** used to generate the noise3d. 0, 1, 2 and 3 are legal values.

This primary substrate can be varied with noise (`Noised = true`) and ridges (`Ridged = true`). Like the `f_hetero_mf` function, the `f_ridged_mf` function can be used with `y` scaled infinitely (`RidgedInf = true`) or not, and has `H`, `Lacunarity`, `Octaves`, `Offset`, `Gain`, and `Noise Generator` values that can be controlled in this version of the macro (`RidH`, `RidLac`, `RidOct`, `RidOff`, `RidG`, `RidNG`). From the POV-Ray documentation:

---

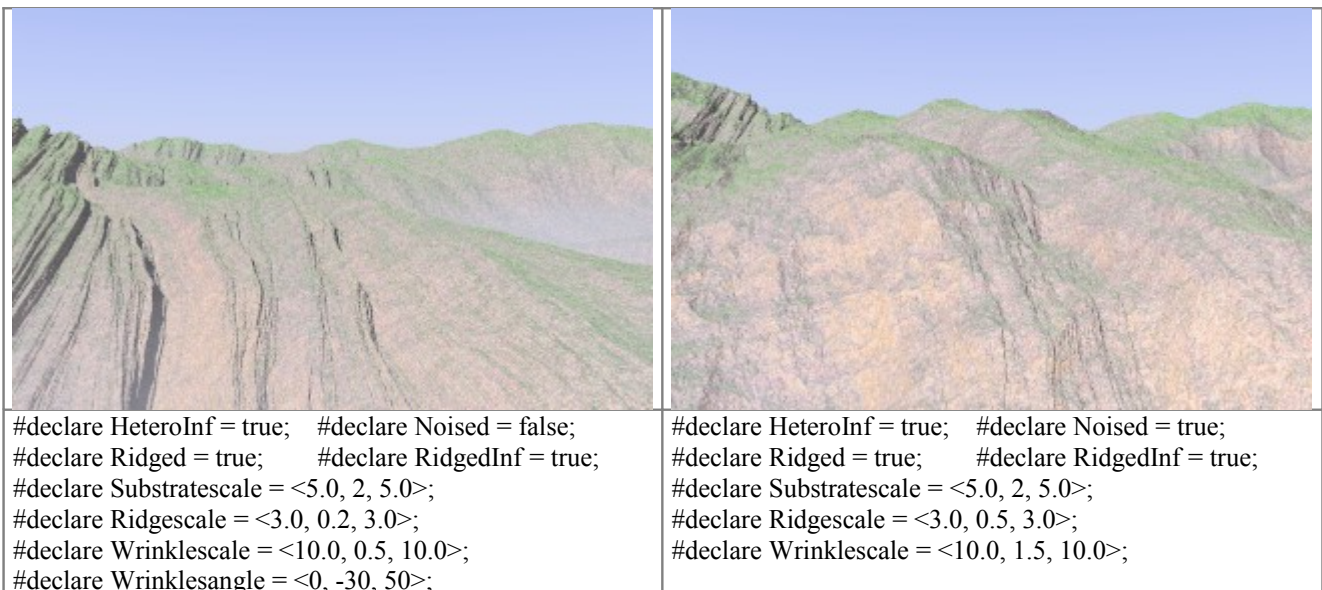
- **H** is the negative of the exponent of the basis noise frequencies used in building these functions (each frequency  $f$ 's amplitude is weighted by the factor  $fE^{-H}$ ). When  $H$  is 1, the fractalization is relatively smooth. As  $H$  nears 0, the high frequencies contribute equally with low frequencies
- **Lacunarity** is the multiplier used to get from one "octave" to the next in the "fractalization". This parameter affects the size of the frequency gaps in the pattern. (Use values greater than 1.0)
- **Octaves** is the number of different frequencies added to the fractal. Each octave frequency is the previous one multiplied by "Lacunarity". So, using a large number of octaves can get into very high frequencies very quickly
- **Offset** gives a fractal whose fractal dimension changes from altitude to altitude. The high frequencies at low altitudes are more damped than at higher altitudes, so that lower altitudes are smoother than higher areas
- **Gain** weights the successive contributions to the accumulated fractal result to make creases stick up as ridges
- **Generator type** used to generate the noise3d. 0, 1, 2 and 3 are legal values.

The substrate and the Ridged functions can be influenced (see Element 7 – Substrate/Pattern Scale). Finally, a fractal pigment function (Mandelbrot or Julia) can be added to this substrate.

By selecting the fractal function (Fractal = true) you may generate cliffs and table mountains. You are advised to read paragraph 3.5.11.14 of the POV-Ray documentation (v. 3.6) for explanation of the different parameters used. Be aware however, that using fractals may result in long renders because the max\_gradient of the isosurface may have to be dramatically increased. This may even be as high as several thousands according to the message window. However, this is not always necessary. Increase slowly until you feel satisfied that there are no apparent holes or artefacts. In addition, start with low iteration values for results that look more natural. However, you might need to increase them.

The Mandelbrot parameter is a Boolean switch for the mandelbrot fractal (Mandelbrot = true) or the Julia fractal (Mandelbrot = false).

Example: #declare Fractal = true;



## Element 2 - Noise:

[parameters: Noise]

With Noised = true, you control the f\_noise3d function with the Noise parameter.

Example: #declare Noise = 0.3;

### Element 3 - Patterns:

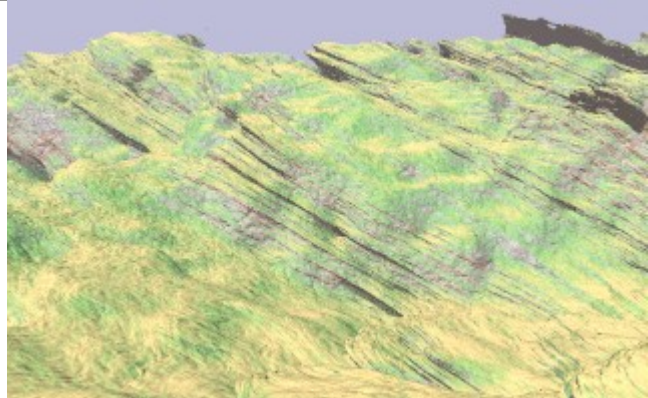
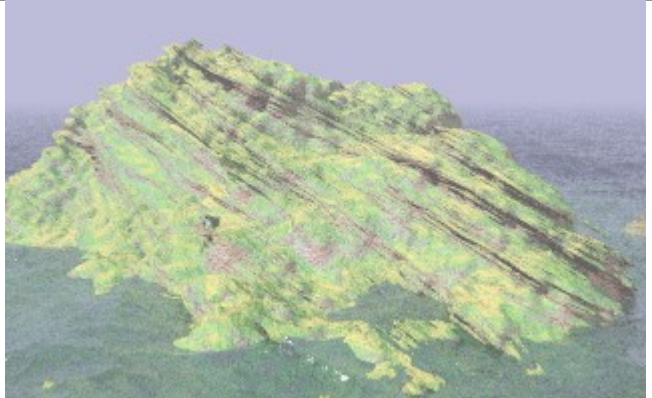
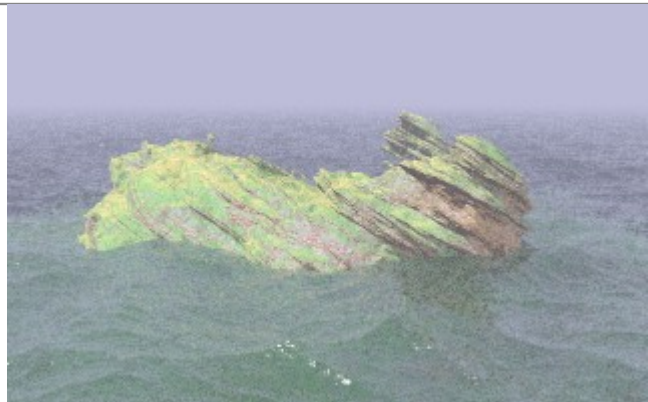
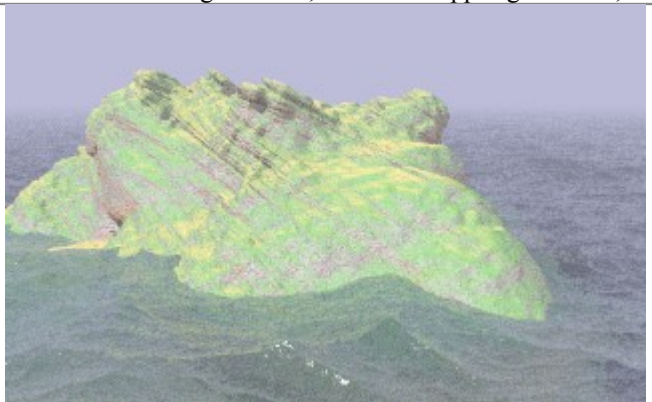
[parameters: Wrinkled; Rippled; Waved; Dented; Isolated; Isolator; Cracked; Hexed; Honeycomb]

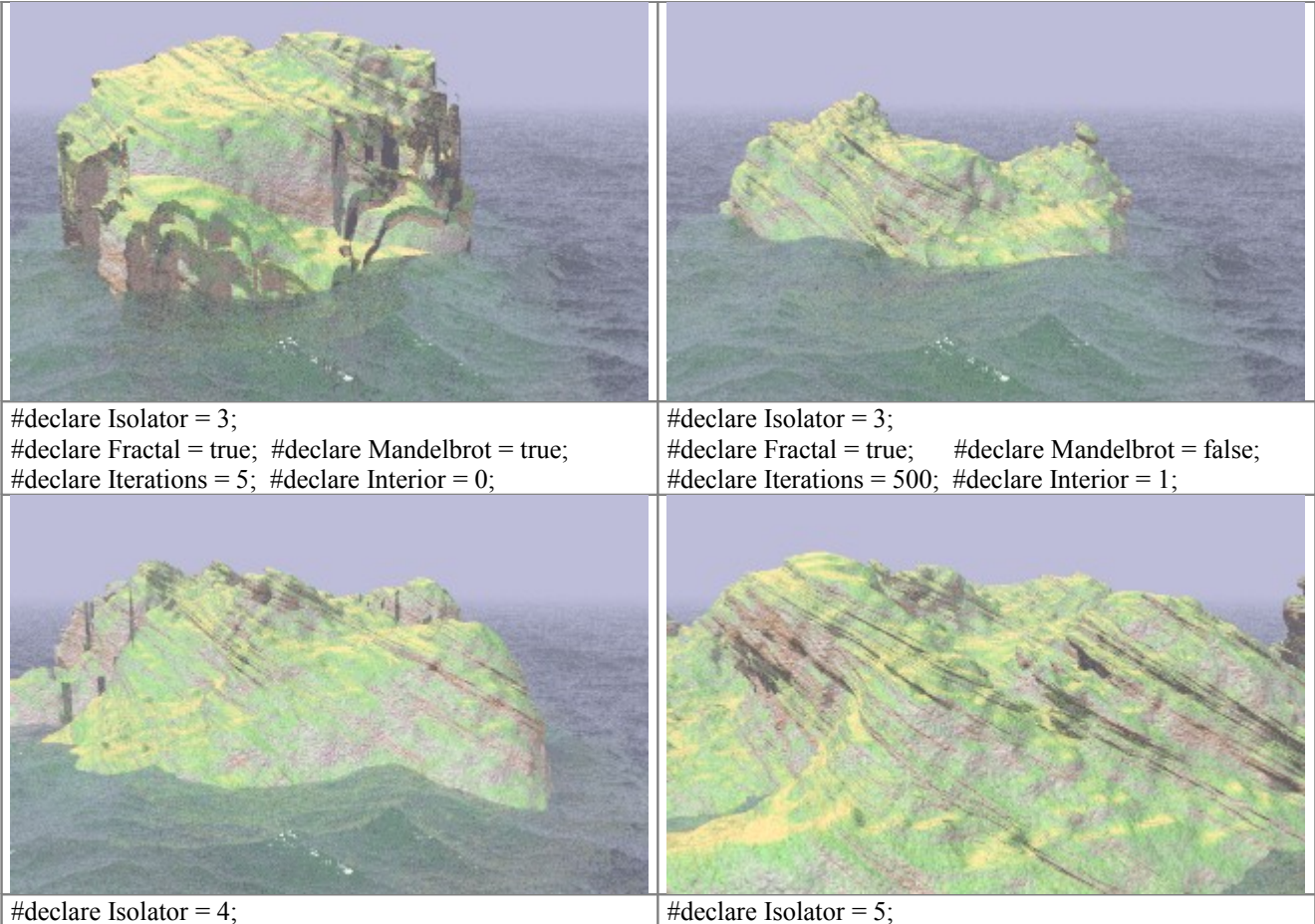
This element groups the available pattern functions used by the macro. A Boolean switch turns them on or off. When turned off, corresponding parameters in other elements are not used. The names of the parameters refer in general to the pattern they represent, e.g. Wrinkled will call the wrinkles pattern function. Exceptions are Isolated with its added parameter Isolator, and Hexed with its added parameter Honeycomb.

In particular, Isolated is an interesting parameter. When true, it allows for the generation of islands or single rocks. The Isolator parameter is the switch (1-5) that controls the basic shape of the island (see examples below and the appendices). See also the specific parameters in Element 9 - Island.

Hexed is an experimental parameter. When true, it calls the f\_hex\_x function that produces a hexagonal pattern. This pattern is very regular and so not very natural looking, but in vertical sections, it may simulate columnar joints, stalagmites or stalactites rather well. The parameter Honeycomb inverses the pattern, producing honeycomb-like structures on the surface. If you want to simulate the Giant's Causeway, it might be that the Dented or Cracked parameters give better results.

Example: #declare Wrinkled = true;

|   |  |
|---|--|
|  |    |
| <pre>#declare Isolated = false;</pre>   | <pre>#declare Isolated = true;      #declare Isolator = 1; #declare Islandgain = 0.5;    #declare Diameter = 1; #declare Wrinklesgain = 0.4; #declare Ripplesgain = 0.7;</pre> |
|  |    |
| <pre>#declare Isolator = 2;</pre>   | <pre>#declare Isolator = 3; #declare Wrinklesgain = 0.2; #declare Ripplesgain = 0.5;</pre>   |



**Element 4 – Pattern 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.

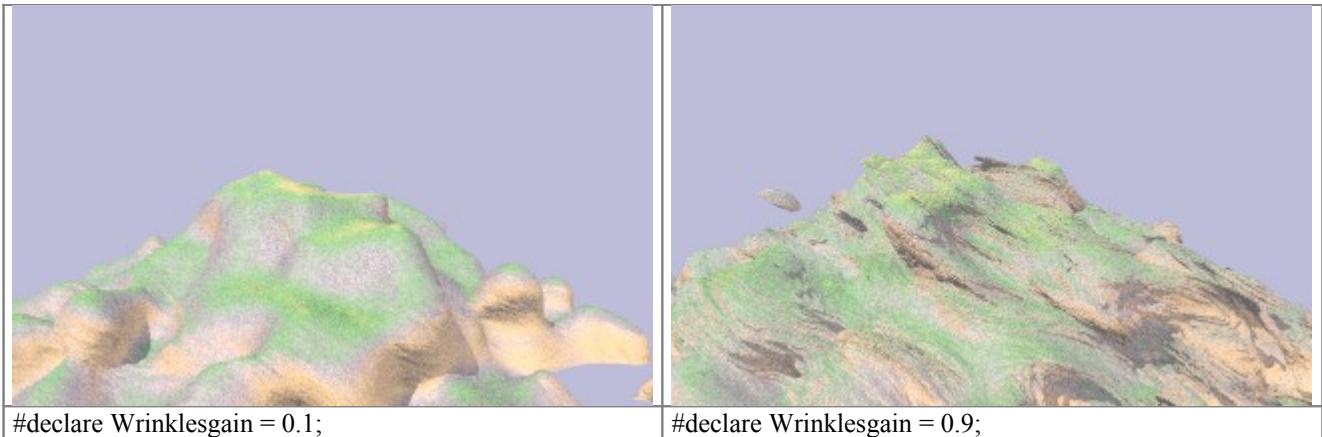
Example: #declare Wrinkleswarp = true;

**Element 5 – Pattern Gain:**

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

The strength of each of the pattern functions can be controlled by increasing or decreasing their 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;



### Element 6 – Pattern 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>;

### Element 7 – Substrate/Pattern Scale:

[parameters: Substratescale; Ridgescale; Fractalscale; Wrinklescale; Ripplescale; Wavescale; Dentscale; Cracklescale; Cracksizes; Hexscale]

The selected substrate or pattern functions can be scaled by adding a scaling vector. The Fractalscale parameter is a vector controlling the size of the fractal. You should also experiment with negative y values! This results in depressions or canyons in the landscape.

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

### Element 8 – Substrate/Pattern Angle:

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

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>;

### Element 9 – Fractal Translation:

[parameters: Fractaltrans]

If in use, the fractal can be positioned where you want it to be for the best effect.

Example: #declare Fractaltrans = <0, 1, -10>;

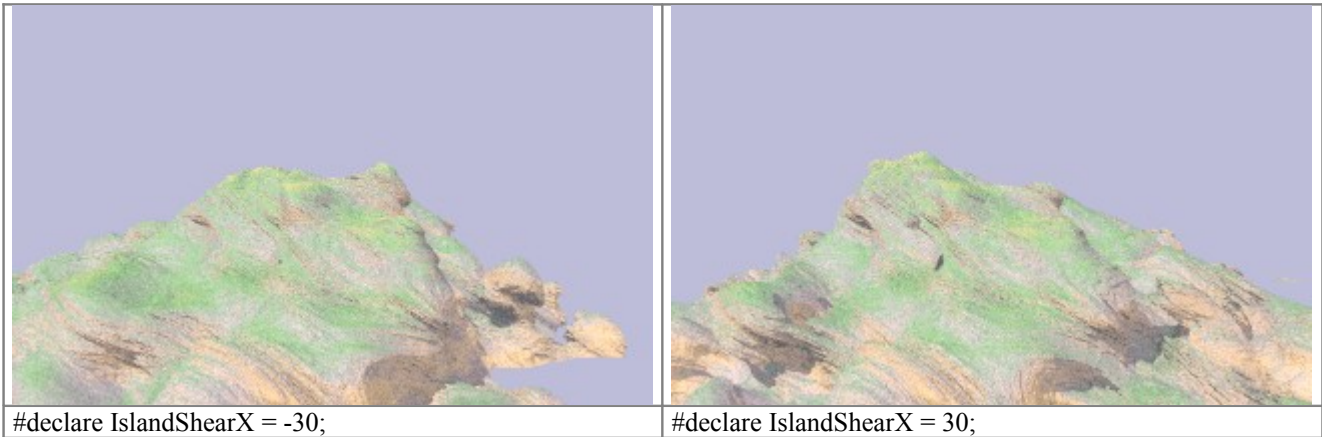
### Element 10 - Island:

[parameters: IslandShearX; IslandShearZ; Islandgain; Diameter]

The functions that can be used to generate islands or single rocks have a few additional parameters. The island can be sheared in the x- or the z-direction, so that it looks asymmetrical (see example below). Think of Gibraltar.

The Diameter parameter controls the diameter of the island. This is not necessarily a POV-Ray unit, but a measure of scaling, depending on the Isolator parameter chosen.

Example: #declare IslandShearX = -10;



### Element 11 - Isosurface:

[parameters: UpperBound; LowerBound; MaxGrad; Xsize; Zsize; Isoscale; Isorot; 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, Isorot, and Isotrans can additionally be used to scale, rotate, and/or translate the complete isosurface according to your own needs.

Example: #declare UpperBound = 2;

### Element 12 - Texture:

[parameters: Landtex; Simpletex; Slope; SlopeValue]

You can use your own textures of course, 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;



## Update history

01-02-05 Version 1.0  
01-03-05 small corrections in macro and docs  
01-04-05 added rotation to the isosurface  
01-05-05 added more functions to the island generator  
01-06-05 added the hexagon pattern functions  
01-08-05 added fractals  
01-18-05 added the appendixes to the docs  
01-20-05 Version 2.0  
01-27-05 Version 2.1. Added controls to f\_hetero\_mf and f\_ridged\_mf functions  
01-27-05 Version 2.2  
05-01-05 Versions 2A and 2B

## 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.

Version 2 - May, 2005

**Appendix 1 – Parameters and defaults for the two versions**

| <b>Geomorph mcr2</b>                      | <b>Default</b>                   | <b>Version</b> |
|---|----------------------------------|----------------|
| <b>(0) Declared landscape parameters:</b> |                                  |                |
| <b>Substrate:</b>                         |                                  |                |
| #ifndef (HeteroInf)                       | #local HeteroInf = false;        | B              |
| #ifndef (HetH)                            | #local HetH = 1;                 | A+B            |
| #ifndef (HetLac)                          | #local HetLac = 1;               | A+B            |
| #ifndef (HetOct)                          | #local HetOct = 1;               | A+B            |
| #ifndef (HetOff)                          | #local HetOff = 0;               | A+B            |
| #ifndef (HetT)                            | #local HetT = 0.2;               | A+B            |
| #ifndef (HetNG)                           | #local HetNG = 2;                | A+B            |
|   |                                  |                |
| #ifndef (Noised)                          | #local Noised = true;            | A+B            |
|   |                                  |                |
| #ifndef (Ridged)                          | #local Ridged = true;            | A+B            |
| #ifndef (RidgedInf)                       | #local RidgedInf = false;        | B              |
| #ifndef (RidH)                            | #local RidH = 0.8;               | A+B            |
| #ifndef (RidLac)                          | #local RidLac = 1;               | A+B            |
| #ifndef (RidOct)                          | #local RidOct = 1;               | A+B            |
| #ifndef (RidOff)                          | #local RidOff = 0;               | A+B            |
| #ifndef (RidG)                            | #local RidG = 0.9;               | A+B            |
| #ifndef (RidNG)                           | #local RidNG = 2;                | A+B            |
|   |                                  |                |
| #ifndef (Fractal)                         | #local Fractal = false;          | A+B            |
| #ifndef (Mandelbrot)                      | #local Mandelbrot = true;        | A+B            |
| #ifndef (Complex)                         | #local Complex = <0.353, 0.288>; | A+B            |
| #ifndef (Iterations)                      | #local Iterations = 5;           | A+B            |
| #ifndef (Exponent)                        | #local Exponent = 3;             | A+B            |
| #ifndef (Interior)                        | #local Interior = 0;             | A+B            |
| #ifndef (InFactor)                        | #local InFactor = 1.0;           | A+B            |
| #ifndef (Exterior)                        | #local Exterior = 1;             | A+B            |
| #ifndef (ExFactor)                        | #local ExFactor = 1.0;           | A+B            |
|   |                                  |                |
| <b>noise and randomness:</b>              |                                  |                |
| #ifndef (Noise)                           | #local Noise = 0.4;              | A+B            |
| #ifndef (Stream)                          | #local Stream = 92322;           | A              |
|   |                                  |                |
| <b>patterns:</b>                          |                                  |                |
| #ifndef (Wrinkled)                        | #local Wrinkled = true;          | A+B            |
| #ifndef (Rippled)                         | #local Rippled = false;          | A+B            |
| #ifndef (Waved)                           | #local Waved = false;            | A+B            |
| #ifndef (Dented)                          | #local Dented = false;           | A+B            |
| #ifndef (Sphered)                         | #local Isolated = Sphered;       | A+B            |
| #ifndef (Isolated)                        | #local Isolated = false;         | A+B            |
| #ifndef (Isolator)                        | #local Isolator = 2;             | A+B            |
| #ifndef (Crackled)                        | #local Crackled = false;         | A+B            |
| #ifndef (Hexed)                           | #local Hexed = false;            | A+B            |

|                           |  |     |
|---------------------------|--|-----|
| #ifndef (Honeycomb)       | #local Honeycomb = false;                | A+B |
| <b>warp:</b>              |  |     |
| #ifndef (Wrinkleswarp)    | #local Wrinkleswarp = false;             | A+B |
| #ifndef (Rippleswarp)     | #local Rippleswarp = false;              | A+B |
| #ifndef (Waveswarp)       | #local Waveswarp = false;                | A+B |
| #ifndef (Dentswarp)       | #local Dentswarp = false;                | A+B |
| #ifndef (Cracklewarp)     | #local Cracklewarp = false;              | A+B |
| <b>gain:</b>              |  |     |
| #ifndef (Substrategain)   | #local Substrategain = 1.0;              | A   |
| #ifdef (Substrategain)    | #local Subyscale = Substrategain;        | B   |
| #ifndef (Ridgegain)       | #local Ridgegain = 1.0;                  | A   |
| #ifdef (Ridgegain)        | #local Ridgeyscale = Ridgegain;          | B   |
| #ifndef (Wrinklesgain)    | #local Wrinklesgain = 0.5;               | A+B |
| #ifndef (Ripplesgain)     | #local Ripplesgain = 0.5;                | A+B |
| #ifndef (Wavesgain)       | #local Wavesgain = 0.5;                  | A+B |
| #ifndef (Dentsgain)       | #local Dentsgain = 0.5;                  | A+B |
| #ifndef (Cracklegain)     | #local Cracklegain = 1.0;                | A+B |
| #ifndef (Hexgain)         | #local Hexgain = 1.0;                    | A+B |
| <b>warp turbulence:</b>   |  |     |
| #ifndef (Wrinklesturb)    | #local Wrinklesturb = <0.5, 0.9, 0.9>;   | A+B |
| #ifndef (Ripplesturb)     | #local Ripplesturb = <0.5, 0.9, 0.9>;    | A+B |
| #ifndef (Wavesturb)       | #local Wavesturb = <0.5, 0.9, 0.9>;      | A+B |
| #ifndef (Dentsturb)       | #local Dentsturb = <0.5, 0.9, 0.9>;      | A+B |
| #ifndef (Crackleturb)     | #local Crackleturb = <0.5, 0.9, 0.9>;    | A+B |
| <b>scale:</b>             |  |     |
| #ifndef (Substratescale)  | #local Substratescale = <1.0, 1.0, 1.0>; | B   |
| #ifndef (Ridgescale)      | #local Ridgescale = <1.0, 1.0, 1.0>;     | A+B |
| #ifndef (Fractalscale)    | #local Fractalscale = <1.0, 1.0, 1.0>;   | A+B |
| #ifndef (Wrinklescale)    | #local Wrinklescale = <0.5, 0.2, 0.5>;   | A+B |
| #ifndef (Ripplescale)     | #local Ripplescale = <0.5, 0.2, 0.5>;    | A+B |
| #ifndef (Wavescale)       | #local Wavescale = <0.5, 0.2, 0.5>;      | A+B |
| #ifndef (Dentscale)       | #local Dentscale = <0.5, 0.2, 0.5>;      | A+B |
| #ifndef (Cracklescale)    | #local Cracklescale = <0.5, 0.2, 0.5>;   | A+B |
| #ifndef (Cracksize)       | #local Cracksize = 2;                    | A+B |
| #ifndef (Hexscale)        | #local Hexscale = <0.2, 0.4, 0.2>;       | A+B |
| <b>angle of rotation:</b> |  |     |
| #ifndef (Fractalangle)    | #local Fractalangle = <0, 0, 0>;         | A+B |
| #ifndef (Wrinklesangle)   | #local Wrinklesangle = <0, 0, -20>;      | A+B |
| #ifndef (Ripplesangle)    | #local Ripplesangle = <0, 0, -25>;       | A+B |
| #ifndef (Wavesangle)      | #local Wavesangle = <0, 0, -5>;          | A+B |
| #ifndef (Dentsangle)      | #local Dentsangle = <0, 0, -10>;         | A+B |
| #ifndef (Crackleangle)    | #local Crackleangle = <0, 0, 0>;         | A+B |
| #ifndef (Hexangle)        | #local Hexangle = <0, 0, 0>;             | A+B |

|                             |                                     |     |
|-----------------------------|-------------------------------------|-----|
|                             |                                     |     |
| <b>fractal translation:</b> |                                     |     |
| #ifndef (Fractaltrans)      | #local Fractaltrans = <0, 0, 0>;    | A+B |
|                             |                                     |     |
| <b>island:</b>              |                                     |     |
| #ifdef (SphereShearX)       | #local IslandShearX = SphereShearX; | A+B |
| #ifndef (IslandShearX)      | #local IslandShearX = 15;           | A+B |
| #ifdef (SphereShearZ)       | #local IslandShearZ = SphereShearZ; | A+B |
| #ifndef (IslandShearZ)      | #local IslandShearZ = 10;           | A+B |
| #ifdef (Spheregain)         | #local Islandgain = Spheregain;     | A+B |
| #ifndef (Islandgain)        | #local Islandgain = 0.5;            | A+B |
| #ifndef (Diameter)          | #local Diameter = 1;                | A+B |
|                             |                                     |     |
| <b>isosurface:</b>          |                                     |     |
| #ifdef (FinalFunction)      | #undef FinalFunction                | B   |
| #ifndef (Evaluate)          | #local Evaluate = false;            | A+B |
| #ifndef (Min factor)        | #local Min factor = 0.6;            | A+B |
| #ifndef (Attenuation)       | #local Attenuation = 0.7;           | A+B |
| #ifndef (UpperBound)        | #local UpperBound = 3;              | A+B |
| #ifndef (LowerBound)        | #local LowerBound = -2;             | A+B |
| #ifndef (MaxGrad)           | #local MaxGrad = 10;                | A+B |
| #ifndef (Threshold)         | #local Threshold = 0.0;             | A+B |
| #ifndef (Xsize)             | #local Xsize = 10;                  | A+B |
| #ifndef (Zsize)             | #local Zsize = 10;                  | A+B |
| #ifndef (Isoscale)          | #local Isoscale = <1, 1, 1>;        | A+B |
| #ifndef (Isorot)            | #local Isorot = <0, 0, 0>;          | A+B |
| #ifndef (Isotrans)          | #local Isotrans = <0, 0, 0>;        | A+B |
|                             |                                     |     |
| <b>texture:</b>             |                                     |     |
| #ifndef (Simpletex)         | #local Simpletex = false;           | A+B |
| #ifndef (Slope)             | #local Slope = true;                | A+B |
| #ifndef (SlopeValue)        | #local SlopeValue = 0.6;            | A+B |

## Appendix 2 – Parameter settings for Islands in the examples

```
// substrate:
#declare HeteroInf = false;
#declare HetH = 1;
#declare HetLac = 1;
#declare HetOct = 1;
#declare HetOff = 0;
#declare HetT = 0.2;
#declare HetNG = 2;

#declare Noised = false;

#declare Ridged = false;
#declare RidgedInf = false;
#declare RidH = 1.2;
#declare RidLac = 3;
#declare RidOct = 6;
#declare RidOff = 1.15;
#declare RidG = 5;
#declare RidNG = 2;

#declare Fractal = true;
#declare Mandelbrot = true;
#declare Complex = <0.1, 0.2>;
#declare Iterations = 50;
#declare Exponent = 3;
#declare Exterior = 1;
#declare ExFactor = 1.0;
#declare Interior = 0;
#declare InFactor = 1.0;

// noise:
#declare Noise = 0.4;           // controller for the f_noise3d function

// patterns:
#declare Wrinkled = true;
#declare Rippled = true;
#declare Waved = false;
#declare Dented = false;
#declare Isolated = true;      // when true, generates an island or an isolated rock
#declare Isolator = 2;        // settings are 1-5
#declare Crackled = false;
#declare Hexed = false;
#declare Honeycomb = true;

// warp:
#declare Wrinkleswarp = true;
#declare Rippleswarp = true;
#declare Waveswarp = false;
#declare Dentswarp = true;
#declare Cracklewarp = false;

// gain:
#declare Wrinklesgain = 0.4;
#declare Ripplesgain = 0.7;
#declare Wavesgain = 0.5;
#declare Dentsgain = 0.5;
#declare Cracklegain = 1.0;
#declare Hexgain = 2.8;
```

---

```

// warp turbulence:
#declare Wrinklesturb = <0.5, 0.9, 0.9>;
#declare Ripplesturb = <0.5, 2, 0.9>;
#declare Wavesturb = <0.5, 0.9, 0.9>;
#declare Dentsturb = <0.5, 0.9, 0.9>;
#declare Crackleturb = <0.5, 0.9, 0.9>;

// scale:
#declare Substratescale = <5.0, 2, 5.0>;
#declare Ridgescale = <3.0, 0.5, 3.0>;
#declare Fractalscale = <1.5, -1, 1.5>;

#declare Wrinklescale = <1.5, 0.2, 1.5>;
#declare Ripplescale = <0.1, 0.2, 0.1>;
#declare Wavescale = <0.5, 0.2, 0.5>;
#declare Dentscale = <0.5, 0.2, 0.5>; // pattern scale
#declare Cracklescale = <0.5, 0.2, 0.5>; // multiplier for Cracklescale
#declare Cracksize = 1;
#declare Hexscale = <0.15, 0.2, 0.15>;

// angle of rotation:
#declare Fractalangle = <0, 0, 0>;
#declare Wrinklesangle = <0, 0, -30>;
#declare Ripplesangle = <0, 0, -30>;
#declare Wavesangle = <0, 0, 0>;
#declare Dentsangle = <0, 0, -10>;
#declare Crackleangle = <0, 0, 0>;
#declare Hexangle = <0, 0, 0>;

// fractal translation:
#declare Fractaltrans = <0, 0, 0>;

// island:
#declare IslandShearX = 0; // shear of the island function in the x direction
#declare IslandShearZ = 0; // shear of the island function in the z direction
#declare Islandgain = 0.5;
#declare Diameter = 1; // diameter of the island function

// isosurface:
#declare UpperBound = 2; // Bounding box upper limit
#declare LowerBound = -5; // Bounding box lower limit
#declare MaxGrad = 11; // max_gradient of the isosurface (change according to need)
#declare Xsize = 15; // increase size of the bounding box (not scale!) for X
#declare Zsize = 15; // increase size of the bounding box (not scale!) for Z
#declare Isoscale = <5, 5, 5>; // scale of the isosurface
#declare Isorot = <0, 0, 0>; // rotation of the isosurface
#declare Isotrans = <0, 3, 0>; // translation of the isosurface

// texture:
##declare Landtex = // define your own texture here
// for the internal textures, use:
#declare Simpletex = false; // just a choice between two nice textures
#declare Slope = true; // true = slope & altitude; false = gradient texture
#declare SlopeValue = 0.6; // (0-1) higher values: more slope, less altitude, and vice versa

```

---

### Appendix 3 - The sea in the examples

```
// TdG_NorthSeaWater macro
#macro TdG_NorthSeaWater (Filter,Transparency,ScaleX,ScaleY,ScaleZ,FD,FP)
material {
  texture {
    pigment {
      color rgbft <0.104167, 0.3125, 0.260417, Filter, Transparency>
    }
    normal {
      waves , 0.05
      turbulence 0.5
      frequency 100.0
      phase 1.0
      ramp_wave
      scale <ScaleX, ScaleY, ScaleZ>
    }
    finish {
      ambient 0.15
      diffuse 0.0
      brilliance 5.0
      phong 1.0
      phong_size 90.0
      specular 0.4
      roughness 0.01
      conserve_energy
      reflection {
        .1,
        .9
        falloff 2
      }
    }
  }
  interior {
    ior 1.33
    fade_distance FD
    fade_power FP
    fade_color rgb <0.0, 0.437467, 0.0>
    media {
      samples 2,2
      absorption rgb <1.0, 0.05, 0.05>
    }
  }
}
#end // end of macro

//Deep Water:
plane { y, -10.0 pigment {color rgb <0.119608, 0.137255, 0.556863>*0.01}}

//-----
// Iso_Sea by Marc Jacquier (2003)
// Originally: Isosurface plugin 1beta by S. Shonfield, morayplugin@yahoo.ca
// adapted to left-handed orientation:

#declare Scale_x=7;
#declare Scale_z=5;
#declare Damp=1.2;
#declare Lacunarity=3;
#declare Octave=6;
#declare Offset=1.18;
#declare Gain=5;
```

---

```
#declare Wave_Height=0.8;
isosurface {
  function {
    y +1-f_ridged_mf(x/Scale_x, 0, z/Scale_z,Damp,Lacunarity,Octave,Offset,Gain,2)*Wave_Height
  }
  contained_by {box{<-500,-11,-500>,<500,1,500>}}
  threshold 0
  accuracy 0.001
  max_gradient 20
  //max_trace 1
  //all_intersections
  TdG_NorthSeaWater (
    0.9, // filter
    0.25, // transparency
    3, // ScaleX
    2, // ScaleY
    1, // ScaleZ
    1, // fade distance
    3 // fade power
  )
  scale <1,1,1>
  rotate <0,0,0>
  translate <0, 0, 0>
} // end isosurface
```