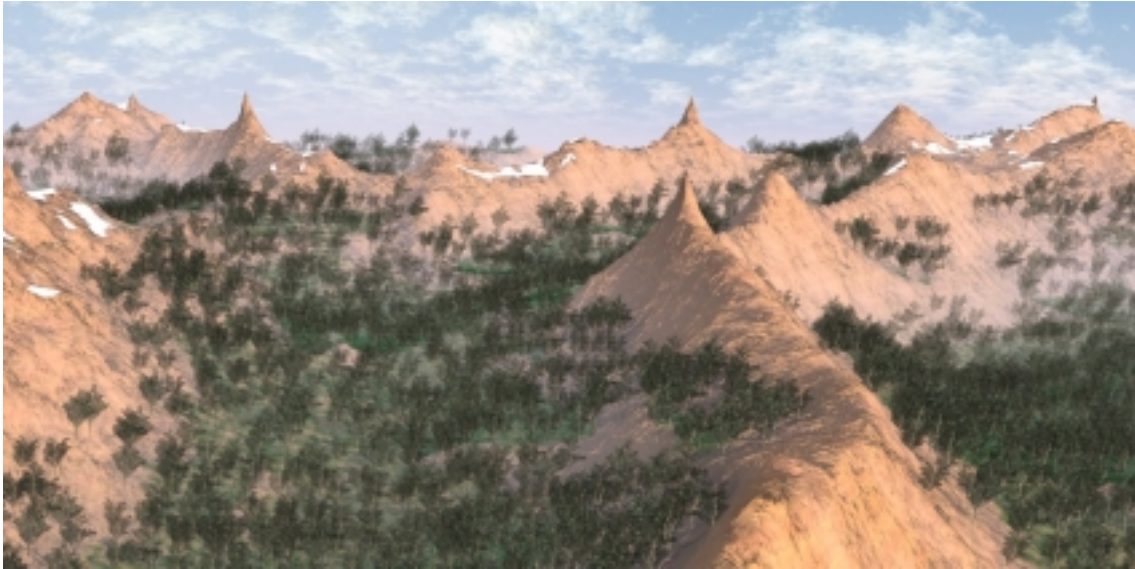


Blender World Forge



© August 2004 – Stefano Selleri a.k.a. S68
Vers. 0.1.0
‘T con zero’

Table of Contents

INTRODUCTION	4
PACKAGE CONTENT	4
DEPENDENCIES	4
INSTALLATION	4
USAGE	5
NOISE	5
FLAT	5
LATLONG	6
TETRA, OCTA & ICO	6
NOISE SETTINGS	7
FLAT MAPPING	8
SPHERICAL MAPPING	8
FILTERS	8
INVERT FILTER	9
EXPONENTIAL FILTER	9
POLYNOMIAL FILTER	9
HORIZONTAL STEP (HSTEP) AND VERTICAL STEP (VSTEP) FILTERS	9
LANDMARKS	10
POSTPRO	12
FULL MESH LIMITED LAYERS	12
VERTEX ONLY LIMITED LAYERS	13
FILE FORMAT	13
PRESETS	16
ASTEROID1.BWF	16
ASTEROID2.BWF	16
MONUMENT.BWF	16
WORLD.BWF	16
DEFAULT.BWF	16
CHANGELOG	17
0.1.0	17
0.0.9	17
0.0.8	17
0.0.7	17
KNOWN BUGS	18
TODO	18

Discalimer

This software is provided 'as it is' no guarantees expressed or implied are given, except that it will occupy some space on your disk(s). The software, the file example.blend and this document are © 2003,2004 Stefano <S68> Selleri and released under Blender Artistic license which you should have received with the package. If this is not the case a copy of said license is available at www.blender.org or directly from the author selleri@det.unifi.it.

Introduction

Blender World Forge (BWF) is a Python script designed to build worlds. Its main purpose is the generation of fractal terrains, on which the user can later on add water, clouds etc.

BWF saw a steady development from the first version 0.0.1 up to version 0.0.12, even if not all of these versions were released to the public. All the 0.0.x versions were based on an open source noise generation package, CGkit. CGkit was pretty nice but has some drawbacks, the major being that, to work with Blender under Linux, it required to be compiled with the same compiler used to compile Blender (at least I have been told so) hence you have to compile both CGkit and Blender by yourself.

The latest development of the Python API occurred in Blender 2.33 presented a bunch of new functions, most important a full set of fractal noises. BWF was hence mature to get completely rid of CGkit and become a Blender-only package. A new series was hence produced, 0.1.x, of which this is the first release, BWF 0.1.0.

Package Content

The **BWF** package should contain:

1. **BWF.pdf** – This file you are reading ☺
2. **BWF.blend** – An example Blender file with the script within
3. **BWF-0.1.0.py** – The script, as a separate text file.
4. **Licence.txt** – The Blender Artistic License under which this package is given.
5. **PreSets** – A directory containing a bunch of pre-set environments, just for your fun.

Dependencies

BWF needs the following softwares to be correctly installed on your computer to work:

1. An operating system of your choice ☺.
2. Blender 2.33 and maybe later version (© The Blender Fund - www.blender.org).
3. Python 2.3.3 or whatever Python is recommended with your version of Blender (© The Python Software Foundation – www.python.org)

Installation

Unpack the package in a directory of your choice.

Once you are done you can load the **bwf.blend** file of this package in Blender.

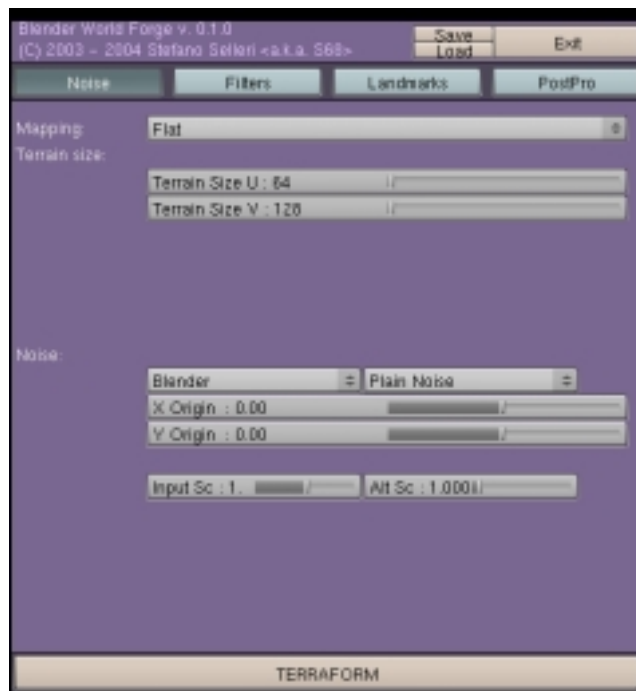
Alternatively you can load **bwf-0.1.0.py** in a text window in any of your .blend files.

Usage

Place your mouse cursor on the left Blender window, the text window containing the **BWF** script, and press **ALT-P**. The Graphical User Interface (**GUI**) here on the right should appear. If not you have a problem.

The **GUI** is divided into four part:

- 1 - A top part with credits and three buttons:
 - a) A **Save** button – This opens a File Selection Window where you can type in a file name where to save all settings. The file name default extension is **.bwf**
 - b) A **Load** button – This opens a File Selection Window where you can select a **.bwf** file for loading some pre-saved setups. A bunch of setups is provide with the distributions.
 - c) An **Exit** button – The **Exit** button does exactly what you are thinking of.
- 2 - A Upper middle part with four radio buttons. These selects the four possible button windows of **BWF**. Namely: **NOISE**, **FILTERS**, **LANDMARKS** and **POSTPRO**. By clicking on these buttons the main central part of the interface shows the pertinent buttons, as described in the following.
- 3 - A middle part, the bigger, main, part, holding whichever button you decided to see by clicking on the aforementioned buttons.
- 4 - A lower part with just one button: **TERRAFORM** to launch execution.



Noise

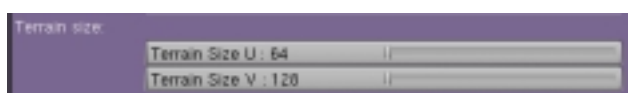
The **NOISE** buttons allows for the definition of the terrain mapping, size – in arbitrary units – and for the definitions of the raw parameters governing the noise functions generating the altimetry.

The first button is a MenuButton and defines the mapping, by selecting the Mapping type the rest of the interface changes.



Flat

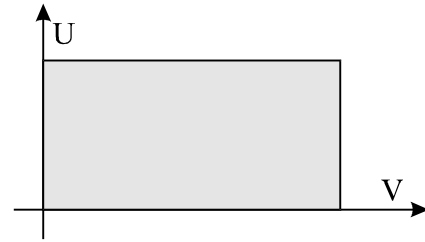
Generates flat terrains. These terrains are created on a rectangular portion of a **UV** plane.



In this case the terrain dimensions are defined via two sliders, **U** and **V**. The default setting is 64x128. Please note that the **UV** plane is mapped onto Blender **XY** plane with a 1:10 ratio, that is, a 10x10 points grid in the **UV** plane is a square of side 1 Blender Unit.

The default settings will then create a terrain of 128x64 vertices (8192) which will be 12.8 x 6.4 Blender units.

If **U** and **V** parameters are set so that the total number of vertices would exceed Blender's built-in limit of 64k vertices in a mesh more than a single mesh will be automatically and seamlessly created, none of which greater than 251x251 vertices.

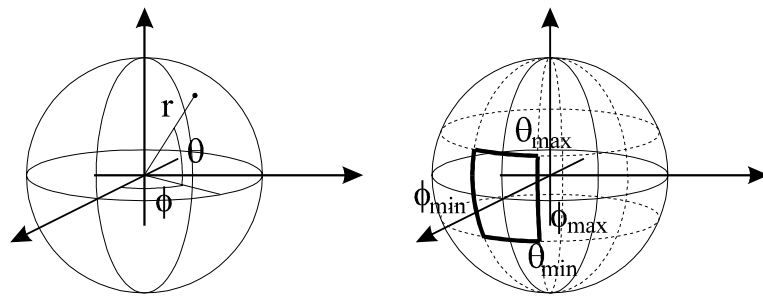


Newest Blender versions do not have the 64k vertex limit any more, but BWF still makes many smaller meshes rather than a single big one. You can still Join them if you so like!

LatLong

The **BWF** script is able to produce entire worlds, portion of worlds and plots of land. For a small plot of land a flat terrain is appropriate, but for a large portion of land – yet still portions – the globe curvature must be taken into account.

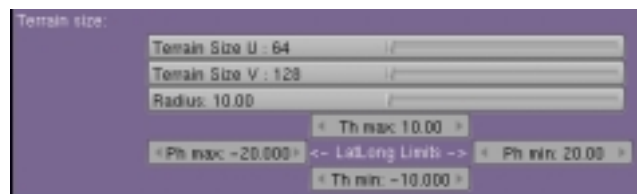
This is done by selecting the **LatLong** Mapping toggle button. This is the most complex mapping. The **UV** plane is warped onto a sphere and **U** becomes the **Theta** angle of a spherical reference while **V** becomes the **Phi** angle. The range of variations of **Theta** and **Phi** are set



independently from the number of **U** and **V** points via 4 numerical buttons placed around the toggle button. The default setting will generate an entire sphere, and, in this case, keeping **V = 2xU** is a good choice.

The radius of the sphere to be created is defined by the **Radius** parameter.

Smaller portions of a sphere can be obtained by changing the **Theta** and **Phi** limits. It is often a good idea to use **LatLong** mapping



for relatively small plot of lands because the **UV** mapping in the spherical coordinates causes an unneeded concentration of vertices at poles. For full globes the following mappings are preferred.

Tetra, Octa & Ico

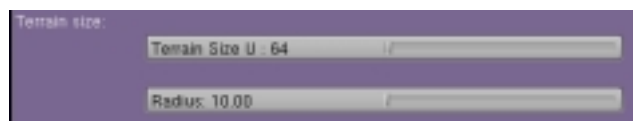
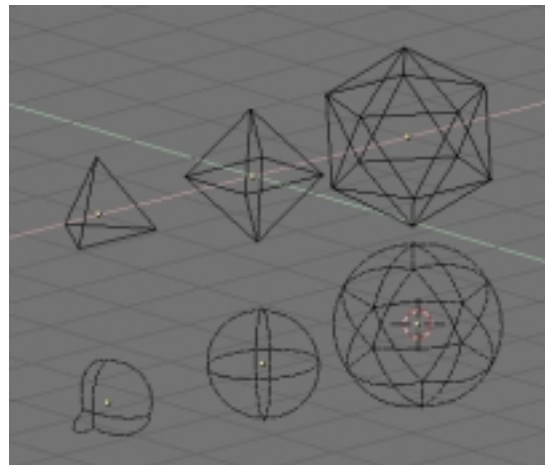
These three options work in a similar way inasmuch they create a full globe on the basis of its regular subdivision in triangles.

Tetra generates a sphere as four triangles, starting from a Tetrahedral base. **Octa** as eight, starting from an Octahedral base. Lastly **Ico** uses twenty triangles, being indeed a level 1 Icosphere.

Each of these triangles is subdivided in *at least* 3 quadrilaterals, which are then further subdivided into smaller quadrilaterals, exactly as using SubSurf would do, except that vertices are mapped exactly onto a sphere and not calculated via the Catmull Clarke algorithm.

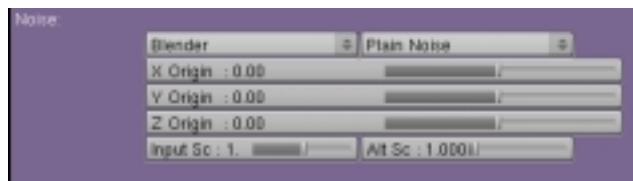
The **Terrain Size** slider dictates the level of subdivision. Please note that *each* triangular patch will be a separate blender Object, limited in 65000 vertices, so the maximum level of subdivision is 146. The default level of 64 generates globes of very high detail, so you might want to tone it down while experimenting!

Please note also that each Triangular patch is generated as three quadrilateral meshes, Joined by the script. The current Python API does not have a Remove Doubles method, so this must be made by hand.



Noise Settings

The lower group of buttons allows for noise definitions. Blender now provides a wide range of **Noise Types** and **Noise Functions**. Mixing very different landscapes can be created.



Blender basically provides 10 noise types, the standard Blender one, which was the base of all textures up to Blender 2.32. Then there is the classic **Perlin Noise**, a **Modified Perlin Noise**, some flavors of **Voronoi** noises and a piecewise constant **Cell Noise**. The doc of Blender explains these noises in detail.

These noise types are then used as a basis for the true Noise Function. You can use the **Plain Noise**, with no modifications, a **Fractional Brownian Motion**, **MultiFractal**, **Hybrid Multifractal**, **Ridged Multifractal**, **Turbulence** and **Voronoi** functions. For each of these some additional settings appear.



Since all of these are built-in Blender functions no much will be said here. The parameters for each are exactly those you have available when you use the Musgrave type of texture in the texture buttons.

The terraforming process operates in 4 stages.

1. A regular mesh, either Planar, LatLong or Spherical is created
2. Noise is added to displace vertices along quote z for planar, and along the radius r for spherical mappings
3. Filters (next section) are applied to the displacement
4. Landmarks (section after next) are applied to the displacement

Point 2 requires the computation of noise as a function. This is done in a **UVW** reference which is separate from Blender's **XYZ** reference. This allows us to explore different locations of the noise space without having to move our Objects in Blender space.

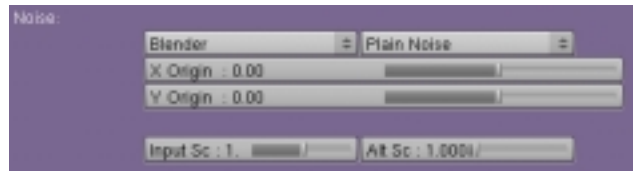
Flat Mapping

If the Mapping is planar then parameters are

$$V = x \times NIS + X_{Origin}$$

$$U = y \times NIS + Y_{Origin}$$

with X_{Origin}, Y_{Origin} the values in the relevant sliders and NIS the value of the Noise **Input Scale** slider.



Since all Blender Noise functions returns values in various ranges, the fractal terrain will have elevations in the same range. For greater flexibility two operations are implicitly performed. The subtraction of 0.5, to make the noise average ideally 0, and additional multiplicative scale parameter, defined by the **Altitude Scaling** slider.

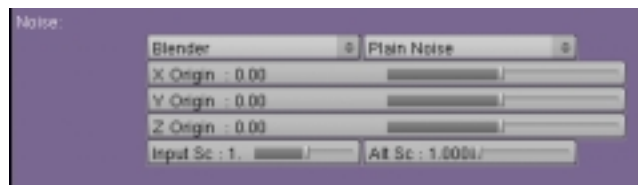
Spherical Mapping

If the Mapping is spherical, whether **LatLong**, **Tetra**, **Octa** or **Ico** then parameters are

$$V = x \times NIS + X_{Origin}$$

$$U = y \times NIS + Y_{Origin}$$

$$W = z \times NIS + Z_{Origin}$$

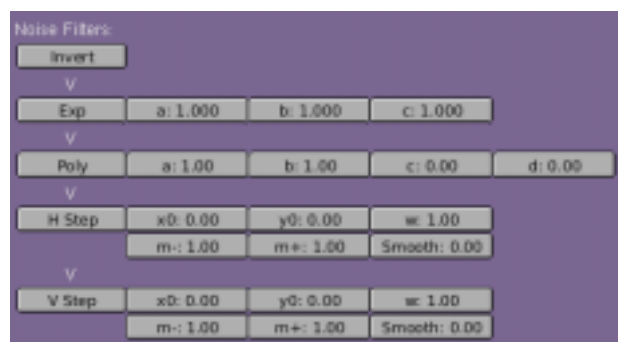


This time three coordinates are used, being the surface non-planar, hence a third parameter is needed. All other parameters stands.

Filters

For a higher degree of flexibility further processing of the noises are possible. After the noise is computed, hence the displacement of the vertex, and before it is added to the vertex some pre-set mathematical functions can be applied.

Please note that more than one filter can be selected and that filters are applied **in the order in which they are listed on the GUI**. This is reminded to you by the 'V' (arrows) signs.



This means that the *Domain* of each function is the *Co-Domain* of the previous function and that the domain of the *first function to be applied* is the output of the noise process.

We will use the $z_{new} \leftarrow f(z_{old})$ formalism in describing filters.

Invert filter

This is the simplest mapping

$$z \leftarrow -z$$

Exponential filter

$$z \leftarrow ae^{bz+c}$$

Which, for positive b will enhance peaks and flatten valleys while, for negative b will flatten peaks and deepen valleys.

Polynomial filter

$$z \leftarrow a + bz + cz^2 + dz^3$$

Which can be very flexible if YKWYAD™¹

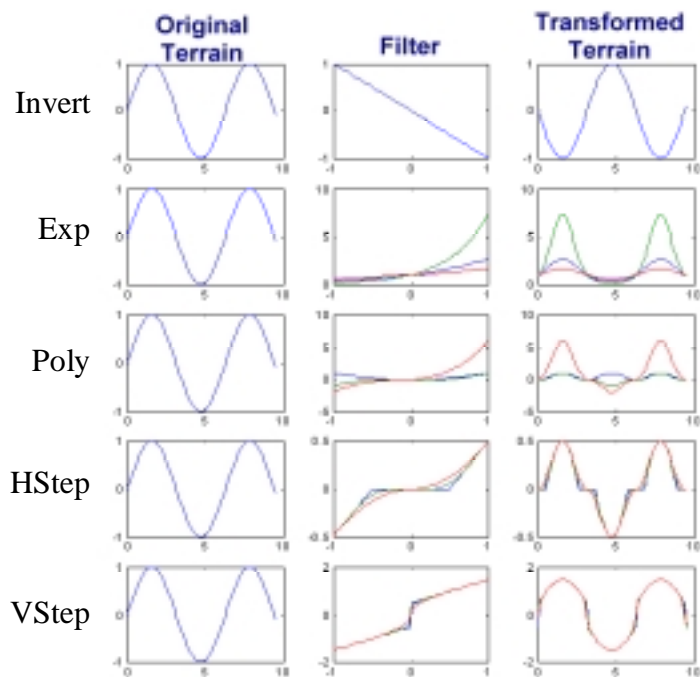
Horizontal Step (Hstep) and Vertical Step (Vstep) filters

These are actually too complex to be given in plain formulas, so look at the graphs.

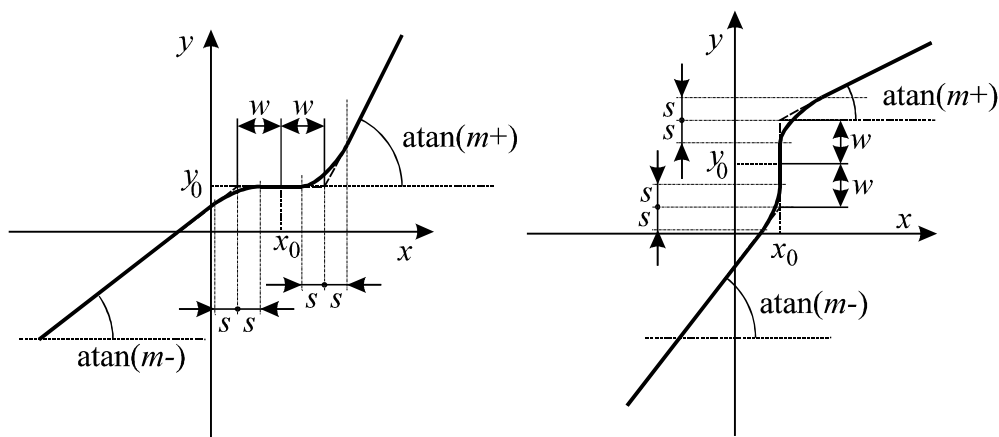
In particular the graphs here on the right shows a sinusoidal terrain on the left column, some possible filters in the middle column, and the relative transformed terrain in the right column.

Top row is relative to the **Inversion** filter. Second row is relative to the **Exponential** filter. Blue lines are for $a=1$, $b=1$, $c=0$. Green lines for $a=1$, $b=2$, $c=0$ and red lines for $a=1$, $b=0.5$, $c=0$. Third row is the **Polynomial** filter, Blue lines are for $a=0$, $b=0$, $c=1$, $d=0$. Green lines for $a=0$, $b=0$, $c=0$, $d=1$ and red lines for $a=0$, $b=1$, $c=2$, $d=3$.

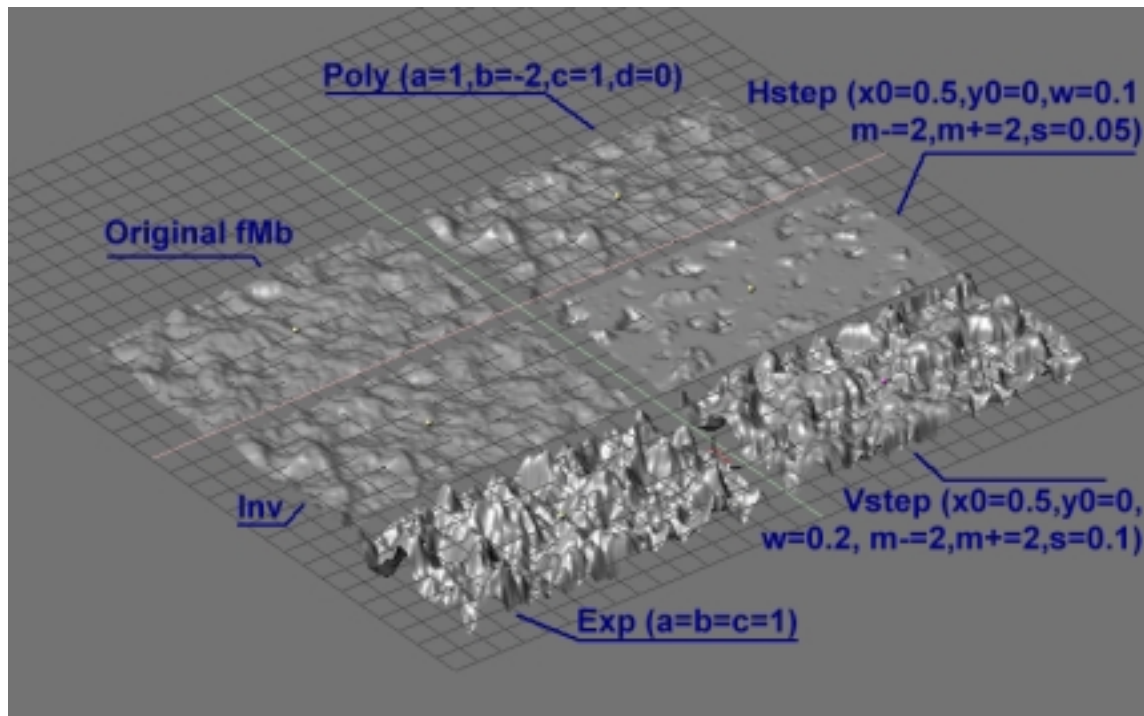
Fourth row is the **Hstep** filter. All three filters have $x_0=0$, $y_0=0$, $w=0.5$, $m=-1$, $m+=0$. Blue line has $s=0$, green $s=0.25$ and red $s=0.5$. Fifth row is the **Vstep** filter with exactly the same parameters. For a meaning of these please look at the following graphs.



¹ You Know What You Are Doing



Basically, if $s=0$ you have a piecewise linear function made by a steep m_- segment, an horizontal (vertical) segment and a steep m_+ segment. If $s>0$ (s cannot be greater than w) then the linear segments are joined by third order interpolating polynomials (for **Hstep**) or arcs of ellipsis/hyperbola (for **Vstep**).



The previous picture shows the effect of the filters on a basic **Fractional Brownian Motion (fBm)** noise.

Landmarks

The third button window allows for the definition of some conspicuous features, or landmarks, on the terrain. Up to now two features are implemented: craters and peaks.

Cospicuous Landmarks:			
Craters No. 0		Seed 0	
r:	1.00	rV:	1.00
h:	1.00	hV:	1.00
d:	1.00	dV:	1.00
s:	1.000	sV:	1.00
Peaks No. 0		Seed 0	
r:	1.00	rV:	1.00
h:	1.00	hV:	1.00
mt:	1.00	mtV:	1.00
mb:	1.000	mbV:	1.00

Both landmarks are structures with cylindrical symmetry defined as an addition to land height according to a given function of the distance from the landmark position.

Landmarks are randomly scattered all over the terrain. If their number **Crater No** and **Peaks No** is zero no landmark are created. Otherwise the required number is created. Location is generated randomly via the Noise function. Such a distribution is governed via the 'seed' parameter.

The mathematical function of the crater is:

$$z = \begin{cases} d \left(\frac{\rho}{r} \right)^{s+1} + h - d & \text{if } \rho < r \\ h \left(\frac{r}{\rho} \right)^s & \text{if } \rho \geq r \end{cases}$$

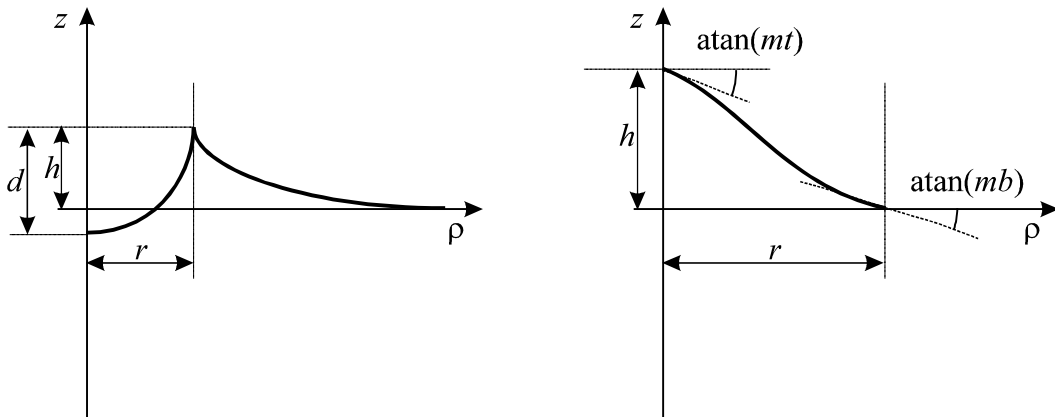
And this should explain the meaning of the first column of parameters: radius **r**, height over terrain **h**, depth below rim **d** and steepness of sides **s**. The figure below should clarify even more. The second column gives a random variance to these values, to obtain craters of all dimensions, height and steepness.

The mathematical function for the peak is

$$z = \begin{cases} a_0 + a_1 \rho^1 + a_2 \rho^2 + a_3 \rho^3 & \text{if } \rho < r \\ 0 & \text{if } \rho \geq r \end{cases}$$

That is a third order polynomial interpolating point (0,h), being **h** the peak height and (r,0), being **r** the peak base radius. Further parameters defining the peak is the steepness at the top **mt** and the steepness at the bottom **mb**. Again these values are given as mean and variance to have peaks of different dimensions and shapes. Note that **mt** and **mb** can be both positive and negative, but you probably want to keep them **negative** unless you know what you are doing.

The following picture should clarify the parameters more.

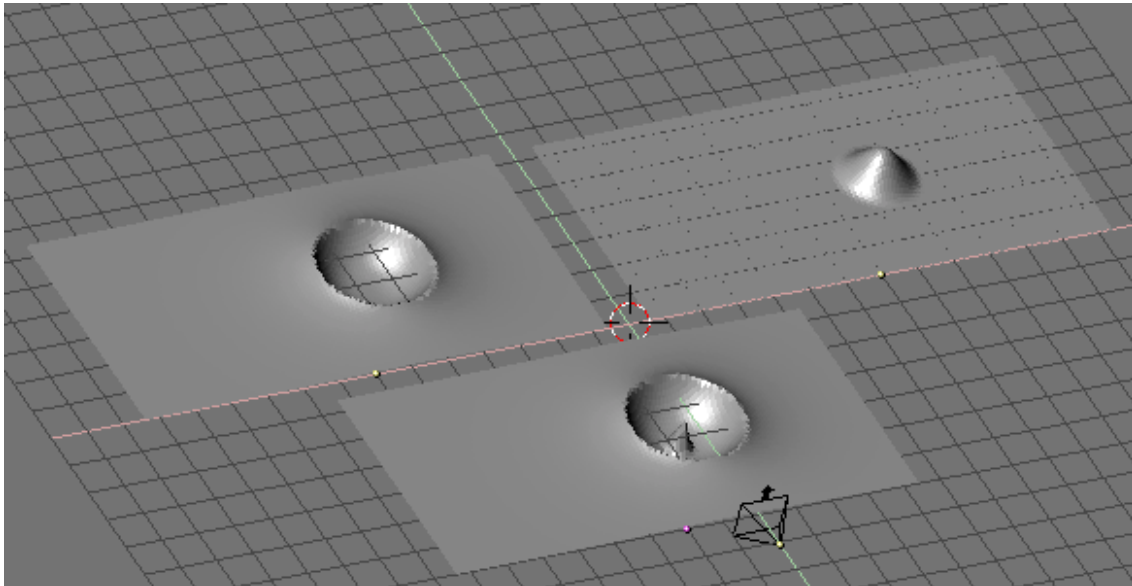


Be warned on two things:

1. If both peaks and craters are used and the seeds are equal for both, you will have a peak at the center of each crater.

2. Crater/Peak function is calculated for **every point once for every landmark**. This can be quite time consuming.

The following figure shows one crater ($r=1.5, h=0.5, d=1, s=2$, variances and seed to 0), one peak ($r=1, h=1, mt=0, mb=0$, variances and seeds to 0) and a combination (same crater parameters, peak has $r=0.3, h=1, mt=-2, mb=0$). Please note that, to better show the landmarks, noise amplitude is here set to 0. A non – zero amplitude gives realistic results.



PostPro

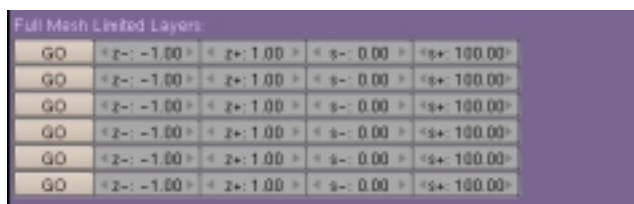
New BWF 0.1.0 finally have some true PostPro.

PostPro is **not** part of the terraforming process, it is postprocessing (M. de La Palice...) You must have an already generated plot of land selected to use PostPro. PostPro works correctly only for **Flat** mappings (for now).

Full Mesh Limited Layers

This option creates a new mesh, from the selected one, containing all the faces of the original mesh obeying to given constraints.

Constraints are given on altitude ($z-$, that is Z minimum and $z+$, that is Z maximum) of each face and on the slope ($s-$, minimum steepness and $s+$, maximum steepnes) of each face.



If a face is completely above $z-$ and below $z+$ and its slope is greater than $s-$ but smaller than $s+$ then the face is duplicated on the newly created object, otherwise it is not. The new mesh is created when the **GO** button is pressed.

This is great to build vegetation coverage.

You can define 6 separate settings for these layers, each has its own **GO** button this is usefull if you save and load your settings, since you can keep six different settings altogether.

Vertex Only Limited Layers

On the other hand, you can create only vertices, randomly scattered on the selected, original, plot of land, so that they abide to given constraints and exhibit a given density.

Vertex Only Limited Layers:					
GO	< z-: -1.00 >	< z+: 1.00 >	< s-: 0.00 >	< s+: 100.00 >	< V: 4.00 >
GO	< z-: -1.00 >	< z+: 1.00 >	< s-: 0.00 >	< s+: 100.00 >	< V: 4.00 >
GO	< z-: -1.00 >	< z+: 1.00 >	< s-: 0.00 >	< s+: 100.00 >	< V: 4.00 >
GO	< z-: -1.00 >	< z+: 1.00 >	< s-: 0.00 >	< s+: 100.00 >	< V: 4.00 >
GO	< z-: -1.00 >	< z+: 1.00 >	< s-: 0.00 >	< s+: 100.00 >	< V: 4.00 >
GO	< z-: -1.00 >	< z+: 1.00 >	< s-: 0.00 >	< s+: 100.00 >	< V: 4.00 >

Altitude and slope constraints are handled exactly as above, but faces are not created in the newly generated mesh, instead vertices are generated, randomly placed on the face. For each face a variable number of vertices is created, so that the average density of nodes per face is the one defined in the **V** slider.

This is great if you dupliver items on the vertex cloud (stones, trees, bushes etc.)

File Format

Blender World Forge loads and save **.bwf** files holding all the terraforming settings.

Such files are in XML standard and using BWF syntax, here it is the file holding all default settings:

```
<BlenderWorld version="0.1.0">
<Mapping>1</Mapping>
<Dimensions USize="64" VSize="128" Radius="10.0"
ThMin="-10.0" ThMax="10.0" PhMin="-20.0" PhMax="20.0">
</Dimensions>
<Noise Type="0" XOffset="0.0" YOffset="0.0" ZOffset="0.0"
NoiseInputScale="1.0" AltitudeScale="1.0"
Function="0" Metric="0" H="1.0"
Lacunarity="2.0" Octaves="2.0" Offset="1.0"
Gain="1.0" Hard="0" Amplitude="0.5" Frequency="2.0" Exp="2.5">
</Noise>
<Invert>
0
</Invert>
<Exp a="1.0" b="1.0" c="1.0">
0
</Exp>
<Poly a="1.0" b="1.0"
c="0.0" d="0.0">
0
</Poly>
<HStep Px0="0.0" Py0="0.0"
Pw="1.0" Ps="0.0"
Pmm1="1.0" Pmp1="1.0">
0
</HStep>
<VStep Px0="0.0" Py0="0.0"
Pw="1.0" Ps="0.0"
Pmm1="1.0" Pmp1="1.0">
0
</VStep>
<Craters Seed="0"
Radius="1.0" RadiusV="1.0"
Height="1.0" HeightV="1.0">
```

```

        Depth="1.0" DepthV="1.0"
        Steep="1.0" SteepV="1.0">
0
</Craters>
<Peaks Seed="0"
    Radius="1.0" RadiusV="1.0"
    Height="1.0" HeightV="1.0"
    TopSteep="0.0" TopSteepV="0.0"
    FootSteep="0.0" FootSteepV="0.0">
0
</Peaks>
<SubLayer
    Zmin="-1.0" Zmax="1.0"
    Smin="0.0" Smax="100.0">
0
</SubLayer>
<SubLayer
    Zmin="-1.0" Zmax="1.0"
    Smin="0.0" Smax="100.0">
1
</SubLayer>
<SubLayer
    Zmin="-1.0" Zmax="1.0"
    Smin="0.0" Smax="100.0">
2
</SubLayer>
<SubLayer
    Zmin="-1.0" Zmax="1.0"
    Smin="0.0" Smax="100.0">
3
</SubLayer>
<SubLayer
    Zmin="-1.0" Zmax="1.0"
    Smin="0.0" Smax="100.0">
4
</SubLayer>
<SubLayer
    Zmin="-1.0" Zmax="1.0"
    Smin="0.0" Smax="100.0">
5
</SubLayer>
<VertexCloud
    Zmin="-1.0" Zmax="1.0"
    Smin="0.0" Smax="100.0"
    Density="4.0">
0
</VertexCloud>
<VertexCloud
    Zmin="-1.0" Zmax="1.0"
    Smin="0.0" Smax="100.0"
    Density="4.0">
1
</VertexCloud>
<VertexCloud
    Zmin="-1.0" Zmax="1.0"
    Smin="0.0" Smax="100.0"

```

```

        Density="4.0">
2
</VertexCloud>
<VertexCloud
    Zmin="-1.0" Zmax="1.0"
    Smin="0.0" Smax="100.0"
    Density="4.0">
3
</VertexCloud>
<VertexCloud
    Zmin="-1.0" Zmax="1.0"
    Smin="0.0" Smax="100.0"
    Density="4.0">
4
</VertexCloud>
<VertexCloud
    Zmin="-1.0" Zmax="1.0"
    Smin="0.0" Smax="100.0"
    Density="4.0">
5
</VertexCloud>
</BlenderWorld>

```

The meaning of all the tags should be self explanatory, as well as all their attribute (Bold and blue) which correspond to the items with the same meaning in the GUI. Red items are numerical values which can assume any value (well, stay in the limits imposed by the GUI, please! And keep them Integer or Float accordingly)

PreSets

As for V. 0.0.7 there are a series of pre-set environments you can load. This will help you in tweaking the various parameter, especially the filters, starting from decent setups rather than having to blindly change the default values!

Please note that, of course, pre-set made with previous version of BWF than 0.1.0 are unusable since the file format changed...

Asteroid1.bwf

A tiny asteroid, very irregular but somewhat smooth, better if you Join all meshe and SubSurf.

Asteroid2.bwf

A medim asteroid, very rough, with few craters

Monument.bwf

A setup for a flat slice of the Monument Valley (more or less)

World.bwf

An entire Earth-like world.

Here the presence of water erosion and continental shelves is given via usage of the HStep filter.

Once you loaded the PreSet and have hit the **TERRAFORM** button you can simply add a Level 4 IcoSphere (With the cursor at the center of the co-ordinate system) and set it to SubSurf Level 1.

Via the **NKEY** menu You can give to the sphere a scale of 7.108 (strange numbers, uhu?). This sphere will be the ocean of your world.

Of course a second, sphere, scaled a little more, with clouds will give you an atmosphere.

Please note that this example will generate around 250.000 vertices!

Default.bwf

The default settings, to go back to them easily!

ChangeLog

No changelogs before 0.0.6... so new stuff is from 0.0.7 onwards!

0.1.0

Major rewrite

- No more Cgkit, only Blender, hence:
 - Different GUI
 - Different Noises
 - Different file format
- PostPro for flat lands :
 - Meshes
 - Vertex clouds

0.0.9

New in 0.0.9 – 28-10-2003

- Complete rewrite of Load-Save stuff, now using XML!
- Progress bars works at last!

0.0.8

New in 0.0.8 – 25-10-2003

- The Tetra, Octa, Ico mapping
- Complete rework of the internal flow, now meshes are created regular and then elaborated. Much simpler, much more elegant
- Z Offsett added, general re-write of the noise mapping algorithm.

Actually 0.0.8 was never released to the public, but I need to keep it for my mental clearness...

- The noise is made to have 0 average, hence the invert filter goes back to $z \leftarrow -z$ and all other filters are much more consistent.
- Load-Save stuff reworked for new parameters.

0.0.7

New in 0.0.7 – 20-10-2003

- The Save and Load Buttons, the Save *overwrites* existing files, and the check on the **.bwf** extension is pretty loose. The Load button does no checking at all, so expect crashes if you try to load a wrong file
- The 'Inverse' Filter is changed from $z \leftarrow -z$ to $z \leftarrow 1 - z$ to maintain the land in the [0-1] range. This in any case must be checked since Zscale setting is there...

Known Bugs

- None (?)

ToDo

More PreSets

PostPro for spherical worlds too!

Then true post processing! Erosion! Lakes! Seas !