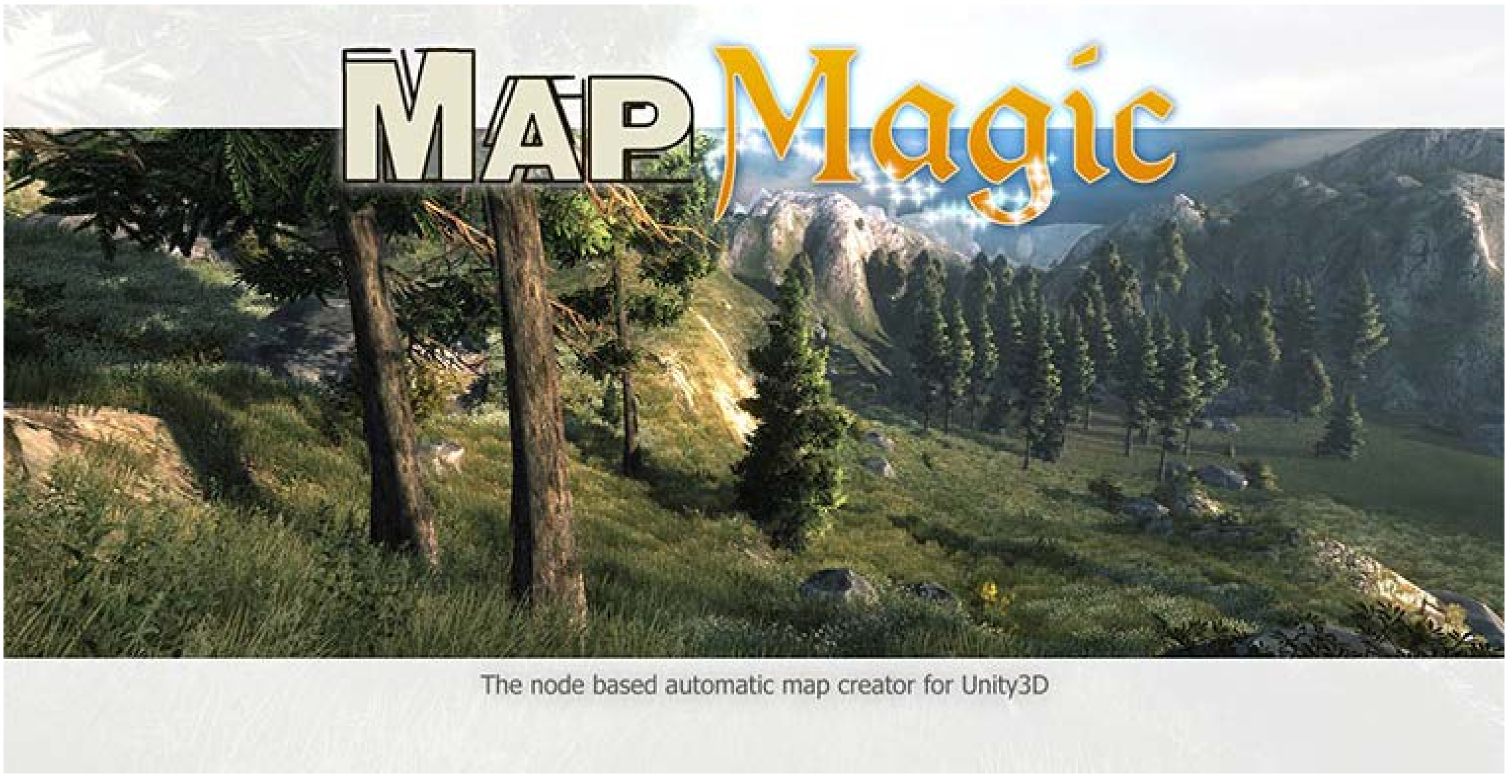
# Home



Welcome to MapMagic - a platform for terrain creation and automatic game object placement.

This tool uses a node-based visual scripting interface to determine creation logic. Each node represents a separate algorithm called a “generator”. Examples of generators include: noise, voronoi, blend, curve, erosion, object scatter, forest, etc. All nodes are presented on a field called the “graph”.

## Main concepts

## Generators

### Map Generators

## Tutorials

Creating villages with mapmagic
Objects and trees biome blending

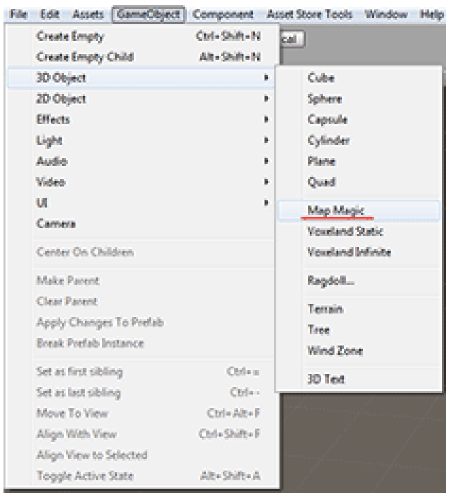
## Questions and Answers

General FAQ: How do I ...?
Problem Solving: ... does not work, what should I do?
Scripting FAQ: How do I ... using scripts?

# Quick start

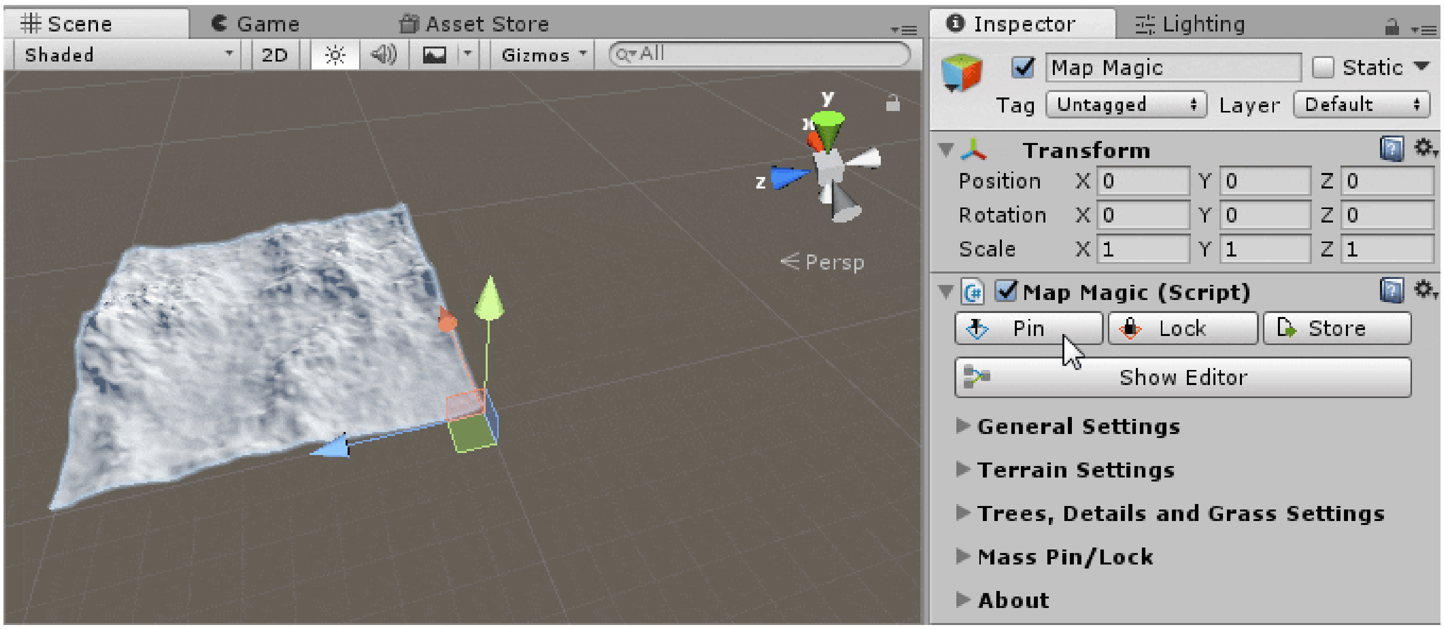
## Part 1: Creating MapMagic

The first thing you need to do is create a MapMagic object, which contains the MapMagic script, the initial terrain, and allows the editor interface to be opened. To create a MapMagic object, click the GameObject menu -> 3D Object -> MapMagic.



A new game object named “MapMagic” will appear in the hierarchy menu. It already has the initial pinned terrain and three example generators, so the terrain is not flat. Clicking this object will show basic settings in the Inspector.

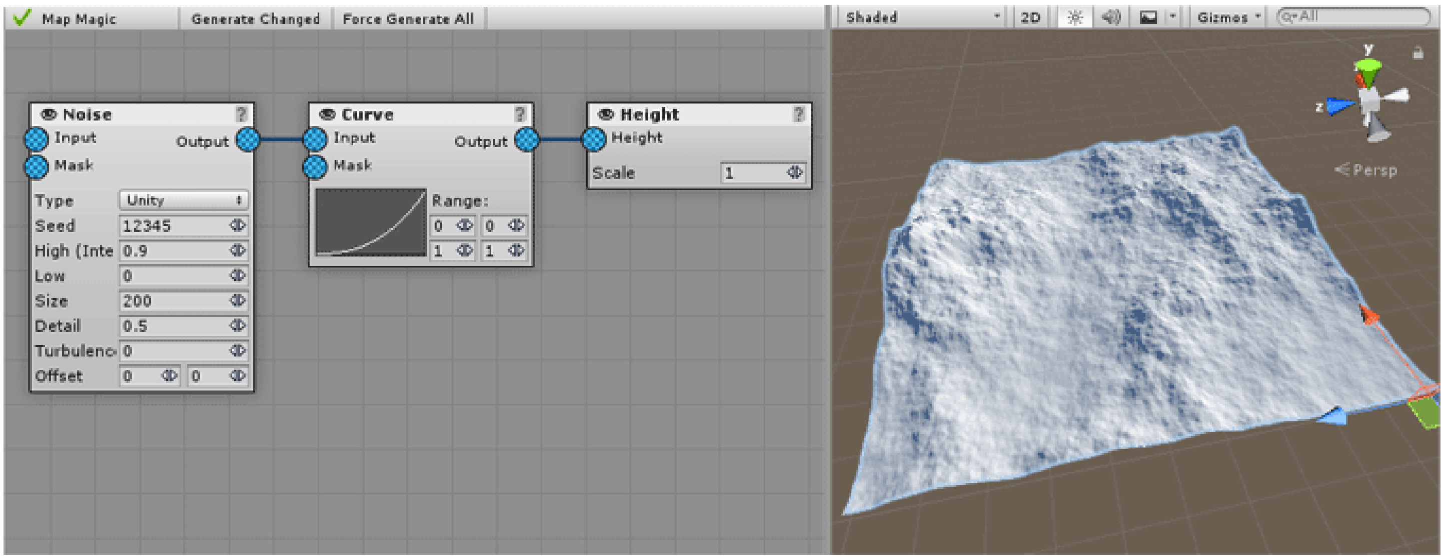
The “Pin Terrain” button allows pinned terrains to be selected in the Scene view. Left-clicking on an empty area will pin a terrain, and it will appear in a moment, after it’s been generated. Left-clicking on an already pinned terrain will unpin it. To exit pinning mode click on the “Pin Terrain” button once more.



Any change made to pinned terrains will be re-written on the next generate. To prevent this, use the terrain lock by pressing the “Lock Terrain” button and selecting a pinned terrain that should be locked. When using a locked terrain, try not to modify a height greatly, otherwise you can get noticeable borders between your unchanged, locked terrains and their neighbors that have a new procedural heightmap with whatever changes you’ve made since locking the previous terrain.

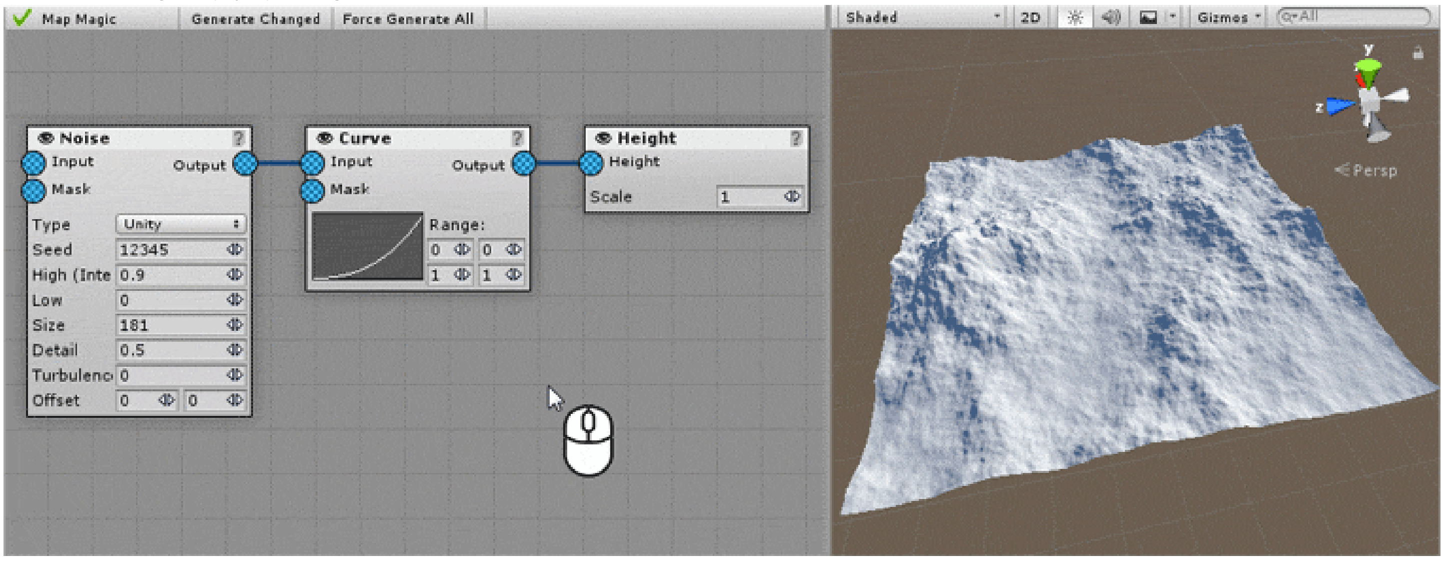
By the default all of the pinned terrains are saved in scene, there is no separate .asset files for them. But in some cases using separate terrain files could be handy - for example when passing some terrains to another project or for reducing the scene size. In this case terrain data could be exported to the standard .asset file by pressing the "Store" button and selecting terrain with a green selection frame.

Within the MapMagic object you created, the “Show Editor” button will open up the MapMagic Editor Window. Here you can see three initial nodes (called generators): Noise, Curve and Height (output generator), all connected together. It's recommended to dock this window somewhere so it will not overlap the scene view.



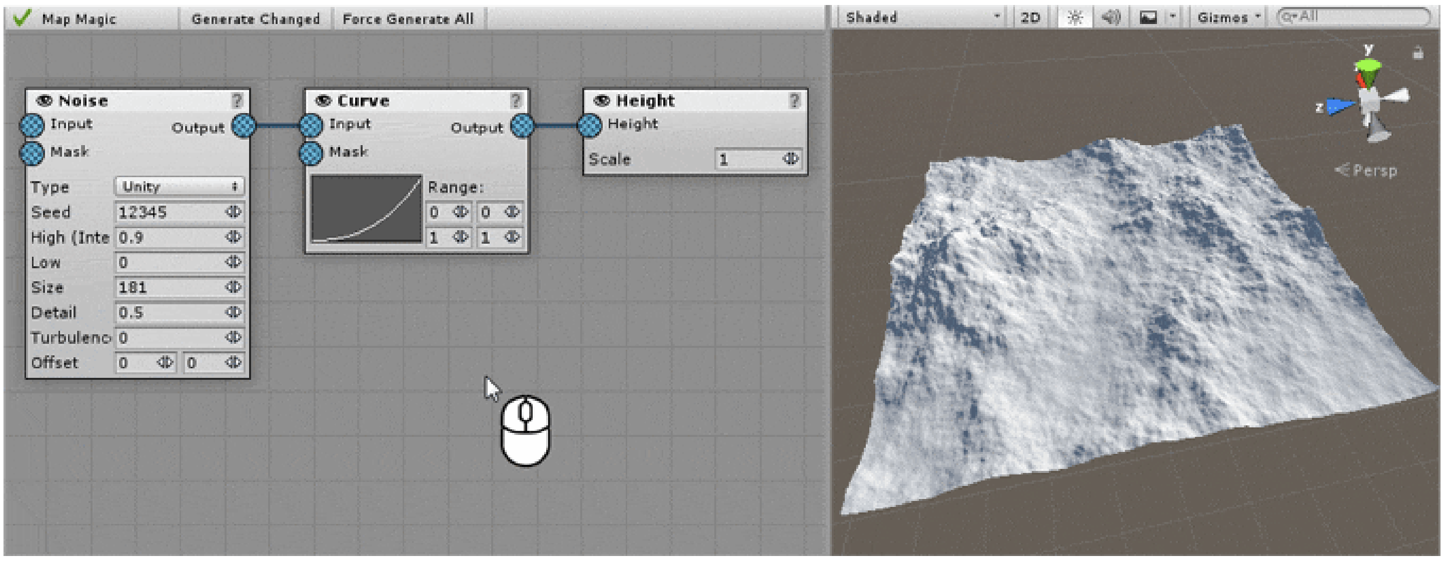
## Part 2: Editor Window

To **pan** (**scroll**) the editor window click and drag middle mouse button anywhere on the graph. It does not matter if you are middle-clicking empty space, generator, field or connection wire:



> You can use Alt + left click combination if you are using 2-button mouse.

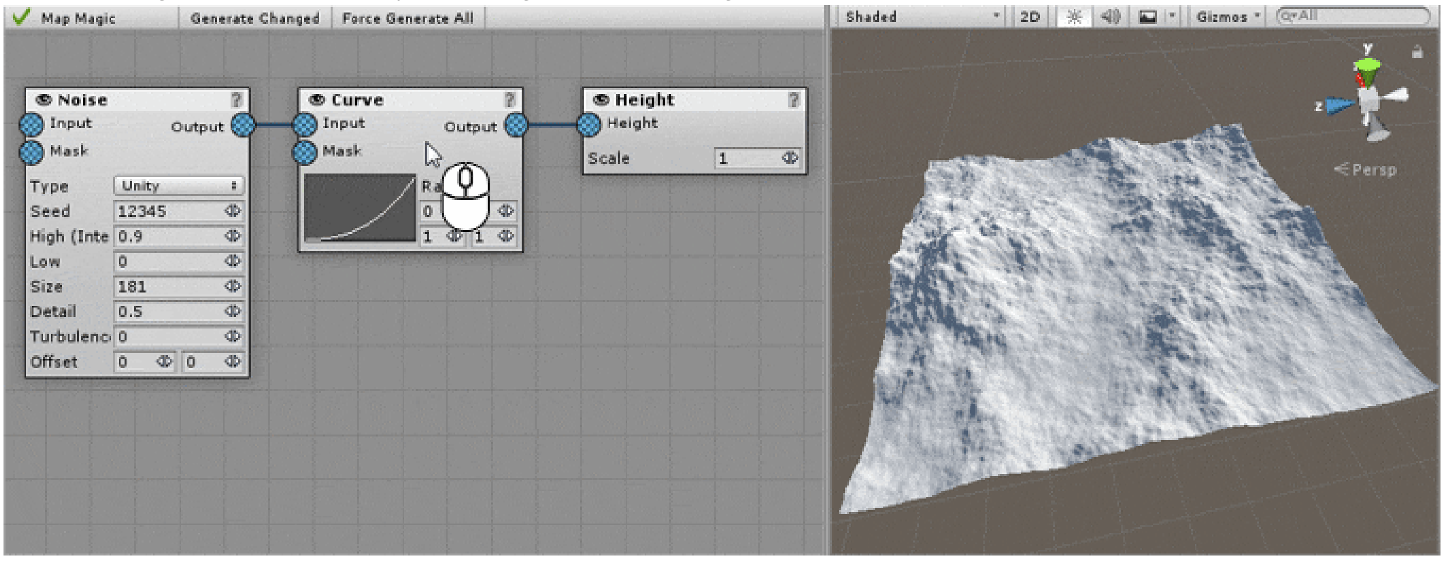
To **zoom** the graph rotate the mouse wheel up or down:



Noise: Type Unity, Seed 12345, High (Inte 0.9, Low 0, Size 181, Detail 0.5, Turbulence 0, Offset 0 0
Curve: Range: 0 0 / 1 1
Height: Scale 1

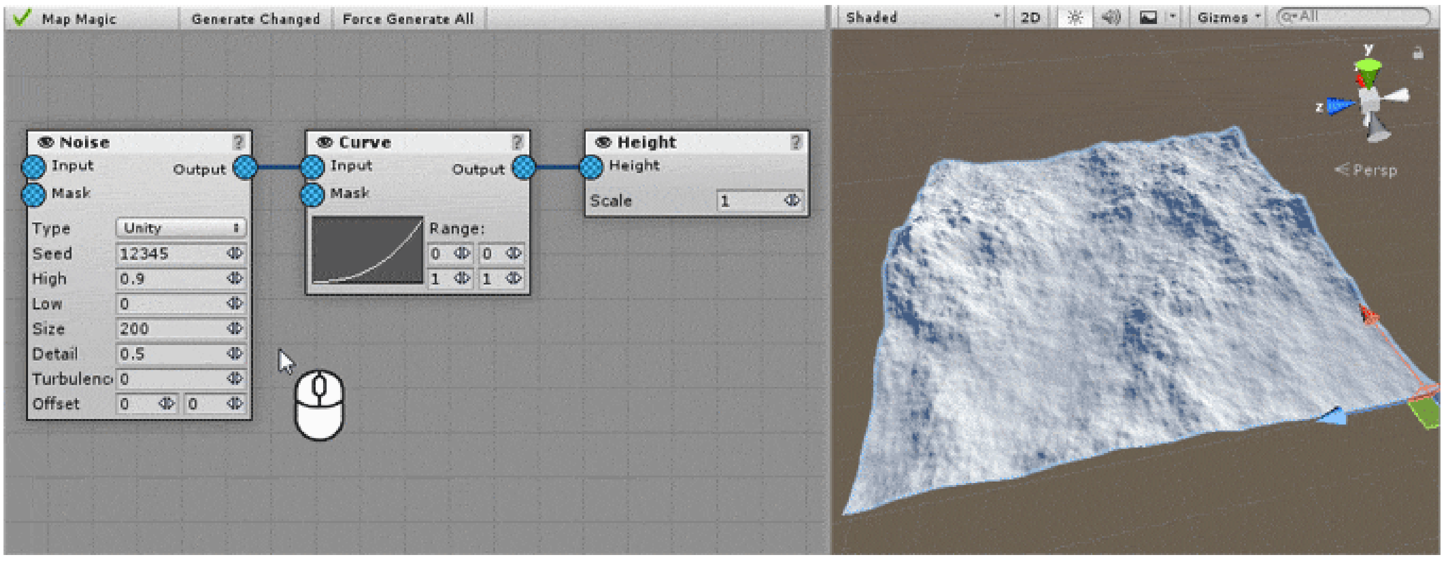
> Alternatively you can use keyboard keys "+" and "-" for zoom.

To **move** the generator left-click anywhere on generator and drag it:



Noise: Type Unity, Seed 12345, High (Inte 0.9, Low 0, Size 181, Detail 0.5, Turbulence 0, Offset 0 0
Curve: Range: 0 0 / 1 1
Height: Scale 1

Try changing the generator values by left-click and drag ◆ on the right side of the property field:

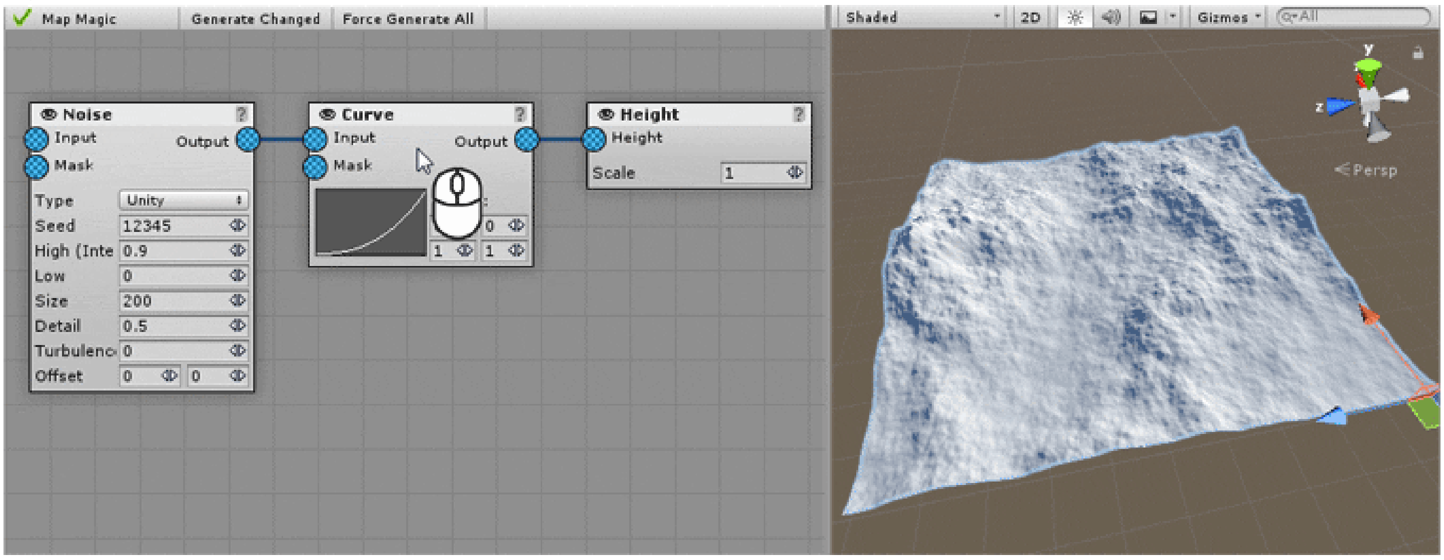


Noise: Type Unity, Seed 12345, High 0.9, Low 0, Size 200, Detail 0.5, Turbulence 0, Offset 0 0
Curve: Range: 0 0 / 1 1
Height: Scale 1

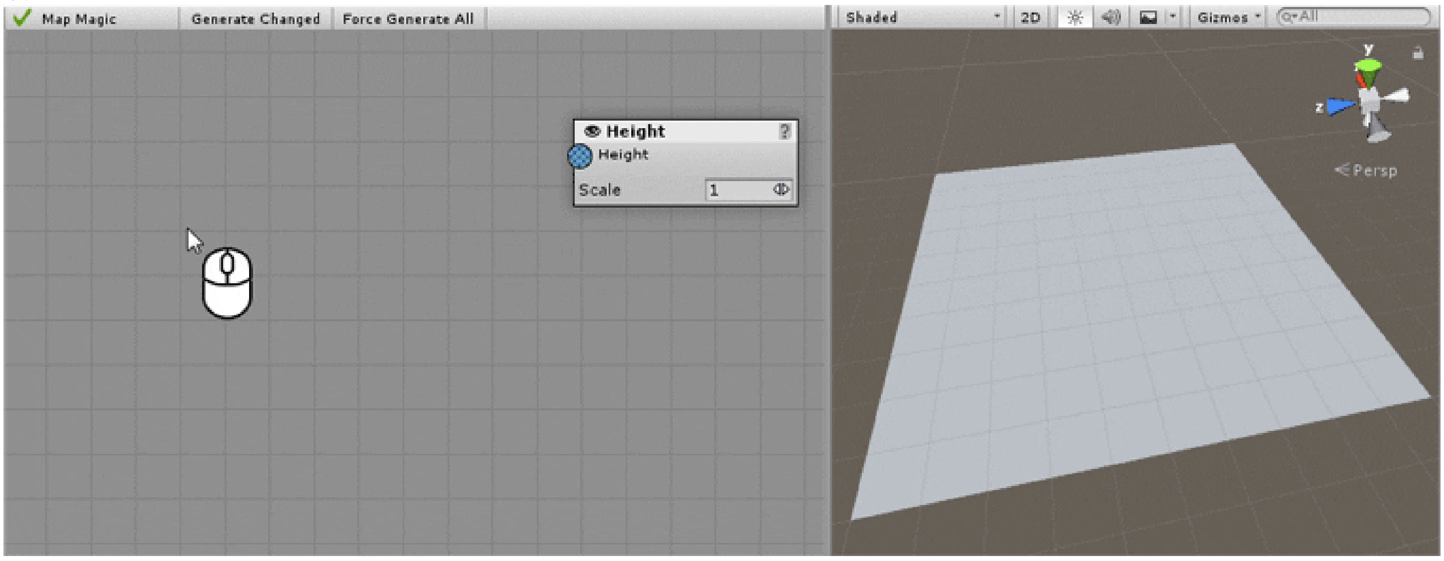
You can click on the field and type the new value the standard way too.

You can see that the resulting terrain is changing too - after a short delay. During this delay the new terrain is calculated and applied. You can see the "Generate" mark in the top left corner of the window meaning that the new terrain is currently calculated.

The generator could be removed by right-clicking anywhere on it and selecting "Remove" from the context menu. Remove all the 3 generators so we could start from scratch.

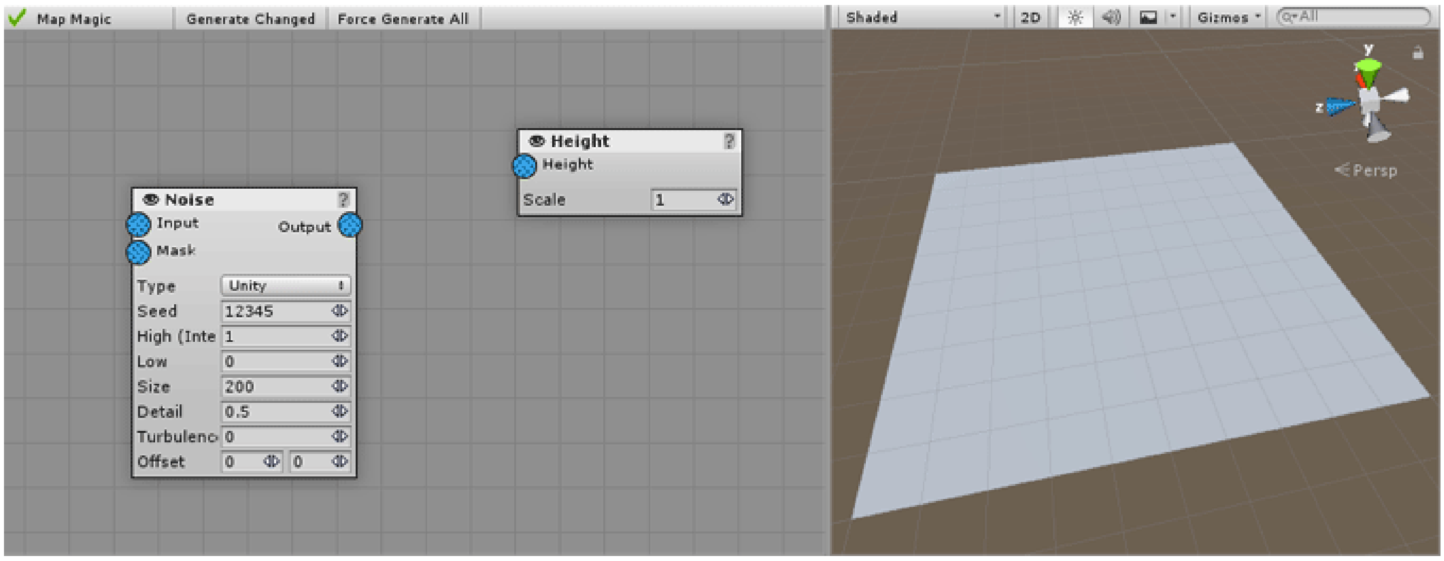


A new generator could be created by right-clicking on background (empty space) and selecting a proper section and generator type in context menu.



Create two generators - similar to those we have removed:

- **Noise**: Create -> Maps -> Noise. This generator creates a fractal noise pattern.
- **Height**: Create -> Output -> Height. This node converts the input map to terrain heightmap and applies it.
  Move created generators to make the graph look this way:

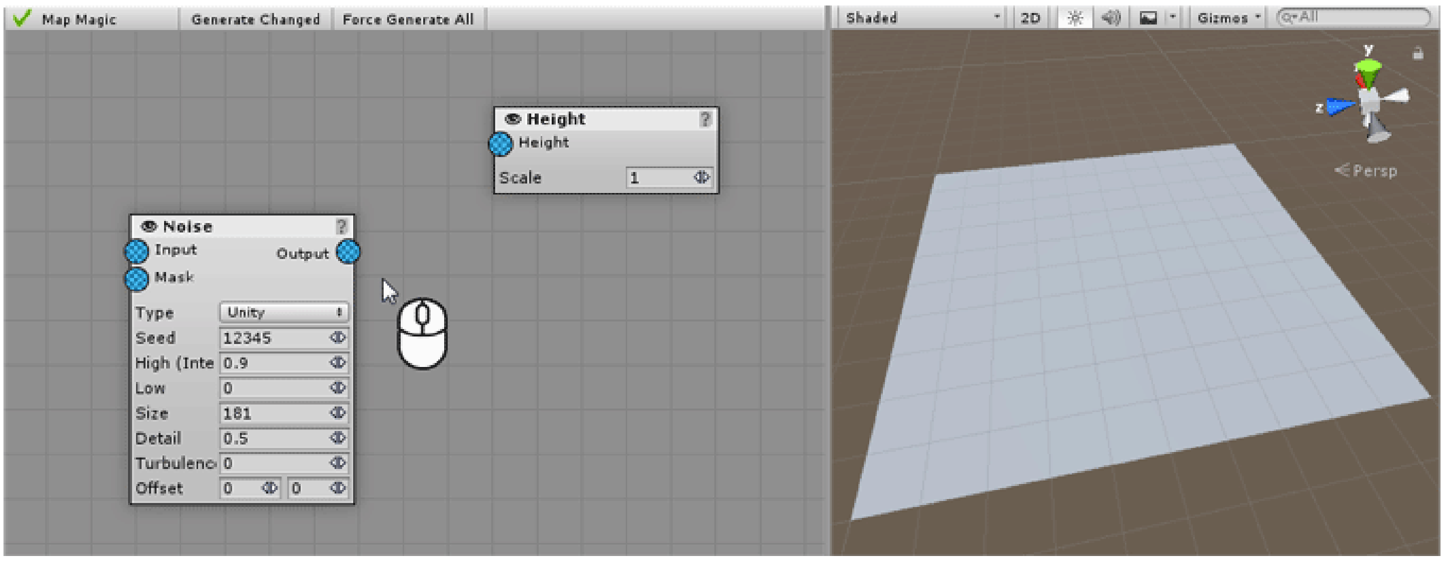


Noise
Input Output
Mask
Type: Unity
Seed: 12345
High (Inte: 1
Low: 0
Size: 200
Detail: 0.5
Turbulenc: 0
Offset: 0 0

Height
Height
Scale: 1

The Noise Generator creates a fractal noise pattern (like this). Consider it's a 2D map. After the Noise Generator has been generated it stores this map in it's output. Now we have to make this output pass to the Height Generator to apply it to terrain.

To do so we have to connect Noise Generator's output to the Height Generator's input by clicking in the blue circle and dragging the mouse:



Noise
Input Output
Mask
Type: Unity
Seed: 12345
High (Inte: 0.9
Low: 0
Size: 181
Detail: 0.5
Turbulenc: 0
Offset: 0 0

Height
Height
Scale: 1

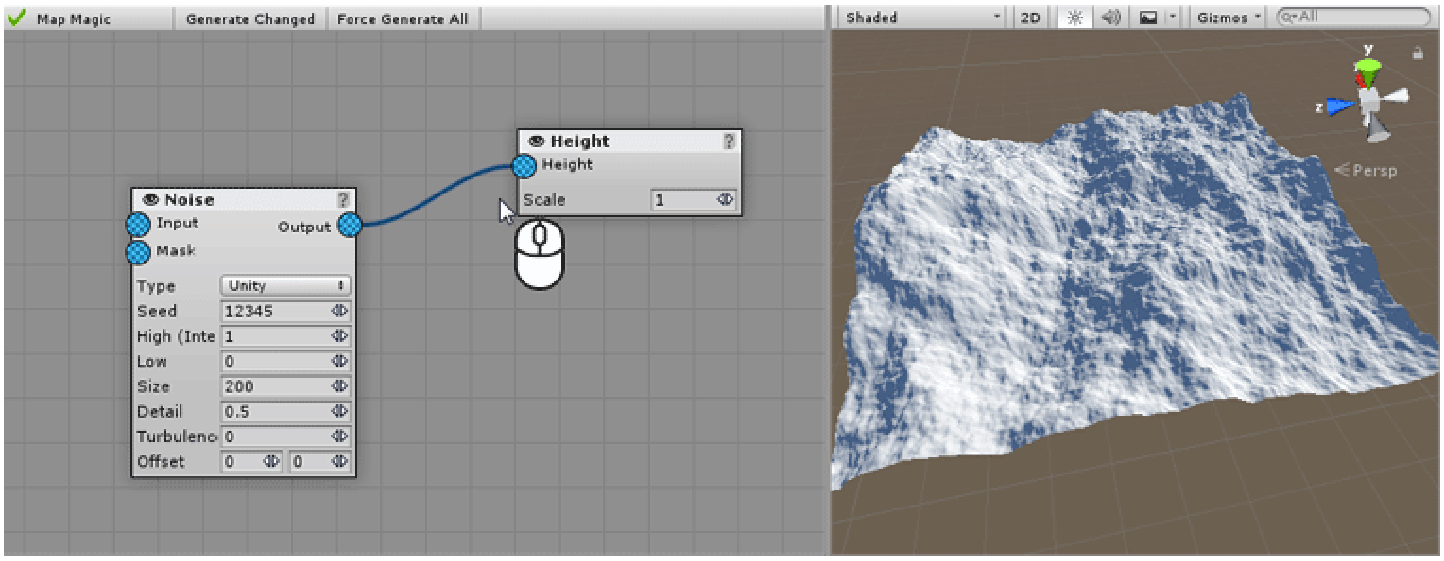
You can see that the noise effect applied to terrain.

> You can also click at the Height's input circle and drag the line to the Noise's output with the same effect.

You can see that the generators are now linked with a blue line. Blue line connects only blue inputs/outputs, blue color symbolize a "map" input/output type: ●. In contrast to blue connections you may see the green lines that connect green inputs/outputs: ●. Green means an "object list" connection. You cannot connect green and blue inputs together since their types are incompatible - to convert an objects list to a map (or vice versa) a special generators should be used.

> Technically, "maps" are not actually textures - they use a special Matrix class that use an array of floats internally. Neither "Object lists" are lists, but spatial hashes.

To unlink the generator you should connect it to void. Click at it's input and drag it to the empty space:



| | |
|---|---|
| Type | Unity |
| Seed | 12345 |
| High (Inte | 1 |
| Low | 0 |
| Size | 200 |
| Detail | 0.5 |
| Turbulenc | 0 |
| Offset | 0 0 |

> **Clicking and dragging an output to an empty space will have no effect.** This is caused by MM's concept: any output can have any number of connections, but any input can be connected only to one output. So, connecting an input to void just unlinks it.

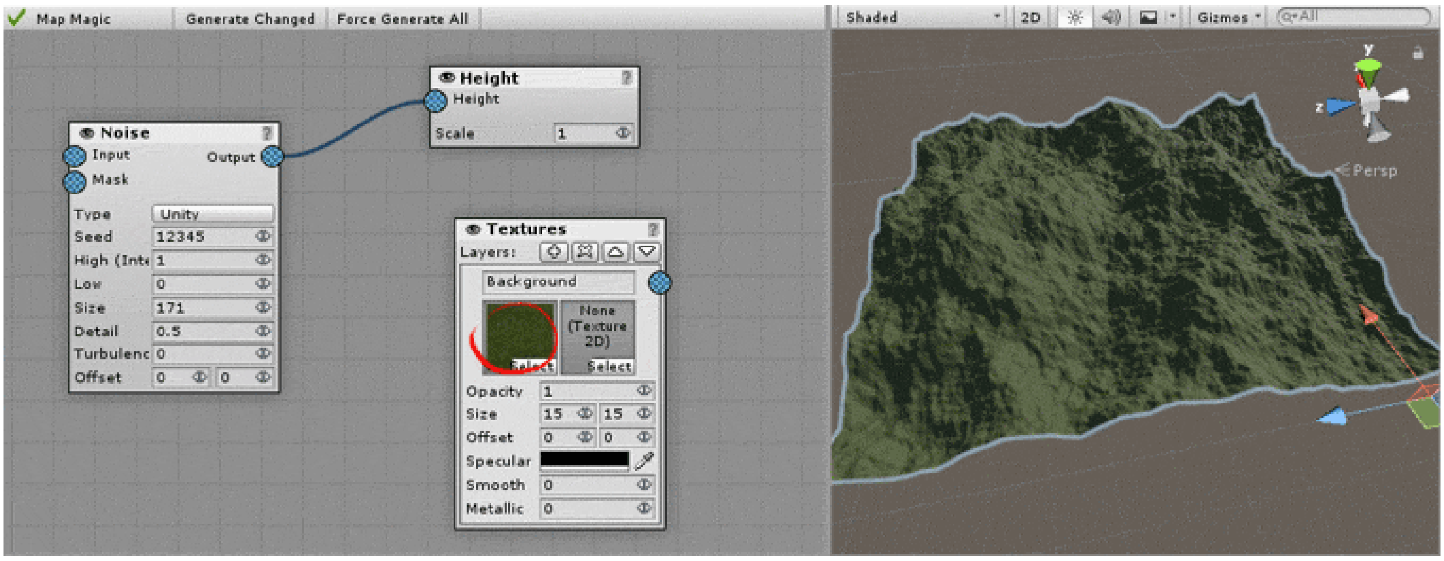
## Part 3: Textures

All of the generators are grouped together under the Create Menu using this logic:

- Map: these nodes work only with 2D maps.
- Object: generators that receive or output object lists. The easy way to find out if generator is in Maps or Objects category is to question yourself if it deal with objects in any way. If id does - then it will be in Objects list, otherwise look for it under the Maps.
- Outputs: a special generators that apply all of the terrain. Unless any of the output generators is not connected MapMagic will not even start to work. These are the main generators that trigger the linked nodes. Their number is quite limited:
  - Height: used to generate height. Receives a map, perceives it like a heightmap, converts it to float[,] format and assigns to terrain.
  - Textures: used to paint terrain with textures. Receives several maps (supposed to be control texture masks) blends them together and applies along with the assigned prototypes (i.e. terrain textures and normalmaps).
  - Grass: to apply grass.
  - Objects: places game objects (transforms) on terrain according to the "object list" input.
  - Trees: same as Objects, but uses the Unity terrain trees engine instead of individual transforms.
  - Custom Shader: same as Texture, but works with MegaSplat, RTP, CTS or other custom terrain shader.
  - Voxeland Outputs: work only when MapMagic is used as a Voxeland generator.

> **Voxeland is a voxel-based 3D terrain, and it requires a different approach to terrain creation rather than 2.5D Unity terrains. Do not try to make a standard map with expectation to convert it to Voxeland - this will require a lot of changes to the graph and in some cases will not even be possible.

Add a Textures output: Create -> Output -> Textures:



Noise
Input Output
Mask
Type Unity
Seed 12345
High (Inte 1
Low 0
Size 171
Detail 0.5
Turbulenc 0
Offset 0 0

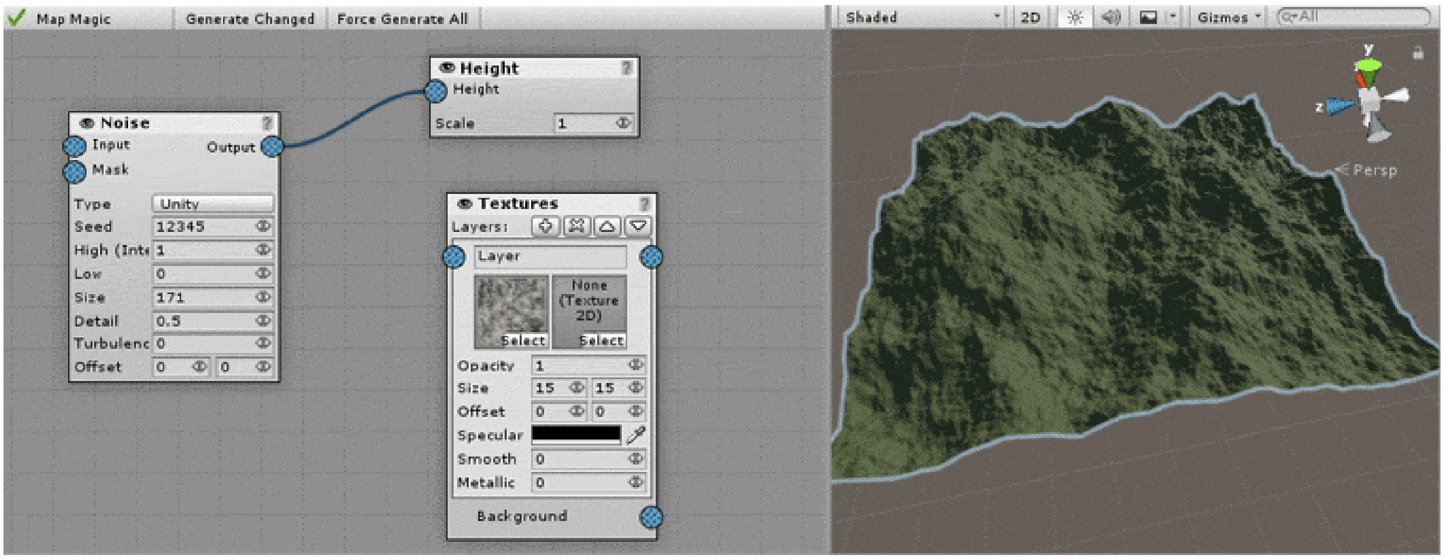
Height
Height
Scale 1

Textures
Layers:
Background
None (Texture 2D)
Opacity 1
Size 15 15
Offset 0 0
Specular
Smooth 0
Metallic 0

Assign a texture in a slot shown on the image - this will fill all the terrain with it.

Textures output has a layered structure. The one we've just added has only one layer called "Background". Each layer corresponds to the texture layer when editing the standard terrain. Each layer has two texture slots - one for the diffuse (albedo) texture, and one for the normal map, and some parameters like texture size, offset, specular color, gloss, etc.

To add a new layer click the ✣ plus button next to the "Layers" label. Assign the other texture to the diffuse slot:

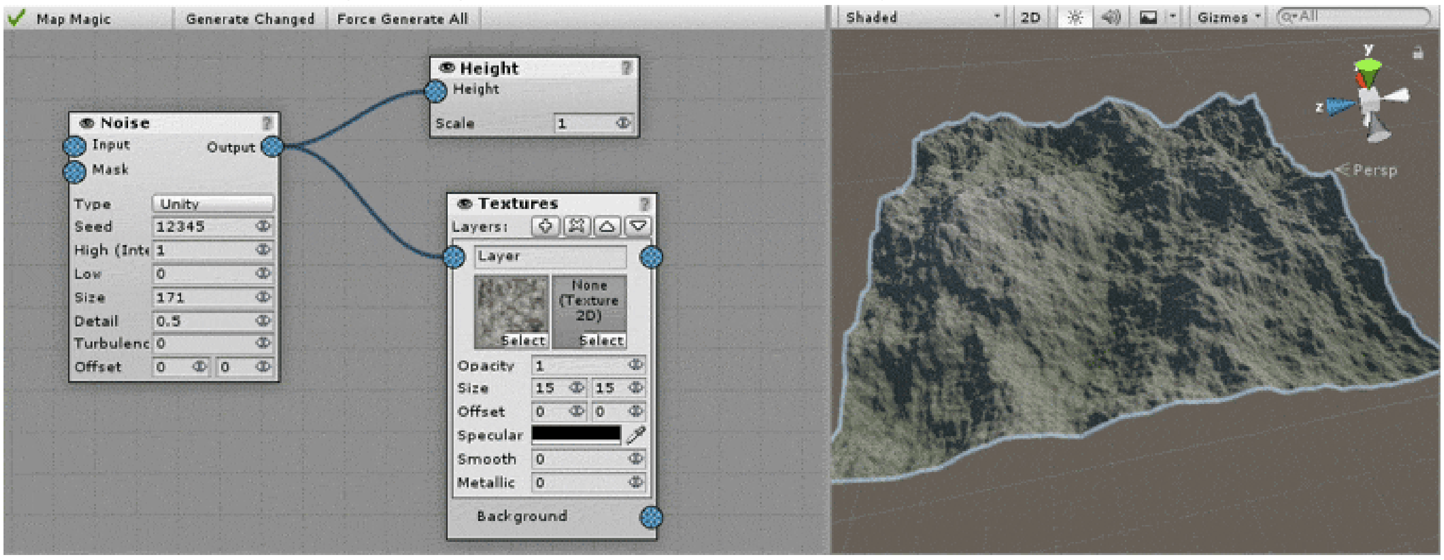


Noise
Input Output
Mask
Type Unity
Seed 12345
High (Inte 1
Low 0
Size 171
Detail 0.5
Turbulenc 0
Offset 0 0

Height
Height
Scale 1

Textures
Layers:
Layer
None (Texture 2D)
Opacity 1
Size 15 15
Offset 0 0
Specular
Smooth 0
Metallic 0
Background

Now try connecting the layer's input to the Noise generator. You can see that the terrain was painted according to the noise map, the same map used for the height: the higher the values, the more visible the layer is:

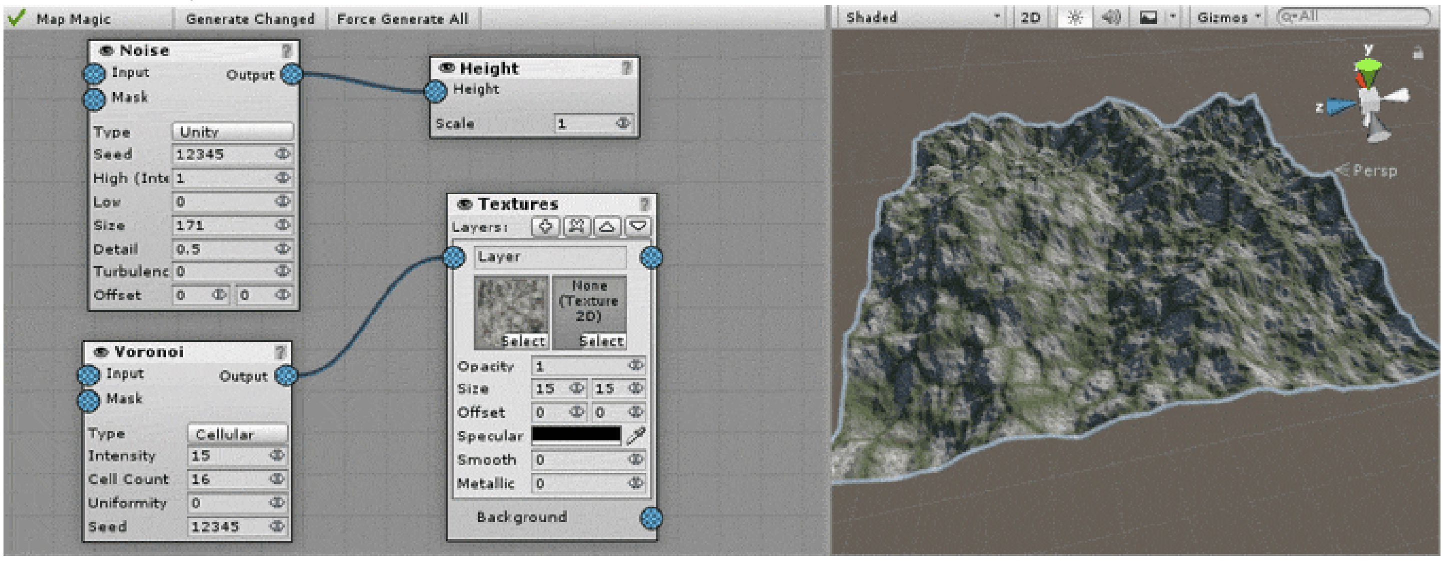


Noise
Input Output
Mask
Type Unity
Seed 12345
High (Inte 1
Low 0
Size 171
Detail 0.5
Turbulenc 0
Offset 0 0

Height
Height
Scale 1

Textures
Layers:
Layer
None (Texture 2D)
Opacity 1
Size 15 15
Offset 0 0
Specular
Smooth 0
Metallic 0
Background

Try creating some absolutely different map like voronoi (Create -> Map -> Voronoi) and connect it to the layer's input. Just raise voronoi's intensity value to 15 to make the effect visible:



Noise
Input
Output
Mask
Type Unity
Seed 12345
High (Inte 1
Low 0
Size 171
Detail 0.5
Turbulenc 0
Offset 0 0

Height
Height
Scale 1

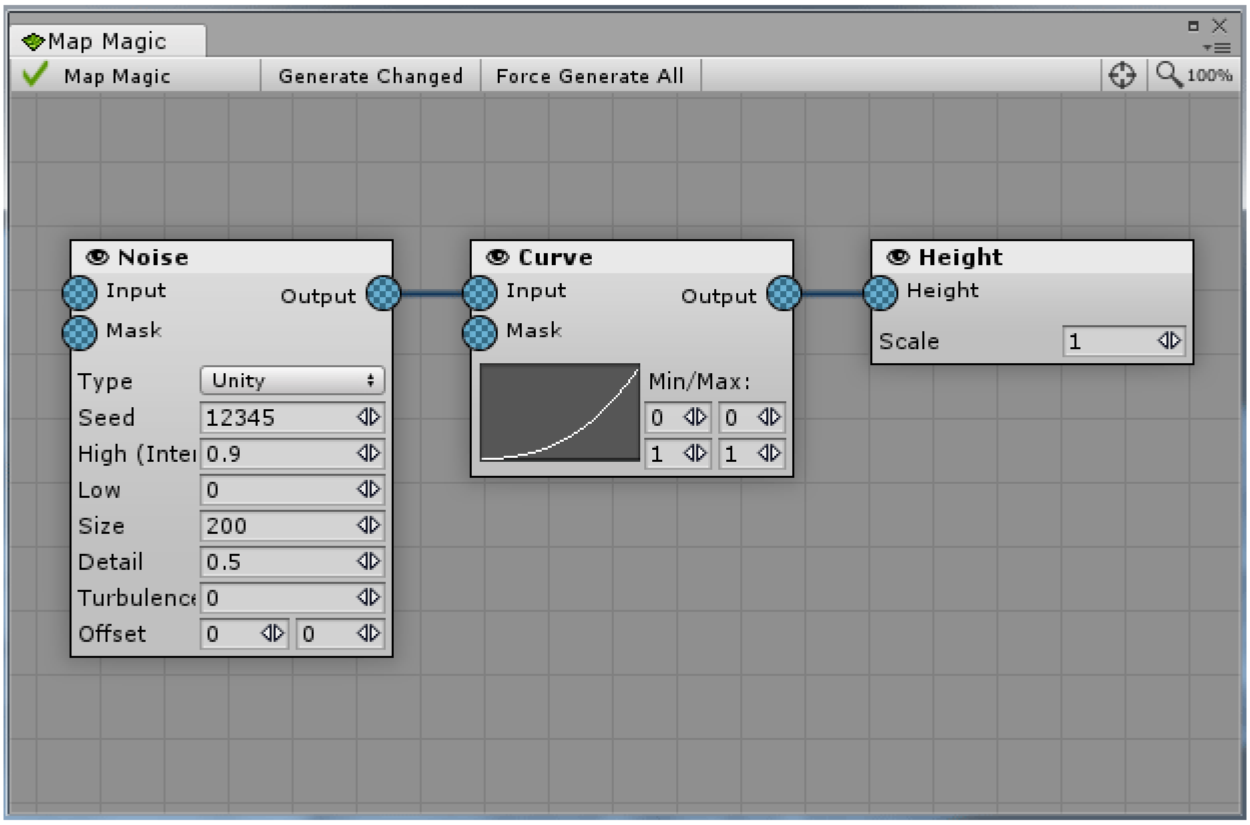
Textures
Layers:
Layer
None (Texture 2D)
Select
Select
Opacity 1
Size 15 15
Offset 0 0
Specular
Smooth 0
Metallic 0
Background

Voronoi
Input
Output
Mask
Type Cellular
Intensity 15
Cell Count 16
Uniformity 0
Seed 12345

Note that Background layer does not have input. It just fill all the other layers' values so that their sum is always 1.

## Editor window

The Editor Window is the main tool for working with MapMagic, so it is recommended to dock it somewhere if you are going to work closely with MM. Try not to make it overlap the Scene View to let you see the generation results. This window could be opened by pressing "Show Editor" button in Inspector when MapMagic, MapMagic data asset or Voxeland are selected.



Type: Unity
Seed: 12345
High (Inte: 0.9
Low: 0
Size: 200
Detail: 0.5
Turbulence: 0
Offset: 0 0

Min/Max: 0 0 / 1 1

Scale: 1

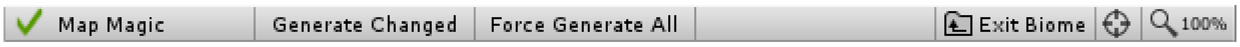
The Editor window can be **scrolled** using the middle mouse button or left mouse button with Alt pressed.

To **zoom** in or out, use the scroll wheel.

To move a generator, left-click and drag anywhere on an individual generator generator window.

> Changing any of the generator's parameters will force the generator and it's dependent generators to re-calculate. Once the generators' results are ready the terrain will be changed.

At the top of the window you can see a toolbar:



From left to right:

- Green tick mark changes its appearance depending on the current generation state:
  - Green tick is displayed when all terrains have been generated successfully and all of them are up-to-date.
  - Grey "Generating" mark (animated) is displayed when at least one terrain is currently in the progress of being generated.
  - Red "Error" mark means that the generator graph has connection error(s): either mandatory connections are empty,

    or connections have the wrong type, or the graph has an infinite loop. Error connections will be displayed in red on graph.
  - When nothing is displayed beside the “Generate” button then terrains have not yet been generated. This is common when the “Instant Generate” feature is turned off in the settings.

- **Map Magic** is the name of the object this data belongs to. Displaying an objects name could be handy when you select a separate data .asset file in project view and edit it or work with multiple Voxeland objects.
- **Generate Changes** button will generate only the changed nodes. This happens automatically when "Instant Generate" is turned on in settings (and it is on by default). But if you'd like to turn "Instant Generate" off for performance or debug purposes use this button to apply all of the changes.
- **Force Generate All** makes all of the nodes re-calculate from scratch. It could be used to fix terrains if something unpredictable occurred. It could help you a lot when writing a custom generator.
- **Exit Biome** button is visible only when you are inside a Biome. Pressing it will return you to the main graph.
- Focus button displayed as the **"target"** icon will focus editor window on the center of your graph. It can help to return editing when you are lost on the infinite graph field.
- Zoom button with a "***magnifier***"* icon will return the zoom value to 100%. On the right of the icon current zoom value is displayed.

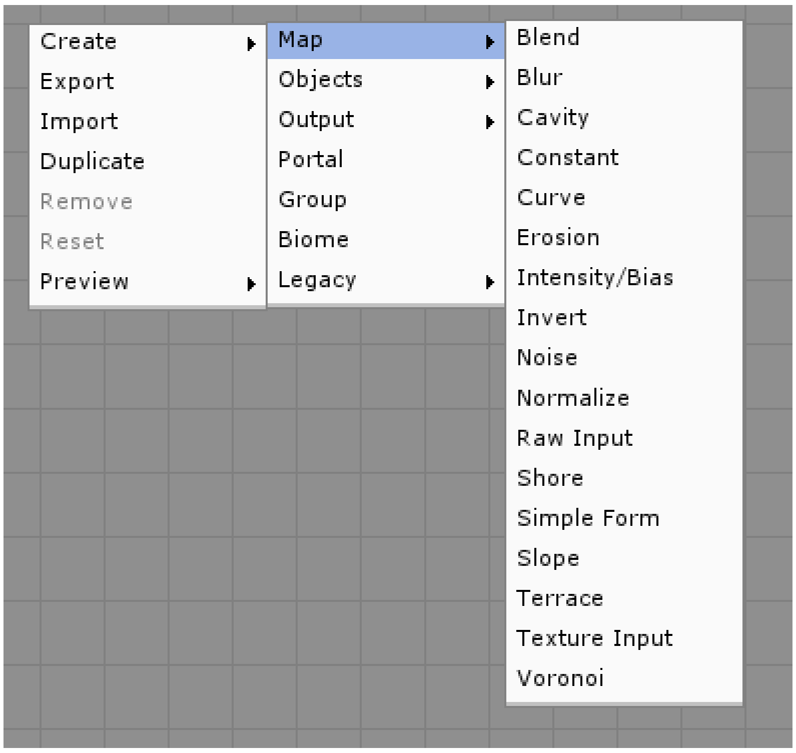
Right-clicking on the empty graph field, generator or the generator output (blue or green circle) will open up a Right-click Menu. With this menu a new generator could be created, exported, removed, previewed, etc.

A newly created generator can be connected to a currently existing generator system. Just drag and drop it’s input or output icons to the other generator’s input or output. Note that the generator’s output can have several connections, while an input accepts only one connection. Some generator inputs are mandatory, that means that they always require some input and generating will fail if they are not connected. Empty mandatory inputs are highlighted with a red asterisk.

>> Right click menu

## Right click menu

Right-clicking on the empty graph, generator or generator output (circle) will open up a Right-click Menu. With this menu a new generator could be created, exported, removed, previewed, etc.



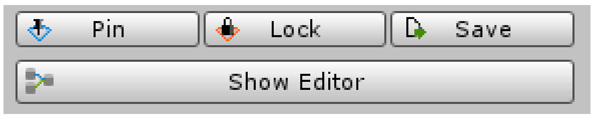
- **Create** will add a new generator node. Note that all of the generators are grouped by type:
  - **Map Generators** connect only with other map generators. Their inputs and outputs are always Maps.
  - **Object Generators** connect only with other object generators. They may or may not have map inputs/outputs, but the Object is always present.
  - **Output Generators** directly set terrains and place objects.
  - The **Portal** is not actually a generator but a connection technique. Don't let the name fool you - it is not about spatial holes to other terrains or dimensions, it's just a way to better organize the graph better. The portal creates a 'wireless' connection when set as an input, that can be paired with a matching output portal that feeds into another place on your graph, minimizing the amount of long cluttered wires displayed.
  - With the help of the **Group** several generators node could be grouped together to quickly move or delete them. Group can be used to annotate some generator blocks too. To engroup the generator simply drag it into the group (so that it does not sticks out).
  - **Biome** generator contains the biome data with it's own output generators.
  - **Legacy** generators list contains all of the generators that were used in previous versions of MapMagic. Creating new ones is not recommended, but in some cases it could be handy to make something compatible with already existing graphs.

- **Export** will store the generator (group) in an XML-like .nodes file so it could be imported with an Import feature. It will export a single generator if clicked on a generator, a group with all the contained generators if clicked on group, or all of the generators if clicked on the empty field.
- **Import** will import exported generators, appending them to currently existing graph.
- **Duplicate** will create a copy of the generator or a group just below. This feature is disabled if clicked on empty field.

- **Remove** will delete the generator or a group. This feature is disabled if clicked on empty field. If clicked on group it will ask if you want to delete all of the contained generators with a popup window.
- **Reset** will reset all of the generator values, returning them to the default ones. Note that this will un-link this generator from the graph.
- Any output from any map generator can be viewed using the **Preview** feature.
    - **On Terrain** will color the terrain in a red to green range of colors according to the preview map values. High map values are colored green, while low values are red.
    - Moreover, a map can be previewed in a separate window. **In Window** will open up a window with a grayscale map displayed. This map can be scrolled and zoomed the way the main graph can.
    - To exit preview mode, right-click anywhere in the Editor and select **Clear**.

> Note that Preview works only with a generator output, not the generator itself, since the generator could have several outputs. **Make sure you are clicking an output circle, not the node window when trying to use a preview**.

## Settings



In playmode MapMagic generates an infinite terrain - new chunks are created when camera approaches them. In editor you should force generate manually, by using the **Pin** feature. Press the Pin button, select and click the chunk in the scene view - now this chunk will be marked as "pinned", it will always persist in scene (but could be disabled in playmode), and it will be re-generated at all the graph changes.

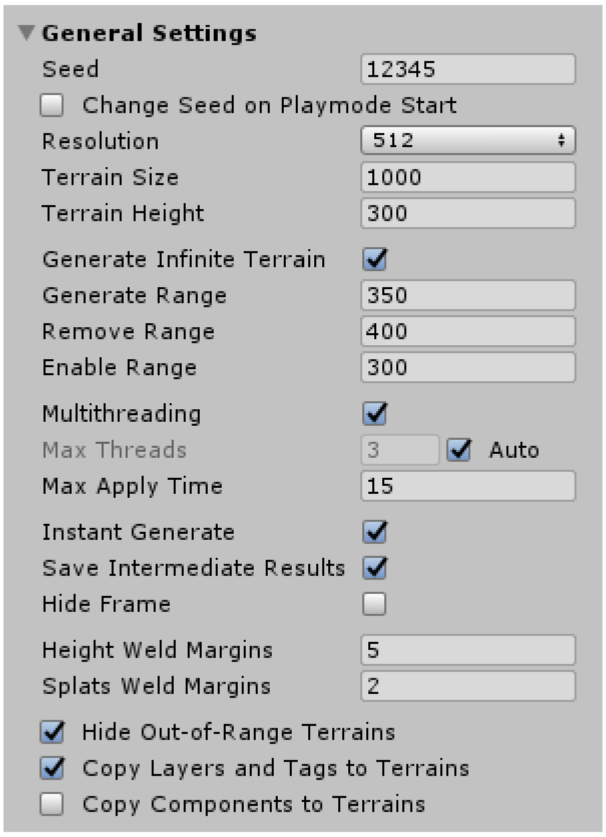
> Pinning terrain is usually used for terrain quality preview or baking. If you are going to use 100 sq km land in your game it's not recommended to pin 10*10 square because of memory, scene size and performance reasons. Pin only 1-4 terrains and let MapMagic generate others in playmode.

Any change made to pinned terrains will be re-written on the next generate. To prevent this, use the terrain lock by pressing the **Lock** button and selecting a pinned terrain that should be locked. When using a locked terrain, try not to modify a height greatly, otherwise you can get noticeable borders between your unchanged, locked terrains and their neighbors that have a new procedural heightmap with whatever changes you've made since locking the previous terrain.

By the default all of the pinned terrains are saved in scene, there is no separate .asset files for them. But in some cases using separate terrain files could be handy - for example when passing some terrains to another project or for reducing the scene size. In this case terrain data could be exported to the standard .asset file by pressing the **Save** button and selecting terrain with a green selection frame.

Show Editor button will open up an Editor Window.

## General Settings


General Settings

| | |
|---|---|
| Seed | 12345 |
| ☐ Change Seed on Playmode Start | |
| Resolution | 512 |
| Terrain Size | 1000 |
| Terrain Height | 300 |
| Generate Infinite Terrain | ✓ |
| Generate Range | 350 |
| Remove Range | 400 |
| Enable Range | 300 |
| Multithreading | ✓ |
| Max Threads | 3 ✓ Auto |
| Max Apply Time | 15 |
| Instant Generate | ✓ |
| Save Intermediate Results | ✓ |
| Hide Frame | ☐ |
| Height Weld Margins | 5 |
| Splats Weld Margins | 2 |
| ✓ Hide Out-of-Range Terrains | |
| ✓ Copy Layers and Tags to Terrains | |
| ☐ Copy Components to Terrains | |

- **Seed**: MapMagic global seed value. This number is used to initialize pseudo-random generators. Note that pseudo-random generators use their own Seed parameter, so the final result depends both on the Global Seed and Generator Seed parameter. Change this value to achieve a different look for all the generators.
- **Change Seed on Playmode Start**: will randomly choose a new seed value once the game/playmode is started. This will generate unique terrain for each game session. Make sure that no terrains are pinned and locked - otherwise they will not be able to weld with the generated terrains properly.
- **Resolution** sets the inputs and outputs map size. Increasing resolution will create a more detailed terrain, but note that generation time will increase quadratically. For example, calculating Erosion Generator - one of the slowest generators - with 5 iterations for the map size of 512 takes about 2 seconds, while the same action for the map size of 1024 takes more than 7 seconds.

> The calculations comparing terrain generate performance show that terrains with size 500 and resolution 512 are generating faster in playmode than terrains with size 2000 and resolution 2048. In first case MapMagic does not generate the areas that are out of range: for instance, with a view range of 800, you will have 3*3 grid of 500/512 terrains within a view range, or 2*2 grid of 2000/2048 terrains. Obviously, the first case is faster. It's not just faster, it's **more than 7 times faster**.
>
> It is recommend using the smaller chunks (and resolution) as possible as long as you can avoid noticeable seams of the erosion, blur or other generators that are masked at the chunk edges. The resolution of 512 seems to be a good balance between the generate speed and the visual quality. Even if you are baking terrain and not using MapMagic in playmode - consider using more low-resolution terrains just to reduce the baking time.

- **Terrain size** sets the length of a terrain’s side in units. This parameter just changes the scale of the terrain object and does not affect performance. Note that changing the size parameter requires terrain height adjustments to make the terrain look proportional.
- **Terrain height** sets the maximum terrain elevation, i.e. map pixels with values of 1 will set terrain at this level.
- **Generate Infinite Terrain** - when enabled, this dynamically generates new terrains in Play mode when the main camera comes close to the edge of the current terrain (based on your range settings), creating a virtually infinite world. Infinite

terrain ranges are measured using ***max(abs(x), abs(z))*** algorithm, so they form a square-size perimeters.
  - **Generate Range** - the minimum distance to the terrain border that triggers terrain generation. Note that the terrain will not appear immediately when the Generate Range is reached, as it will take some time to generate, depending on the graph generator's complexity. This parameter ensures that all of the terrains within range are generated or started generating.
  - **Remove Range** - the chunks that are further than this distance from camera are removed to free up memory. It's recommended to use some buffer distance between the generate and remove ranges to prevent the terrain re-generate when the player will walk some steps back where he came from.
  - **Enable Range** - the chunks that are further than this distance are disabled. The closer chunks are enabled if they are generated. Used to hide far chunks to gain performance without actually destroying them. Does not work when "Hide out of range terrains" is off.

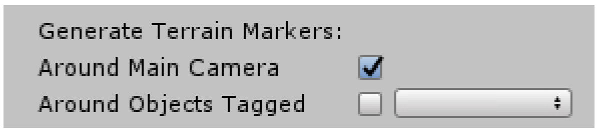
- **Multithreaded**: when enabled, Map Magic generate works in a several separate threads. Turning this parameter off forces Map Magic to work in the main thread only using co-routines only (this could be useful for debugging of custom generators or for compatibility reasons).
  - **Max Threads**: the number of threads MapMagic uses to generate maps. The recommended value is number of current processor threads -1.
  - **Max Threads Auto**: sets the number of threads to processor thread count - 1. This way, on the different hardware the number of threads will be different. Auto mode works both in editor and in build.
  - **Max Apply Time**: Although generate happens in other threads, applying generated results to the terrain, the final stage, is done using co-routines. This value sets the maximum time spent per frame by co-routine in milliseconds. Increasing this value will shorten the overall apply time, but can create a noticeable lag. Note that in some cases co-routine could not be split in smaller parts, so setting this value to 1 will not help to avoid some profiler spikes.

- **Instant Generate**: when this parameter is enabled there is no need to press the "Generate" button after changing any of the generator parameters: Map Magic triggers the change and forces generation automatically as you make changes to each node in the graph.
- **Save Intermediate Results**: Map Magic keeps the output results for each terrain for each generator, so on generator change there is no need to re-generate everything: only the changes in the generator and its dependences are calculated. When the parameter is disabled MapMagic will clear all generator maps after generation has been completed. Disabling this option can reduce the memory footprint, but each time any of the generators change the entire graph will be re-generated from scratch, which can take some time depending on your graph's complexity.
- **Hide Frame**: hides Unity selection frame/highlight (depending on Unity version) when MapMagic object is selected. This can avoid the "wired" look of all of the objects placed on terrains.
- **Weld Margins**: Since all of the terrains are generated independently some algorithms like blur or erosion could give different results on a terrain's borders. To get rid of this difference Map Magic creates a smooth transition from one terrain to another. The Margins parameter is the size of the transition in pixels. When the parameter is 0 then terrain welding is off. Note that it's not a panacea for terrain welding - in most generators you will have to use "Safe Borders" to generate more or less close results at the borders.
  - **Height Weld Margins**: margins applied for heightmap welding
  - **Splat Weld Margins**: margins that are applied for terrain texturing welding. Since texturing differences are less noticeable than height differences, it is recommended to set this parameter lower than Height Weld Margins.

> Keep in mind that using high margins values results in longer apply time (i.e. the time when Unity freezes after generating) and can cause floating objects bugs (when terrain is lowered under the objects to perform the weld). Keep these values as low as possible and use the Safe Borders parameter to negate the terrain difference.

**Hide Out of Range Terrains**: hides pinned terrains if they are out of the generate range, otherwise they are always displayed. Disabling this can give the effect of terrains floating in the air and reduces performance. Use "Enable Range" value to set the hide range.
- **Copy Layers and Tags to Terrains**: when turned on Map Magic will copy all of the tags and layers assigned to the Map Magic object to terrains on generate.
- **Copy Components to Terrains**: when turned on Map Magic will copy all the scripts assigned to the Map Magic object (except MapMagic itself) to terrains on their generate.

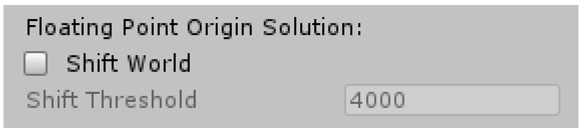
## Generate Terrain Markers


Generate Terrain Markers:
Around Main Camera ✓
Around Objects Tagged ☐

By the default the main camera is used as a trigger object to generate new chunks. All the generate, remove and enable ranges are counted from the main camera, it is the main generate marker. You can add the other markers enabling **Around objects tagged** and selecting a proper tag from the field. This could be useful if you have several characters that could have a vast distance between them.

Moreover, you can turn off using camera as a generate marker by disabling **Around Main Camera** - for example, if you camera moves between those characters quickly and you don't want to generate terrain while in motion.

## Floating Origin Solution

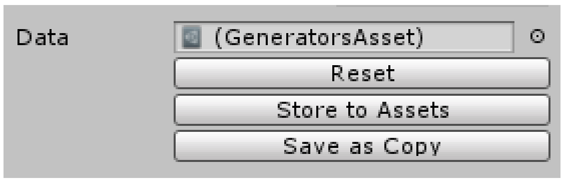

Floating Point Origin Solution:
☐ Shift World
Shift Threshold 4000

One of the problems related with the vast or infinite worlds lies in floating point precision limitation. When camera is placed far from the zero coordinate lighting, shadows and z-fighting problems could be experienced. There's a solution called Floating Origin that can help to solve that. It places the camera closer to origin when it it exceeds some threshold from zero coordinate. All of the world objects are placed relatively along with the camera so that player does not notice that shift had happened.

MapMagic already has Floating Origin integrated, so it safely offsets camera and MapMagic object. Since MM works in multiple threads and co-routines this offset can happen during the terrain generate - in this case MapMagic ensures that the generated terrains appear at the right position, with the objects scattered properly.

To turn on Floating Origin feature enable the **Shift World** toggle. Set the **Shift threshold** to the offset step, so that the camera will not exceed the double of this distance (for threshold of 4000 maximum camera position is 8000 on any axis).

## Data

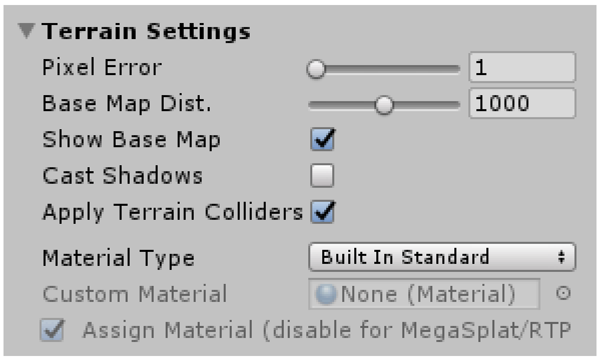

Data (GeneratorsAsset)

By the default the graph you are creating is stored in the scene where MapMagic object is saved. But you can export the graph to use it in another scene or in a biome.

You can assign a saved data in **Data** slot - this will replace your current graph with an assigned one. Re-selecting a MapMagic object is required for changes to take effect.

- **Reset** button will clear current graph and replace it with a blank one;
- **Store to Assets** will export current graph as an .asset file and assigns the stored scene in a data slot. All the further graph changes will cause changing of the stored asset.
- **Save as Copy** will duplicate current graph and save it as an .asset file. Changing the graph will not affect the stored asset.

## Terrain Settings

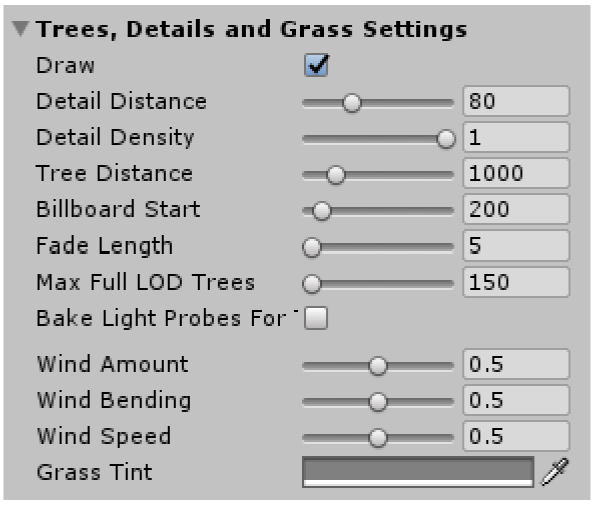

Terrain Settings
Pixel Error 1
Base Map Dist. 1000
Show Base Map ✓
Cast Shadows ☐
Apply Terrain Colliders ✓
Material Type Built In Standard
Custom Material None (Material)
✓ Assign Material (disable for MegaSplat/RTP

Are equivalent to the settings used for each terrain in the standard terrain's Settings tab. The only difference is that they are applied to all of the MapMagic terrains.

- **Pixel Error**: The accuracy of the mapping between the terrain maps (heightmap, textures, etc) and the generated terrain; higher values indicate lower accuracy but lower rendering overhead. Unlike the standard terrain setting you can set the value to 0 to force the terrain render full-resolution mesh.
- **Base Map Distance**: The maximum distance at which terrain textures will be displayed at full resolution. Beyond this distance, a lower resolution composite image will be used for efficiency.
- **Show Base Map**: toggles if base map will be enabled at all.
- **Cast Shadows**: Does the terrain cast shadows?
- **Apply Terrain Colliders**: disabling will turn off terrain and tree colliders for all of MapMagic chunks
- **Material Type**: The material used to render the terrain. This will affect how the color channels of a terrain texture are interpreted.
    - **Built In Standard**: This is the PBR (Physically-Based Rendering) material introduced in Unity 5.0.
    - **Built In Legacy Diffuse**: This is the legacy built-in terrain material from Unity 4.x and before.
    - **Built In Legacy Specular**: This built-in material uses BlinnPhong (diffuse and specular term) lighting model and has optional normal map support.
    - **Custom**: Use a custom material of your choice to render the terrain. This material should use a shader that is specialised for terrain rendering (e.g. it should handle texture splatting properly).

- **Custom Material** assign a custom material in this slot.
- **Assign Material**: If you are using Custom Shader Output then the material assigned here is not applied to the terrain directly, but is used as a template. It is copied for each chunk, the copy has it's unique per-terrain control textures assigned, and then the copy applied to terrain. To switch to the template mode disable this toggle.

## Trees, Details and Grass Settings



| | |
|---|---|
| Draw | ✓ |
| Detail Distance | 80 |
| Detail Density | 1 |
| Tree Distance | 1000 |
| Billboard Start | 200 |
| Fade Length | 5 |
| Max Full LOD Trees | 150 |
| Bake Light Probes For | ☐ |
| Wind Amount | 0.5 |
| Wind Bending | 0.5 |
| Wind Speed | 0.5 |
| Grass Tint | |

Are equivalent to the settings used for each terrain in the standard terrain's Settings tab. The only difference is that they are applied to all of the MapMagic terrains.

- **Draw**: Should trees, grass and details be drawn?
- **Bake Light Probes For Trees**: If this option is enabled, Unity will create internal Light Probes at the position of each tree and apply them to tree renderers for lighting. Otherwise trees are still affected by light probe groups.
- **Detail Distance**: The distance (from camera) beyond which details will be culled.
- **Detail Density**: The number of detail/grass objects in a given unit of area. The value can be set lower to reduce rendering overhead.
- **Tree Distance**: The distance (from camera) beyond which trees will be culled.
- **Billboard Start**: The distance (from camera) at which 3D tree objects will be replaced by billboard images.
- **Fade length Distance over which trees will transition between 3D objects and billboards.
- **Max Mesh Trees**: The maximum number of visible trees that will be represented as solid 3D meshes. Beyond this limit, trees will be replaced with billboards.
- **Wind Speed**: The speed of the wind as it blows grass.
- **Wind Size**: The size of the "ripples" on grassy areas as the wind blows over them.
- **Wind Bending**: The degree to which grass objects are bent over by the wind.
- **Grass Tint Overall color tint applied to grass objects.

## Mass Pin

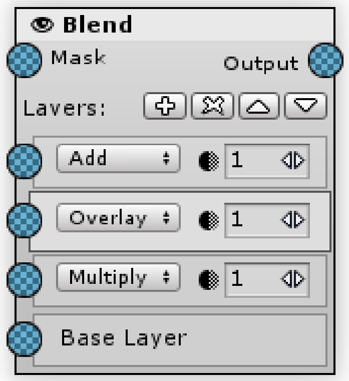
Bunch pin or lock of the terrains. Select a rectangle using the offset and size and press pin or lock.

> **Rectangle size is in MM chunks, not the world units.**
>
> **This feature is designed to be used with streaming plugins. Using it in all other purposes is not recommended because of performance reasons.**

## About

Plugin summary info. You can look for the current version here.

# Blend

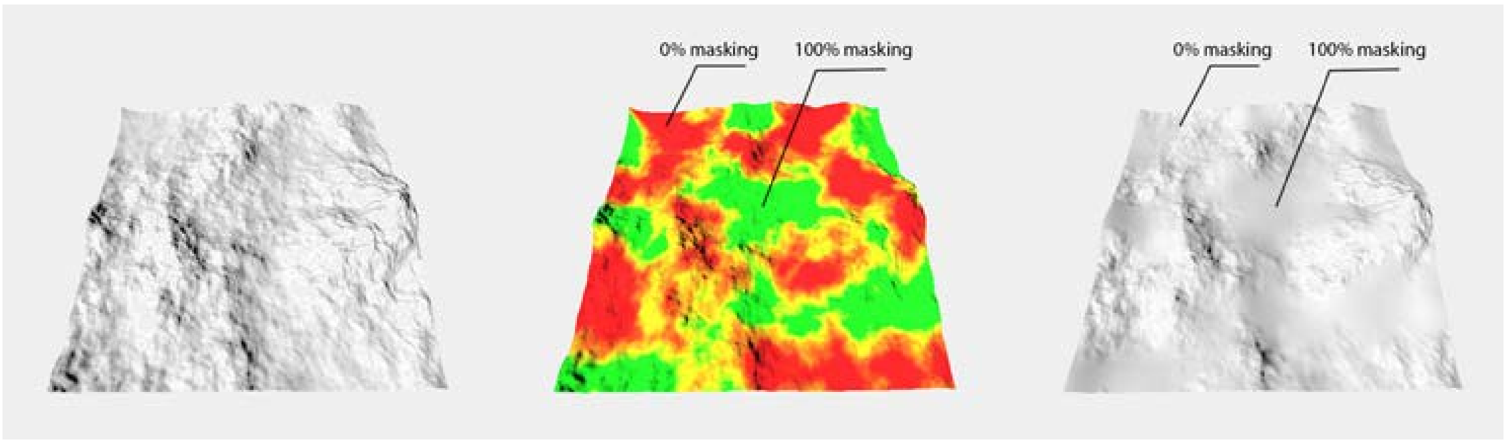


Blends two or more maps together using the specified blend algorithm. This generator acts quite similar to the layer blending mode in Photoshop or other graphics editor. Think of the "Base" map input as a background layer, and the layers above are the blending layers.

Some generators like noise or voronoi have their own blend inputs: additive (Input) and multiplicative (Mask). In most cases it is enough to use them, but in some cases you might need the other blend type or need to blend generator chains or generators that do not have blend inputs. Blend generator was created to solve that.

## Inputs:

- ● Blend input in all of the blend layers (including Base layer).
- ● Mask - multiplies the generator intensity by the mask map's value. For the mask pixels that have a value of 0 the generator effect is invisible, for the pixels that have a value of 1 the intensity is unchanged.



*Mask, using the example of **Blur** Generator:*
*| Source heightmap | Mask map | Masking result |*

## Outputs:

- ● Output - the blended result

## Properties:

Blend Generator input maps are stacked using the layers mode. Each layer could be selected by clicking on it. Above the layers, next to the "Layers:" label you can see layer control buttons:

- Add: will add new layer atop of the selected one.
- Remove: removes currently selected layer.
- Up: will change the layers order by moving the selected layer up.
- Down: will change the layers order by moving the selected layer down.

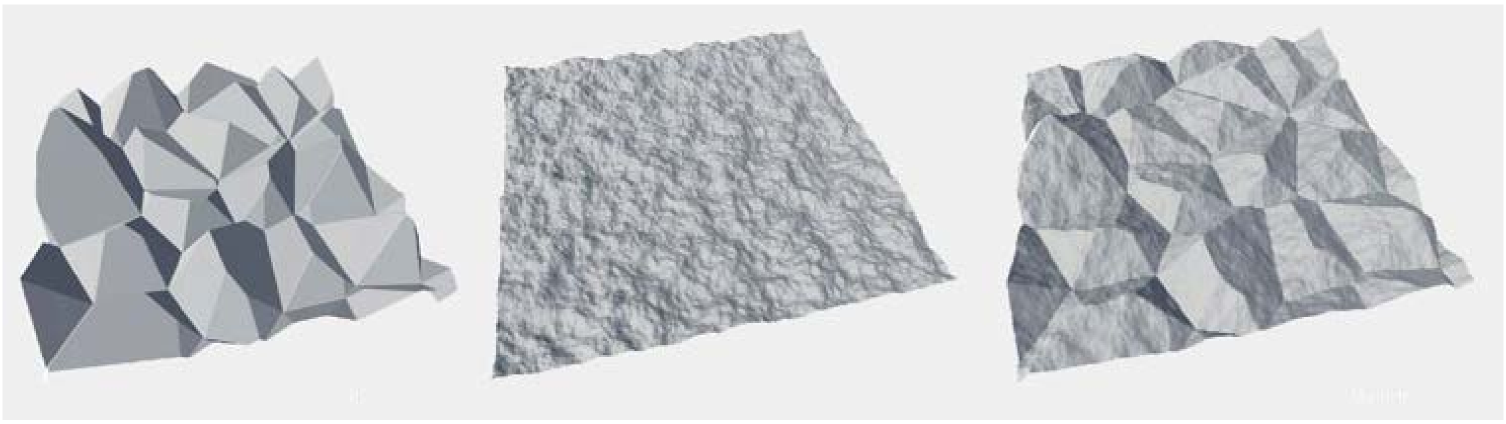
Base layer could not be removed or sent up or down.

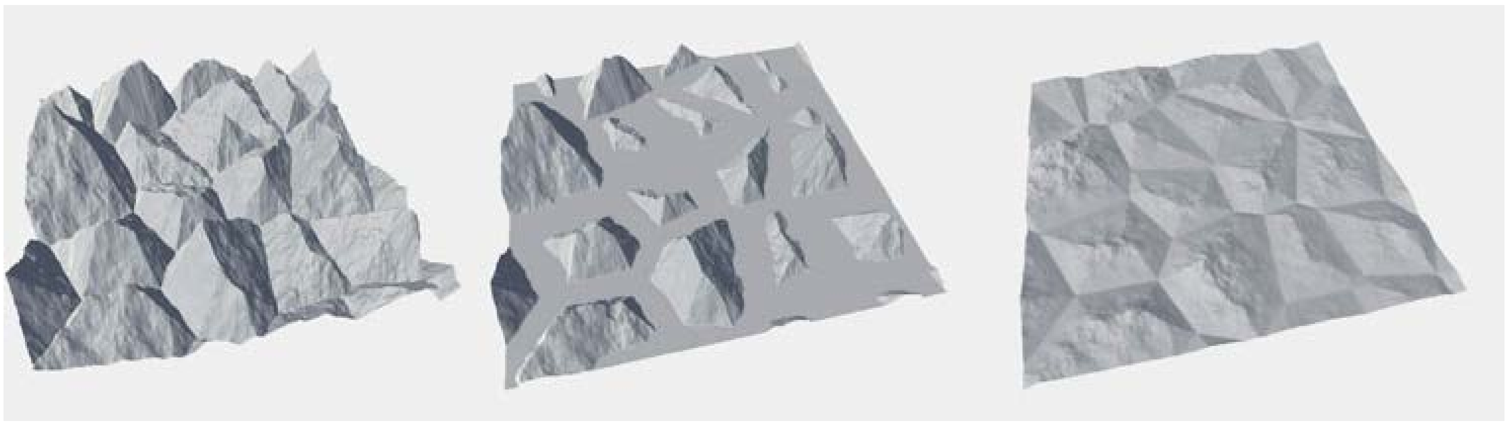
Each layer (except the Base one) has the algorithm and opacity properties.

- Algorithm - determines how exactly the layers are blended
- Opacity - the blending layer opaqueness factor. If the value is 0 the blending result is not visible.

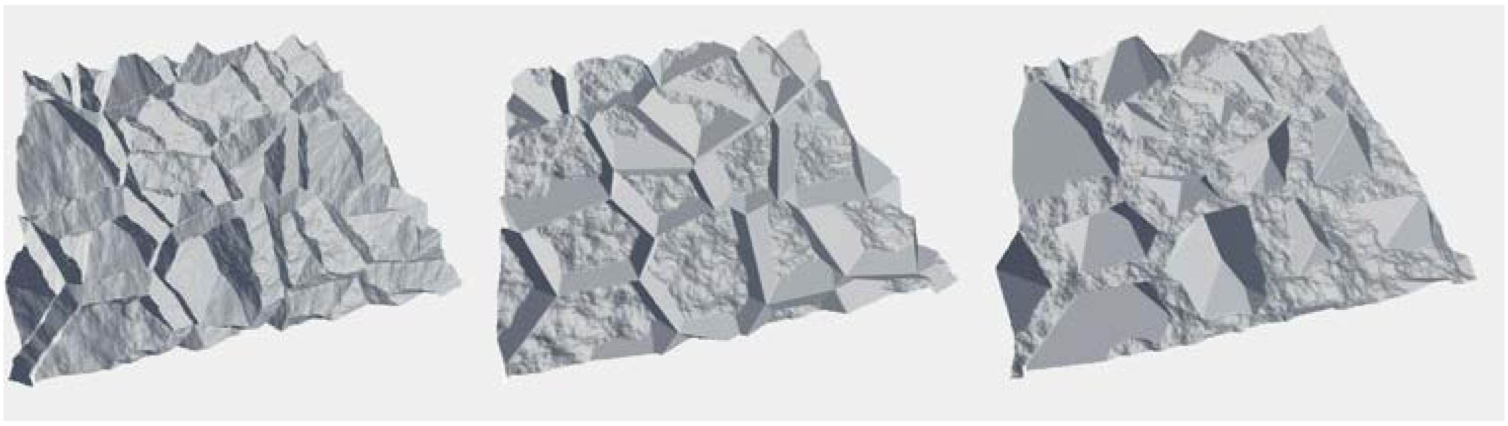
Here are the algorithms the blend generator offers:

- Mix: simply blends two maps using the opacity value Algorithm's formula is { return b };
- Add: adds the Blend layer to the Base layer, i.e. the sum of the layers { a+b; }
- Subtract: subtracts the Blend layer from Base layer { a-b; }
- Multiply: multiplies the two layers. Note that since input values in most cases are less than 1, the multiplied value would be lesser than any of the input ones. { a*b; }
- Divide: the opposite of Multiply. The result can often exceed 1, so use it with care. { a/b; }
- Difference: the delta value between two layers { Mathf.Abs(a-b); }
- Min: the minimum value between both layers. { Mathf.Min(a,b); }
- Max: the maximum value between both layers. Try experimenting with Min and Max blend algorithms as they can bring clean and useful results { Mathf.Max(a,b); }
- Overlay: this effect is often used to process images but the feasibility of using it for blending height or splat maps is questionable.
  if (a > 0.5f) return 1 - 2*(1-a)*(1-b);
  else return 2*a*b;


*Base layer: voronoi, Blend layer: Noise, Mix algorithm with opacity 0.5*



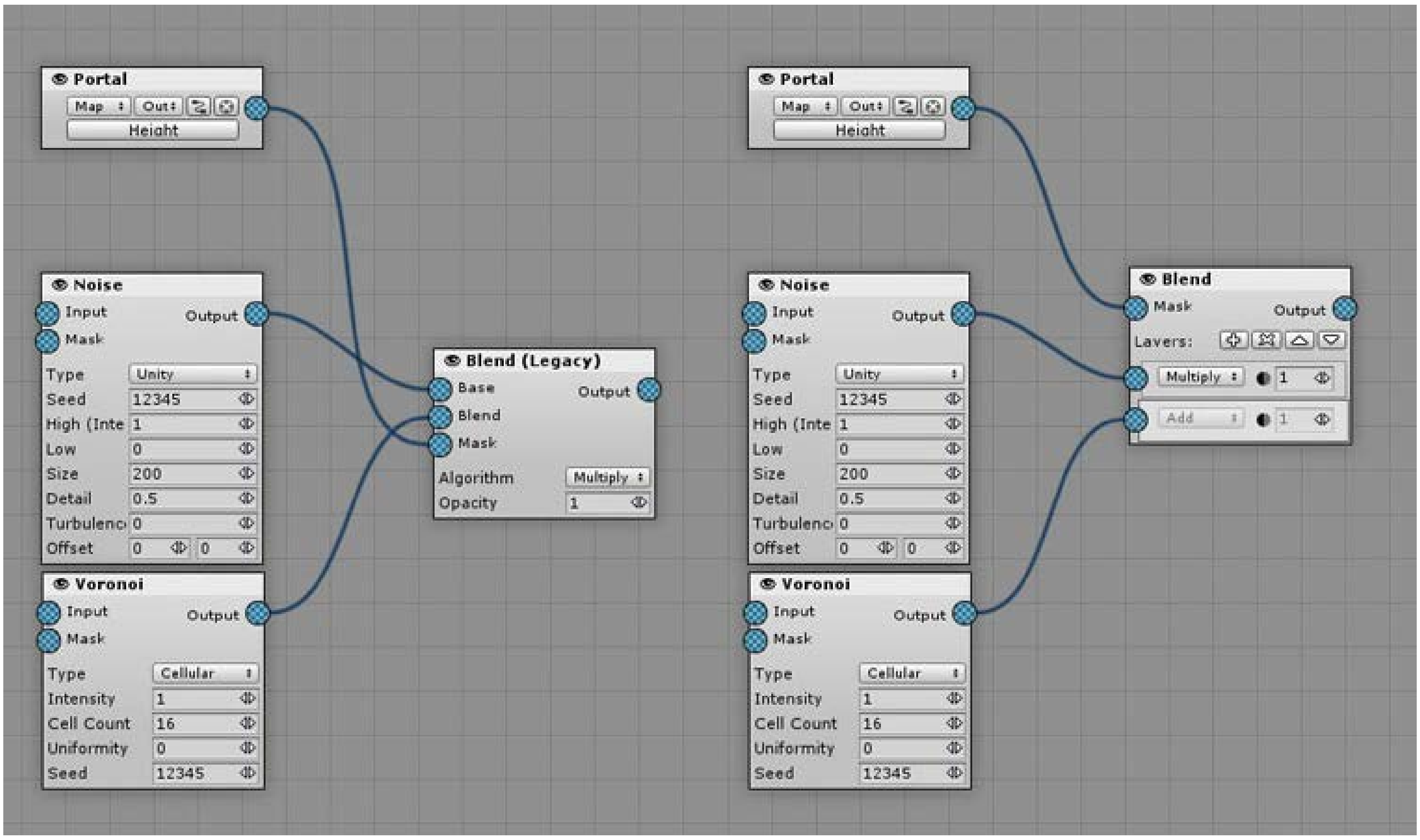
*Algorithm: Add, Subtract, Multiply*



*Algorithm: Difference, Min, Max*

Base layer opacity is always 1, and the algorithm is always additive.

In earlier versions of MapMagic Blend Generator had only two layers. If you are looking through MapMagic tutorial you can see the the legacy Blend - note that it works absolutely the same way the new Blend does, and could be replaced without any issues:


Portal
Map | Out
Height

Noise
Input
Mask
Output
Type: Unity
Seed: 12345
High (Inte: 1
Low: 0
Size: 200
Detail: 0.5
Turbulenc: 0
Offset: 0 0

Voronoi
Input
Mask
Output
Type: Cellular
Intensity: 1
Cell Count: 16
Uniformity: 0
Seed: 12345

Blend (Legacy)
Base
Blend
Mask
Output
Algorithm: Multiply
Opacity: 1

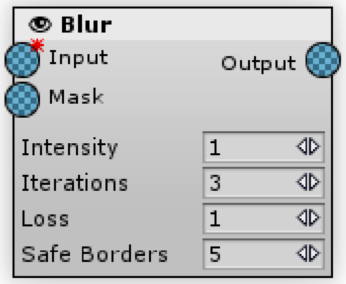
Portal
Map | Out
Height

Noise
Input
Mask
Output
Type: Unity
Seed: 12345
High (Inte: 1
Low: 0
Size: 200
Detail: 0.5
Turbulenc: 0
Offset: 0 0

Voronoi
Input
Mask
Output
Type: Cellular
Intensity: 1
Cell Count: 16
Uniformity: 0
Seed: 12345

Blend
Mask
Output
Layers:
Multiply 1
Add 1

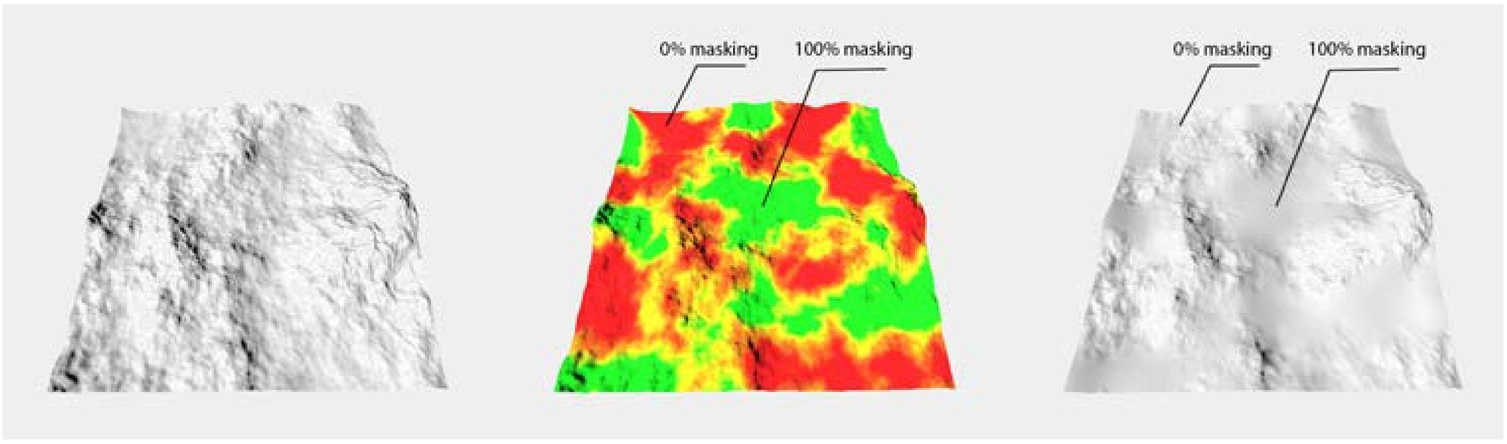
# Blur


Blur
Input
Output
Mask
Intensity 1
Iterations 3
Loss 1
Safe Borders 5

Smooths the input map.

## Inputs:

- ● Input - the default map to be processed by the generator.
- ● Mask - multiplies the generator intensity by the mask map's value. For the mask pixels that have a value of 0 the generator effect is invisible, for the pixels that have a value of 1 the intensity is unchanged.



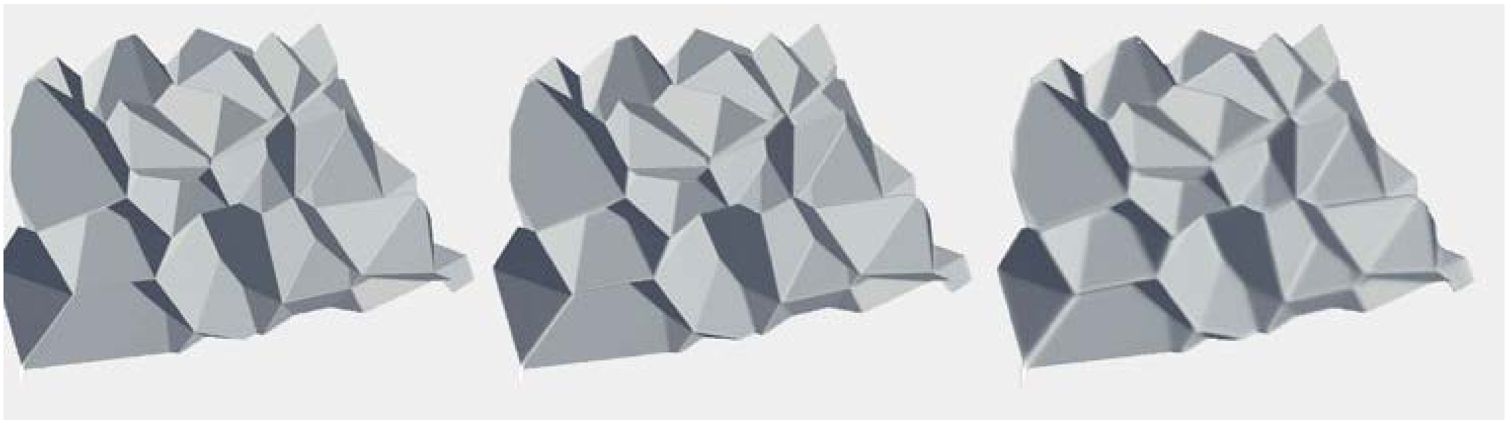
| *Source heightmap* | *Mask map* | *Masking result* |

## Outputs:

- ● Output - stores the generator's processing result.

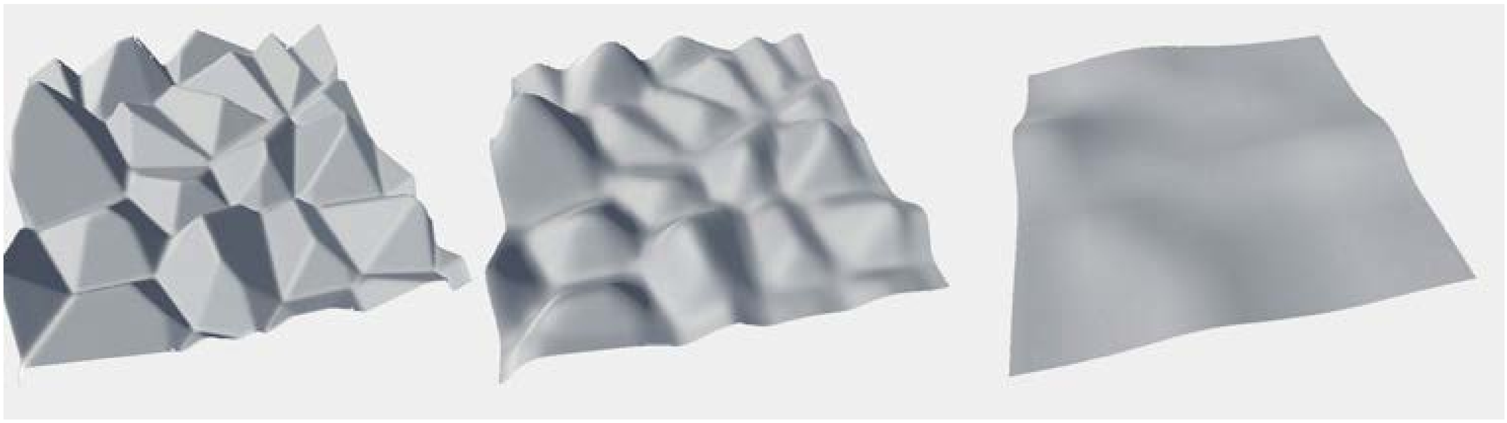
## Properties:

- Intensity: amount of blur applied per iteration.
- Iterations: number of blur iterations applied. Note that increasing the iterations count will reduce blur performance. The maximum reasonable amount of iterations is about 10, if you want more map smoothness use the Loss parameter.



*Iterations: 0, 1, 10*

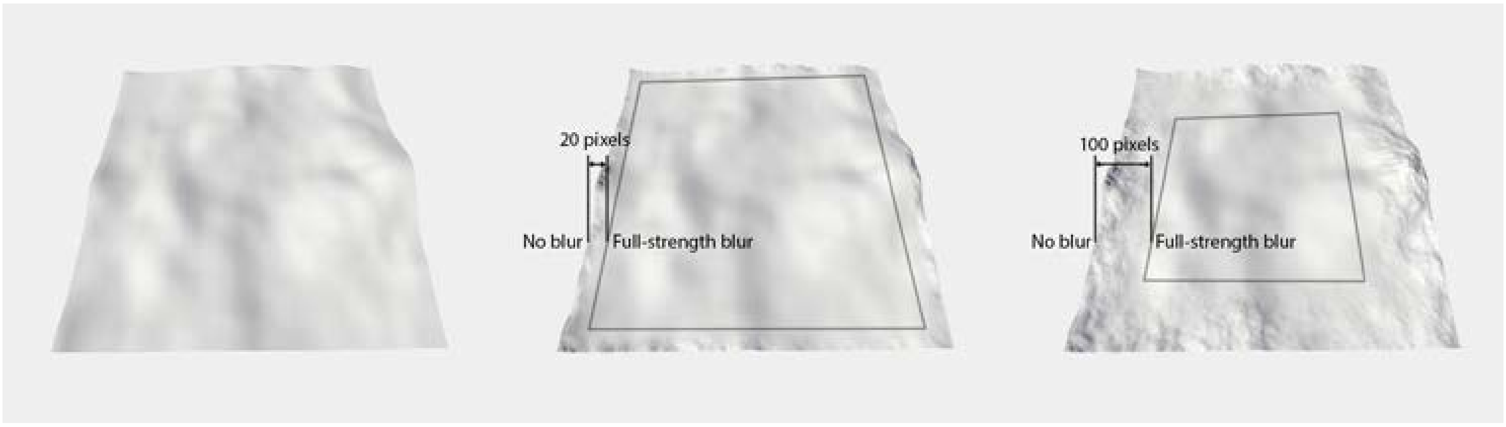
- Loss: forces the generator to skip some of the input pixels. This will result in a bit more 'pixelated' look, but will give the map extra smoothness. A value of 1 means that no pixels will be skipped. The Loss parameter has virtually no effect on performance so it could be used to create extremely smooth maps. Use it together with high Intensity and Iterations counts to avoid noticeable pixelization.



*Loss (with Iterations = 10): 1, 20, 60*

> Do not try to make your map extra-blurry by increasing Iterations value only. This will result in poor Blur performance. Use Loss value instead.

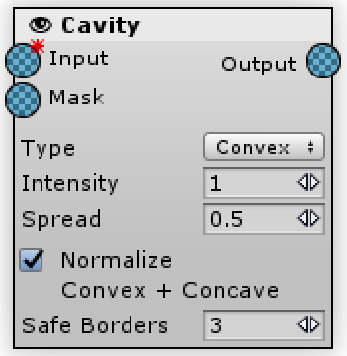
- Safe Borders: masks terrain borders so that the generator effect on the borders is zero, and increases with distance from the border. This parameter sets the amount (in pixels) until the effect's full strength. When set to 0 border masking is off. This parameter is essential for better terrain welding, so always enable it if this generator outputs to height.



*Safe Borders: 0 (off), 20, 100*

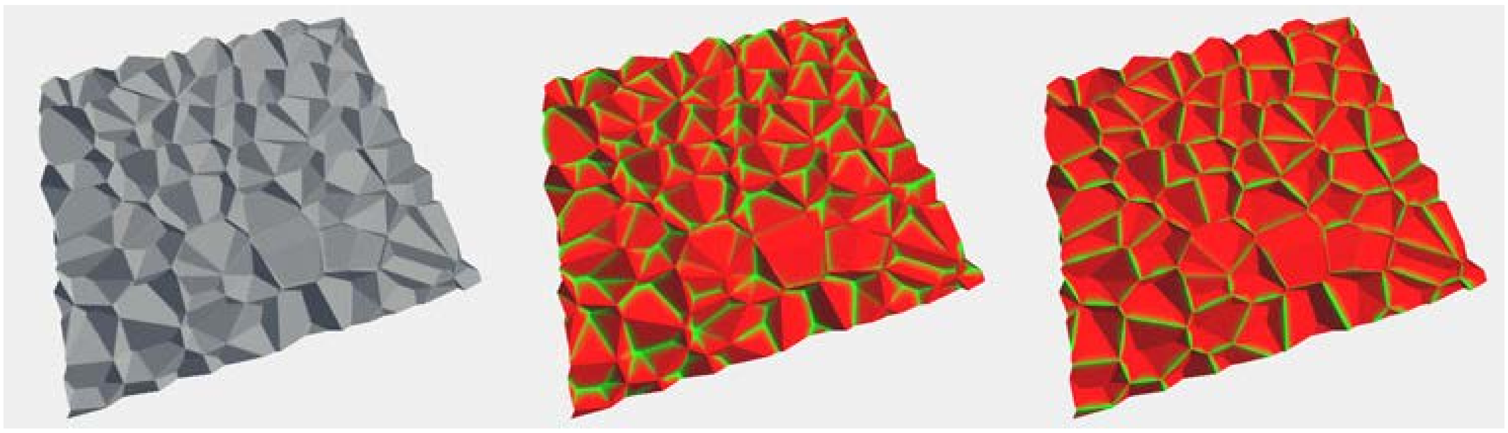
> Note that turning Safe Borders off (setting it to 0) will result in seams between chunks.

# Cavity



Type: Convex
Intensity: 1
Spread: 0.5
✓ Normalize Convex + Concave
Safe Borders: 3

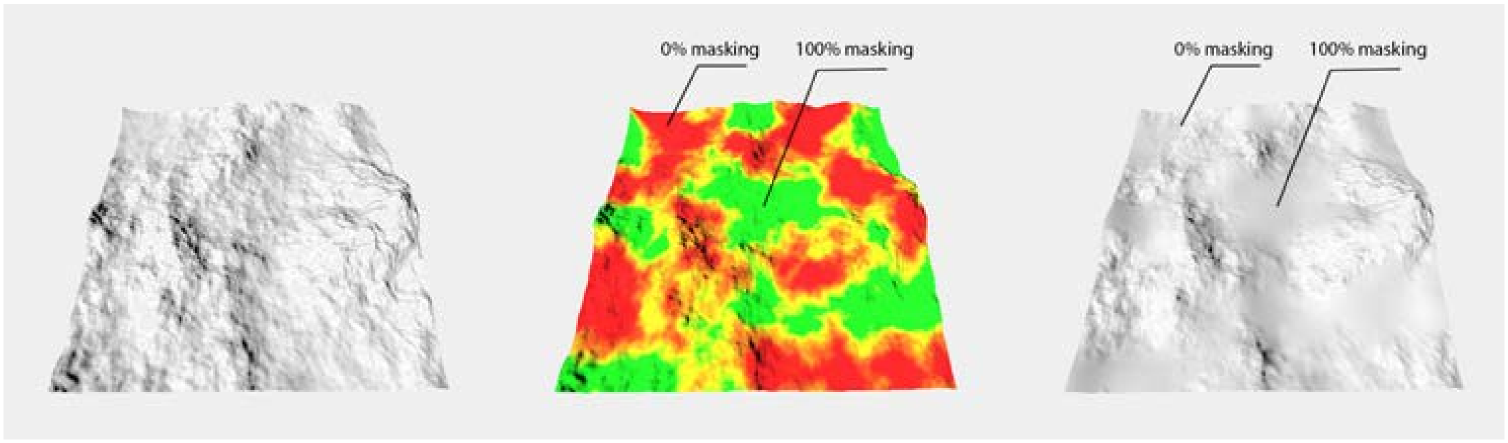
Generates the maps of concavity and convexity. All of the bulges and cambers are regarded as convex and the hollows are regarded as concave. Note that output maps do not intersect: one pixel can not be convex and concave at the same time.



| *Source height* | *Convex* | *Concave* |

## Inputs:

- ● Input - the heightmap to be processed by the generator.
- ● Mask - multiplies the generator intensity by the mask map’s value. For the mask pixels that have a value of 0 the generator effect is invisible, for the pixels that have a value of 1 the intensity is unchanged.



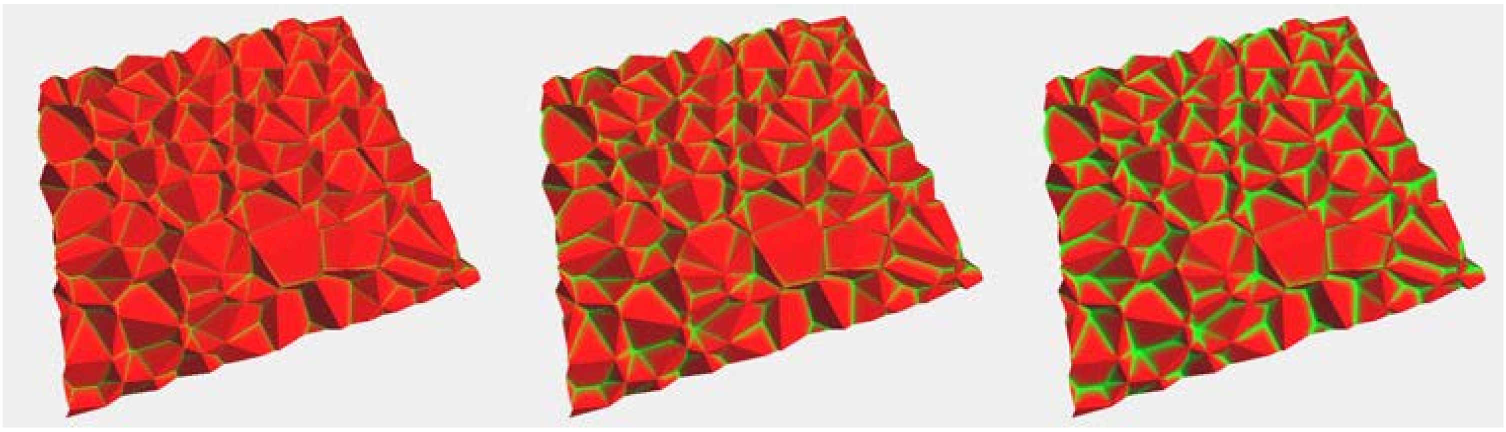
*Mask, using the example of **Blur** Generator:*
| *Source heightmap* | *Mask map* | *Masking result* |

## Outputs:

- 🌐 Output - stores the bulging or hollow areas (depending on a cavity type) and adjustment pixels.

## Properties:

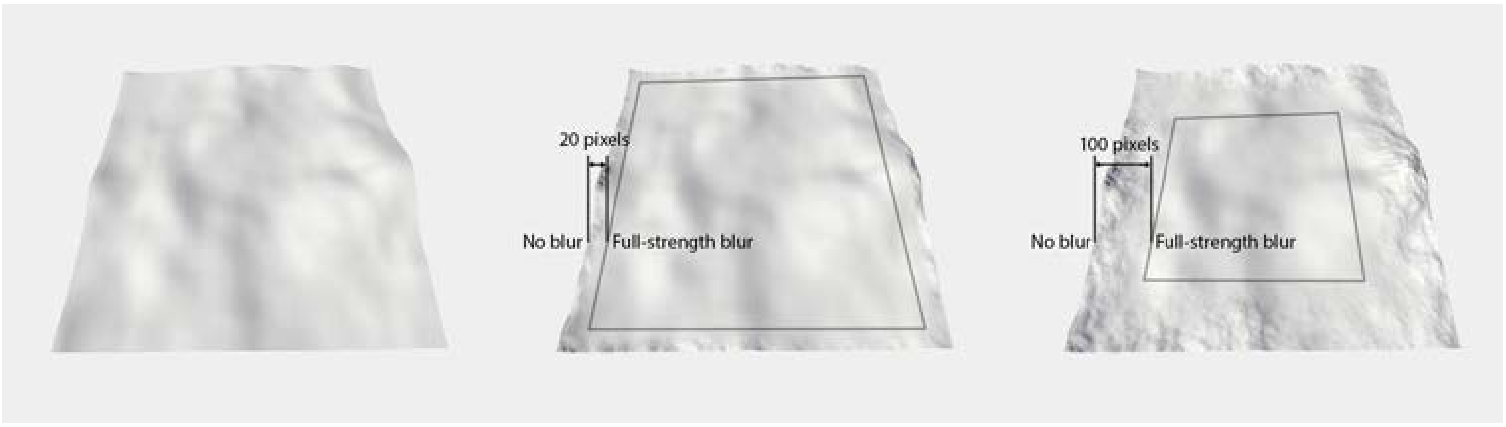
- Intensity - the amount of influence of the generator on the input map. When the value is set to 0 the generator effect is not visible. In other words, it is the generator's effective opacity.
- Spread - since really curving terrain areas are rather rare, the Cavity Generator can expand resulting information on more level areas.



*Spread: 1, 2.5, 5*

- Normalize Convex+Concave: ensures that all of the convex values combined with concave ones are always equal to 1. If this parameter is off only convex or concave map is calculated, its antipode is not taken into account. Turn it off to perform rude but faster calculations.

- Safe Borders: masks terrain borders so that the generator effect on the borders is zero, and increases with distance from the border. This parameter sets the amount (in pixels) until the effect's full strength. When set to 0 border masking is off. This parameter is essential for better terrain welding, so always enable it if this generator outputs to height.



*Using the example of **Blur** Generator:*
*Safe Borders: 0 (off), 20, 100*

> Note that turning Safe Borders off (setting it to 0) will result in seams between chunks.

# Constant

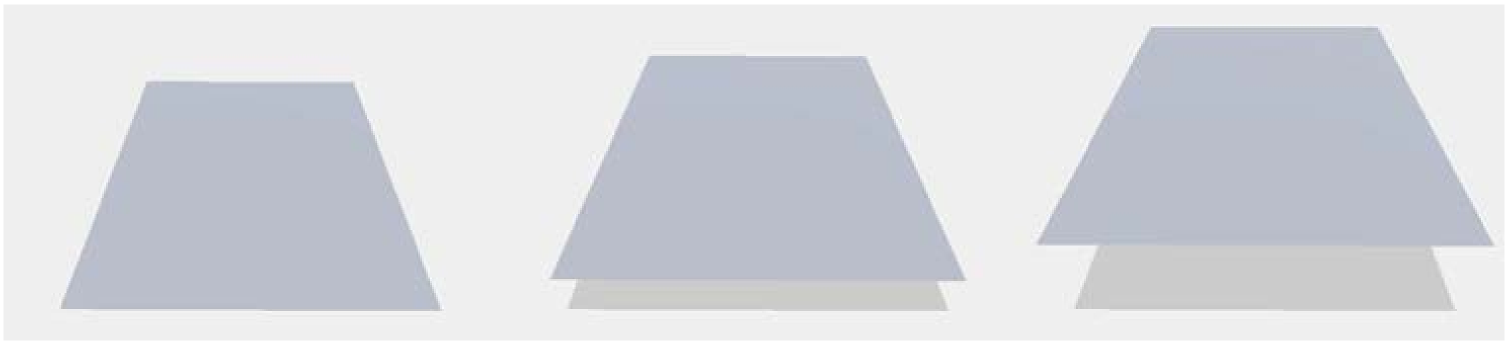
Creates a map filled with a given value. This is the most simple generator, used to create transparent masks or flat heightmaps.

## Output

The planar at a certain level.

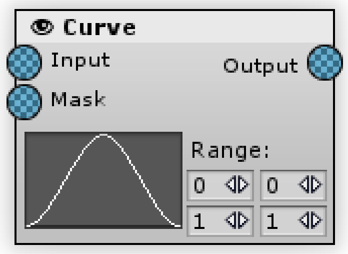
## Properties:

- Value: the map will be filled with this value on generate. Value ranges between 0 (no fill, transparent fill or zero ground level depending on the output where it will be used) and 1 (fully opaque fill or maximum terrain height).



*Value: 0, 0.5, 1*

# Curve



Adjusts map values using a user-defined curve. It works similar to the curves adjustment in graphics editors like Photoshop. The horizontal axis of the curve chart is the input value, vertical axis is the output value.

Map Magic uses Unity's animation curve interface for curve editing, so working with the Curve Generator is done the same way. Keys and tangents can be dragged with the left mouse button, the right mouse button is used to create new keys and access key properties.

This generator provides plenty of opportunities for map editing: it can be used to invert maps, adjust minimum or maximum values, clamp map values or even make ladder slopes (although a terrace generator could be more handy for this).

For example, to give the map more contrast just move left and right keys horizontally closer to the central line. To invert the map move the left key to the top and the right key to the bottom so that the diagonal line is inverted (note that key tangents should be set to auto to achieve a straight line).

## Inputs:

● Input(mandatory) - a source map

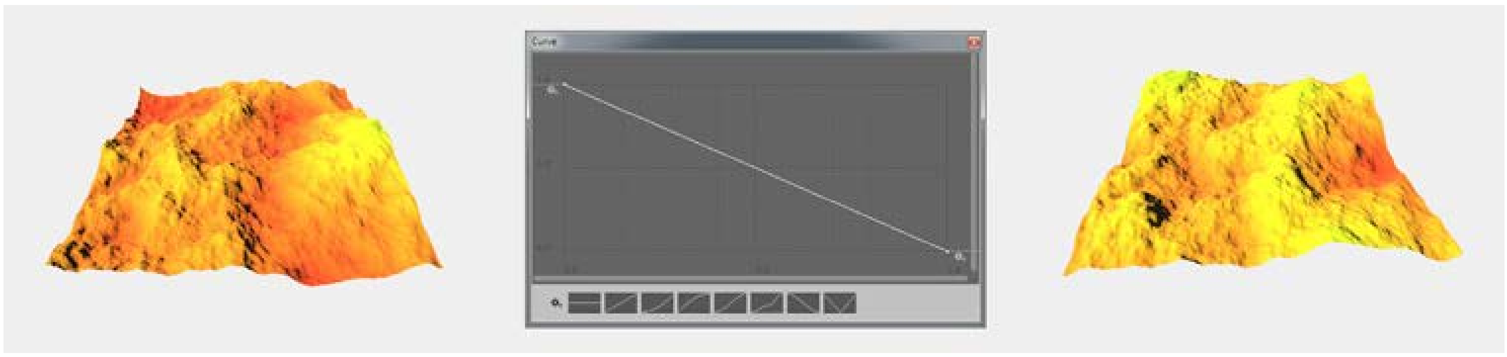
## Outputs:

● Output - a processed map

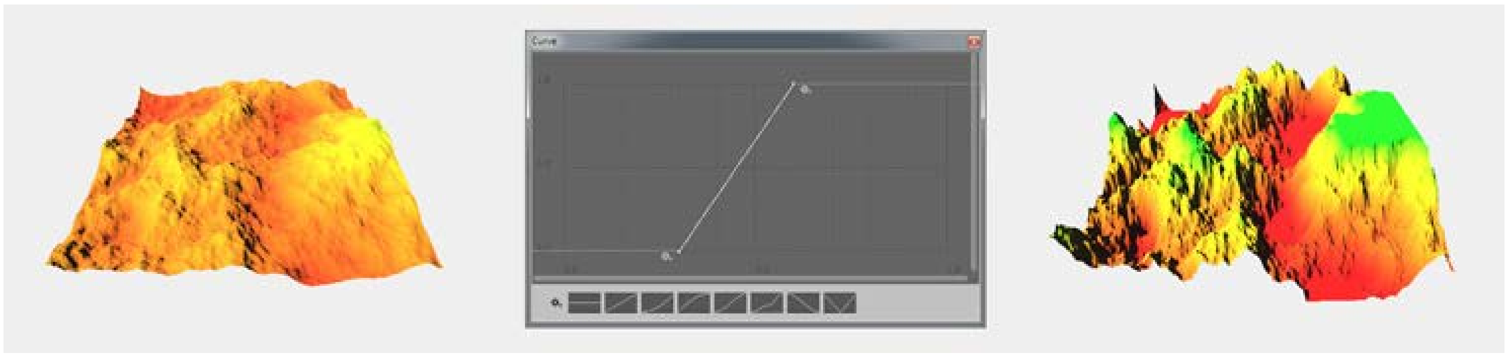
## Properties:

- Curve field: displays a curve preview. Clicking on it will open up a Curve Editor window - the standard Unity curve editing tool (see Unity manual: https://docs.unity3d.com/Manual/EditingCurves.html for more information).
- Range: by default the whole value range from 0 to 1 is used for editing. For more precise editing, minimum and maximum values could be set for input and output. The left row sets min and max for the input values (curve horizontal), while the right row sets output min and max (vertical).

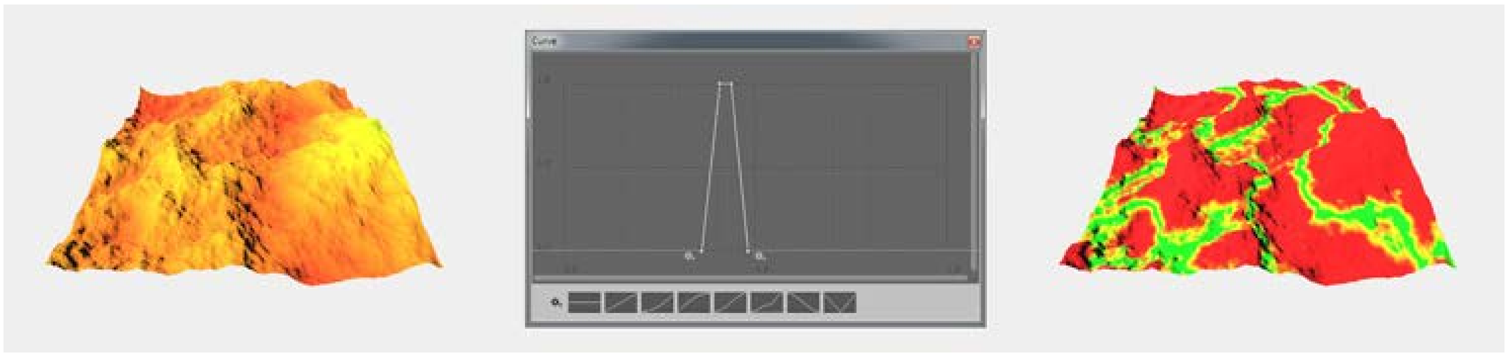
## Usage Examples:



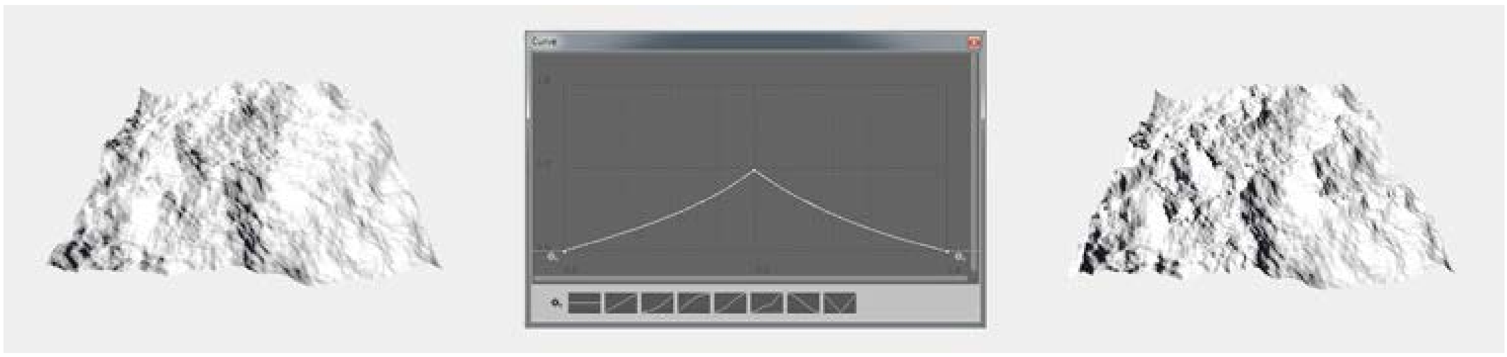
*Invert*



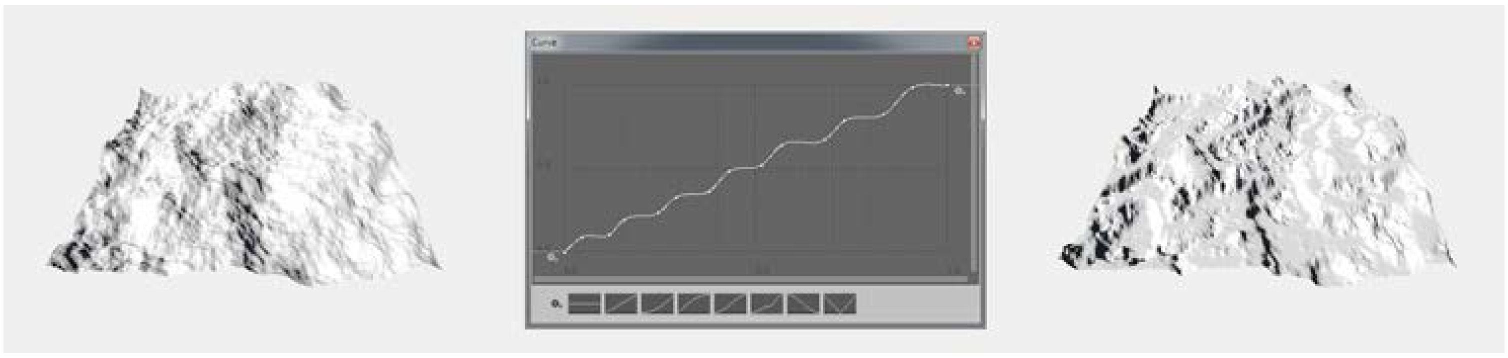
*Contrast*



*Select Range*

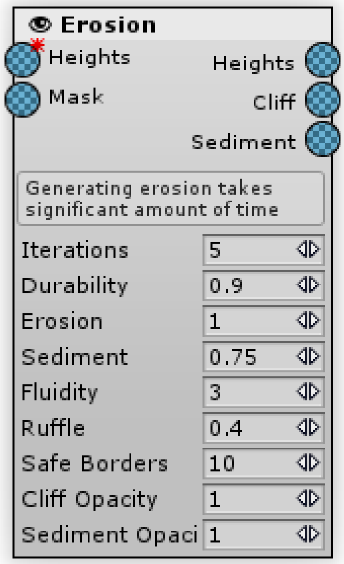


*Turbulence Ridge*



*Terrace*

# Erosion



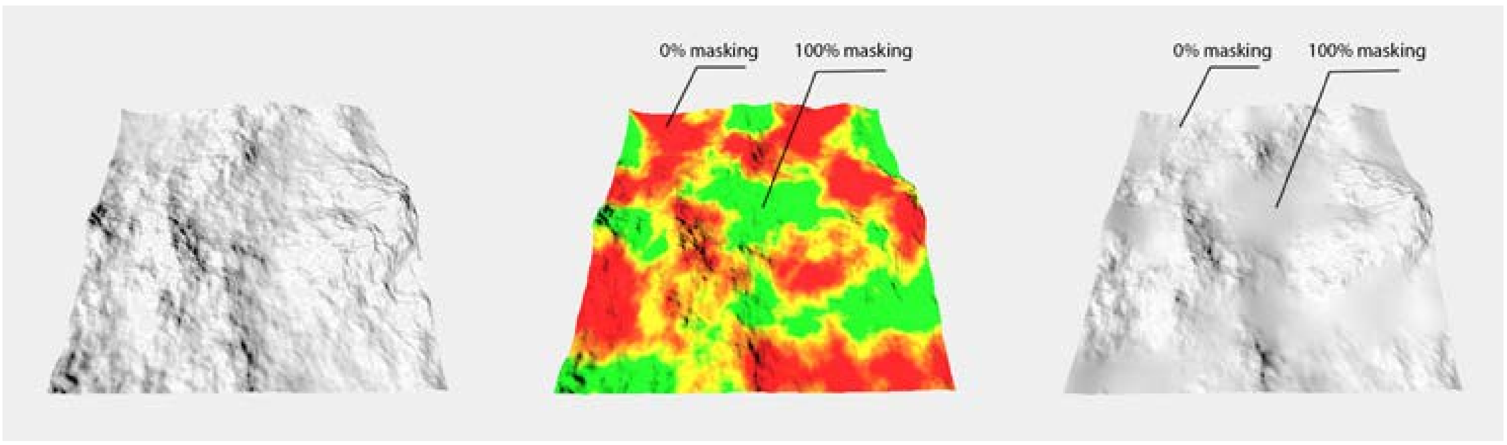
Generating erosion takes significant amount of time

| | |
|---|---|
| Iterations | 5 |
| Durability | 0.9 |
| Erosion | 1 |
| Sediment | 0.75 |
| Fluidity | 3 |
| Ruffle | 0.4 |
| Safe Borders | 10 |
| Cliff Opacity | 1 |
| Sediment Opaci | 1 |

Reproduces water’s flowing action on the terrain’s surface. Flows erode cliff formation, transport eroded stratum to another location and settle it as a sediment. Note that all of the flows, erosion and sediment calculations are iterative and very resource intensive, this makes the Erosion Generator the slowest generator of all the MapMagic generators. It usually takes about 3 seconds to generate erosion with the default parameters on a modern computer - compared to any other generator which are nearly realtime. But the result is usually worth it.

## Inputs:

- ● Heights - the heightmap to be processed by the generator.
- ● Mask - multiplies the generator intensity by the mask map’s value. For the mask pixels that have a value of 0 the generator effect is invisible, for the pixels that have a value of 1 the intensity is unchanged.



*Mask, using the example of **Blur** Generator:*
| *Source heightmap* | *Mask map* | *Masking result* |

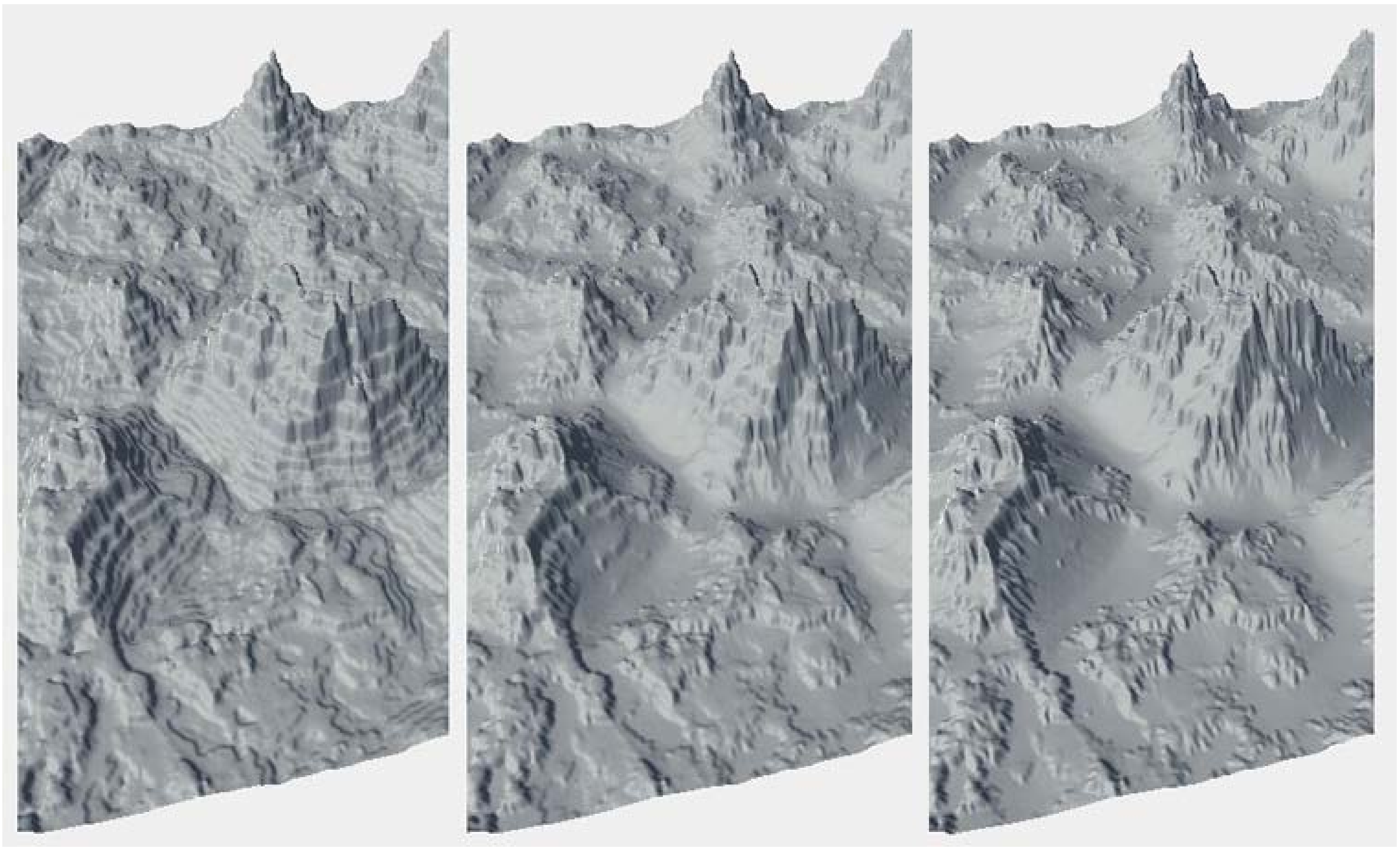
## Outputs:

- ● Heights - stores the generator’s changed heightmap result

- ● Cliff - stores eroded depth information. The more soil that was eroded from the current pixel - the higher the map pixel value. This map is multiplied by Cliff Opacity value.
- ● Sediment - stores the sediment depth information. The more sediment that was brought to the map's pixel - the higher its value. This map is multiplied by Sediment Opacity value.

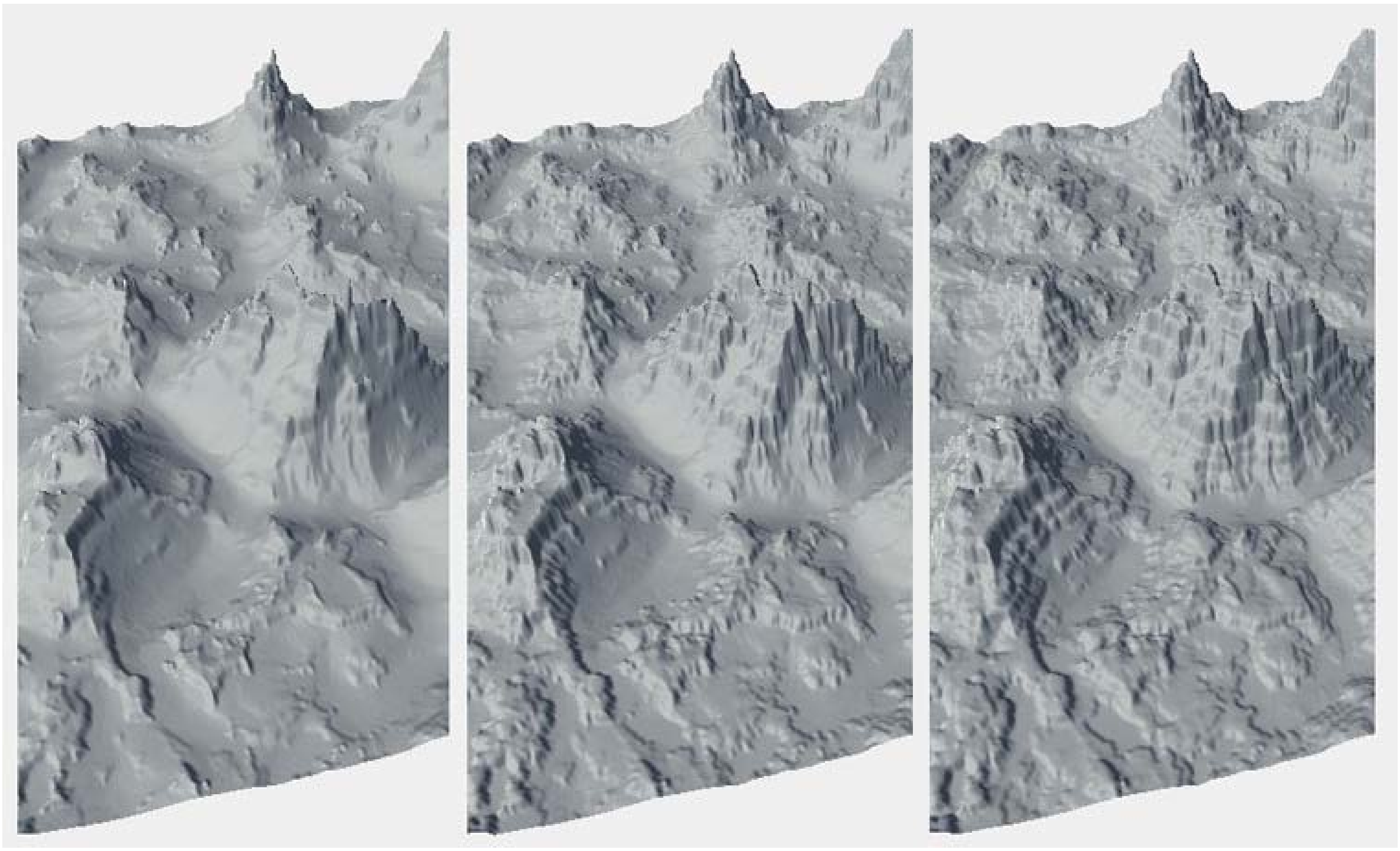
## Properties:

- Iterations: number of passes the generator should perform. This parameter determines the performance directly. In most cases there is no need to set it more than 5-7 as further erosion becomes less noticeable.

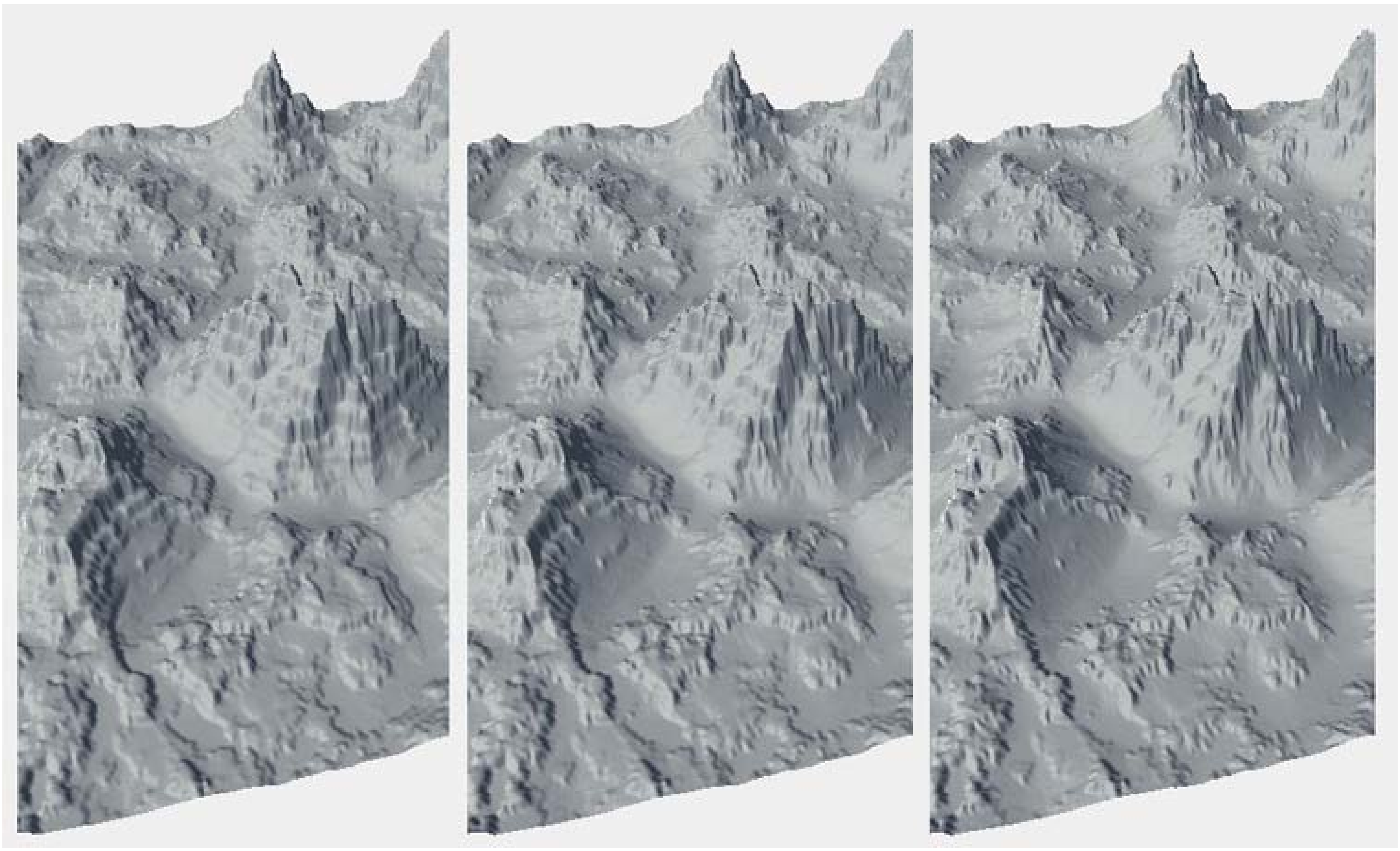


*Iterations: 1, 5, 15*

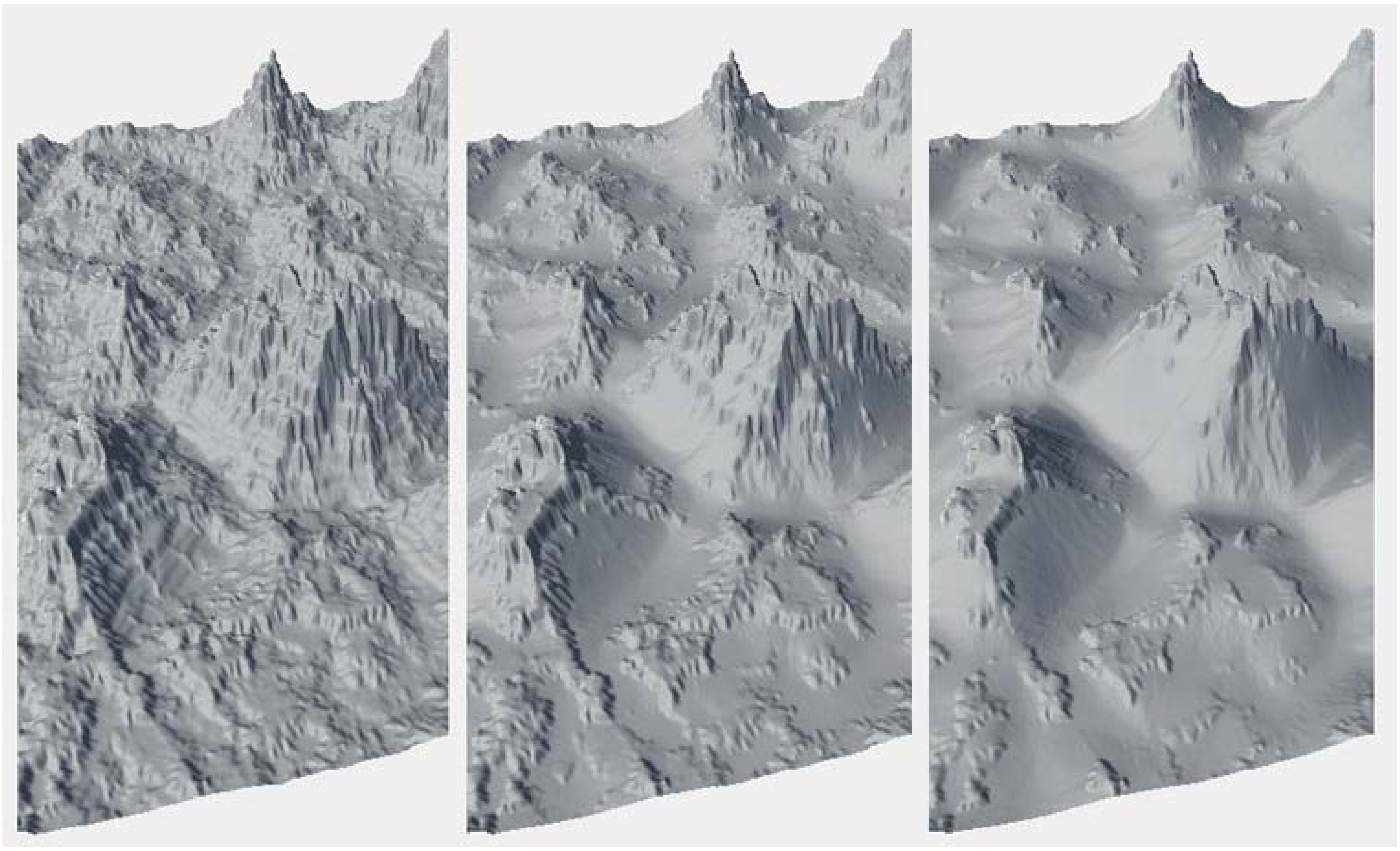
- Durability: determines how durable the input terrain is - i.e. how much it is affected by the flow erosion. Lower values will erode the land more with each iteration but the overall result will be less accurate and predictable, so it is recommended to set this value to 0.8 or higher. If value is set to 1 then the land will not be eroded at all. NOTE: since this parameter acts similarly to Erosion value it will be probably removed in future versions.


*Durability: 0, 0.9, 0.98*

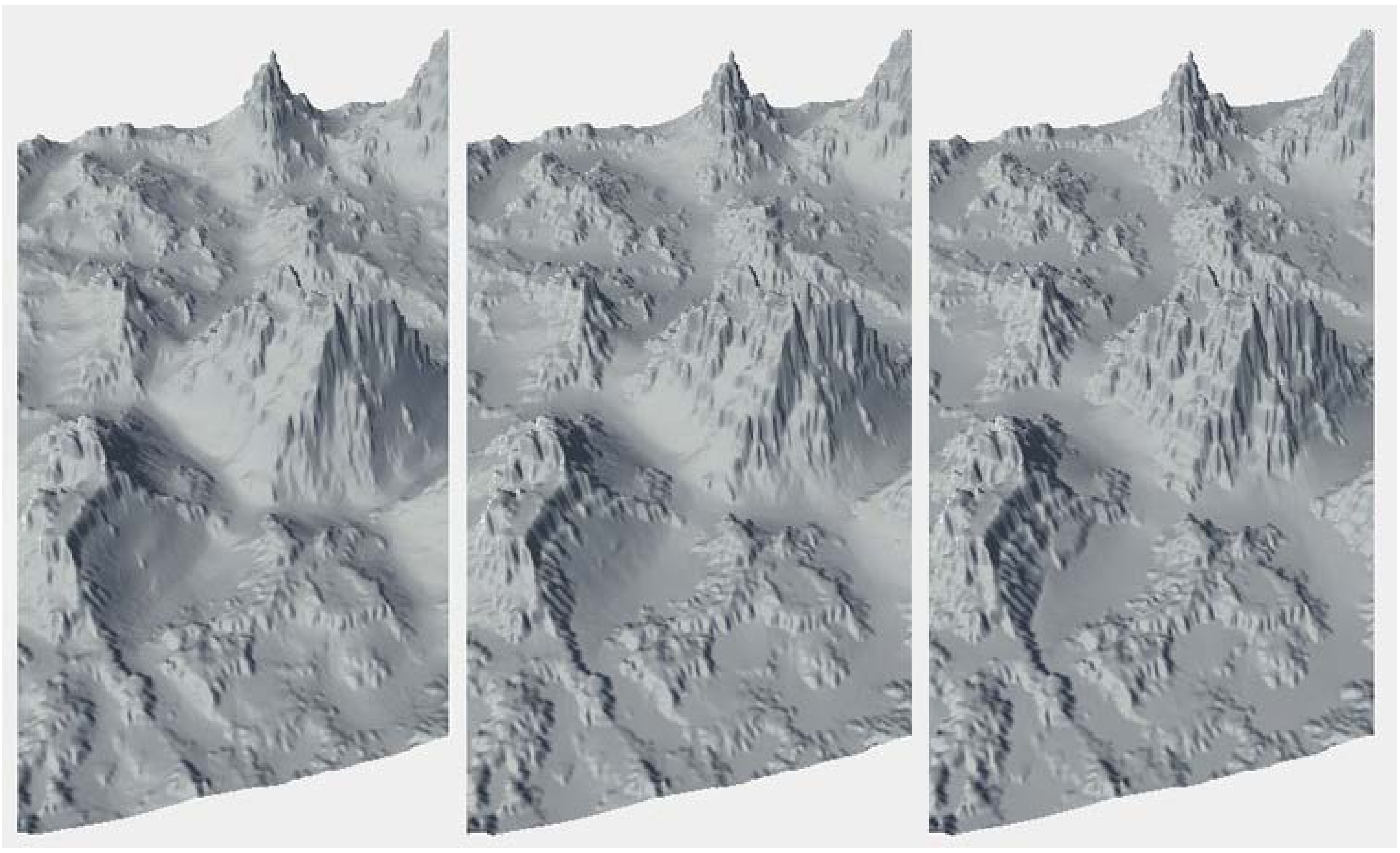
- Erosion: this factor multiplies the amount of stratum that will be raised by the flow to be transported to sediment.


*Erosion: 0.5, 1, 2*

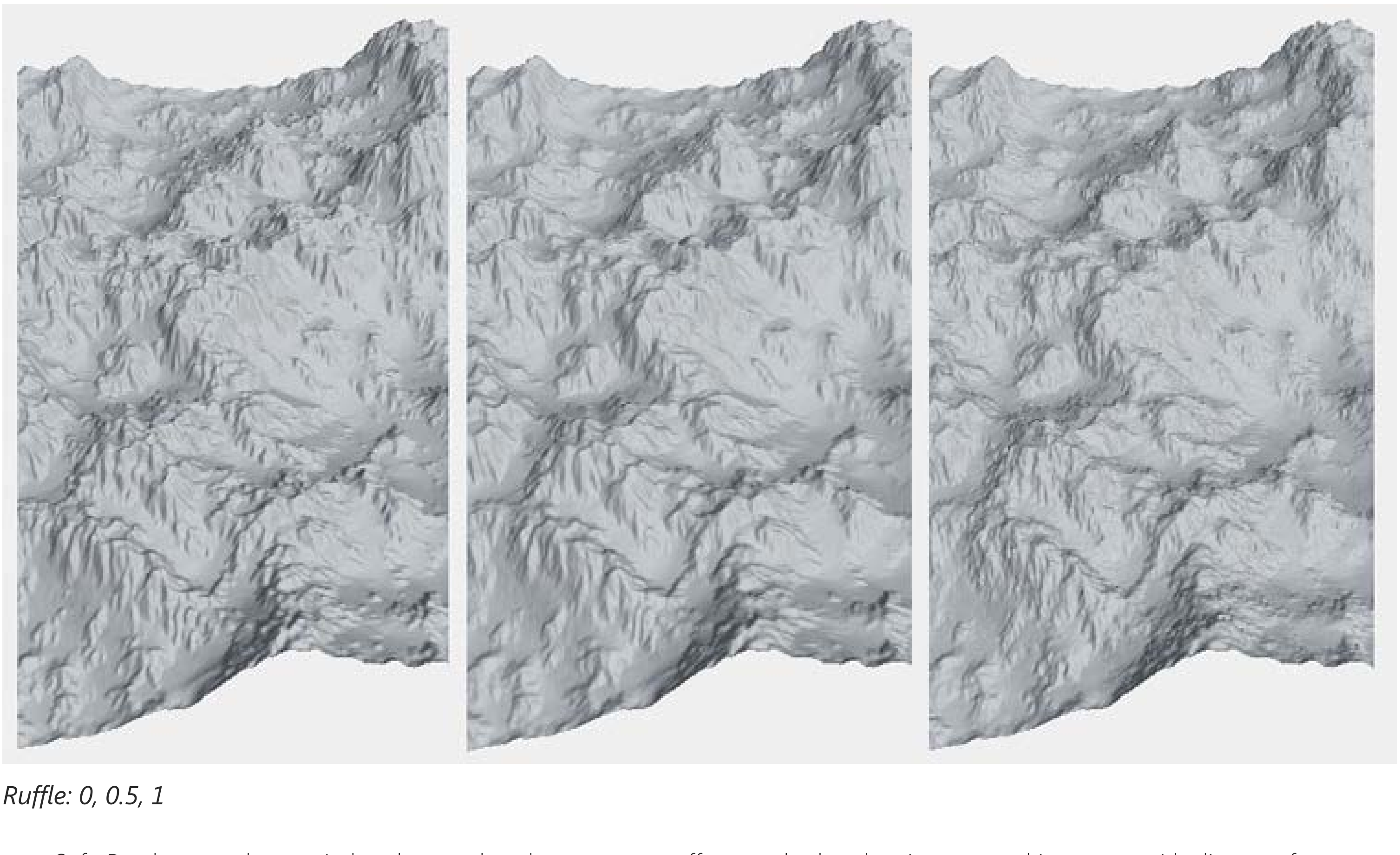
- Sediment: this factor multiplies the amount of stratum that will be settled out of the flow. A value 0 means that all of the eroded land will be disintegrated or carried away somewhere so that no sediment will be left at all, just eroded land will remain. A value 2 means that the stratum amount will be magically doubled before it settles. A realistic value should be slightly below 1, but fine results can be achieved within all of the 0-2 value range.


*Sediment: 0, 1, 3*

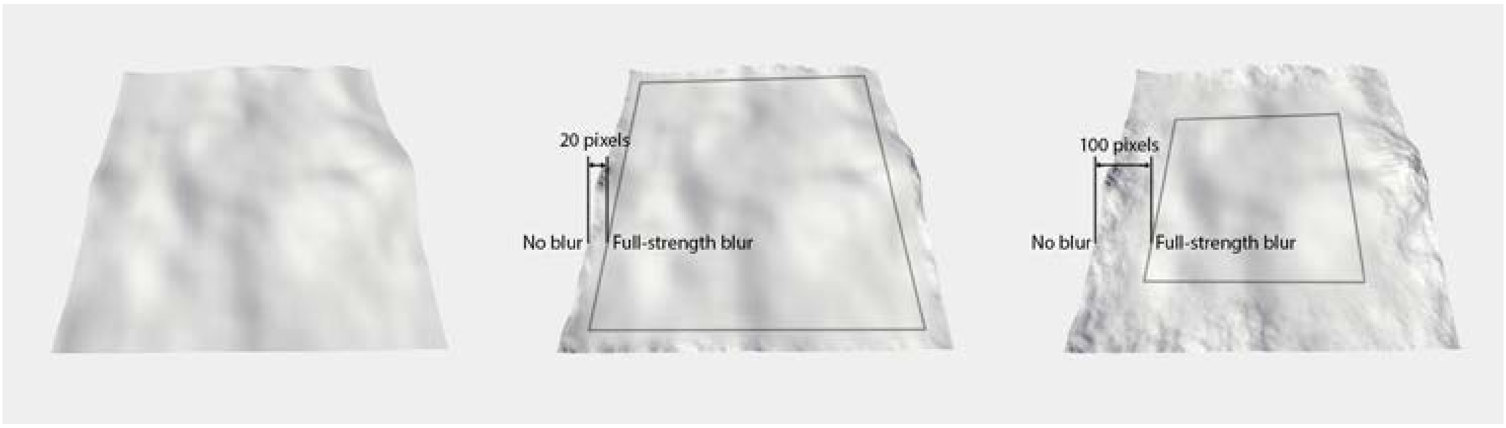
Fluidity iterations: the amount of passes used to calculate flow runs. A higher value means the farther sediment will travel from where the stratum was originally eroded. Together with the Iterations parameter it affects performance greatly.


*Fluidity: 1, 5, 15*

- Ruffle: adds some randomness to the amount of stratum that is being eroded. This results in a more noisy cliff look and a bit lesser erosion depth.


*Ruffle: 0, 0.5, 1*

- Safe Borders: masks terrain borders so that the generator effect on the borders is zero, and increases with distance from the border. This parameter sets the amount (in pixels) until the effect's full strength. When set to 0 border masking is off. This parameter is essential for better terrain welding, so always enable it if this generator outputs to height.

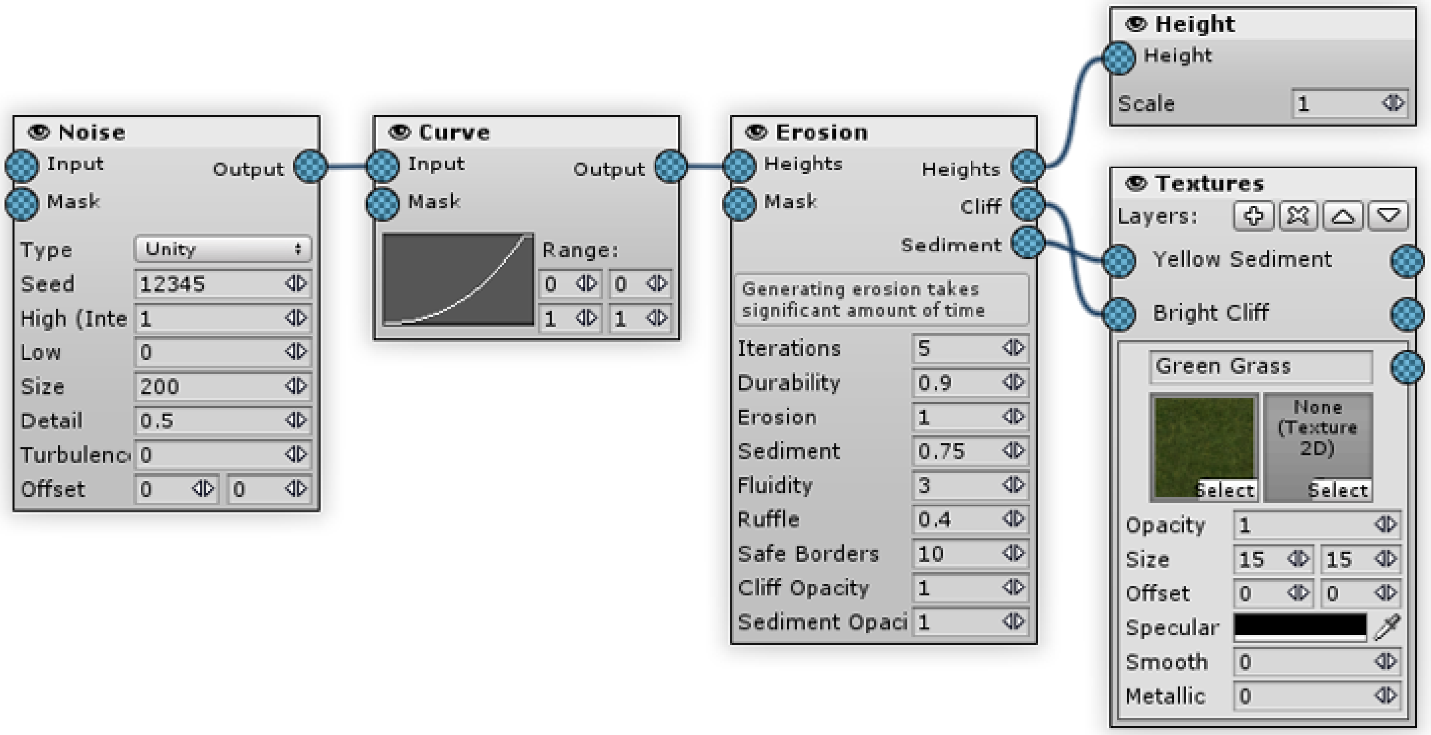


*Using the example of **Blur** Generator:*
*Safe Borders: 0 (off), 20, 100*

> Note that turning Safe Borders off (setting it to 0) will result in seams between chunks.

- Cliff and Sediment opacity: factors that multiply Cliff and Sediment outputs to make them more visible.

## Usage Example:



**Noise**
Input | Output
Mask

| | |
|---|---|
| Type | Unity |
| Seed | 12345 |
| High (Inte | 1 |
| Low | 0 |
| Size | 200 |
| Detail | 0.5 |
| Turbulenc | 0 |
| Offset | 0   0 |

**Curve**
Input | Output
Mask
Range: 0   0 / 1   1

**Erosion**
Heights | Heights
Mask | Cliff
Sediment

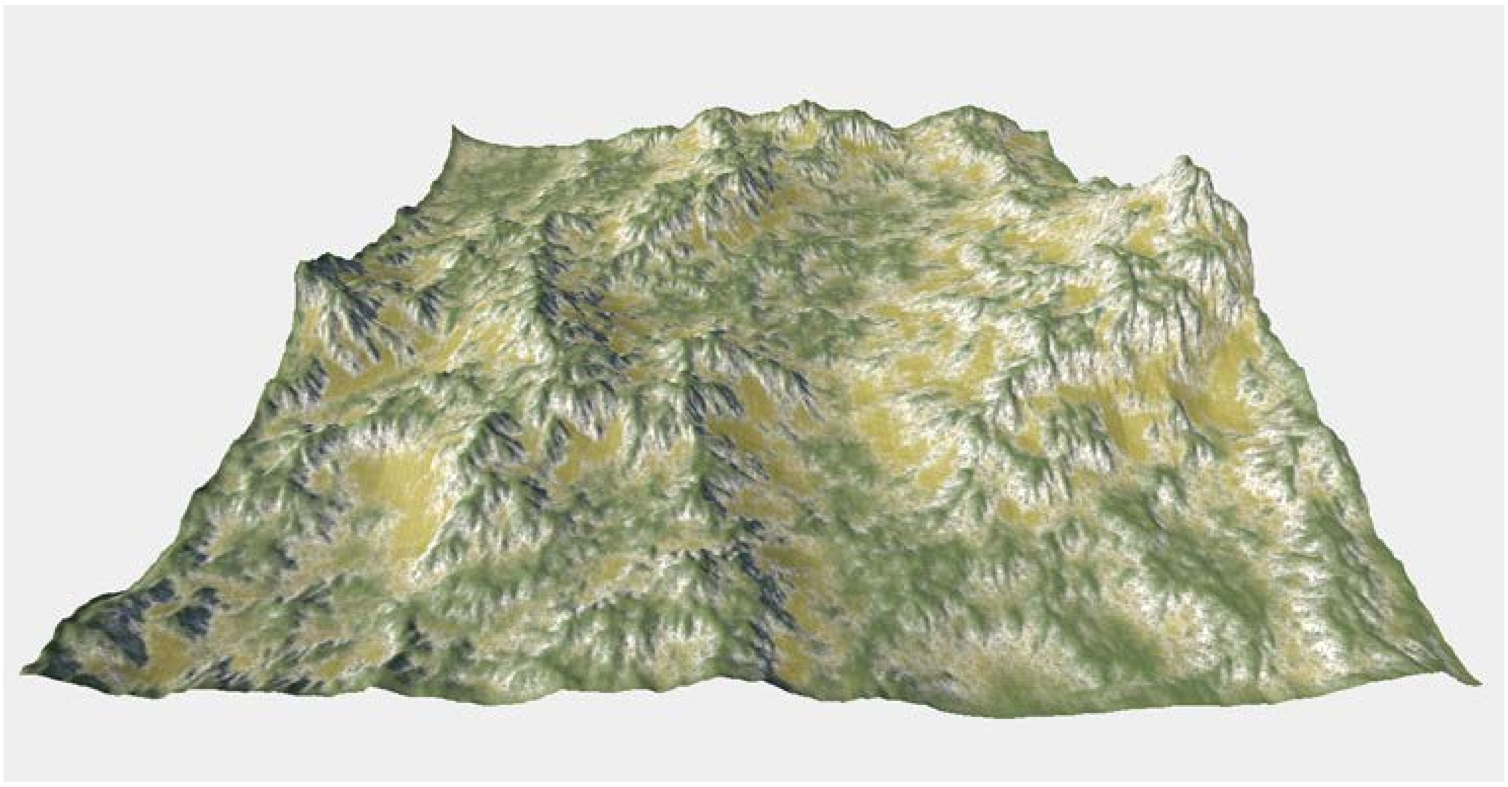
Generating erosion takes significant amount of time

| | |
|---|---|
| Iterations | 5 |
| Durability | 0.9 |
| Erosion | 1 |
| Sediment | 0.75 |
| Fluidity | 3 |
| Ruffle | 0.4 |
| Safe Borders | 10 |
| Cliff Opacity | 1 |
| Sediment Opaci | 1 |

**Height**
Height
Scale 1

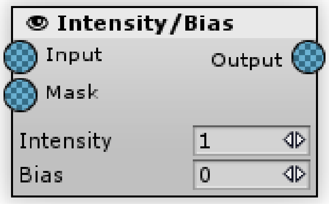
**Textures**
Layers:
Yellow Sediment
Bright Cliff
Green Grass
None (Texture 2D)
Select   Select

| | |
|---|---|
| Opacity | 1 |
| Size | 15   15 |
| Offset | 0   0 |
| Specular | |
| Smooth | 0 |
| Metallic | 0 |



*Note that the Sediment layer is applied after the Cliff one in Textures Output*

# Intensity_bias

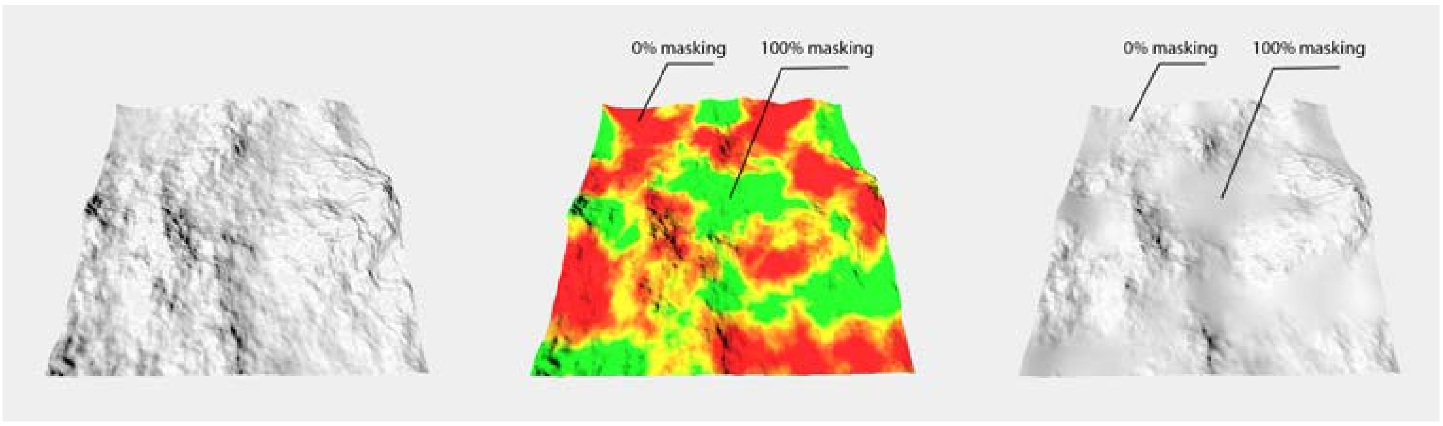


Intensity 1
Bias 0

A simple way to adjust map contrast. It's useful if you don't want to deal with curves for some reason.

## Inputs:

- ● Input - the default map to be processed by the generator.
- ● Mask - multiplies the generator intensity by the mask map's value. For the mask pixels that have a value of 0 the generator effect is invisible, for the pixels that have a value of 1 the intensity is unchanged.

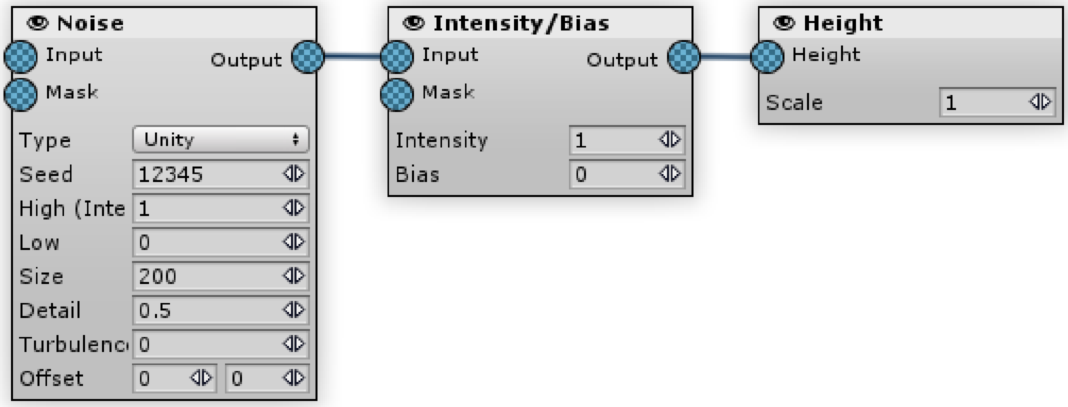


*Mask, using the example of **Blur** Generator:*
| *Source heightmap* | *Mask map* | *Masking result* |

## Outputs:

- ● Output - stores the generator's processing result.

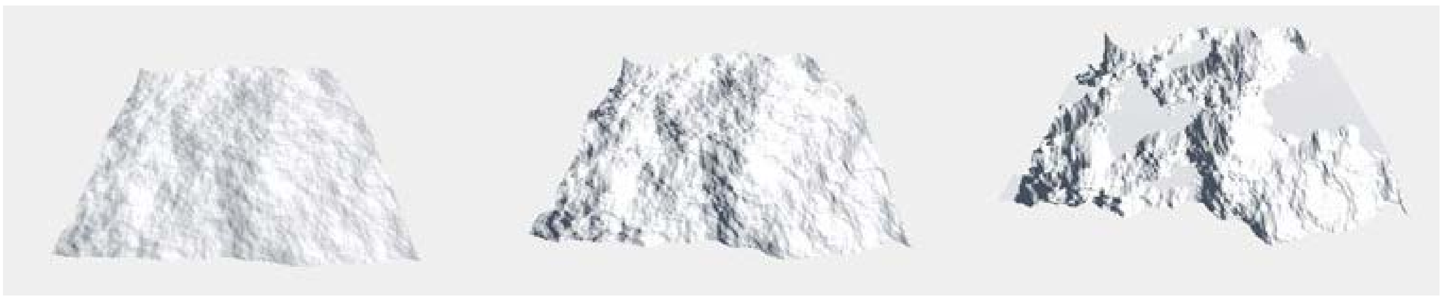
## Properties:



Noise: Type Unity; Seed 12345; High (Inte 1; Low 0; Size 200; Detail 0.5; Turbulenc 0; Offset 0 0
Intensity/Bias: Intensity 1; Bias 0
Height: Scale 1

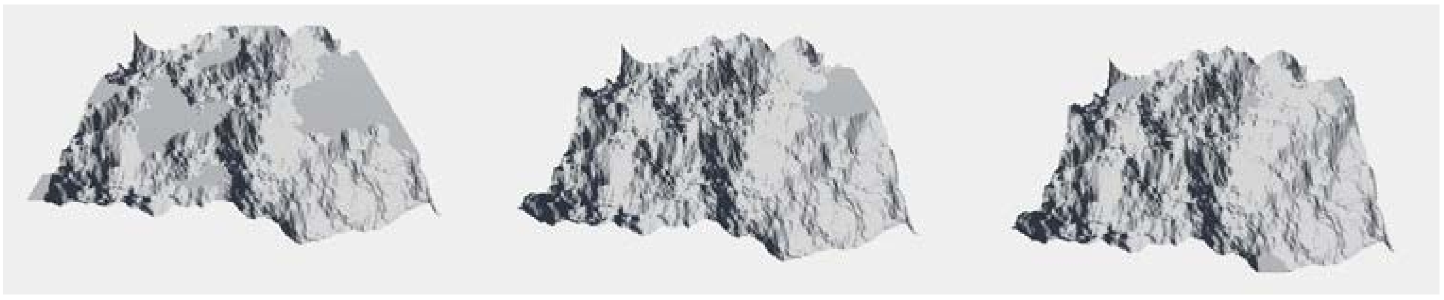
*Graph used in examples*

- Intensity: multiplies input values with the value specified. Consider it as "Contrast".



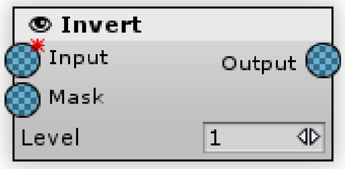
*Intensity: 0.5, 1, 2*

- Bias: adjusts output values by adding value specified. It does not affect anything if Intensity value is 1 and gains effect the more Intensity is higher or lower from 1.

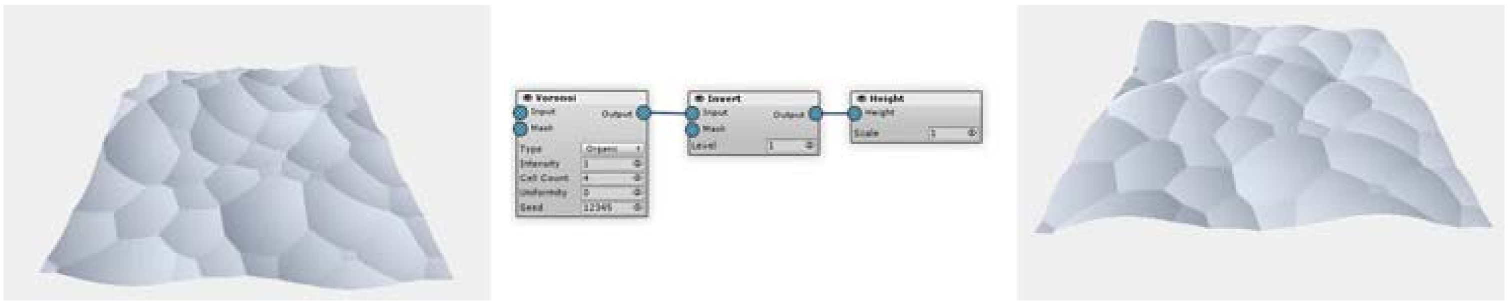


*Bias (Intensity=2): 0, 0.25, 0.5*

# Invert

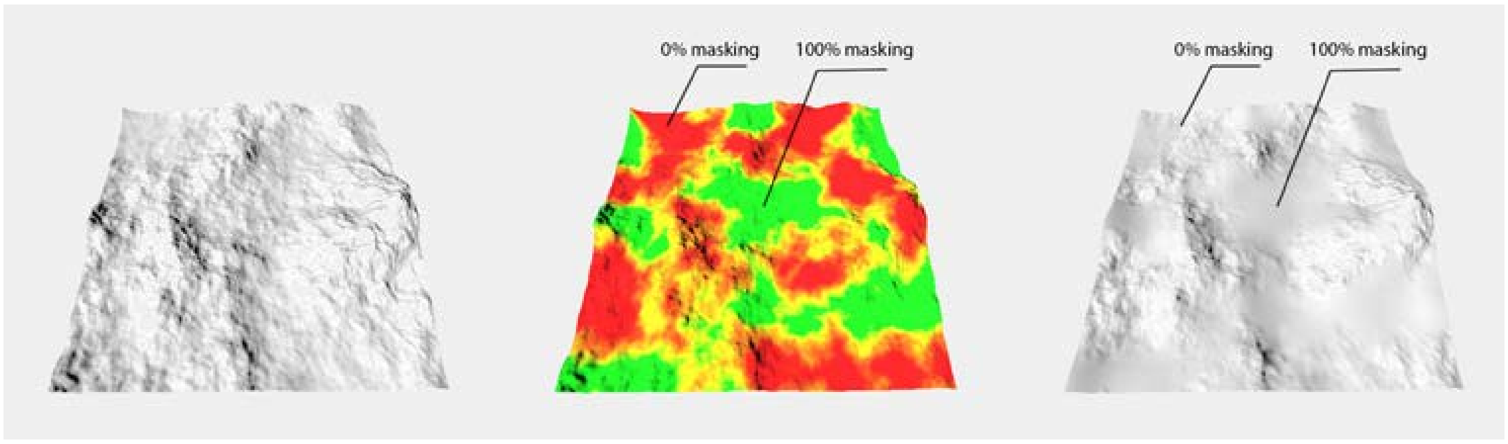

Invert
Input
Output
Mask
Level 1

A simple way to invert a map. It's useful if you don't want to deal with curves for some reason. Mathematically it's 1-value algorithm, or, to be more exact, Level-value.



## Inputs:

- Input - the default map to be processed by the generator.
- Mask - multiplies the generator intensity by the mask map’s value. For the mask pixels that have a value of 0 the generator effect is invisible, for the pixels that have a value of 1 the intensity is unchanged.



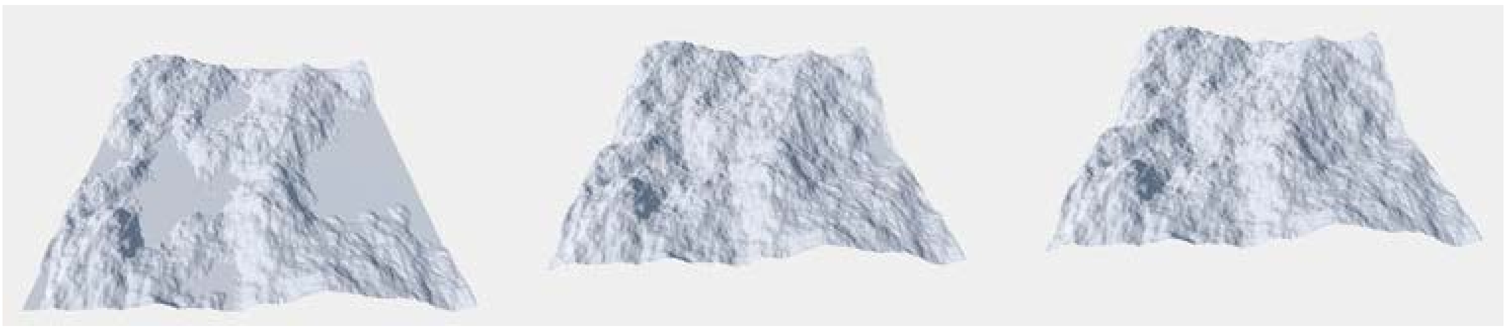
*Mask, using the example of* ***Blur*** *Generator:*
| *Source heightmap* | *Mask map* | *Masking result* |

## Outputs:

- Output - stores the generator’s processing result.

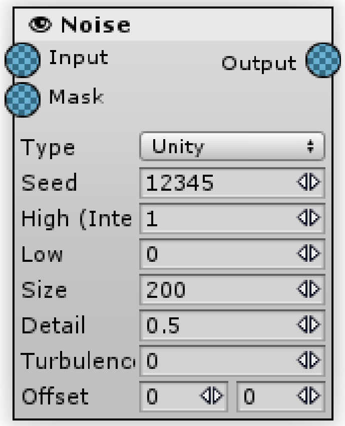
## Properties:

- Level: subtrahend in invert algorithm. Determines the height of inverted map.



*Level: 0.5, 0.75, 1*

# Noise



Type: Unity
Seed: 12345
High (Inte: 1
Low: 0
Size: 200
Detail: 0.5
Turbulenc: 0
Offset: 0 0

The Noise Generator is one of the most basic generators in MapMagic. It generates a fractal Perlin noise map that is widely used as a base for various map creation algorithms.

## Inputs:

- ● Input - if no input is specified, noise generator returns the pure noise. If input is specified, noise is added (+) above the input map.
- ● Mask - a map that controls a generator’s level of intensity. It multiplies (*) the noise value with a mask value.

> It is common for the Noise Generator to be used with no inputs connected, but in order to easily combine it with the other maps these 2 inputs are used. Connecting noise with an "Input" will result in applying noise additive, and with a "Mask" will multiply map with a noise.

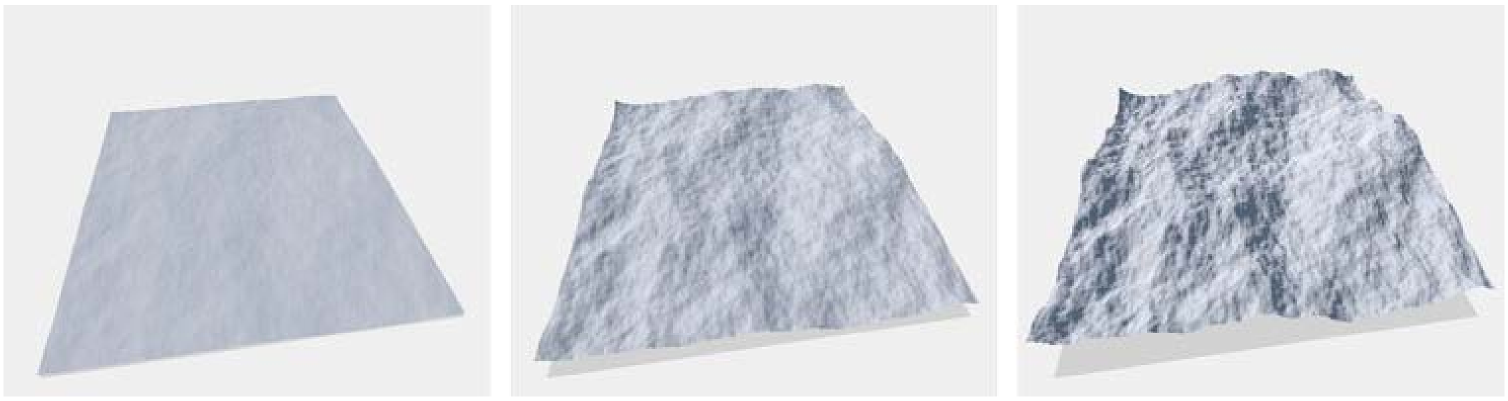
## Outputs:

● Output - a noise map (if no input specified) or an input map with a noise added (if input specified).

## Properties:

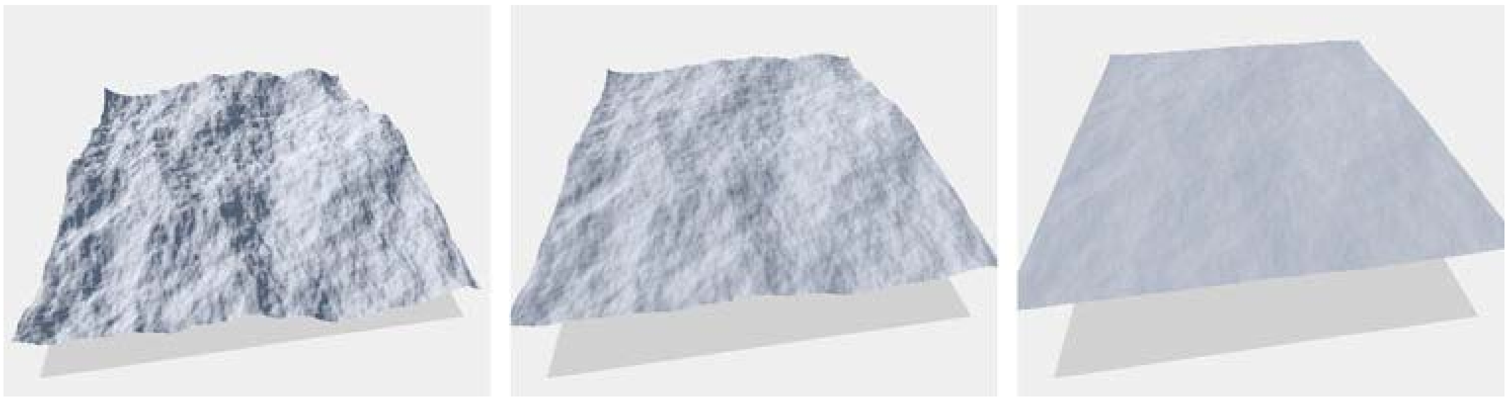
- Seed - a number used to initialize pseudo-random noise generator. If two generators use equal seed numbers the resulting pattern will be the same.

- High (intensity)- sets the highest noise value. As the name says, consider it as the noise intensity.



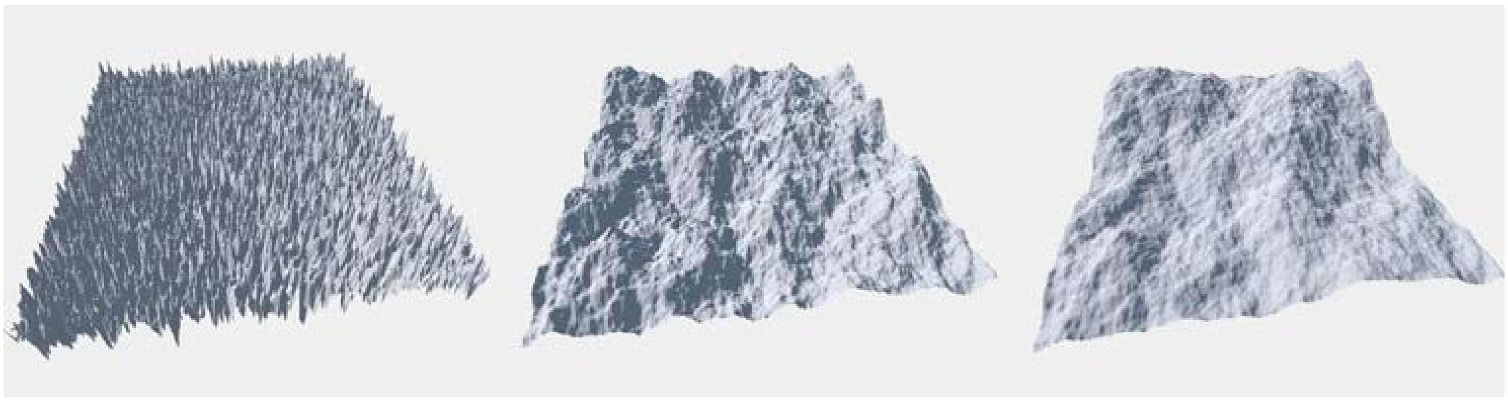
*High (intensity): 0.1, 0.5, 1*

- Low - sets the minimum noise value. All of the noise values generated lay within the Low-High range. You can increase this value to make the noise less contrast, or lower it below zero level to make the noise sharp.



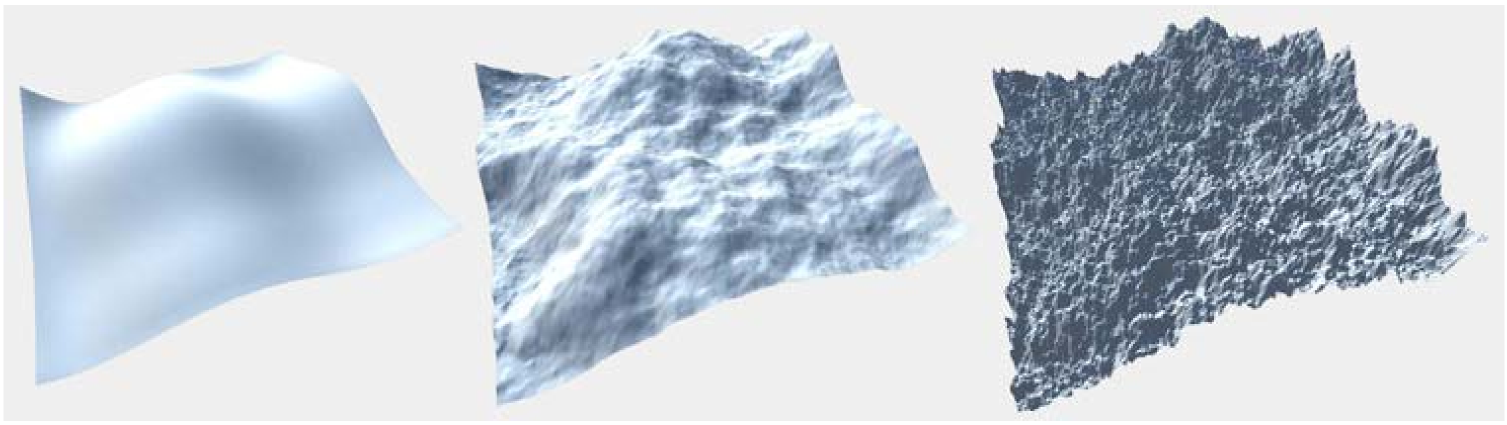
*Low: 0, 0.5, 0.9*

- Size - this parameter determines the size of the biggest fractal. Noise above this value ceases to be fractal being a standard perlin noise. Lower size values result in a very homogeneous and predictable noise, while high values can create diverse noise and scenic heightmaps.



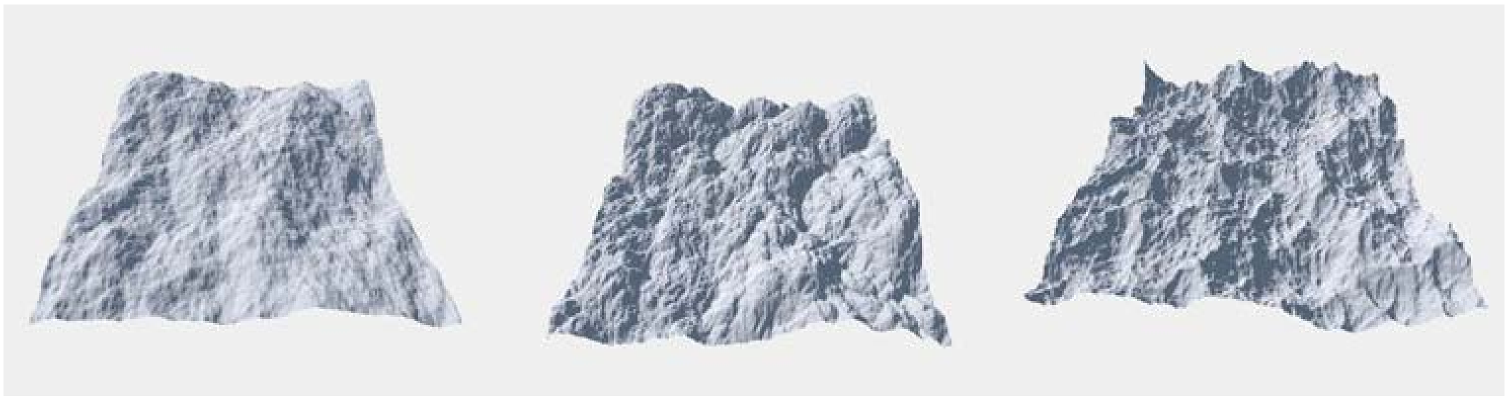
*Size: 10, 100, 200*

- Detail - this parameter determines the big and small fractals' bias. When the parameter is higher than 0.5 then small fractals have greater impact, which results in a more 'noisy' map. When parameter is below 0.5 small fractals are less significant than the big ones, which results in a smoother noise.



*Detail: 0, 0.4, 0.6*

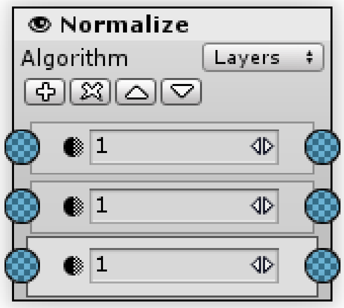
- Turbulence - creates "bubbles" (positive) or "ridges" (negative).



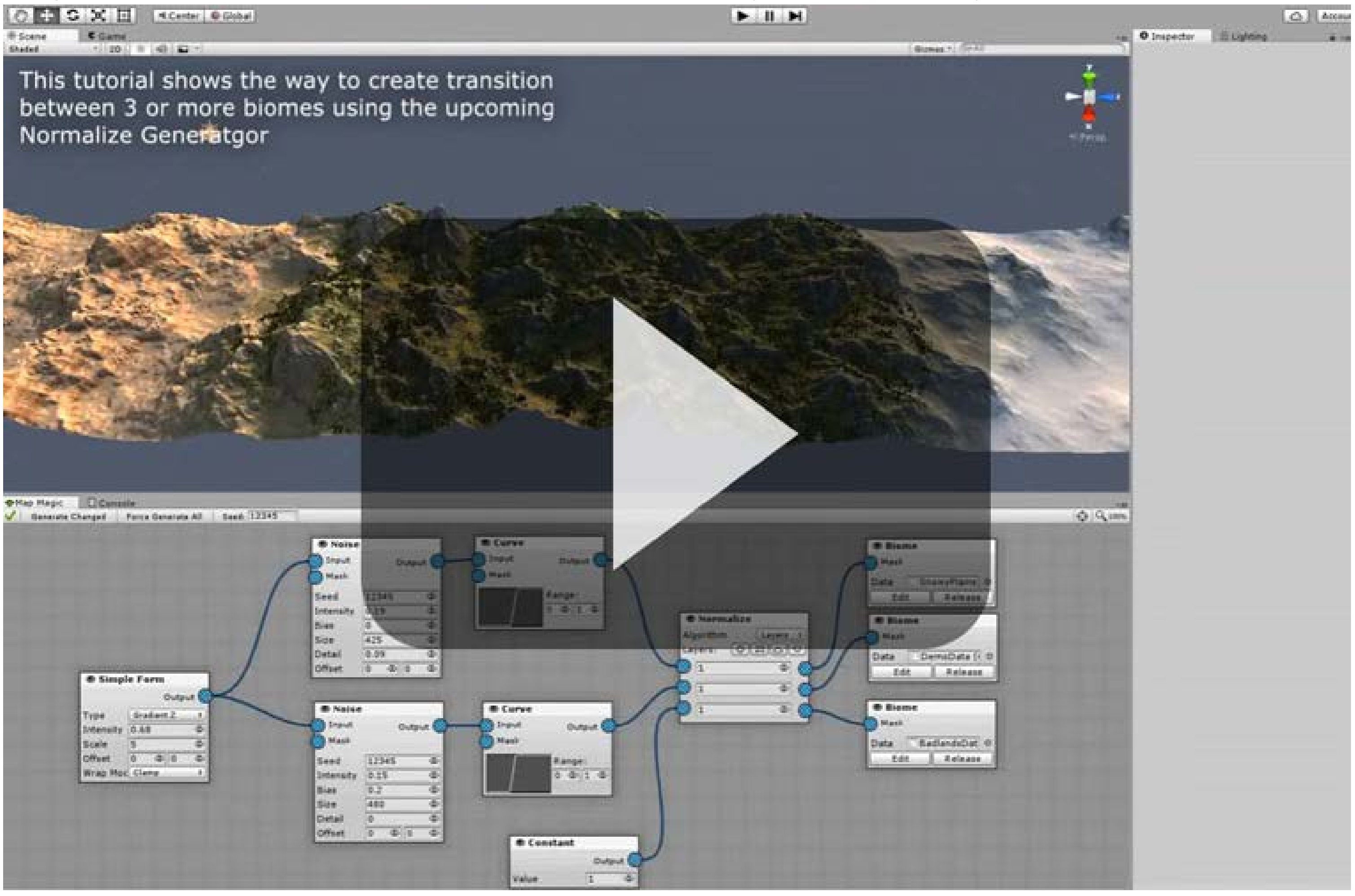
*Turbulence: 0, 1, -1*

- Offset - this parameter defines the noise pattern position, shifting it along the x and z axes. These values can be positive or negative. Offset applies to all of the terrains and sums with the terrain position, so the noise maps continue seamlessly on all the terrains.

# Normalize



This generator is used to combine several maps so that their sum is always 1. It could be handy to prepare biome masks:


This tutorial shows the way to create transition between 3 or more biomes using the upcoming Normalize Generatgor
Simple Form
Noise
Curve
Constant
Normalize
Biome

## Inputs:

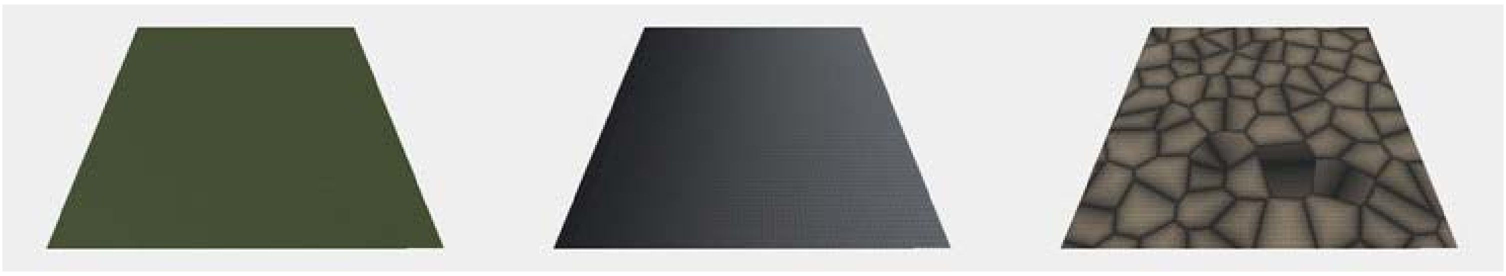
- ● Input in all of the layers - source non-normalized maps.

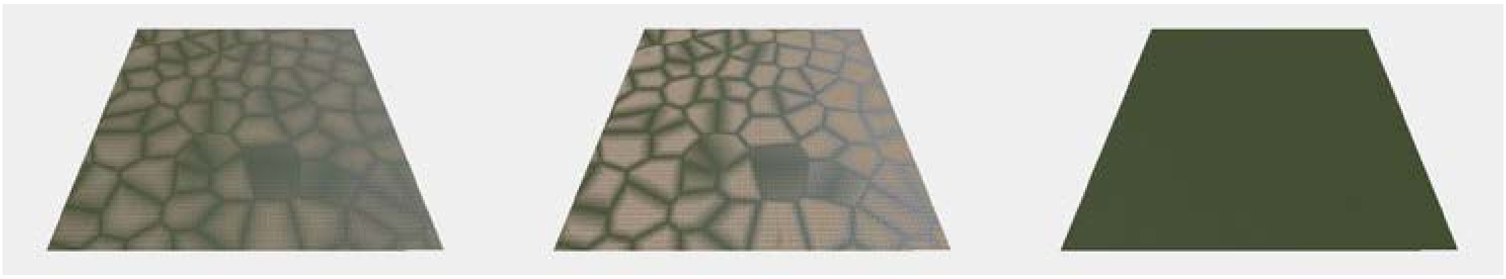
## Outputs:

- ● Map outputs in all if the layers - input maps processed so that any pixel sum of the map values is equal to 1.

## Properties:

- Algorithm: determines the way maps are normalized: ** Sum: for every pixel the generator is calculating sum of all map values. Then each value is divided by that sum. This way all of the layers intensity is scaled proportionally, and no layer has superiority. The layers order does not matter when using Sum algorithm. ** Layers: This generator acts quite similar to the layer blending mode in Photoshop or other graphics editor, and the same way Texture Output textures are blended. Layers are applied from bottom to top, so the top layer is applied without any changes at all.


*Normalize Source Maps: Grass, Cliff, Sand*

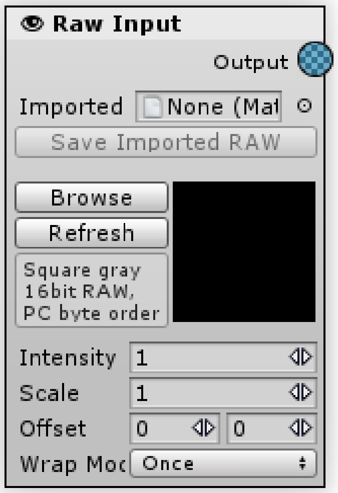

*Normalize algorithms: Sum, Layers, Layers (layer order inverted, so 100% filled grass at the top displaces all other maps).*

Normalize Generator input maps are stacked using the layers mode. Each layer could be selected by clicking on it. Above the layers you can see the layer control buttons:

- ✚ Add: will add new layer atop of the selected one.
- ✖ Remove: removes currently selected layer.
- △ Up: will change the layers order by moving the selected layer up.
- ▽ Down: will change the layers order by moving the selected layer down.

◐ Each layer has an Opacity value that multiplies layer's map before applying it.

# Raw_input



Raw Input generator was designed to import .raw heightmaps as a base for MapMagic terrains. In order to use it the .raw file should be properly prepared. It should be:

- square (width should be equal to length)
- grayscale
- 16 bits per channel
- and it should be saved as .raw file with PC byte order It does not matter where the heightmap is saved, whether it is the Assets folder or not, it will be converted to matrix format on import and saved within the scene. A saved .raw file can be added with the help of the 'Browse' button. Please note that MapMagic does not keep track of heightmap changes, so the 'Refresh' button should be pressed after any change.

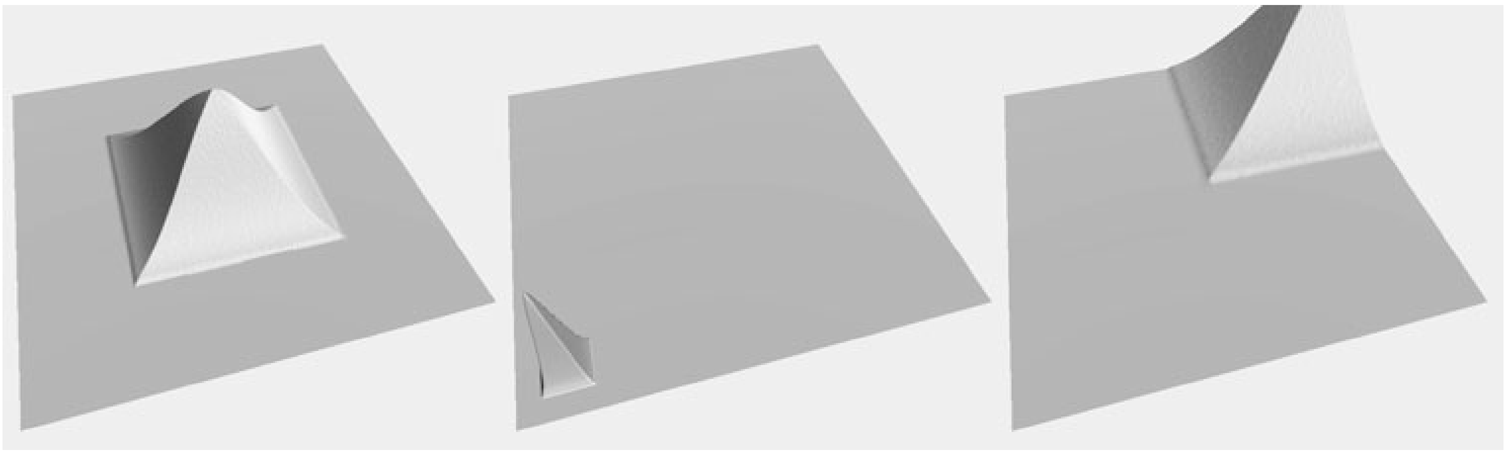
Once the RAW File is loaded it is **strongly recommended** to make an asset of it using "Save Imported RAW" button, otherwise it will not be loaded on scene load and will not work in playmode and build.

## Outputs:

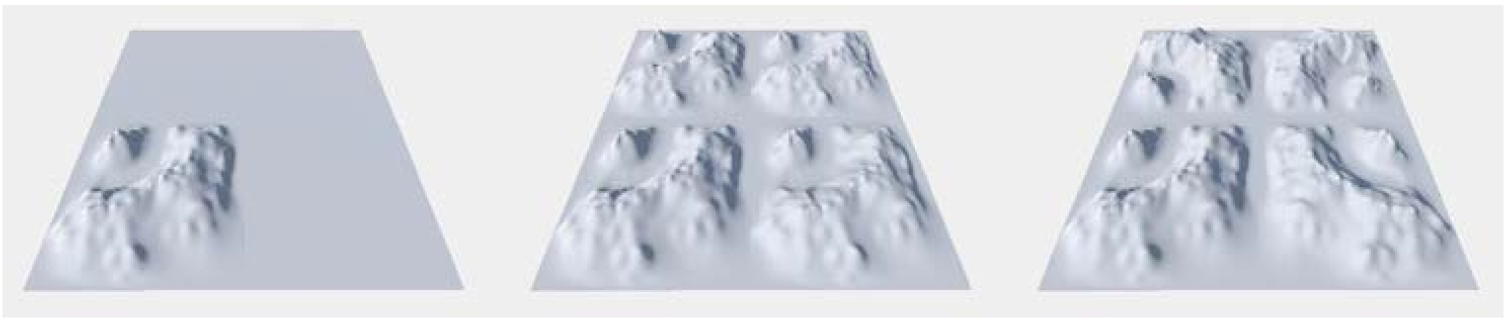
- ● Output - the RAW image applied as a MapMagic map.

## Properties:

- Intensity: the amount of influence of the generator on the input map. When the value is set to 0 the generator effect is not visible. In other words, it is the generator's effective opacity.
- Scale: sets the size of heightmap. By default size is equal to one chunk (terrain).

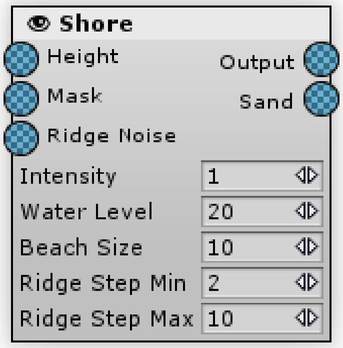

*Scale: 1, 0.2, 2*

- Offset: moves heightmap along X or Z-axis (in world units).
- Wrap Mode: determines how the heightmap should be repeated so that every terrain point will not miss a map:
    - Once: repeat map only once
    - Tile: repeat map endlessly the way the standard texture could be tiled
    - PingPong: will mirror every 2nd map and repeat map endlessly. Use this if you want to fill your land with a heightmap that is not tilable.


*Wrap Mode: Once, Tile, PingPong*

> **If your terrain is not pinned in 0,0 then you might not see any RAW Input effect by default. This happens because it is tiled once at the scene origin. Set Wrap Mode to Tile or PingPong or adjust the Offset to see the RAW Input effect**

## Shore


Shore

Height
Mask
Ridge Noise

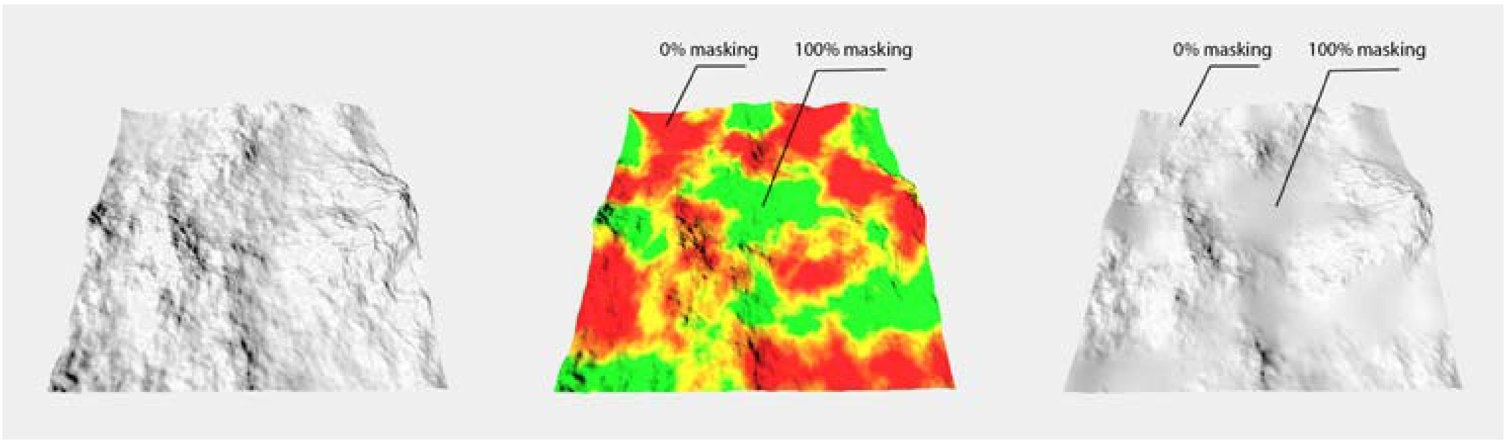
Output
Sand

| | |
|---|---|
| Intensity | 1 |
| Water Level | 20 |
| Beach Size | 10 |
| Ridge Step Min | 2 |
| Ridge Step Max | 10 |

Creates a shore line with a beach and a ridge.

### Inputs:

- ● Input - the heightmap to be processed by the generator.
- ● Mask - multiplies the generator intensity by the mask map's value. For the mask pixels that have a value of 0 the generator effect is invisible, for the pixels that have a value of 1 the intensity is unchanged.



| *Source heightmap* | *Mask map* | *Masking result* |

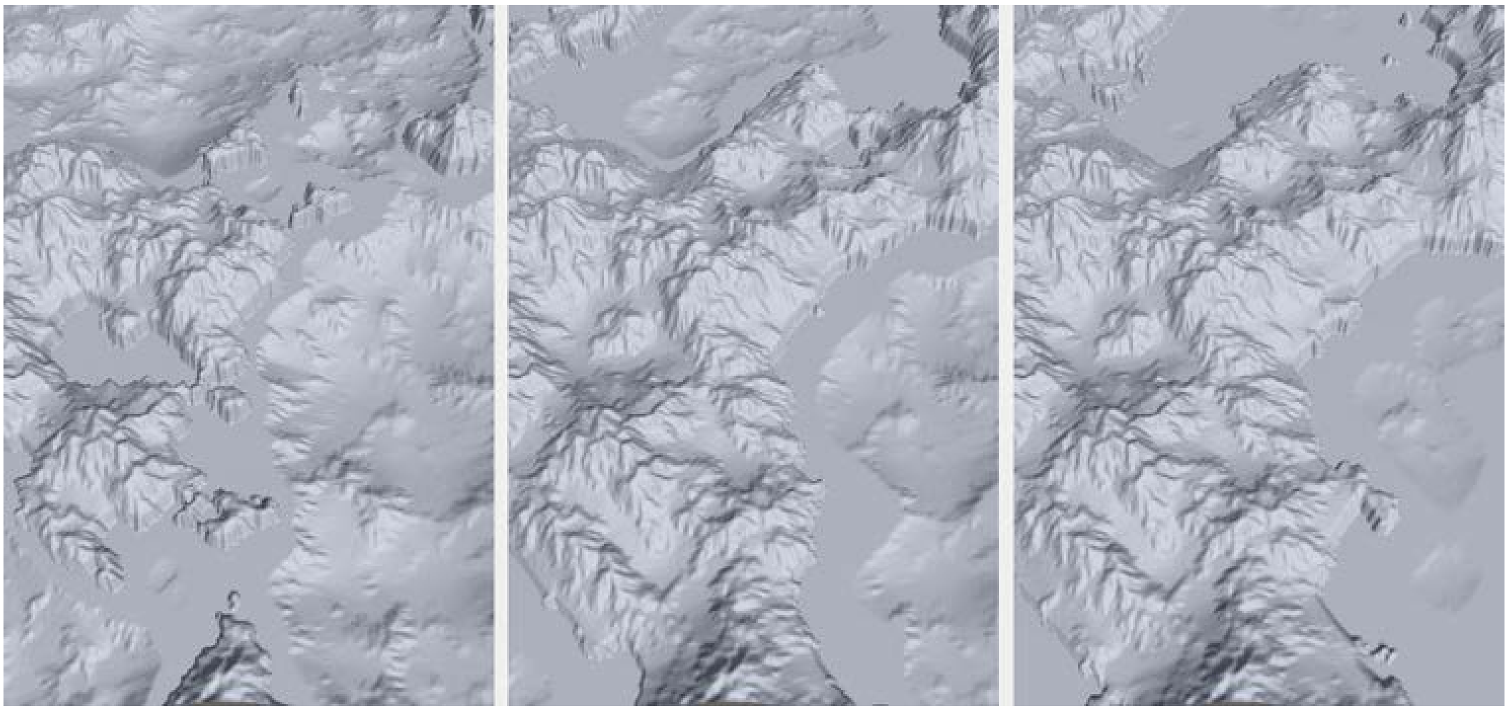
- Ridge Noise - a map to control diversity of the ridge. Usually just a noise map used here just to make a ridge less homogeneous.

### Outputs:

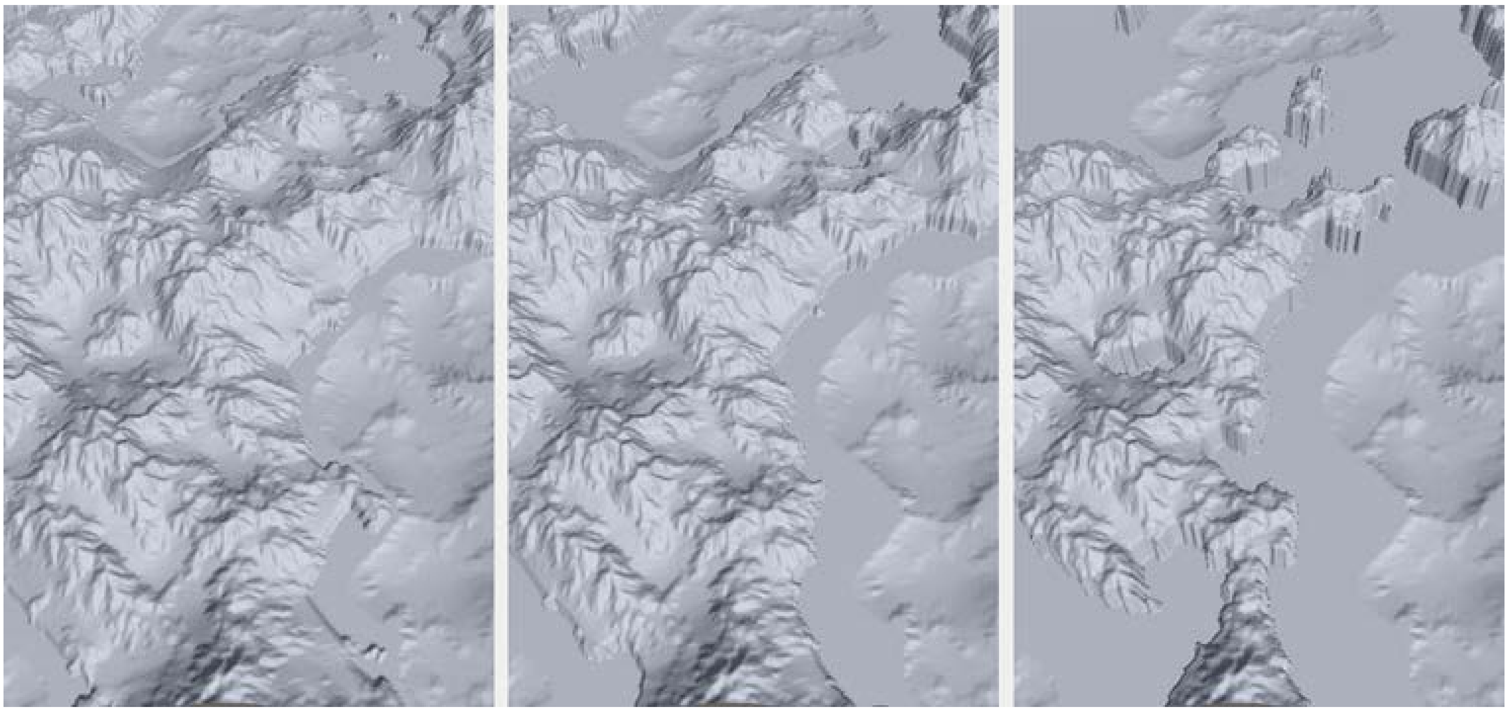
- ● Output - stores the generator's processing result.
- ● Sand - a map for the Texture Output or mask for placing beach and underwater objects. Determines the beach sand (or pebbles) opacity.

### Properties:

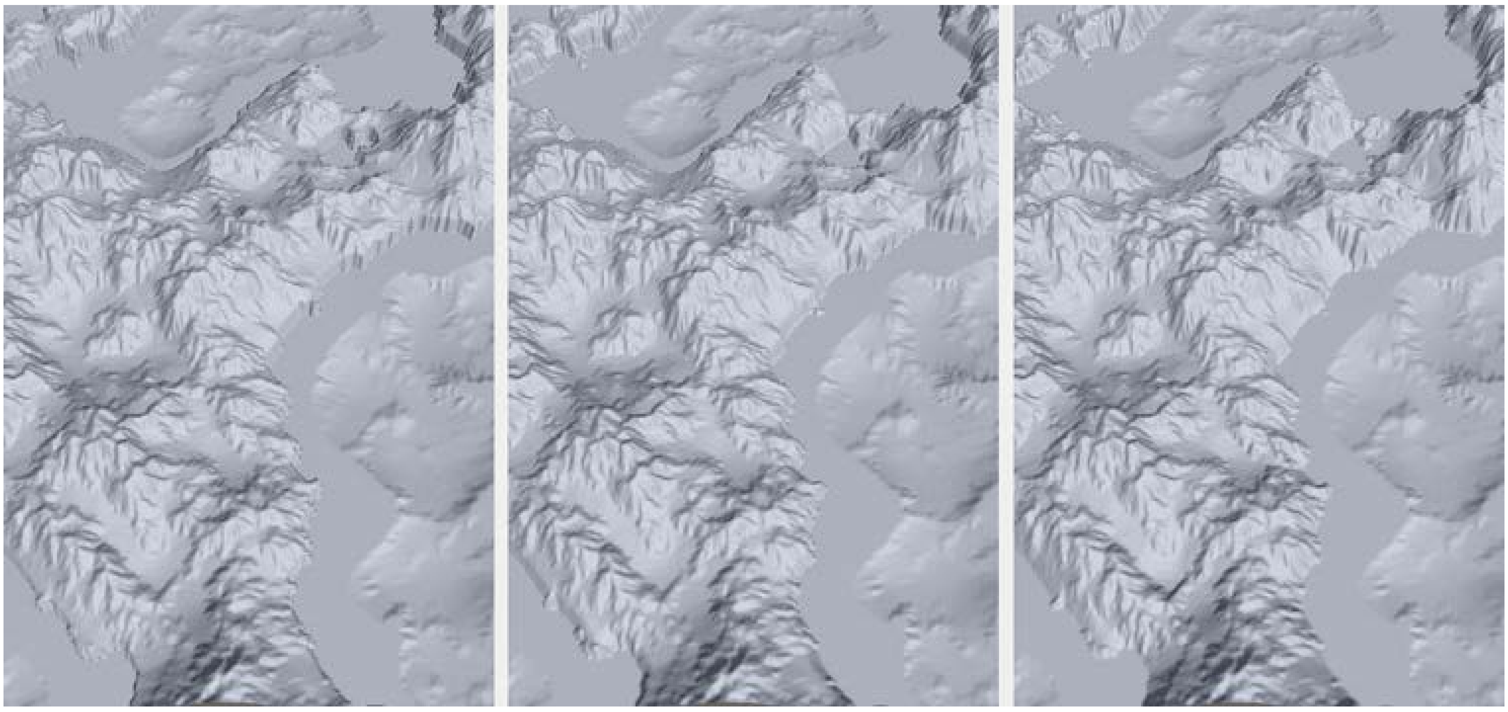
- Intensity: the influence of the generator. Hint: this parameter can also modify the beach incline (setting the lesser value makes the beach more inclined).
- Water Level: the level of the beach from the MM object pivot, in units.


*Water Level: 100, 50, 10*

- Beach Size: the size of the beach

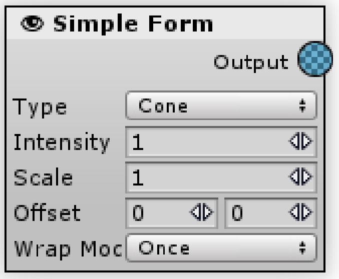

*Beach Size: 10, 20, 40*

- Ridge Step (Min): the steepness of the ridge. When the Ridge Noise map is assigned it is used as a ridge minimum steepness.


*Ridge Step: 0, 10, 20*

- Ridge Step Max: the maximum ridge steepness when the Ridge Noise map is assigned. When no Ridge Noise map is assigned this property is not used.

# Simple_form


Simple Form
Output
Type: Cone
Intensity: 1
Scale: 1
Offset: 0 0
Wrap Moc: Once

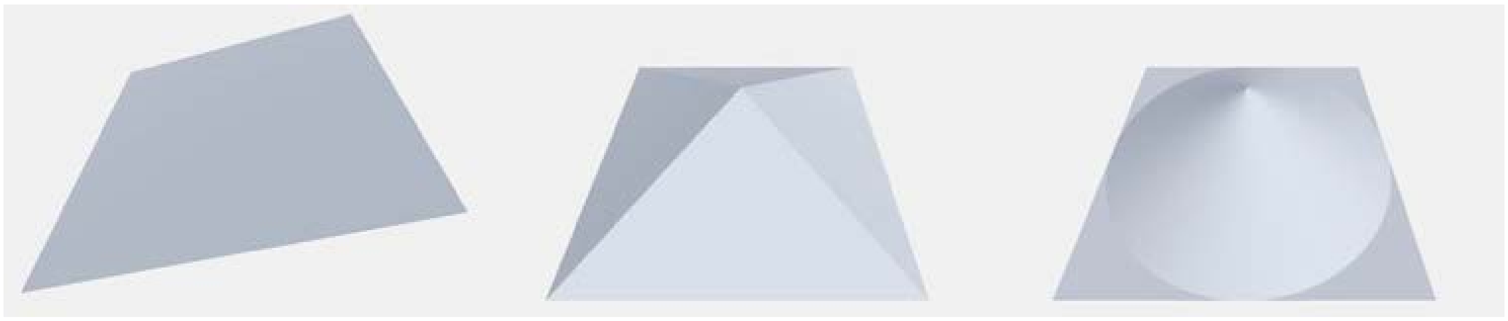
Easy way to create simple heightmap object. Could be used to mask points of the compass with the Gradient or create an island base with the Cone.

## Outputs:

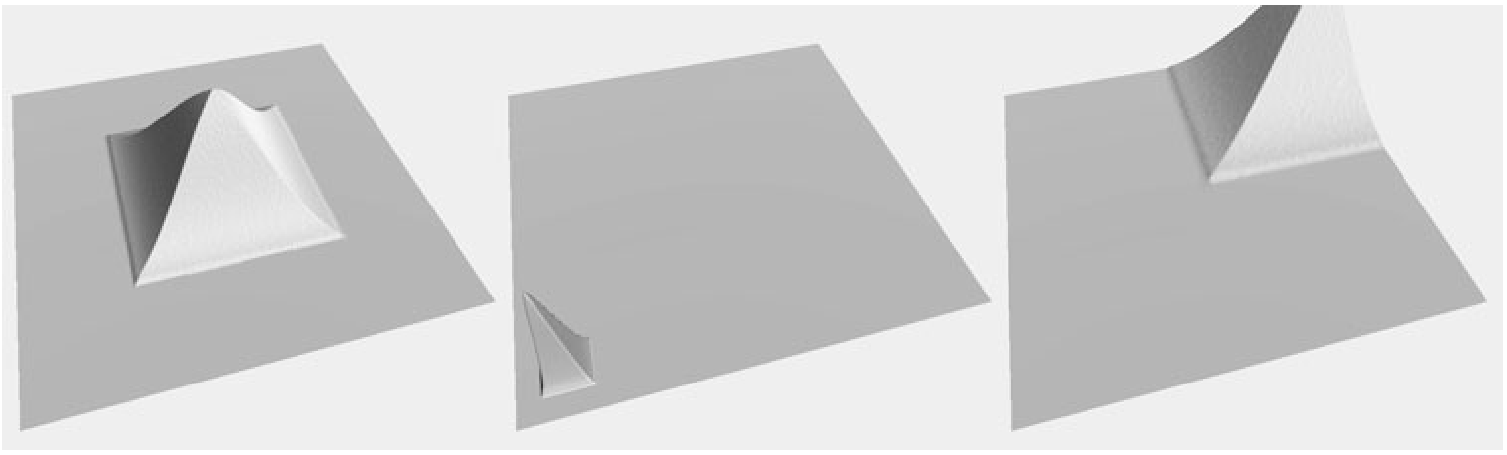
- ● Output - stores the generator's processing result.

## Properties:

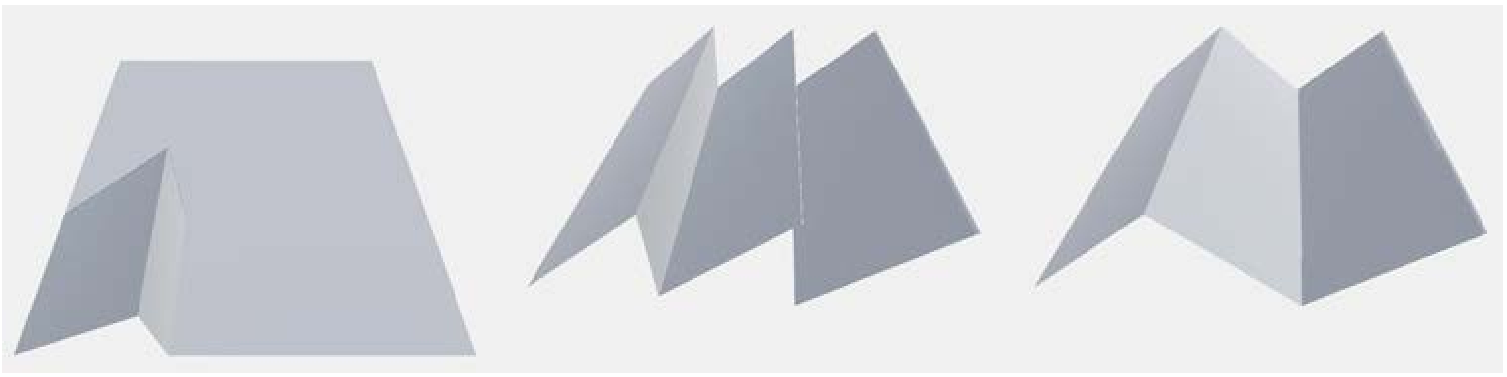
- Type:
  - Gradient X: fills a map with an x-axis aligned gradient.
  - Gradient Z: fills a map with an z-axis aligned gradient.
  - Pyramid: creates a pyramidal form.
  - Cone: creates a cone.


*Type: Gradient, Pyramid, Cone*

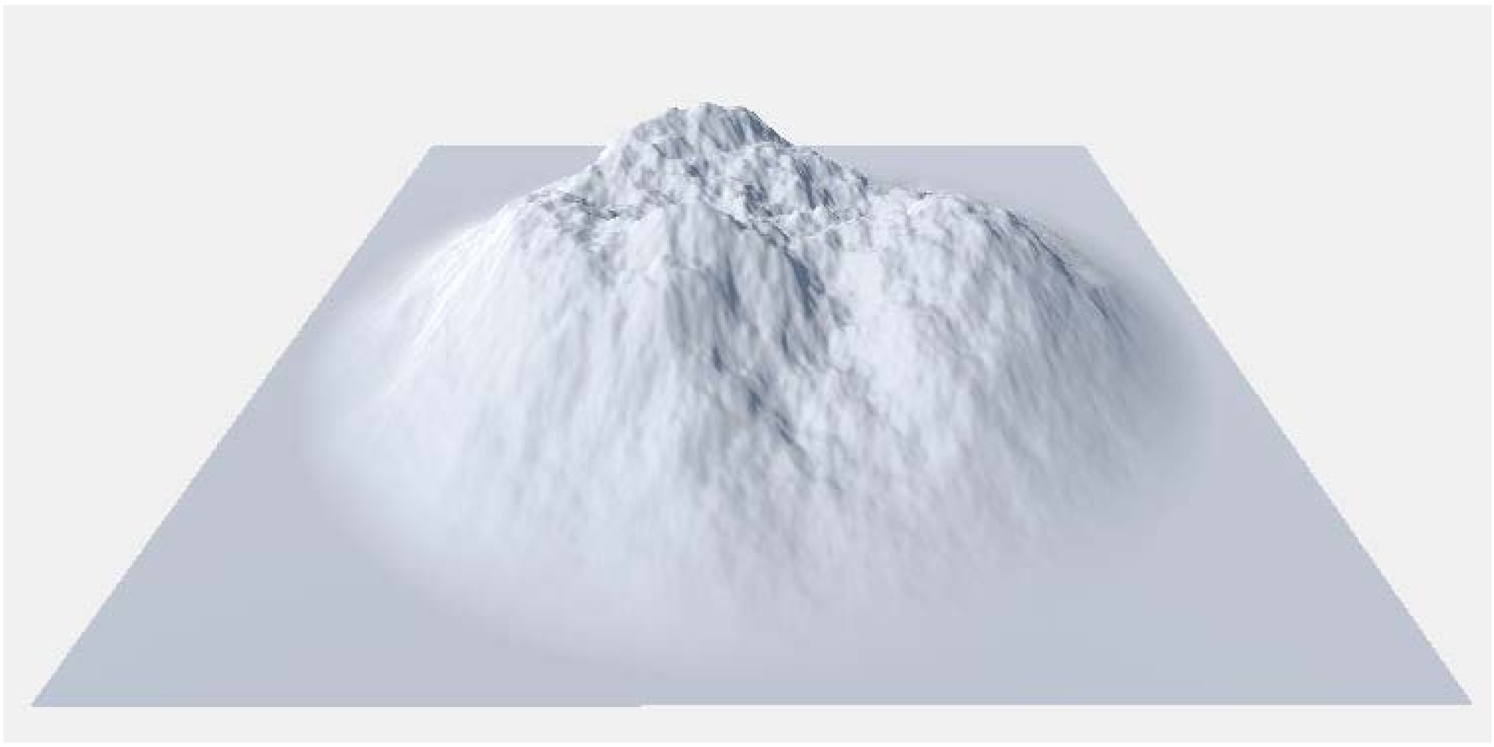
- Intensity: the height of the simple form figure.
- Scale: sets the size of heightmap. By default size is equal to one chunk (terrain).

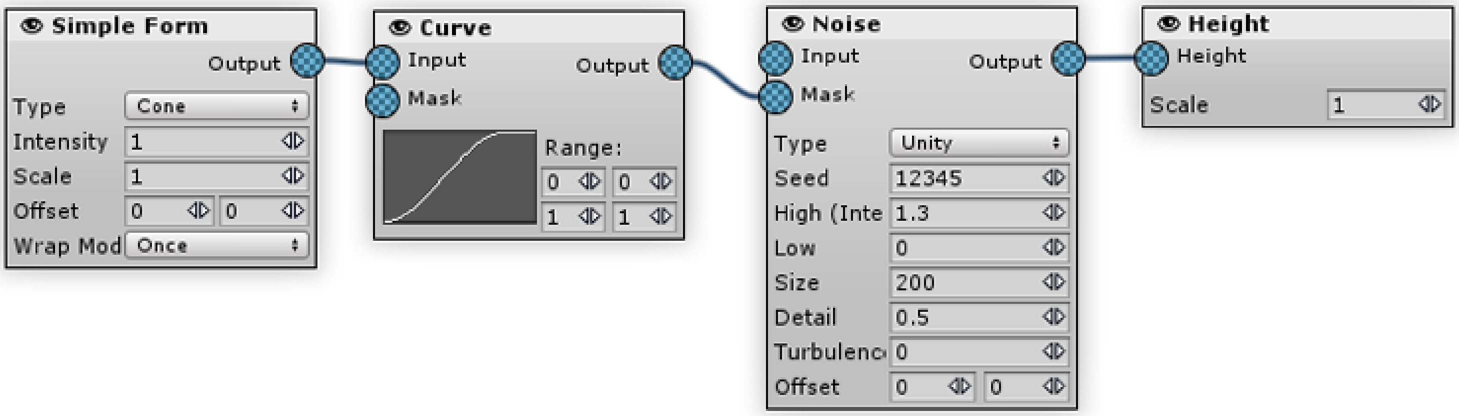

*Scale (pyramid, adjusted by Curve): 1, 0.2, 2*

- Offset: moves figure along X or Z-axis (in world units).
- Wrap Mode: determines how the heightmap should be repeated so that every terrain point will not miss a map:
    - Once: repeat map only once
    - Tile: repeat map endlessly the way the standard texture could be tiled
    - PingPong: will mirror every 2nd map and repeat map endlessly. Use this if you want to fill your land with a figure that is not tilable.


*Wrap Mode: Once, Tile, PingPong*

## Usage Examples:





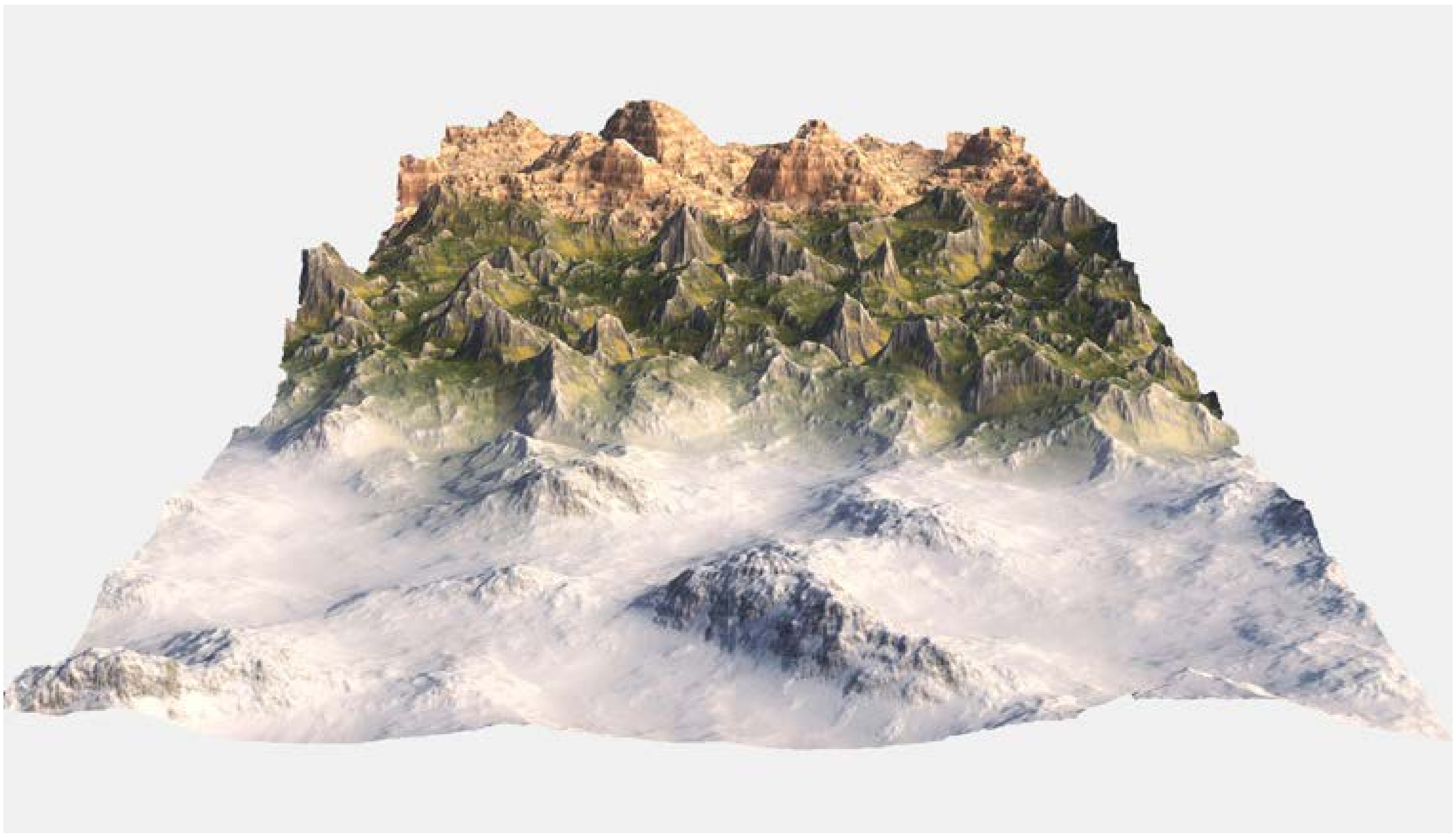
Simple Form
Output
Type: Cone
Intensity: 1
Scale: 1
Offset: 0 0
Wrap Mod: Once

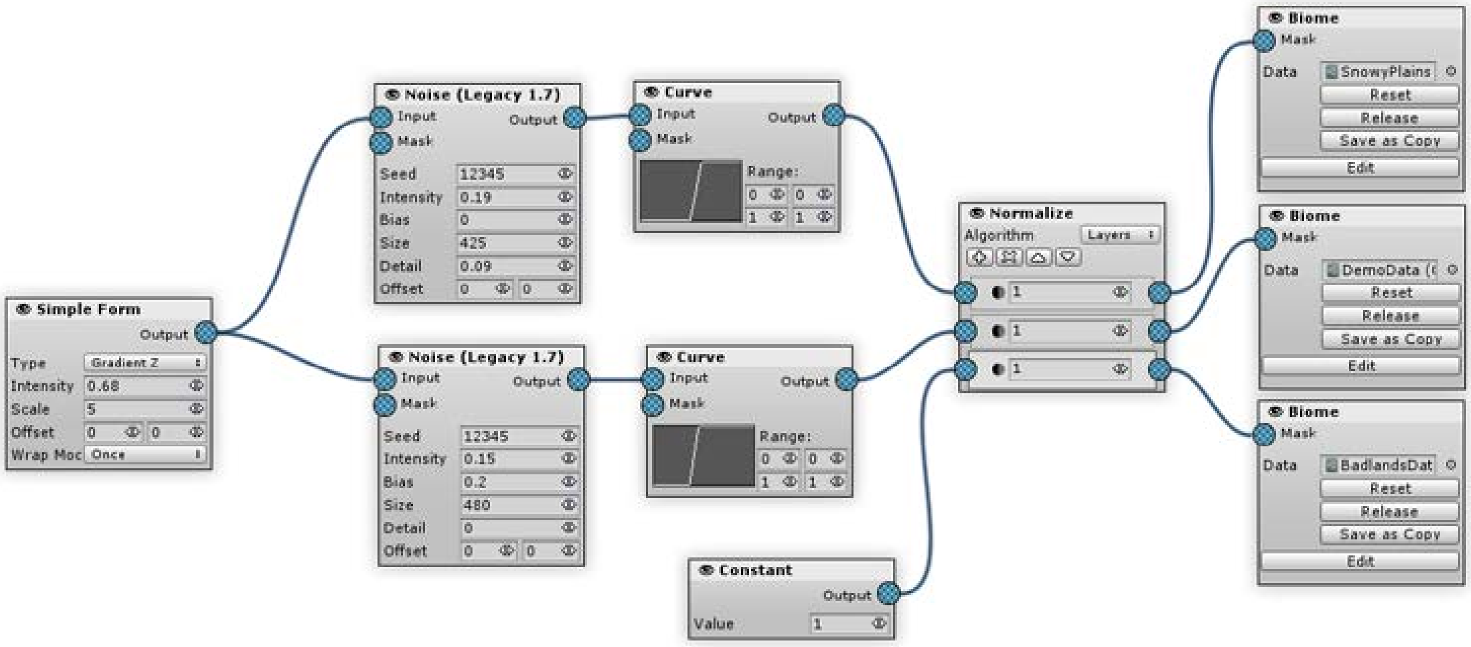
Curve
Input
Mask
Output
Range:
0 0
1 1

Noise
Input
Mask
Output
Type: Unity
Seed: 12345
High (Inte: 1.3
Low: 0
Size: 200
Detail: 0.5
Turbulenc: 0
Offset: 0 0

Height
Height
Scale: 1

*A base for the island*





Simple Form
Output
Type: Gradient Z
Intensity: 0.68
Scale: 5
Offset: 0 0
Wrap Moc: Once

Noise (Legacy 1.7)
Input / Mask / Output
Seed: 12345
Intensity: 0.19
Bias: 0
Size: 425
Detail: 0.09
Offset: 0 0

Curve
Input / Mask / Output
Range: 0 0 / 1 1

Noise (Legacy 1.7)
Input / Mask / Output
Seed: 12345
Intensity: 0.15
Bias: 0.2
Size: 480
Detail: 0
Offset: 0 0

Curve
Input / Mask / Output
Range: 0 0 / 1 1

Constant
Output
Value: 1

Normalize
Algorithm: Layers
1
1
1

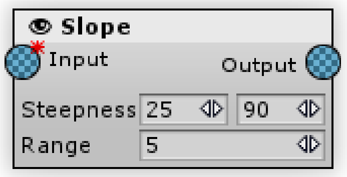
Biome
Mask
Data: SnowyPlains
Reset / Release / Save as Copy / Edit

Biome
Mask
Data: DemoData ((
Reset / Release / Save as Copy / Edit

Biome
Mask
Data: BadlandsDat
Reset / Release / Save as Copy / Edit

*To blend biomes from North to South*

## Slope



Generates the height difference map. For each map pixel it calculates the average height delta to four nearby pixels. Could be useful for separating horizontal and vertical surfaces (to fill them with grass and cliff respectively).

### Inputs:

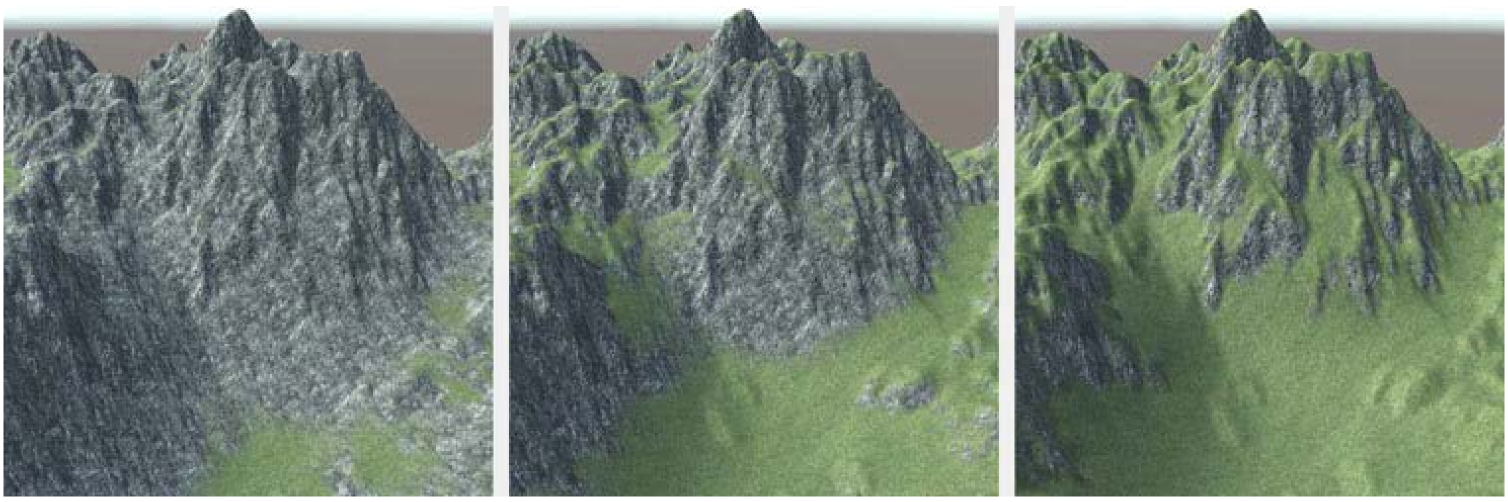
- Input: the default map to be processed by the generator.

### Outputs:

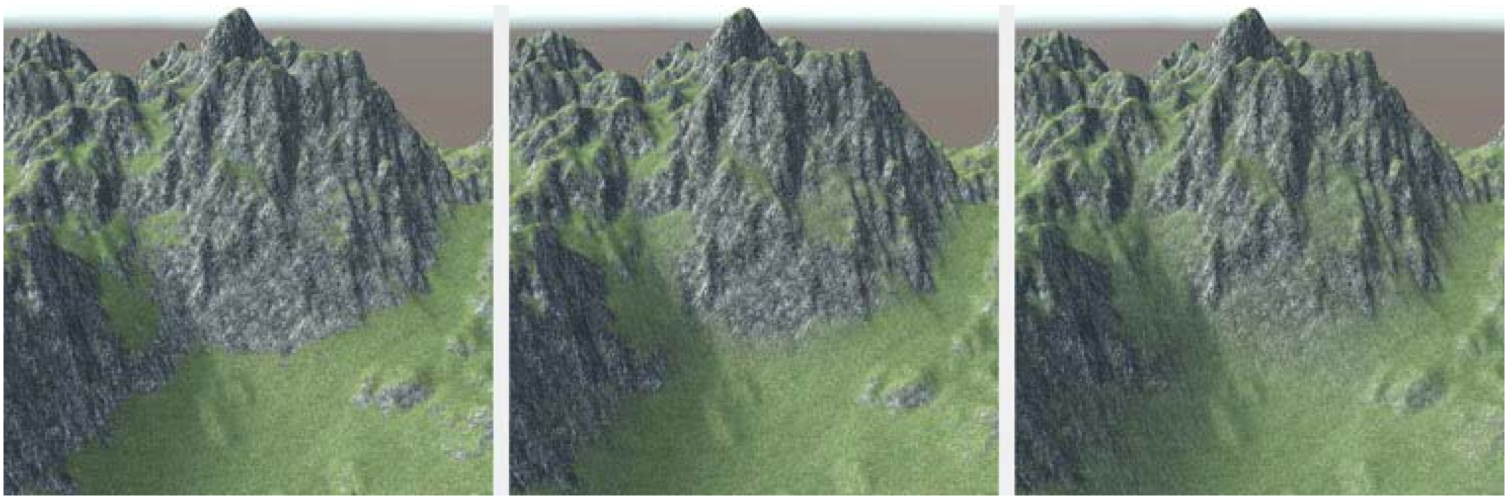
- Output: stores the generator's processing result.

### Properties:

- Steepness: defines the incline range that will be filled with slope map (in degrees). Minimum parameter sets the start incline of the stop, maximum - the end of the slope.


*Steepness: 5-90, 35-90, 80-90*

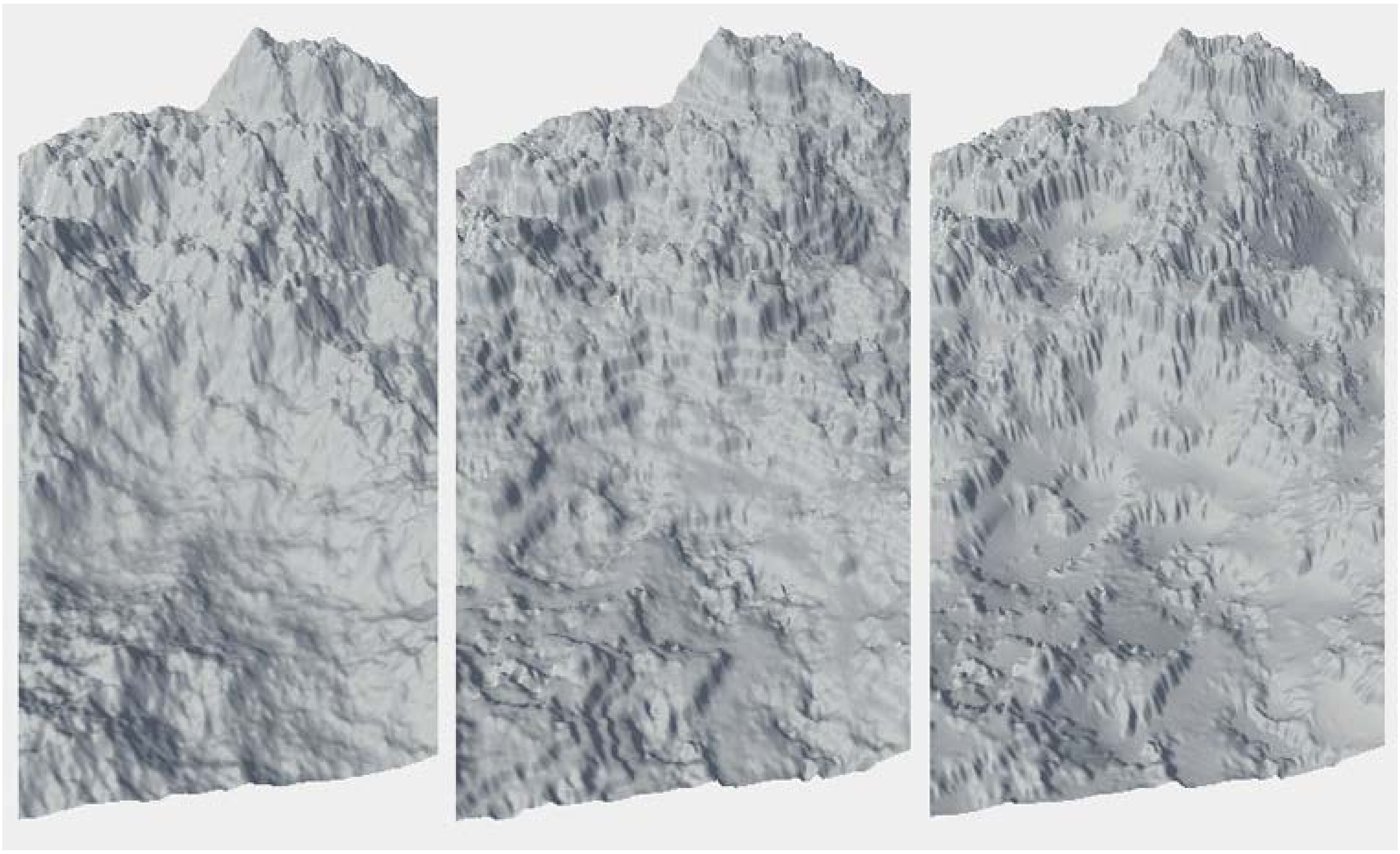
- Range: how gradual will be a slope transition (in degrees)



*Range: 1, 5, 15*

## Terrace

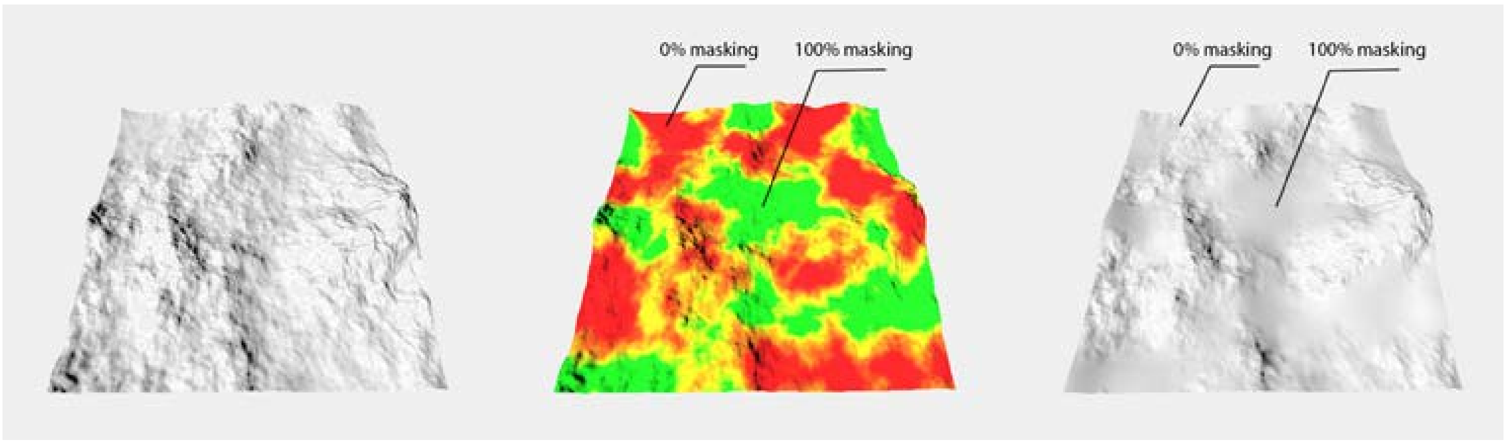
Produces step-like land forms on the slopes of hills. Used together with a Noise (as terrace's input) and Erosion (originates from terrace's output) Generator, it can produce fine landscapes. Don't underestimate the Terrace Generator - even if it seems that pure terrace output is not good enough looking, together with erosion it can make a terrain much more realistic and diverse.



| *Initial terrain* | *Terrace generator* | *Terrace+Erosion* |

### Inputs:

- Input - the default map to be processed by the generator.
- Mask - multiplies the generator intensity by the mask map's value. For the mask pixels that have a value of 0 the generator effect is invisible, for the pixels that have a value of 1 the intensity is unchanged.



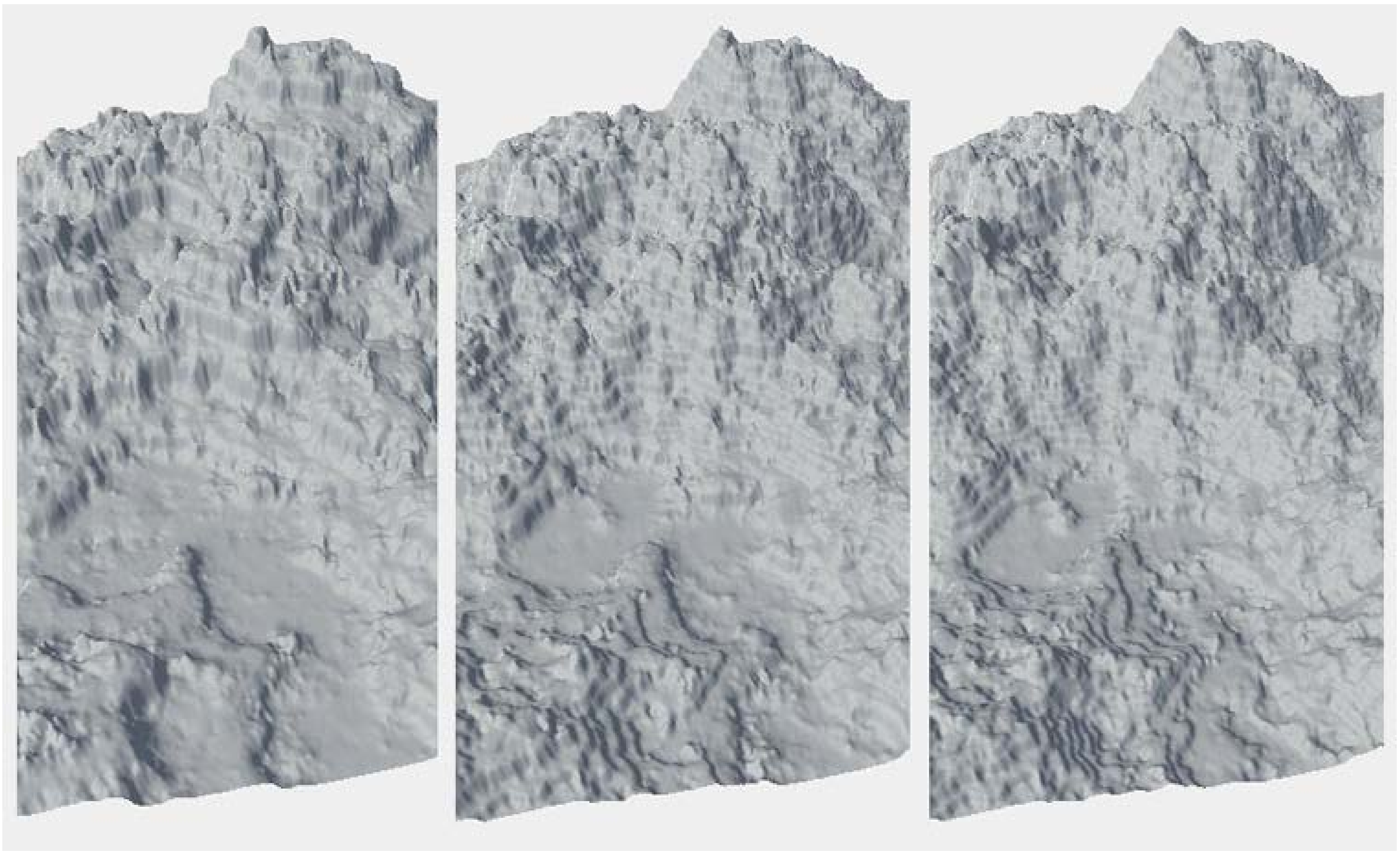
*Mask, using the example of **Blur** Generator:*
*| Source heightmap | Mask map | Masking result |*

## Outputs:

- Output - stores the generator's processing result.

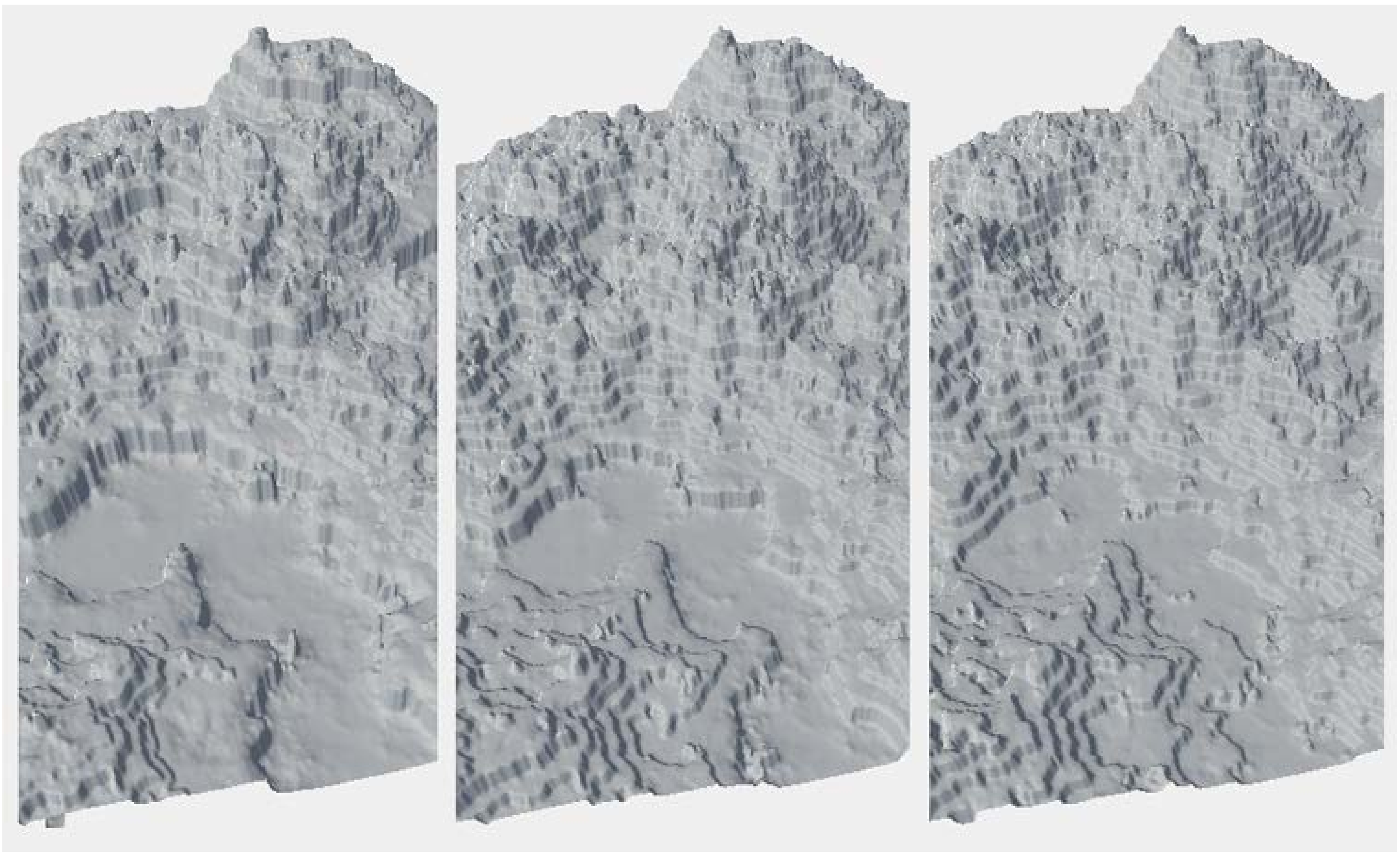
## Properties:

- Tread count: each of the terrace's flat surfaces is called a tread. This parameter sets the number of treads - from 2 (at the top and bottom of terrain) to any reasonable number (note that making a terrace height lesser than a unit often does not make sense). This parameter affects the generator performance but since Terrace Generator is relatively fast every value below 100 will not have an appreciable influence on total the generation time.

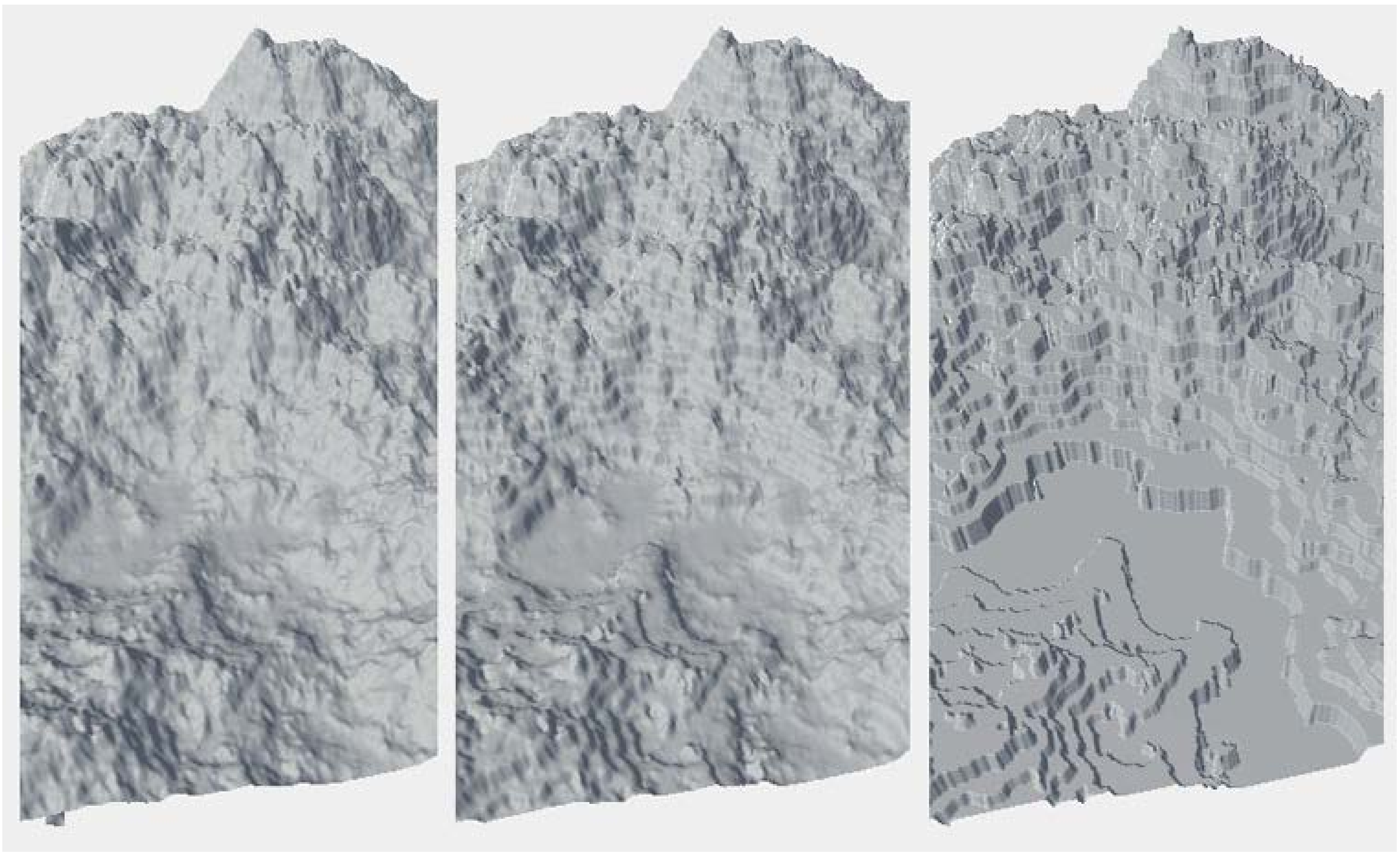


*Treads: 10, 20, 30*

- Uniformity: the terrace's step's height difference. When set to 1 all of the steps have the same height and when set to 0 the height may vary from zero to twice the uniform height.

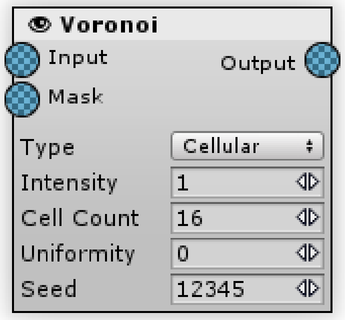

_Uniformity: 0, 0.5, 1

- Steepness: determines how inclined the step’s vertical plane(riser) is and how acute the bend between the tread and riser would be. A value of 1 will generate absolutely vertical risers with maximum tread length, while a value of 0 will set the tread’s length to zero (but that does not mean that terracing would be turned off - to make the terrace effect even less noticeable use the Intensity parameter).


*Steepness: 0, 0.5, 1*

- Intensity: how visible the effect of the generator is, i.e. generator opacity.

# Voronoi

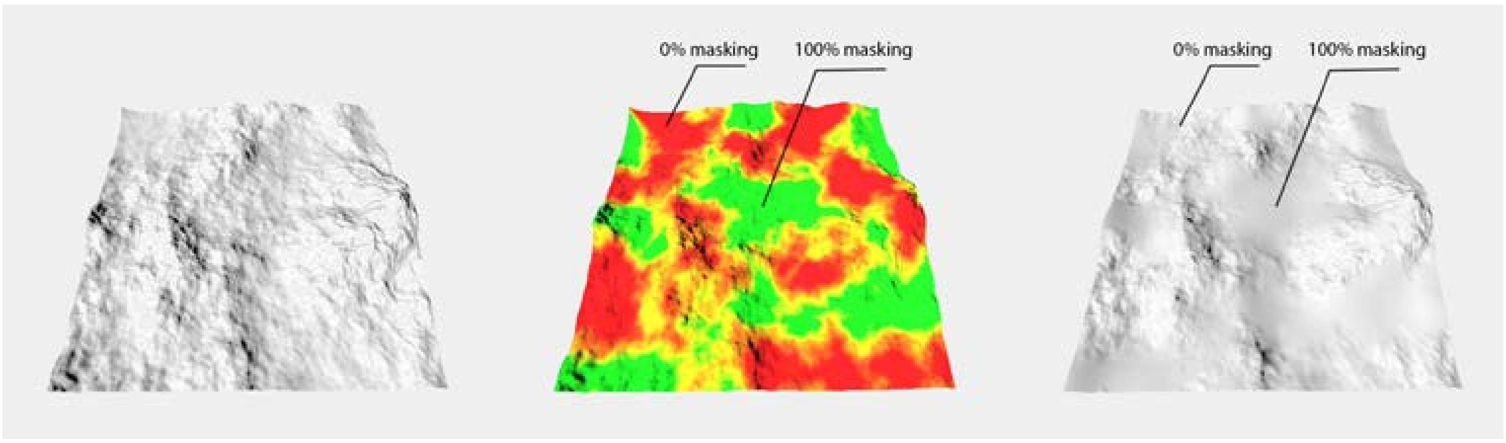


| | |
|---|---|
| Type | Cellular |
| Intensity | 1 |
| Cell Count | 16 |
| Uniformity | 0 |
| Seed | 12345 |

Uses a Voronoi pattern to create maps. Splits the map into cells, generates a random point for each cell, and fills the map using evaluated distances to the closest point and the second closest point. A Voronoi map looks like a mosaic composed of irregular convex polygons.

## Inputs:

- ● Input - if no input is specified Voronoi generator returns the pure Voronoi map. If an input is specified, the Voronoi effect is added (+) above the input map.
- ● Mask - multiplies the generator intensity by the mask map's value. For the mask pixels that have a value of 0 the generator effect is invisible, for the pixels that have a value of 1 the intensity is unchanged.



*Mask, using the example of **Blur** Generator:*
| *Source heightmap* | *Mask map* | *Masking result* |

## Outputs:

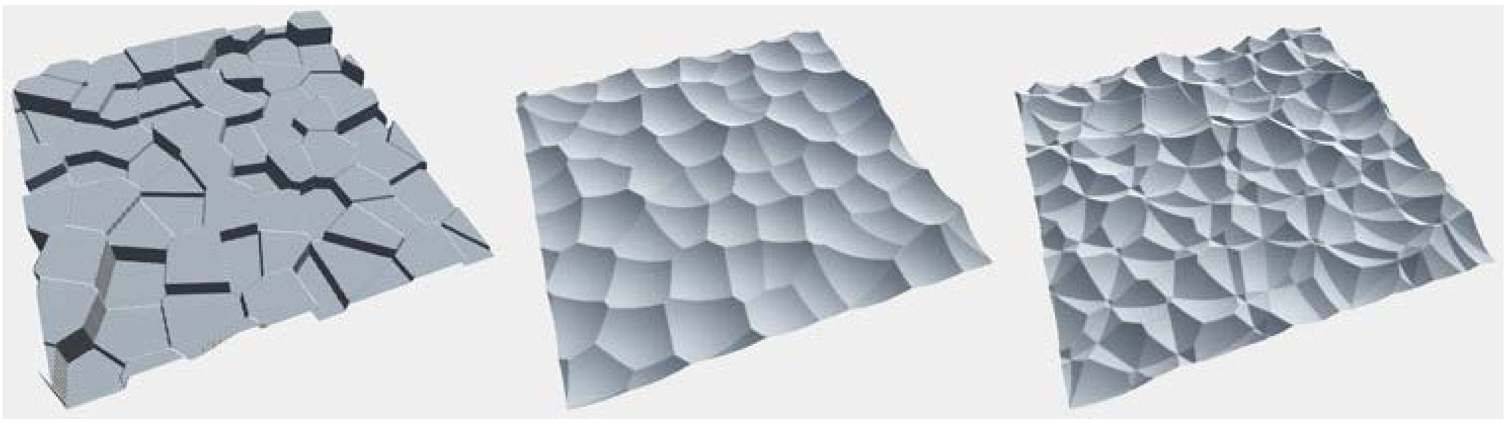
- Output - a voronoi pattern (if no input specified) or an input map with a voronoi pattern added (if input specified).

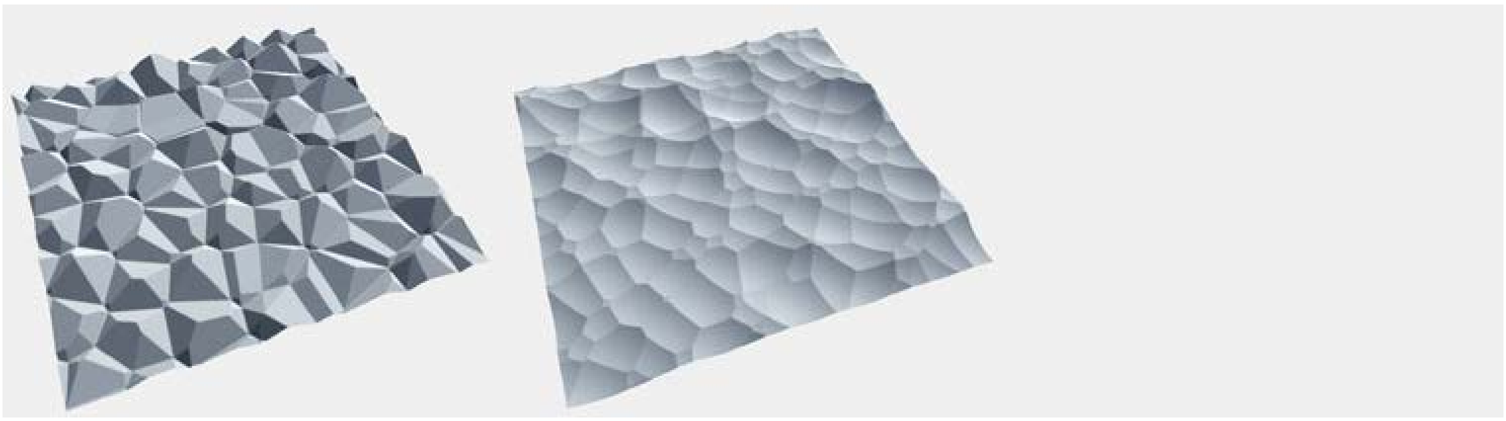
## Properties:

- Type: determines how exactly the final height is calculated:
  - Flat: fills all of the closest pixels with cell point's value
  - Closest: fills pixel with distance to the closest point value
  - Second Closest: same as Closest, except the distance to the second closest point is calculated

Cellular: distance to the second closest point minus the distance to the closest point.

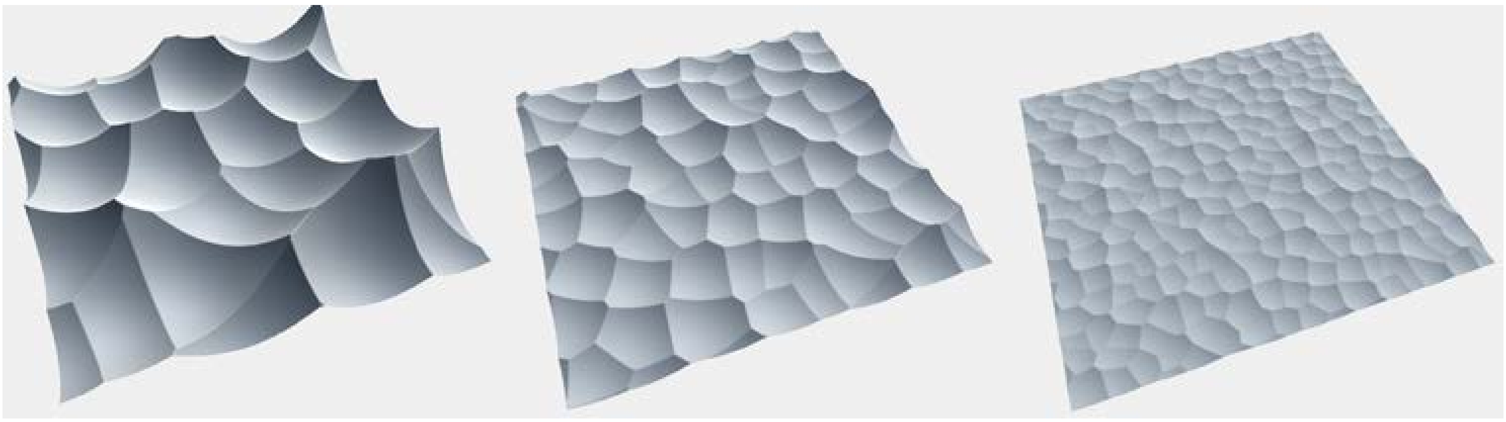
- Organic: distance to the second closest point plus the distance to the closest point.


*Flat, Closest, Second Closest*

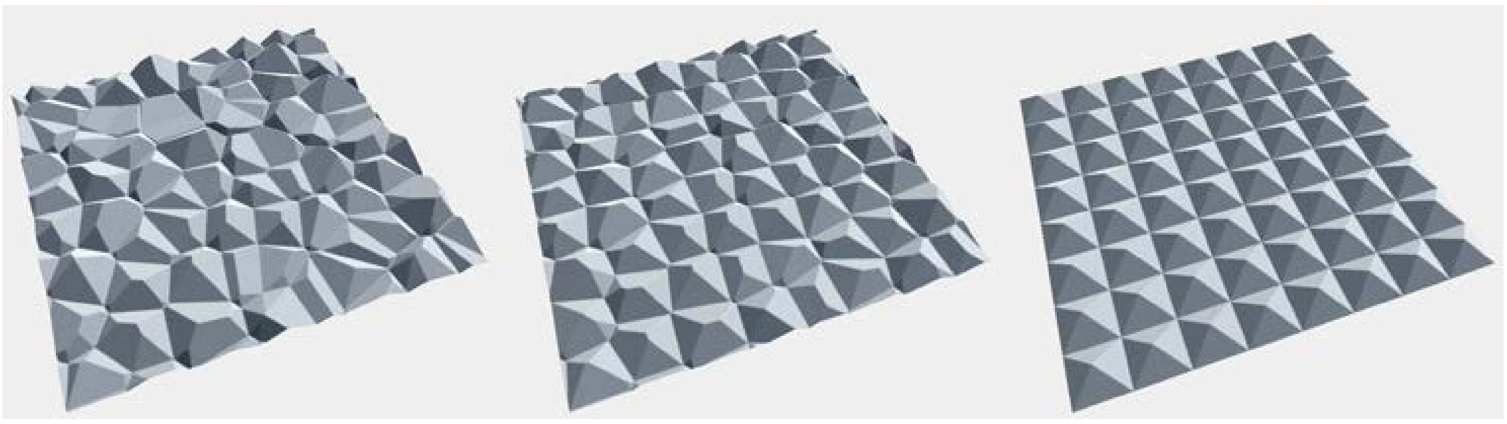

*Cellular, Organic*

- Intensity - the amount of influence of the generator on the input map. When the value is set to 0 the generator effect is not visible. In other words, it is the generator's effective opacity.

- Cell count: the quantity of cells. This parameter sets the size of the Voronoi pattern: The larger the count, the smaller the polygon size. For better terrain welding it is recommended to use a power of two cell count, however it is not mandatory.


*Cell Count: 4, 8, 16*

- Uniformity: determines the offset of the cell's point, and as a consequence, the diversity of the polygons. With a value of 1 all polygons are equal, and they get more unique with the decrease in Uniformity value.


*Uniformity: 0, 0.5, 1*

- Seed: a number used to initialize pseudo-random Voronoi generator.

# Adjust

Modifies the height, rotation and size parameters of the objects. Note that object rotation and scale parameters are not taken into account by Object Output unless it has “Rotate” and “Scale” checkboxes turned on.

## Inputs:

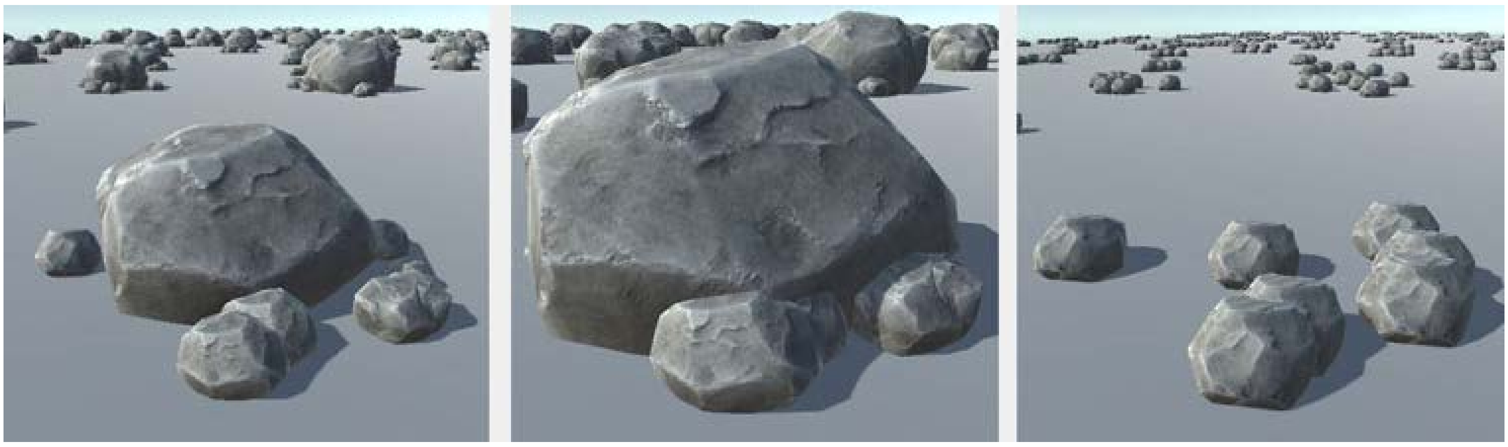
- Input - the default object list (hash) to be processed by the generator.
- Intensity - a map that controls a generator’s level of intensity. If some object is placed at the map pixel with value of 0, no adjust will be applied to it. If the pixel value is 1 then it will be adjusted using full intensity.

## Outputs:

- Output - stores the generator’s processing result.

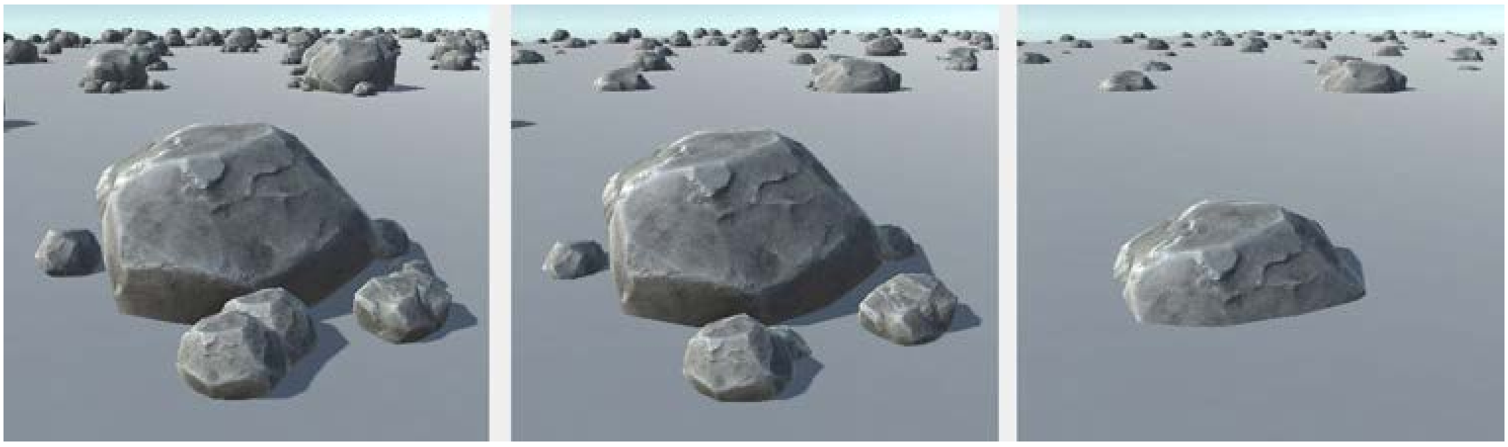
## Properties:

- Type: when the “Relative” type is selected all of the adjustment changes are applied to the existing object’s height, rotation and size. The “Absolute” type resets the old values and then places (and scales) the object as if it was placed on zero level, having no rotation and having a scale of 1.

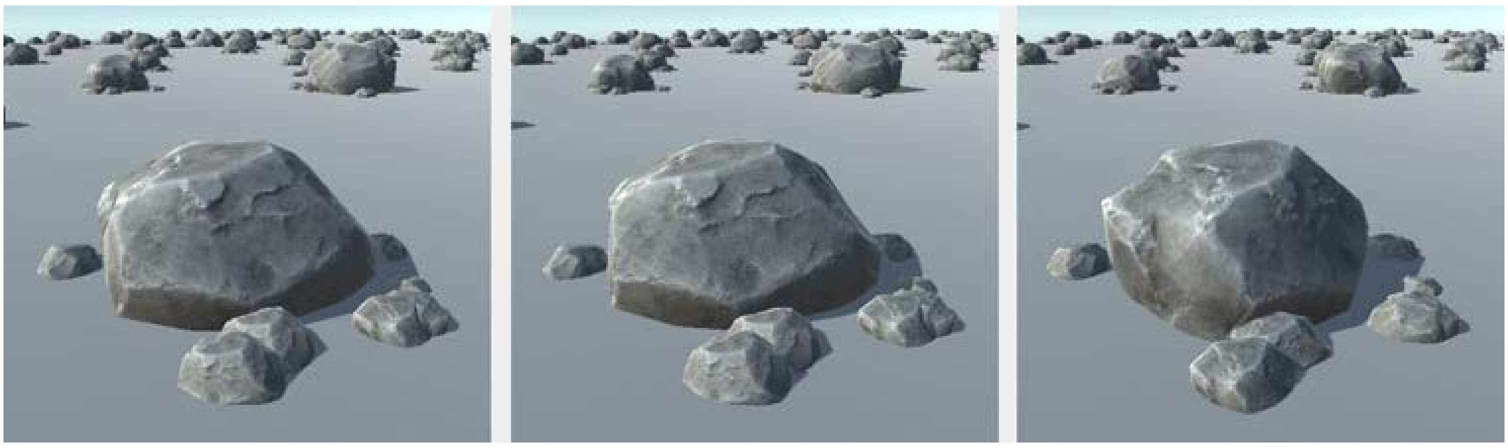


| *Initial Scene* | *Relative Size Adjustment (1.5x)* | *Absolute Size Adjustment (1.5x)* |

- Height, Rotation, Size parameters: all of these have minimum and maximum values, The resulting value will be selected at random and will lie within the minimum and maximum.



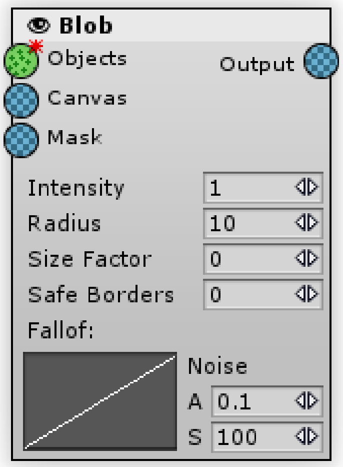
*| No height adjustment | Height: -5, 0 | Height: -5, -5 |*



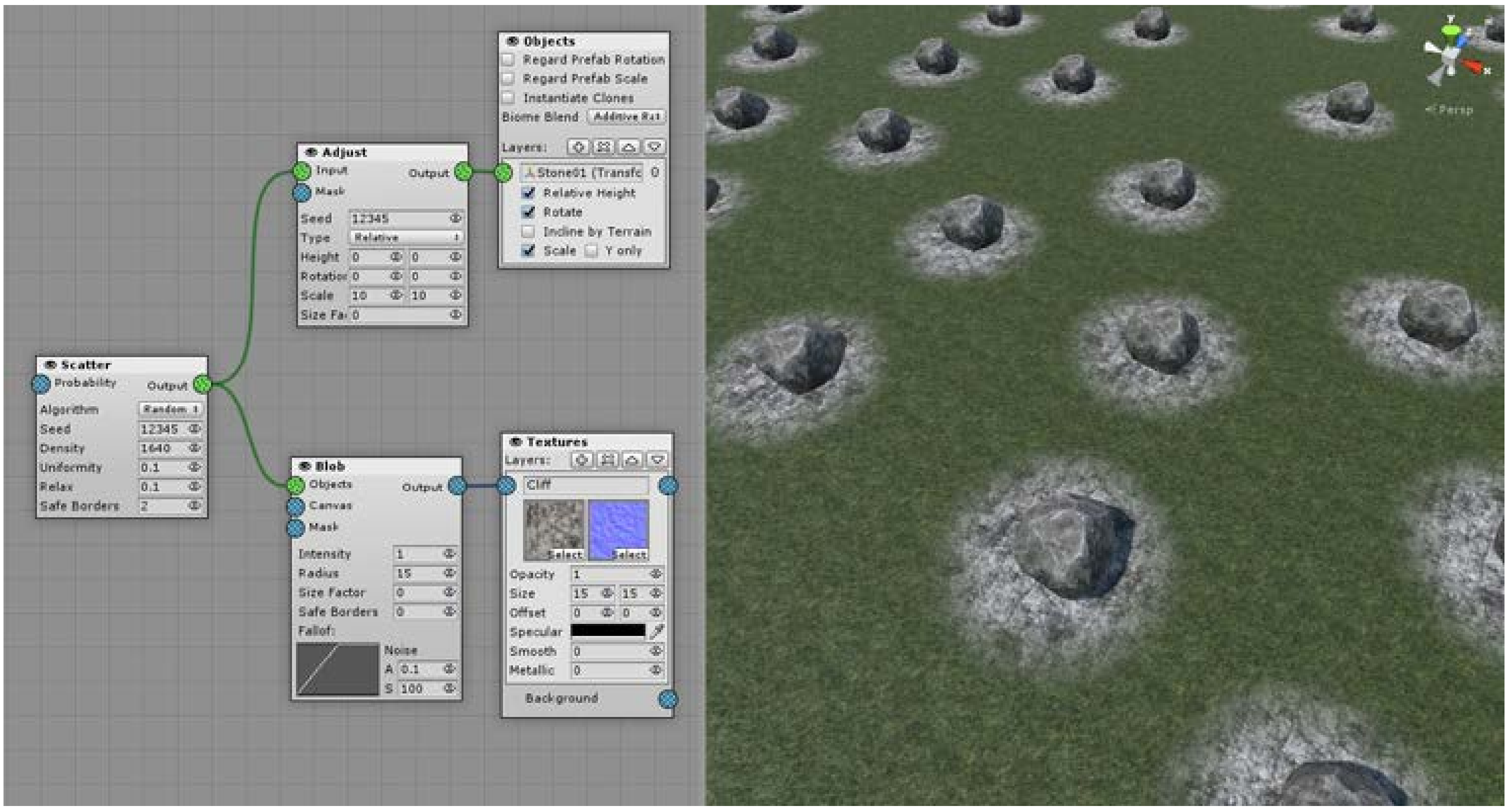
*| No rotation adjustment | Rotation: 0-10 | Rotation: 0-360 |*

- Size Factor: multiplies the changes applied by this generator depending on the initial object size. For example, if the object's size is 2 then all of the adjustment values will be doubled if the factor is equal to 1. This is useful for proportional object changes: if the object should be lowered to half of it's height, for instance. When the value is 0 all of the objects are adjusted the same regardless of their size.

# Blob


Blob
Objects
Output
Canvas
Mask
Intensity 1
Radius 10
Size Factor 0
Safe Borders 0
Fallof:
Noise
A 0.1
S 100

Blob Generator draws circles on the map map in places where objects are located. It can be used for painting maps under objects.


Scatter
Probability
Output
Algorithm Random
Seed 12345
Density 1640
Uniformity 0.1
Relax 0.1
Safe Borders 2

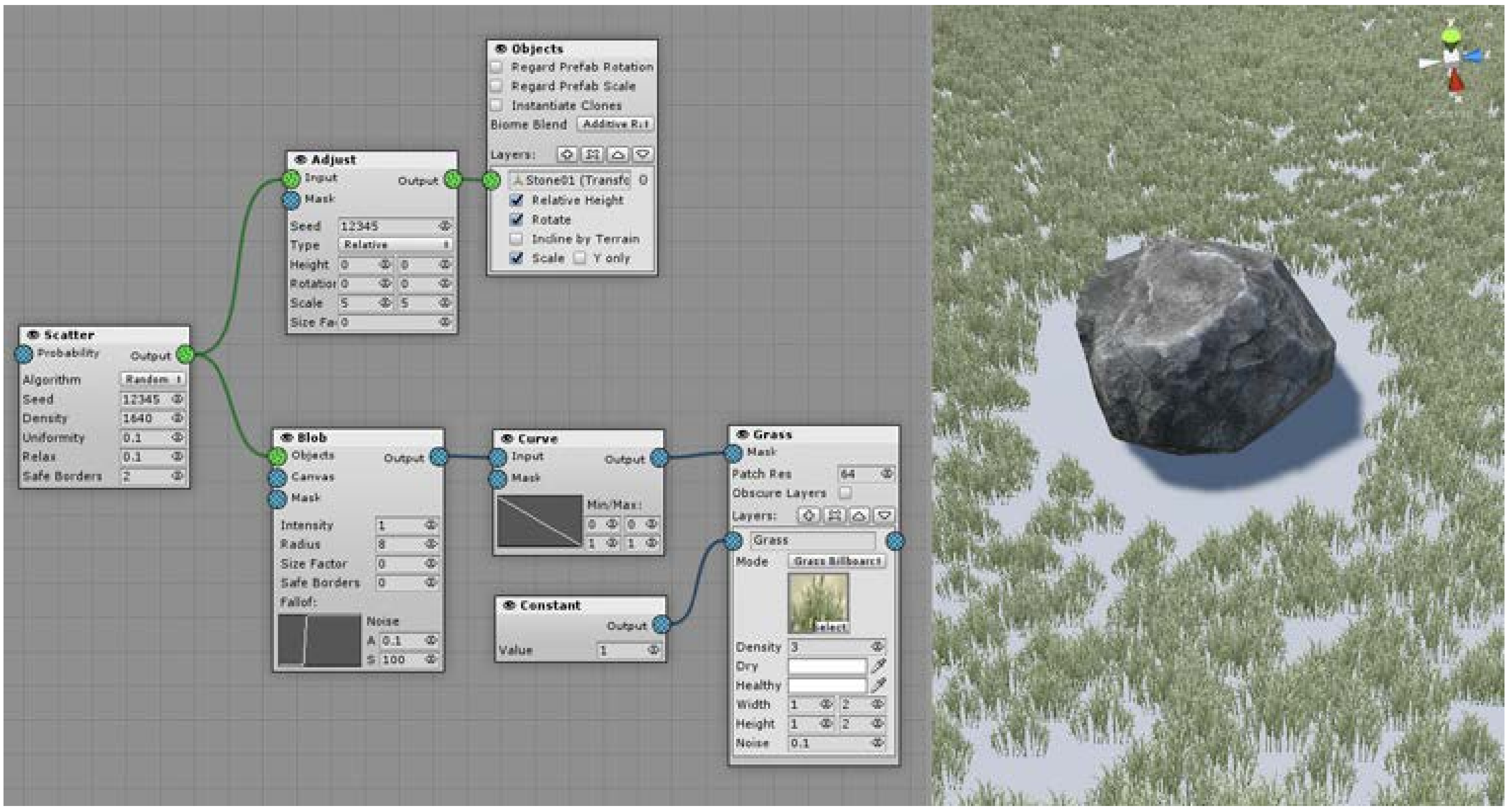
Adjust
Input
Output
Mask
Seed 12345
Type Relative
Height 0 0
Rotation 0 0
Scale 10 10
Size Fa 0

Objects
Regard Prefab Rotation
Regard Prefab Scale
Instantiate Clones
Biome Blend Additive Rst
Layers:
Stone01 (Transfc 0
✓ Relative Height
✓ Rotate
Incline by Terrain
✓ Scale Y only

Blob
Objects
Output
Canvas
Mask
Intensity 1
Radius 15
Size Factor 0
Safe Borders 0
Fallof:
Noise
A 0.1
S 100

Textures
Layers:
Cliff
Select
Select
Opacity 1
Size 15 15
Offset 0 0
Specular
Smooth 0
Metallic 0
Background

Blob Generator could be also used for creating special masks, for example, a grass mask. to avoid grass growing under the objects:



## Inputs:

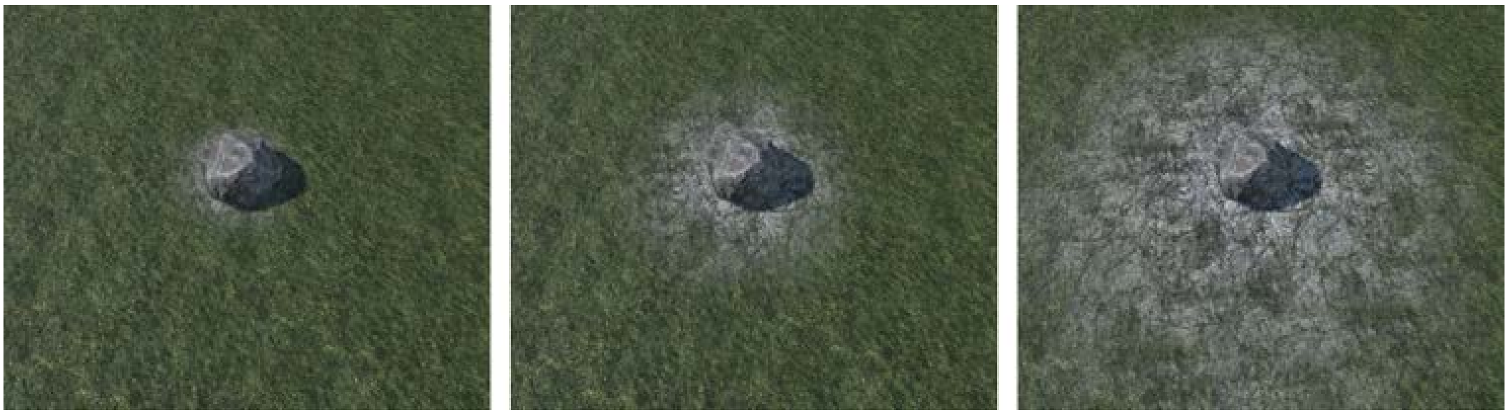
- Objects (mandatory): objects that would be the centers of the flat areas.
- Canvas: the map to stamp blobs. If not specified then blank map will be used.
- Mask: the map that controls generator's intensity.

## Output

Input canvas with object stamps.

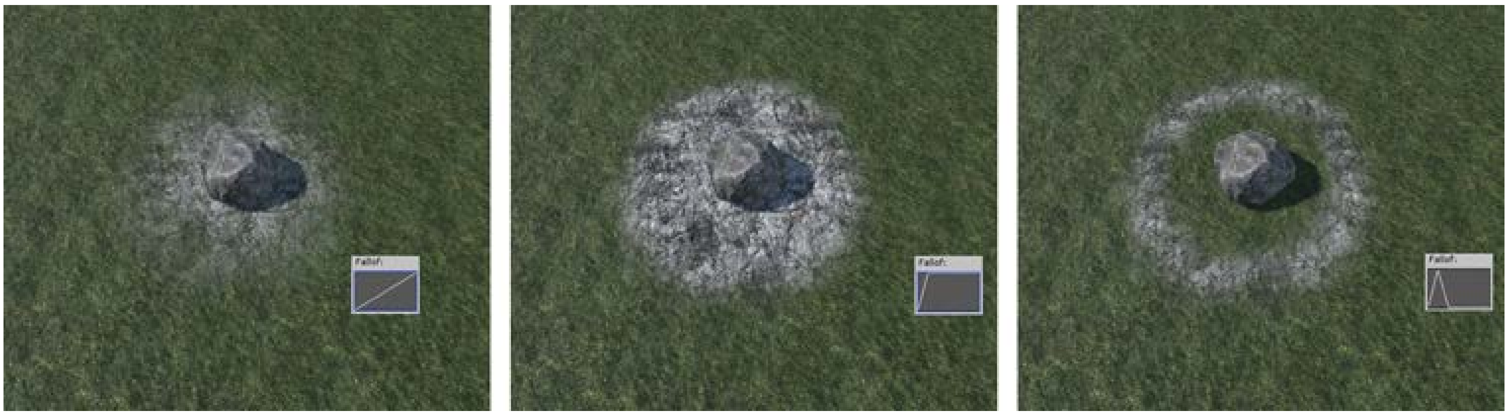
## Properties:

- Radius: the size of the blob stamps.

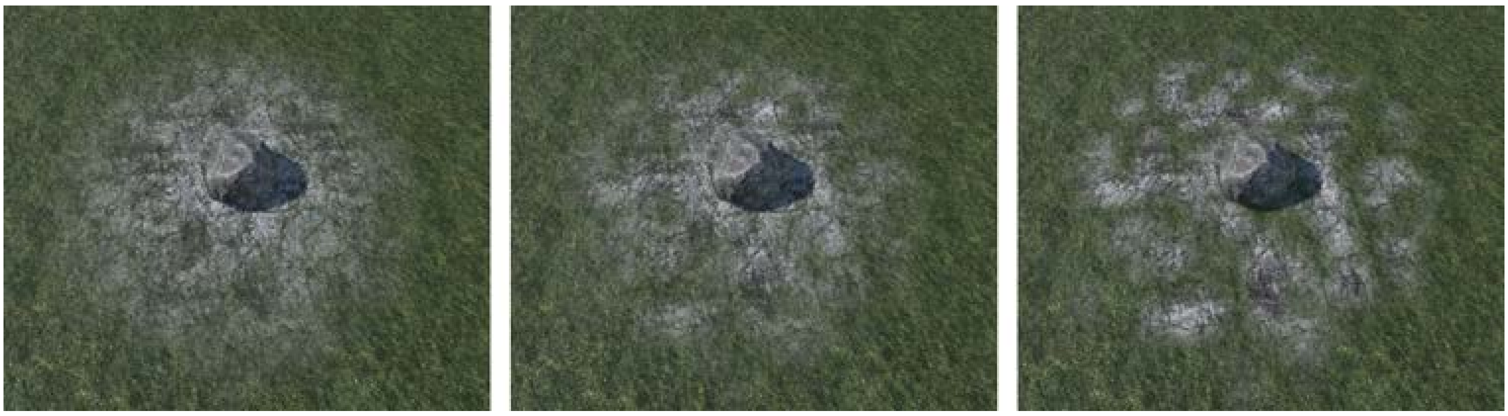

*Radius: 5, 10, 20*

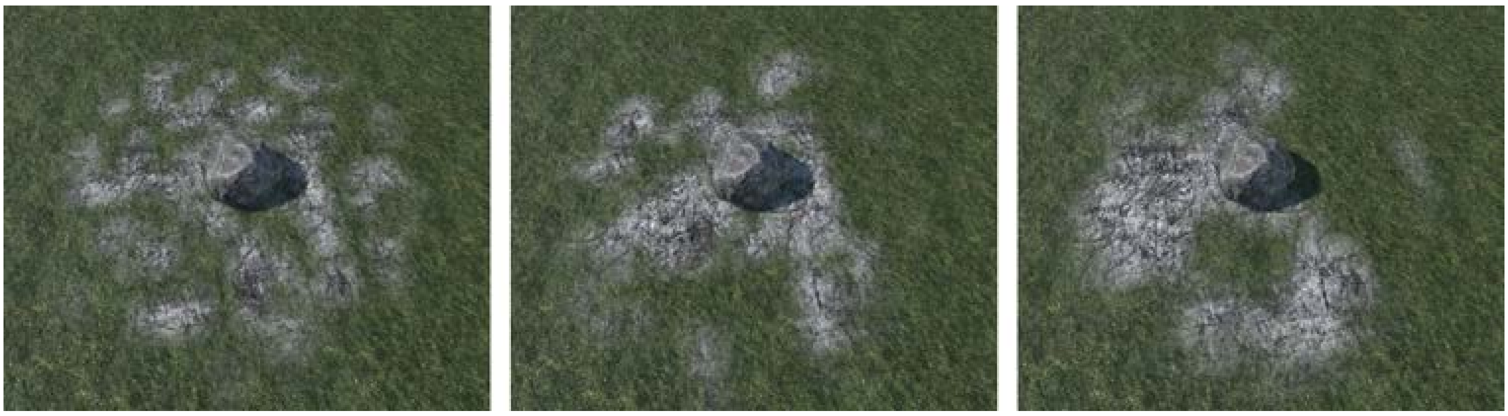
- Size Factor: multiplies the changes applied by this generator depending on the initial object size. For example, if the object's size is 2 then all of the adjustment values will be doubled if the factor is equal to 1. This is useful for proportional object changes: if the object should be lowered to half of it's height, for instance. When the value is 0 all of the objects are adjusted the same regardless of their size.

- Safe Borders: if not equal to zero generator effect will be fading down near the chunk border to prevent seams between chunks. Property value determines the fade length in world units.
- Fallof: flatten intensity depending on the distance from the object center. Left of the curve is periphery, right is the object center. Think of it like a flat area profile.

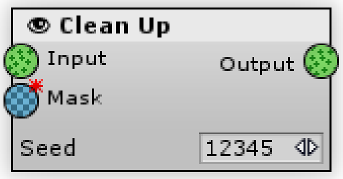


- Noise: if not zero some noise amount is added to the fallof to prevent unnatural clear slopes look. "A" property sets the amount of the noise, and "S" - the noise size.

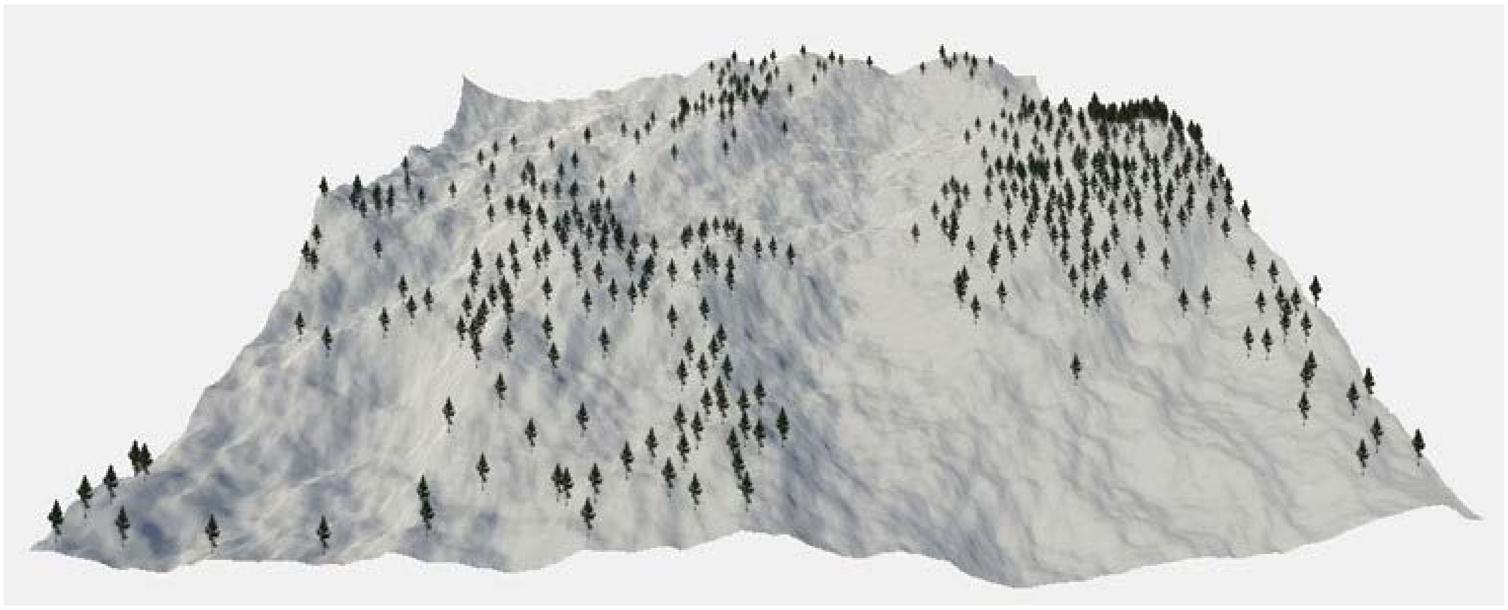

*Noise Amount (A): 0, 2, 10*

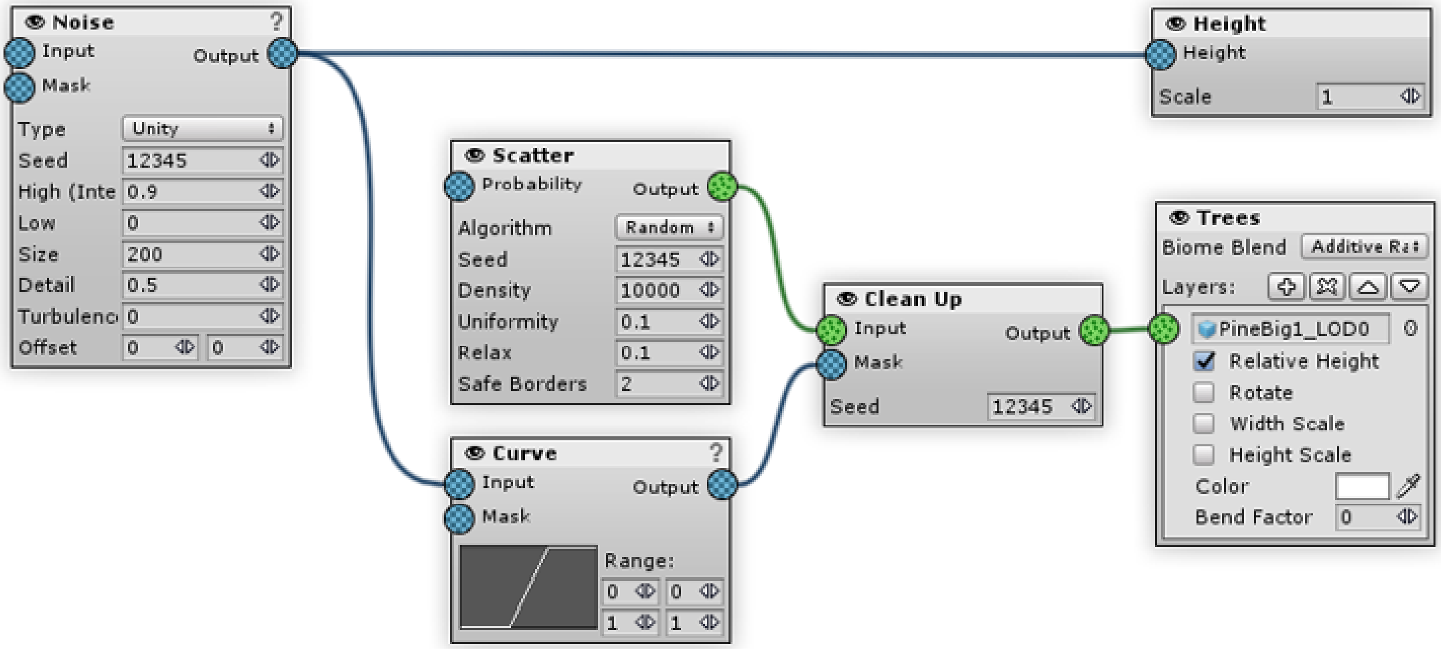

*Noise Size (S): 1, 2, 5*

# Clean_up



A generator to remove objects using objects intensity mask. It will leave the object if the mask value is 1, remove it if the mask value is 0. If the mask value is something in-between, it will remove the object at random using Seed property, the less the value the more the chance to remove the object.





*Leaving only the trees on tops using the heightmap as a mask*

## Inputs:

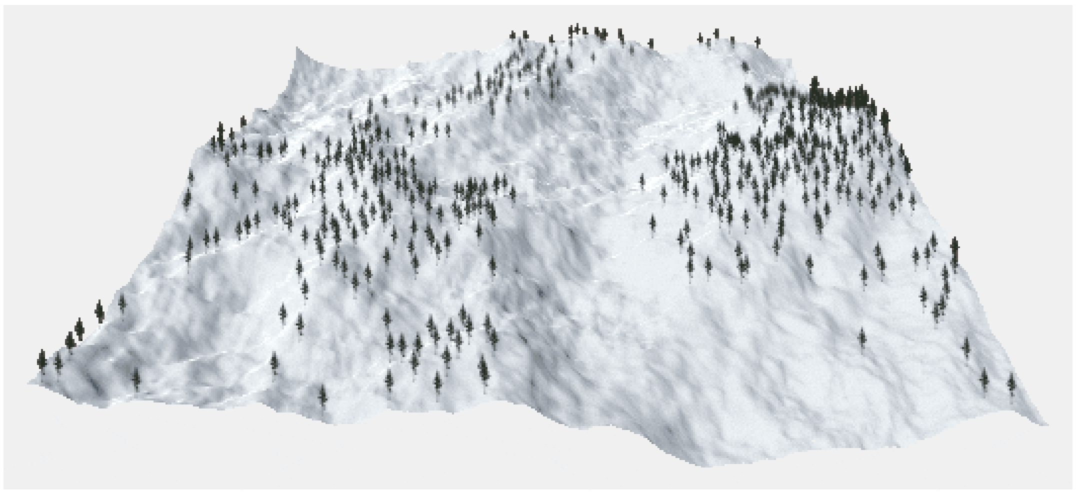
- ● Input - the default object list (hash) to be processed by the generator.
- ● Mask - desired objects intensity mask. The higher the values, the more objects will remain.

## Outputs:

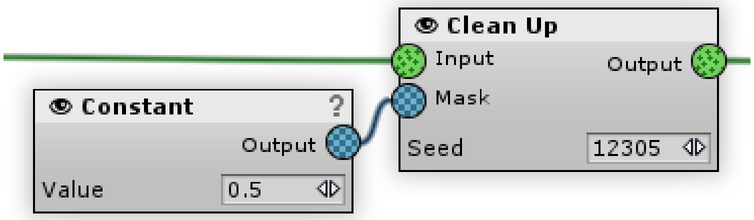
- ● Output - the list (hash) with the objects being removed.

## Properties:

- Seed - a seed value for the random generator that decides if objects should be removed.

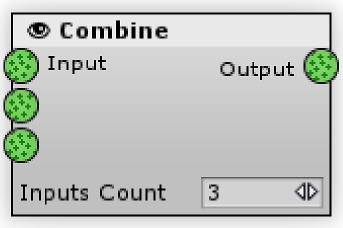

*Different seed values*

## Other Usage Examples:



*Removing every 2nd object*

## Combine



Gathers objects from multiple lists (hashes) in one.

### Inputs:

- ● Multiple Inputs - the lists (hashes) with the objects that will be combined in output.

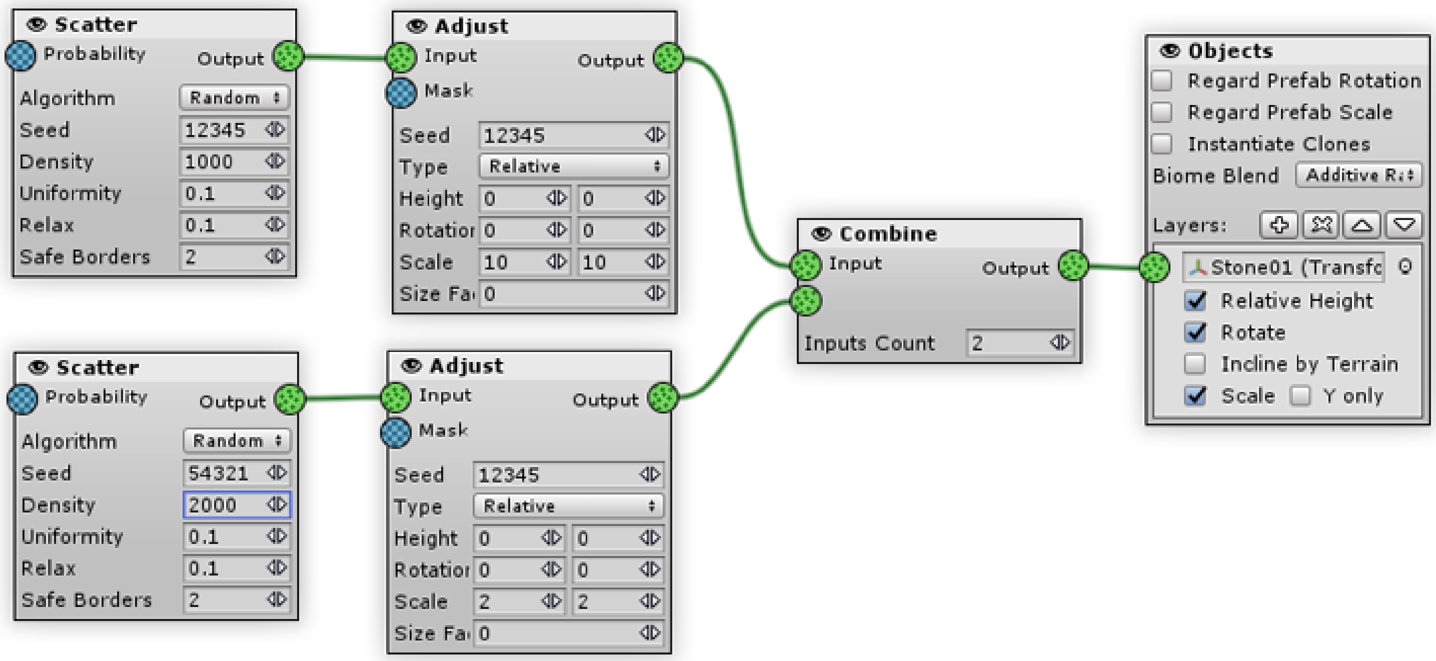
### Outputs:

- ● Output - the list (hash) that has all the objects from all the input lists (hashes) combined.

### Properties:

- The number of inputs can be set with the Inputs Count parameter.

### Usage Example:



Scatter
Probability Output
Algorithm Random
Seed 12345
Density 1000
Uniformity 0.1
Relax 0.1
Safe Borders 2

Adjust
Input Output
Mask
Seed 12345
Type Relative
Height 0 0
Rotatior 0 0
Scale 10 10
Size Fa 0

Scatter
Probability Output
Algorithm Random
Seed 54321
Density 2000
Uniformity 0.1
Relax 0.1
Safe Borders 2

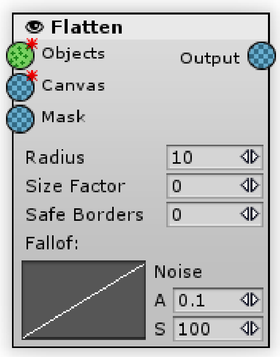
Adjust
Input Output
Mask
Seed 12345
Type Relative
Height 0 0
Rotatior 0 0
Scale 2 2
Size Fa 0

Combine
Input Output
Inputs Count 2

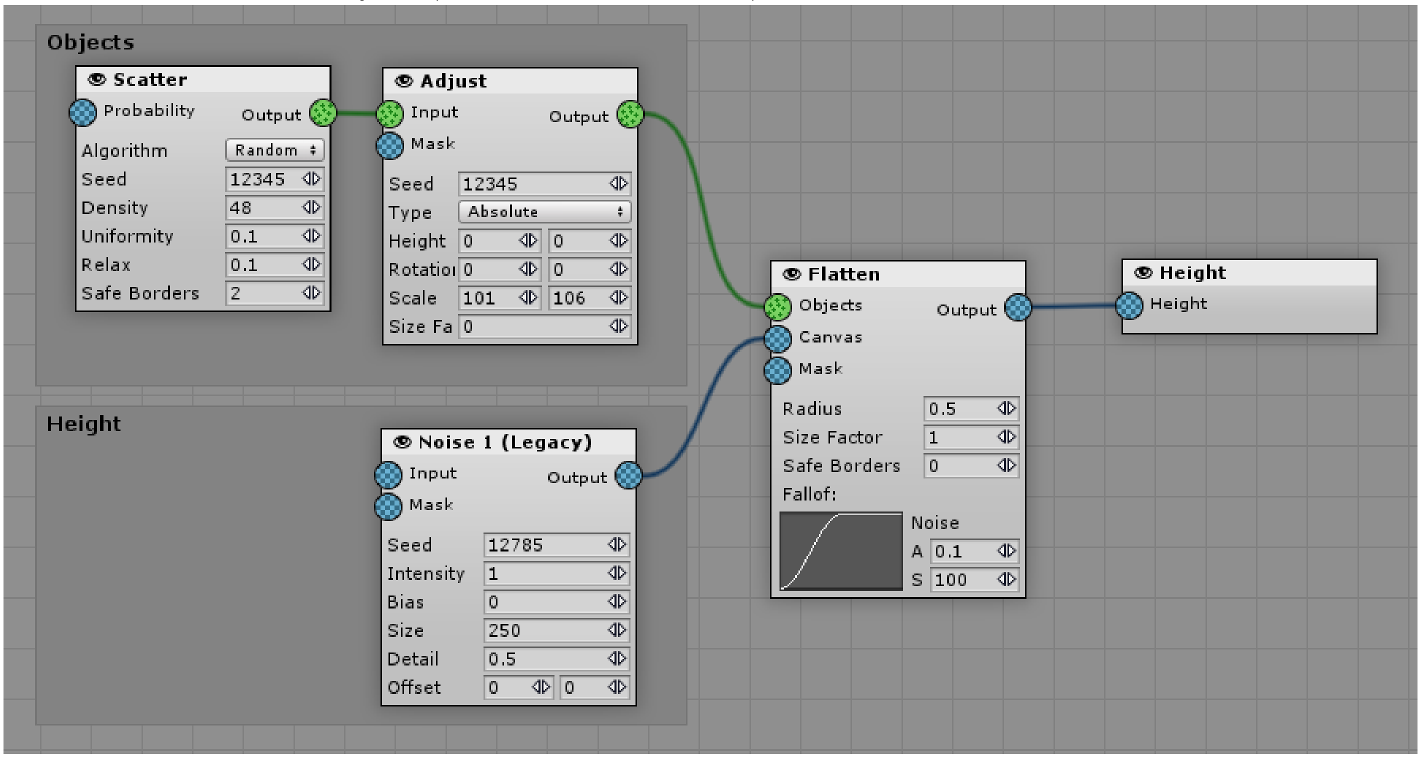
Objects
☐ Regard Prefab Rotation
☐ Regard Prefab Scale
☐ Instantiate Clones
Biome Blend Additive R
Layers:
Stone01 (Transfo
☑ Relative Height
☑ Rotate
☐ Incline by Terrain
☑ Scale ☐ Y only

*Combining big and small stones in one output*

## Flatten


Flatten
Objects
Output
Canvas
Mask
Radius 10
Size Factor 0
Safe Borders 0
Fallof:
Noise
A 0.1
S 100

Used to create a flat land under the objects to place houses or structures that require a flat land.


Objects

Scatter
Probability
Output
Algorithm Random
Seed 12345
Density 48
Uniformity 0.1
Relax 0.1
Safe Borders 2

Adjust
Input
Output
Mask
Seed 12345
Type Absolute
Height 0 0
Rotatioı 0 0
Scale 101 106
Size Fa 0

Height

Noise 1 (Legacy)
Input
Output
Mask
Seed 12785
Intensity 1
Bias 0
Size 250
Detail 0.5
Offset 0 0

Flatten
Objects
Output
Canvas
Mask
Radius 0.5
Size Factor 1
Safe Borders 0
Fallof:
Noise
A 0.1
S 100

Height
Height

## Inputs:

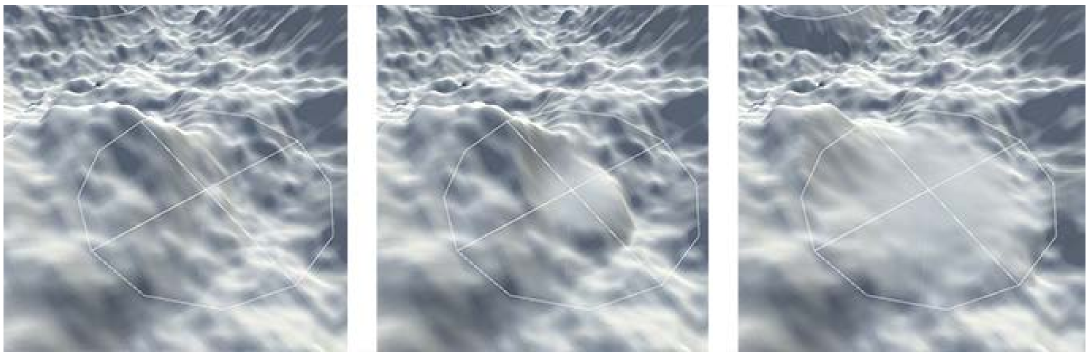
- 🟢 Objects (mandatory): objects that would be the centers of the flat areas.
- 🔵 Canvas (mandatory): current graph heightmap that will be adjusted with this generator.
- 🔵 Mask: the map that controls generator's intensity.

## Outputs:

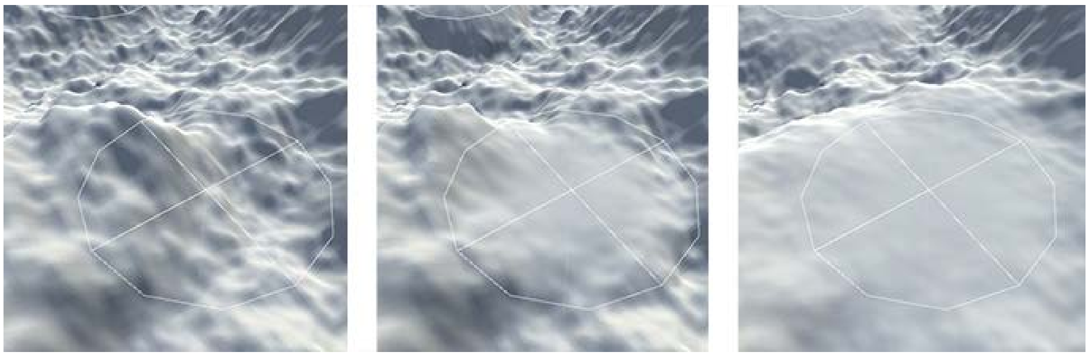
- 🔵 Output: height, input canvas with flattened object stamps.

## Properties:

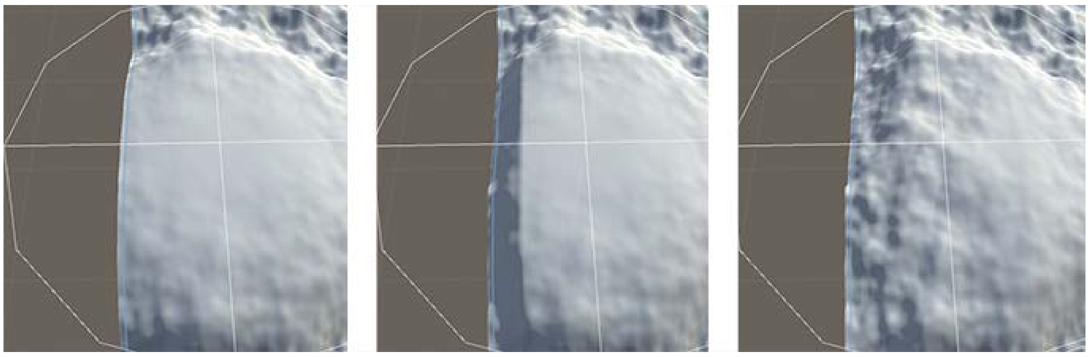
- Radius: the radius of the adjusted areas (not the flat land, but the whole Fallof area).


*Radius: 0, 25, 50*

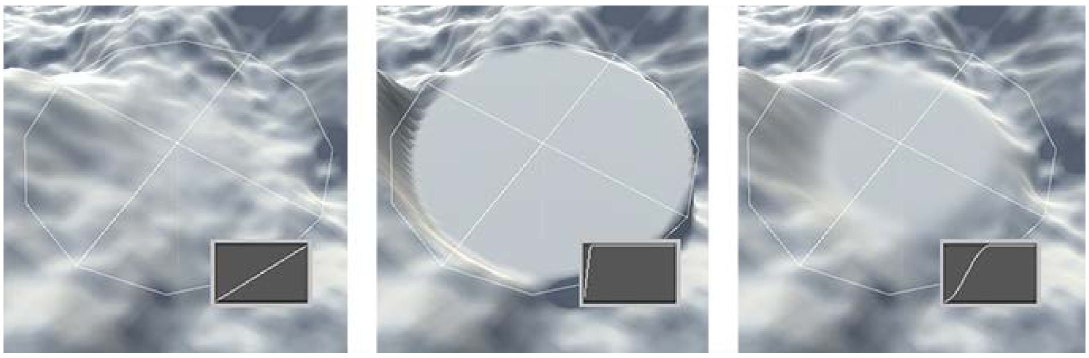
- Size Factor: multiplies the changes applied by this generator depending on the initial object size. For example, if the object's size is 2 then all of the adjustment values will be doubled if the factor is equal to 1. This is useful for proportional object changes: if the object should be lowered to half of it's height, for instance. When the value is 0 all of the objects are adjusted the same regardless of their size.


*Size Factor: 0, 1, 2*

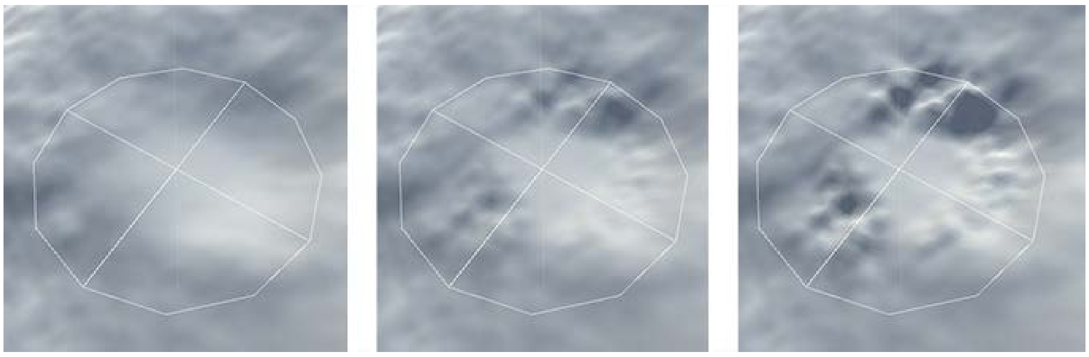
- Safe Borders: if not equal to zero generator effect will be fading down near the chunk border to prevent seams between chunks. Property value determines the fade length in world units.


*Safe Borders: 0, 10, 50*

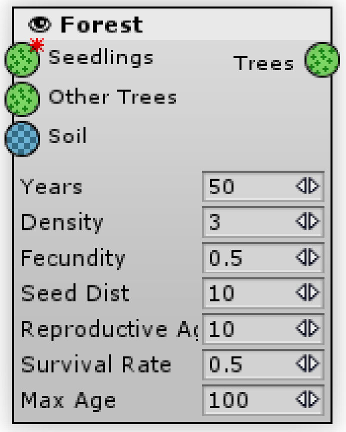
- Fallof: flatten intensity depending on the distance from the object center. Left of the curve is periphery, right is the object center. Think of it like a flat area profile.


*Fallof*

** Noise: if not zero some noise amount is added to the fallof to prevent unnatural clear slopes look. "A" property sets the amount of the noise, and "S" - the noise size.

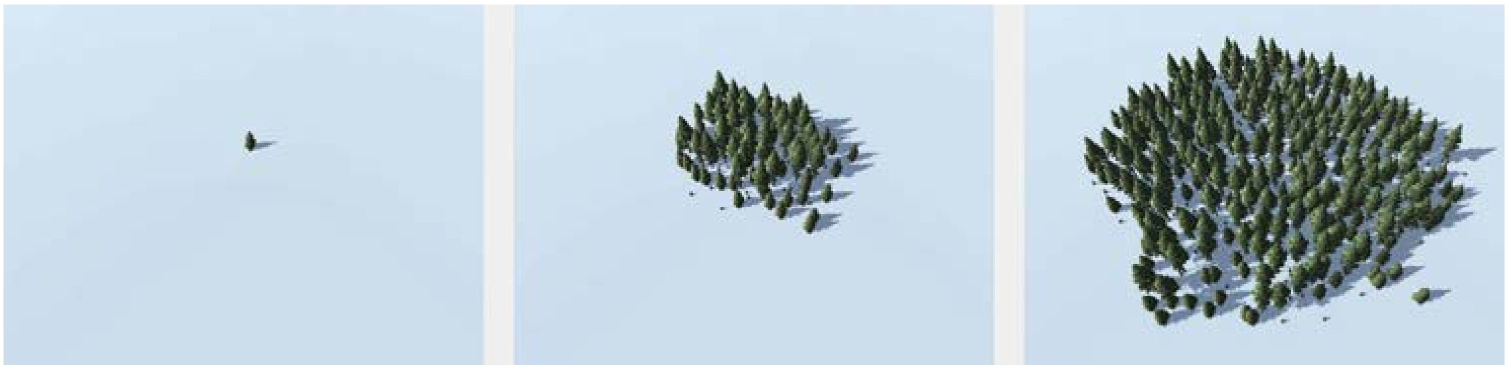

*Noise: 0, 1, 2*

# Forest


Forest
Seedlings
Trees
Other Trees
Soil

| | |
|---|---|
| Years | 50 |
| Density | 3 |
| Fecundity | 0.5 |
| Seed Dist | 10 |
| Reproductive Ag | 10 |
| Survival Rate | 0.5 |
| Max Age | 100 |

Emulates a natural forest growth: seed dispersal, the growth of trees, the adequacy of lighting, and soil quality. Forest Generator can generate several forests of one tree type. As forests grow and expand they will be joint together into a single whole. Each forest starts with a single tree - a seedling.



Along with Erosion Generator this Generator takes some time to compute. Keep in mind that a forest is a system with negative feedback. Every little change in the beginning causes an absolutely different result in the end. So generating a forest with slightly different inputs can cause a very different final picture. For example, changing the seedling's position can cause the entire forest to die, while other forest arrays would appear in different places.

## Inputs:

- ● Seedlings - the initial trees. Each of the seedlings starts a new forest.
- ● Extra Shade - additional tree objects that shade the trees. For example, when planting a birch forest it could be more long-lived trees like oaks. This object hash prevents the forest from growing in the other forest's area.
- ● Soil - a map that controls the chance of the tree to live and to produce seeds. Poor quality (low map values) raises the tree's chances of dying. Think of it like soil quality but this also could be a height or slope factor or their multiplication. This parameter can control not only the soil quality, but other aspects like height or slope factor.

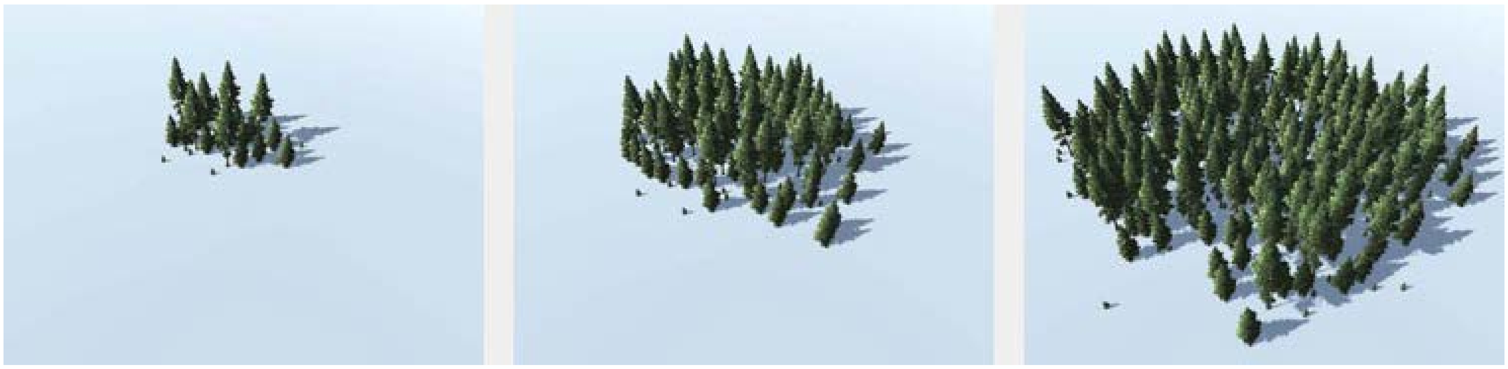
## Outputs:

- ● Trees - living trees object hash. The age of each tree in a Tree's output is recorded as an object size. One year corresponds to 1 unit in size: the first year's sapling's size would be 1, while a hundred year old tree's size would be 100.

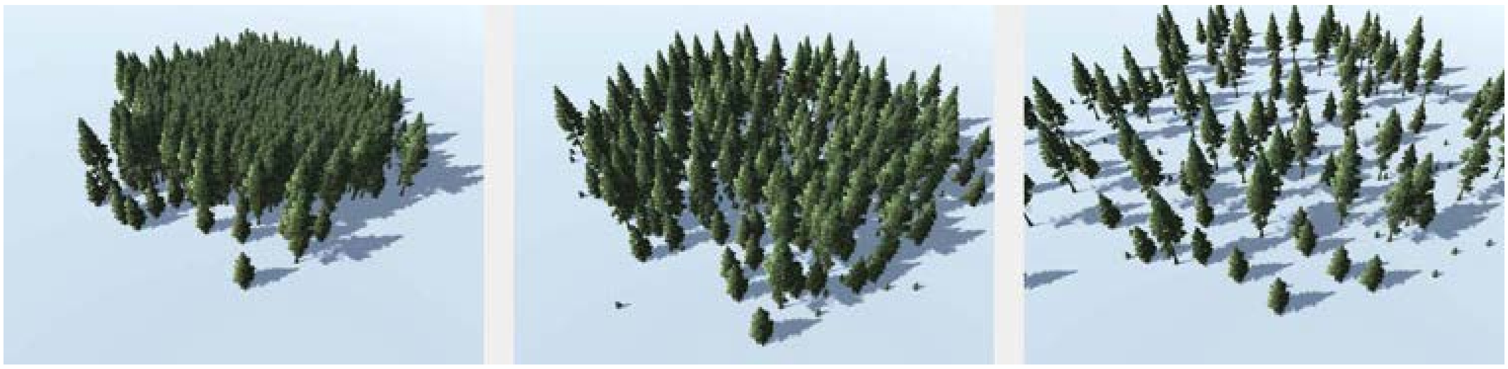
Use Split Generator to sort trees, with the conditions properly set: for example for young trees the Size Condition should be 0-10, for medium trees 10-30, for big ones 30-200.

## Properties:

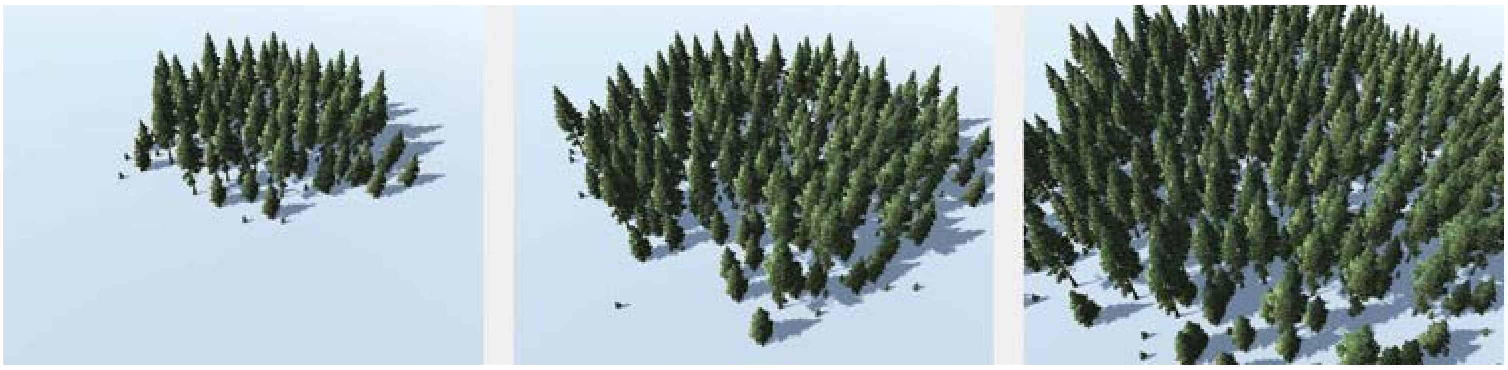
- Years: number of years passed since the first seedling started to grow. In most cases this parameter determines the size of the forest, but in some cases the forest can shrink after it matures and even die - this often happens for tall trees (i.e. high Shade Dist value) and poor soil.


*Years: 50, 100, 150*

- Density: the maximum number of trees in an area of 10*10 units.


*Density: 10, 3, 1*

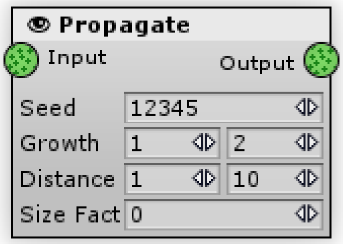
- Fecundity: how many seedlings the tree produces per year. The more the value the faster the forest spreads.
- Seed Dist: how far a tree can throw a seed. Increasing this parameter can make a forest spread fast, but for more realistic results it is recommended to set this parameter a little higher than Shade Dist.


*Seed Dist: 5, 10, 15*

- Reproductive Age: the age at which the tree starts to produce seedlings around it.
- Survival Rate: a chance for the tree to survive each year. This value is multiplied with a Soil Mask.
- Max Age: the maximum tree life time. The tree will die when it reaches this age, but it can die earlier because of bad

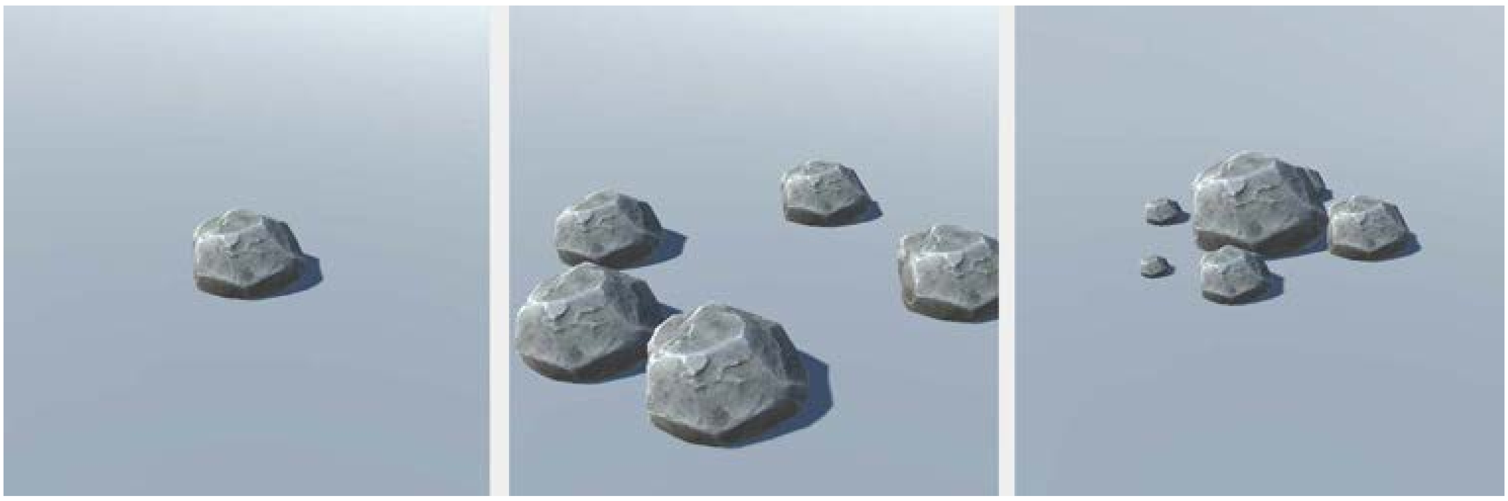
conditions (i.e. Survival Rate).

## Propagate



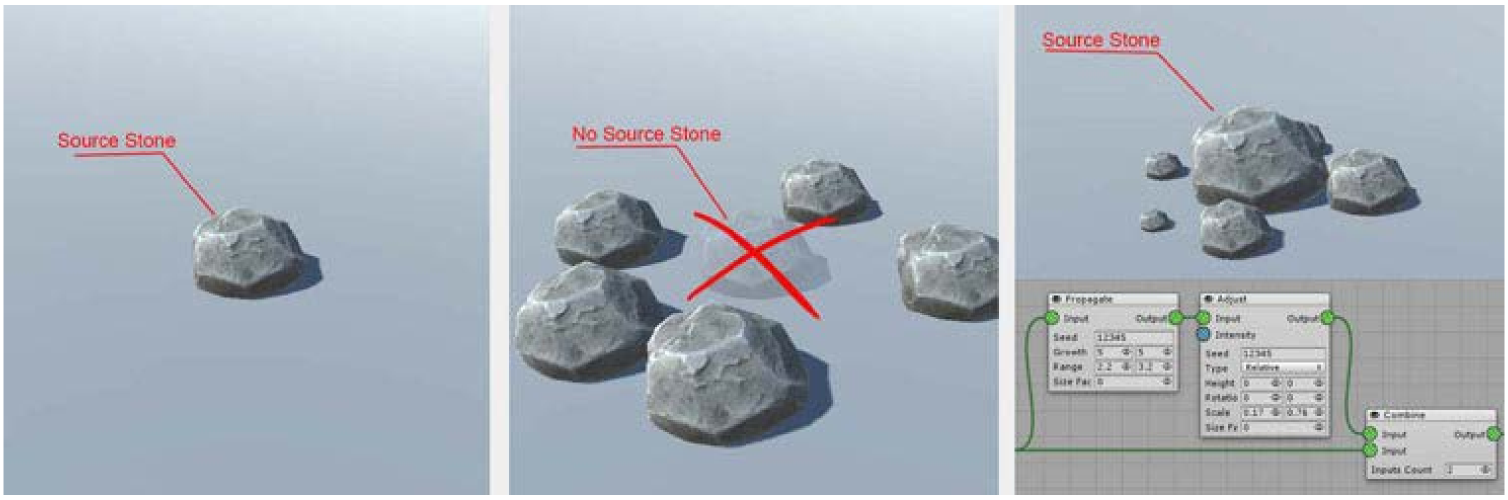
Propagate
Input Output
Seed 12345
Growth 1 2
Distance 1 10
Size Fact 0

For each of the input objects Propagate Generator will create a certain number of clones and will offset the created clones from the object's position by a certain distance in a random direction on a plane.



| *Original object* | *Propagate with 5 clones* | *Propagate + Size Adjust + Combine with original* |

Note that this generator will not return the input objects themselves, it will just output their offset clones. Use Combine Generator to combine source objects with propagated ones.



Propagate
Input Output
Seed 12345
Growth 5 5
Range 2.2 3.2
Size Fac 0

Adjust
Input Output
Intensity
Seed 12345
Type Relative
Height 0 0
Rotatio 0 0
Scale 0.17 0.76
Size Fc 0

Combine
Input Output
Input
Inputs Count 2

| *Original object* | *Propagate with 5 clones* | *Propagate + Size Adjust + Combine with original* |

### Inputs:

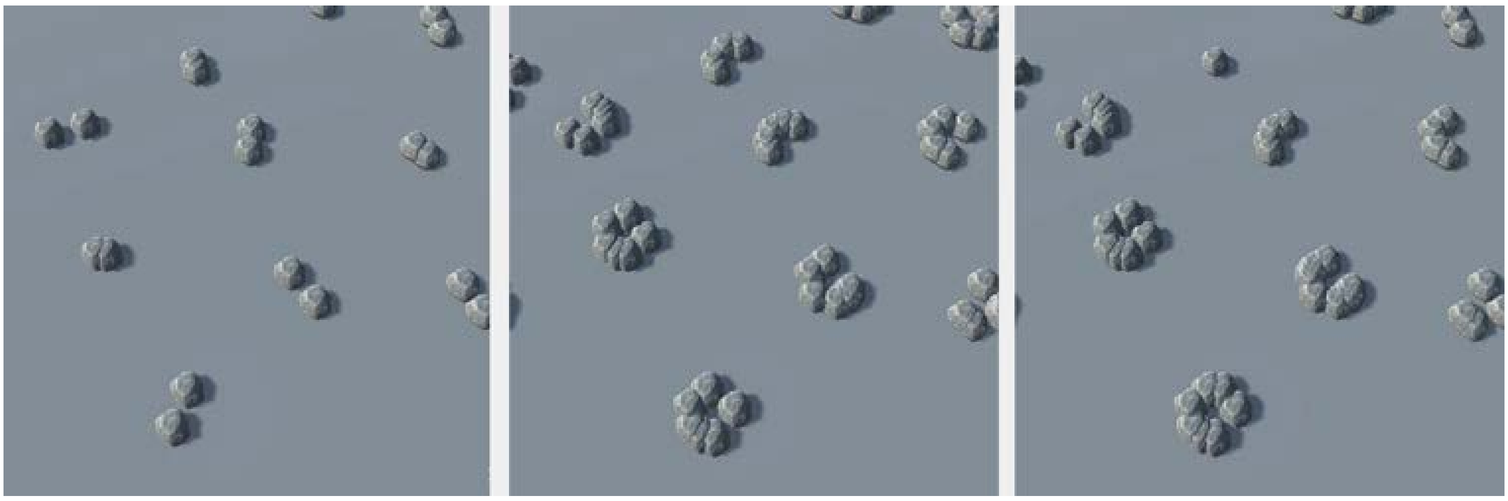
- 🟢 Input - the hashes with the objects that will be combined in output

## Outputs:

- ● Output - a hash that has all the objects from all the input hashes combined.

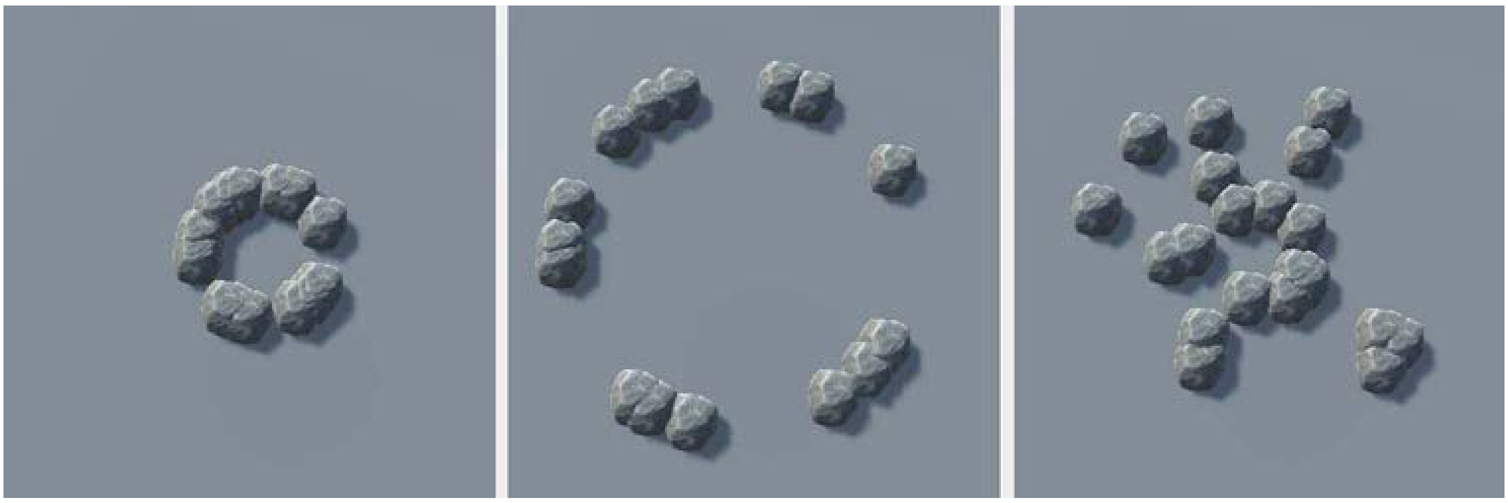
## Properties:

- Growth: minimum and maximum number of clones created for each object by the Generator. This value is a float. For example, when the growth range is set to 5 - 5.5 the Generator will create 5 clones for 75% 1of the objects and 6 clones for 25%, so the average number of clones per object will be 5.25.



*Growth: 2-2, 6-6, 1-8*

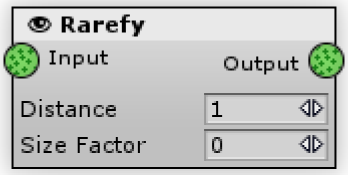
- Distance: minimum and maximum distance for clone's offset from the object's original position.



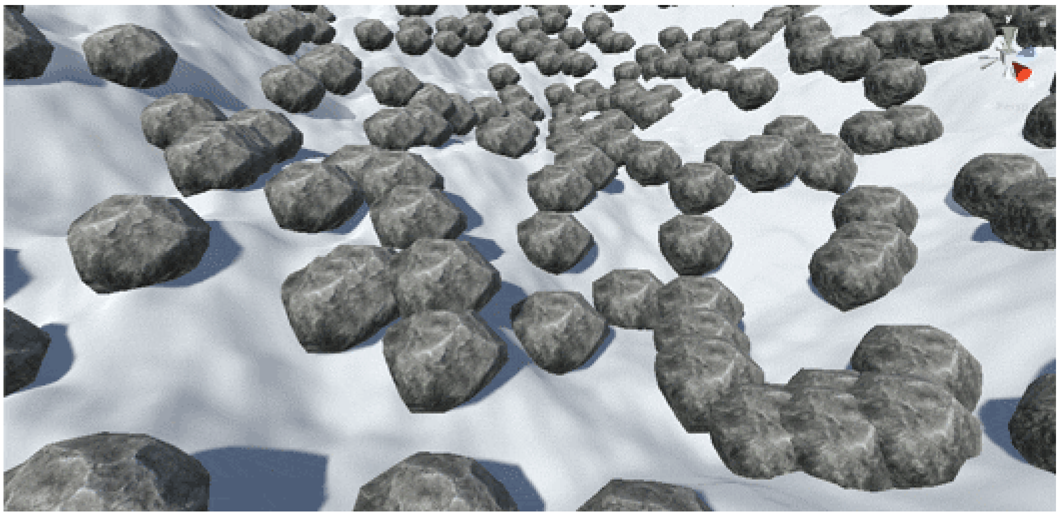
*Distance: 1-1, 4-4, 1-4*

- Size Factor: impact factor of the original object's size on the Growth and Range parameters. When set to 0 the object size is not taken into account, when set to 1 Growth and Range parameters are multiplied by object size. Keep in mind that big objects will generate many more clones that can end up overflowing the terrain with cloned objects.

# Rarefy


Rarefy
Input
Output
Distance 1
Size Factor 0

Removes objects that are located close to each other withing a single list (hash).



Note that it works within a single objects input. In order to remove closely located objects in two or more inputs you've got to use Subract Generator.

## Inputs:

- 🟢 Input - the default object list (hash) to be processed by the generator.

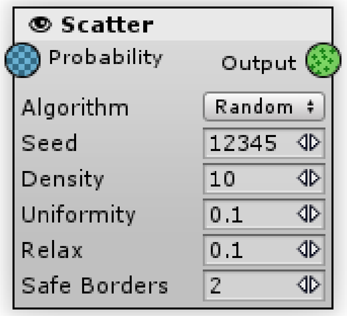
## Outputs:

- 🟢 Output - stores the generator's processing result.

## Properties:

- Distance: the minimum distance that is allowed between the objects. If objects are located closed then one of them is removed.
- Size Factor: multiplies the distance value depending on the initial object size. For example, if the object's size is 2 then all of the adjustment values will be doubled if the factor is equal to 1. This is useful for proportional object changes: if the object should be lowered to half of it's height, for instance. When the value is 0 all of the objects are adjusted the same regardless of their size.

# Scatter


Scatter
Probability
Output
Algorithm: Random
Seed: 12345
Density: 10
Uniformity: 0.1
Relax: 0.1
Safe Borders: 2

Scatters objects in the terrain area, randomly creates new object positions. Usually it's the first generator that starts the objects nodes chain.

## Inputs:

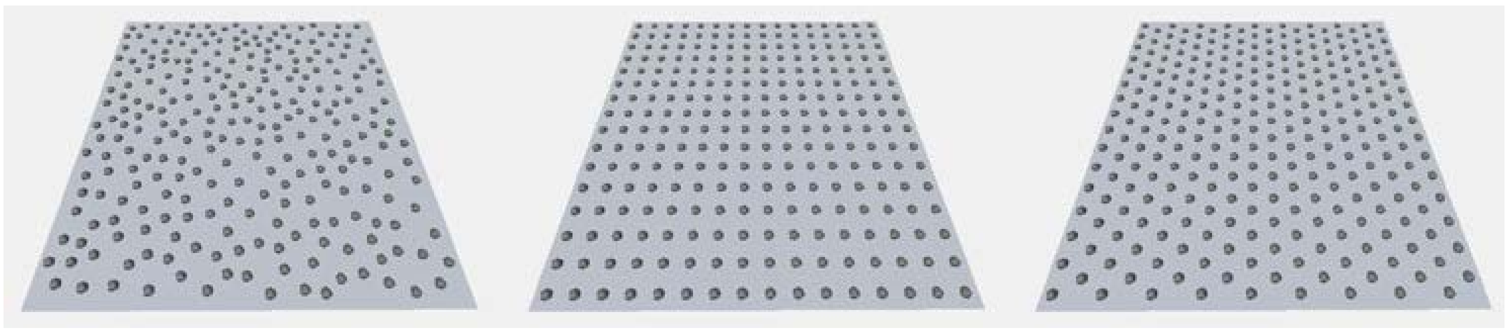
- ● Probability - the chance to spawn an object at the current map pixel. The higher the pixel values in some of the map area - the more objects will be spawned there.

## Outputs:

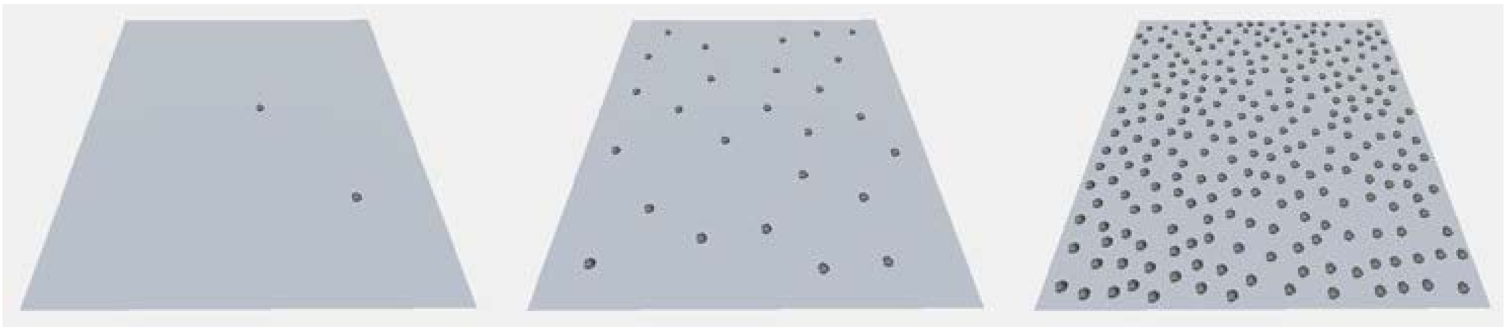
- ● Output - scattered objects

## Properties:

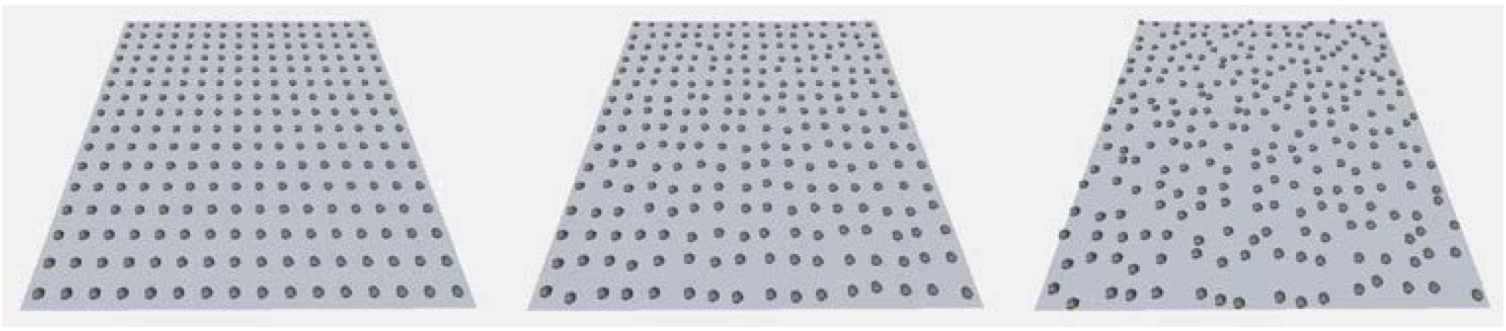
- Algorithm: determines the pattern objects are placed before randomizing:
  - Random: no pattern, pure random
  - Square Cells: objects are placed using square grid
  - Hex Cells: objects are placed using triangle (hex) grid


*Algorithm: Random, Square Cells, Hex Cells*

- Density: the quantity of scattered objects per a square kilometer (100*100 units). Note that when using a probability map the final count is less than the value because of generator's probability occlusion.

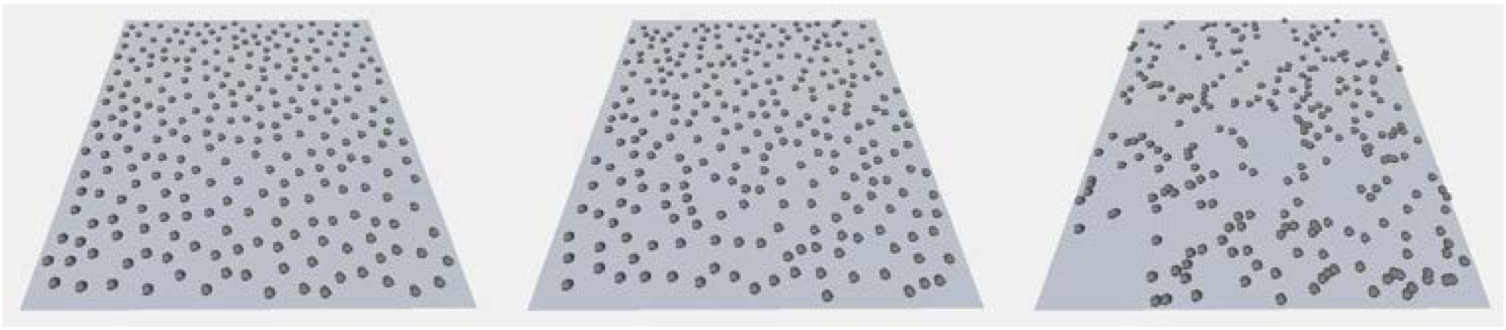

*Density (terrain size is 500*500 meters): 10, 100, 1000*

- Uniformity: determines how evenly objects are distributed along the terrain.
    - When using square or hex cells algorithm this parameter determines objects offset range after they have been placed on terrain

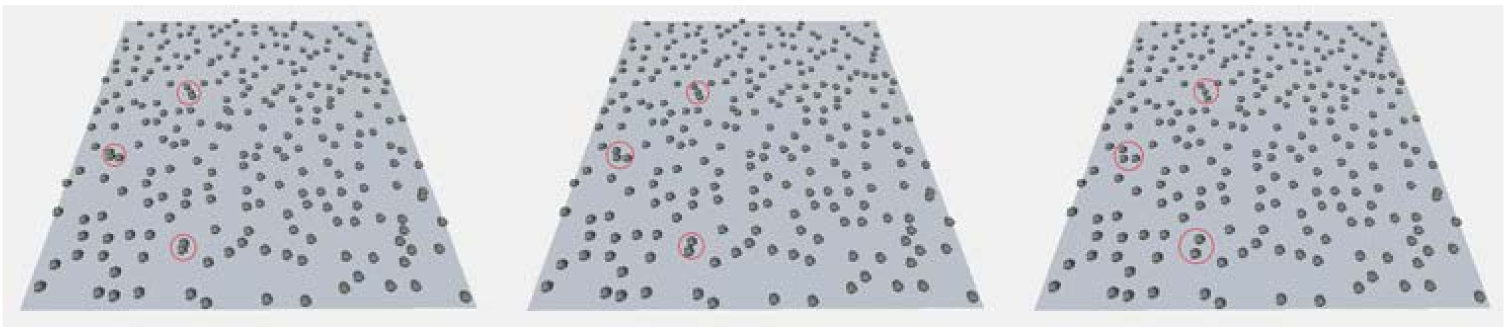

*Uniformity (square cells): 1, 0.7, 0.3*

- When using Random algorithm this parameter determines the number of iterations to find equidistant object position.

> **Note that using high Uniformity value with high objects count in Random mode can result in greatly increased generate time. Usually there's no point in using Uniformity more that 0.1 in this case. Consider switching to Square or Hex Cells mode when the objects count is <1000.**

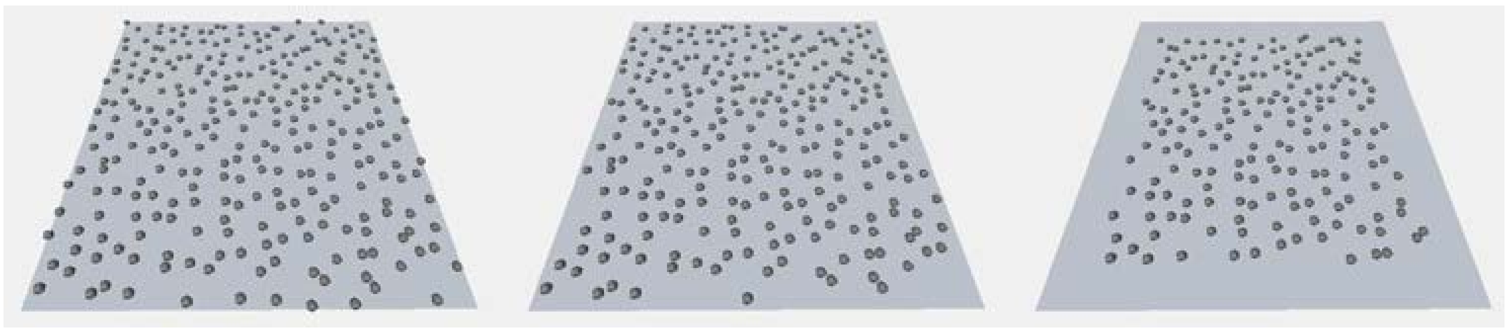

*Uniformity (random cells): 1, 0.1, 0.001*

- Relax: pulling apart closely located objects after the random has been applied. Useful at low Uniformity values.

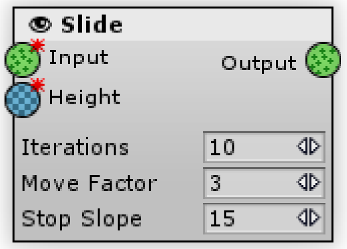


*Relax: 0, 0.25, 0.5*

- Safe Borders: an offset from the chunk borders to prevent seams (for instance, when using Blob or Flatten with the scattered objects).


*Safe Borders: 0, 10, 50*

## Slide



Iterations 10
Move Factor 3
Stop Slope 15

Pulls objects downhill. Returns objects that were moved down according to the terrain normals.

### Inputs:

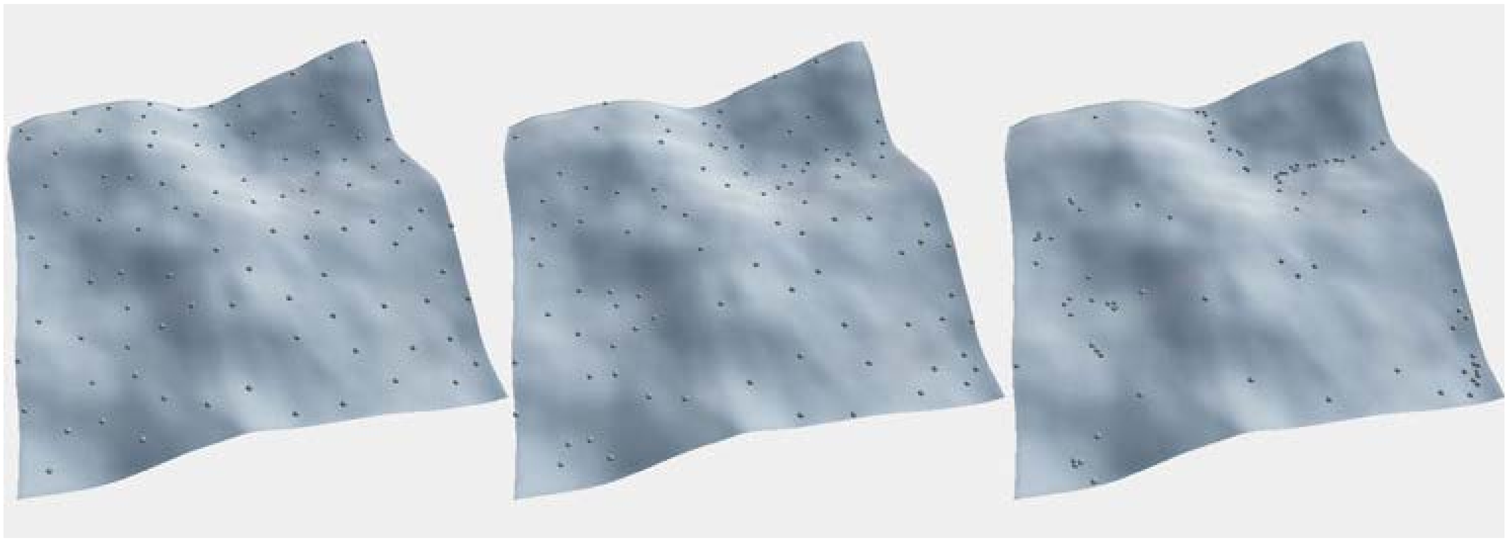
- ● Input - the object hash to be processed by the generator.
- ● Height - a terrain heightmap. The objects use it to calculate the slope direction at their positions and therefore their movement direction.

### Outputs:

- ● Output - stores the generator’s processing result.

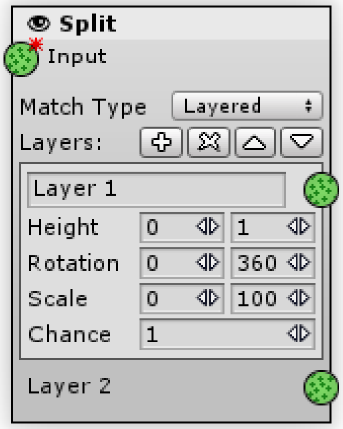
### Properties:

- Iterations: object move direction is evaluated every iteration. Increasing iteration count will result in objects traveling further, preserving a generator fidelity but increasing generation time.

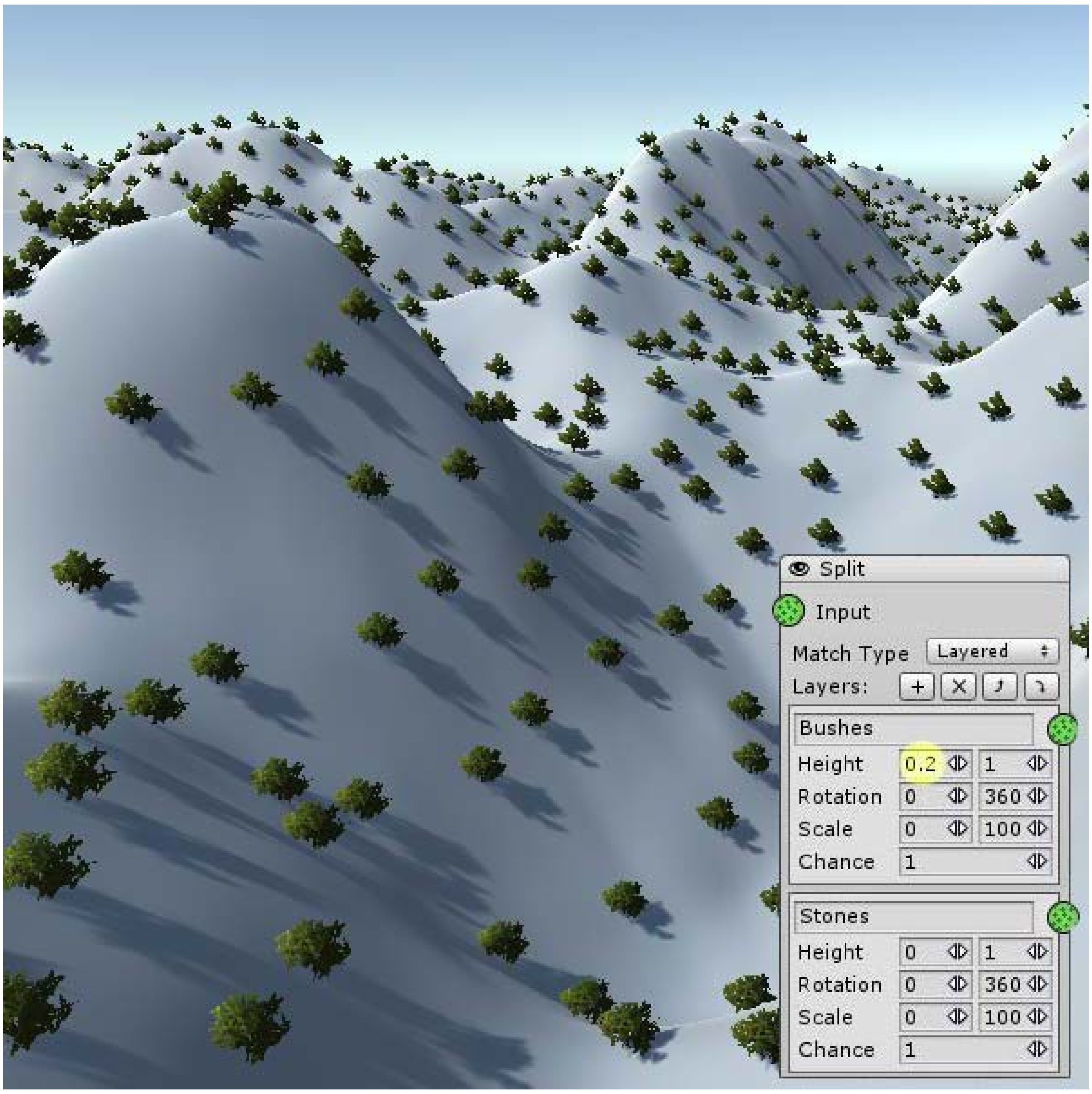

*Iterations: 0, 10, 50*

- Move Factor: The distance an object travels per iteration. Increasing this value will move objects further, but very far distances can reduce generation quality: it can make an object move upward or bounce above narrow hollows.
- Stop Slope: stops the object on horizontal surface. The object is not moved anywhere when it reached the slope inclined less than this value (in degrees, 0 is flat and 90 is vertical).

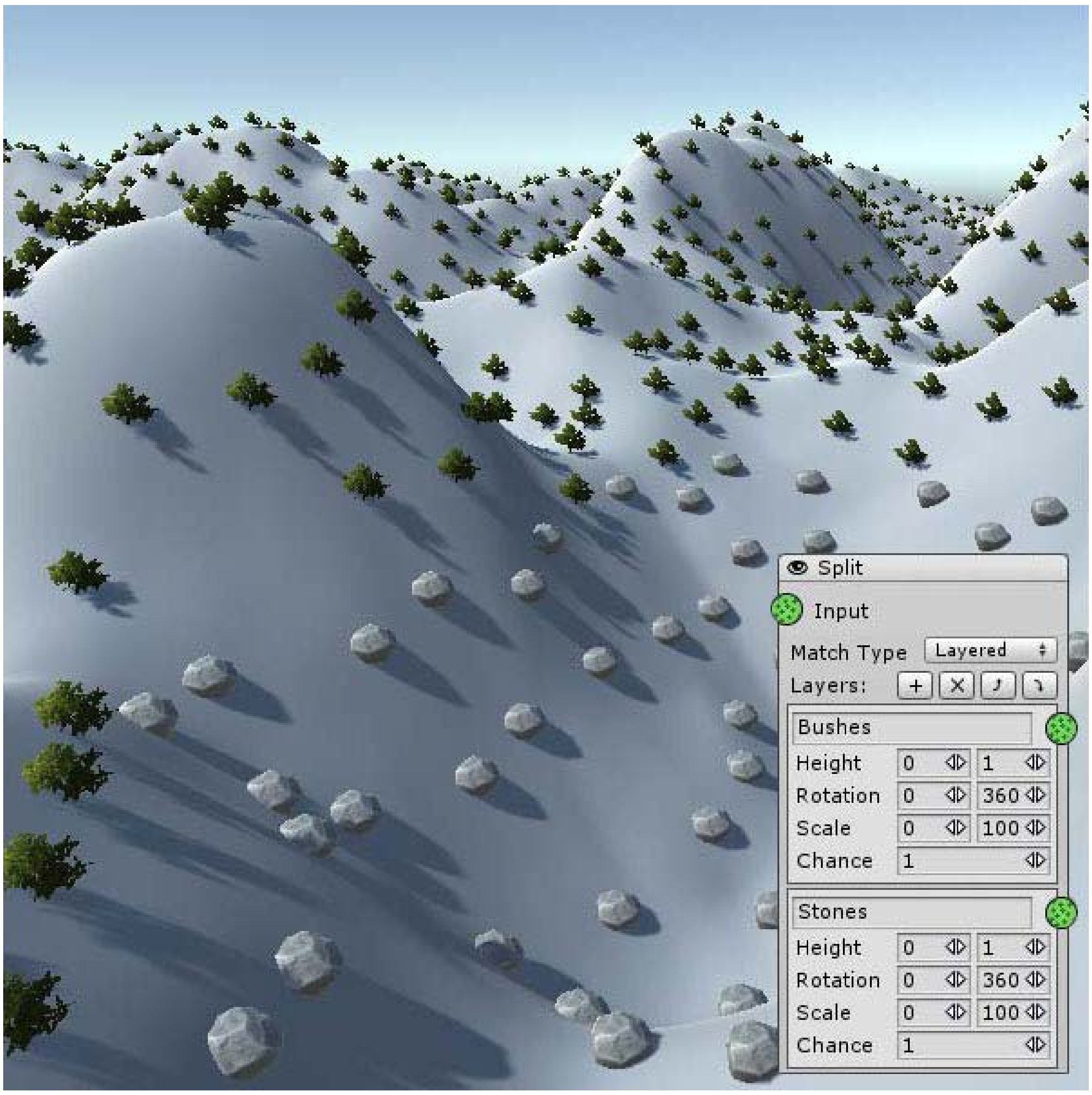
# Split


Split
Input
Match Type: Layered
Layers:
Layer 1
Height 0 1
Rotation 0 360
Scale 0 100
Chance 1
Layer 2

Creates several new outputs and fills them with input's objects using certain conditions. Split Generator has a layered structure. Each layer has its own output as well as minimum and maximum range parameters. For each of the input objects, Split Generator finds a layer that matches the object properties and writes this object to the layer output.

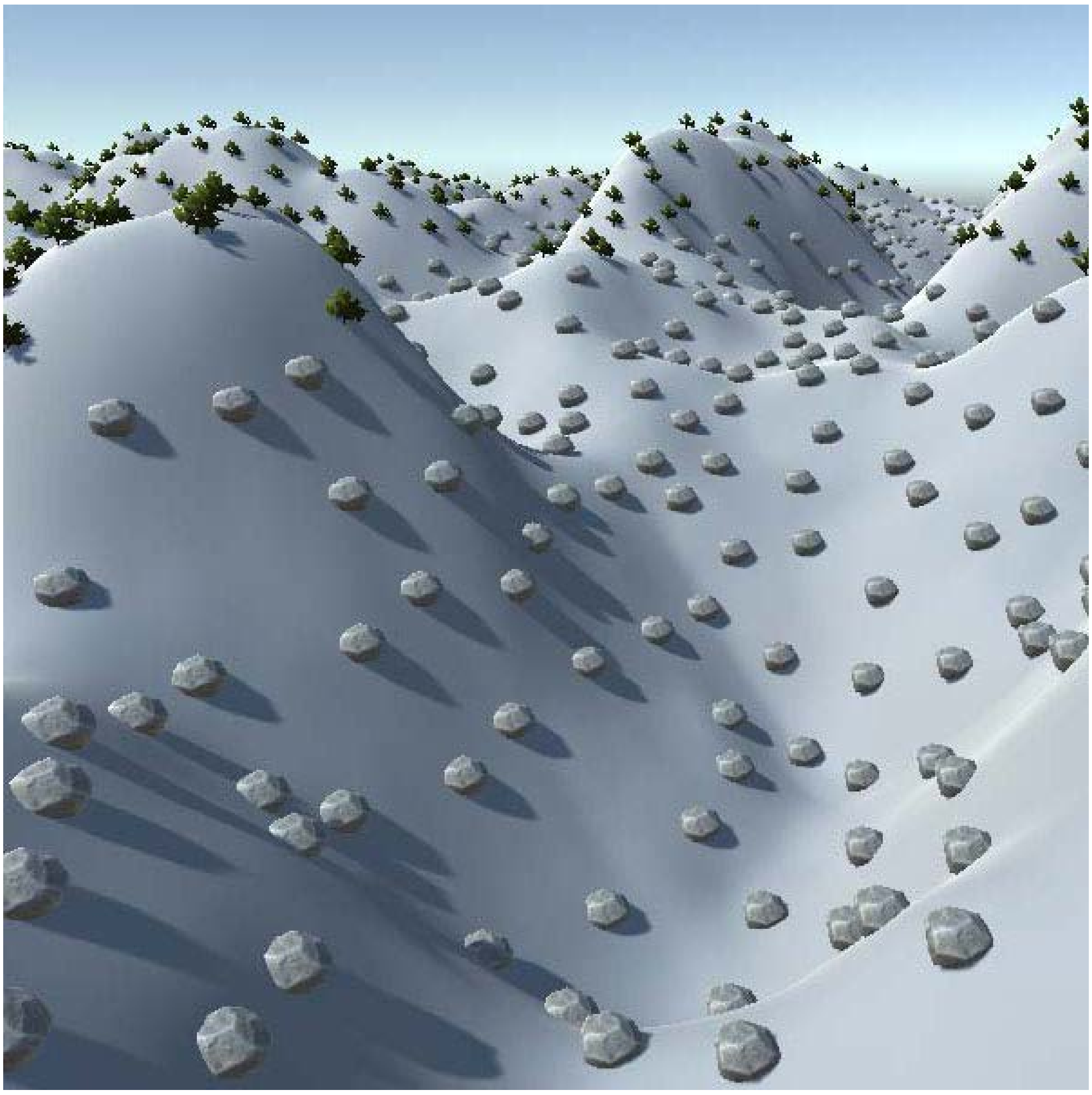

Split
Input
Match Type: Layered
Layers:
Bushes
Height 0.2 1
Rotation 0 360
Scale 0 100
Chance 1
Stones
Height 0 1
Rotation 0 360
Scale 0 100
Chance 1

*Two layers with the same conditions. _Using the top one (bushes)*


Split
Input
Match Type: Layered
Layers:

| | | |
|---|---|---|
| Bushes | | |
| Height | 0 | 1 |
| Rotation | 0 | 360 |
| Scale | 0 | 100 |
| Chance | 1 | |
| Stones | | |
| Height | 0 | 1 |
| Rotation | 0 | 360 |
| Scale | 0 | 100 |
| Chance | 1 | |

*Bushes height starts from 0.2*
*for every object below Stones layer is used*


*Bushes height starts from 0.3*

## Inputs:

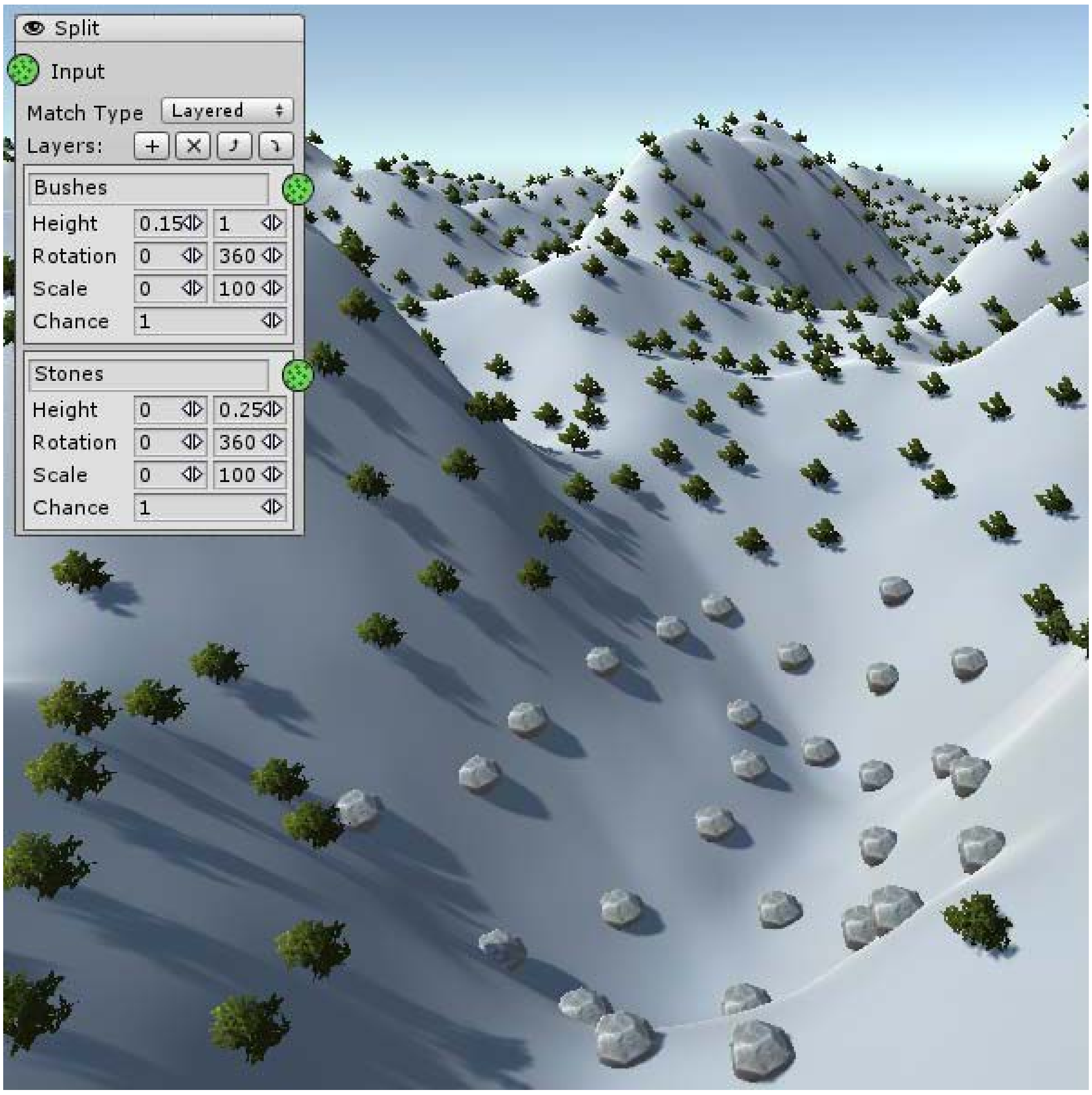
- 🟩 Input - the default object's hash to be processed by the generator.

## Outputs:

Each of the layers has an 🟩 Output, which stores only those objects from an initial hash that passed the layer conditions filter.

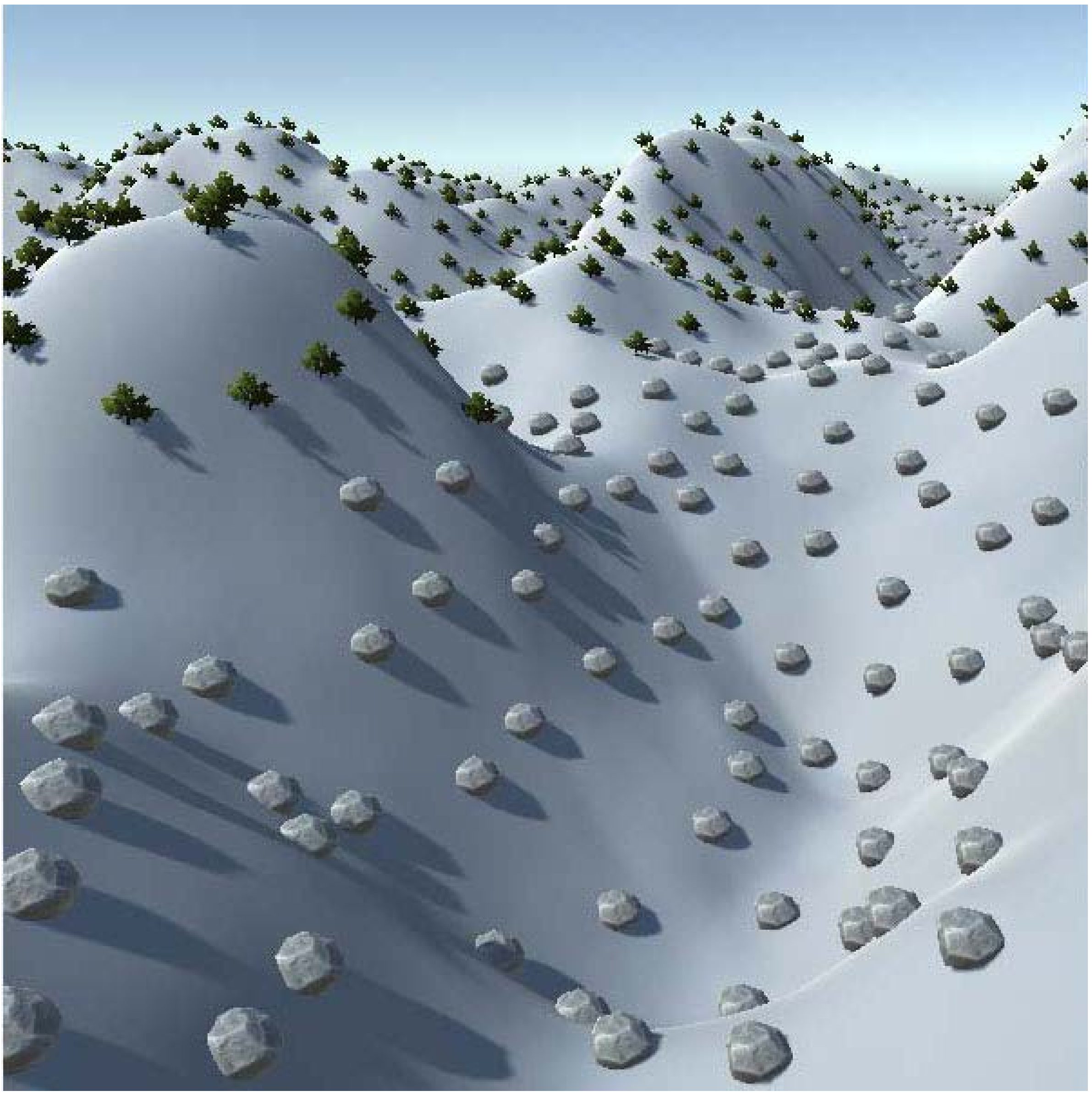
## Properties:

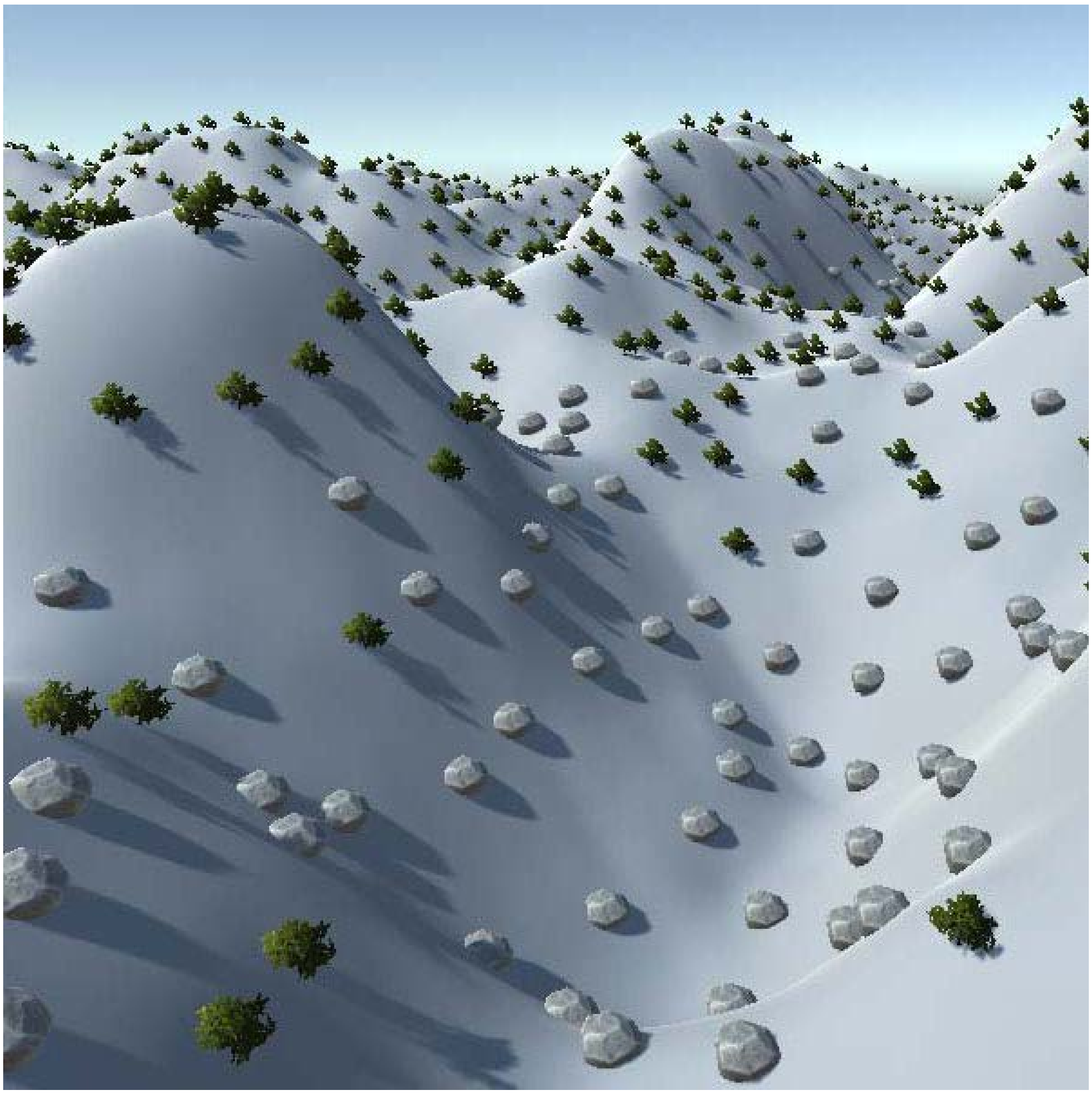
- Match Type: Note that the input object will be passed to only one layer output. If the object's properties matches several layers' conditions it will use the top layer if the "Layered" Match Type is selected or it will select a layer at random in the case of "Random" Match Type.



| | | |
|---|---|---|
| Bushes | | |
| Height | 0.15 | 1 |
| Rotation | 0 | 360 |
| Scale | 0 | 100 |
| Chance | 1 | |
| Stones | | |
| Height | 0 | 0.25 |
| Rotation | 0 | 360 |
| Scale | 0 | 100 |
| Chance | 1 | |

*Stones Height: 0 - 0.25*
*Bushes Height: 0.15 - 1*
*Bushes atop*


*Stones atop*



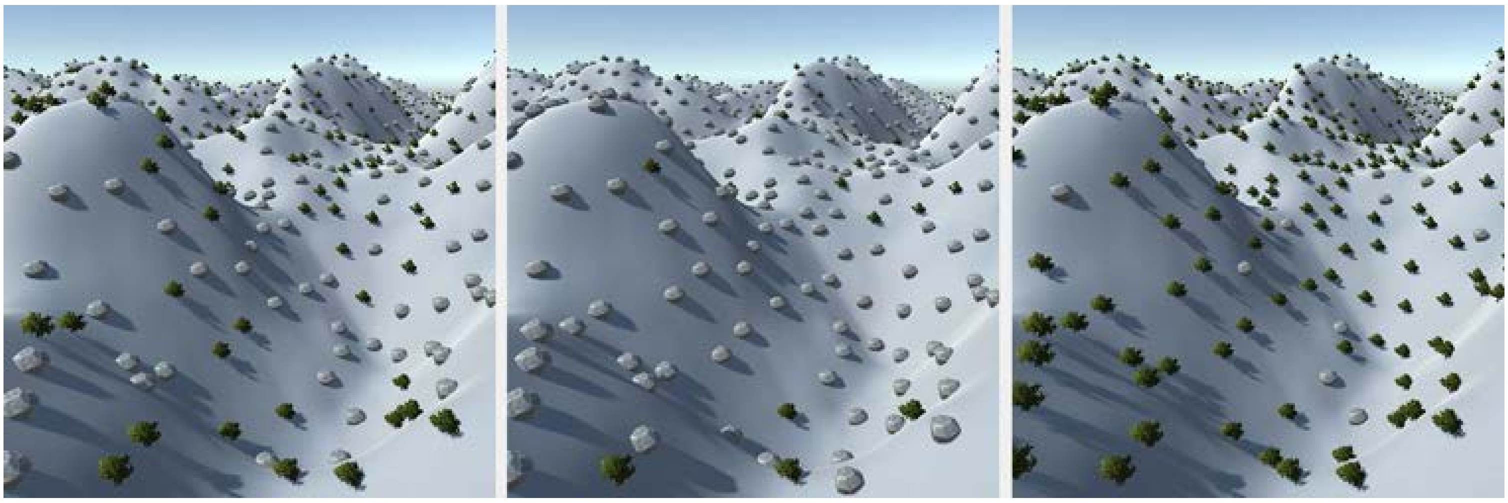
*Random match type*
*Note that in range 0.15 - 0.25 bushes and stones are mixed*

Split Generator uses the layered system, similar to the system used in Output Generators. Each layer could be selected by clicking on it. Above the layers, next to the "Layers:" label you can see layer control buttons:

- ✚ Add: will add new layer atop of the selected one.
- ✖ Remove: removes currently selected layer.
- △ Up: will change the layers order by moving the selected layer up.
- ▽ Down: will change the layers order by moving the selected layer down.

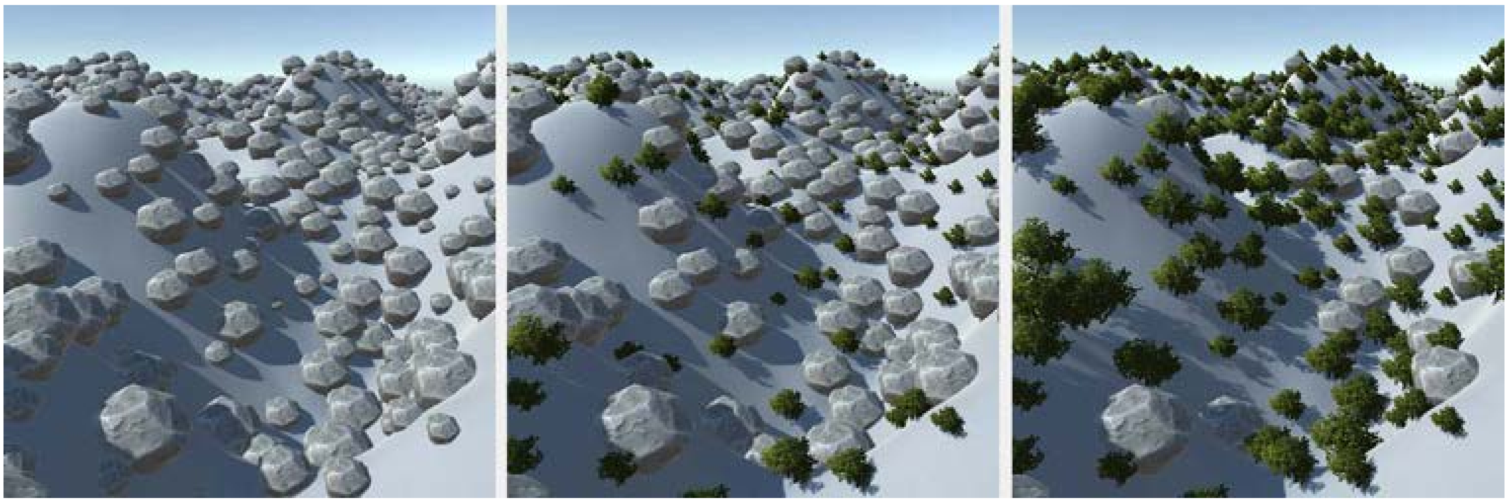
Each of the layers has the following properties:

- Height, Rotation, Size conditions: these parameters have minimum and maximum values. If the input object's parameters lay within the range of all layer conditions then this layer will be used(if not overlapped by an upper layer).



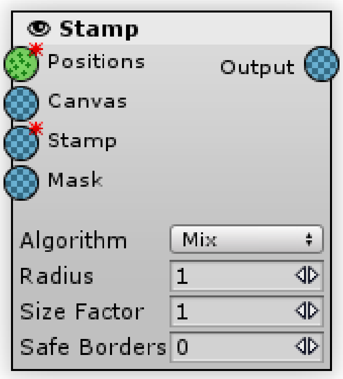
*Stones Scale Condition: 1-4, 2-4, 3-4*

- Chance: used if Match Type is set to Random. If an object matches two or more layers then it is selected at random using the Chance parameter. It determines the probability of using the layer - higher values will give more chances.



| *Bushes Chance: 1, Stones Chance: 1* | *Bushes: 0.1, Stones: 1* | *Bushes: 10, Stones: 1* |

## Stamp


Stamp

Positions

Canvas

Stamp

Mask

Output

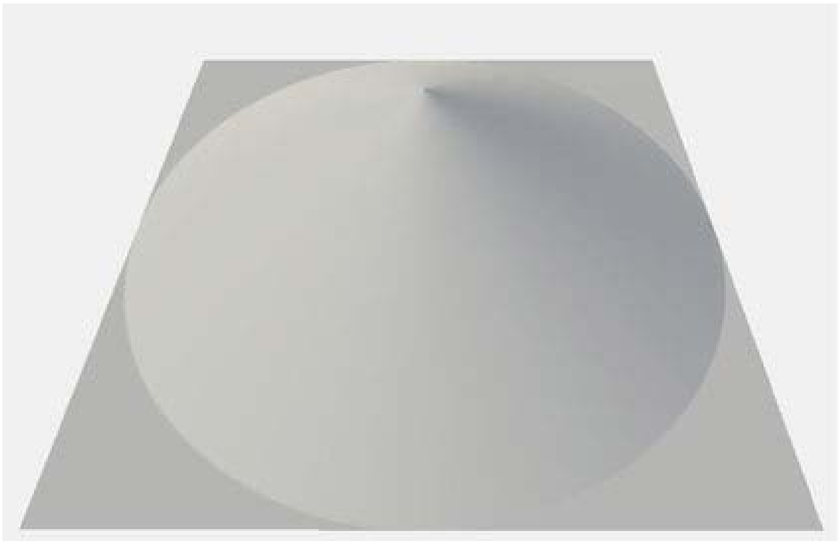
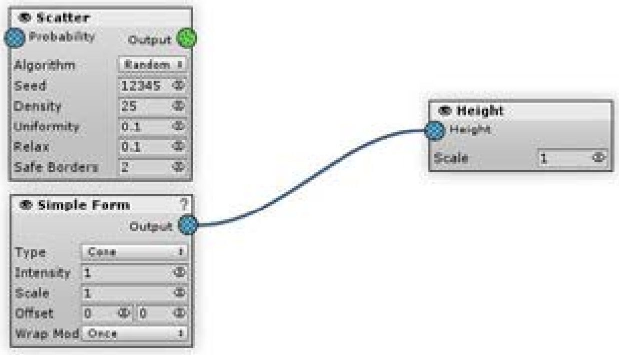
Algorithm Mix

Radius 1

Size Factor 1

Safe Borders 0

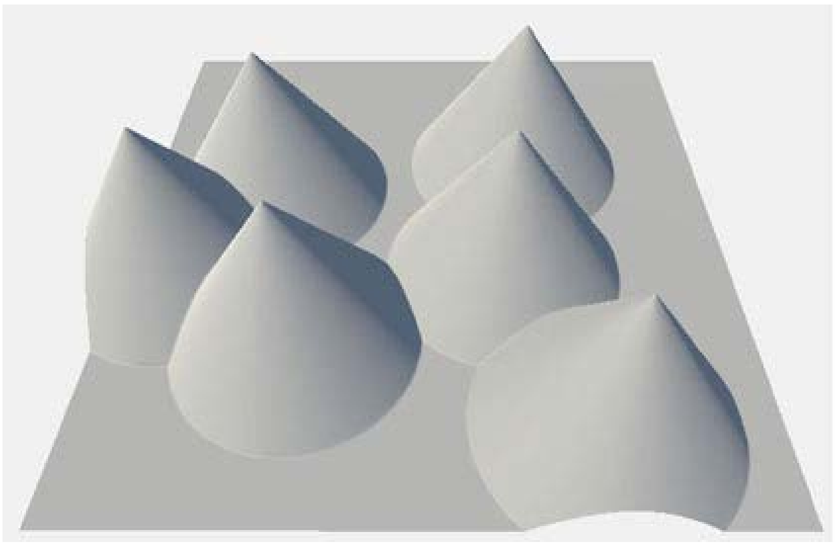
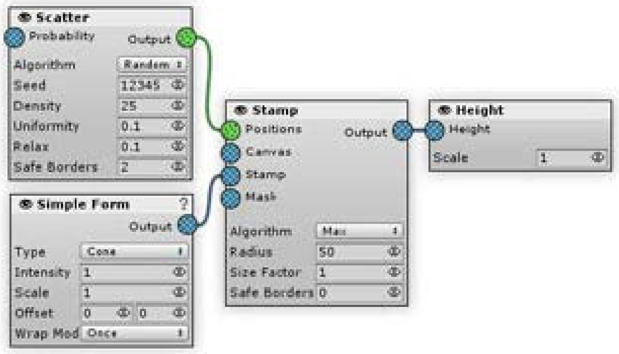
Takes a map that is used on the whole chunk, scales it down (in most cases), and stamps it on the object positions.



Scatter: Probability, Output; Algorithm Random; Seed 12345; Density 25; Uniformity 0.1; Relax 0.1; Safe Borders 2

Simple Form: Output; Type Cone; Intensity 1; Scale 1; Offset 0 0; Wrap Mod Once

Height: Height; Scale 1

*Stamp Source*



Scatter: Probability, Output; Algorithm Random; Seed 12345; Density 25; Uniformity 0.1; Relax 0.1; Safe Borders 2

Simple Form: Output; Type Cone; Intensity 1; Scale 1; Offset 0 0; Wrap Mod Once

Stamp: Positions, Canvas, Stamp, Mask, Output; Algorithm Max; Radius 50; Size Factor 1; Safe Borders 0

Height: Height; Scale 1

*Stamp Result*

### Inputs:

- Positions: object positions to perform stamps.
- Canvas: a map stamps will be applied to. If nothing specified the empty map is used.
- Stamp: source map that will be scaled and stamped.
- Mask: multiplies the generator intensity by the mask map's value. For the mask pixels that have a value of 0 the generator effect is invisible, for the pixels that have a value of 1 the intensity is unchanged.

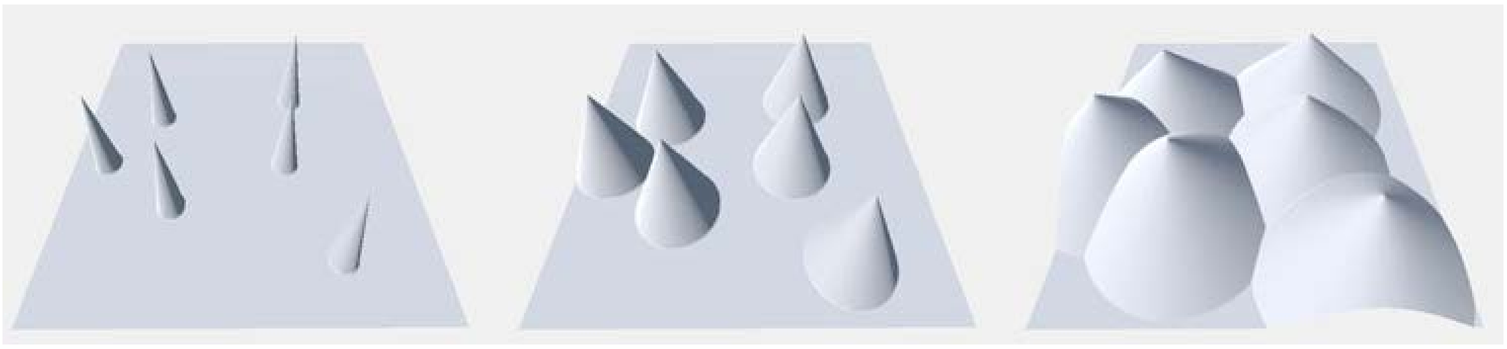
> **In some cases (especially when using Simple Form, RAW or Texture Input as Stamp source) when the terrain is *not* pinned at 0,0 you might notice that stamp generator seem to have no effect. This happens because the stamp generator takes the stamp from the current chunk, not from the one with 0,0 coordinate. So, if the current chunk has nothing to display then it will get nothing to stamp. You can check this by previewing stamp source on the terrain. Usually the easiest way to fix is enabling WrapMode in Simple Form or RAW/Texture Input**

## Outputs:

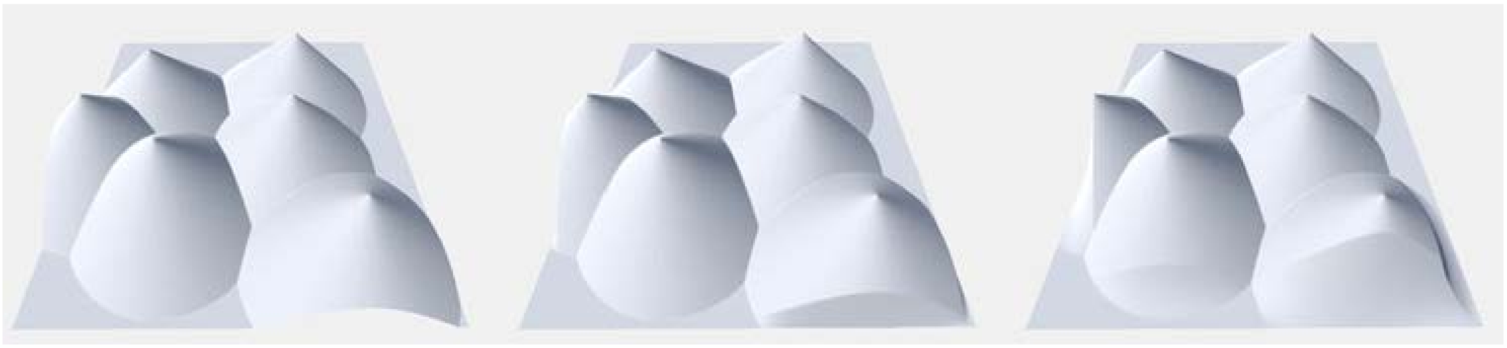
- Output - canvas (or empty) map with stamps applied.

## Properties:

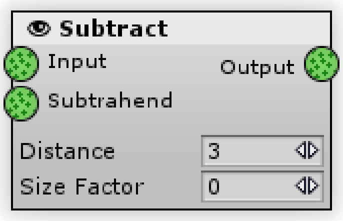
- Radius: the size of the stamps


*Radius: 10, 30, 70*

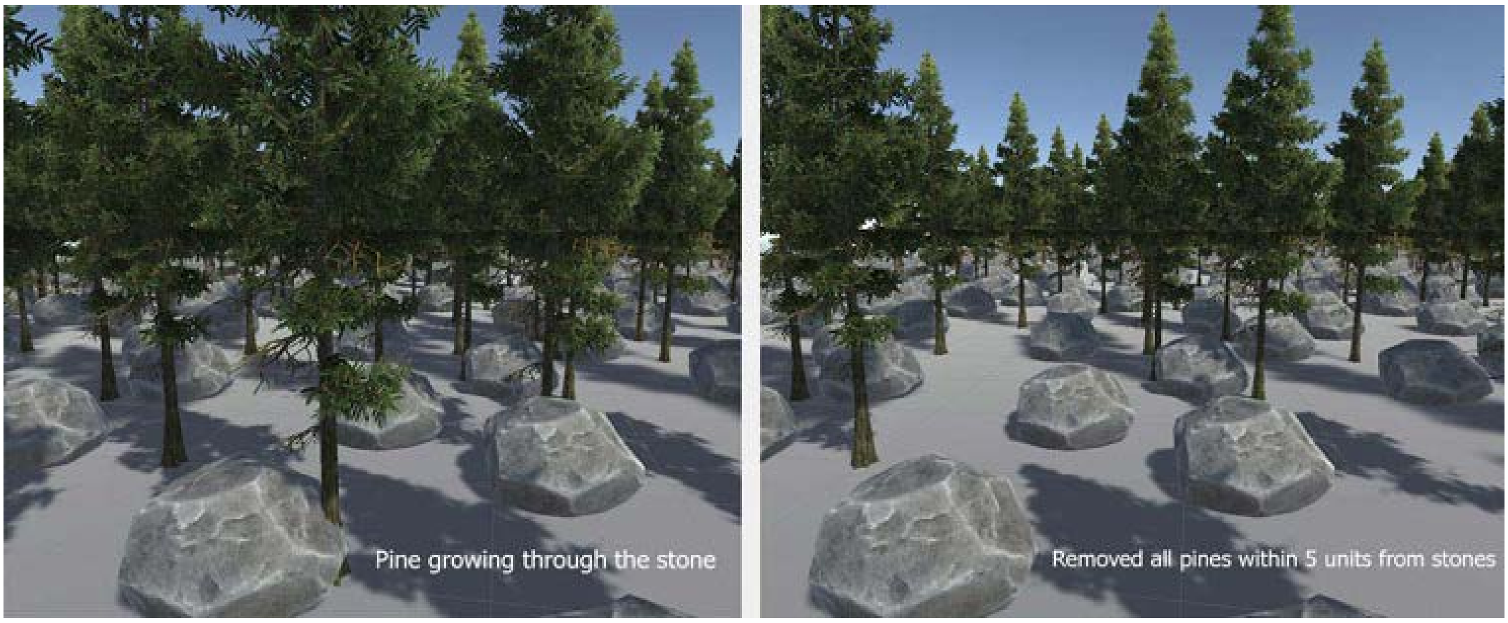
- Size Factor: multiplies the changes applied by this generator depending on the initial object size. For example, if the object's size is 2 then all of the adjustment values will be doubled if the factor is equal to 1. This is useful for proportional object changes: if the object should be lowered to half of it's height, for instance. When the value is 0 all of the objects are adjusted the same regardless of their size.
- Safe Borders: if not equal to zero generator effect will be fading down near the chunk border to prevent seams between chunks. Property value determines the fade length in world units.

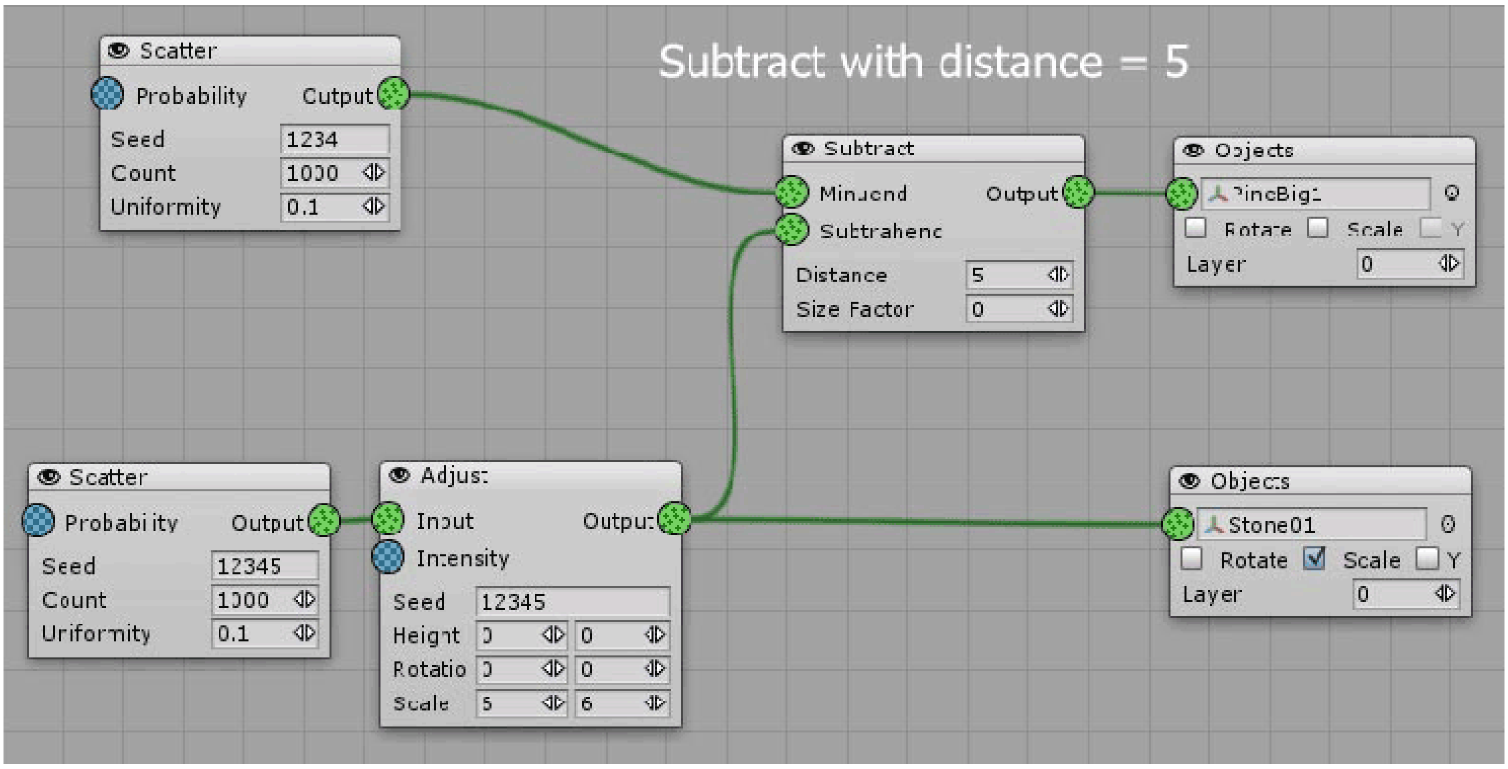

*Safe Borders: 0, 10, 50*

# Subtract


Subtract
Input
Output
Subtrahend
Distance 3
Size Factor 0

Returns the modified Minuend Input so that all objects positioned at a certain distance from Subtrahend Input objects are removed.


Pine growing through the stone
Removed all pines within 5 units from stones



Scatter: Probability, Output; Seed 1234; Count 1000; Uniformity 0.1
Subtract: Minuend, Output; Subtrahend; Distance 5; Size Factor 0
Objects: PineBig1 0; Rotate ☐; Scale ☐; Y ☐; Layer 0
Scatter: Probability, Output; Seed 12345; Count 1000; Uniformity 0.1
Adjust: Input, Output; Intensity; Seed 12345; Height 0 0; Rotatio 0 0; Scale 5 6
Objects: Stone01 0; Rotate ☐; Scale ✓; Y ☐; Layer 0

*Subtracting stones from pines removes all pine objects within range of 5*

## Inputs:

- ● Input - minuend, the object list (hash) whose objects will be removed.
- ● Subtrahend - a reference hash that defines which minuend objects will be removed. This object list (hash) is not changing and is not included in output.

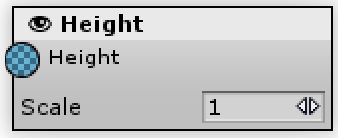
## Outputs:

- ● Output - a minuend object list (hash) with the defined objects removed

## Properties:

- Distance: if the distance between the minuend and any of the subtrahend objects is closer than this value, the object will not be included in the output(i.e. it will be removed). Note that the distance is measured in map resolution units, not in world units.
- Size Factor: makes a distance dependable from closest subtrahend object’s size. Useful for clearing areas around objects whose size may vary(like stones).

# Height



Applies the Input map as a terrain height.

## Output Generator

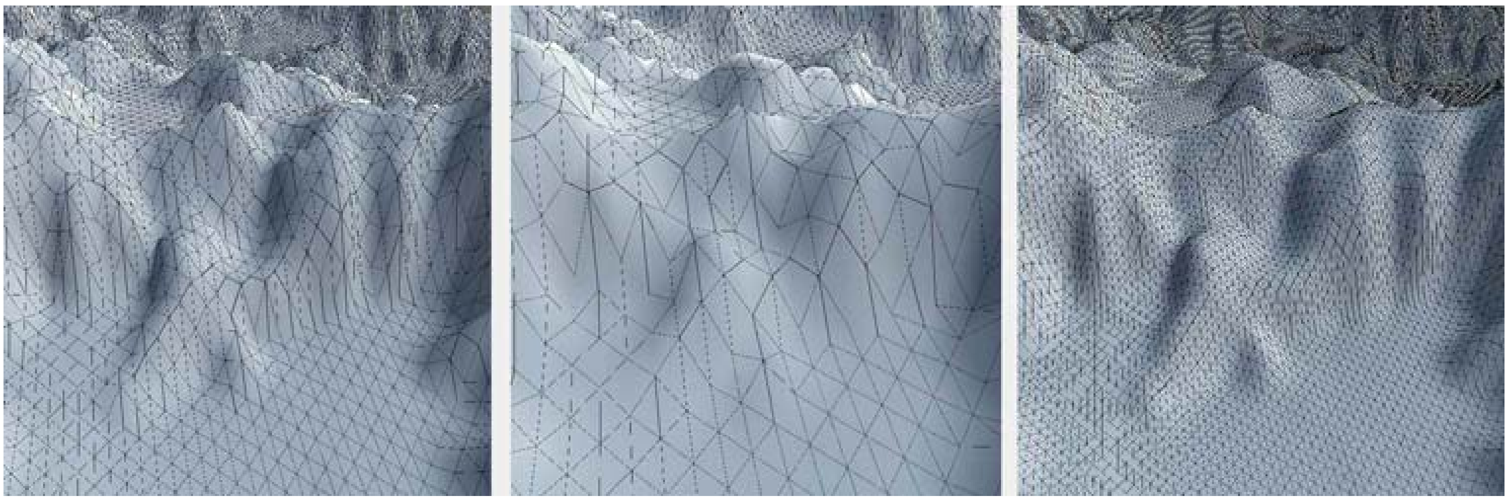
Output Generators bring all the map and object inputs to life: convert maps to terrain heightmap, splatmap or detailmap and place real objects on terrain. All of the generator chains should end up in one (or more) Output Generators - otherwise they will not be generated at all.

## Inputs:

- ● Input: a map that will be applied to terrain.

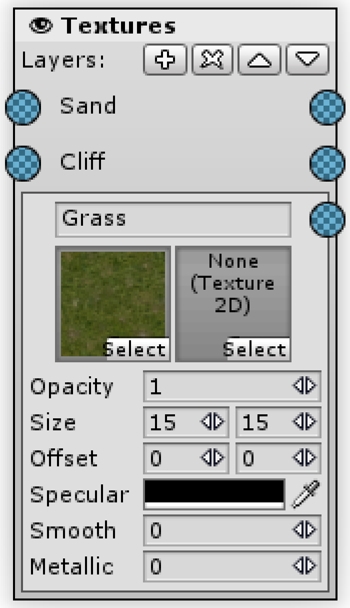
## Properties:

A scale parameter resizes the final heightmap before applicationapply: when set to 0.5 the terrain heightmap resolution will be the half of the map resolution that is set in Settings. When the parameter is equal to 2 it will upscale the map twice with bilinear filtering.



*Scale: 1, 0.5, 2*

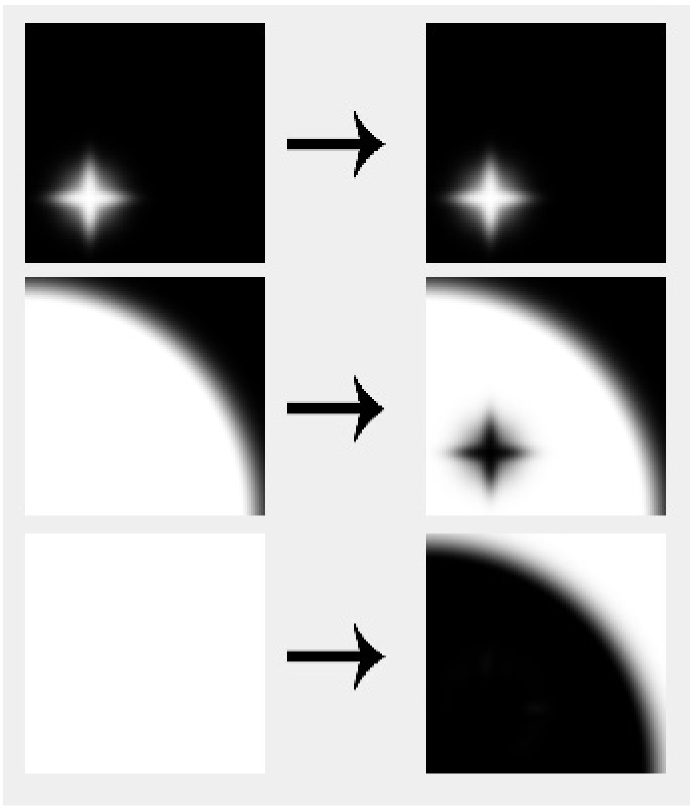
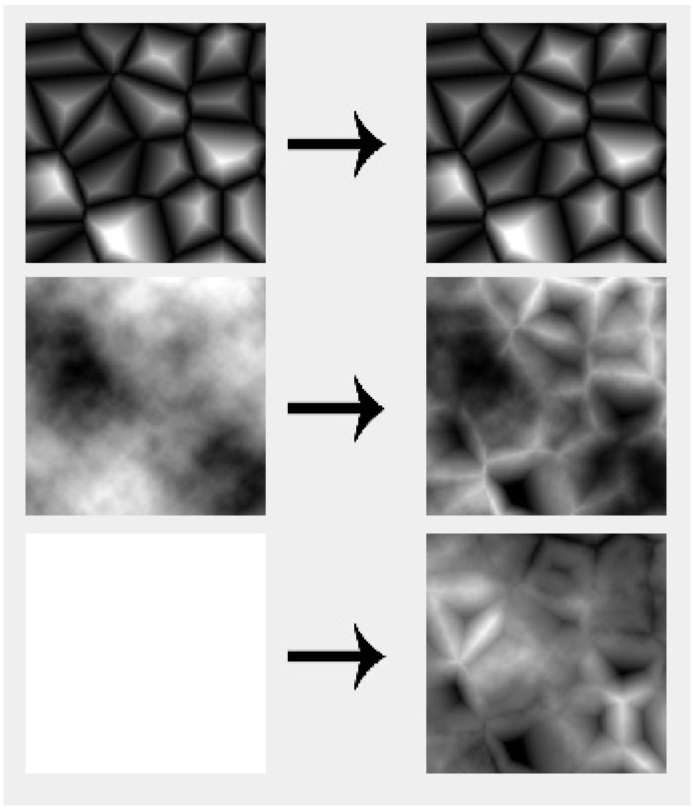
# Textures


Textures
Layers:
Sand
Cliff
Grass
None (Texture 2D)
Select
Select
Opacity 1
Size 15 15
Offset 0 0
Specular
Smooth 0
Metallic 0

Applies the terrain texturing information, i.e. “colors” the terrain with textures.

Texture Output final result depends on the layer order. Layers are blended together similarly to the layer system in Photoshop and other graphics editors: Each of the upper layers overlaps the lower ones. A mathematical algorithm for each layer generator multiplies all underlying layer values with the inverse value (1-value) of the current layer.

Texture Output layers can have output connections. These connections store a processed and blended layer mask. The sum of the output map values is always equal to 1. These outputs could be used for further map processing (for example, for planting grass using the grass map so it really gets on the terrain).

Each layer could be selected by clicking on it. Above the layers, next to the "Layers:" label you can see layer control buttons:

- Add: will add new layer atop of the selected one.
- Remove: removes currently selected layer.
- Up: will change the layers order by moving the selected layer up.
- Down: will change the layers order by moving the selected layer down.

The Background layer does not require an input since it is regarded as completely filled (a constant of 1). This layer could not be removed or sent up or down.

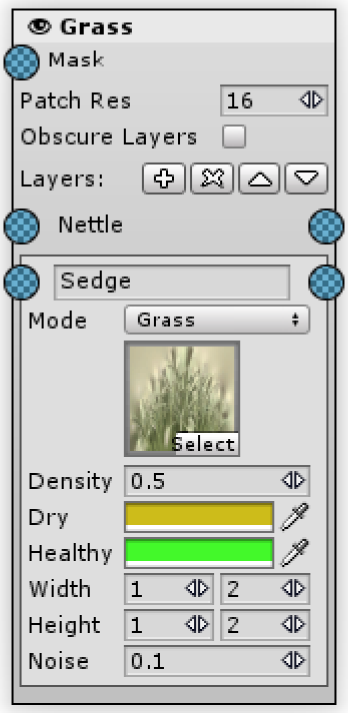
Note that due to better terrain welding splat prototypes the size parameter should be equal to a power of 2.

## Properties:

The layer properties are similar to the standard terrain texture parameters (splat prototypes).

- Opacity multiplies all of the input values before applying it

# Grass



Paints grass on terrain using layer's maps as a mask. This generator can place detail meshes as well.

## Inputs

- ● Mask - the overall map of grass density. For the mask pixels that have a value of 0 the grass is not planted, for the pixels that have a value of 1 grass is planted using the full density.

## Properties:

- Obscure Layers toggle makes the grass behave the same way as Texture Output layers. If turned on for each new layer the grass quantity is blended using the layer opacity. If turned off all grass layers are set up independently.
- A patch res parameter that applies to all of the layers. It specifies the size in pixels of each individually rendered grass patch. A larger number reduces draw calls, but might increase triangle count since detail patches are culled on a per batch basis. A recommended value is 16. If you use a very large detail object distance and your grass is very sparse, it makes sense to increase the value.

- Each layer has the following properties:

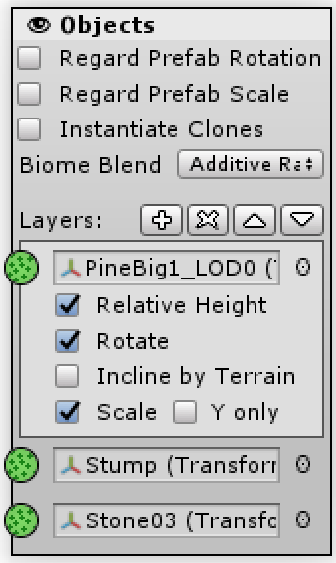
  - Mode: determines whether it will be a standard grass, a billboard grass or a detail mesh
    - Grass is the default method to display grass using the static planes.
    - Billboard grass images will rotate so that they always face the camera.
    - Vertex Lit mode is used to place a detail meshes.
    - Object mode is used to place custom grass bushes with opacity.

  - Texture field (for the grass modes): a grass albedo (diffuse) texture

  Object field (for the object/vertex lit mode): a detail mesh prefab
- The Noise Spread value controls the approximate size of the alternating patches, with higher values indicating more variation within a given area. (Tech note: the noise is actually generated using Perlin noise; the noise spread refers to the scaling applied between the x,y position on the terrain and the noise image.)
- The alternating patches of grass are considered more “healthy” at the centres than at the edges and the Healthy/Dry Color settings show the health of grass clumps by their color. (Tech note: the noise is actually generated using Perlin noise; the noise spread refers to the scaling applied between the x, y position on the terrain and the noise image.)
- Width and Height values specify the upper and lower limits of the size of the clumps of grass that are generated.

> The grass layer parameters are similar to the standard terrain grass and detail properties

# Objects



Applies objects to terrain. For each layer it instantiates the required number of objects and places them in the position and height prescribed by layer's input. Object size and rotation are used too, if the object's layer "Rotate" and "Scale" parameters are checked. Instantiated objects are grouped as terrain's child transforms.

Objects Output uses the object pool to instantiate objects faster: when the terrain gets out of generate distance its objects are not destroyed, but used to generate new terrain that appears within generate range. So be careful changing objects that are placed on terrain: a changed object will appear here and there as the player walks across the land. Use a separate prefab to make such unique objects.

## Properties:

- Regard Prefab Rotation: in some cases your prefab might have an initial connection (for example, 90 degree X-axis connection for the objects that were exported from 3DS Max). When this feature enabled, the final object rotation will be combined, allowing you to place those objects properly. If it's disabled the prefab instantiated will have (i.e. 0,Y,0) rotation no matter of it's origin transform.
- Regard Prefab Scale: same as previous for the scale. If enabled, the final scale will be multiplied with the initial prefab scale. Useful for placing small objects that were upscaled in a prefab.
- Instantiate Clones: by default in editor mode new objects will be created maintaining the prefab connection, in playmode objects are instantiating using prefab clones. Enabling this will make MM instantiate objects using prefab clones both in editor and playmode. This will increase the apply speed in editor, but the objects instantiated will loose prefab connections.
- Biome Blend: the way objects are blended between the biomes

> Settings mentioned above are static, i.e. shared between all the Object Output Generators in scene (including biomes).

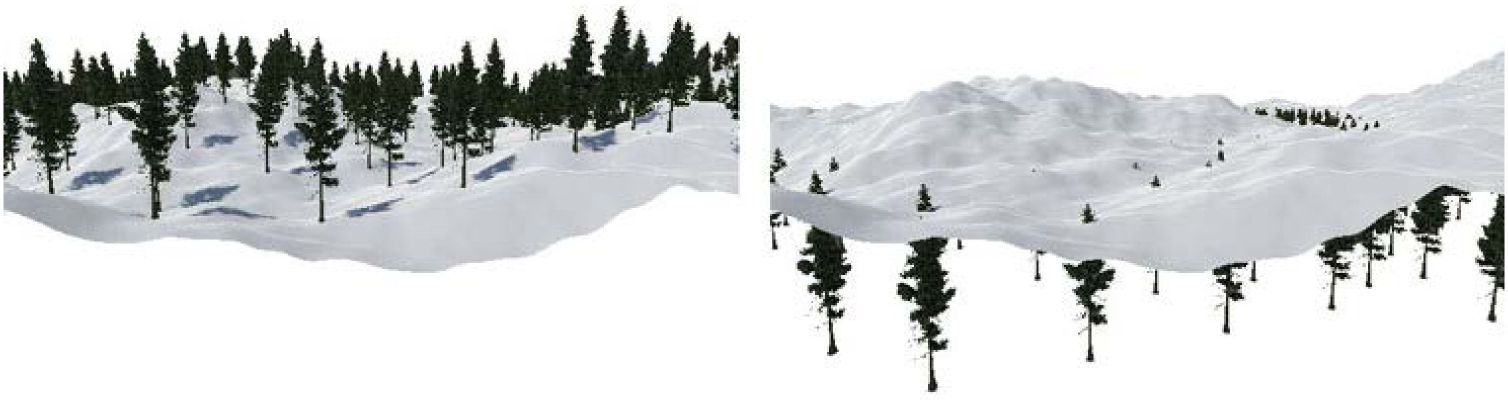
Object Generator input maps are stacked using the layers mode. Each layer could be selected by clicking on it. Above the layers, next to the "Layers:" label you can see layer control buttons:

- ✚ Add: will add new layer atop of the selected one.

- Remove: removes currently selected layer.
- Up: will change the layers order by moving the selected layer up.
- Down: will change the layers order by moving the selected layer down.

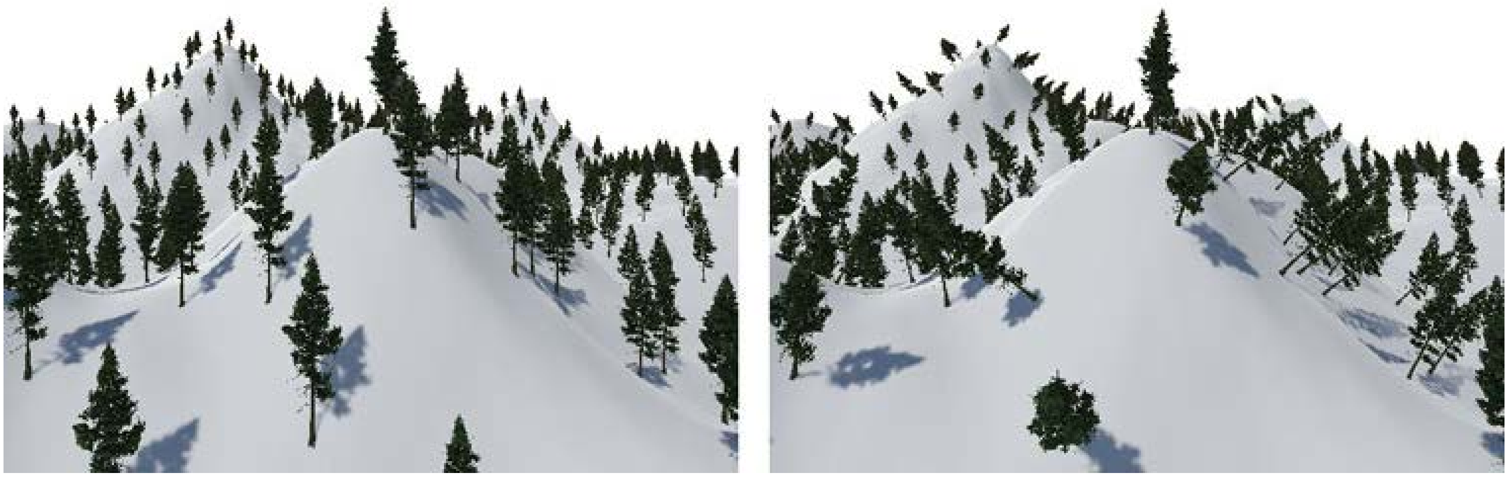
Each layer has these properties:

- Prefab field: a prefab that will be instantiated for each input object
- Relative Height: if enabled the objects are placed relatively to the terrain level. If the object height level is 0, it will be placed exactly on the terrain, if 13 then it will be placed 13 meters above the terrain. When disabled, the object will be placed relatively to the scene Y zero level: if object level is 13 it will be placed at the height 13 no matter of the terrain height in current position.



*Relative Height On / Off*

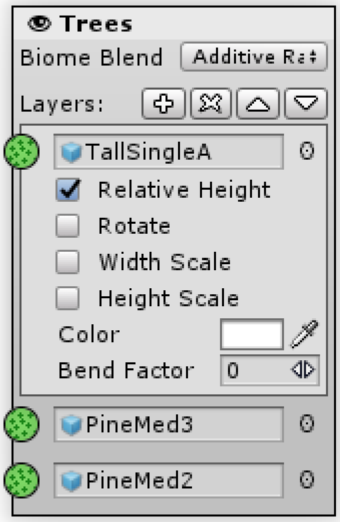
- Rotate: will rotate an object around the Y axis according to its rotation value set in the input object's hash.
- Incline By Terrain: will force objects to take the terrain normal.



*Incline By Terrain On / Off*

- Scale: will scale an object. If the Y parameter is turned on then the object will be scaled along the Y-axis (height) only. If it is off then the object will be scaled uniformly.

## Trees



Tree Output works similarly to Objects Output: for each layer it instantiates several trees of a given type at the positions prescribed by layer's input. The only difference is that trees in this case are not the Transforms but the terrain Tree Instances.

### Properties:

- Biome Blend: the way trees are blended between the biomes

Tree Generator input maps are stacked using the layers mode. Each layer could be selected by clicking on it. Above the layers, next to the "Layers:" label you can see layer control buttons:

- Add: will add new layer atop of the selected one.
- Remove: removes currently selected layer.
- Up: will change the layers order by moving the selected layer up.
- Down: will change the layers order by moving the selected layer down.

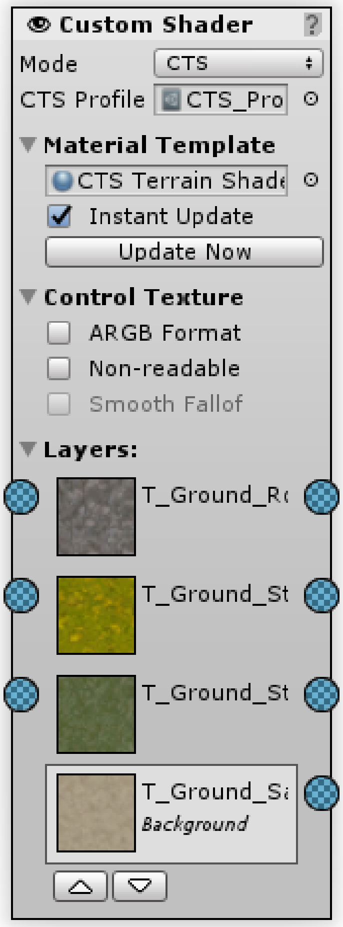
Each layer has these properties:

- Prefab field: a prefab that will be instantiated for each input object
- Rotate: will rotate an object around the Y-axis according to its rotation value set in the input object's hash.
- Scale Width, Scale Height: the X/Z and Y scale axes can be toggled independently. Switching Width on and Height off will make the trees scale along X and Z axises, while Y scale will always be equal to 1, and vice versa. To scale uniform check both toggles, to disable scaling uncheck both.

> **Note that prefab trees that do not have LODs (i.e. those who use billboard imposters) could not be rotated or scaled. It's not a MapMagic bug, it is the way Unity works.** Possible solution. Note that while the tree is not rotated, it's collider does, so turn off rotation and scale feature if you are using imposter trees.

- Color Tint: multiplies tree type with a given color.
- Bend Factor: the influence of the tree to the wind zone.

# Custom_shader



A special output generator designed to work with a 3rd party terrain shaders:

- Relief Terrain Pack;
- MegaSplat;
- Complete Terrain Shader
  An appropriate asset should be installed.

Furthermore, this output generator can work with almost any terrain shader that uses control textures. If you are using your own terrain shader switch mode to "Custom".

This generator should be used instead of Texture Output generator.

## Properties:

- Mode: selects the type of the shader that should be used.
- CTS Profile / MS Texture List / RTP Component: the main shader asset where all of the shader properties are kept. It's used in MapMagic to get available layers and to perform Instant Update.
- In custom mode you will see the control textures list. You can set the control textures names here, and total control textures count below.

## Material Template:

A custom material source that will be used to create chunk materials. The material assigned here is not actually used on the terrain - MapMagic clones it for each chunk, assigns the proper unique control maps to copy and applies copy to terrain.

This field is the instance of the Terrain Settings / Custom Material field.

> Theoretically MaterialPropertyBlock could be used instead of material template. It would be faster and will not require Instant Update, but it cannot be serialized (i.e. saved for pinned terrains). You can try experimenting with MaterialPropertyBlock by un-commenting the block in Apply method.

Since every chunk uses it's own individual material changing CTS Profile / MS Texture List / RTP Component properties will have no visual effect. To copy all of the shader asset properties to terrains you have to click "Update Now" button.

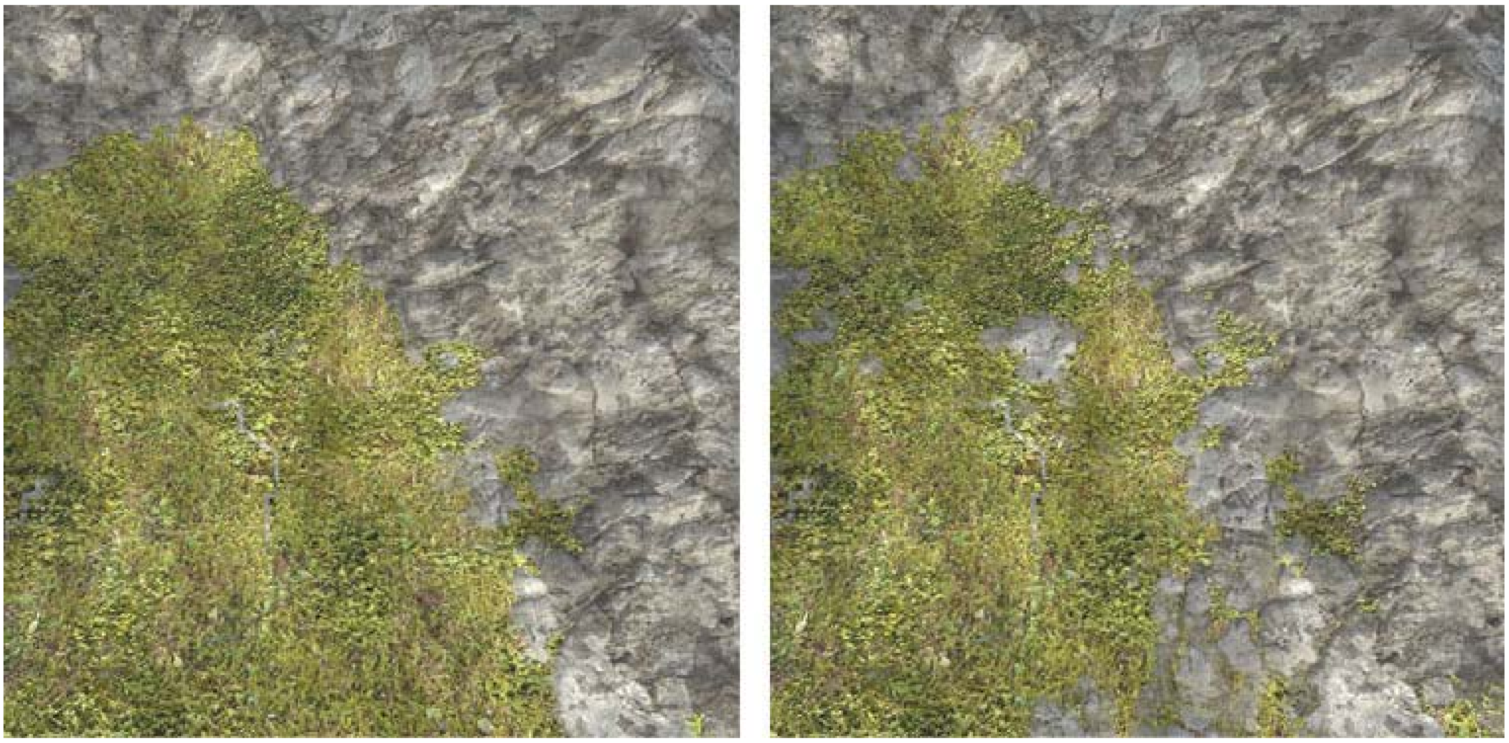
Instant Update will click this button every frame the scene view changes.

> **Don't forget to turn "Instant Update" mode off after you're happy with the shader settings since updating all the materials every frame is resource intensive.**

## Control Texture:

Textures that control map blending in shaders are called "Control Textures". That's the very textures that are unique for each of the terrain.

- ARGB Format will force using ARGB instead of RGBA. This is required for MegaSplat since v.1.14
- Non-readable will turn off the control textures read/write mode after the texture data has been applied. This will reduce the texture size, but may conflict with the custom shader asset.
- Smooth Fallof (MegaSplat) will make gradients between textures a bit more smooth and increase texture transitions areas.


*Smooth Fallof: Off / On*

## Layers:

Input maps are stacked using the layers mode. Each layer could be selected by clicking on it. Below the layers you can see layer control buttons:

- ✚ Add (MegaSplat mode): will add new layer atop of the selected one.
- ✖ Remove (MegaSplat mode): removes currently selected layer.
- △ Up: will change the layers order by moving the selected layer up.
- ▽ Down: will change the layers order by moving the selected layer down.

Base layer could not be removed or sent up or down.

Up and down buttons are not displayed if the layer count is zero.

To add, remove or change layer properties an appropriate asset should be used. "Refresh All" in RTP settings might be required after the first properties change.

Each of the layers (exept Background) has it's own Inputs and Outputs. Layers are blended similarly to Texture Output generator.

### Warnings and Fixes

In order to use Custom Shader output properly you've got to set up MapMagic itself:

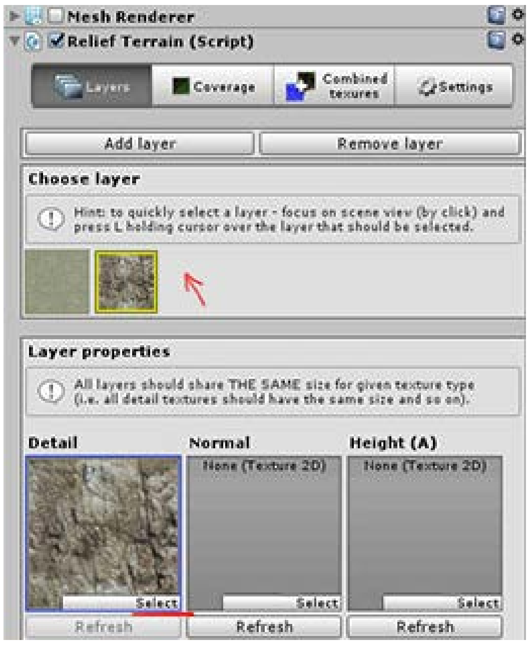
- Switch Material Type to custom in Terrain Settings;
- Assign a proper Custom Material template with CTS/MegaSplat/RTP shader;
- Turn off "Assign Custom Material" feature - it will apply the Custom Material to terrain, while we need to use it's clones with a custom control maps;
- Disable base map for terrains since most shaders do not use it;
- Assign CTS Profile / MS Texture List in a proper slot in generartor. In case of RTP ReliefTerrain component should be assigned to MapMagic object.

In case if you forget to do something from that list you will see a warning with a "Fix" button. Pressing this button will fix it automatically.

## Tutorials:

### Using RTP:

- Create a new scene, add MapMagic object, open editor and add CustomShader Output;
- Switch Mode to RTP;
- Fix all the warnings pressing "Fix" buttons. On adding RTP component you might see a message that _RTP_LODmanager object has been created;
- Select MapMagic object. In ReliefTerrain component, Layers tab, select each of the layers in Choose Layer (they are not visible by default) and assign it's diffuse, normal and height textures;


Mesh Renderer

Relief Terrain (Script)

Layers | Coverage | Combined texures | Settings

Add layer | Remove layer

**Choose layer**

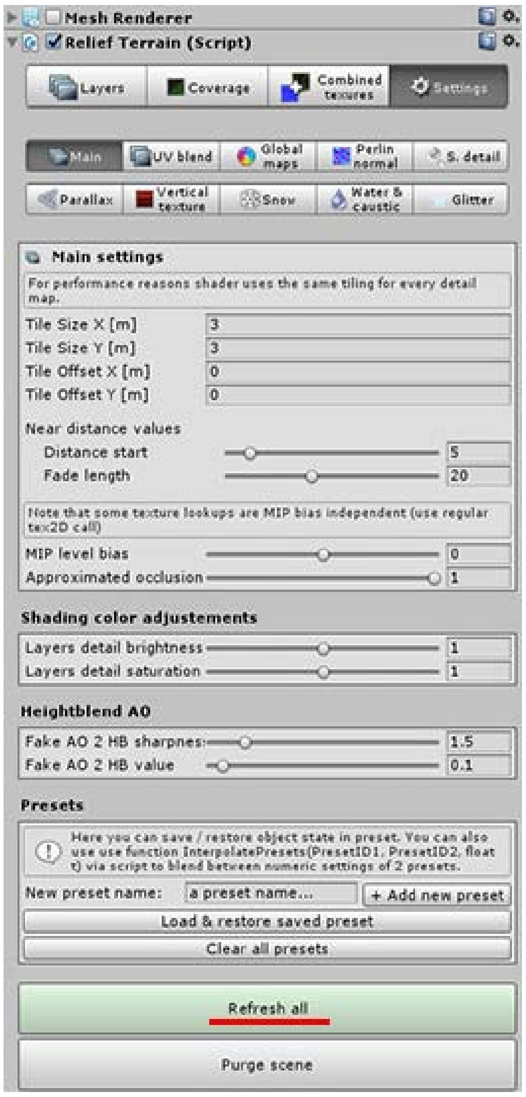
Hint: to quickly select a layer - focus on scene view (by click) and press L holding cursor over the layer that should be selected.

**Layer properties**

All layers should share THE SAME size for given texture type (i.e. all detail textures should have the same size and so on).

| Detail | Normal | Height (A) |
|---|---|---|
| | None (Texture 2D) | None (Texture 2D) |
| Select | Select | Select |
| Refresh | Refresh | Refresh |

- Go to RTP Settings / Main and press Refresh All button;


Mesh Renderer

Relief Terrain (Script)

Layers | Coverage | Combined texures | Settings

Main | UV blend | Global maps | Perlin normal | S. detail

Parallax | Vertical texture | Snow | Water & caustic | Glitter

**Main settings**

For performance reasons shader uses the same tiling for every detail map.

| | |
|---|---|
| Tile Size X [m] | 3 |
| Tile Size Y [m] | 3 |
| Tile Offset X [m] | 0 |
| Tile Offset Y [m] | 0 |

Near distance values

| | |
|---|---|
| Distance start | 5 |
| Fade length | 20 |

Note that some texture lookups are MIP bias independent (use regular tex2D call)

| | |
|---|---|
| MIP level bias | 0 |
| Approximated occlusion | 1 |

**Shading color adjustements**

| | |
|---|---|
| Layers detail brightness | 1 |
| Layers detail saturation | 1 |

**Heightblend AO**

| | |
|---|---|
| Fake AO 2 HB sharpnes: | 1.5 |
| Fake AO 2 HB value | 0.1 |

**Presets**

Here you can save / restore object state in preset. You can also use use function InterpolatePresets(PresetID1, PresetID2, float t) via script to blend between numeric settings of 2 presets.

New preset name: a preset name... | + Add new preset

Load & restore saved preset

Clear all presets

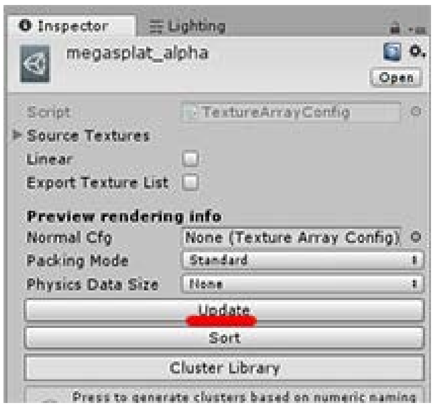
Refresh all

Purge scene

- In Custom Shader Output node / Material Template press Update Now;
- Connect Custom Shader Output node layers to the other graph.

## Using MegaSplat:

- Create a new scene, add MapMagic object, open editor and add CustomShader Output;
- Switch Mode to MegaSplat;
- Assign MegaSplat texture list. It is generated from the MS Texture Array Config when pressing "Update" button;



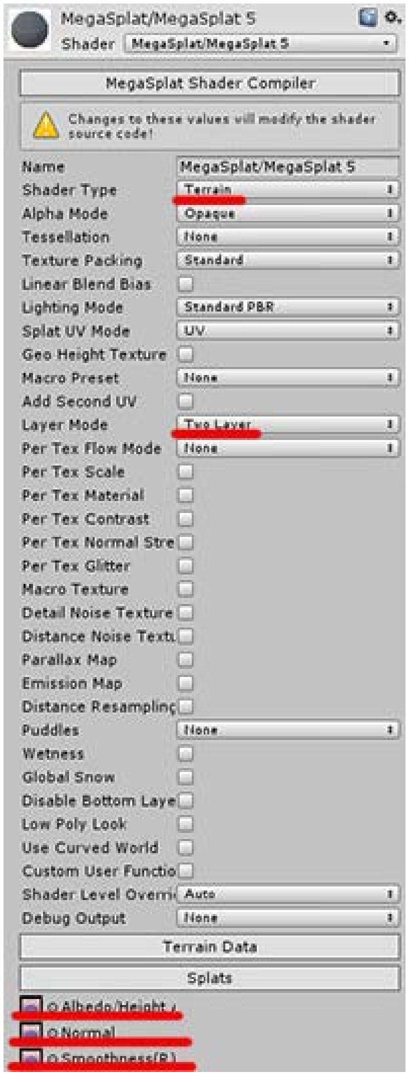
megasplat_alpha

| | |
|---|---|
| Script | TextureArrayConfig |
| Source Textures | |
| Linear | ☐ |
| Export Texture List | ☐ |

**Preview rendering info**

| | |
|---|---|
| Normal Cfg | None (Texture Array Config) |
| Packing Mode | Standard |
| Physics Data Size | None |

- Fix all the warnings pressing "Fix" buttons. Fixing a material will create a new MegaSplat N shader in MegaSplat folder;
- In the material in Material Template slot:
  - Switch Shader Type to terrain;
  - Layer Mode: Two Layer;
  - Assign in Albedo/Height, Normal and Smoothness texture arrays slots appropriate texture arrays;



MegaSplat/MegaSplat 5
Shader MegaSplat/MegaSplat 5

Changes to these values will modify the shader source code!

| | |
|---|---|
| Name | MegaSplat/MegaSplat 5 |
| Shader Type | Terrain |
| Alpha Mode | Opaque |
| Tessellation | None |
| Texture Packing | Standard |
| Linear Blend Bias | ☐ |
| Lighting Mode | Standard PBR |
| Splat UV Mode | UV |
| Geo Height Texture | ☐ |
| Macro Preset | None |
| Add Second UV | ☐ |
| Layer Mode | Two Layer |
| Per Tex Flow Mode | None |
| Per Tex Scale | ☐ |
| Per Tex Material | ☐ |
| Per Tex Contrast | ☐ |
| Per Tex Normal Stre | ☐ |
| Per Tex Glitter | ☐ |
| Macro Texture | ☐ |
| Detail Noise Texture | ☐ |
| Distance Noise Textu | ☐ |
| Parallax Map | ☐ |
| Emission Map | ☐ |
| Distance Resampling | ☐ |
| Puddles | None |
| Wetness | ☐ |
| Global Snow | ☐ |
| Disable Bottom Laye | ☐ |
| Low Poly Look | ☐ |
| Use Curved World | ☐ |
| Custom User Functio | ☐ |
| Shader Level Overri | Auto |
| Debug Output | None |

Albedo/Height
Normal
Smoothness(R)

- In Custom Shader Output node / Material Template press Update Now;
- If you got MegaSplat version <1.14 turn on ARGB Format in Custom Shader Output / Control Texture;

- Add new layers in Clusters section, select textures and connect them to the other graph.

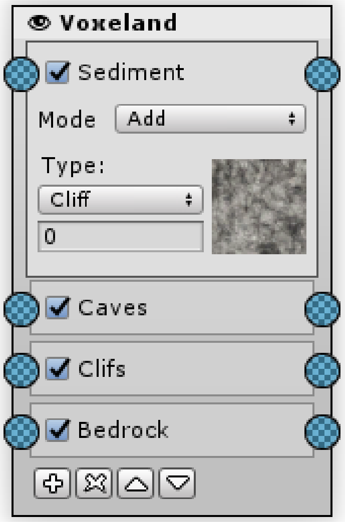
## Using CTS:

- Create a CTS profile with the standard way: create terrain, assign textures, add CTS component, create and apply profile (menu/Component/CTS/Create and apply profile), delete terrain;
- Create MapMagic object, open editor and add Custom Shader Output;
- Switch Mode to CTS;
- Fix all the warnings pressing "Fix" buttons;
- In Custom Shader Output node / Material Template press Update Now;
- Connect Custom Shader Output node layers to the other graph.

## Using proprietary terrain shader

- Create MapMagic object, open editor and add Custom Shader Output;
- Switch Mode to Custom;
- Set the number of control textures used in your shader;
- Set the names of your control textures as they are shown in a shader (they are usually starting with underscore);
- Fix all the warnings pressing "Fix" buttons;
- Assign or set up material template that uses your shader;
- Connect layers to the graph, set the proper control texture and channel for each layer.

# Voxeland



Outputs Voxeland terrain. Requires Voxeland to be installed. This generator replaces the standard both Height and Texture outputs.

> **Voxeland Output works only when MapMagic is is used as a Voxeland's generator. In order to use it you have to create Voxeland terrain, switch Generate Type to MapMagic, and create a new graph.**

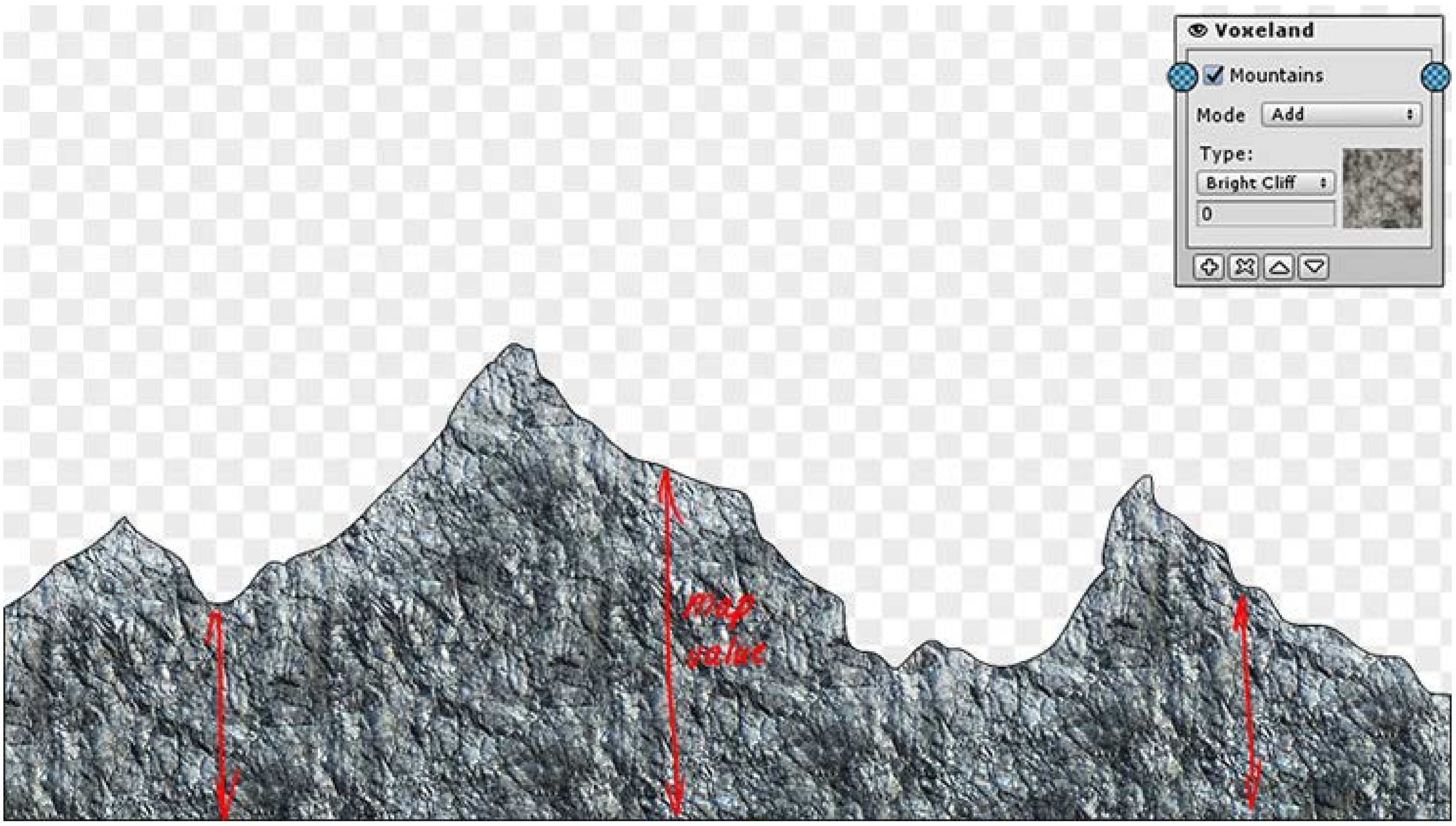
Voxeland Output block types are stacked using the layers mode. Each layer could be selected by clicking on it. Below the layers you can see layer control buttons:

- ✚ Add: will add new layer atop of the selected one.
- ⌘ Remove: removes currently selected layer.
- △ Up: will change the layers order by moving the selected layer up.
- ▽ Down: will change the layers order by moving the selected layer down.

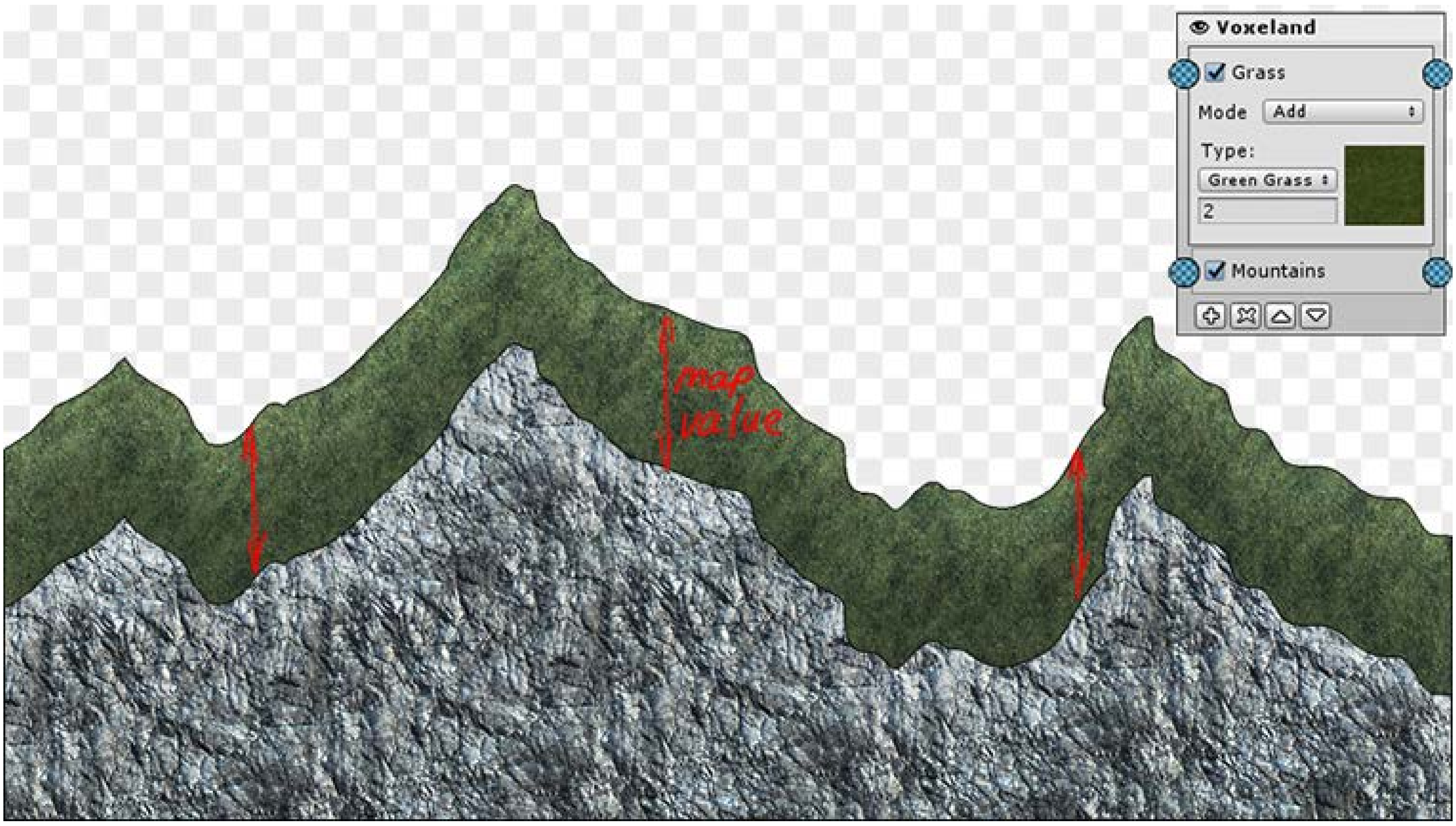
Unlike the usual 2D terrains, Voxeland is a true 3D terrain that consists of blocks. Voxeland data stores the information about all of the blocks, not only the top once. For example, if you dig the grass that should have a bedrock cliff underneath, you will find that the cliff blocks are still here. This way Voxeland terrain could be just "painted" with some texture, instead all of the blocks in a volume should be set. Voxeland Output offers a way to do that using layers.

Each layer could be applied using one of these blending types:

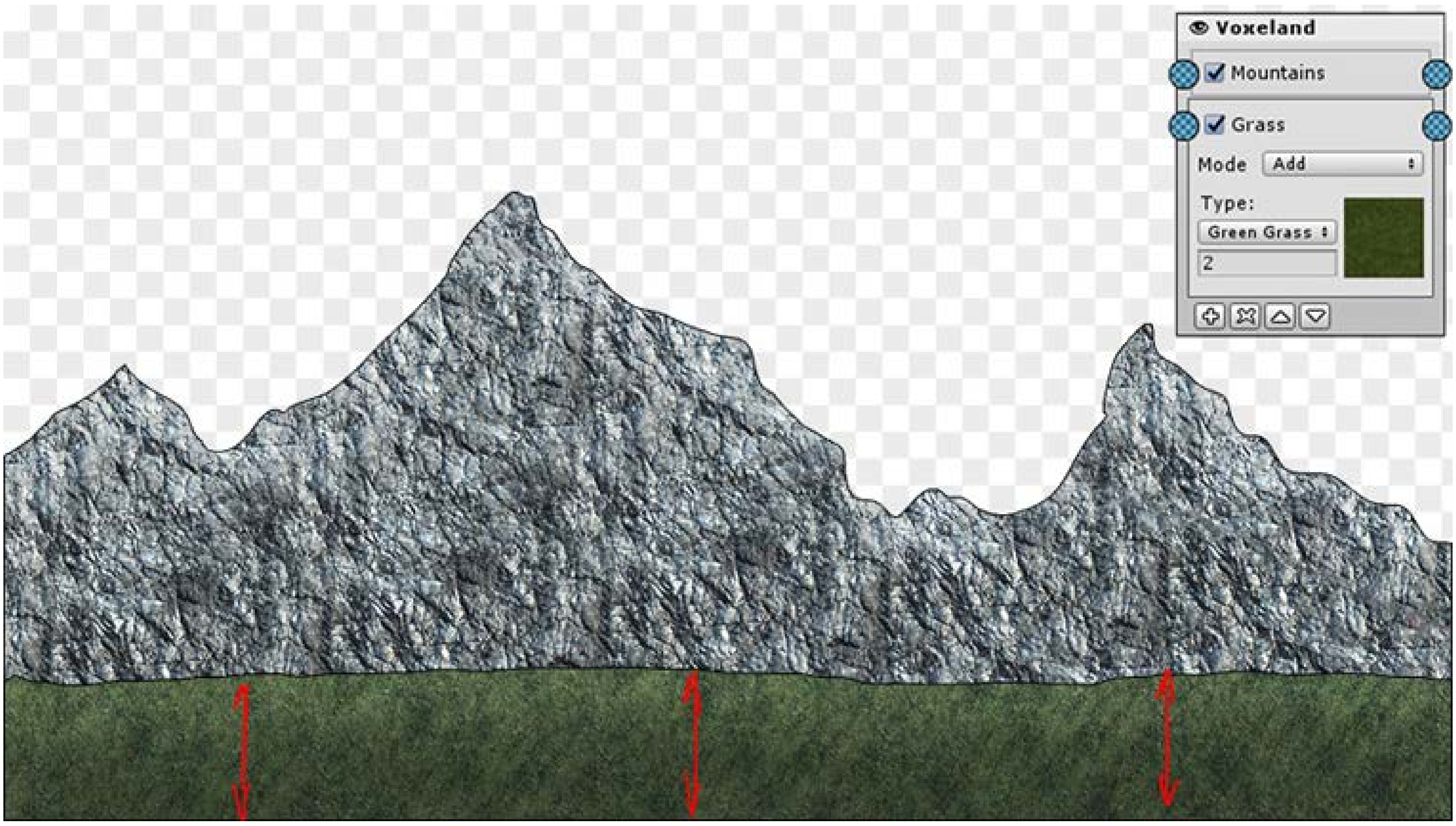
- **Add**: adds the layer above already existing terrain. Just like a dump track unloading gravel.



*First layer (cliff) applied with the "Add" mode.*

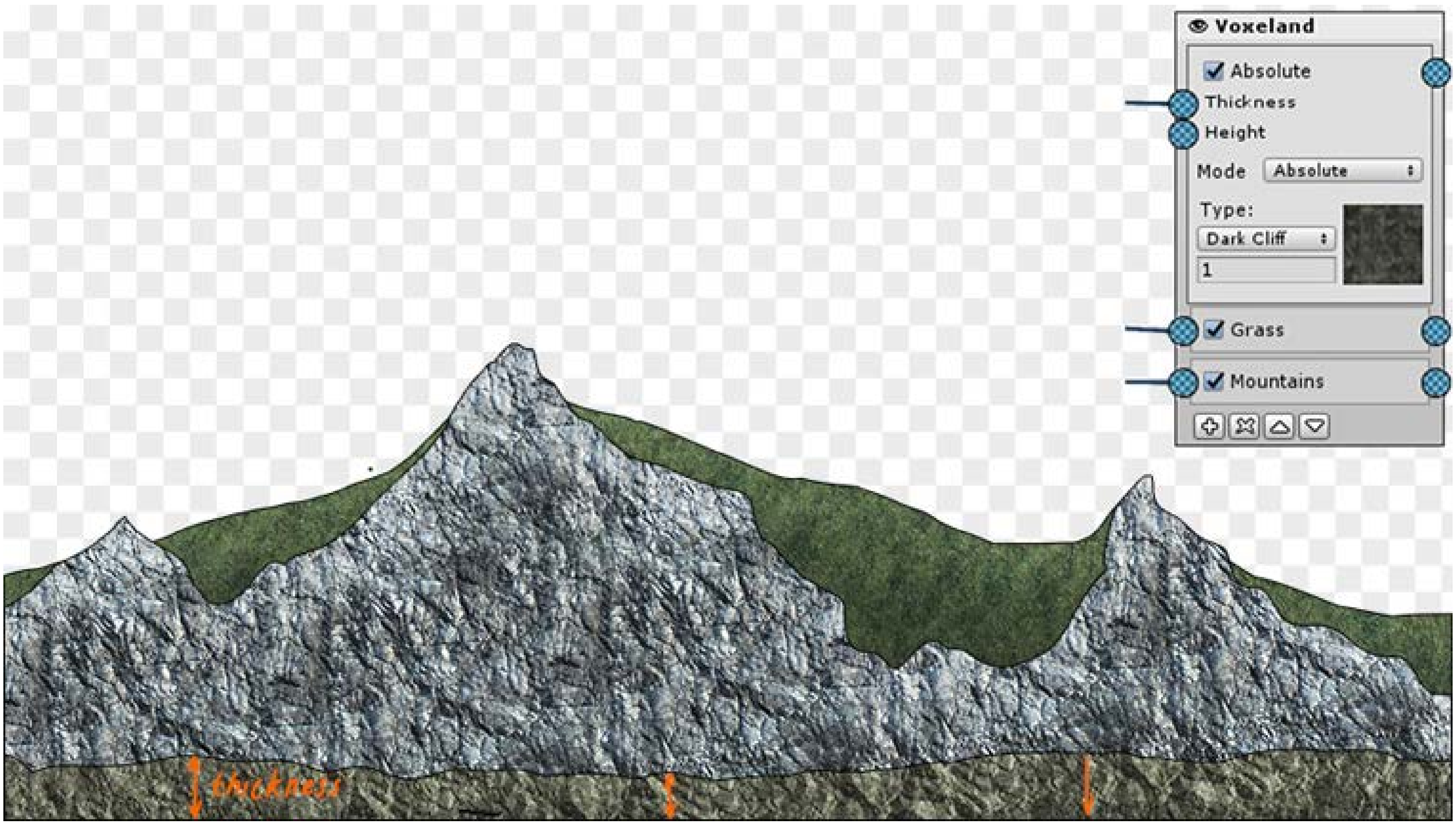


*Second layer (grass) applied with the "Add" mode.*

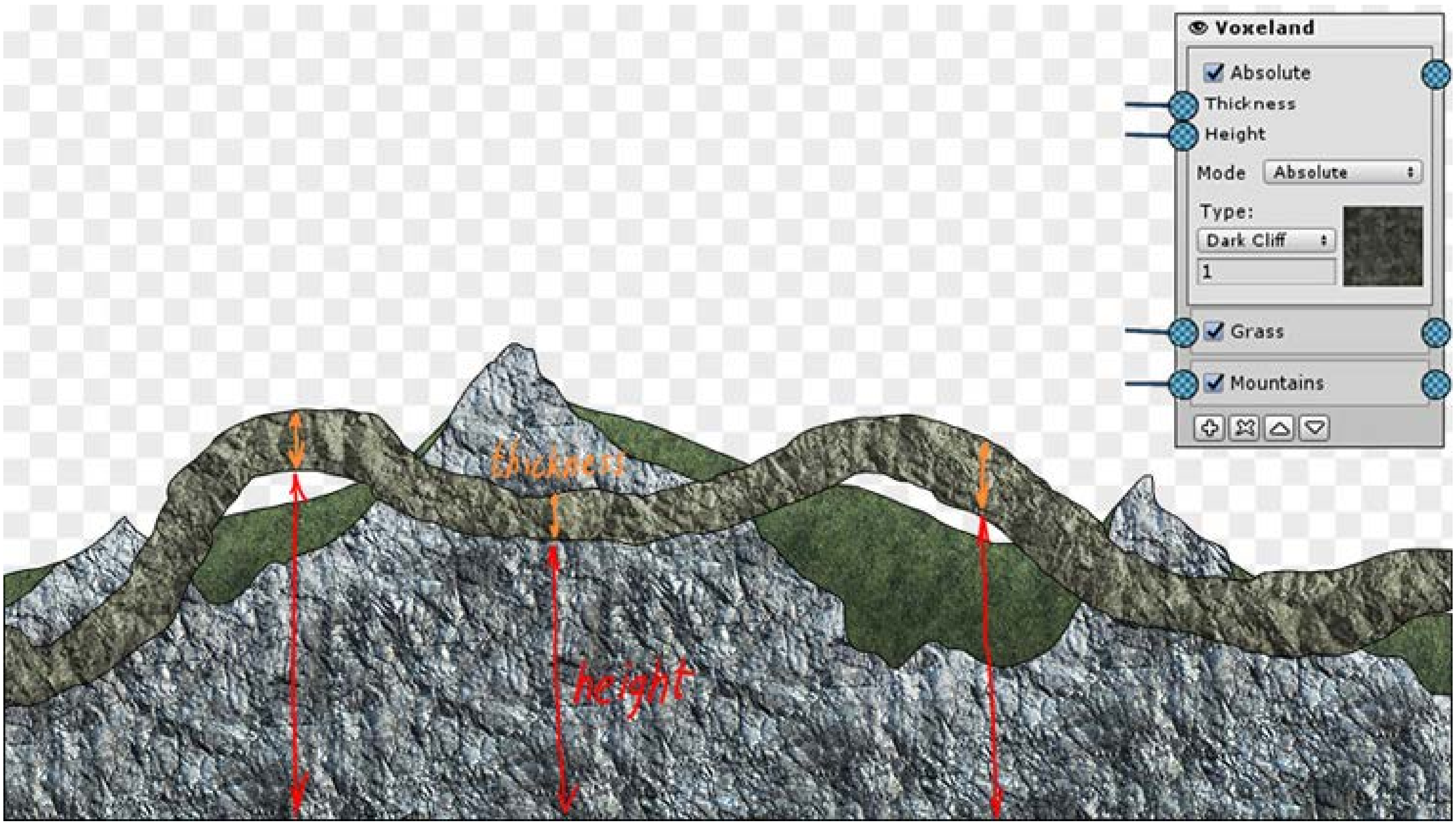


*Same layers applied in a reverse sequence.*

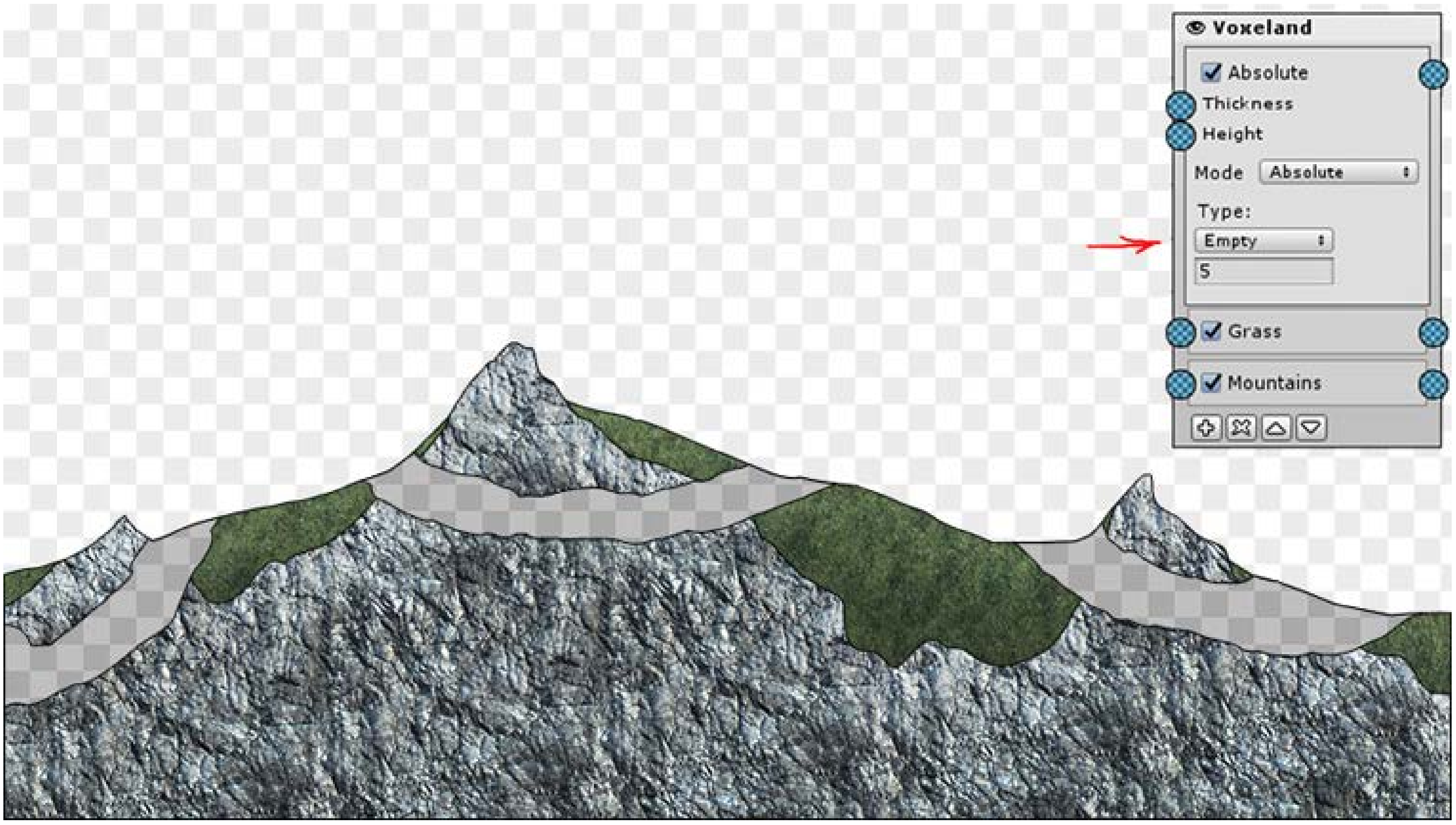
- **Absolute**: simply sets the layer using the height map provided. The way the layer set depends only on a height values, and does not depend on the current terrain height. So it can replace already existing layers or even hover above the terrain.



*If the height is not provided the layer will be applied at the zero level*

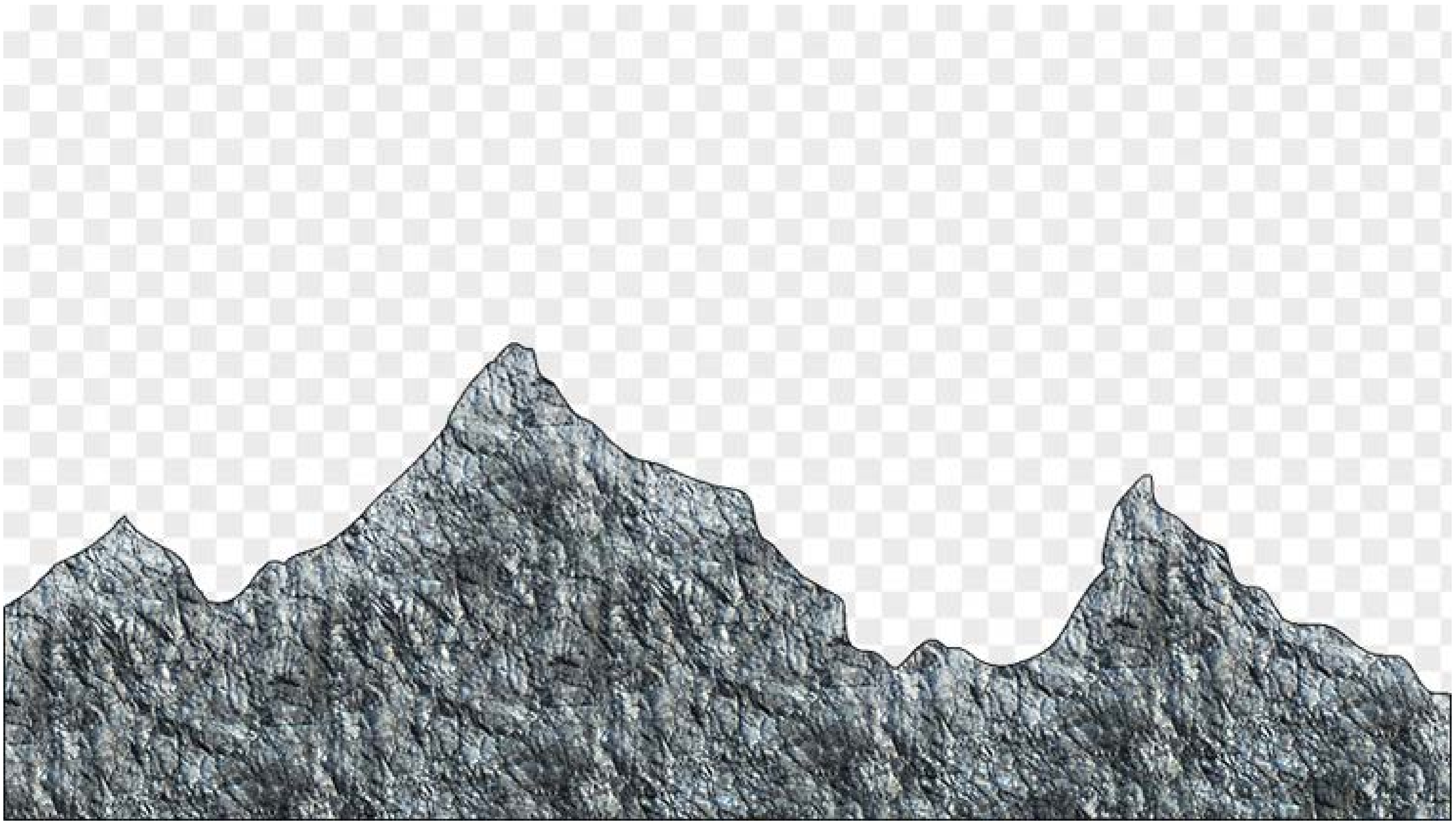


*Providing a height will make it intersect the terrain, crossing all of the existing layers, including the "air" one.*

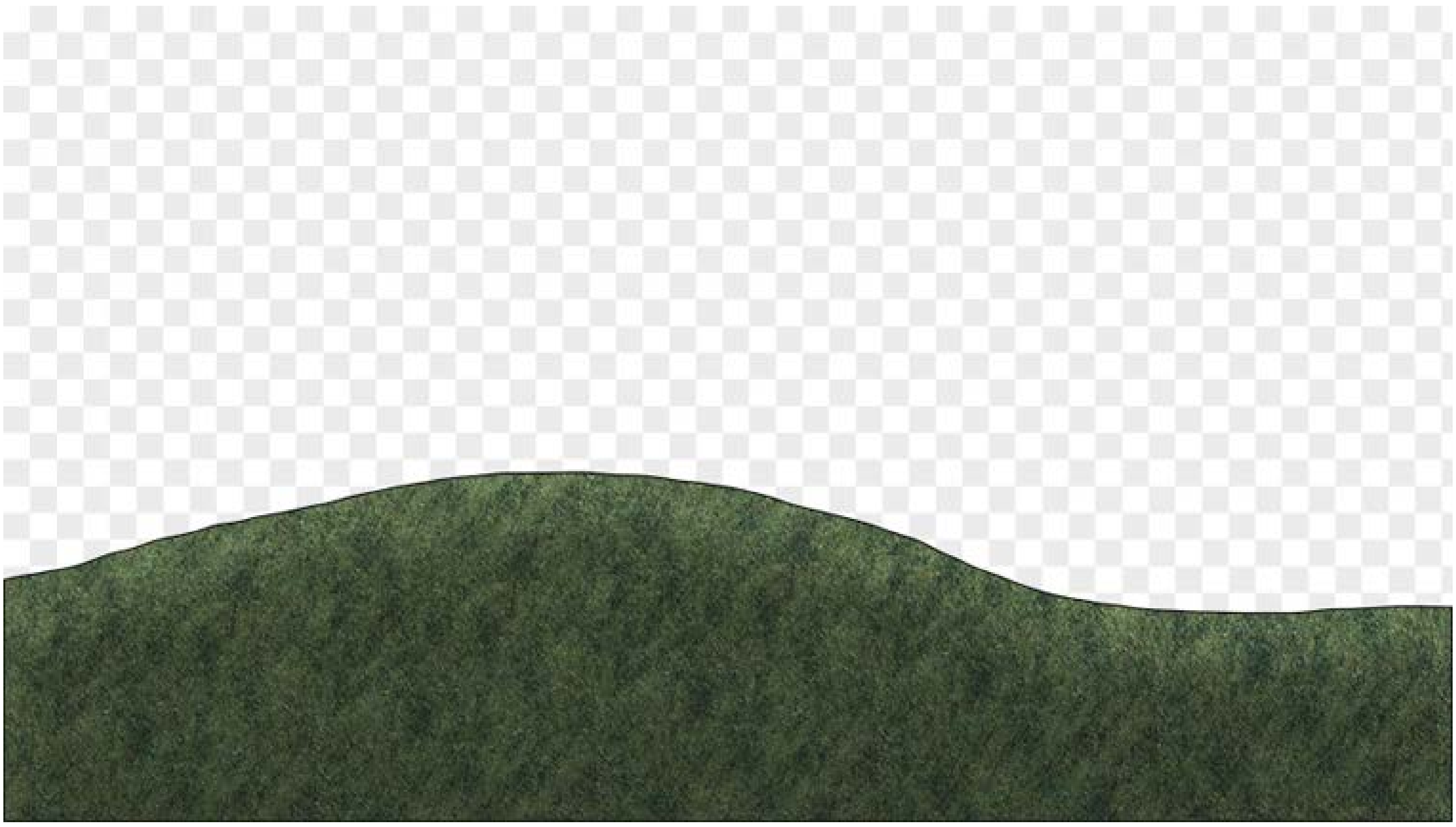


*By switching the layer type to "Empty" you can cut the terrain with the layer, making caves in it.*

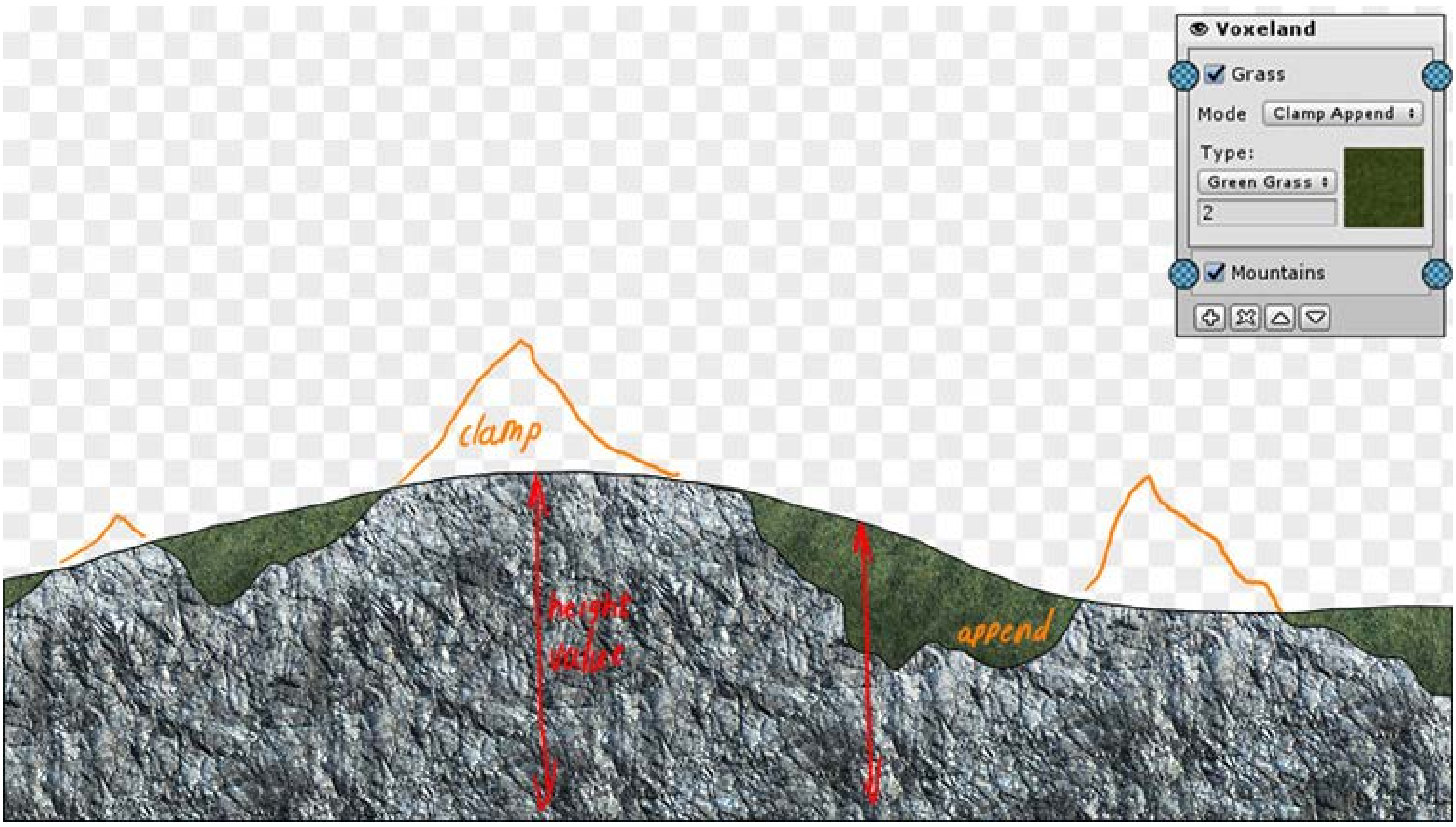
- **Clamp/Append**: applies the layer at the zero level, comparing two heightmaps:
  - If the layer height is lower than the terrain height, it clamps the terrain to the level height.
  - If the layer height is higher that the terrain height, it adds layer type blocks to the terrain until it reach the layer height. This way the final terrain height should be equal to the layer height.



*Terrain heightmap*

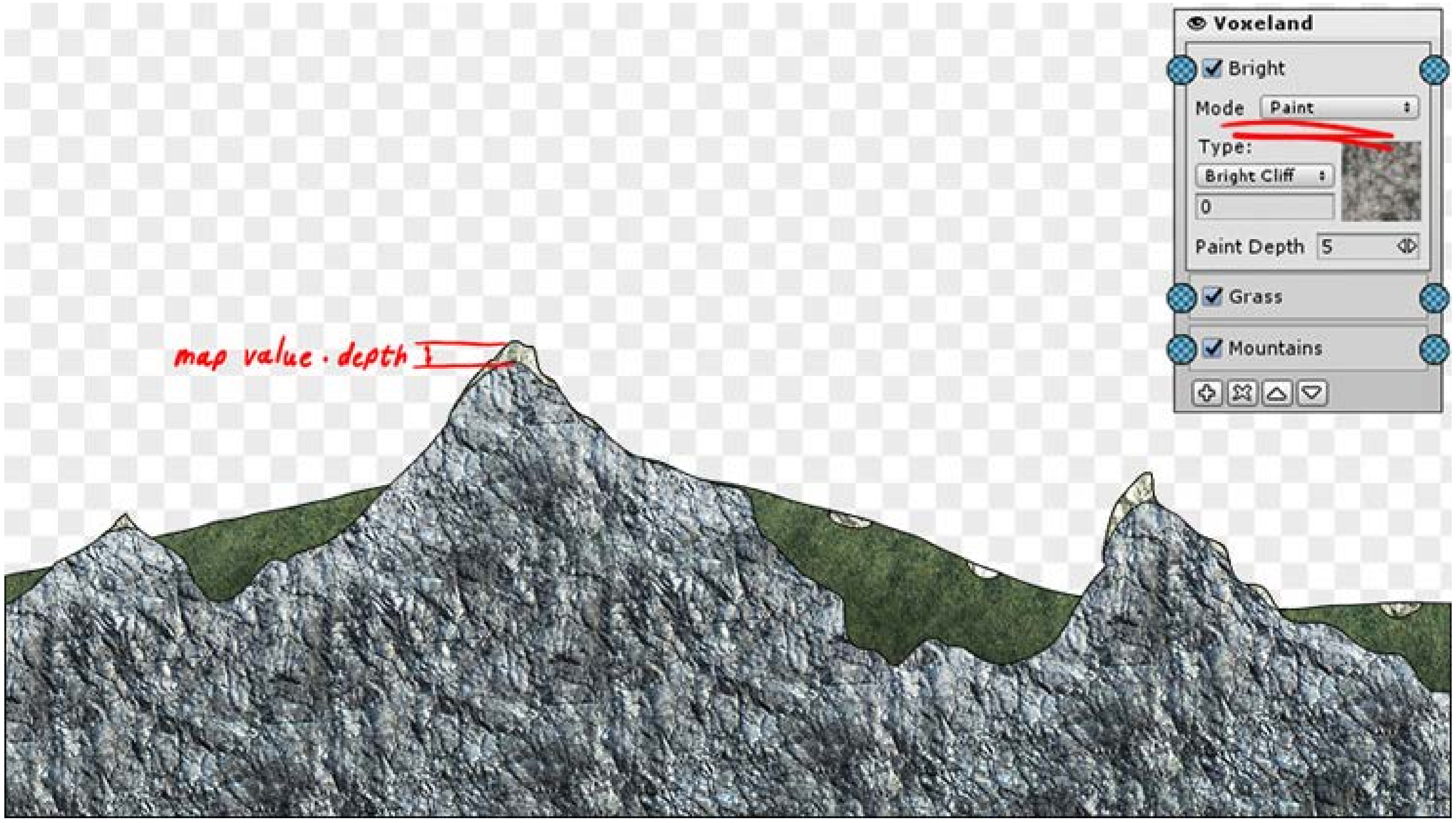


*Layer heightmap*



*Clamp/Append effect*

- **Paint**: "paints" with the blocks on the terrain. The depth of the paint effect is determined by the "Paint Depth" value multiplied by layer value. When the layer value is 0 no paint effect is applied, when the layer value is 1 paint effect is equal to Paint Depth value.

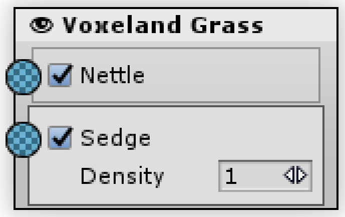


*Paint effect*

> You can stack several layers with the same block type.

Note that the **existing MapMagic maps** could hardly be converted to Voxeland terrain. Textures Output cannot be just replaced with Voxeland Output node. Voxeland Output uses absolutely different (and more complex) algorithm, it does not just blends layers in a Photoshop manner, but it creates layers in 3D space. This requires not only the significant graph changes, but changing the whole approach to the graph creation. That's why I'm not converting Island or Demo scenes to be used with Voxeland - they will require the whole graph created from scratch, and it will not have much in common with the original one.

# Voxeland grass

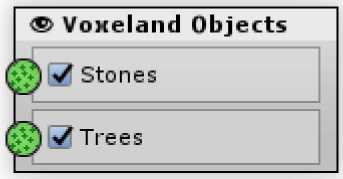


Outputs Voxeland terrain grass. Requires Voxeland to be installed. This generator replaces the standard Grass output.

> **Voxeland Output works only when MapMagic is is used as a Voxeland's generator. In order to use it you have to create Voxeland terrain, switch Generate Type to MapMagic, and create a new graph. See Voxeland+MapMagic tutorial for details.**

Use Voxeland "Grass Blocks" settings to add or remove layers or change their order.

## Voxeland objects



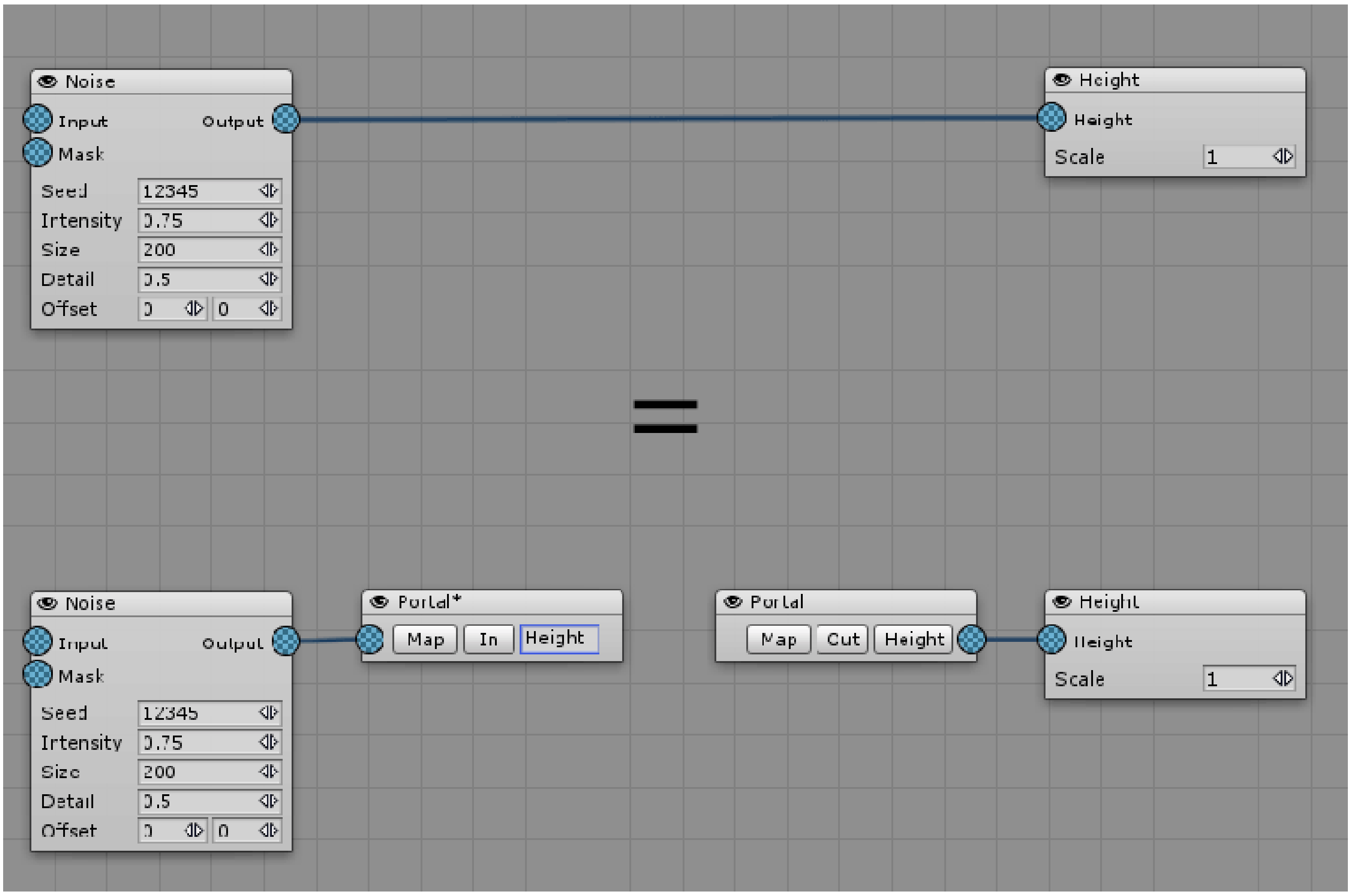
Outputs Voxeland terrain objects. Requires Voxeland to be installed. This generator replaces the standard Objects output.

> **Voxeland Output works only when MapMagic is is used as a Voxeland's generator. In order to use it you have to create Voxeland terrain, switch Generate Type to MapMagic, and create a new graph. See Voxeland+MapMagic tutorial for details.**

Use Voxeland "Object Blocks" settings to add or remove layers or change their order.

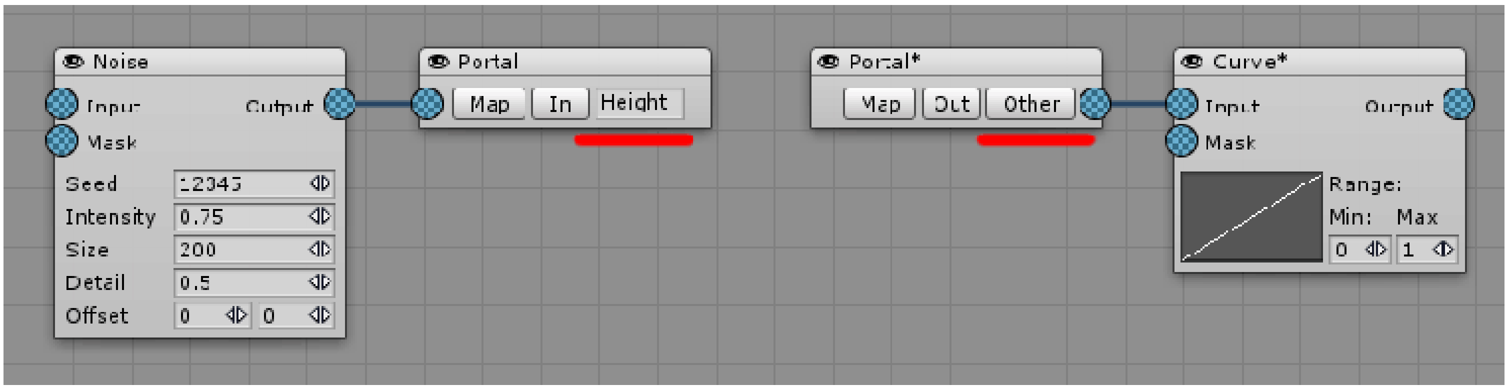
## Portal

Portals are the special generators used to organize the graph in a more convenient way. Sometimes it is necessary to connect some inputs and outputs on opposite sides of the graph. It is not easy to connect generators using the furthest zoom and makes it hard to read the graph. Portals are the mean to solve the problem. One portal (an input form) has an Input connection and the other portal (an output form) has an Output connection. On generate the input portal just passes the object to the output one, connecting the two generators together. There is no functional difference between a portal connection and the standard one.

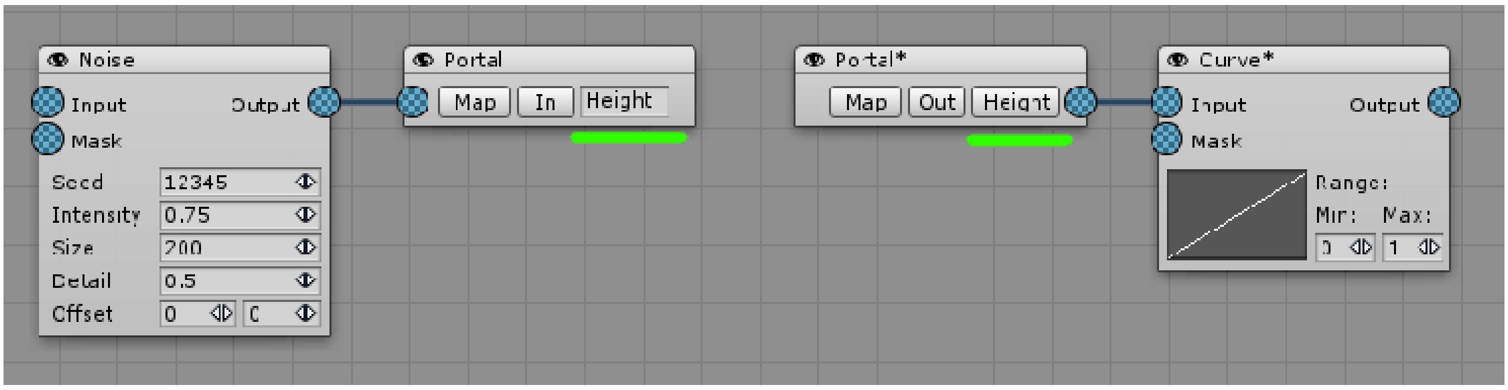


*Think of portal connection as a standard connection with no line displayed.*

To switch between Input Portal and Output portal use the In/Out button in the middle. To change the portal In/Out type (Map or Object) use the Map/Obj button to the left. One more thing that's important to know about portals in order to use them is the portal article. Only those portals that are linked share the same article - in the example above, the article is called "Height". So to link the portals you've got to type portal article in an Input Portal and then select a typed name in an Output Portal. An article could be any name you like and which describes portal map the best in your opinion.

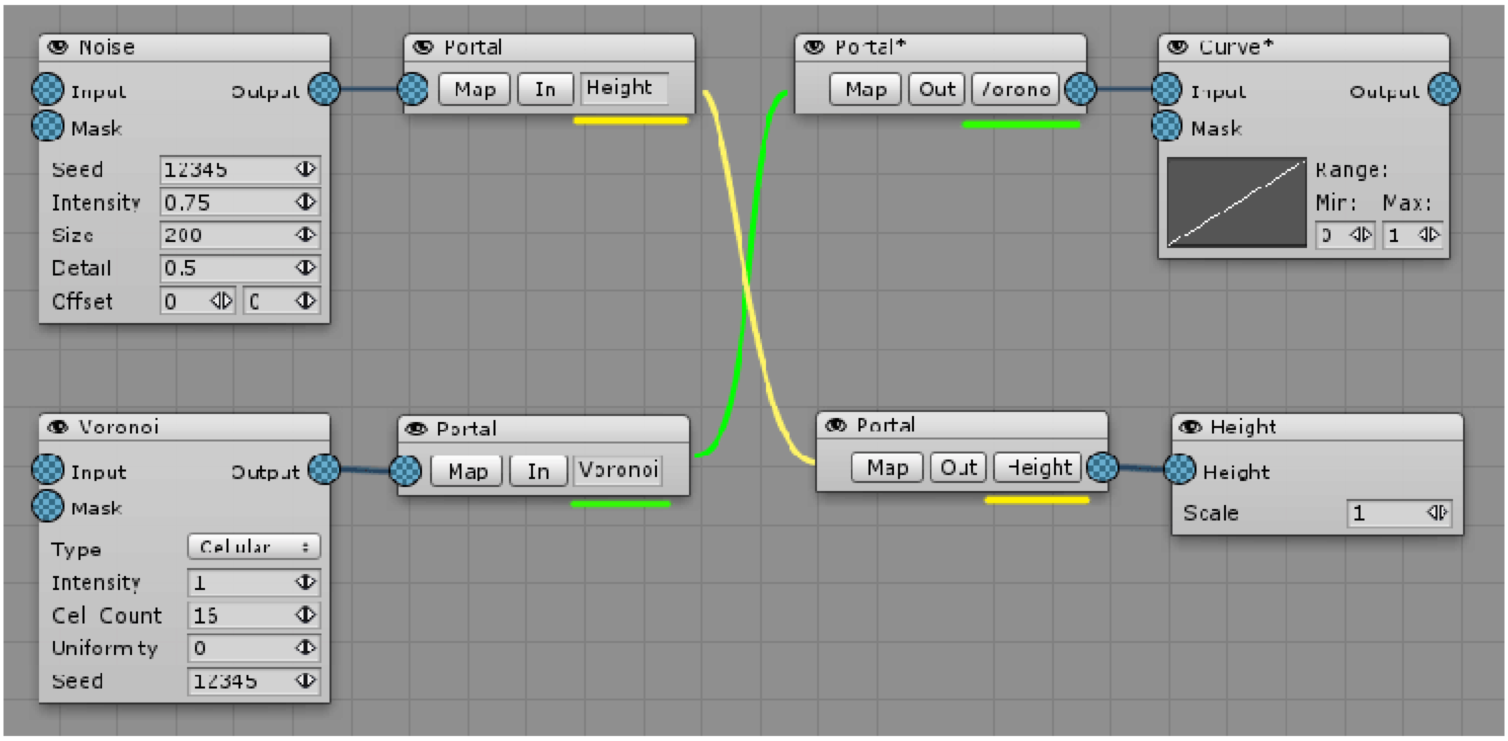


*Portals are not linked because of an article mismatch*



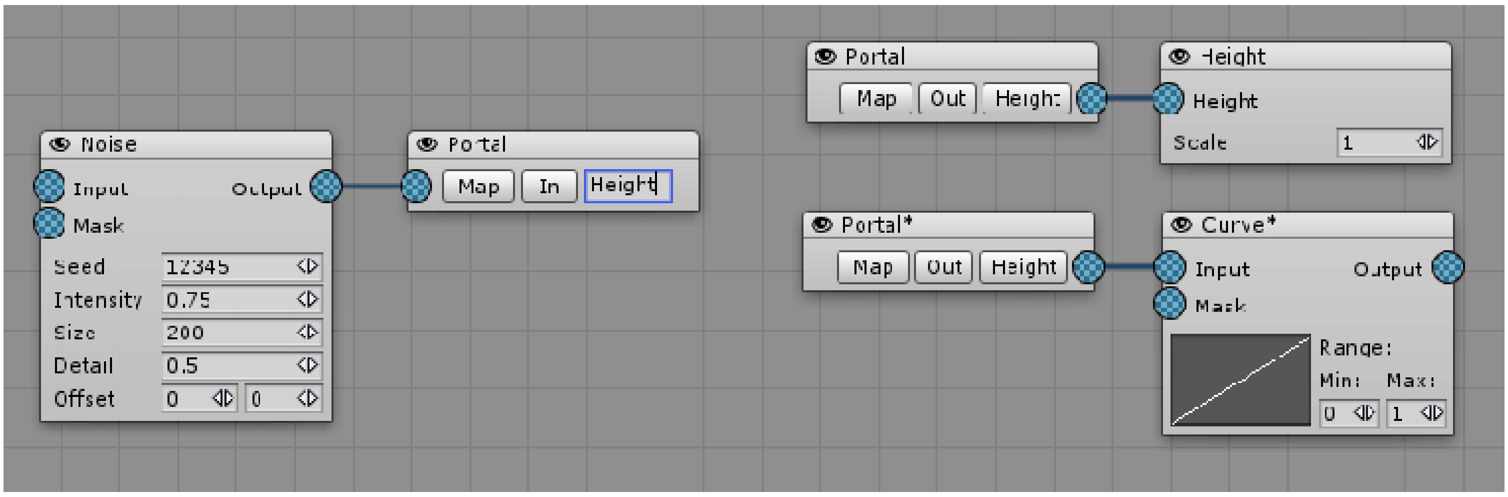
*Portals are linked because they share the same article*

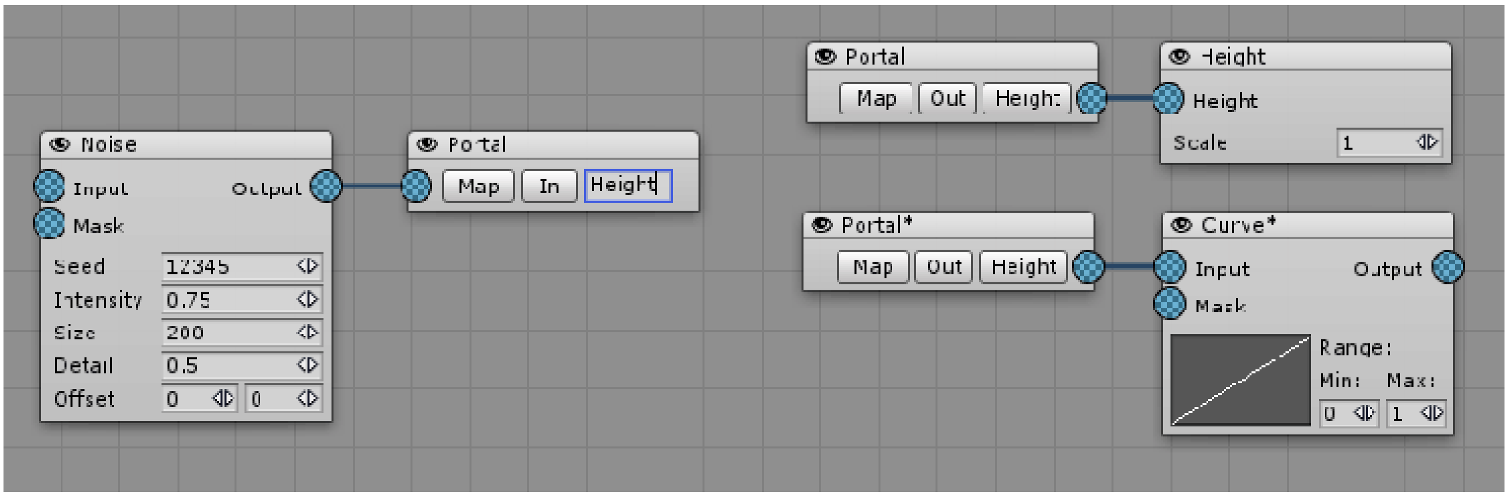
In a complex graph you would like to use portals with a different articles. Keep in mind that only those portals that share the same name are connected.



*Portals are cross-connected using the article principle*

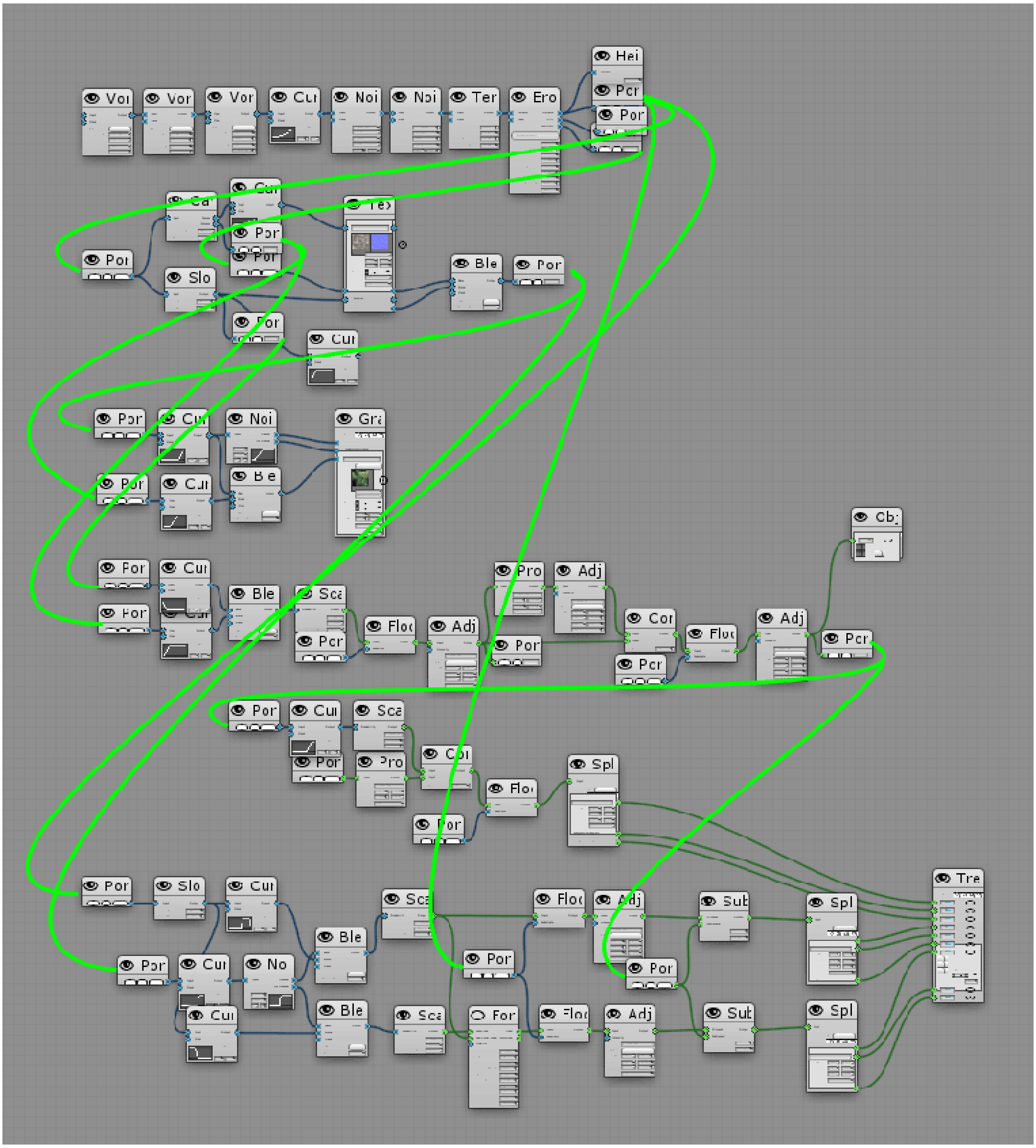
Multiple portal connections are also possible. To create them, use one Input Portal and multiple Output Portals. More complex combinations are also possible.





*Possible portals connections*

One might ask: "Why use these portals are if they could be replaced with direct connections?" It is even more confusing when using a simple graph when learning Map Magic. But I'd like to show a portal connection from a rather complex graph:



*Demo scene portal system*

So portals should not be used with every connection and not even on every fourth connection but they could be used from time-to-time to make a graph a bit more pretty and readable. Most likely even a complex graph will not have more than a dozen different portal articles.

## Properties

- Map/Object: portal input & output. Change the default "Map" it if you are going to create an objects portal.
- In/Out: portal type. In-portal have the input, but does not have output. Out-portal is the one that has the output, but

don't have input.

- Button: shows portal connections as lines (for graph debug purpose).
- Focus Button (in output portal): focuses editor window on the linked input portal.
- Name field: type portal name (for input portal) or select a link from the list (for output portal).

# Creating a custom generator

MapMagic can be extended with custom generators. All generators are made as modules, so a custom generator can be created without changing MapMagic’s code and without the need to re-write it on every MapMagic upgrade.

A generator template can be found in the SampleGenerator.cs file. You can copy this file and store it anywhere in your project. But for a better understanding an example of gradual generator creating will be given.

First of all let’s start with an empty class. All the generators derive from a base Generator class. It should be marked with a Serializable attribute and have a GeneratorMenu attributes used to create the generator with a right-click menu:

```
[System.Serializable]
[GeneratorMenu (
    menu="MyGenerators",
    name ="MyGenerator",
    disengageable = true,
    helpLink = "http://myGeneratorHelp" ) ]
public class MyCustomGenerator : Generator
{

}
```

GeneratorMenu attribute has the following arguments:

- menu: a base menu where Generator will be stored. It could be "Map", "Objects", "Outputs", or something custom. If you don't want to put your generator into a category (make it like Portal or Biome) leave it blank, but not null: name = "".
- name: the way Generator name is displayed in right-click menu.
- disengageble: the generator could be turned off with an 👁 eye icon next to the generator name at the graph.
- helpLink: generator help or usage description on the web. This link is opened when clicking ? help button next to the generator name at the graph.

Custom generators should override two abstract functions: Generate and OnGUI.

```
[System.Serializable]
[GeneratorMenu (menu="MyGenerators", name ="MyGenerator", disengageable = true)]
public class MyCustomGenerator : Generator
{
    public override void Generate (
        CoordRect rect,
        Chunk.Results results,
        Chunk.Size terrainSize,
        int seed,
        Func stop = null)
    { }
    public override void OnGUI public override void OnGUI (GeneratorsAsset gens)
    { }
}
```

Generate function accepts the following arguments:

- CoordRect rect: current terrain coordinates. In map space - measured in map pixels rather then world units. So, for the second chunk with a map resolution 512 it will be 512-1024 even if the chunk size is 1000 world units.
    - Coord Offset: rect offset, x and z, ints.
    - Coord Size: rect dimensions, x and z, ints.

- Chunk.Results results: a class to store the generated results, contains generate products of all of the generator in the chunk, not only the current one.
- Chunk.Size terrainSize: all the necessary information about a chunk size that is set in General Settings:
    - int resolution: currently used map resolution
    - float dimensions: current terrain size in world units
    - float height: maximum terrain height in world units
    - float pixelSize, get: returns the size of the pixel in world units (i.e. dimensions/resolution). Useful for converting from map space to world space.

- int seed: a global seed value used in current generate
- Func stop: stop callback to check if generate is canceled from the main thread (it could be canceled if the new generate was started when constantly changing node value). If stop(val) returns true then generator should cancel generate and return as soon as possible. In further versions float val will be used to get current generate progress, but in MM 1.8 it's not used.

OnGUI method take the GeneratorsAsset (i.e. graph) it belongs to as an argument.

Now let's add the generator's inputs and outputs. To work properly, all the generator's inputs and outputs should:

- be properly initialized
- included in the inputs or outputs enumerator
- have a graphical representation in the generator's OnGUI function

So let's add the Input and Output variables. Inputs and outputs constructors generally take two arguments: the first is the name of the input/output as it is displayed in the node and the second is its type in the form of InoutType enum (InoutType.Map or InoutType.Objects). In addition, Input can have an optional "mandatory" property: if set to true it will display a red mark for unconnected input. Warning: remove MyGenerator from a graph before applying the following code and create it again when scripts are compiled. Otherwise it will have a null input/output.

```
[System.Serializable]
[GeneratorMenu (menu="MyGenerators", name ="MyGenerator", disengageable = true)]
public class MyCustomGenerator : Generator
{
    //input and output vars
    public Input input = new Input("Input", InoutType.Map, mandatory:true);
    public Output output = new Output("Output", InoutType.Map);

    public override void Generate public override void Generate (CoordRect rect, Chunk.Resul
ts results, Chunk.Size terrainSize, int seed, Func stop = null) { }
    public override void OnGUI (GeneratorsAsset gens) { }
}
```

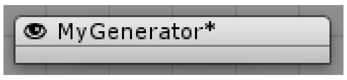
On generating MapMagic iterates a generator's inputs and outputs to process each of them. Enumerables are used to do this, just adding the variables will not make them accessible to MapMagic. Each input should be included with "yield return" keyword in an overridden Inputs enumerable, while each output should be included in an overridden outputs enumerable.

```
[System.Serializable]
[GeneratorMenu (menu="MyGenerators", name ="MyGenerator", disengageable = true)]
public class MyCustomGenerator : Generator
{
    //input and output vars
    public Input input = new Input("Input", InoutType.Map, mandatory:true);
    public Output output = new Output("Output", InoutType.Map);

    //including in enumerator
    public override IEnumerable Inputs() { yield return input; }
    public override IEnumerable Outputs() { yield return output; }

    public override void Generate public override void Generate (CoordRect rect, Chunk.Resul
ts results, Chunk.Size terrainSize, int seed, Func stop = null) { }
    public override void OnGUI (GeneratorsAsset gens) { }
}
```

If the generator has, let's say, three inputs and two outputs it should start with something like this:

```
[System.Serializable]
[GeneratorMenu (menu="MyGenerators", name ="MyGenerator", disengageable = true)]
public class MyCustomGenerator : Generator
{
    public Input firstIn = new Input("First", InoutType.Map);
    public Input secondIn = new Input("Second", InoutType.Map);
    public Input thirdIn = new Input("Third", InoutType.Map);
    public Output firstOut = new Output("First", InoutType.Map);
    public Output secondOut = new Output("Second", InoutType.Map);

    public override IEnumerable Inputs()
        { yield return firstIn; yield return secondIn; yield return thirdIn; }
    public override IEnumerable Outputs()
        { yield return firstOut; yield return secondOut; }

    public override void Generate public override void Generate (CoordRect rect, Chunk.Resul
ts results, Chunk.Size terrainSize, int seed, Func stop = null) { }
    public override void OnGUI (GeneratorsAsset gens) { }
}
```

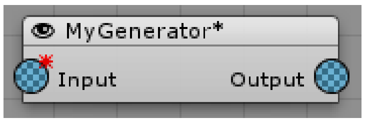
Now we have a correct and properly set-up generator. It has only two drawbacks: it does nothing and it has no displayed parameters in the GUI - not even input and output connection nodes.



We will fix the last one by exposing the input and output in the OnGUI function. This function is called when the generator node is rendered on the graph and uses a Layout wrapper, instead of UnityEditor.EditorGUILayout, which allows element scrolling, zooming and dragging. Right now we will not dig into Layout, we will just add a new line with a 20-pixel height(with zoom=100%) layout.Par function and then draw the input and output (a circle with an icon and a label with input/output name) at that line by calling their own OnGUI functions.

```
[System.Serializable]
[GeneratorMenu (menu="MyGenerators", name ="MyGenerator", disengageable = true)]
public class MyCustomGenerator : Generator
{
    //input and output vars
    public Input input = new Input("Input", InoutType.Map, mandatory:true);
    public Output output = new Output("Output", InoutType.Map);

    //including in enumerator
    public override IEnumerable Inputs() { yield return input; }
    public override IEnumerable Outputs() { yield return output; }

    public override void Generate public override void Generate (CoordRect rect, Chunk.Resul
ts results, Chunk.Size terrainSize, int seed, Func stop = null) { }

    public override void OnGUI (GeneratorsAsset gens)
    {
        layout.Par(20);
        input.DrawIcon(layout);
        output.DrawIcon(layout);
    }
}
```

Now you can see that MyGenerator has two map connections, and the input connection is displayed as mandatory.



Our 3 inputs and 2 outputs generator will have 3 lines, one input and one (or none) output per line:

```
    public override void OnGUI (GeneratorsAsset gens)
    {
        layout.Par(20); firstIn.DrawIcon(layout); firstOut.DrawIcon(layout);
        layout.Par(20); secondIn.DrawIcon(layout); secondOut.DrawIcon(layout);
        layout.Par(20); thirdIn.DrawIcon(layout);
    }
```

Now let’s add a generate function. Let’s say it will do a simple function - it will invert an input map’s value(1-value). There's already a generator for that, and the same effect could be achieved with curves but as an example it'll do. Here is the body of the Generate function. This function overrides the base Generator’s function.

```
[System.Serializable]
[GeneratorMenu (menu="MyGenerators", name ="MyGenerator", disengageable = true)]
public class MyCustomGenerator : Generator
{
    //input and output properties
    public Input input = new Input("Input", InoutType.Map, mandatory:true);
    public Output output = new Output("Output", InoutType.Map);
    //including in enumerator
    public override IEnumerable Inputs() { yield return input; }
    public override IEnumerable Outputs() { yield return output; }

    public override void Generate (CoordRect rect, Chunk.Results results, Chunk.Size terrain
Size, int seed, Func stop = null) { }

    public override void OnGUI (GeneratorsAsset gens)
    {
     layout.Par(20); input.DrawIcon(layout); output.DrawIcon(layout);
    }
}
```

Now we will load inputs using the input.GetObject function. This function returns an object type, so we have to cast it to the map's type, which is internally called "Matrix". This matrix will be called src (source).

```
[System.Serializable]
[GeneratorMenu (menu="MyGenerators", name ="MyGenerator", disengageable = true)]
public class MyCustomGenerator : Generator
{
    //input and output properties
    public Input input = new Input("Input", InoutType.Map, mandatory:true);
    public Output output = new Output("Output", InoutType.Map);

    //including in enumerator
    public override IEnumerable Inputs() { yield return input; }
    public override IEnumerable Outputs() { yield return output; }

    public override void Generate (CoordRect rect, Chunk.Results results, Chunk.Size terrain
Size, int seed, Func stop = null)
    {
       Matrix src = (Matrix)input.GetObject(chunk);
    }

    public override void OnGUI (GeneratorsAsset gens)
    {
     layout.Par(20); input.DrawIcon(layout); output.DrawIcon(layout);
    }
}
```

Now let's add two guard clauses to the Generate function. They will exit the function before executing its body to prevent errors or to save computing time. The first one occurs in the case where an input gets a null object - if it is not connected or the previous generator returned null. In this case this generator should not set any objects. The second one will stop the generate using the "stop" callback. Due to the multithreaded approach it can change its value at any time, and often it does - during the generate function execution from the main thread. It is recommended to check if the generate has not been stopped from time to time in your generator code.

```
[System.Serializable]
[GeneratorMenu (menu="MyGenerators", name ="MyGenerator", disengageable = true)]
public class MyCustomGenerator : Generator
{
    //input and output properties
    public Input input = new Input("Input", InoutType.Map, mandatory:true);
    public Output output = new Output("Output", InoutType.Map);

    //including in enumerator
    public override IEnumerable Inputs() { yield return input; }
    public override IEnumerable Outputs() { yield return output; }

    public override void Generate (CoordRect rect, Chunk.Results results, Chunk.Size terrain
Size, int seed, Func stop = null)
    {
        Matrix src = (Matrix)input.GetObject(chunk);

        if (stop!=null && stop(0)) return;
    }

    public override void OnGUI (GeneratorsAsset gens)
    {
     layout.Par(20); input.DrawIcon(layout); output.DrawIcon(layout);
    }
}
```

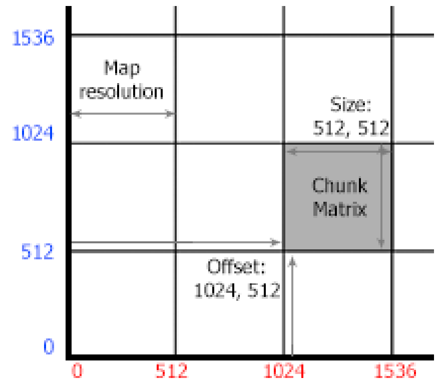
The other way the function can exit early is if the generator is not enabled. But in this case it should pass the input object to the output. So it uses output.SetObject function and returns.

```
[System.Serializable]
[GeneratorMenu (menu="MyGenerators", name ="MyGenerator", disengageable = true)]
public class MyCustomGenerator : Generator
{
    //input and output properties
    public Input input = new Input("Input", InoutType.Map, mandatory:true);
    public Output output = new Output("Output", InoutType.Map);

    //including in enumerator
    public override IEnumerable Inputs() { yield return input; }
    public override IEnumerable Outputs() { yield return output; }

    public override void Generate (CoordRect rect, Chunk.Results results, Chunk.Size terrain
Size, int seed, Func stop = null)
    {
        Matrix src = (Matrix)input.GetObject(chunk);

        if (stop!=null && stop(0)) return;
        if (!enabled) { output.SetObject(chunk, src); return; }
    }

    public override void OnGUI (GeneratorsAsset gens)
    {
     layout.Par(20); input.DrawIcon(layout); output.DrawIcon(layout);
    }
}
```

We're now at the most interesting part. We will do some simple calculations to invert a map. But before doing that let's make sure that it will not modify our original "src" matrix. This matrix could be used by other generators and, more importantly, it will be used by this generator each time it's parameter changes if "save intermediate results" is checked in the MapMagic settings. So do not forget this simple rule: do not change the input maps or object hashes, use new matrices or hashes to write values. So we will create a new matrix to write our changed src values and call it dst (destination). This matrix should have the same size and the same coordinates as the source matrix. The size and offset information is kept in a matrix.rect struct. We will use src.rect in a new matrix constructor.

```
[System.Serializable]
[GeneratorMenu (menu="MyGenerators", name ="MyGenerator", disengageable = true)]
public class MyCustomGenerator : Generator
{
    //input and output properties
    public Input input = new Input("Input", InoutType.Map, mandatory:true);
    public Output output = new Output("Output", InoutType.Map);

    //including in enumerator
    public override IEnumerable Inputs() { yield return input; }
    public override IEnumerable Outputs() { yield return output; }

    public override void Generate (MapMagic.MapMagic.Chunk chunk)
    {
        Matrix src = (Matrix)input.GetObject(chunk);

        if (stop!=null && stop(0)) return;
        if (!enabled) { output.SetObject(chunk, src); return; }

        Matrix dst = new Matrix(src.rect);
    }

    public override void OnGUI (CoordRect rect, Chunk.Results results, Chunk.Size terrainSiz
e, int seed, Func stop = null)
    {
     layout.Par(20); input.DrawIcon(layout); output.DrawIcon(layout);
    }
}
```

In order to invert the matrix we have to iterate all the src pixels and set them to dst = 1 - src. It's very simple to do this using matrices. Matrix is a wrapper of a standard single-dimensional array of floats, with an interface that of a 2D array but makes it a bit faster. And it takes matrix offset into account - so that matrix coordinates start not from zero, but from the offset values.



For example, if we want to get some pixel at a coordinates, let's say [768,256] we will call matrix[768,256]. If we want to iterate a matrix that has an offset [1024,0] and a size [512,512] we should use the following code:

```
for (int x=1024; x<1024+512; x++)
   for (int y=0; y<0+512; y++)
      float val = matrix[x,y];
```

By the way, since the matrix is used for 3D objects like terrain, and it is applied horizontally, it’s better to use the Z-axis name instead of Y(which usually indicates height).

So, the simple invert generator code should look like this:

```
[System.Serializable]
[GeneratorMenu (menu="MyGenerators", name ="MyGenerator", disengageable = true)]
public class MyCustomGenerator : Generator
{
    //input and output properties
    public Input input = new Input("Input", InoutType.Map, mandatory:true);
    public Output output = new Output("Output", InoutType.Map);

    //including in enumerator
    public override IEnumerable Inputs() { yield return input; }
    public override IEnumerable Outputs() { yield return output; }

    public override void Generate (CoordRect rect, Chunk.Results results, Chunk.Size terrain
Size, int seed, Func stop = null)
    {
        Matrix src = (Matrix)input.GetObject(chunk);

        if (stop!=null && stop(0)) return;
        if (!enabled) { output.SetObject(chunk, src); return; }

        Matrix dst = new Matrix(src.rect);
        for (int x=src.rect.offset.x; x< src.rect.offset.x+src.rect.size.x; x++)
           for (int z=src.rect.offset.z; z< src.rect.offset.z+src.rect.size.z; z++)
               dst[x,z] = 1 - src[x,z];
    }

    public override void OnGUI (GeneratorsAsset gens)
    {
     layout.Par(20); input.DrawIcon(layout); output.DrawIcon(layout);
    }
}
```

We can speed up the code a bit to avoid summing of rect with offset using the Coord struct. Think of Coord like a Vector2, except that it uses ints instead of floats.

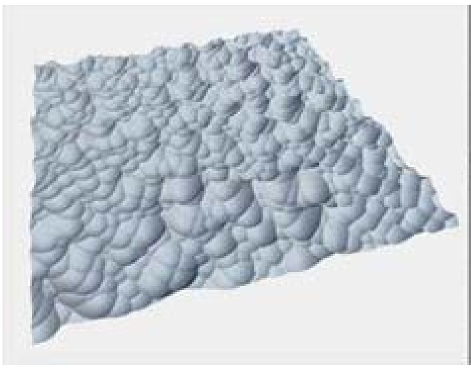
```
Coord min = src.rect.Min; Coord max = src.rect.Max;
for (int x=min.x; x< max.x; x++)
   for (int z=min.z; z< max.z; z++)
    dst[x,z] = 1 - src[x,z];
```

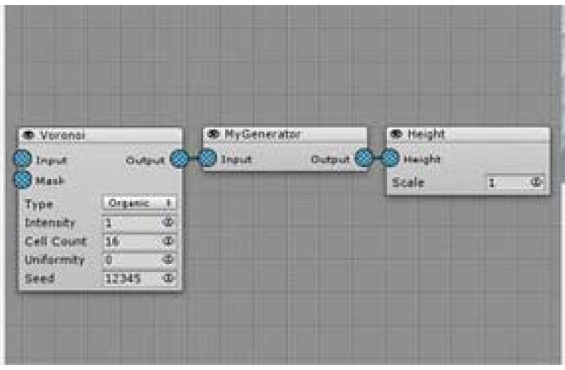
And the final step to make a generator work - it should store the generated result using the output.SetObject function. This function takes two arguments: the first one is a chunk to store an object and the second one is the result object itself.

But before storing an object check that the thread is not stopped. Just in case.

```
[System.Serializable]
[GeneratorMenu (menu="MyGenerators", name ="MyGenerator", disengageable = true)]
public class MyCustomGenerator : Generator
{
    //input and output properties
    public Input input = new Input("Input", InoutType.Map, mandatory:true);
    public Output output = new Output("Output", InoutType.Map);

    //including in enumerator
    public override IEnumerable Inputs() { yield return input; }
    public override IEnumerable Outputs() { yield return output; }

    public override void Generate (CoordRect rect, Chunk.Results results, Chunk.Size terrain
Size, int seed, Func stop = null)
    {
        Matrix src = (Matrix)input.GetObject(chunk);

        if (stop!=null && stop(0)) return;
        if (!enabled) { output.SetObject(chunk, src); return; }

        Matrix dst = new Matrix(src.rect);

        Coord min = src.rect.Min; Coord max = src.rect.Max;
        for (int x=min.x; x< max.x; x++)
           for (int z=min.z; z< max.z; z++)
           dst[x,z] = 1 - src[x,z];

         if (stop!=null && stop(0)) return;
         output.SetObject(chunk, dst);
    }

    public override void OnGUI (GeneratorsAsset gens)
    {
     layout.Par(20); input.DrawIcon(layout); output.DrawIcon(layout);
    }
}
```

Now if we create this generator and connect it to the organic Voronoi Generator you will see that it produces bubbles instead of caverns - so it works the way we expect.





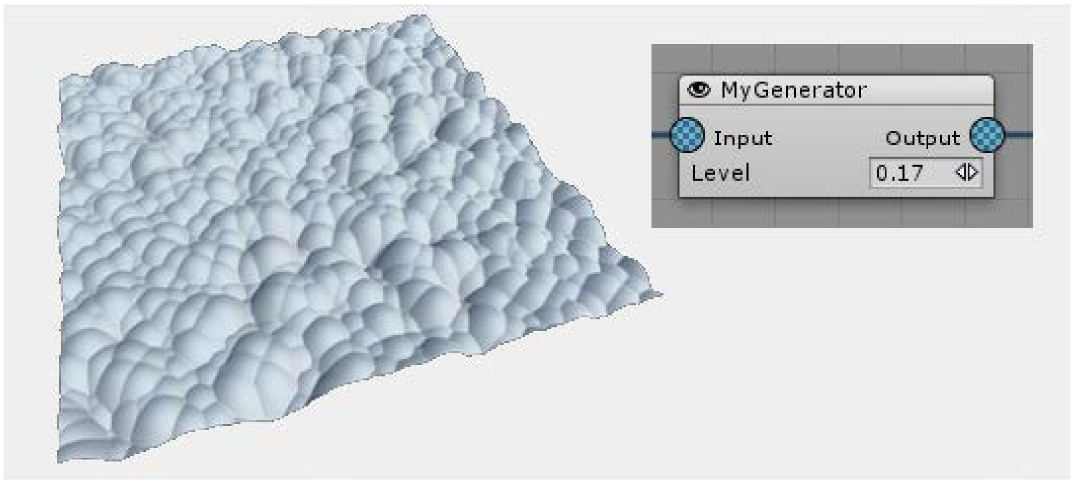
It has one drawback though - it creates a map somewhere at the top of the terrain which makes it hard to edit or blend the map with the other ones. This happens because we subtracted src values from 1 - the top terrain level. So if it was replaced with a custom value we will get a more adequate result. And note that we should monitor the dst value to prevent it from going below zero.

So let's add a "level" variable and expose it on the GUI using the layout.SmartField. A smart field is a common MapMagic field with a text input and a draggable icon to the right. The SmartField function takes two required arguments: the ref of a value we want to expose and the text that should be displayed in front of it. And we will use two additional parameters: minimum value (min:0) and maximum value (max:1) because, obviously, the level could not be less than zero or higher than 1.

By default there is no need to create a new line using layout.Par() because this function is already called in a smart field unless the newLine parameter is set to false.

```
[System.Serializable]
[GeneratorMenu (menu="MyGenerators", name ="MyGenerator", disengageable = true)]
public class MyCustomGenerator : Generator
{
    //input and output properties
    public Input input = new Input("Input", InoutType.Map, mandatory:true);
    public Output output = new Output("Output", InoutType.Map);

    //including in enumerator
    public override IEnumerable Inputs() { yield return input; }
    public override IEnumerable Outputs() { yield return output; }
    public float level = 1;
    public override void Generate (CoordRect rect, Chunk.Results results, Chunk.Size terrain
Size, int seed, Func stop = null)
    {
        Matrix src = (Matrix)input.GetObject(chunk);

        if (src==null || chunk.stop) return;
        if (!enabled) { output.SetObject(chunk, src); return; }

        Matrix dst = new Matrix(src.rect);

        Coord min = src.rect.Min; Coord max = src.rect.Max;
        for (int x=min.x; x< max.x; x++)
           for (int z=min.z; z< max.z; z++)
              float val = level - src[x,z];
        dst[x,z] = val>0? val : 0;

         if (chunk.stop) return;
         output.SetObject(chunk, dst);
    }

    public override void OnGUI (GeneratorsAsset gens)
    {
        layout.Par(20); input.DrawIcon(layout); output.DrawIcon(layout);
        layout.Field(ref level, "Level", min:0, max:1);
    }
}
```

Now we've got a nice looking generator, which has an input, output, and an adjustable parameter. It's also seamless: It appears as if it's been in MapMagic forever, but the really cool thing is that it is literally a plugin for MapMagic, we've done it without touching the main code. It could be removed by just deleting the file, sent to a colleague or published online.



# Scripting_faq

## I want to generate and return heightmap data at some custom area, is there any way to do it?

In some cases you might want to make MapMagic generate without actually creating terrains or using multithreading, just call pure generate function using some particular graph and terrain coordinates. There is a way to do it. Actually, it is the way Voxeland is using MapMagic to generate it's terrain. You will need to use graph's Calculate function for that.

```
Calculate (int offsetX, int offsetZ, int size, Chunk.Results results, Chunk.Size terrainSize, 
int seed)
where
- offsetX and offsetZ  is the offset of the area to generate (in map pixels)
- size  is the dimentions of the area
- results  store all of the generate data, including the heightmap. Top access heightmap Matri
x use results.heights value.
- terrainSize  is a struct that holds information about world parameters: terrain resolution, 
terrain dimentions (it's size in world units), and maximum height value
- seed  is a global seed used to generate unique terrain
```

Note that this function is called not on MapMagic object itself - you can even have no MapMagic in scene at all - but on the graph, a saved .asset file.

Here is a usage example:

```
MapMagic.GeneratorsAsset mapMagicGens = MapMagic.instance.gens; //finding current graph asset
MapMagic.Chunk.Results results = new MapMagic.Chunk.Results(); //preparing results
MapMagic.Chunk.Size size = new MapMagic.Chunk.Size(MapMagic.instance.resolution, MapMagic.inst
ance.terrainSize, MapMagic.instance.terrainHeight);
mapMagicGens.Calculate(100, 100, 300, results, size, 12345);
```

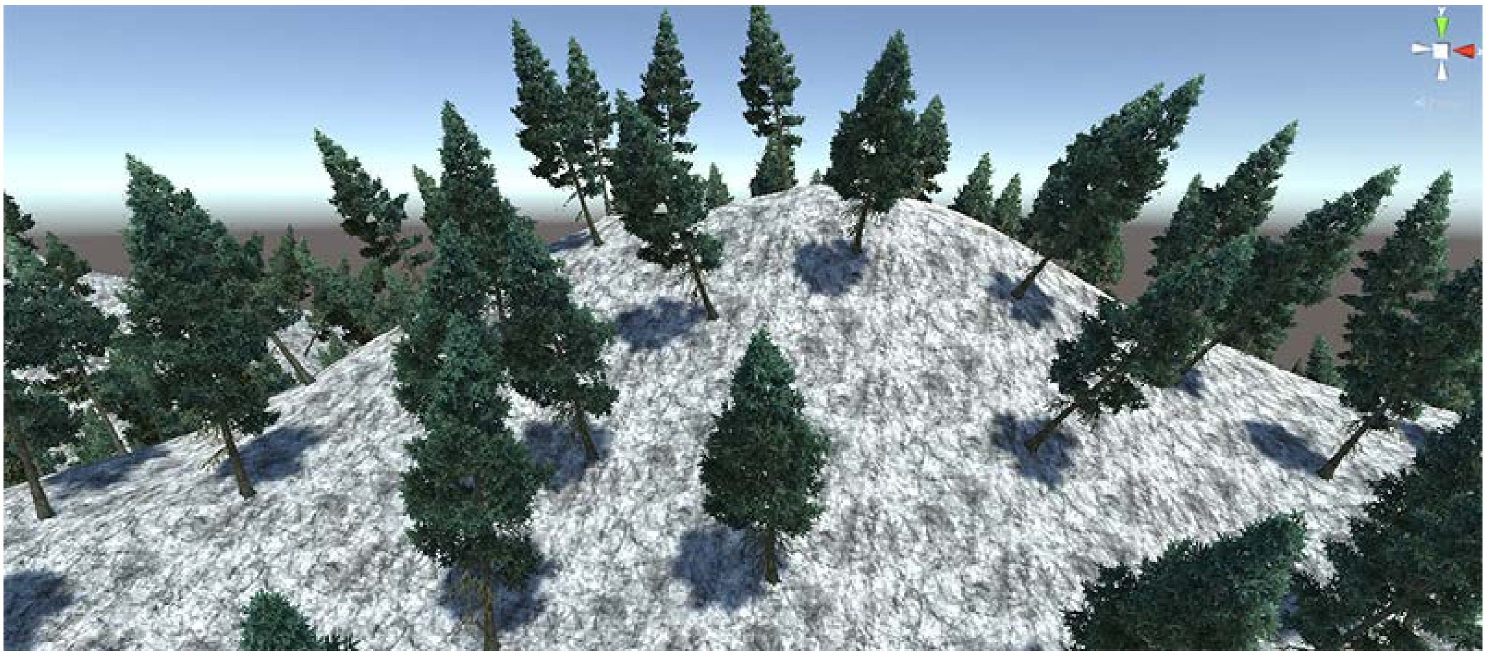
Here is how it used in Voxeland:

```
area.results = new MapMagic.Chunk.Results();
MapMagic.Chunk.Size size = new MapMagic.Chunk.Size(area.rect.size.x, area.rect.size.x, heightF
actor);
if (mapMagicGens != null)
    mapMagicGens.Calculate(area.rect.offset.x, area.rect.offset.z, area.rect.size.x, area.resu
lts, size, seed);
```

## How to post-process objects applied to terrain?

This example shows the way to incline objects along the terrains. Actually, objects could be inclined out of the box since v.1.8, this code just shows a practictical example of using OnApply event:

```
    using UnityEngine;

    namespace MapMagic
    {
        [ExecuteInEditMode]
        public class AlignToNormal : MonoBehaviour
        {
            public void OnEnable ()
            {
                MapMagic.OnApply -= Align; //just in case it was not called on disable
                MapMagic.OnApply += Align;
            }

            public void OnDisable ()
            {
                MapMagic.OnApply -= Align;
            }

            public void Align (Terrain terrain, object obj)
            {
                TransformPool.InstanceDraft[][] tfmTypeInstances = obj as TransformPool.InstanceDraft[][];
                if (tfmTypeInstances == null) return; //in case it is height/splat/grass/forest apply

                for (int t=0; t<tfmTypeInstances.Length; t++)
                {
                    TransformPool.InstanceDraft[] tfmInstances = tfmTypeInstances[t];

                    //there could be a check if an object of this type should be rotated

                    for (int i=0; i<tfmInstances.Length; i++)
                    {
                        TransformPool.InstanceDraft tfmInstance = tfmInstances[i];

                        Vector3 terrainNormal = terrain.terrainData.GetInterpolatedNormal(
                            tfmInstance.pos.x/terrain.terrainData.size.x,
                            tfmInstance.pos.z/terrain.terrainData.size.z);
                        Vector3 sideVector = Vector3.Cross(terrainNormal, tfmInstance.rotation*Vector3.forward);
                        Vector3 frontVector = Vector3.Cross(sideVector, terrainNormal);

                        tfmInstance.rotation = Quaternion.LookRotation(frontVector, terrainNormal);
                        tfmInstances[i] = tfmInstance; //saving struct to array
                    }
                }
            }
        }
    }
```



Add the script anywhere in scene - for instance, to Map Magic object. Actually you can do almost anything with such a scripts.

## How can you export terrain data?

You can export terrain data to asset file using this editor function:

```
        void ExportTerrainData (Terrain terrain)
        {
            if (!export) return;
            export = false;

            string path= UnityEditor.EditorUtility.SaveFilePanel(
                "Save Terrain as Unity Asset",
                "Assets",
                "ExportedTerrain.asset",
                "asset");
            if (path!=null && path.Length!=0)
            {
                path = path.Replace(Application.dataPath, "Assets");

                Terrain terrain = GetComponent<Terrain>();
                float[,,] splats = terrain.terrainData.GetAlphamaps(0,0,terrain.terrainData.al
phamapResolution, terrain.terrainData.alphamapResolution);

                UnityEditor.AssetDatabase.CreateAsset(terrain.terrainData, path);
                terrain.terrainData.SetAlphamaps(0,0,splats);
                UnityEditor.AssetDatabase.SaveAssets();
            }
        }
```

Note that somehow just exporting with CreateAsset removes splat map information, but this code has already a workaround for that. A ready script could be found here. Just assign it to the terrain object and press "export" - it works like a button.

## How to set random seed each time the game starts?

You can use mapMagic.ForceGenerate method - it will make terrain re-generate after seed change:

```
mm.seed = newSeed;
mm.ForceGenerate();
```

Or you can start with a disabled mapMagic object, and enable it once the seed is set in your script's Start method:

```
void Start()
{
mm.seed = newSeed;
mm.enabled=true;
}
```

Personally, I prefer the second approach since it works both in multithreading and non-multithreading mode: it does not start unneeded old-seed generate on scene start that will be erased with ForceGenerate.

If you are going to change only generator of one type (for example, the noise seed):

```
foreach (MapMagic.NoiseGenerator1 noiseGen in mapMagic.gens.GeneratorsOfType<MapMagic.NoiseGenerator1>())
```

Note that there should not be pinned terrains in the scene - since they have an initial seed value they will not match the new terrains created with the new seed.

## How to change the dynamic generator linking from the code?

https://forum.unity3d.com/threads/map-magic-world-generator-a-node-based-procedural-and-infinite-game-map-tool.390920/page-20#post-2859860

# General_faq

## Why do you use multithreading instead of GPU?

I'd like to mention that not everything that run on GPU is automatically fast. I've made some experiments with the compute shaders and found out that it could be accelerated only with a flexibility loss - for instance, have to give up using multiple output connections. The reason I prefer multithreading to GPU calculations is that it fits dynamically generated infinite terrain better. GPU generating usually looks great in the editor, but in playmode, when GPU load is usually 100% it can squeeze graphics calculations and produce a some lag. While in most cases the processor cores are not so loaded (maybe except the main unity thread), which allows to use them almost unnoticeable.

MapMagic uses multithreading instead, calculating all of the generators asynchronously in a several separate threads, and just applies final the result in a main thread. Calculating terrain is mostly fast (dozens milliseconds per generator per thread) except some generators like erosion, which can take up to several seconds. All the generating runs in a background and creates no lag in game.

## Is there any way to change standard MapMagic graphs to make them work with Voxeland?

Unfortunately, I don't believe it's possible. Textures Output cannot be just replaced with Voxeland Output node. Voxeland Output uses absolutely different (and more complex) algorithm, it does not just blends layers in a Photoshop manner, but it creates layers in 3D space. This requires not only the significant graph changes, but changing the whole approach to the graph creation. That's why I'm not converting Island or Demo scenes to be used with Voxeland - they will require the whole graph created from scratch, and it will not have much in common with the original one. You can find MM+Voxeland tutorial here, JIC: https://youtu.be/jlLALBsAUZY

## Is there any way for custom generator to get neighbour chunk information?

No. It's one of MMWG's concepts: chunks are independent. Nor they get any data from a neighbor chunk, neither they send it. Otherwise each chunk will require all of the neightbour chunks to generate, and they will require all of their neigbours to be generated, and so on up to infinity.

## Is there any way to export terrain as model?

Actually it's a subject for a whole new plugin, like this one: Terrain To Mesh (https://www.assetstore.unity3d.com/en/#!/content/47276). However, there is a way to export terrain into 3D editor without a plugin - since MM generates the standard terrains their heightmap could be exported using the "Export Raw" button in the terrain's settings tab. This heightmap could be could be applied to the plane with displace modifier. But terrain data could be exported as a standard Unity terrain using this script: https://www.dropbox.com/s/tnb3v1s6r5h1ss0/ExportTerrainData.cs?dl=0 Place it anywhere inside Assets folder, assign it to terrain and check "Export" (it works like a button).

## Is there any way to create villages using MM?

## How to remove objects below certain level?

## I've created my graph/biome/generator and want to share it, what is the best way to do it?

Feel free to append Community Graphs and Community Generators pages. You can also publish your screenshots on MMWG Unity forums thread.

## Where I can find the graphs shown in the biomes videos?

You can download badlands and snow plains graphs from the Community Grpahs page. Note that the textures are not included, I hasve no rights to share them. Sorry.

## Is there any way to use navmesh or backed pathfinding on a dynamic terrains?

It's better not to use anything baked on a dynamic terrain: not a lightmap nor a navmesh. A dynamic pathfinding should be used instead, using waypoint navigation, collision detection and steering behaviors. An open terrain is not a maze, so in most cases it will be enough. Apex Path boasts dynamic pathfinding, but I have not tested it. Some users claim that they are using Apex Path alongside MapMagic with great results. (end of page 7) However, for pinned terrains that can have some complex POI like villages or cities a manual navmesh could be created with the standard tools.

## How to use RTP 8-layers mode?

You need to switch RTP to 8-layer manually using _RTP_LODmanager: RTP features first pass -> 8 LAYERS in first pass.

## How to change the dynamic generator linking from the code?

https://forum.unity3d.com/threads/map-magic-world-generator-a-node-based-procedural-and-infinite-game-map-tool.390920/page-20#post-2859860

## What is the best way to restrict how far the player can go? My idea was to pin the last terrain and draw a mountain or a invisible wall, is this the way that you guys are working with it?

I think that the best way to create the invisible walls is to actually place invisible walls. It should be a collider mesh that should not be associated with MM in any way, just a separate game object that limits player movement. Or it could be a character controller script that does not allow character exceed some bounds. Let MM generate the terrains on the background, and set the map bounds by other means.

## I'm using custom terrain material with my own shader. Is there any way to make it work with MapMagic?

You can use Custom Shader Output (CSO) in "Custom"(default) mode. It uses per-channel generate of control maps using r,g,b and alpha inputs and assigns them to terrain material using the given property name.

# Problem_solving

## I have never been able to get that "preview" option working

Click the blue circle, not the generator itself. Note that MapMagic object has to be selected to display preview gizmos.

## I cannot pin a terrain. When I click "Pin" button nothing happens.

Try resetting Unity layout. Click a "Layout" button at the top right corner of the Unity window and select a new layout. You can switch back to the layout you like then.

## I'm getting "Lod "GROUP < 0X7FFF" errors" in a playmode. Why?

You've encountered one of the Unity limitations: there could not be more than 32K objects with LODs. It's not the matter of MapMagic in general, you have to use less loded objects or disable LODGroup components. Personally I got rid of this by disabling LODGroups for the objects that are far from the camera. I used a coroutine to process only several objects per frame to avoid hiccups.    It seems that Unity has fixed this issue in 5.6, or increased the number of possible LOD Groups.

## Trees are not scaled

## I'm using Forest generator with a grass mask. I have trees in the sea, but the grass preview shows that I shouldn’t have. Any idea?

Forest generator can create new trees, but it does not remove them - this also applies to initial seedlings. To remove them connect grass mask to scatter's probability input.

## Why do I have seams on the chunks borders?

Chunks are absolutely independent and do not exchange any data - for instance, when blurring one chunk it cannot get pixels from the neighbor one to perform a perfect blur. That's why some generators have "Safe Borders" value - it masks the generator effect closer to chunk edges. Many generators have this value set to 2 by the default, you can try increasing it to minimize the seam effect.

## I've got this warning when using custom grass detail mesh or high grass density without a mask, and the grass does not appear in some "stripped" areas:

```
The combined number of detail object vertices in one single patch is exceeding the limit (6
5k). Try decreasing detail density or detail resolution per patch.
UnityEditor.DockArea:OnGUI()
```

The combined grass mesh is too complicated: you have to split it. Decrease Patch Res value parameter in Grass Output. If it is already 8 try to simplify your grass mesh or reduce grass density.

## How to make uNature work with MapMagic?

Depending on MapMagic and UNature versions you can get a "type or namespace cannot be found" error in the console on scripts compile. In order to fix this use the quick fix: https://www.dropbox.com/s/crl0d9xppxmc6f4/UNMapMagic_Manager.cs?dl=0.

Store it in *uNature\Scripts\Core\Extensions\Integrations\MapMagic\Integration* replacing the older file.

1. Make sure that **UN_MapMagic** is defined in Scripting Define Symbols (Edit->Project Settings->Player). If it does not - simply restart Unity (or add it manually).
2. Click Window->uNature->Extensions and find Map Magic. Expand the arrow and make sure "Activated" is checked.
3. Open the scene with Map Magic and click Window->uNature->Foliage and then "Create Foliage Manager"
4. Click on Main Camera and add UNSeeker component
5. Set the Map Magic object / terrain to a different layer. For example, the uNature_Terrain layer (don't use Default layer)
6. Go into the Foliage Manager and find "Prototype Global Settings" change "Maps Generation Mask" to uNature_Terrain (or whatever layer you picked). It will automatically find the grass / foliage that was setup on the terrain and load them in "Prototypes Management" in the Foliage Manager"

After all of that in some cases the grass can appear at the zero level for pinned terrains. This could be fixed by clicking "Regenerate Heights" in Foliage Manager.

## Simple Form with a Stamp does not work on multiple terrains.

I guess that the problem is in the Simple Form wrap mode. Set it to "tile" to use on multiple terrains.

## I use CTS and Custom Shader Output. If I change texture in CTS Profile it is not updated on the terrain.

You should press "Bake Textures" button of CTS Profile. This is the way CTS works.