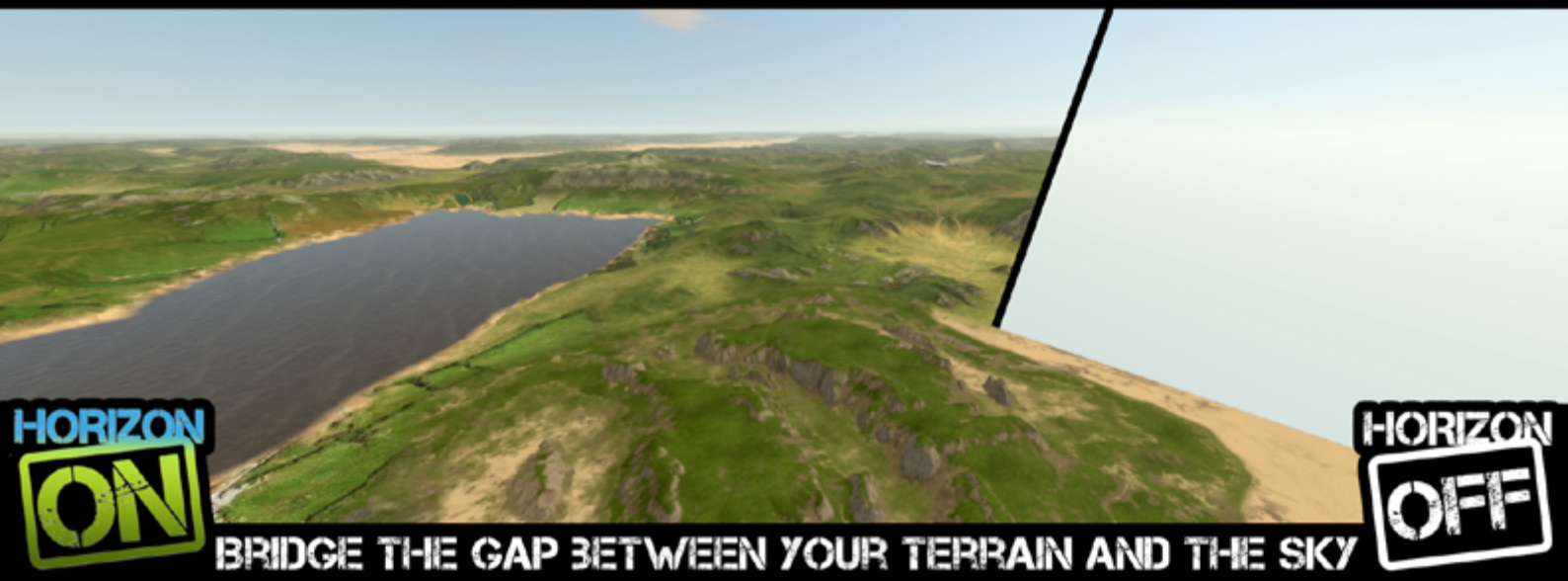
HORIZON[ON]



Manual v1.4

Welcome & thank you for purchasing Horizon[ON]

Horizon[ON] is an allround solution to make horizons look great, even on tiny terrains. Terrains usually have to be much bigger than the accessible area just to hide the square shape of your world, this is even worse if players can reach elevated positions. Also even with bigger terrains the horizon line still is harsh and unrealistic. With Horizon[ON] these problems are solved and you don't need to make terrains huge just to get a somewhat decent horizon. No more distance based blurring or other expensive image effects. Instead it offers you a simple but powerful way to blend your terrain with the skybox or any other background. It makes it easy to let your terrain appear much larger than it actually is and maximizes the walkable area of your terrain, so less of the costly unity terrain(or mesh terrain) is wasted just for background scenery.

We wish you fun and success using our product!

Nathaniel, Tom & Peter

p.s.: Please take some time and rate Horizon[ON] on the Unity Assetstore. We did our best to make our product stand out but your support is what makes it successful!

Features:

Main

- extend your terrain towards the horizon at a minimum cost
- smooth blending
- works for terrains of any size
- extremely adjustable & artist friendly workflow
- very low on draw calls
- mobile friendly

Shaders/Materials

- multi_compile shaders - use only the features you really need
- clever material management
- works with OpenGL, DX9, DX11,
- works in Linear and Gamma Space
- ambient override and tint to easily match your scenery
- 4 Layers for different terrain types on one horizon
- detail textures and normal maps
- emissive channels for distant city lights, volcanoes or else...
- water & ice (oceans, rivers, lakes...)
- height based fog
- vertical cliffs
- displacement
- tesselation(DX11 only)
- dynamic snow
- super fast trees
- super fast buildings
- spectacular mountains

Blendobjects

- easily add hills and cliffs to the horizon with a minimum of polygons
- add tesselated parts to the horizon for great detail
- all perfectly blended with the horizon object

Useable as a terrain solution(!)

(Note: its not meant to be a replacement for powerful terrain solutions like RTP)

- suitable very well for top-down games, flight simulators & strategy games.
- even in first person it can look great(depending on your needs of course)

Important Note!!!

Please import Horizon[ON] into an empty project and make yourself familiar with it before importing it into an existing project. Horizon[ON] uses shader keywords to enable and disable features. In unity 4.6 exists a limitation of 64 shader keywords and if you have already a lot of shaders in your project, it can be that upon import of Horizon[ON] you exceed this limit. If that is the case you will get an error, saying :

Maximum number (64) of shader keywords exceeded, ...

If that happens to you, don't panic, Horizon[ON] uses these keywords only to enable certain features, if you know already which features you are going to use you can easily define them in the shadercode,

from:
#pragma multi_compile _EXAMPLE_KEYWORD_OFF _EXAMPLE_KEYWORD_ON
to
#define _EXAMPLE_KEYWORD_ON

On the pages 19 and 20 of this manual you find a detailed explanation of why and how to deal with shader keywords. Please read this section thoroughly before you import Horizon[ON] into an existing project!!!

About Horizon[ON] in general:

There are a multitude of different workflows possible with Horizon[ON] and its difficult to foresee how you are going to use it. Horizon[ON] is designed in a way that it puts very little restrictions on your creativity. You could use it in miniature scale just as well as in large scale. Horizon[ON] is not even limited to terrains in the common sense, you could use it for an ant simulator just as well as for a flight simulator, you could use it to mimic the death star surface from starwars or as a background for a 2.5D sidescroller, you could even use it as a dungeon solution in a topdown game.

There wont be tips to do these things in this manual but there will be enough information to get you a deeper understanding of how Horizon[ON] works. You probably have to experiment a little if you want to adapt Horizon[ON] for such cases. I just want to tease your creativity a bit and suggest that you could find ways to use it for more than just a terrain extension if you think outside of the box. In case you have made something cool with Horizon[ON], i would be very happy to hear from you. Please come and visit the Horizon[ON] forum thread now and then and share your experience. Feedback is also very appreciated.

For now though, we will just focus on what Horizon[ON] is actually made for, to bridge the gap between your terrain and the sky!

First steps with Horizon[ON]:

Importing works as usual. After import and taking a look at the example scene, you can decide if you want to read the manual first or go more for the learning by doing approach...

The "Horizon[ON] - Getting Started Guide" will help you with your first steps. There are also some videos that show Horizon[ON] in action. This manual contains more detailed information, "best practices" and useful tips to get the most out of horizon in your project.

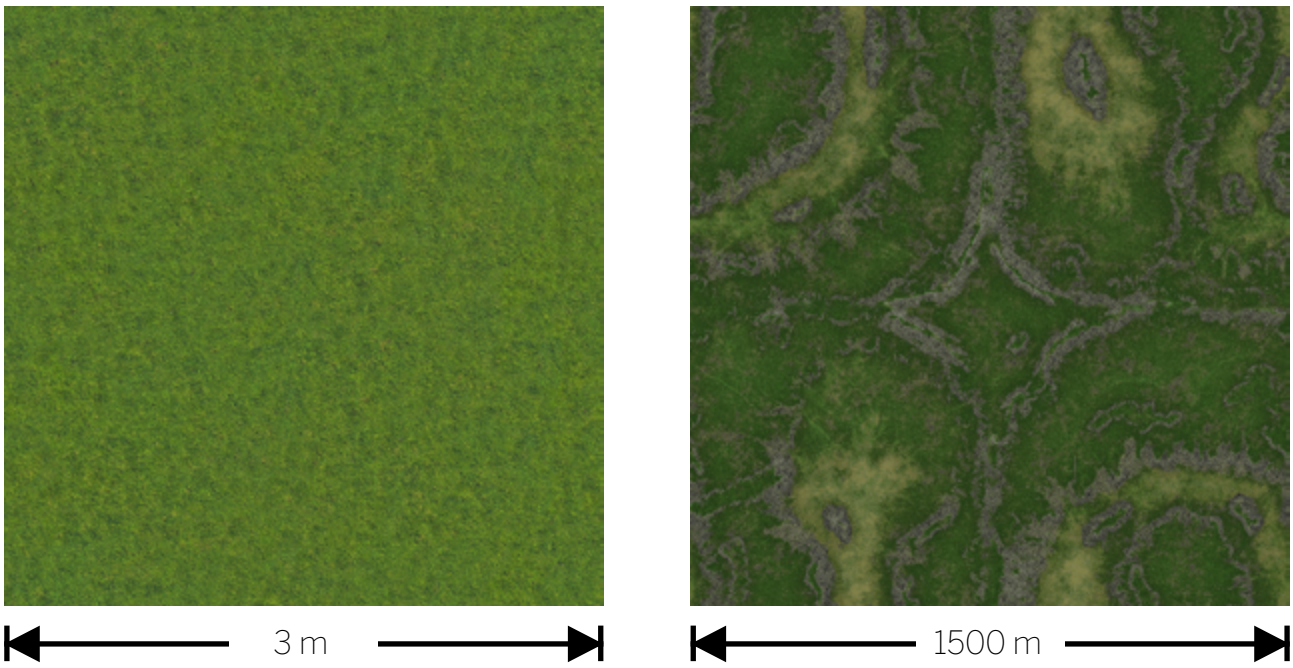
Learning an extensive new tool can be sometimes frustrating but i ensure you that learning how to use Horizon[ON] is worth it. A good starting point is the example scene together with the next few pages. Also remember that every button in the HorizonMaster(the main interface for your work) has tooltips.

So let's go and dive right into the bits and bolts of Horizon[ON].

Understanding the basic concept

Horizon[ON] works with layers just like a terrain shader, you can have up to 4 layers + water(which acts just like an extra layer). The main difference to a terrain shader is the scale of the textures in worldspace. A terrain shader usually has textures that range somewhere between 1-40 meters in size. A layer on a terrain shader for example could be a grass texture, of 3x3 meters size in worldspace. This means it is a tiling texture, after 3 meters it will repeat. Horizon[ON] works just like that but instead of grass it would be an image of a terrain region, like a desert, or a mountain range, or hills for example. Such a region does not contain only grass like in the terrain exaple, it would look more like a satelite image of such a region. It would have a much bigger size before it will repeat In the example below i used 1500 m, but of course it is up to you to chose the size of your layer.

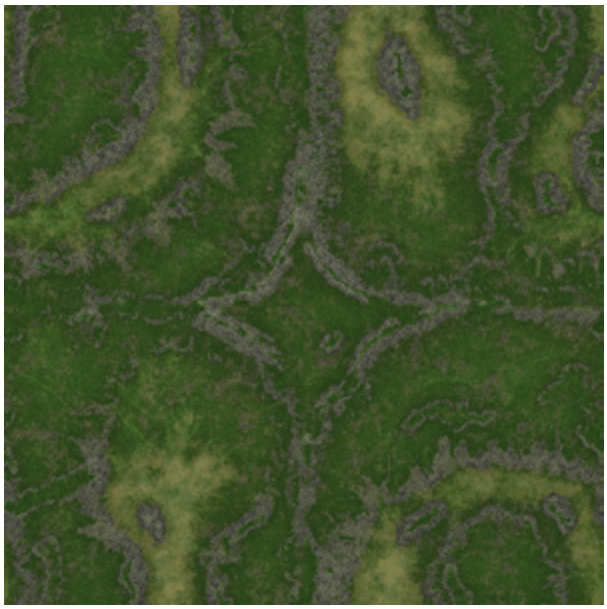
Below you see a comparison, left a typical grass texture as it would be used on a terrain shader, right a typical texture as it would be used with Horizon[ON].



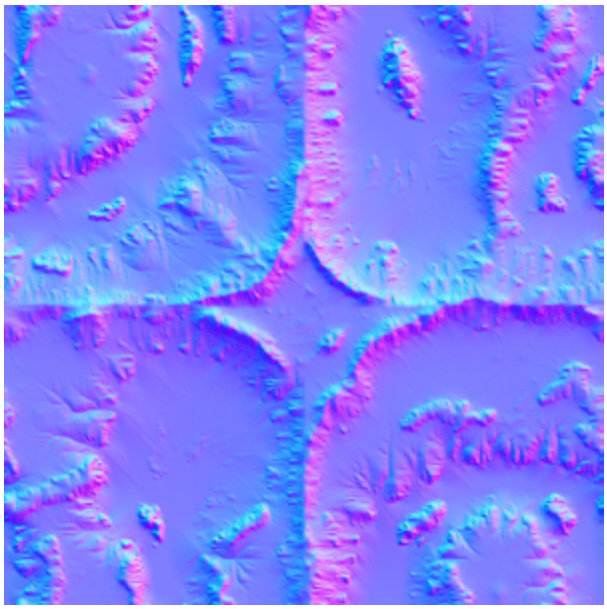
As you can see, the right texture spans a bigger area, and it shows more things than just grass, basically, it can contain all kind of terrain styles at once. Just like the left texture, the Horizon[ON] texture is a tiling texture, which means it can repeat without obvious seams.

Note:
If you want to know more about how to edit a texture to make it tileable, look online, there are hundreds of tutorials on this topic out there!

A layer of Horizon[ON] does not neccesarily only consist of a diffuse(or albedo) texture, it can also make use of a normal map (if the normal map feature is enabled). Usually you want your normal map to be roughly matching with your diffuse texture but even if they dont match you can get good results. Below you can see the normal map that i made to fit with the diffuse texture:

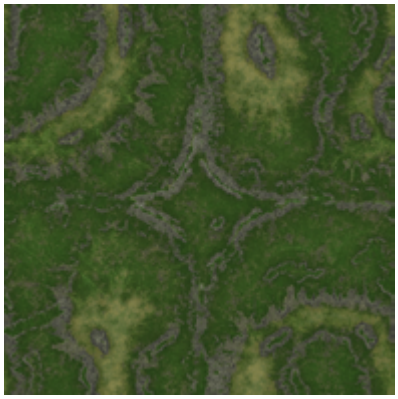


1500 m

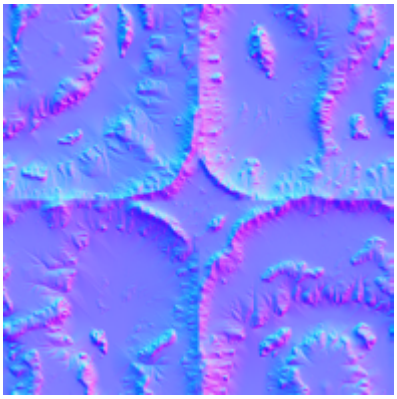


1500 m

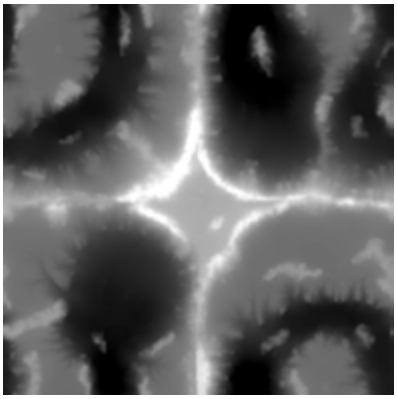
With Horizon[ON] you can also use a Heightmap for displacement, which is optional. Such a heightmap is an 8 bit greyscale image that contains elevation data. Like the normal map you may want it to be fitting with you other layer textures(diffuse, normal). Here an example(below right):



1500 m



1500 m



1500 m

The heightmap of a layer is special as it need to be stored in a shared texture(this is because of the texture sampler number limit) the heightmap of layer 1 goes into the alpha channel, the heightmap of layer 2 goes into the red channel, the heightmap of layer 3 goes into the green channel and the heightmap of layer 4 goes into the blue channel. So you will have one combined heightmap texture for all of your layers.

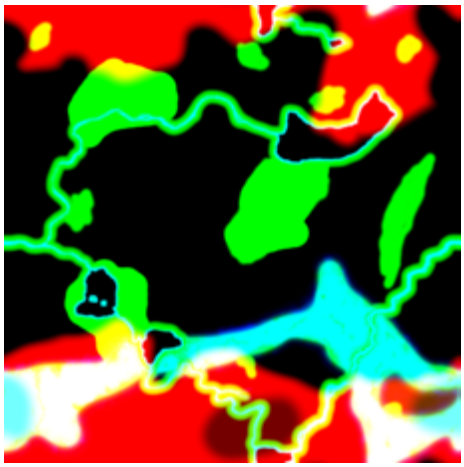
Apart from diffuse, normal and height you can also have emission, this would be stored in the alpha channel of the diffuse texture. Emission is very useful if you want to have a cityscape and like to have streetlights in the night, another usecases is glowing lava for vulcanic environments.

To recapitulate, a layer can consist of:

1 diffuse(or albedo) texture (optionally with emission stored in the alpha channel)
1 normal map (optional, depending on if the normalmap feature is enabled)
1 greyscale 8bit heightmap (optional - only needed if you use displacement)

The Mask(a.k.a. splatmap):
If you only use one layer you dont need to do more but if you have more than one layer you need to use a mask to specify where a layer is shown in world space. Just like the layers also the mask should be tileable to make most out of the texture resolution.

The mask can look like this:



Each channel specifies where a layer is drawn in world space. But the mask has only 4 channels but if we 4 layers + water(i mentioned that water is just like an extra layer) we have 5 different areas to assign. This works by placing one layer over the other. which means that layer 1 will be drawn everywhere and with the first channel in the mask you assign layer 2. **Layer 1** - everywhere, **layer 2** - drawn where the mask's red channel is white, **layer 3** - drawn where the mask's green channel is white, **layer 4**, drawn where the mask's blue channel is white, **water** - drawn where the mask's alpha channel is white.

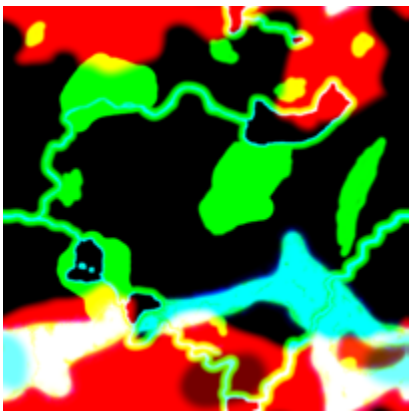
Red - Layer 2

Green - Layer 3

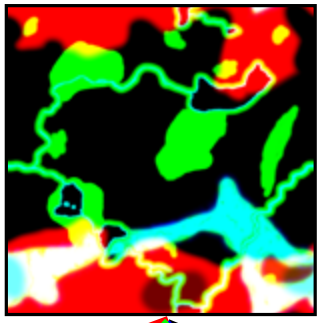
Blue - Layer 4

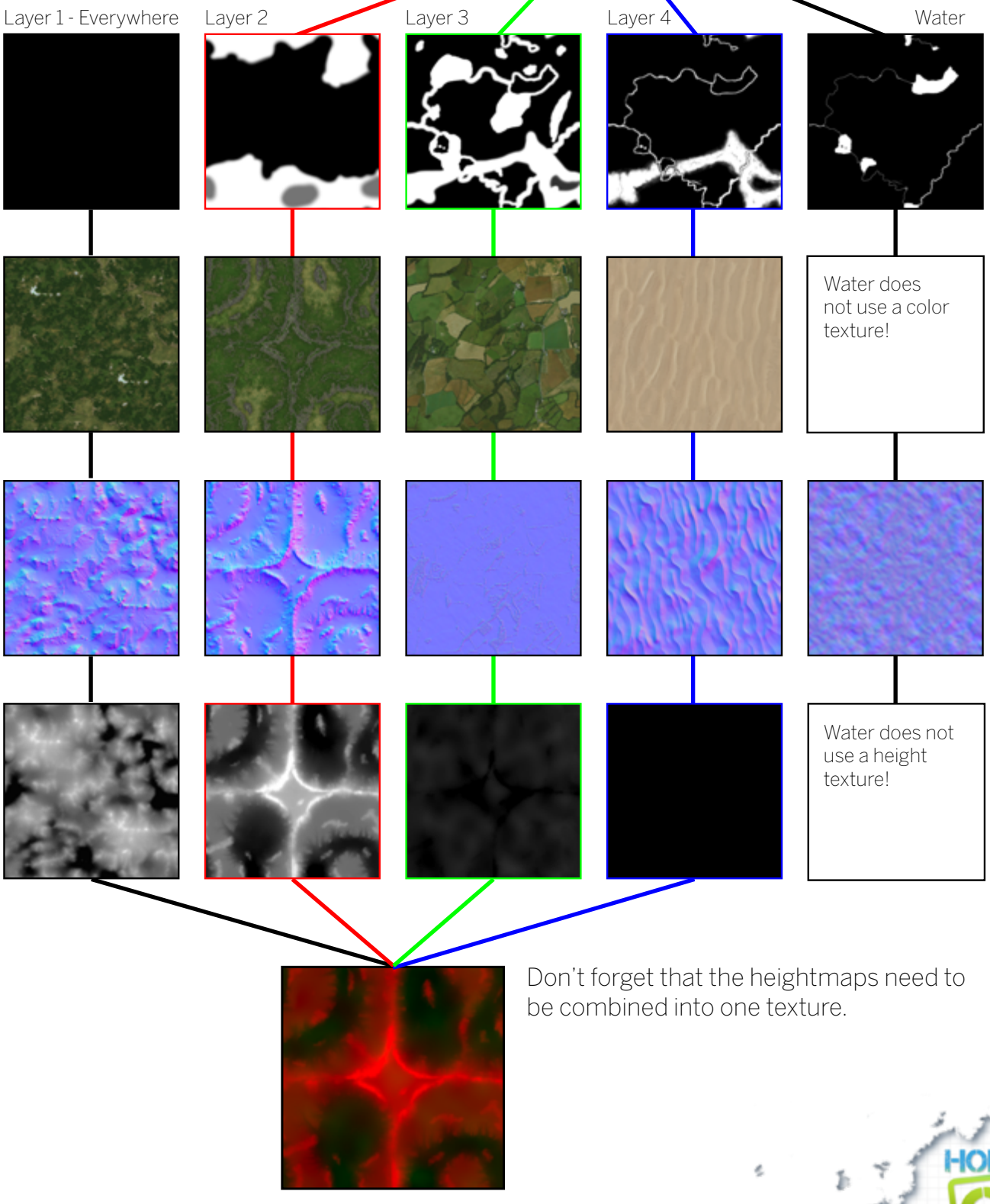
Alpha - Water

The result looks like this(below):
Mask only on the left, mask blended with the result in the middle, result only on the right.



Here you can see how the mask is used to blend the layers on Horizon[ON].
This diagram will hopefully help you to understand the concept better. To really grasp it, i recommend that you experiment with the example scene. Modifying the included textures is an easy way. You can always restore the original textures by getting them from the assetstore again.





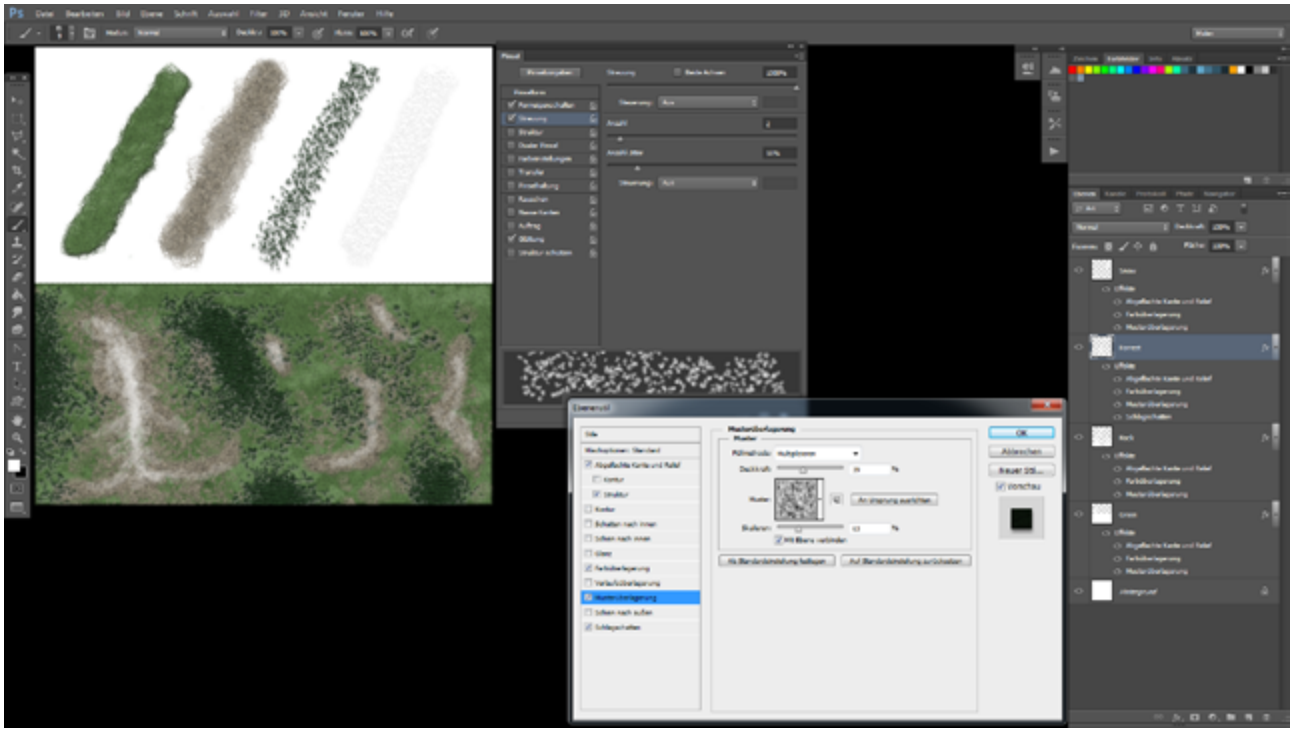
Don't forget that the heightmaps need to be combined into one texture.

How to create Textures & Maps for Horizon[ON]

There are lots of ways to create textures and maps and it is a very general skill teached in many tutorials on the web. To use Horizon[ON] to its maximum potential, you or an artist in your team, need to be able to author textures in one or another way. As it is definitely far out of the scope to teach you texturing in this document, i will assume that you either have a texture artist that you can assign this task to, or that you know about texturing yourself. I will just touch very briefly on some techniques.

Painting textures

Textures for horizon can easily be painted. A good way to do it is to use layer effects in photoshop. you can assign different patterns to layers and then paint with these patterns:



Also heightmaps can be painted and normal maps as well, although it is easier to paint a heightmap and convert it to a normal map with unity texture importer(or a 3rd part tool like xNormal(Free - external), Crazybump(Paid - external), BittmapToMaterial(Paid - Unity), etc.). Painting textures is obviously the most straight forward way,

Using Aerial photos or satelite imagery

This is also a good way and with many tools you can not only capture the color but also the height. A great tool is World Composer by Nathaniel Doldersum(look on the Unity Asset Store). You can also just take screenshots from google maps.

Generating textures

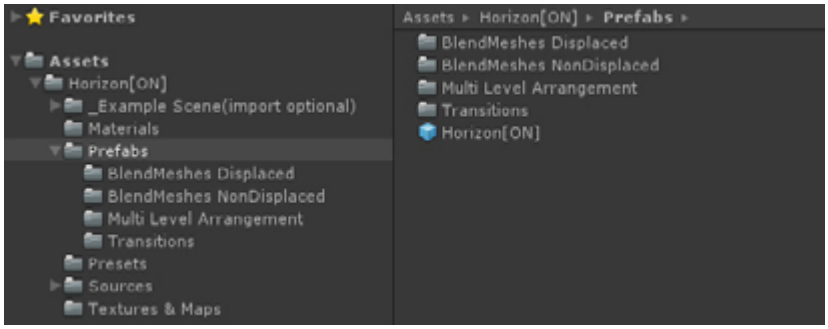
There are a few great tools for this available, the one i use myself is WorldMachine(Paid-External), another is TerrainComposer(Paid - Unity).

Combining all of the above

You can blend the results in an image editor.
By the way, each layer in the example scene uses one of the above techniques. Layer1 is generated(WorldMachine), Layer2 is a combination of painting and generating, layer 3 is a satelite image, Layer 4 is painted.

The Content of Horizon[ON]

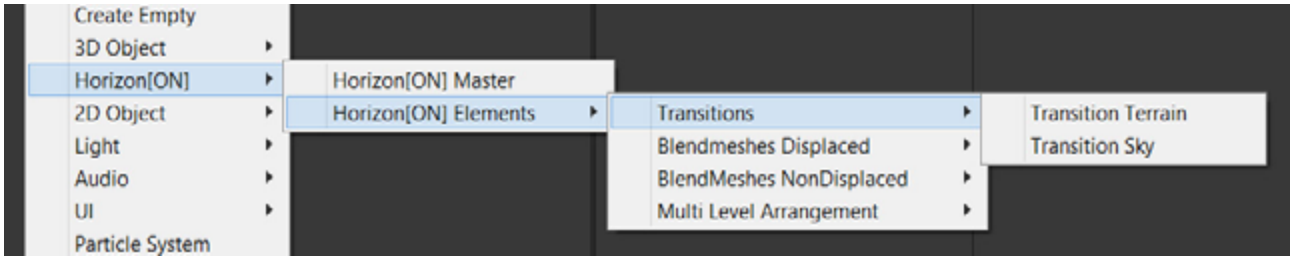
Here you can see the folder structure after import. The most important folders are of course "Prefabs" and "Materials" (and their sub-folders). The contents of the "Textures & Maps" folder can give you an impression of how masks and textures can look like. The example scene contains things that are needed for the "Getting Started Guide".



For the most part you wont need to care about the content of the "Sources" folder. It contains the shaders, Editor scripts and other internal parts of Horizon[ON].

Adding Prefabs to your scene

To add prefabs to your scene you can either, drag prefabs from the prefabs folder to the hierarchy, or use the create menus. The easiest way is to right click in the heirarchy and use the menu like shown in the picture:



The Horizon[ON] Master

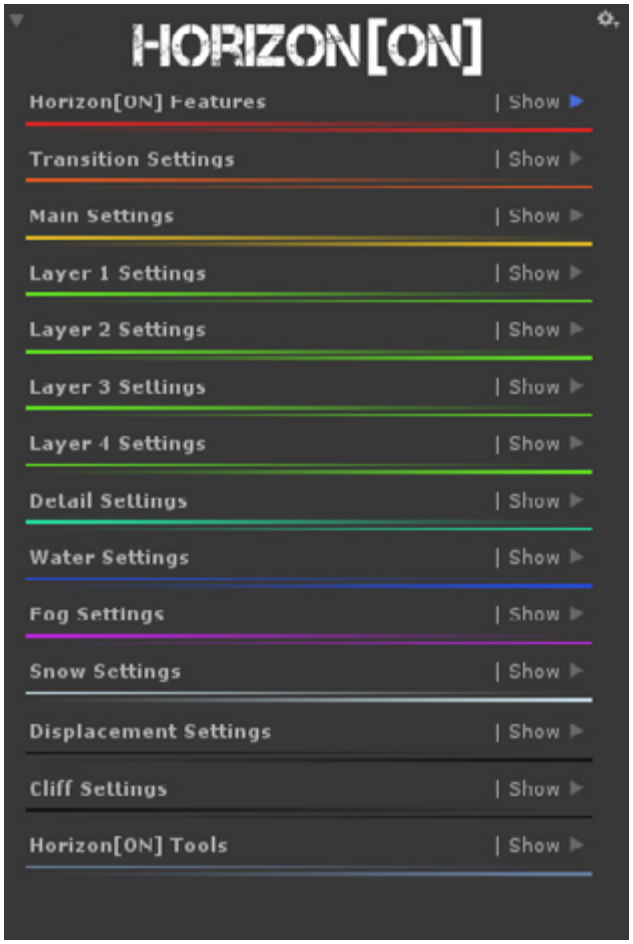
The Horizon[ON] Master is always the basis and the first thing you need to create. It is a script that makes it possible to utilize all shaders/materials that come with Horizon[ON] at once. It is crucial that the materials stay synchronized and Horizon[ON] Master will make that really easy. It is a component of the Horizon[ON] prefab which in most cases should be the parent of all the Horizon[ON] Elements. It will automatically scan its children for the used materials and depending on which materials are used, it will present you different options. What ever parameters you adjust on it will affect all its children. It is definitely possible (and in some rare usecases even unavoidable) to adjust parameters on the materials it self.

While there are some differences in the material inspectors and the Horizon[ON] Master, it will still give you full access to all parameters. Moreover it will also expose only the values that are actually availible in the current configuration of features and elements which are in use. The material inspectors will also do that to some extent but the Horizon[ON] Master is definitely a more convenient thing to work with. Using the material inspector is really only neccesary for advanced users that need to treat some materials diffenently than others for some unforseen reasons.

Here you can see the Horizon[ON] prefab (with the master component) when you have added it to the scene:

Since there are a lot of settings, i built in foldouts for each feature and the other important sections. Clicking on the triangle will fold/unfold them.

The Horizon[ON] Master in detail



This is the Horizon[ON] Master when it has children of all types and when all features are enabled. As you can see it shows a lot fo categories. I will explain each category in greater detail further down in this manual. For now here is an overview:

Horizon[ON] Features - This section lets you enable and disable features.
Transition Settings - Here you can control the transition from terrain to the horizon and the transition from the horizon to the sky.
Main settings - here you can control things like the mask, and global aspects, like global color tint, etc.
Layer (1,2,3,4) Settings - depending on how many layers are enabled. It will give you access to their parameters and their textures.
Detail Settings - here you can control the paramaters and textures of the detail feature.
Water Settings - here you can control all aspects of the water feature.
Fog Settings - Things like fog amount, fog start distance and transition length can be adjusted here.
Snow Settings - All aspects of the dynamic snow feature can be adjusted here.
Displacement Settings - displacement height and other settings related to displacement .
Cliff Settings - all settings for the cliffs.
Horizon[ON] Tools - all kind of things that do not fit into the other categories can be found here.

Like mentioned, the master script wont give you any options if the Horizon[ON] object does not have children. The easiest way to add child elements to it, is to right click it in the hierarchy and then select what you need from the create menu.

Horizon[ON] Features

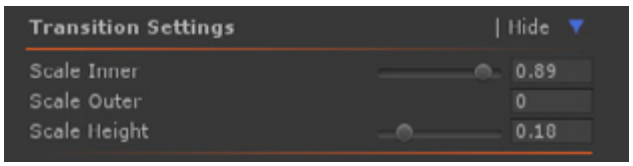
Use these checkboxes to enable or disable certain features. This will compile a shader variant that has only the selected features.

Layercount - Set how many layers you want to use.
Enable Specularity - Check if you want to use direct specularity.
Enable Reflections - Check if you want to use reflection probes/sky reflection.
Enable Normalmapping - Check if you want to use normalmaps.
Enable Emissiveness - Check if you want to use emissiveness.
Enable Detail Textures - Check if you want to use detail textures.
Enable Water - Check if you want to use water.
Enable Fog - Check if you want to use layered fog.
Enable Snow - Check if you want to use dynamic snow.

Of course, the more features you enable the slower Horizon[ON] will be. This is mainly of concern for mobiles, as on desktop even with everything enabled Horizon[ON] should perform **very** good. Also on mobile you might run into compatibility issues if you just enable everything. It depends on the devices and unfortunately it is impossible to guarantee that everything will work on every possible device out there.

Depending on which features you select, there might be some options invisible in the categories below and also in the material inspectors.

Transition Settings

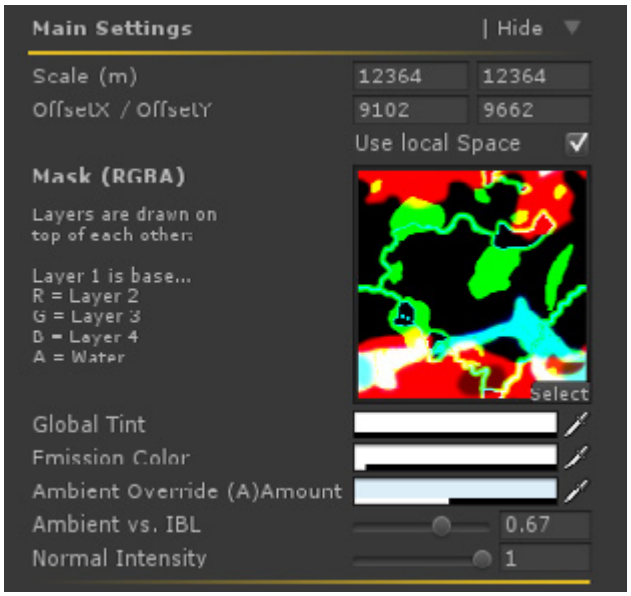


If there is a child element of the type "Transition" these options are availible for you to adjust.

Scale Inner - this is the transition length from the terrain to the horizon. The longer the transition is, the smoother it will appear but it will also make you lose more of your useable terrain area, so it's best to find a good compromise.
Scale Outer - here you can adjust how far the object should extend outwards. Basically, bigger is better, definitely scale it as far as your Camera farclip plane allows.
Scale Height - this is the transition height from the horizon to the sky. If you want the horizonline to be smooth, use a high value, if you want your horizon line to be sharp, use a small value.

"Scale Inner" only affects the "Transition Terrain" element. "Scale Outer" and "Scale Height" only affect the "Transition Sky" element.
Note that you have to scale the Horizon[ON] Transition elements first in the transform(!).
E.g. If your terrain is 5000m big you also need to put that value into the transforms scale.
(Uniformly - X: 5000, Y: 5000, Z: 5000)

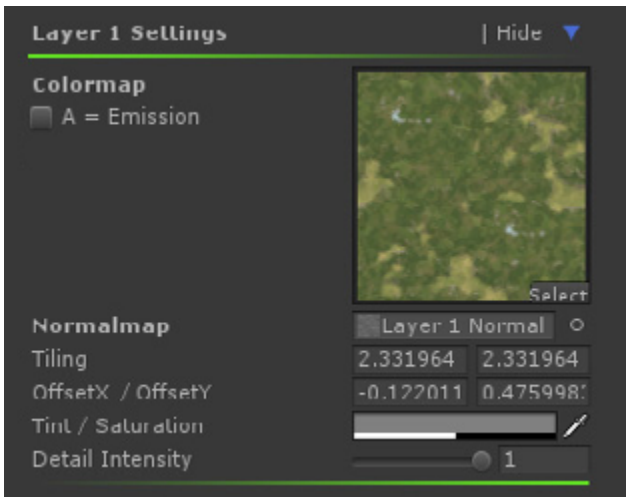
Main settings



(Note: not all of the properties listed below are visible by default - you may have to enable features in the features section to see them)

- **Scale (m)** - here you can adjust how big the mask(aka. splatmap) is going to be. This is in meters (scene units). also you can offset it. This mask can and should be tileable.
- **OffsetX / OffsetY** - Where in worldspace shoud the mask be placed. You can drag the Labels individually to move the mask around.
- **Use local Space** - Usually the mask and the maps will be projected in worldspace, so even if you move the object the textures will stay in place. If you don't want that(for example if you need to do world shifting) you can check this and the projection will be moved with the object.
- **Mask (RGBA)** - Assign your custom mask here. This mask works basically like a splatmap but can be painted in an image editing program. It does not need to be normalized, so layers are lerped on top of each other. Layer 1 is placed everywhere, layer 2 is placed where the red channel is white, layer 3 is placed where the green channel is white, layer 4 is placed where the blue channel is white and water will be placed where the alpha channel is white.
- **Global Tint** - Tints the whole Horizon[ON] object. This can help to get a perfect blend with your terrain. Also its is needed to adjust the brightness of Horizon[ON] for the color space that you use(linear vs. gamma).
- **Emission Color** - If you use the alpha channel of a layer as an emission mask it will take the according pixels of the layer colormap as emission color, this color can be tinted (multiplicative) using this color picker. The alpha channel acts as a brightness multiplier(useful to get HDR range).
- **Ambient Override (A) Amount** - You can make Horizon[ON] to use a different ambient light color than the one set in the render settings. The alpha channel controls how strong the override is.
- **Ambient Diffuse Intensity** - Adjusts how much Horizon[ON] is influenced by the Ambient light.
- **Ambient Reflection Intensity** - Adjust how bright reflections are on water.
- **Ambient vs. IBL** - Blends between the sky ambeint light and the simple Ambient light color(this also takes the Ambient Overide into account). This is useful in various cases, e.g. to animate a change in lighting conditions.
- **Normal intensity** - If you use normalmapping (features tab) you can use this to control the overall strength of the layer normal maps.

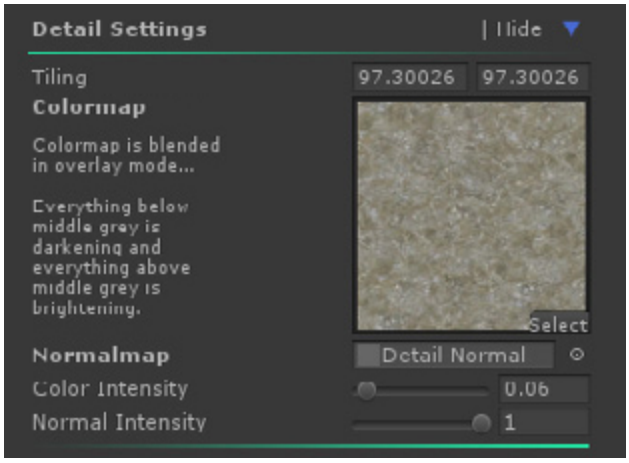
Layer Settings (1, 2, 3, 4)



(Note: not all of the properties listed below are visible by default - you may have to enable features in the features section to see them)

- **Colormap** - In most cases this will be a large scale image of a certain terrain type, e.g. a satellite image of a desert, a jungle, a city, fields, etc. Something you made with WorldMachine or just hand painted maps are of course suitable as well.
- **A = Emission** - You can use the alpha channel of your colormap as an emissive mask. This is useful e.g. for city lights or lava...
- **Normalmap** - Should fit your colormap but a normal map created from a grayscale perlin noise can do the trick as well in many cases. A normalmap generated in WorldMachine or Terrain Composer that matches your colormap is ideal.
- **Tiling** - Controls how often the colormap and normalmap will be tiled relative to the mask in the Main Settings tab.
- **OffsetX / OffsetY** - Use it to offset the layer. As with all offset fields you can click and drag the labels independently - this means for example that dragging "Offset X" will only affect the left float field.
- **Tint / Saturation** - Tints this layer. Middle grey is neutral, so you can darken or birghen your layer here. The alpha value of the color picker controls the saturation of this layer. This can be usefull if you want to animate seasons - e.g. a reddish tint in autumn and some desaturation in winter.
- **Detail Intensity** - If you have detail textures enabled in the features tab you can use this slider to adjust how strong the detail textures will be applied to this layer.

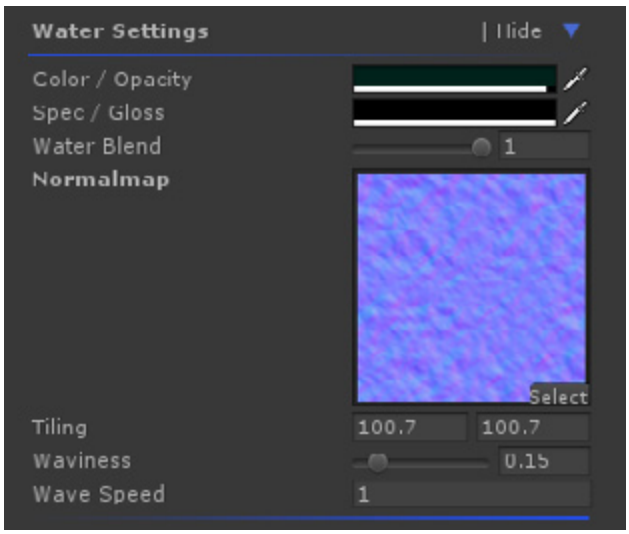
Detail Settings



(Note: not all of the properties listed below are visible by default - you may have to enable features in the features section to see them)

- **Tiling** - Controls how often the colormap and normalmap will be tiled.
- **Colormap** - Using detailmaps can improve the look dramatically. The detail colormap is applied to all active layers and the intensity can be adjusted per layer(see layer properties above). It is using an overlay blendmode (as you might know it from photoshop), that means every pixel brighter than a middle gray will be blended additively (brightens the underlying layer) and every pixel darker than a middle gray will be blended multiplicative (darkens the underlying layer). Note that middle gray is different depending if your texture import settings have "sRGB Texture" checked or not.
- **Normalmap** - Using a detail normalmap can improve the look as well and can make the layer textures appear of higher resolution than they actually are.
- **Colormap intensity** - This slider controls how strong the detail colormap will be blended with the underlying layers overall.
- **Normalmap intensity** - This slider controls how strong the detail colormap will be blended with the underlying layers overall.

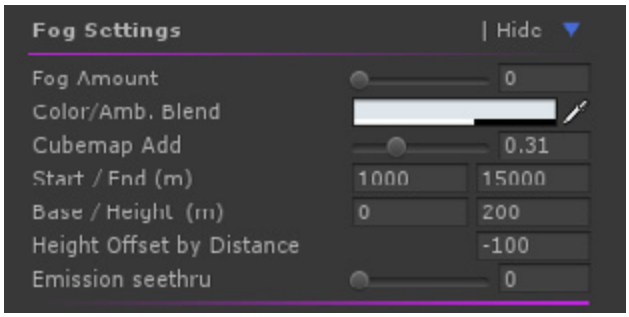
Water Settings



(Note: not all of the properties listed below are visible by default - you may have to enable features in the features section to see them)

- **Color / Opacity** - Sets the color of the Water (Tip: pick a color from a satellite image). The opacity is controlled by the alpha value of the color picker. You can make the water fully transparent and the reflection/specularity is still preserved.
- **Spec / Gloss** - Used to adjust the specularity (RGB in the color picker) and glossiness (A in the color picker).
- **Water blend** - You can use this to gradually hide or reveal the water. This can be used to simulate partially wet areas.
- **Normalmap** - Used for the water waves.
- **Tiling** - controls how long and wide your waves are going to be.
- **Waviness** - Controls the strength of the water waves. Set it to zero to simulate ice.
- **Wave Speed** - Self explanatory, Controls the speed of the waves.

Fog Settings



(Note: only available if "Enable Layered Fog" in the features tab is selected).

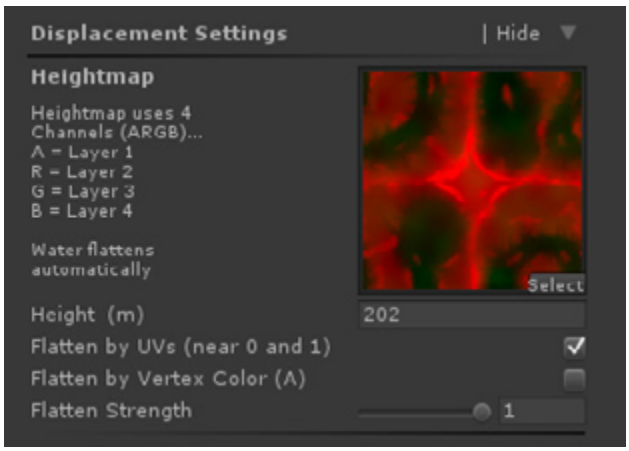
- **Fog Amount** - Controls how strong the fog will be applied to the Horizon[ON] objects.
- **Color/Amb. Blend** - Controls the color of the fog. The Alpha value controls how much from the ambient light color should be used as fog color.
- **Cubemap Add** - Controls how much of the reflection cubemap should be added to the fog color. This is usefull if you want to give some directionality to your fog color, it's also helpful sometimes to match the fog color perfectly with your skybox.
- **Start / End (m)** - The left input field controls how far from the camera position the fog starts. The right input field controls the horizontal distance from the start distance in which the fog will reach 100% opacity.
- **Base / Height** - The left input field controls the height where the fog starts(100% opacity). The right input field controls the vertical distance from the start height in which the fog will reach 0% opacity.
- **Height offset by distance** - Shifts the fog height by distance. Brings further away fog higher or lower.
- **Emission seethru** - Controls how much of the emission color will punch through the fog.

Snow Settings

(Note: not all of the properties listed below are visible by default - you may have to enable features in the features section to see them)

- **Snow Amount** - controls the amount of snow.
- **Snow Color** - controls the color of the snow.
- **Base / Height** - The left input field controls the height where the snow shoud start. The right input field controls how long the transition is (this value works best if it pretty large, otherwise you may get a too sharp transition).
- **Slope Damping** - controls if snow can lay on vertical surfaces.
- **Color Damping** - Makes the snow amount dependent of the underlying color.

Displacement Settings

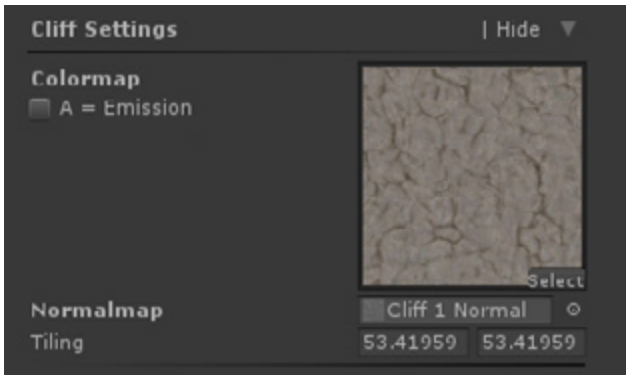


(Note: only available if a child exists which uses the displacement shader).

- **Heightmap** - This is the heightmap which is used for displacement. It can utilize all its 4 channels, each channel is used for one layer. The alpha channel is used for layer 1, the red channel is used for layer 2, the green channel is used for layer 3 and the blue channel is used for channel 4. Ideally the heighmaps are matching the colormap and normalmap of the corresponding layer. Water will be automatically flattened.
- **Height** - This controls the max. displacement amount in meters. Where a channel of the heightmap is white, it's corresponding layer will be displaced to the specified height
- **Flatten by UVs** - if this is enabled it will reduce the displacement using the UVs of the mesh. If the UVs are near o or near 1 the displacement is reduced if the uvs are at 0,5 the displacement is not reduced. A lot of the prefabs in the displacement category use this feature, so it is on by default and if you dont have a good reason to turn it of you should leave it on.
- **Flatten by Vertex Color** - This can be enabled if you want to use vertexpainting to reduce the diplacement localy.
- **Flatten Strength** - this controls how steep the falloff of the displacement reduction is for "Flatten by UVs"
- **SubDivisions** - (not illustrated on this picture) This controlls the amount of subdivisions if you use tesselation(DX11 only).

More details about displacement in the elements section of this manual.

Cliff Settings

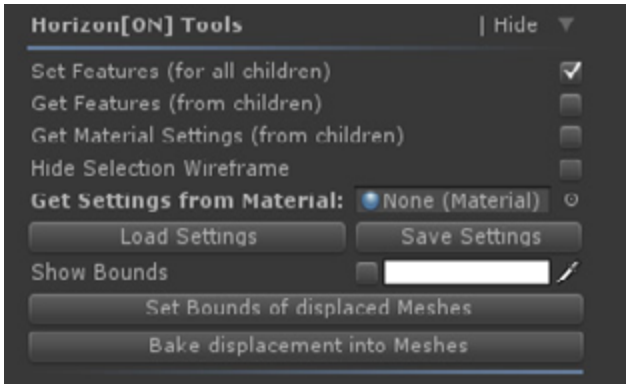


(Note: only available if a child exists which uses the cliff shader).

- **Colormap** - the colormap used for cliffs.
- **A = Emission** - like on the layer settings.
- **Normalmap** - the normalmap used for cliffs.
- **Tiling** - scales the cliff textures.

The textures of the cliffs are projected 2 planar. This makes it possible to have no stretching on steep surfaces without utilizing mesh UVs.

Horizon[ON] Tools



(Note: not all of the properties listed below are visible by default - you may have to enable features in the features section to see them)

- **Set Features** - If this is enabled Horizon will set the features for all of its children, this is enabled by default. However if you want to enable/ disable certain features for only some of the children you might want to turn this off.

- **Get Features** - If this is enabled it will get the features from its children and apply it to the Horizon[ON] Master. Its not recommended to enable this as it is possible that it screws up your feature settings if you add a new child with a different feature set, so use it with caution. If you are not adding new children it is safe though. This is probably never used but it might be useful if you want to nest multiple Horizon[ON] Master objects.

- **Get Material Settings** - Similar as the setting above but for the material settings. It also has to be treated with caution for the same reason and probably is only useful if you want to nest Horizon[ON] Master objects.

- **Hide Selection Wireframe** - If you enable this it will hide the selection wireframe for all of its children. Keep in mind that this will be remembered by then children so if you unparent a child while its wireframe was hidden you'll have to reparent it to the Horizon[ON] Master to be able to reenable the selection wireframe. However, this is very useful for tweaking the material settings without being distracted by the wireframes.

- **Get Settings from Material** - If you drag a material into the slot, Horizon[ON] Master will adopt all of its settings.

- **Load Settings** - Here you can load previously saved settings.

- **Save Settings** - You can save the current settings into a new preset by using this button. This will also be useful if you have multiple setups and you are not yet sure which you are going to use.

- **Show Bounds** - If there are children which use displacement it is important that they have correct bounds, otherwise they might be culled incorrectly. This would lead to disappearing objects or not optimal performance. You can visualize the bounds of these objects by enabling the checkbox. The color picker lets you choose a color for the bounds visualization.

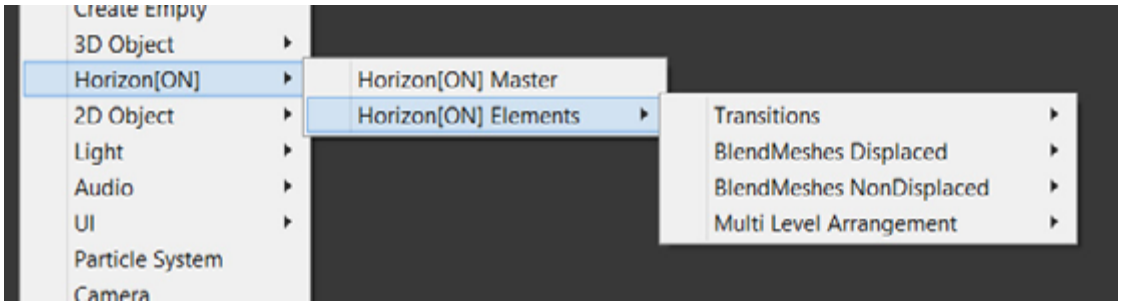
- **Set Bounds of displaced meshes** - If you click this button Horizon[ON] Master will set the correct bounds. Note that you have to set the Y-scaling of the objects(and its parents) to 1 otherwise the bounds will be scaled. Please use the show bounds feature(above) when you are working on the bounds to make sure everything is as you expect it to be.

- **Bake displacement into meshes** - If you click this button, the meshes which use displacement will be replaced by meshes that have the displacement baked into them. This is usefull to create colliders(which are needed for tree painting), to have correct normals on layer transitions, and to optimize the resulting meshes in a 3d program. See the "About Bounds, Culling & Displacement" Section further down below to undestand what this powerful feature can do for you.

The Horizon[ON] Elements

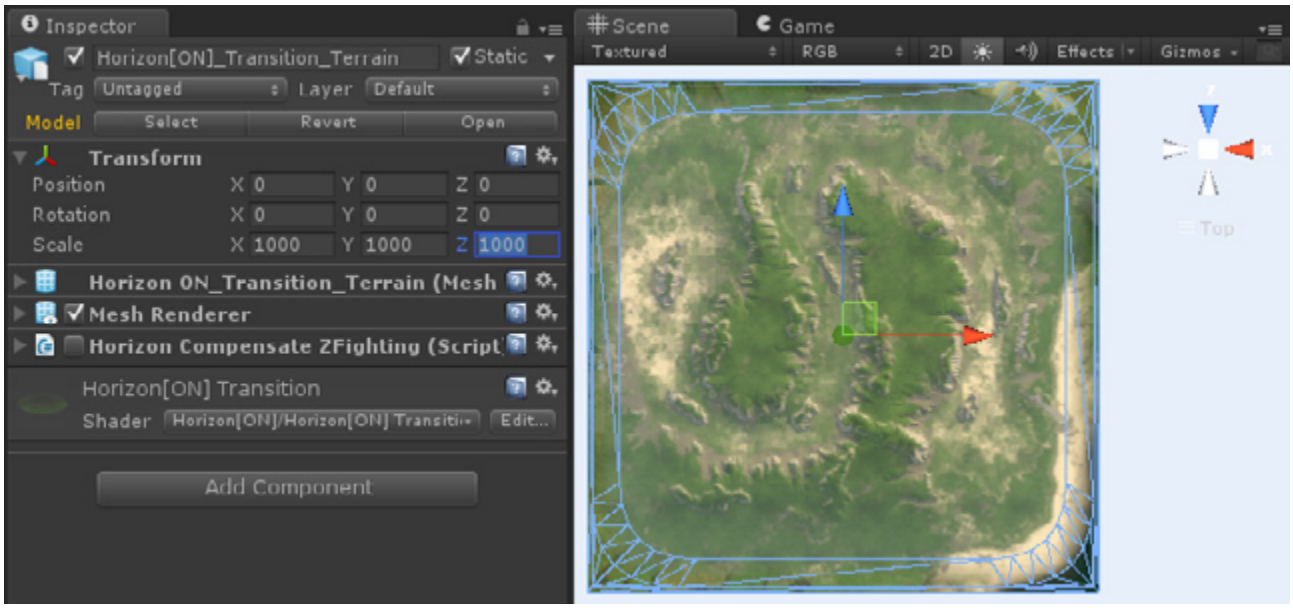
There are 4 main categories of elements:

- **Transitions** - there are 2 elements in this category, "Transition Terrain" and "Transition Sky".
- **BlendMeshes Displaced** - this category contains elements that use displacement and can be placed on a flat horizon.
- **BlendMeshes NonDisplaced** - this category contains elements that don't use displacement and can be placed on a flat horizon.
- **Multi Level Arrangements** - this category contains elements that are used to compose a horizon in as special way.



The Scale of Horizon[ON] Elements

It is important that the elements have the right scale, you set the scale of elements just like you are used to, by adjusting the scale of the transform. Like in this picture where i added a "Transition Terrain" element as a child to the Horizon[ON] object, i need to set the right scale. The default is a scale of 1000, if your terrain is for example 2500, you need to set the scale of the "Transition Terrain" element to a scale of 2500 as well.

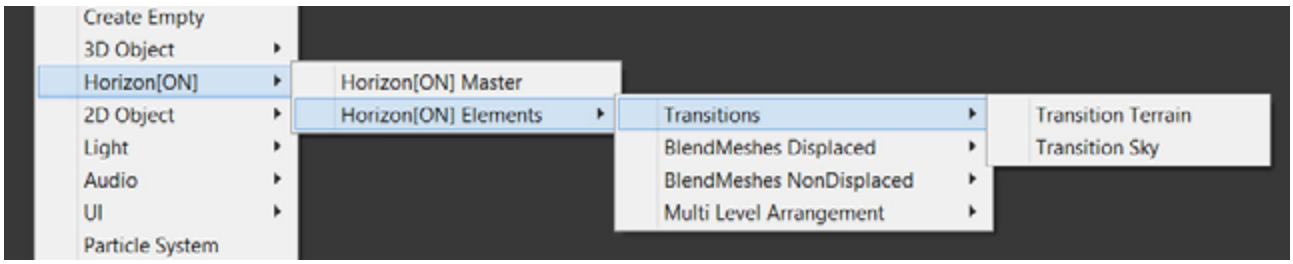


Some elements need to be scaled uniformly and some others can be scaled freely. Displaced elements use the scale of the Y-axis do define their vertical bounds. More info about scaling can be found in the description of the elements.

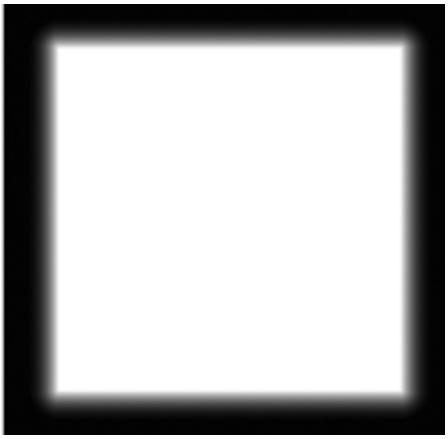
Transition Elements

There are 2 elements in this category:

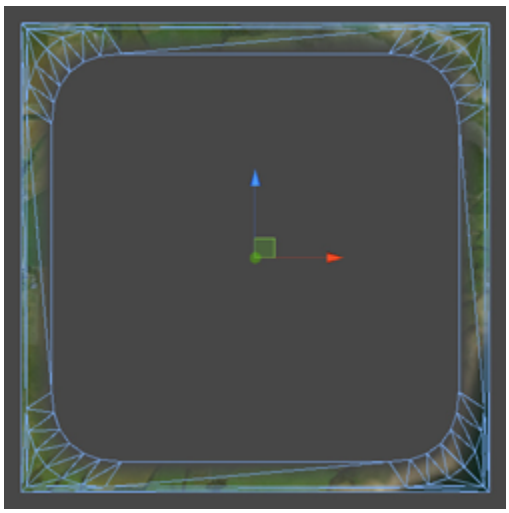
- **Transition Terrain**
- **Transition Sky**



The "Transition Terrain" Element

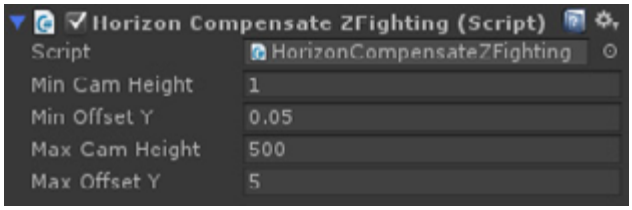


You will need this element if you want to smoothly blend your terrain with Horizon[ON]. To be able to do this you need to flatten your terrain on the outside. Usually the very outside edge is not reachable anyway as it would in the worst case let the player fall off of your terrain or reveal an ugly end of your world. However it is not necessary to smoothly blend Horizon[ON] and your terrain in all cases but if you want to do so this is currently the best (and easiest) way.
Flattening the terrain is relatively simple. You just need to export its heightmap to an image editor that supports raw files and draw a black area around it. If you use Photoshop you can use an inverted rounded rectangle with a black glow multiplied on top of your heightmap. Just like in the image on the right. I recommend to level your heightmap so it uses its full dynamic range.



The "Transition Terrain" is a mesh that has a transparent falloff on the inside, this falloff can be scaled by using the slider in the "Transition Settings" on the Horizon[ON] Master...

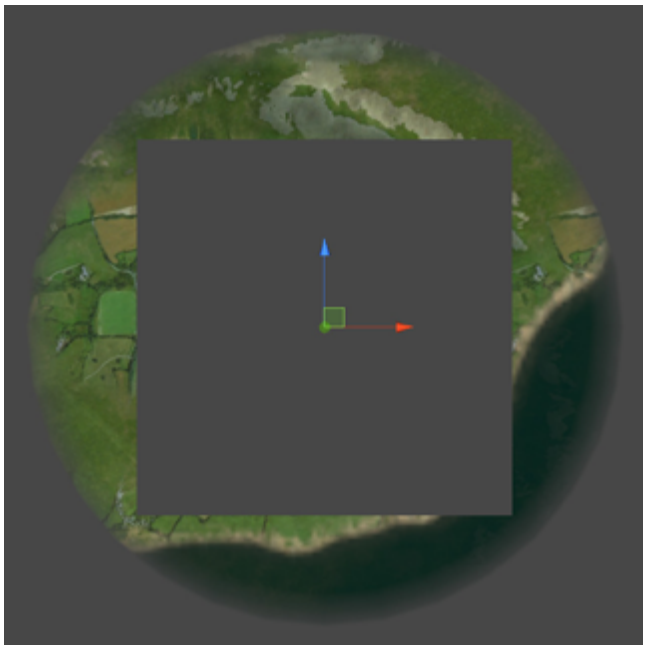
The falloff should be scaled to match your flattened area on the terrain.
The transform of the transition object should be scaled exactly as big as your terrain is. The position X and Z should be in the middle of your terrain (i recommend to offset your terrain by the half of its size so it is placed exactly in the middle of your scene - called the scene origin). The position Y should be at the bottom of your terrain(that would be 0 in almost all cases).



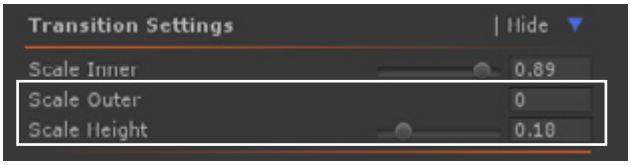
By default there is a script attached to the prefab, called "HorizonCompensateZFighting". Since we want our transition to be as close as possible to the terrain(so we can not look under the transition and see the skybox in the gap) we could run into zFighting problems if we are in great height above it and the distance is very small.
This script will offset the transition based on the main camera position (global Y-axis). Depending on your case it might be necessary to use this script or not. However if you use it, you will need to set the transition to be non static. This would then be (in the most cases) the only element of Horizon[ON] which is not set to static.

The "Transition Sky" Element

You will need this element if you want to smoothly blend the horizon with the sky and have a rounded world.



If you are not using multi level arrangements you can set the scale of the object just like the scale of your terrain. Scaling should be uniform. It should be positioned in the middle and bottom of your terrain in most cases. You can then control how far it extends by using the "Scale Outer" and "Scale Height" values on the Horizon[ON] Master.

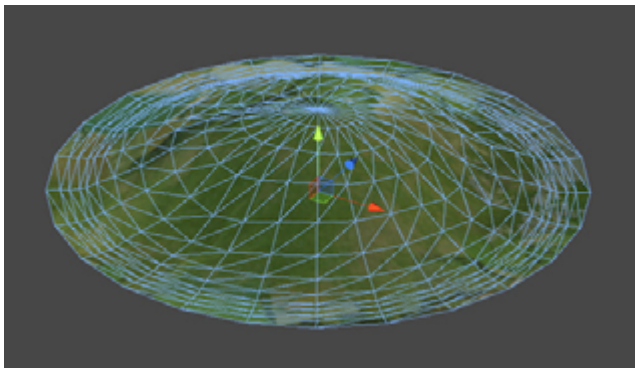


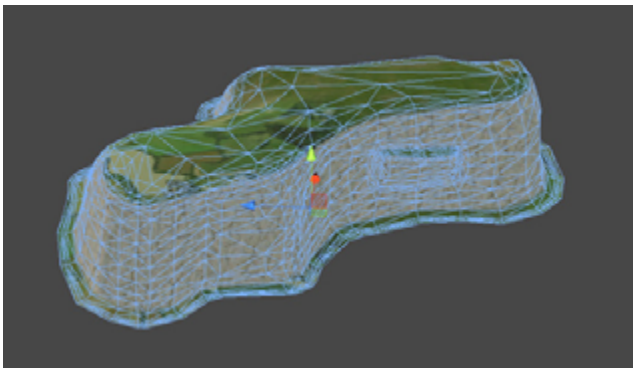
You should take care that your camera far clip plane is big enough so it wont clip the transiton.
In many cases this object is also the basis to place elements from the "BlendMeshes" categories on top. If you are using multi level arrangements, this object should be the outmost object(and therefor should have the largest scale - dont forget to scale it uniformly).

BlendMeshes NonDisplaced Elements

There are 2 elements in this category:

- **Hill Example**
- **Cliff Example**





These are called exampes because you can easily model more of them in a 3D program. You dont even need to take care about UVs. These objects can be placed anywhere on the horizon if it is flat. You can also scale them in any way you like. These objects are great to add vertical detail to your horizon if you are strongly constrained by performance(Mobile) and cannot use displacement.

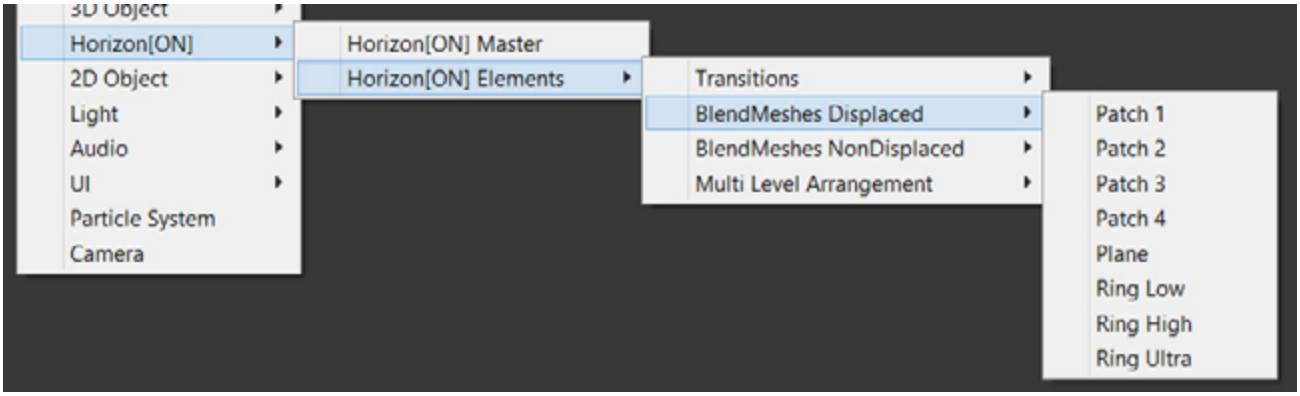
Cliffs and Hills are very similar, the difference is that cliff objects use 2 planar mapping in world space for vertical surfaces. The great thing about these objects is that they can have mesh colliders attached to them, which opens a lot of possibilities. Because they are textured in worldspace they can be intersected nicely, the textures would always be continous in such cases. Those objects would be fantastic candidates for creative misusing. Cliffs for example could be used as dungeon walls or obstacles on a minigolf course.

About modeling your own BlendMeshes:
It is very easy and any shape is possible, the only thing you should take care of, is that their pivots are in the middle and on the bottom. The normals of the vertices that touch the underlying surface need to point straight up to get perfect blending. This is very easy to do, just select all vertices that are on zero height and change their normals... depending on your 3d program you have to look for a command/modifier like "User Normals" or "Explicit Normals".

BlendMeshes Displaced Elements

There are 8 elements in this category:

- **Patch 1**
- **Patch 2**
- **Patch 3**
- **Patch 4**
- **Plane**
- **Ring Low**
- **Ring High**
- **Ring Ultra**



Patch 1, Patch 2, Patch 3, Patch 4 and Plane
These are basically all pre-tesselated planes in different shapes, they use the Flatten by UVs feature of the displacement shader. You can use them to displace parts of the horizon locally. You can move them around freely and also scale them on the X and Z axis. Like all blendmeshes they should be placed on the same height as the underlying surface.

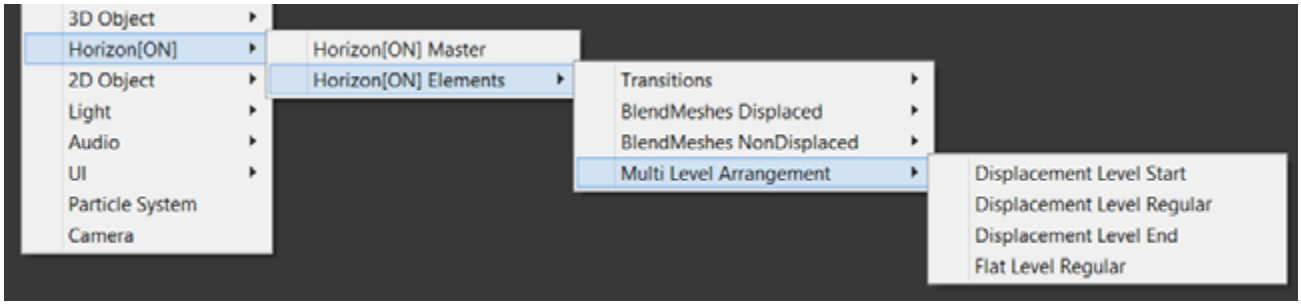
Ring Low, Ring High and Ring Ultra
These are also pretesselated Planes but shaped and arranged so they are forming a ring. You can place these rings around your terrain. Low, High and Ultra just says how much they are pre-tesselated. otherwise they are all the same. They also use the "Flatten by UVs" feature of the displacement shader.

If you are using DX11 and want to make use of tesselation you can assign the "Horizon[ON] Tesselation (DX11)" material to these elements instead of the "Horizon[ON] Displacement" material.

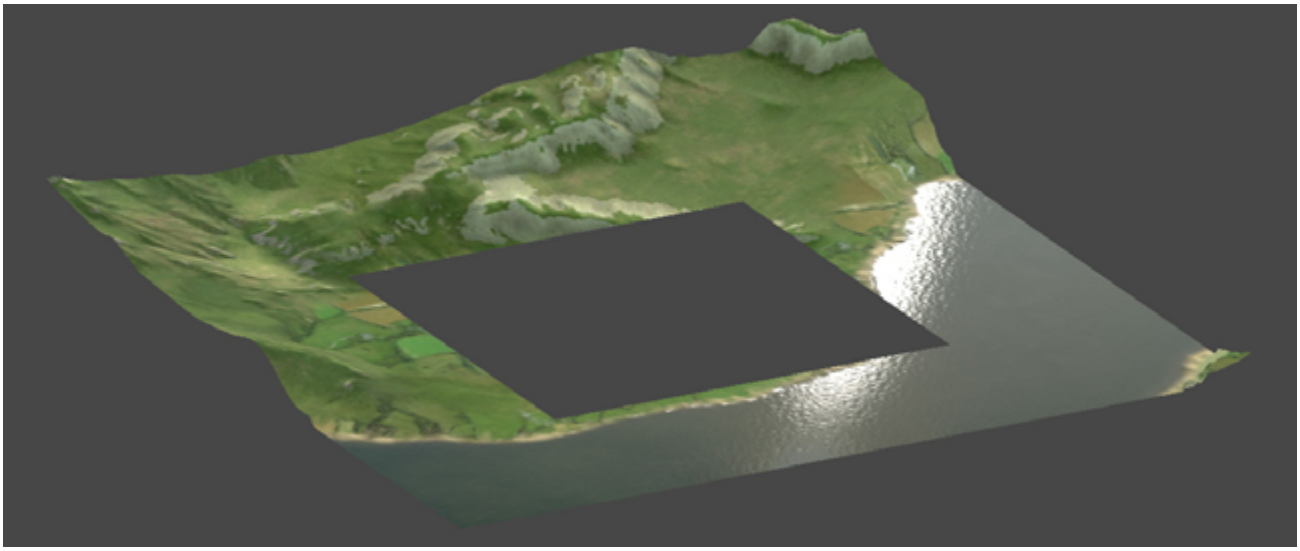
Multi Level Arrangement Elements

There are 4 elements in this category:

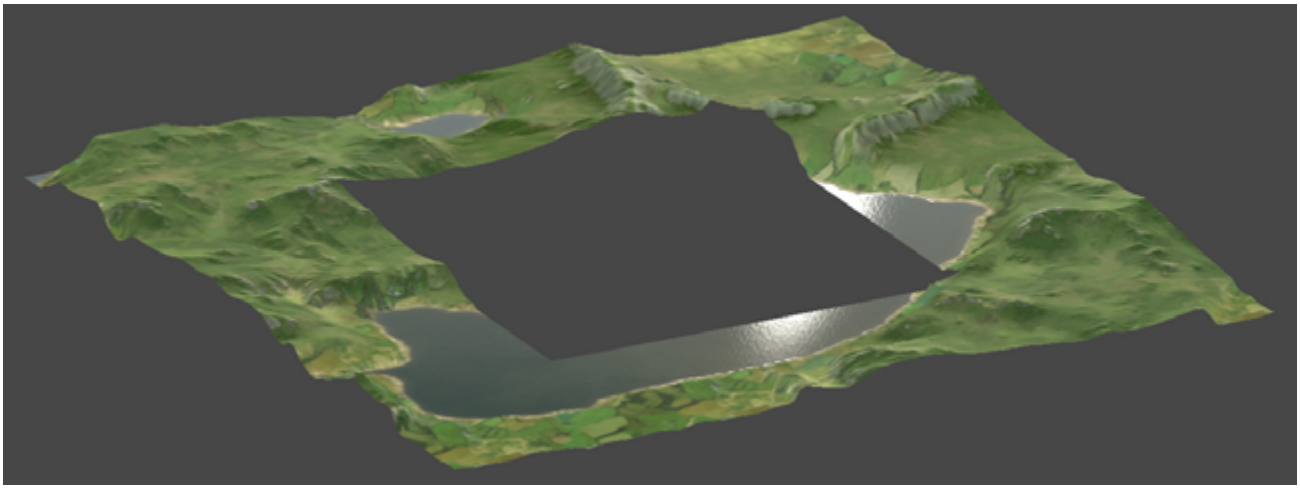
- **Displacement Level Start**
- **Displacement Level Regular**
- **Displacement Level End**
- **Flat Level Regular**



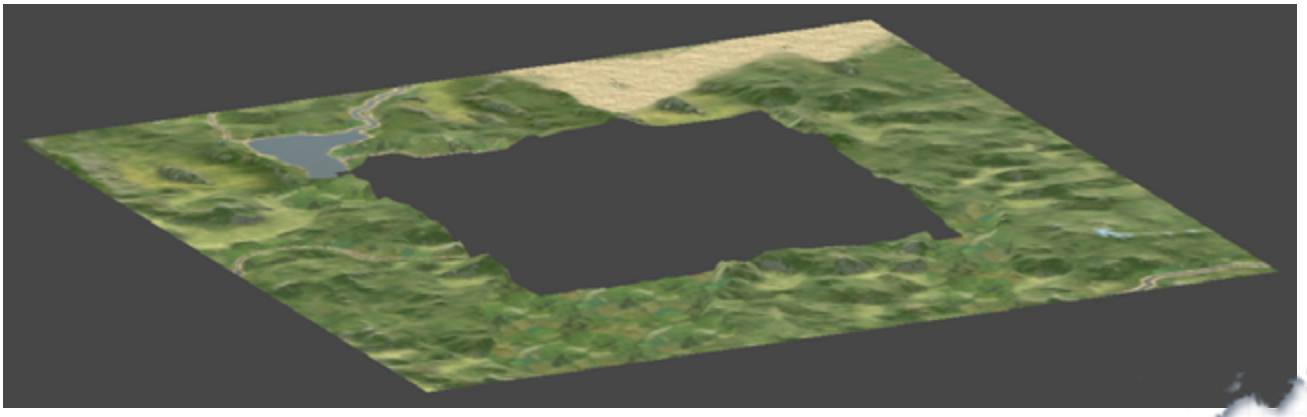
Diplacement Level Start
This is a set of 12 planes, arranged so that they enclose the terrain. You can scale the parent on the X-axis and on the Z-axis so it fits your terrain, so if your terrain is 2000x2000 you would need to set the scale to X: 2000 and Z: 2000. The scale of the Y-axis is used to set the vertical bounds. The planes are as well pretesselated and use the "Flatten by UVs" feature of the displacement shader. The UVs are set in a way so the displacement is flattened towards the center.



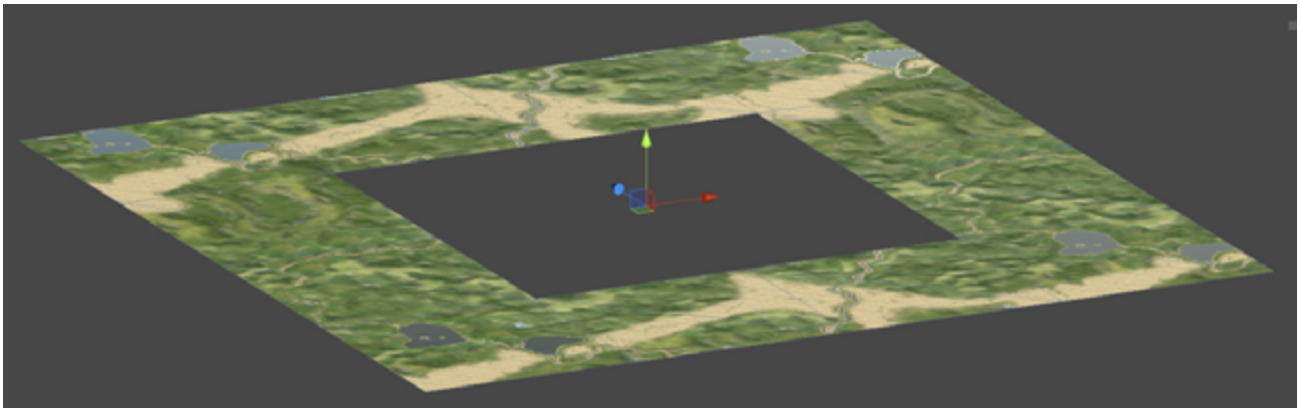
Diplacement Level Regular
This is exaclty the same setup like above but the diplacement is not flattenend. You would scale this always twice as big as the previous level. So, this would enclose the "Displacement Level Start". You can also duplicate it and enclose another "Displacement Level Regular". You can do this recursively as many times as you wish but usually no more than a total of 5 levels is needed.



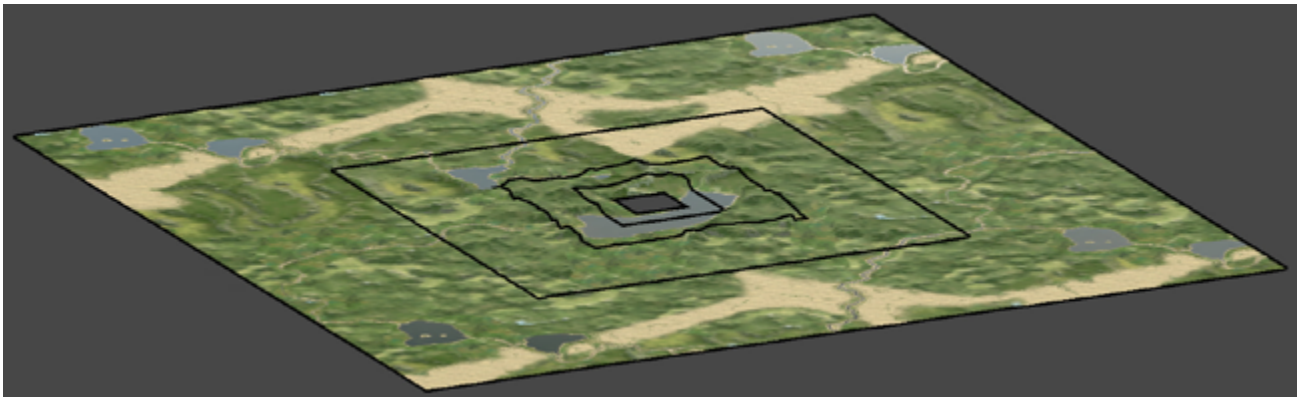
Diplacement Level End
Exaclty the same setup like "Displacement Level Start" but the other way around. It is flattended on the outside. Scale this always twice as big as the previous level. So, this would enclose the "Displacement Level Regular".



Flat Level Regular
Just the same as "Displacement Level Regular" but without displacement.



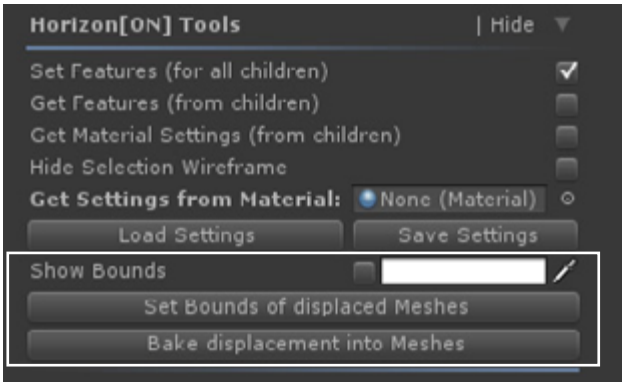
The Levels combined, a typical multilevel arrangement:
Displacement Level Start, Displacement Level Regular, Displacement Level End, Flat Level Regular in order, always scaled twice as big as the previous level.



Since all the levels have the same amount of triangles but are scaled always twice as big there is a logarythmic falloff of detail, which means the triangles stay roughly at the same size on the screen. Much detail in the foreground less detail in the background.

About Bounds, Culling & Displacement

Bounds:
Correct bounds are important for efficient and accurate culling. Since culling does not take vertex displacement into account we need to take care about it. On the Horizon[ON] Master in the Tools section, you can enable "Show Bounds" to see how the bounds fo your displaced elements look like. We control the bounds by the Scale of the Y-axis of these elements. You can manually adjust the bounds if you like but the easier way is to use the "Set Bounds of displaced Meshes" button. By doing so all displaced elements will be checked, its position, scale, mask & layers will be taken into account and the scale of the Y-axis will be set to the correct value.



Baking of displacement:
Why would you want to do that? Well, there could be a number of reasons, lets say that you need collision on your displaced elements. Maybe you just need it to be able to place objects on displaced elements more conveniently, maybe its essential for your gameplay. Maybe you want to use displaced elements on mobile but vertex displacement via textures is not an option there. If you want to bake the displacement into your meshes i recommend to duplicate your Horizon[ON] object fist so that you can go back more easily.
You can just click the "Bake displacement into Meshes" button. It could take a few seconds. When its done you will notice that the displacement seems to be twice as strong. That is because the meshes are now already deformed and the shader is putting additional displacement on top of that. So after baking, you need to change the material from "Horizon[ON] Displacement" to "Horizon[ON] Object".

About the Materials and the shaders:

While the 10 materials of Horizon[ON] share most parameters, they serve different purposes. Every material uses a shader with the same name as the material. These shaders have some unique properties:

Horizon[ON] Transition Terrain
This material uses transparency for the fade between the terrain and Horizon[ON]. Vertex offset offset is used for the size of the transition, it does not take the normals of the mesh into account.

Horizon[ON] Transition Sky
This material uses transparency for the fade between the sky and Horizon[ON]. Vertex offset offset is used for the size of the transition, it does not take the normals of the mesh into account.

Horizon[ON] Flat
This material is the most simple in the collection. It is intended for flat surfaces.

Horizon[ON] Object
This material is just the same as the one above but it takes the normals of the mesh into account. It is used for "Blend Meshes" like hills. If you create your own hill shapes use this material.

Horizon[ON] Cliff
This material is just the same as the one above but it uses 2planar mapping for vertical surfaces.

Horizon[ON] Displacement
This material is just the same as the "Horizon[ON] Object" but it uses vertical displacement by texture. It is used for all objects which use displacement.

Horizon[ON] Tesselation(DX11)
This material is just the same as the "Horizon[ON] Displacement" material with additional tesselation.

Horizon[ON] Trees
This material is used for trees. Only a the snow and fog parameters are synchronized by the Horizon[ON] Master script. It's shader is made with Shader Forge and so you can customize it to your needs or use it as a basis to createoe other materials for Horizon[ON].

Horizon[ON] Buildings
This material is used for Buildings. Only a the snow and fog parameters are synchronized by the Horizon[ON] Master script. It's shader is made with Shader Forge and so you can customize it to your needs or use it as a basis to createoe other materials for Horizon[ON].

Horizon[ON] Mountains
This material is used for Mountains. Only a the snow and fog parameters are synchronized by the Horizon[ON] Master script. It's shader is made with Shader Forge and so you can customize it to your needs or use it as a basis to createoe other materials for Horizon[ON].

Reusing of Materials:

The way Horizon[ON] is designed you dont need to duplicate materials for each and every scene, instead material parameters are set via the Horizon[ON] Master script. It holds all the settings and synchronizes the materials on Awake. This prevents material inflation and keeps your project as tidy as possible, also it is much easier to manage things in one place and avoids accidental desynchronization of material parameters.

Syncronizing Materials: As mentioned above, when you use the Horizon[ON] Master to manage your materials they will be synchronized automatically but there may be cases where you cant use the Horizon[ON] Master but still need to be able to synchronize some or all properties of the materials. There is a script that can help you to do this. You can find it in the Folder "Horizon[ON]/Sources/Scripts/". The script is called **"HorizonCopyMaterialProps"**. To use it, assign it to a gameobject which uses the material that you want to synchronize with another. Then drag a gameobject which uses the target material into the slot. You can then either copy all material properties or just the ones that are enabled(features).

List of all shader variables (which might be needed for scripting):

```
// Disc Scaling ==========
if (m.HasProperty ("_ScaleInner")) m.SetFloat ("_ScaleInner", yourFloat);
if (m.HasProperty ("_ScaleOuter")) m.SetFloat ("_ScaleOuter", yourFloat);
if (m.HasProperty ("_ScaleHeight")) m.SetFloat ("_ScaleHeight", yourFloat);
// Main Settings ==========
if (m.HasProperty ("_MapScaleOffset")) m.SetVector ("_MapScaleOffset", yourVector4);
if (m.HasProperty ("_MaskRBlend1GBlend2BBlend3AWater")) m.SetTexture ("_MaskRBlend1GBlend2BBlend3AWater", yourTexture);
if (m.HasProperty ("_LocalSpace")) m.SetFloat ("_LocalSpace", (float)yourBool);
if (m.HasProperty ("_Tint")) m.SetColor ("_Tint", yourColor);
if (m.HasProperty ("_EmissionColor")) m.SetColor ("_EmissionColor", yourColor);
if (m.HasProperty ("_AmbientOverrideAAmount")) m.SetColor ("_AmbientOverrideAAmount", yourColor);
if (m.HasProperty ("_DiffIBLMulti")) m.SetFloat ("_DiffIBLMulti", yourFloat);
if (m.HasProperty ("_SpecIBLMulti")) m.SetFloat ("_SpecIBLMulti", yourFloat);
if (m.HasProperty ("_AmbientIBL")) m.SetFloat ("_AmbientIBL", yourFloat);
if (m.HasProperty ("_GlobalNormalmapIntensity")) m.SetFloat ("_GlobalNormalmapIntensity", yourFloat);
// Layer 1 Settings ==========
if (m.HasProperty ("_BaseColormap")) m.SetTexture ("_BaseColormap", yourTexture);
if (m.HasProperty ("_BaseColormap")) m.SetTextureScale ("_BaseColormap", yourVector2);
if (m.HasProperty ("_BaseColormap")) m.SetTextureOffset ("_BaseColormap", yourVector2);
if (m.HasProperty ("_TintSaturation")) m.SetColor ("_TintSaturation", yourColor);
if (m.HasProperty ("_BaseEmission")) m.SetFloat ("_BaseEmission", (float)yourBool);
if (m.HasProperty ("_BaseNormalmap")) m.SetTexture ("_BaseNormalmap", yourTexture);
if (m.HasProperty ("_BaseDetailIntensity")) m.SetFloat ("_BaseDetailIntensity", yourFloat);
// Layer 2 Settings ==========
if (m.HasProperty ("_BlendColorMap1")) m.SetTexture ("_BlendColorMap1", yourTexture);
if (m.HasProperty ("_BlendColorMap1")) m.SetTextureScale ("_BlendColorMap1", yourVector2);
if (m.HasProperty ("_BlendColorMap1")) m.SetTextureOffset ("_BlendColorMap1", yourVector2);
if (m.HasProperty ("_TintSaturationBlend1")) m.SetColor ("_TintSaturationBlend1", yourColor);
if (m.HasProperty ("_BlendEmission1")) m.SetFloat ("_BlendEmission1", (float)yourBool);
if (m.HasProperty ("_BlendNormalmap1")) m.SetTexture ("_BlendNormalmap1", yourTexture);
if (m.HasProperty ("_BlendDetailIntensity1")) m.SetFloat ("_BlendDetailIntensity1", yourFloat);
// Layer 3 Settings ==========
if (m.HasProperty ("_BlendColormap2")) m.SetTexture ("_BlendColormap2", yourTexture);
if (m.HasProperty ("_BlendColormap2")) m.SetTextureScale ("_BlendColormap2", yourVector2);
if (m.HasProperty ("_BlendColormap2")) m.SetTextureOffset ("_BlendColormap2", yourVector2);
if (m.HasProperty ("_TintSaturationBlend2")) m.SetColor ("_TintSaturationBlend2", yourColor);
if (m.HasProperty ("_BlendEmission2")) m.SetFloat ("_BlendEmission2", (float)yourBool);
if (m.HasProperty ("_BlendNormalmap2")) m.SetTexture ("_BlendNormalmap2", yourTexture);
if (m.HasProperty ("_BlendDetailIntensity2")) m.SetFloat ("_BlendDetailIntensity2", yourFloat);
// Layer 4 Settings ==========
if (m.HasProperty ("_BlendColormap3")) m.SetTexture ("_BlendColormap3", yourTexture);
if (m.HasProperty ("_BlendColormap3")) m.SetTextureScale ("_BlendColormap3", yourVector2);
if (m.HasProperty ("_BlendColormap3")) m.SetTextureOffset ("_BlendColormap3", yourVector2);
if (m.HasProperty ("_TintSaturationBlend3")) m.SetColor ("_TintSaturationBlend3", yourColor);
if (m.HasProperty ("_BlendEmission3")) m.SetFloat ("_BlendEmission3", (float)yourBool);
if (m.HasProperty ("_BlendNormalmap3")) m.SetTexture ("_BlendNormalmap3", yourTexture);
if (m.HasProperty ("_BlendDetailIntensity3")) m.SetFloat ("_BlendDetailIntensity3", yourFloat);
// Detail Settings ==========
if (m.HasProperty ("_DetailColormap")) m.SetTexture ("_DetailColormap", yourTexture);
if (m.HasProperty ("_DetailColormap")) m.SetTextureScale ("_DetailColormap", yourVector2);
if (m.HasProperty ("_DetailColormap")) m.SetTextureOffset ("_DetailColormap", yourVector2);
if (m.HasProperty ("_DetailColormapIntensity")) m.SetFloat ("_DetailColormapIntensity", yourFloat);
if (m.HasProperty ("_DetailNormalmap")) m.SetTexture ("_DetailNormalmap", yourTexture);
if (m.HasProperty ("_DetailNormalmapIntensity")) m.SetFloat ("_DetailNormalmapIntensity", yourFloat);
// Water Settings ==========
if (m.HasProperty ("_WaterColorAColorBlend")) m.SetColor ("_WaterColorAColorBlend", yourColor);
if (m.HasProperty ("_WaterSpecGloss")) m.SetColor ("_WaterSpecGloss", yourColor);
if (m.HasProperty ("_WaterNormalmap")) m.SetTexture ("_WaterNormalmap", yourTexture);
if (m.HasProperty ("_WaterNormalmap")) m.SetTextureScale ("_WaterNormalmap", yourVector2);
if (m.HasProperty ("_WaterNormalmap")) m.SetTextureOffset ("_WaterNormalmap", yourVector2);
if (m.HasProperty ("_WaterBlend")) m.SetFloat ("_WaterBlend", yourFloat);
if (m.HasProperty ("_WaterWaves")) m.SetFloat ("_WaterWaves", yourFloat);
if (m.HasProperty ("_WaterWaveSpeed")) m.SetFloat ("_WaterWaveSpeed", yourFloat);
// Fog Settings ==========
if (m.HasProperty ("_OverlayFogAmount")) m.SetFloat ("_OverlayFogAmount", yourFloat);
if (m.HasProperty ("_OverlayFogColorAfromAmbient")) m.SetColor ("_OverlayFogColorAfromAmbient", yourColor);
if (m.HasProperty ("_OverlayFogAmountFromReflCubemap")) m.SetFloat ("_OverlayFogAmountFromReflCubemap", yourFloat);
if (m.HasProperty ("_OverlayFogStartDistance")) m.SetFloat ("_OverlayFogStartDistance", yourFloat);
if (m.HasProperty ("_OverlayFogDistanceTransition")) m.SetFloat ("_OverlayFogDistanceTransition", yourFloat);
if (m.HasProperty ("_OverlayFogStartHeight")) m.SetFloat ("_OverlayFogStartHeight", yourFloat);
if (m.HasProperty ("_OverlayFogHeightTransition")) m.SetFloat ("_OverlayFogHeightTransition", yourFloat);
if (m.HasProperty ("_OverlayFogDistance2Height")) m.SetFloat ("_OverlayFogDistance2Height", yourFloat);
if (m.HasProperty ("_OverlayFogEmissivePunchThru")) m.SetFloat ("_OverlayFogEmissivePunchThru", yourFloat);
// Snow Settings ==========
if (m.HasProperty ("_SnowAmount")) m.SetFloat ("_SnowAmount", yourFloat);
if (m.HasProperty ("_SnowColor")) m.SetColor ("_SnowColor", yourColor);
if (m.HasProperty ("_SnowSpecGloss")) m.SetColor ("_SnowSpecGloss", yourColor);
if (m.HasProperty ("_SnowHeight")) m.SetFloat ("_SnowHeight", yourFloat);
if (m.HasProperty ("_SnowHeightTransition")) m.SetFloat ("_SnowHeightTransition", yourFloat);
if (m.HasProperty ("_SnowSlopeDamp")) m.SetFloat ("_SnowSlopeDamp", yourFloat);
if (m.HasProperty ("_SnowOutputColorBrightness2Coverage")) m.SetFloat ("_SnowOutputColorBrightness2Coverage", yourFloat);
// Displacement Settings ==========
if (m.HasProperty ("_Parallax")) m.SetFloat ("_Parallax", yourFloat);
if (m.HasProperty ("_ReduceByVertexAlpha")) m.SetFloat ("_ReduceByVertexAlpha", (float)yourBool);
if (m.HasProperty ("_ReduceByUVBorder")) m.SetFloat ("_ReduceByUVBorder", (float)yourBool);
if (m.HasProperty ("_ReduceByUVBorderLength")) m.SetFloat ("_ReduceByUVBorderLength", yourFloat);
if (m.HasProperty ("_ParallaxMap")) m.SetTexture ("_ParallaxMap", yourTexture);
if (m.HasProperty ("_EdgeLength")) m.SetFloat ("_EdgeLength", yourFloat);
// Cliff Settings ==========
if (m.HasProperty ("_CliffColormap")) m.SetTexture ("_CliffColormap", yourTexture);
if (m.HasProperty ("_CliffColormap")) m.SetTextureScale ("_CliffColormap", yourVector2);
if (m.HasProperty ("_CliffColormap")) m.SetTextureOffset ("_CliffColormap", yourVector2);
if (m.HasProperty ("_CliffEmission")) m.SetFloat ("_CliffEmission", (float)yourBool);
if (m.HasProperty ("_CliffNormalmap")) m.SetTexture ("_CliffNormalmap", yourTexture);
```

Shader Keywords limit in Unity

In Unity 4.6 exists a shader Keywordlimit and if you are getting an error message saying:
Maximum number (64) of shader keywords exceeded, ...
It means that all shaders in your project together use more than 64 Keywords. This is annoying but not a big problem, at least not with Horizon[ON]. If you didn't get this error before you imported Horizon[ON], you can solve it by defining the features. **The steps to do this are the same as in the following chapter.** You have to do it on all of the 7 shaders that come with Horizon[ON]. The downside is that you are not able anymore to enable or disable the features that you have defined in the inspector. Using the checkboxes will not have any effect.

Modifying the shadercode to reduce buildsize (Unity 4.6)

One drawback of the multi-compile shaders is that there are thousands of shader variants precompiled and all of them will be included in your builds. Of course we can not provide you with thousands of shader variants to choose from either. So, if you are satisfied with your results and you know which features are needed in your final production, you have to remove some keywords from the used shaders, to make sure only what you need is actually included in your build. If you are not a programmer and you feel overwhelmed just by looking at shadercode, don't let it scare you, it's very simple to do.

First thing you have to do is to analyze which shaders you are actually using, for example if you only use the "Horizon[ON] Transition" Material and no others you only need to care about modifying one shader: Horizon[ON] Transition.shader. Next thing you want to check is, which features dou you actually use, you can do that by simply looking at the features section of Horizon[ON] Master or directly at the materials.

You'll find the shaders here: Assets\Horizon[ON]\Sources\Shaders\...
There are 7 shaders that you may need to edit:

- **Horizon[ON] TransitionTerrain.shader**
- **Horizon[ON] TransitionSky.shader**
- **Horizon[ON] Flat.shader**
- **Horizon[ON] Object.shader**
- **Horizon[ON] Cliff.shader**
- **Horizon[ON] Displacement.shader**
- **Horizon[ON] Tesselation(DX11).shader**

The place to change the multi compile options is in all shaders the same, look for line 185:

```
185            // Enable/Disable Features here! ============================================================
186            #pragma multi_compile _LAYERCOUNT_ONE _LAYERCOUNT_TWO _LAYERCOUNT_THREE _LAYERCOUNT_FOUR
187            #pragma multi_compile _IBLDIFF_AMBIENT _IBLDIFF_CUBEMAP _IBLDIFF_SKYSHOPSH _IBLDIFF_UNITYSH
188            #pragma multi_compile _DIRECTSPEC_OFF _DIRECTSPEC_ON
189            #pragma multi_compile _IBLSPEC_OFF _IBLSPEC_ON
190            #pragma multi_compile _NORMALMAPS_OFF _NORMALMAPS_ON
191            #pragma multi_compile _EMISSIVENESS_OFF _EMISSIVENESS_ON
192            #pragma multi_compile _DETAIL_OFF _DETAIL_ON
193            #pragma multi_compile _WATER_OFF _WATER_ON
194            #pragma multi_compile _OVERLAYFOG_OFF _OVERLAYFOG_ON
195            #pragma multi_compile _SNOW_OFF _SNOW_ON
196            // ============================================================================================
197            // Backup of the original multicompile options! ===========================================
198 //           #pragma multi_compile _LAYERCOUNT_ONE _LAYERCOUNT_TWO _LAYERCOUNT_THREE _LAYERCOUNT_FOUR
199 //           #pragma multi_compile _IBLDIFF_AMBIENT _IBLDIFF_CUBEMAP _IBLDIFF_SKYSHOPSH _IBLDIFF_UNITYSH
200 //           #pragma multi_compile _DIRECTSPEC_OFF _DIRECTSPEC_ON
201 //           #pragma multi_compile _IBLSPEC_OFF _IBLSPEC_ON
202 //           #pragma multi_compile _NORMALMAPS_OFF _NORMALMAPS_ON
203 //           #pragma multi_compile _EMISSIVENESS_OFF _EMISSIVENESS_ON
204 //           #pragma multi_compile _DETAIL_OFF _DETAIL_ON
205 //           #pragma multi_compile _WATER_OFF _WATER_ON
206 //           #pragma multi_compile _OVERLAYFOG_OFF _OVERLAYFOG_ON
207 //           #pragma multi_compile _SNOW_OFF _SNOW_ON
208            // ============================================================================================
```

As you can see i have put a copy of the original code there and commented it out as a backup. You dont have to fear to screw something up and you dont need to backup the shader files.

Changing “#pragma multi_compile” to “#define”

All you need to do is to open the shaderfiles in MonoDevelop and go to line 185. Remove the keywords that are not needed. For example if dont use emission you can remove `_EMISSIVENESS_ON` , if you dont use fog you can remove `_OVERLAYFOG_ON` and so on.
After you have done this for all features, you change `#pragma multi_compile` to `#define` . That is all you have to do. Save the shaders and go back to unity. If you have done this correctly you should not see any warnings or errors. Restart Unity if the errors didn’t go away.

Here is an example how it could look like in context:

Before:

```
185  // Enable/Disable Features here! ==========
186  #pragma multi_compile _LAYERCOUNT_ONE _LAYERCOUNT_TWO _LAYERCOUNT_THREE _LAYERCOUNT_FOUR
187  #pragma multi_compile _IBLDIFF_AMBIENT _IBLDIFF_CUBEMAP _IBLDIFF_SKYSHOPSH _IBLDIFF_UNITYSH
188  #pragma multi_compile _DIRECTSPEC_OFF _DIRECTSPEC_ON
189  #pragma multi_compile _IBLSPEC_OFF _IBLSPEC_ON
190  #pragma multi_compile _NORMALMAPS_OFF _NORMALMAPS_ON
191  #pragma multi_compile _EMISSIVENESS_OFF _EMISSIVENESS_ON
192  #pragma multi_compile _DETAIL_OFF _DETAIL_ON
193  #pragma multi_compile _WATER_OFF _WATER_ON
194  #pragma multi_compile _OVERLAYFOG_OFF _OVERLAYFOG_ON
195  #pragma multi_compile _SNOW_OFF _SNOW_ON
196  // ==========
```

After:

```
185  // Enable/Disable Features here! ==========
186  #define _LAYERCOUNT_ONE
187  #define _IBLDIFF_AMBIENT
188  #define _DIRECTSPEC_OFF
189  #define _IBLSPEC_OFF
190  #define _NORMALMAPS_OFF
191  #define _EMISSIVENESS_OFF
192  #define _DETAIL_OFF
193  #define _WATER_OFF
194  #define _OVERLAYFOG_OFF
195  #define _SNOW_OFF
196  // ==========
```

Difference between Unity 4.6 and Unity 5 and beyond:

Basically it is just the same thing but in Unity 5 the limit of keywords is 128 (256 for Unity 2017) instead of 64 and instead of “#pragma multi_compile”, “#pragma shader_feature” is used. That means you are much less likely to run into issues because of exceeding the limit. **It is not necessary to do the steps to reduce your build size.** The unity 5 version of Horizon[ON] is using the “#pragma shader feature“ and the benefit is that unused shader variants will not be included into the build.

However, if you get a warning saying:
Maximum number (128) of shader keywords exceeded, ...
You have to do the same steps as described above but the section you need to change in the shaders would look like this:

```
185  // Enable/Disable Features here! ==========
186  #pragma shader_feature _LAYERCOUNT_ONE _LAYERCOUNT_TWO _LAYERCOUNT_THREE _LAYERCOUNT_FOUR
187  #pragma shader_feature _IBLDIFF_AMBIENT _IBLDIFF_CUBEMAP _IBLDIFF_SKYSHOPSH _IBLDIFF_UNITYSH
188  #pragma shader_feature _DIRECTSPEC_OFF _DIRECTSPEC_ON
189  #pragma shader_feature _IBLSPEC_OFF _IBLSPEC_ON
190  #pragma shader_feature _NORMALMAPS_OFF _NORMALMAPS_ON
191  #pragma shader_feature _EMISSIVENESS_OFF _EMISSIVENESS_ON
192  #pragma shader_feature _DETAIL_OFF _DETAIL_ON
193  #pragma shader_feature _WATER_OFF _WATER_ON
194  #pragma shader_feature _OVERLAYFOG_OFF _OVERLAYFOG_ON
195  #pragma shader_feature _SNOW_OFF _SNOW_ON
196  // ==========
```

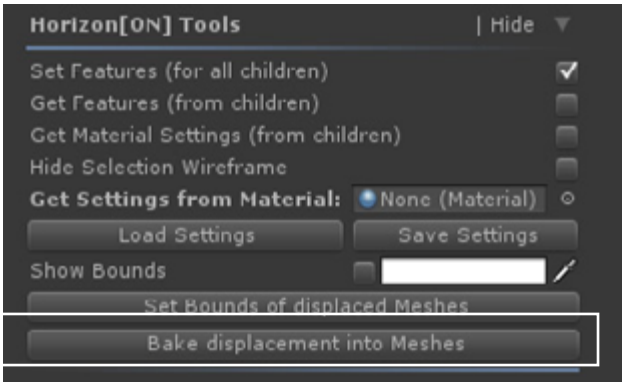
So, in case of Unity 5, you change `#pragma shader_feature` to `#define` and remove the unneeded Keywords. That is all.

New in Version 1.2 - The Deco Painter script

This is a script that makes it easy to place trees and buildings on the Horizon[ON] elements. It is not the most advanced prefab painter in the world but it should be enough to get you started and to get you an idea of how decorating Horizon[ON] can be done. There are tons of assets on the store that do similar things and if you dont find this script to be enough for you needs please consider to get a more advanced one. If there is high demand for this (which i doubt) i might release a more general and improved version of this deco painter as a separate product. for now you have to live with the limited functionality or extend it on your own, or purchase an asset that is specifically made for the purpose of painting prefabs.

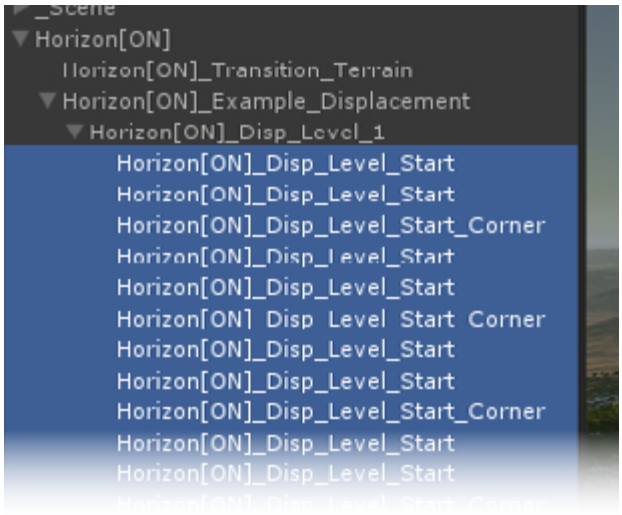
Anyway, here is a brief introduction but first we need to quickly take care of something else: colliders

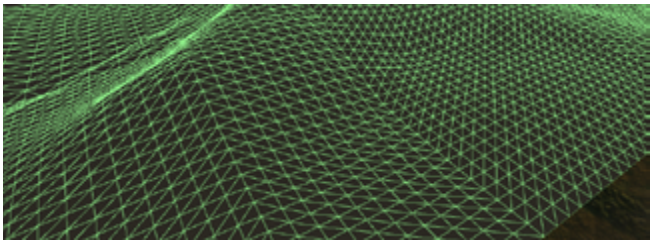
For painting to work you need to have collision meshes. The easiest way to get collision meshes is to just duplicate your Horizon[ON] parent. Rename it to something like "Colliders" then go into the tools section of its Horizon Master and clik on the "Bake displacement into Meshes" button.



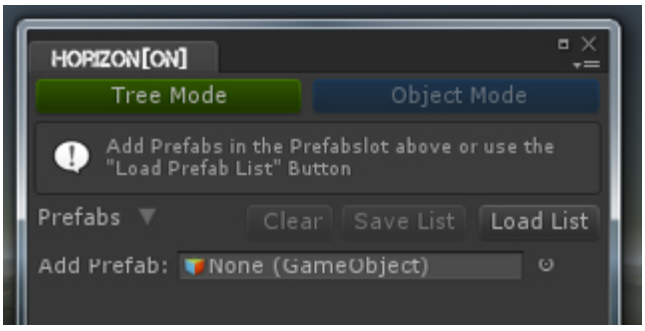
You can now delete all unneccesary objects that do not actually add anything to the collision, like the transition meshes... now select all objects that you want to have collision(the object that you want to paint on).

Put them directly under the parent which we named "colliders" and delete all unnecesary objects. Also we don't need the "Horizon Master" component on it anymore so we can remove it. Now we select all the children again and add a MeshCollider component to them. Next step is to remove all Meshrenderer components and Meshfilter components as we dont need them for collision. Now we have a good basis to paint on:



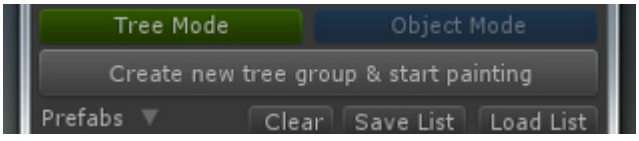


If you go to Window/Horizon[ON]/Horizon[ON] Deco Painter in the menu bar of unity, the Deco Painter window will open.



As you can see it has 2 general modes. One for trees one for buildings(could be used for other objects as well). The first thing you need to do is to add the prefabs you wan to paint. Included with Horizon[ON] are 24 tree variations and 20 building variations. You have to drag them in one by one. but once you have done this you can save them as a list for later use. Alternaitvely you can load one of the lists that come with Horizon[ON]. If tree mode is selceted you can only load tree lists and the same is true for objects.

Once you added your first prefab you can see that the notification disappears and a button takes its place:



Basically it is ready to paint now. Lets quickly clear the list if you have already added something and click on "Load List". Navigate to the Presets folder and select the "Horizon[ON] DecoPainterPrefabList_Trees_A.prefab"

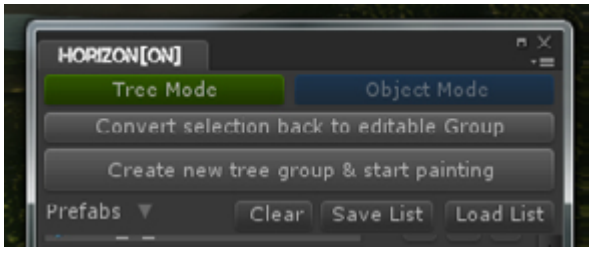
Lets try the painting. Click on "Create new tree group & start painting", you can see a new empty Gameobject is created called "TreeGroup (editing!)" also you can see that the interface has changed a bit:

There are some Infos on how many vertices/triangles/ objects you have currently placed in this group. Also there are some painting options, they are pretty self explanatory and it is absolutely no problem that you just experiemnt with them...

Click and drag with the mouse over the horizon. you can see how treess are added and that the trees are not regular bilboards as you might expect but they are pseudo billboards... they always look towards the scene origin, since the player will never walk on Horizon(at least not if you are using it in the intended way), this is perfectly enough and **saves a lot of performance**. Click now on "Close current treegroup and combine".

A notification will popup which tells you about the vertcount/trianglecount and all the trees you just painted will be cobined into a single mesh. With the trees that come with Horizon[ON] there are 24 tree variations combined into a single texture atlas. The trees are made out of 6 vertices giving a bit better shape and also less overdraw. Since 10k+ Trees can be combined into a single mesh and there is no billboarding happening you can expect best possible performance even with very dense forests.

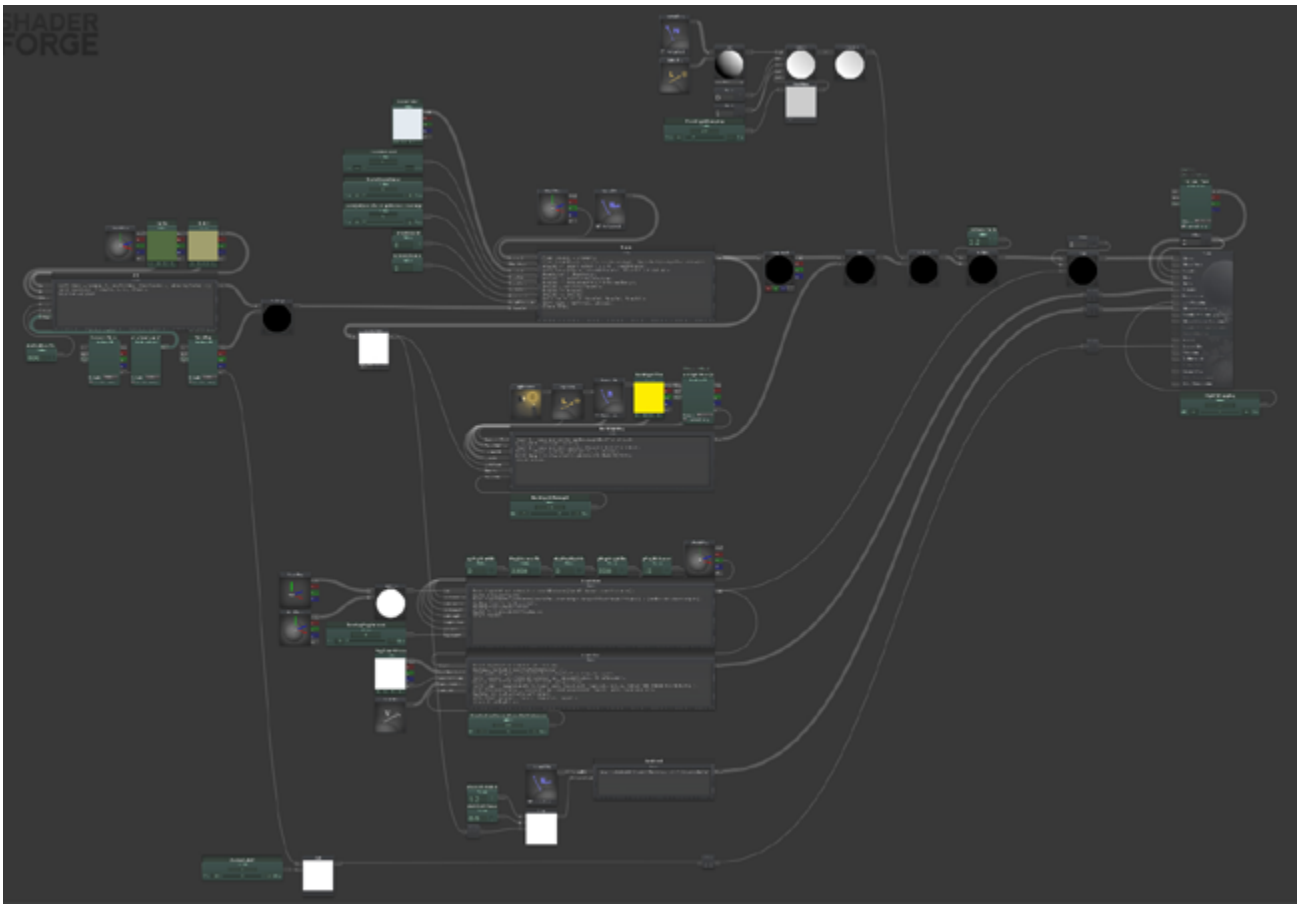
If you need to convert the combined treegroup back to an editable group, you can do so be selecting the object in the hierarchy and then clicking the "Convert selection back to editable Group" button.



In Object Mode, it is pretty much the same as in tree mode, though you have to place buildings one by one. Scaling them up and orienting them with your mouse drag.

New in Version 1.2 - ShaderForge Example shaders

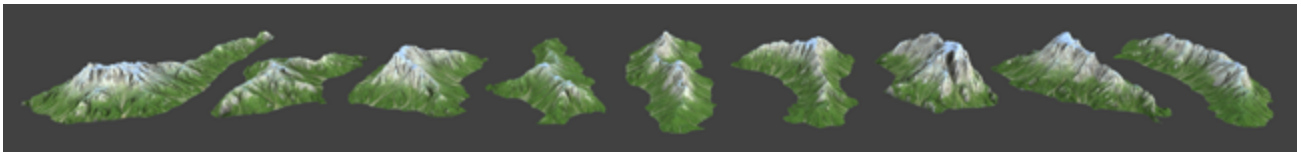
The Shaders for the Buildings and Trees react to Snow and Fog, they use the same property names as the Horizon[ON] shaders. That means you can steer the snow and fog parameters with the Horizon Master and you can make a controller script that controls everything in runtime(look for the included script "HorizonControllerExample.cs"). These shaders are made with ShaderForge and provide you a good example of how you can make your Shaderforge shaders work with Horizon[ON].



The Layout is nice and tidy so you can get a quick overview how things are done...

New in Version 1.4:

Undo support added.
Further optimized shaders.
Normal seams between meshes after baking displacement are now automatically fixed.
Removed Marmoset Skyshop Support to streamline shaders (it's obsolete now since Unity has all the features built in).
Improved user interface.
Example scene is now available in linear and gamma colorspace.
Tweaked textures and settings of the example scenes.
Improved Building shader
Improved Tree Shader
Added 9 Decoration Mountains
Added Post Processing Profiles for the example scenes
Updated Manual



Making Textures and Masks for Horizon[ON]

Making Textures for the layers:
There are so many different ways to make or generate textures that i can only suggest some. Generally you have to think of Horizon[ON] the same way as you would think of a terrain but in a bigger scale. Instead of putting rock textures, dirt textures or grass textures, etc. on it, we put textures of whole areas on it. So, for example a desert texture, a jungle texture, a city texture, etc. would be suitable. These textures can be generated, drawn, or photographed. The, in my opinion, best way to make such textures, is to use a combination of all these techniques.

Making a Mask for Horizon:
The mask that is included in the examplescene was just painted in photoshop. There is really not much science to it but it definitely helps if you are at least to some extent familiar with photoshop(or another image editing program). For painting i use regular layers in a separate file and i copy the results into the individual channels of the mask file that i use on Horizon[ON]. This way you have more control and more tools availible compared to painting directly into the channels. For example you might want to subtract one layer from another or use a layer as a guide for painting. If you paint directly in the color channels of the mask image this is not possible.

Useful Links

Programs & Tools that might help you to create textures for Horizon[ON]:

WorldComposer: Its a very cool tool for unity, which allows you to grab satelite images from the web very easily. The satelite images are of very high quality and it even has tools to remove shadows or clouds from the images.
https://www.assetstore.unity3d.com/en/#!/content/13238

TerrainComposer: It is very powerful and versatile, you can create unity terrains with the data you get from WorldComposer. You can modify it and even generate fitting normal maps.
https://www.assetstore.unity3d.com/en/#!/content/7065

Worldmachine: Very powerful terrain generator, you can create very realistic terrains with it and output all maps that you need for Horizon[ON].
http://www.world-machine.com

Crazybump: Can be useful to generate normal and heightmaps from images.
http://www.crazybump.com/

Video Tutorials that might help you:

Making textures tileable: This thechnique can be used to make your layer textures and you your mask tileable.
http://youtu.be/4SiYt92RINo

Creating Splatmaps: You can use this or similar techniques to create a mask for Horizon[ON].
http://youtu.be/U-c9FnsXT_s

FAQ's

Q: Does Horizon[ON] work with Unity 5's Lighting?
A: Yes, sure, if you mean U5's spherical harmonic ambient lighting, yes.

Q: Does Horizon work with Enlighten?
A: No, it does not really need to. Terrains and generally outdoor scenes, are pretty much ideal situations for dynamic IBL(Image Based Lighting - Light coming from a skybox). Lightmapping or precalculated lighting on an object of thousands of square km size would not be a good idea anyway. You can however use Enlighten on your closeup terrain(s) and geometry and still use Horizon[ON].

Q: How well does Horizon work with "Time of Day" or other sky assets.
A: Very well, Horizon[ON] has been extensively tested with TOD and it is a perfect combination. Not only produces TOD very good looking skys, it also gives great dynamic lighting on Horizon[ON]. Also it comes with an ImageFX for atmospheric scattering(basically a more realistic fog). I can recommend TOD wholeheartedly.

Q: How does the blending(transition) from my terrain to Horizon[ON] work.
A: There are a few ways to do it. The regular way would be to flatten the terrain on the outside edges and overlay the transition object from Horizon[ON] onto your terrain. In my opinion this is the best option as it is easy to do and gives great visual results. Usually you can not let the player get too close to the edge of the terrain without revealing limited range and the square shape of your environment. With Horizon[ON] you can allow the player to get very close to the edge ad still hold the illusion of an infinite world. There are also other way to do it. In my tests it was perfectly feasible to just place the terrain a bit above Horizon[ON] and do no transition at all. The edge of the terrain will just look like a hill in fron of the horizon. This way the player can still come fairly close to the edge, especially if the player can not take very elevated positons, but even if so it can still work fine. Another method would be just a intersection between your terrain and the geometry of Horizon[ON]. if you match the colors of Horizon[ON] well enough with the colors of your terrain this can work quite well too, especially if you put additional detail like rocks, trees, buildings, roads, etc. over the parts where the intersection occurs. Lastly there is also the possibility to create your own transition mesh and use that but this option is the most tricky one and is usually not needed. Which of the described methods works best in your case, you have to find out on your own as i can not generalize all kind of terrains and the usecases.

Q: Can objects like trees and buildings be placed on Horizon[ON].
A: Sure and i recommend to decorate Horizon[ON] further, especially in proximity of the walkable terrain. Horizon[ON] comes with a building and tree placer script, also there are some trees and buildings included that are very efficient to render(many times more efficient that SpeedTree trees for example). You can also place any other objects on Horizon[ON].

Q: How can i match Horizon[ON]'s built in Fog and Snow with my objects,
A: I included a building, a tree and a mountain shader made with shader forge. You could either use these directly or you can duplicate them and make your own shaders out of it. The included shaders have a node setup that uses the same properties as the Horizon[ON] shaders, that means you can control these shaders and Horizon[ON]'s shaders at the same time and get the same results(Fog Density, Snow Amount, etc.). E.g. When you increase the snow amount it would be done so for all objects on Horizon[ON]. Also there is an asset called "UBER Standard Shader Ultra", i recommend looking into it, It is in my opinion the best and most complete shader solution for Unity. It also allows you to control things like snow with very little effort and very convincing results. With very little scripting you could sync Horizon[ON]'s snow with the snow from UBER.

Q: Does Horizon[ON] use a lot of textures and therefor much memory?
A: No, not neccesarily. While you can crank up the resolution of your textures to the maximum for astonishing results, it is by no means neccesary to do so for a AAA look. Horizon[ON] does utilize tiling textures and detail textures to make the impression of huge textures with much higher resolution than are actually used. You could easily have a very good looking horizon with detail objects and a terrain in the middle and your scene would still fit into a 10mb Webplayer.

Q: Can Horizon[ON] be used in only one direction or does it have to be all around?
A: No, it does not have to surround your terrain. If your player will never look behind a hill for example, i recommend to remove all the geometry behind it. What can never be seen should not exist in your scene.

Q: Is it possible to use Horizon with a mesh based terrain instead of a Unity terrain?
A: Absolutely, the terrain in the example scene is such a case... what you put in the cetern of Horizon[ON] is up to you.

Q: How does the making of a new horizon work, do i paint the heightmap just like with Unity's terrain?
A: No, it is different. Horizon[ON] supports up to 4 layers + water, each of these layers can be described as a whole terrain type(meaning that it has its own heightmap, colormap, normal map, etc.). These layers are then blended together using a 4 channel mask image, which you can draw in photoshop or Gimp.

Q: Ok, so how do i make a new layer then?
A: That depends on your favourite workflow and what you expect from the final look. Possible tools to use are: Worldmachine, TerrainComposer, WorldComposer, ZBrush, MudBox, Photoshop/Gimp... What you need to end up with, is a topological image of a terrain type(e.g. desert) for your new layer. You dont need to make a heightmap if you dont use displacement and you dont need to make a normal map if you never going to enable normal mapping. However if you do use the full feature set of Horizon[ON], you get the best results by making a heightmap and normal map that fits your colormap. If you have your layer finished you should make it tileable so you can get a much higher texel resolution. I personally would recommend to either use world machine to generate the needed maps or to use a satelite data grabbing tool like "WorldComposer". With the satelite data you get a Colormap and a fitting heightmap. This heightmap can be used to generate a normal map from(eg. using CrazyBump, xNormal, etc.... you can also generate a normal map in WorldMachine or Terrain Composer from a heightmap).

Q: How many polygons does Horizon[ON] use?
A: That depends heavily on your case, you can have it ranging from less than 50 triangles, to a few hundred k of triangles. There are many ways to reduce the polycount drastically in a lot of cases you can reduce the triangle count by a factor of 10 or more in the optimizing stage... baking the displacement(given that you use displacement), exporting the meshes, combining them to one and letting a polygon optimizer run over it, is a good idea, especially if you want to squeeze the most performance out of Horizon[ON]. Though horizon is still very fast with a few polys more... Also take note that shadows can impact your stats dramatically.

Troubleshooting

Problem: My Scene is too bright/too dark, what is wrong?
Solution: You can adjust the brightness of Horizon[ON] using the Global Tint ColorPicker in the Main settings tab. It could just be that your project is set to Linear Space in the Project settings and Horizon[ON] is adjusted for Gamma Space, or vice versa. Horizon[ON] can be used both in Gamma Color Space or in Linear Color Space, you just have to adjust for it.

Problem: My Horizon is too jaggy and my mountains look like pyramids!
Solution: Horizon[ON] uses geometry, depending on the tiling of your layer heightmap(s) the geometry might just not be tesselated enough. that could mean that you need to reduce the tiling(increase the size of the layer that causes problems) or you need to use more meshes and scale them down. The example scene is set up for a terrain of 1000m side length, if your terrain is much bigger and you just scale up the geometry the space between the vertices gets too big to display all the detail in your heightmap. Take a look how the example scene is composed out of elements, you can arrange them differently to suit your needs. The solution i recommend is to scale your layers bigger. Horizon[ON] is designed for layer maps that cover kilometers not meters.

Problem: I have Z-Fighting between the terrain transition and the terrain.
Solution: Make sure your transition terrain mesh is not set to static, there is a script called HorizonCompensateZFighting.cs, it is resposible for moving the transition up and down based on the camera height. If the transition is set to static it cant be moved and that leads to z-fighting. Also you might need to change the parameters on the script for your case. Check Page 13 for more details.

Problem: I am on mobile and Horizon[ON] is pink.
Solution: Well, most likely you are on GLES2 and you have too many features enabled... disable some features and try again. GLES2 can only handle 8 textures, that includes cubemaps, so you need to keep that limitation in mind. Also make sure you dont use a displacement shader or a tesselation shader on mobile. For displacement on mobile you can still bake the displacement and switch to the incuded object shader.

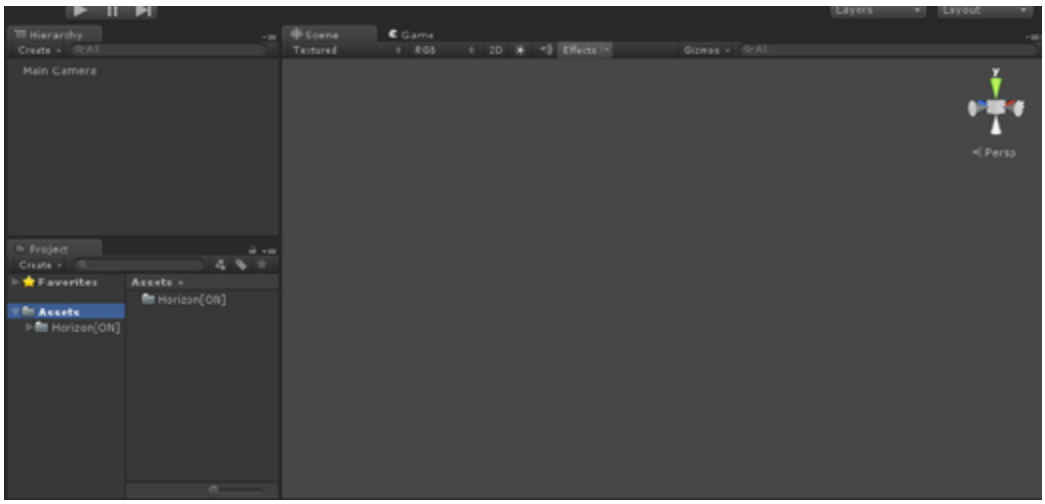
Problem: I am on mobile and the performance is bad.
Solution: Again it proabaly has to do with the amount of features. Make sure you dont go too crazy with the feature count if you are on mobile. Also, current mobiles have much higher screen resolutions than their tiny gpus can handle. Some devices go beyond 2560x1600 which is a bit over the top given the screen size and the gpu power. Using a half resolution still gives good results on such small screens and is much more resonable considering the gpu power of a mobile. Also, you have to be very concious about how many polys you push on a mobile. With Horizon[ON] you have many ways to keep the polycount low. Exporting a baked displacement mesh and reducing it in a 3d program might help or just rely on blend meshes.

Problem: I use Time of Day and its atmospheric scattering ImageFX the transition meshes look wrong or there are seams.
Solution: That is because this effect does not hande object that do not write into depth buffer(e.g. transparent objects). Remove the Skytransition and bring the desitity of the atmospheric scattering effect to a level where the border is just fully covered in fog. For the terrain transition make sure ther is an opaque object right below the transition. You can either make your terrain a b it bigger or just place a plane under it.

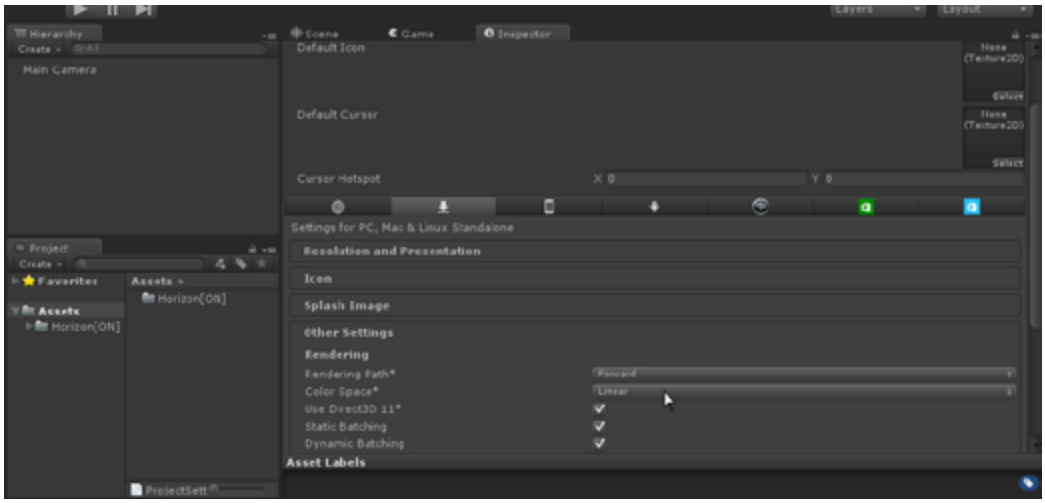
Problem: I have trees and somehow the transparency has problems with the sky transition.
Solution: I have tested it thouroughly and i did not encounter any issues, however it can be that your shaders use anot ideal queue. By default, the transition sky shader is set to “Queue”=“Geometry+501”. Look for this in the transition sky shader and see if raising or lowering this value helps. You might also need to inspect other shaders in your projetct if there is something wrong regarding the queue.

Problem: I have a different problem, what should i do?
Solution: Post about it in the Horizon[ON] thread on the unity forum. I am sure we can sort it out and i might add your problem to this trouble shooting guide later on.

Getting Started



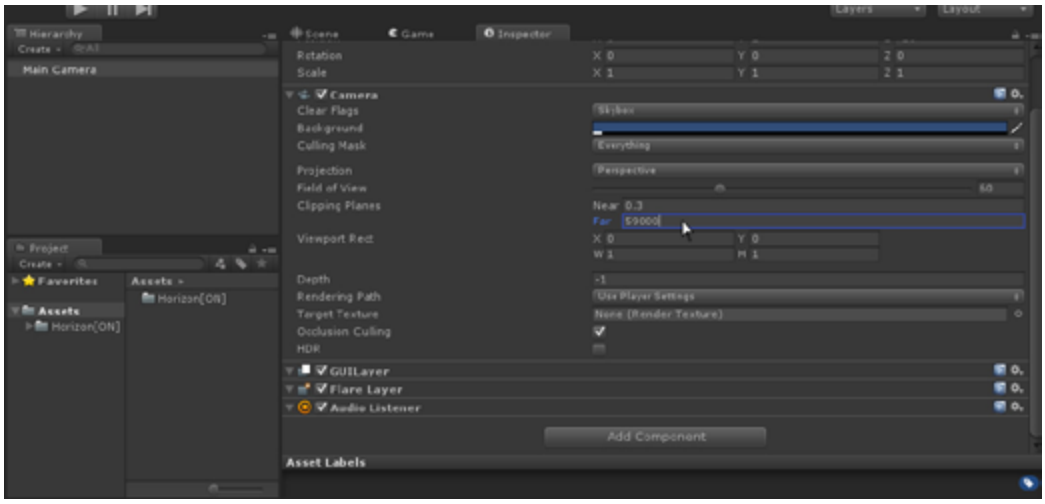
We will start with an empty scene, preferably in a new project(this way we can make sure everything works, if you use an existing project there might be incompatibility issues and the getting started tutorial is not the right place to adress these). First we need to do a few little preparations. Let's go to the Project Settings/Player



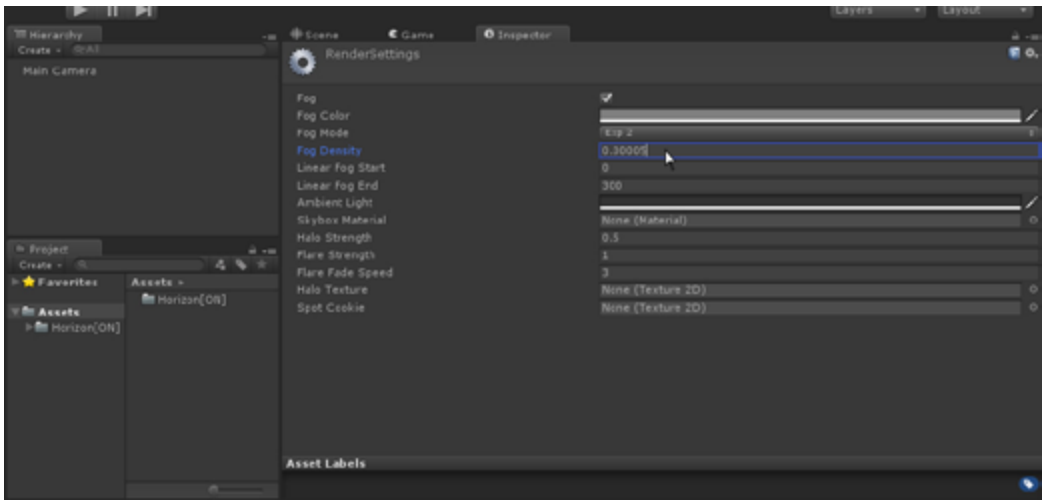
In Horizon[ON] 1.2 and above it is not necessary to switch to linear. The example scene and default material setup has been made for Gamma & Linear Color Space.

Depending on which color space you use, you might need to adjust the "Global Tint" in the Main Settings and eventually tint the layers individually to get the look that you want.

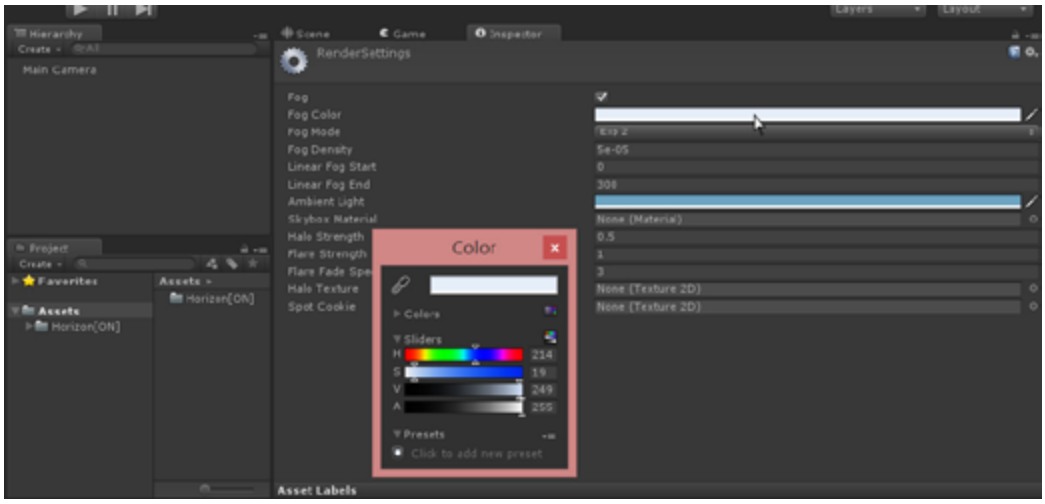
In the Player Settings we will switch the color space to linear. While it is not necessary to work in linear color space it is recommended if you go for a realistic look, as the gamma color space is just wrong in the way how light is treated.



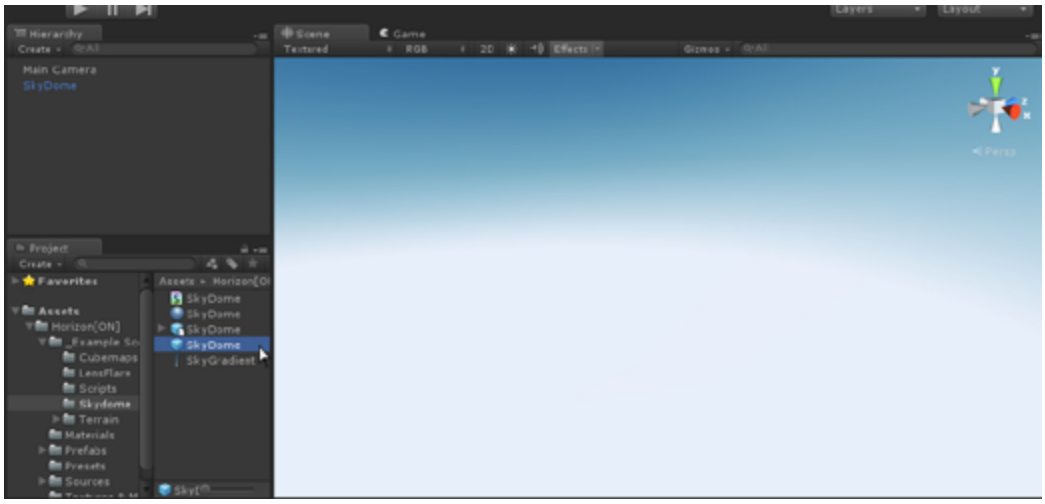
Now that this is done let's select the main camera and set the far clip plane to a pretty high value of 25600. This is of course also depending on your scenes but in general horizon will look most convincing with a high view distance. For the sake of this tutorial we will use this value.



Next we will go to the Render Settings and enable fog, also we set the fog mode to EXP2 and the density to 0.0002.

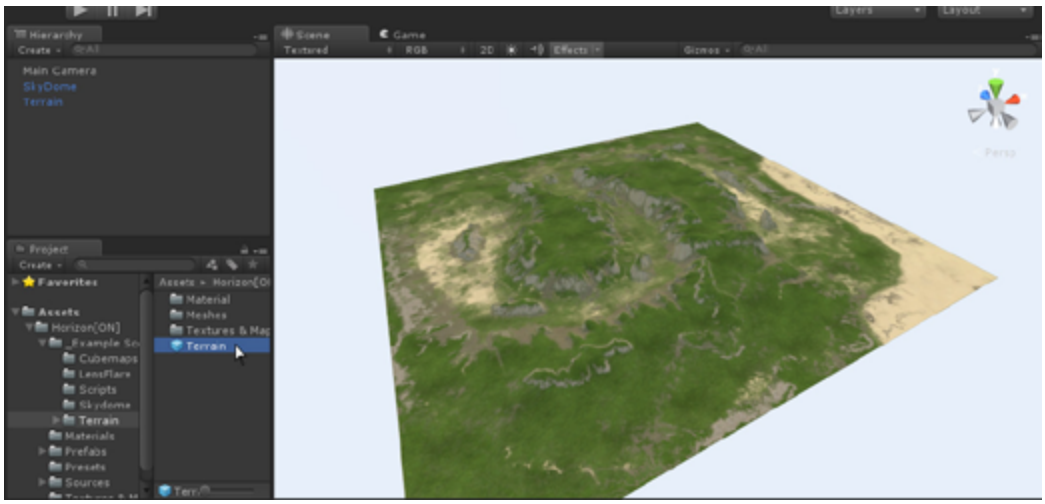


Now we should choose an appropriate fog color. You can switch the color picker to HSV by clicking on the little Icon above the values.We pick a hue of 214, a saturation of 10 and a value of 249.

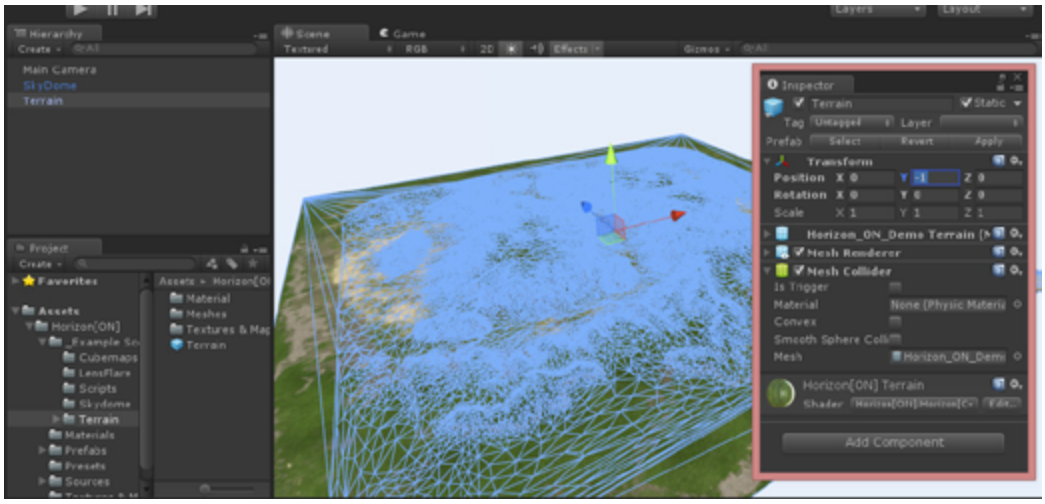


In Horizon[ON] 1.2 and above the skydome prefab has been removed, you can use Unity's default skybox.

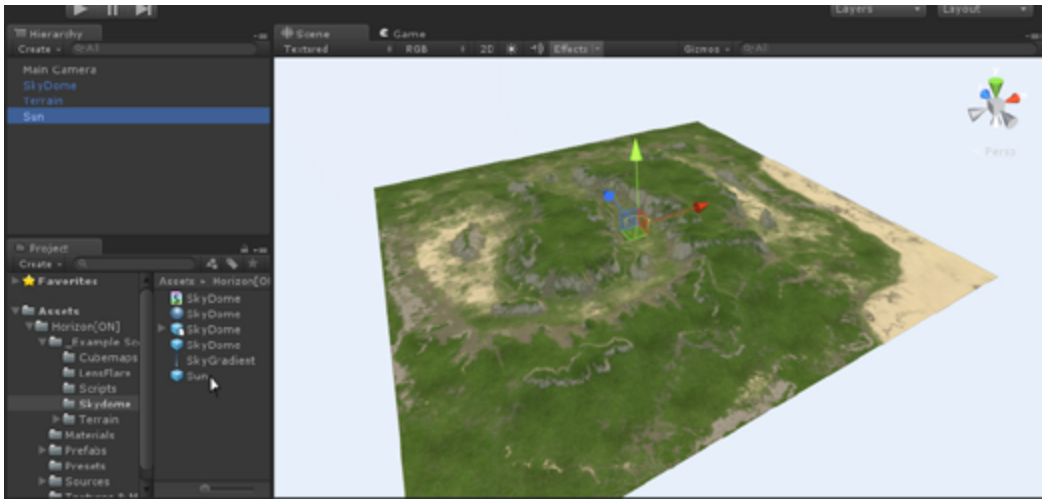
Now we need to add a sky to the scene, In the Folder Horizon[ON]/_Example Scene/Skydome you can find a prefab called „Skydome". Drop it into the hierarchy and make sure its position is 0 ,0 ,0.



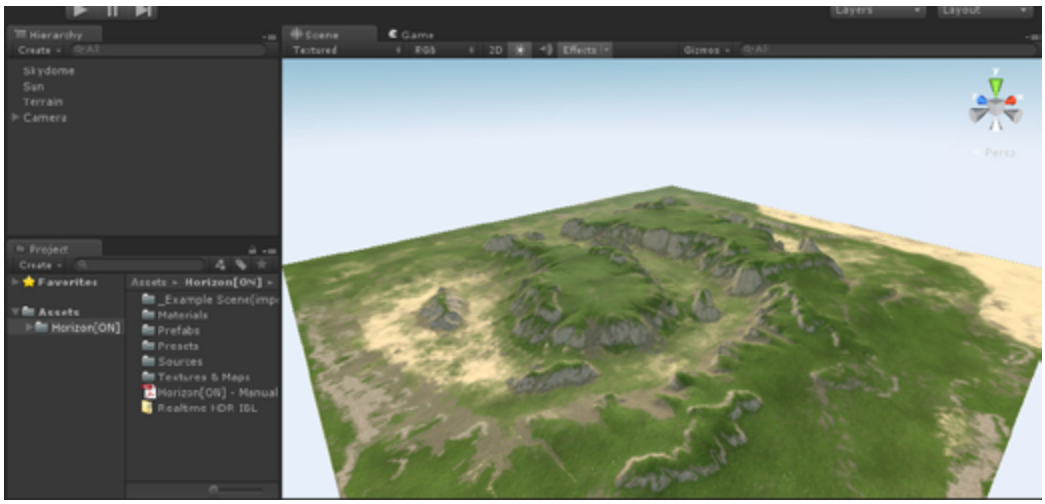
We need a terrain too, so let's add the one from the Folder Horizon[ON]/_Example Scene/Terrain. Just drop the prefab into the hierarchy.



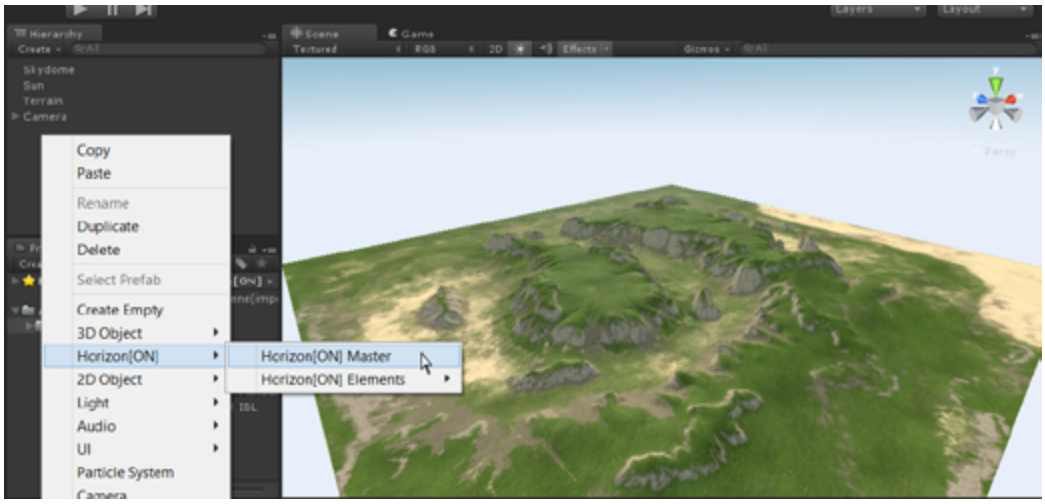
Select the newly added terrain and set its position to (0 , -1 ,0). This is only for now as we want to see the transition from terrain to Horizon[ON] without any z-fighting. In the manual i will explain more on how to solve z-fighting between Horizon[ON] and terrains.



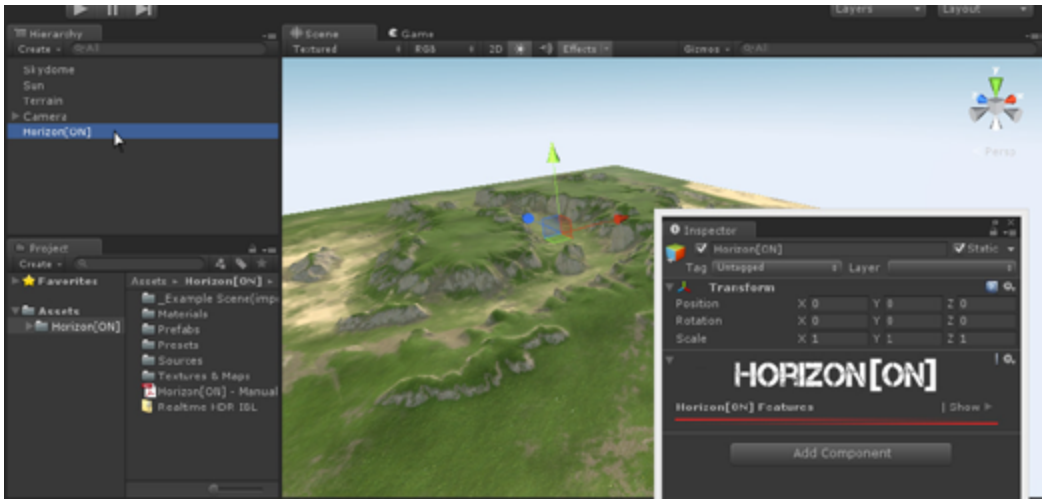
Rename the directional light to "Sun" and add it in the Lighting tab at "Sun Source". Set the intensity at around 1.2 and change the x rotation on the transform to 16.



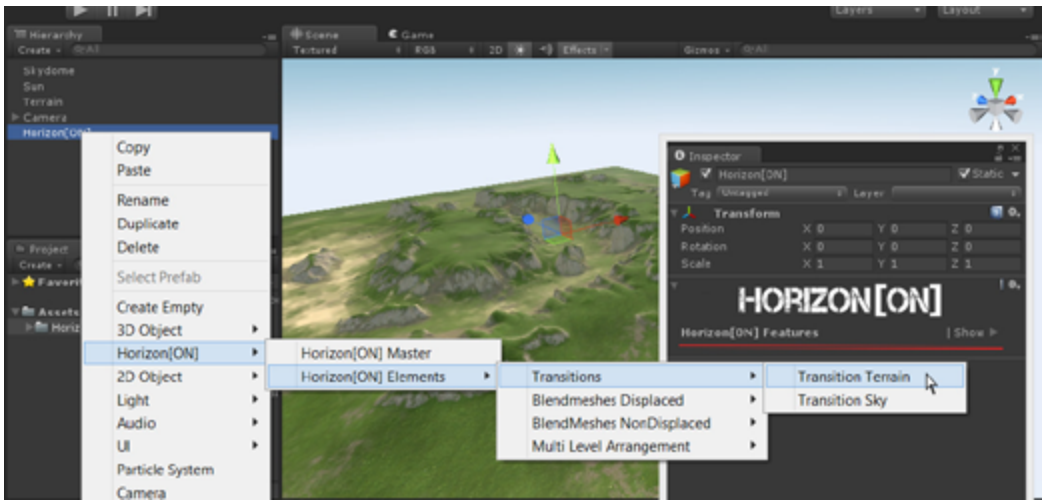
Now we are done with the preparations, your scene should now look similar to this, maybe adjust your scene camera to get a similar viewing angle.



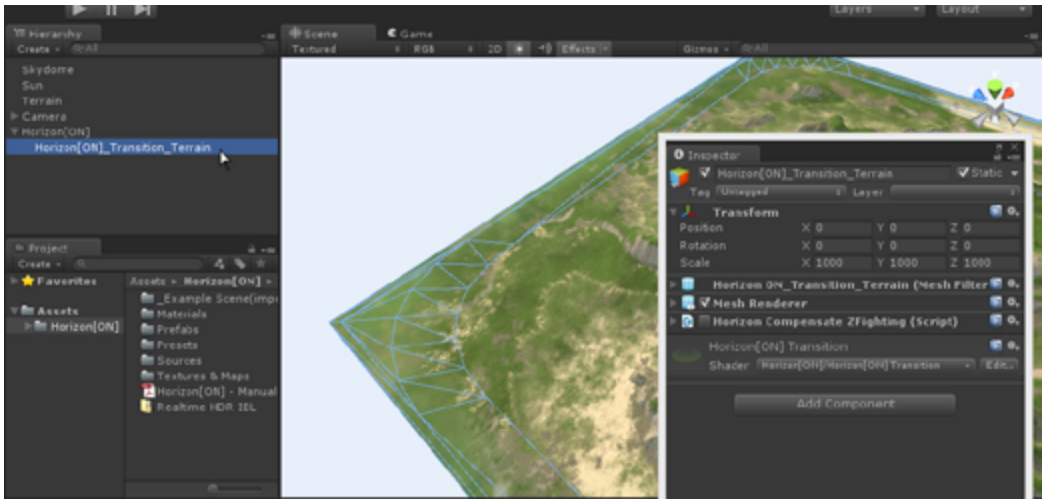
Now we add the Horizon[ON] Master to the scene. Right click on empty space in the hierarchy window and select Horizon[ON]/Horizon[ON] Master.



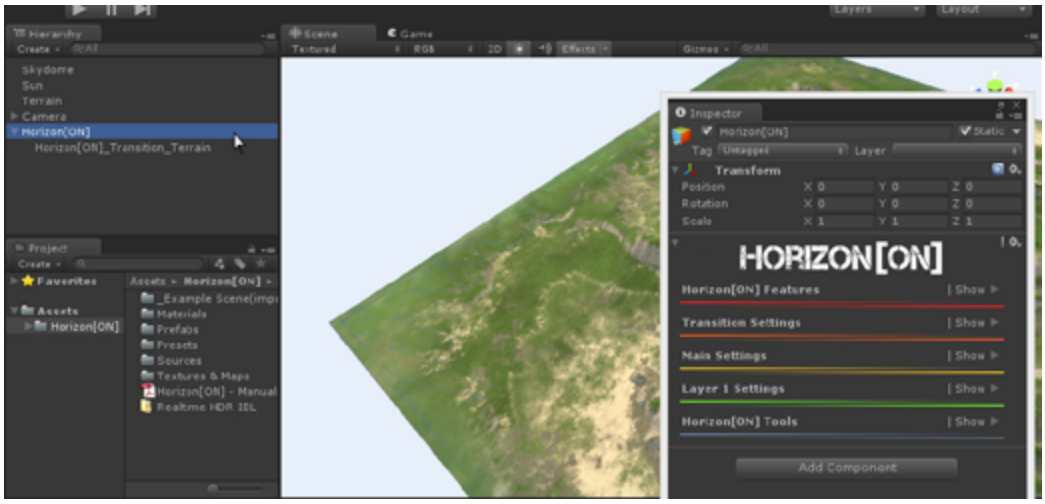
It should be selected now and you can see this is just an empty container at the moment. Horizon Master will help you to manage all the Horizon[ON] Elements. Some of these elements we are going through in this guide. Some of them you'll have to try yourself. There are 4 categories of elements: Transitions, Blendmeshes Displaced, Blendmeshes NonDispaced and Multi Level Arrangements(which contain sub elements).



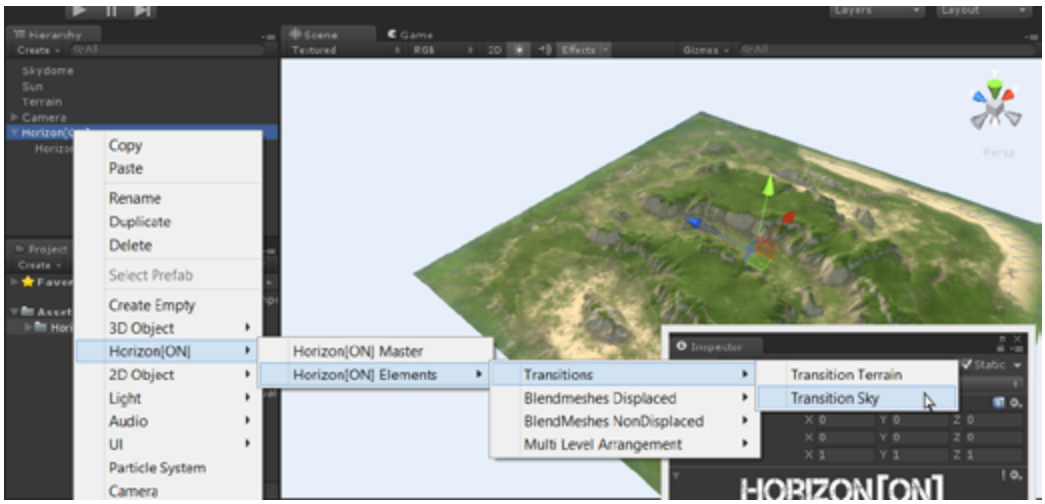
So, we can add the first element. We need a transition from terrain to Horizon[ON]. Right click the Horizon[ON] Master object in the hierarchy(this way the newly created element will automatically be a child of Horizon[ON] Master).
Select: Horizon[ON] / Horizon[ON] Elements / Transitions / Transition Terrain.



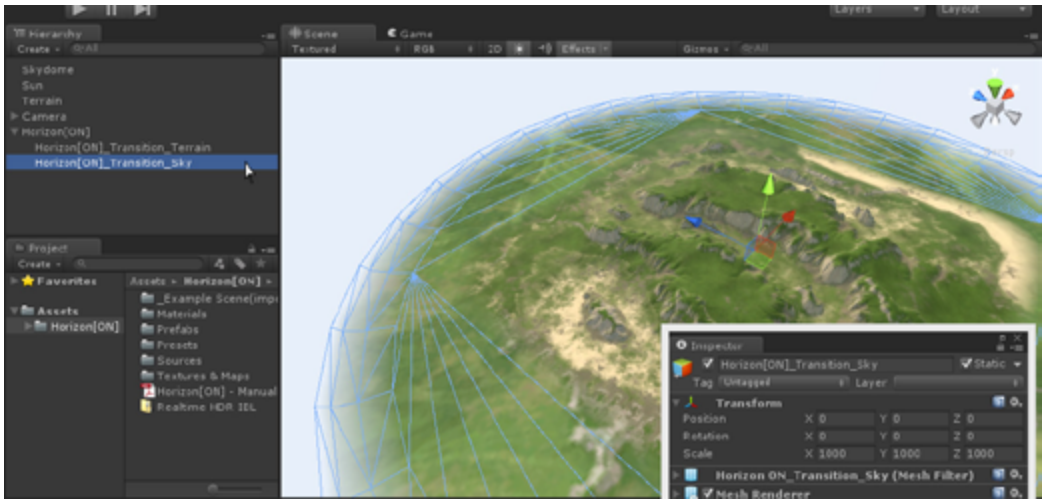
Whenever you add a new element it will automatically be selected. You can also see that the transition object has a disabled script called „Horizon Compensate ZFighting"attached. We ignore that for now, i explain it in the Manual. Now the material of the transition is not yet synced. Once you reselect the Horizon[ON] Master it will take care of its children.



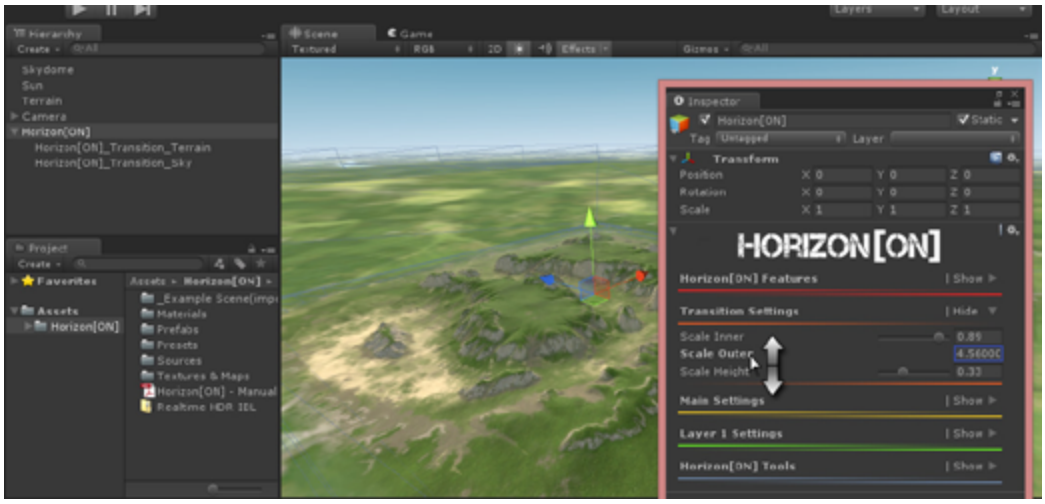
Do that now. Reselect the Horizon Master. As you can see there are now a few more options available. That is because Horizon[ON] Master will check its children for the used materials and expose its parameters depending on which features you enable. This will help you to keep multiple materials/shaders in sync and give you the ability to adjust parameters on multiple shaders at once.



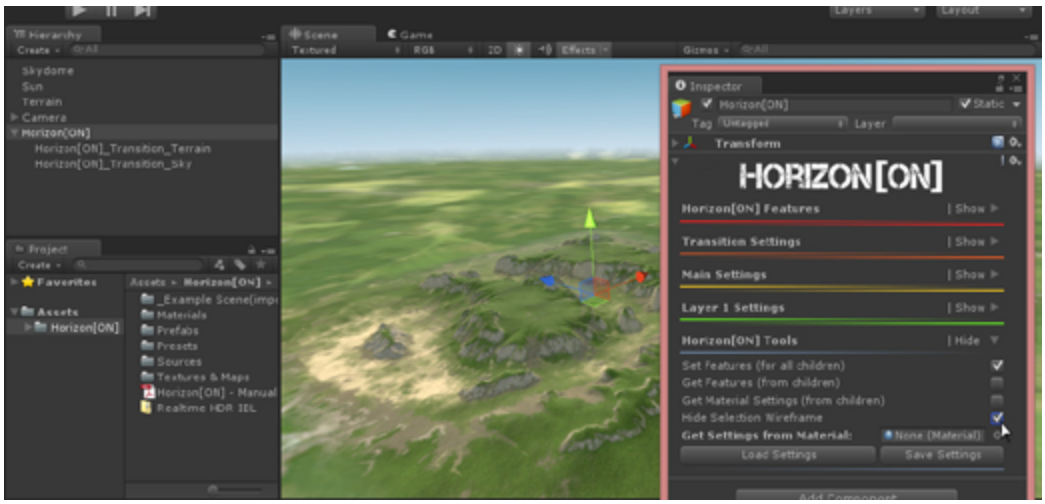
So now that we have placed the first element, let's carry on with the next. Right click on our Horizon[ON] object and... Select: Horizon[ON] / Horizon[ON] Elements / Transitions / Transition Sky.



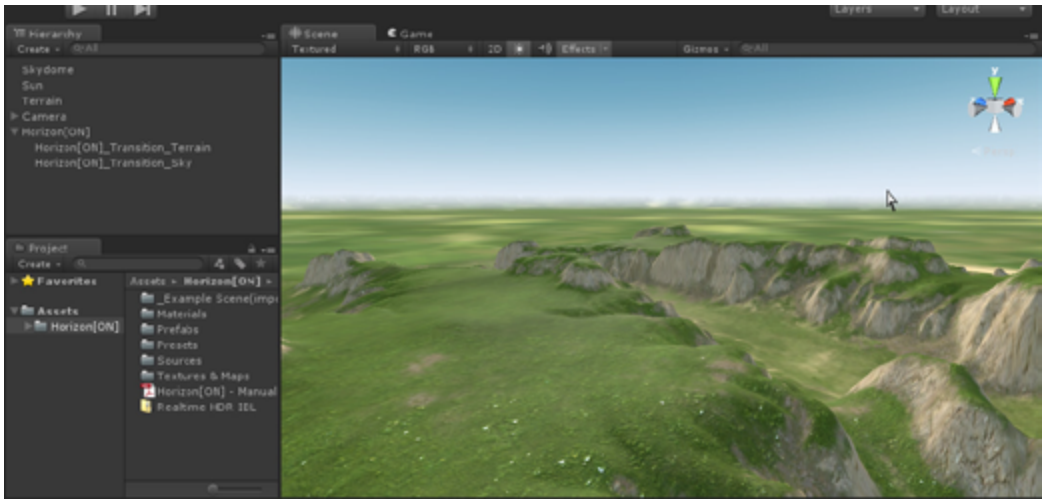
Ok, now our world is not square anymore, that is already a big improvement but it is still not much bigger then our terrain. We will take care of that in the next step. Reselect the parent.



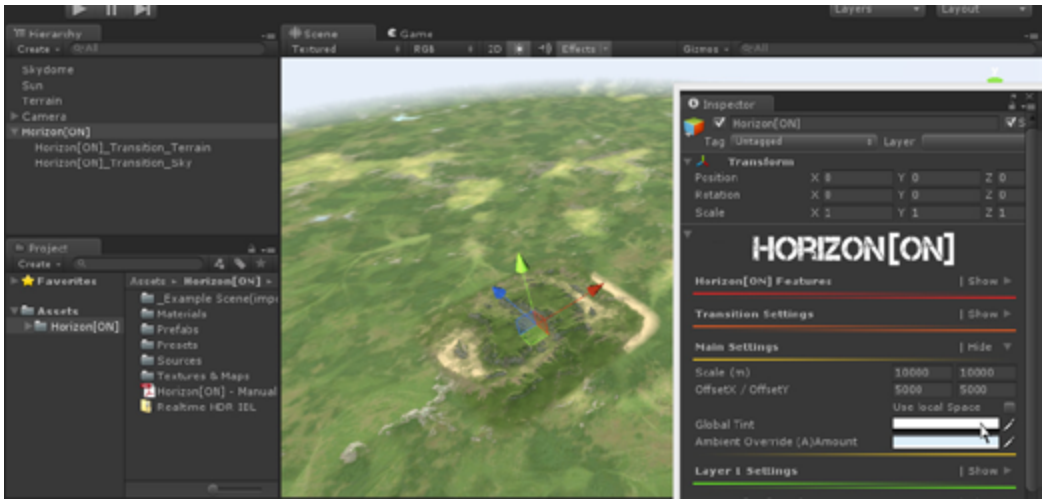
In the Inspector click on the little triangle next to „Transition Settings | Show" to unfold this section. Here are 3 values to adjust, „Scale Inner", „Scale Outer" and Scale Height, play a bit with them to see what they do. As you can see Scale Outer is a float field and has no slider but you can click and drag on the label to adjust its values. This can be done on all float fields that have no slider. When you are familiar with what these values do, adjust it like in the screen above.



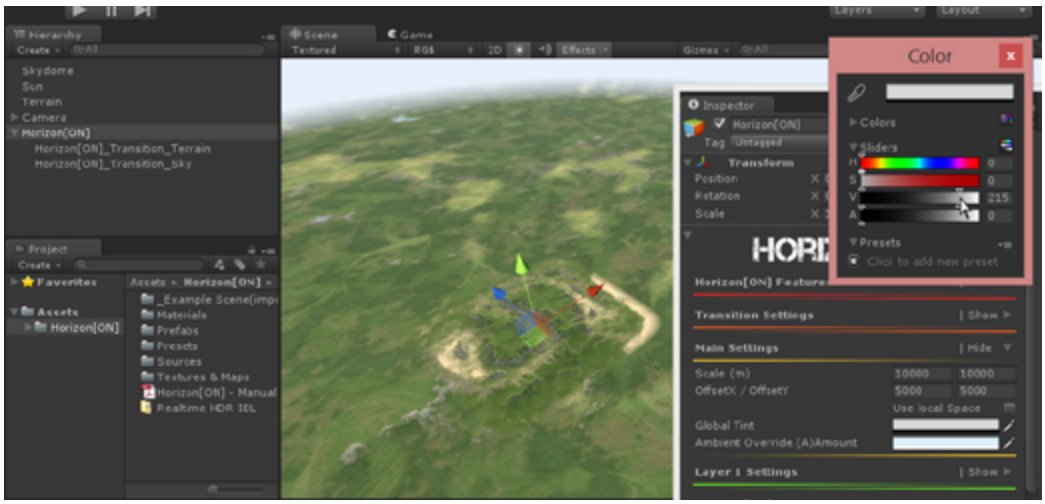
Often the wireframe of the selection is a bit distracting when adjusting material parameters. There is a solution for that problem in the tools section of the Horizon[ON] Master script. Unfold the tools and tick the checkbox next to „Hide Selection Wireframe“. Keep in mind though that this setting will be remembered by the children.



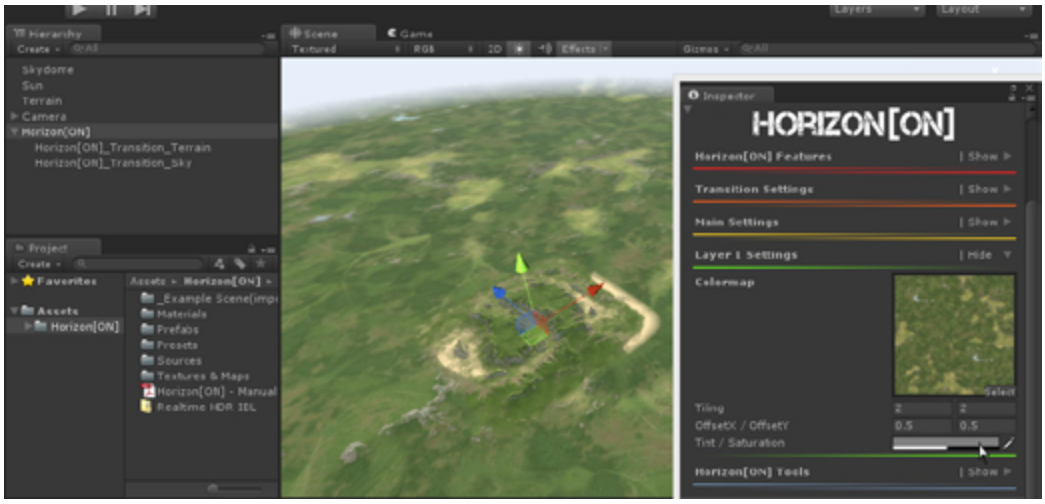
Looks already quite nice. The colors and are roughly matching but we can fine tune that.



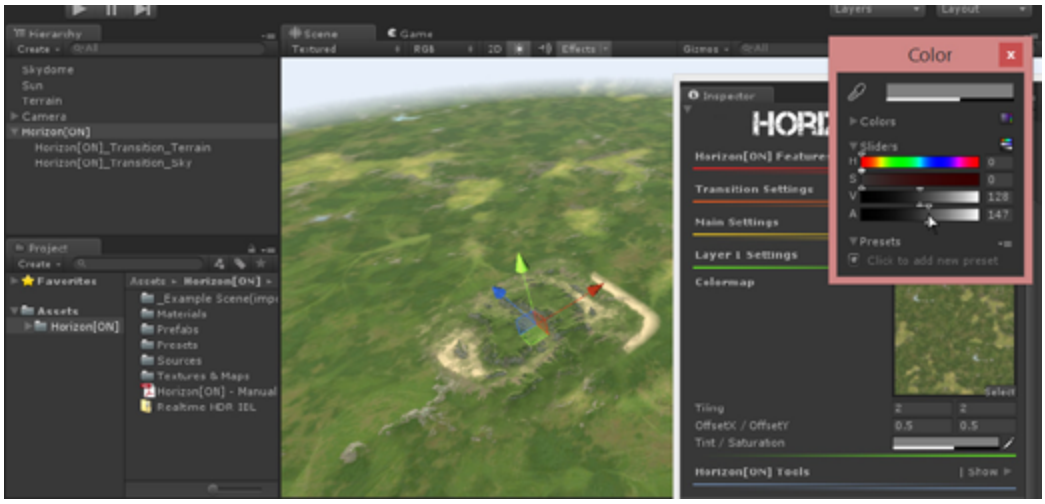
In the „Main Settings“ section click on Global Tint and...



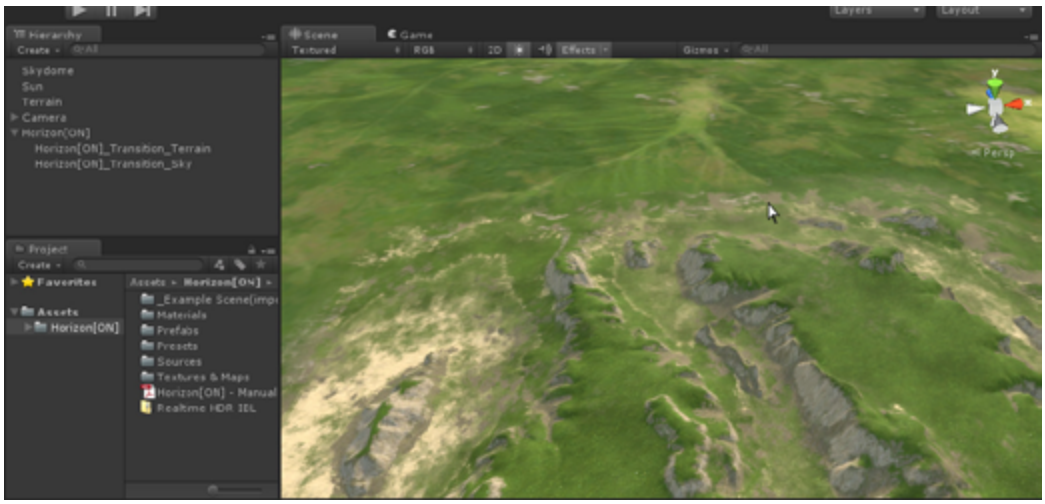
...bring down the value a little bit. Now the brightness is perfectly matching but the horizon is a bit pale compared with the terrain. Let's unfold the „Layer 1 Settings".



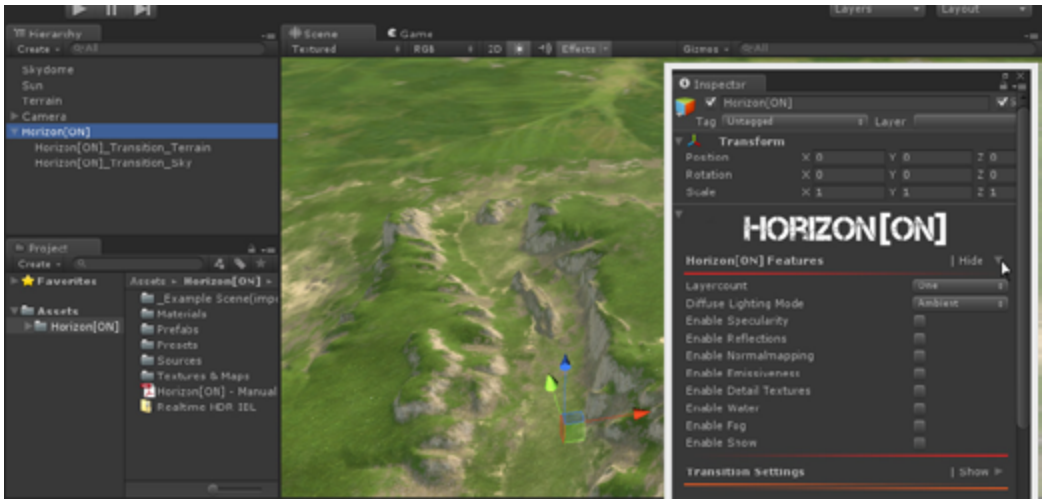
Layers also have their own tint color and they also have a saturation value, which is set by the alpha channel in the color picker.



Let's increase the saturation a little bit.
Great, apart from the sandy corner of the terrain it is almost impossible to tell where the terrain ends.

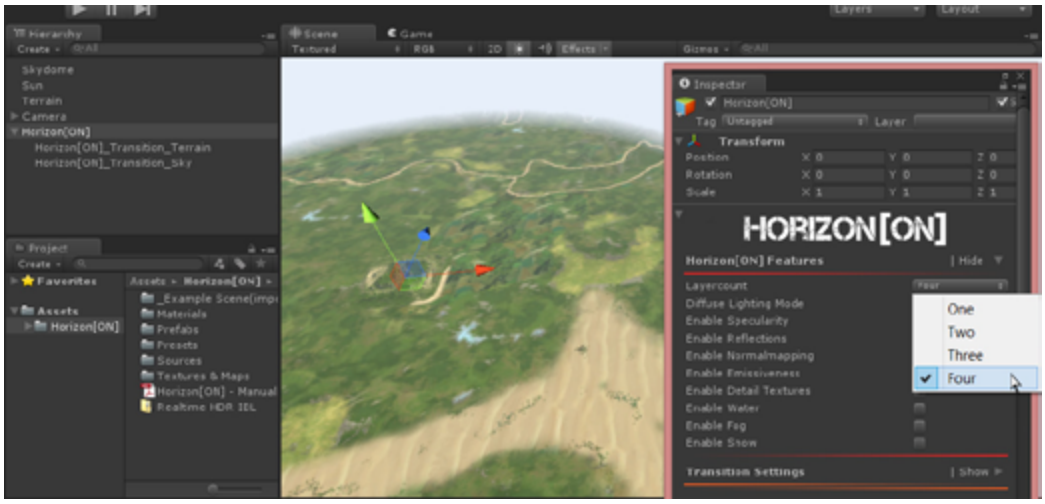


Great match but in close up we see a difference in high resolution detail. But first let's take care of the sandy corner.

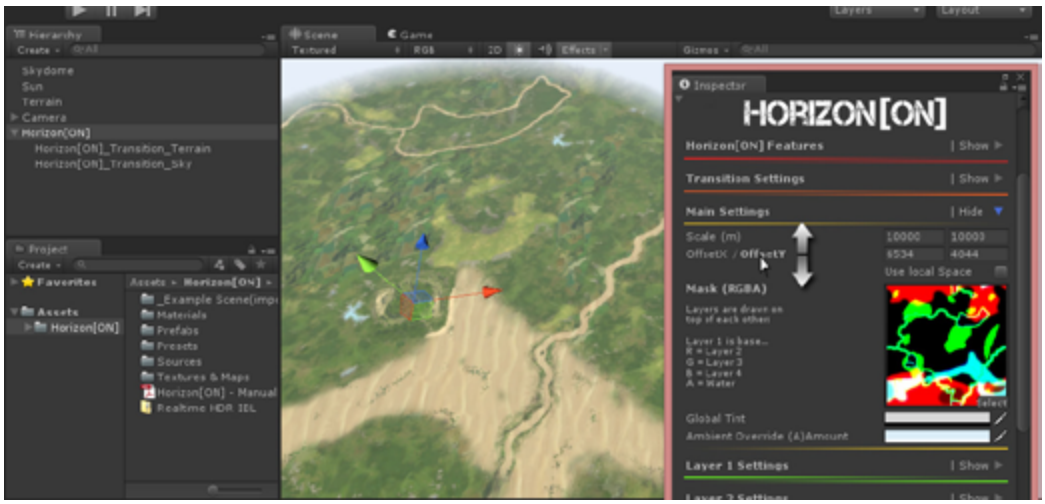


In Horizon[ON] 1.4 the "Diffuse Lighting Mode" enum field has been removed as it is now controlled from unity's lighting tab. Please ignore it in the illustrations.

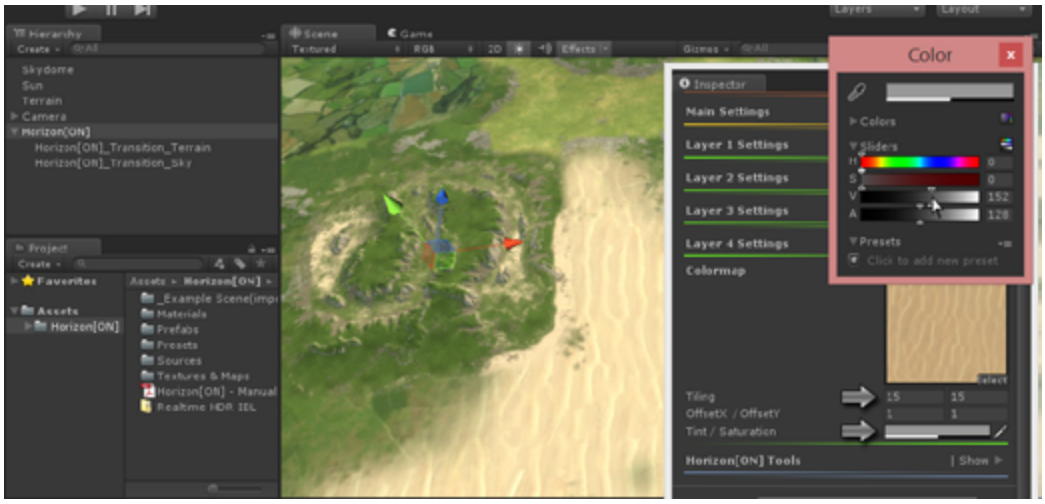
Open the features section.



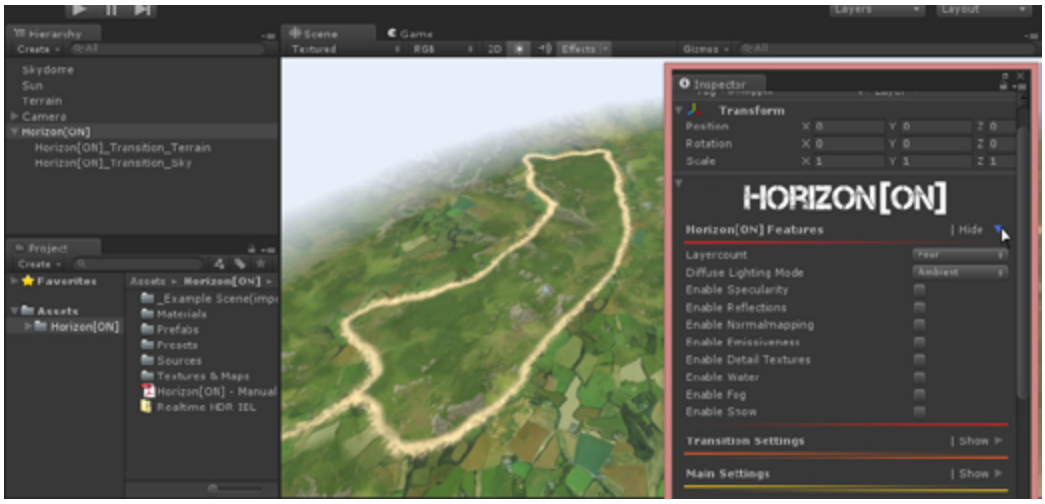
Let's just extrapolate what you have learned and skip layers 2 and 3, we will switch directly to a layercount of 4.



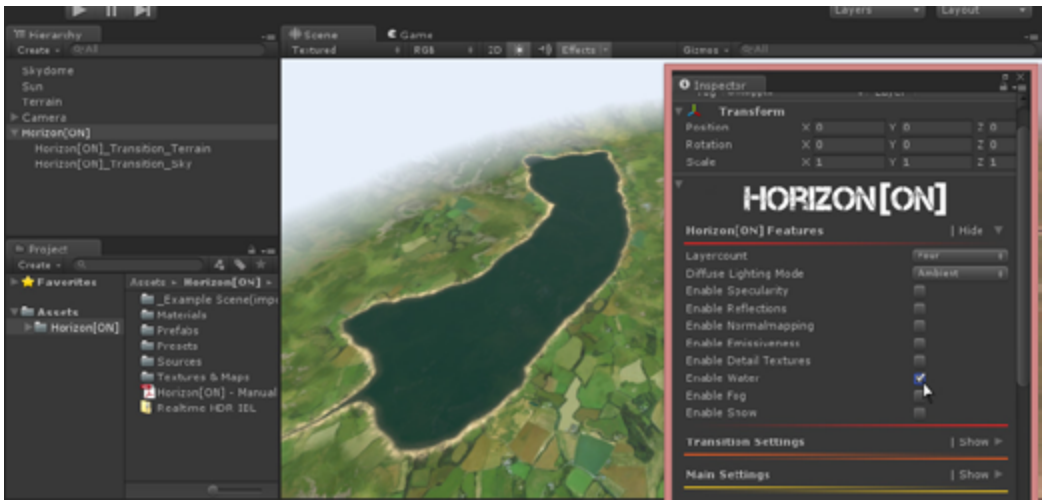
Now that we increased the layercount there are a few more options available in the „Main Settings“ section. We see that there is a mask used to define where which layer is drawn. We need to find a sandy area which we can move towards our terrain. Of course it should match the sandy corner of the terrain. Click and drag the „OffsetX“ and „OffsetY“ labels until you have it matching roughly like in the above image.



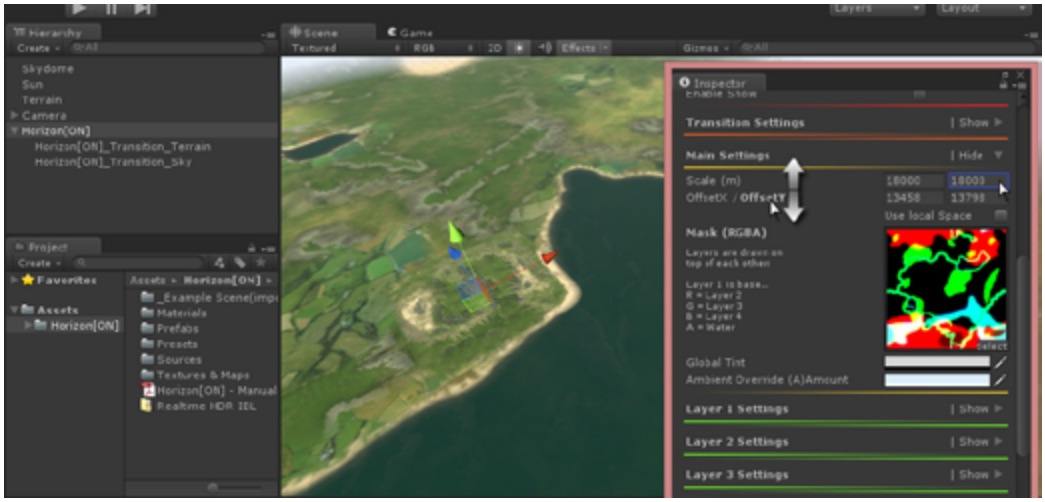
Great, though the sand is a little bit too dark. Open the „Layer 4 Settings“ section and change the value in the „Tint / Saturation“ color picker. A little bit brighter should give us a better blend. We will also set the tiling to a value of 15.



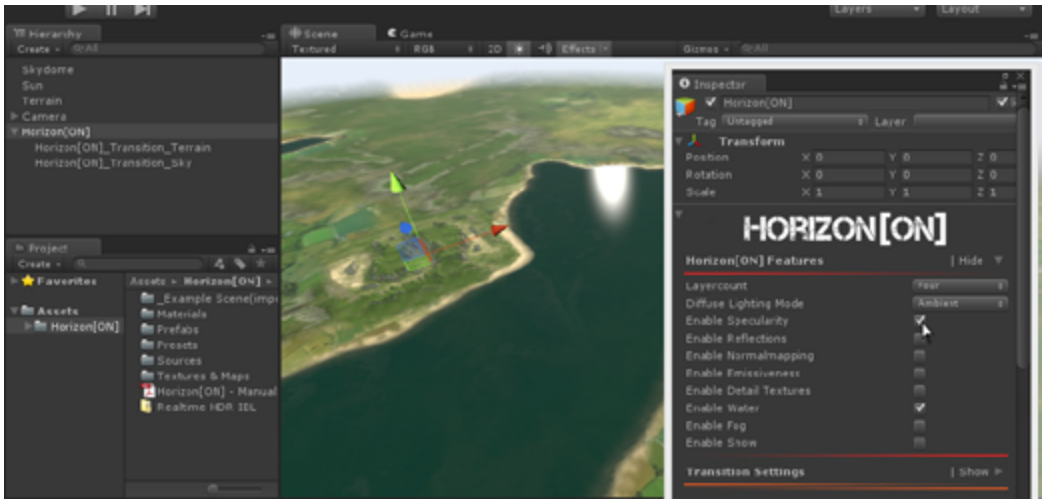
Awesome, but what is this? Well, this is how i painted the mask, i wanted to have the sand as a beach area, surrounding water and in the depths of the water should be a little bit greenish detail, let’s see if that works out.



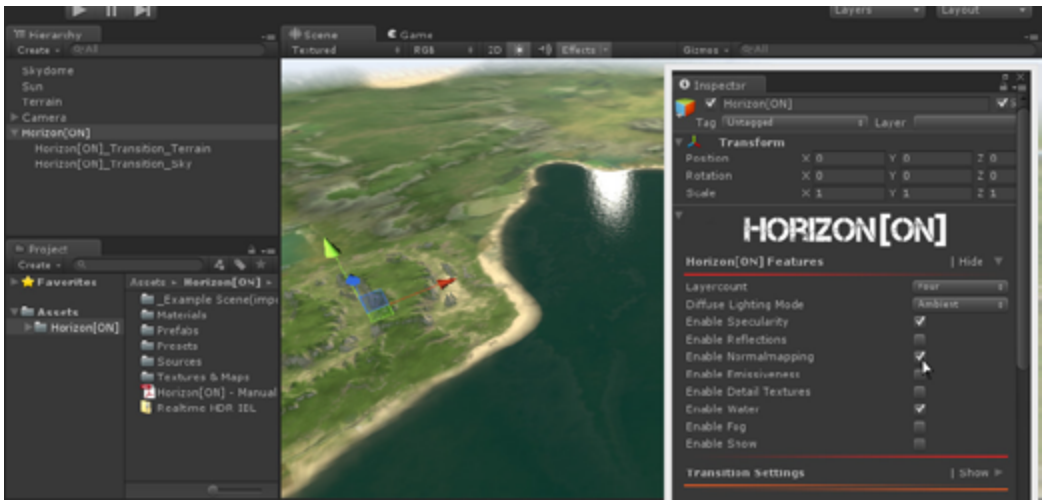
Hmm, yes i think that is good. Actually this corner on the lake looks like it could fit our sandy part of the terrain. Let's go back to the „Main Settings Section".



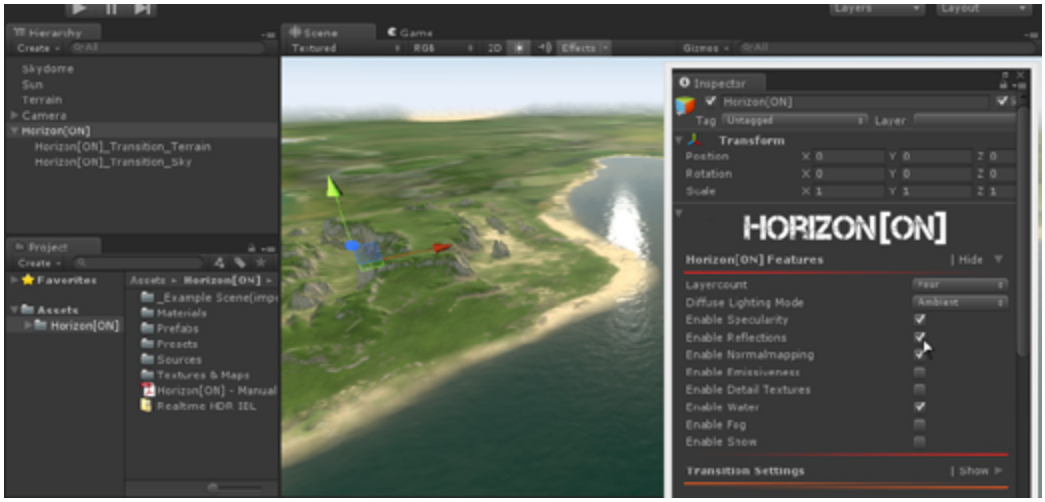
It fits but the terrain is a bit to big, or our mask is too small... We will put our mask scale to around 12300 meters. We also need to adjust the offset but you know already how that works. I like the water but in reality water is not just painted on, it should reflect the sun.



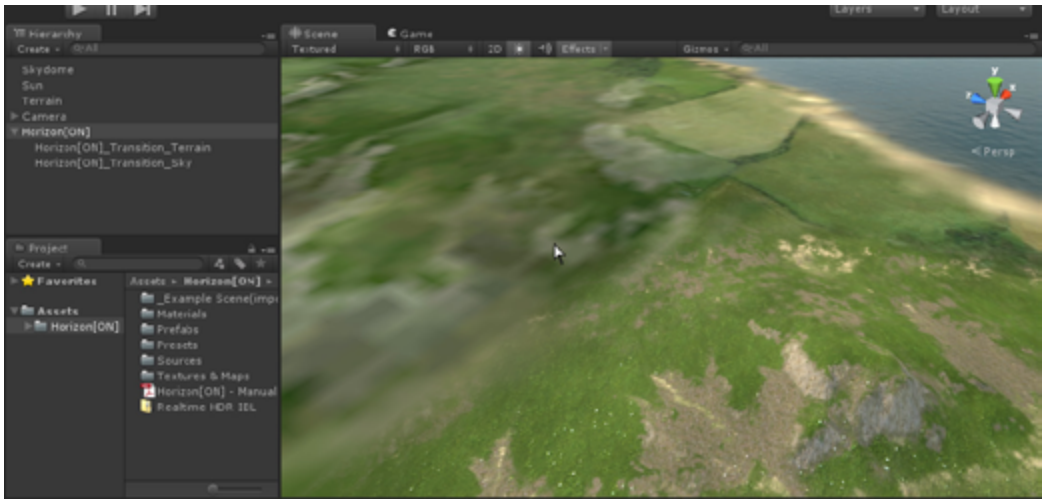
So, let's go back to the „Horizon[ON] Features" Section and check „Enable Specularity". Awesome but i have another complaint. The water is way too smooth, its like a perfect mirror... Seems like we need to enable normalmaps.



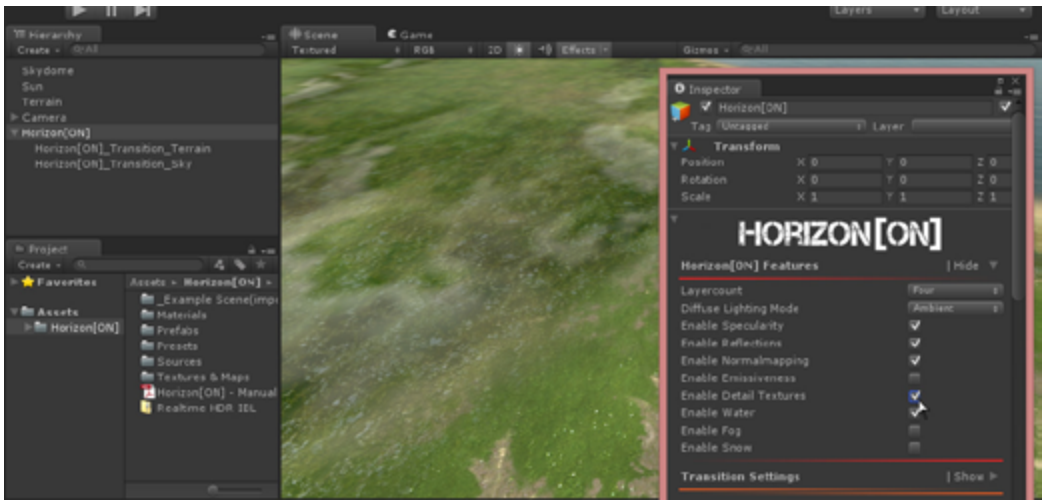
How to enable and disable features you know already, i guess i don't need to tell you... Probably you have done it already before you turned the page. Hey, wow, now not only the water looks better also the rest looks more interesting. There is still something fishy about the water though... (Pun intended).



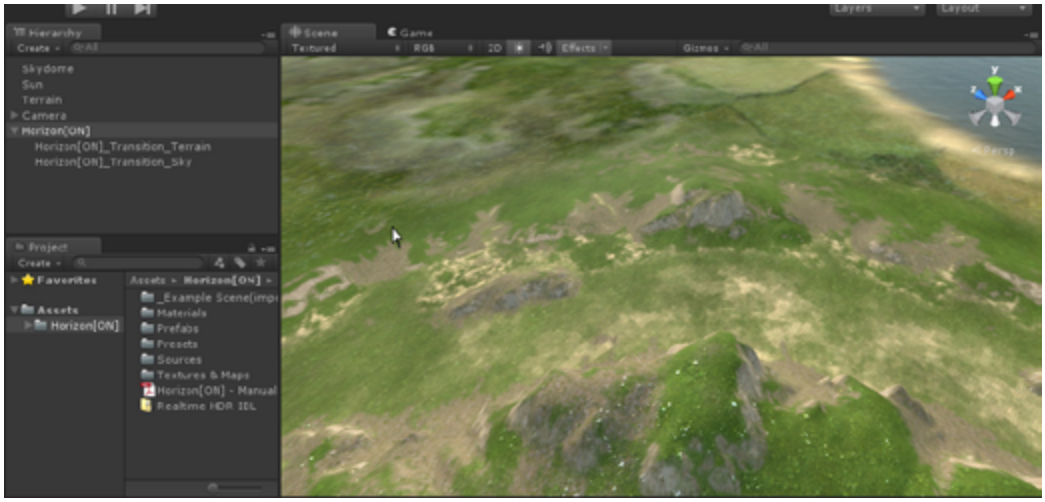
I know what is missing on the water. Water should not only reflect the sun, it should also reflect the sky...
We need to enable reflections.



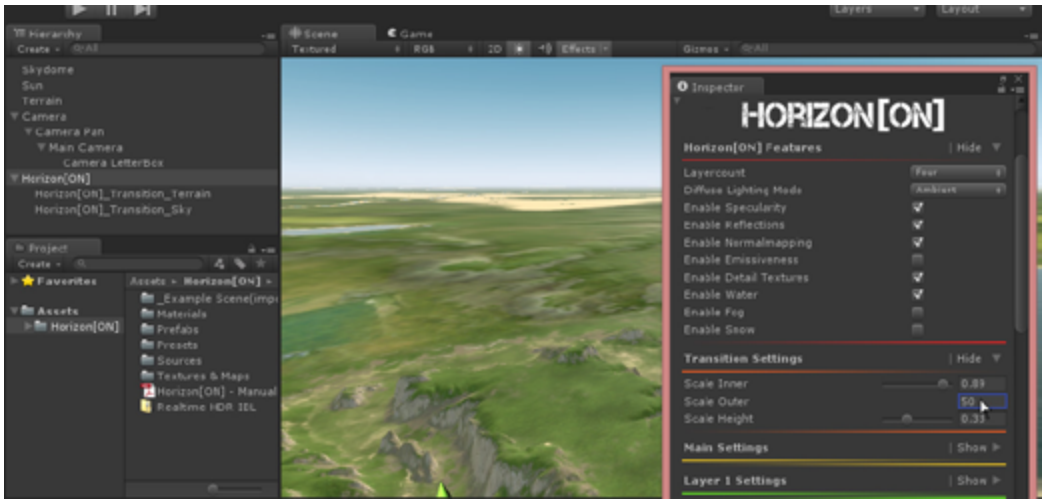
Like i mentioned already 2 pages earlier, the terrain has more high frequency detail compared with Horizon[ON]. Sure, we could just crank up the resolution of the layer textures but i think there is a better way. Since we are already in the features section let's check if detail textures do the trick.



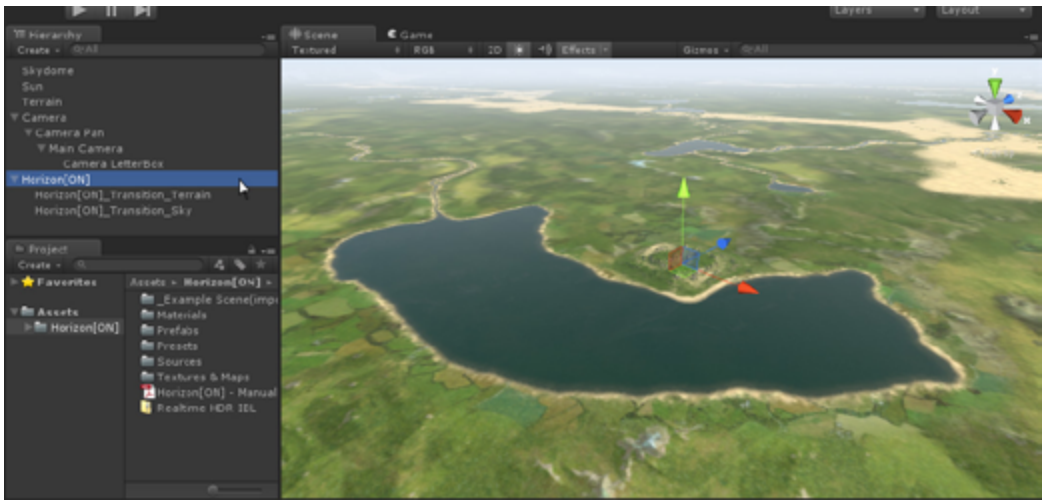
Not bad at all.



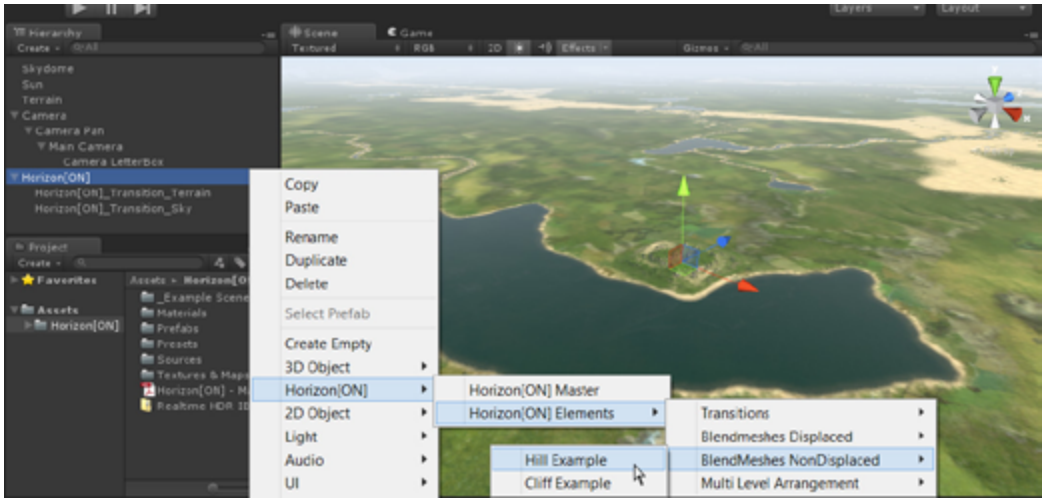
Looking top down we are already there but the horizon line is not far enough away.



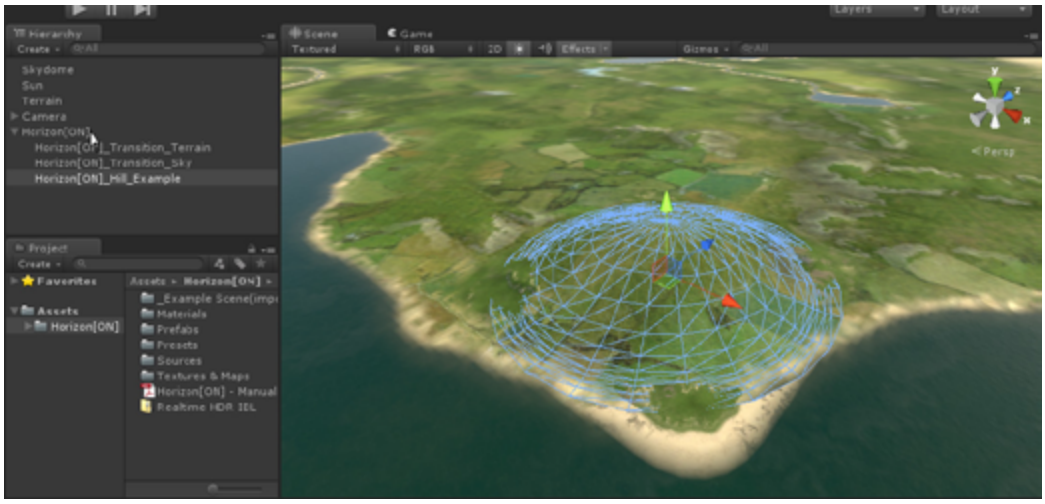
Do you remember where to change it? Yes right, „Scale Outer“ in the „Transition Settings“ should be increased. We should put it as far away as we can but of course it should not exceed the far clip plane of our camera.



Nice, in case it is not active on your side you can enable „Effects“ in the top bar of the scene view. This will not only let us see if the fog is matching but also our water will be animated. One thing is a major letdown though, our horizon is completely flat. Seen from above this is no problem at all but a flat viewing angle breaks the illusion.

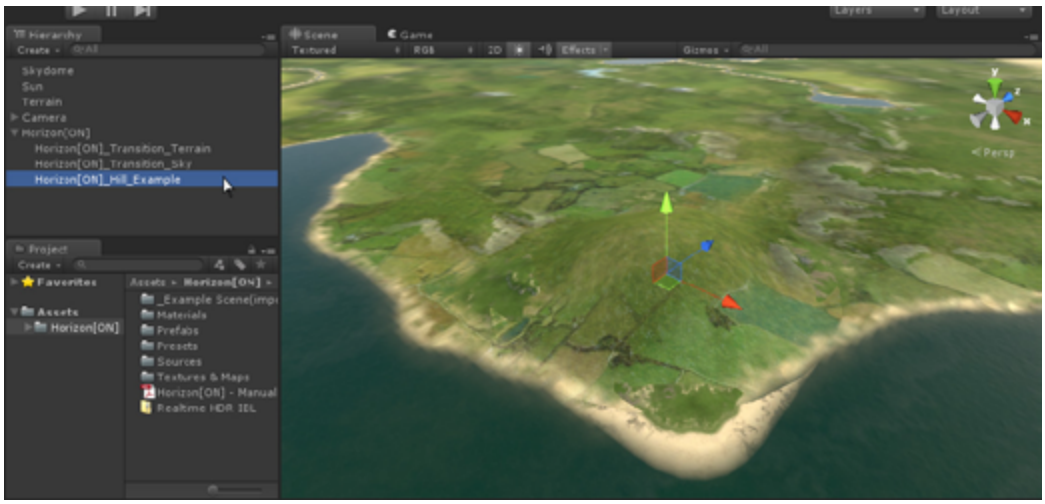


We can try something, you'll like this i guess. Let's right click our Horizon[ON] object in the hierarchy(we need another child element) and go to: Horizon[ON] / Horizon[ON] Elements / Blendmeshes NonDisplaced and choose „Hill Example“.

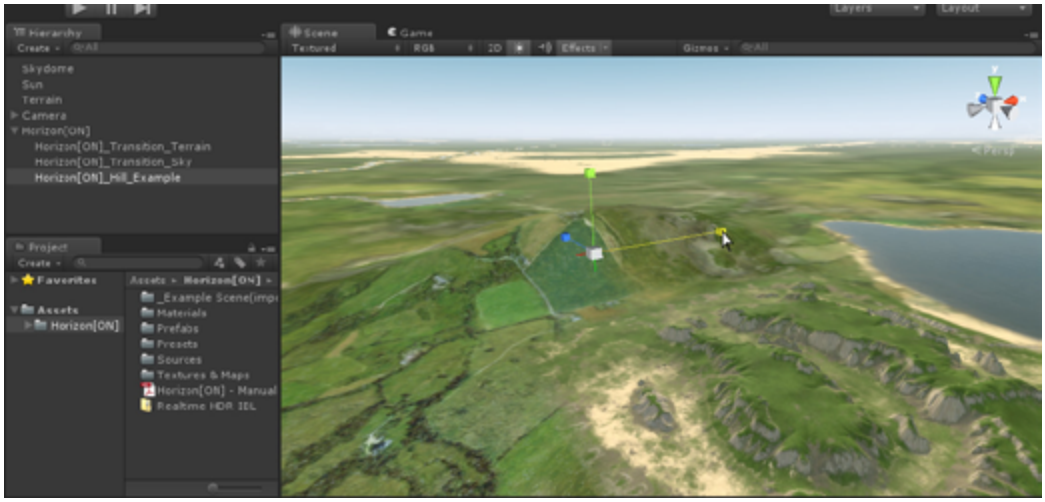


In Horizon[ON] 1.2 it is not necessary to reselect the parent for synchronisation. It should get synched automatically when you add it as a child.

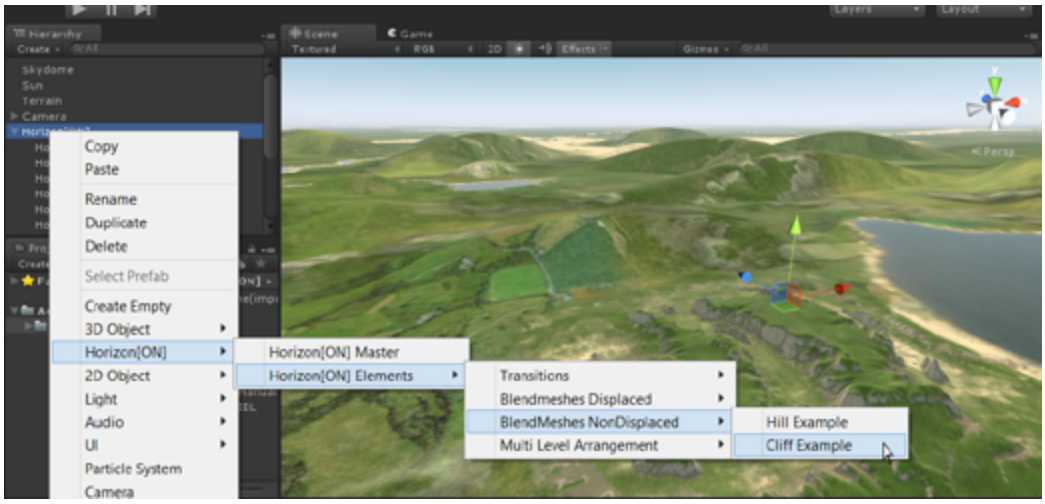
A camoufaged UFO has landed on our terrain. Actually it is a blend mesh, reselect the Horizon[ON] Master to make sure the material is synced. Also it should disable the distracting wireframe.



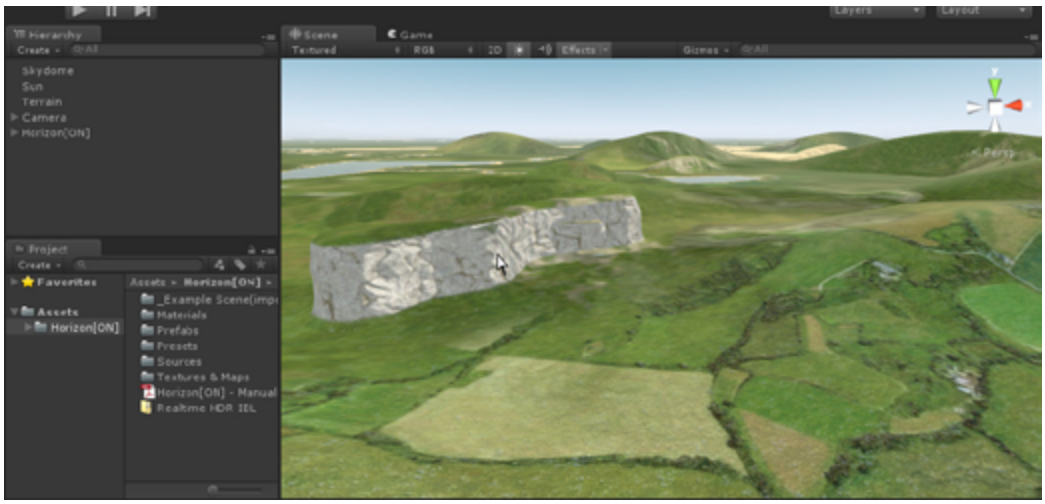
Great, now back on our „Hill Blendmesh“ you can move it around, this is where the fun begins! Just place it anywhere you like. You can move it freely but you should make sure that the Y position is always on the same level as the underlying surface. Also you should make sure that your transform is set to „Pivot“ and not „Center“.



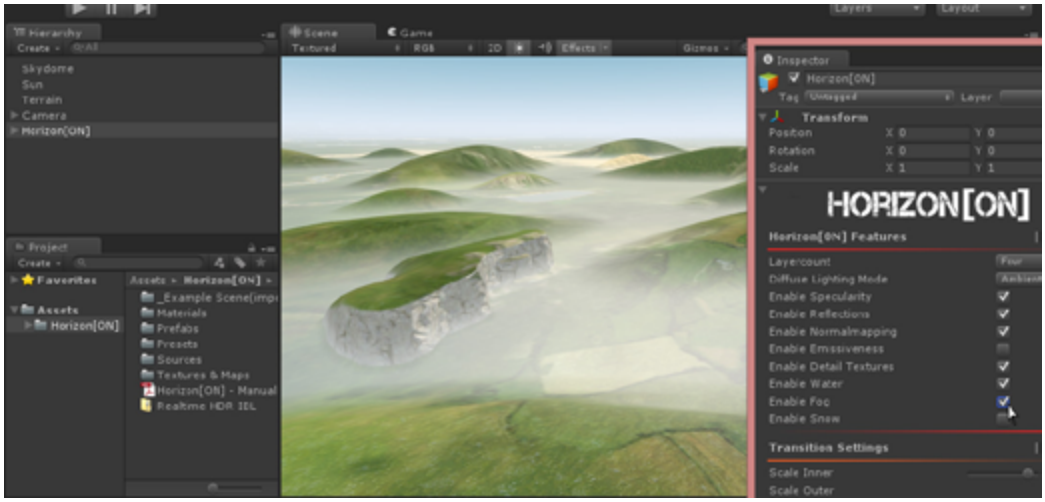
You can also scale it however you like. Make it long and flat or round and pointy, its up to you. You can either duplicate it or get more hills from the „create menu“. Place some more hills if you like or just continue with the next step.



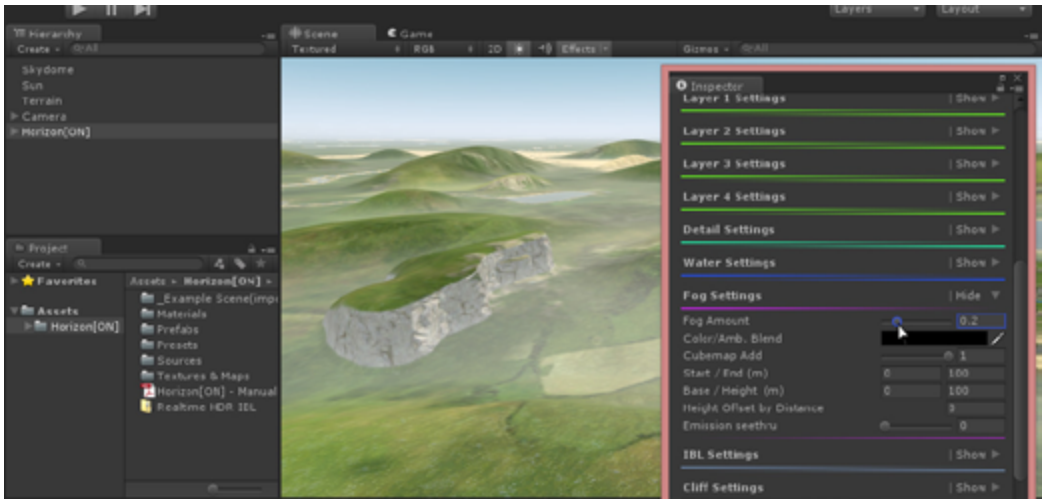
There is also another example of blendmesh, the cliff blendmesh. I named them example to emphasize that you can create your own blendmeshes. It is very easy to do so, you don't even need to care about UVs.



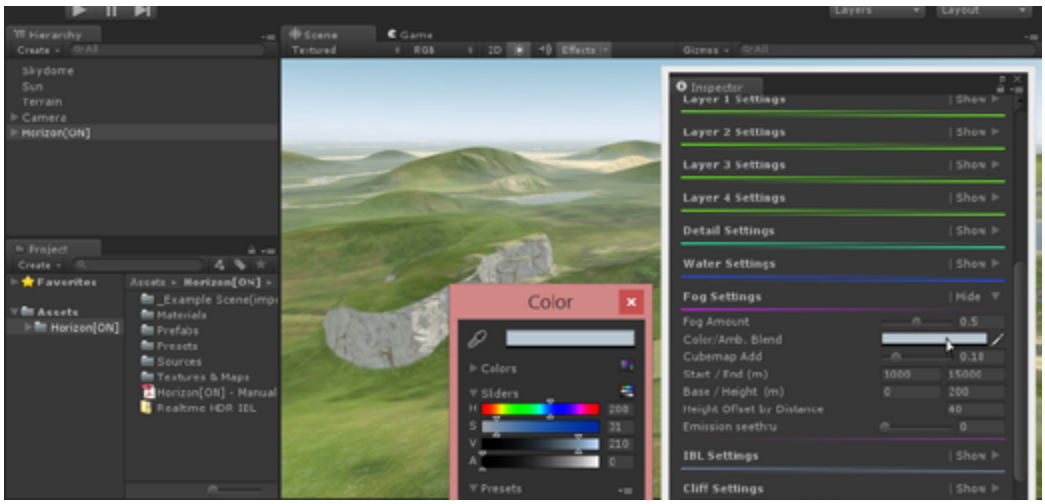
Just like with the „Hill Example“, you can move and scale the cliff freely. Experiment a bit with it.



Let us try out the fog feature. Enable the fog feature and let’s dig a little deeper.



Open the Fog Settings section and we’ll take a look at the options, The most obvious thing there is the Fog Amount.

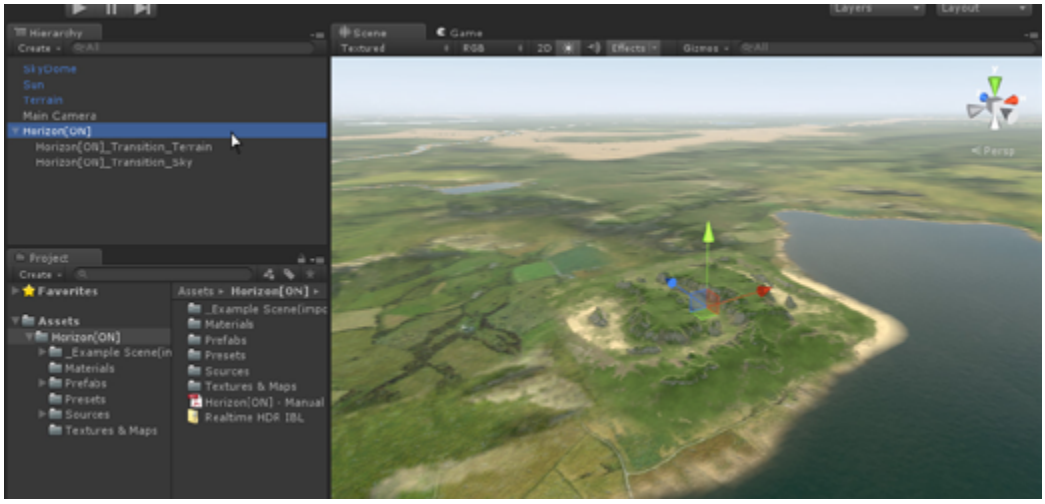


What the Color does is also pretty clear but the alpha value maybe not, if you increase it the fog color will be taken more and more from the ambient light color. As these 2 are usually close together (if not just the same color), it might be easier if you want to care only about one value. Cubemap Add is also very interesting, it adds color from the reflection cubemap to the fog color, this is nice to get more directional dependency of the fog and can help to match the fog with the skybox. Take a moment and experiment with the other values, if you hover the mouse over a label there will be a tooltip explaining the functionality. You can take the values in the image above as a guideline if you want to do so.

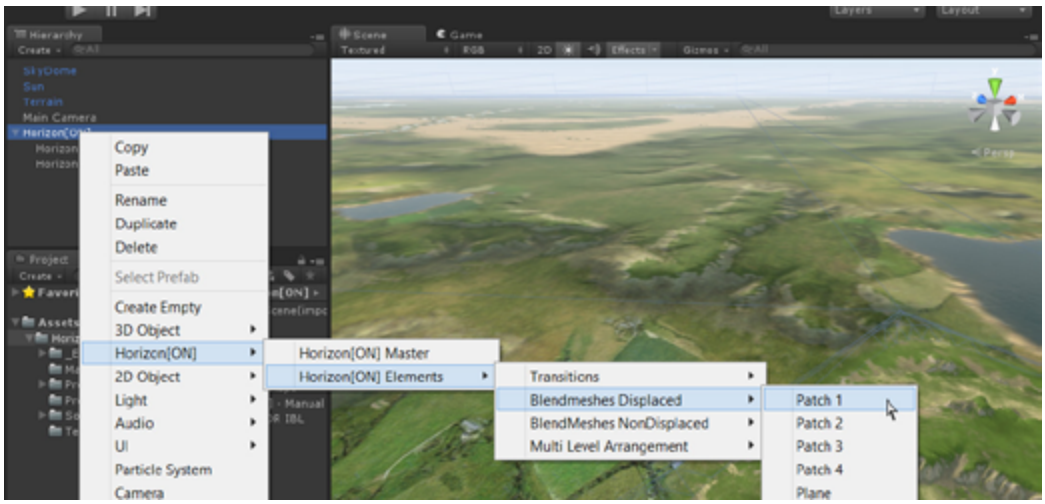
In Horizon[ON] 1.2 changing the diffuselighting mode is no longer needed. It will take whatever settings you choose in Unity's Lighting window.

In Horizon[ON] 1.4 the "Diffuse Lighting Mode" Enum dropdown has been removed, please ignore this step.

Try also to set the „Diffuse Lighting Mode" to Cubemap in the features section. You should be familiar with IBL already. If not, look it up. It is basically what the famous Skyshop does, Cubemaps generated with Skyshop are compatible.

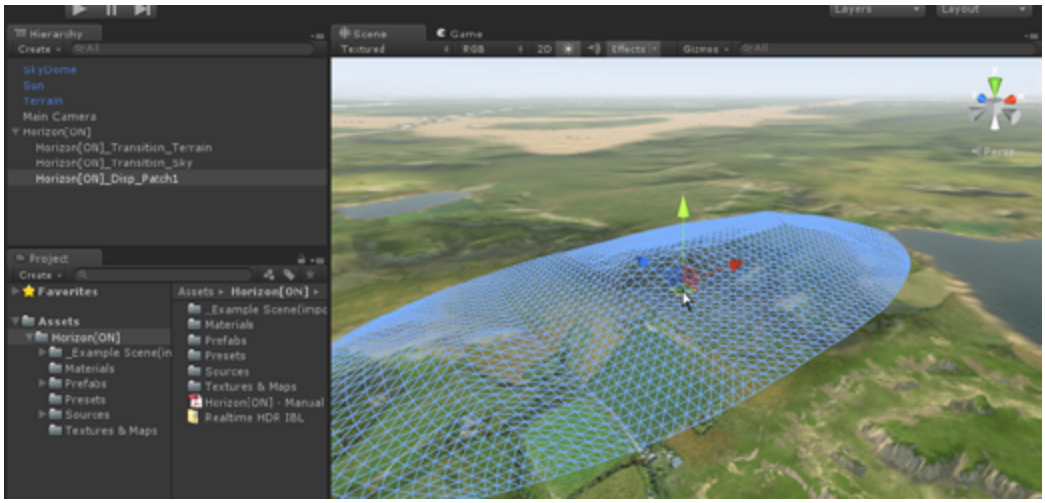


Ok, now we can take a look at another type of blendmesh. The displaced category... Those are my favorite ones. In case you have plastered your horizon already with hills and cliffs you might want to make some room for the next experiments.



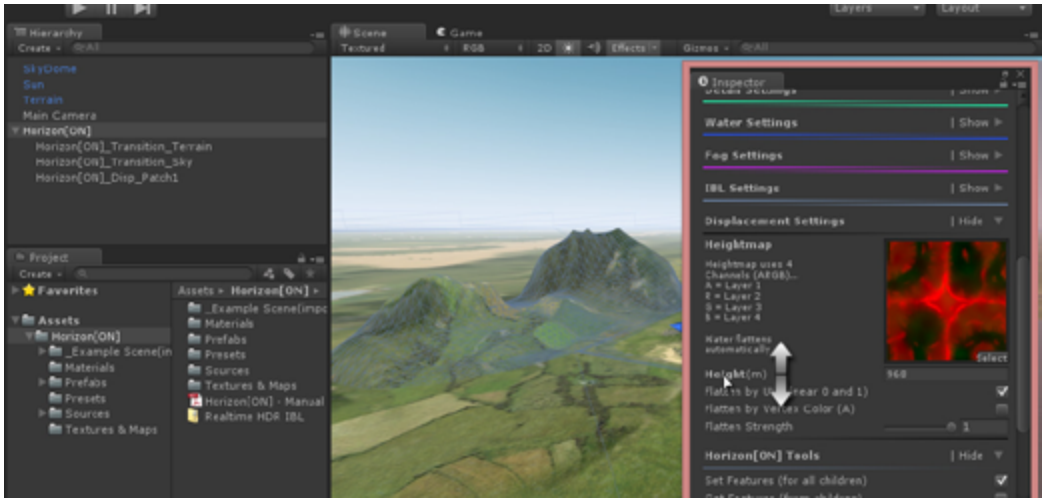
Again, right click on the Horizon[ON] Master object in the hierarchy and take a look in: Horizon[ON] / Horizon[ON] Elements / Blendmeshes Displaced.

Feel free to place them as you like, these can only be scaled on X and Z the Y axis is driven by a 4 channel heightmap where each channel relates to one layer.

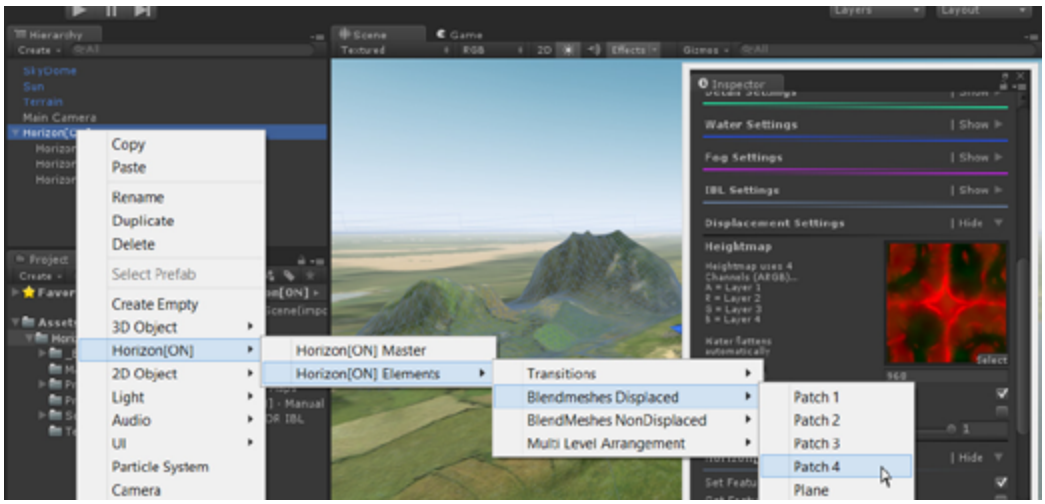


In Horizon[ON] 1.2 and above it is not necessary to reselect the parent for synchronisation. It should get synched automatically when you add it as a child.

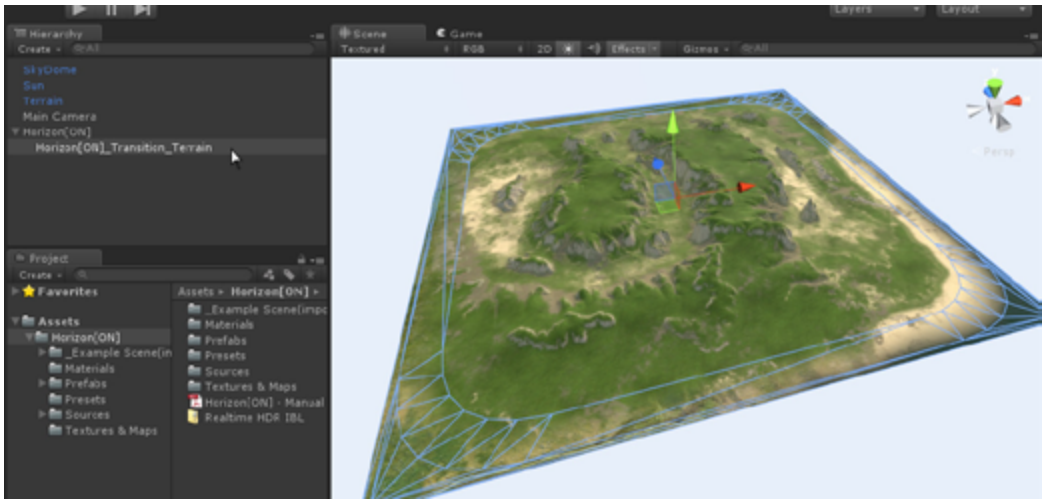
Don't forget to sync the material by selecting the parent after placing the first one(This is only necessary once for each shared material).



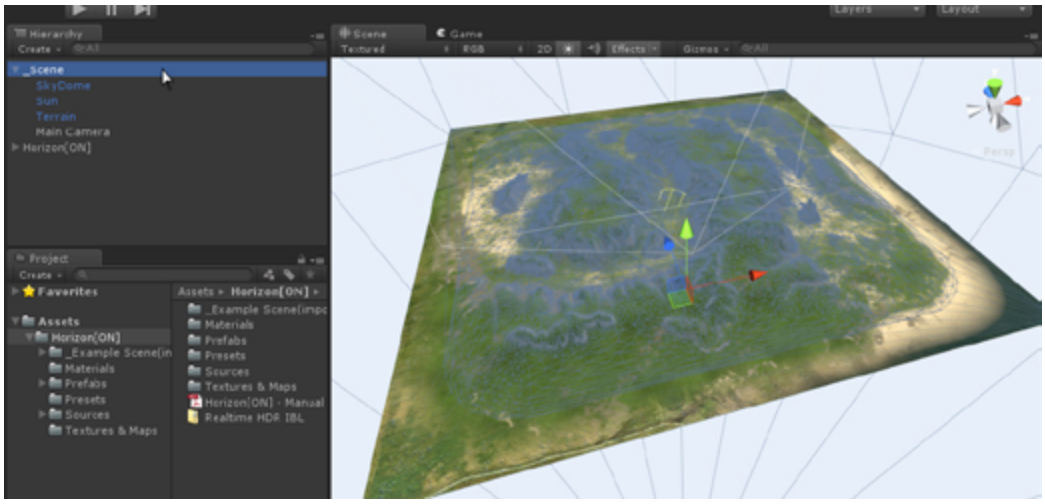
Let's have some fun with the "Displacement Settings". Play with the height and the other settings. Don't forget that for everything are tooltips available. So if you need an explanation for a certain feature, it is right there under your mouse cursor.



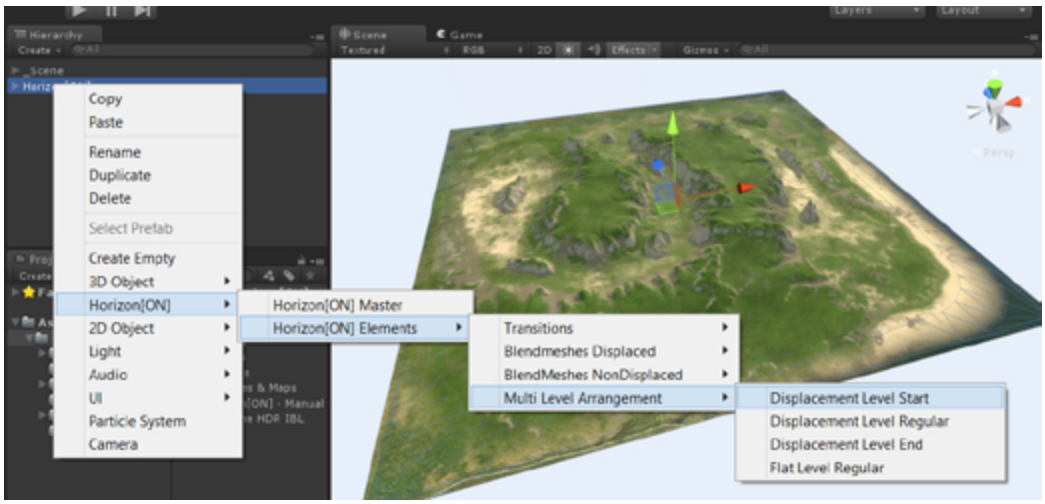
Try all the blendmeshes from the displaced category. They are great fun to play with, i like how they automatically conform to Horizon[ON] and flatten on the water. Especially how the rivers cut through the mountains is nice.



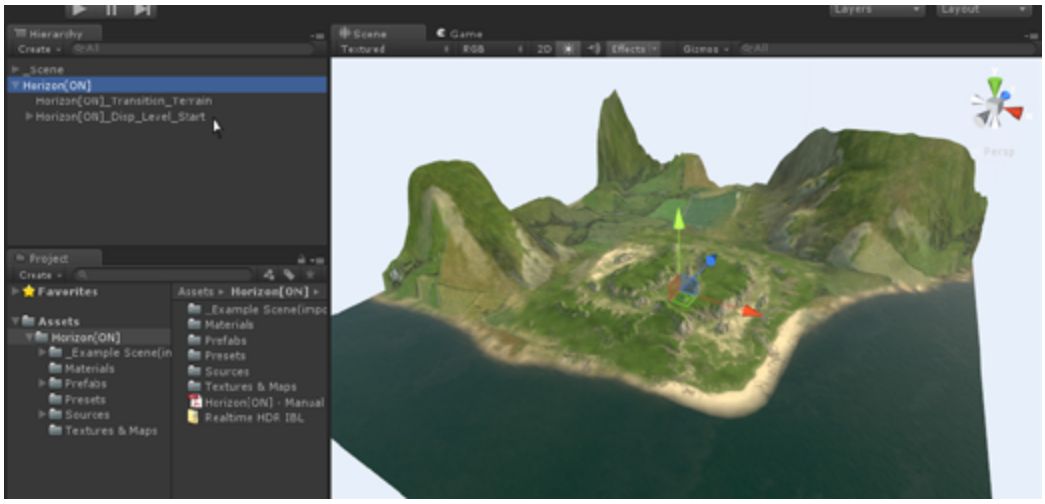
Time to look at the most impressive displacement feature of Horizon[ON], but we need more room... Delete all children of the Horizon[ON] Master, only leave the Horizon[ON]_Transition_Terrain in there.



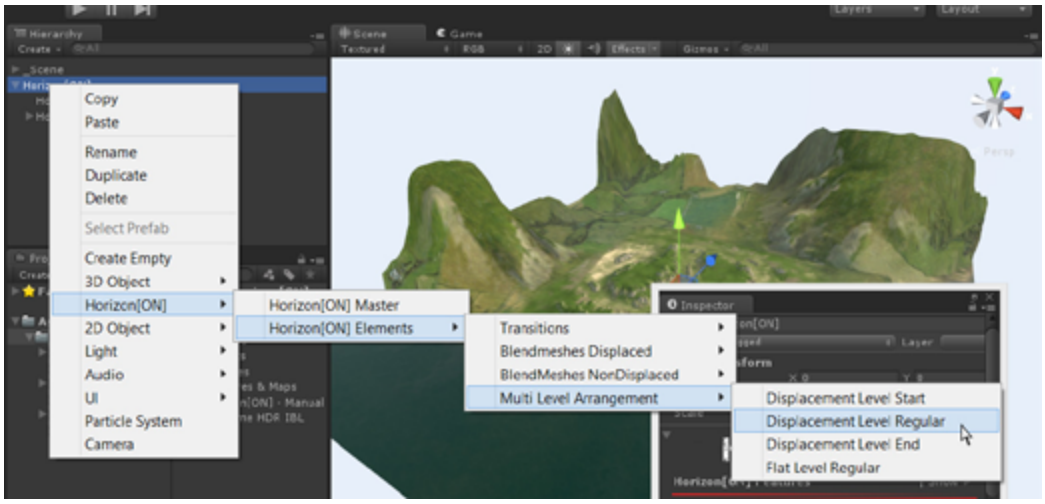
Let us also put all the other objects into a new gameobject. I called it _Scene but it is not of importance how it is called. It will just help us to keep the scene more organized.



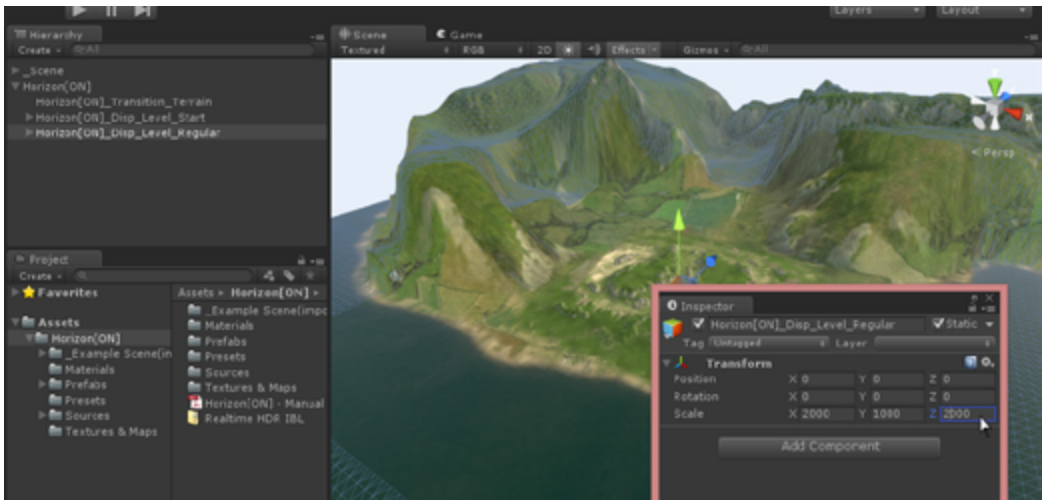
Now that we have made room, we can right click our Horizon[ON] Master object and look into the „Multi Level Arrangement" category, let's select „Displacement Level Start".



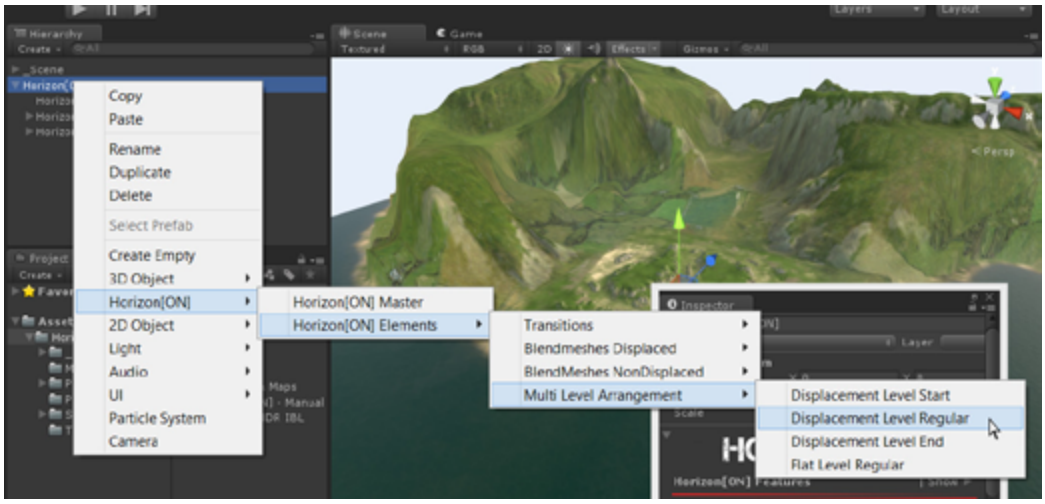
As you can see, this object encloses our terrain completely and flattens towards the center. It consists of 12 children, 2 planes on each side and another plane in each corner. These planes have UVs set up in a special way, so the shader knows where to flatten the displacement. Select the parent so the material gets synced in case it is not already.



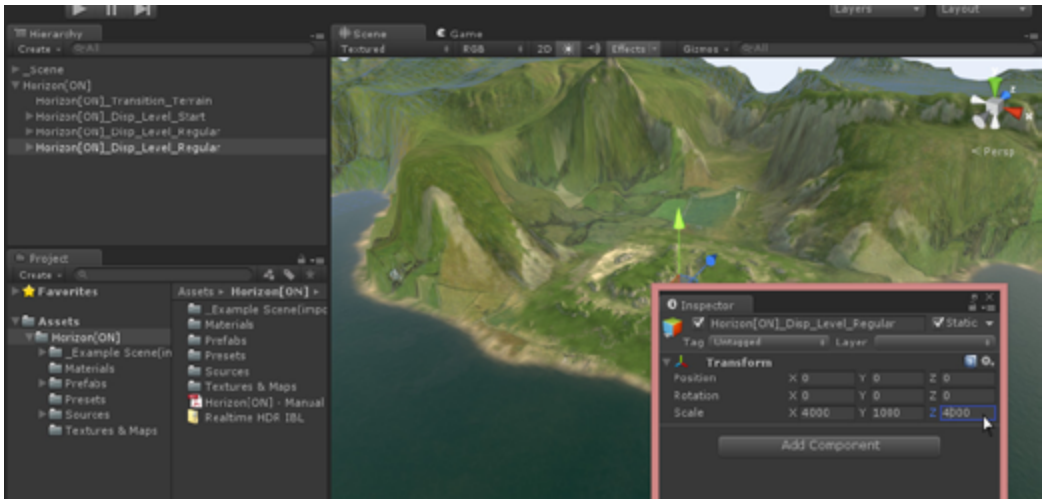
We will now add another element from this category. Select „Multi Level Regular". This is now the first time we need to change the scale of the object to make it fit. By the other elements, e.g. the „Transition Terrain", was fitting our terrain right away. This is because by standard the elements are scaled to the default size of 1000 meters. If our terrain was of another size, for example 2000 by 2000, we would need to put that values in the transform.



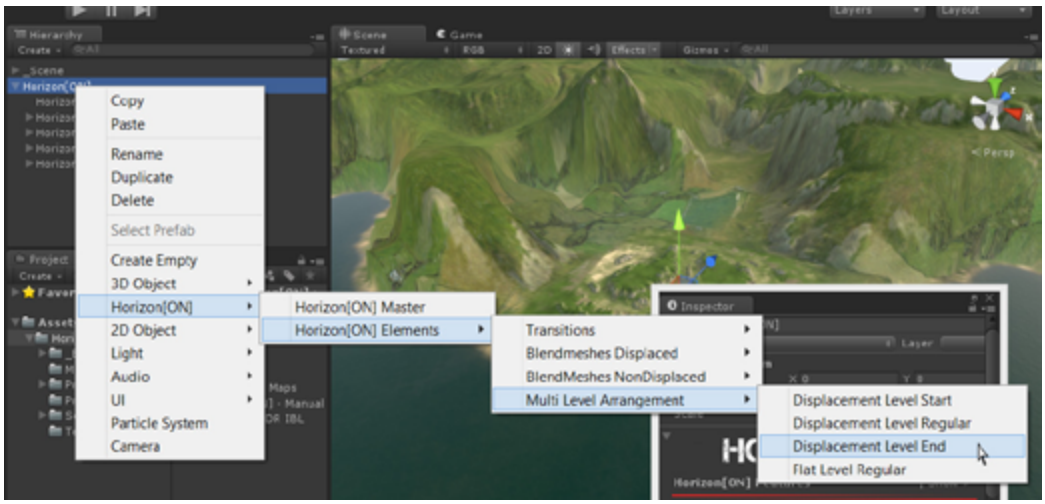
Let's fit the scale of our new element. We need to make it twice as big as our previous level. So put the X and Z Scale to 2000. As you can see it fits the „Displacement Level Start" perfectly.



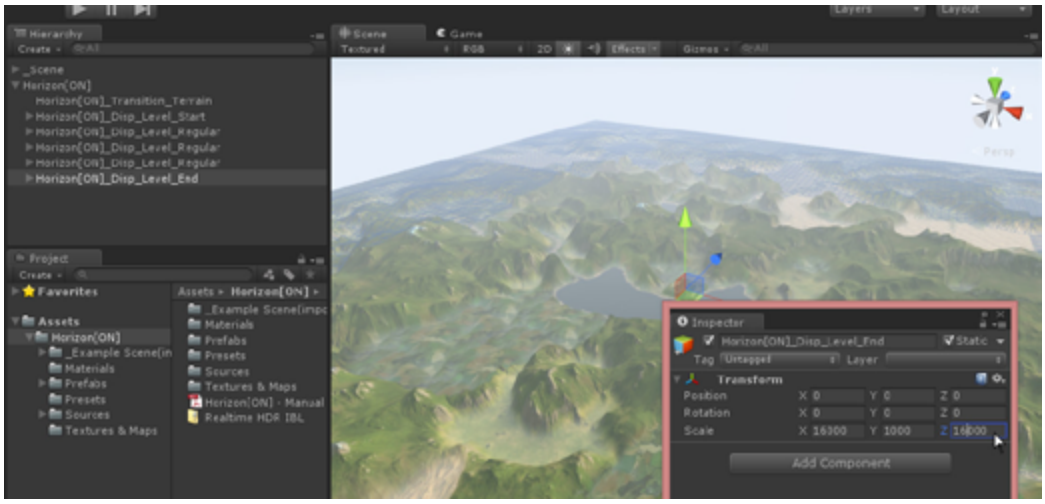
Let's carry on and repeat the process. We create another „Displacement Level Regular"....



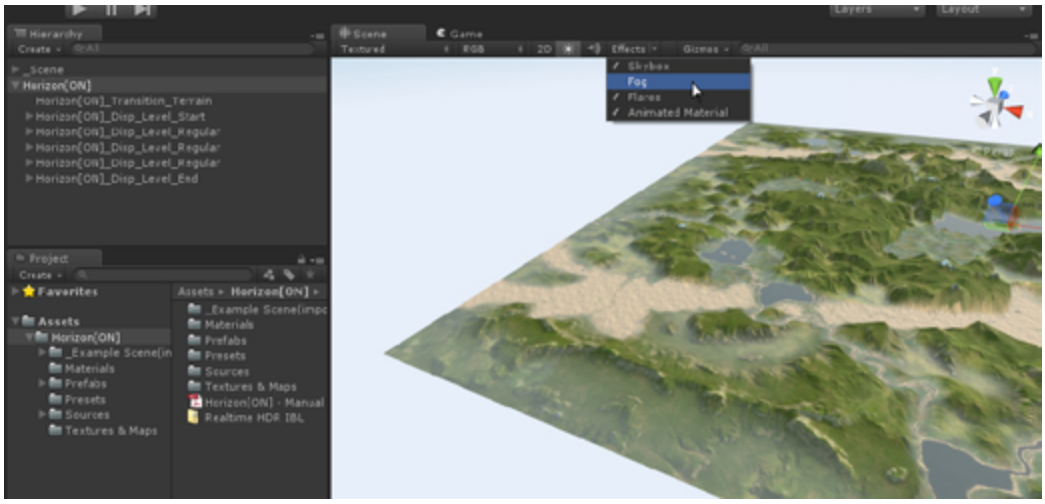
... and set the scale twice as big as the previous level, so now we need to set the X and Z Scale to 4000. We need another „Displacement Level Regular", you can either duplicate the previous one or use the create menu. Double up the scale once again. You can create as many levels enclosing the previous levels as you like but for now 3 of the „Displacement Level Regular" will be enough.



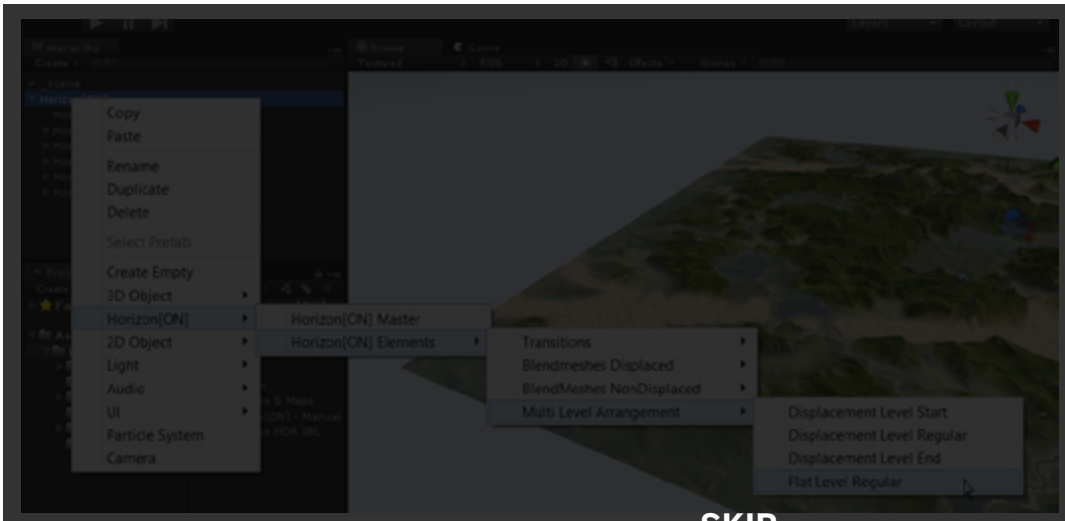
Since these are not flattened towards the outside edge we need now to place a „Displacement Level End“ element.



It need to be twice as big as the previous level as well, so let‘s now put the scale on X and Z to 16000.

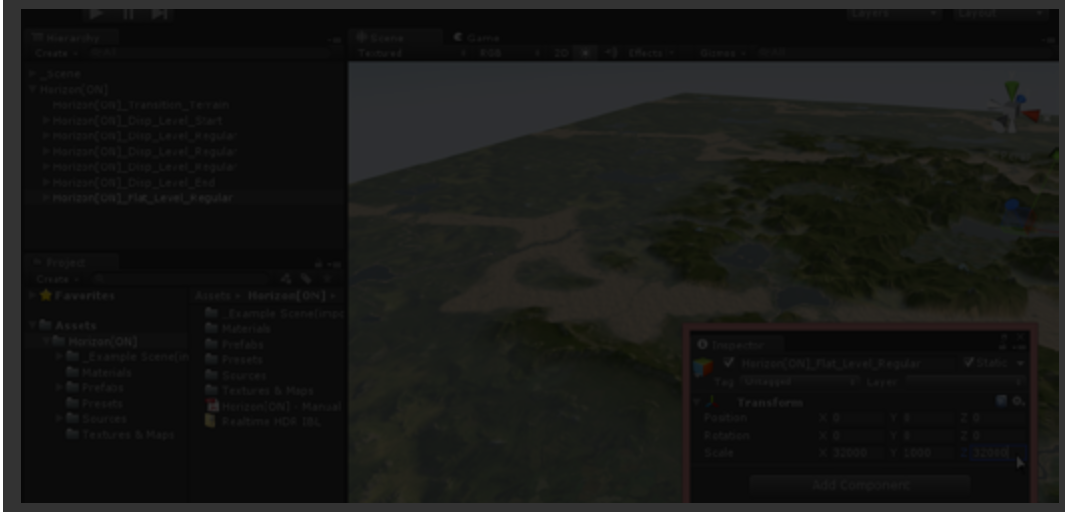


You may need to disable fog for a moment to be able to see something in a more zoomed out state. You can see that the outside edge is again perfectly flat.

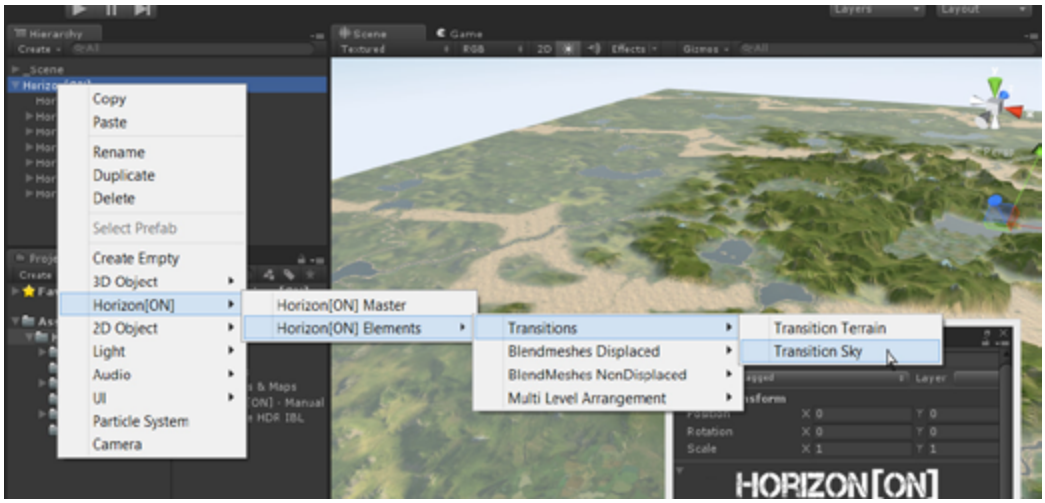


SKIP

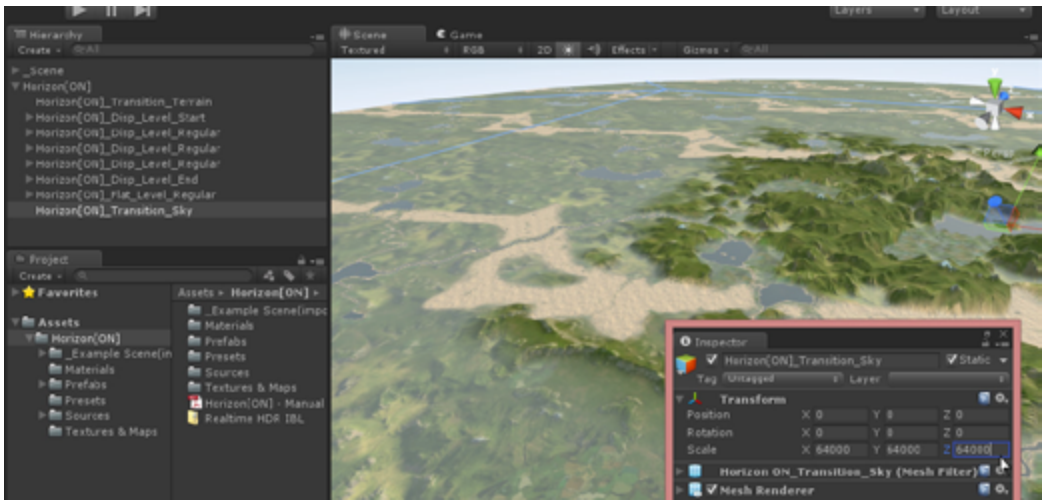
If we had a bigger view distance we could add another level. We are already pretty far in the outskirts of our horizon and we don't need to put much detail there so we could continue flat. "Flat Level Regular" is what you'd should take in such a case.



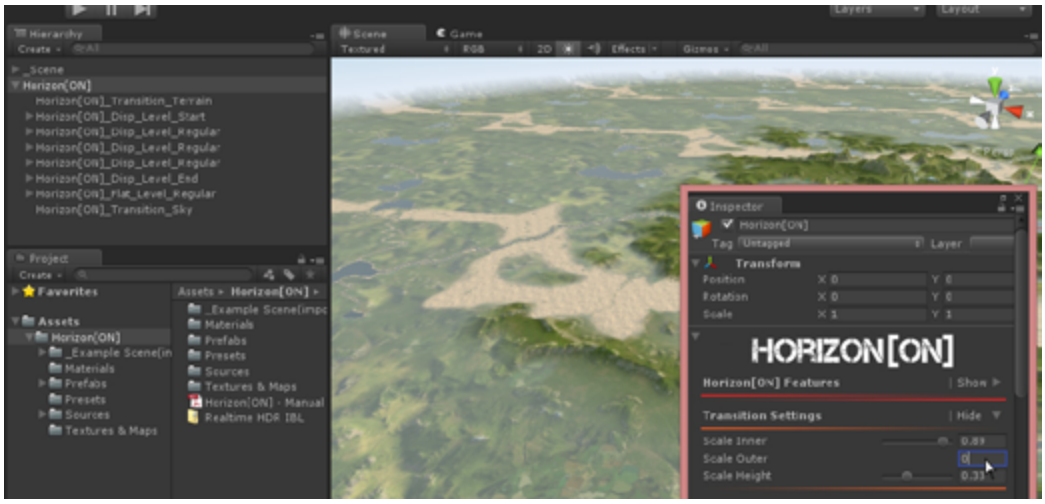
Of course it would need to be twice as big (scale 32000).



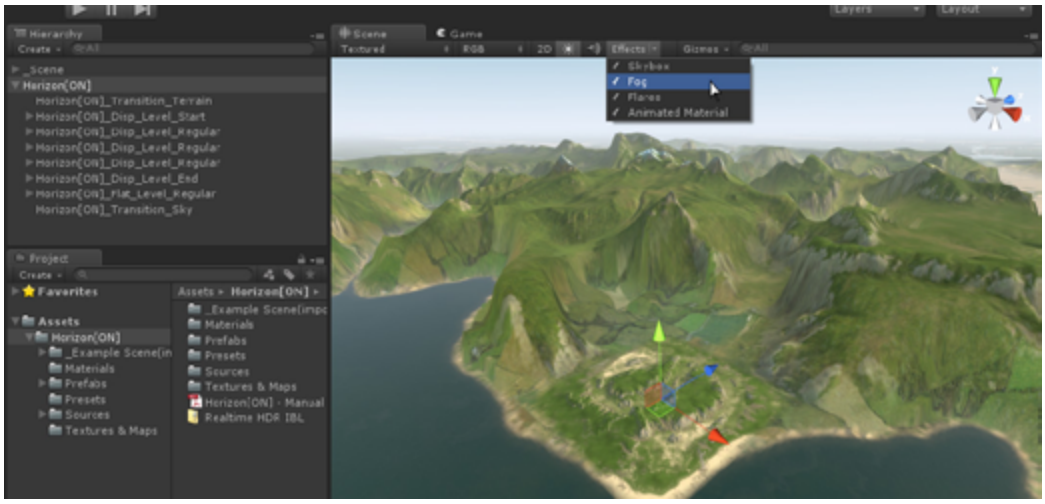
Now it is time to place our sky transition. Let's go to the „Transitions" category and pick „Transition Sky".



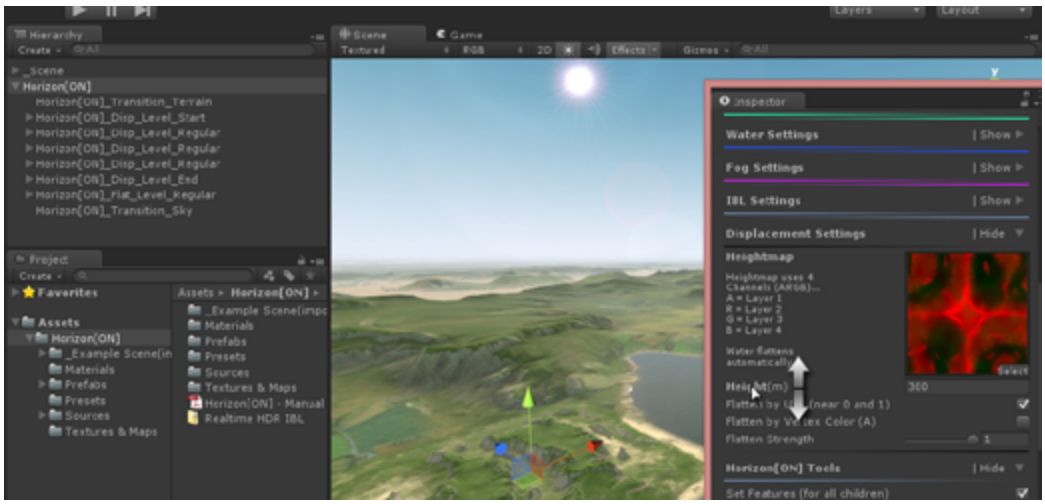
As you are probably expecting we need to scale it up. This time we put the scale on X, Y and Z to 32000. Here the Y axis is important as the mesh is not flat and we want the scale to be uniform. We can see that the mesh is now clearly intersecting with our far clip plane in the game view, so let's fix that.



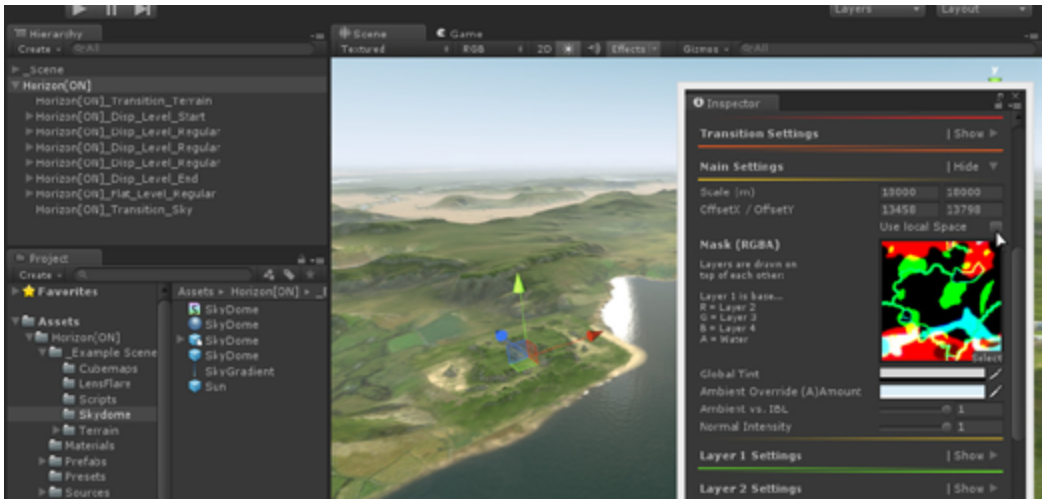
We need to go to the „Transition Settings" on our Horizon[ON] Master object and put „Scale Outer" to 0. We do that because unlike with the previous setup we already have a big horizon and don't need to scale it up additionally. The upward bend on the transition should now end right before our camera far clip plane.



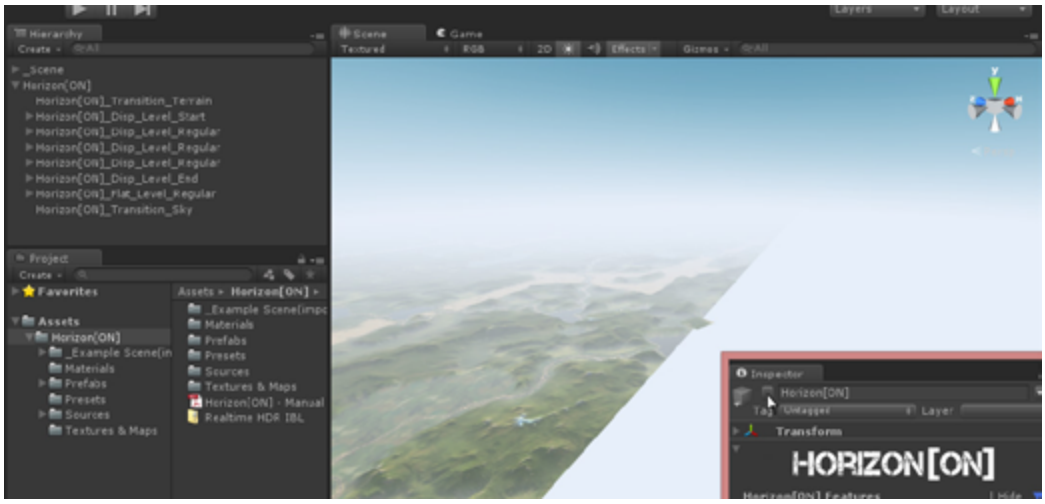
We don't want to see this upward bend though, its only there to give a nice fade towards the sky. So we can re-enable fog now.



Actually we are done now, you can go back into the „Displacement Settings“ and play with the „Height (m)“ value. While you have selected the Horizon[ON] Master object you can also try to move it around on the X and Z axis. You will notice that the textures will stay in place regardless. You may want that or not so there is a way to change that.



Go to the „Main Settings“ and check „Use local Space“. Move the Horizon[ON] Master object again and see the difference. In case you want to do world shifting, this setting might be useful.



The picture above shows a with and without montage, amazing how tiny our terrain appears to be...
Ok, we are finished, you notice i didn't touch on too many details. This guide is really only to get you started.
Take a look at the main manual for explanations of all sliders and knobs and explore a bit on your own.

Have fun using Horizon[ON],
Peter