Manual

Mesh Materializer

Thank you for your purchase of **Mesh Materializer**.

If you have any questions or suggestions, let me know through the forum:
http://forum.unity3d.com/threads/mesh-materializer.293796/

Mesh Materialize can convert any mesh to flat style for imitating low poly effect and bake additional data to the mesh vertex color.

Mesh Materialize works in editor and runtime(build).

Examples videos:

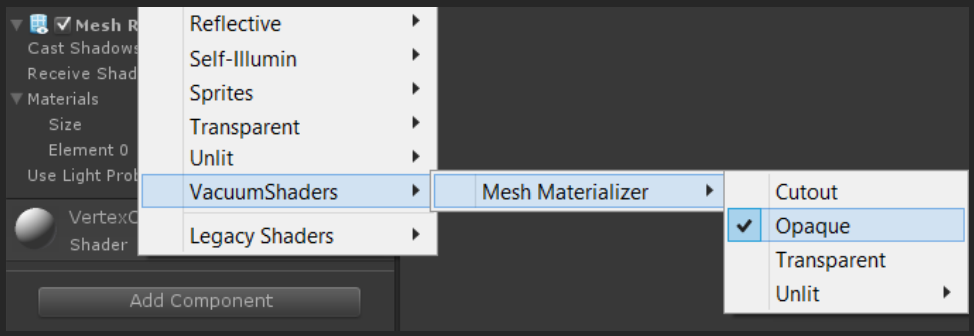
1. Simple mesh conversion http://youtu.be/TufXfjxSRnc
2. Combine simple mesh http://youtu.be/yfcS9pEZqb8
3. Skinned mesh conversion http://youtu.be/yMaZqTgqOh8
4. Terrain conversion http://youtu.be/odF0_MTqZKk
5. Color adjustments https://youtu.be/BmXvm381KhQ
6. Procedural material baking https://youtu.be/bt95FLkwQKU

Mesh Materializer is accessible from menu Window/VacuumShaders/Mesh Materializer.

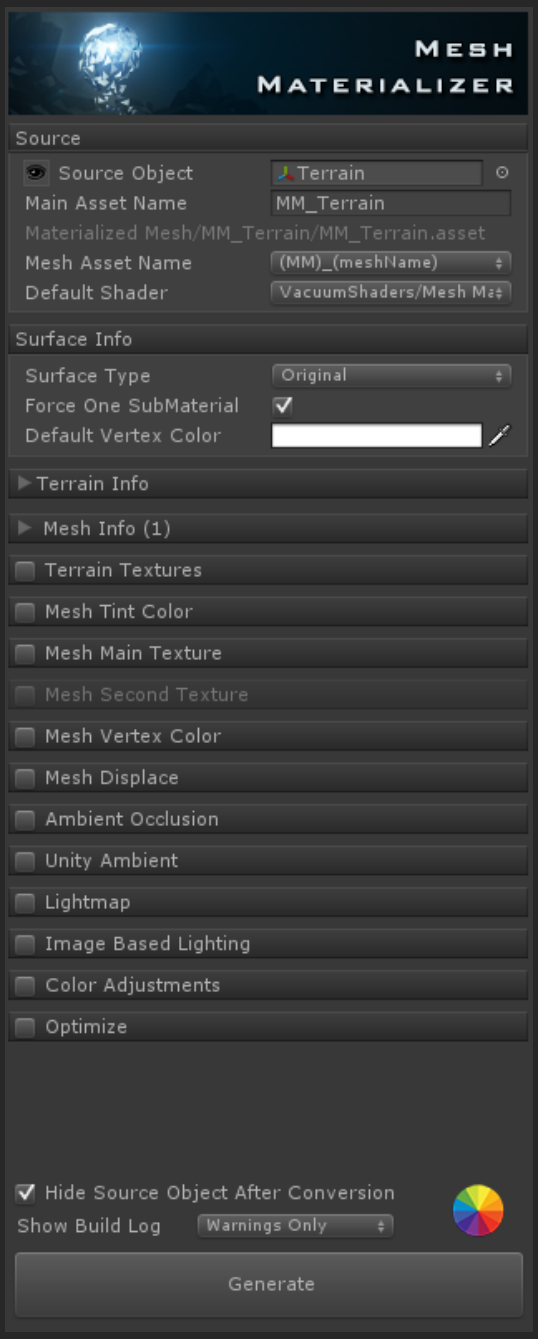
To convert mesh drag its transform from Hierarchy window to Mesh Materializer *Source object* field.

Generated files will be saved inside Assets/Materialized Meshes/ MAIN ASSET NAME / folder.
Mesh Materializer creates ready to use prefab with converted mesh and materials and spawnes it into the scene.

Package contains vertex color shaders with optional Texture, Specular, Reflection, IBL and Emission features.



Mesh Materializer is accessible from menu Window/VacuumShaders/Mesh Materializer.



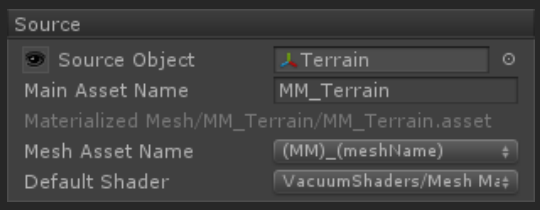
To define which data will be saved inside mesh vertex color just turn on appropriate check box.

Required colors and textures are read from *Source Object's* material(s).

Lightmap info is read from scene.

Image Based Lighting, Ambient Occlusion and Displace are calculated by **Mesh Materializer**.

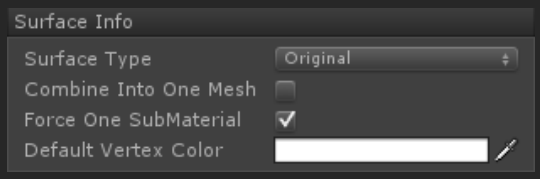
Source



Source Object – Object that should be converted (including its children in hierarchy).
Main Asset Name – This is the name of prefab asset that will be created inside Assets/Materialized Meshes/ MAIN ASSET NAME / folder. Same name will have prefab material.
Mesh Asset Name – Name format of mesh assets created by conversion. These mesh assets will be created in the same folder as main asset.
Default Shader – After conversion Mesh Materializer will create material for the asset, material will use this shader.

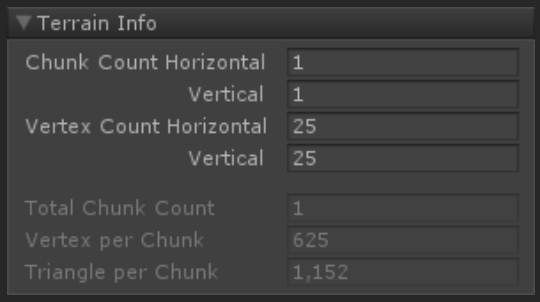
 - Button changes *Source Object's* visibility. If *Source Object* is child, than visibility is controlled by its parent object and this button is disabled.

Surface Info



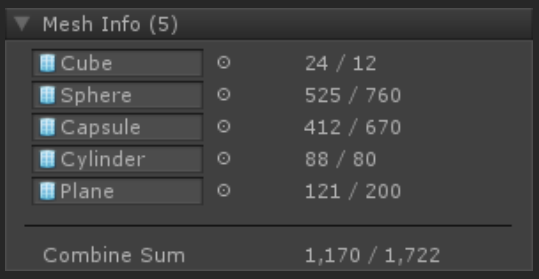
Surface Type – should be saved original triangle type or it should be flattened. For flattened meshes vertex count = 3 * triangle count. If resultant mesh will have more than 65,000 vertices it will be split.
Combine Into One Mesh – If *Source Object* contains only simple meshes, they can be combined into one mesh (one mesh asset file).
Force One Submaterial – generated mesh will combine all submaterials into one. Meshes with one submaterial require only 1 draw call for rendering.
Default Vertex Color – mesh vertex color if it is supposed to be saved with mesh. Also default color is used if mesh material has no baked colors.

Terrain Info



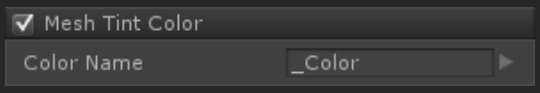
Chunk Count Horizontal/Vertical – Instead of converting source terrain into one mesh, it can be divided into multiple chunks.
Vertex Count Horizontal/Vertical – Chunk vertex count. Each chunk may have max 65,000 vertices.

Mesh Info



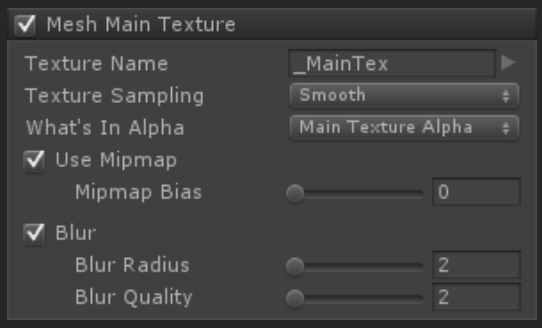
Displays simple and skinned mesh info (vertex and triangle count). If **Combine Into One Mesh** is turned on, displays resultant mesh vertex and triangle count.
If mesh vertex count after flattening will be more than 65,000 info is displayed in red color.

Mesh Tint Color



Adds color baking.
Color Name is parameter inside mesh material from where color value should be read.
Pick-up button displays color parameters that are accessible within selected mesh material(s).

Mesh Main Texture



Adds texture baking.

Texture Name is parameter inside mesh material from where texture value should be read.
Pick-up button displays texture parameters that are accessible within selected mesh material(s).
To bake texture it should be readable. Check “Read/Write Enabled” inside texture import settings.
Unity readable texture formats are - ARGB32, RGBA32, BGRA32, RGB24, Alpha8 and DXT.

Texture Sampling (enabled only if surface type is Flat)

- Smooth
- Flat Hard
- Flat Smooth
- Flat Smoother

What’s In Alpha – how to interpret provided texture alpha

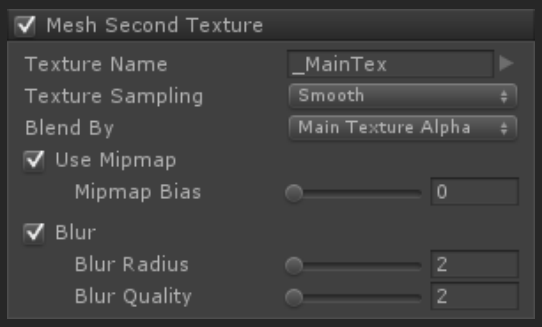
- Texture Alpha – generated vertex color alpha will be the same as texture alpha
- One - generated vertex color alpha will be 1 (one)
- Zero - generated vertex color alpha will be 0 (zero)
- Blend Add – if second texture is also baked, generated vertex color alpha will be sum of Main and Second texture alphas.
- Blend Multiply - – if second texture is also baked, generated vertex color alpha will be multiply of Main and Second texture alphas.

Use Mipmap – texture color value is read from texture mipmap.
To use mipmap texture should have checked “Generate Mip Maps” inside texture import settings.

Blur texture before sampling.

- Blur Radius – size of Gaussian blur.
- Blur Quality – iteration count for Gaussian blur.

Mesh Second Texture



Adds second texture baking (blending), if Main Texture is enabled.

Texture Name is parameter inside mesh material from where texture value should be read.
Pick-up button displays texture parameters that are accessible within selected mesh material(s).
To bake texture it should be readable. Check “Read/Write Enabled” inside texture import settings.
Unity readable texture formats are - ARGB32, RGBA32, BGRA32, RGB24, Alpha8 and DXT.

Texture Sampling (enabled only if surface type is Flat)

- Smooth
- Flat Hard
- Flat Smooth
- Flat Smoother

Blend By – how two textures (Main and Second) should be blended.

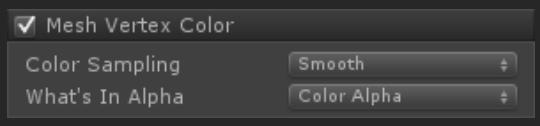
- Add – arithmetical sum of two textures.
- Multiply – arithmetical multiplication of two textures
- Main Texture Alpha - blend two textures by Main texture alpha
  lerp(MainColor, SecondColor, MainAlpha)
- Main Texture Alpha Invert - blend two textures by Main texture inverted alpha
  lerp(MainColor, SecondColor, 1 - MainAlpha)
- Second Texture Alpha - blend two textures by Second texture alpha
  lerp(MainColor, SecondColor, SecondAlpha)
- Second Texture Alpha Invert - blend two textures by Second texture inverted alpha
  lerp(MainColor, SecondColor, 1 - SecondAlpha)
- Vertex Color Alpha - blend two textures by mesh vertex color alpha (if mesh has vertex color)
  lerp(MainColor, SecondColor, vertexAlpha)

Use Mipmap – texture color value is read from texture mipmap. To use mipmap - texture should have checked “Generate Mip Maps” inside texture import settings.

Blur texture before sampling.

- Blur Radius – size of Gaussian blur.
- Blur Quality – iteration count for Gaussian blur.

Mesh Vertex Color



Adds mesh vertex color baking (if mesh has vertex color).

Color Sampling (enabled only if surface type is Flat)

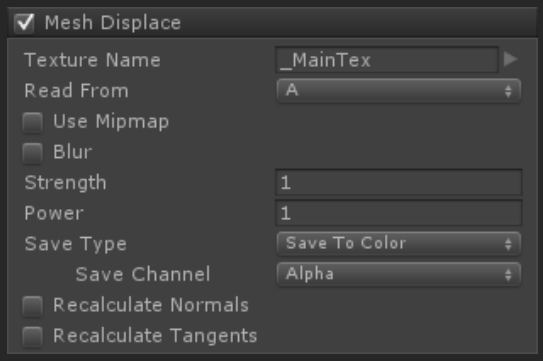
- Smooth
- Flat

What's In Alpha – how mesh vertex color alpha should be interpreted

- Color Alpha
- One
- Zero

If texture is also baking then its generated alpha is multiplied by these alpha value.

Mesh Displace



Adds displace baking.

Texture Name is parameter inside mesh material from where texture value should be read. Pick-up button displays texture parameters that are accessible within selected mesh material(s). To bake texture it should be readable. Check "Read/Write Enabled" inside texture import settings. Unity readable texture formats are - ARGB32, RGBA32, BGRA32, RGB24, Alpha8 and DXT.

Read From – Channel displace is read from.

Use Mipmap – Read displace value from texture mipmap

Blur – Blur texture before reading it.

Strength – Displace strength.

Power – Mathematical “power” of displace value.

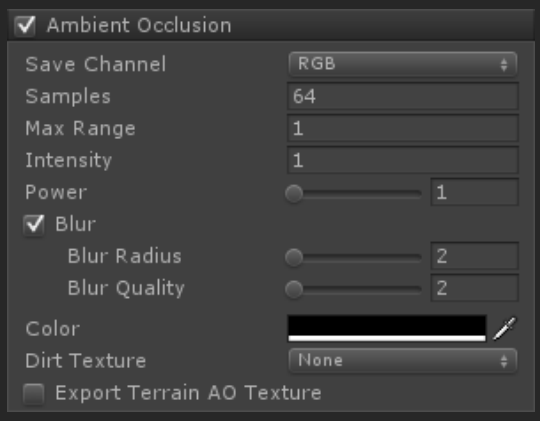
Save Type – How should be saved displace

- Displace Vertex
- Save To Color – Instead of displacing vertices, displace data will be saved inside color (RGB or Alpha channel). Color value will be multiplied to already baked color data (tint color, textures, IBL, AO).
- Displace vertex and save to color.

Recalculate Normals – After displace vertex normals need recalculation.

Recalculate Tangents – After displace tangent buffer needs recalculation.

Ambient Occlusion



Adds ambient occlusion generating and baking.

Color Sampling (enabled only if surface type is Flat)

- Smooth
- Flat

Save Channel – where is saved generated ambient occlusion? Inside RGB or Alpha.

Samples – Iteration count while generating AO. More samples better quality – bigger generating time.

Max Range – Ray length while generating AO.

Intensity – Generated AO value intensity.

Power – Mathematical "power" of the generated AO value.

Blur – (only for terrains) blurs resultant AO values.

Color – AO color if it is saved inside RGB.

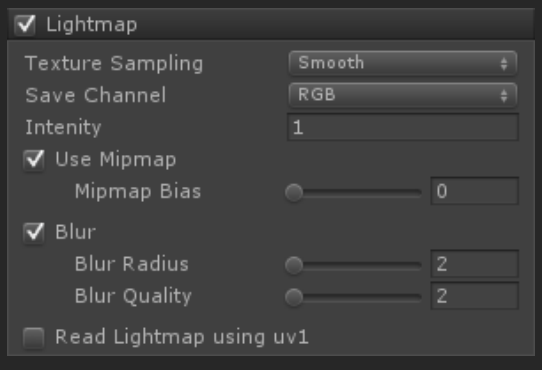
Dirt Texture – AO color will be multiplied by dirt texture color(RGB) and/or dirt texture alpha(RGBA).

Export Terrain AO Texture – (only for terrains) export resultant AO as texture inside *Main Asset* folder.

Unity Ambient (Unity 4)

Adds Unity Ambient color baking.

Lightmap (Unity 4)



Adds lightmap baking (if mesh has lightmap data).

To bake lightmap, texture should be readable.
Unity readable texture formats are - ARGB32, RGBA32, BGRA32, RGB24, Alpha8 and DXT.

Texture Sampling (enabled only if surface type is Flat)

- Smooth
- Flat Hard
- Flat Smooth
- Flat Smoother

Save Channel – where is saved generated lightmap? Inside RGB or Alpha.
Color value will be multiplied to already baked color data (tint color, textures, IBL, AO).

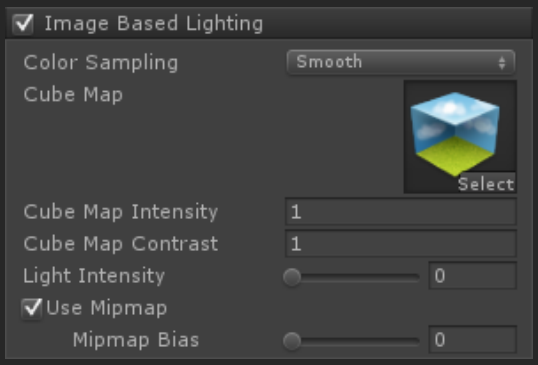
Intensity – generated color value intensity.

Blur texture before sampling.

- Blur Radius – size of Gaussian blur.
- Blur Quality – iteration count for Gaussian blur.

Read Lightmap using uv1 - By default Lightmap is read using uv2, but it can also be read by uv1.

Image Based Lighting



Adds Image Based Lighting (IBL) baking.

To bake IBL, cube map should be readable.
Unity readable texture formats are - ARGB32, RGBA32, BGRA32, RGB24, Alpha8 and DXT.

Color Sampling (enabled only if surface type is Flat)

- Smooth
- Flat

Cube Map – IBL cube that should be baked.

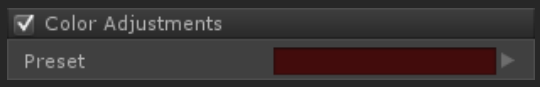
Cube Map Intensity – IBL cube intensity

Cube Map Contrast – IBL cube contrast

Light Intensity - IBL light intensity [0, 1]

Use Mipmap – IBL color value is read from cube mipmap (if it has mipmap).

Color Adjustments



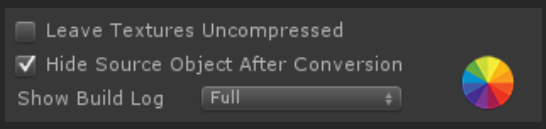
Adds color adjustments baking.

Preset - Color adjustments preset name. Pick-up button displays available color adjustment presets.

Optimize

Optimize mesh by excluding certain buffers from mesh.

Misc



Leave Textures Uncompressed – Before conversion MM checks texture format, if it is in unsupported format MM converts it to 'Automatic Truecolor' for flawless baking . After conversion texture restores its original compression format.

Texture compressing/uncompressing needs some time and increase total conversion time. Leaving textures uncompressed may decrease conversion time up to 50%.

Note: If texture is uncompressed turning off this check box does not restore texture original compression format as MM does not store such info.

Hide Source Object After conversion –

Show Build Log – Outputs useful info about conversion.

– Opens **Color Adjustments** window.

Color Adjustments is accessible from menu Window/VacuumShaders/Color Adjustments or by clicking  button inside Mesh Materializer window.

Color adjustments are applied only to the selected mesh vertex colors and only while CA window is active. If CA window losses focus selected mesh will restore its original vertex color.

Apply changes button will apply color adjustments changes to the mesh vertex color (without undo).

All optional parameters are made similar to the Photoshop controls.

Color Adjustments are applied only to the RGB, not Alpha.

Working with presets

Mesh Materializer and Color Adjustments can be saved in presets.

Presets are accessible from contex (right click) menu for each window.