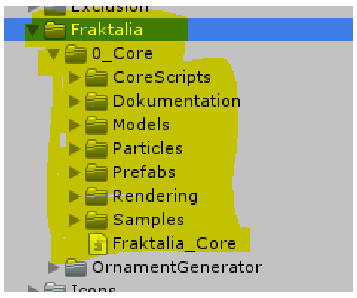
CORE LIBRARY

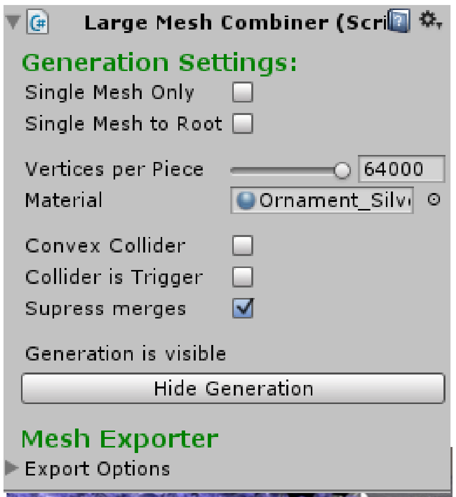


The core library is the content included in every paid asset that use at least one feature included in the library. The library is used by all Mesh based assets I develop as the library contains important functionality and sub systems like the Large Mesh Combiner, Native Mesh Combine, Procedural UV Generator and frequently used math systems. Also the core library provides content related to multi material system and includes a tool which allows you to create TextureArrays which are important for multi-material meshes.

In the documentation, each system has a development flag for which has following states:

- **FINAL**: Final state means that the specific system will not be updated anymore except bug fixes found in the future.
- **COMPLETE**: This state means that the system is released and there is almost nothing left to do. Content is only added if something is found worth to be included. Complete systems are also those which are pretty small.
- **RELEASED:** This is content which is released but there is room for more so it probably get updates regularly.
- **DEPRECATED**: A system marked as deprecated is similar to FINAL but is replaced by a better, more advanced system.
- **BETA**: Content which is already available but may contain unknown bugs and is in active development.

LARGE MESH COMBINER: FINAL

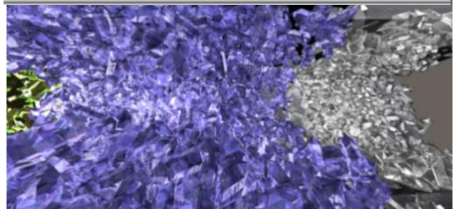


Large Mesh Combiner (Scri

Generation Settings:

Single Mesh Only ☐
Single Mesh to Root ☐
Vertices per Piece 64000
Material Ornament_Silv
Convex Collider ☐
Collider is Trigger ☐
Supress merges ✓
Generation is visible
Hide Generation

Mesh Exporter
Export Options



The Large Mesh Combiner is the core system in some of my assets like the crystal generator. Whenever a large amount of meshes and vertices are involved, this system may be included there.

In Unity the maximum amount of vertices a mesh can have is somewhere at 64000 vertices at the moment. However to reduce draw calls, many meshes as possible should be combined into one mesh instead of multiple game objects with very small meshes.

However procedural generated meshes easily exceed this vertices limit and this is why this system is required. If the amount of vertices is greater than the maximum, the system simply fills up the mesh until its limit is reached and creates a new mesh which is filled with the next vertices.

The screenshot is the mesh of a large crystal from the crystal generator. This structure is so huge that it needs 10 Pieces where each of them has 32000 vertices as the limit is set to 32000.

When you set the game object to static, generated stuff by this system will also be static. By default, the generation and their pieces are hidden from the hierarchy. You can hide or unhide them. When set to hide, generations will also be hidden.

If you change the material, it changes the material of all mesh pieces. Additionally if you only have one piece, you can flag the Single Mesh Only flag which hard limits the maximum vertex count which means that the generation will stop when the limit is reached resulting in incomplete results. If Single Mesh Only is set, you can also bind the result as a component to the generator by setting the “Single Mesh to Root” flag.

Results can have colliders if the algorithms enforce it. The settings “Convex Collider” or “Collider Is Trigger” will affect all generated colliders.

Meshes can be created as animations. Assets like the Ornament Generator allow the creation piece by piece which is optimized. The meshes are generated using divide and conquer methods and the result will be merged meshes with the maximum vertex counts. This optimization can be turned off by setting “Supress Merges” to true. The result will be more pieces with lesser vertices which are often wanted when they can be destroyed.

The state of this sub system is **FINAL** as it was developed in a time before Burst Compilation existed.

EXPORTING:

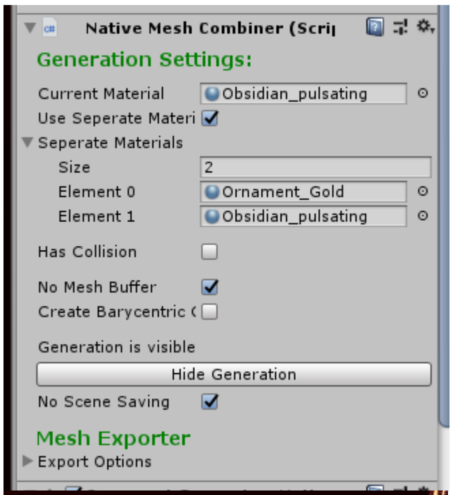
You can export every individual mesh piece by clicking on one of those buttons. The standard path is the assets folder. However you can set your custom path and mesh name. You currently can export as .asset or .obj file.

Keep in mind that you need every piece in order to get the full mesh you want to export if the amount of pieces is greater than 1.

CUSTOMISATION:

The script “MeshPieceAttachment” is a simple script with a virtual effect function. This function is called whenever a new piece is added. By implementing children of this class and overriding the effect function, you can alter the properties of the pieces. Attach your custom derivate as component to the game object which has the LargeMeshCollider on it to use it.

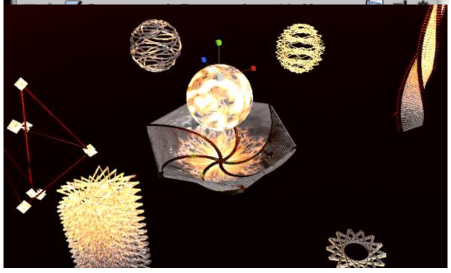
NATIVE MESH COMBINER: COMPLETE


Native Mesh Combiner (Scri

Generation Settings:

Current Material: Obsidian_pulsating
Use Seperate Materi: ✓
Seperate Materials
Size: 2
Element 0: Ornament_Gold
Element 1: Obsidian_pulsating
Has Collision: ☐
No Mesh Buffer: ✓
Create Barycentric (: ☐
Generation is visible
Hide Generation
No Scene Saving: ✓

Mesh Exporter
Export Options



The Native Mesh Combiner is a new core system in some of my assets which utilize the burst compiler for increased performance. Scripts which generate meshes and have Ultra or Native in their name will require this script. The settings are almost similar to the LargeMeshCombiner with slight differences.

Since Unity 2018.1.0 the maximum amount of vertices a mesh can have increased from 64000 to 4 billion vertices. This allows the creation of meshes with an enormous vertex density. However such creation is too slow for real time application without the usage of the new burst compiler and job system.

The screenshot shows meshes generated by the Ornamen Generator. These are generated and animated in real time. Now the most expensive calculation is the collider generation which is still very complex.

When you set the game object to static, generated stuff by this system will also be static. By default, the generation and their pieces are hidden from the hierarchy. You can hide or unhide them. When set to hide, generations will also be hidden.

If you change the material, it changes the material of all mesh pieces. But it is also possible to use separate materials for every slot. For example the Ornament Generator uses 2 slots. One for crystals and one for pipes.

Additionally if you only have one piece, you can flag the Single Mesh Only flag which hard limits the maximum vertex count which means that the generation will stop when the limit is reached resulting in incomplete results. If Single Mesh Only is set, you can also bind the result as a component to the generator by setting the “Single Mesh to Root” flag.

Results can have colliders if the algorithms enforce it. The settings “Convex Collider” or “Collider Is Trigger” will affect all generated colliders. Also this core component requires the Burst Package to be installed and every Burst placeholder to be removed in order to get the best performance.

If “NoSceneSaving” is flagged, the generated mesh will not be saved into the scene. Since the NativeMeshCombiner can generate meshes in real time, it is possible to simply generate procedural meshes after scene loading so these highly complex meshes do not have to be stored inside the scene which usually causes the file size to explode and increases loading times.

However this also means that such meshes are not stored inside the scene and must be rebuilt in order to see the resulting mesh. Therefore the script using the NativeMeshCombiner is responsible for the regeneration of the mesh.

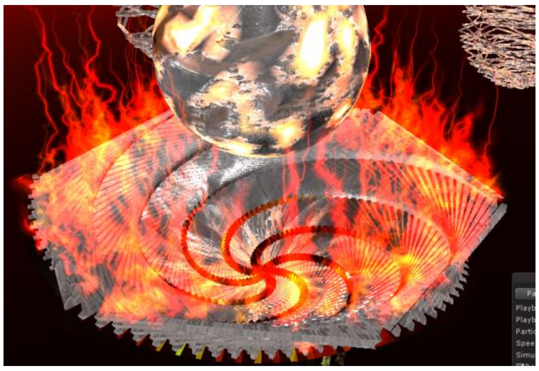
The NativeMeshCombiner also has export options similar to the LargeMeshCombiner. The standard path is the assets folder and can be set to your custom path and mesh name.

A new option is **Create Barycentric Colors** which writes barycentric coordinates into the vertex colors which can be used by wireframe or seamless tessellation shader.

CUSTOMISATION*:*

The final result can be further modified by simply having specific scripts attached to the game object containing the Native Mesh Combiner.

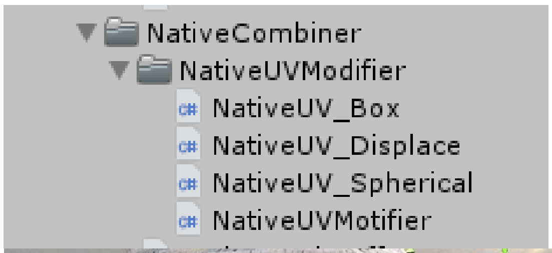
AUTOMATIC COMPONENT ATTACHMENT:



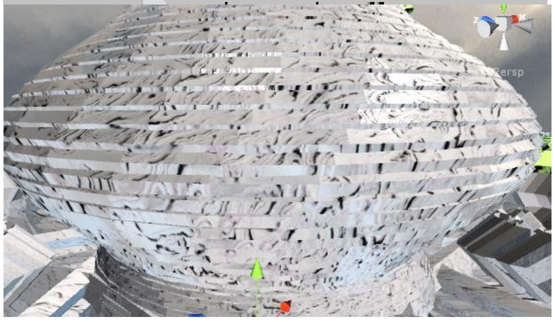
The script “MeshPieceAttachment” is a simple script with a virtual effect function. This function is called whenever a new piece is added. By implementing children of this class and overriding the effect function, you can alter the properties of the pieces. Attach your custom derivate as component to the game object which has the NativeMeshCombiner on it to use it.

The most interesting attachment is the MeshPieceAttachment_Particle which adds particle effects to the generated mesh. For more information, check the Mesh Piece Attachment section.

UV MODIFICATION:



- NativeCombiner
  - NativeUVModifier
    - NativeUV_Box
    - NativeUV_Displace
    - NativeUV_Spherical
    - NativeUVMotifier



The NativeMeshCombiner allows the manipulation of the UV coordinates of the resulting mesh. To manipulate the UV coordinates, simply attach components which derive from NativeUVModifier to the game object containing the NativeMeshCombiner. The modifications stack so multiple UV modifiers can be combined. For example the procedural generated object below uses UV_Spherical and UV_Displace to generate a non-repetitive appearance.

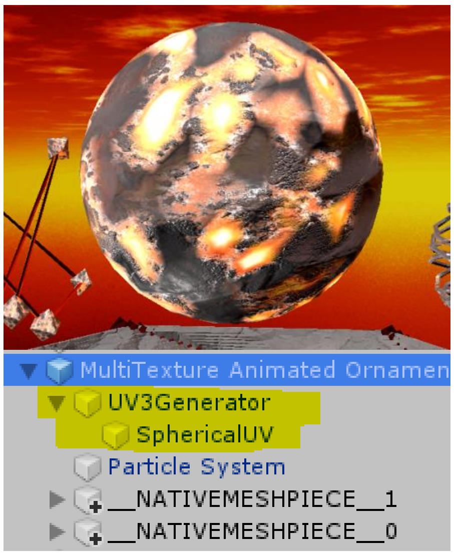
Creating custom UV_Modifier is a more complicated process because it utilizes burst compilation for increased performance. Therefore coding knowledge about unity job system is crucial. However to simplify the creation, a template file is included. The template file is supposed to be duplicated and renamed. It contains 3 classes which are required:

- TEMPLATE_Data: This struct is used as data container for the job system and should contain all data which can be set in the inspector
- TEMPLATE: The monobehaviour class which can be attached as component to an game object.
- TEMPLATE_Job: This is the struct which uses burst compilation. Your desired algorithm should be implemented inside the Execute function. The template implements spherical UV mapping as example.

PROCEDURAL UV GENERATION

Mesh generated by the Native Mesh Combiner can be further enhanced by modifying the additional UV channels which are UV3, UV4, UV5, UV6, UV7 and UV8. UV2 is used by Unity so it is not possible to modify this channel.

To modify an additional UV channel, simply attach a GameObject containing a ProceduralUVGenerator script to the Native Mesh Combiner GameObject. The ProceduralUVGenerator then require algorithms to generate the data which is then written into the UV channel defined inside the UV generator.



MESH PIECE ATTACHMENT COMPLETE

Mesh piece attachments are scripts which further modify procedurally generated content which are usually 3D Meshes. Their effect is utilized by the procedural generation system as soon as it is attached as component. Some procedural generators use multiple slots.

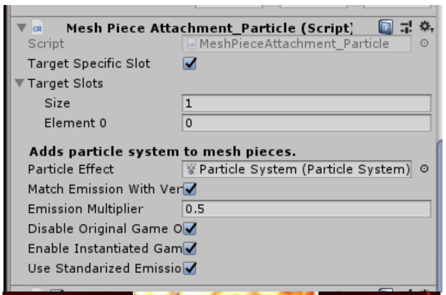
So if multiple slots are used, the attachment is applied to every slot unless **Target Specific Slot** is checked. If checked, only slots defined inside the **Target Slots** list are modified.

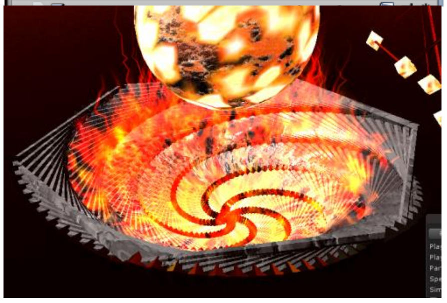
Every attachment has 2 main functions which can be overridden for custom attachments:

`Effect(GameObject piece):` `Is called whenever the attachment is applied. This is called once when a new game object is generated.`

`UpdatePiece(GameObject piece, Mesh proceduralMesh):` `Some effects require updating. This function is called whenever the procedural mesh is modified.`

PARTICLE


```
Mesh Piece Attachment_Particle (Script)
Script                     MeshPieceAttachment_Particle
Target Specific Slot       ✓
Target Slots
  Size                     1
  Element 0                0

Adds particle system to mesh pieces.
Particle Effect            Particle System (Particle System)
Match Emission With Ver    ✓
Emission Multiplier        0.5
Disable Original Game O    ✓
Enable Instantiated Gam    ✓
Use Standarized Emissio    ✓
```



The most interesting attachment is the MeshPieceAttachment_Particle as it adds a particle effect to the mesh pieces. It requires a particle system as input parameter. The attachment adds the assigned particle system to the mesh pieces.

When **MatchEmissionWithVertices** is true, the emission rate is automatically set so the amount of emission per vertex remains constant. This is important when vertex count of the mesh is not constant.

If **UseStandarizedEmission** is true, the emission rate of the original particle system is always 100. The emission amount is then only adjustable using the **Emission Multiplier**. A good way to design the particle system is to modify the attached particle system and then copy the values and paste it to the original particle system. As the emission is set procedurally, the emission value does not impact the original particle system when pasting the modified parameters over the original particle system.

DISABLE RENDERER

This attachment simply disables the mesh renderer of the procedurally generated game object. The result is then invincible. This is one of the most simple mesh piece attachments possible.

DECAY

Also pretty simple as this attachments attaches a timer which will destroy the procedurally generated game object when the timer expires.

PROCEDURAL UV GENERATOR RELEASED

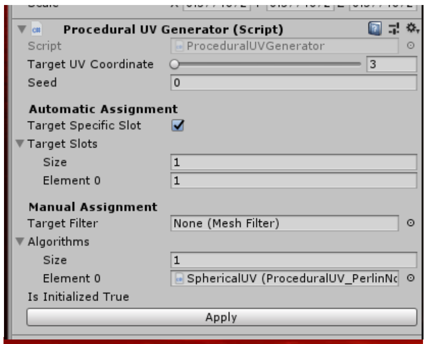
The Procedural UV Generator generates data for additional UV coordinates which are between UV3-8. This generator is used by the Native Mesh Combine to define which slot should be modified. Therefore it has the option to target all generated slots or just specific ones defined inside the Target Slots list.

Otherwise it is also possible to just assign a target filter manually if nothing else is automatically using the generator.

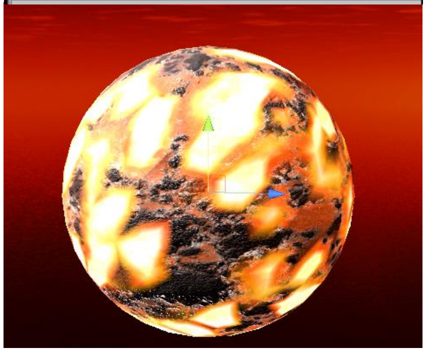
The Seed is used for randomness.

The generated UV data is calculated using algorithms. For example the right sphere is the default sphere mesh but modified using PerlinNoise_Sphere algorithm.

After writing the additional data into the UV coordinate, no changes will be visible unless a shader is using those coordinates. Therefore the samples are using the Multi-Material shader which is included.



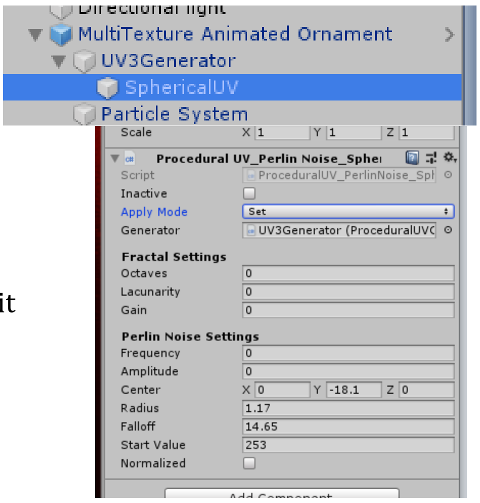
```
Procedural UV Generator (Script)
Script                   ProceduralUVGenerator
Target UV Coordinate     3
Seed                     0

Automatic Assignment
Target Specific Slot     ✓
Target Slots
    Size                 1
    Element 0            1

Manual Assignment
Target Filter            None (Mesh Filter)
Algorithms
    Size                 1
    Element 0            SphericalUV (ProceduralUV_PerlinNc
Is Initialized True
```



ALGORITHMS

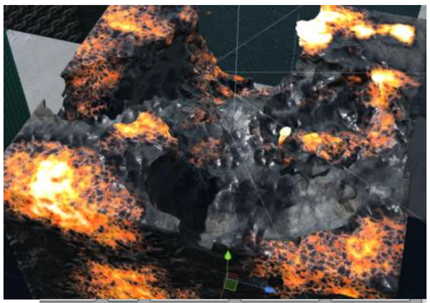
In order to generate data, you have to attach at least one game object containing a Procedural UV component. The generator will update the result whenever you change the parameters so it is recommended to play around with the prefabs provided in the sample scenes.

Also the apply mode allows you to mix together different algorithms. The first algorithm has the apply mode “Set” and other ones should use different apply modes except “Set” as it would overwrite the previous computation.



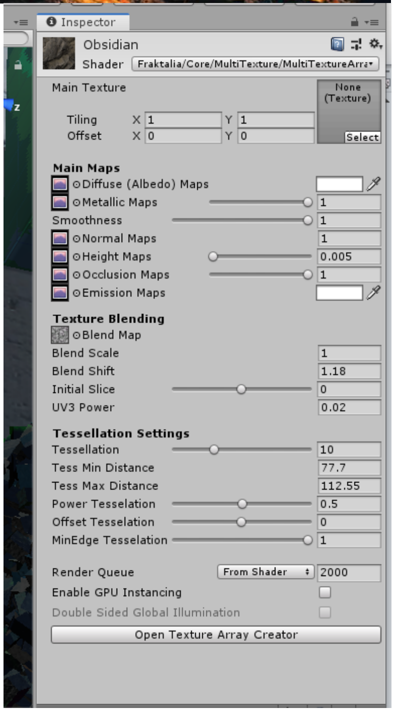
```
MultiTexture Animated Ornament
    UV3Generator
        SphericalUV
    Particle System

Scale                X 1    Y 1    Z 1

Procedural UV_Perlin Noise_Sphe
Script               ProceduralUV_PerlinNoise_Spl
Inactive             ☐
Apply Mode           Set
Generator            UV3Generator (ProceduralUV(

Fractal Settings
Octaves              0
Lacunarity           0
Gain                 0

Perlin Noise Settings
Frequency            0
Amplitude            0
Center               X 0    Y -18.1    Z 0
Radius               1.17
Falloff              14.65
Start Value          253
Normalized           ☐
```

MULTI MATERIAL RELEASED



Multi-Material features are implemented gradually during the development of the Voxel Generator but other assets are also starting to use this functionality.

Instead of normal texture2D as input parameter, TextureArrays are used instead. For the texture lookup we usually use UV coordinates but an additional Index coordinate is used to define which texture should be read. This allows seamless blending between textures. Unfortunately Unity does not provide a way to create texture arrays. Therefore a tool is included which bakes normal textures into texture arrays.



Obsidian

Shader Fraktalia/Core/MultiTexture/MultiTextureArra

Main Texture: None (Texture) Select

	X	Y
Tiling	1	1
Offset	0	0

Main Maps

Diffuse (Albedo) Maps	
Metallic Maps	1
Smoothness	1
Normal Maps	1
Height Maps	0.005
Occlusion Maps	1
Emission Maps	

Texture Blending

Blend Map	
Blend Scale	1
Blend Shift	1.18
Initial Slice	0
UV3 Power	0.02

Tessellation Settings

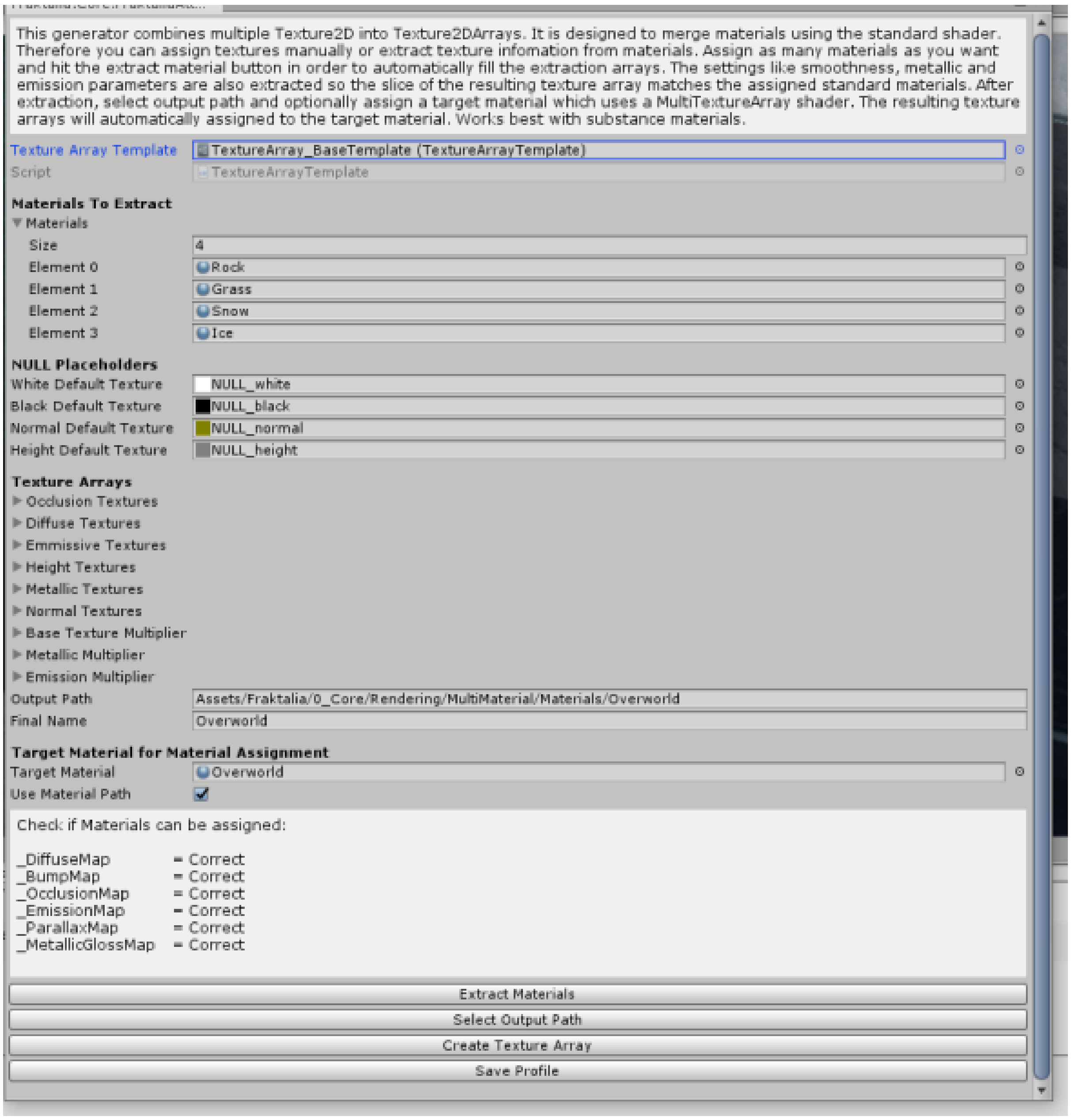
Tessellation	10
Tess Min Distance	77.7
Tess Max Distance	112.55
Power Tesselation	0.5
Offset Tesselation	0
MinEdge Tesselation	1

Render Queue	From Shader 2000
Enable GPU Instancing	☐
Double Sided Global Illumination	☐

The voxel block in the right sample uses the Obsidian material which uses is a surface shader copying the normal standard shader but uses TextureArrays instead. As a bonus, a tessellation version is also included which is seamless if barycentric coordinates are written into the vertex colors.

The shader uses the UV3 coordinate as Index value which is modified by the Hull Generators (Voxel Generator) or by the Procedural UV Generator which is included in every asset related to procedural content generation.

Also this material directly gives you access to the texture array creator



This generator combines multiple Texture2D into Texture2DArrays. It is designed to merge materials using the standard shader. Therefore you can assign textures manually or extract texture infomation from materials. Assign as many materials as you want and hit the extract material button in order to automatically fill the extraction arrays. The settings like smoothness, metallic and emission parameters are also extracted so the slice of the resulting texture array matches the assigned standard materials. After extraction, select output path and optionally assign a target material which uses a MultiTextureArray shader. The resulting texture arrays will automatically assigned to the target material. Works best with substance materials.

Texture Array Template	TextureArray_BaseTemplate (TextureArrayTemplate)
Script	TextureArrayTemplate

Materials To Extract

▼ Materials

Size	4
Element 0	Rock
Element 1	Grass
Element 2	Snow
Element 3	Ice

NULL Placeholders

White Default Texture	NULL_white
Black Default Texture	NULL_black
Normal Default Texture	NULL_normal
Height Default Texture	NULL_height

Texture Arrays

- ▶ Occlusion Textures
- ▶ Diffuse Textures
- ▶ Emmissive Textures
- ▶ Height Textures
- ▶ Metallic Textures
- ▶ Normal Textures
- ▶ Base Texture Multiplier
- ▶ Metallic Multiplier
- ▶ Emission Multiplier

Output Path	Assets/Fraktalia/0_Core/Rendering/MultiMaterial/Materials/Overworld
Final Name	Overworld

Target Material for Material Assignment

Target Material	Overworld
Use Material Path	✓

```
Check if Materials can be assigned:

_DiffuseMap        = Correct
_BumpMap           = Correct
_OcclusionMap      = Correct
_EmissionMap       = Correct
_ParallaxMap       = Correct
_MetallicGlossMap  = Correct
```

This tool allows the creation of texture arrays by filling the texture arrays. Since you would work with many textures, the tool comes with a functionality to directly extract the information from individual standard materials. Substance Materials are ideally suited to be extracted.

Therefore the easiest way is to fill the materials list with materials. The index will then match the order of the list. In the image above, Rock is the first and Ice is the last one. When clicking Extract Materials, the assigned materials are scanned and the texture arrays are filled with textures, ready to be baked.

Also the extractor considers the color settings of the assigned material (Main Color, Metallic, Smooth, Emission)

You can optionally assign a target material so the texture arrays are automatically assigned to the material so you can immediately see the result.

LAST NOTES:

If you have any questions about this system, suggestions, bug reporting don't hesitate to contact me

Contact Information:

E-Mail: m.hartl@fraktalia.org

Homepage: http://fraktalia.org/