Mesh Materializer API is accessible:

- (c#) `using VacuumShaders.MeshMaterializer;`
- (java) `import VacuumShaders.MeshMaterializer;`

## Simple and Skinned mesh materializer

```
static public Mesh MaterializeMesh(Renderer _renderer, out CONVERTION_INFO[] _buildInfo,
                                   out string[] _buildInfoFull, params MMData[] _data)
```

Function returns converted mesh.

_renderer – active gameobject renderer.

_buildInfo – this variable will contain conversion info.

_buildInfoFull – this variable will contain conversion info with detail explanation.

_data – array of conversion data. Available data type and their parameters are exactly same as inside **Mesh Materializer** window:

- MMData_SurfaceInfo
- MMData_MeshTextureAndColor
- MMData_MeshDisplace
- MMData_AmbientOcclusion
- MMData_IndirectLighting
- MMData_Lightmap
- MMData_Optimize

Note:

Textures and Models need to be readable.
Unity readable texture formats are - ARGB32, RGBA32, BGRA32, RGB24, Alpha8 and DXT.

If conversion fails, returned mesh will be null, _buildInfo and _buildInfoFull will contain info about fail.

## Simple Mesh combiner and materializer

```
static public COMBINE_INFO CombineMeshes(Transform _parent, out Mesh _combinedMesh)
```

Combines all meshes (including child) in _parent and returns result in _combineMesh.

`COMBINE_INFO` contains info about combine (success or problems).

```
static public COMBINE_INFO CanBeCombined(Transform _parent)
```

Checks if meshes (including child) in _parent can be combined.

```
static public Mesh MaterializeMeshGroup(Transform _parent, out CONVERTION_INFO[] _buildInfo,
                                        out string[] _buildInfoFull, params MMData[] _data)
```

Same function as `MaterializeMesh()` but with combining meshes (including child) of _paranet.

## Terrain Conversion and maps extractor

```
static public Mesh[] MaterializeTerrain(Terrain _terrain, out CONVERTION_INFO[] _buildInfo,
                                        out string[] _buildInfoFull, params MMData[] _data)
```

Function returns converted terrain as mesh array.

_terrain – active terrain object.

_buildInfo – this variable will contain conversion info.

_buildInfoFull – this variable will contain conversion info with detail explanation.

_data – array of conversion data. Available data type and their parameters are exactly same as inside **Mesh Materializer** window:

- MMData_SurfaceInfo
- MMData_TerrainToMesh
- MMData_TerrainTexture
- MMData_AmbientOcclusion
- MMData_IndirectLighting
- MMData_Lightmap
- MMData_Optimize

```
static public Texture2D[] ExtractTerrainSplatmaps(Terrain _terrain)
```

Extracts terrain splatmaps, if has any.

```
static public void ExtractTerrainBasemap(Terrain _terrain, out Texture2D _diffuseMap,
                                         out Texture2D _normalMap, int _width, int _height)
```

Extracts terrain basemap as diffuse and normal textures.

_diffuseMap and _normalMap will be null if _terrain has no such data.

```
static public Texture2D ExtractTerrainHeightmap(Terrain _terrain, bool _remap)
```

Extracts terrain heightmap.

_remap – Remaps texture values to be inside [0, 1] range.

```
static public int ExtractTerrainTexturesInfo(Terrain _terrain,
                                             out Texture2D[] _diffuseTextures,
                                             out Texture2D[] _normalTextures,
                                             out Vector2[] _uvScale, out Vector2[] _uvOffset)
```

Extracts terrain's paint texture info.

# MMData classes

```
public class MMData_SurfaceInfo : MMData
{
	public enum SURFACE_TYPE { Original = 0, Flat }

	public SURFACE_TYPE surfaceType;
	public bool forceOneSubMaterial;
	public Color defaultColor;
}

public class MMData_TerrainToMesh : MMData
{
	public bool isEnabled;

	public int chunkCountHorizontal;
	public int chunkCountVertical;
	public int vertexCountHorizontal;
	public int vertexCountVertical;
}

public class MMData_TerrainTexture : MMData
{
	public enum SAMPLING_TYPE { Smooth, FlatHard, FlatSmooth, FlatSmoother }
	public enum ALPHA { TextureAlpha, TextureAlphaInvert, One, Zero}

	public bool isEnabled;

	public SAMPLING_TYPE samplingType;
	public ALPHA alpha;

	public bool useMipmap;
	[Range(0.0f, 1.0f)]
	public float mipmapBias;

	public bool useBlur;
	[Range(1, 64)]
	public int blurAmount;
	[Range(0, 5)]
	public int blurDownSample;
}
```

```
public class MMData_MeshTextureAndColor : MMData
{
	public enum SAMPLING_TYPE { Smooth, FlatHard, FlatSmooth, FlatSmoother }
	public enum ALPHA { MainTextureAlpha, MainTextureAlphaInvert, One, Zero, SeconTextureAlpha,
				SeconTextureAlphaInvert, BlendAdd, BlendMultiply, BlendDecal, VertexAlpha,
				VertexAlphaInvert }
	public enum BLEND_TYPE { Add, Multiply, Decal, Detail, MainTextureAlpha, MainTextureAlphaInvert,
					SecondTextureAlpha, SecondTextureAlphaInvert, VertexColorAlpha,
					VertexColorAlphaInvert }


	public bool isEnabled;
	public SAMPLING_TYPE samplingType;
	public bool includeVertexColor;
	public string colorName;

	public string texture_1_Name;
	public bool texture_1_useMipmap;
	[Range(0.0f, 1.0f)]
	public float texture_1_mipmapBias;
	public bool texture_1_useBlur;
	[Range(1, 64)]
	public int texture_1_blurAmount;
	[Range(0, 5)]
	public int texture_1_blurDownSample;

	public string texture_2_Name;
	public BLEND_TYPE texture_2_blendType;
	public bool texture_2_useMipmap;
	[Range(0.0f, 1.0f)]
	public float texture_2_mipmapBias;
	public bool texture_2_useBlur;
	[Range(1, 64)]
	public int texture_2_blurAmount;
	[Range(0, 5)]
	public int texture_2_blurDownSample;

	public ALPHA alpha;
}
```

```
public class MMData_MeshDisplace : MMData
{
	public enum READ_FROM { R, G, B, A, Grayscale }
	public enum SAVE_TYPE { DisplaceVertex, SaveToColor, DisplaceVertexAndSaveToColor }
	public enum SAVE_CHANNEL { RGB, Alpha }


	public bool isEnabled;

	public string textureName;
	public READ_FROM readFrom;
	public float strength;
	public float power;
	public bool useMipmap;
	[Range(0.0f, 1.0f)]
	public float mipmapBias;
	public bool useBlur;
	[Range(1, 64)]
	public int blurAmount;
	[Range(0, 5)]
	public int blurDownSample;

	public SAVE_TYPE saveType;
	public SAVE_CHANNEL saveChannel;

	public bool recalculateNormals;
	public bool recalculateTangents;
}

public class MMData_IndirectLighting : MMData
{
	public enum SOLVER { SolidColor, CubeMap, SceneSkybox }
	public enum SAMPLING_TYPE { Smooth, Flat }

	public bool isEnabled;

	public SAMPLING_TYPE samplingType;
	public SOLVER solver;
	public Color skyColor;
	public Cubemap cubeMap;
	public float intensity;
	public float contrast;
	public bool useMipMps;
	[Range(0.0f, 1.0f)]
	public float mipmapBias;
	[Range(-1.0f, 1.0f)]
	public float lightStrength;
}
```

```
public class MMData_Lightmap : MMData
{
	public enum SAMPLING_TYPE { Smooth, FlatHard, FlatSmooth, FlatSmoother }
	public enum SAVE_CHANNEL { RGB, Alpha }


	public bool isEnabled;

	public SAMPLING_TYPE samplingType;
	public SAVE_CHANNEL saveChannel;
	public float gamma;
	public float lightmapHDR;
	[Range(0.0f, 1.0f)]
	public float intensity;
	public bool grayscale;
	public bool useMipmap;
	[Range(0f, 1f)]
	public float mipmapBias;
	public bool useBlur;
	[Range(1, 64)]
	public int blurAmount;
	[Range(0, 5)]
	public int blurDownSample;

	public bool readLightmapUsingUV1;
}



public class MMData_Optimize : MMData
{
	public bool isEnabled;

	public bool saveUV;
	public bool saveUV2;
	public bool saveUV3;
	public bool saveUV4;
	public bool saveNormal;
	public bool saveTangent;
	public bool saveColor;
	public bool saveSkin;
}
```

# Color Adjustment

Color Adjustment API is accessible:

- (c#) `using VacuumShaders.MeshMaterializer.ColorAdjustment;`
- (java) `import VacuumShaders.MeshMaterializer.ColorAdjustment;`

```
public class CAData_HueSaturationLightness : CAData
{
	public bool isEnabled;


	[Range(-180.0f, 180.0f)]
	public float hue = 0.0f;

	[Range(-100.0f, 100.0f)]
	public float saturation = 0.0f;

	[Range(-100.0f, 100.0f)]
	public float lightness = 0.0f;
}

public class CAData_BrightnessContrast : CAData
{
	public bool isEnabled;

	[Range(-1.0f, 1.0f)]
	public float brightness = 0.0f;

	[Range(-1.0f, 1.0f)]
	public float contrast = 0.0f;

	[Range(0.0f, 1.0f)]
	public float redCoeff = 0.5f;
	[Range(0.0f, 1.0f)]
	public float greenCoeff = 0.5f;
	[Range(0.0f, 1.0f)]
	public float blueCoeff = 0.5f;
}
```

```
public class CAData_Levels : CAData
{
	public enum CHANNEL { RGB, R, G, B }


	public bool isEnabled;
	public CHANNEL channel;

	public float inputMin = 0.0f;
	public float inputMax = 255.0f;
	[Range(0.1f, 3.0f)]
	public float inputGamma = 1.0f;
	public float outputMin = 0.0f;
	public float outputMax = 255.0f;

	public float inputMinR = 0.0f;
	public float inputMaxR = 255.0f;
	[Range(0.1f, 3.0f)]
	public float inputGammaR = 1.0f;
	public float outputMinR = 0.0f;
	public float outputMaxR = 255.0f;

	public float inputMinG = 0.0f;
	public float inputMaxG = 255.0f;
	[Range(0.1f, 3.0f)]
	public float inputGammaG = 1.0f;
	public float outputMinG = 0.0f;
	public float outputMaxG = 255.0f;

	public float inputMinB = 0.0f;
	public float inputMaxB = 255.0f;
	[Range(0.1f, 3.0f)]
	public float inputGammaB = 1.0f;
	public float outputMinB = 0.0f;
	public float outputMaxB = 255.0f;
}


public class CAData_ColorSpace : CAData
{
	public enum COLOR_SPACE { Original, Gamma, Linear }


	public bool isEnabled;
	public COLOR_SPACE colorSpace;
}
```

```
public class CAData_ColorOverlay : CAData
{
	public enum BLEND_MODE { Normal, Darken, Multiply, ColorBurn, LinearBurn, Lighten, Screen,
					ColorDodge, LinearDodge, Overlay, HardLight, VividLight, LinearLight,
					PinLight, HardMix, Difference, Exclusion, Subtract, Divide }

	public bool isEnabled;

	public BLEND_MODE blendMode;
	public Color color;
	[Range(0.0f, 1.0f)]
	public float blendIntensity;
}



public class CAData_Invert : CAData
{
	public bool isEnabled;

	public bool invertR;
	public bool invertG;
	public bool invertB;
}

public class CAData_Alpha : CAData
{
	public bool isEnabled;

	public bool invert;
	public float power;
	[Range(-1f, 1f)]
	public float offset;
	public float strength;
}
```

`public Color Apply(Color _srcColor)` – Applies adjustment of the current class to the _srcColor