Flat Mesh Generator API is accessible:

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

Simple and Skinned mesh conversion

```
static public Mesh GenerateFlatMesh(Renderer _renderer, out CONVERTION_INFO[] _buildInfo,
                                    out string[] _buildInfoFull, FlatMeshOptions _data)
```

_renderer – active gameobject renderer (must contain mesh and material components).

_buildInfo – this variable will contain conversion info.

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

_data – `FlatMeshOptions` class object contain info for generating flat mesh.

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 detail info.

Simple Mesh combiner

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

Combines child meshes in _parent and returns result in _combineMesh.

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

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

Checks if meshes in _parent can be combined.

```
static public Mesh GenerateFlatMeshesAndThenCombine(Transform _parent, out CONVERTION_INFO[]
                                          _buildInfo, out string[] _buildInfoFull,
                                          FlatMeshOptions _data)
```

Same function as `GenerateFlatMesh()` but with combining meshes of _paranet.

Terrain Conversion

```
static public Mesh[] GenerateFlatTerrain(Terrain _terrain, out CONVERTION_INFO[] _buildInfo, out
                                   string[] _buildInfoFull, FlatTerrainOptions _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 – `FlatTerrainOptions` class object contain info for generating flat mesh.

```csharp
public class FlatMeshOptions
{
    //Enum////////////////////////////////////////////////////////////////
    public enum SAMPLING_TYPE { Hard, Smooth }
    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
                           }

    //Variables////////////////////////////////////////////////////////////
    public SAMPLING_TYPE samplingType;

    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 string colorName;

    public bool   includeVertexColor;
    public ALPHA  alpha;
    public bool   mergeSubMaterials;
    public bool   combineMeshes;
}
```

```csharp
public class FlatTerrainOptions
{
    //Enum////////////////////////////////////////////////////////////////
    public enum SAMPLING_TYPE { Hard, Smooth }
    public enum ALPHA { TextureAlpha, TextureAlphaInvert, One, Zero }

    //Variables///////////////////////////////////////////////////////////
    public int chunkCountHorizontal;
    public int chunkCountVertical;
    public int vertexCountHorizontal;
    public int vertexCountVertical;

    public SAMPLING_TYPE samplingType;

    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 ALPHA alpha;
}
```

Check run-time example scenes.