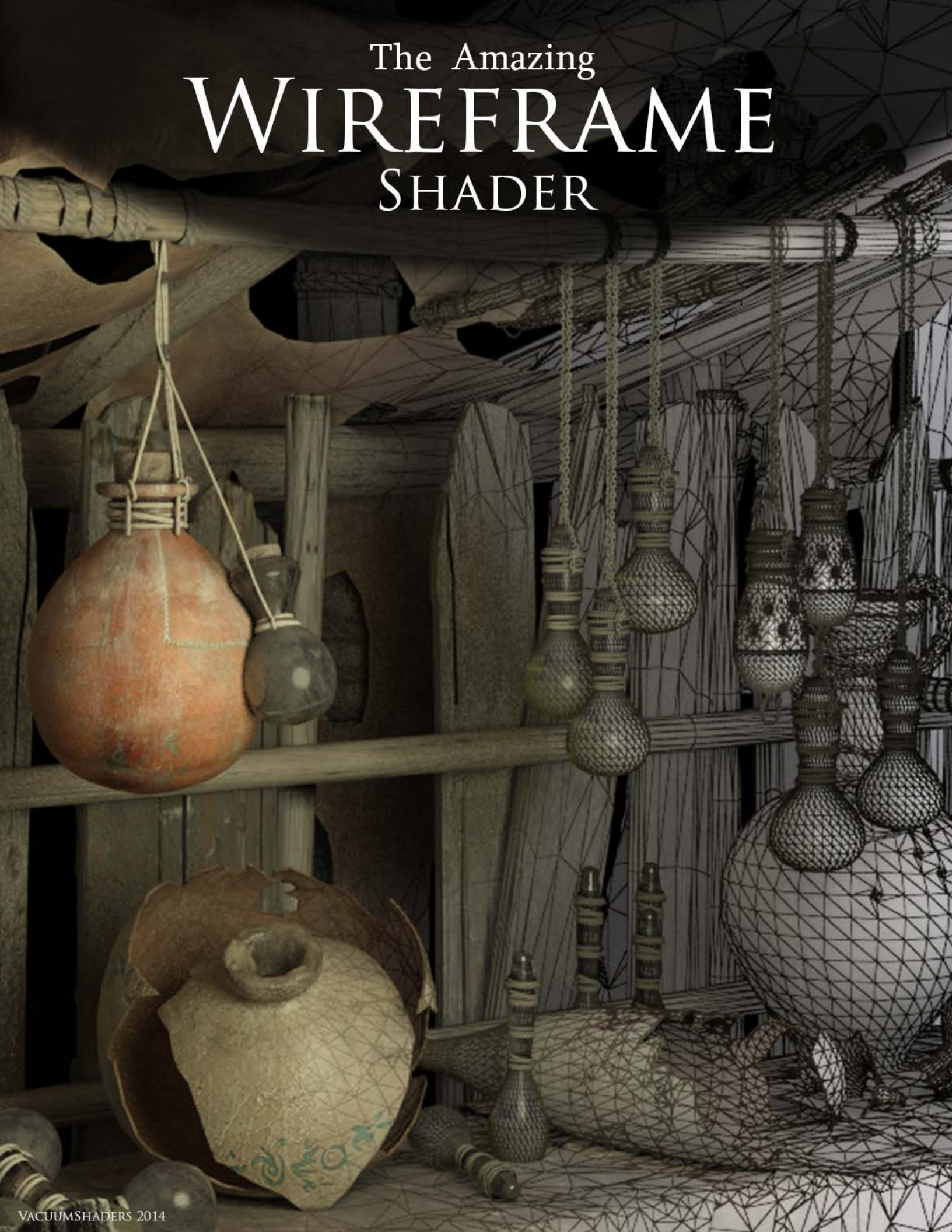
The Amazing WIREFRAME SHADER



VACUUMSHADERS 2014

Thanks for purchasing **The Amazing Wirframe** shader.

Please consider leaving a review or just rating the asset if you find it useful.

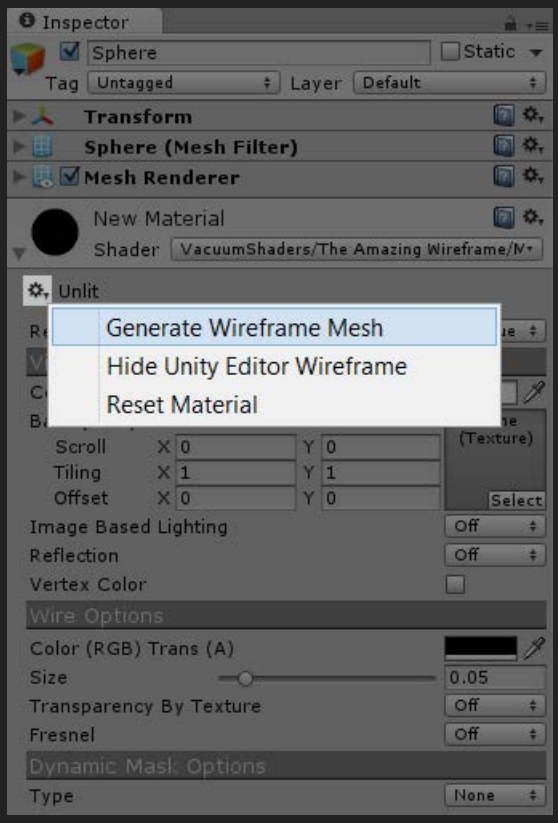
Thanks!

Quick Start

Create material and set wireframe shader VacuumShaders/The Amazing Wireframe/Mobile/Unlit. Apply material to a mesh.

Mesh will be rendered completely black (color of wireframe by default), it means that mesh has no barycentric coordinates for wireframe rendering.

For generating mesh with barycentric coordinates select mesh inside scene (Hierarchy window) and inside material editor click on 'Gear' icon, choose Generate Wireframe Mesh option.



Generated mesh asset file will be saved inside Assets/(Temporary)/Wireframe folder and automatically replace original mesh from MeshFilter or SkinnedMeshRenderer.

Now shader will render wireframe.

There 3 options for wireframe rendering:

1. Baking baricentric coordinates inside mesh

Pros:

- Can be used on any device
- Fast during rendering
- Supports quad wireframe rendering.

Cons:

- Only mesh with triangle count less than 21,000 can be used. (mesh with more triangles are not guaranteed to be converted)
- Run-time baking may be slow. Package includes two functions for that purpose: Fast (non-optimized) and slow (generates optimized mesh)
- Not all mesh features are supported, e.g. BlendShapes
- Meshes with multiple sub-materials during baking will be converted into 1 sub-material.

2. Reading wireframe from simple texture

Pros:

- No mesh data baking is required
- Can be used on any device
- No limit on triangle count
- All mesh features are supported, including BlendShapes and multiple sub-material.
- Fastest rendering
- Supports quad wireframe rendering (technically any shape of wireframe can be rendered)

Cons:

- Required texture must be in high resolution (not required, but for better visual it's desired)
- Can be problematic for run-time generated meshes, unless all required textures are provided.

3. Dynamically calculating barycentric coordinates inside shader

Pros:

- No mesh data baking is required
- No limit on triangle count
- All mesh features are supported, including BlendShapes and multiple sub-material
- Fast rendering. Even barycentric data is generated inside shader, their calculation needs 0 (zero) instruction
- Perfect for run-time generated meshes
- Supports tessellation shaders

Cons:

- Requires device with GeometryShader support
- Cannot render quad wireframe

Material editor features are self-explanatory, but some of them need more instructions:

Dynamic Mask

Dynamic mask is world-space modifier defining areas where wireframe is visible.

Dynamic mask object controller parameters are not displayed inside material editor and can only be updated from script.

- Plane – Wireframe visibility is defined by plane equation and takes into account two parameters:
    1. Plane world-space position `half3 _V_WIRE_DynamicMaskWorldPos`
    2. Plane world-space normal `half3 _V_WIRE_DynamicMaskWorldNormal`

    (Wireframe is visible only on positive side of the plane)

- Sphere - Wireframe visibility is defined by sphere equation and takes into account two parameters:
    1. Sphere world-space position `half3 _V_WIRE_DynamicMaskWorldPos`
    2. Sphere radius `half _V_WIRE_DynamicMaskRadius`

    (Wireframe is visible only inside sphere)

- Box - Wireframe visibility is defined by Unity built-in Cube mesh and shader needs three parameters:
    1. Box world-space size(scale) is defined by two parameters `half3 _V_WIRE_DynamicMaskBoundsMin` and `half3 _V_WIRE_DynamicMaskBoundsMax`
    2. Position and rotation with `_V_WIRE_DynamicMaskTRS` matrix. It transform-rotation-scale matrix calculated by `Matrix4x4.TRS().inverse;` method.

Check Example_2 (Mask - Plane), Example_3 (Mask - Sphere) and Example_3 (Mask - Box) example scenes.

Global Illumination

If using together with world-space modifiers like Distance Fade or Dynamic Mask wire object position must be updated from script.

Shader expects wire object position (world-space) to be inside `half3 _V_WIRE_ObjectWorldPos` variable.

Check Example_10 (Dynamic GI - Distance Fade) example scene.

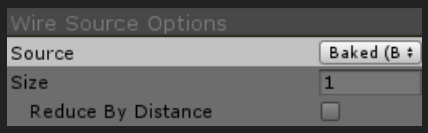
Unity Standard shaders

All shaders inside package share same options and effects. Except **Standard** shaders.

Not all wireframe effects are available within **Standard** shaders.

Projector

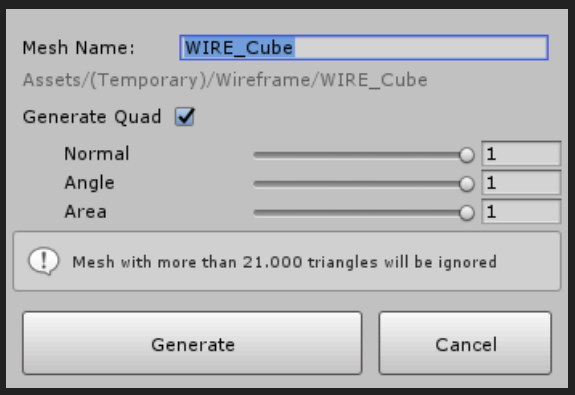
Effect completely relies on the Unity Projector effect. Available two options (Source) for wireframe rendering:



- Baked – Mesh projecting on must have barycentric coordinates baked inside.
- Dynamic – Uses geometry shader and does not need mesh with baked barycentric coordinates.

In both cases mesh can use any shader, not necessary Wireframe shader.

Check Example_4 (Projector) example scene.



Wireframe (barycentric coordintes) baking

Barycentric coordinates can be baked suing material editor. If mesh is selected inside scene, then material editor will have enabled 'Gear' icon with menu option for mesh baking.

Can generate Triangle and Quad wirefarmes.

Triangle wireframe is guaranteed on mesh with less than 21,000 triangles. If mesh uses more triangles baking process may fail (due to increasing vertex count during baking) and no data will be baked inside mesh.

Quad wireframe can be baked with mesh using less than 21,000 triangles.
Quad shape rendering is not 100% guaranteed. It completely relies on original mesh triangle and vertex structure.
Some parts of quad mesh still can be rendered as triangle.

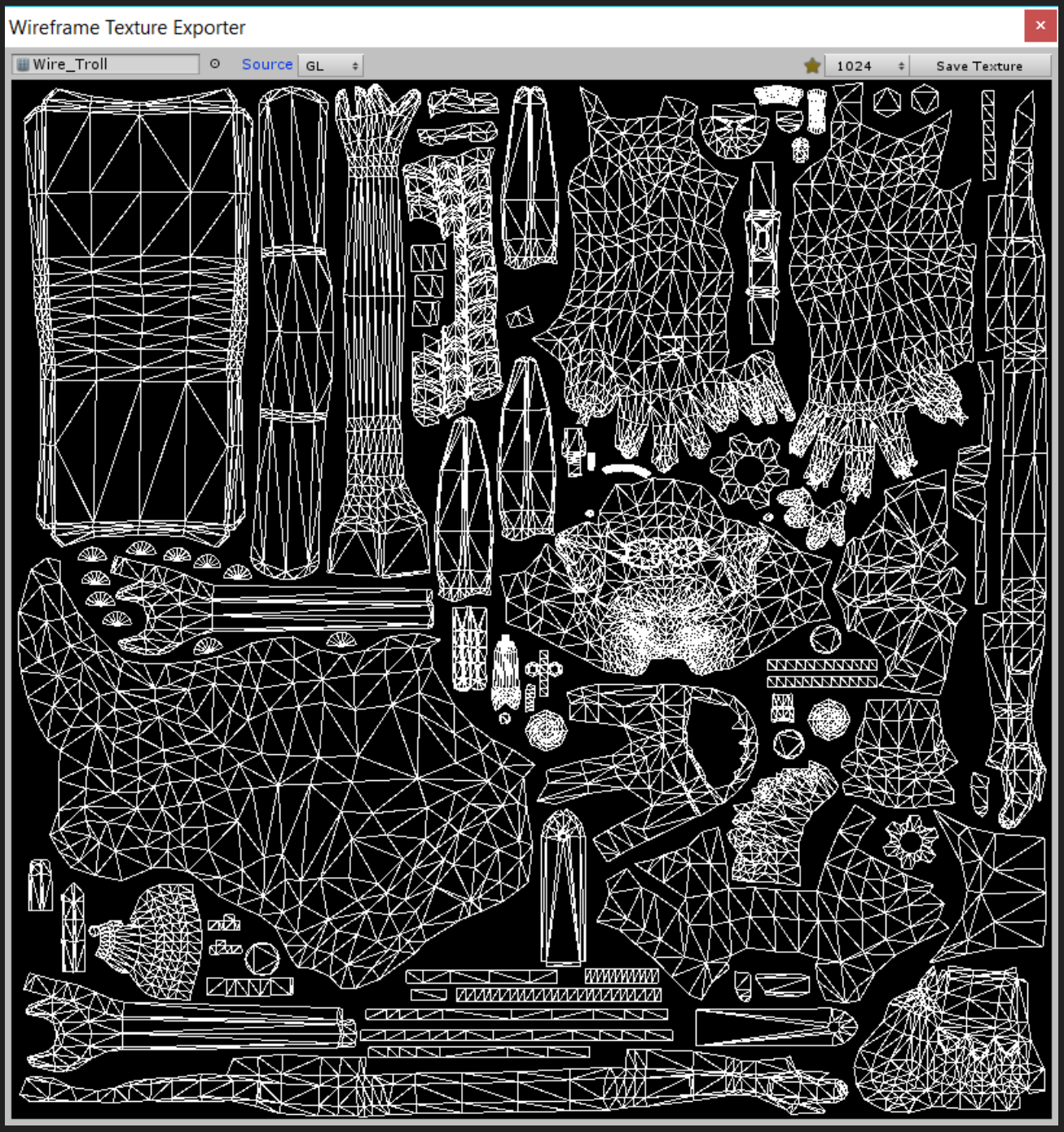
Quad converter uses three coefficients (Normal, Angle and Area), in most cases their default values will produce the best result.

Wireframe (texture) baking

Tool available from Menu/VacuumShaders/Wireframe Texture Exporter.

GL source uses Unitybuilt-in GL renderer for mesh wireframe renering, supports only trainagle shapes.

Using **Baked** source allows reading data from mesh baked barycentric coordinates and supports quad shaped wireframe baking.



Texture is exported in PNG format without black background and only with transparent wireframe data.

Run-time API

Two methods for Triangle wireframe generating:

- `Mesh WireframeGenerator.Generate(Mesh _origMesh)` – Generates optimized mesh (used by editor tools).
  Is guaranteed on mesh with less than 21,000 triangles. If mesh uses more triangles baking process may fail (due to increasing vertex count during baking) and will return ***null***.
- `Mesh WireframeGenerator.GenerateFast(Mesh _origMesh)` – Generates non-optimized mesh. Is the fastest method but mesh triangle count has limit – 21.000

For generating Quad wireframe:

- `Mesh WireframeGenerator.GenerateQuads(Mesh _origMesh)` – _origMesh triangle count must be less than 21,000.
  Quad shape rendering is not 100% guaranteed. It completely relies on original mesh triangle and vertex structure.
  Some parts of quad mesh still can be rendered as triangle.

All methods are available within `VacuumShaders.TheAmazingWireframeShader` namespace.

Check Example_5 (Runtime) and Example_8 (Runtime Quad Converter) example scenes.