

Wireframe Shader

Copyright © 2021 Amazing Assets

amazingassets.world

TABLE OF CONTENTS

1. Quick Start	3
2. Editor Window - Wireframe Mesh Generator	5
2.1 Settings	6
2.2 Generate	7
2.3 Save	8
3. Editor Window - Wireframe Texture Generator	9
3.1 Settings	10
3.2 Save	10
4. Dynamic Wireframe Rendering	11
5. Dynamic Mask Controller	12
6. Custom Shaders	13
6.1 Hand-written shaders	13
6.2 Unity Shader Graph	14
6.3 Amplify Shader Editor	19
7. Run-time API	21

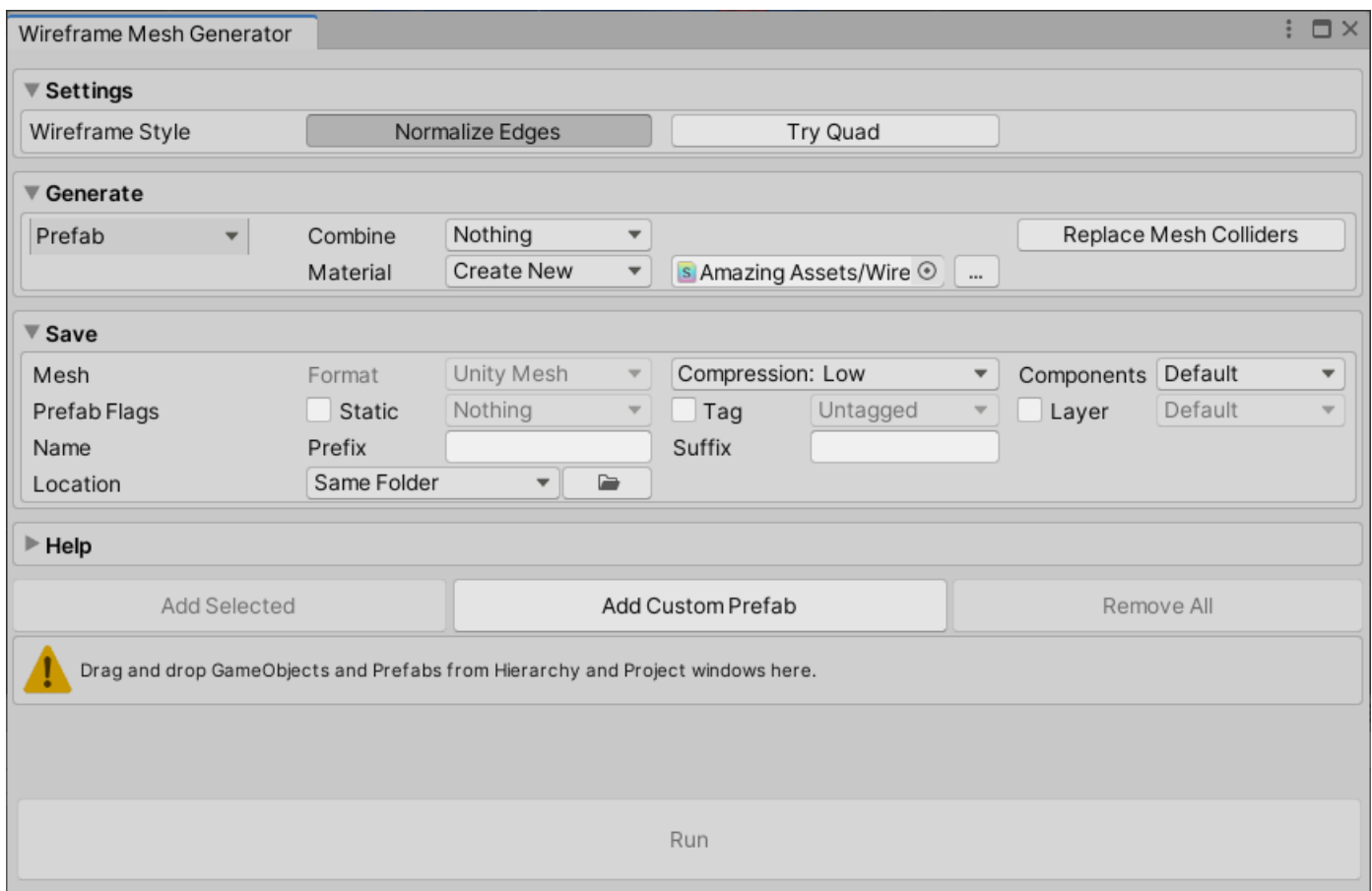
1. QUICK START

For wireframe rendering it is necessary to:

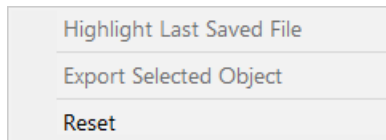
1. Bake wireframe data (barycentric coordinates) inside mesh UV4 buffer.
2. Use special shader to read that baked data from a mesh and based on it render wireframe.

Wireframe Shader package includes editor tool and run-time API for baking barycentric coordinates inside a mesh and collection of special shaders for wireframe rendering.

To bake wireframe data inside a mesh use **Wireframe Mesh Generator** editor tool from **Unity Main Menu/Window/Amazing Assets**.



Drag and drop game objects with [MeshFilter](#) or [SkinnedMeshRenderer](#) from the Hierarchy or prefab files from the Project windows here.



Use context menu to reset window settings.

After adding object to the conversion list, editor window displays technical information about used meshes.

Add Selected		Add Custom Prefab			Remove All		
Target		Mesh	Submesh	Vertices & Triangles	Wireframe	Index	
toonSportsCar		5	1	4,817 / 5,732	17,196 (256%)	16 Bits	-

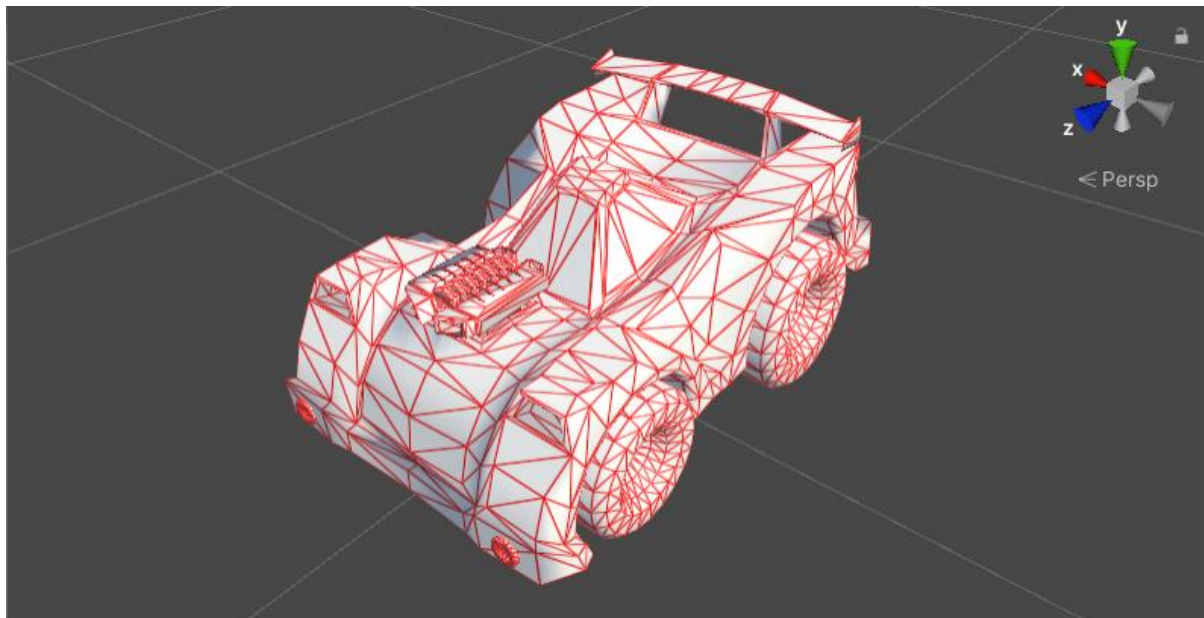
Wireframe data is generated for each individual triangle of a mesh and after baking it, mesh vertex count may increase.

Editor window displays **Vertices & Triangles** count of a source mesh and how much vertices generated **Wireframe** mesh will have. Triangles count never change, but vertex count always will be 3x more than triangles count, as triangle consists of three vertices and wireframe data baked per-vertex is unique for each one.

On the screenshot above, [toonSportsCar](#) object has 4,817 vertices and 5,732 triangles. After baking wireframe data, new mesh will have same amount of triangles and $3 \times 5,732 = 17,196$ vertices.

After adding objects to the conversion list, click on the **Run** button at the bottom of the window.

This will generate new prefab file with wireframe mesh and new material for wireframe rendering and instantiate this prefab in the scene, in the same location as the source object.



To see generated mesh, move it inside Scene view or hide source object

Use context menu to highlight saved files inside Project window and see all generated files.

2. EDITOR WINDOW - WIREFRAME MESH GENERATOR

Editor tool for generating wireframe meshes. Available from **Unity Main Menu/Window/Amazing Assets**.

Wireframe Mesh Generator

▼ Settings

Wireframe Style

Normalize Edges

Try Quad

▼ Generate

Prefab

Combine

Nothing

Replace Mesh Colliders

Material

Create New

Amazing Assets/Wiref

...

▼ Save

Mesh

Format

Unity Mesh

Compression: Low

Components

Default

Prefab Flags

☐ Static

Nothing

☐ Tag

Untagged

☐ Layer

Default

Name

Prefix

Suffix

Location

Same Folder

► Help

Add Selected

Add Custom Prefab

Remove All

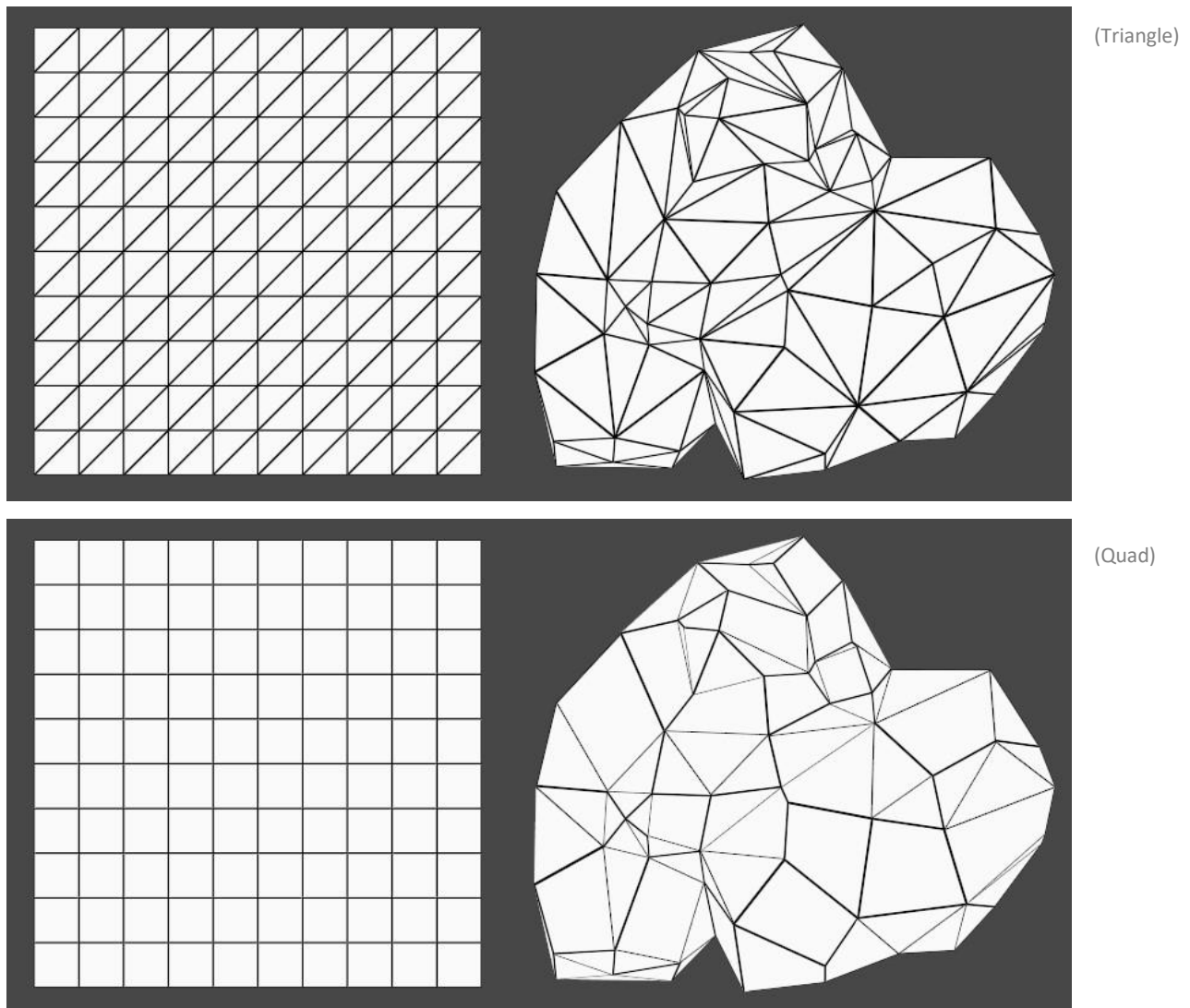
Target	Mesh	Submesh	Vertices & Triangles	Wireframe	Index	↺
toonSportsCar	5	1	4,817 / 5,732	17,196 (256%)	16 Bits	-
Boat	1	1	327 / 167	501 (53%)	16 Bits	-
Fox	1	1	2,266 / 3,450	10,350 (356%)	16 Bits	-
Leves	1	1	1,724 / 862	2,586 (50%)	16 Bits	-
Medieval_house_lite	57	1 - 2	13,930 / 9,496	28,488 (104%)	16 Bits	-
Medieval_house_lite	57	1 - 2	13,930 / 9,496	28,488 (104%)	16 Bits	-
Rock	1	1	736 / 642	1,926 (161%)	16 Bits	-
Trunk	1	1	1,429 / 1,682	5,046 (253%)	16 Bits	-
Candle	1	1	665 / 716	2,148 (223%)	16 Bits	-
Candles	1	1	2,313 / 3,204	9,612 (315%)	16 Bits	-
Plane	1	1	4 / 2	6 (50%)	16 Bits	-

Run (25 Objects / 145 Meshes)

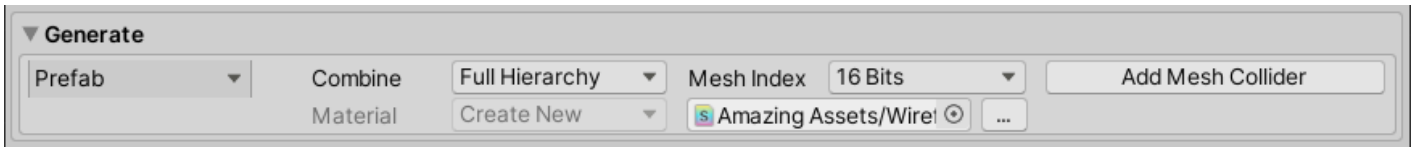
2.1 SETTINGS



- **Normalize Edges** - Wireframe baking algorithm will try to make width off all edges of the wireframe triangle equal. This effect can be noticeable when using **Thickness** values inside wireframe materials.
- **Try Quad** – Wireframe rendering in quad style is pure approximation and its quality depends on the mesh vertex & triangle layout. Images below demonstrate Triangle and Quad wireframe style rendering for Unity's built-in Plane (has good vertex & triangle layout) and custom meshes.

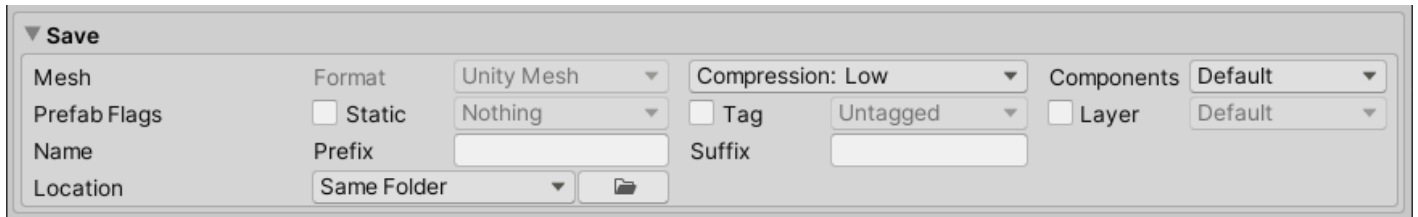


2.2 GENERATE



- **Prefab/Mesh Only** – Generates full prefab file that is an exact copy of the source object but with newly generated wireframe mesh and material. Or just wireframe mesh asset file.
- **Combine** – Mesh combine options:
 - **Nothing** – Generated wireframe mesh has the same amount submeshes as the source mesh.
 - **Submeshes** – Generated mesh has one combined submesh.
 - **Full Hierarchy** – Combines all components (all child game objects) inside source game object and generates one mesh with one submesh. Does not work with skinned meshes.
- **Mesh Index** – Desired [index format](#) for fully combined mesh.
- **Add/Replace Mesh Collider** – Adds or replaces MeshCollider's mesh inside generated prefab object, to be the same wireframe mesh as in the used MeshFilter.
- **Material** – Material options for generated prefab:
 - **Use Original** – New material asset is not created and generated prefab uses material from the source object.
 - **Create New** – Creates new material asset using selected shader.
 - **Create Duplicate** - Creates new wireframe material asset using selected shader and copies properties from the source object material there.


2.3 SAVE



▼ Save

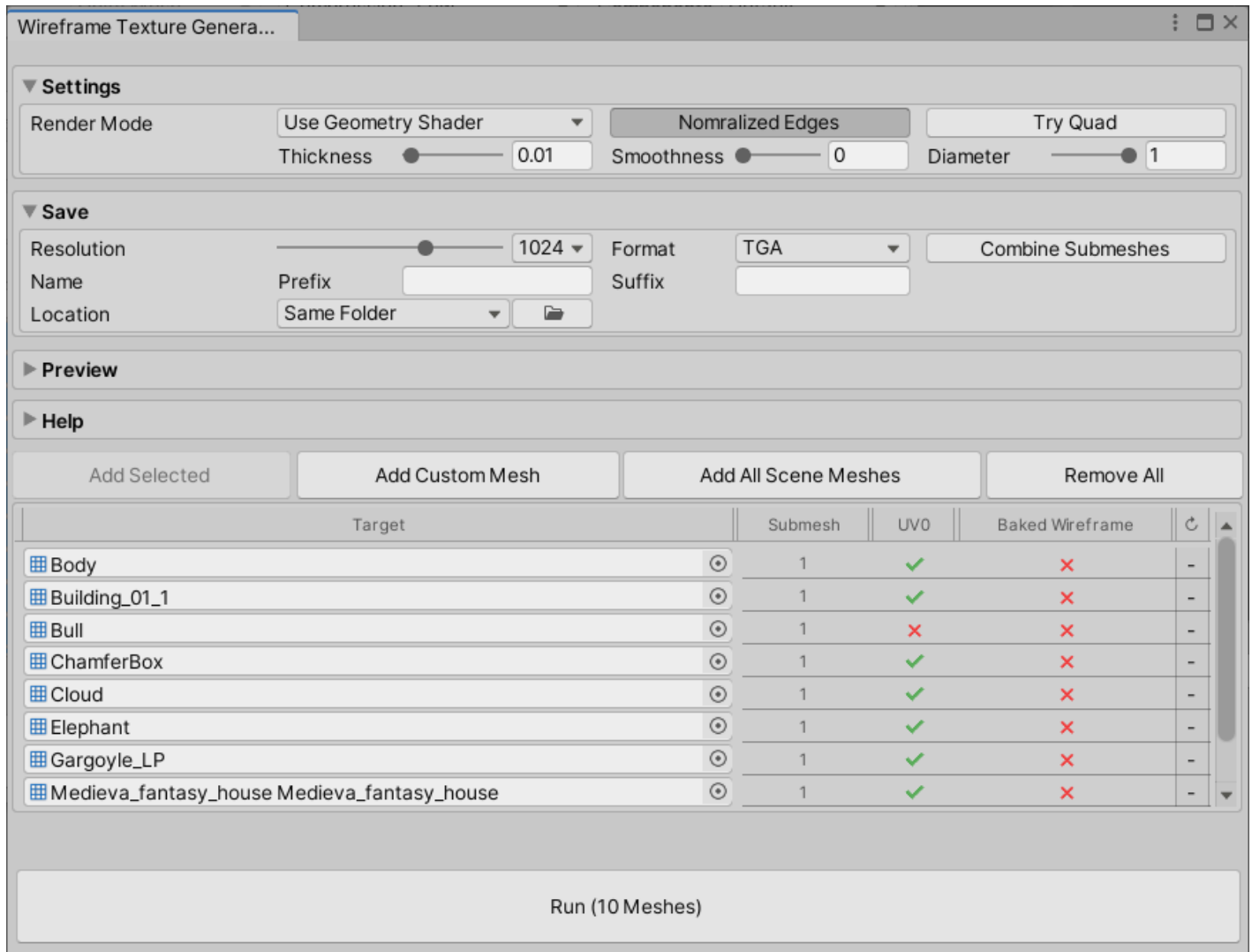
Mesh	Format	Unity Mesh	Compression: Low	Components	Default	
Prefab Flags	<input type="checkbox"/> Static	Nothing	<input type="checkbox"/> Tag	Untagged	<input type="checkbox"/> Layer	Default
Name	Prefix		Suffix			
Location	Same Folder					

- **Mesh Format** - Generated mesh file is saved in Unity **.asset** format.
- **Compression** - Allows reducing generated file size by lowering numerical accuracy of a mesh. Instead of 32-bit floats, lower size fixed number will be used to represent mesh data.

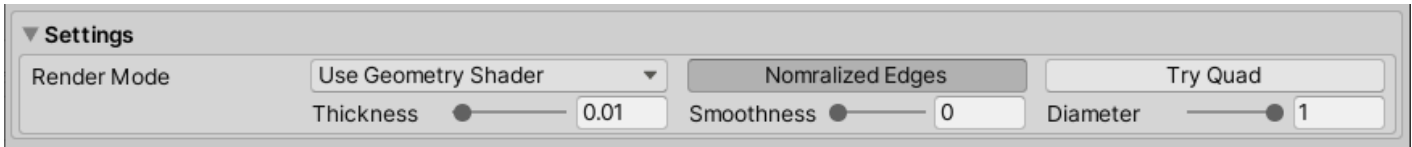
Note, more compression introduces more artifacts in vertex data (*position, normal, uv, etc.*).
- **Components** – By default generated mesh has all components that are available in the source mesh. Using this option unwanted components can be excluded from the generated mesh and file size will be reduced.
- **Prefab Flags** – Overrides generated prefab's flags.
- **Name** – Adds prefix & suffix to the all generated files names.
- **Location** – Generated files save location. Can be any folder on the hard drive, inside or outside of a project.
-  **Aa** - Allows using parent object's name as save subfolder or includes it in the name of the generated files.

3. EDITOR WINDOW - WIREFRAME TEXTURE GENERATOR

Editor tool for generating wireframe textures. Available from **Unity Main Menu/Window/Amazing Assets**



3.1 SETTINGS



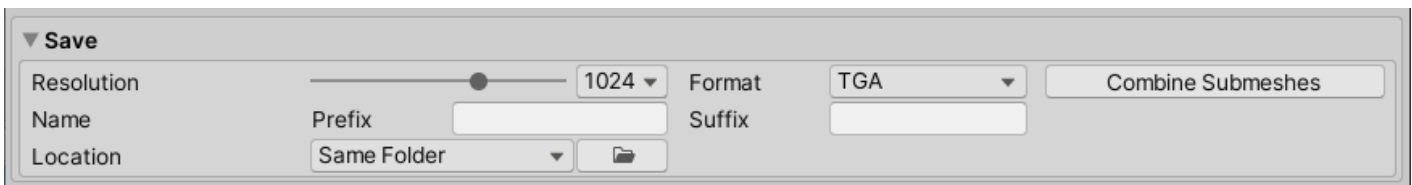
➤ **Render Mode** - Wireframe texture render modes:

- **Read Wireframe From Mesh** – Wireframe texture is rendered using wireframe data baked inside a mesh.
- **Use Geometry Shader** – Wireframe texture is calculated and rendered using a shader, no data baking is required.

➤ **Thickness/Smoothness/Diameter** – Wireframe rendering visual options.

Note, For generating wireframe texture, mesh needs to have proper UV0. Overlapping or inaccurate UV0 can render wireframe incorrectly.

3.2 SAVE



➤ **Resolution** - Generated texture save resolution. Can be in the range of [16, 8192].

➤ **Format** – Texture save format: JPG, PNG or TGA.

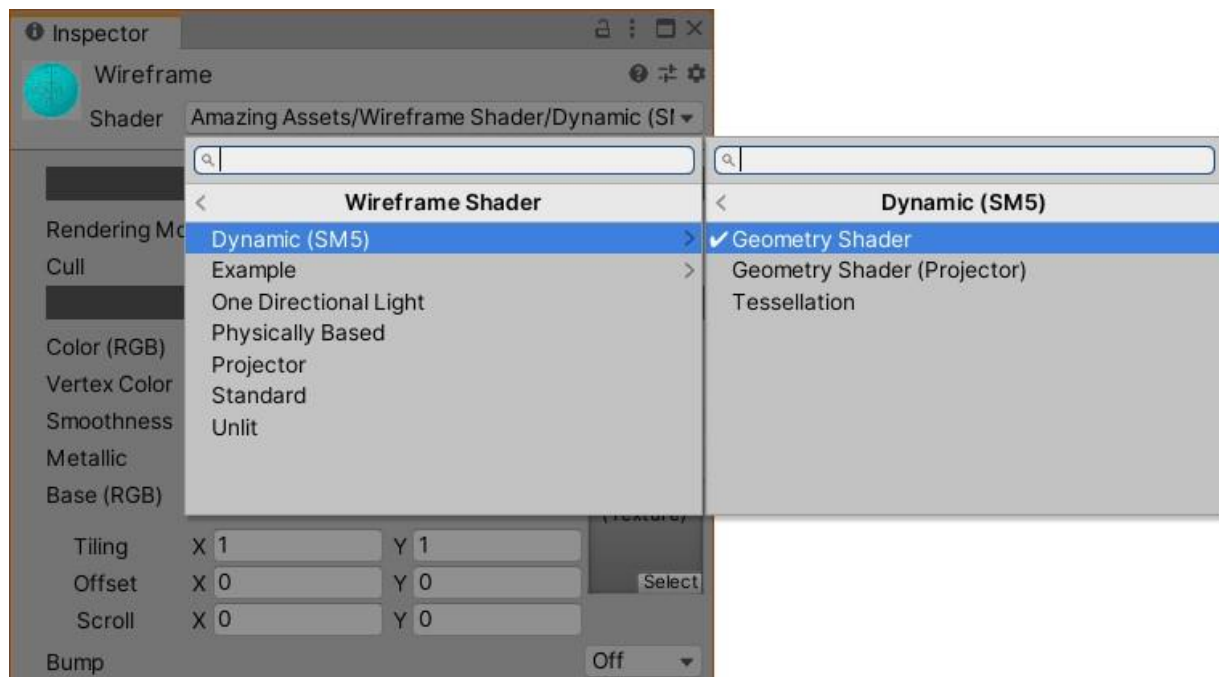
➤ **Combine Submeshes** – For each submesh of a mesh is generated separate wireframe texture file. By enabling this option, all submeshes will be combined to render one wireframe texture file.

Note, if submeshes have overlapping UVs, wireframe texture also will have overlapping zones.

4. DYNAMIC WIREFRAME RENDERING

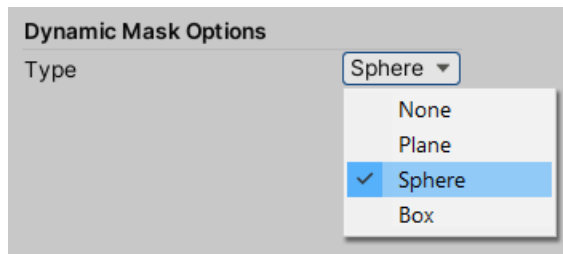
Package contains collection of the Dynamic Wireframe shaders that do not need wireframe data to be baked inside a mesh, instead everything is calculated by shaders themselves. Those shaders require device with Shader Model 5.0 and GeometryShaders support.

Currently dynamic wireframe rendering is supported only by **Built-in** render pipeline.



5. DYNAMIC MASK CONTROLLER

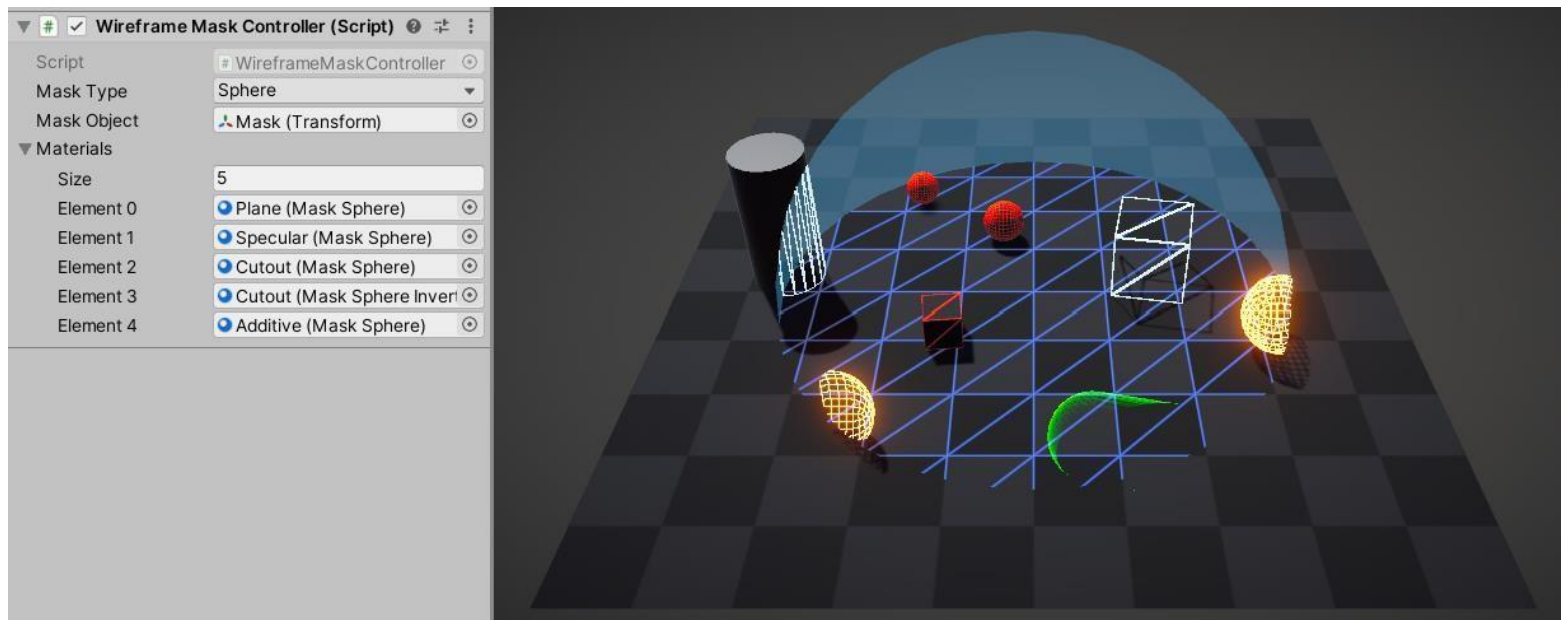
Package included shaders can use Plane, Sphere and Box objects for masking wireframe effect. It can be turned on/off from the material editor.



For updating material with mask object's transformation data (position, rotation, scale), it is necessary to use **WireframeMaskController** script.

One instance of this script can update only one type of a mask for multiple materials. Scene can contain any number of the **WireframeMaskController** scripts.

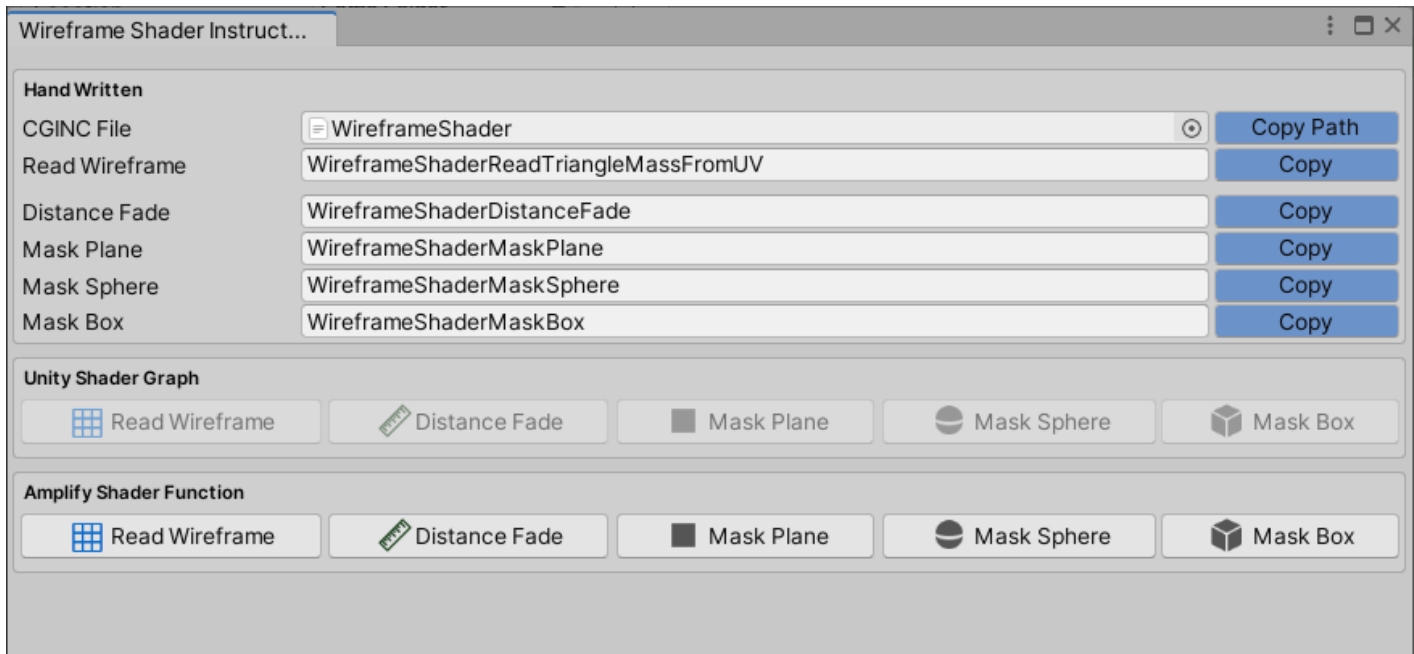
Package includes example scenes demonstrating all 3 types of the dynamic masking in action.



Package included shaders can use only one type of a mask, but for custom shaders it is possible to mix them all together. Custom wireframe shaders are explained in the chapter below.

6. CUSTOM SHADERS

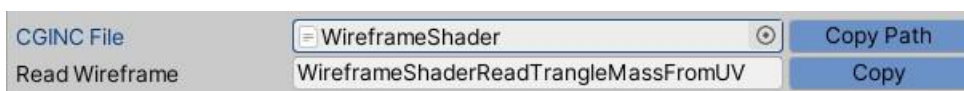
Wireframe data baked inside a mesh can be read by handwritten shader, Unity Shader Graph and Amplify Shader Editor. All prepared methods, nodes and file paths can be ready from the **Unity Main Menu/Window/Amazing Assets/Wireframe Shader Instructions** window.



6.1 HAND-WRITTEN SHADERS

Steps requiring for reading wireframe data inside a hand written shader:

- Inside shader define path to the **WireframeShader.cginc** file.
This can be easily done by clicking on the **Copy Path** button inside **Wireframe Shader Instructions** window. Keyboard memory now will contain path to the **cginc** file, just paste it into a shader.




- Inside shader send vertex TEXCOORD3 (containing wireframe data) from vertex to the pixel shader stage.
- Use **WireframeShaderReadTriangleMassFromUV** method inside pixel stage to read wireframe from the TEXCOORD3 buffer.

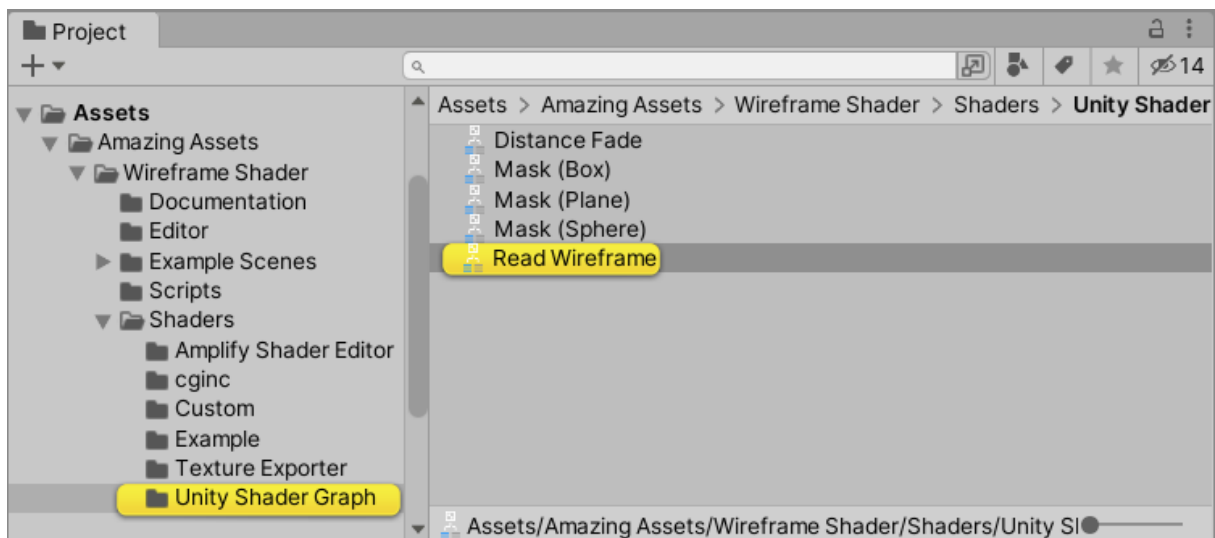
Package includes example shaders (**Amazing Assets\Wireframe Shader\Shaders\Example** folder) for demonstrating steps described above.

Additionally shader API allows to use 4 type of the mask methods for hiding parts of a wireframe: Distance Fade, Plane, Sphere and Box masks. Each methods returns grayscale value in the range of [0, 1] and can be simple multiplied with the return result of the [WireframeShaderReadTriangleMassFromUV](#) method. Clicking on the **Copy** button will copy method name and variables it receives to the keyboard memory.

Note, Plane, Sphere and Box mask methods use object transformation info that needs to be updated using script. Check mask example scenes and [WireframeMaskController](#) script there updating material properties.

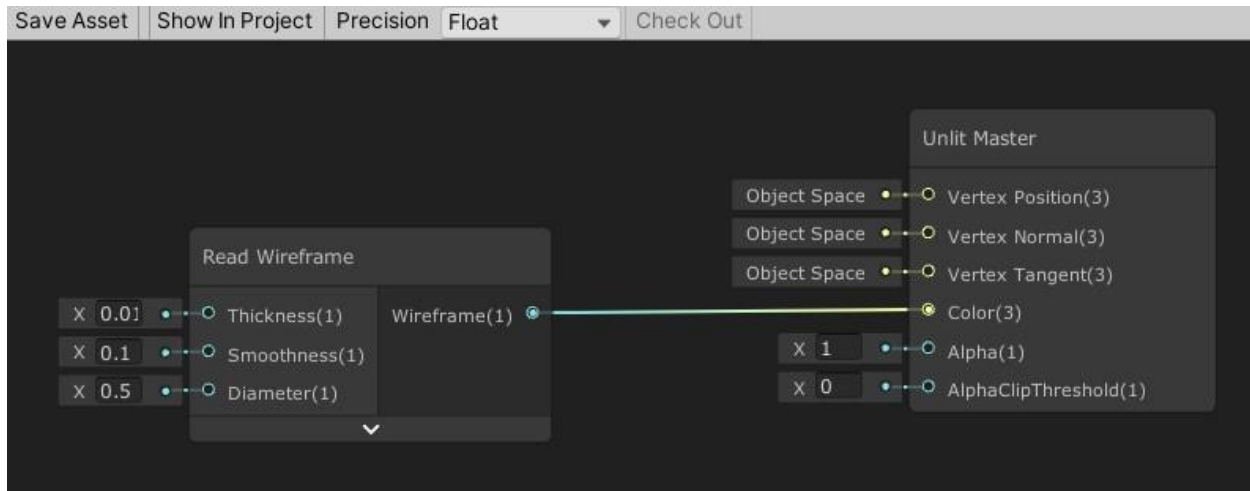
6.2 UNITY SHADER GRAPH

Inside **Wireframe Shader Instructions** window click on the  **Read Wireframe** button. This will highlight Shader Graph node inside Project window.

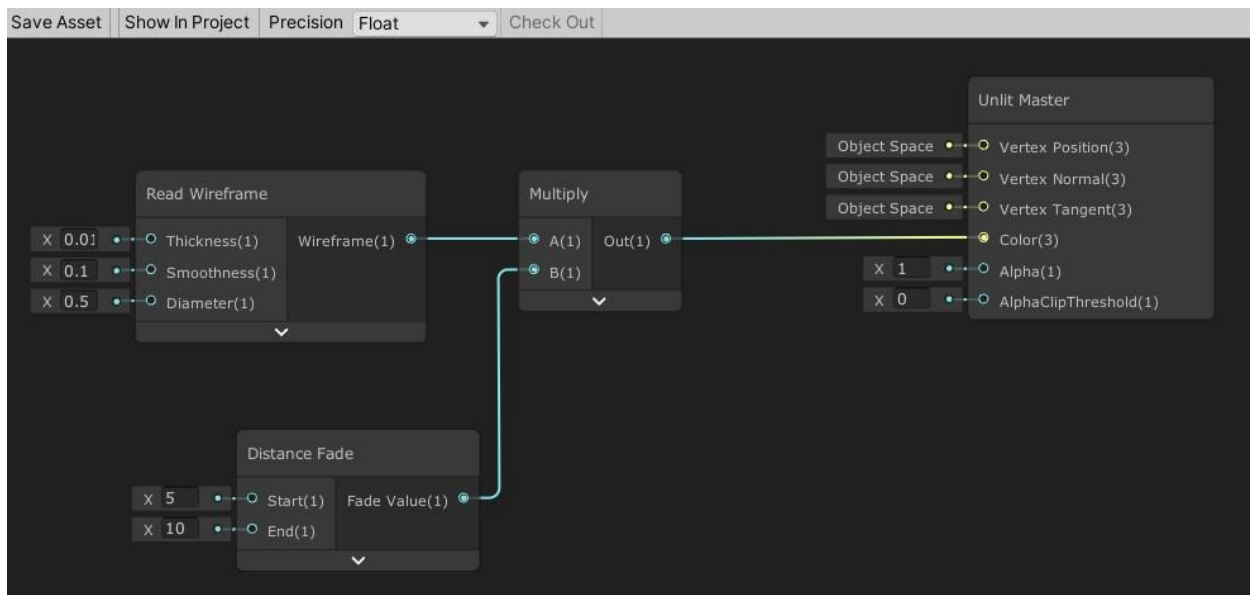


Drag & and drop node file inside shader graph.

Node reads wireframe data baked inside mesh UV4 buffer. Output result is in the range of [0, 1].

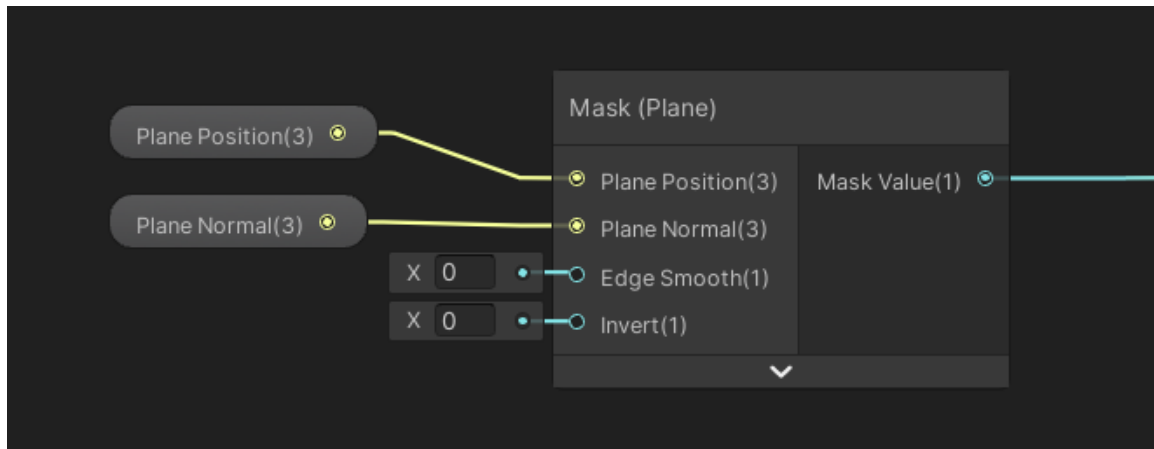


Package includes additional nodes for creating distance fade and dynamic mask effects. Output of those nodes are in the range of [0, 1] and can be multiplied with **Read Wireframe** (or any other) node.

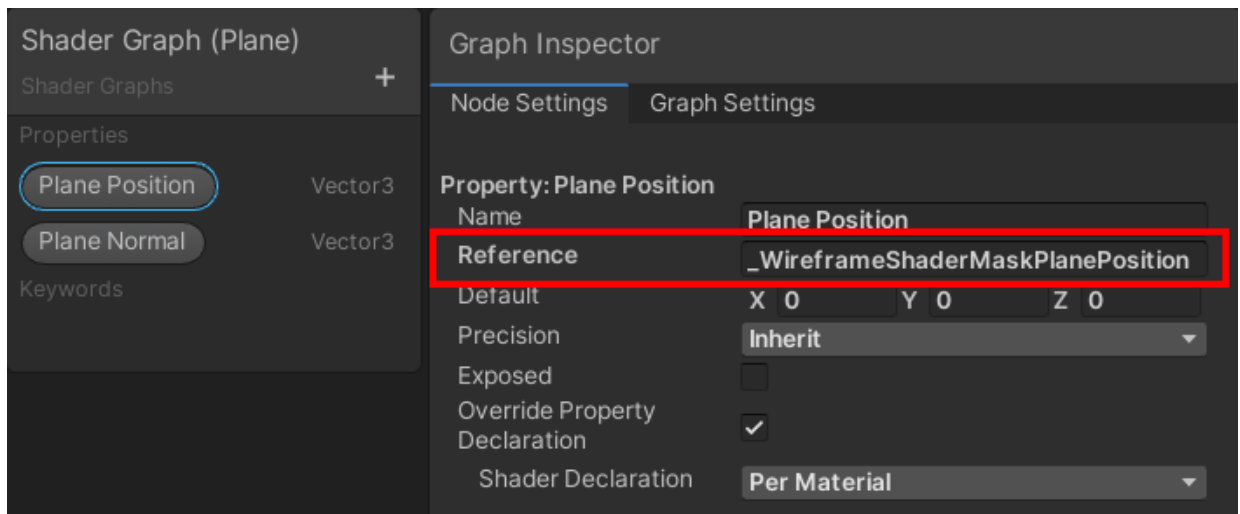



Note, for proper using Plane, Sphere and Box masks, shader needs declaration of the several properties and updating them using script:

- Plane mask node:



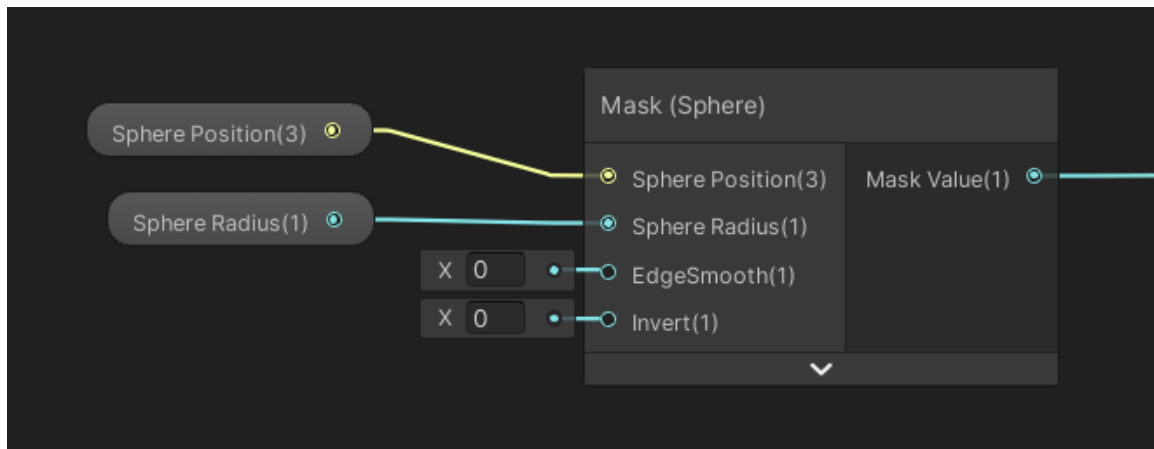
Plane Position property of the **Vector3** type and **Plane Normal** property of the **Vector3** type are mandatory. Those properties need to be updated using script. To update those properties using [WireframeMaskController](#) script, use **_WireframeShaderMaskPlanePosition** for **Mask Position** and **_WireframeShaderMaskPlaneNormal** for **Mask Normal** properties reference names.



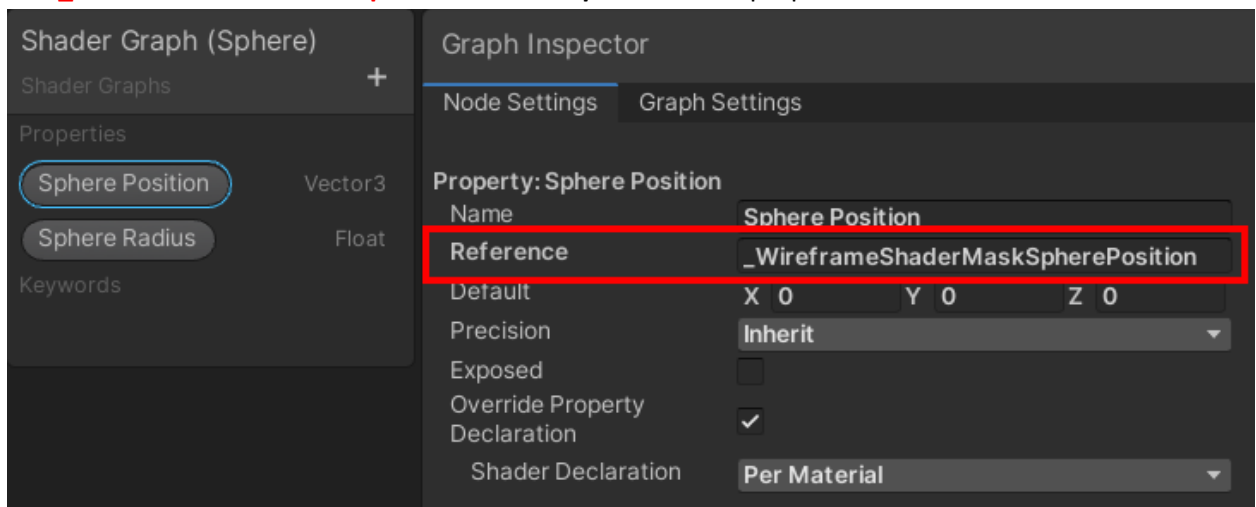
Note, when clicking on the  **Mask Plane** button inside **Wireframe Shader Instructions** window, Unity's Console window will print Plane mask's properties reference names.


To hide **Plane Position** and **Plane Normal** properties inside material editor, uncheck **Exposed** property inside **Node Settings** and make sure **Shader Declaration** field is set to **Per Material**.

- Sphere Mask Node:



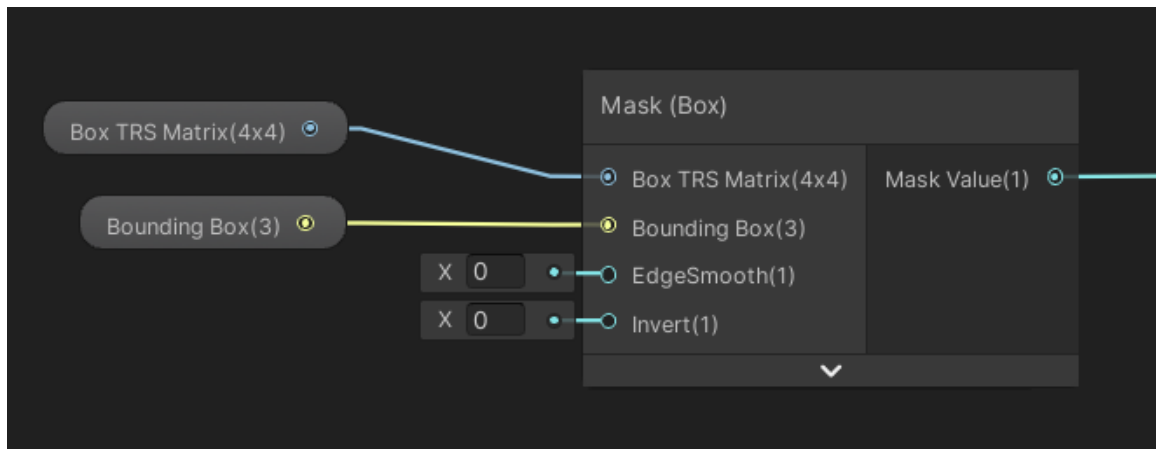
Sphere Position property of the **Vector3** type and **Sphere Radius** property of the **float** type are mandatory. Those properties need to be updated using script. To update those properties using [WireframeMaskController](#) script, use **_WireframeShaderMaskSpherePosition** for **Sphere Position** and **_WireframeShaderMaskSphereRadius** for **Sphere Radius** properties reference names.



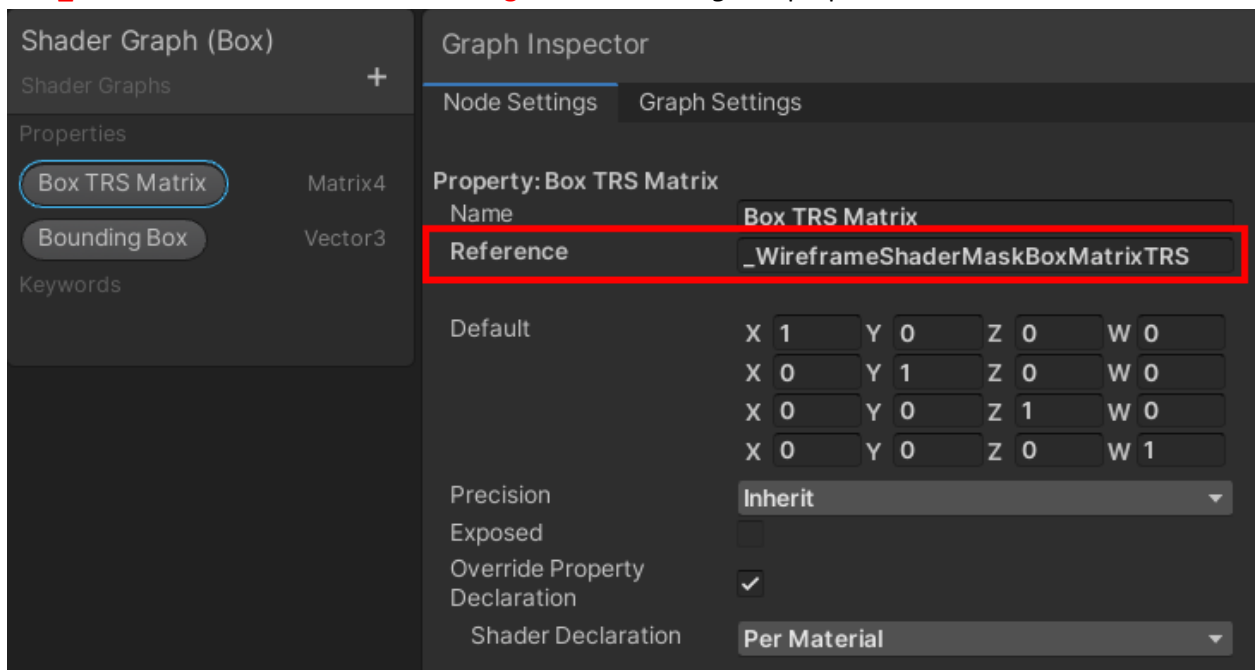
Note, when clicking on the  **Mask Sphere** button inside **Wireframe Shader Instructions** window, Unity's Console window will print Sphere mask's properties reference names.


To hide **Sphere Position** and **Sphere Radius** properties inside material editor, uncheck **Exposed** property inside **Node Settings** and make sure **Shader Declaration** field is set to **Per Material**.

- Box Mask Node:




Box TRS Matrix property of the **Matrix4** type and **Bounding Box** property of the **Vector3** type are mandatory. Those properties need to be updated using script. To update those properties using [WireframeMaskController](#) script, use **_WireframeShaderMaskBoxMatrixTRS** for **Box TRS Matrix** and **_WireframeShaderMaskBoxBoundingBox** for **Bounding Box** properties reference names.

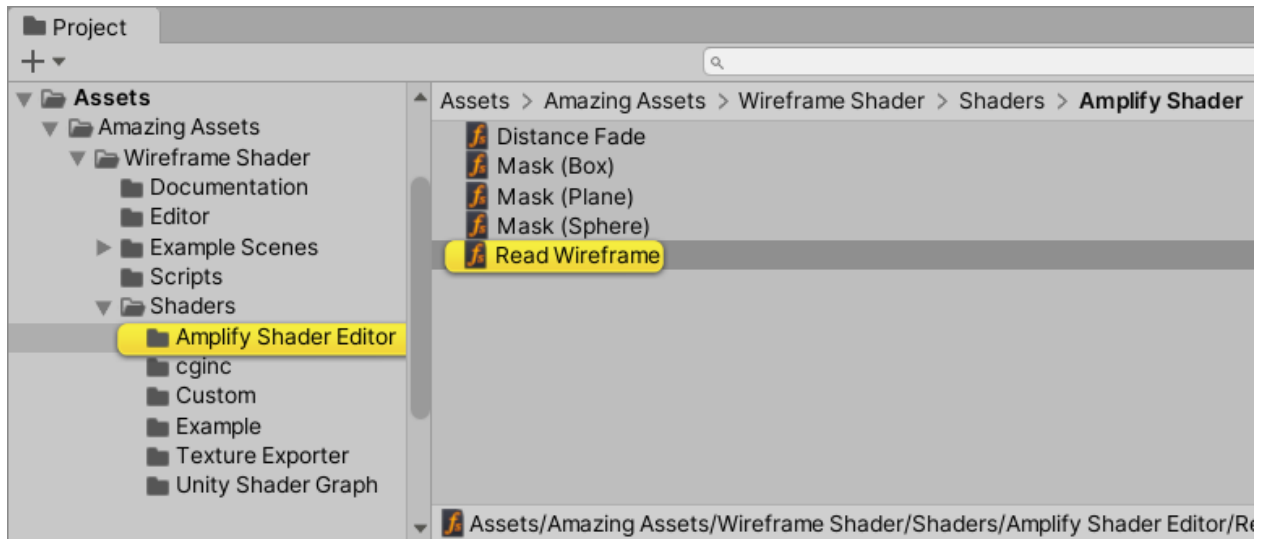


Note, when clicking on the  **Mask Box** button inside **Wireframe Shader Instructions** window, Unity's Console window will print Box masks's properties reference names.

To hide **Box TRS Matrix** and **Bounding Box** properties inside material editor, uncheck **Exposed** property inside **Node Settings** and make sure **Shader Declaration** field is set to **Per Material**.

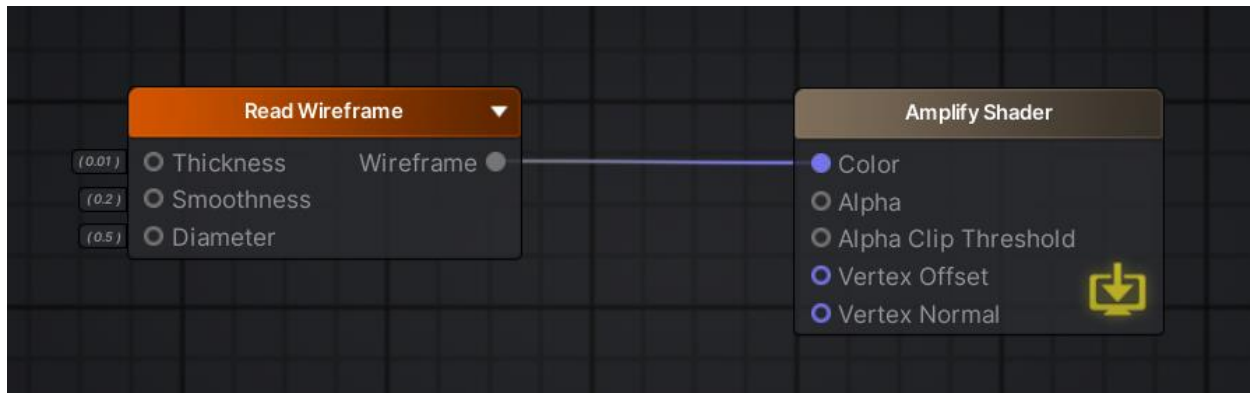
6.3 AMPLIFY SHADER EDITOR

Inside **Wireframe Shader Instructions** window click on the  **Read Wireframe** button. This will highlight Amplify Shader Editor node inside Project window.

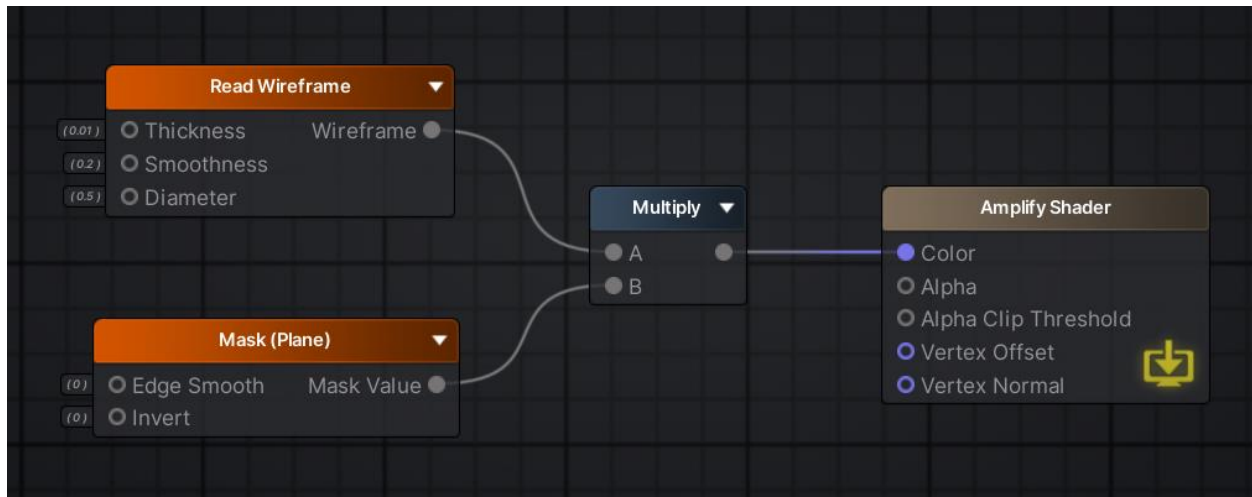


Drag & and drop node file inside shader editor.

Node reads wireframe data baked inside mesh UV4 buffer. Output result is in the range of [0, 1].



Package includes additional nodes for creating distance fade and dynamic mask effects. Output of those nodes are in the range of [0, 1] and can be multiplied with **Read Wireframe** (or any other) node.



Note, when using Plane, Sphere or Box mask nodes inside shader, use [WireframeMaskController](#) script to update those masks transformation data inside material.

7. RUN-TIME API

Run-time API for calculating wireframe data for a mesh or generating wireframe texture can be brought into scope with this using directive:

```
C#
using AmazingAssets.WireframeShader;
```

Unity [Mesh](#) class now will have 2 additional extension methods:

Mesh GenerateWireframeMesh(bool normalizeEdges, bool tryQuad)

Generates new mesh with wireframe data baked inside uv4 buffer (note, inside shader uv4 coordinate of a mesh is read using TEXCOORD3 semantic).

Resultant mesh is in 16 bit index buffer format if it has less than 65,535 vertices (21,845 triangles). Otherwise mesh uses 32 bit index buffer format.

Texture2D GenerateWireframeTexture(bool useGeometryShader, int submeshIndex, bool normalizeEdges, bool tryQuad, float thickness, float smoothness, float diameter, int resolution)

Generates wireframe texture.

bool useGeometryShader – If enabled, wireframe texture is generated using GeometryShaders and source mesh does not need wireframe data to be baked inside it. For run-time use and build, project must include **Amazing Assets/Wireframe Shader/Shaders/Texture Exporter/Texture Exporter.shader** file. If this option is not used then wireframe texture is calculated from data baked inside a mesh.

int submeshIndex - Index of a submesh for which is rendered wireframe texture. Use value of -1 to export submesh combined one texture.

bool normalizeEdges – Wireframe triangle edges will be approximately of the same width.

bool tryQuad – Renders wireframe in **quad** shape instead of a **triangle**. Result highly depends on a mesh vertex & triangle layout.

float thickness, smoothness, diameter – Visual characteristic of the rendered wireframe.

int resolution - Texture resolution. Must be power of 2, in the range of 16 – 8192.