POLYGRID

DYNAMIC WIREFRAME EFFECTS

v1.0.1

Created by Ripcord Development
info@ripcorddev.com

This project has been thoroughly tested for bugs before being sent to the Unity Asset store. If you do find an issue with this package, please contact us before posting a negative review on the Unity Asset Store. We are more than willing to help solve any issues you may encounter.

PolyGrid allows you to dynamically generate a complex looking effect with the click of a button. The simple set up combined with lots of configuration options will make it easy to integrate the **PolyGrid** effect into your project.

PREFABS

This package only contains a couple prefabs. The first is a prefab of a fully configured **PolyGrid** object, the same one that appears in the demo scene. The other is a prefab for a **PolyGrid** vertex object. This object is simply a default Unity Quad with a custom shader applied to it.

SHADERS

This package contains two custom shaders. While these shaders aren't required to use **PolyGrid** they certainly help define the look of this package.

- **Billboard** – This shader forces the attached object to always face the camera. It ignores the rotation of the object but respects the object's scale. It supports a texture with an alpha channel and allows for colour tinting. This shader is best used on the vertex objects.
- **UnlitAlphaTint** – An unlit shader that supports a texture with an alpha channel and allows for colour tinting. This shader is best used on the edge objects.

MATERIALS

PolyGrid contains two sets of materials, one set for vertex objects, and another for edge objects. Try using different combinations to create the effect that's right for your project or use these as a guide for creating your own.

SCRIPTS

PolyGrid is setup and controlled through a single script. There are support scripts that are used as well, but **PolyGrid** takes care of adding and configuring those. Below is an overview of each script and what it does.

- **PolyGrid** – This script creates a dynamic grid of vertex and edge objects, as well as sets the visual style and motion for the grid. The grid is built in the editor, rather than at runtime.
    - **AnimateGrid()** – This function starts moving the vertices in the gird.
    - **ResetGrid()** – This function returns all vertices in the grid to their original position's. Once returned, the vertices stop moving.

Grid Construction

- **Vertex Prefab** – The gameObject that will be used for all vertex objects.
- **Vertex Material** – The material that will be applied to all vertex objects.
- **Edge Material** – The material that will be applied to all edge objects.

Grid Dimensions

- **Rows** – The number of vertex rows the polyGrid object will have.
- **Columns** – The number of vertex columns the polyGrid object will have.
- **Spacing** – The distance from one vertex to the next along any axis.

Grid Appearance

- **Show Vertices** – If true, the vertex objects of the polyGrid will be visible.
- **Show Edge Rows** – If true, the horizontal edges of the polyGrid will be visible.
- **Show Edge Columns** – If true, the vertical edges of the polyGrid will be visible.
- **Vertex Size** – The scale of all vertex objects.
- **Edge Thickness** – The width of all edge objects.
- **Vertex Colour** – The colour that will be applied to all vertex objects.
- **Edge Colour** – The colour that will be applied to all edge objects.

Grid Movement

- **Vertex Range** – The maximum distance a vertex can move along any axis, positive or negative, away from its original position.
- **Min Time** – The minimum number of seconds required for a vertex to move from one position to another.
- **Max Time** – The maximum number of seconds required for a vertex to move from one position to another.
- **Speed Multiplier** – A multiplier that modifies the time a vertex requires to move from one position to another.
- **Auto Start** – If true, the PolyGrid will start moving on scene start, otherwise it will need to be triggered manually.

Note:

- *The settings in the PolyGrid component are restricted to maximum and minimum values. If you need to go beyond any of these values, you'll need to set them manually within the PolyGridEditor script.*

- **PolyVertex** – This script controls the movement of each individual PolyGrid vertex. *This script is automatically assigned and configured.*

This script requires the free iTween package to function properly, see below

- **PolyEdge** – This script links the points of the edge object's LineRenderer to the individual vertices in the PolyGrid. *This script is automatically assigned and configured.*

Editor Scripts – These scripts control the appearance of certain scripts within the inspector. Editor scripts must remain within a folder called **Editor**.

- **PolyGridEditor** – This script controls the appearance of any instance of **PolyGrid** within the editor. It allows a **PolyGrid** to be built directly in the editor as well as restricts values to specified ranges.

3rd Party Scripts

- **iTween**

The free iTween package must be installed for this package to function properly

This package makes use of the freely available iTween package for Unity, created by Bob Berkebile (pixelplacement). To learn more about iTween or to help support its development, you can check out the documentation here:
http://www.pixelplacement.com/itween/index.php

iTween can be downloaded from the Unity Asset Store here:
https://assetstore.unity.com/packages/tools/animation/itween-84

To install iTween simply import its asset package. Nothing further needs to be done.

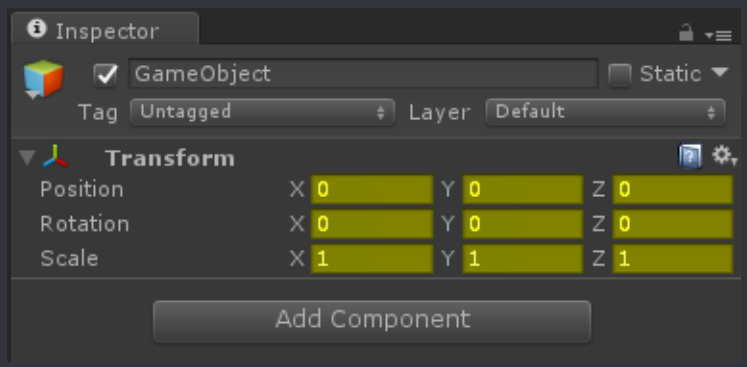
GETTING STARTED

Quick Steps:

1. Create empty GameObject
2. Reset transform of GameObject
3. Add PolyGrid component
4. Set Vertex Prefab, Vertex Material, and Edge Material values
5. Press Build Grid

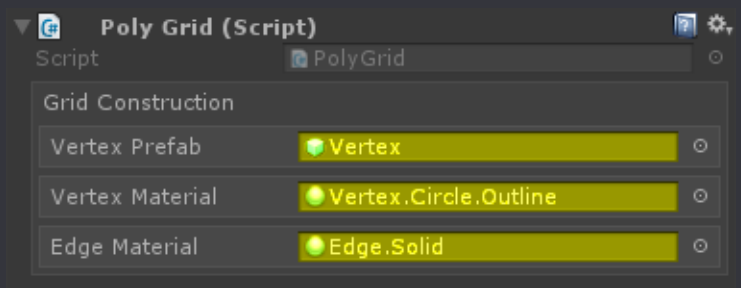
Detailed Steps:

1. To get started with PolyGrid create an empty game object (Menu > GameObject > Create Empty or press CTRL+SHIFT+N).
2. With the new empty GameObject selected reset its transform values so that position and rotation are 0, and scale is 1.



You can quickly reset the transform (or any component) by right-clicking the component title and selecting Reset.

3. Add the PolyGrid component to the empty GameObject. You can do this by either selecting and dragging the PolyGrid script onto the empty GameObject, or with the empty GameObject selected you can press the Add Component button and navigate to RipcordDev > PolyGrid > PolyGrid.
4. All the values of the PolyGrid script have preset values except for those in the Grid Construction section. These values will need to be set manually. You can assign whichever objects and materials you want for these items but to match the look of the demo scene you will need to assign the following:



5. When you are finished adjusting the other settings to customize the look of your PolyGrid, press the Build Grid button at the bottom of the component to generate the grid in the scene.

Notes:

- *You will need to press Build Grid each time you make changes to the settings in PolyGrid. This will rebuild the grid using the latest settings.*
- *The grid is initially built at a zeroed-out transform. Once it has been built though, you can move, rotate, and scale the PolyGrid object however you like.*

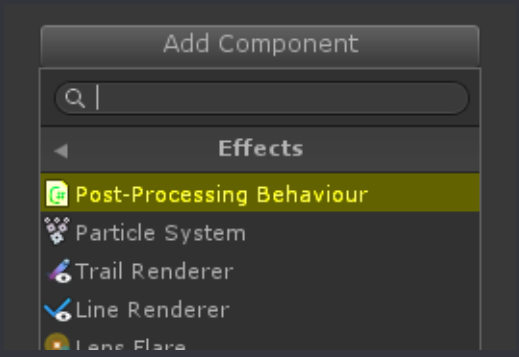
SCENE SETUP

The following steps are only required if you want to use the full PolyGrid demo as seen in screenshots and the WebGL build. If you want to just get right to using PolyGrid for your own project you can skip this section.

The demo scenes in this package make use of a free asset package provided by Unity. To replicate the exact look and functionality of the demo scenes, you'll need to import this package.

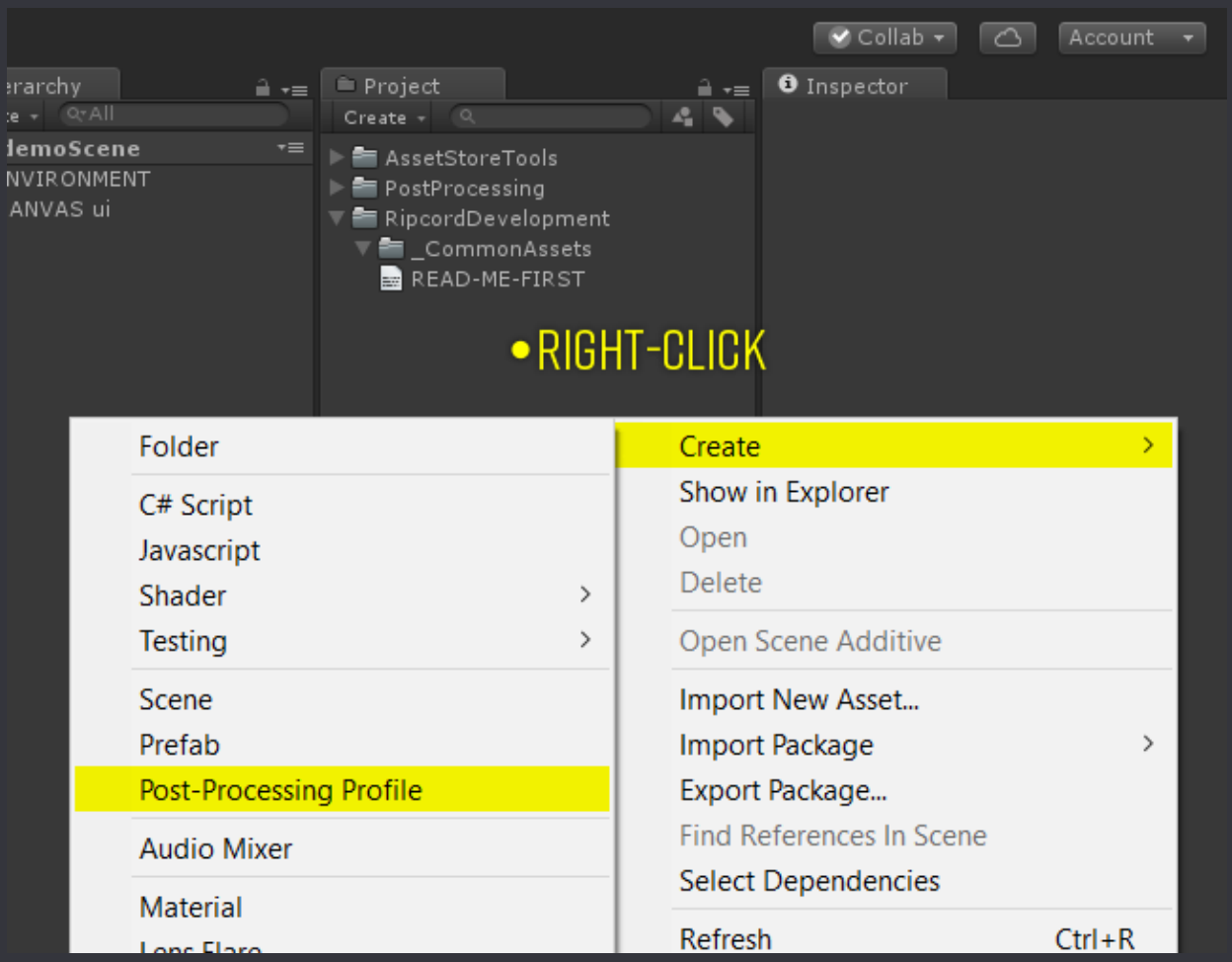
Download this package from the Unity Asset Store and import it into your project - Post Processing Stack

1. Add the Post-Processing Behaviour component by selecting your main camera, pressing the Add Component button. Select Effects > Post-Processing Behaviour to add the component.

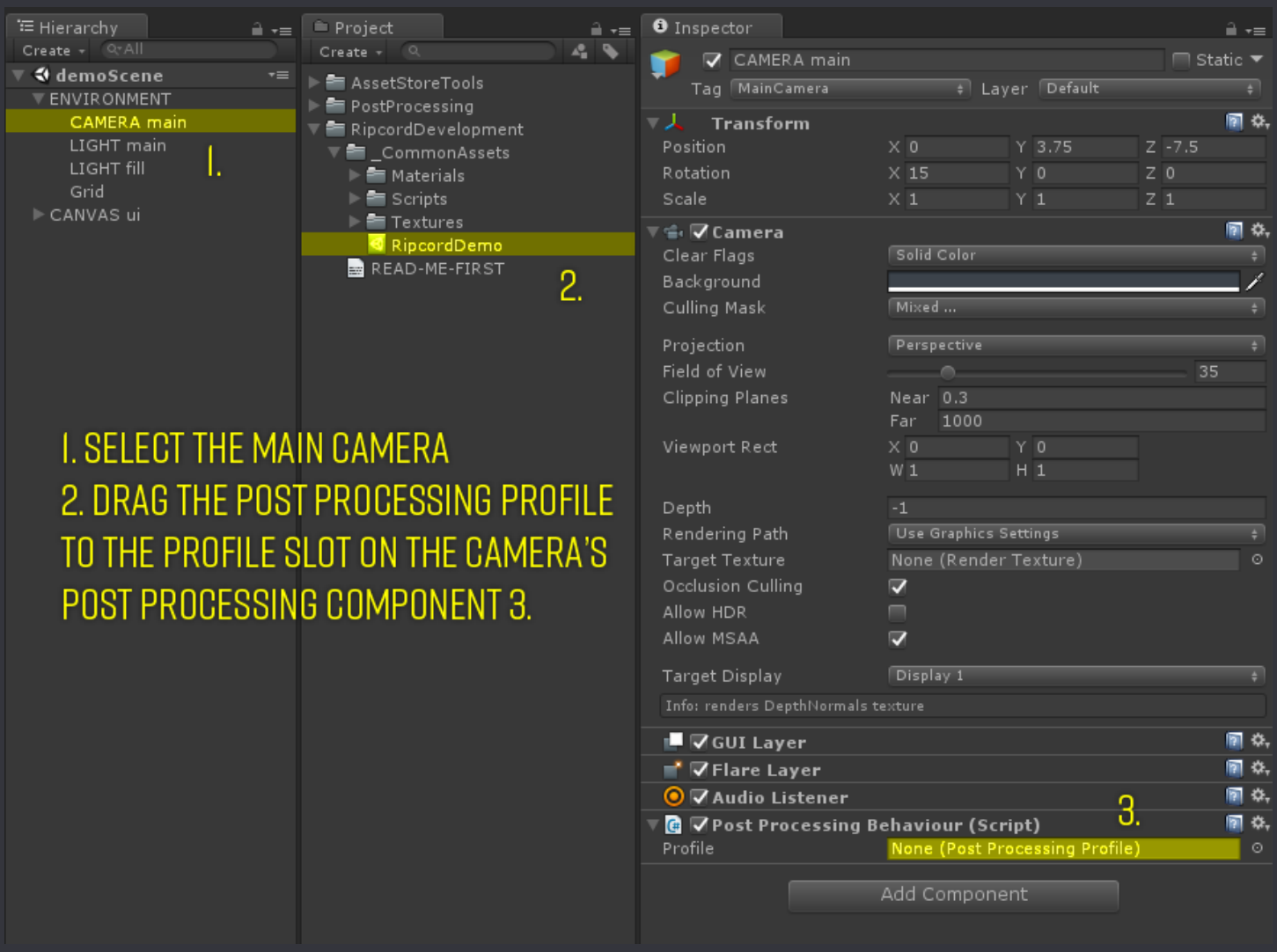


   With the Post-Processing Behaviour added to your camera, you can either use the included RipcordDemo Post Processing Profile or you can build your own. If you are going to use the include profile you can skip to step 3.

2. Right-click in your Project panel and select Create > Post Processing Profile. Name the profile whatever you like.

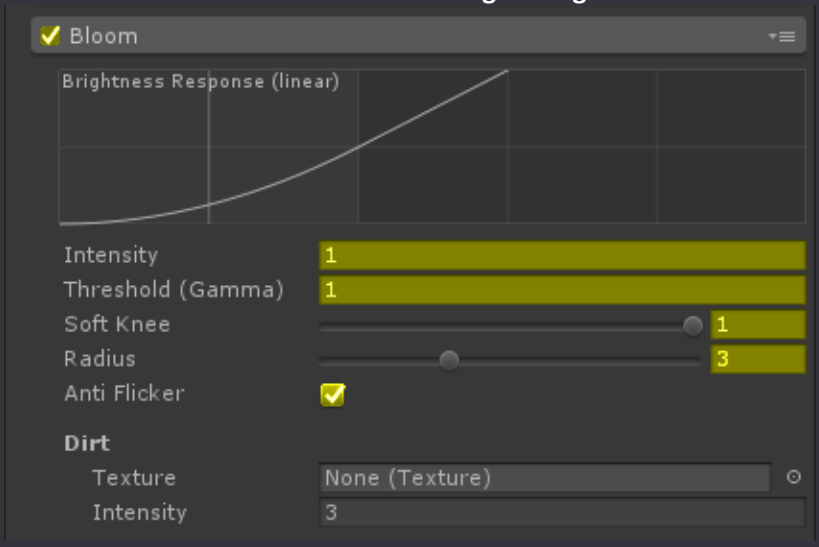


3. Assign your profile to the Post-Processing Behaviour component.

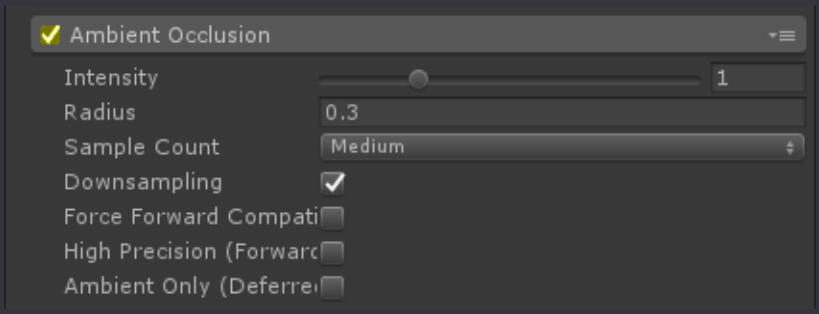


4. Select the Post Processing Profile in the Project panel and apply the following settings:

Bloom Post Processing settings



Ambient Occlusion Post Processing settings



WRAP UP

PolyGrid's simple interface allows you to step up some complex looking effects with ease. Try it out and please share your results!

The code has been heavily commented and modularized to make this package as easy to understand as possible. If you have any questions or comments, please don't hesitate to reach out.

If you find this package useful, please don't forget to leave positive feedback on the Unity Asset Store**. If you have any issues, please contact me with as much information about the issue as you can and I will get back to you as soon as possible.**

Thank you!

RIPCORD // DEVELOPMENT

www.ripcorddev.com