SHADER CONTROL

RELEASE NOTES

Index

Introduction

Thank you for acquiring Shader Control!

Shader Control is a powerful tool that allows you to:

1.- Control which shaders and keywords are compiled in your build.

From the “Build View”, you can choose to ignore some shaders or certain keywords, making your compilation faster and reducing the build size.

You can even specify which keywords combinations will be included in your build. This feature is important to reduce the amount of variants since in most cases you will be interested in only a set of shader features and not all permutations.

2.- Identify which shaders use keywords in your project and modify them to reduce the total count of keywords used (important if you have exceeded the 256 keyword limit!).

The “Project View” allows you to:

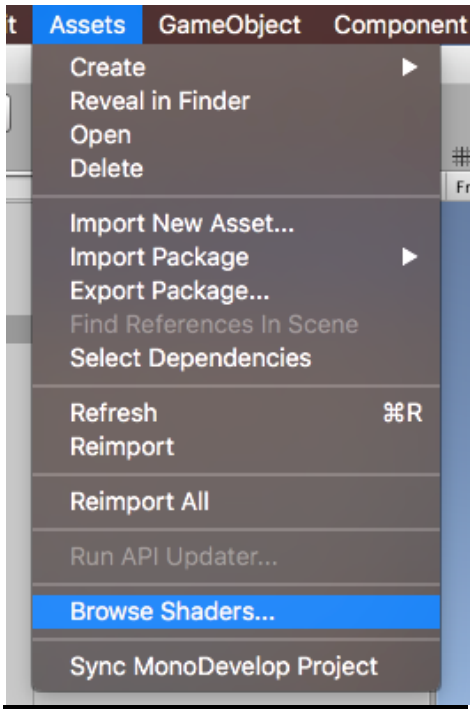
- Quickly locate and list shaders in your project along keywords used.
- Disable/enable keywords per shader.
- Calculate shader variants based on shader modifications.

The key difference between Build View and Project View is that Build View shows all shaders and keywords used during Unity build process, including internal shaders, while the Project View only lists shaders that are accessible in your project folders (internal Unity shaders cannot be modified but you can skip them during build phase).

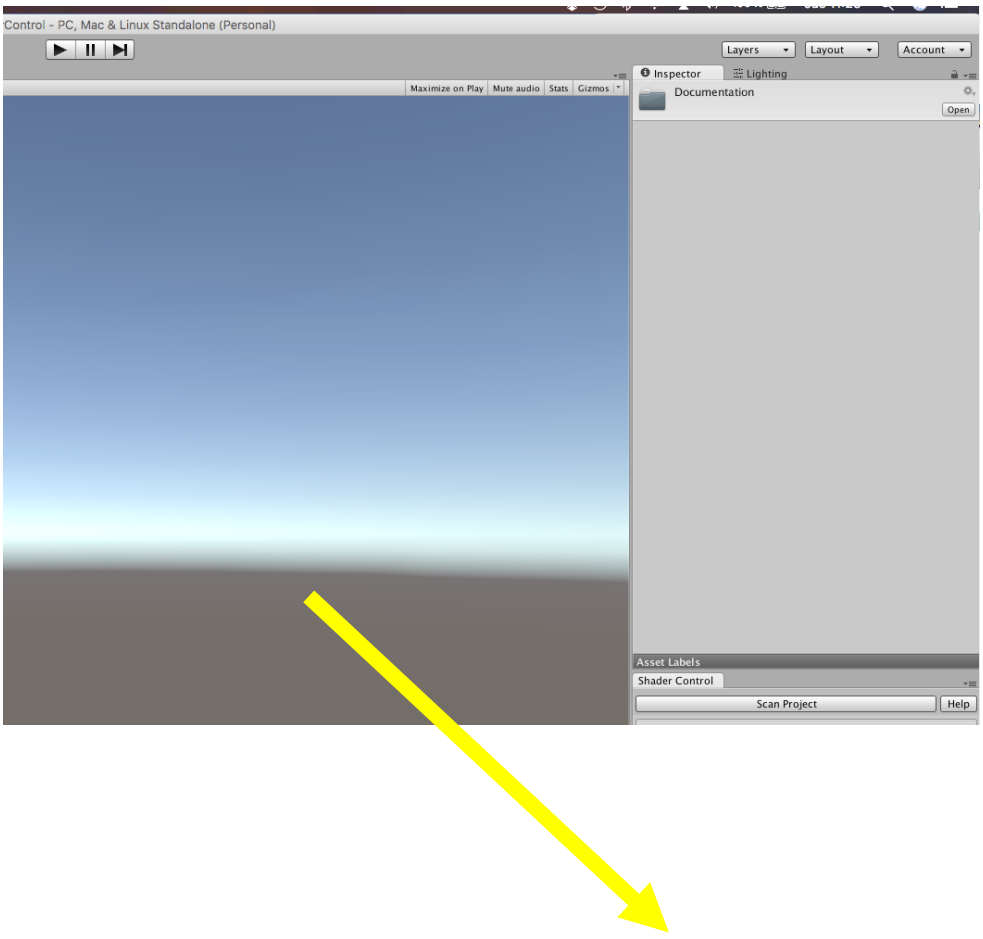
Quick Start

1. Import the asset into your project

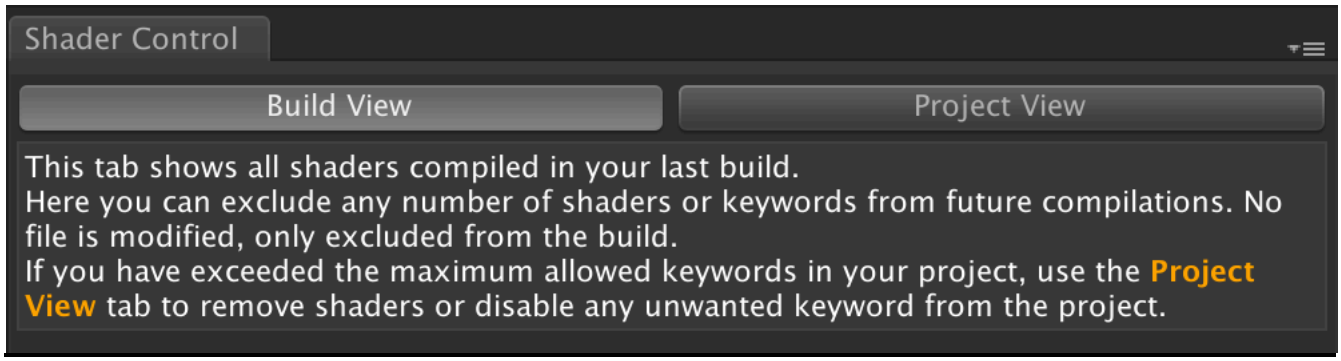
2. Go to menu Assets and select Browse Shaders…



3. Shader Control window will show. It’s useful to drag it and dock it under the Inspector panel:



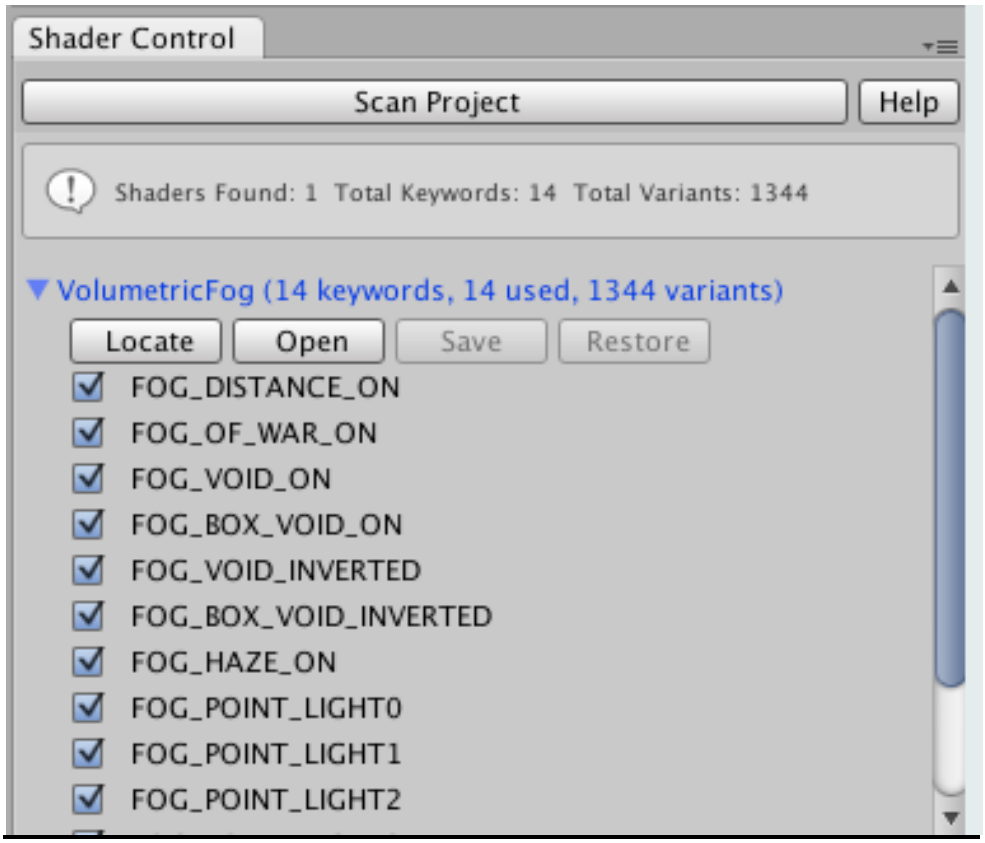
4. The main interface is divided in two areas: “Build View” and “Project View”.



5. Use the “Build View” to control which shaders / keywords are included in your build. The data shown in this tab is from your last build since Shader Control is installed thus you need to make a quick compilation before any data is shown. Press “Quick Build” to instruct Shader Control to optimize the next build. This quick build won’t be functional because shaders are not compiled in the build but Shader Control can collect the data very quickly. After doing a quick build, next builds are normal.

   Note: the Build View can show any shader, included hidden/internal Unity shaders whereas the Project View can only show shaders with source code in your project. Because of this, you will probably prefer to use the Build View most of the time except if you need to reduce the total keyword count.

6. Use the “Project View” to list which shaders are in your project and modify them automatically to remove some keywords. Click “Scan Project” and expand any shader to list its keywords and options:



Click “**Locate**” to select the shader in the project panel or “**Open**” to open it with default system editor (if it does not open, make sure an application is configured to open files with .shader extension).

Click on the checkbox to disable or enable a keyword. Then click “**Save**” to update the shader file.
Shader Control will create a backup copy with same filename ending with “_backup”. Click “**Restore**” to recover the backup copy.

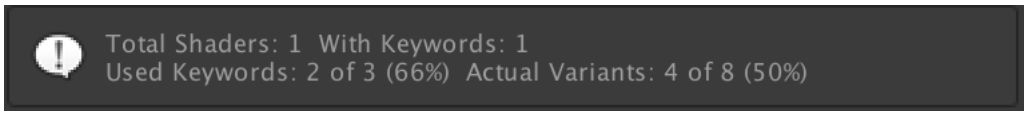
Each keyword you disable will reduce the total shader variants, which will contribute to a reduced build size and compilation time.

“**List Materials**”: this option will be shown next to the shader when one or more materials use it in your project. Click to show a list of the materials and have the ability to quickly locate them in the project panel.

“**Prune Keyword**”: if a shader source is not available (for instance, the Standard shader), Shader Control will still offer you a view of the materials that use that shader . It will also give you the option to prune any of the keywords used by those materials belonging to that shader. Prune keyword will disable the keyword reference in the materials listed. It won’t modify the shader itself, because the source is not available, but the materials won’t contribute to the keyword limit in the project.

Window Stats

The statistics (total shaders, keywords count, etc.) show in Build View is accurate because this data is collected from the build. The Project View only shows data based on the shader files found in your project (it does not include hidden shaders).



“Total Shaders”: the number of files with .shader extension found in the project.

“With Keywords”: number of shaders that use at least one keyword.

“Used Keywords”: the total number of enabled keywords that count towards the Unity limit.
When one or more keywords have been disabled by Shader Control, it will show the current enabled keywords vs the original amount. In the picture above, the original total keywords is 3 and one keyword has been disabled using Shader Control hence it shows “2 of 3”.

“Actual Variants”: the total number of shader variants (different compilations) that are generated by Unity according to the keywords permutations.

The “**Clean All Materials**” button disables any keyword stored in any material that’s also disabled in the shader source file.

When you disable a keyword using Shader Control and press Save, it does two things:

- Modifies the shader and comment out the keyword so it does not generate shader variants.

- Disables that keyword from any material that might be using it.

The button “Clean All Materials” just repeat the second step but does this for all materials in the project. It takes every material and examines the corresponding shader for any referenced keyword in that material. If that keyword is disabled in the shader source code, then it’s also removed from the material. Usually this step is not needed but pressing the button is harmless and ensures that there’re not disabled keywords set on the materials of the project that could be set by scripts.

About shader keywords

Shaders are extremmely optimized pieces of code that run in the GPU. Because they need to use the fewer number of instructions, shader keywords related to their features or options can be used to create compiled variations of the same code base. This way the code actually executed in the GPU is locally optimized with fewer register usage and less conditional jumps, resulting in faster execution.

For example, in a fog shader, an option could be to blend the fog using 2 colors instead of only one, creating an additional gradient effect. Since you may want to use a single color, a keyword could be used to generate two variants of the same fog code at compilation time, one that uses the second color and another one that just uses one color. This keyword could be named FOG_USE_GRADIENT and is speficied in the shader using a special compiler instruction called “#pragma multi_compile”. When the compiler detects a #pragma multi_compile line, it will generate several variations of the shader according to the keywords found.

Following the above example, in the shader file you could find:
#pragma multi_compile __ FOG_USE_GRADIENT

The underscore tells the compiler it must generate a shader variant without any keyword, and then a variant using the keyword FOG_USE_GRADIENT. This keyword can be detected later in the shader code using a compiler conditional that surrounds a piece of code. That piece of code will be include in that compilation if the keyword is enabled.

Many complex shaders, also called “uber shaders”, make use of lot of keywords so as much as possible code can be used in the same render pass. This approach creates faster shaders compared to chaining effects on sequential render passes.

But uber shaders usually adds lot of features (shader keywords) that many users won’t use, producing unnecesary variants hence bigger build size and compilation time. This is the case where Shader Control tool is very useful. By inspecting the shader keywords you can decide to remove the keywords from the shader file itself, avoiding compiling unused variants.

Note that modifying shaders incorrectly can render them useless or produce undesirable visual effects! Make sure you have a backup copy of your shaders or project before applying any automatic change. Also you may need to contact the asset author regarding some shader keywords to ensure you can safely remove them from the shader file.

Create Shader Variant Collections

Shader Control can also create shader variant collection assets which can be used to speed up shader loading time. For a general overview and uses of shader variant collection assets, please read the official Unity documentation: https://docs.unity3d.com/Manual/OptimizingShaderLoadTime.html

A shader variant collection includes one or more shaders along a number of different variants (combination of keywords). The purpose of a Shader Variant Collection is telling Unity that your game will use those variants so you can warm them up using scripting or during scene load avoiding hiccups due to shader compilation at runtime.

A Shader Variant Collection of any shader can be created from the Build View. Locate the shader, expand it and click the "Advanced" button. In this new window you will be able to specify different variants and also create a shader variant collection.

Note that when specifying specific variants from this window, Shader Control will ignore any other variant for that shader which will save time and space in your build.

Support & Contact info

We hope you find the asset easy and fun to use. Feel free to contact us for any enquiry.
Visit our Support Forum for usage tips and access to the latest beta releases.

Kronnect
Email: contact@kronnect.com
Kronnect Support Forum: https://www.kronnect.com/support