Shaders Limiter

v1.2.0

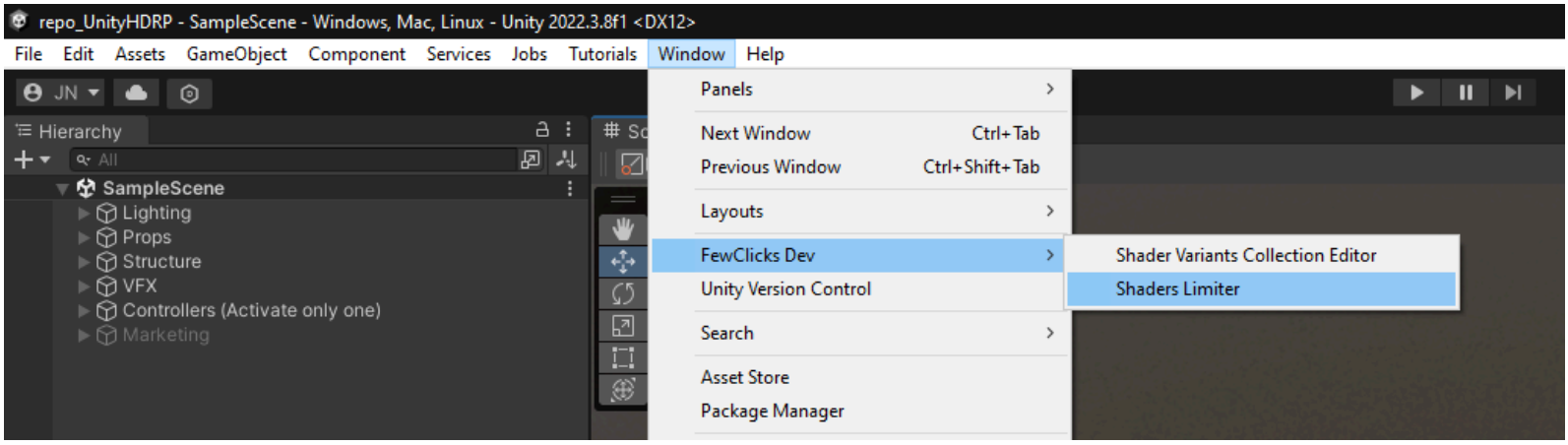
Shaders Limiter is an editor tool that helps reduce the build time and memory usage of your Unity project. It comes with a small shader keyword library to help you better understand what's going on under the hood.

Core features

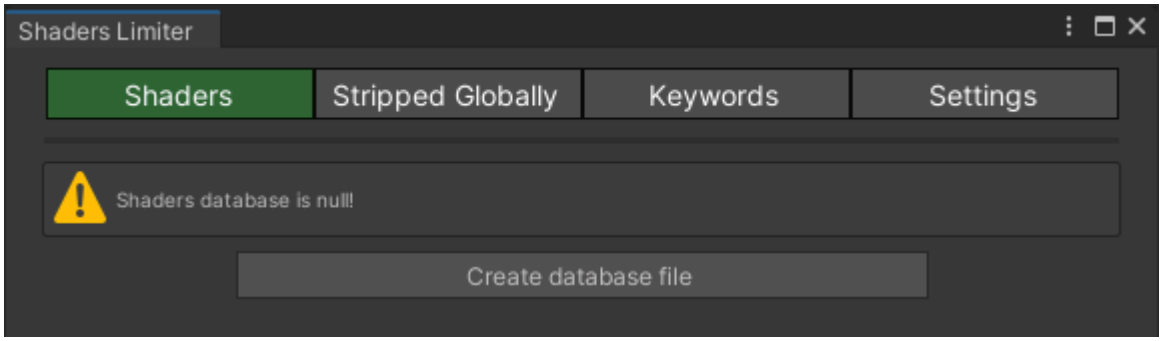
- Stripping shaders based on keywords (per shader or globally)
- Stripping all shaders outside of those specified in the shader variant collections
- A preview of all shader keywords specified by Unity in the project
- A preview of unique, all, and currently enabled global keywords used by the project
- Shaders' keyword library that can be easily extended
- Shader variant collection inspection, modification, and merging

[0] Installation

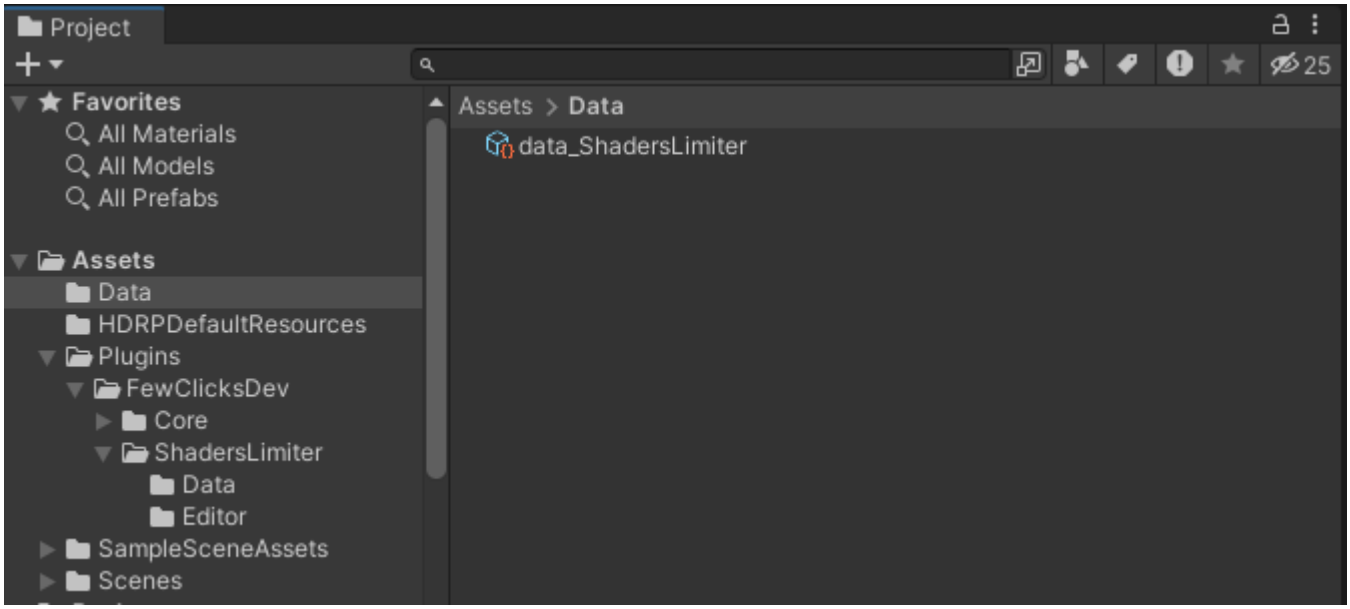
After you download and import the package from the Asset Store, new menu items will appear under the 'Window' tab. Click on it to open the window.



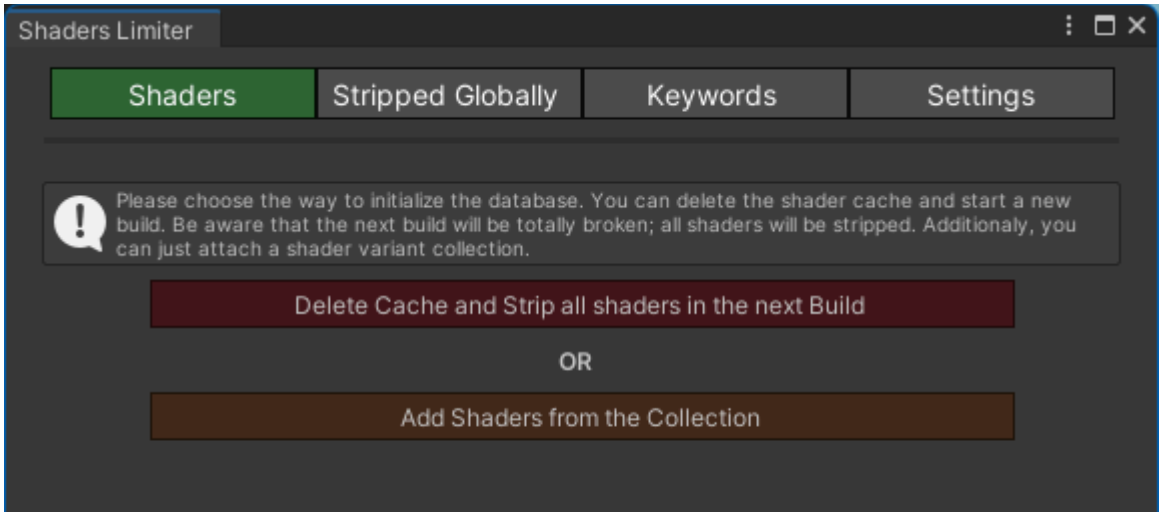
The tool's window will pop up with the information that the database hasn't been created yet. Click on the button below.



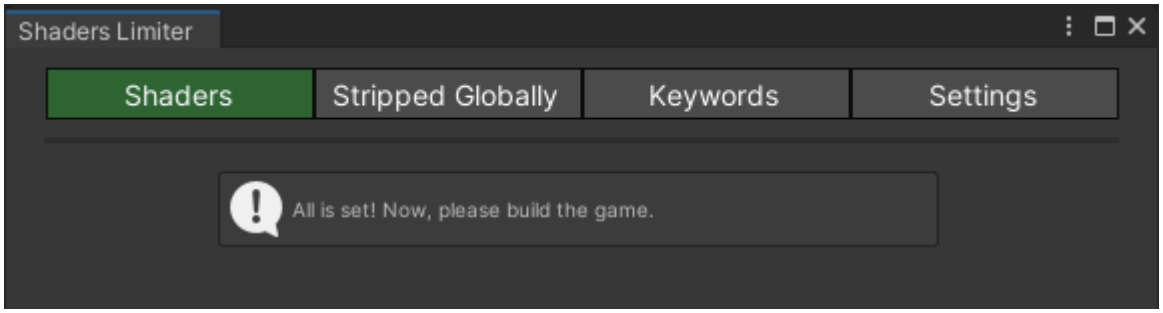
Choose the right place for the database (it's a scriptable object, so it must be within the 'Assets' directory).



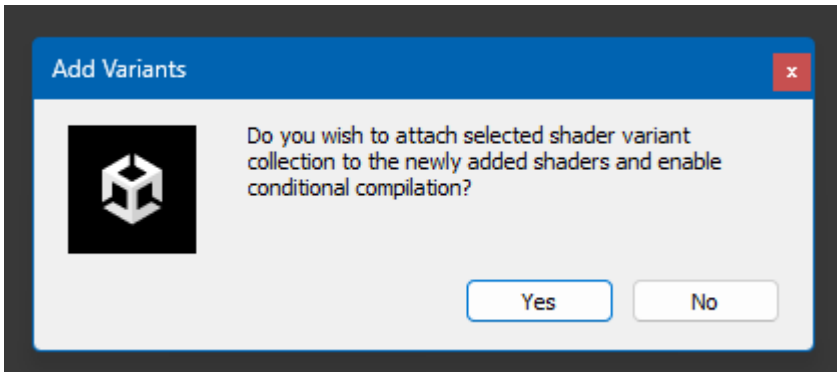
A newly created database is empty. Please press the red or orange button, depending on the way you wish to initialize the database.



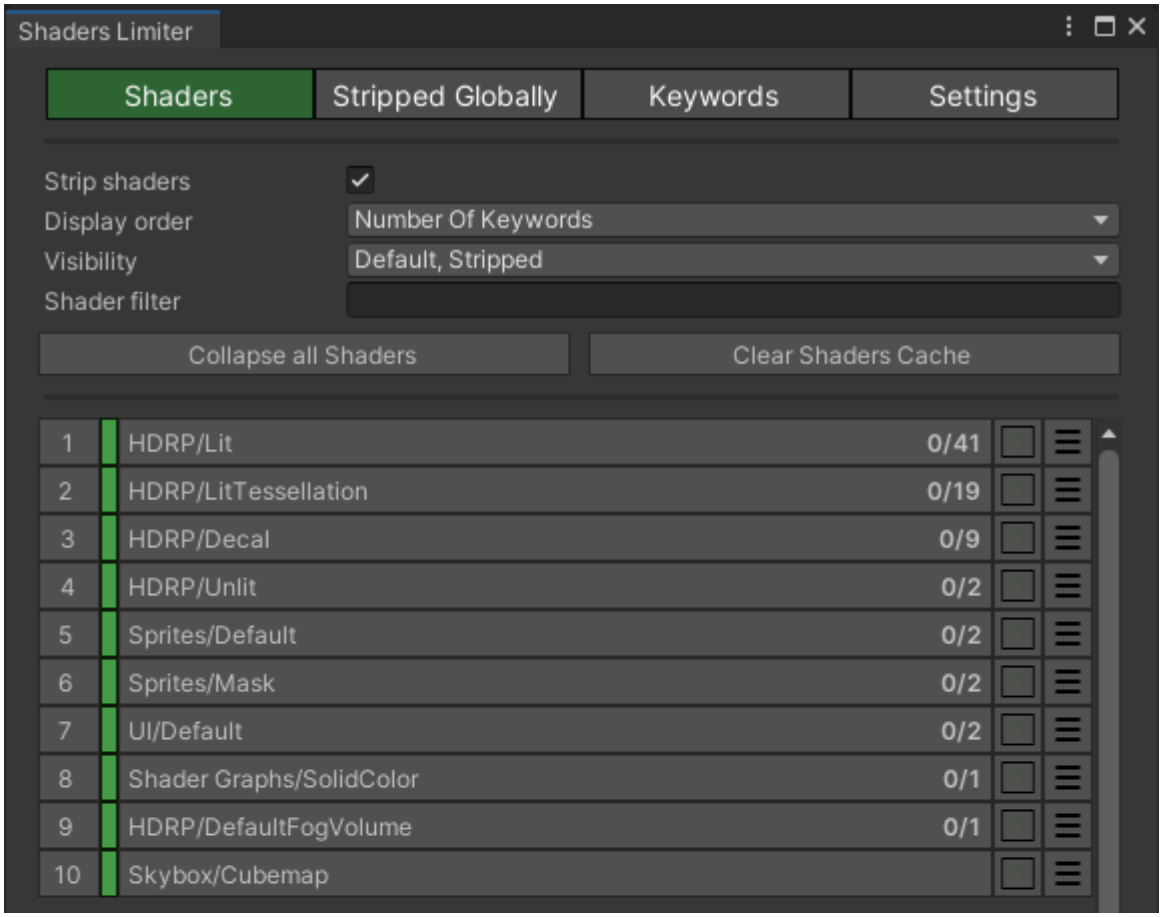
When using the “**Delete Cache**" option, after you click the button, the editor window will look like the screenshot below. It indicates that everything is set correctly; please build your project! If you are using addressables, don’t forget to build them first.



If you decide to go with the “**Add Shaders from the Collection**” options, you will be prompted to attach a shader variant collection asset. The tool will also ask if you want to attach this collection to all shaders and enable conditional compilation on them (only variants included in the collection will be compiled during the build time).



After the successful initialization, don't forget to set the "**Strip shaders**" flag to true to enable shader stripping in the upcoming builds.

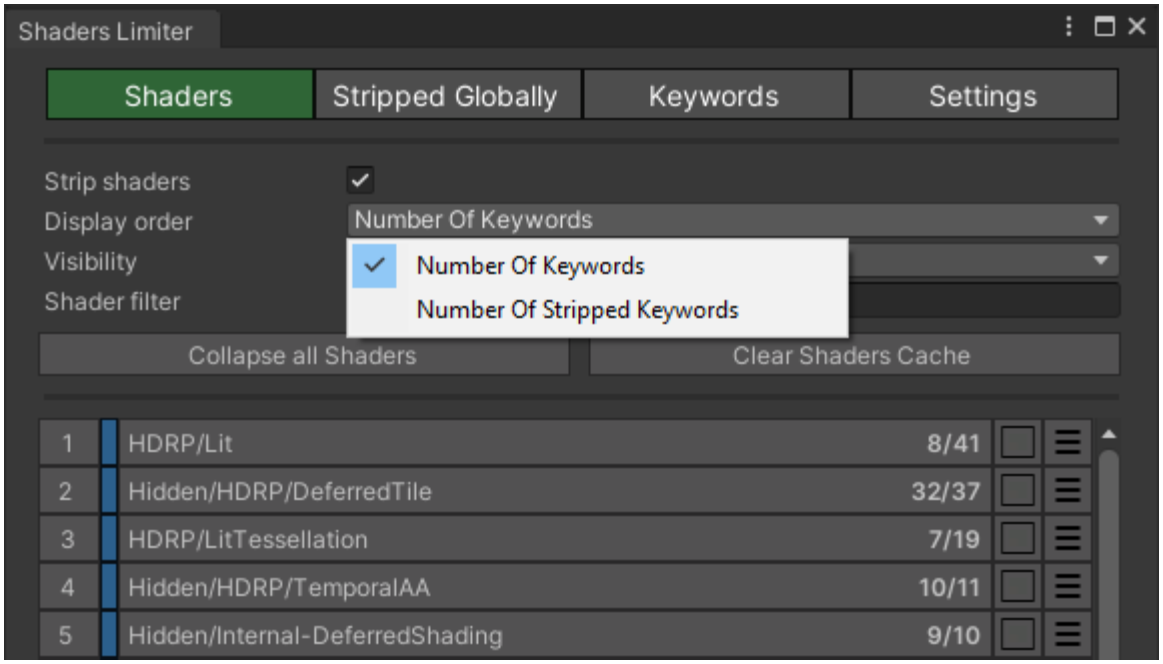


[1] Shaders

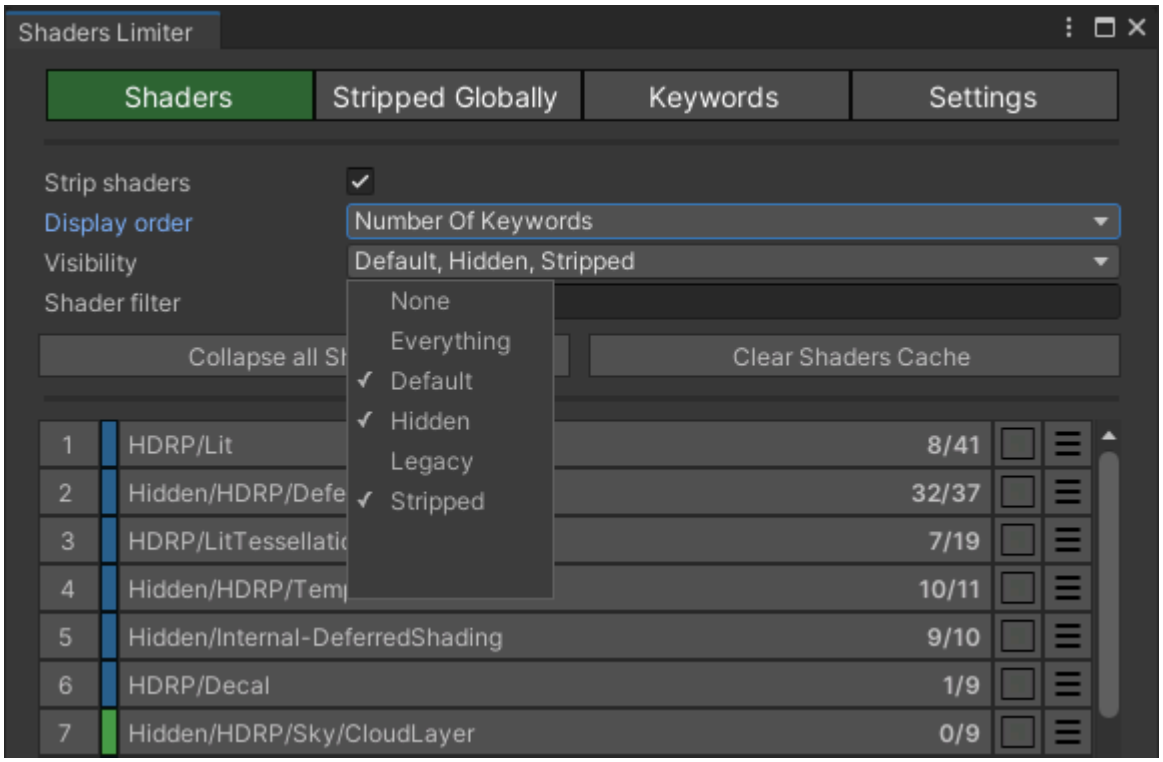
Strip Shaders - this is the main setting of the tool, turning it on and off.

Keep in mind that Unity recompiles shaders only if it detects any changes. If you already stripped all of the shaders and started a new build without changing anything, setting this flag to false won't make any difference. Unity will assume that all shaders are already compiled and ready to be added to the game. It's usually a good practice to 'Clear Shaders Cache' if you set 'Strip Shaders' to false.

Display Order - lets you sort all shaders based on total number or stripped keywords.



Visibility - lets you filter out already stripped, hidden, or legacy shaders.

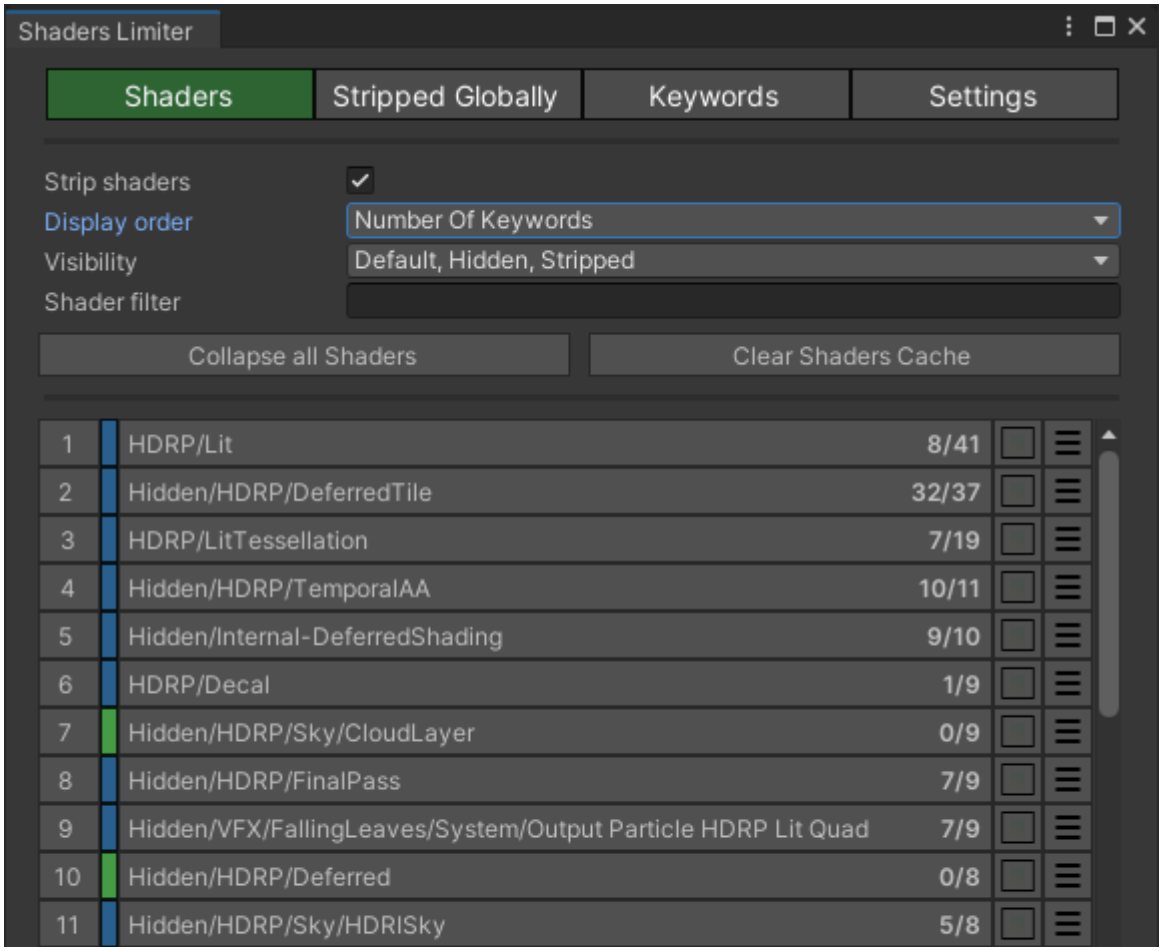


Shader filter - lets you filter out shaders based on the provided string.

Collapse all Shaders - helper function that lets you hide extended views of all shaders.

Clear Shaders Cache - function that will delete all shader caches inside the Library folder, forcing Unity to recompile all of them from the start.

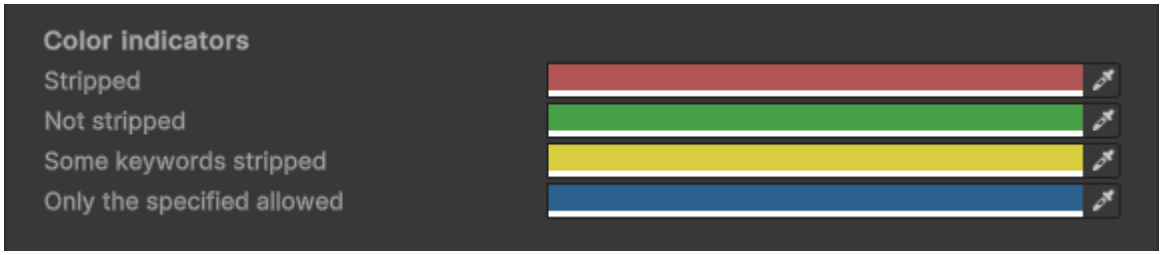
Shaders list



Based on your display order and visibility settings, this part of the window shows all shaders that have been compiled by Unity during builds. Each shader is represented by the elements below.

Shader index - just a number indicating an order in the current list.

State color - small colored label indicating if or how shaders are being limited in the builds. You can change colors in the Settings tab.



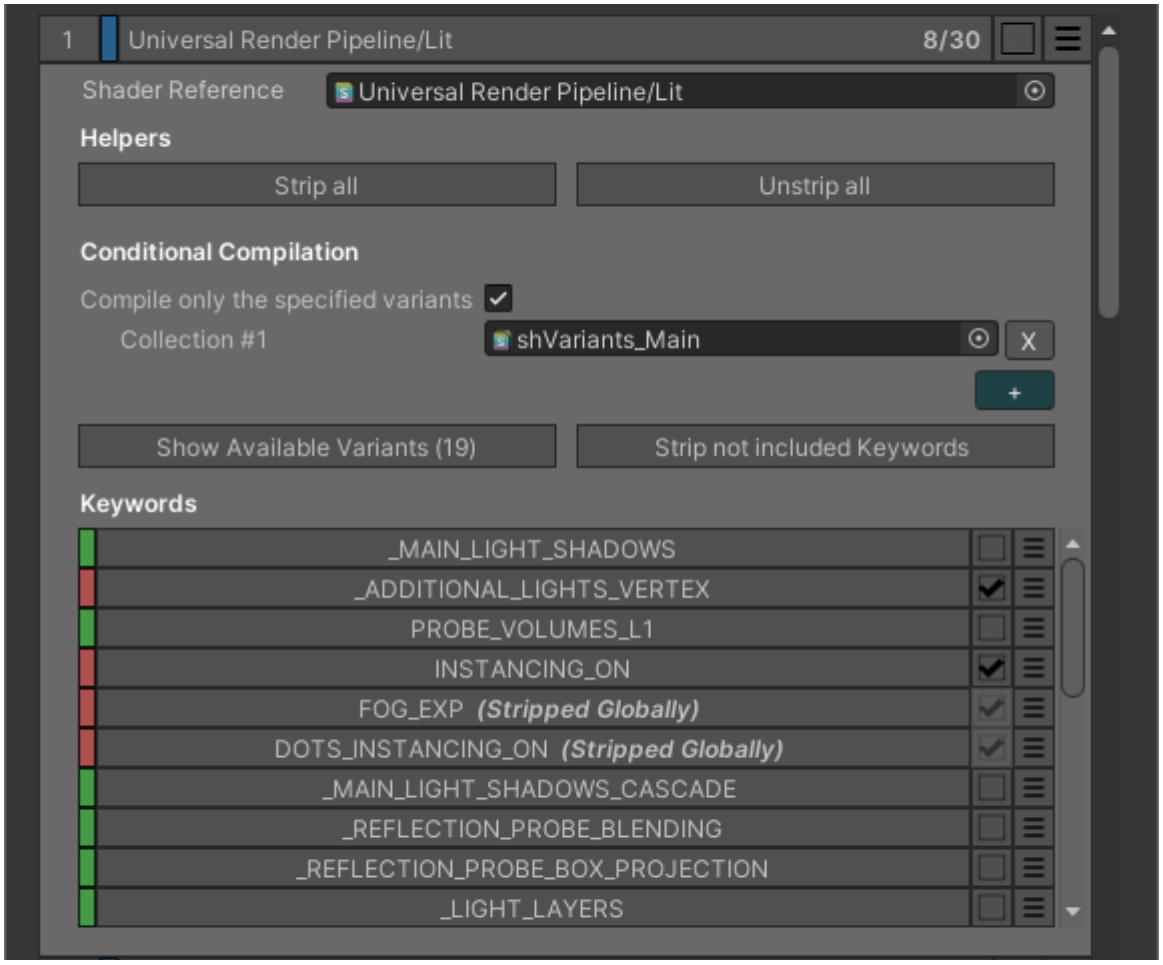
Label - it's the name of the shader; click it to open the expanded view.

Stripped count - a number indicating how many keywords are stripped out of the total number that this shader is using.

Toggle - used to strip the shader completely from the build.

Options menu - it provides some functionality that is described in the 'Additional options' section.

Shaders expanded view



Shader Reference - actual shader file that is associated with the shader to strip.

Strip all - lets you strip all shader keywords.

Unstrip all - lets you unstrip all shader keywords.

Compile only the specified variants - stripping will be determined by the shader variant collections specified below.

You can use as many collections as you want, but I recommend using only one collection for the whole project that contains all the shaders from all your levels.

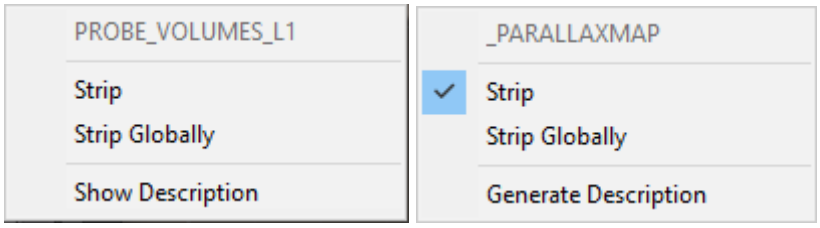
Show available Variants - it will open an additional window where you can inspect all shader variants that have been found in the provided collections. Number in the brackets is

Strip not included keywords - this helper will strip all shaders that haven't been found in the provided shader variant collections.

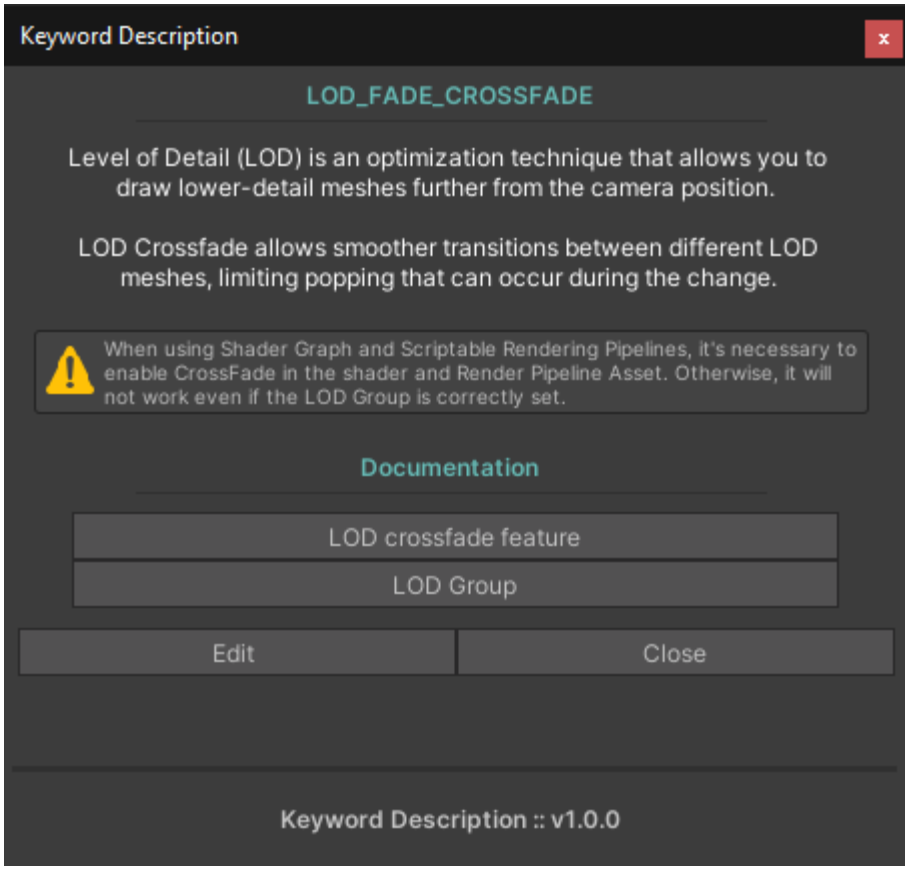
Keywords - complete list of all keywords that have been found during the build process. You can strip them one by one by clicking on the label or toggling next to it. The color indicator will be either red or green.

Globally stripped keywords will have an additional label next to the name. They can't be unstripped on the shader level; use the options menu or the 'Stripped Globally' tab to manage them.

Each keyword has an option menu that can be opened by clicking the 'Option' button on the right side. You can use it to strip the keyword globally, show or generate a description.

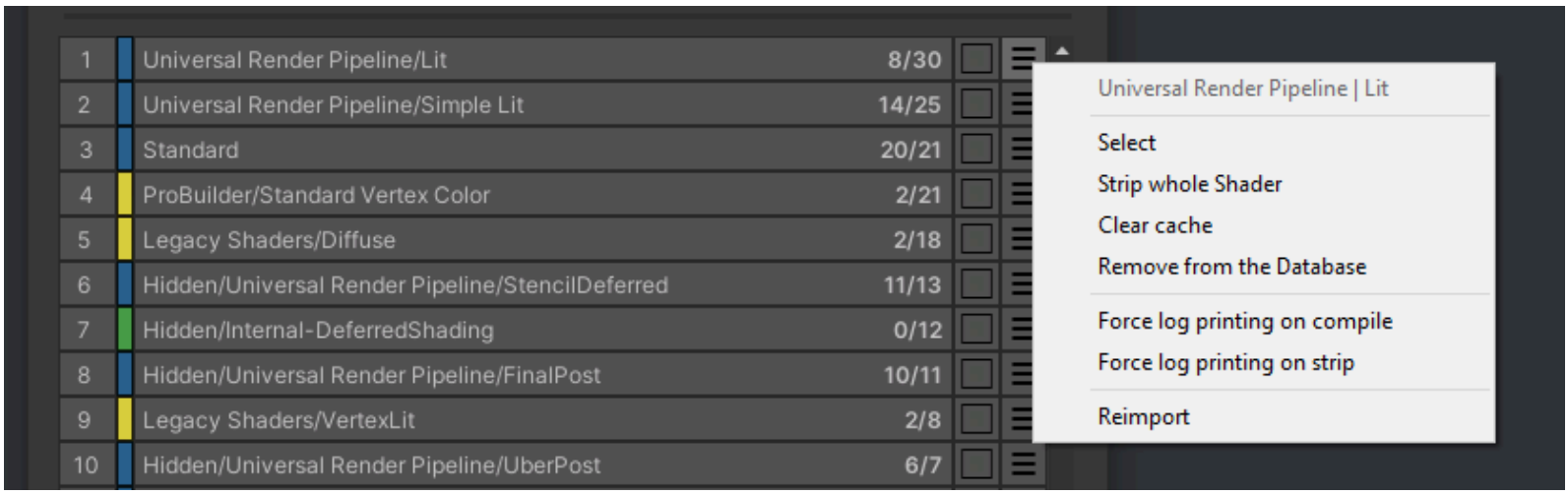


Keyword description - it gives you some overall information about the keyword and can help you better understand Unity by providing a description and documentation links. You can add your own descriptions for other team members or edit existing ones.



Additional options

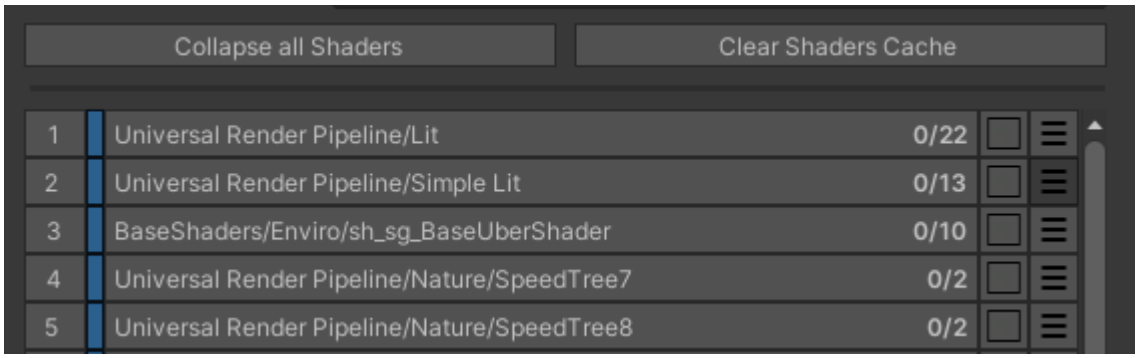
Each shader has an options button right next to the ‘Strip’ toggle. Pressing it will open a context menu visible below. The light gray text at the top is the shader name.



Select - select and ping the shader's reference in the project window.
Strip whole Shader - toggle the ‘Strip’ flag on the shader. This action will reimport the shader and delete its cache, forcing Unity to recompile it in the next build.
Clear cache - clear shader cache that is located in the ‘Library’ folder.
Remove from the Database - remove shader tracking from the database. Keep in mind that if the shader is compiled by Unity during the next build, it will return to the database.

Force log printing on compile - force Shaders Limiter to print logs on shader variant compilation even if the main logs flag is disabled.
Force log printing on strip - force Shaders Limiter to print logs on shader variant stripping even if the main logs flag is disabled.

Force log printing on compile and strip will also change the options button tint to indicate the override, as seen in the screenshot below.

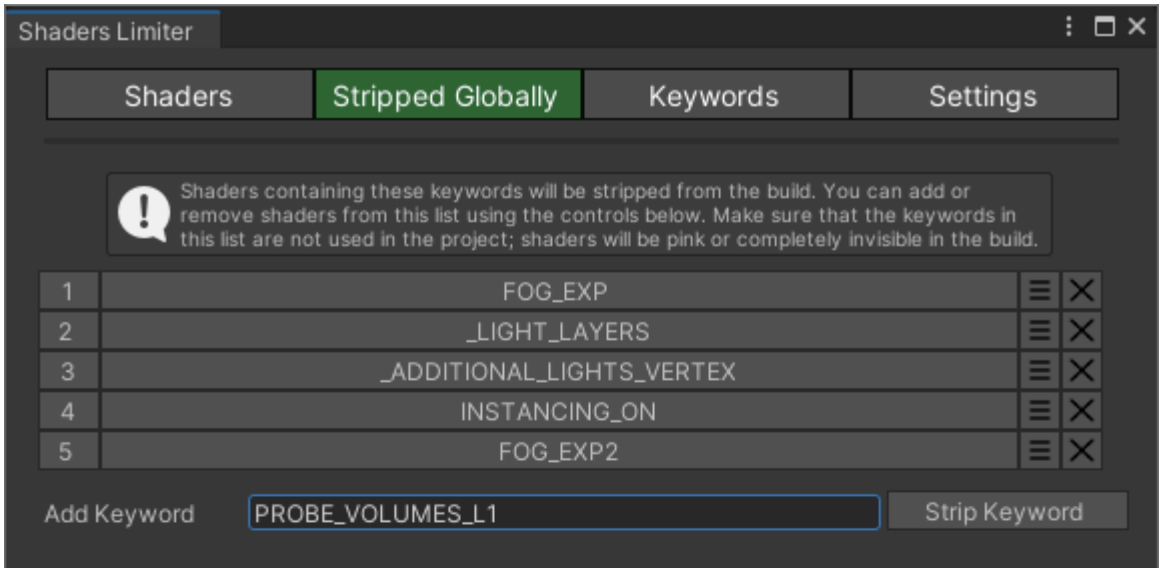


Reimport - reimport the shader. This can be used to find new shader variants that were not tracked by the variant collection. Note that built-in shaders can’t be reimported.

[2] Stripped Globally

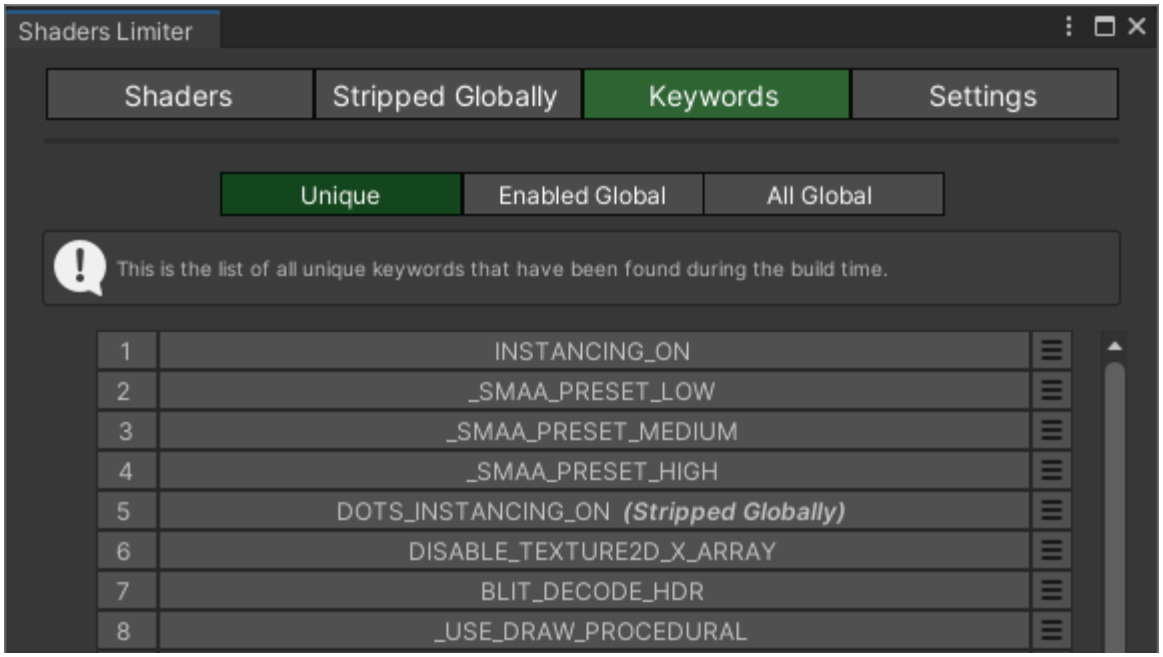
Each shader keyword that you set as “Stripped Globally” will show up in this tab for you to see and manage. Here, you can unstrip, show, or generate descriptions for them.

At the bottom of the keywords list, there is a text field for you to strip keywords from a string. Keep in mind that shader keywords don't contain empty spaces, and they are case-sensitive.



[3] Keywords

This tab is split into three submenus. Similar to Shader's expanded view, here you can inspect and strip keywords globally.



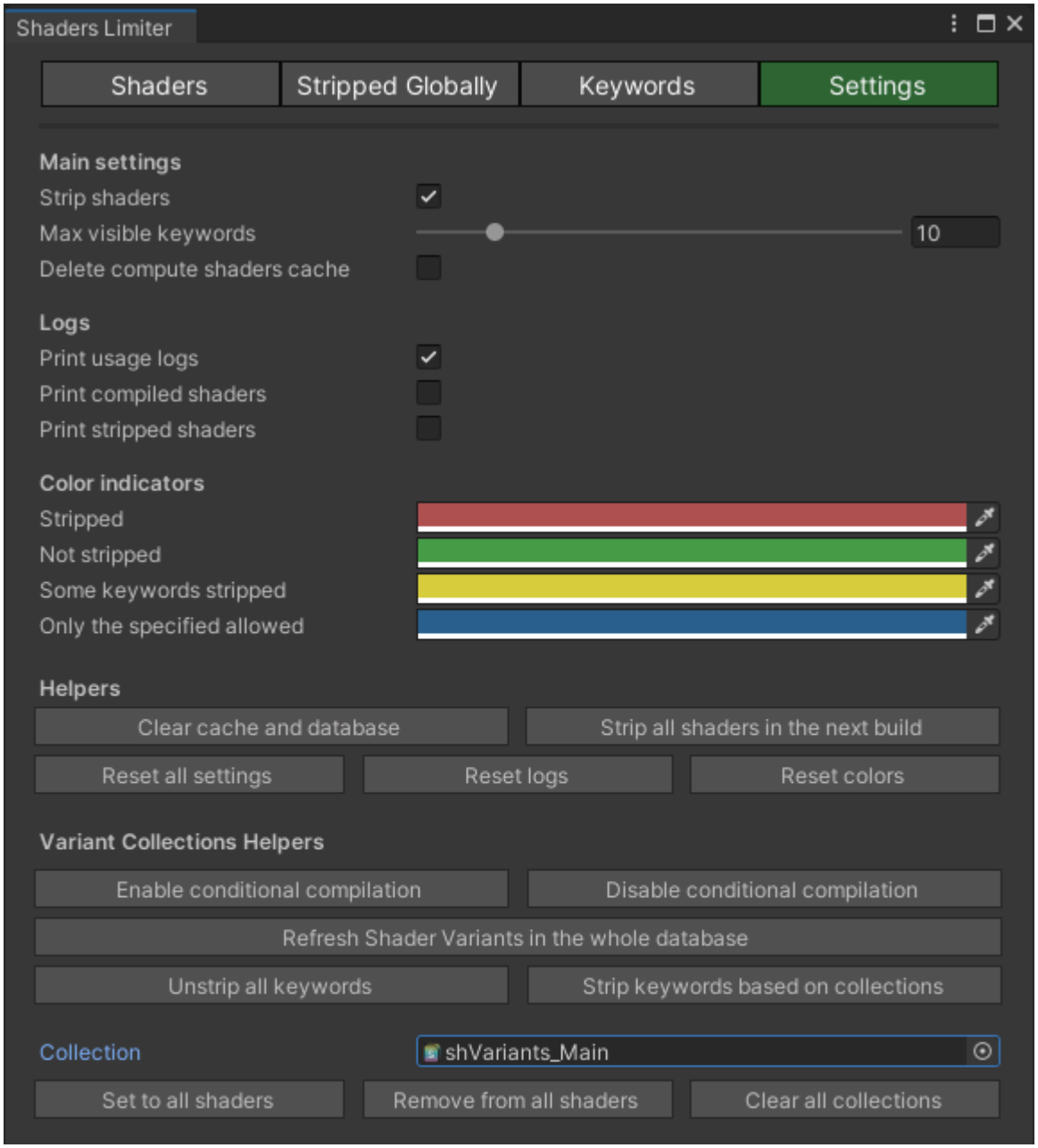
Unique - here you can preview all the unique shader keywords that you have in your project shaders.

Enabled Global - this list contains all of the global shader keywords that are currently enabled in your Unity project. They are based on your project version, used rendering pipeline, target platform, and project settings.

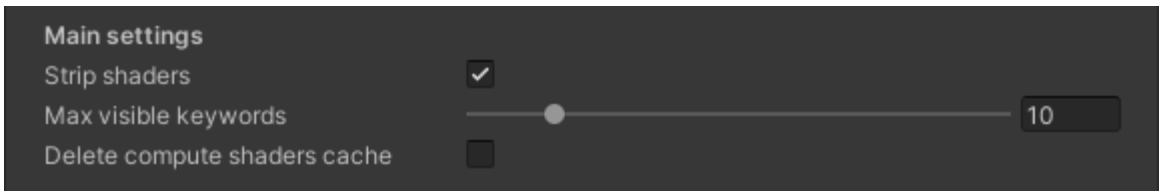
All Global - here you can preview all global keywords that Unity is aware of.

[4] Settings

In this tab, you can adjust all available tools' settings. It also contains some helper buttons that can alter all shaders to strip in your project. It is split into several sections.



Main Settings

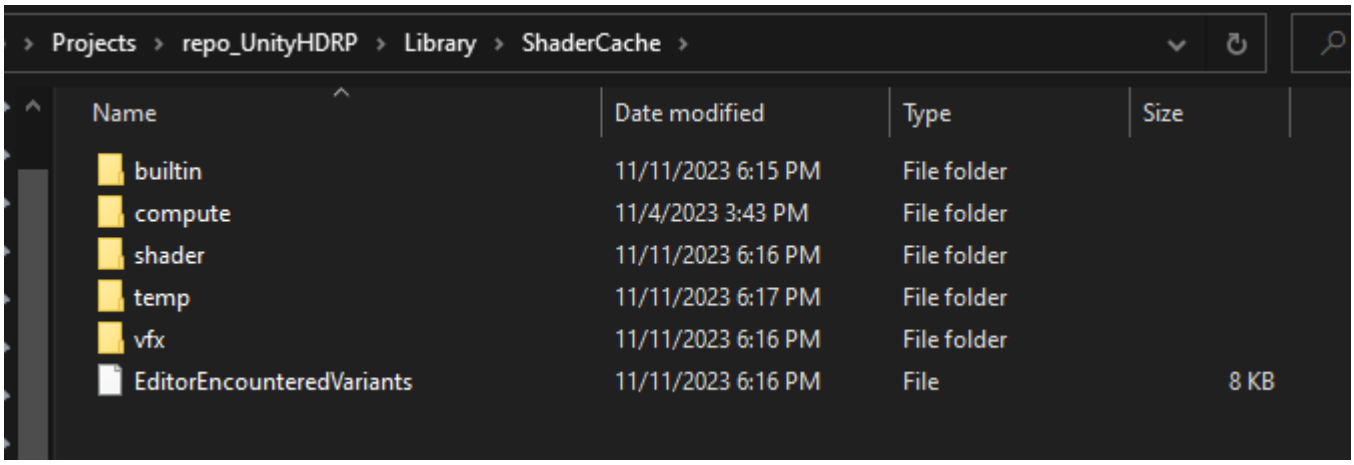


Strip shaders - this is the main setting of the tool, the same as in the ‘Shaders’ tab. Use it to enable or disable this tool.

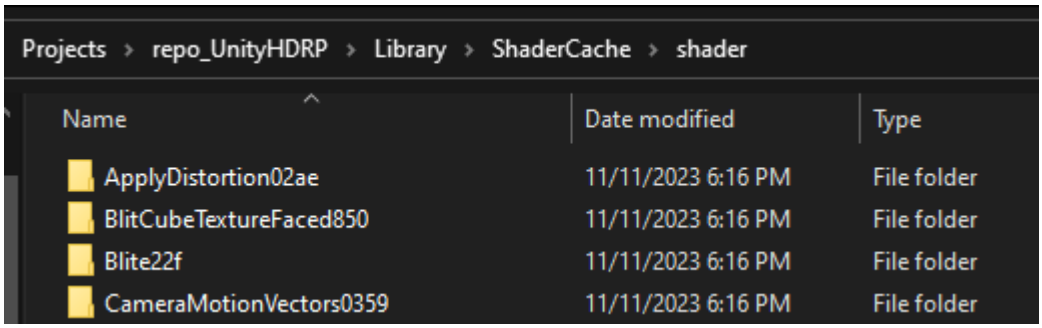
Max visible keywords - number of keywords that are visible in the expanded shader view. Ten seems like a good default value, but depending on your shader complexity and your monitor orientation, you can increase or decrease this value for your convenience.

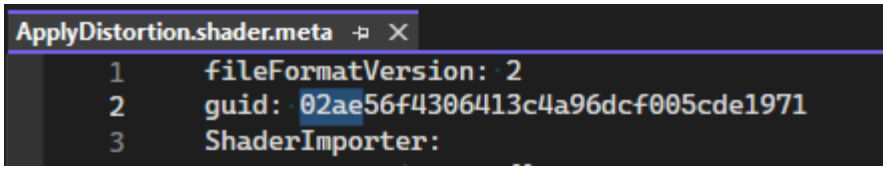
Delete compute shaders cache - when you delete shader caches to force Unity to recompile them from scratch, you can set this flag to true to include compute shaders. They are not being stripped by this tool, so it's up to you if you want to delete their cache or not.

Shader cache can be found in the Library folder. They are grouped based on their type and usage. File ‘EditorEncounteredVariants’ contains all shader variants that Unity Editor already compiled while you were playing. They can be saved into Shader Variant Collections using Project Settings / Graphics.

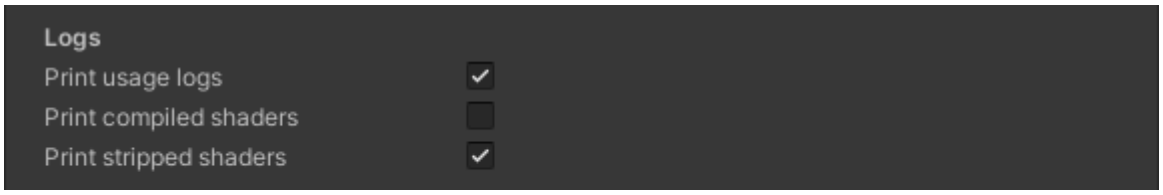


To delete a single shader cache, Shaders Limiter generates a path based on the shader name and the first four characters of its GUID. You can see an example below for the shader ‘ApplyDistortion’.





Logs



Print usage logs - flag enabling some of the tool usage logs. It's up to you if you want to get any information about what's going on when using Shaders Limiter.

Print compiled shaders - flag that can help you debug which shaders are being compiled during the build.

Keep in mind that printing thousands of logs to the console can impact overall performance, so it's recommended to keep this flag to 'False' unless necessary. The Same rule applies to 'Print stripped shaders'.

Print stripped shaders - flag that can help you debug which shaders are being stripped during the build.

Both 'Print compiled shaders' and 'Print stripped shaders' can be overridden per shader using the additional options menu.

Color Indicators

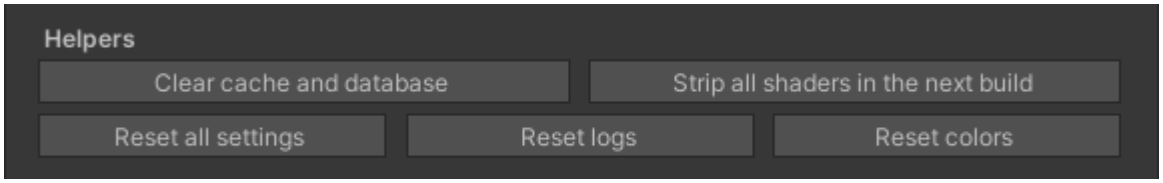
Stripped - color of all stripped shaders and keywords. Shader that is set as not stripped, but all of its keywords are, will have that color too.

Not Stripped - default color for not stripped shaders and keywords.

Some keywords stripped - color indicating that for a given shader, some of its keywords.

Only the specified allowed - color that indicates stripping of this shader is based on the provided shader variant collections.

Helpers



Clear cache and database - this helper will clear all shader cache (forcing Unity to recompile all shaders during the next build) and will clear the database, removing all setup.

It's the best way to start from scratch, without deleting the database file.

Strip all shaders in the next build - it sets the same flag that is used during the first build after the installation. It will strip all shaders, causing the next build to be unplayable.

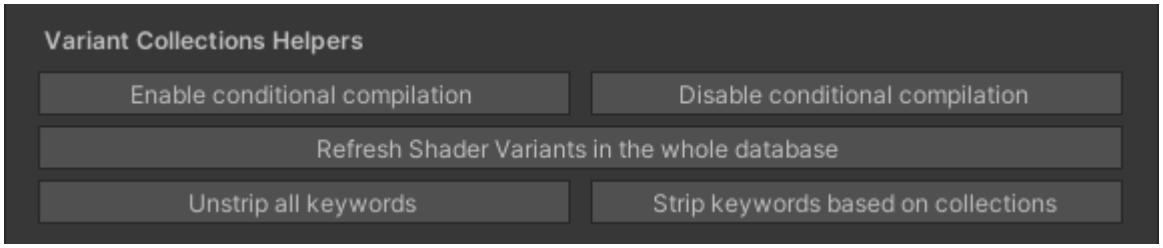
This can be used if you want to start from scratch or if you just added some super complicated shaders to the project and don't want to wait for thousands of variants to compile during the next build.

Reset all settings - reset all settings to the defaults.

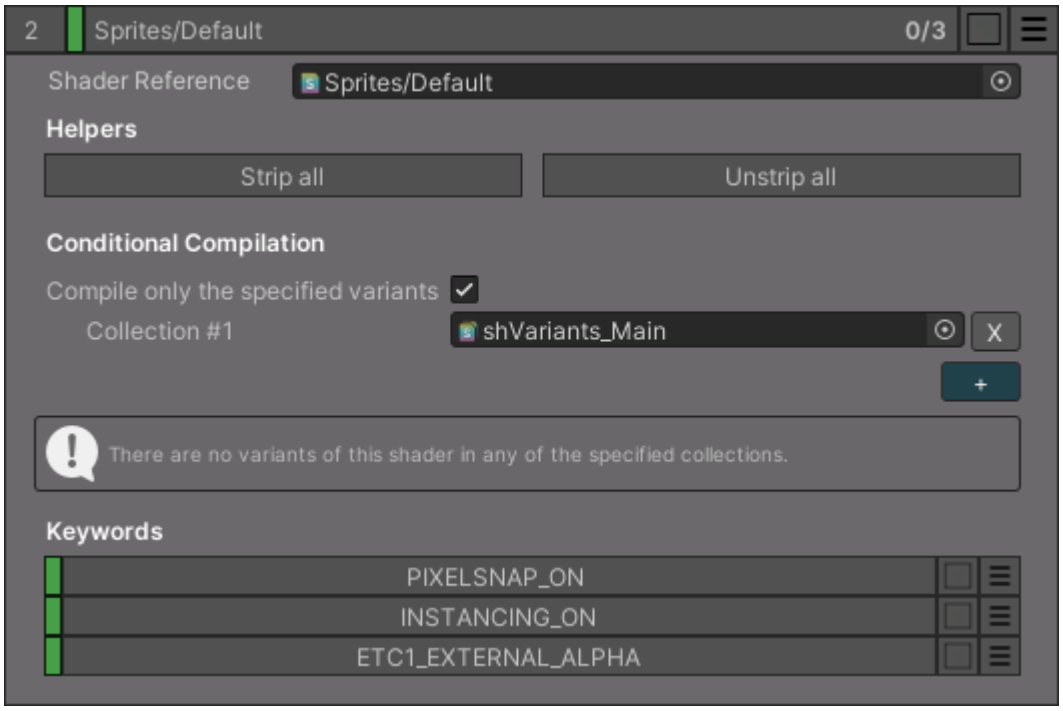
Reset logs - reset log settings to their default.

Reset colors - reset colors to their defaults.

Variant Collections Helpers



Enable conditional compilation - enabling “Compile only the specified” flag on all shaders to strip. Keep in mind that this flag will have no effect if there are no included shader variants in the attached collection. The tool will inform you about that in the shader expanded view. The color indicator won’t change either.



Disable conditional compilation - disabling the “Compile only the specified” flag on all shaders to strip.

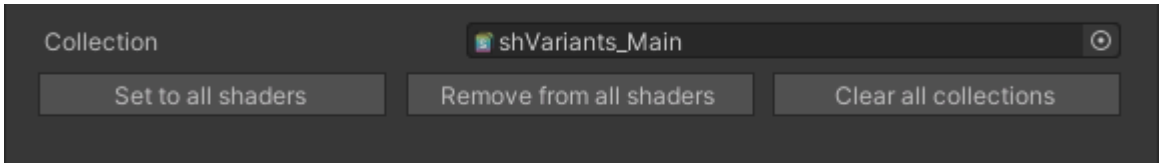
Refresh Shader Variants in the whole database - this button will force the tool to recalculate all variants saved in the shaders to strip. Press it after you have modified a shader variant collection attached to the database.

Unstrip all keywords - helper button that will unstrip all shader keywords (excluding globally stripped) in all shaders to strip. This can be used to make sure that you don’t strip any unnecessary keywords except for the ones that are not present in the provided shader variant collections.

This can be done per shader basis in the shader expanded view.

Strip keywords based on collections - another helper button that will at first unstrip all shader keywords in the whole database and then strip keywords that are not included in the provided shader variant collections.

This can be done on a per shader basis in the shader expanded view.



Collection - shader variant collection reference that will be used for functions available below it.

Set to all Shaders - this button will add a referenced shader variant collection to all Shaders to strip. It's a great way to start using this tool: make a master collection with all needed shader variants and then attach it to all shaders in the database.

Remove from all Shaders - it will remove referenced variant collections from all shaders to strip. It can be used for collections cleanup.

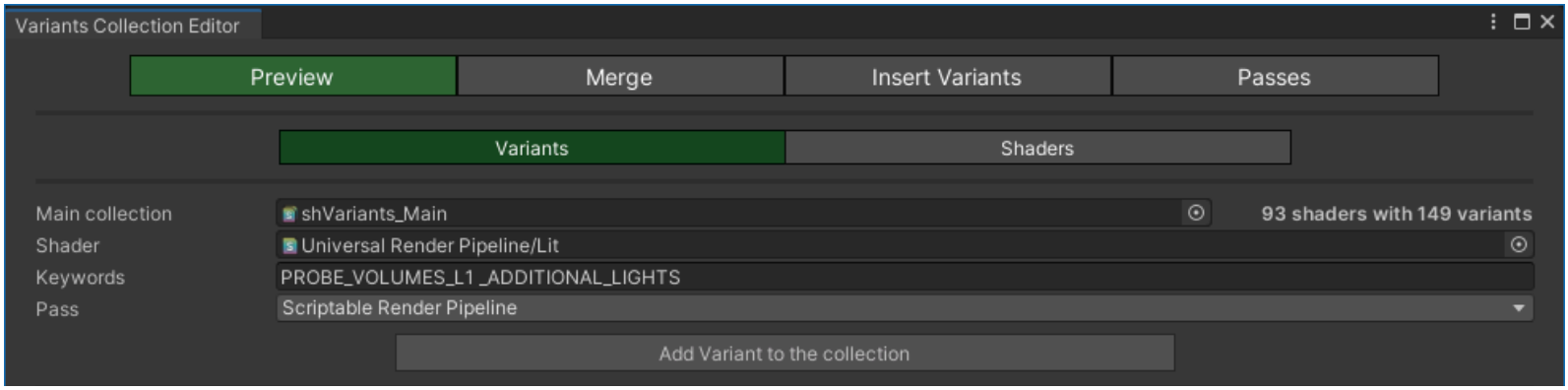
Clear all collections - this button will remove all shader variant collections from all shaders to strip.

[5] Shader Variants Collection Editor

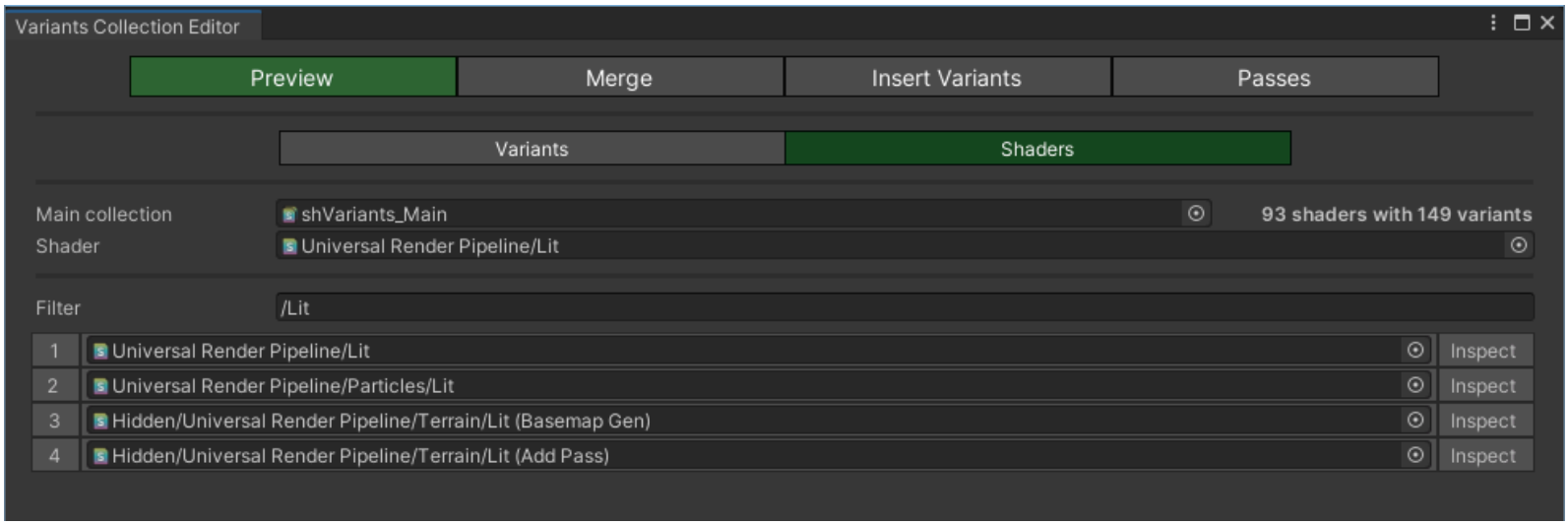
This window can be used to manage shader variant collections attached to the Shaders Limiter database. This tool is split into several tabs, each containing different functionality.

Preview

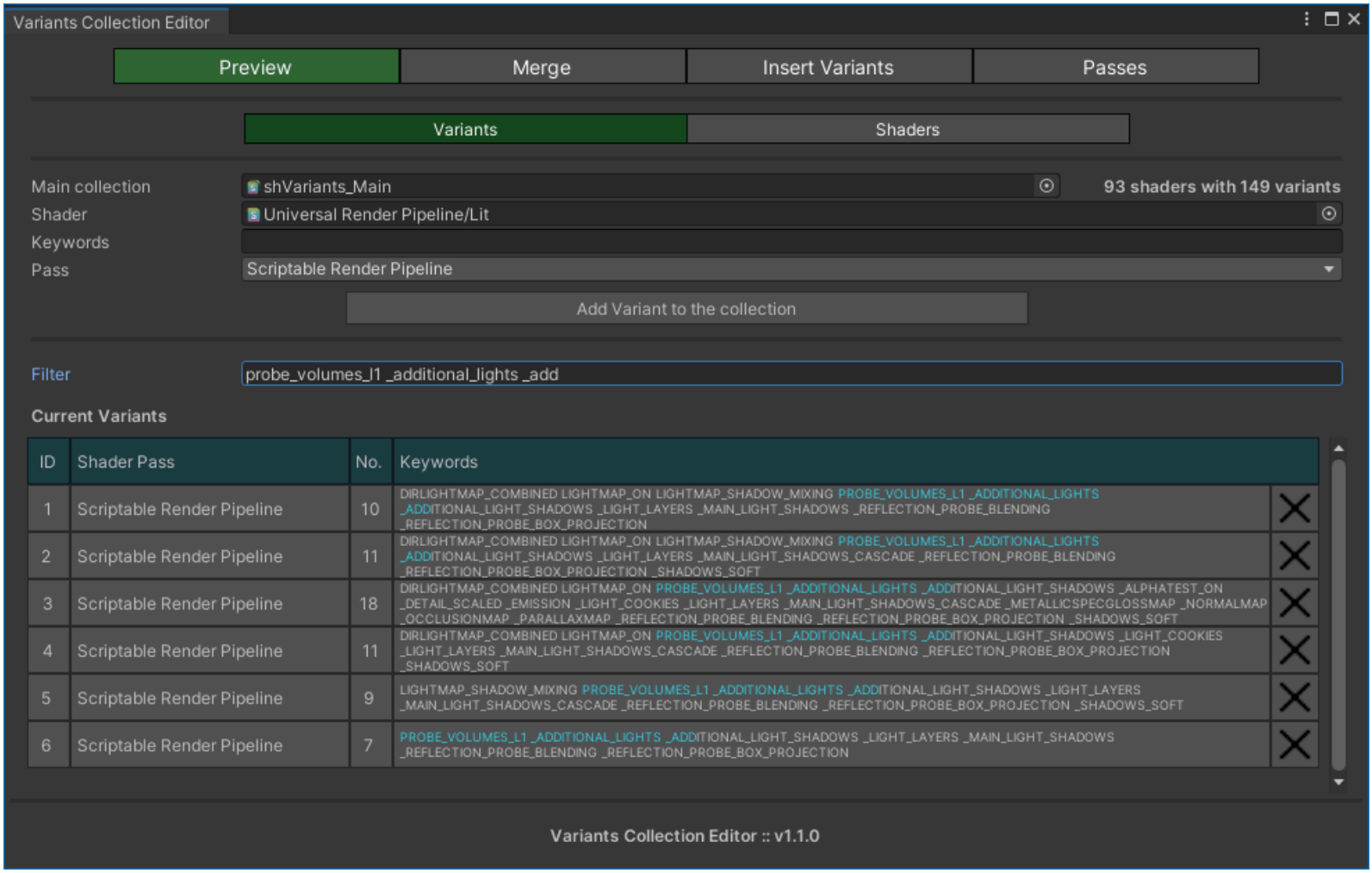
In this tab, you can preview all shader variants found in the attached variant collection based on the selected shader reference. From there, you can add a new variant to the string using the selected shader pass and pasted shader keywords. Additionally, you can remove shader variants that you think are not used anymore by your project.



If you are looking for a specific shader with a specific variant, you can use the ‘Shaders’ tab, find your shader using the filter, and click ‘Inspect’ to preview it in the ‘Variants’ tab.



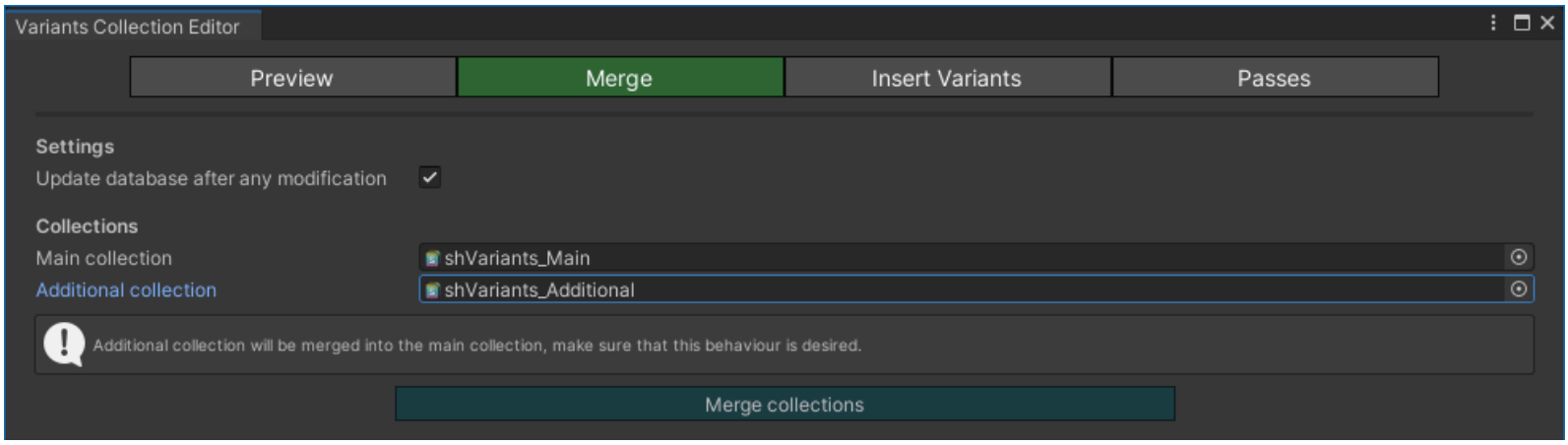
Here, you can filter furthermore for specific keywords.



Merge

In this tab, you can merge two shader variant collections together. It can be useful if you want to have one main collection with all variants attached to the Shaders Limiter database but still want to warm up different collections separately.

‘Update database after any modification’ will automatically refresh cached variants in the Shaders Limiter database. This can take some time, so if you are merging several collections at once, you can disable it until you are about to merge the last one.

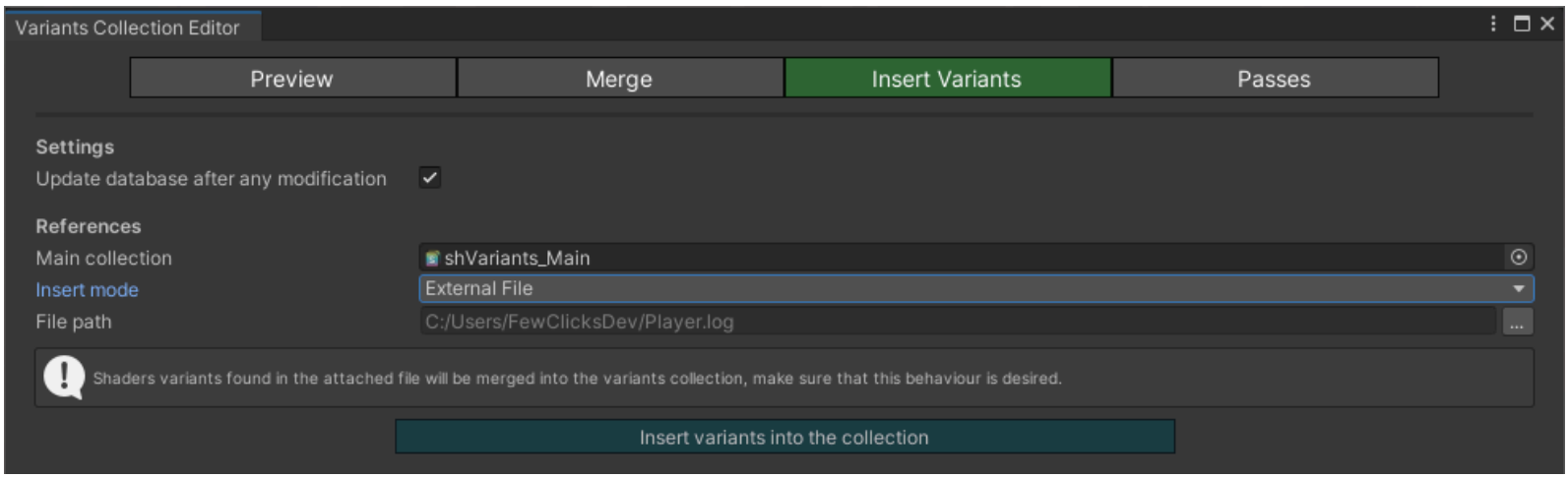


Insert Variants

In this tab, you can add new shader variants using several sources:

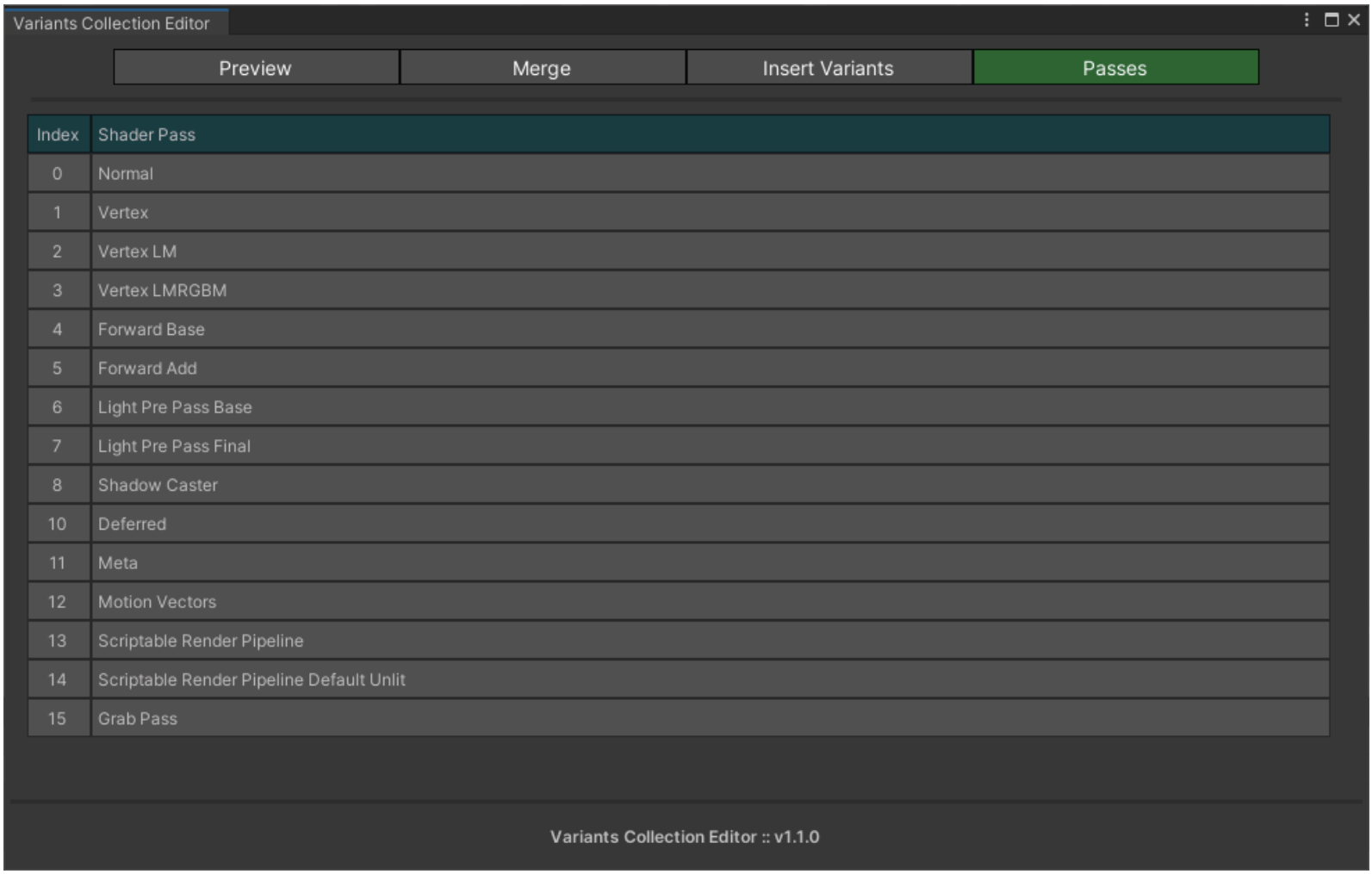
- **Log file**: file with the .log extension that is inside your assets folder.
- **Text file**: text asset with the .txt extension that is inside your assets folder.
- **External file**: text asset with the .log or .txt extension that is anywhere on your drive.
- **Pasted string**: text copied from the .log or .txt file.
- **Editor encountered variants**: variants that are currently tracked by Unity (can be found in the 'Library/ShaderCache' folder or 'ProjectSettings/Graphics' at the bottom of the page).

To collect them, you have to enable at least one option in the project settings: 'Log shader compilation' or 'Strict shader variant matching'. More about that can be found in the FAQ section of this document.



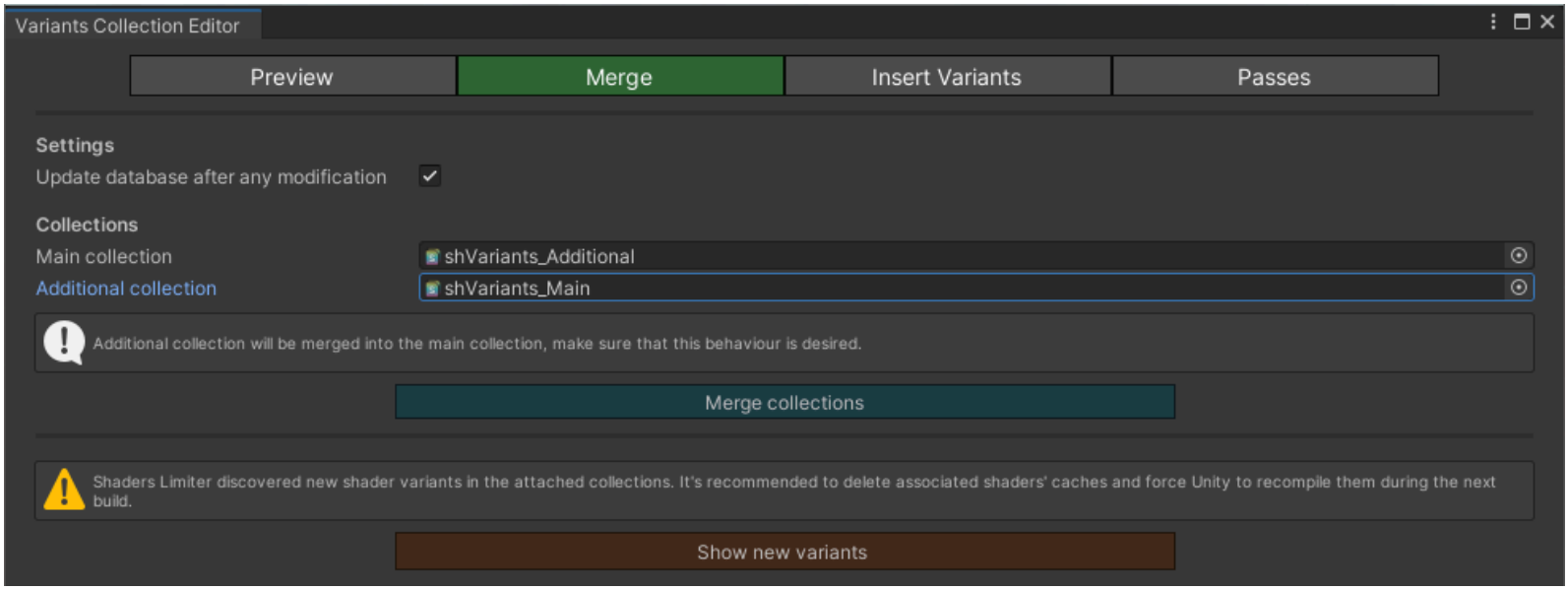
Passes

In this tab, you can preview all possible shader passes available in Unity.

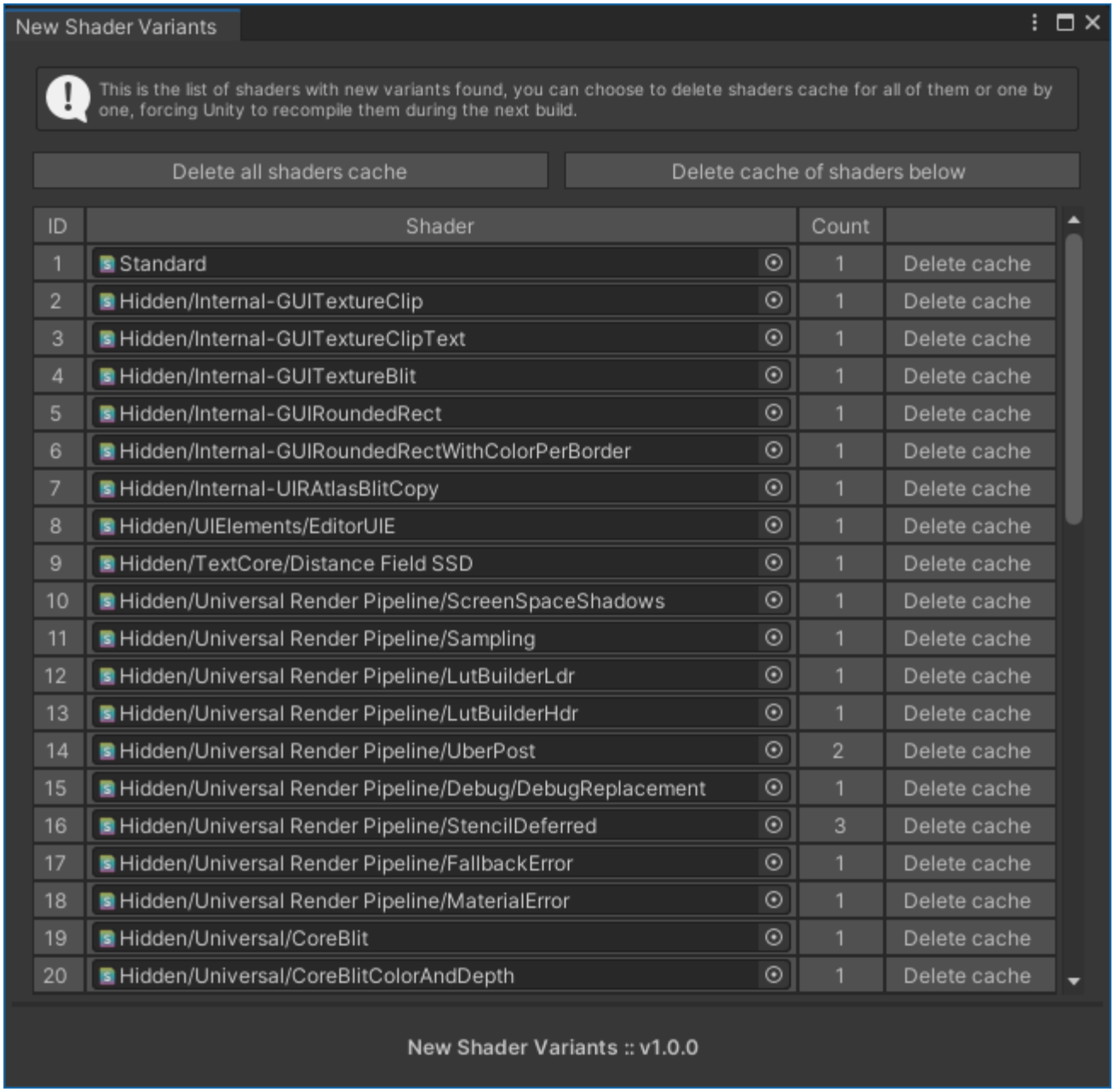


New Variants

After successfully merging two collections together or inserting new variants into one, there will be a button that will let you preview what changed during this operation.



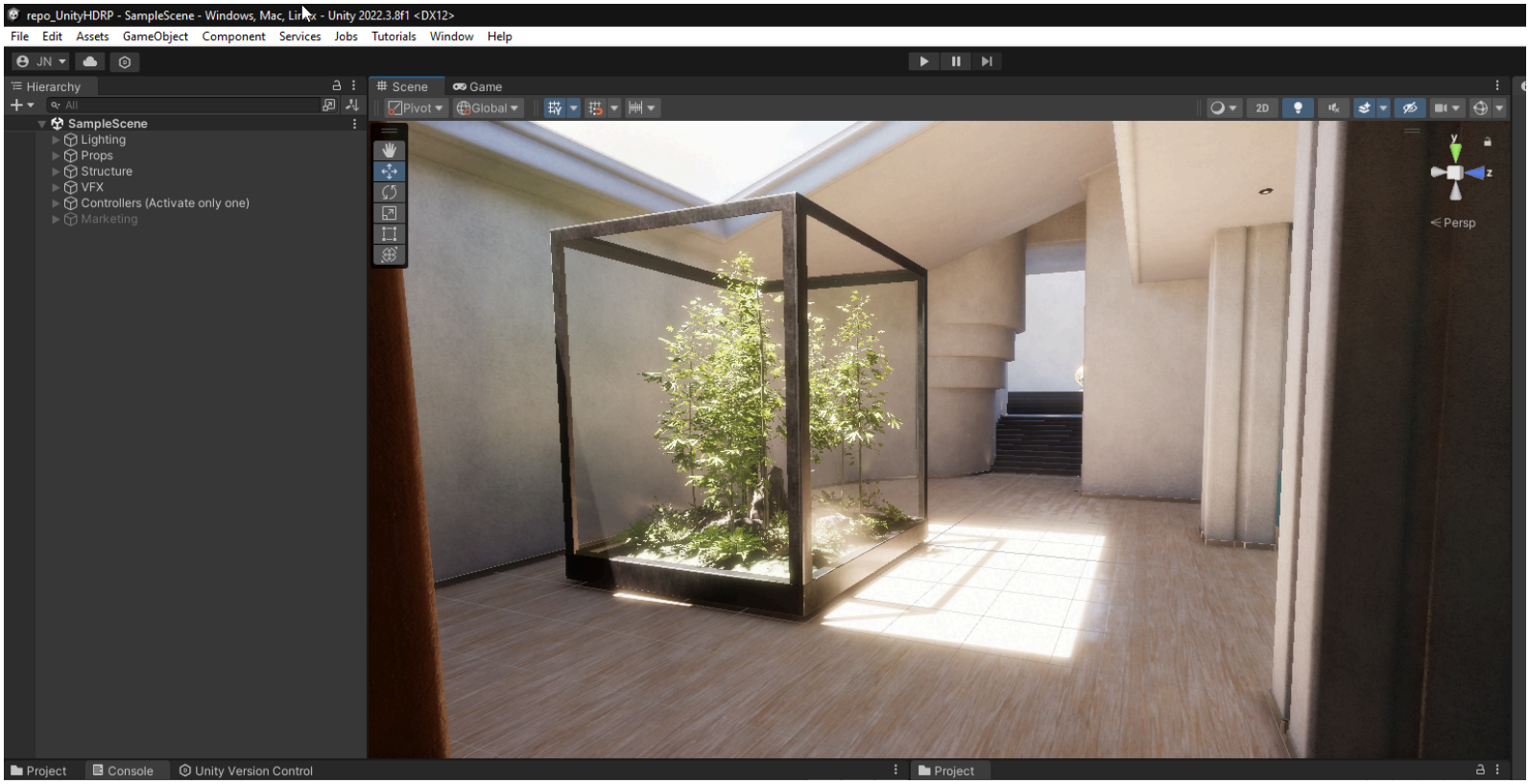
If you decide to click it, a new window will appear, showing you which shaders have been changed and how many variants have been added. Additionally, there will be an option to delete the cache of those shaders, forcing Unity to recompile them during the next build, making sure that new variants will be compiled and included in your project.



[6] Example Usage

This section will provide an example use case for this tool. All this starts with downloading an HDRP template scene using Unity Hub.

https://blog.unity.com/engine-platform/explore-learn-create-with-hdrp-scene-template

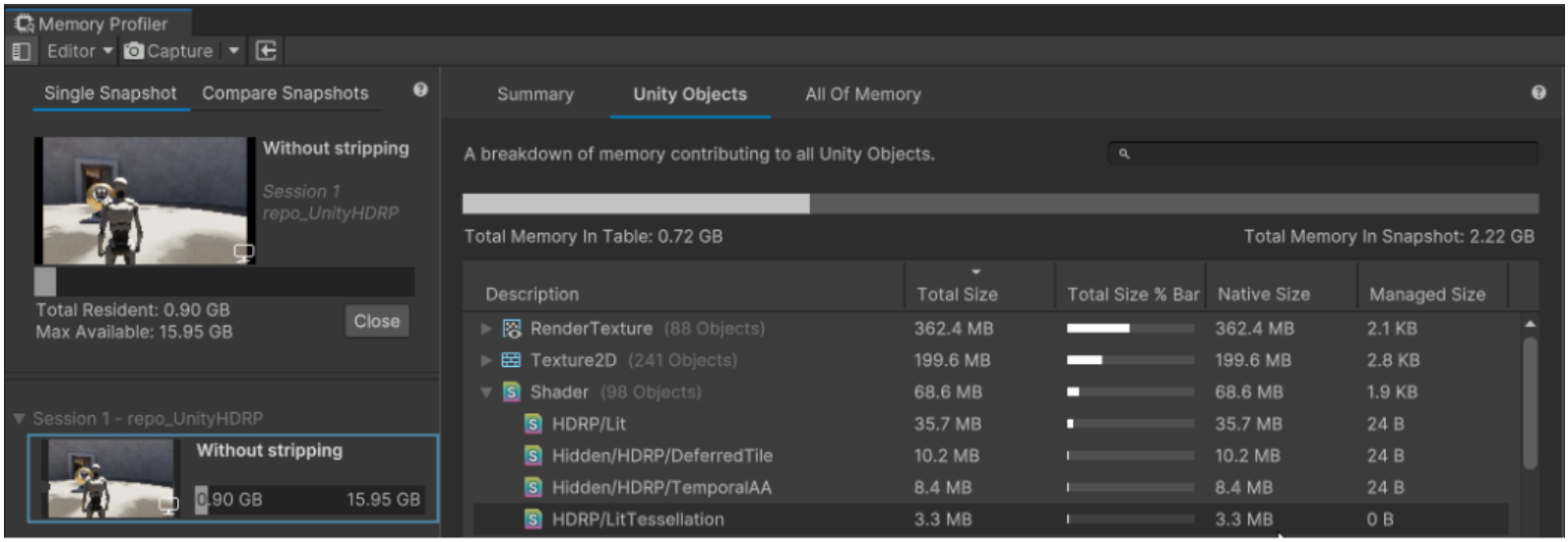


It’s a tutorial, so I would like to know what’s the starting point of our optimization. I built the game (the development build), attached a memory profiler, and took a snapshot of the memory.

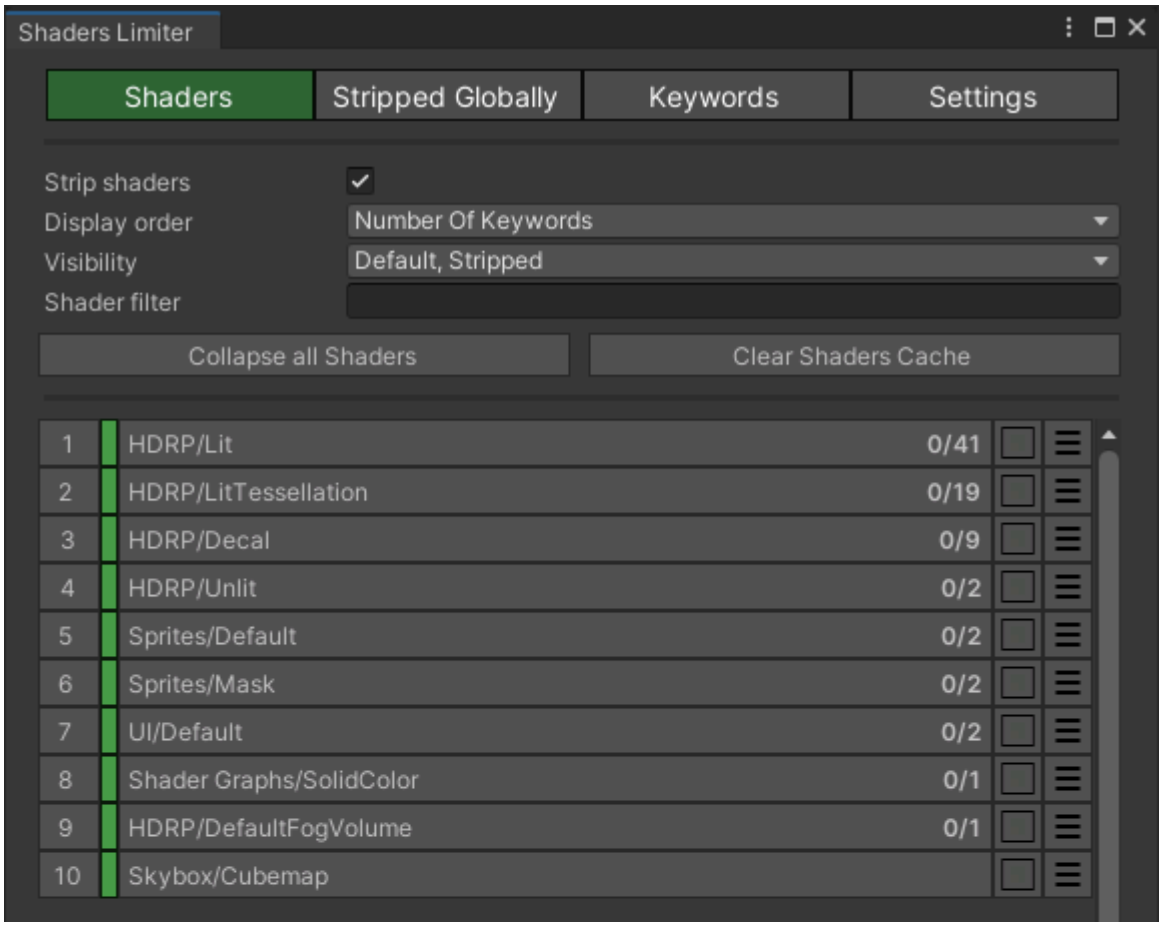
You can learn more about the Memory Profiler package here.
https://docs.unity3d.com/Packages/com.unity.memoryprofiler@1.0/manual/index.html

Please don’t profile the memory in the editor; it will give you false results.

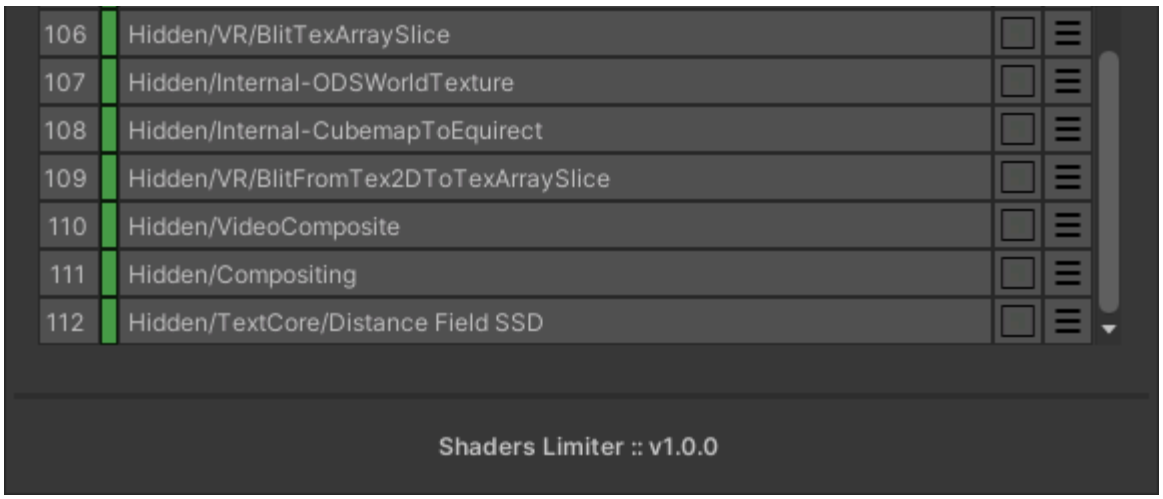
You can see the snapshot below. At the moment, shaders occupy 68.6MB of memory. We can start optimizing.



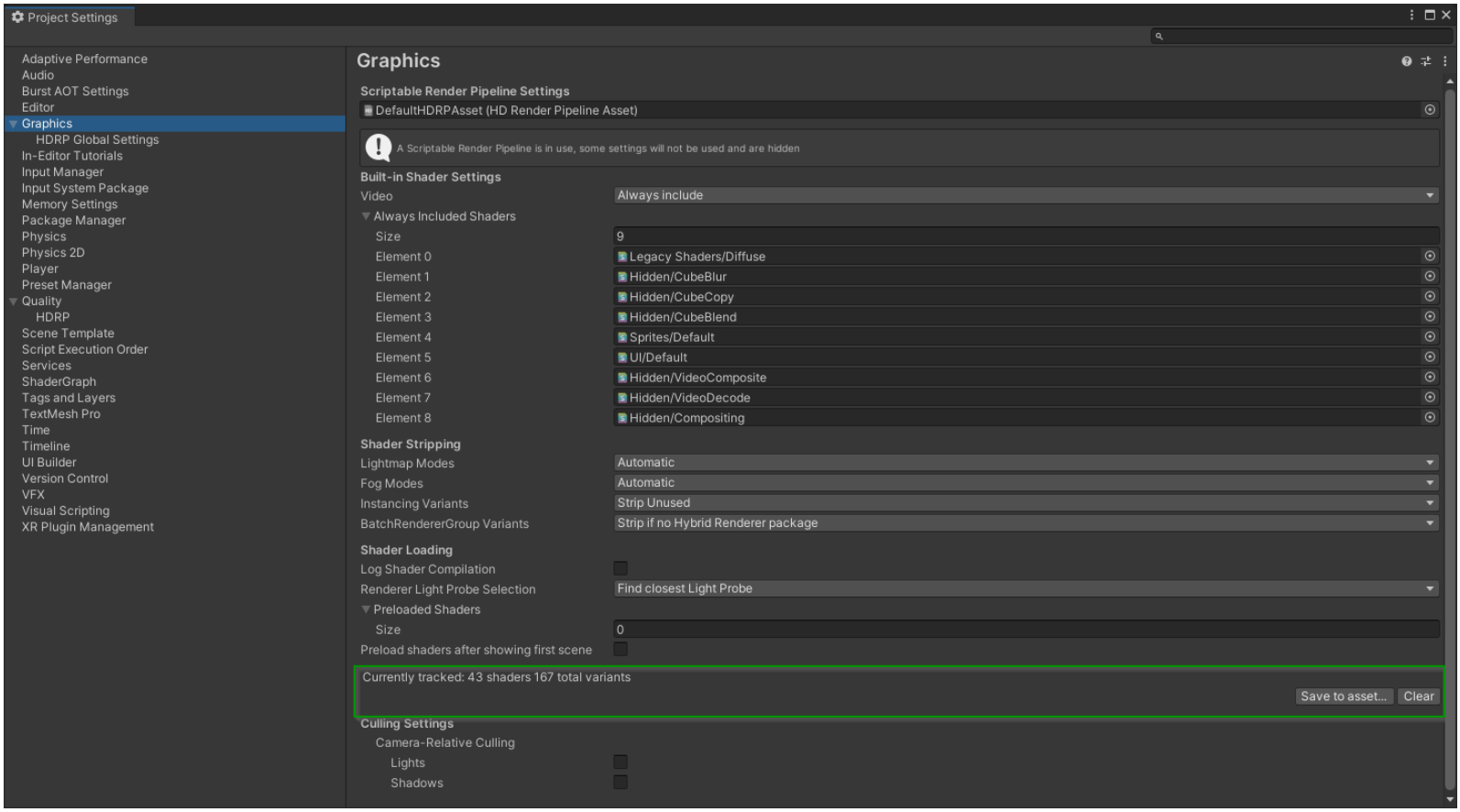
Install the Shaders Limiter, create a database, build the game again, and enable shader stripping. You should end up with a database similar to this.



Above, you can see only Default and Stripped shaders. The total number of shaders that this project is using is much larger.

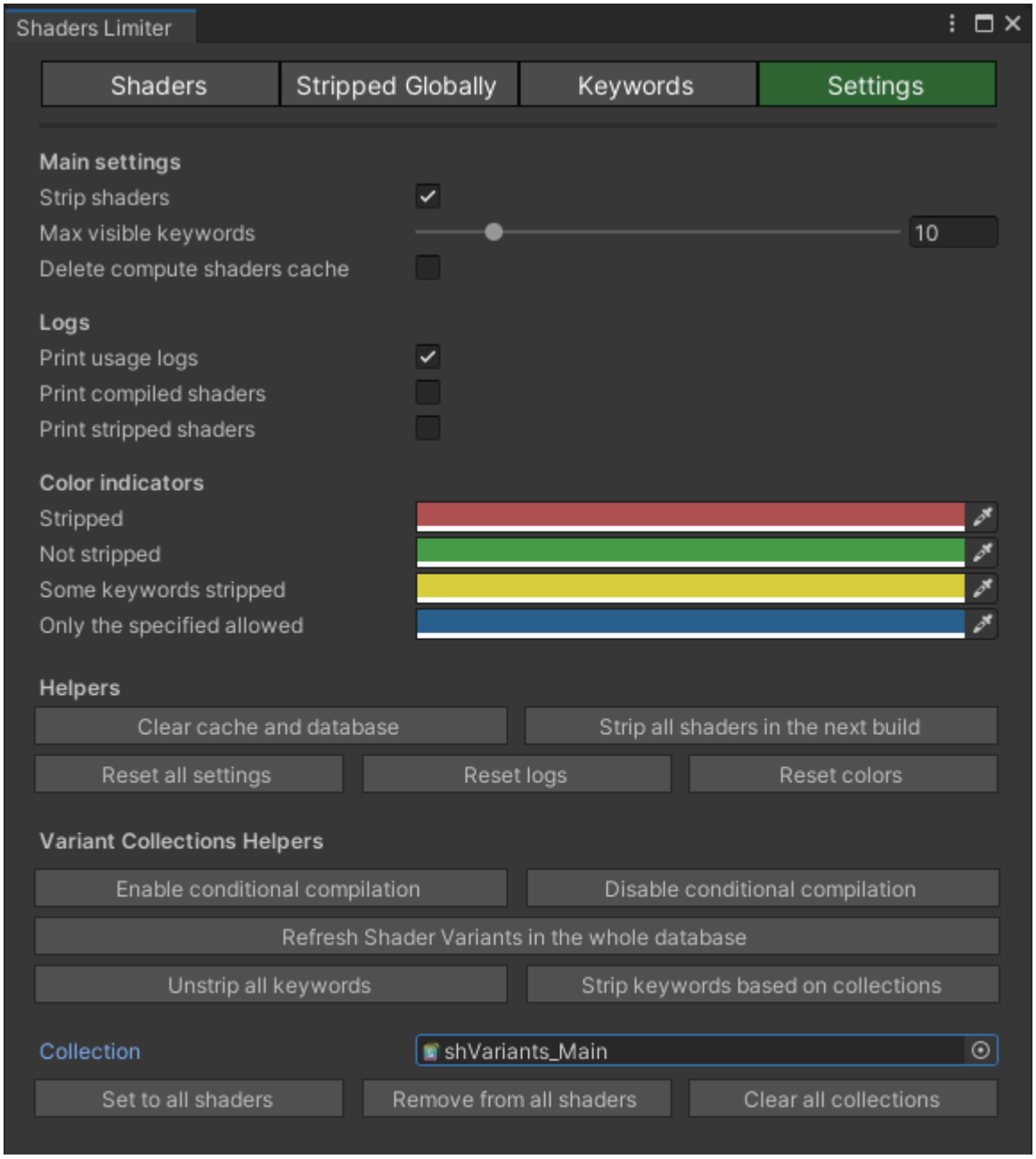


Everything is set; we can create a shader variant collection. Unity tracks all compiled shaders and saves them in the library. You can see how many shaders are tracked in Project Settings / Graphics. Save tracked shaders and variants as an asset.

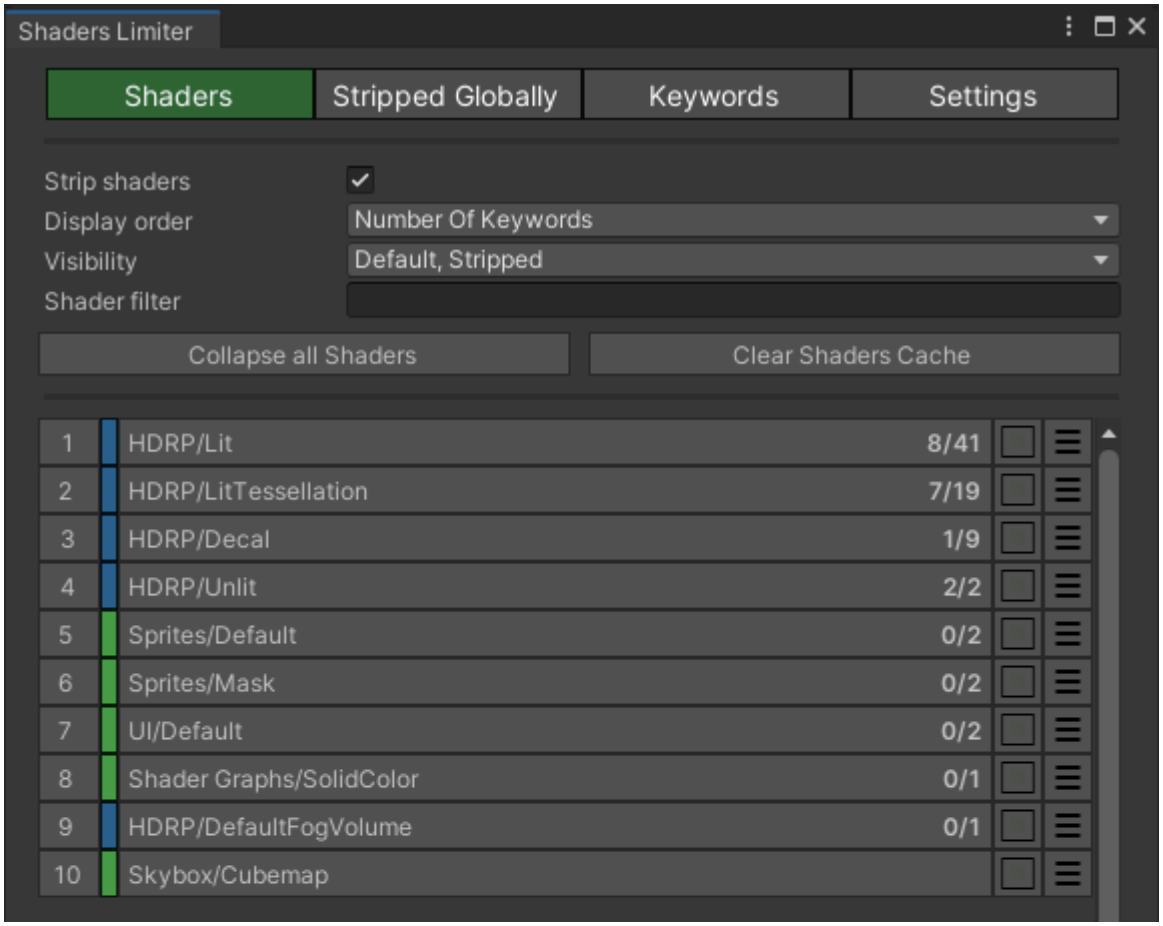


If you suspect that some shaders are not tracked by Unity, you can find that shader and ‘Reimport’ it. This should fix the issue.

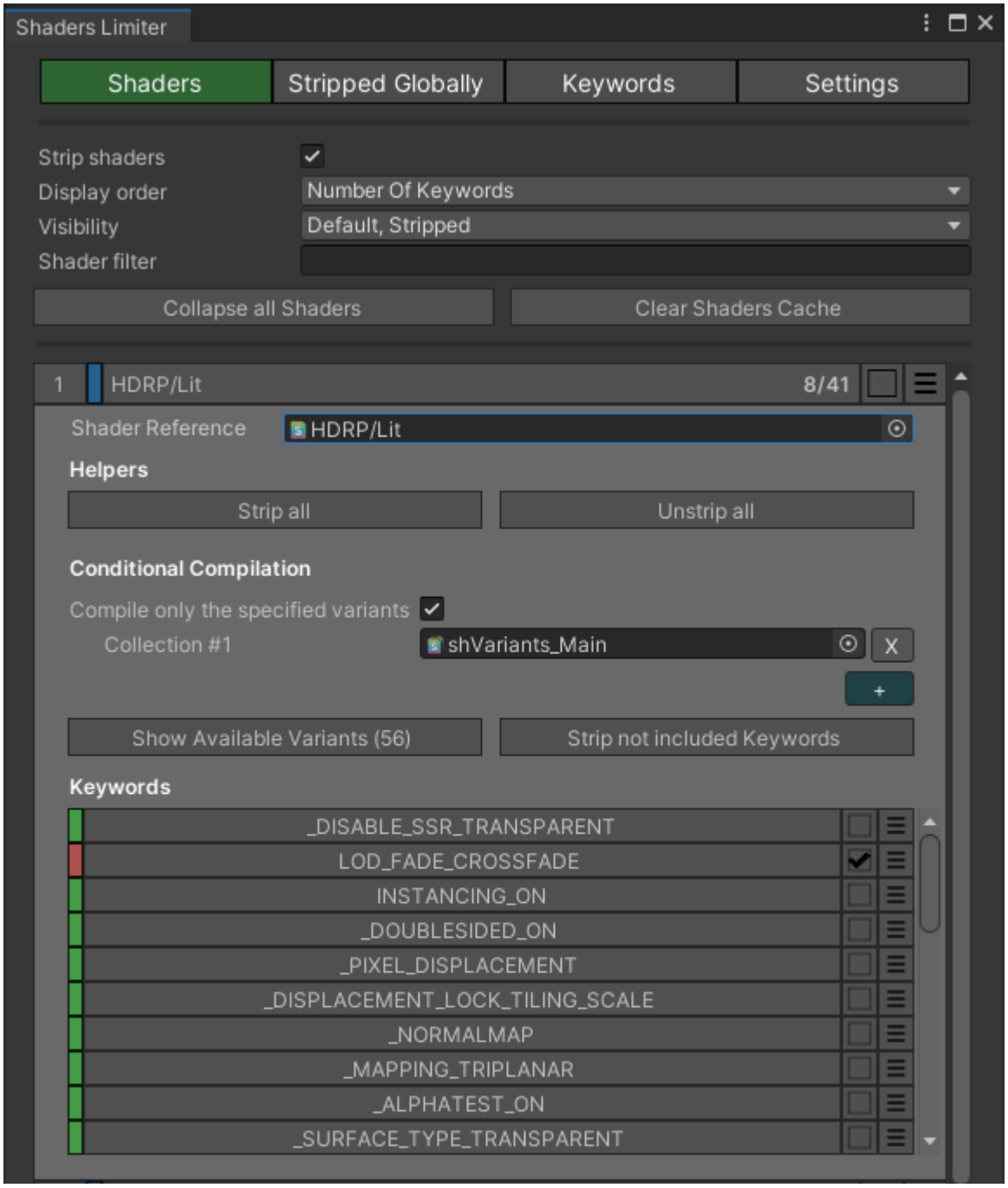
From there, we can go back to the Shaders Limiter window, Settings tab. We set a collection reference to our newly created one and press “Set to all shaders”.



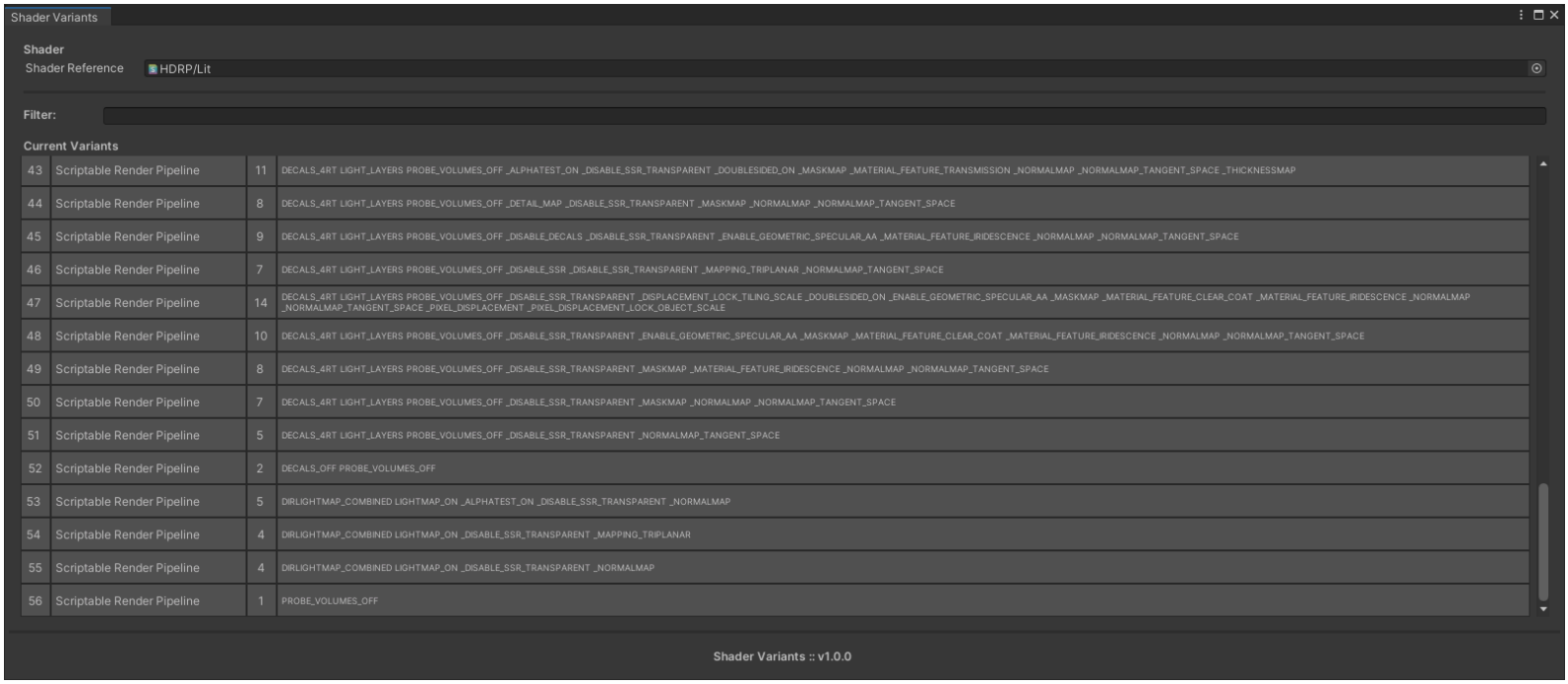
Additionally, we can press ‘Strip keywords based on collections’ to see how many keywords we don’t need. Our ‘Shaders’ tab should look like that.



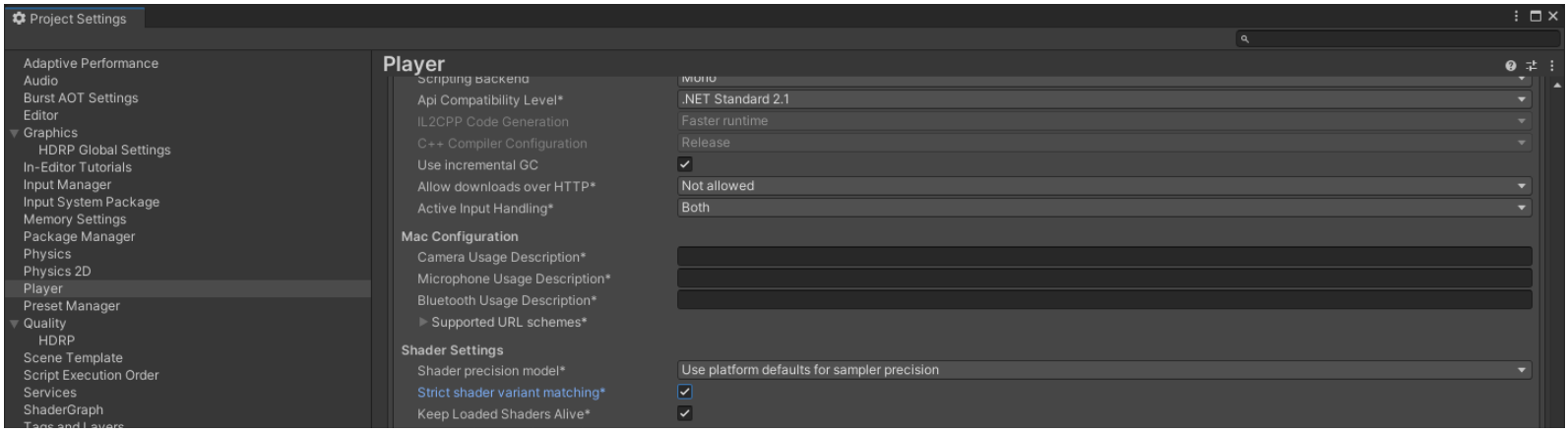
As you can see, HDRP/Lit has the most keywords; you can see its variants by going to the shader expanded view and pressing ‘Show Available Variants’.



Here, you can see that we are currently tracking 56 variants. Unity would try to compile hundreds, if not thousands, of variants without shader variant stripping.



We can go one step further and set the flag ‘Strict shader variant matching’ to true. It will force Unity to draw a magenta shader if any of the variants are not included in the build, instead of finding the closest one.

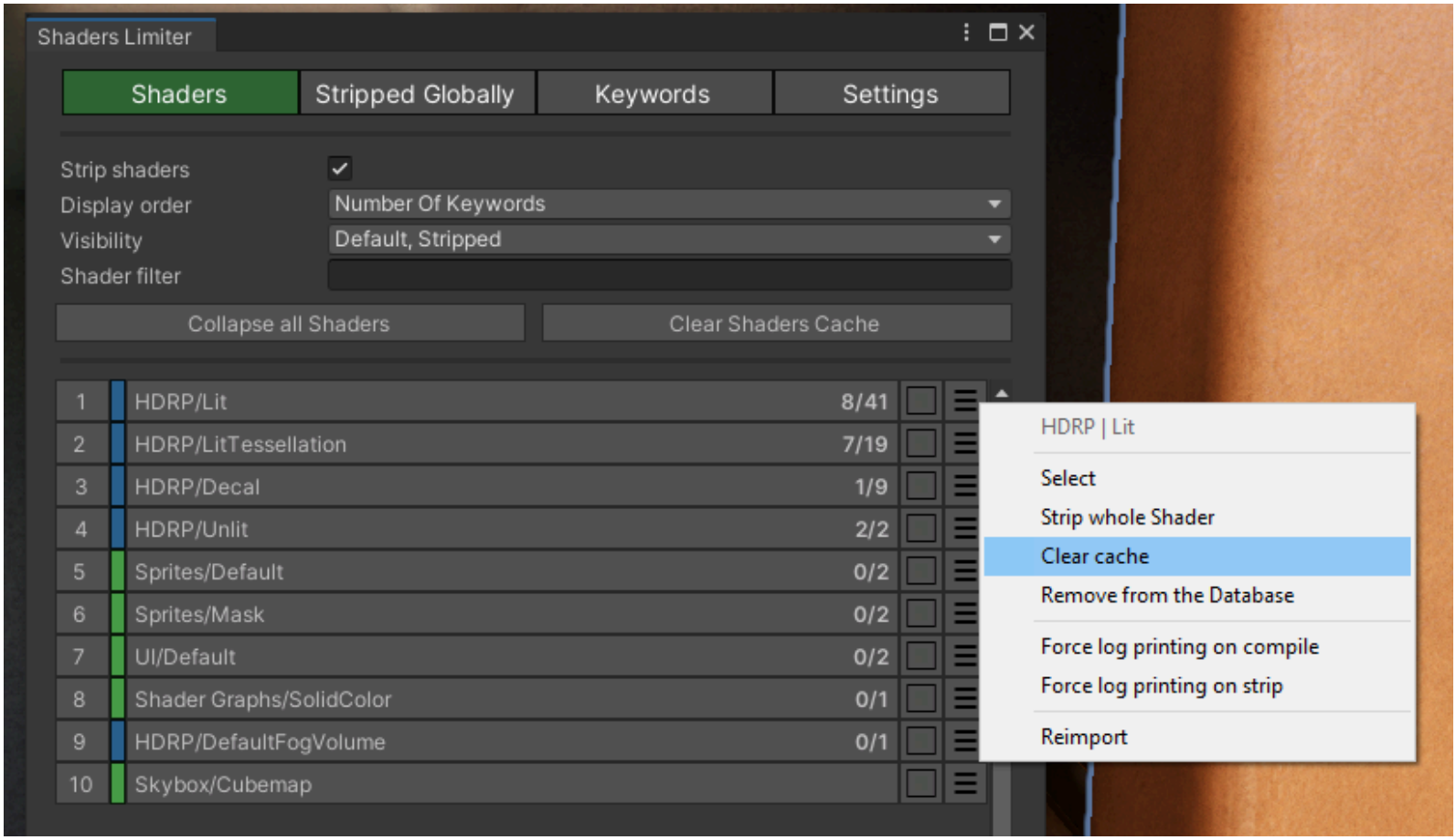


Here you can read more about this feature.

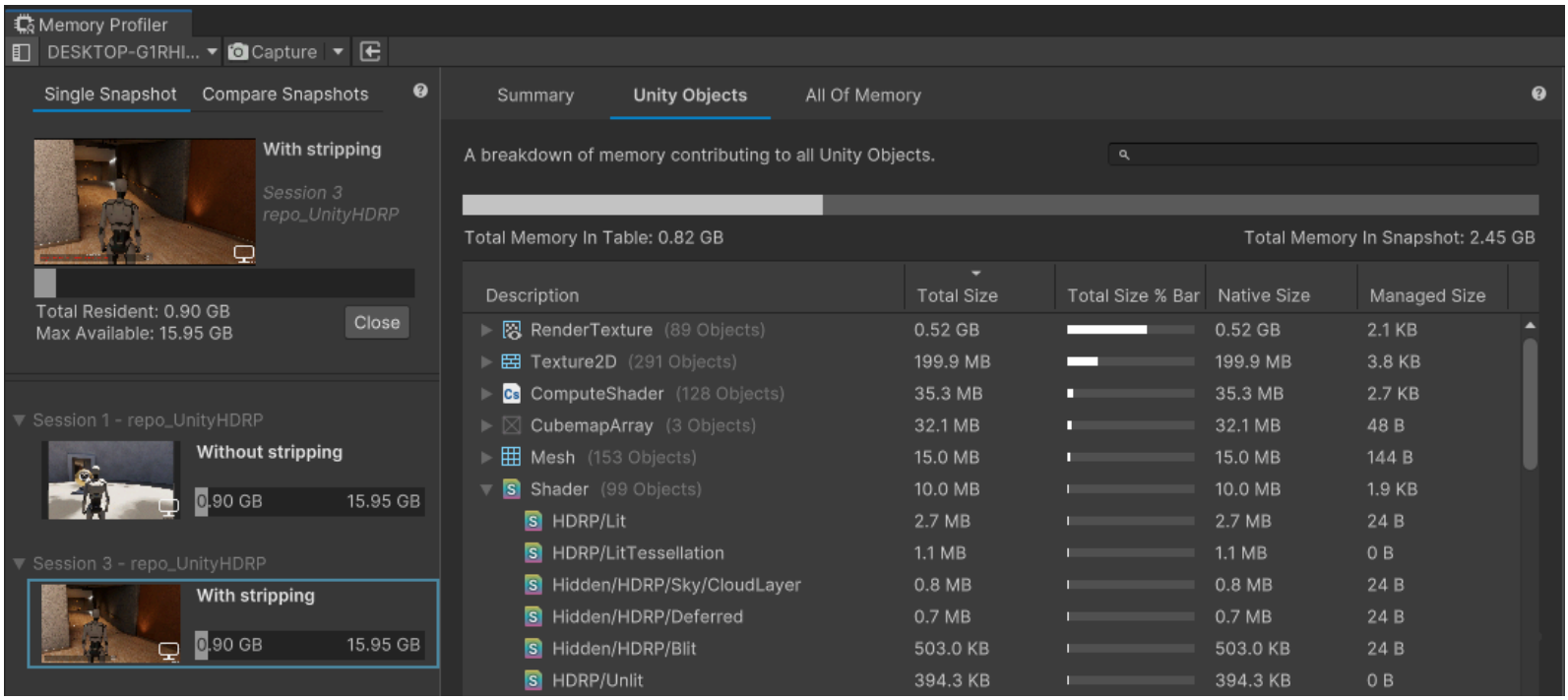
https://docs.unity3d.com/2022.1/Documentation/ScriptReference/PlayerSettings-strictShaderVariantMatching.html

It’s time to build the project again. To make sure that Unity will recompile all shaders, we can press ‘Clear Shaders Cache’. If you are doing build iterations and you are sure that only one shader has changed, you can

press ‘Clear cache’ in the options menu of the given shader instead of deleting it whole for all shaders in the project.



When a project is built, we can open the memory profiler and make another snapshot. Here are the results. As you can see, Shader's memory went from 68.6MB to 10.0MB. Results may vary based on your project specification, used rendering pipeline, shader complexity, and some other factors, but usually the more complicated the project, the bigger the difference.



If your build somehow is fullscreen magenta or some of your meshes or UI elements are pink, you can try to re-import some shaders, add them to your variants collection, and build the game again. Make sure that Unity is recompiling the shaders (delete cache, change something in the materials, etc.) because maybe you still have generated cache where everything was stripped.

[7] FAQ

What is Shader Stripping?

Shader stripping is a process during project building that excludes shader variants that won't be used anyway. It speeds up the building process and saves you some memory (build size and runtime memory usage).

Here are some of Unity's documentation links:

https://docs.unity3d.com/Manual/shader-variant-stripping.html
https://docs.unity3d.com/Packages/com.unity.render-pipelines.universal@7.1/manual/shader-stripping.html
https://docs.unity3d.com/Packages/com.unity.render-pipelines.high-definition@12.1/manual/Shader-Stripping.html

What is the Shader Variant Collection?

A shader variant collection is a list of shader variants (same shader, different keywords). You can easily create them using the button in Project Settings/Graphics Settings. They can be used to prewarm shader variants to avoid runtime compilation performance drops or to make sure that even shaders that are not included in the scenes are present in the build.

Here are some of Unity's documentation links:

https://docs.unity3d.com/Manual/shader-variant-collections.html
https://docs.unity3d.com/ScriptReference/ShaderVariantCollection.html

What to do if everything / some assets are pink?

After the first build with Shaders Limiter, your build will be pink because all shaders were stripped. It's normal behavior. The first build was meant to be fast and provide as much information as possible. Your next build should be fine; just strip shaders that you don't want to include and build your project again.

If only some of your meshes are pink, it probably means that you stripped too much. In the development build, you should see logs on the console about what's missing. You can use frame debugger to localize the mesh and then find that mesh in the editor.

If you set "Strict shader matching" in the tool settings to 'true' and some of your meshes are pink, search for missing shader variants in the console and then add that variant to your collection. You can strip the shader cache (of that shader, not whole) to make sure that Unity will recompile it during the next build.

Here are some of Unity's documentation links:

https://docs.unity3d.com/2020.1/Documentation/Manual/class-Shader.html
https://docs.unity3d.com/Manual/shader-compilation.html
https://docs.unity3d.com/2022.1/Documentation/ScriptReference/PlayerSettings-strictShaderVariantMatching.html

Are there any shaders that shouldn't be stripped?

Some third-party assets (e.g. Beautify from Kronnect) are using their own shader stripping, so even when Shaders Limiter tracks them, please don't enable any kind of stripping on them because it may break your game (something will be pink or invisible).

How can I automatically collect the shader variants needed for my project?

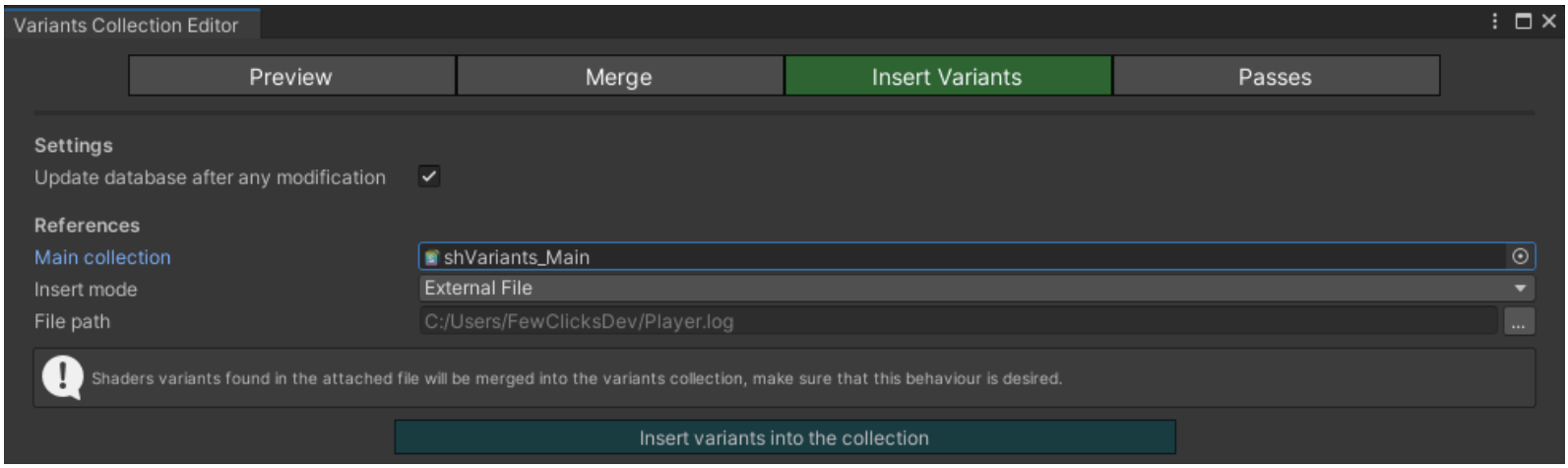
The easiest way to do it is by building your game in development mode with two options enabled. The first one forces Unity to print compiled shader variants to the console. It can be found in '*Project Settings/Graphics*' under the name '*Log Shader Compilation*'.

```
Compiled shader: Hidden/InternalClear, pass: <Unnamed Pass 7>, stage: vertex, keywords <no keywords>
Compiled shader: Hidden/InternalClear, pass: <Unnamed Pass 7>, stage: pixel, keywords <no keywords>
Compiled shader: Hidden/Internal-GUITexture, pass: <Unnamed Pass 0>, stage: vertex, keywords <no keywords>
Compiled shader: Hidden/Internal-GUITexture, pass: <Unnamed Pass 0>, stage: pixel, keywords <no keywords>
Compiled shader: Universal Render Pipeline/Lit, pass: ShadowCaster, stage: vertex, keywords <no keywords>
Compiled shader: Universal Render Pipeline/Lit, pass: ShadowCaster, stage: pixel, keywords <no keywords>
Compiled shader: Universal Render Pipeline/Lit, pass: ShadowCaster, stage: pixel, keywords _ALPHATEST_ON
Compiled shader: Sprites/Default, pass: <Unnamed Pass 0>, stage: vertex, keywords <no keywords>
Compiled shader: Sprites/Default, pass: <Unnamed Pass 0>, stage: pixel, keywords <no keywords>
Compiled shader: Hidden/BlitCopy, pass: <Unnamed Pass 0>, stage: vertex, keywords <no keywords>
Compiled shader: Hidden/BlitCopy, pass: <Unnamed Pass 0>, stage: pixel, keywords <no keywords>
```

The second option is to set '*Strict shader variant matching*' in the '*Project Settings/Player/Other Settings*'. This will force Unity to print any missing shader variant in the built project (because maybe you stripped too much). They will look something like the picture below.

```
Shader BaseShaders/Foliage/sh_sg_fl_GrassShader, subshader 0, pass 3, stage vertex: variant _USESUBSURFACE _USEWIND _WRITE_RENDERING_LAYERS not found.
Shader BaseShaders/Enviro/sh_sg_BaseUberShader, subshader 0, pass 3, stage vertex: variant _USE_LAYERS_2_LAYERS _WRITE_RENDERING_LAYERS not found.
Shader BaseShaders/Enviro/sh_sg_BaseUberShader, subshader 0, pass 3, stage vertex: variant _USE_LAYERS_BASELIT _WRITE_RENDERING_LAYERS not found.
Shader BaseShaders/Skybox/sh_SkyboxFlow, subshader 0, pass 0, stage vertex: variant _ENABLEFOG_ON not found.
```

Either way, just open 'Window/FewClicks Dev/Shader Variants Collection Editor' on the 'Insert Variants' tab, assign a variant collection, select the log file, and press the "Insert variants into the collection" button. All found variants will be added automatically, and your Shaders Limiter's database will be updated.



Here are some of Unity's documentation links:

https://docs.unity3d.com/Manual/shader-how-many-variants.html

[8] Review and Feedback

If you enjoyed using this tool please consider leaving a review on the **Unity Asset store** !
Thank you very much for any feedback.

Need some help or have an idea how to make this tool even better? Don't hesitate to write me an email at **contact@fewclicksdev.com**.

[9] Release Versions

v1.1
» Improved stripping speed and memory allocations during the build.
» Preview of all shaders found in the shader variant collection.
» Adding shader variants to the collection from game logs, pasted console string and the ones currently tracked by Unity.
» Automatic database refresh after shader variant collections merging and modifications.
» Track recently updated shader variants to delete cache and force Unity to recompile them during the next build.
» Jump between different Shaders Limiter windows using generic menus (3 dots next to the close button).

v1.0
» Initial release