NiloToon Manual

ver 0.17.2 (2025-08-05)

- ~~**Unity2020.3**~~ or below is not supported
- **Unity2021.3** is supported (will be removed in **2025** for better **RenderGraph** support)
- **Unity2022.3** is supported
- **Unity6.0~6.3** is supported, requiring **NiloToon0.17.2** or above

Online document link:

NiloToon Manual

Known issue:

- **Parallel Import** is not yet supported, enabling it will **corrupt** your .fbx generated **prefab**! (please **disable** Project Settings>Editor>**Parallel Import**, see this)
- When using **Deferred rendering** or **Forward DepthPriming**, **Opaque(Alpha)** with **ZWrite** material will not work. It is by URP's design, try switch to **Forward+** without DepthPriming (supports 256 visible lights) if it is a problem
- **Motion Vector** for **Classic Outline** is not yet supported correctly (so TAA/DLSS/motion blur/STP is not 100% correct for **Classic Outline** area), and MotionVector will not render if material enabled **Classic Outline** > **Support Dof** toggle
- **NiloToonSelfShadow**'s filtering is wrong in **Unity 6.3**

Quick links:

Table of contents

Use Cases / Gallery

To view NiloToon's render result, or NiloToon user's creation, you can visit this github website for photo preview and video links:
NiloToonURP's render result preview

Warudo + NiloToon

Warudo supports importing NiloToonURP characters via a **.warudo** file:

- **Warudo** (Warudo on Steam)(Warudo official site)

Warudo Pro

Warudo is unique in being a 3D livestream software that supports NiloToon shaders. However, this support feature is exclusive to the **Warudo Pro** version and isn't available in the Warudo free version.

Should you wish to acquire the **Warudo Pro** version, please reach out via email to Warudo at (**info@warudo.app**).

Build .warudo (NiloToon)

To build **.warudo** character file with NiloToon shader in unity, you should use:

- **Unity2021.3.18f1** (Warudo requires exactly this version)
- **NiloToon 0.16.12** (Warudo uses NiloToon 0.16.12, if you built a .warudo using an older NiloToon version, you need to build again using NiloToon 0.16.12 for Warudo to render your NiloToon character correctly)

To help you build your character to a .warudo faster and easier,
you can download our special version of Warudo's SDK project,
which is the regular Warudo SDK project + URP & NiloToon setup already:

- Download the project here -> **For Warudo(NiloToon0.16.12)**

Warudo chara mod doc

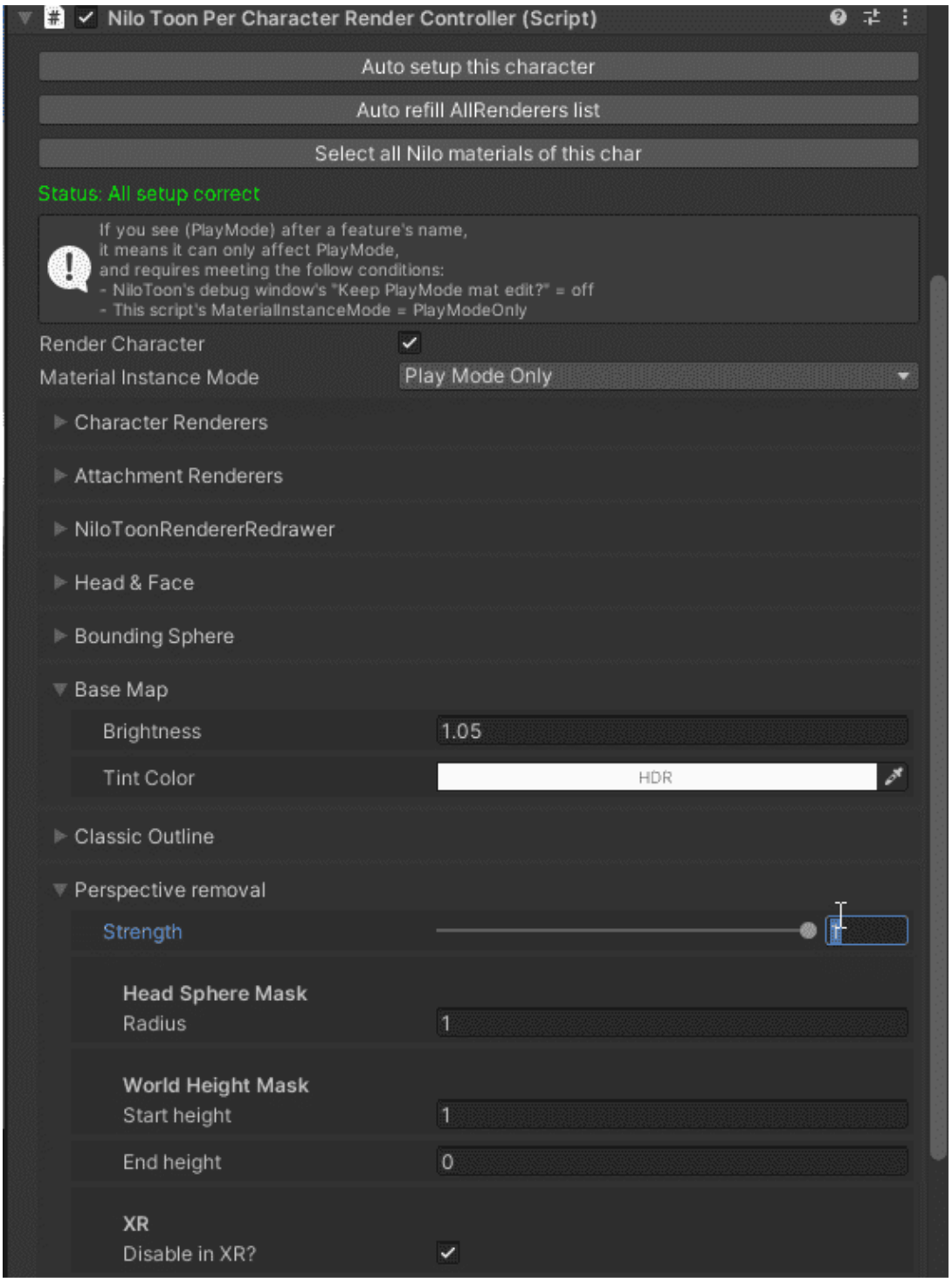
Here are the documents about building character mod to Warudo:

- **Overview | Warudo Handbook**
- **Unity & Warudo SDK Installation (For Step3 Option1: Download this instead -> For Warudo only(NiloToon0.16.12))**
- **Creating Your First Mod | Warudo Handbook**
- **Character Mod | Warudo Handbook**

NiloToon chara settings

Before building the .warudo file, you may adjust a few properties of NiloToonPerCharacterRenderController on your character prefab:

- **Perspective removal = 1**, this will make the .warudo file look flat in any camera fov inside Warudo, usually it will produce a better result for a regular Warudo + obs overlay streaming use case
- **Base Map > Brightness**, you may increase it a bit(e.g., 1.05) if you find that the default rendering inside Warudo is too dark. Keep it as 1 if the default rendering color inside Warudo looks correct.



NiloToon shader stripping

[Important note]

If you see features like:

- **dissolve**
- **dither**
- **screen space outline**
- **URP shadow**

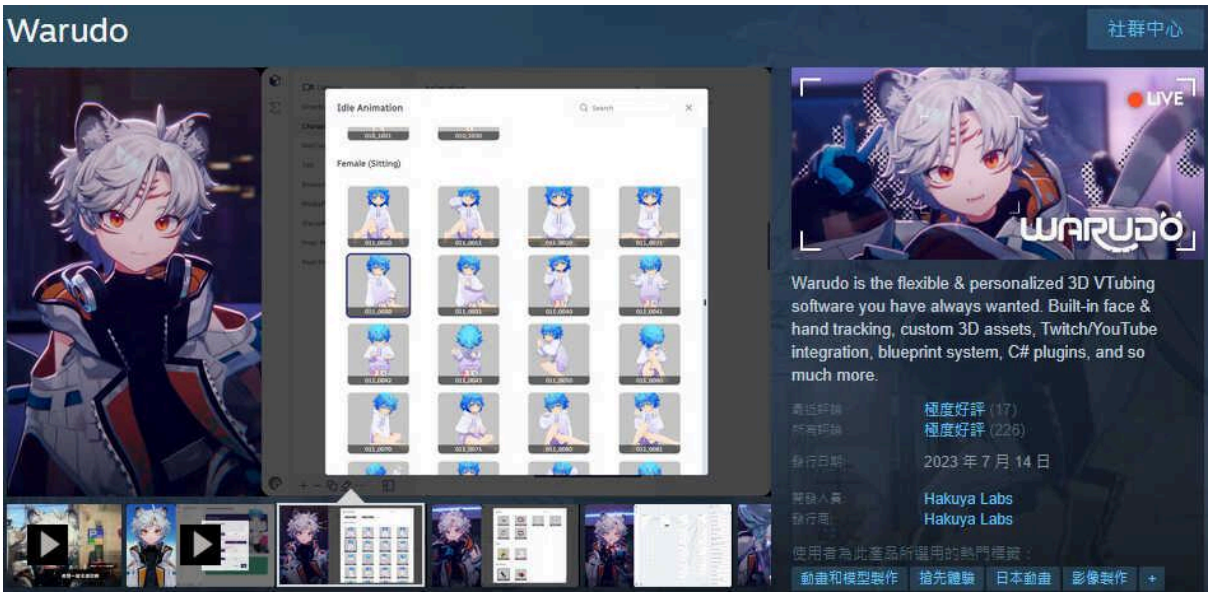
in the Unity editor correctly, but failed to see them after building .warudo file and importing to Warudo, very likely it is due to NiloToon shader stripping settings when you build the .warudo file.

Following this to solve the problem.

Mocap tools + NiloToon

Here is a list of motion capture tools on steam that supports NiloToon.

Warudo Pro



- Warudo on Steam
- Warudo official site

MELIGO Creator

- MELIGO on Steam
- MELIGO official site

*If you developed a software that supports NiloToon, you can notify us to put your software info here.

Download demo builds

If you just want to play NiloToon's demo build on your device without using UnityEditor, you can download the demo build via these links:

- **PC** .exe demo (2021.3LTS build) for your PC (**maximum quality setting**):
  Download NiloToonURP demo build (PC)

- **Android** .apk demo (2021.3LTS build) for your android phone (**balanced quality setting**):
  Download NiloToonURP demo build (Android)

- **Quest2/3** .apk demo (2022.3LTS build) for your quest 2/3 (**fastest quality setting**):
  Download NiloToonURP demo build (Quest2/3)

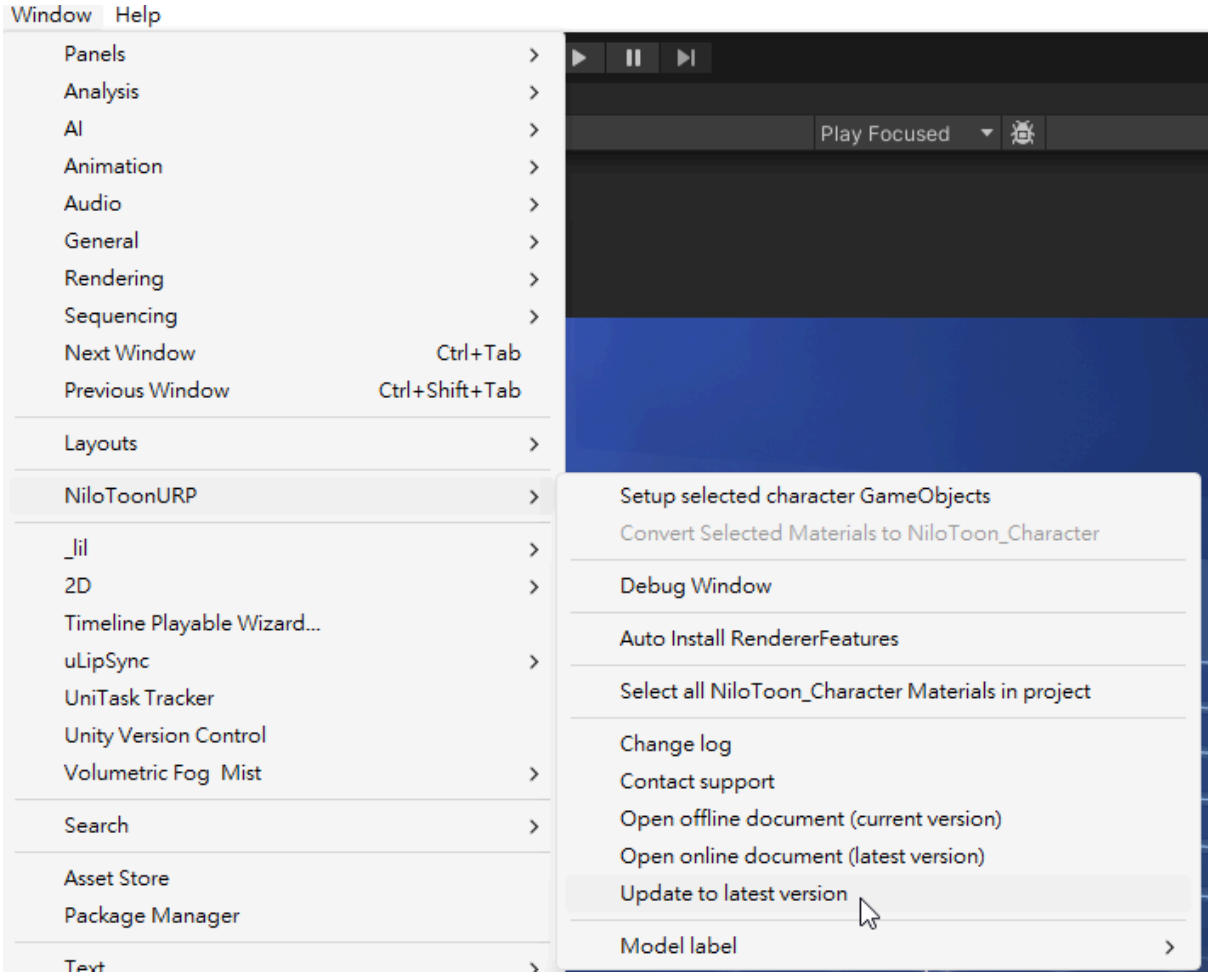
Download package/zip

To download NiloToon's **.unitypackage** for importing to your projects, or if you want a complete NiloToon demo project source(.**zip**), you can:

A) download them using the google drive link:
[Download the latest NiloToonURP(via google drive)]

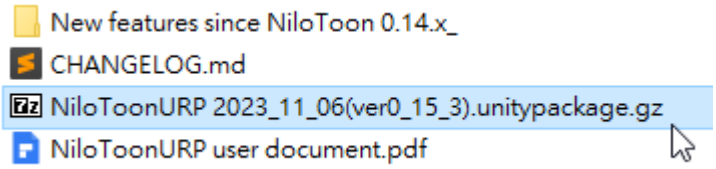
or

B) click this **Update to latest version** button (it is the same as doing the option A above)

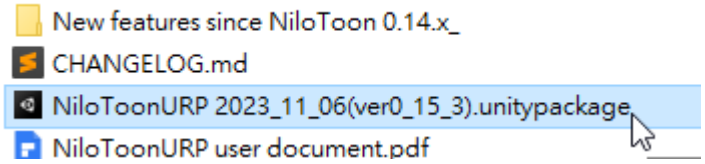


You need NiloToon's google drive permission to access the above google drive link. If you forget the email address of your purchased NiloToonURP's download permission, contact us.

fixing .unitypackage.gz
We highly recommend downloading each required file(.zip / .unitypackage) separately!
If you download a **folder** instead of a **file** via google drive, google drive will compress everything inside the folder and convert the downloaded **.unitypackage** as **.unitypackage.gz**



For windows, to extract the **.unitypackage**, don't decompress the **.gz** file again, just **remove the .gz extension** by simply **renaming** the file (select the file > press F2 > remove .gz > hit enter).



For example, you downloaded:

- NiloToonURP 2023_11_06(ver0_15_3)**.unitypackage.gz**

You should **rename** the file to:

- NiloToonURP 2023_11_06(ver0_15_3)**.unitypackage**

then you can use the **.unitypackage** as usual, for example, import it into your Unity Editor.

*Do not try to download the whole **NiloToonURP all versions** root folder, it is too big and google will download many separated zip(s), which may miss files or be difficult to download everything correctly.

Play demo project in Editor

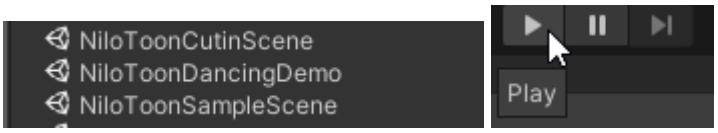
If you're new to Unity, NiloToon or URP, we recommend downloading the latest NiloToonURP demo project before integrating NiloToon into your actual Unity project. The demo project provides examples that you can use as a learning reference and a playground project for experimentation. This demo project will help you understand and effectively use NiloToon in the future. To get this demo project:

1) Download **NiloToonURP demo project [date & version].zip**, you can download the complete project .zip file inside the following google drive folder: Download the latest NiloToonURP(via google drive)

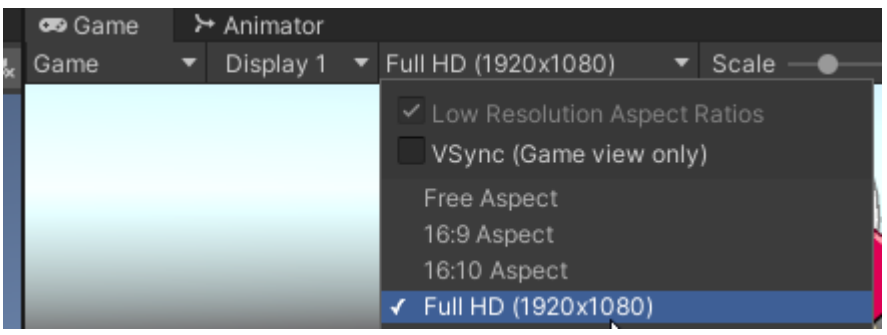
2) Unzip the project zip to a folder, picking a **fast SSD drive** is highly recommended (after unzip, your folder should contains **Assets,Packages,ProjectSettings** folders)

3) Start UnityHub, click Project > Add > Add project from disk, select the folder that you just unzip which contains **Assets,Packages,ProjectSettings,** open the project using the **latest Unity 2022.3LTS / Unity6.0-6.2**. Click **Yes** for all pop up windows. The first time opening the project may take **hours** to reimport assets if you **didn't pick a fast SSD** for this project.

4) After re-import finished, open and play any NiloToon scene (you can search all NiloToon scenes in the **project window** using **t:scene** in the **search bar**)

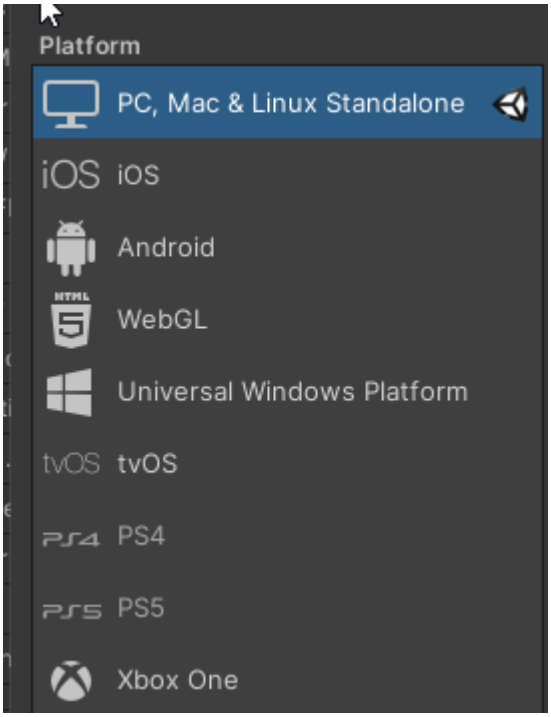


5) You will now see the same result of the downloadable demos inside your editor's game window (you need to use **1920x1080** or above game window size) .



6) (Optional) build the project to your target platform to test (it will compile for **a few hours** if you are building to that platform the first time due to shader

compilation, then the next time you build it again it will be much faster, usually 5~10 minutes). Build speed mainly depends on SSD speed,CPU speed



- For **PC** platform, you can simply build the project without modification (e.g., just press "**Ctrl+B**" in editor after you open the project)
- For **Android/iOS** platform, see **How to build the demo project for Android/iOS?**
- For **Web** platform, see **How to build the demo project for Web?**

7) (Optional) If there are problems/bugs in your target platform's build, always contact us, we will help you to solve it (**Bug report**).

Install package & setup

To use NiloToon's .unitypackage in your existing or new project, follow these steps:

0a.Convert to URP

[Important step]

Before importing NiloToonURP's .unitypackage, you need to ensure your Unity project is using **URP**. See this to understand how to convert a Built-in RP project to an URP project, else you will see **pink error shader** in material and NiloToon will **not compile**.

0b.Backup project

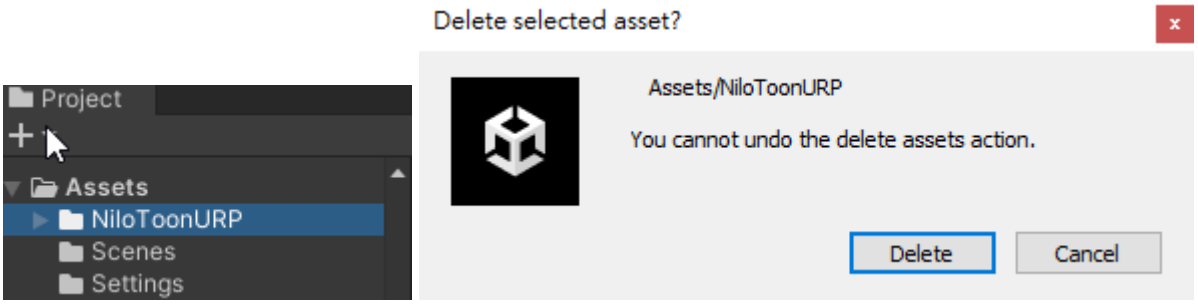
(optional) Before any update, back up your project (commit using any version control tools like **GitHub Desktop**/**Plastic SCM**, or just make a copy of your project). In case something bad happens, you can still perform a safe recovery.

1a.Delete old NiloToon

(optional) If you want to avoid all update problems:

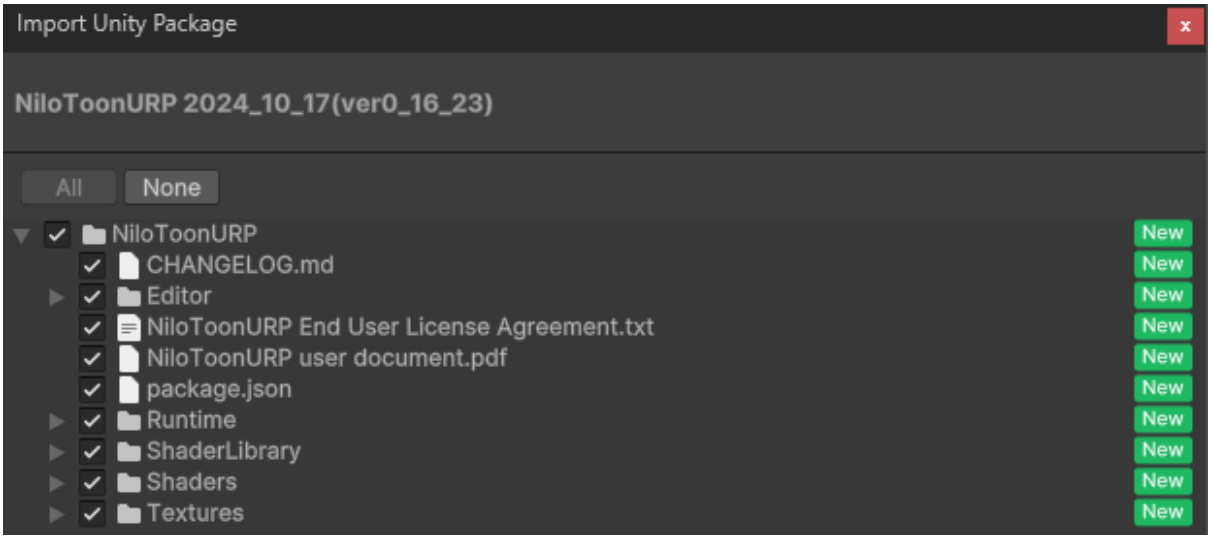
1. open an empty scene in Unity Editor
2. **delete** your project's old **NiloToonURP** folder for a clean reinstall in next step

However, if you are familiar with the update process, deletion of the NiloToonURP folder may not be necessary. Sometimes NiloToon may change internal folder directory, when that happened, deleting the old **NiloToonURP** folder is required



1b.Import .unitypackage

[Important step] Import NiloToonURP_[version].unitypackage



1c.Auto dependency

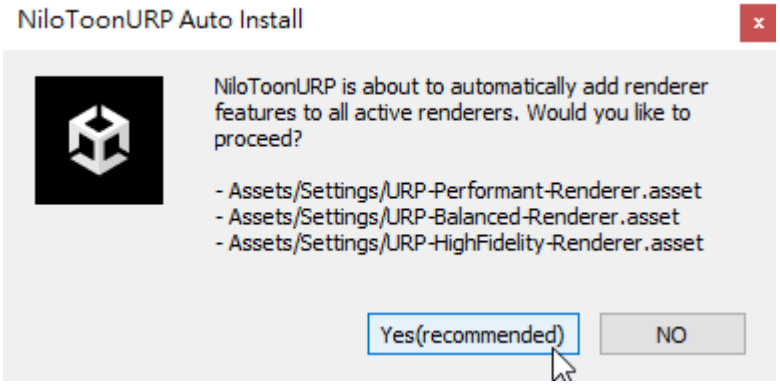
(no action needed) NiloToon will install **Collections**(1.2.4 or higher) in the package manager automatically, you don't need to open Package Manager.

1d.Auto install window

[Important step]

A **NiloToonURP Auto install** window may pop up, if you don't want to manually install NiloToon's renderer feature, we highly recommend clicking **Yes** to let NiloToon install all required NiloToon renderer features for your project's active renderer.

(in case you want to do it manually later, click No)



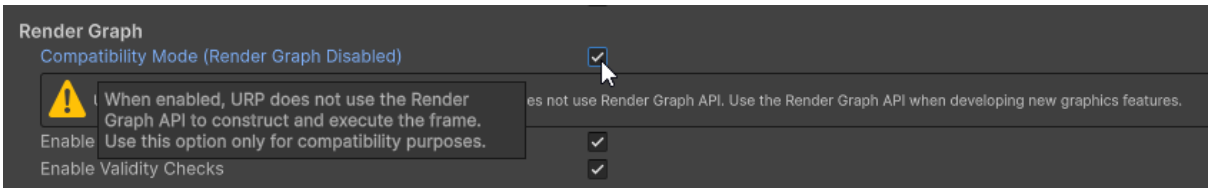
1e.Render Graph

If you enabled Render Graph, NiloToon will still work correctly for almost all functions, but the following features may not work correctly:

- NiloToon Anime PostProcess (in Forward/Forward+ mode)

To show these NiloToon features correctly, you can enable:

Project Settings>Graphics>**Compatibility Mode = on**



In **Unity6.3**, Compatibility Mode will be removed by URP, so we will recommend you try to turn Compatibility mode **off** if possible for any projects that you plan to upgrade to Unity6.3 in the future.

1f.Preserve config in hlsl

In most cases this step is not required, so you can safely skip this step unless you are using VertExmotion in NiloToon.

(optional) If you need VertExmotion to work with NiloToon, open NiloToonCharacter_ExtendDefinesForExternalAsset.hlsl, re-enable the settings if you need them.

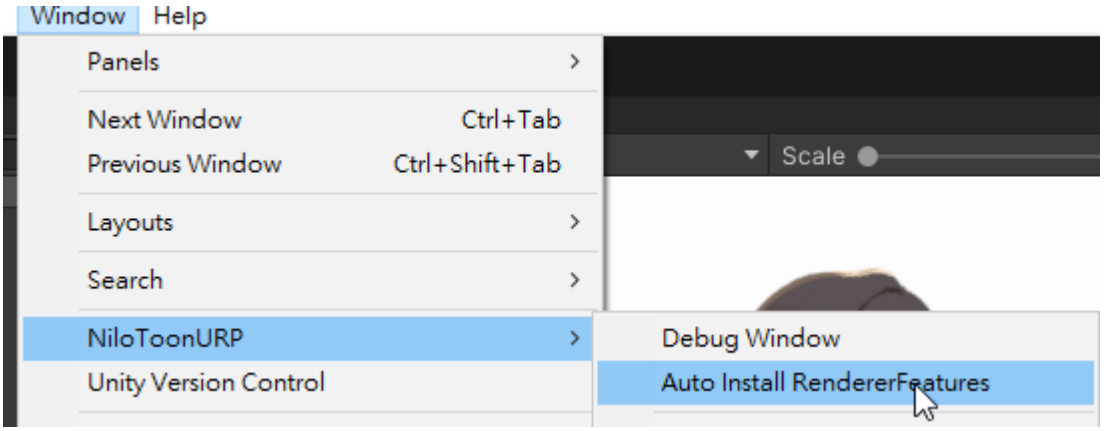
1g.Restart Unity

[Important step]

Install & update of NiloToon finished, you need to **restart the editor** to ensure NiloToon update correctness.

2a.Retrigger Auto install

(Only if you clicked **NO** in step(1d))

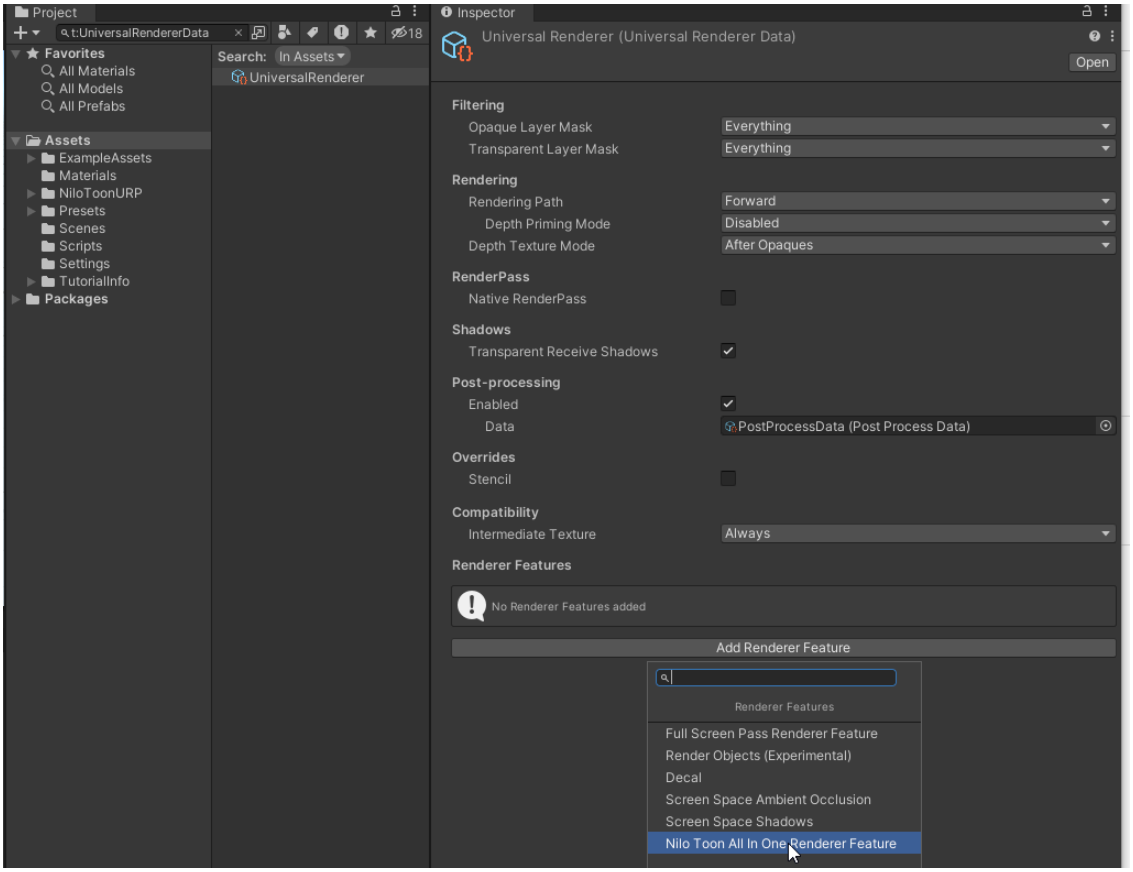


You can always click: Window > NiloToonURP > **Auto Install RendererFeatures** button to let NiloToon add all renderer features automatically for you again

2b.+NiloRendererFeature

(If you skipped both step(1d) and (2a))

If you don't want NiloToon to auto install renderer features for you, you can do it manually. Select your active URP **3D** Renderers (search t:UniversalRendererData in your project window), manually add a NiloToonAllInOneRendererFeature to it.



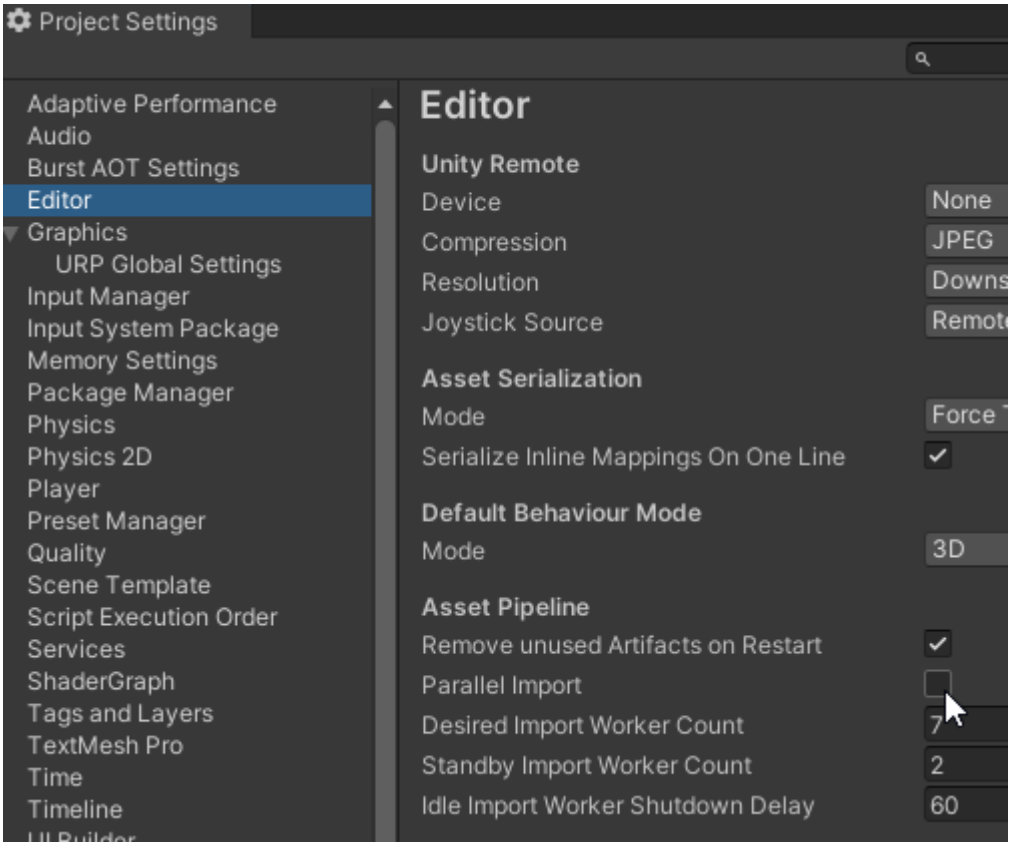
*You should also add NiloToonAllInOneRendererFeature to other active renderers if needed, for example, other renderers for each quality settings in Project Settings > Quality

3a.Parallel Import = Off

[Important Step]

Set Project Settings > Editor > Asset Pipeline > **Parallel Import** = **false**

else your .fbx generated **prefab** may become **invalid** since NiloToon requires single thread(non-parallel) fbx import



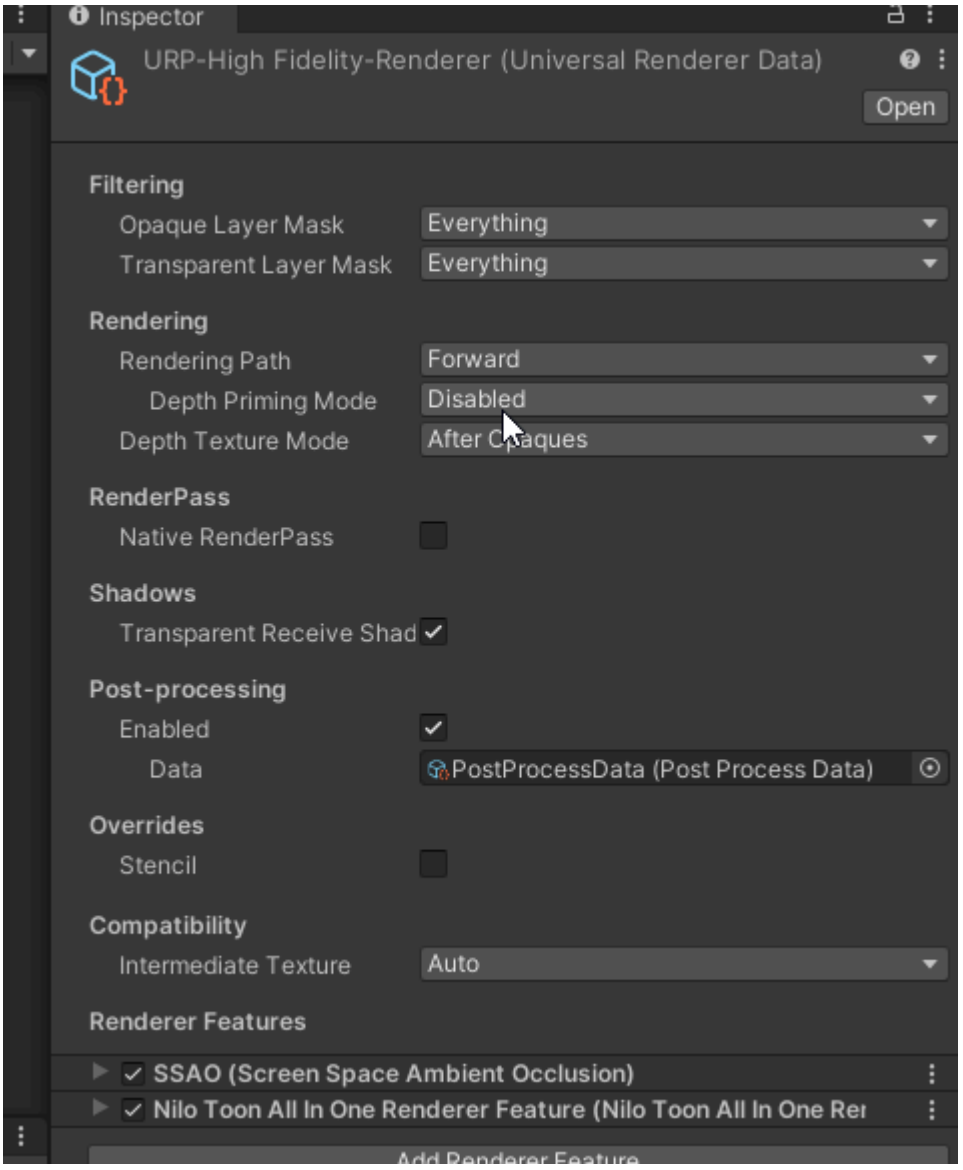
*For the reason why it is required, and solution to fix the problem, see this

3b.Depth Priming = Off

[Important Step]

Select all your URP Renderers (search t:UniversalRendererData in your project window) that uses NiloToonAllInOneRendererFeature:

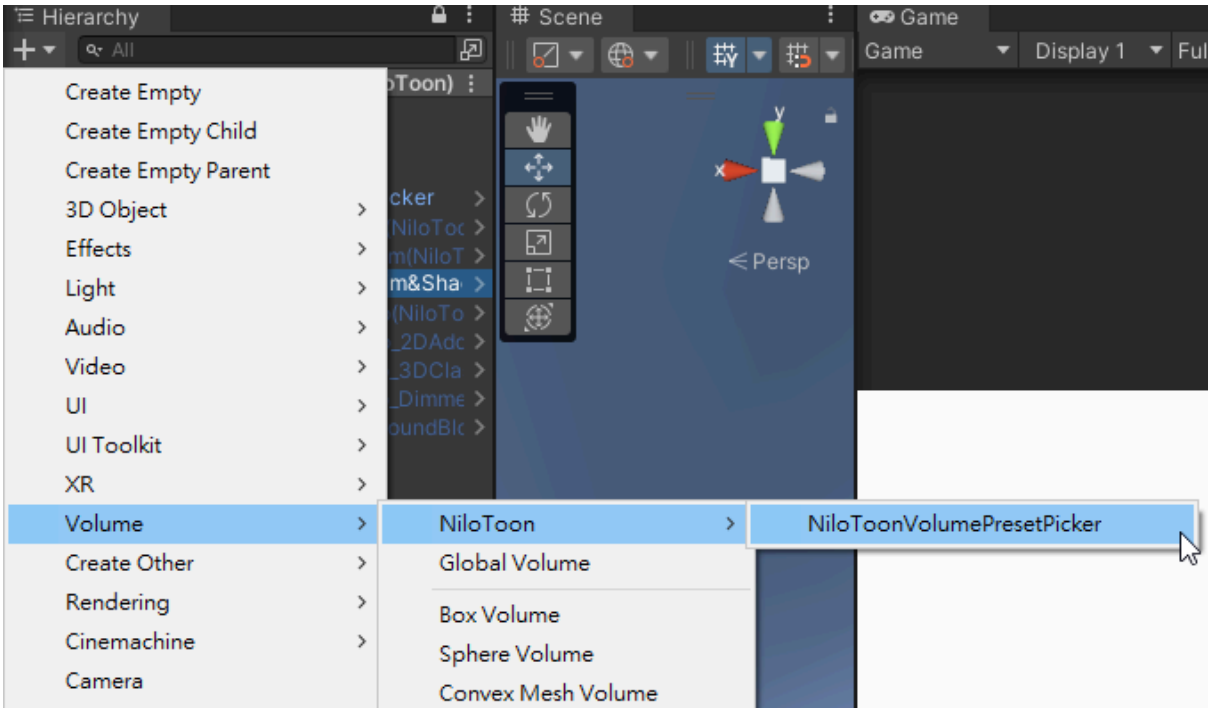
- Set **Depth Priming Mode** to **Disabled**, else face material using NiloToon_Character shader can not render **2D shadow** on the face.
- Set **NativeRenderPass** to **Off** as **NiloToon renderer feature** may not work correctly with **NativeRenderPass** (What is the purpose of this setting? - Unity Forum)



4a.+VolumePresetPicker

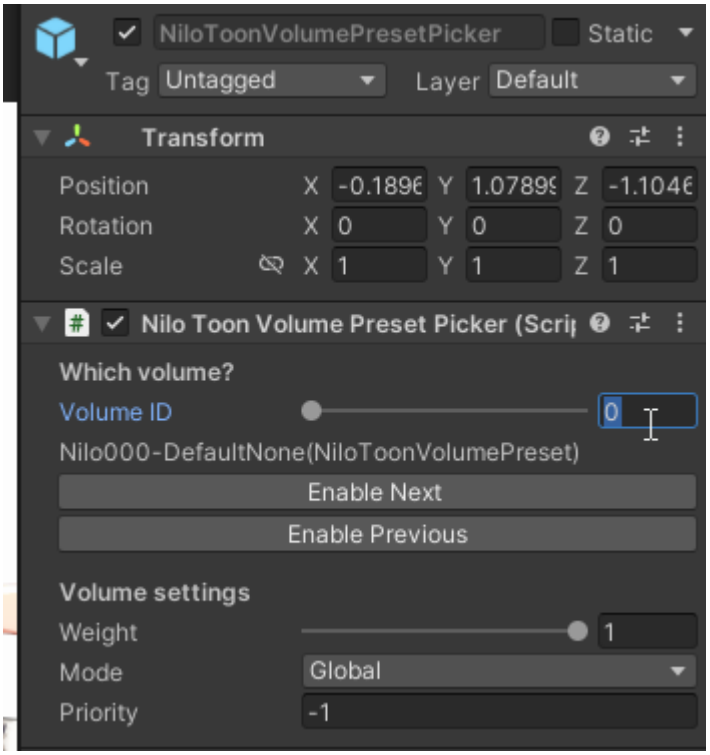
(optional) Add a **NiloToonVolumePresetPicker** to the scene by clicking:

+ > Volume > NiloToon > NiloToonVolumePresetPicker



NiloToonVolumePresetPicker allows you to pick a volume from a group of volume presets provided by NiloToon. After the setup in the later steps finishes, you can try different volumes later when setting up the character. Pick Volume ID = 1 or 2 (**Nilo001** or **Nilo002**) will be a good starting choice.

- Drag **Volume ID** or click **Next/Previous** to switch between volume presets
- Drag **Weight** to control the effect intensity

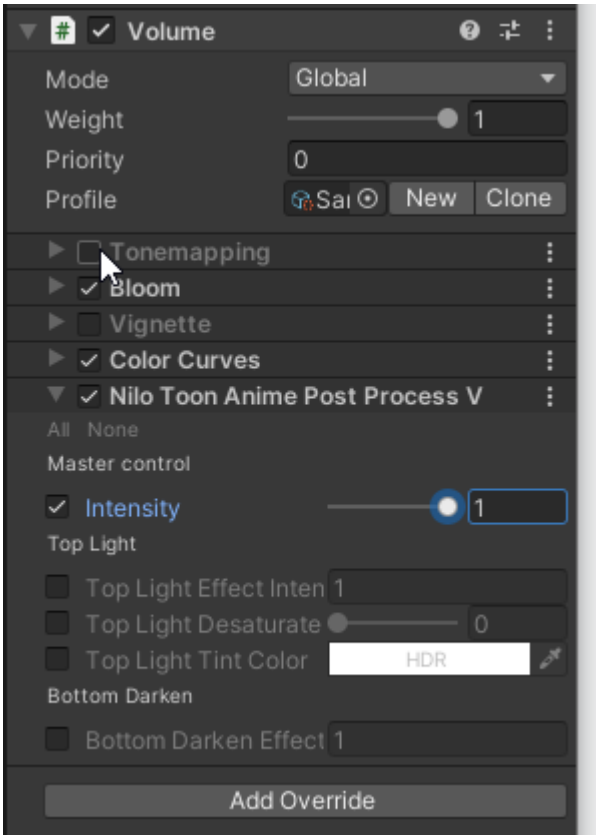


*If you want to use/edit one of the volumes, you should always **clone** the volume profile as your own volume profile asset, so you won't edit the volume profiles inside the NiloToonURP folder. Doing the clone can prevent future NiloToon updates from overwriting your volume profile edit.

4b.NiloToon Tonemap

Disable URP Tonemap

[Important Step]
We highly recommend you disable your Post-process Volume profile's **URP Tonemapping**.

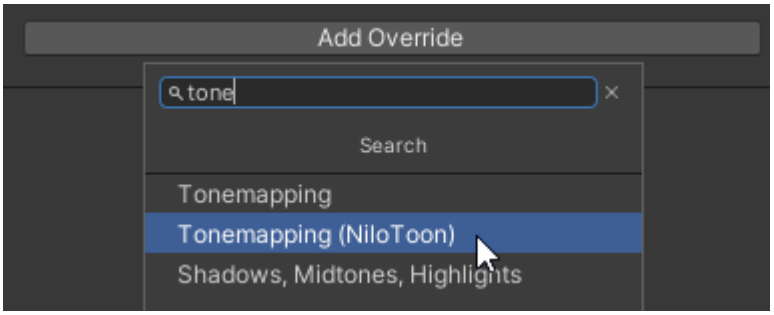


Technically there is no conflict with NiloToon, but **URP Tonemapping** usually destroys your character's NPR color design, especially **ACES**. In the VTuber industry, since character usually has a much higher priority/importance than the environment, it is very common to disable tone mapping to protect character's original NPR color design, so the 3D character model's color looks similar to the Live2D reference color or the 3D BaseMap color, even if that means bright light will clamp to rgb maximum/white color.
(You can keep **URP Tonemapping** turned on if you know tone mapping is needed in your project, although we **highly not recommend it**, please consider Use NiloToon Tonemapping if possible to produce a better tone mapping result for both the character and environment)

Use NiloToon Tonemap

It is ok to not use any tonemapping if you control the lighting intensity carefully, but if tonemapping is absolutely needed by a realistic PBR environment(e.g., your scene has very bright HDR volumetric light beams, light), and the **URP Tonemapping**

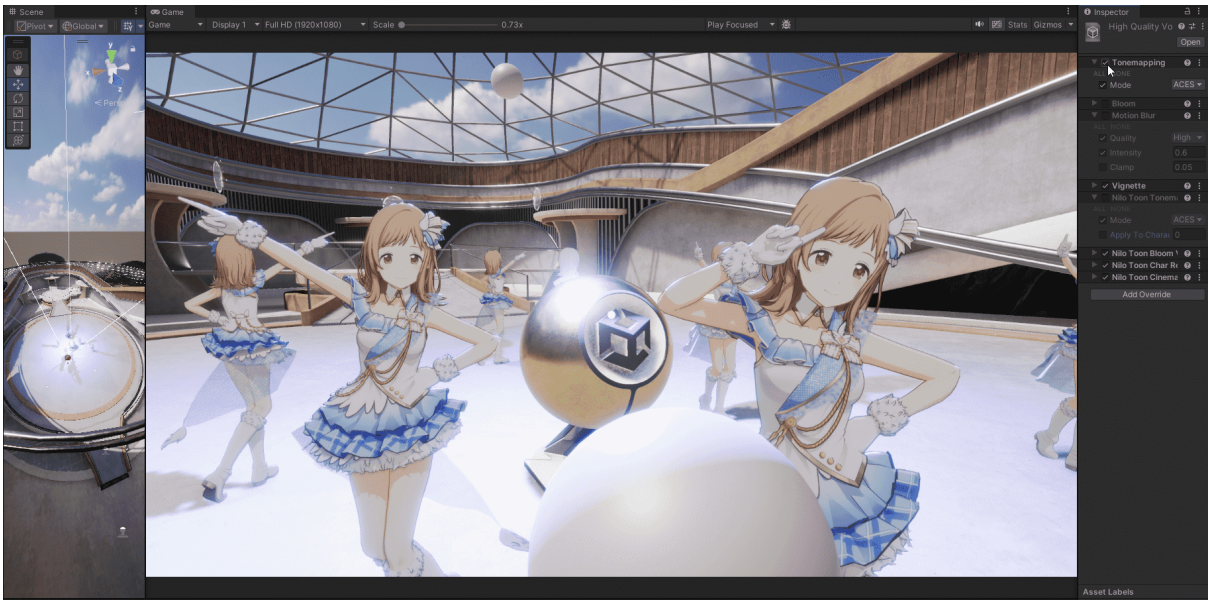
makes your character color looks dirty and bad, you can try to disable **URP Tonemapping** and use **NiloToonTonemapping** instead.



Here is an example video showing the use of NiloToon's tonemapping:
[NiloToon Unity6 concert demo]

NiloToonTonemapping is a similar tone mapping when compared to **URP Tonemapping**, but **NiloToonTonemapping** **will have extra options and controls, and will not apply to NiloToon character's area by default**

For example, when using **URP's Tonemapping(ACES)**, while the environment looks great, the character color will usually become dark, gray, desaturated and unappealing, see image below.



(Image above = using **URP's Tonemapping(ACES)**, character color is not good)

Switching to **NiloToonTonemapping(ACES)**, now tone mapping **will not apply to the character area by default, but still affects other non-character areas as usual**, which usually improves character's color result, see image below to see the difference/improvement.

(Image above = the character color becomes much better after switching from **URP's Tonemapping(ACES)** to **NiloToonTonemapping (ACES)**)

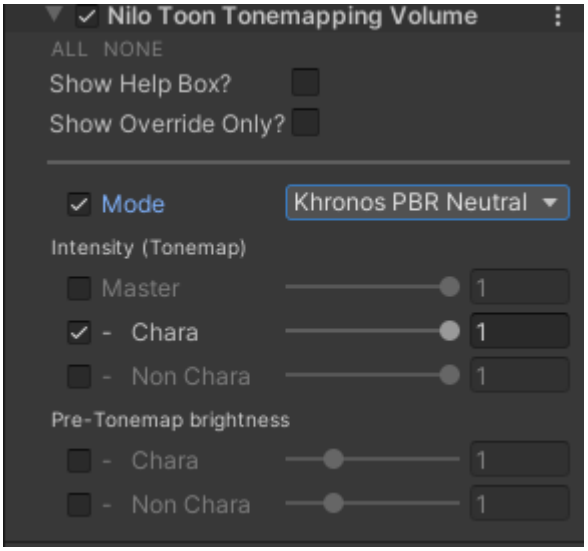
But please keep in mind that applying tone mapping to non-character areas only will make the character area over bright when compared to the environment, especially when there are additive effects applied on top of the character area like volumetric light beam, volumetric fog, HDR particles. For solutions see:

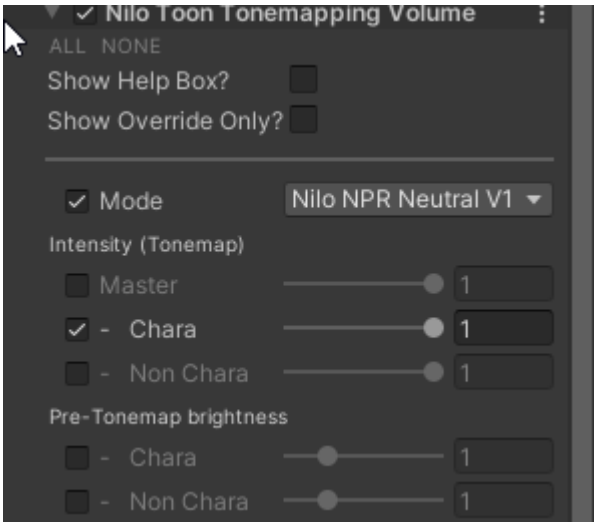
- Nilo Neutral setup
- Nilo Hybird ACES setup

Nilo Neutral setup

If you need to affect both **environment** and **character** together with the **exact same tone mapping**, and the HDR range/light intensity is not very high. We recommend trying these first:

1. using **KhronosPBRNeutral** or **NiloNPRNeutral**
2. set **Intensity (Tonemap) > Chara** to **1**





(images above are example setup of **KhronosPBRNeutral** or **NiloNPRNeutral** applying to the whole screen, the same tonemapping will affect both non-character and character pixels due to setting **Intensity(Tonemap) > Chara** to **1**)

These tone mapping modes are designed for producing a nicely tonemapped result color that is closer to **no tone mapping**, with a disadvantage of only supporting a very limited HDR brightness range.

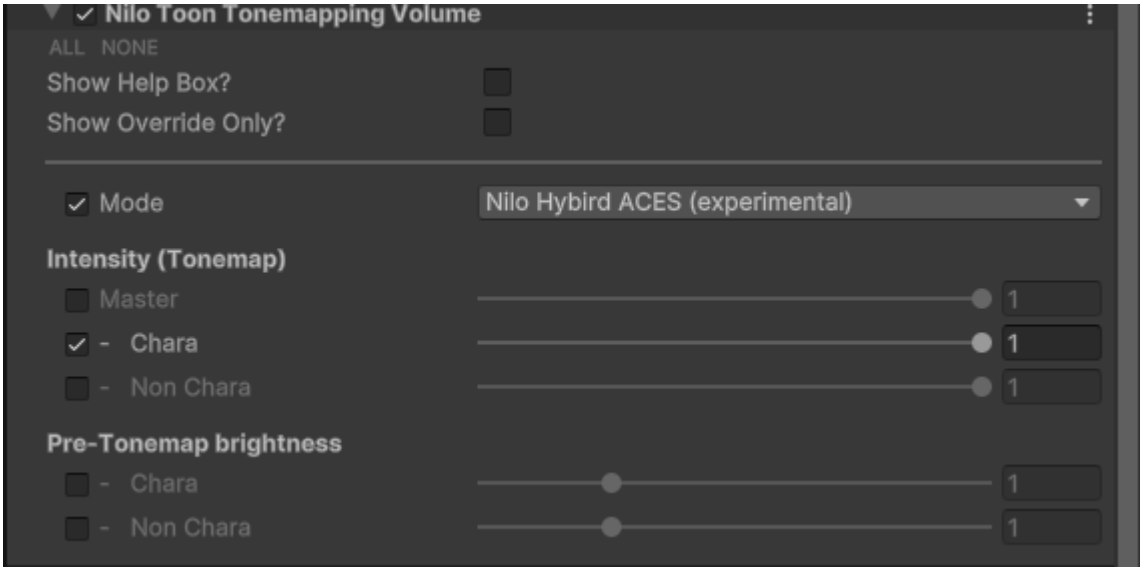
Nilo Hybrid ACES setup

Let's watch the result tonemapping first as an example:

- NiloConcertDemo

If you need **ACES** for environment like the above video, for example when you are working on a **realistic PBR scene(e.g., realistic 3Dlive concert or Archviz scene), extremely bright HDR lights, light beams, smoke and volumetrics**, we only recommend **NiloToon Hybird ACES**, you can use it by :

1. using **NiloToon Hybird ACES**
2. set **Intensity (Tonemap) > Chara** to **1**



Using the above setup means:

- applying **ACES** to non-character pixels = great for pbr or realistic environment pixels
- applying **NiloToon hybird ACES** for character pixels, which usually makes the character color look so much better for colors in the **LDR range** when compared to the regular **ACES**, and will produce a color similar to **ACES** in **HDR range**

(images above are example setup of **Nilo Hybird ACES** applying to the whole screen, tonemapping will affect character also due to setting Intensity(Tonemap) > Chara to 1)

You can also edit **Pre-Tonemap brightness**, which is a simple brightness multiplier (a.k.a **exposure**) right before tonemapping.Usually used for bringing the brightness/color of non-character and character pixels closer to each other, to help the character merge more naturally into a realistic scene.

Limitations

When using **NiloToonTonemapping** in any mode other than **None**, it is expected you:

- do not use **URP's Tonemapping**
- use **NiloToon's bloom** instead of **URP's bloom**

If you try to use URP's bloom when NiloToonTonemapping is active, you will see URP's bloom result reduced a lot.

The reason is due to the following render steps:
1. URP render all 3D object's **HDR** color as usual, the result is in **HDR**
2. NiloToon apply NiloToonBloom on top of the **HDR** color, the result is still in **HDR**
3. NiloToon apply NiloToonTonemapping, the result is now in **LDR**!
4. URP apply bloom, using the **LDR** as input but it is actually expecting **HDR** input, it is not expected. (this is why URP's bloom is reduced a lot, and why we recommend disable URP's bloom)
5. URP apply tonemapping, using the **LDR** as input but it is actually expecting **HDR** input, it is not expected (this is why we recommend disable URP's Tonemapping)

5. Project setup DONE

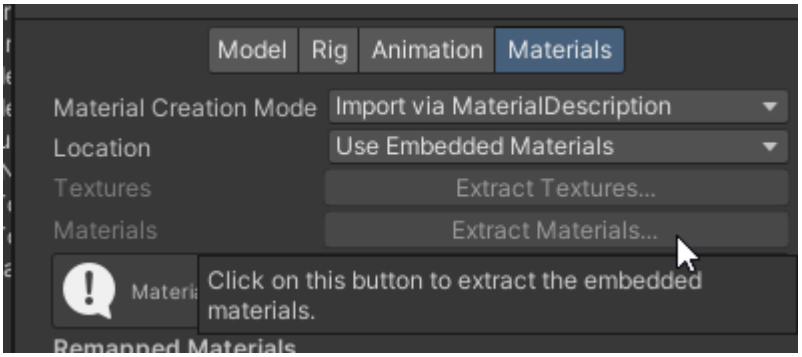
per project set up DONE, now you should focus on **Setup new character (Auto)** and **volume profile.** We will recommend doing a quick and minimum character setup first, and put your character prefab into the target environment to have an early check of the overall color.

Setup new chara (Auto)

This section will show the usual steps to set up a new character to use NiloToon, a new character usually means the **prefab** generated from **.fbx** or **.vrm** files.

0.Extract materials

[Important Step]
Select your character **fbx/vrm** in the project window, make sure your character's materials are all **extracted and editable**.
That means you should have clicked **Extracted Materials** from **.fbx/.vrm/.blend/.maya…** file's inspector already, the materials are **not sub-assets** of the model file anymore, but are actual **editable materials** in the project window)



*For **VRM0.x**, you don't have the option to extract materials, it is done automatically

1.Setup Chara prefab

[Important Step]
This is the key step to setup a character prefab, pick one of the following methods to set up your character prefab.

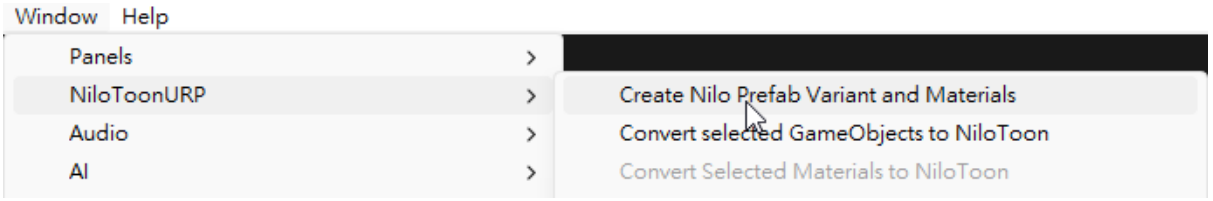
*Before you begin, make sure you don't have any **missing scripts** inside the prefab, because saving unity prefab requires no missing script.

There are a total of 3 methods to setup a character, we recommend **1A** > **1B** > **1C**, most of the time we only use **1A**:
1A) Non destructive auto setup (recommended)
1B) Destructive auto setup
1C) Manual setup (not recommended)

1A. Non destructive

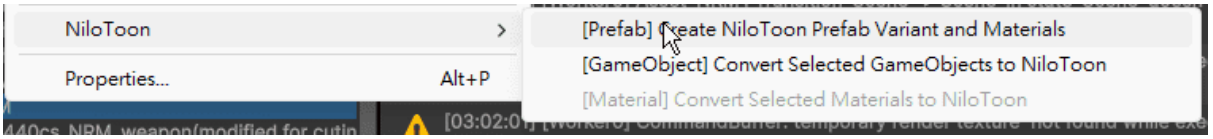
If you **do not** want NiloToon to **overwrite/edit** any existing prefab and materials, always pick this!
If you want to convert a **vrm0.x** character prefab to NiloToon, always pick this!

After selecting your character's Prefab in the **project window**, click:
Window > NiloToonURP > Create Nilo Prefab Variant and Materials



or

After right clicking the prefab in **project window**, click:
NiloToonURP > [Prefab] Create Nilo Prefab Variant and Materials



Clicking this will not edit any of your character's existing prefab or materials, and only generate new prefab variant + material clones. After this setup everything is "**Added files**" in your **version control**, it means you will keep the original character prefab and materials unmodified, and have a new character prefab variant that uses a new set of NiloToon materials.

This setup allows you to keep updating the original character file (e.g., .vrm / .fbx files) in the future, and keeping the original prefab/material unmodified.

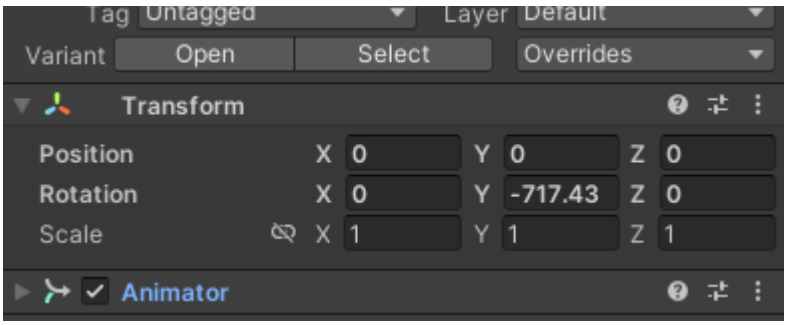
*After the click, a **pop-up** window will appear, always click **yes** to convert the new material clones to use NiloToon, see **Convert Mat to Nilo**

1B. Destructive

If you did 1A/1C already, you should skip this step.

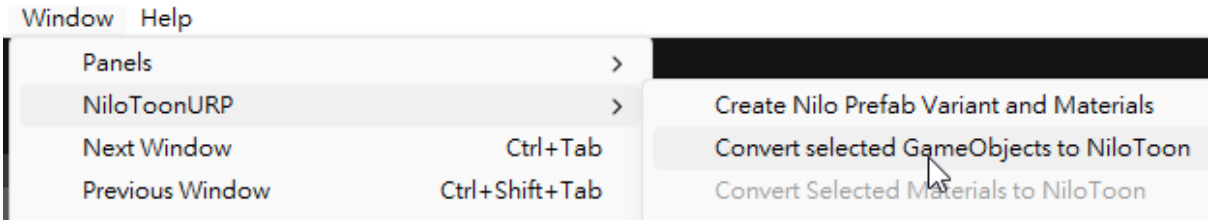
If your original material settings are not needed anymore, and you do want NiloToon to **overwrite** existing character prefab/materials, pick this!

In **hierarchy** window, after **selecting** your character prefab/gameobject's root gameobject(A GameObject that includes all renderers and bones of a character, usually this GameObject is where the character **Animator** component is generated),



click:

Window > NiloToonURP > Convert selected GameObjects to NiloToon



Clicking this will edit your character prefab and materials directly. After this setup everything is "**Modified files**" in your version control, it means you will lose the original character shader/material settings, since NiloToon overwrites them using NiloToon shader.

*After the click, a **pop-up** window will appear, always click **yes** to convert the new material clones to use NiloToon, see **Convert Mat to Nilo**

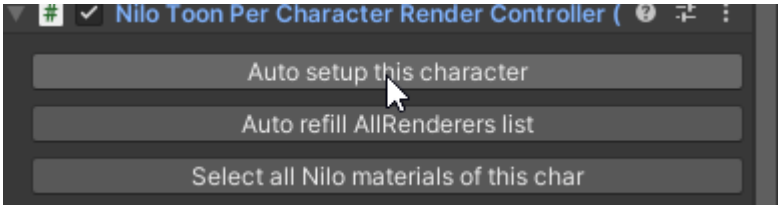
1C. Manual

If you did 1A/1B already, skip this step.

Most of the time you can skip this step, since you should be using 1A/1B in most cases.

1. After selecting your character's root GameObject/Prefab,
manually add a new script **NiloToonPerCharacterRenderController** to this game object (you can type **nilo per** to find the script).

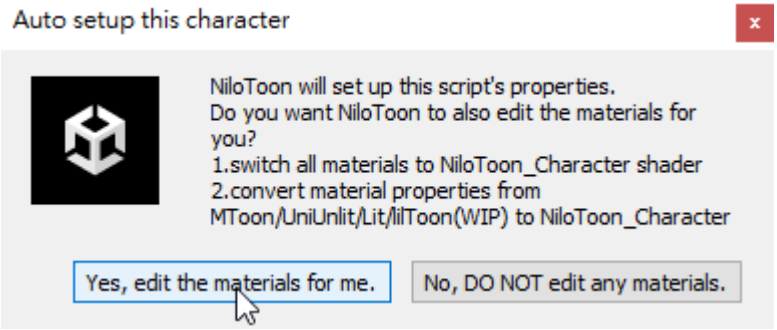
2. Then click the "**Auto setup this character**" button



*After the click, a **pop-up** window will appear, always click **yes** to convert the new material clones to use NiloToon, see **Convert Mat to Nilo**

2 Convert Mat to Nilo

After doing the above **1A/1B/1C method**, a “Auto setup this character” pop up window will appear, click **yes** to this pop up window,



this will automatically handle material and shader conversion to NiloToon’s shader, the following shaders are supported in the shader conversion:

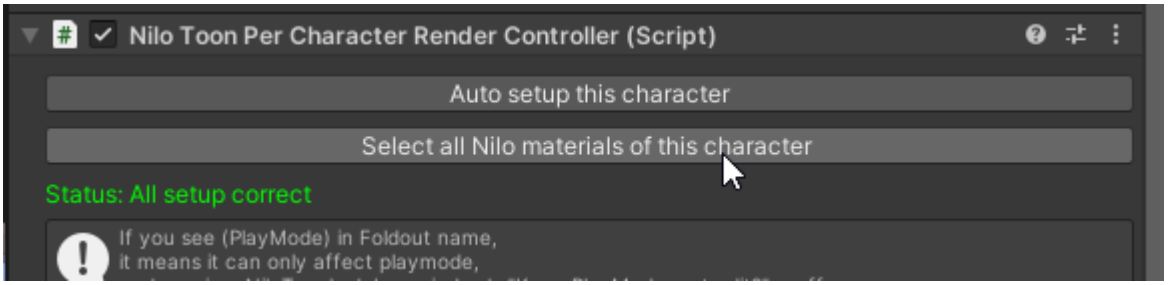
- **URP Lit/Unlit/Complex Lit/Simple Lit** to NiloToon
- **vrm materials(VRM’s MToon)** to NiloToon
- **lilToon** to NiloToon

3 Enable Alpha Clipping

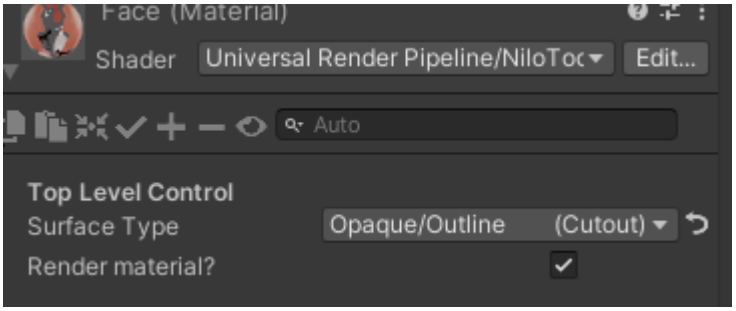
(you can skip this step if alpha clipping is working correctly already)

If the **BaseMap** textures has an alpha channel for alpha clipping, and NiloToon didn’t enable alpha clipping

you can click on the **Select all Nilo materials of this character** button

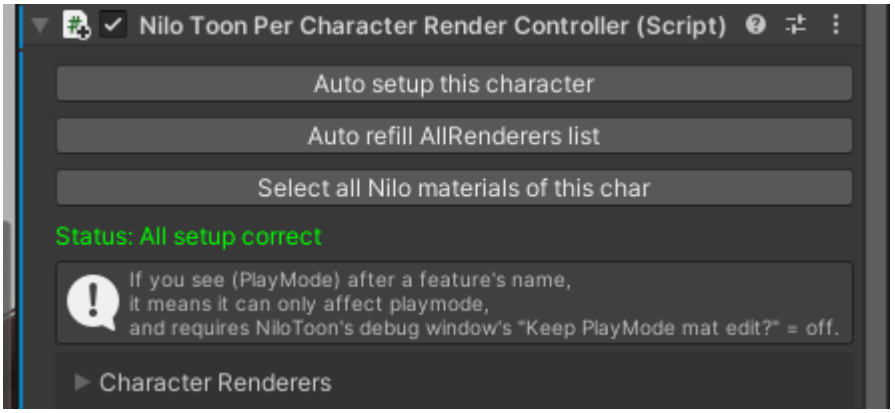


This will select all materials of this character, then convert **Opaque/Outline** to **Opaque/Outline (Cutout)** for all character's material, after that, NiloToon character materials will use the alpha channel of the BaseMap for alpha clipping



4 Set Per Chara script

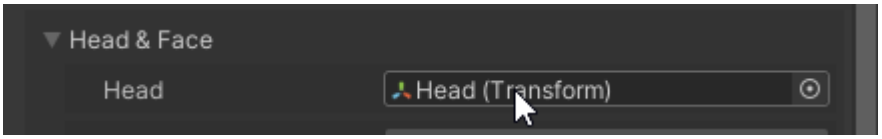
NiloToonPerCharacterRenderController will try to set up some fields of the script using the character's bone structure. If successful, you will see Status: All setup correct same as the picture below



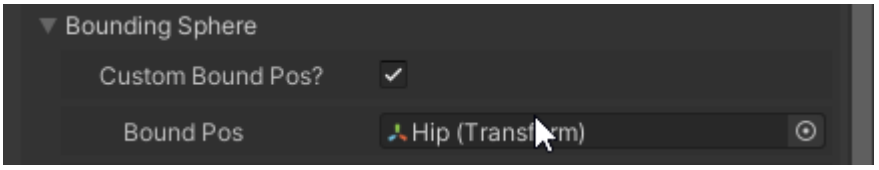
Head & Hip bone

In rare cases, if you model doesn't have **hip** bone and **head** bone, you will need to manually assign them into **NiloToonPerCharacterRenderController**'s:

- Head & Face > **Head** (Transform). Assign the **head bone** here.



- Bounding Sphere > **Bound Pos** (Transform). Assign the **hipbone** here.

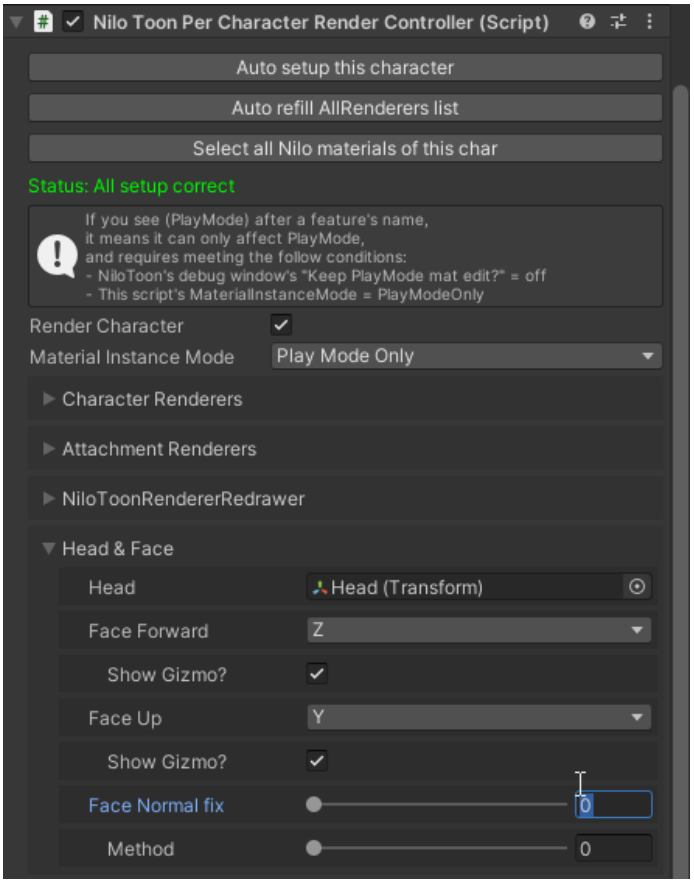


*If your prefab is not a character, you can assign any gameobject that is around the center of the prefab as a hip bone.

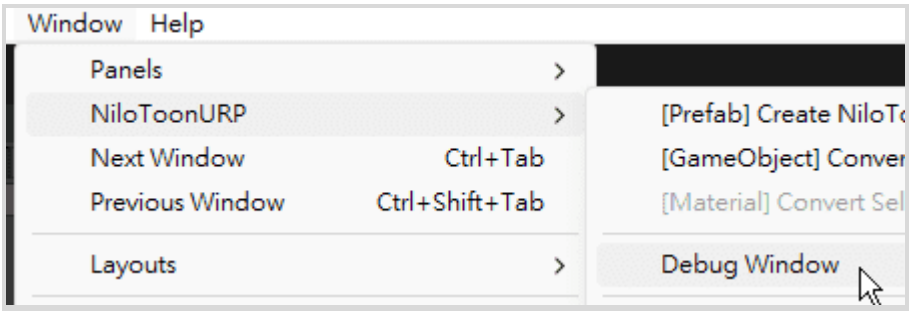
Face Normal fix

NiloToonPerCharacterRenderController will also try to flatten your face's lighting normal by default (**Face Normal fix** = **1** by default), if your model has special normal for lighting, and you don't like NiloToon's flatten face normal, you can set **Face Normal fix** = **0**.

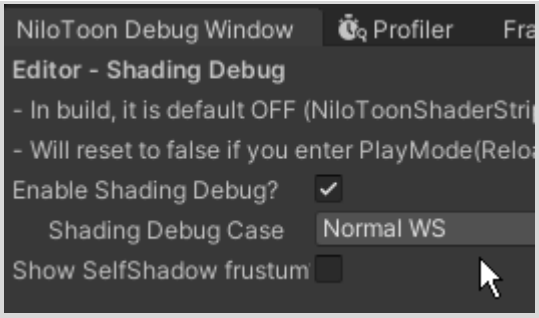
When **Face Normal fix** = **0**, NiloToon shader will use the face model's original vertex normal directly for lighting.

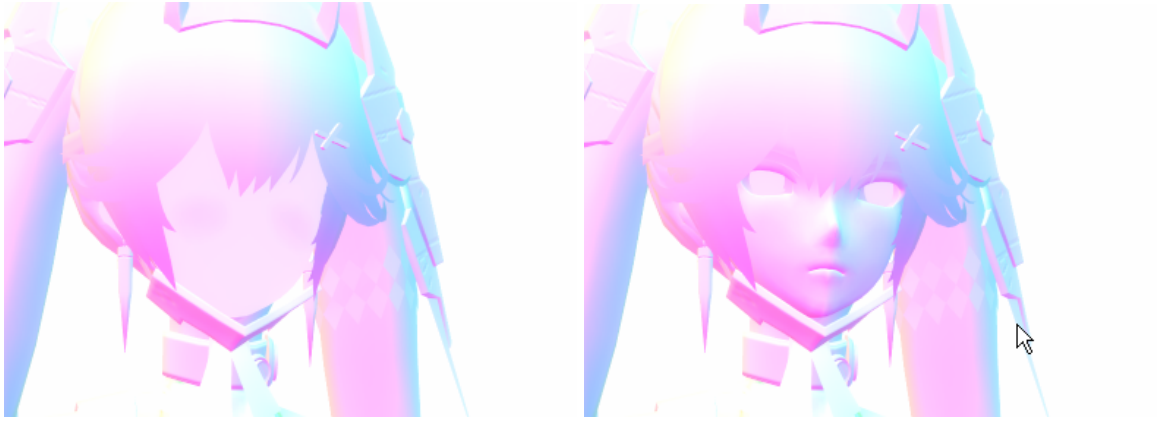


To **debug** face normal, you can use **Window > NiloToonURP > DebugWindow**



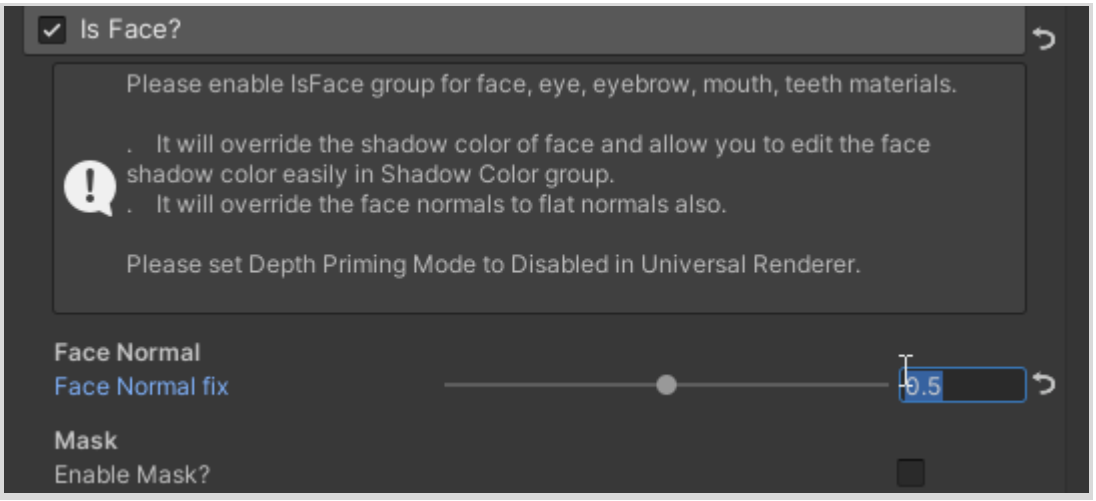
Turn on Enable Shading Debug, and check NormalWS





(Left: **Face Normal fix** = **1** | Right: **Face Normal fix** = **0**)

If you want to control **Face Normal fix** **per material**, **use Is Face? > Face Normal fix**



Eyeglasses

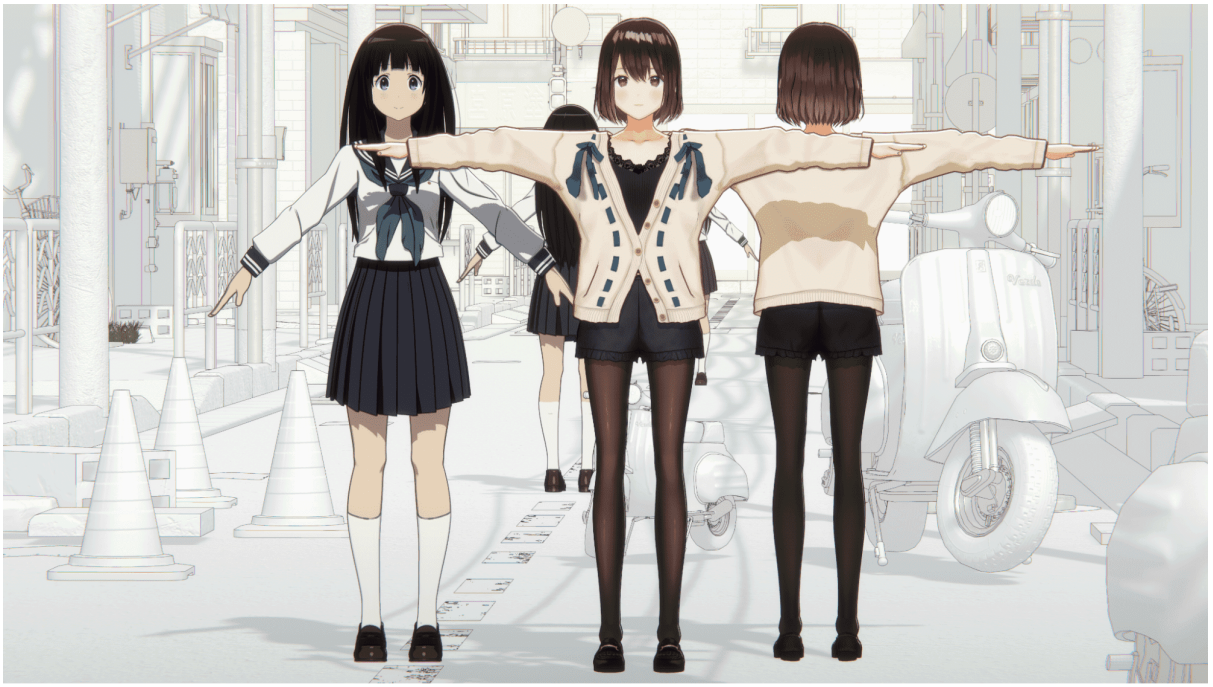
For **eyeglasses**, which will drop a **2D shadow** on the face.
If you don't like this 2D shadow produced by the eyeglasses, you can reduce/remove the 2D shadow dropped by the eyeglasses.

To do so, inside the eye glasses material:

1. Enable **Is Face?**
2. Edit **Is Face? > Face Normal fix** to **0**

This will treat the eyeglasses material as part of the face, but NiloToon will not edit it's normal due to step(2).

After all the steps above, now you should already see your model set up correctly,

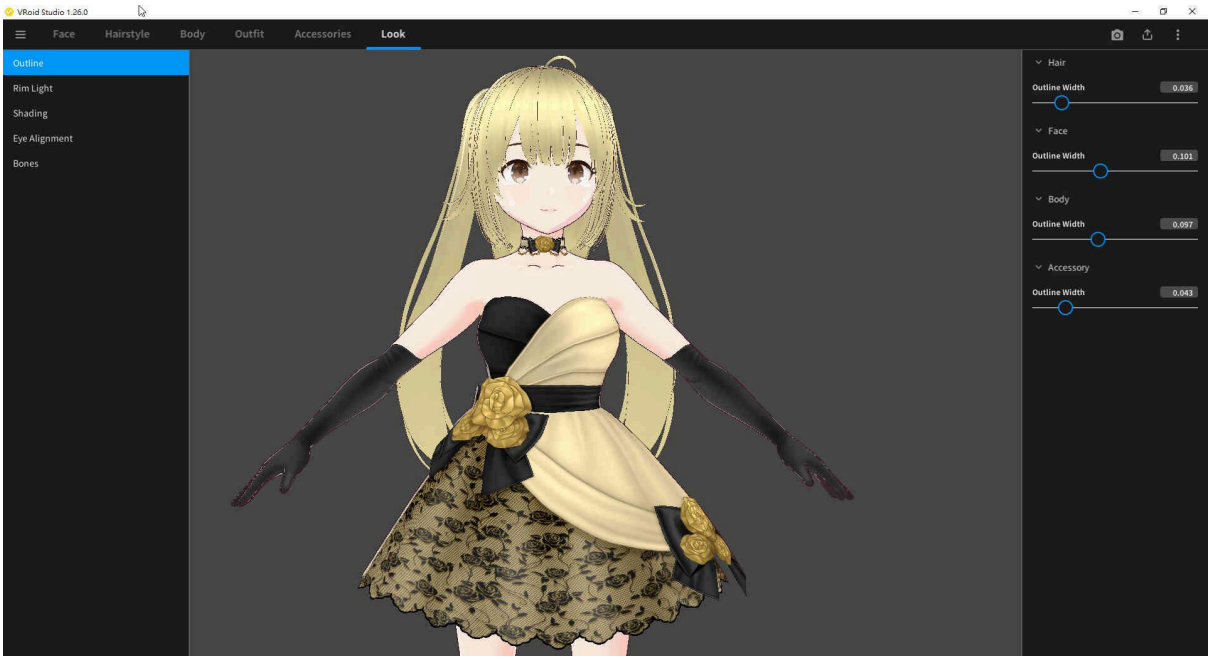


if you want to improve the result, continue reading [Improve chara mat]

VRoid exported vrm

VRoid studio will export vrm file, and NiloToon's auto setup will work for any vrm files automatically, the images below are the expected result of:

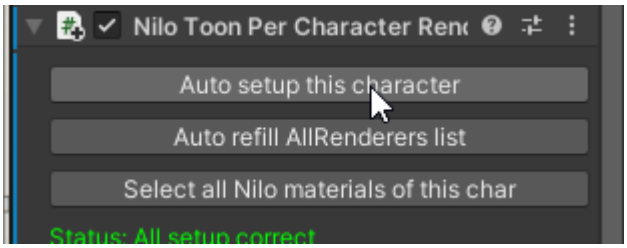
(1) VRoid Studio export vrm file



(2) after NiloToon auto setup using the above steps

Convert BRP mat to URP

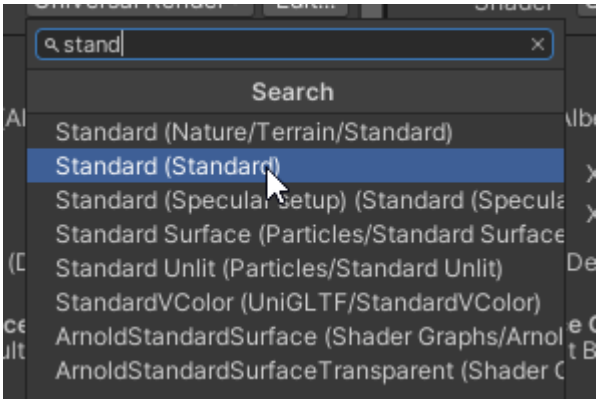
(!): You should skip this section if your character materials are already using URP shaders, or you are setting up the character using any auto methods in the above section



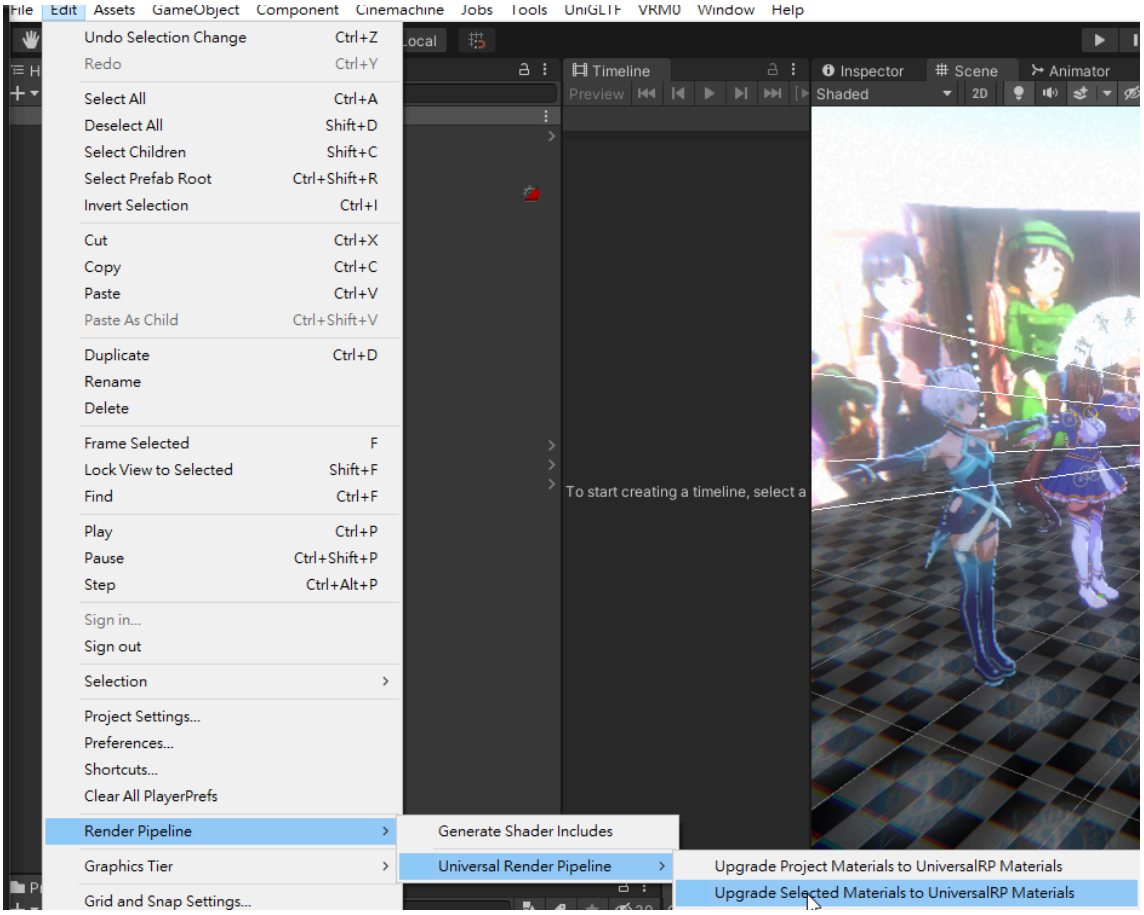
If you have a character using Bult-in RP materials and you decided to not using the **Auto setup this character button** (which we highly recommend to use the auto button and completely skip this section), you can still convert the materials to NiloToonmanually (which we don't recommend).

1.select all character's BRP materials

2.switch material's shader to **Standard**(the Standard shader of Built-in RP), doing this will allow URP to convert the Built-in RP Standard shader to a matching URP Lit shader in next step



3.click this button



If you skip the above steps and switch a built-in RP material to NiloToon character shaders directly, you will lose texture references on the material (become all white), since the naming of Built-in RP(**_MainTex**) is not the same as URP(**_BaseMap**).

Improve chara mat

After NiloToon's auto material conversion, the result will not always be perfect. You can improve/fix the character materials by manually editing some important material settings, below are some common important buttons and properties of character materials, we recommend **at least** reading:

[Important Material Inspector buttons]

- Group UI display mode
- Show only modified properties
- Revert button

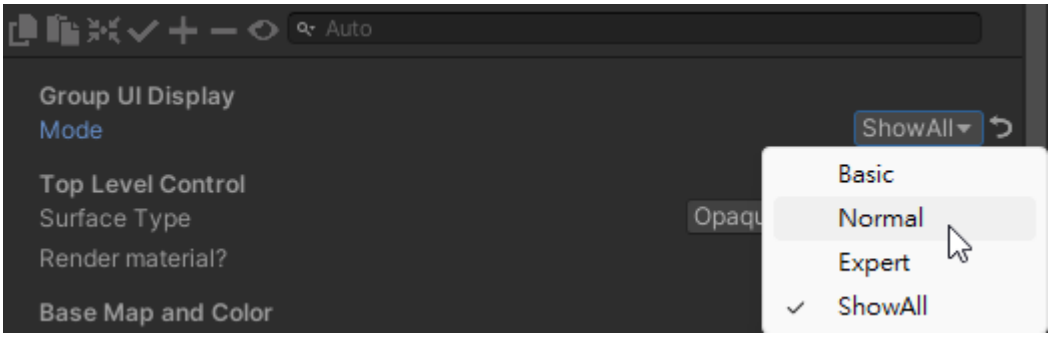
[Important Groups/Properties]

- Surface Type
- Is Skin? / Is Face?
- Render Face(Cull)
- Classic Outline
- Shadow Color
- Lighting Style

in order to perform the most basic setup for NiloToon materials.

Group UI Display mode

This is an important option!



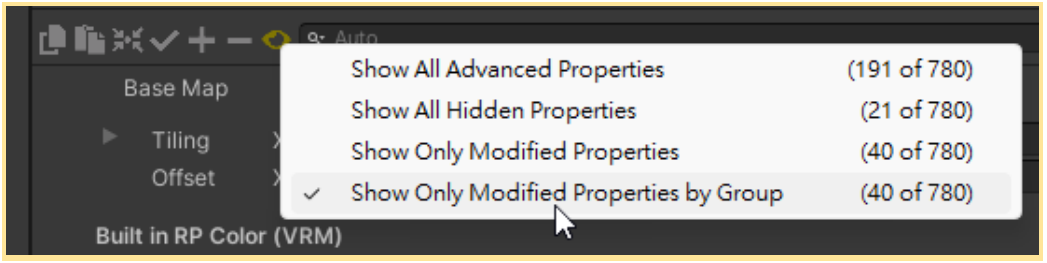
At the top of the NiloToon_Character material, you will see **Group UI Display > Mode**, there are 4 **modes** you can pick in **Group UI Display**,

- **Basic** shows a minimum number of groups, only great for first time user
- **Normal** shows all core and commonly used groups (**default** mode)
- **Expert** shows a larger number of groups
- **ShowAll** shows all groups, including all experimental and rarely used groups

Group UI Display Mode only controls the visibility of the feature groups in the material, it doesn't affect the effectiveness of the group, so it is possible for a group to be active, but invisible due to **Group UI Display Mode**. If there are groups that are not visible in the material inspector, we recommend setting **Group UI Display Mode** to **ShowAll** to make all groups visible.

Show only modified

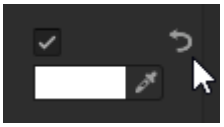
This is an important button!



NiloToon by default will show all groups and properties defined by **Group UI Display Mode**. There can be many groups and properties that may overwhelm you. In order to let you check only **what has been actually edited** in the material, you can click on the **eye icon** at the top of the material inspector, and select **Show Only Modified Properties by Group.**
Now **only modified properties/groups** will be shown, and all properties/groups with default value will be hidden.
It is great to use **Show Only Modified Properties by Group** when you want to understand all actual edited settings in the material that are not default values (all modified properties will have the **revert button**). **Show Only Modified Properties by Group** is especially useful if you need to understand what exactly are the settings that are not default values, for understanding, debugging or studying that material.

Revert button

This is an important button!



If a property or group has been modified (not default value), a revert button(white arrow) will appear. You can always click the revert button on any modified properties or groups to **revert** them to the **default value**. Useful when you edit some properties or groups, don't like the new result and want to reset to default value.
You can also view it as a marker to quickly understand which properties have been modified(not default value anymore), since only modified properties will have a revert button.

Tooltip

If you want to know more about a property or a group, put the mouse on the property or group, a detailed tooltip will appear explaining what that property or group is.

In the image above, you can put the mouse on any group or property to view tooltip details.

SurfaceType

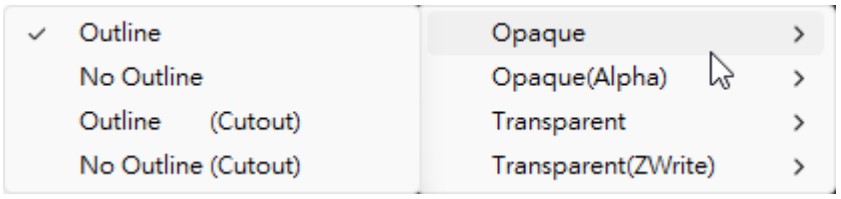
This is the first option you should check, it is the most important setting!
Similar to the URP **Lit** shader's **Surface Type**, in the NiloToon_Character shader, it has a similar **Surface Type**. You should select the correct **Surface Type** of the material first before editing other properties, common **Surface Type** are:

- **Opaque/Outline**: the **default** surface type, opaque with classic outline, **majority of materials** will use this, e.g. any opaque hair/body/face/cloth that without the need of semi-transparent
- **Opaque**: same as **Opaque/Outline**, but without outline, great for material that don't need outline e.g. eye, eye brow, mouth, teeth, nail, thin hair, thin cloth
- **Opaque(Alpha)/Outline**: same as **Opaque/Outline**, but added semi-transparent alpha blending. e.g., semi-transparent **front hair**, semi-transparent cloth with mainly high alpha value so that it **should cast shadow map**
- **Transparent(ZWrite)/Outline**: the classic Unity transparent, but with ZWrite and outline, e.g., semi-transparent plastic outer jacket/cloth with mainly low alpha value so that it **should not cast shadow map**
- **Transparent**: the classic Unity transparent without ZWrite and outline, e.g., any transparent materials that don't need outline and don't cast shadow map, e.g. semi-transparent glass material of an eye glass, emoji, face expression, vfx

*If you need to display **Classic Outline**, select the one with **/Outline suffix**, else select the one with **/No Outline suffix**

*If you need to apply **Alpha Clipping (a.k.a. Alpha CutOut)**, select the one with **(CutOut) suffix**, else select the one without **(CutOut) suffix**

*Changing the **SurfaceType** will **reset** the **render queue** to the default render queue of the selected SurfaceType. If you need a specific render queue, make sure to set the render queue again after changing **SurfaceType**



Is Skin? / Is Face?

This is an important setting!

The **skin** and **face** material will require special care to look good, in material the

- **Is Skin?**
- **Is Face?**

groups will be useful for this task.

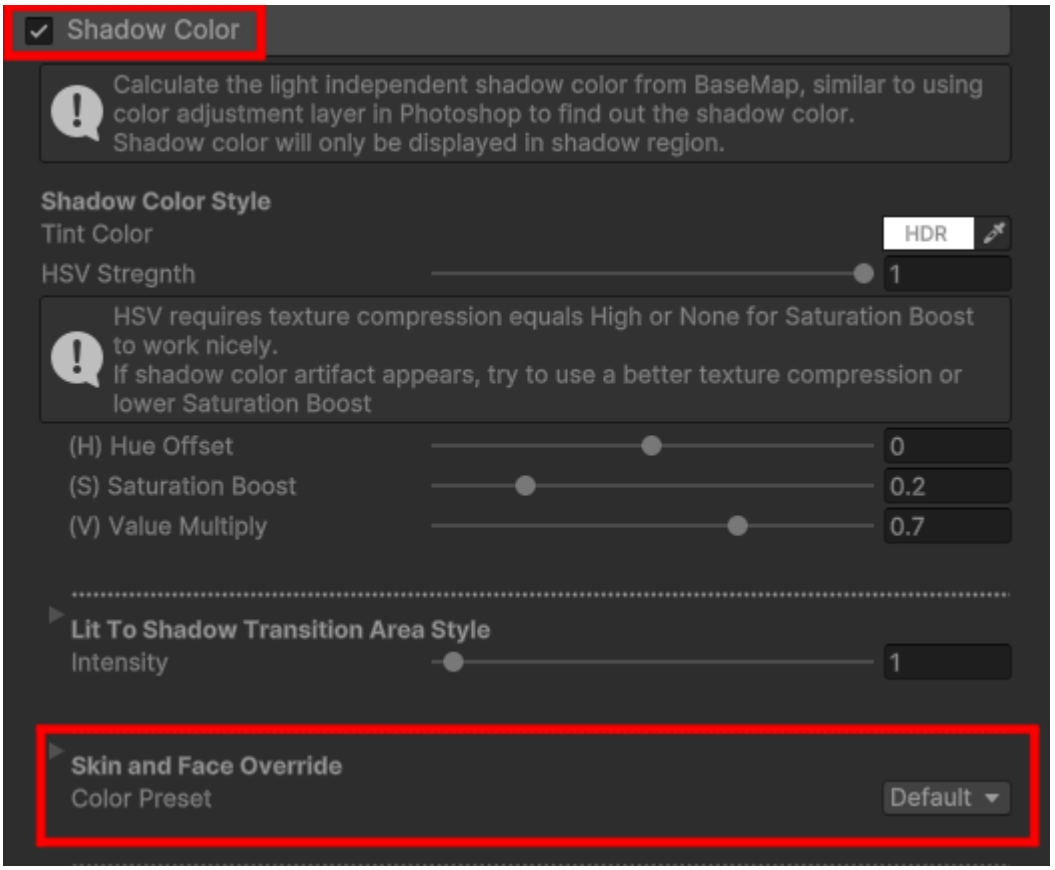
Settings and effect

Find all **skin** materials (e.g. face, mouth, body, hand, leg), turn on **Is Skin?**, else turn it off. If the material mixed skin and non-skin part's into the same material, use **Is Skin?** > **Mask** to mask it

Enable **Is Skin** will:

- make the material's **Shadow Color** use an **overridden reddish skin shadow color** in the **Shadow Color** group, instead of using the default shadow color(the default shadow color may not look good for skin if you didn't turn on

Is Skin group)



*To edit the skin color, use Shadow Color > Skin and Face Override,

Find all **face** materials (e.g. face, mouth, eye & eyebrow), turn on **Is Face?**, else turn it off. If the material mixed skin and non-skin part's into the same material, use **Is Skin?** > **Mask** to mask it

Enable **Is Face** will:

- edit face's shading normal in shader(shading normal will become flat/round, defined by **NiloToonPerCharacterRenderController** script's **fix face normal** slider)
- override to use values from **Lighting Style (face)** group instead of the standard **Lighting style** group
- make the material's **Shadow Color** use an **overridden reddish skin shadow color** in the **Shadow Color** group, instead of using the default shadow color(the default shadow color may not look good for skin if you didn't turn on **Is Face** group)

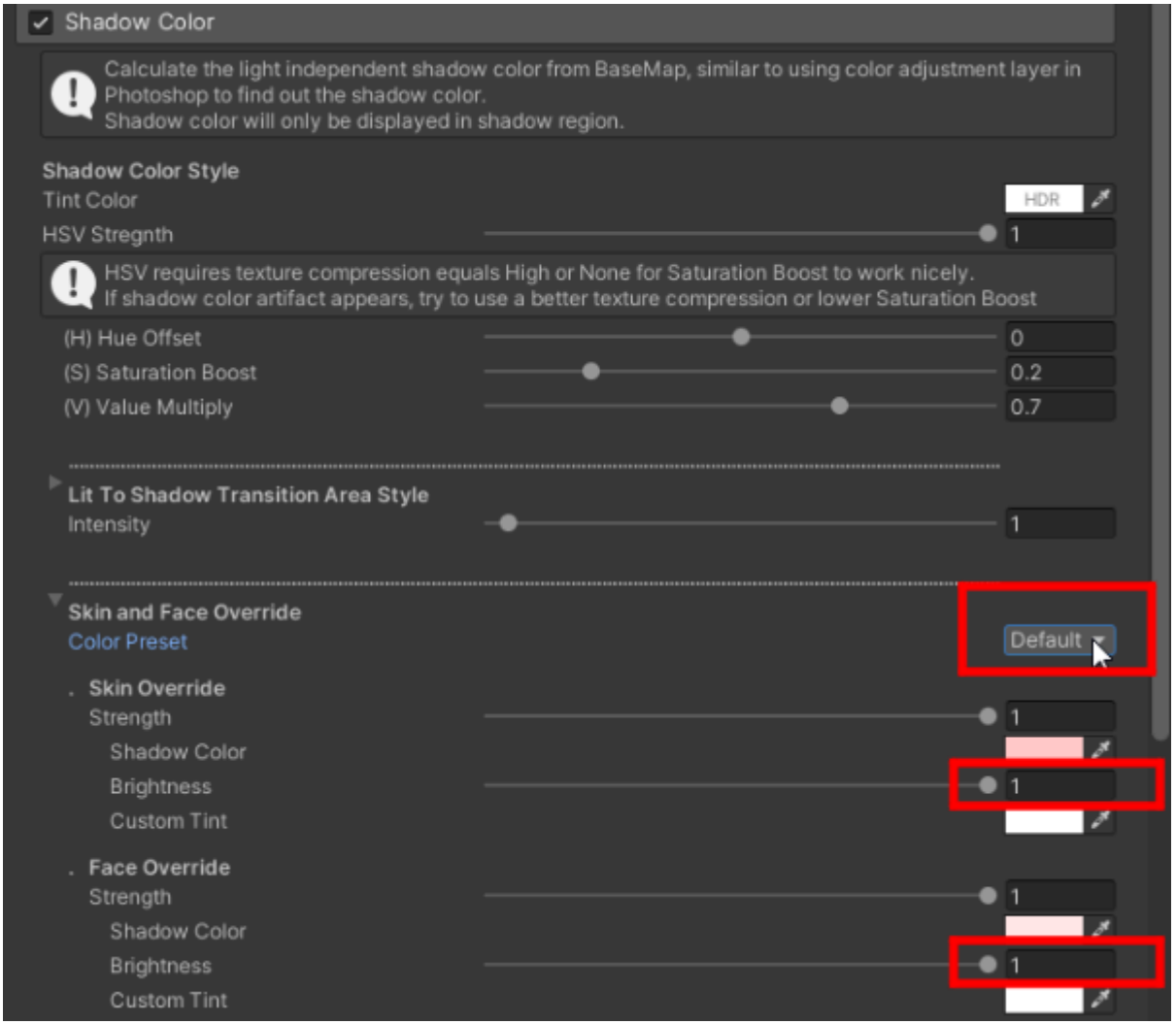
*Don’t enable **Is Skin?** for non-skin materials
*Don’t enable **Is Face?** for non-face materials

ShadowColor skin/face

After marking the Skin and Face area in material, you can now control the skin and face shadow color in material using this section in material:
Shadow Color > Skin and Face Override

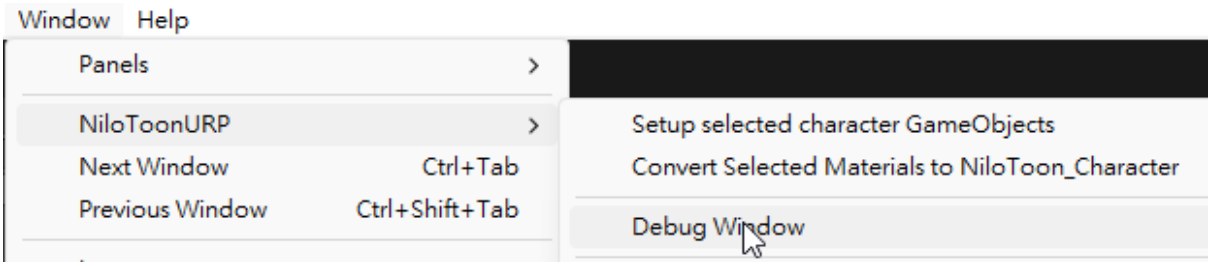
If you don't like the current skin/face shadow color, do the following:

- Pick a good **Color Preset** (Red/Orange/Purple and it's variants), it will define shadow color's **Hue** and **Saturation**
- Control **Brightness** / **Custom Tint** for both skin and face, it is ok to lower the **Brightness** to 0.8~1 to make the shadow color darker, especially for **male** character or character with a more **realistic style**



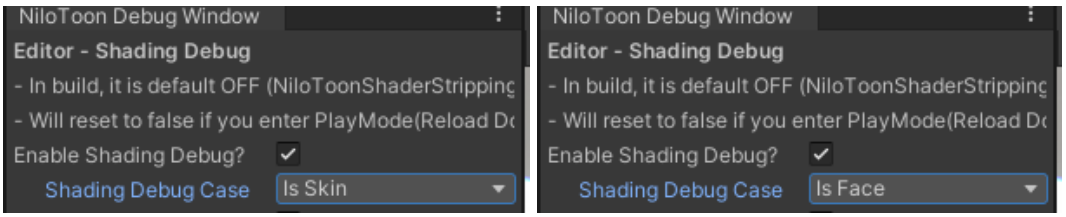
Debug skin/face area

You can debug the character's **Skin** and **Face** area using NiloToon's debug window, click: Window > NiloToonURP > Debug Window



then

- select **Is Face** or **Is Skin** in Shading Debug case
- enable **Enable Shading Debug?** toggle



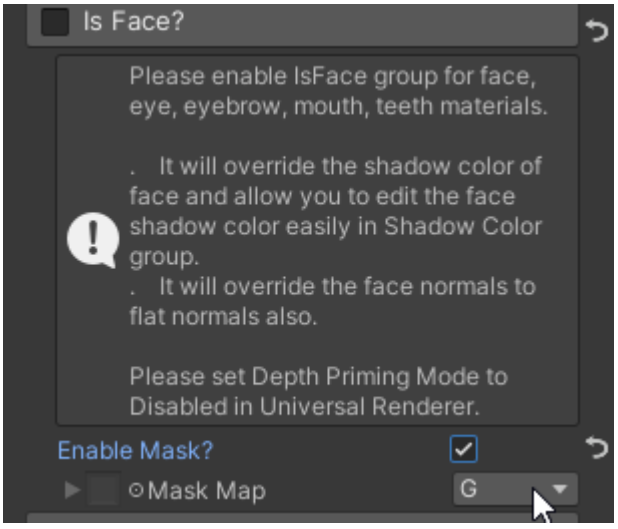
Then you can debug the skin and face area as a grayscale value, where the marked area will be shown as white, else black.

Neck shadow split

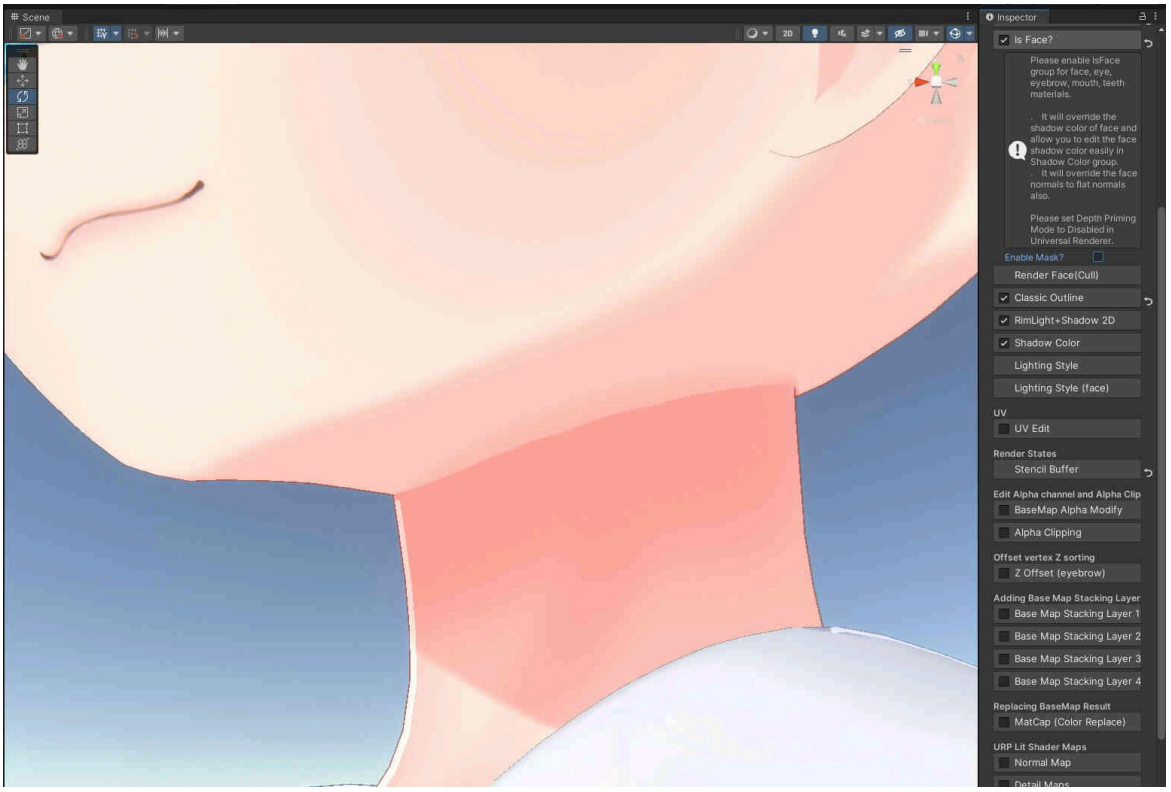
This is an advanced topic, you can skip it if this is not a problem in your model.

Standard solution

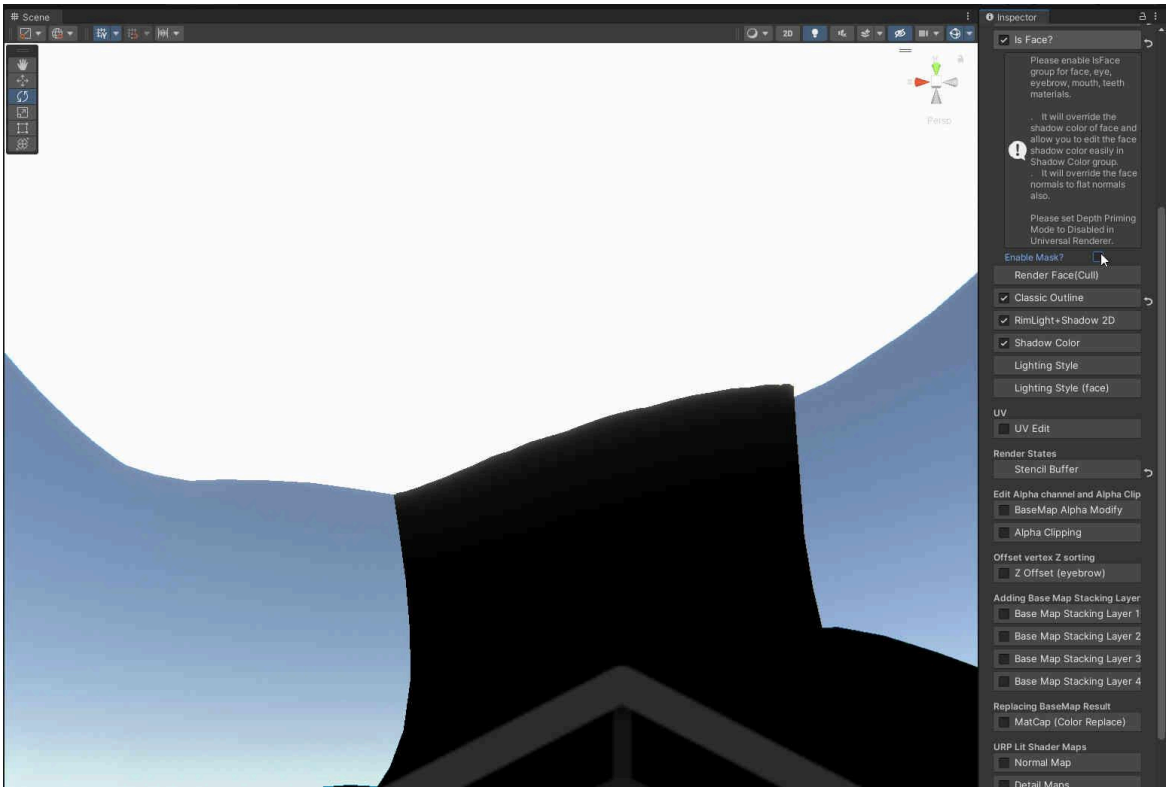
Is Skin? and **Is Face?** both have a **Mask Map** feature for you to mask out areas that should not be considered as **Is Skin?** or **Is Face?** using an extra texture.



A common problem about shadow usually appears when the face material (Enabled **IsFace**) and body material (Disabled **IsFace**) split around the **neck**, which will make **shadow looks separated around the neck**(a visible line at neck area, where shadow color and shadow map rendering are not matching due to 2 materials having different **IsFace** settings)

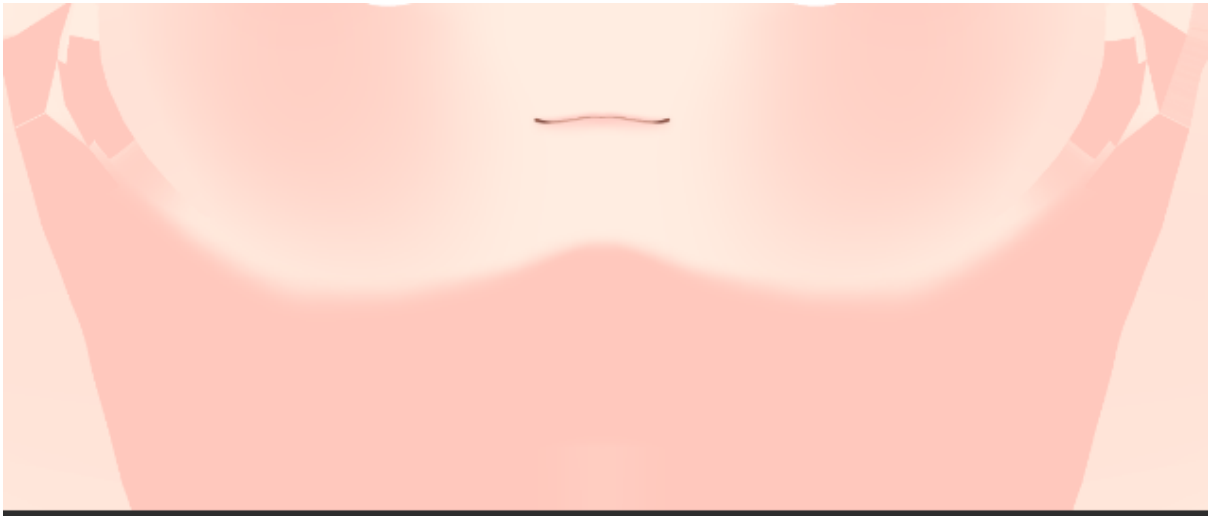


(Image above shows the **shadow split problem** at the neck)



(Image above shows the current **debug view** of **Is Face**, you can see the **bottom part of the head** is also considered as Is Face, which might not be ideal since it will produce a split line in shadow)

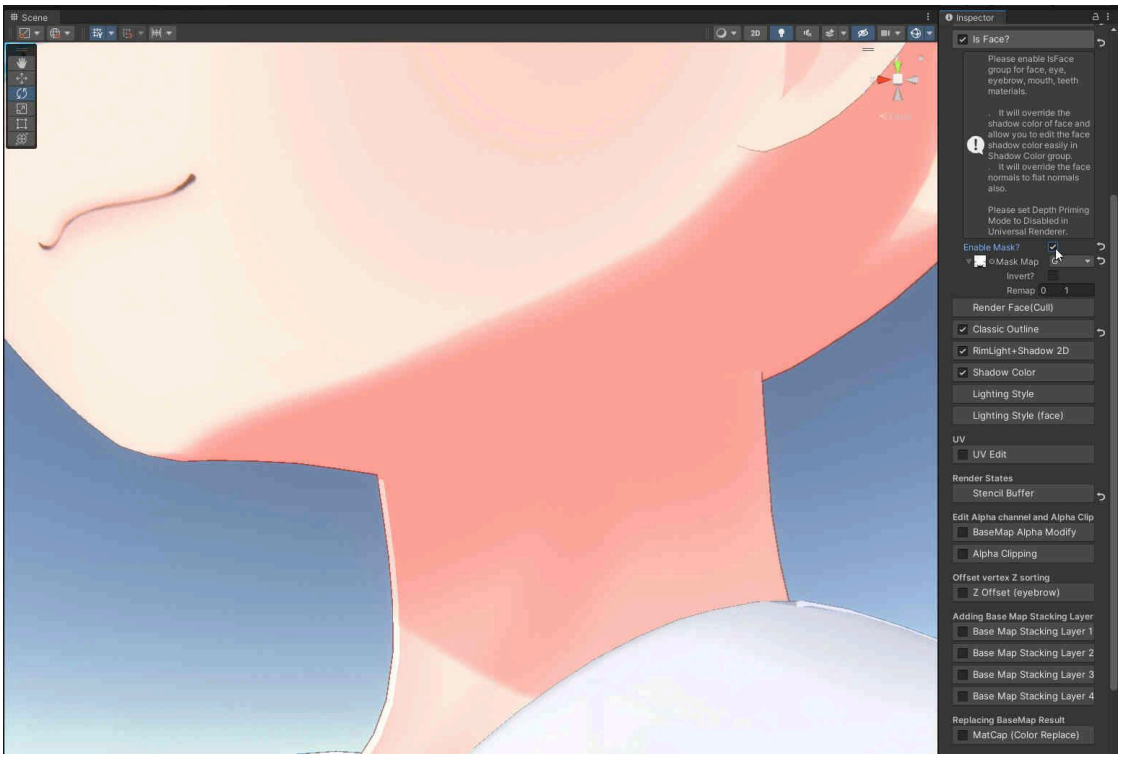
A common solution to this problem is to draw an extra mask map to make the **bottom part of the head** not considered as **Is Face**, usually the **bottom part of the head** is where the artist will paint shadow color in the basemap.



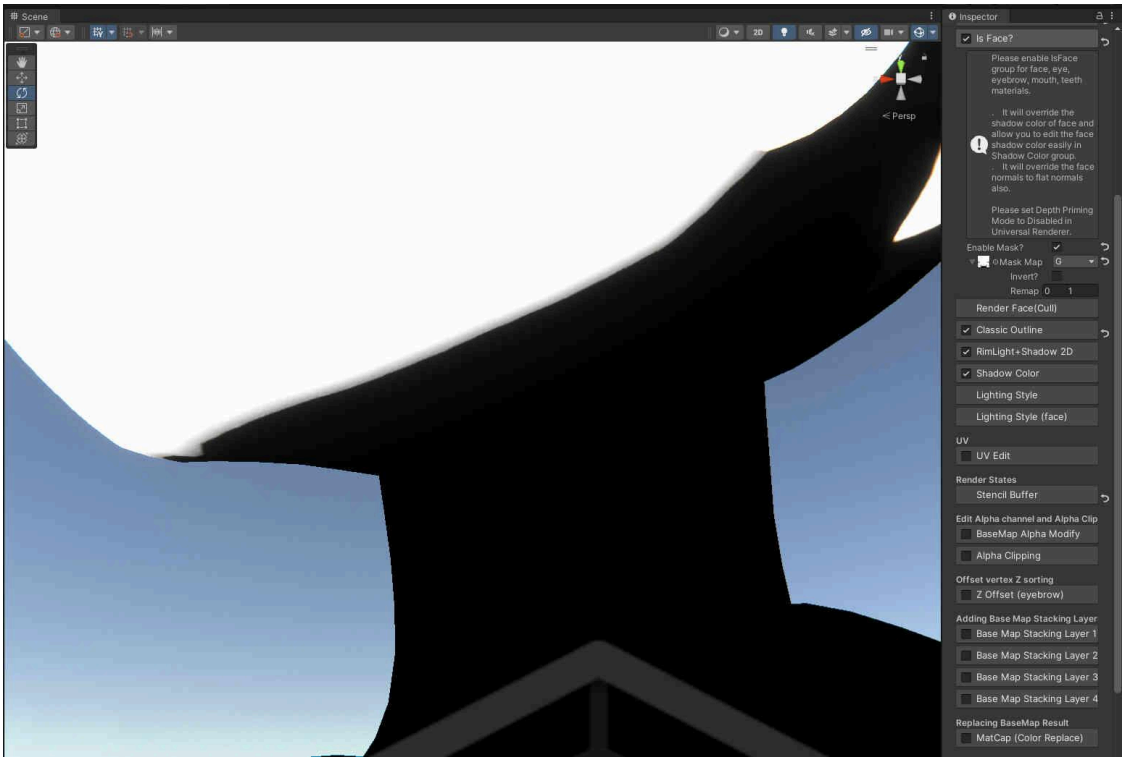
(The above texture is the BaseMap, the shadow color is the **bottom part of the head**)

(The above texture is the extra **Is Face mask map**, where the Shadow color/**bottom part of the head** from BaseMap is converted to black and the rest is converted to white. You can add a little bit blur to it to make the transition smoother)

After assigning the above **Is Face**'s mask map, you should see a better shadow rendering result! You can rotate the directional light to ensure everything looks ok in all lighting conditions



(Image above shows the result after solving the **shadow split problem** by drawing an extra mask map for **Is Face**, which makes the **bottom part of the head** **not** considered as **Is Face**)



(Image above shows the **new debug view** of **Is Face** after the fix, where the **bottom part of the head** is **not** considered as **Is Face** anymore, which usually generates much better shadow rendering!)

Occlusion map

In some extreme light angles, the shadowmap on the neck will not look stable or will look like a sharp cut. In this situation you can add an **occlusion map** with soft gradient to force part of the neck as shadow, which can easily hide the problematic area.

Let's see an example:

(Image above = before adding occlusion map, a sharp split of shadow is visible due to the shape of the 3D model geometry)

(Image above = after adding occlusion map, lighting in all direction will look great now, and the result shadow will mix with shadow map perfectly, this is what we usually do for customer's model for a perfect neck shadow rendering)

(Image above = the new painted occlusion map)

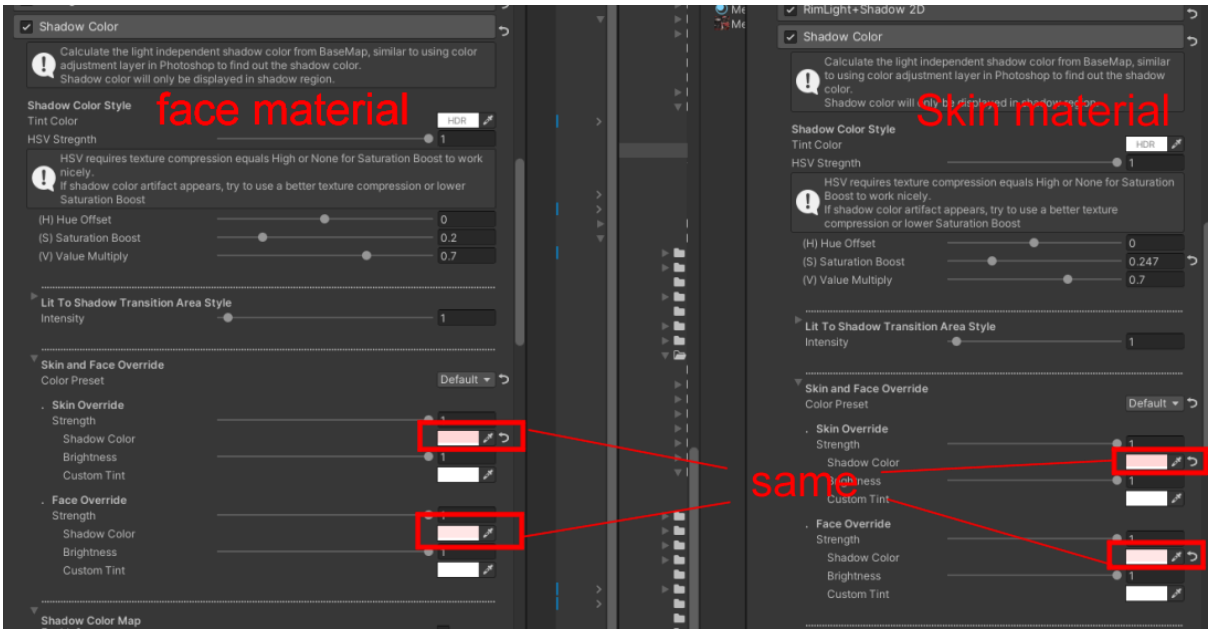
Easy solution

Since the above face mask requires you to paint manually or is good at using tools like PhotoShop or Clip Studio Paint to generate a mask, it may be too difficult for a non-artist to do it. In this situation you can consider the easy solution.

The idea of the easy solution is to do the opposite of the Standard solution we just introduced, easy solution hides the shadow split line by making the neck material become **Is Face** also, it will work well only if there is cloth or accessory covering the lower part of the neck as cover, and you use the neck material's **Is Face** mask to make the shadow split line appears only under the cloth or accessory. In other words, you move the split line to a lower position of the neck, where the split line is now covered by cloth or accessory, so the shadow split problem is not visible to the viewer due to cover.

Sync Shadow Color

If the shadow color of all face and skin materials need to be the same, you should use the same shadow color for all face materials and skin materials, see the image below



Render Face(Cull)

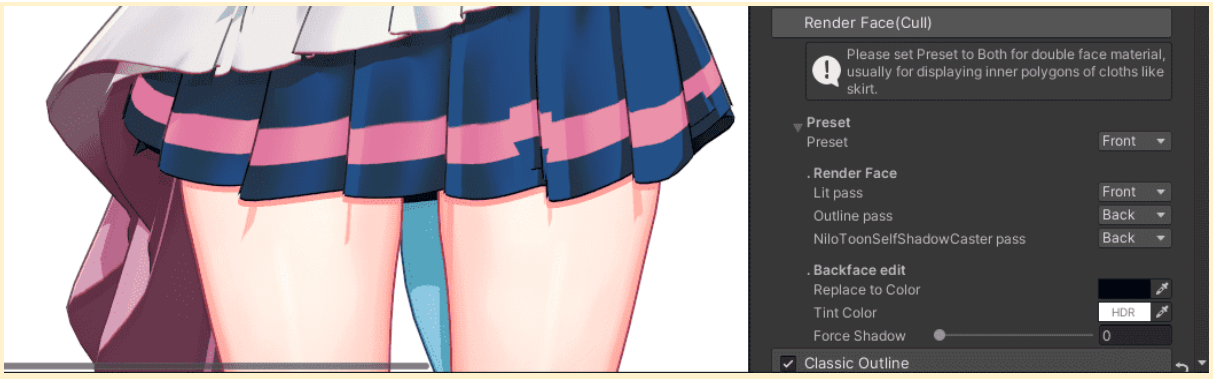
Same as the **Render Face** of URP's **Lit** shader, NiloToon_Character shader has the same **Render Face**. The default preset is **Front**, so only the **front face** of the material will be considered in the **color buffer**'s rendering, and only the **back face** of the material will be considered in the **NiloToon self shadow map**'s rendering.

If the material is for a single-face mesh but wants to produce a double-side rendering, for example, a single-face **skirt or cloth** that is meant to be rendered as double face, you can change the **Render Face** Preset to **Both.**

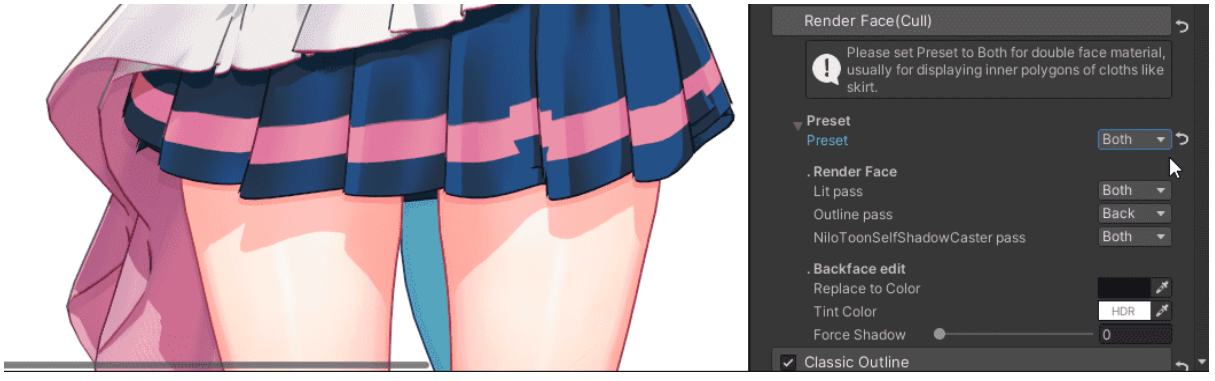
*If your **skirt** or **cloth** material`s **NiloToon self shadow map** looks broken or incomplete (having big holes in shadow area, or shadow is completely invisible), you should try setting **Render Face = Both**, this will render a complete shadow, which will solve the **NiloToon self shadow map** problem in most cases. This is due to **NiloToon self shadow map** rendering **Back face only** by default to reduce Shadow Acne, it has an assumption that the mesh is an enclosed 3D mesh, so single face mesh like skirt will need you to manually edit **RenderFace** to **Both**.

Let's see an example:

- Before fix, **RenderFace Preset** = **Front** (skirt shadow & back face not render correctly)



- After fix, **RenderFace Preset** = **Both** (skirt shadow & back face now render correctly)

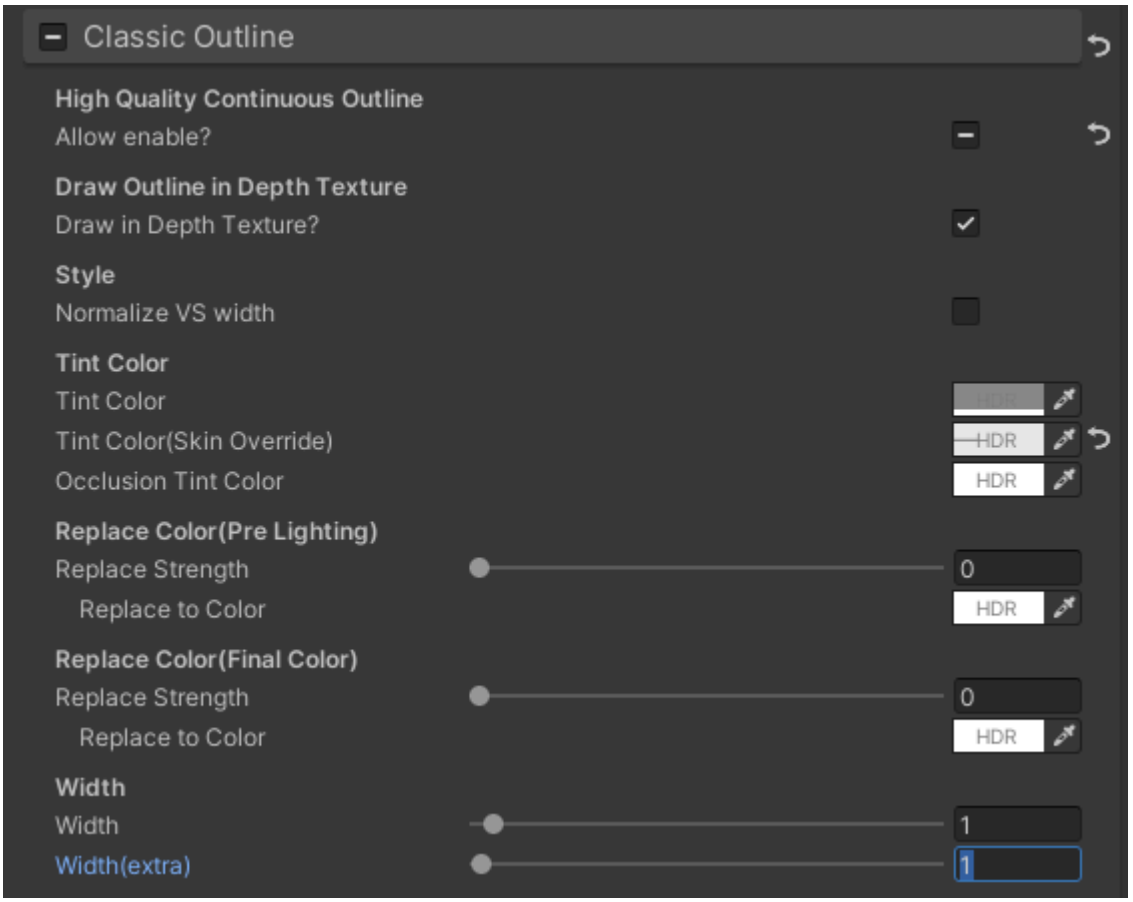


Classic Outline

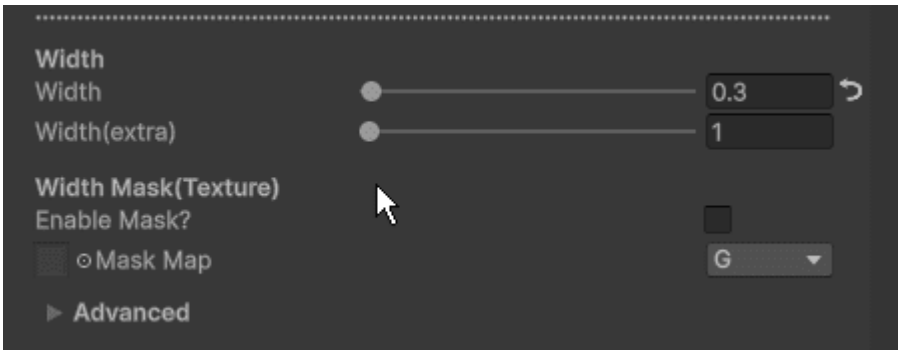
This is an important option, recommend at least setting the Width!

Width

If you can't see outline, make sure in all your material's Classic Outline section, **Width(extra)** is large enough, because **Width**(_OutlineWidth) will preserve values saved in the material by previous shaders, sometimes preserved **Width**(_OutlineWidth) is not directly usable if you are switching from another toon shader(e.g. RealToon/Poiyomi) to NiloToon shaders.



You can also control width by a texture,



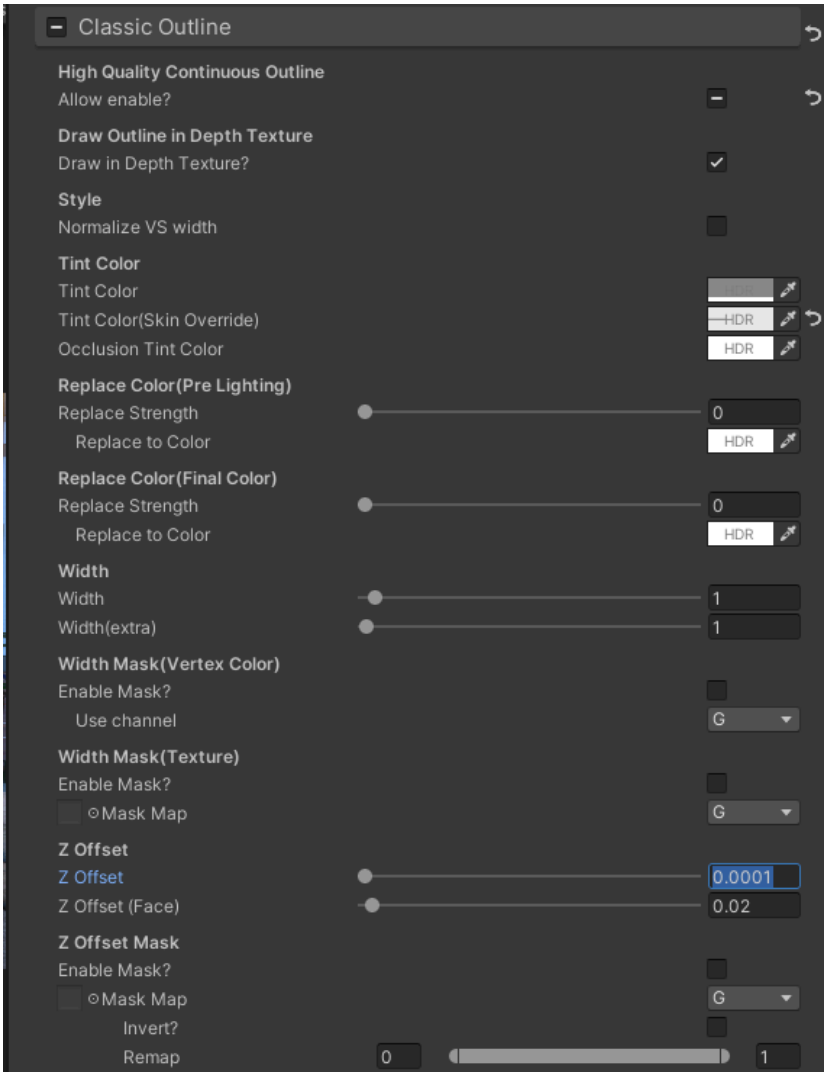
it is important to hide/reduce outlines for certain areas, for example:

- set 0% width for **eye**, never show any outline for the eye!

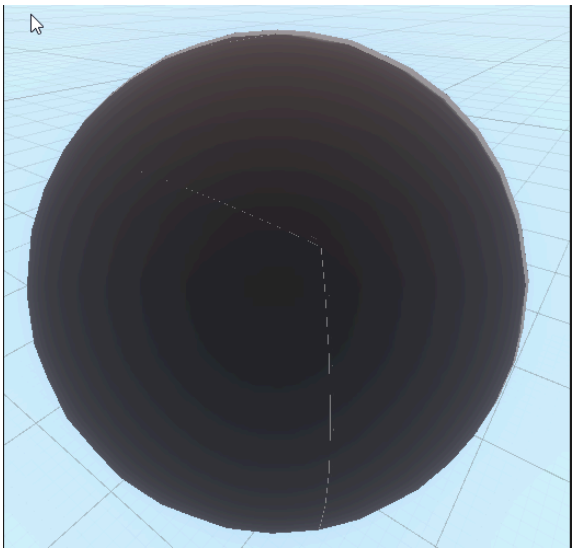
- set 0-20% width for **mouth** and **teeth**
- set 0% for any **hair outline artifact** (highly depends on how the 3D modeler build the hair)
- set 50% width for **finger**
- set 50% width for **front hair**
- set smaller width for **small accessory**

ZOffset

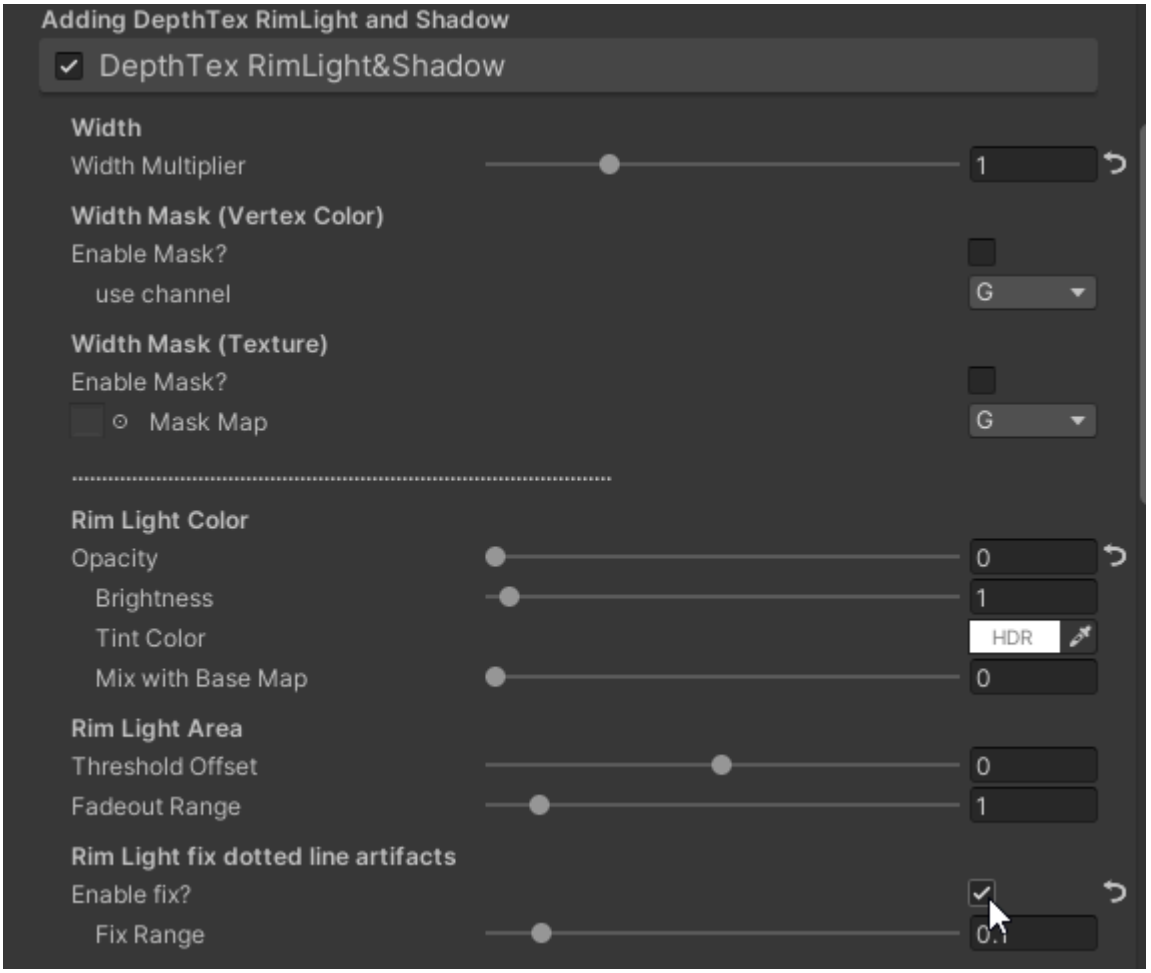
If you want to hide ugly outline artifact, you can also try to increase Classic Outline's **Maskable ZOffset**, you can use a mask map to control where to apply the ZOffset



3.4 If you see weird rim light line on your material,



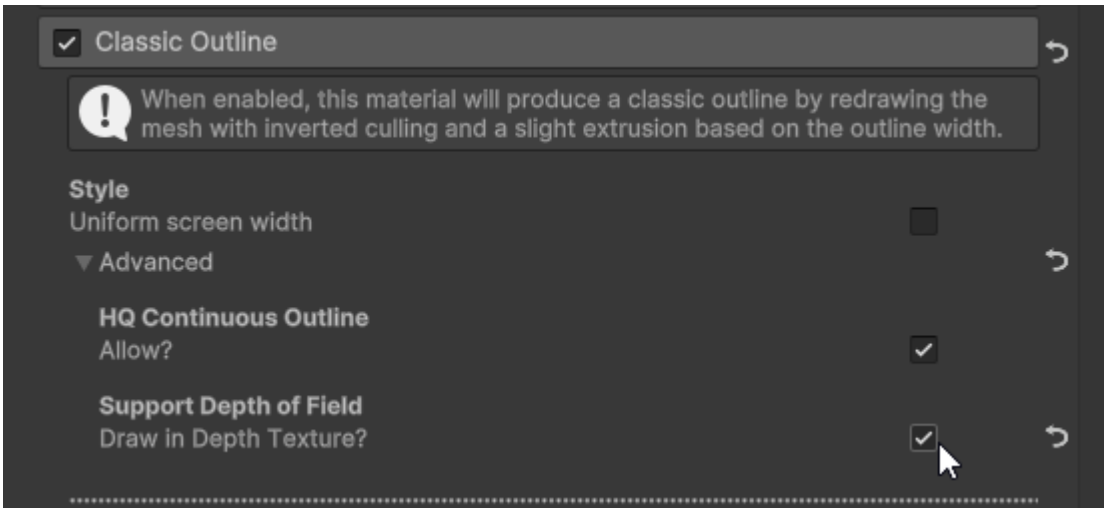
3.4a you can enable **Rim Light fix dotted line artifacts**, it is slower to render, but it can solve the problem



Draw in Depth Texture?

For **Draw in Depth Texture?** in the **Class Outline** section, we recommend **disabling** it if you need a motion vector, for example, when you use TAA/MotionBlur/DLSS (it is disabled by default).

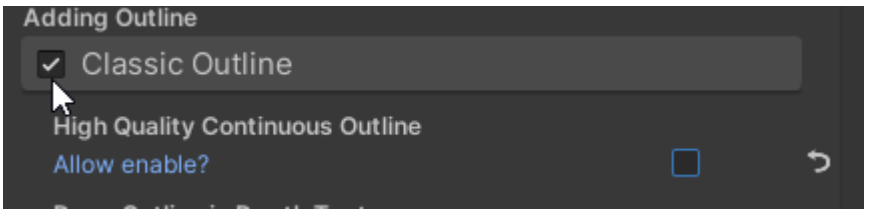
Only enable **Draw in Depth Texture?** if you don't need motion vector, and want effects like Depth of Field to perform correctly



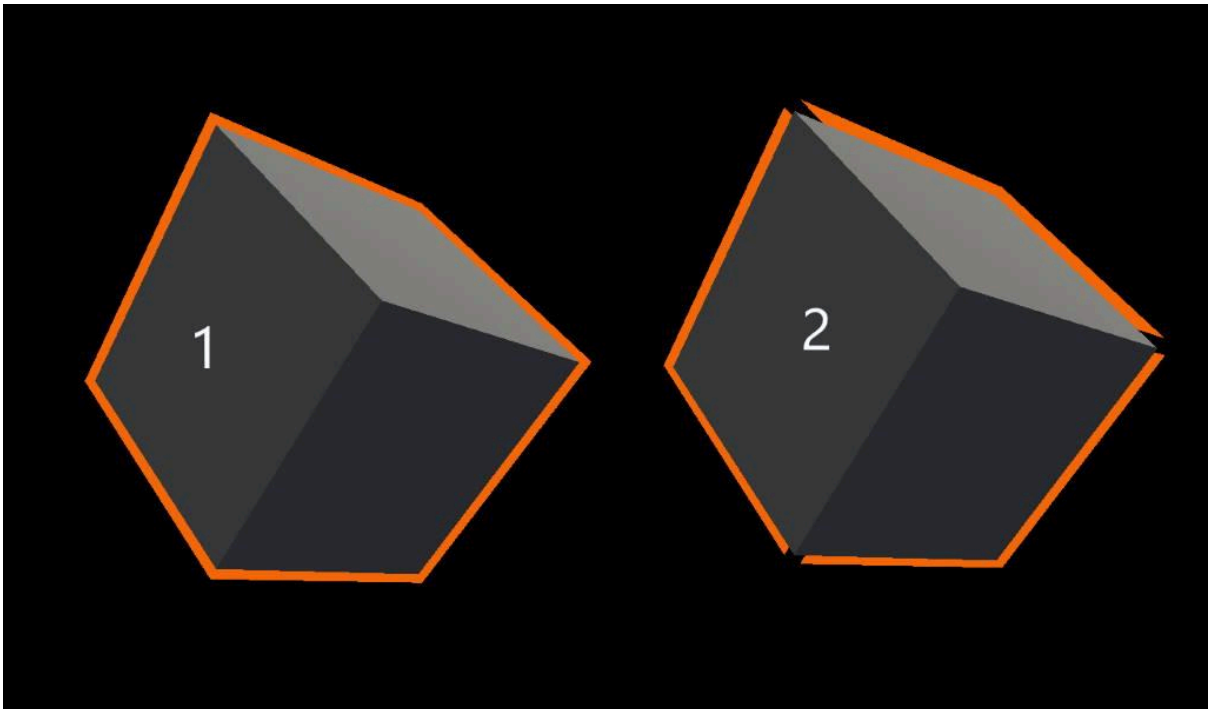
Disabling **Draw in Depth Texture?** will always solve the problem, but material will not write outline's depth into URP's _CameraDepthTexture, which makes post process effects like **Depth of Field** not correct on outline pixels.

Continuous Outline

If you end up still having an outline problem, you can try disable **High Quality Continuous Outline**(_OutlineUseBakedSmoothNormal)



The outline will always render correctly if you turn **High Quality Continuous Outline**(_OutlineUseBakedSmoothNormal) off, but ugly breaking outlines may appear at corners of a polygon(e.g., when turned off = cube2).



GuiltyGear CEDEC2024 note

CEDEC 2024 has a talk about improving outline, it can be useful if you are setting up the outline material. NiloToon supports all features from this talk in the **Classic Outline** group of the material..
「ギルティギア」シリーズ、輪郭線を自在に操る「背面法」実装テクニック【CEDEC2024】 - GAME Watch

In summary:

1. NiloToon_Character shader will auto adjust outline width according to distance to camera, you don't need to do anything (clamped at certain distance, in order to prevent outline that is too thick)
2. NiloToon_Character shader will auto adjust outline width according to fov, you don't need to do anything

3. You can adjust material outline width by a 0~1 value from vertex color or texture (Great for hiding eye/teeth/small object outline)
4. you can adjust material outline ZOffset by a 0~1 value from vertex color or texture (Great for hiding bad outline of hair, mouth, arm/leg skin, dynamic animated cloth)

Also, the method "closing outline" at the end of the slides means reducing outline width on specific vertices manually to "close" the outline, preventing floating outline, it can be time consuming and requires skill to do it, but can produce the best result of classic outline. (In practice, very few do this since it is not easy, but if you aim for best quality outline, you can do it for visible floating outline)

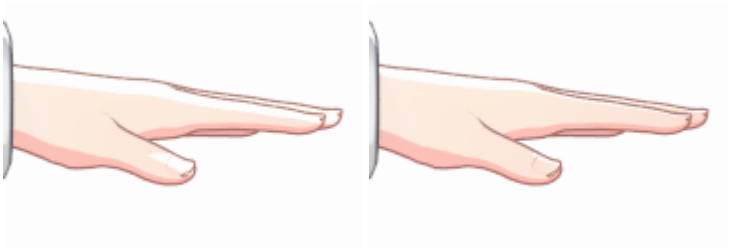
RimLight+Shadow 2D

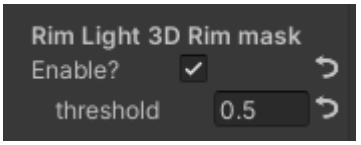
For objects like hand & finger / glove / small accessory, sometimes the 2D rim light may not be suitable for them.

For materials where a thinner, non-uniform width rim light is preferred, particularly for **hands**, **fingers**, and **gloves**. We highly recommend enabling **3D Rim masks** to improve the rim light result, you can find the settings here:

- RimLight+shadow2D > Rim Light 2D > Setting > Advanced > Rim Light 3D Rim mask

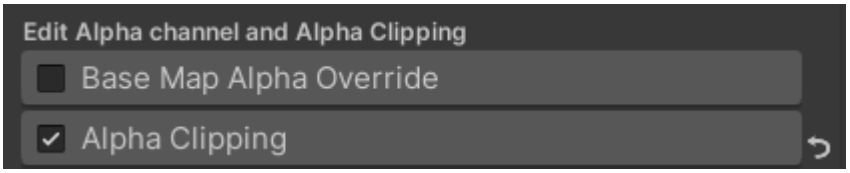
which will produce a **more natural rim light** in most cases, we usually at least enable it for **hands**, **fingers**, and **gloves**.



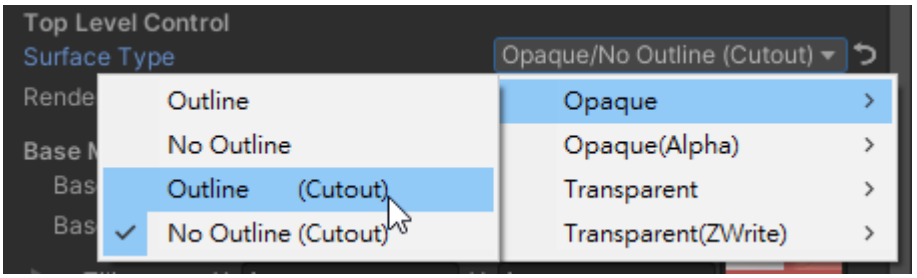


Alpha Clipping

If the BaseMap contains alpha data for alpha clipping, you can enable **Alpha Clipping** toggle



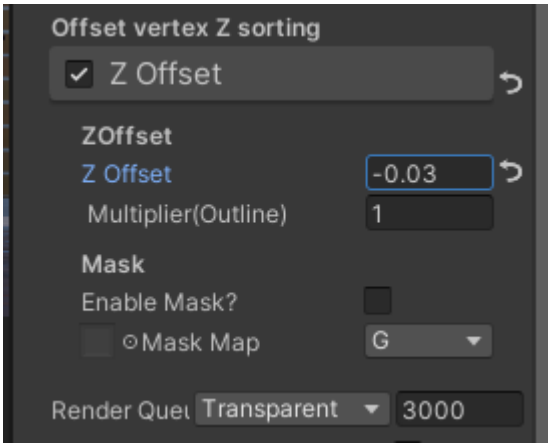
or select a new **Surface Type** with **(Cutout)** suffix



Z Offset (eyebrow)

(this step is optional, depending on the art style you want)

If the material is an eyebrow material, you can drag ZOffset to around -0.03 or more, this will make the eyebrow render over the hair.



- If you want to render semi transparent eyebrow on hair, see this

Shadow Color

Important Setting!

If you don't like the default color of the shadow, you can control shadow color in the **Shadow Color** group.

For **non-skin and non-face** area, you can edit these params:

- Tint Color
- HSV Strength
- H
- S
- V

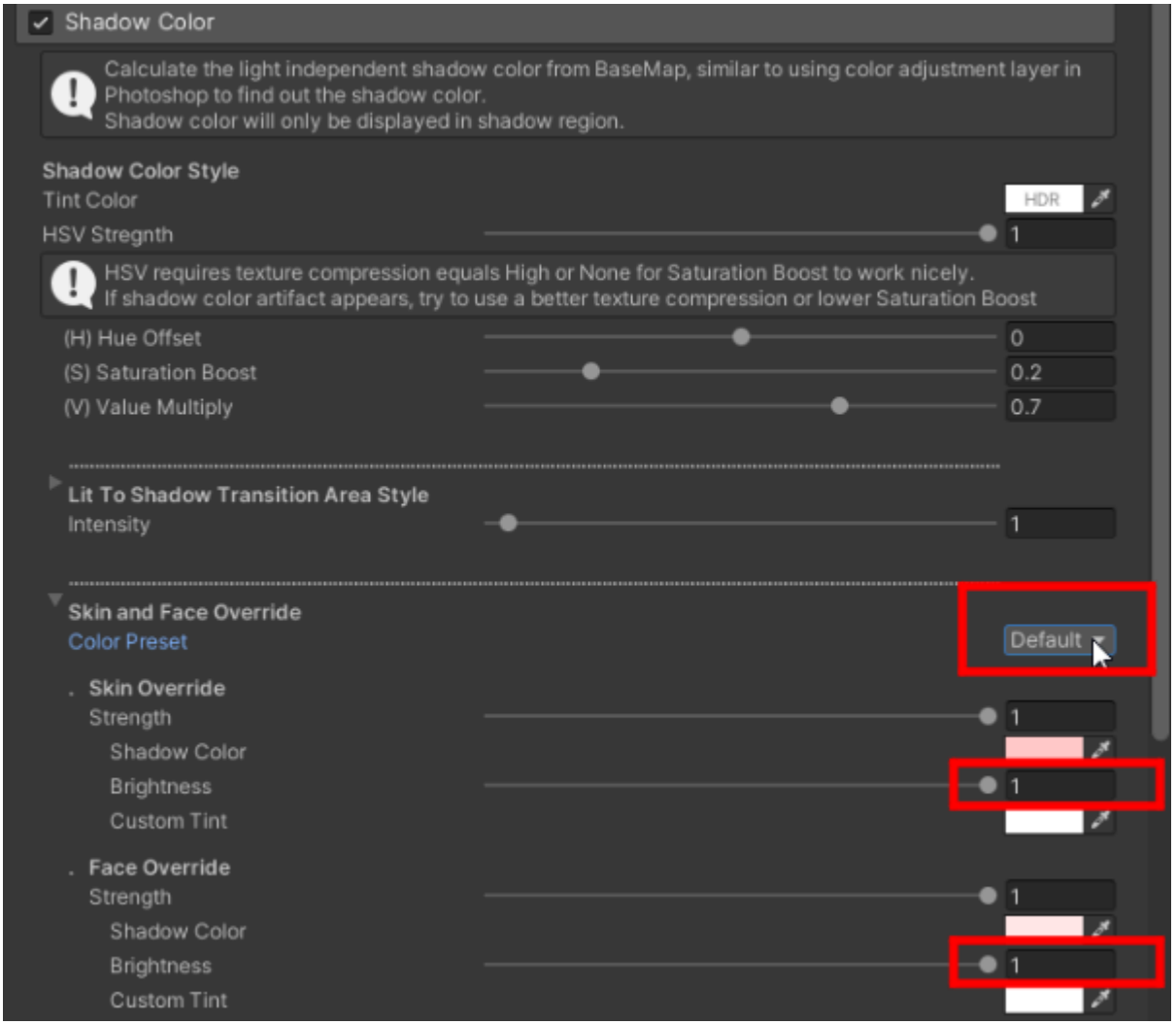
For **skin and face** area, you can edit the settings highlighted in red:

- Color Preset
- Brightness (Skin Override)
- Brightness (Face Override)

*We highly recommend picking a good skin shadow color using the character's 2D art as a color reference

*The settings highlighted in red are only for overriding **skin and face** shadow color.

*Common range of **Brightness** is 0.8~1

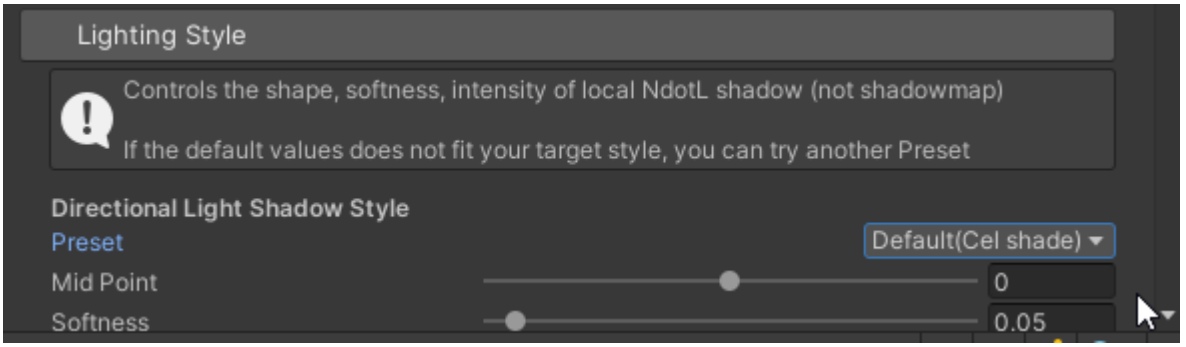


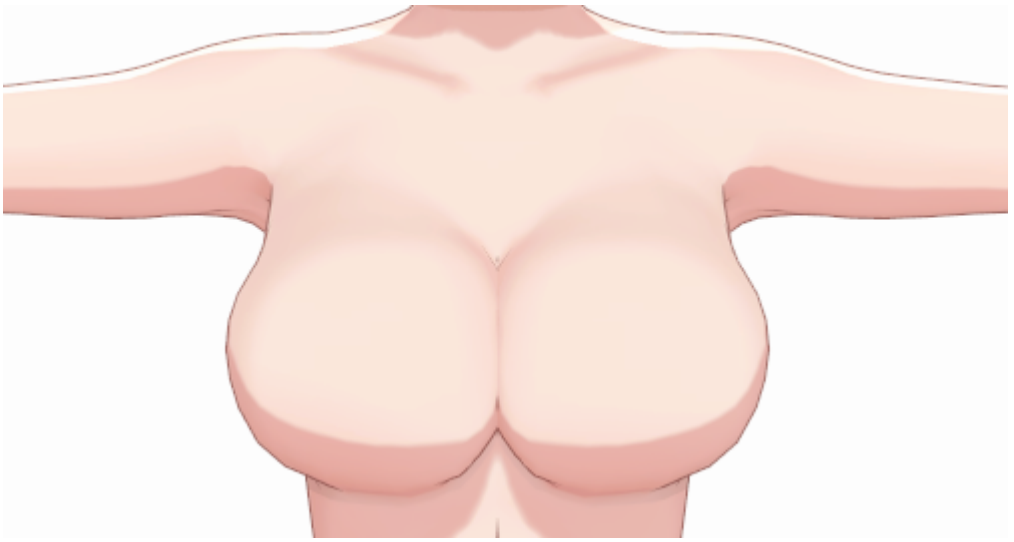
Lighting Style

Important Setting!

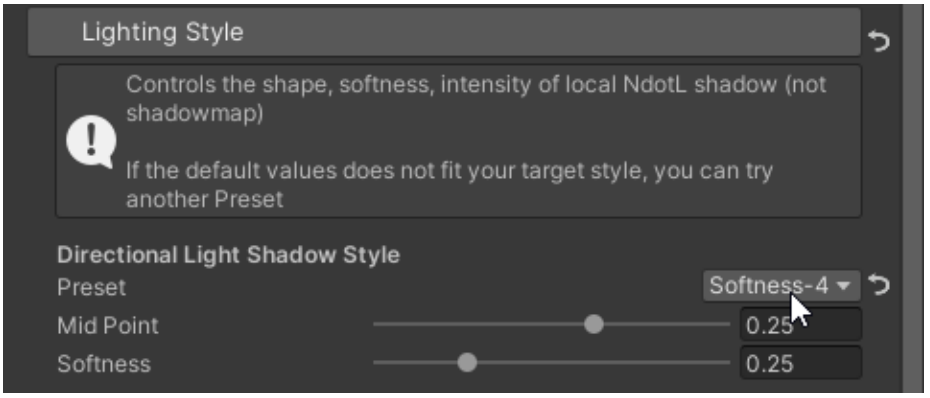
If you don't like the default cel style of the shadow, you can control the shadow style/softness in the **Lighting Style** group, you can simply pick a **preset** to control **shadow softness** (e.g., if you want the shadow become softer with more gradients, pick a preset with a **higher softness**.

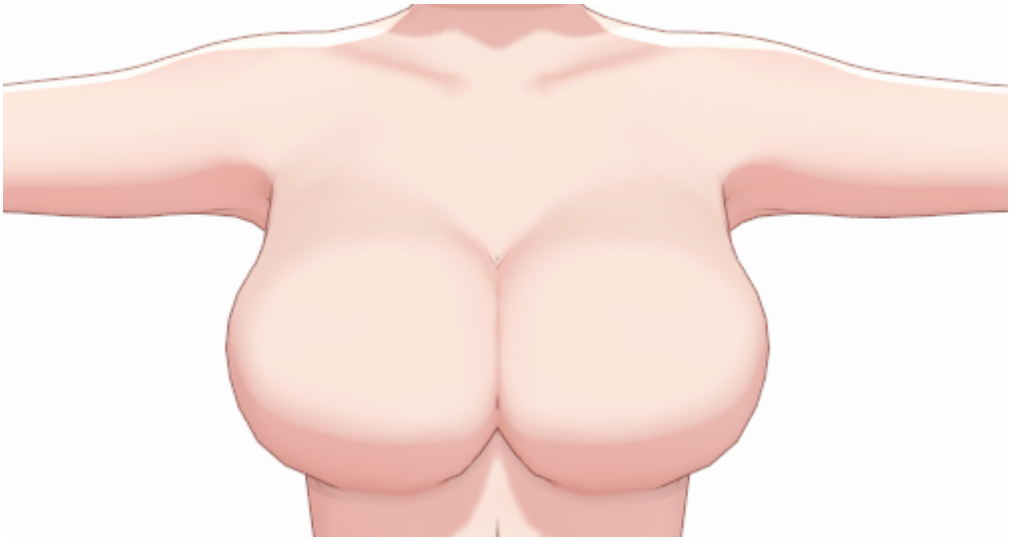
Lighting Style is an important feature if you want a softer look of a material, for example, softer **hair / skin / fur /cloth**…



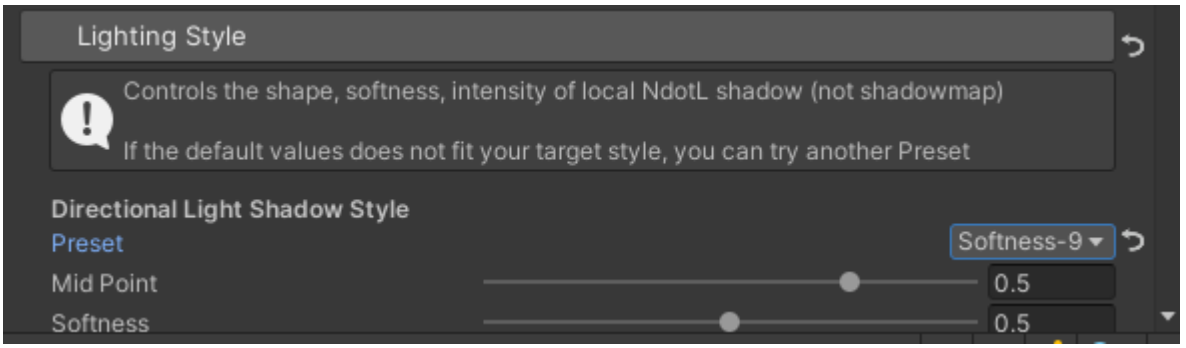


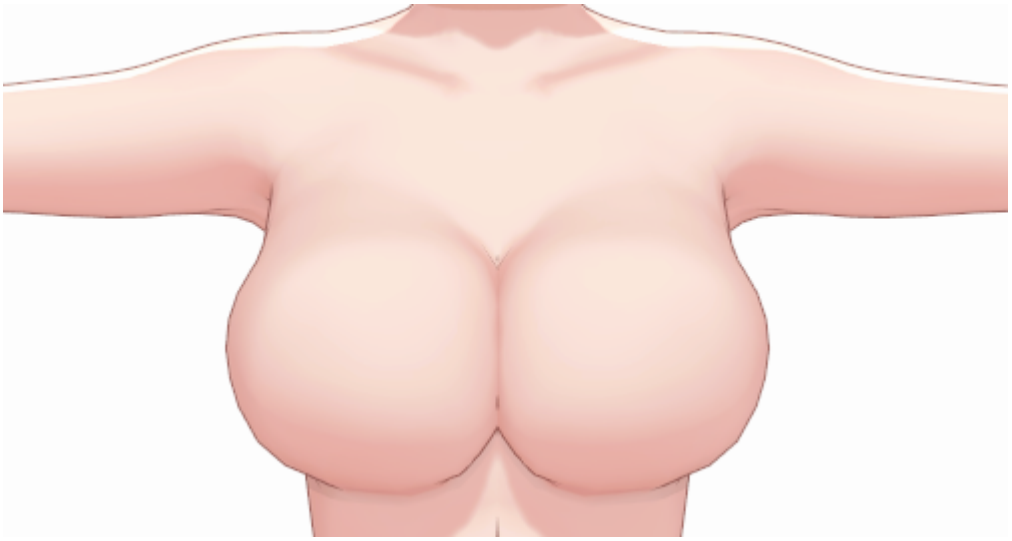
Picture above is the default lighting style (**Cel shade**)





Picture above is a slightly softer version lighting style (**Softness-4**)





Picture above is a very soft version lighting style (**Softness-9**)

Softer shadow preset in Lighting Style is also important if you want to produce a more **realistic** material, these videos have a soft shadow preset in the Lighting Style for your reference:

[NiloToon Unity URP shader] Girls Frontline 2 Exilium - Sharkry rendering

[NiloToon Unity URP shader] Girls Frontline 2 Exilium (2023 CBT3) - MP7 Che…

[NiloToonURP] Girls Frontline 2 Exilium (2023 CBT3) - Vepley rendering

Softer **fur** rendering:
普普亞on X: "咪咪國王@youjomiruku、@youjo_miruku的一些動作測試

Softer **skin** rendering:
https://youtu.be/cVw0KQ7q1yY?si=EEnVBPYnJlAp9YCR

Soft **skin** and **cloth** rendering together:
【粛聖!! ロリ神レクイエム☆】パトラちゃんに罵倒してもらうだけ #vtuber #周防パトラ #shorts - YouTube
粛聖!! ロリ神レクイエム☆ / 周防パトラ（9さい＋81××さい）cover

2-pass transparent

If you have a single material with both opaque and alpha parts combined together(e.g. cloth, eyeglass, hair etc), it may be difficult to draw this material correctly without special care, since the drawing result will depend on the triangle's order, which means the result can be random depending on the camera viewing angle. A solution that produces a similar result to lilToon's **2-pass transparent**, is to use **NiloToonRendererRedrawer**.

Steps to solve the problem, use the following method to produce a 2-pass transparent:
1. In the original material, set:
    - SurfaceType = **Opaque(Alpha)/Outline** or **Transparent(ZWrite)/Outline**

2. then create a material variant of (step1)'s material, set:
    - SurfaceType = **Opaque/Outline (Cutout)**
    - with alpha Clipping's **threshold = 0.99~1**
    - make sure **render queue** is earlier(smaller) than the original material in step 1

3. attach **NiloToonRendererRedrawer** to the renderer, redraw step (2)'s material variant
4. done, you should be able to produce a result similar to lilToon's 2-pass transparent now

The idea is to:
1. Make the material variant in Step2 draw only the pixels with alpha ~= 1 first, any pixels with alpha < 1 will be removed in the material variant due to alpha cutout(alpha clipping)
2. Then draw the original material in Step(1) as usual(by Unity's renderer)l, since the depth buffer is filled by Step2's material variant already, the chance of the original material drawing the correct result will be much higher.

If using **NiloToonRendererRedrawer** is causing problems or not allowed, you can append extra material to a renderer if the target material is the last one of the material list, see this for details.

Wrong rendering

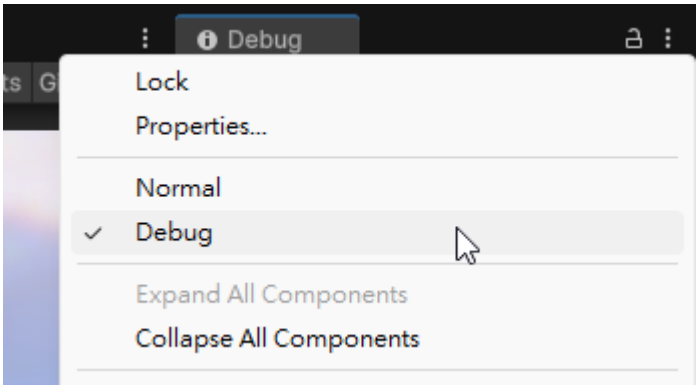
If after the auto material conversion to NiloToon, one of the following happened:

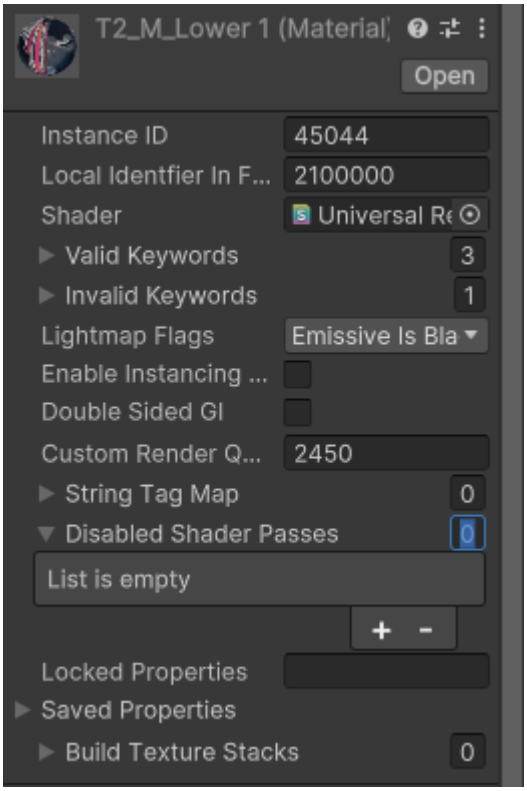
- the **2D rim light** looks completely wrong
- missing **URP shadow caster** pass

It is likely that your material disabled some shader pass automatically by the Unity fbx importer, for example:

- **DepthOnly** pass
- **ShadowCaster** pass

Try to enter the **debug mode** inspector, and reset the **Disabled Shader Passes**.





Sometimes the Unity fbx importer will disable shader pass if the importer considers the material is transparent, but you use it as Opaque or Opauqe(cutout), which produce a 2D rim light. In this situation the material will have **Disabled Shader Passes** and you need to reset manually in the debug mode inspector.

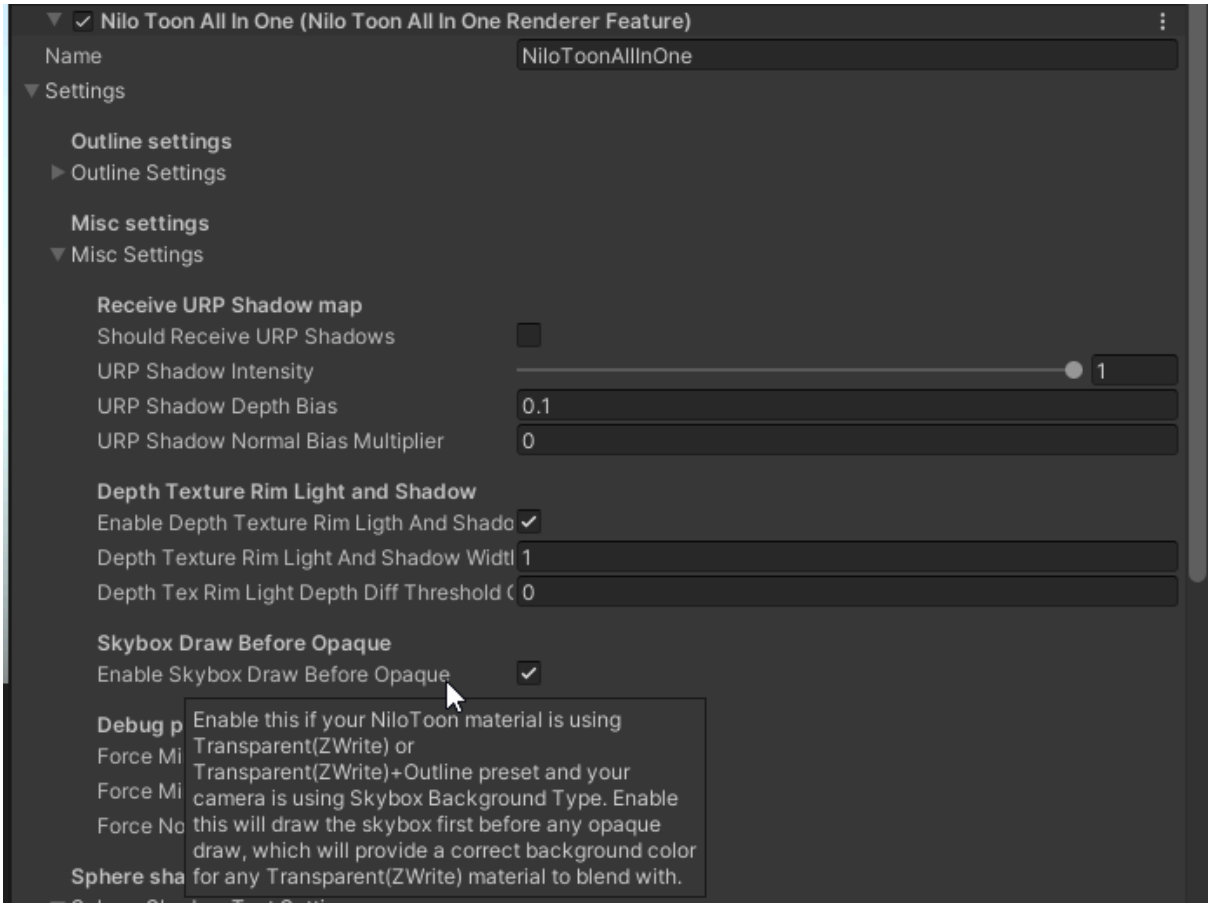
Non-material settings

Renderer feature

There are some useful settings in the NiloToonAllInOneRendererFeature, here are some of them.

Skybox AlphaBlend issue

If any transparent material's blending background is **Skybox**, due to URP's skybox rendering timing, result may flicker/wrong if your material renders **before** skybox (before render queue 2500). You will need to enable `EnableSkyboxDrawBeforeOpaque` in NiloToon's renderer feature to provide a valid background for these materials to blend with.



If you don't need this skybox redraw, you can turn it off to improve performance.

Receive URP Shadow

If you want your NiloToon character to receive shadow map shadow from other shaders(e.g., Lit shader / shader graph shader), you can try to enable **Should Receive URP Shadows** in NiloToon's renderer feature. This will make NiloToon character to receive the regular URP shadow map (receive shadow from any shader that supports ShadowCaster pass)

*By default, **Should Receive URP Shadows** is disabled, due to **NiloToonSelfShadow** is enabled

If you want to avoid character's self URP shadow and only receive URP shadow from far objects, set a high **URP Shadow Depth Bias**(e.g., set 1 = 1m), this can avoid most of the URP self shadow by the character, but still allows your character to receive shadow from far objects (e.g., a building / house / tree / bus / flying birds)

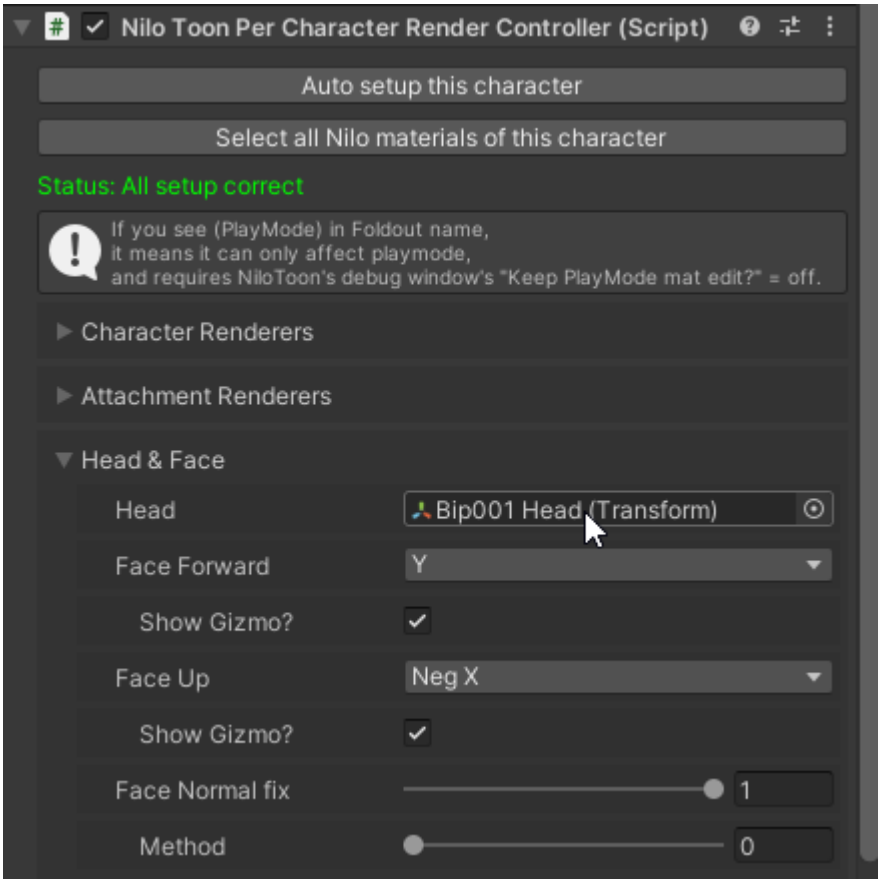
If all the settings are correct, you should see the URP's lit shader cast shadow on the NiloToon character, like this https://youtu.be/gtgME6MJpk4?si=h7_lCwAnVRYD7yV0&t=82

Per Chara script

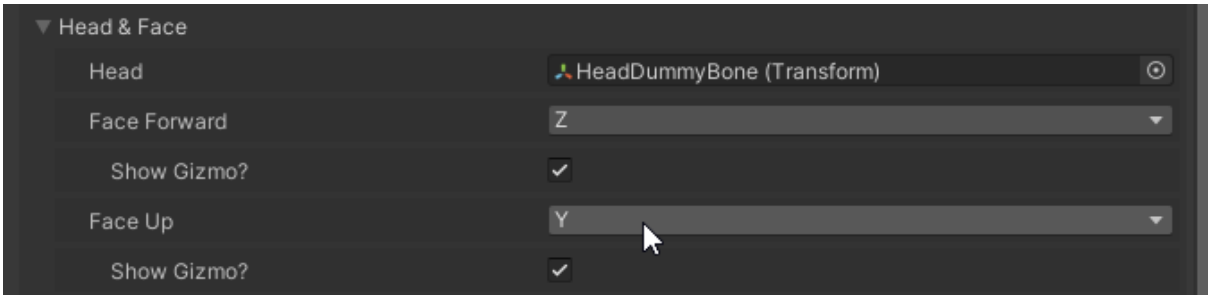
NiloToon requires knowing the character's Head bone and Face direction for shading.

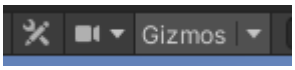
Setup bones &bound

If it is not yet set up correctly, drag the character's head bone to the Head slot.
*In most cases the **Auto setup** should did that for you already



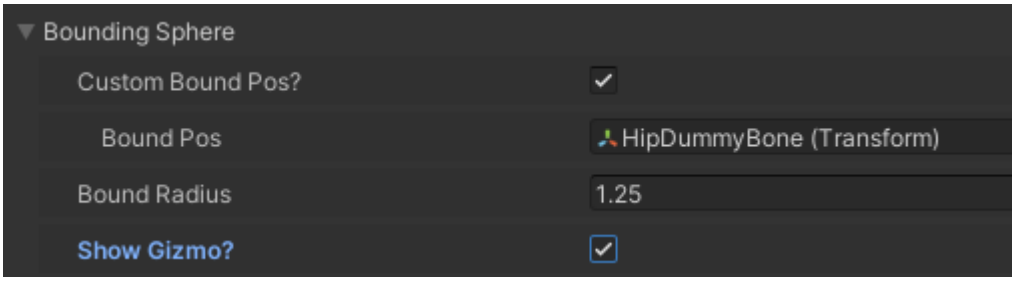
Edit **Face Forward** and **Face Up** until the 3D blue arrow is following the character face's forward direction and the 3D green arrow is pointing upward
(you need to enable **Gizmo** for the scene window in order to see the blue and green arrows).
*In most cases the **Auto setup** should did that for you already

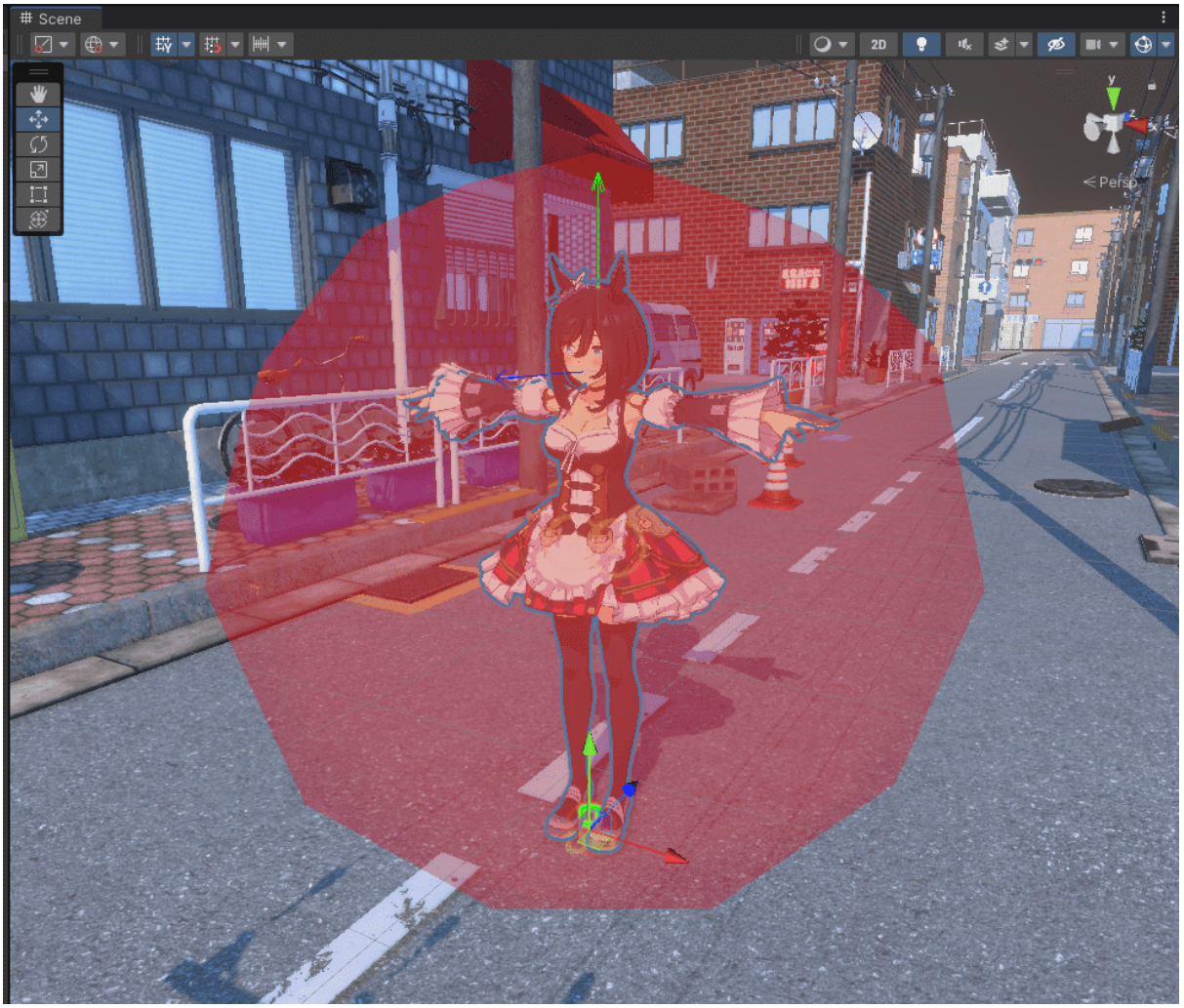




Drag character's hip / pelvis bone to Bounding Sphere's **Bound Pos** slot,
*In most cases the **Auto setup** should did that for you already

Edit the Bound Radius until the red gizmo sphere tightly contain the character
*In most cases the **1.25M default radius is enough** already for a normal size human character



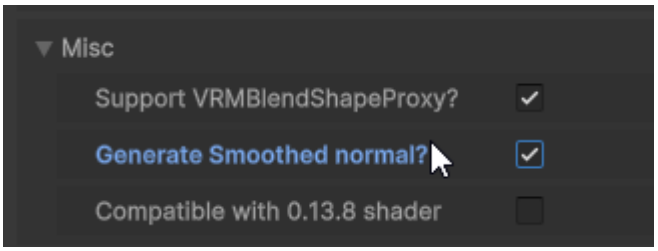


VRM HQ outline

- For fbx, NiloToon will bake a smoothed normal automatically by default.
- For vrm, NiloToon will not bake a smoothed normal by default

If the outline of the vrm character prefab is not looking good (e.g., split or broken outline on hair or hard edge), you can try to generate a smoothed normal outline for the vrm mesh in Runtime.

When you enable **Generate Smoothed normal**, NiloToon will generate outline data OnAwake. If your vrm model doesn't have any special vertex normal for outline, likely you should enable this option to improve **Classic Outline** quality



Summary

Per character setup DONE!
You can now play the scene and try editing the param of:

- NiloToonPerCharacterRenderController (that script on the root of your character)
- character material's setting
- Volume's NiloToon_____Volume setting

Additional Light settings

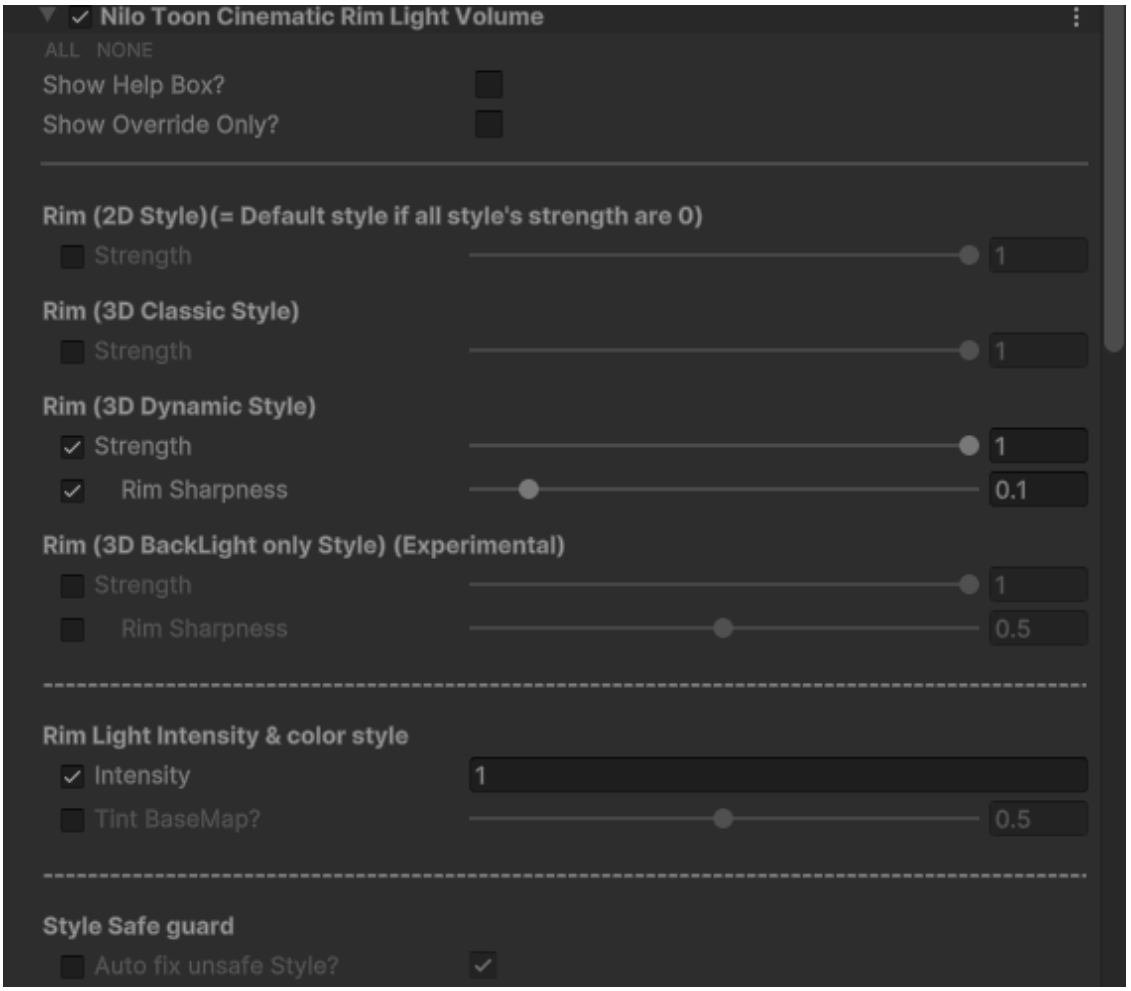
NiloToon by default will only render a **2D rim light** on the character, if you need to produce a **3D soft rim light** produced by additional light, read the section below about **NiloToonCinematicRimLightVolume**.

Dynamic style rim light

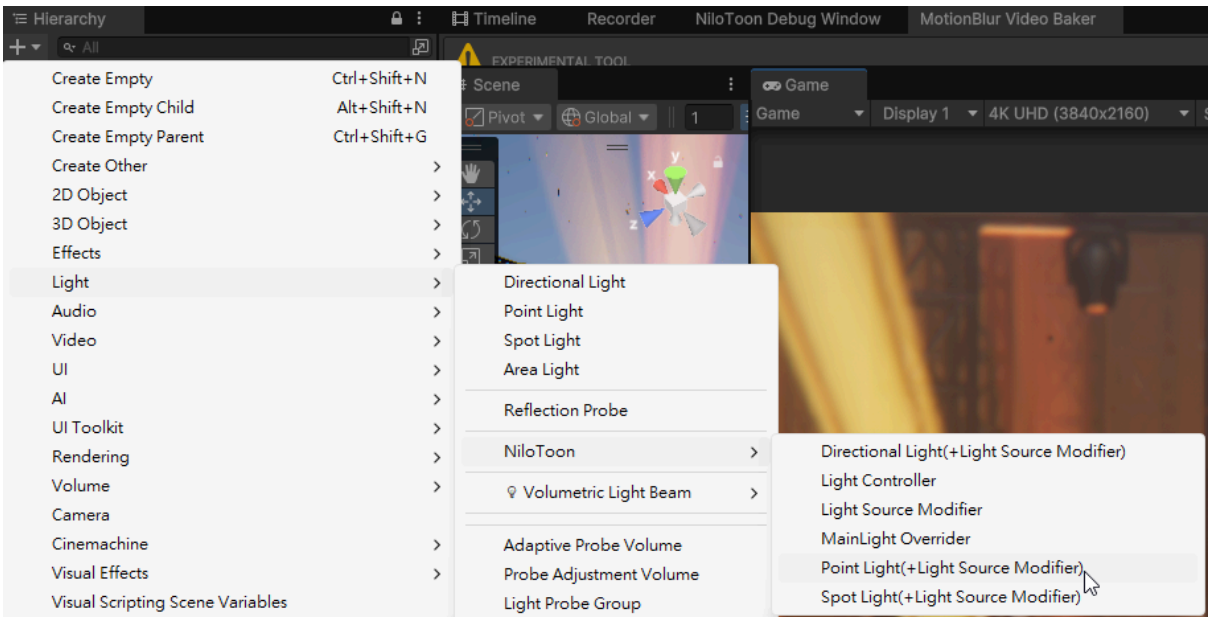
If you need to produce additional 3D style soft rim light, for example, you want the character to receive soft rim light produced by a spot/point light/additional directional light, you can do the following, but let's see a few examples first to understand what the soft rim light effect is:

- 粛聖!! ロリ神レクイエム☆ / 周防パトラ（9さい＋81××さい）cover
- 【#空澄セナ4周年記念LIVE】～私がここにいる理由～【空澄セナ/ぶいすぽっ！】
- 〖＃夜乃くろむ3D 〗これがOTONAの魅力ってやつだよ。の巻〖 ぶいすぽっ！ / 夜乃くろむ 〗
- 【 #小雀とと生誕記念3Dライブ 】～森に響く音色～【 ぶいすぽ / 小雀とと 】
- [청백가요대전] Shhh - 냐냐 베베리 왕꿈틀이 한결 (쾌이엇) KISS OF LIFE 4K

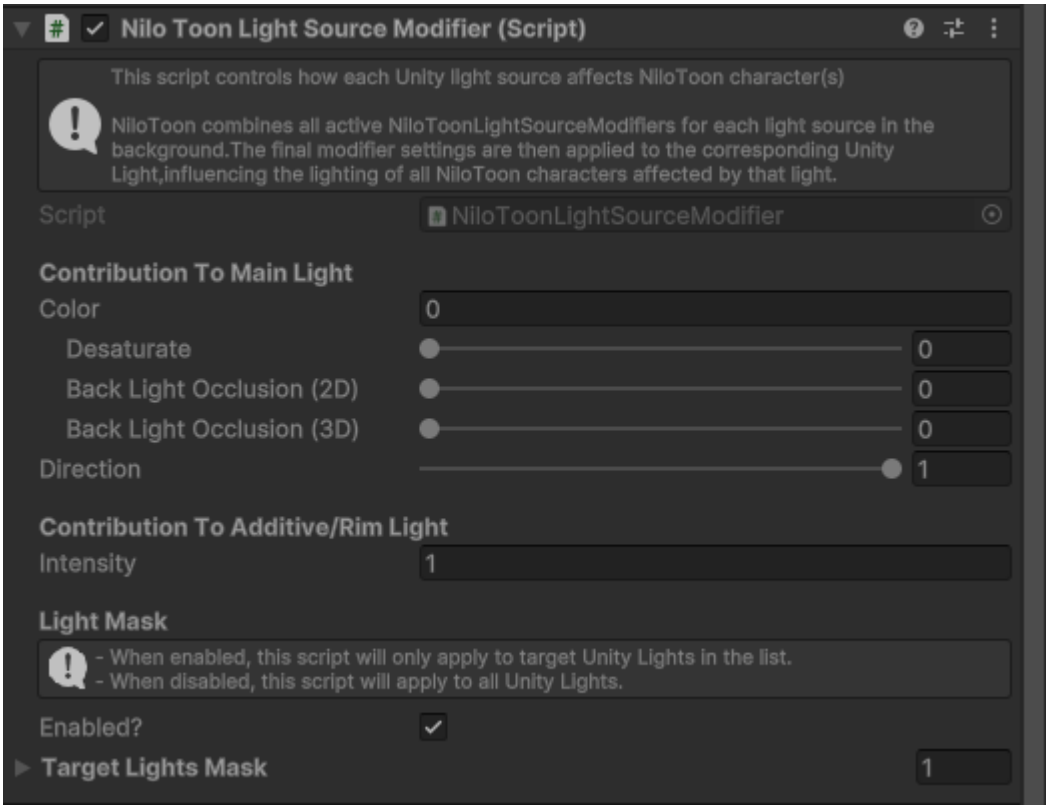
1. Setup **NiloToonCinematicRimLightVolume** in your volume, override the **3D Dynamic Style > Strength to 1**. The style will try to render a Left/Right side rim light when the character is lit by spot/point light/additional directional light.
   *You can freely adjust **Rim Sharpness** and **Intensity** for your target art style



2. Add a new **Spot/PointLight(+Light source Modifier)** using this menu, place the light to a location so that the light can affect the character



3. Use these settings for the **NiloToonLightSourceModifer** script attached to the light. If you don't want the light affecting main light direction, set **Direction** to **0**



4. Setup done, now this light will produce a soft 3D style rim light, you can also enable **soft shadow** or **light cookie** for this light to produce improved lighting effects

Concert style back light

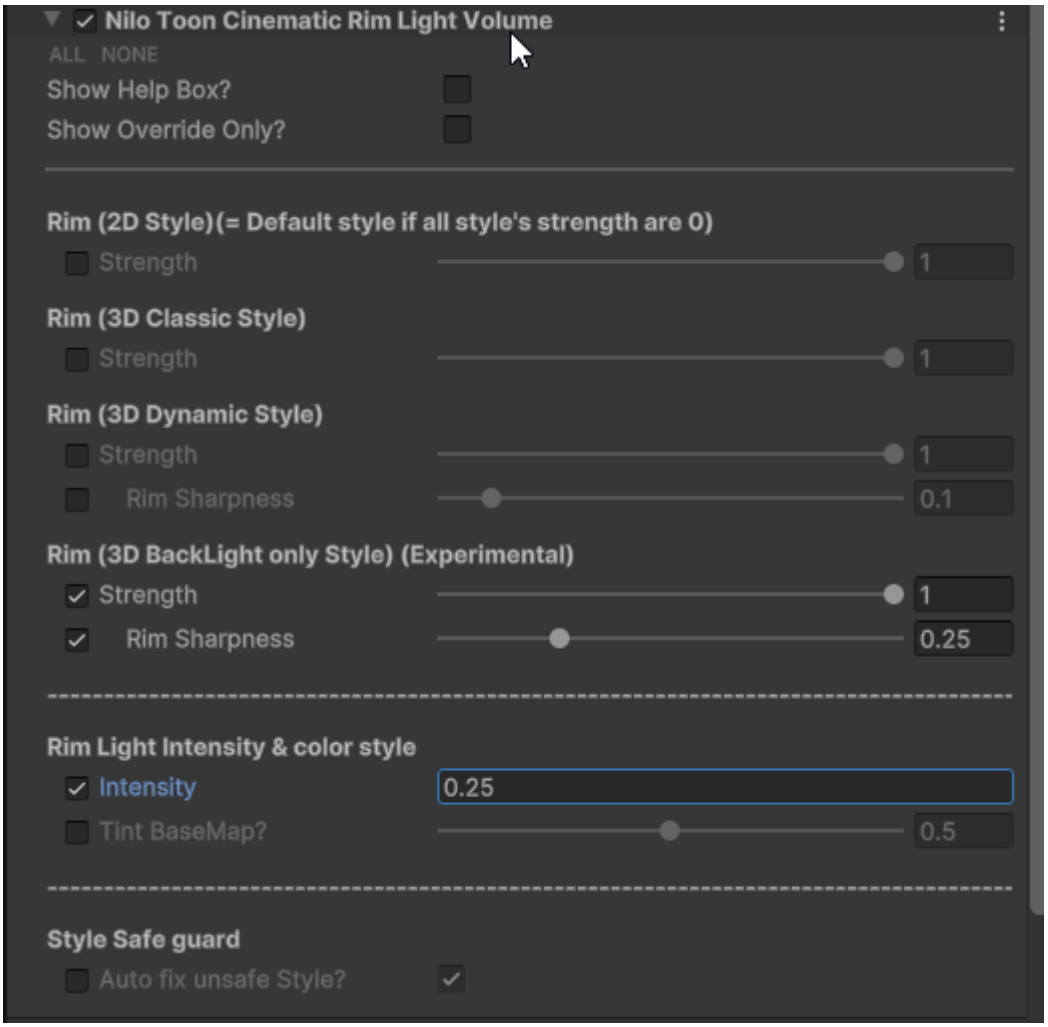
For concert 3DLive / MV that has many animating spot lights at the back of the character, we usually use the **3D BackLight only Style** in **NiloToonCinematicRimLightVolume**.
Examples:

- NiloToon Unity6 Concert Demo MV
- BOOM - Evil (Official Video)
- Snow halation 이세계아이돌 COVER| 러브 라이브! OST

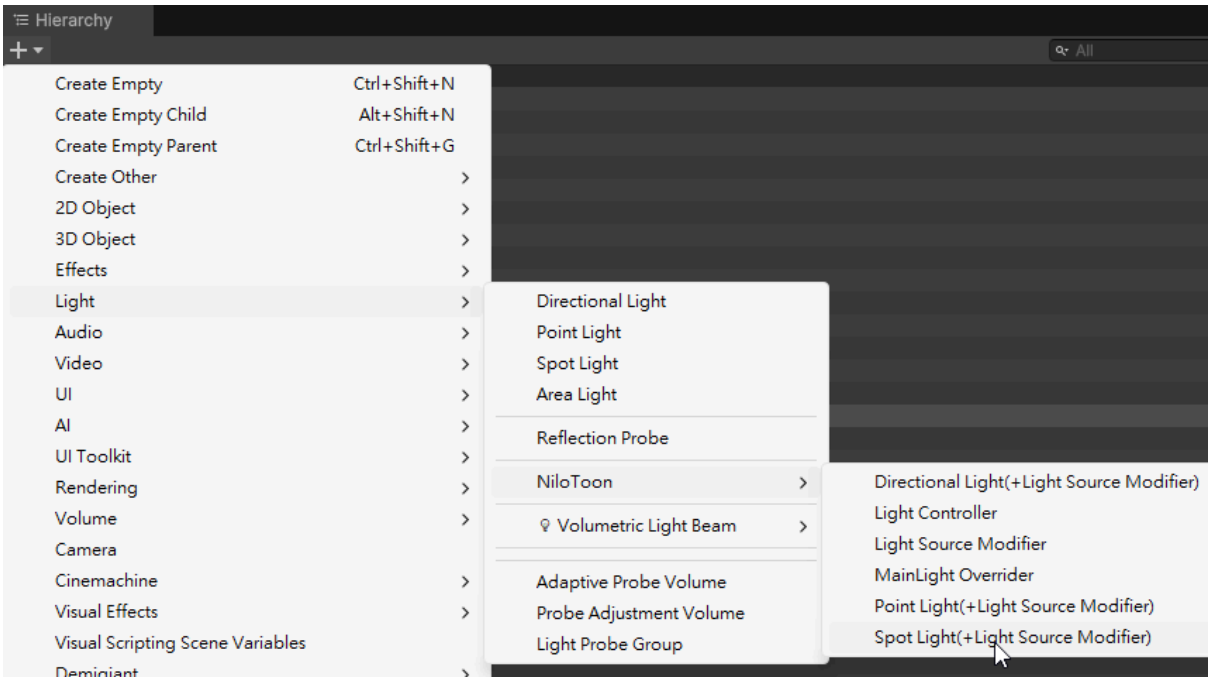
This style will render a realistic rim light / back light **only when the light is behind the character** (in camera's view), same as real-life rim light / back light. It works extremely great for 3D concert's flashing and animating back light (spot light).

To use this style:

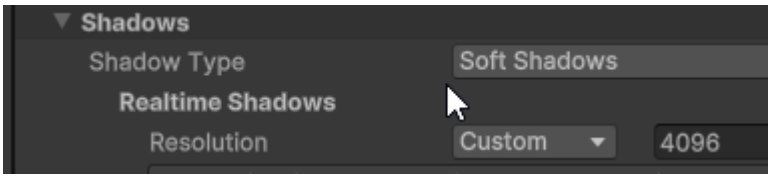
1. set up **NiloToonCinematicRimLightVolume** in your volume, override the **3D BackLight only Style > Strength to 1**.
   *You can freely adjust **Rim Sharpness** and **Intensity** for your target art style



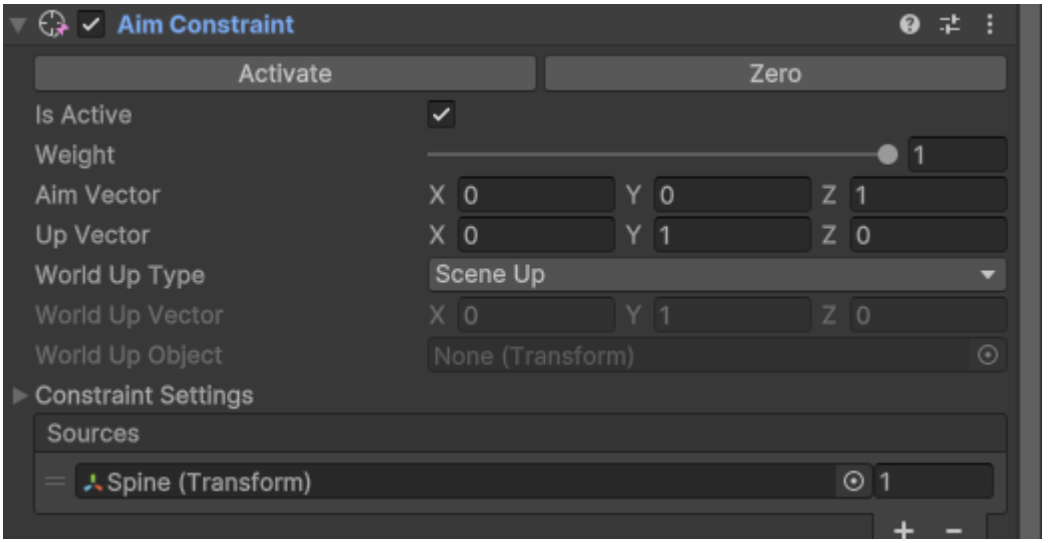
2. Add a new **SpotLight(+Light source Modifier)**, place it behind the character.



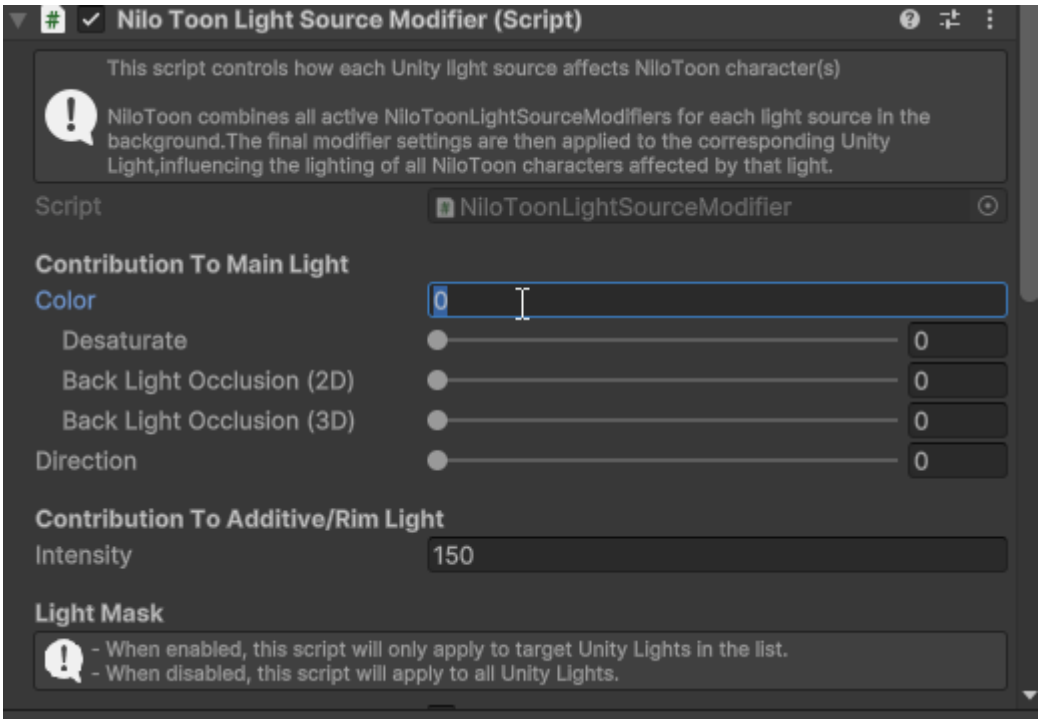
Enable soft shadow with at least 2048 resolution, shadow is important to produce a realistic rim light, preventing light leaking to the front of the character.



3. You can add the Aim Constraint script to force the light always aim at the target character's bone, so no matter how the character moves, your back spot light will always aim at the character to produce a stable back light / rim light



4. Use these settings for the **NiloToonLightSourceModifer**.



You likely don't want the back spot light affecting main light color and direction, so we recommend setting both **Color** and **Direction** to **0,** this will make sure this light only produces back light/rim light and **not affecting any main light's lighting**.
*You can freely edit the Intensity per light

5. Animate the above spot light in timeline to produce a flashing back light, using tools like https://github.com/murasaqi/Unity_StageLightManeuver

All the above settings are suggestions only, feel free to use any setting that work for your project

PC project setting example

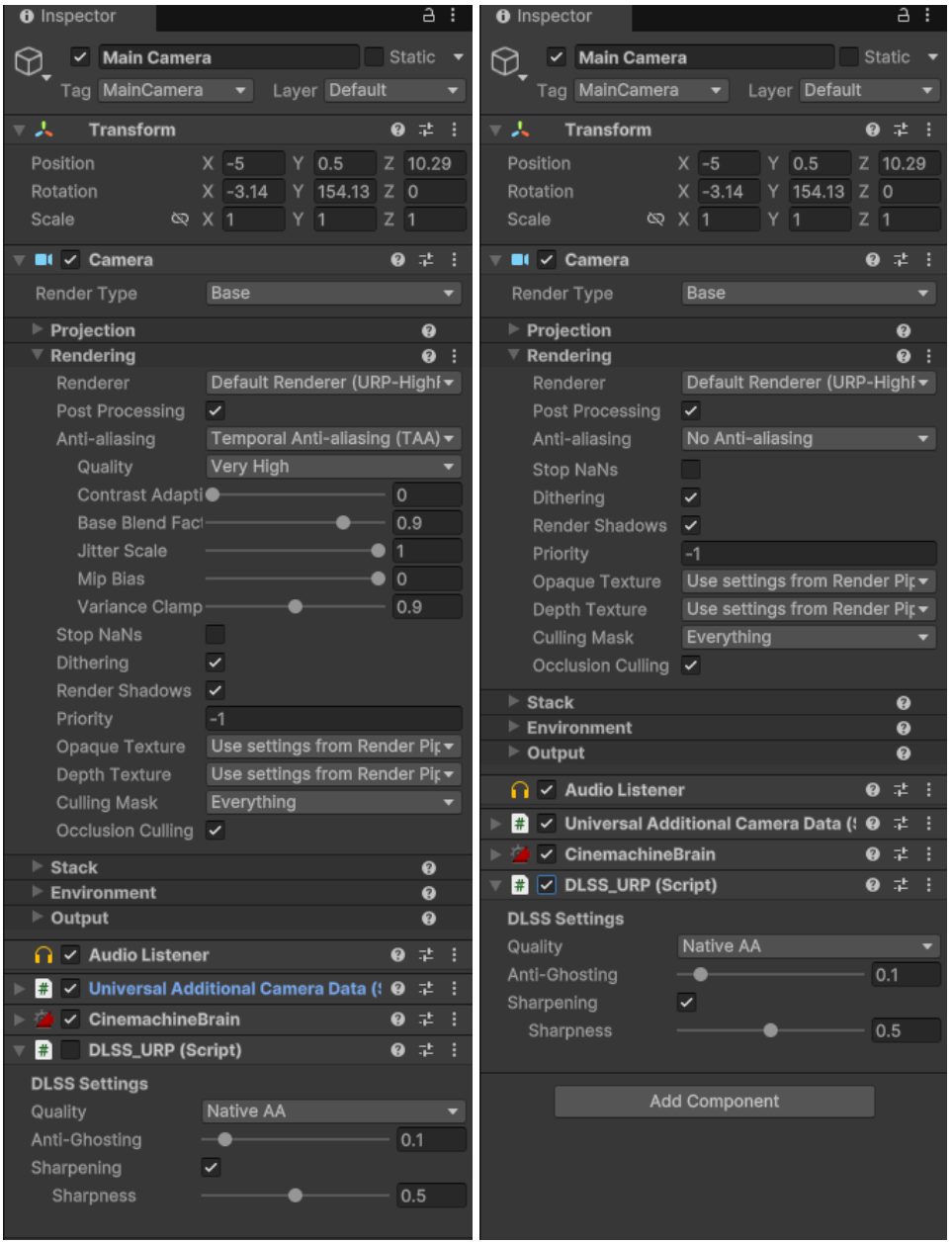
This section will show example settings from our **Unity6 PC concert** demo template project:

- 닐로툰 콘서트 데모 4K 60fps (nilotoon concert demo 01 4K 60fps)
- NiloConcertDemo Unity6LTS

Doesn't mean that you should copy all the settings, these settings are just for your reference if you are also working on a **concert** project.

Camera

- **TAA** only(**RenderScale** = **2**), or **DLSS**.**NativeAA** only(**RenderScale** = **1**)
- **MSAA** = **off** due to VRAM limit pressure in 4K/8K. **FXAA/SMAA** are not used
- **Dithering** may be enabled if **color banding** from volumetric light appears

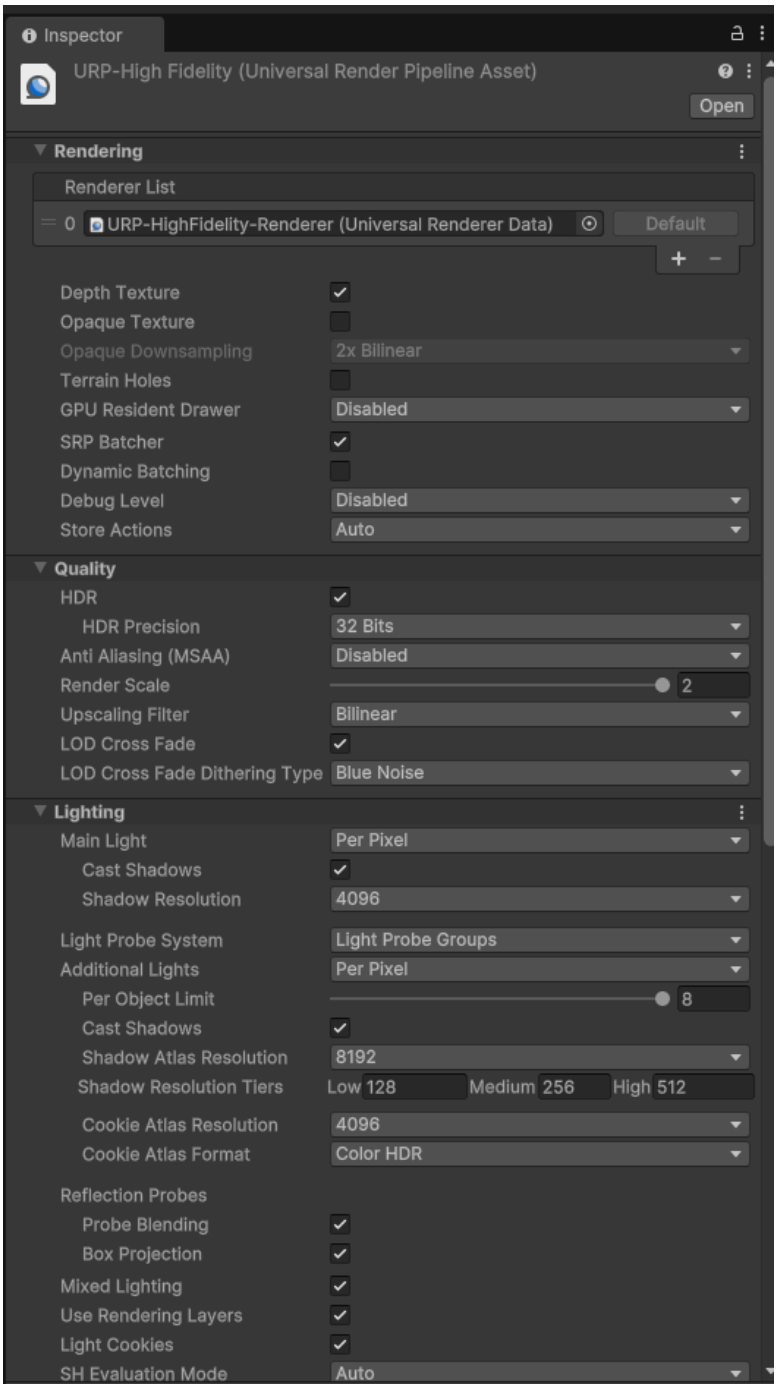


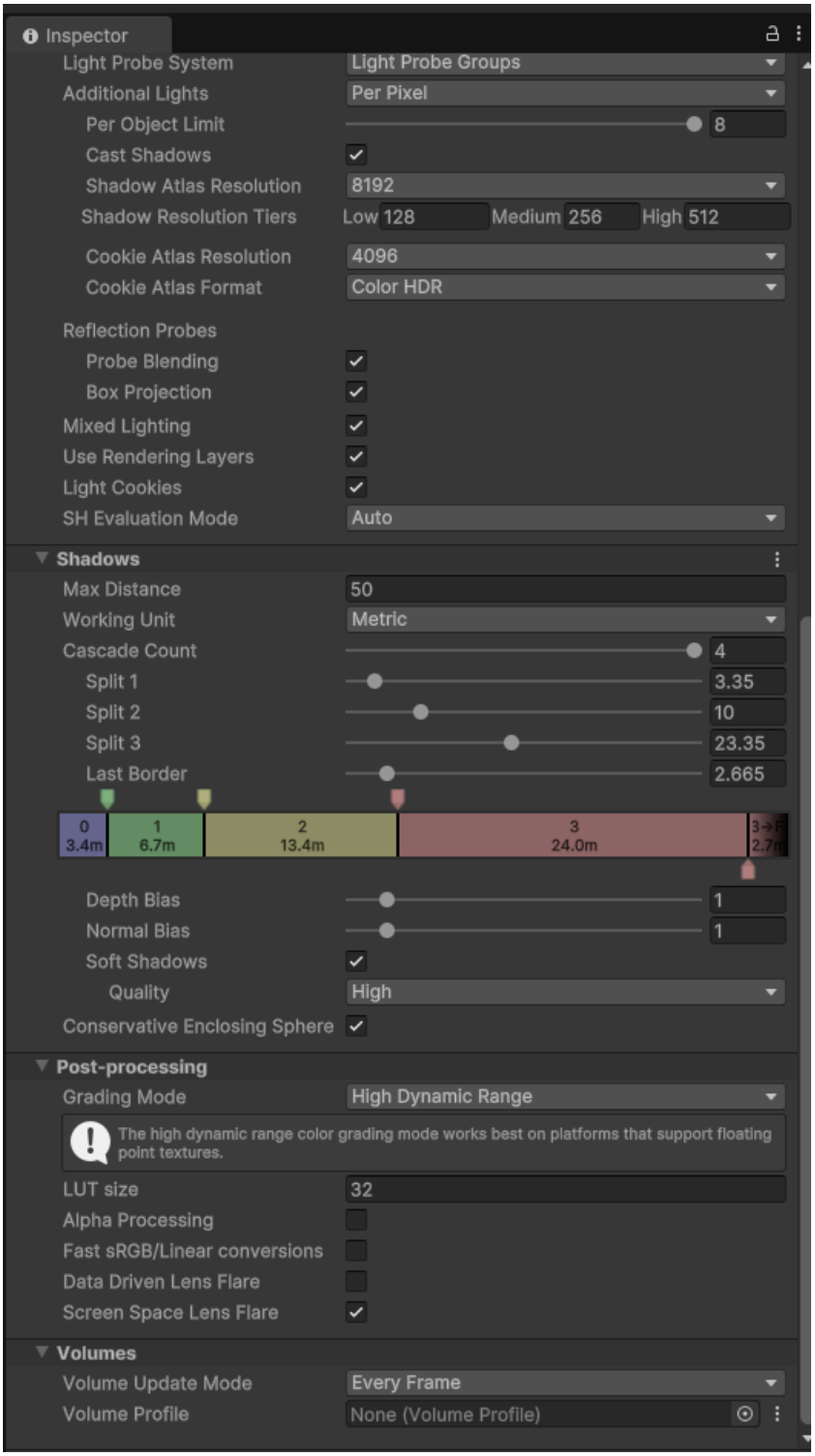
*Left Image = **TAA** only(**RenderScale = 2**)(DLSS = off), great for normal resolution like **1080p,1440p** or even **4K** if you have a good GPU (e.g., RTX 4080 or higher)

*Right Image = **DLSS** only(**RenderScale = 1**) (Camera AA = off), only for high resolution like 4K ~ 8K

URP Asset

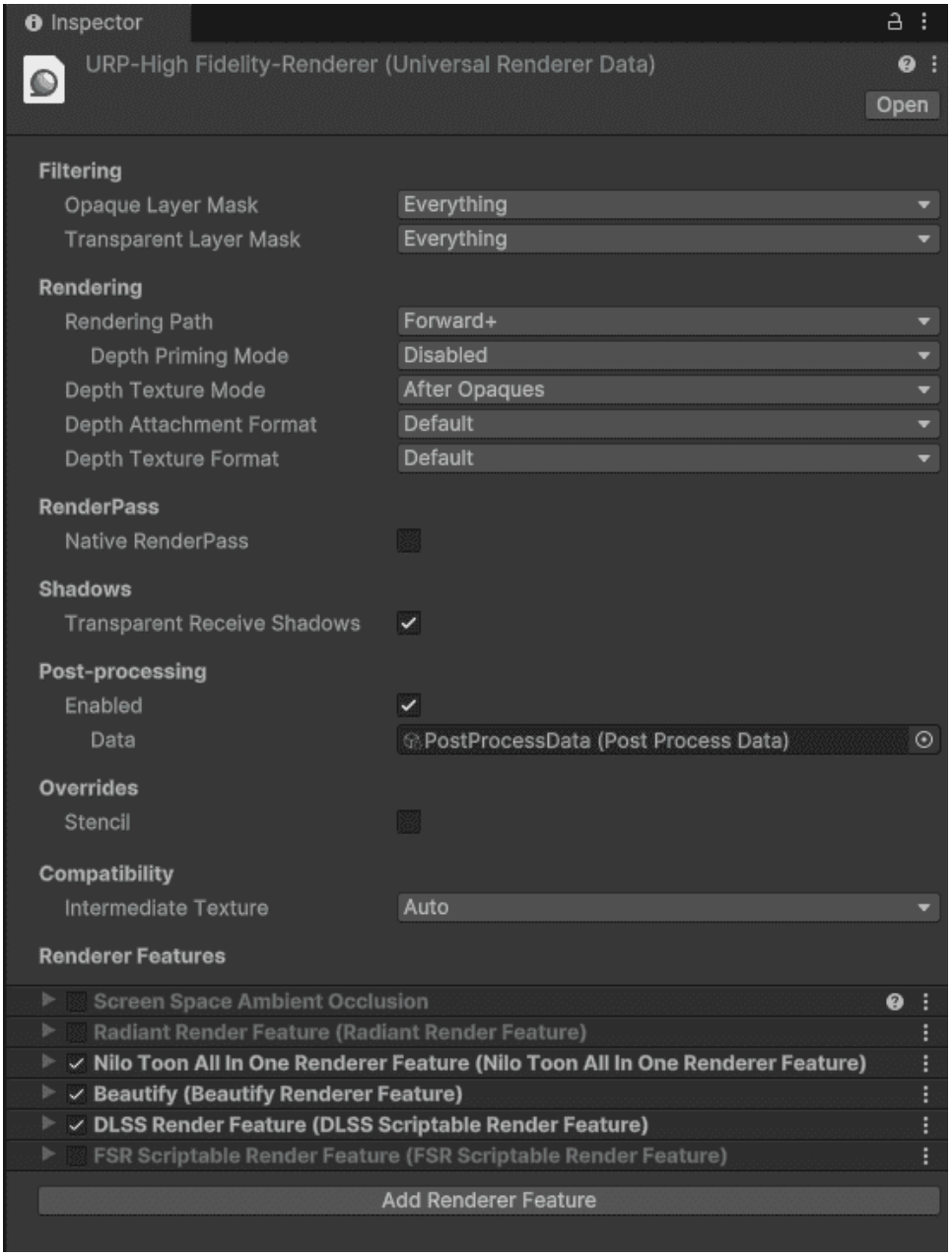
- **GPU Resident Drawer** = **Off.** It has a high CPU base cost, so turn it on only if you have **lots of instancing**(e.g., Kitbashing scene with a total of >10k instance count)
- **SRP Batcher** = **On**, unless you have a reason to disable it
- **HDR** is needed for concert-type HDR lighting + bloom
- **MSAA** = **off** due to VRAM limit pressure or TAA requirement
- **RenderScale** improves anti-aliasing quality a lot! If you have a good GPU, make this as high as possible(1~2) until fps is affected. When RenderScale >1, URP is doing super sampling(SSAA) under the hood. When RenderScale = 2, it means a 2x2 = 4x GPU cost (VRAM & fragment shading) for the best image quality
  *When DLSS is used, RenderScale is always locked at 0.3333~1 depending on DLSS's Quality option, you can't use DLSS & RenderScale > 1 together




Inspector
Light Probe System
Light Probe Groups
Additional Lights
Per Pixel
Per Object Limit
8
Cast Shadows
Shadow Atlas Resolution
8192
Shadow Resolution Tiers
Low 128
Medium 256
High 512
Cookie Atlas Resolution
4096
Cookie Atlas Format
Color HDR
Reflection Probes
Probe Blending
Box Projection
Mixed Lighting
Use Rendering Layers
Light Cookies
SH Evaluation Mode
Auto
Shadows
Max Distance
50
Working Unit
Metric
Cascade Count
4
Split 1
3.35
Split 2
10
Split 3
23.35
Last Border
2.665
0
3.4m
1
6.7m
2
13.4m
3
24.0m
Depth Bias
1
Normal Bias
1
Soft Shadows
Quality
High
Conservative Enclosing Sphere
Post-processing
Grading Mode
High Dynamic Range
The high dynamic range color grading mode works best on platforms that support floating point textures.
LUT size
32
Alpha Processing
Fast sRGB/Linear conversions
Data Driven Lens Flare
Screen Space Lens Flare
Volumes
Volume Update Mode
Every Frame
Volume Profile
None (Volume Profile)

Renderer

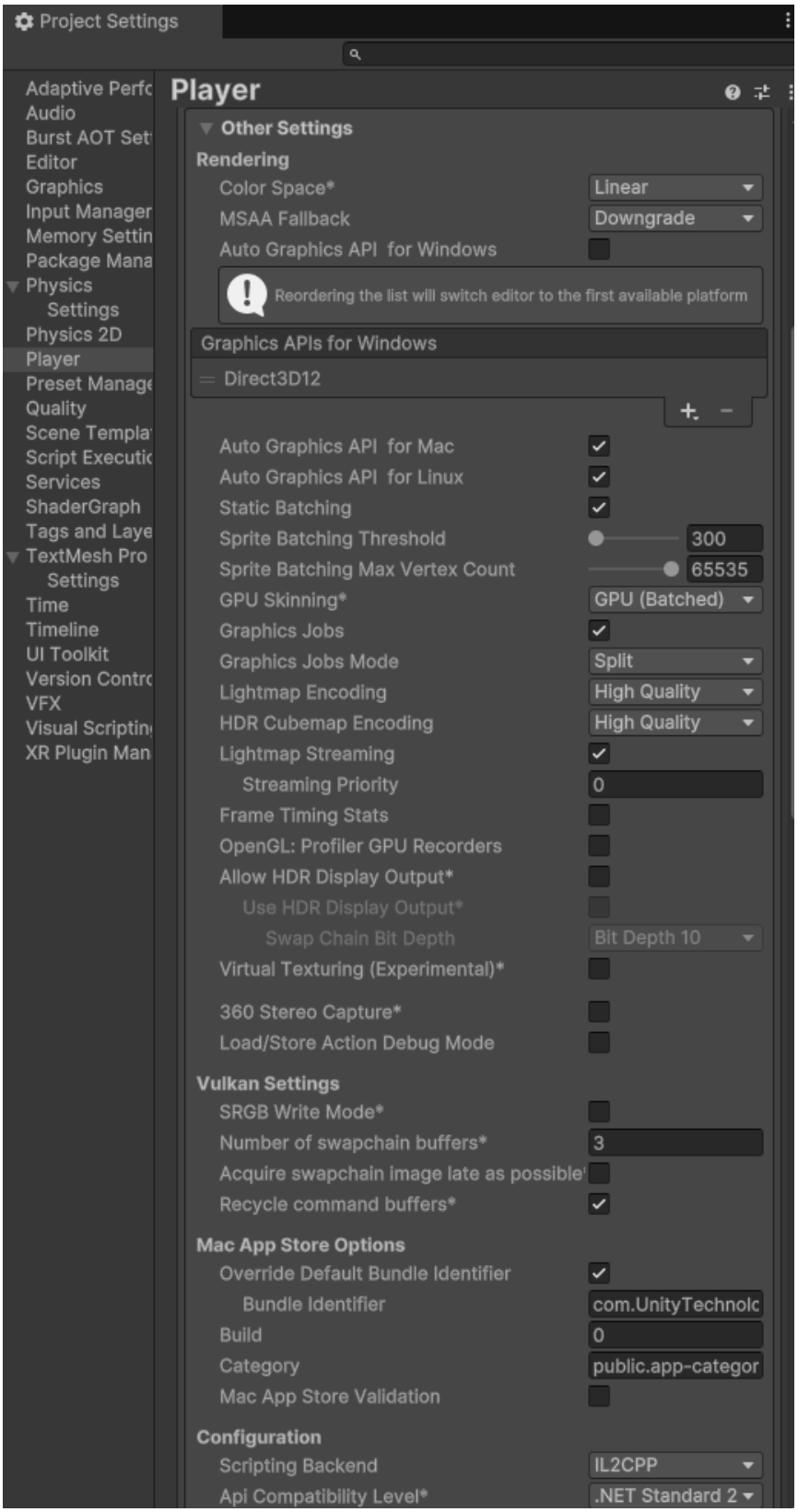
- **Rendering Path = Forward+** for supporting a maximum of 255 visible realtime lights, lights that aim at the character will also **enable soft shadow** to improve lighting quality on character and stage ground. Other **non-important lights** will **disable shadow** due to CPU performance cost, and only turn on shadow if absolutely needed
- **Depth Priming Mode** = **Disabled** (for producing face 2D shadow)
- **Native RenderPass** = **Off**

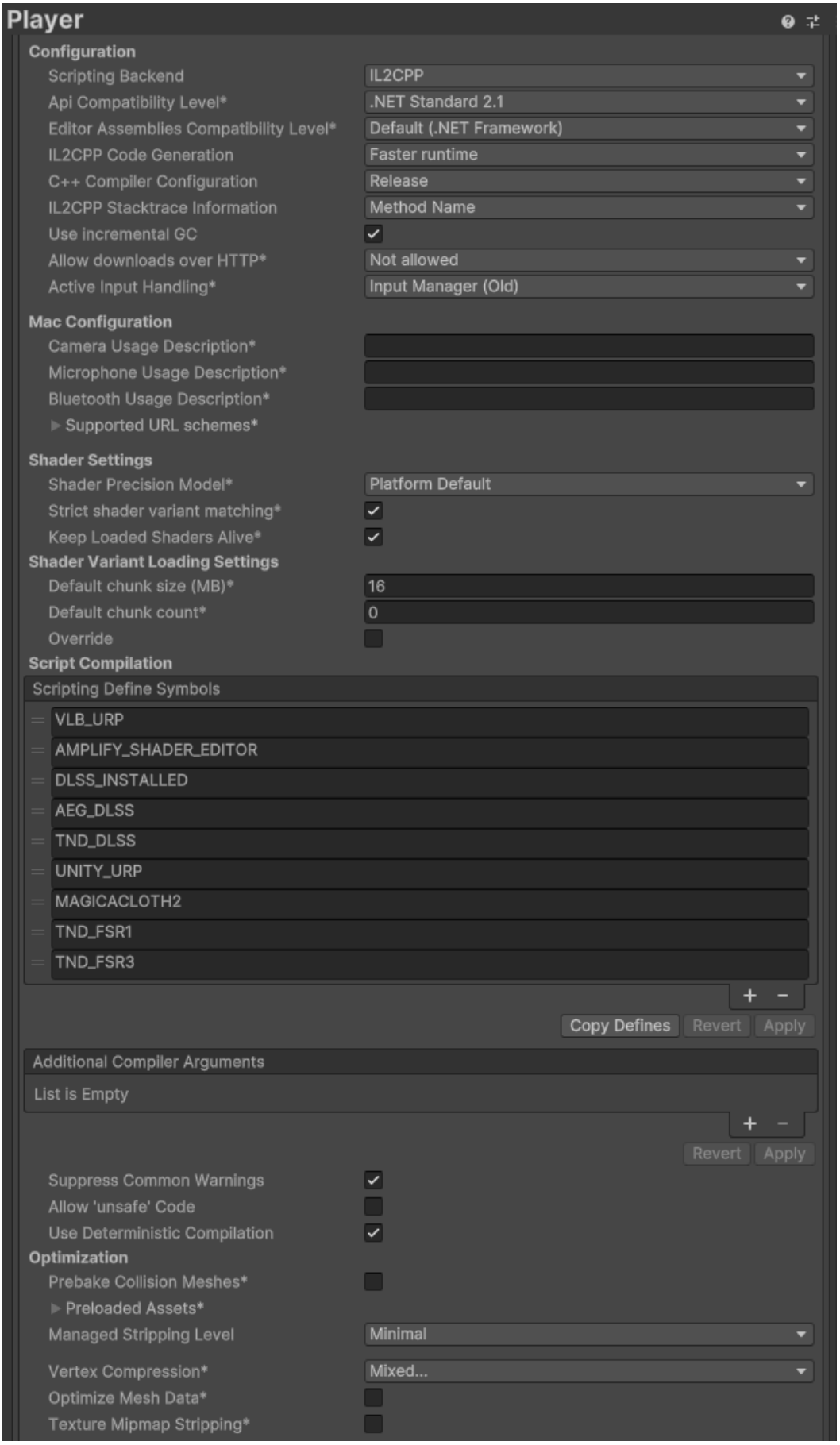


*For **Beautify**, it is **optional**, we are only using the **Anamorphic Flare** feature in Beautify's volume(the horizontal screen space HDR light flare), and the auto **default saturation** boost. Using URP's Screen space lens flare may produce similar results if you don't want to use Beautify.

Project Setting> Player

- important settings = **ColorSpace(Linear), GraphicsAPI(DX12), GPU Skinning(Batched), Graphics Jobs(enable+Split), IL2CPP, incrementalGC(On)**
  * **DX12** is not a must, but it is highly recommended due to the faster CPU render loop. If DX12 doesn't work for you, fallback to use **DX11**

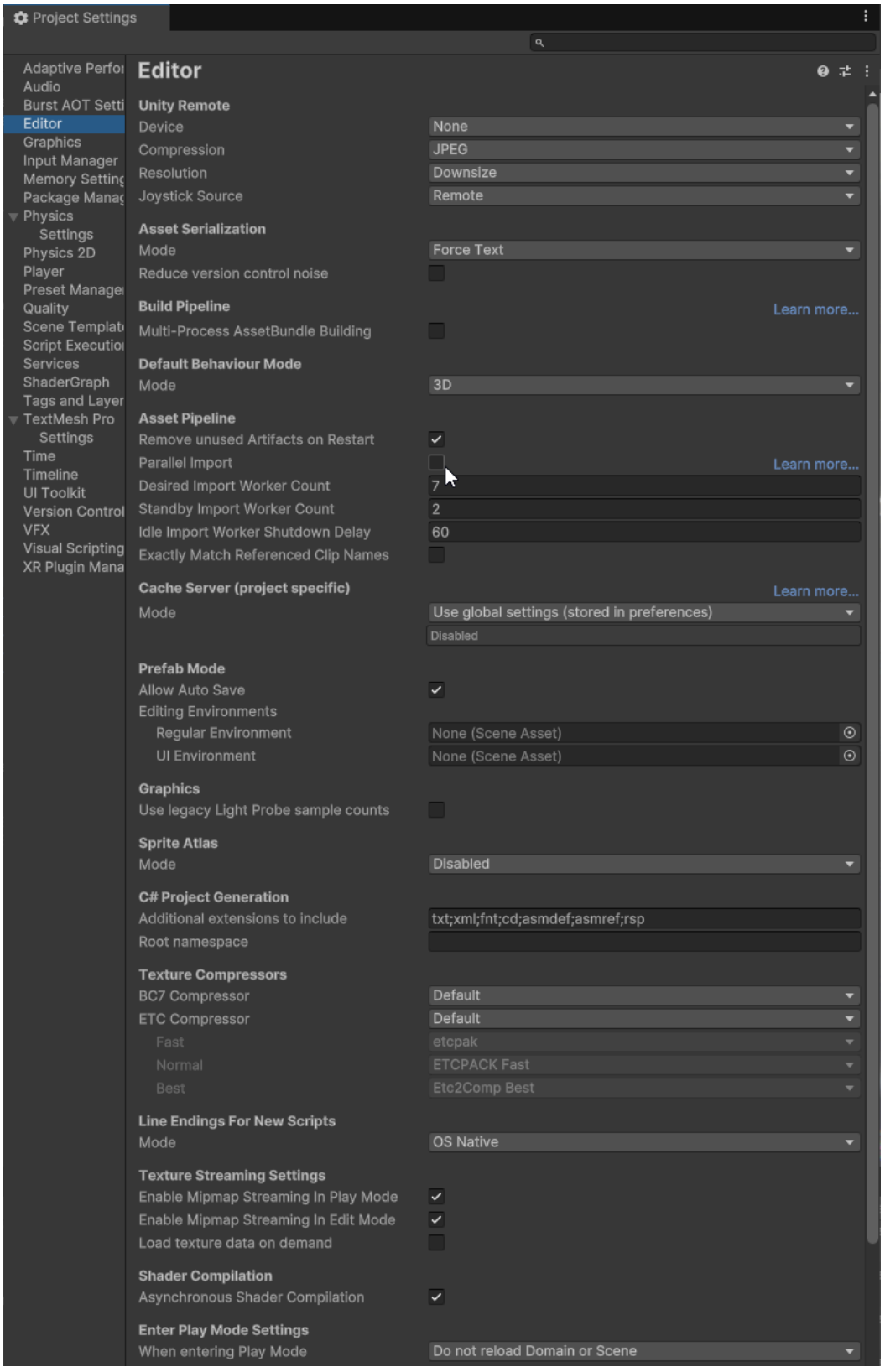




*__Scripting Define Symbols__ are automatically generated, you don't need to follow

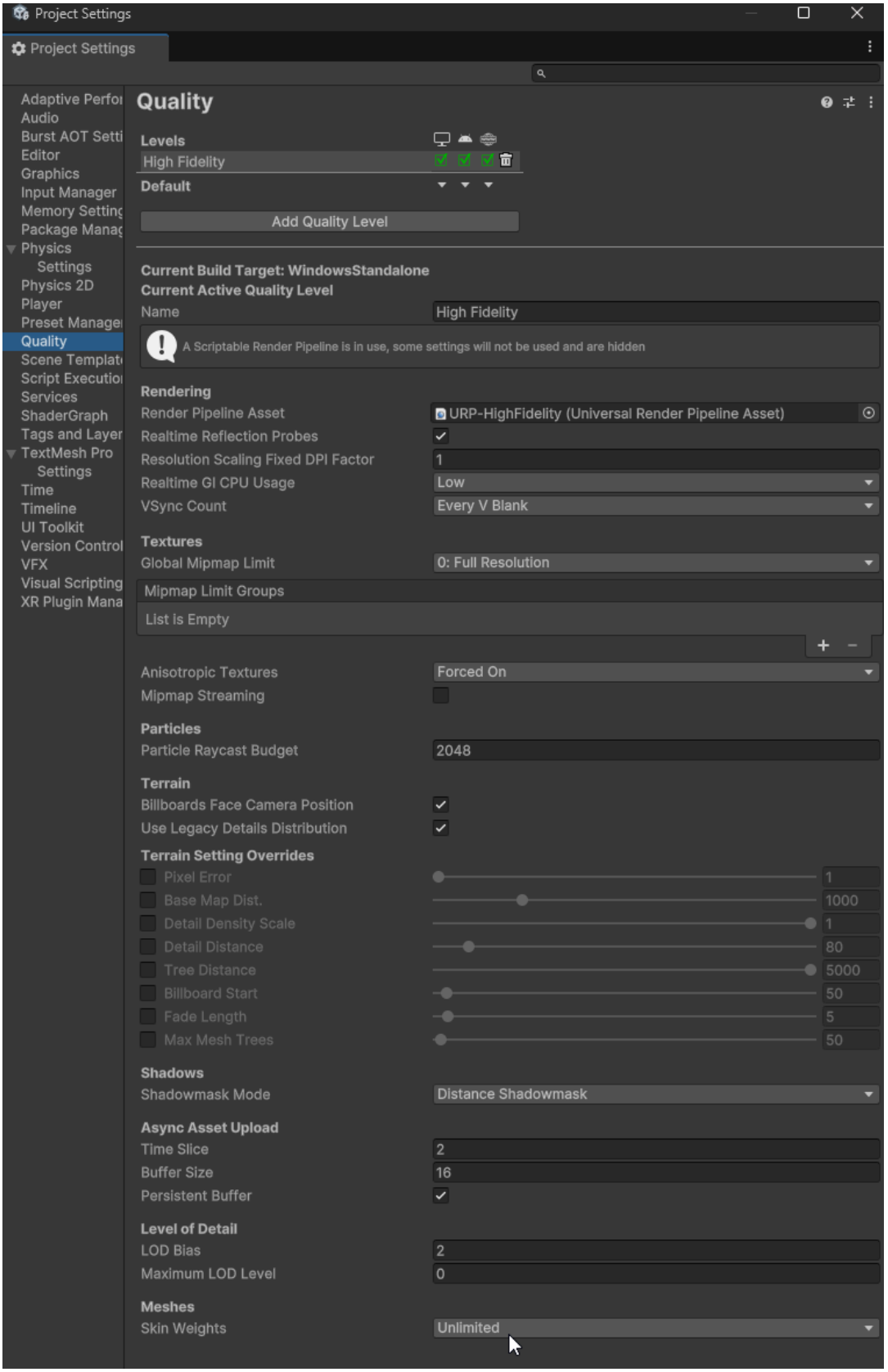
Project Setting> Editor

- Make sure **Parallel Import** = **off**, else your NiloToon fbx prefab will corrupt (see this)
- **Enter Play Mode setting** = **Do not reload Domain or Scene**. It is not a must, but it saves lots of time every time you enter playmode in the editor. NiloToon supports it, but not all your other assets supports it, so be careful when using it



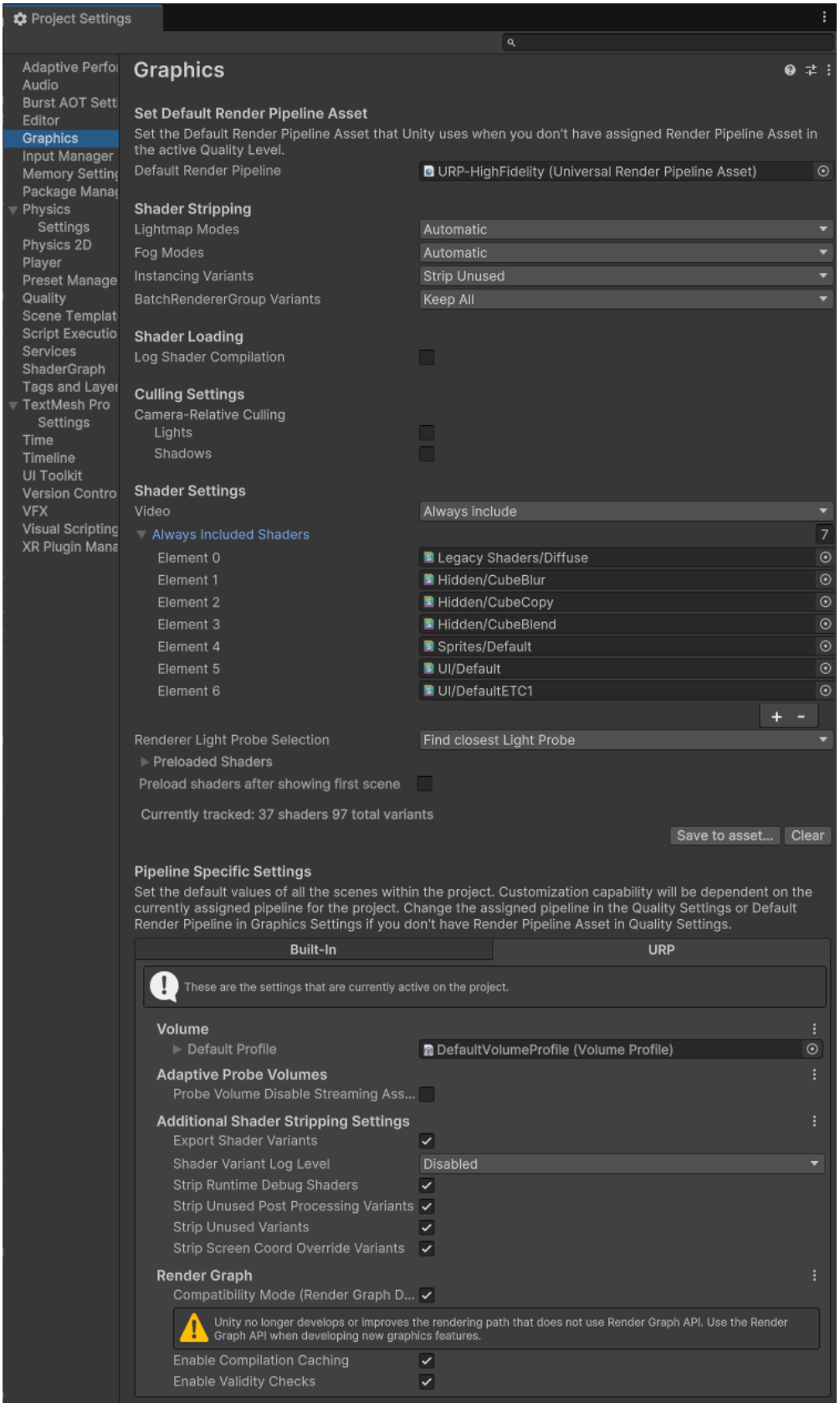
Project Setting> Quality

- Make sure **Meshes > Skin Weights** is **Unlimited**, it will improve the result of character skinning



Project Setting> Graphic

- **Render Graph > Compatibility Mode** = **On** or **Off** will both work. If you find that the rendering result for NiloToon is different between them, contact us!



Notable assets used

1. Character shader: **NiloToon** (NiloToon_Character shader)
2. Postprocess volume: **NiloToon** (NiloToon's Bloom, Tonemapping, Cinematic RimLight control, character render control, shadow control, motion blur…We used most of the NiloToon volumes)
   (**Nilo028-Concert005_StyleDark_ACES(NiloToonVolumePreset)** in **NiloToonVolumePresetPicker** is similar to the one that we use here)
3. Lighting style control: **NiloToon** (**NiloToonLightSourceModifier** script added to a few lights that affect character, setting **Color = 0** & **Direction = 0** for producing a 'rim light only' effect on character)
4. Real-time Motion blur: **NiloToon** (see NiloToon Motion blur tools)
5. Offline Motion blur baking: **NiloToon** (see NiloToon Motion blur tools)

—----------

6. Light fixture prefabs/animation timeline: Unity_StageLightManeuver
7. Light beam: Volumetric Light Beam
8. Lit smoke particle: Vfx Graph - 6-way lighting (Require Unity6)
9. Confetti particle: opaque VFX graph with motion vector
10. Ground planar reflection: PIDI : Planar Reflections 5 - URP Edition
11. Horizontal bloom flare: Beautify 3 / URP's Screen space lens flare
12. Far back ground fog: Lux URP Esstenstails
13. Video player for stage's big monitor: KlakHap
14. Character cloth physics: Magica Cloth 2
15. Mouth AIUEO anim: uLipSync or ARKit animation
16. Character animation: UnityChan Candy Rock Star
17. Stage environment prefab: Stage constructor

For details of all possible tools that we may use for concert, see Tool for 3D Live/MV

New to Unity6?

If you want to upgrade from **Unity2022.3** to **Unity6**, and want some comments/reviews from us before spending time trying Unity6, you can read the following section. These are only **our subjective opinions** about Unity6.

We wrote the following comments based on working on these **Unity6** projects:

- NiloToon Unity6 Concert demo project

Which Unity6 is better?

We will always recommend using **the latest Unity6LTS** that you can download via UnityHub, not any **alpha/beta** Unity6.

Unity6 stability

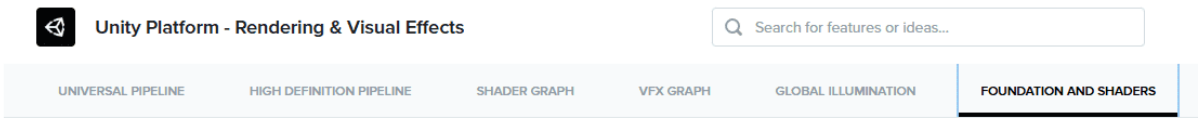
From our experience,
- Unity2021.3 = **7/10** stability, quite stable due to less features
- Unity2022.3 = **6/10** stability, and in some early Unity2022.3.x versions they are **0/10** due to URP RenderTexture memory leak and crash
- **Unity6000.0** = **8/10** stability, we used Unity6 for a few months and we believe it is surprisingly quite stable

Is upgrade painful?

- Upgrade from **unity2020.x -> Unity2021.3** = very painful, renderer feature C# / shader hlsl code rewrite needed
- Upgrade from **unity2021.3 -> unity2022.3** = painful, many API change about **RTHandle** & renderer feature
- Upgrade from **unity2022.3 -> unity6000.0** = this upgrade is quite easy, there should be a few minor API changes (if you enabled **RenderGraph compatibility mode**). You may face **'RenderTargetHandle' is obsolete: 'Deprecated in favor of RTHandle'** problem if your tools are only designed for Unity2021's **RenderTargetIdentifier** renderer feature API, this time you need to update all of them to use **RTHandle** API.

What's new in Unity6

You can always check the Unity6 roadmap, remember to check all tabs to find useful features for your project:



https://unity.com/roadmap/unity-platform/rendering-visual-effects

Useful features

From all Unity6 new features, we like the following features the most:

1. 6-way lighting smoke VFX graph, we use it a lot for smokes vfx that are lit by lights in concert projects like this

2. DX12 split graphics jobs, much better render loop **CPU performance**, no negative effect to visual result
3. DX12 graphics jobs for editor, much better render loop **CPU performance**, no negative effect to visual result
4. Batched GPU skinning, better **GPU performance** when character models have lots of SkinnedMeshRenderer(s) and blendshape(s), no negative effect to visual result
5. GPU ResidentDrawer, may improve **CPU performance** only when rendering large amounts (>10K) of gameobjects that contain the same MeshRenderers for GPU instancing. It has a fixed CPU cost per frame, so enabling it will reduce performance if GPU instancing is not utilized in a large amount.
6. Adaptive Probe Volumes (APV), we treat it like an auto and improved version of the old **light probe** system, no more manual placement needed
7. Screen Space Lens Flare, usually used for concert's lighting effect, see example here
8. VFX Graph Motion Vectors, it makes opaque particles like confetti correctly affects motion blur/TAA/DLSS, we use it a lot for confetti in concert project like this

Useless features

These features are not bad, but we have better alternatives already.

1. URP object motion blur, it is not looking good enough, so we added **NiloToonMotionBlurVolume** for Unity6 that suit our visual needs. You can use URP's object motion blur when faster GPU performance is needed
2. STP, it doesn't look great in motion, so DLSS is always better if you need better visual results from upscaling. For our AA choice, see this, we don't use STP

Change in Unity6

For NiloToon only, not much changed in Unity6, you should not see any major NiloToon breaking change after upgrading from Unity2022.3 to Unity6.

But there are a few NiloToon improvements that only exist in Unity6:
- better character **motion vector pass** = produce better and correct motion vector, it means better DLSS/TAA/motion blur visual result, it is a breaking change in Unity6 but in a good way
- new **NiloToon Motion Blur Volume**

Upgrade to Unity6

If you want to upgrade to Unity6, do the following:

1. **backup** project (commit in version control / just copy the project files)
2. upgrade to **latest NiloToon** (atleast NiloToon 0.16.x latest)
3. check playing in editor, confirm everything is still working as expected
4. restart the project using the **latest Unity6LTS**
5. if you see visual problem, try enable/disable **RenderGraph compatibility mode** to check for difference
6. check for any **error log**, it is possible that some **assets/tools** inside your project are **too old** for **Unity6**, update them
7. check for any **remaining error log**, Unity6 has breaking C# API change, but it is not that much and fixing them should be not difficult
8. try playing in Editor, look for any remaining error log

9. try building the application

Low fps & solution

There can be many different things that makes your editor/build's performance(fps) low, so you should always:

1. find what is actually slowing you down
2. optimize only that thing (do not make random edits and hope it helps)
3. recheck fps
4. if it increases fps with no side effect, keep the change, push to version control
5. go back to 1

Here we use a **concert PC project** as an example for the following Q&A. Many other concert projects face a similar performance bottlenecks:

1.CPU or GPU bound?

If you lower the Game window **resolution** and **RenderScale** to the **lowest possible**:

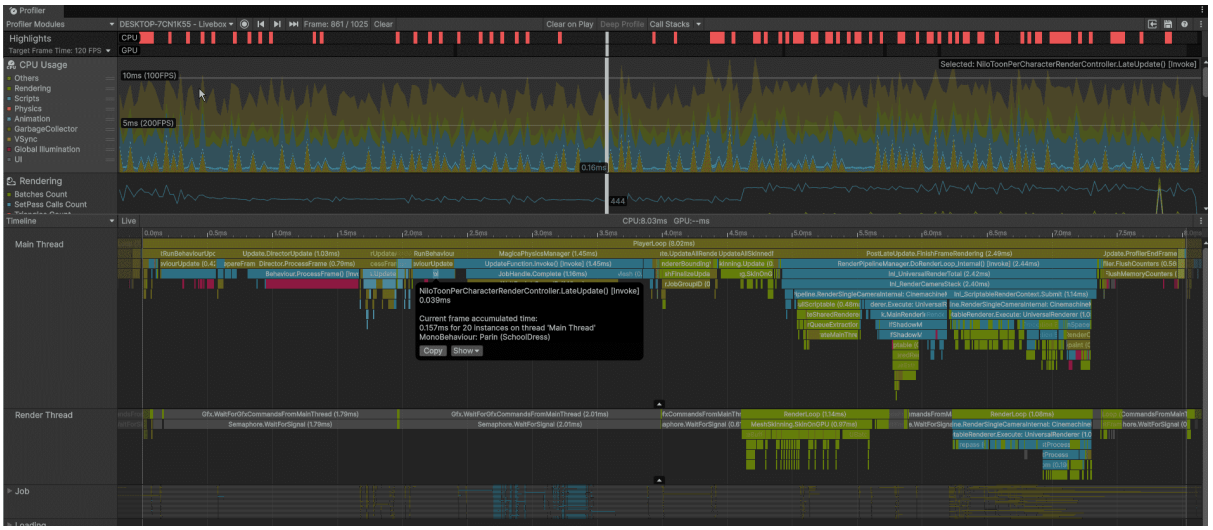
- fps stays the same = you are likely **CPU bound**
- fps increases a lot = you must be **GPU bound**

*Most concert project we worked on are **CPU main thread** bound

2.How many characters?

For **1-5** characters, it is quite common, and it should not slow down too much due to NiloToon.

*Each **NiloToonPerCharacterRenderController** has a fixed CPU main thread cost, you can check the sum of time cost in the profiler. If you use a lot of **NiloToonPerCharacterRenderController** for prop/items, be careful about the total of CPU main thread time cost. (Image below shows a **4 character + 16 prop = 20 NiloToonPerCharacterRenderController** situation, they will produce a total of 0.x ms CPU cost)



3. Hardware requirement

If the editor is slow, what is your hardware? For example, our PC for Unity editor is:

- CPU = Intel Core i7-14700
- GPU = NVIDIA RTX 4080 super

capable for medium to small size concert projects running in the editor

4. Editor or Build?

For **CPU**, there can be significant **CPU** performance differences between **editor** and **build**. Editor play mode typically has more editor-only overhead and tends to be slower compared to running in a built executable. If possible, build and test **CPU** performance in build.

For **GPU**, it is ok to test performance in editor, since the GPU performance in build will not have a big difference when compared to the editor.

*In the **concert PC project**, ⅓ of the CPU time are Editor only tasks

5. Common CPU bottlenecks:

To solve CPU bound problems,

Reduce/disable them:

- PIDI **planar reflection** (each reflection is using an extra camera)
- more than 1 camera (using only 1 3D main camera is recommended)
- too many characters with **NiloToonPerCharacterRenderController** (e.g. > 5)
- too many prop/items with **NiloToonPerCharacterRenderController** (e.g. > 20)
- too heavy **MagicaCloth** 1/2, vrm spring bone
- too heavy timeline / director update, usually due to lighting animation
- stage light timeline update, when using heavy tools like this one -> https://github.com/murasaqi/Unity_StageLightManeuver
- too many visible lights
- too many light enabled **unneeded shadowmap**
- non-SRP batching material in frame debugger
- CPU particles with high spawn count, like sea of penlight or confetti
- NiloToon's "Keep play mode mat edit?"; it prevents SRP batching in editor
- NiloToon's **High Quality Culling** in NiloToon renderer feature's **Char Self Shadow** section; it costs a full 0.x ms camera SRP culling.

Try enable them:

- **DX12** graphics API (DX11 is usually slower in multi-thread)
- DX12 split graphics jobs, much better render loop **CPU performance**, no negative effect to visual result
- DX12 graphics jobs for editor, much better render loop **CPU performance**, no negative effect to visual result
- GPU ResidentDrawer, may improve **CPU performance** only when rendering large amounts (>10K) of gameobjects that contain the same MeshRenderers for GPU instancing. It has a fixed CPU cost per frame, so enabling it will reduce performance if GPU instancing is not utilized in a large amount.
- GPU Skinning

*If you want to find CPU bottlenecks, using **Profiler** and **FrameDebugger** are very suitable, these 2 tools will help you find most of the CPU performance problems

*See also Unity6 Useful features

6. Common GPU bottlenecks:

The reason for GPU bottlenecks is usually much straightforward, and is similar for most projects. To solve GPU bound problems,

Reduce/disable them:

- High Game window **resolution** (e.g. >= 4K)
- High **RenderScale** (e.g., >= 1) (but sometimes, 2 is still preferred for the best quality)
- enabled **MSAA** in high resolution/RenderScale (disable MSAA, and use TAA if possible)
- too much **volumetric** light beam / **volumetric** fog overdraw (stack together and cover many pixels)
- **no mipma**p for high resolution textures
- too heavy **post-processing** (try disable volume and check fps difference)

Try enable them:

- Batched GPU skinning, better **GPU performance** when character models have lots of SkinnedMeshRenderer(s) and blendshape(s), no negative effect to visual result

For detailed example settings for resolution & AA to reduce GPU cost, including **mobile**, see AA/Quality presets.

For detail doc about DLSS for improving GPU performance, see DLSS

Edit NiloToon source code

If you are not a programmer, you can skip this section
Although we don't recommend editing NiloToon's source code, since it will make future updates difficult. If you absolutely need to do it, here is a suggestion for version control:

Usually in your version control, you need 2 extra branches to improve future merging.

[Extra Branch A (NiloToon source update only)]
- you push new NiloToon .unitypackage updates into this branch only, without any other edits by you

[Extra Branch B (Your NiloToon edits)]
- you merge the above "Branch A" into this branch from time to time, and push only your NiloToon code change into this Branch B

You have a real development branch / main branch,i.e., **[Main Branch]**. You will merge **Branch B** into **[Main Branch]** from time to time.

With this 3 branch setup in version control, it may reduce many merge conflicts, allow NiloToon to continue to update, while not removing your change to NiloToon's code.

Still, it is possible to produce merge conflict when you update NiloToon(e.g., when you merge "Branch A" to "Branch B"), depending on which part of the NiloToon's source code you edited, if NiloToon's developer and you edited the same code line, conflict will still be produced.

Edit NiloToon's hlsl

For example you want to apply a global color tint effect to NiloToon's shader, before editing any **NiloToon** related hlsl, you should first open the following files from the project window:
- **NiloToonCharacter_ExtendFunctionsForUserCustomLogic**
- **NiloToonEnvironment_ExtendFunctionsForUserCustomLogic**

These are hlsl designed for users to edit, they have different empty functions(entry point) for you to inject any custom logic into the shader at different timing in the shader, similar to MonoBehaviour's empty function like Update() / LateUpdate().

So if you want to apply change to NiloToon's hlsl, see if the above files works for you first, using them will make development much cleaner and easier, since we will not edit these files for a long time, so version control conflict will be very rare, you can safely develop extra rendering logic into them and don't worry future breaking change.

Use Rider for hlsl

We highly recommend using **Rider** for hlsl development, it is the only tool that we use for hlsl development since we can write/read hlsl just like working with C#, important function to work with hlsl in Rider are:

- **Go to Declaration** or **Usages** (**F12**), great for jumping into URP/NiloToon's function/struct, or finding usage of a function/struct
- **Go back** to previous location (**Ctrl + -**), usually used after F12, to go back to the line you are programming
- **Extract method** (**Ctrl + .**) or (**Ctrl + R, M**)
- **Rename** (**Ctrl + R, R**)

Build & shader stripping

While building the following files from the editor:

- application for a target platform
- .warudo
- assetbundles

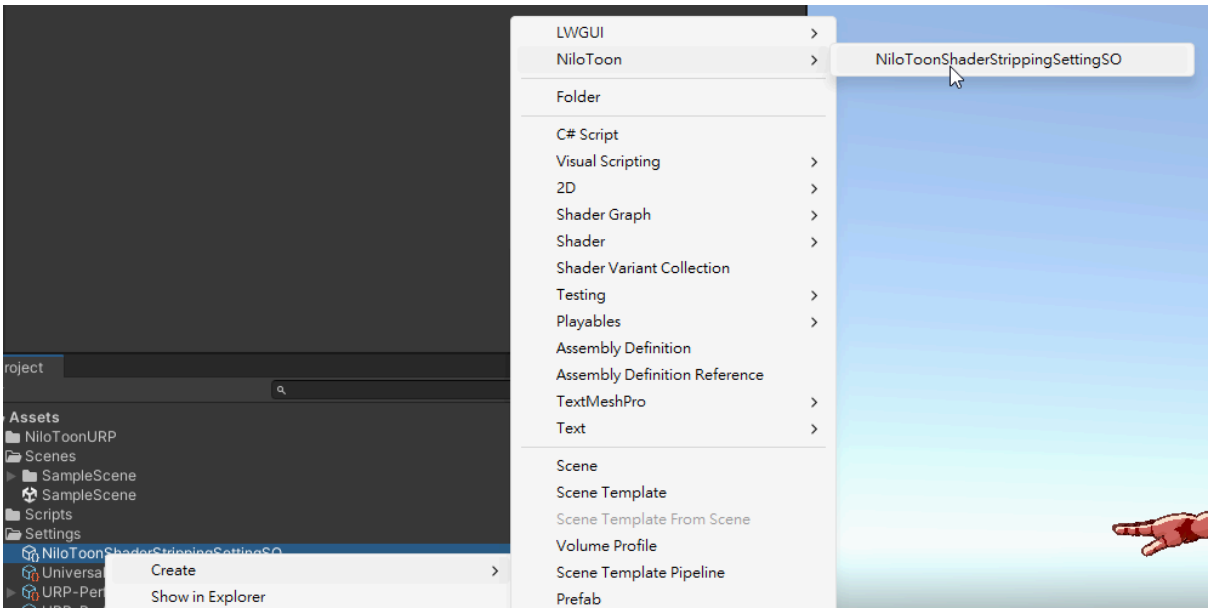
usually it doesn't require any extra steps for NiloToon to run in your build correctly, but if you used some features in the editor that is not included by default for a target platform, for example:

- Dissolve
- Dither fadeout
- Screen Space Outline
- Receive URP shadow

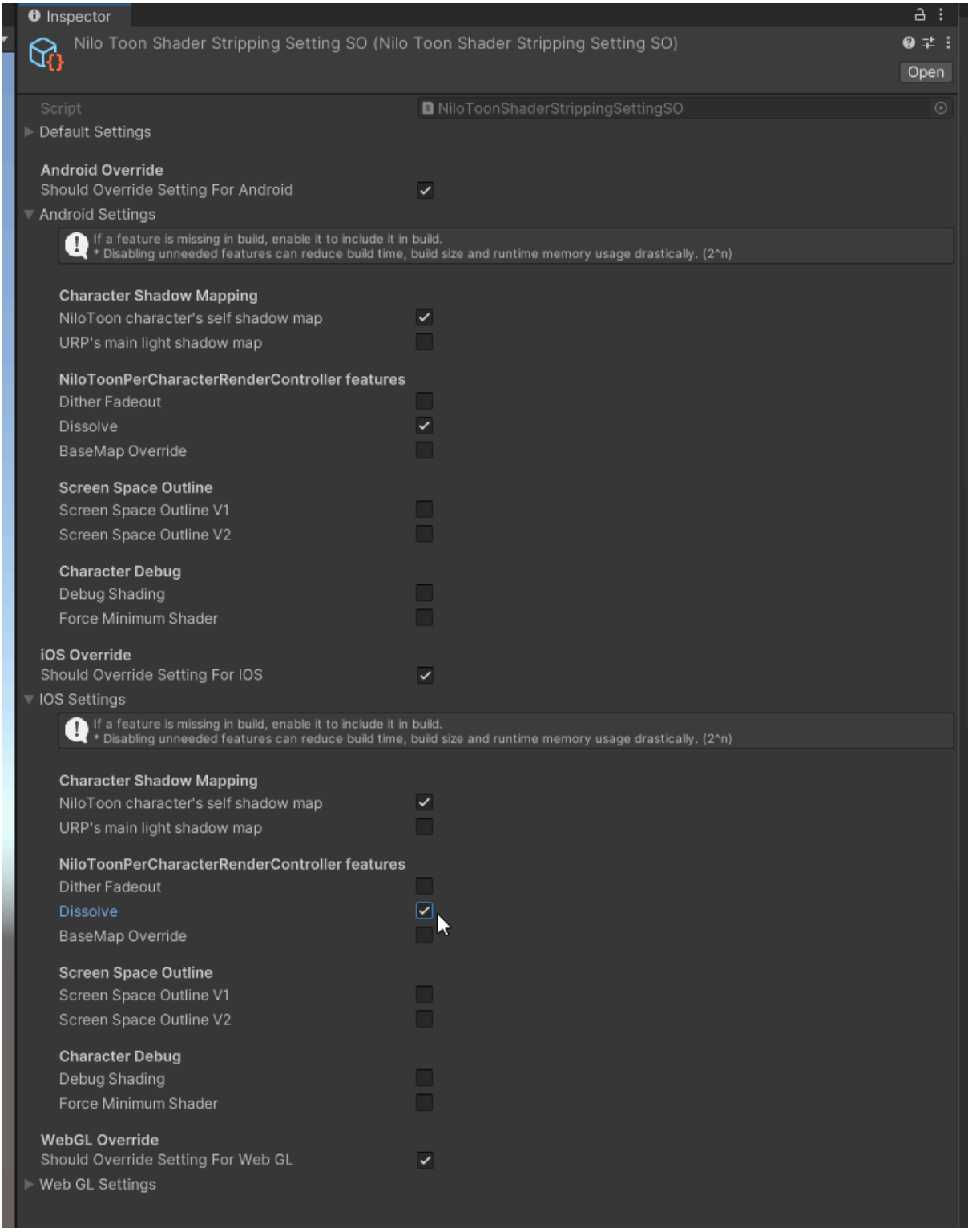
You will need to manually enable those features for a specified platform, so NiloToon can include those feature's shader variant in build.

For example, if you find that some features(e.g., Dissolve / Dither Fadeout / Screen Space Outline / Receive URP shadow) are missing only in build, but works perfectly in editor, follow the steps below to solve the problem:

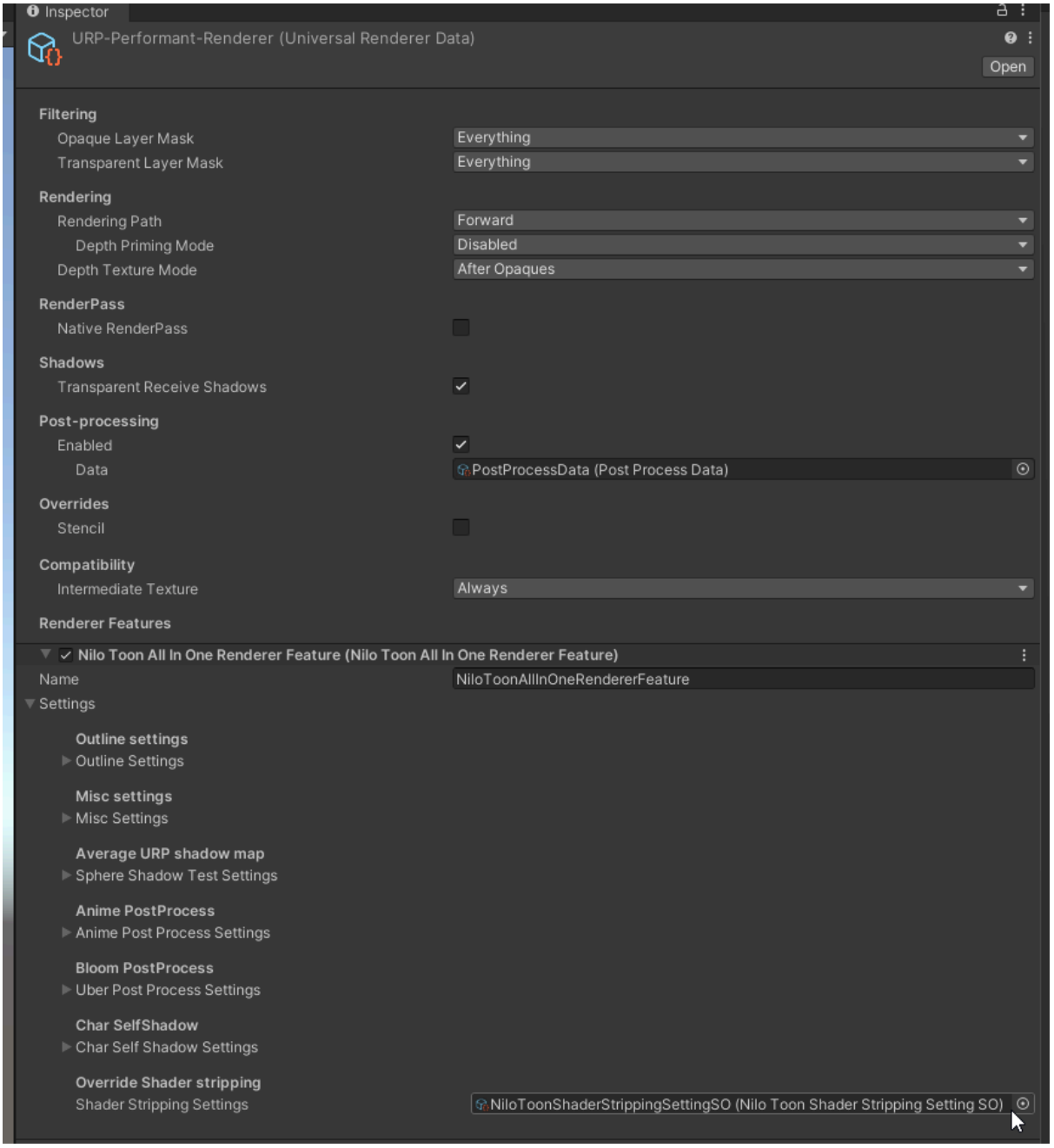
1. Create a NiloToonShaderStrippingSettingSO asset



2. Enable **the required features** for your target platform. For example, enable **dissolve** for android and iOS if you find that **dissolve** is missing only in android and iOS



3. assign that NiloToonShaderStrippingSettingSO from step(1-2) to **all** of your target NiloToon renderer feature



4. build again, then the problem will be solved! The same applies to a simple direct build, asset bundle, addressable, .warudo

If you find that the build time is too long, and you want to remove some useless shader variant in build instead, you can do the opposite.
See Reduce Shader memory & build time

NiloToon(Lite) - mobile/VR

Since NiloToonURP(full version) contains many features for PC and console, it has lots of overhead in rendering even if you don't use those features in mobile/VR.

If NiloToonURP(full version) is too slow for your mobile/VR game project, and you are willing to try another shader that we specifically developed for mobile and VR, try using NiloToon(Lite) shader:
NiloToon(Lite) - mobile & XR high-fps shader

You will see a **3x~5x GPU speed up** when comparing with the NiloToonURP(full version).

For example, if **NiloToonURP(full version)** is costing the GPU **10ms** in your mobile/VR game, **NiloToon(Lite)** will likely cost only **2~3ms**, and it scales way better when there are lots of characters and objects, and high resolution rendering.

Pros:

- It is an extremely fast shader that we built using our past experience from working in the mobile game industry, it is unlikely you can find a faster toon shader in the asset store or github. This shader is designed to work great on weak GPU phones and Quest2
- Simpler to use than NiloToonURP(full version), since there is way less features and options
- Perfectly fine to use it on any renderers, not limited to character only

Cons:

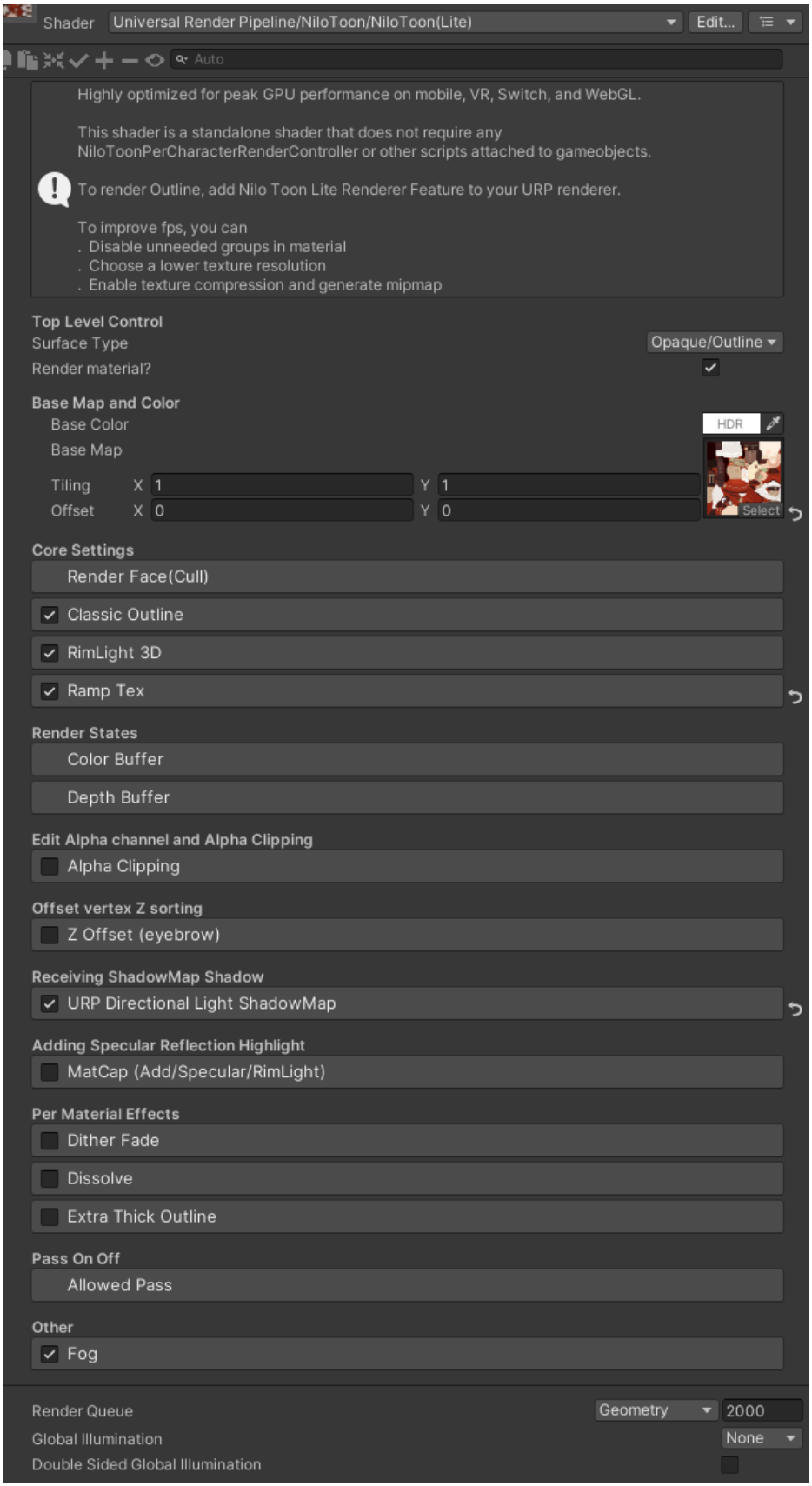
- It is WIP and in early phase of development, things like Shader name or property name may change in future updates
- It has a very low priority in our development plan, update is expected to be very slow (let us know if you want more updates or features, since we don't know if there is any demand for this mobile/VR shader from our customers, most of our customers are VTuber/Virtual Production companies that only cares high-end PC platform as a workstation)
- It has way less features than the full version NiloToon
- No document, only helpbox and tooltip within the material inspector
- NiloToon(Lite) only contains 1 shader and 1 renderer feature, so you don't have any helpful buttons like the auto install or auto setup features
- Doesn't affected by any NiloToon(full version)'s script, renderer feature or volume, it is a standalone shader that runs completely outside of NiloToon(full version)'s system to achieve the best performance

*If you have NiloToonURP imported in your project already, after importing NiloToon(Lite), you may need to manually **delete** the **LWGUI** folder inside the **NiloToonURP(Lite)** folder since it is a duplicated copy of LWGUI, you only need 1 copy of LWGUI

*NiloToon(Lite) has a minimum default setting (most feature default = off), since it has only one goal - fastest GPU performance on mobile/VR. You should only enable the feature that you absolutely need in the material. The less feature group you enable in the material, the better the GPU performance

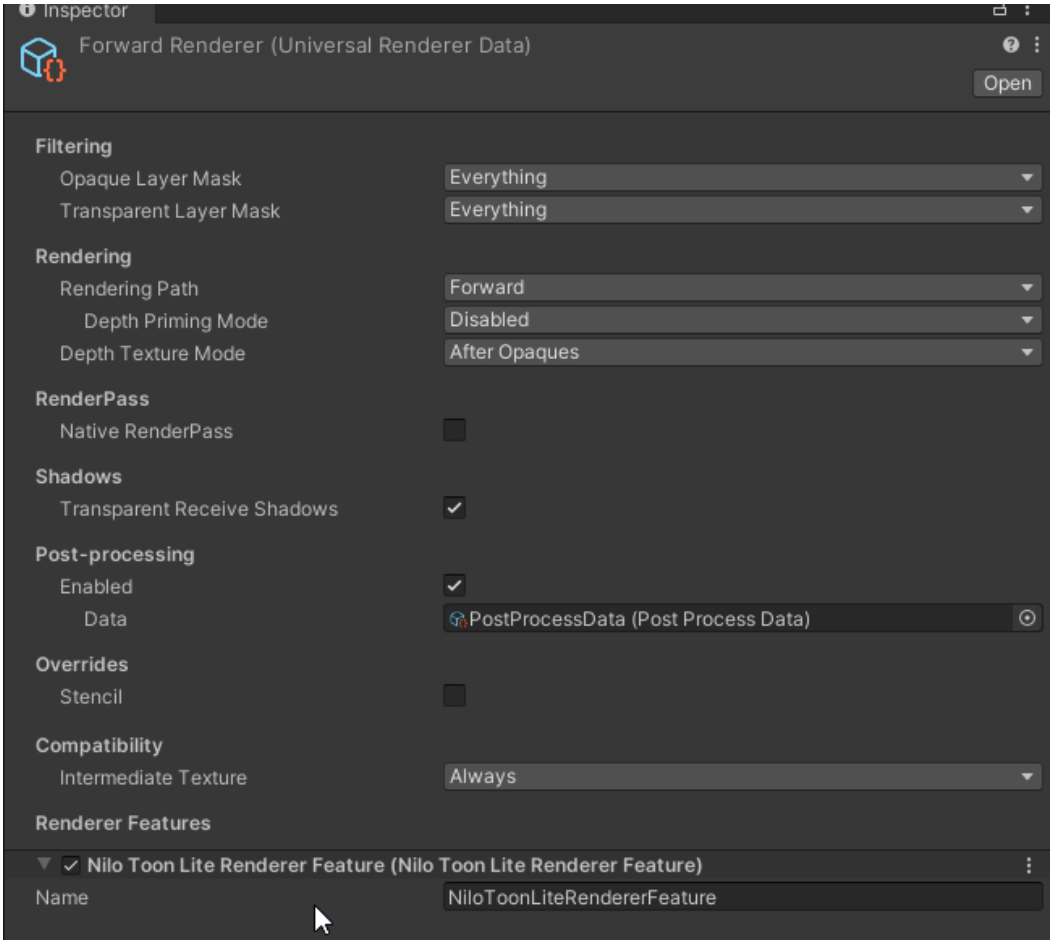
For detailed example settings for resolution & AA to reduce GPU cost, including **mobile**, see AA/Quality presets.

Material Inspector of NiloToon(Lite) shader:



Example render result of NiloToon(Lite) shader, you can render lots of character on mobile with a much higher fps than NiloToon(full version):

To produce Classic Outline for NiloToon(Lite), you need to manually add a **NiloToonLiteRendererFeature**, it is not NiloToon(full version)'s renderer feature:



To achieve the best performance, you should disable NiloToonURP(full version)'s renderer feature to reduce workload.

Road map

Feel free to skip this section if you just want to use the current version of NiloToon right now.

Similar to URP's Road map, we will record NiloToon's roadmaps here.

In progress(highest priority)

- **Dissolve** override **per material**
- **Support Parallel import**: rewrite outline UV8 baker, to support parallel import with fast reimport
- **2-pass depth texture for classic outline**: (requires render graph) to improve classic outline's Depth of field & STP result without requiring any settings by user
- **2-pass motion vector on classic outline**: (requires render graph) improving classic outline's motion blur/TAA/STP/DLSS result without requiring any settings by user

In progress

- **Improve lilToon -> NiloToon material auto convertor**: porting more important lilToon properties to NiloToon in the auto setup process(e.g., **Matcap 1 & 2**, **Glitter**), so you can convert models(e.g., booth lilToon character models) from lilToon to NiloToon in a shorter time.
- **Concert template PC Unity6 project:** a complete 3D live **Unity6LTS** project zip (using render graph)

Planned

- **2-pass ForwardLit for easier semi-transparent render**: If a material mixed Opaque and semi-transparent parts together, rendering can be difficult
- **Optimize mobile performance**: NiloToon 0.13.8 has acceptable mobile performance since many new features targeting PC have not been added in NiloToon 0.13.8. Future version aims to optimize the shaders, providing 'Shader Quality' in renderer feature, so shader can choice a GPU performance similar to NiloToon0.13.8 again with limited features

Under consideration

- Merge NiloToon(Lite) into NiloToon, so when users build for the mobile platform, they can select the feature set in NiloToon's renderer feature (e.g., Minimum/Lite/Core/Full) based on platform or quality level. The goal is to give option to user to downgrade the graphics for better performance, without doing any extra work(e.g. don't require user to change the shader of all character materials, it should be as simple as editing an option like a "NiloToon Shader Quality" in the renderer feature)
- **lilToon Fur** pass function
- **lilToon Glitter** function
- A new '**Override Basemap to _CameraOpaqueTexture**' function with uv distorting, for rendering glass/diamond/water like material

No Plan / Canceled

- **HDRP support** (Canceled due to the new **Unified Renderer** in Unity7) -The originally plan is to write RP abstraction, then include Unity6 HDRP support within the same NiloToon package, keeping the same NiloToon UI/UX between URP & HDRP, rename the product name from NiloToonURP -> NiloToon. (In the past, added as

'under consideration' due to users trying to render NiloToon characters in URP & render environments in HDRP)

- **Unreal Engine(UE5)** support - (Marked as '**No Plan**' due to inquiries about this video 역광 | VIICHAN SOLO CONCERT ✦), we can't make a production-ready UE5 port of NiloToon **without editing UE5's source code**, and it is difficult to maintain it to support every future UE 5.x updates. We believe if NiloToon needs to edit UE5's source code to successfully make a good UE5 port, it is useless to most NiloToon customers since most customers will not consider using a UE5 engine with edited source code by us.
  Also, if we have to do it, the Unity NiloToon will have very slow updates and customer support for a long time, which is unfair to customers who only use Unity, so we have no plan on a NiloToon Unreal Engine(UE5) port.

Release history

Feel free to skip this section if you just want to use the current version of NiloToon right now.

Similar to URP's Road map, we record some recent release history here in text.
For full change log, see change log

New in NiloToon 0.17.0

Released on 2025-06-13

- (Big change)All NiloToon passes fully support Render Graph!!!!!!!!!!!! Users can enable RenderGraph now and should not see any visual difference.
- NiloToon Character shader: support Unity6.1 Forward+ / Deferred+ correctly (CLUSTER_LIGHT_LOOP)

New in NiloToon 0.16.0

Released on 2024-02-19

In short, compared to NiloToon 0.15.15, NiloToon 0.16.0 introduces these significant changes:
- [breaking change] Rewrote the code for additional lights; all additional lights are now treated as the main light by default, allowing you to light a character nicely with just point/spot lights (great for interiors!)
- Added a new 'NiloToonCharacterLightController' script, which allows you to control how each character receives lighting(e.g, tint or add color to main light for a few target characters)
- Added a new 'NiloToonAdditionalLightStyleVolume' volume, which enables you to control how all lights affects all characters globally
- Added a new 'NiloToonLightSourceModifier' script, which enables you to control how each light affects all characters. For example, you can create a few point lights with different roles -> (1)a normal point light (2)a point light without rim light (3)a point light with only rim light (4)a point light that doesn't affect nilotoon characters...For (4), it is especially handy if you'd rather not use Unity URP's rendering masks, since URP's rendering layers mandate individual settings for each renderer's layer, this can be time-consuming and clutters your scene's game objects and prefab files.
- Many new options in the menu "GameObject > Create Other > NiloToon"
- Many new options in the menu "GameObject > Light > NiloToon"
- Introduced 'soft shadow' for shadow maps to reduce aliasing, enabled by default (medium quality)
- Updated the NiloToon Character shader: Basemap Stacking Layer 1-10 now includes +7 UVs (UV1-4, MatcapUV, CharBoundUV, ScreenSpaceUV) and 4 blend mode options (normal, add, screen, multiply). For example, you can create a basemap stacking layer using matcapUV with multiply or add blend methods.
- Updated NiloToonPerCharacterRenderController's' 'Basemap Override' group: +7 UVs (UV1-4, MatcapUV, CharBoundUV, ScreenSpaceUV), and 4 blend mode options (normal, add, screen, multiply)
- Added a new 'Color Fill' group to NiloToonPerCharacterRenderController, a utility function to redraw the character with different effects (e.g., x-ray vision, show characters that are behind a wall similar to CS2/Valorant)
- Added a new 'Generate Smoothed normal?' toogle to NiloToonPerCharacterRenderController, you can enable it for non-fbx prefabs(e.g., vrm character), so those non-fbx character prefabs can still have a similar outline quality as fbx characters
- Fixed all memory leaks and GC allocation issues that were found
- Significant CPU optimization when multiple characters are visible (approximately 1.72x CPU speed up for a test scene with 30 unique characters and 800 unique characters materials)
- HLSL code is now much easier to read in Rider

Note from the Developer (Unity6/RenderGraph/Warudo)
- NiloToon 0.16.x is confirmed to be the last version that still supports Unity2021.3.
- In NiloToon 0.17.0, we will completely remove support for Unity2021.3 to begin working on Unity6 (RenderGraph) support.
- After a few more 0.16.x releases, we will provide a stable NiloToon 0.16.x version to Warudo to be used as the official NiloToon version for Warudo(Unity2021.3).

Note from the Developer (Additional Light Rewrite)
Skip this message if you do not need to light characters using point/spot lights.
NiloToon 0.16.0 includes a major rewrite of the code for additional lights!

In the previous version, NiloToon 0.15.15, the treatment of additional lights was similar to Unity's method, where the lights' colors were additively blended onto the character. As a Unity URP shader, our original design aimed to align with Unity's URP additional light behavior as closely as possible, although we know that it is not the best for toon shader.

However, the additive light method only looked good under very carefully set up conditions, like this [video](https://youtu.be/ST9qNEfPrmY?si=98mqByySiRE7kcTO). In most regular use cases, NiloToon 0.15.15's point/spot lights could easily make the character appear overly bright or look strange.

Therefore, in NiloToon 0.16.0, we decided it was time to make a significant breaking change to ensure additional lights provide good results by default. Now, any additional lights are treated the same as the main light, so no matter how you light the character and regardless of the light type used, the lighting result will always maintain the same quality as the main directional light. This means scenes that mainly use point/spot lights, such as interiors, rooms, caves, etc., will now be much easier to light characters nicely and consistently with NiloToon 0.16.0.

*If you want to produce the old additional light result, you can:
- In Nilotoon Additional Light Style Volume: set color & direction = 0
- (optional)In Nilotoon Cinematic Rim Light Volume: disable “auto fix unsafe style”
——————————中文——————————
简而言之，与 NiloToon 0.15.15 相比，NiloToon 0.16.0 引入了以下重大变化：
- [重大变更] 重写了额外灯光的代码；现在所有额外的灯光默认被当作主灯光处理，只需使用点光源/聚光灯就能很好地照亮角色（非常适合室内！）
- 添加了一个新的 `NiloToonCharacterLightController` 脚本，允许你控制每个角色接收光照的方式（例如，为几个目标角色给主光源着色或添加颜色）
- 添加了一个新的 `NiloToonAdditionalLightStyleVolume` 体积，使你能够全局控制所有灯光如何影响所有角色
- 添加了一个新的 `NiloToonLightSourceModifier` 脚本，使你能够控制每个灯光如何影响所有角色。例如，你可以创建几个具有不同角色的点光源 -> （1）一个普通点光源（2）一个没有边缘光的点光源（3）一个只有边缘光的点光源（4）一个不影响 nilotoon 角色的点光源...对于（4），如果你不愿意使用 Unity URP 的渲染遮罩，这特别方便，因为 URP 的渲染层要求为每个渲染器的层进行单独设置，这可能既费时又会使你的场景的游戏对象和预制文件变得杂乱无章。
- 菜单 "GameObject > Create Other > NiloToon" 中增加了许多新选项
- 菜单 "GameObject > Light > NiloToon" 中增加了许多新选项
- 引入了 'soft shadow'（柔和阴影）用于阴影图，以减少走样，默认启用（中等质量）
- 更新了 NiloToon 角色着色器：基础图堆叠层 1-10 现在包括 +7 UVs（UV1-4, MatcapUV, CharBoundUV, ScreenSpaceUV）和 4 种混合模式选项（正常，添加，屏幕，乘法）。例如，你可以使用 matcapUV 与乘法或添加混合方法创建基础图堆叠层。
- 更新了 NiloToonPerCharacterRenderController 的 'Basemap Override' 组：+7 UVs（UV1-4, MatcapUV, CharBoundUV, ScreenSpaceUV），和 4 种混合模式选项（正常，添加，屏幕，乘法）
- 在 NiloToonPerCharacterRenderController 中添加了新的 'Color Fill' 组，一个实用功能，用于用不同效果重新绘制角色（例如，X光视觉，显示墙后的角色，类似 CS2/Valorant）
- 在 NiloToonPerCharacterRenderController 中添加了新的 'Generate Smoothed normal?' 开关，你可以为非 fbx 预制件启用它（例如，vrm 角色），以便这些非 fbx 角色预制件仍然可以拥有与 fbx 角色相似的轮廓质量
- 修复了发现的所有内存泄漏和 GC 分配问题
- 当多个角色可见时进行了显著的 CPU 优化（在一个测试场景中，有 30 个独特角色和 800 个独特角色材质，大约提速了 1.72 倍）
- 现在在 Rider 中 HLSL 代码更易于阅读

开发者说明（Unity6/RenderGraph/Warudo）
- 确认 NiloToon 0.16.x 是最后一个仍然支持 Unity2021.3 的版本。
- 在 NiloToon 0.17.0 中，我们将完全移除对 Unity2021.3 的支持，以开始支持 Unity6（RenderGraph）。
- 在再发布几个 0.16.x 版本后，我们将提供一个稳定的 NiloToon 0.16.x 版本给 Warudo，作为 Warudo（Unity2021.3）的官方 NiloToon 版本。

开发者说明（附加灯光重写）
如果你不需要使用点光源/聚光灯来照亮角色，请跳过此消息。
NiloToon 0.16.0 包含对附加灯光代码的重大重写！

在之前的版本 NiloToon 0.15.15 中，附加灯光的处理方式类似于 Unity 的方法，灯光的颜色会加性地混合到角色上。作为一个 Unity URP 着色器，我们的原始设计旨在尽可能地与 Unity 的 URP 附加灯光行为保持一致，尽管我们知道这对卡通着色器来说并不是最好的。

然而，加性光方法只有在非常仔细设置的条件下才看起来不错，就像这个[视频](https://youtu.be/ST9qNEfPrmY?si=98mqByySiRE7kcTO)。在大多数常规用例中，NiloToon 0.15.15 的点光源/聚光灯可能会很容易使角色显得过分明亮或看起来奇怪。

因此，在 NiloToon 0.16.0 中，我们决定是时候做出重大的突破性变化，以确保附加灯光默认提供良好的结果。现在，任何附加灯光都被视为与主灯光相同，所以无论你如何照亮角色，无论使用哪种类型的灯光，照明结果始终会保持与主方向光相同的质量。这意味着主要使用点光源/聚光灯的场景，如室内、房间、洞穴等，现在使用 NiloToon 0.16.0 很容易且一致地很好地照亮角色。

*如果你想产生旧的附加光结果，你可以：
- 在 NiloToon Additional Light Style Volume 中：设置颜色和方向 = 0
-（可选）在 NiloToon Cinematic Rim Light Volume 中：禁用 “auto fix unsafe style”
——————————日本語——————————

簡単に言うと、NiloToon 0.15.15に比べて、NiloToon 0.16.0は以下の重要な変更を導入しました:
- [破壊的変更] 追加ライトのコードを書き直しました。すべての追加ライトは、デフォルトでメインライトとして扱われるようになり、ポイントライト/スポットライトだけでキャラクターをきれいに照らすことができます（インテリアに最適！）
- 新しい「NiloToonCharacterLightController」スクリプトを追加しました。これにより、各キャラクターがライティングをどのように受け取るかを制御できます（例えば、いくつかのターゲットキャラクターに対してメインライトの色を調整したり、色を追加したりすることができます）
- 新しい「NiloToonAdditionalLightStyleVolume」ボリュームを追加しました。これにより、すべてのライトがすべてのキャラクターに対してグローバルにどのように影響するかを制御できます
- 新しい「NiloToonLightSourceModifier」スクリプトを追加しました。これにより、各ライトがすべてのキャラクターにどのように影響するかを制御できます。例えば、(1)通常のポイントライト(2)リムライトなしのポイントライト(3)リムライトのみのポイントライト(4) nilotoonキャラクターに影響を与えないポイントライト...(4)は特に便利です。Unity URPのレンダリングマスクを使用したくない場合に役立ちます。URPのレンダリングレイヤーは、各レンダラーのレイヤーに個別の設定が必要であり、これは時間がかかり、シーンのゲームオブジェクトやプレハブファイルを散らかす可能性があります。
- メニュー「GameObject > Create Other > NiloToon」に多くの新しいオプションを追加
- メニュー「GameObject > Light > NiloToon」に多くの新しいオプションを追加
- エイリアシングを減らすために「ソフトシャドウ」をシャドウマップに導入し、デフォルトで有効になっています（中品質）
- NiloToonキャラクターシェーダーを更新しました：Basemap Stacking Layer 1-10には+7 UVs（UV1-4、MatcapUV、CharBoundUV、ScreenSpaceUV）と4つのブレンドモードオプション（ノーマル、アド、スクリーン、マルチプライ）が含まれるようになりました。例えば、matcapUVを使ってmultiplyまたはaddブレンド方法でbasemap stacking layerを作成することができます。
- NiloToonPerCharacterRenderControllerの「Basemap Override」グループを更新：+7 UVs（UV1-4、MatcapUV、CharBoundUV、ScreenSpaceUV）、および4つのブレンドモードオプション（ノーマル、アド、スクリーン、マルチプライ）
- NiloToonPerCharacterRenderControllerに新しい「Color Fill」グループを追加しました。これは、異なる効果（例えば、X線ビジョン、壁の後ろにいるキャラクターをCS2/Valorantのように表示するなど）でキャラクターを再描画するためのユーティリティ機能です。
- NiloToonPerCharacterRenderControllerに「Generate Smoothed normal?」トグルを新たに追加しました。これを有効にすると、非fbxプレハブ（例えば、vrmキャラクター）でも、fbxキャラクターと同様のアウトライン品質を持つことができます。
- 発見されたすべてのメモリリークとGC割り当ての問題を修正しました。
- 複数のキャラクターが可視状態のときのCPU最適化が大幅に改善されました（約1.72倍のCPUスピードアップ、30個のユニークなキャラクターと800個のユニークなキャラクターマテリアルを含むテストシーンで）。
- HLSLコードがRiderで読みやすくなりました。

開発者からの注意（Unity6/RenderGraph/Warudo）
- NiloToon 0.16.xはUnity2021.3をサポートする最後のバージョンであることが確認されています。
- NiloToon 0.17.0では、Unity2021.3のサポートを完全に削除して、Unity6（RenderGraph）サポートに取り組むことになります。
- あと数回の0.16.xリリースの後、Warudo（Unity2021.3）で公式のNiloToonバージョンとして使用される安定したNiloToon 0.16.xバージョンを提供します。

開発者からの注意（追加ライトの書き換え）
ポイントライト/スポットライトを使用してキャラクターを照らす必要がない場合は、このメッセージをスキップしてください。
NiloToon 0.16.0には、追加ライトのためのコードの大幅な書き換えが含まれています！

前バージョンのNiloToon 0.15.15では、追加ライトの扱いはUnityの方法に似ており、ライトの色はキャラクターに加算的にブレンドされていました。Unity URPシェーダーとして、私たちの元のデザインはできるだけUnityのURP追加ライトの動作に沿うことを目指していましたが、トゥーンシェーダーには最適でないことは知っていました。

しかし、加算ライトの方法は、[このビデオ](https://youtu.be/ST9qNEfPrmY?si=98mqByySiRE7kcTO)のように非常に慎重にセットアップされた条件下でのみ見栄えが良かったです。ほとんどの通常の使用例では、NiloToon 0.15.15のポイントライト/スポットライトはキャラクターを過度に明るくしたり、奇妙に見せたりすることが容易でした。

したがって、NiloToon 0.16.0では、追加ライトがデフォルトで良い結果を提供するようにするための重要な変更を加える時が来たと判断しました。これで、追加ライトはメインライトと同じように扱われるので、キャラクターを照らす方法や使用するライトの種類に関係なく、照明結果は常にメインの方向光と同じ品質を維持します。つまり、インテリア、部屋、洞窟など、主にポイントライト/スポットライトを使用するシーンは、NiloToon 0.16.0を使ってキャラクターを簡単かつ一貫して綺麗に照らすことができるようになりました。

*以前の追加ライトの結果を出力したい場合は、以下の操作を行うことができます:
- NiloToon Additional Light Style Volumeで、色と方向を0に設定します。
- （オプション）NiloToon Cinematic Rim Light Volumeで、「auto fix unsafe style」を無効にします

——----------

간단히 말해서, NiloToon 0.15.15에 비해 NiloToon 0.16.0은 이러한 중요한 변경 사항을 소개합니다:
- [주요 변경] 추가 라이트 코드를 재작성했습니다; 이제 모든 추가 라이트는 기본적으로 메인 라이트로 처리되어, 점/스팟 라이트만으로도 캐릭터를 멋지게 조명할 수 있게 되었습니다(인테리어에 적합!).
- 새로운 'NiloToonCharacterLightController' 스크립트를 추가했습니다. 이 스크립트를 통해 각 캐릭터가 조명을 받는 방식을 제어할 수 있습니다(예: 몇몇 타겟 캐릭터들에게 메인 라이트 색상을 틴트하거나 추가하는 등).
- 새로운 'NiloToonAdditionalLightStyleVolume' 볼륨을 추가하여, 전체 캐릭터에 대한 모든 라이트의 영향을 전역적으로 제어할 수 있게 되었습니다.

- 새로운 'NiloToonLightSourceModifier' 스크립트를 추가하여, 각 라이트가 모든 캐릭터에 미치는 영향을 제어할 수 있게 되었습니다. 예를 들어, 다양한 역할을 가진 몇 개의 포인트 라이트를 생성할 수 있습니다 -> (1)일반 포인트 라이트 (2)림 라이트가 없는 포인트 라이트 (3)림 라이트만 있는 포인트 라이트 (4)nilotoon 캐릭터에 영향을 주지 않는 포인트 라이트... (4)는 특히 Unity URP의 렌더링 마스크를 사용하지 않으려는 경우 유용합니다. URP의 렌더링 레이어는 각 렌더러의 레이어에 대한 개별 설정을 요구하기 때문에, 이는 시간이 많이 걸리고 장면의 게임 오브젝트 및 프리팹 파일을 복잡하게 만들 수 있습니다.
- 메뉴 "GameObject > Create Other > NiloToon"에 많은 새로운 옵션을 추가했습니다.
- 메뉴 "GameObject > Light > NiloToon"에 많은 새로운 옵션을 추가했습니다.
- 앨리어싱을 줄이기 위해 'soft shadow'를 그림자 맵에 도입했으며, 기본적으로 활성화되어 있습니다(중간 품질).
- NiloToon 캐릭터 쉐이더를 업데이트했습니다: 베이스맵 스택 레이어 1-10에 이제 +7 UV(UV1-4, MatcapUV, CharBoundUV, ScreenSpaceUV)와 4가지 블렌드 모드 옵션(노멀, 애드, 스크린, 멀티플라이)이 포함되어 있습니다. 예를 들어, matcapUV를 사용하여 멀티플라이나 애드 블렌드 방식으로 베이스맵 스택 레이어를 만들 수 있습니다.
- NiloToonPerCharacterRenderController의 'Basemap Override' 그룹을 업데이트했습니다: +7 UV(UV1-4, MatcapUV, CharBoundUV, ScreenSpaceUV)와 4가지 블렌드 모드 옵션(노멀, 애드, 스크린, 멀티플라이).
- NiloToonPerCharacterRenderController에 새로운 'Color Fill' 그룹을 추가했습니다. 다른 효과로 캐릭터를 다시 그릴 수 있는 유틸리티 기능입니다(예: X레이 비전, CS2/Valorant와 유사하게 벽 뒤에 있는 캐릭터를 보여줌).
- NiloToonPerCharacterRenderController에 새로운 'Generate Smoothed normal?' 토글을 추가했습니다. 이를 비fbx 프리팹(예: vrm 캐릭터)에 활성화할 수 있어, 이러한 비fbx 캐릭터 프리팹도 fbx 캐릭터와 유사한 퀄리티의 윤곽을 유지할 수 있습니다.
- 발견된 모든 메모리 누수와 GC 할당 문제를 수정했습니다.
- 여러 캐릭터가 보일 때 상당한 CPU 최적화를 달성했습니다(30개의 고유 캐릭터와 800개의 고유 캐릭터 재질을 사용한 테스트 장면에서 약 1.72배의 CPU 속도 향상).
- 이제 HLSL 코드가 Rider에서 훨씬 쉽게 읽을 수 있습니다.

개발자의 메모 (Unity6/RenderGraph/Warudo)
- NiloToon 0.16.x는 여전히 Unity2021.3을 지원하는 마지막 버전임이 확인되었습니다.
- NiloToon 0.17.0에서는 Unity2021.3에 대한 지원을 완전히 중단하고 Unity6 (RenderGraph) 지원 작업을 시작합니다.
- 0.16.x 버전을 몇 개 더 출시한 후, Warudo에 사용될 공식 NiloToon 버전으로 안정적인 NiloToon 0.16.x 버전을 제공할 예정입니다(Unity2021.3용).

개발자의 메모 (추가 조명 재작성)
점/스팟 라이트를 사용하여 캐릭터에 조명을 제공할 필요가 없다면 이 메시지는 건너뛰세요.
NiloToon 0.16.0은 추가 라이트 코드에 대한 주요 재작성을 포함하고 있습니다!

이전 버전인 NiloToon 0.15.15에서는 추가 라이트의 처리가 Unity의 방식과 유사했으며, 라이트 색상이 캐릭터에 가산적으로 혼합되었습니다. Unity URP 쉐이더로서, 우리의 원래 디자인은 가능한 한 Unity의 URP 추가 조명 동작과 일치하려고 했지만, 툰 쉐이더에는 최적이 아니라는 것을 알고 있었습니다.

그러나 가산 조명 방법은 [이 비디오](https://youtu.be/ST9qNEfPrmY?si=98mqByySiRE7kcTO)와 같이 매우 신중하게 설정된 조건 하에서만 좋아 보였습니다. 대부분의 일반적인 사용 사례에서 NiloToon 0.15.15의 점/스팟 라이트는 쉽게 캐릭터를 과도하게 밝게 만들거나 이상하게 보이게 했습니다.

따라서 NiloToon 0.16.0에서는 추가 라이트가 기본적으로 좋은 결과를 제공하도록 중요한 변경을 결정할 때라고 생각했습니다. 이제 모든 추가 라이트는 주 라이트와 동일하게 처리되므로, 캐릭터에 어떻게 조명을 하든, 사용하는 라이트의 유형에 상관없이, 조명 결과는 항상 메인 방향성 라이트와 동일한 품질을 유지할 것입니다. 이것은 내부, 방, 동굴 등 주로 점/스팟 라이트를 사용하는 장면들이 이제 NiloToon 0.16.0으로 캐릭터를 아름답고 일관성 있게 조명하는 것이 훨씬 쉬워졌음을 의미합니다.

*예전의 추가 라이트 결과를 만들고 싶다면, 다음과 같이 할 수 있습니다:
- Nilotoon Additional Light Style Volume에서: 색상 & 방향 = 0으로 설정
- (선택사항) Nilotoon Cinematic Rim Light Volume에서: "auto fix unsafe style"을 비활성화

New in NiloToon 0.15.0
Released on 2023-10-24
- UI rework on NiloToon character shader and NiloToonShaderStrippingSettingSO
- Tested with all 5 available Unity versions from the UnityHub(Unity2021.3.31f1 ~ Unity2023.3.0a10)
- Solved all GC Alloc and RTHandle performance problem
- Build is now faster, smaller, and has lower shader memory usage if user rely on the default stripping settings
- We expect NiloToon 0.15.x will become the next stable version, replacing 0.13.6 within the year 2023

New in NiloToon 0.14.0
Released on 2023-08-23
This version now supports Unity2021.3 to Unity2023.2, and supporting all URP's new features.

[NiloToon now supports new URP features in Unity2021.3LTS]

A1. Point Light Shadows
A2. Deferred Rendering Support (NiloToon shaders will still render as Forward in Deferred Rendering with 8 light per renderer limitation)
A3. Decals
A4. Light Cookies
A5. Reflection Probe Blending
A6. Reflection Probe Box Projection

[NiloToon now supports new URP features in Unity2022.3LTS]
B1. Forward+ Support (Forward+ can render a maximum of 256 lights per camera, instead of Forward's maximum of 8 lights per renderer)
B2. Rendering Layer
B3. Light layer
B4. Decal layer
B5. LOD Crossfade (Only NiloToonEnvironment shader has this feature supported, since it is not very useful to apply LOD Crossfade to characters)
B6. TAA support (alternative of MSAA when MSAA is too slow in high resolution render)
B7. All types of light cookies

[NiloToon added feature]
C1. a new volume - NiloCinematicRimLightVolume, which will turn all additional lights into rim light for NiloToonCharacters materials
C2. a new MonoBehaviour - NiloToonCharacterMainLightOverrider, you can override character's mainlight color and direction, without editing any URP's directional light
C3. NiloToonCharacter material can now be used independently without a NiloToonPerCharacterRenderController script attached to the prefab(but when used without having a NiloToonPerCharacterRenderController to control that material, that NiloToonCharacter material's NiloToon self shadow map and average URP shadow will not be rendered)
C4. NiloToonCharacter material's Classic outline can now be rendered for transparent queue(2501-5000) materials. You can use "Transparent(ZWrite)/Outline" and "Opaque/Outline" surface type preset materials together now, outlines from both type of materials will be displayed correctly and not be blocked by each other anymore
C5. Rewrite NiloToon self shadow map algorithm, it will not show shadow artifact easily now when multiple characters are visible and far to each other.
C6. Auto update NiloToonPerCharacterRenderController's allRenderers, any child transform change will trigger an update in edit mode. The update will consider child NiloToonPerCharacterRenderController also
C7. NiloToonBloom added HSV control, high saturation bloom can produce another art style
C8. Rewrite NiloToonCharacter's surfaceType preset, allow you to setup a material more easily with a better dropmenu
C9. NiloToonCharacter material UI update (upgrade to LWGUI 1.11.1), including a new toolbar and group search mode

[NiloToon important bug fix]
D1. fix all possible crash that we know when building the player(compiling shader variants)
D2. now NiloToon will have 0 GC alloc per frame(unless using "NiloToonRendererRedrawer" or "Terrain Crash Safe Guard" enabled)
——————————————中文——————————————
- 支援Unity2021.3~2023.2

[支援所有URP(Unity2021.3)版本的新功能]
A1. Point Light Shadows
A2. Deferred Rendering Support (NiloToon shaders 在 Deferred Rendering 仍然會以 Forward 渲染(有8 light per renderer 的限制))
A3. Decals
A4. Light Cookies
A5. Reflection Probe Blending
A6. Reflection Probe Box Projection

[支援所有URP(Unity2022.3)版本的新功能]
B1. 支援Forward+, 可以令角色收到256個additional light
B2. 支援URP rendering layer
B3. 支援light的rendering layer
B4. 支援decal的rendering layer
B5. LOD crossfade(只有NiloToonEnvironment shader有包括支持, 因為角色的LOD crossfade似乎不太常用)
B6. 支援TAA (當你不想使用MSAA(例如在4K渲染時太慢), 可以試試TAA)
B7. 支援所有light cookie

[NiloToon重要新功能]
C1. 增加了一個「令additional light變rim light」的volume - NiloCinematicRimLightVolume, 理論上角色可以隨便接受大量spotlight/point light但仍會好看不會過亮

C2. 增加「NiloToonCharacterMainLightOverrider」MonoBehaviour，可以使用新的gameobject來改變角色的main light燈光方向和顏色，而不需要改動URP的任何light
C3. NiloToonCharacter material可以獨立使用，不需要依賴nilotoon per character script (但會NiloToon的average shadow如selfshadowmap不會渲染)
C4. 內層Opauqe+Outline物體 和 外層半透明+Outline物體，都能同時出現描邊，不會互相檔住
C5. 重寫了NiloToon self shadow mapping，在多角色而且各自距離比較遠時都不會再出現難看的影子問題
C6. 自動設置NiloToonPerCharacterRenderController的allRenderers，在child transform有任何變動時會在edit mode自動更新，會考慮child NiloToonPerCharacterRenderController
C7. NiloToonBloom 增加 HSV 控制，例如高saturation的bloom可以做出新的bloom風格
C8. 重寫 NiloToonCharacter的surfaceType preset，新的dropmenu使用上會更直觀
C9. NiloToonCharacter material UI 更新 (使用 LWGUI 1.11.1), 增加新的 toolbar 和 search mode

[NiloToon重要修正]
D1. 修正所有build時會令unity crash的問題
D2. 修正所有C# GC，現在是0 GC Alloc per frame(除非使用 "NiloToonRendererRedrawer" 或 "Terrain Crash Safe Guard")

VertExmotion Support

1.open NiloToonCharacter_ExtendDefinesForExternalAsset.hlsl

2.enable the setting to support that asset(by changing 0 to 1)

```
10  ////////////////////////////////////////////////////////////////////////
11  // To NiloToonURP user:
12  // edit 0->1 if you need NiloToon to include target asset's support
13  ////////////////////////////////////////////////////////////////////////
14  #define NILOTOON_SUPPORT_VERTEXMOTION 1
```

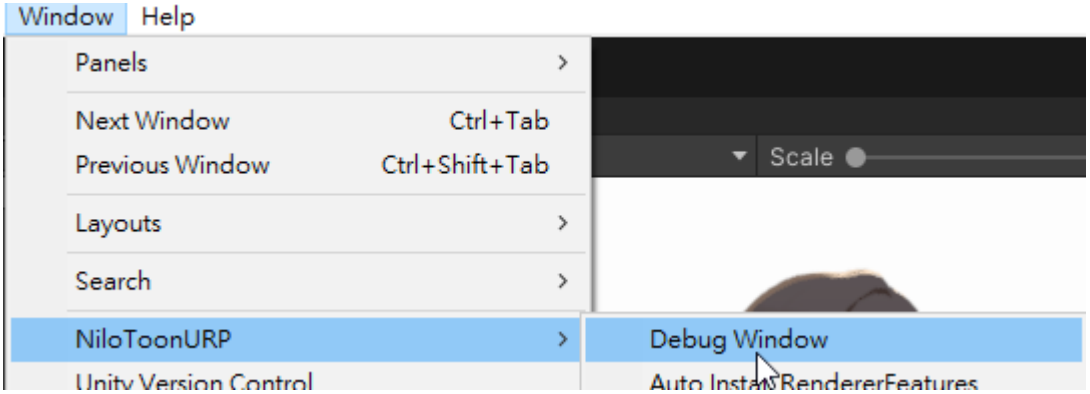
*If your asset's support does not exist in NiloToon, you can always contact us!

Keep Play Mode Mat Edit

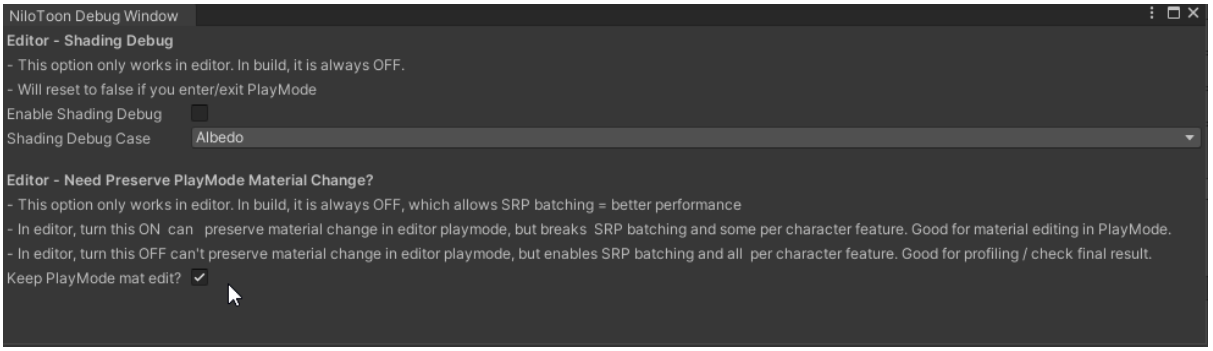
By default, NiloToon will convert all materials to material instances in editor playmode and in build, by calling **Renderer.materials** in C#. This will allow NiloToon to control the materials and avoid the use of **material property block** at the same time, which means **SRP batching** will work in playmode, and higher CPU performance will be produced in build.

There are some situation that you don't care CPU performance, and just want to keep playmode's material edit in editor, in this situation you can:

1.Open Window/NiloToonURP/Debug window,



2.enable **Keep PlayMode mat edit?**



Enabling **Keep PlayMode mat edit?** toggle will make NiloToon NOT turning material into material instance/clone in editor playmode anymore, so the materials used by the character and the materials inside the project window are the same(any material change in play mode will be kept, any material change in project window will affect characters in play mode).

But enabling **Keep PlayMode mat edit?** will make some **per character features** disabled in editor playmode, and it will **prevent SRP batching** due to the use of **material property block**, which means much higher CPU cost and **lower fps**

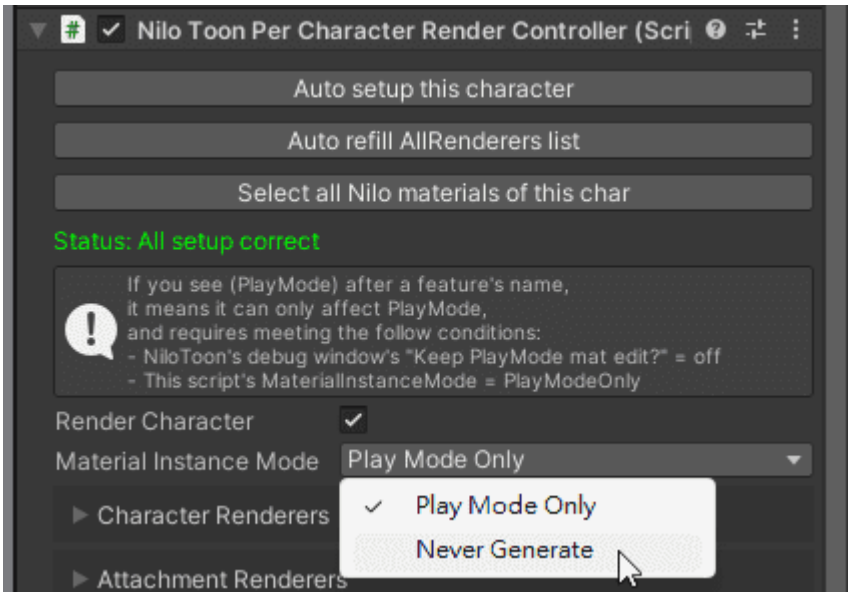
When should I enable it?

Enable Keep PlayMode mat edit? if you are:
- **material artists** that want to enable **Keep PlayMode mat edit?** when editing materials, and you don't care lower CPU performance and fps

Disable Keep PlayMode mat edit? if you are:
- **programmers/tech artist/engineer** making optimization/profiling, disable **Keep PlayMode mat edit?** will make the editor playmode reflect the actual build performance closer
- **users** that want the editor playmode having **higher fps**

What about In build?

Please note that **PlayMode mat edit?** will not have any effect in build.
To make the material not become a material instance in build, you need to use **NiloToonPerCharacterRenderController**'s **Material Instance Mode** option.
This option is applied per character, and will affect both editor playmode and build.



Long Shader build time

If you are doing your first build for a platform, or first build after NiloToon's shader update, it may take a long time to compile all new shader variants.

To reduce the number of shader variants(reduce build time, build size and runtime shader memory), see:

- NiloToon shader uses too much memory / build time is too long! How to reduce it?.

*if it is still slow after the first build, it is not normal, contact us!

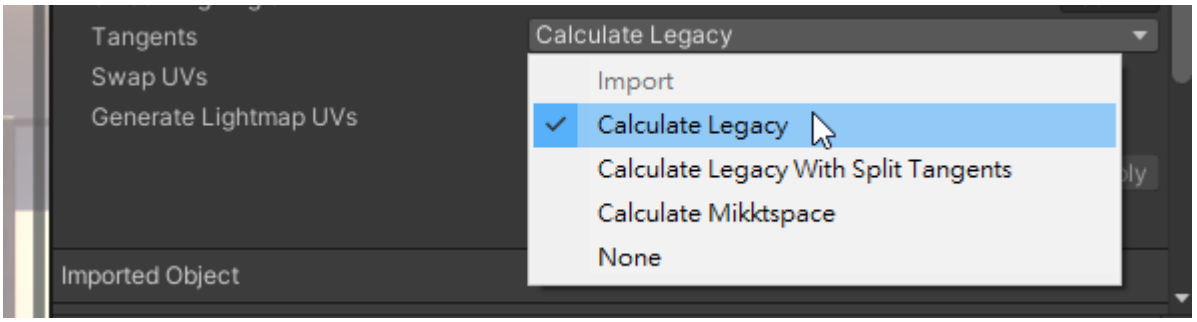
Model missing tangent

If you see this **red** **error** in editor console,

[Error] NiloToon can't bake smooth normal, because _____ (UnityEngine.Mesh)(Assets/*/_____.fbx) don't have tangents, you can let Unity calculate tangents for you (click .fbx->Model tab->Tangents = Calculate ...).

This problem appears if the model mesh doesn't have normal or tangent.

Make sure your model has valid normal and tangent:
- if your model has **tangents** in .fbx, use **Import**
- if your model doesn't have **tangents** in .fbx, click the .fbx, set **Tangents = Calculate Mikktspace or other calculate options**)



Chara Shadowmap explain

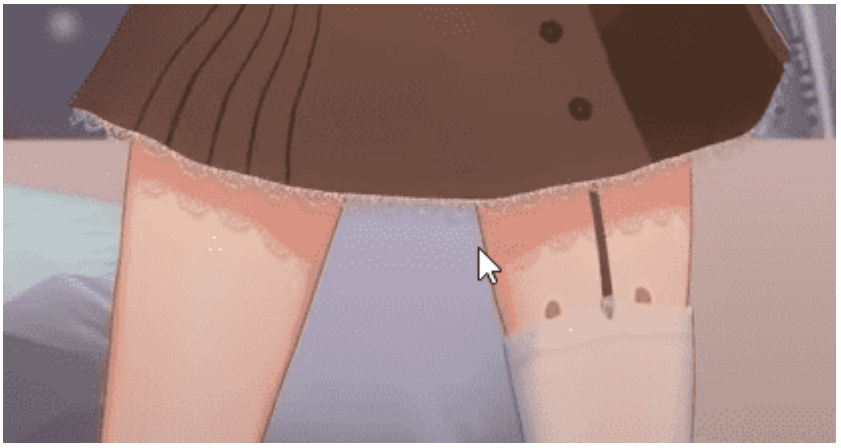
To explain NiloToon's self-shadow map, let's look at a few NiloToon videos as examples. In these videos:

- 【3DMV】SOS /covered by すいみゃ【シャニマス】
- 【3D初披露】墜入霓的世界，我們來了！｜澪Rei、@橙Yuzumi、@煌Kirali #霓NEOn3DSHOWCASE
- 学園アイドルマスター（学マス）- NiloToon test 240521
- more examples(https://github.com/ColinLeung-NiloCat/UnityURPToonLitShaderExample)

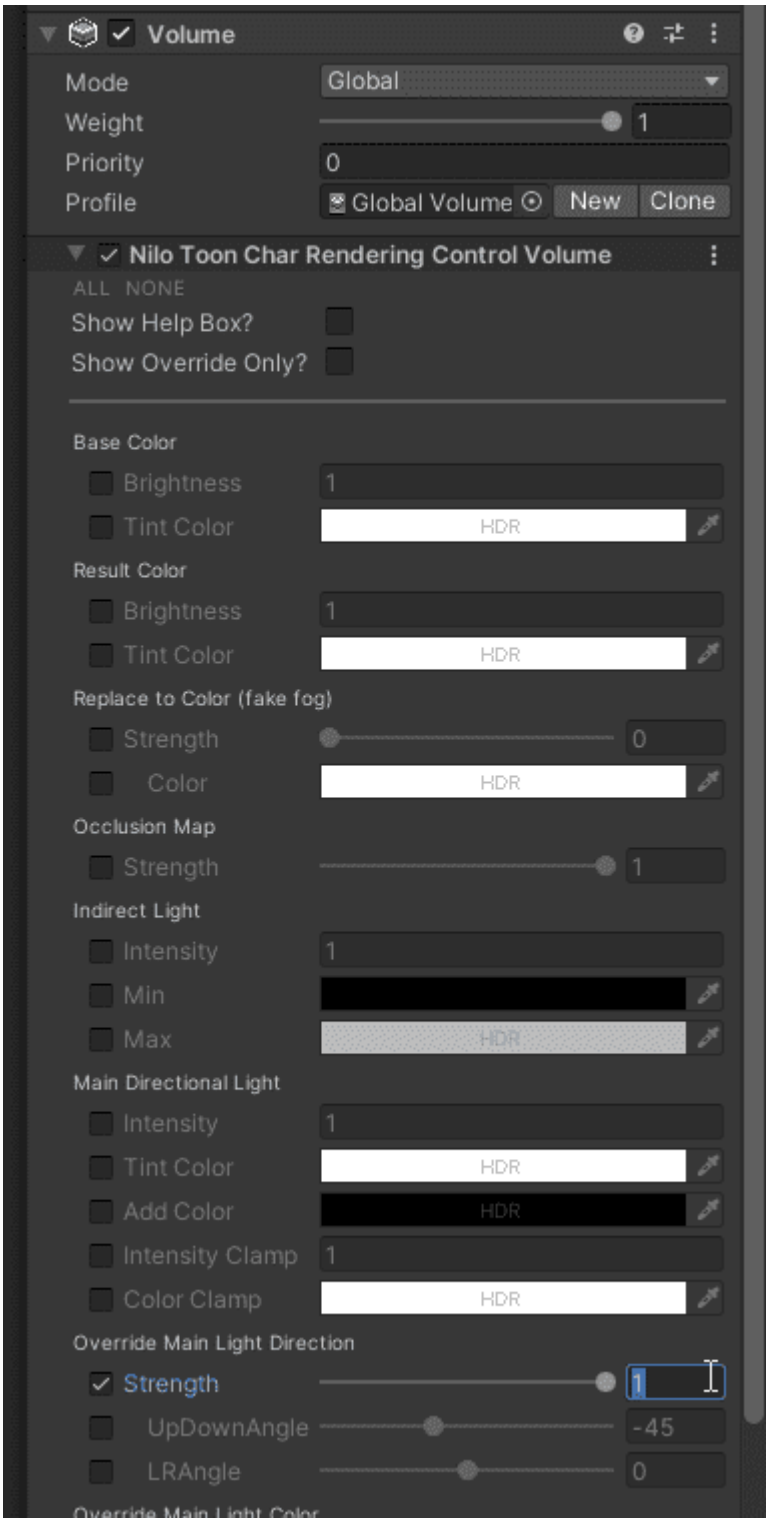
Graphics result / goal

The **skirt** produces an extremely sharp dynamic shadow on the **leg** using NiloToon's self-shadow map system, this skirt shadow is **critical** for **idol outfit** in concert type events.

Any URP shadow settings will not affect NiloToon's shadow results, as NiloToon has its own shadow map system.



You can produce this type of skirt shadow by controlling the **directional light** **angle**(e.g., 45 degrees downward), or using the **Override Main Light Direction** in **NiloToonCharRenderingControlVolume** (see the image below)



View Source code

If you are not a programmer trying to write your own shadowmap system, you may skip the following sections.

To read the NiloToon source code that produces the above self-shadow map shadows, you can open NiloToonURP's:

- **NiloToonCharSelfShadowMapRTPass**.cs: a C# scriptable render pass that draws to a new ShadowMap RenderTexture from the shadow camera's point of view (not an actual camera), only drawing the character's depth
- **NiloToonCharacter**.shader -> **NiloToonSelfShadowCaster** pass: the shader pass used in the above NiloToonCharSelfShadowMapRTPass, used for drawing the character's depth data in the ShadowMap
- **NiloToonCharacter**.shader -> **ForwardLit** pass: a regular lit color pass that samples NiloToonCharSelfShadowMapRTPass's resulting ShadowMap, performs filtering (soft shadow blur), and calculates the final shadow area

Why shadow so sharp?

NiloToon's shadow map system only considers characters and uses a close-fit tight sphere bound to render them, so the **shadow map area utilization** is much higher than URP's shadow map system.
*Higher shadowmap area utilization = less waste in shadowmap area

For example:
- a 2048 size **URP shadow map** for the whole scene(e.g., cover a complete concert stage) will result in a very blurry shadow for character, quality is not usable for character since character only use a small part of the shadow map
- a 2048 size **NiloToon shadow map** for 5 NiloToon characters will result in an acceptable shadow for character, since characters use the shadowmap fully
- a 2048 size **NiloToon shadow map** for 1 NiloToon character will result in a very sharp shadow for that character
- a 4096 size **NiloToon shadowmap** for 1 character, will result in a close to perfect shadow for that character, especially when NiloToon's soft shadow is enabled
- a 8192 or above **NiloToon shadowmap** for 1 character, will result in a perfect shadow even for close up, usually used for offline rendering for trailer/MV

*To understand the shadow map area utilization, you can use frame debugger to view the drawing of NiloToon' self shadow map, the less space wasted, the higher the shadow map utilization.

If you need to run NiloToon shader on mobile, we have a demo apk for you already:
https://drive.google.com/file/d/1CBSYiniHTAaNtlkd5X7TUxjAsQ7eBNxK/view

Mobile friendly?

Yes, for mobile, shadow map can have low GPU cost if:
- shadow map size is small (<=1024, or 2048 for powerful phone)
- shadow map format is correct (RenderTextureFormat.Shadowmap, depthBufferBits = 16)
- do not perform custom soft shadow filtering, relying on hardware shadow blur only (SAMPLE_TEXTURE2D_SHADOW with SAMPLER_CMP can do a slight shadow blur using hardware bilinear filter, it has 0 GPU cost to produce the blur)

The GPU cost / quality of shadow map is scalable, for example:
- for mobile: 1024~2048
- for PC/console: 4096~8192
- for offline MV rendering: 8192~16k, it will produce sharp shadow even in extreme close up, great for video production like song MV

Workflow Benefit

The shadowmap method requires **no setup to work**, since everything is calculated real-time using the final character skinned mesh per frame.
For example, in all of the above example videos, NiloToon's self shadow map will work without any special setup for the character prefab after the init auto NiloToon setup.

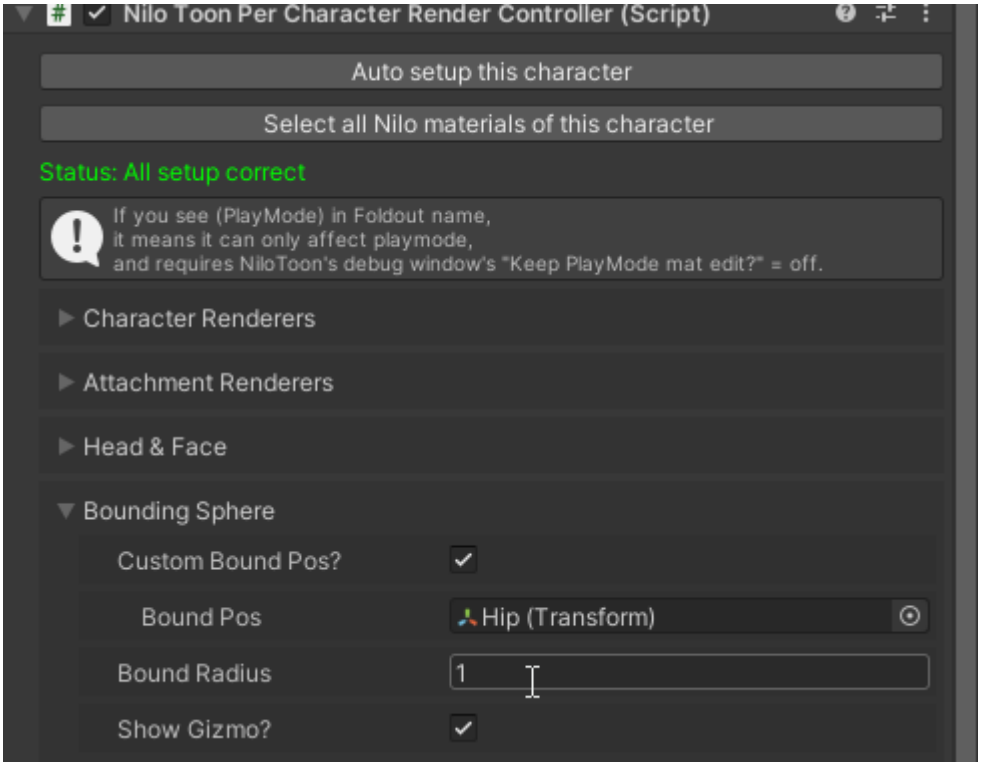
For:
1. **sharp skirt shadow casted on the leg**
2. **arm shadow casted on body**
Most of our clients in the V-Tuber industry like it a lot, especially for concert-type 3D live shows that you can find here:
(https://github.com/ColinLeung-NiloCat/UnityURPToonLitShaderExample).

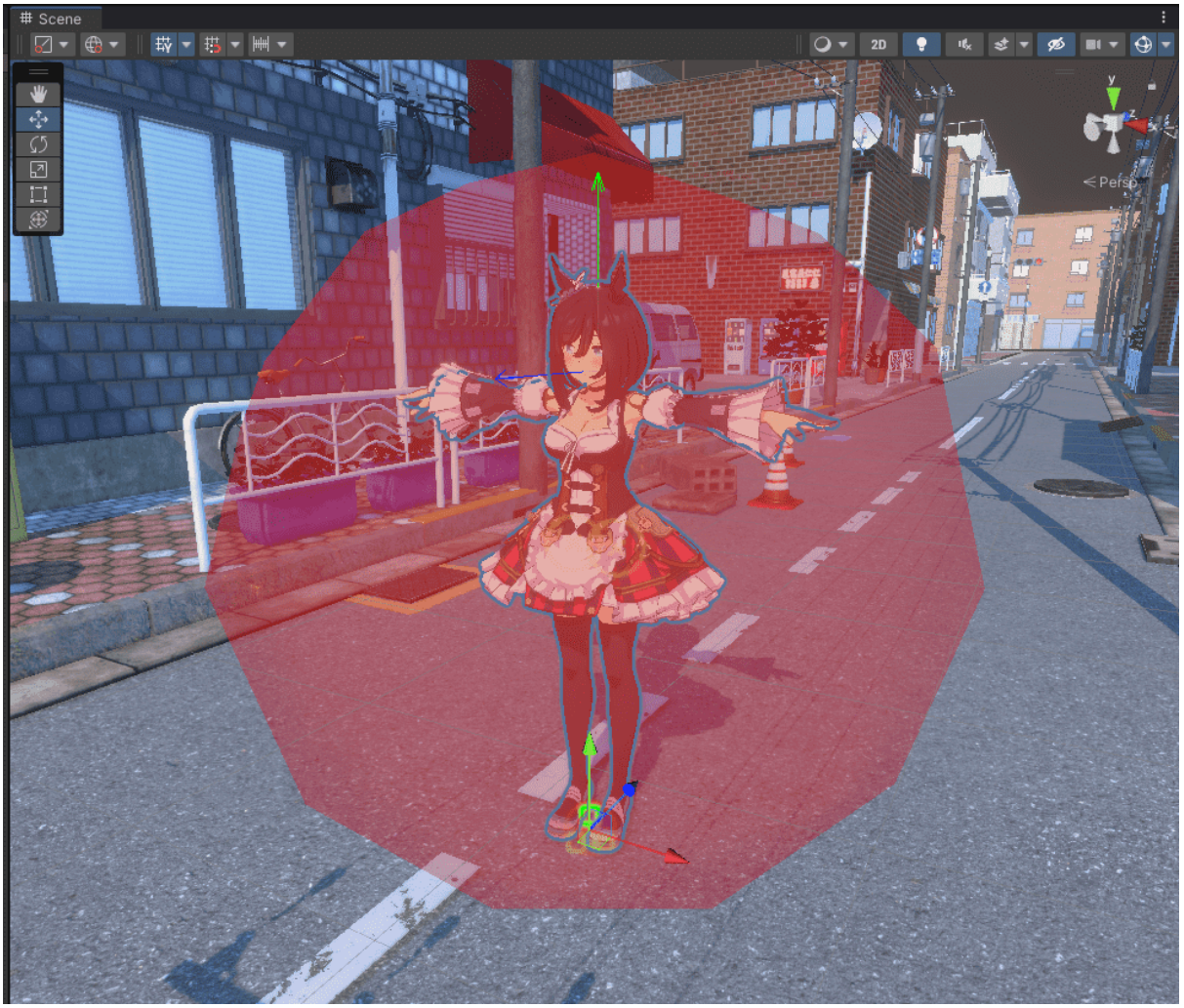
Render steps

1. For each character, use a sphere with a 1.25m radius to encapsulate the whole character, with the character's hip bone as the sphere's center. This sphere should encapsulate the character fully; if not, increase the radius.
2. For example, on a concert stage with 5 idol members, five 1.25m spheres will be on the stage, each encapsulating the corresponding character.
3. Find a bigger sphere S that closely fits and encapsulates all spheres from step(2).
4. Imagine you have an orthographic camera; make the view area closely fit sphere S from step(3), get the MVP matrix M.
5. Use matrix M from step(4) to render all characters's depth into a Shadowmap.
6. Now the shadowmap RT's rendering is complete, it contains all character's depth from shadow light's perspective, with very high shadow map area utilization.
7. In the character lit pass, transform the vertex using the same matrix M, calculate the depth, just as in step(5).
8. After step(7)'s matrix transform, sample the shadow map using vertex position xy as uv, getting the depth from the shadow map.
9. Now you have **self depth** and **shadowmap depth**, compare both depths; if the sampled sample map depth is nearer to the light, that shading pixel is in the shadow area. When comparing depths, a little of bias is needed to avoid Shadow Acne

Chara shadowmap weird

If your character's self shadow map appears and disappears wrongly when the camera moves, make sure the per character script(**NiloToonPerCharacterRenderController**)'s bounding sphere is set up correctly, you will have to make the red gizmo sphere contains the character completely and tightly, please consider any possible animation of the character also.

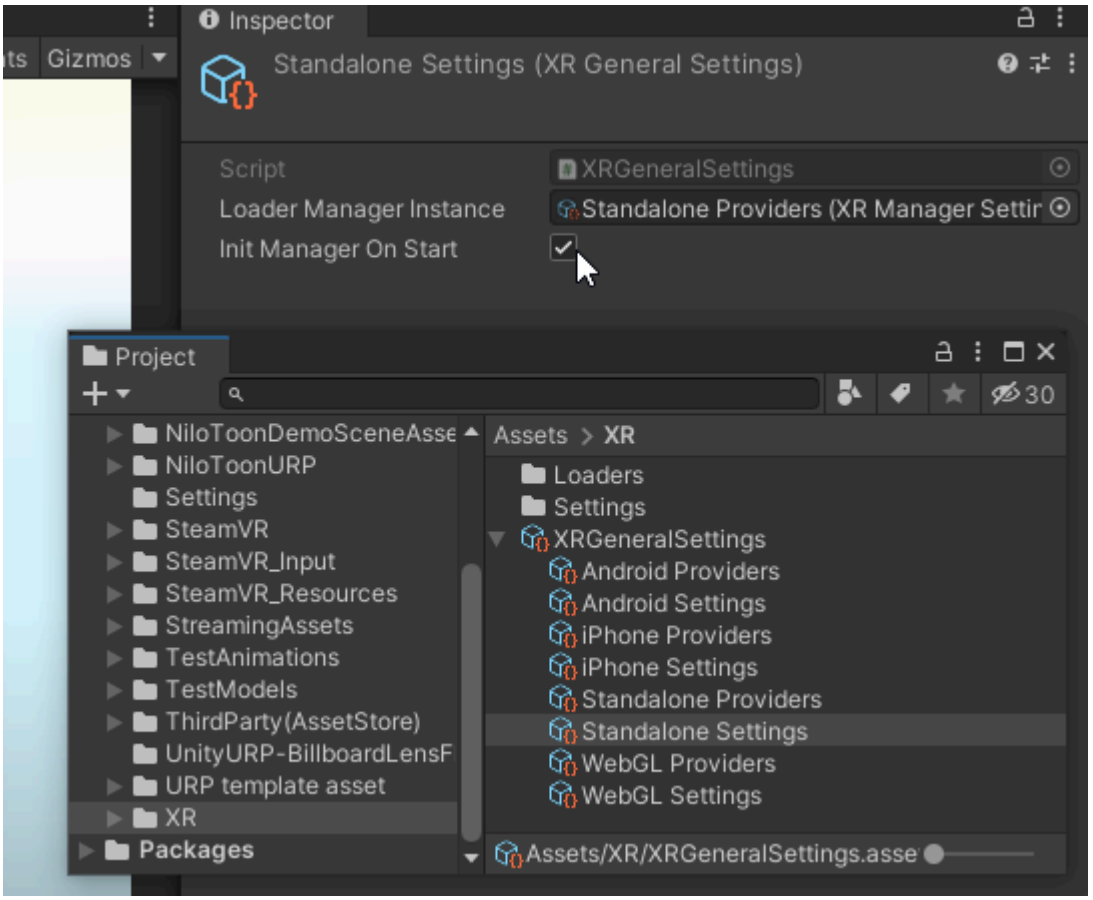




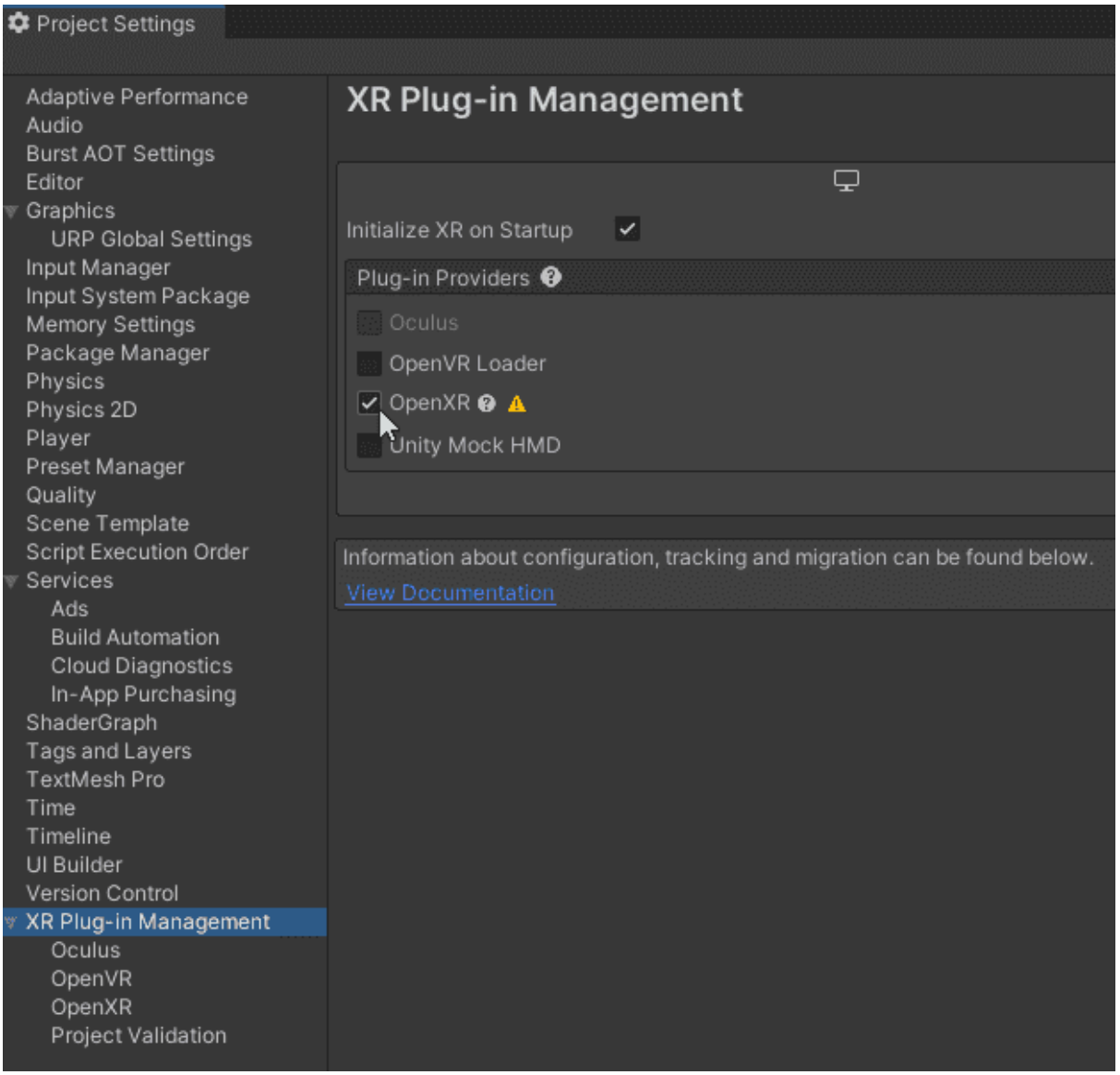
Once the red sphere contains the character completely, NiloToon’s self shadow map will work correctly.

PCVR in Play Mode

- In NiloToon's demo project, find **Assets/XR/XRGeneralSettingsPerBuildTarget** in project, click **Standalone Settings**, enable **Init Manager On Start**



Enable **OpenXR** in Project Settings's **XR Plug-in Management**



- make sure your VR device(e.g. Quest2's AirLink/SteamVR/VirtualDesktop…) is ready and connected to PC
- open any NiloToon's scene
- click play in editor, now in your VR device, you should be able to control the camera movement by your headset
- you can still change scene in play mode, using your **mouse**(there is no control from the VR controllers)

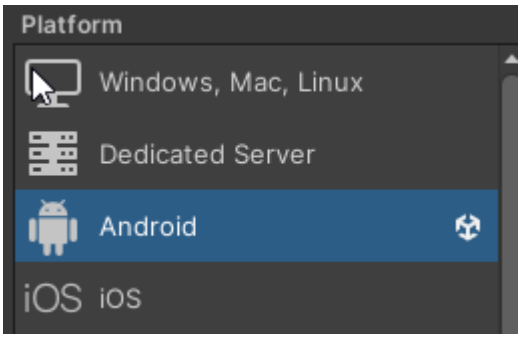
If you want to turn off VR, revert all the above change:
- disable **Init Manager On Start**
or
- disable **OpenXR** in the **XR Plug-in Management**

VR disabled features

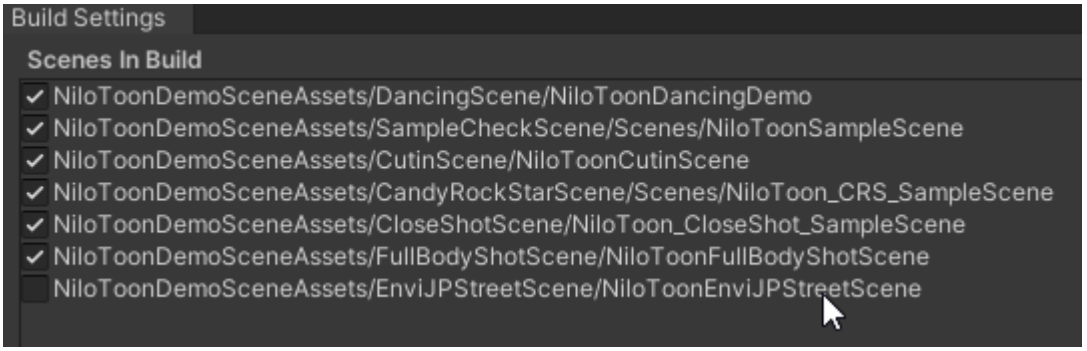
We disabled some features in VR since they may be inappropriate in a VR camera following the user’s head:
- **Anime PostProcess**

Build demo proj(apk/iOS)

1. Switch platform to **Android/iOS**



2. (optional) Don't include **NiloToonEnviJPStreetScene** in Build Settings, this step can reduce build time and runtime shader memory usage, since this scene is not used in Android build

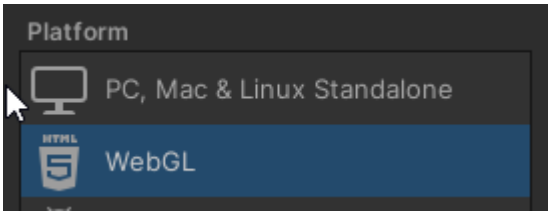


3. Connect your android/iOS phone to your computer via USB, then click "**Build and run**" (or Ctrl+B). Make sure **USB debugging** in enabled in your android device

4a) For android, you should see the .apk running on your android phone after the build finished

4b) For iOS, you should continue your build in **XCode** window

Build demo proj(Web)



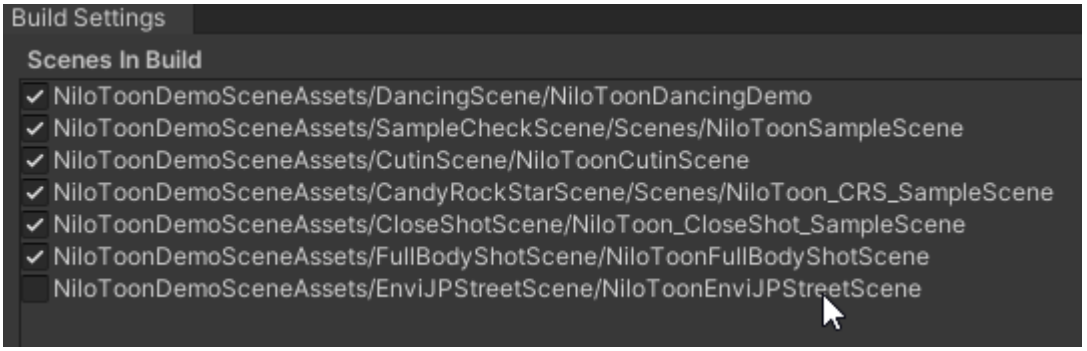
1)Switch platform to **Web**

2)Delete the following folders in project window:
-Assets/**MMD4Mecanim**
-Assets/**SteamVR**
-Assets/**SteamVR_Input**
-Assets/ThirdParty(GitHub)/**uLipSync**

3)Click "**Build and run**" (or **Ctrl+B**)

4)You should see the demo running in your Web browser after the build finished

*If you stuck at loading(0%) in your browser after build success, close the web page.Don't include **NiloToonEnviJPStreetScene** in **Build Settings**, then build again



*__VideoPlayer__ in __Dance!__ scene (__NiloToonDancingDemo__) will not work in Web build, it is expected

GLES2.0 not supported

Since NiloToon and URP shader are using SRP batcher, the recommended OpenGLES version is **OpenGLES 3.1 or up**.

OpenGLES 3.0 is still supported(e.g. very old android mobile / **WebGL 2.0**).

OpenGLES 2.0 is NOT supported by NiloToon & Unity2022LTS anymore.

Planar Reflection issue

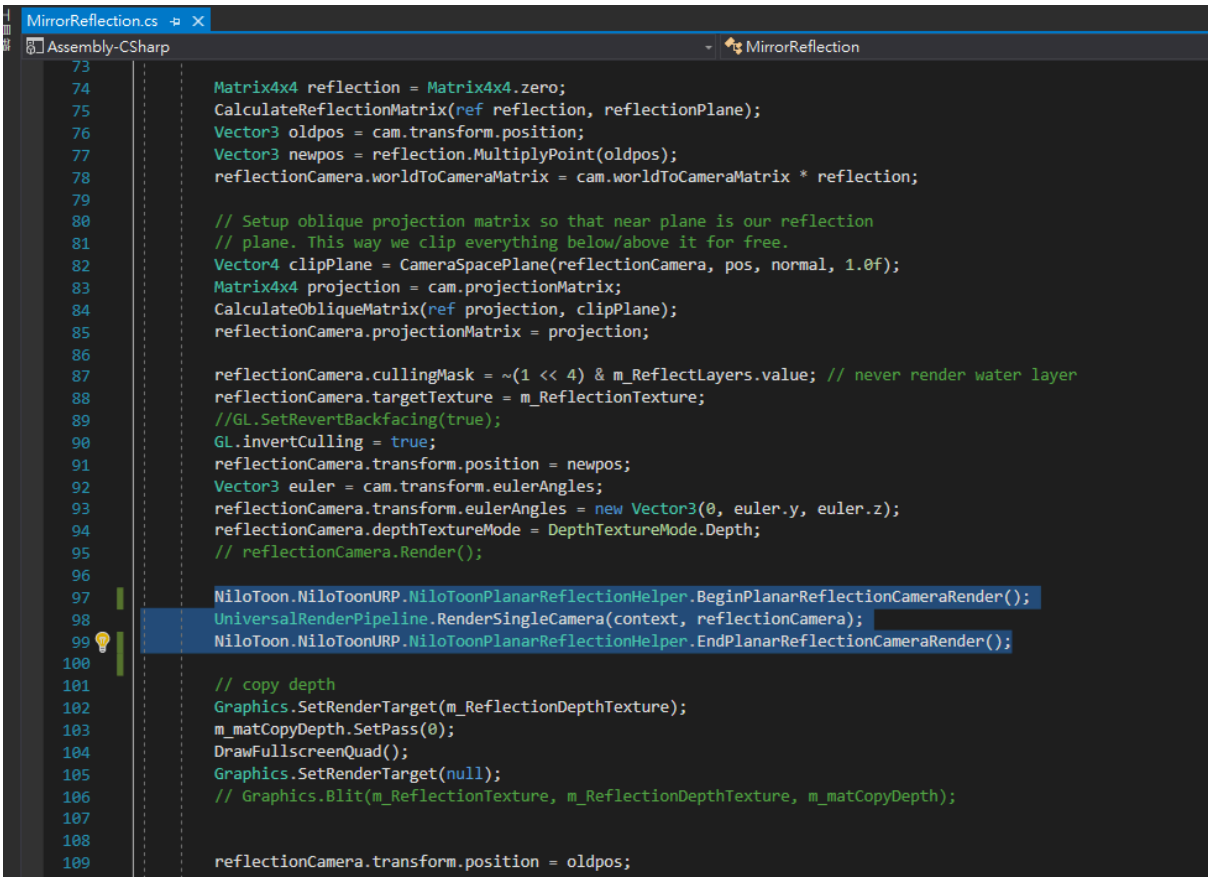
When rendering NiloToon shader in planar reflection camera, some part of the model disappeared (e.g. **PIDI : Planar Reflections 5**)

Usually NiloToon will handle this automatically when camera name contains any of the following keywords,

- **Planar**
- **Reflection**
- **Mirror**

But in case NiloToon failed to solve the problem, you can manually add the following 2 lines to every **UniversalRenderPipeline.RenderSingleCamera** call in your **reflection camera script**, this will solve the problem:

```
NiloToon.NiloToonURP.NiloToonPlanarReflectionHelper.BeginPlanarReflectionCameraRender(); // then add this line right before RenderSingleCamera()
UniversalRenderPipeline.RenderSingleCamera(context, reflectionCamera); // first, find this line in your planar reflection script
NiloToon.NiloToonURP.NiloToonPlanarReflectionHelper.EndPlanarReflectionCameraRender(); // then add this line right after RenderSingleCamera()
```



The picture above is an example of adding the above 2 lines to solve the problem.

After the fix, your NiloToonURP shader character will render correctly in planar reflection.

*If your planar reflection is implemented using **renderer feature**, you will need to use the **CommandBuffer** overload methods instead

```
// add this line before DrawRenderers in a RendererPass
NiloToonPlanarReflectionHelper.BeginPlanarReflectionCameraRender(ref cmd);

// remember to execute cmd first before calling DrawRenderers
context.ExecuteCommandBuffer(cmd);

context.DrawRenderers(cullResults, ref drawSetting, ref filterSetting);

cmd.clear();

// add this line after DrawRenderers in a RendererPass
NiloToonPlanarReflectionHelper.EndPlanarReflectionCameraRender(ref cmd);
context.ExecuteCommandBuffer(cmd);
```

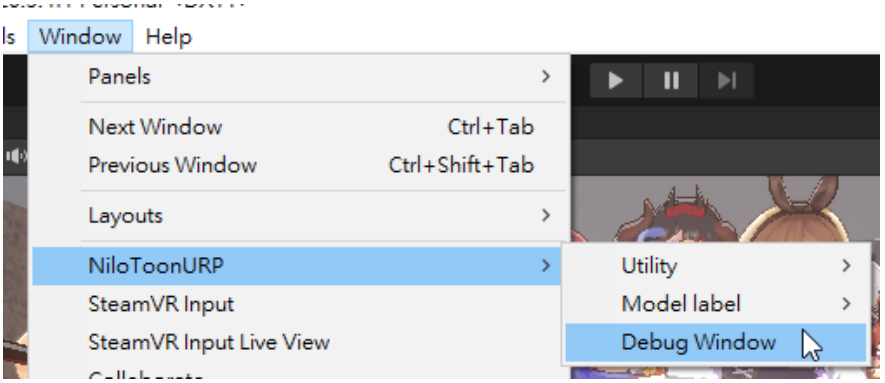
When using **PIDI : Planar Reflections 5**, you can also try to set

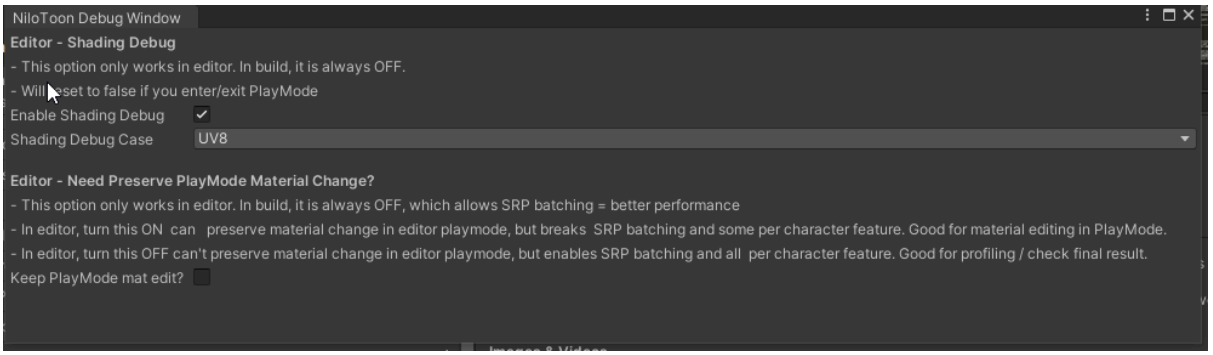
- **PostFX Support =** Disabled

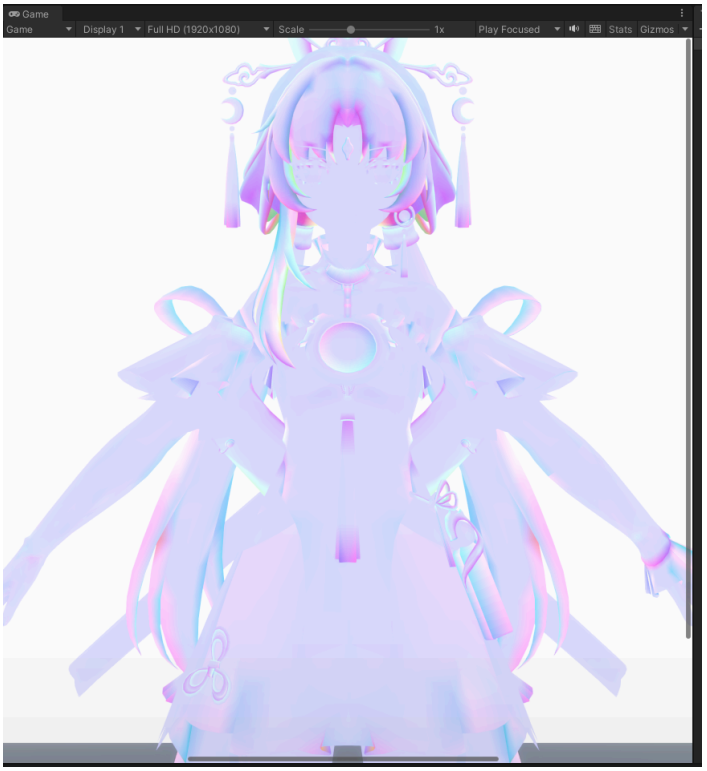
if character still disappears in some camera angle

Debug UV8(smoothed normal)

If you open NiloToon's **debug window** and debug **UV8** as color, and all the character's rendering becomes a light blue and purple (becomes colors like **normal map**, and NOT red yellow and green color like **UV**), then your .fbx's smoothed normal for outline is already generated and stored inside mesh's **UV8** correctly. **HQ continuous outline** in character material's **Classic Outline** group will work when the debug color is correct.

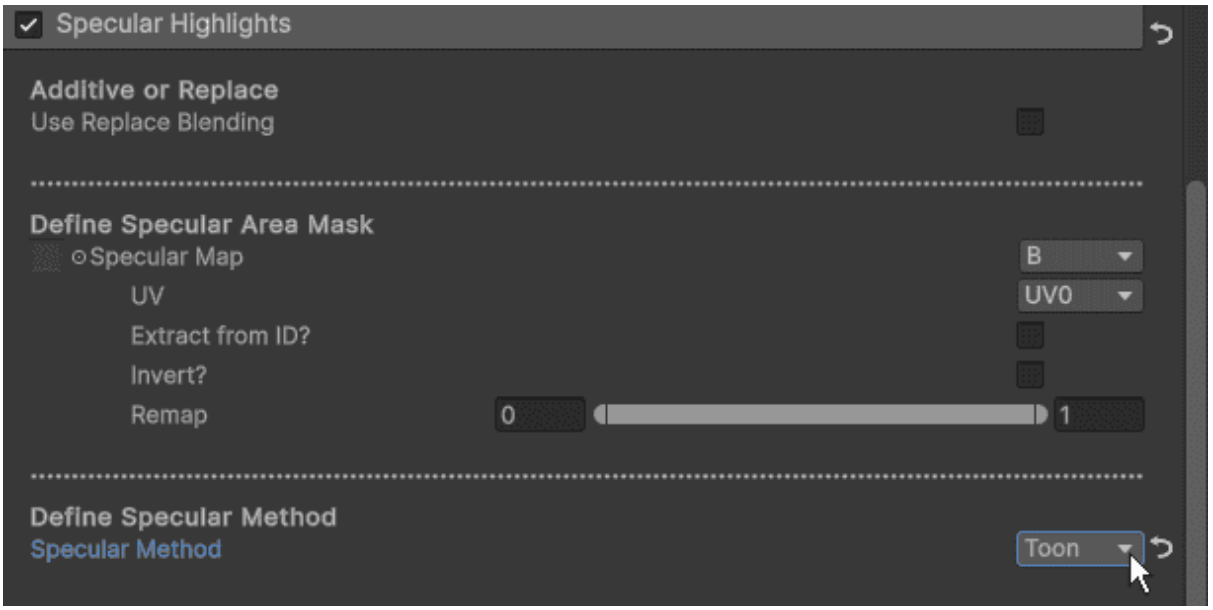




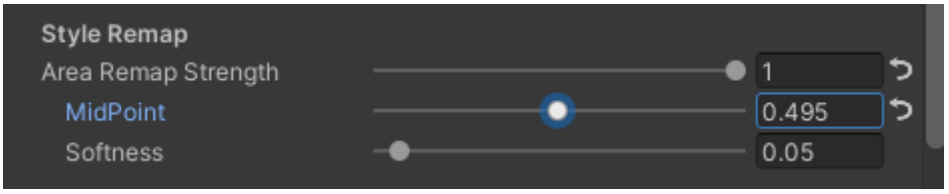


Hair/toon style specular

For hair material, usually using **Specular Method = Toon** will provide better toon specular results for hair, since it is easier to control the specular area/shape artistically.



Then you will want to set the following 3 params until specular looks good for hair/toon style. Below is an example for hair specular



For example:

- Assign the specular mask first, it will limit where the specular can appear
- Set Area Remap Strength to 1
- Adjust **MidPoint** until the specular looks good (e.g., 0.5)
- Set **Softness** to 0.05

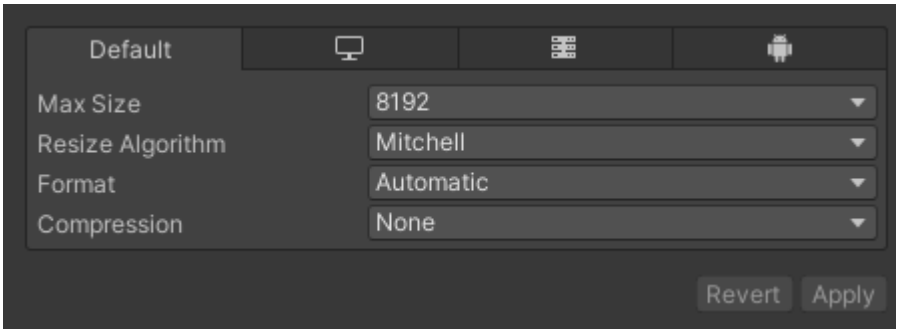
*For **realistic** specular like the **PBR** shader, you should use **Specular Method = GGX PBR** and adjust **smoothness** instead of using **Style Remap**, see [PBR Specular Highlights]

Shadow Color artifacts (hue)

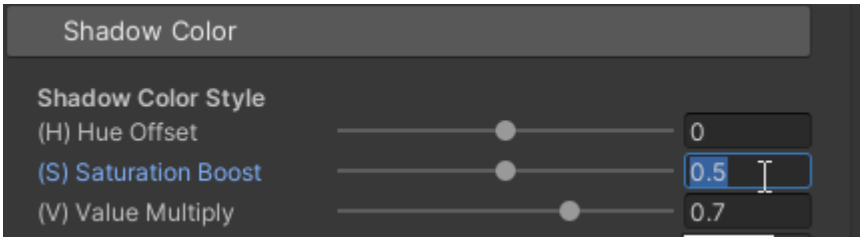
This problem exists when **Base Map**'s saturation is low(white/gray color), where hue is not well defined in the texture(random hue with low saturation), and when NiloToon shader tries to apply **saturation boost** for shadow area, it will make the random hue more visible. This problem will be more problematic after Base Map's texture compression.

To solve it,

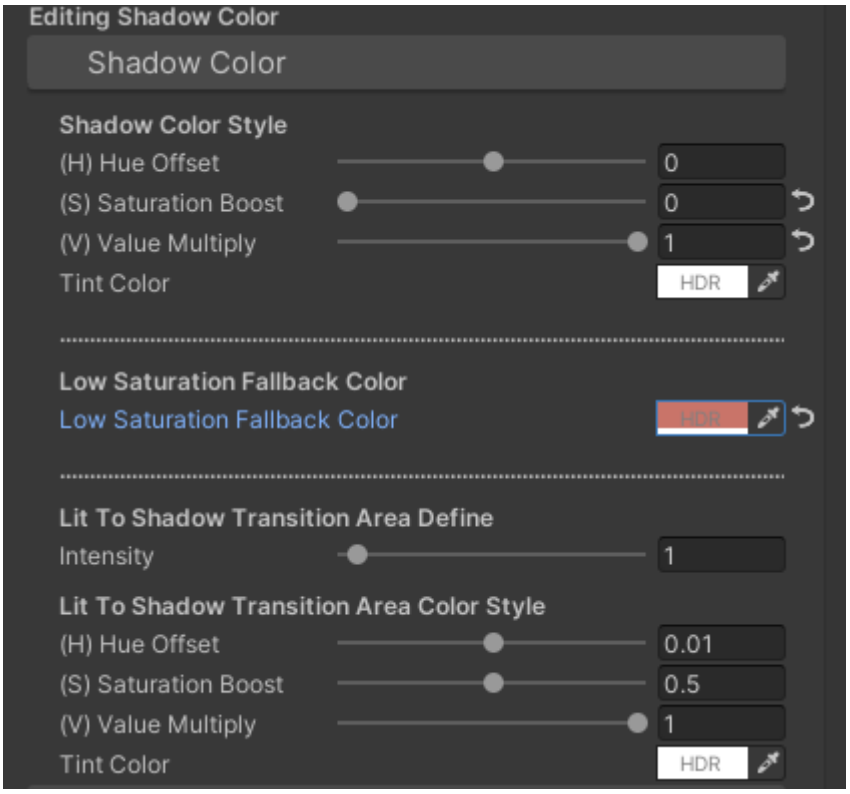
1) for the color texture (**Base Map**), select the texture and try using **high quality** compression , or **None** compression if you don't care about performance too much.

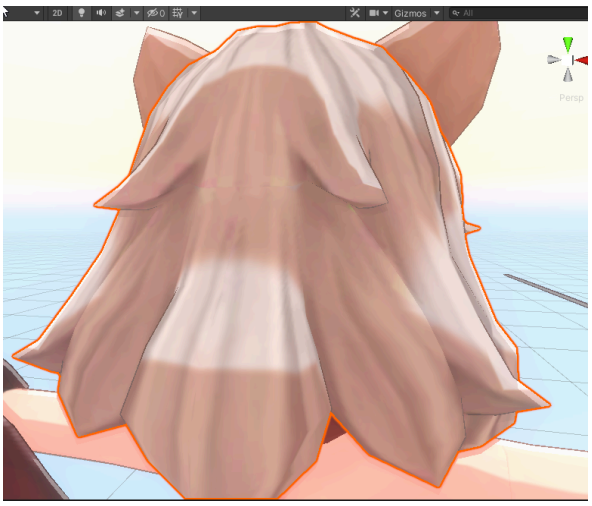


2) in NiloToon material's **Shadow Color** section, use a **lower** **Saturation Boost**, the default **0.2** can be too high

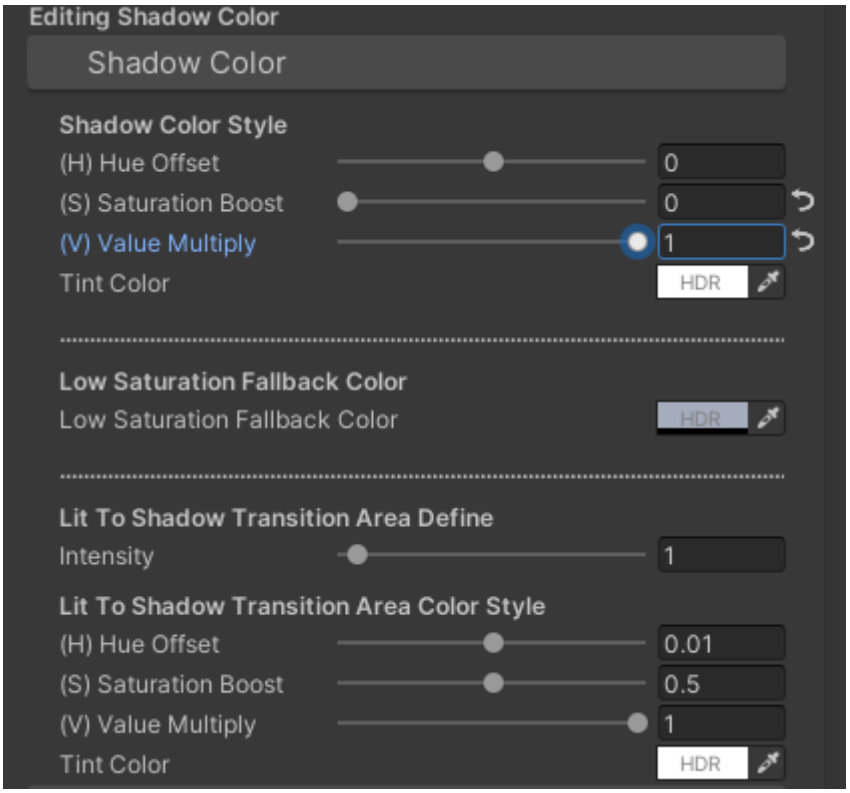


3)(not recommended) in NiloToon material's **Shadow Color** section, set a better **Low Saturation Fallback Color** in material may avoid artifacts




2D
Gizmos
All
Persp

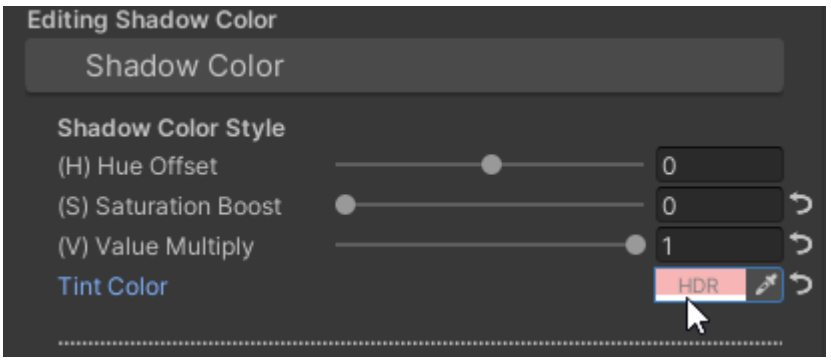
4a) if the problem still can’t be solved,



In material’s **Shadow Color** Style section,

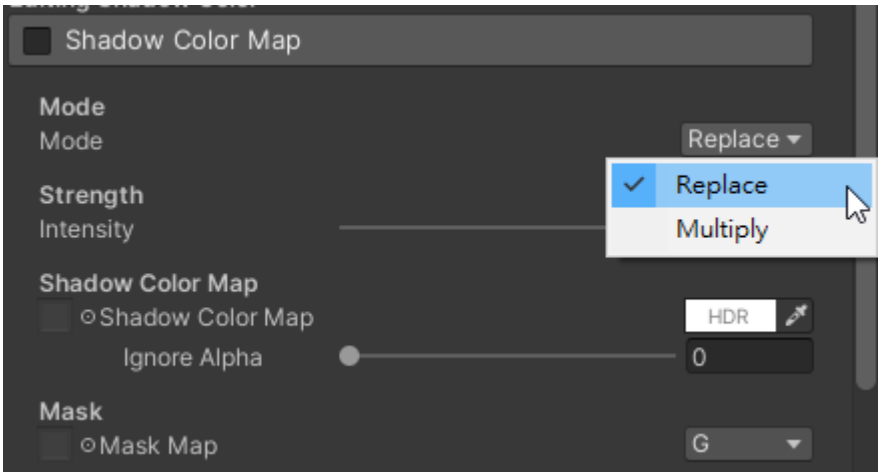
- set **HueOffset** to 0
- set **SaturationBoost** to 0
- set **Value Multiply** to 1
- set **Low Saturation Fallback Color**’s **alpha** to **0**

4b)Doing the above **step (4a)** will cancel all auto shadow color logic, and make the resulting shadow color become the same as lit color(can’t see any shadow). Now the material will not produce random hue shadow color artifacts anymore, you can use **Tint Color** to control the final shadow color manually.



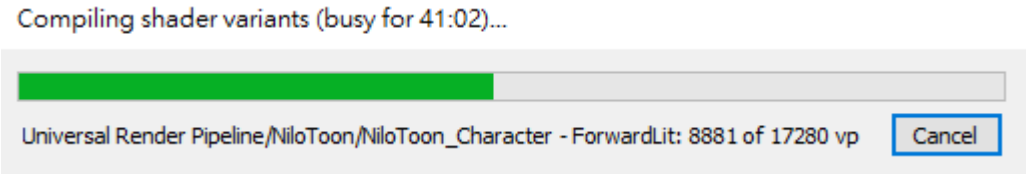
We usually do this for hair and single color cloth materials, since these materials will work better when using just a single shadow tint color.

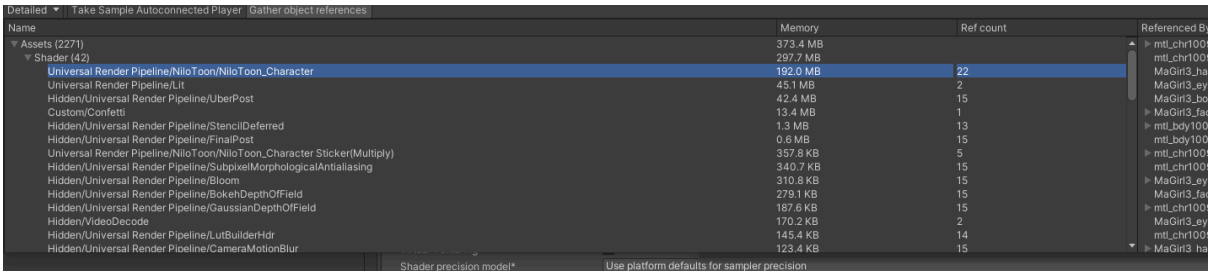
5) If at the end the resulting shadow color is still not good, you can replace the shadow color with your own texture. Although in most cases you don't need to use **Shadow Color Map**, since producing this Shadow Color override texture takes time and skill, it is the last resort.

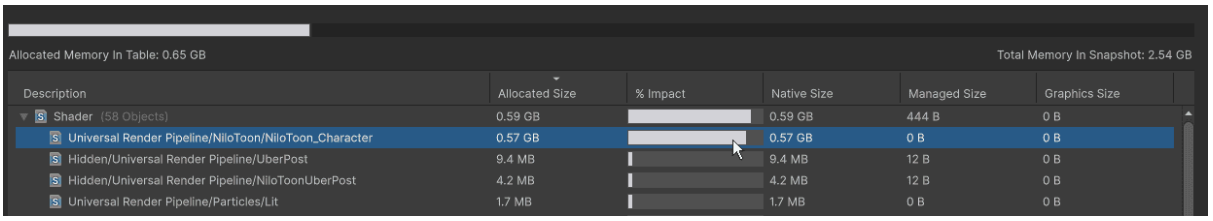


*Some models have this **Shadow Color Map** provided already, usually provided by material using **lilToon/vrm MToon**. If that is the case, you can use the **Shadow Color Map** section to produce the shadow color that the modeler has provided. The auto conversion from **lilToon/vrm MToon** to NiloToon should preserve the shadow color map for you already.

Reduce Shader memory & build time







Similar to URP's Lit shader, ComplexLit shader, UTS3 or RealToon,
NiloToonCharacter.shader uses many
#pragma multi_compile _ _FEATURE
and
#pragma shader_feature_local _FEATURE
in shader to support different use cases.

Shader **memory** usage and **build time** will be **~2^n** for **n** required on+off features, which means the number of shader variants goes up REALLY fast(**exponentially**) when a large amount of features are enabled.

Lowering shader memory for mobile builds is important, else an out of memory crash is possible.
The following pages will introduce solutions to keep shader memory usage and build time low.

0: Use Unity6.1 or newer

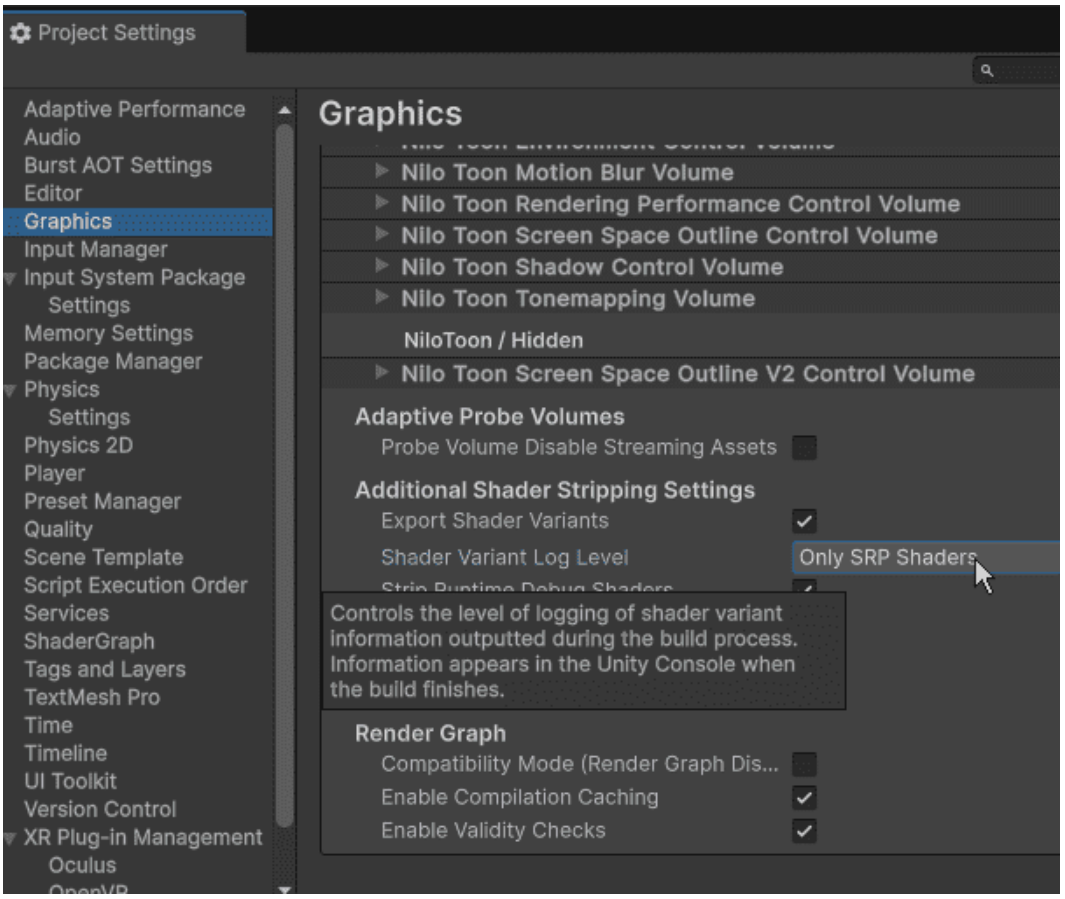
In **Unity6.1**, NiloToon's multi_compile of:

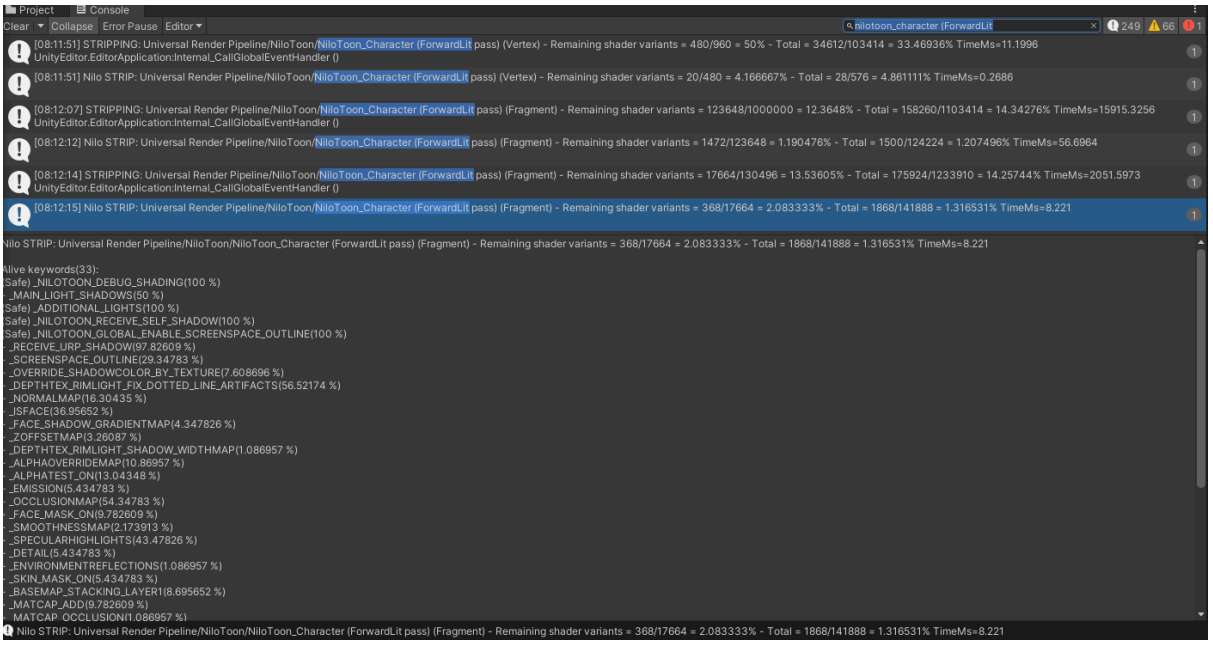
- **fog**

is optimized into a non shader variant method, which will lower the memory usage and build time a lot, exchanged by a little bit more fragment calculation.

*To lower the overall memory usage, enabling **RenderGraph**(disable compatibility mode) can also help.

1: Build stripping info





After you enable **URP asset**'s **Shader Variant Log**, NiloToon's stripping system will log more information when building the player/assetbundle. It will show important information or hints about what keywords can be optimized. For example, you can paste **NiloToon_Character (ForwardLit pass)** in the console window's search to filter the console log after build.

2: Best practice of URP

- Manual: Reduce shader variants in URP
- Manual: Strip feature shader variants in URP

For all **URP assets** included in **Quality settings**, if possible you should try to:

- **Disable** a feature in **all URP Assets** in your build, so URP keeps only variants where the feature is disabled.
- **Enable** a feature in **all URP Assets** in your build, so URP keeps only variants where the feature is enabled.

If you have a feature turn ON in a URP asset for a quality, and turn OFF in another URP asset for another quality, and both URP assets are included and enabled in Quality setting for a platform, URP will not strip that feature and keep both the ON+OFF variants in build, so build time and shader memory usage will be ~2x for each feature like this!

For example, in your quality setting window, you have 2 quality settings enabled for Android:

- Low
- High

Situation A)

- In **Low** setting, main light's **cast shadow** = off
- In **High** setting, main light's **cast shadow** = on

This will **2x** shader variant, since it is possible to have **cast shadow** = on or **cast shadow** = off in build, and URP must include both shader variant in build

Situation B)

- In **Low** setting, main light's **cast shadow** = off
- In **High** setting, main light's **cast shadow** = off

This will **1x** shader variant, since **cast shadow** = on will be stripped when build, only **cast shadow** = off will be included in build

Situation C)

- In **Low** setting, main light's **cast shadow** = on
- In **High** setting, main light's **cast shadow** = on

This will **1x** shader variant, since **cast shadow** = off will be stripped when build, only **cast shadow** = on will be included in build

Also, **disabling unneeded quality** for a platform can help, since it will likely reduce the possible feature combination required.

For example, in the picture below, **unneeded quality** (ExtremeHigh & Highest) are **disabled** for android

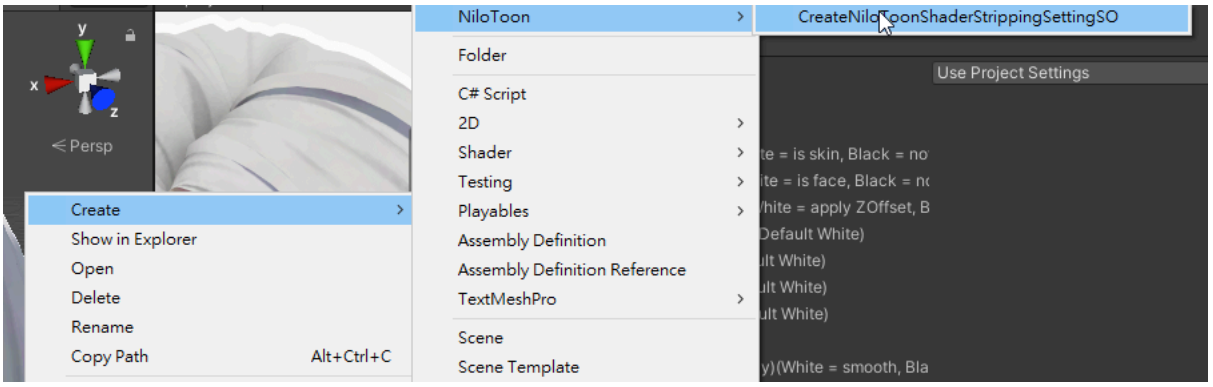
Also, avoid including URP Assets in your build that use different **rendering paths** because this causes Unity to create two sets of variants for each keyword.

After finishing the best practice of URP shader stripping, then you can continue to use the Step(3) below to strip more keywords.

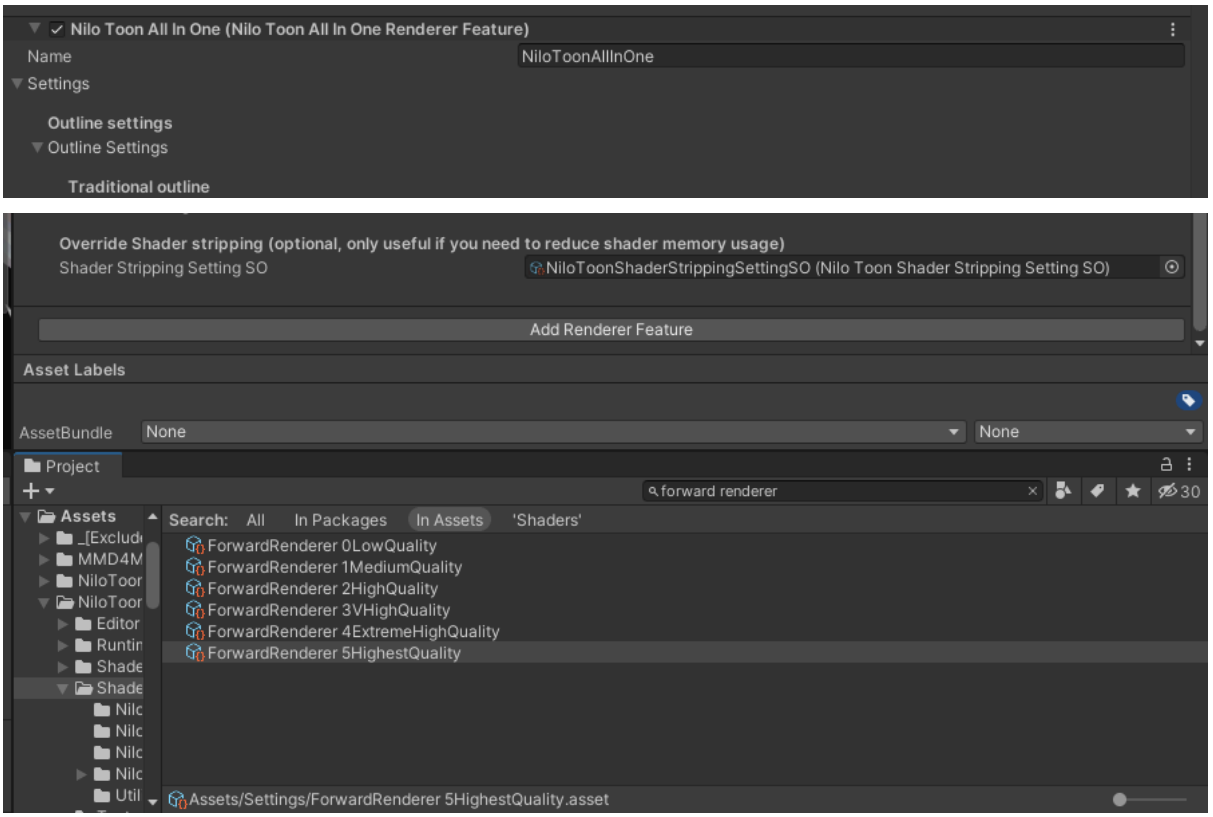
3: NiloToonShaderStrippingSettingSO

To lower **multi_compile** memory usage of **NiloToon_Character** shader, you can use **NiloToonShaderStrippingSettingSO**

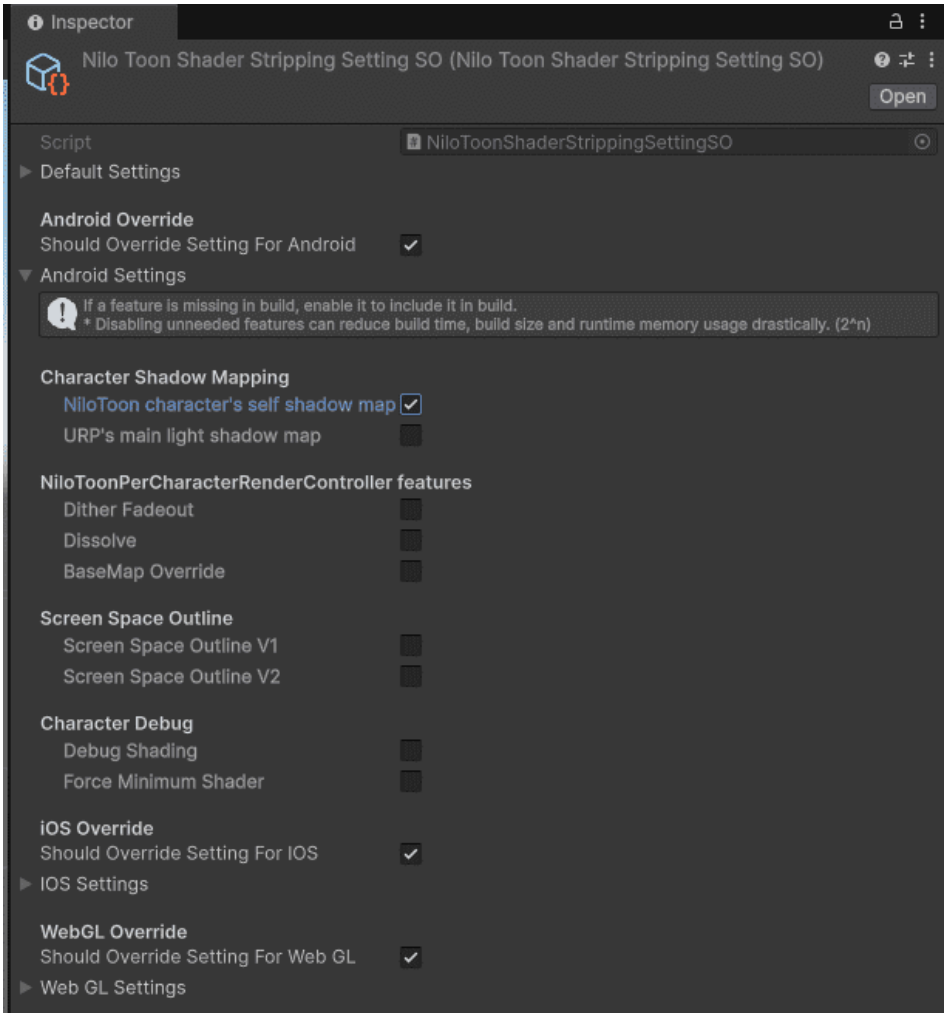
a) right click in the project window, create a **NiloToonShaderStrippingSettingSO**



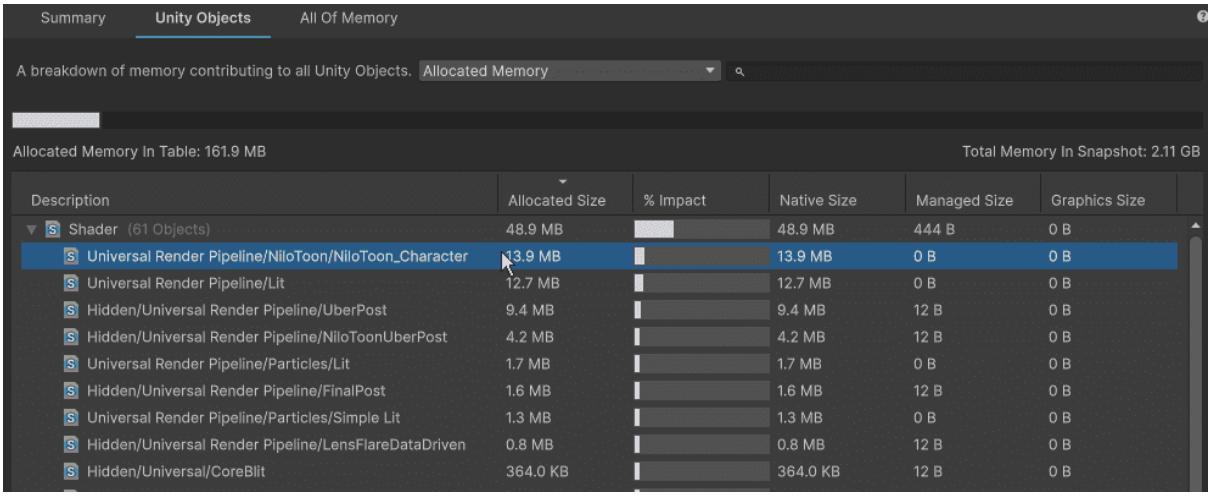
b) drag that single **NiloToonShaderStrippingSettingSO** to **every** NiloToonAllInOne renderer features included in build, not just a random one.



c) Inside that **NiloToonShaderStrippingSettingSO**, You can **turn off** features that you don't need in build **per platform**. For each platform, turning each keyword off will reduce **~40 to 50%** runtime **shader memory** usage and **build time**.
It works for **player build**, **assetbundle** and **addressable**.



*To verify if **NiloToonShaderStrippingSettingSO** is working or not, we recommend **disable every feature** inside it first for your target platform, and build, see if **NiloToon_Character** shader **memory** usage and build time are reduced a lot using the **Memory Profiler**.



If it is working correctly, in Memory Profiler, you should see the **NiloToon_Character** memory lower to around **10~50MB** (in the above screenshot, **13.9MB** for **NiloToonDemoProject**'s **Unity6LTS Android** build, using NiloToon0.16.35). Now

you can **add back** the keywords inside **NiloToonShaderStrippingSettingSO** that you actually need in your game and build again, then check again the memory.

4: Optimize material

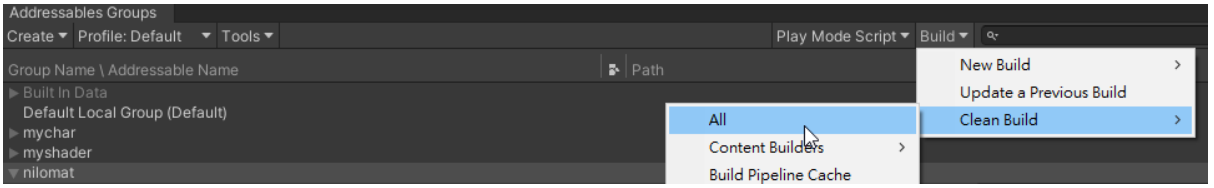
There are LOTS of **shader_feature** in nilotoon's shader, but if you don't enable them in any of your materials in build, they will never increase shader memory / performance cost / build time.

The more unique shader keyword combinations in all your materials(included in build/asset bundle/addressable), the more memory usage is needed in runtime, also build time will be longer.

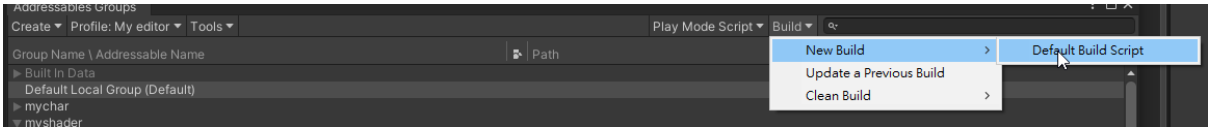
The only way to reduce shader_feature memory usage is to reduce the number of unique combinations of shader keywords in your materials. But before you do that, make sure you have done the above **step(1-3)** first, which usually solves the problem already in most of the situations.

*If you are using **Addressables,** after setting up **NiloToonShaderStrippingSettingSO**, if you can't see any change in the next **Addressables** build, remember to:

- 1.Clean all previously addressables built data



- 2.then build again.



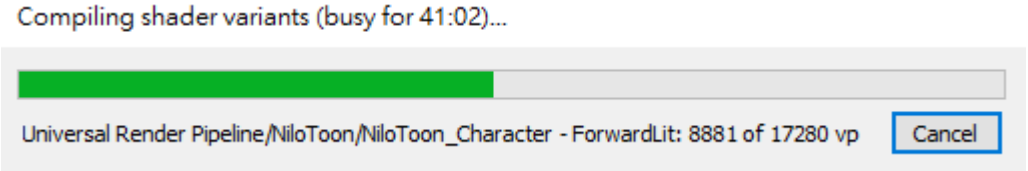
Addressables will not consider **NiloToonShaderStrippingSettingSO**'s edit as a content asset change, so clicking build will do nothing due to the last build's cache being still 'valid'.

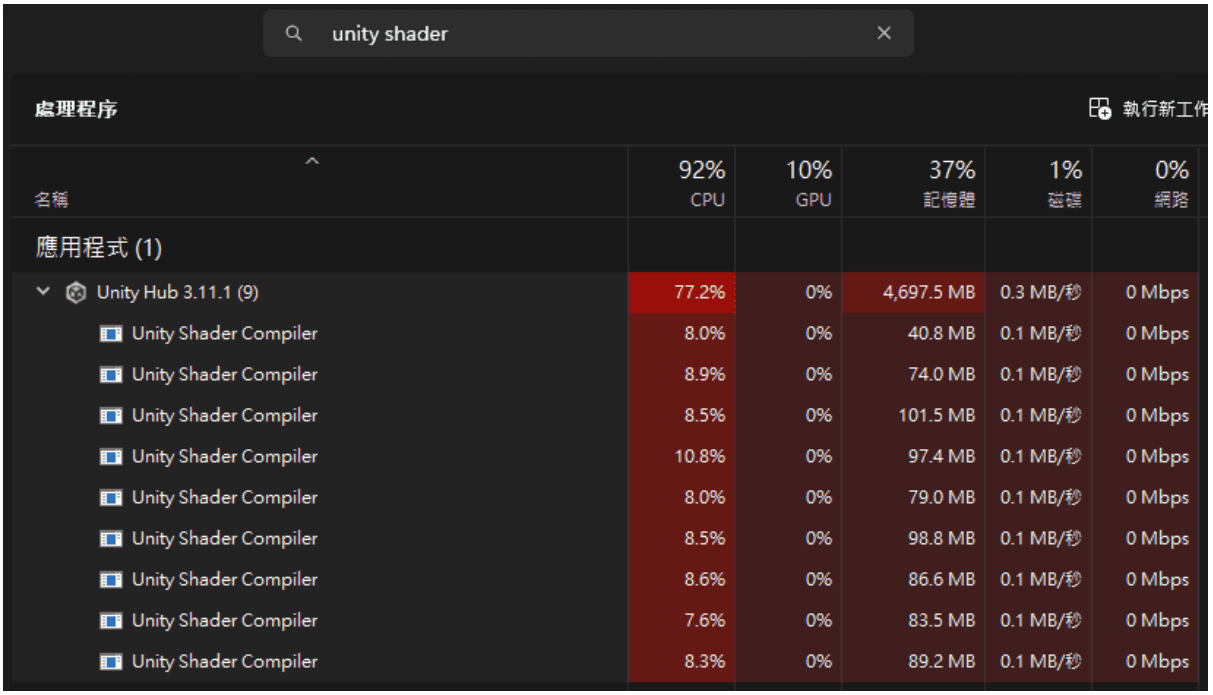
5: Additional resources

Official - Shader Stripping Improvements in URP - Unity Forum
Improve shader compilation time using the URP Config Package - Unity Forum
Improvements to shader build times and memory usage in 2021 LTS

6: Lower build CPU%

If you build an application or assetbundle, Unity will spawn one **Unity Shader Compiler** for each **CPU thread** you have, to maximize the speed of the shader compile.



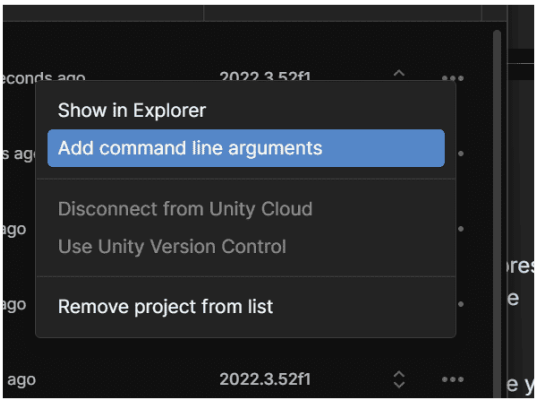


By default, there is no limitation on the number of **Unity Shader Compiler**, so Unity will spawn as many **Unity Shader Compiler** as possible, your PC will become very slow and unable to do other task smoothly while in the build process(e.g., work on another project, using the browser, play games) or even freezing the PC if all the **Unity Shader Compiler** are working too hard (100% full CPU usage)

If you are willing to slow down the shader compile speed in exchange for a much better responsiveness for your PC to work on other tasks while in the build process, you can use this **editor command line argument** to limit a project's max thread count:

```
-job-worker-count PutYourTargetThreadLimitNumberHere
```

To use the above argument, open **UnityHub**,
select the target project and click **Add command line arguments**,

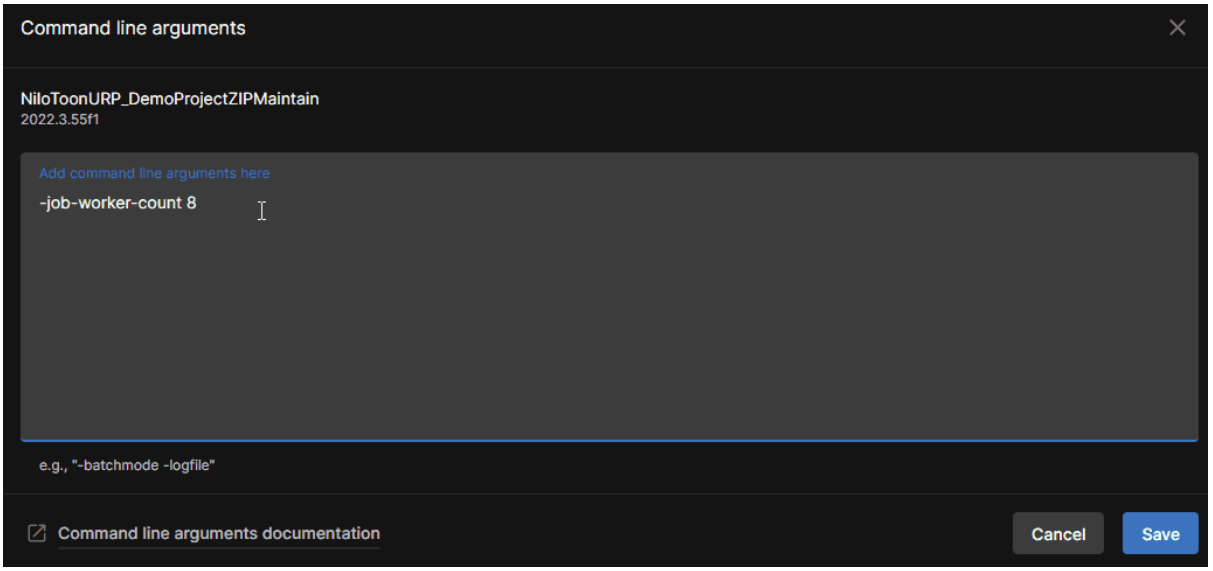


For example, a PC with CPU **Intel i7-14700F** will allow Unity to spawn a maximum of **28 Unity Shader Compilers**; it may make the PC too unresponsive for other tasks.

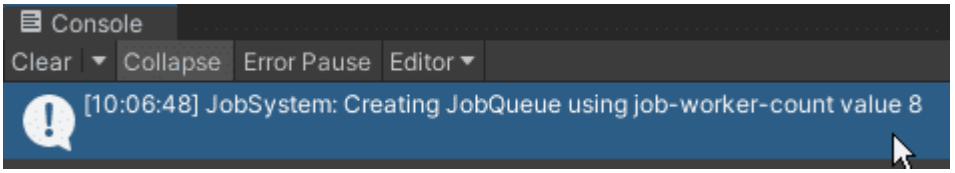
To use a lower thread limit, In the window below, enter:

```
-job-worker-count 8
```

hit **Save**,



restart the editor, you should see the following log



then you can build again. This argument will limit the number of **Unity Shader Compilers** to **8+1**, which improves the responsiveness a lot when building, but it will also slow down the shader compile speed a lot, which means it will take much longer to complete the shader compile.

You should pick a number (in this example it is 8) that suits your **CPU thread count** and your **target responsiveness**. If you are unsure, you can start from 8, and +4 each time until the build starts to slow down your PC too much when you are doing other tasks (e.g., playing a game).

If a PC is used only as a build machine, you should not use this argument to limit the shader compile speed, since you want the fastest shader compiler speed without limiting the thread count.

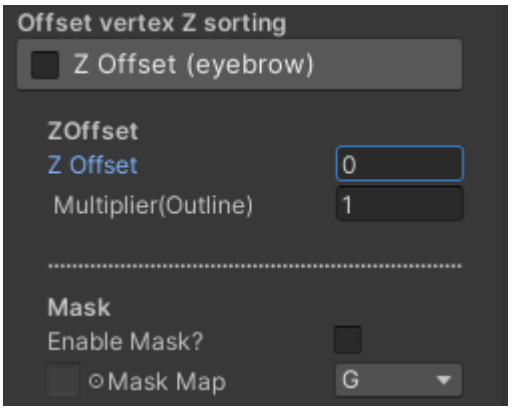
Eyebrow on hair(opaque)

If you only need to simply draw **eyebrows/eyelash** **on top of hair**(without semi-transparent blending with hair), the simplest and quickest method is to use the **Z Offset (eyebrow)** group in your **eyebrow/eyelash material**, no extra **material** / **stencil** / **NiloToonRendererRedrawer** is needed, doing this step alone to your eyebrow/eyelash material will work already.

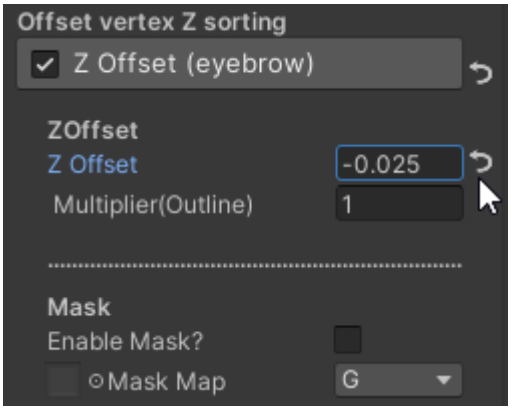
Example result: 【粛聖!! ロリ神レクイエム☆】パトラちゃんに罵倒してもらうだけ #vtuber #周防パトラ #shorts - YouTube

Z Offset method

Before edit eyebrow material:



After edit eyebrow material (just by enabling Z Offset):



-0.02 / **-0.03** is usually a good start, it means pushing the eyebrow 2cm / 3cm towards the camera. Don't set it too large (e.g., -0.1), eyebrow may go infront of unwanted objects, like the hand.

Z Offset mask

If the eyebrow/eyelash is not an isolated material (for example, there is only a single face material, which contains the whole face with eyebrow), use **Z Offset (eyebrow)** group's **Mask Map** to make the ZOffset only apply to the eyebrow/eyelash area.

Stencil method

If you don't want to use ZOffset, and the eyebrow material is an isolated material, it is possible to do it in the traditional way using stencil without using ZOffset.

For example:

1. eyebrow material writes to stencil first

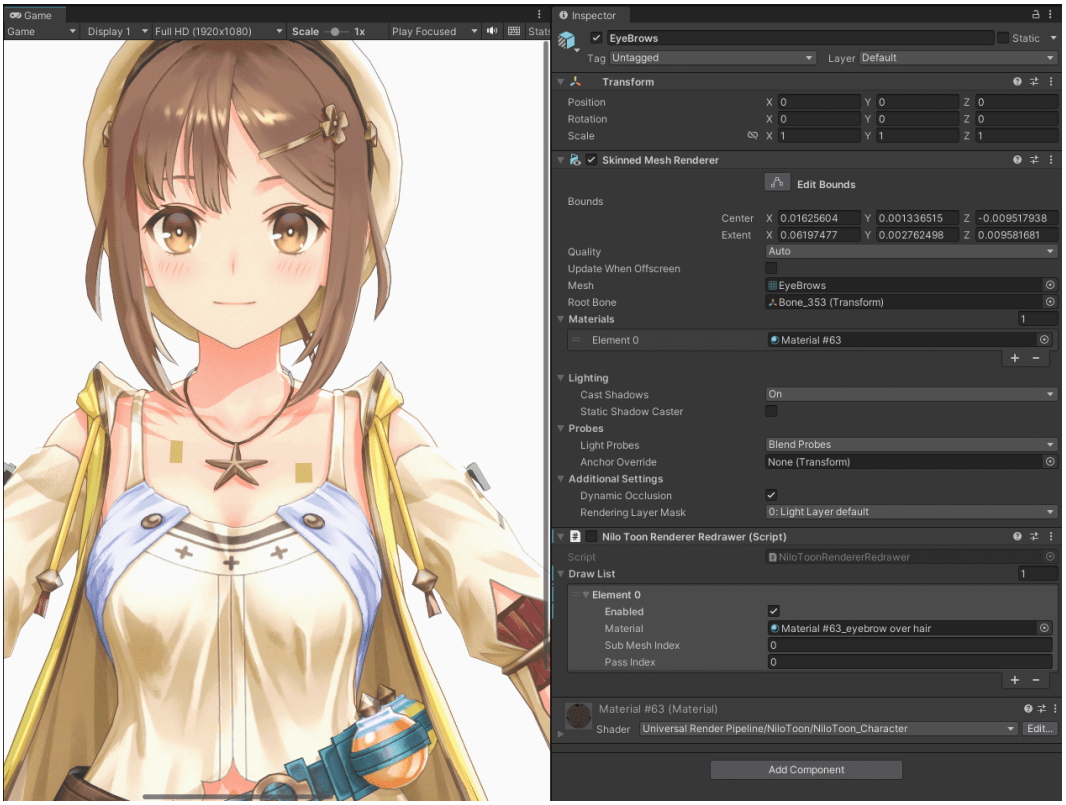
2. hair will use the stencil to reject the render only for eyebrow pixels
3. you will get a result similar to the above ZOffset method, but you may see a wrong occlusion problem. For example, you may see eyebrow from the back side of the hair if your hair material doesn't split front hair into an isolated material

Summary

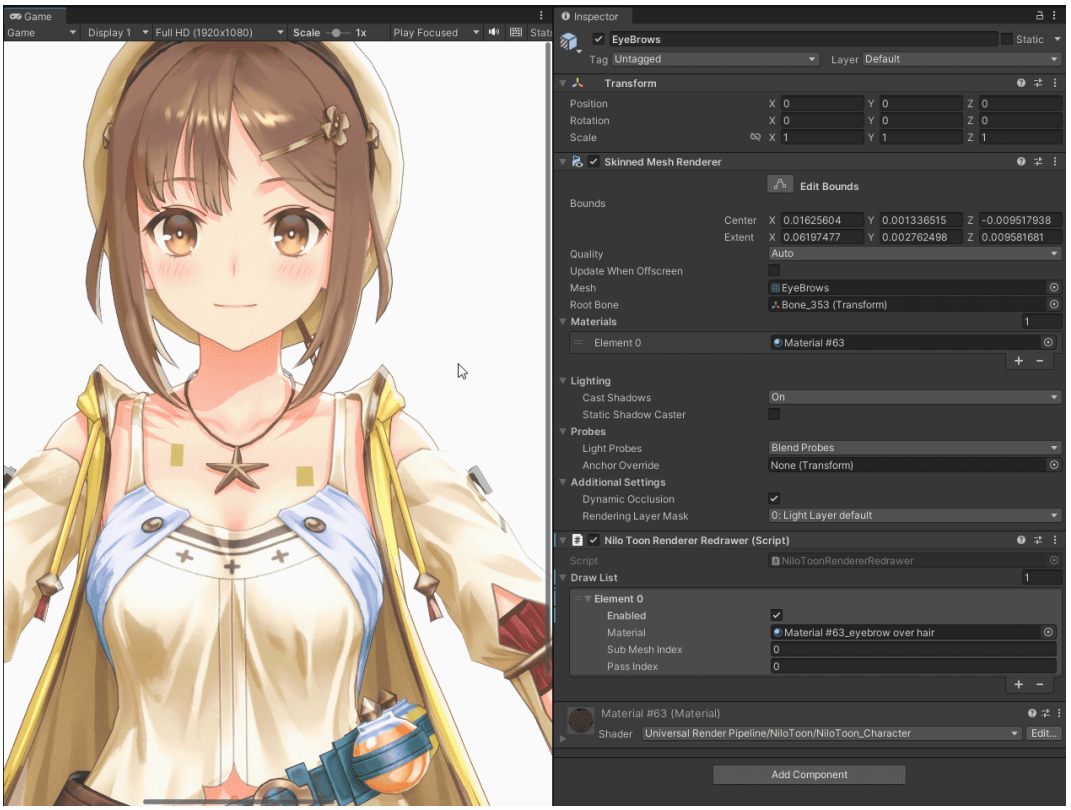
While this ZOffset / Stencil method is simple and fast, sometimes you want to blend eyebrows with the hair in a **semi-transparent** way for a better art style, to do this, see the next section **Eyebrow on hair(transparent)**.

Eyebrow on hair(transparent)

Before:

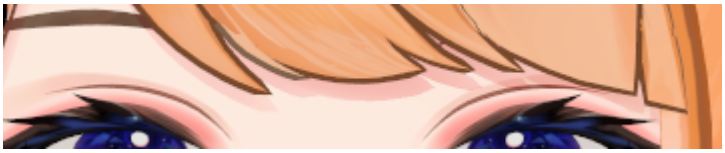


After(look at the difference of **eyebrow** area):

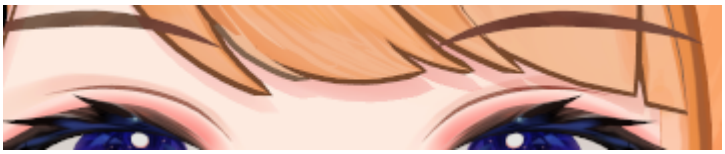


Let’s see an example with simple introduction:

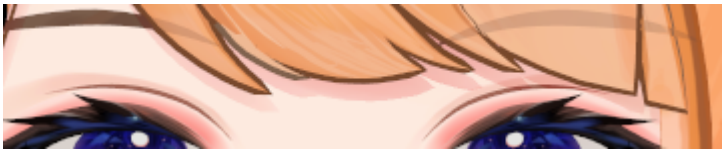
1. original result (without doing anything special)



2. let the eye brow render on top of the hair, if you want this style, see this



3. Semi-transparent blending of eyebrow with hair, this section is about this style



In the section below, we will introduce **Method A / B / C,** you can pick **Method A / B / C** in to achieve a similar result.
We usually pick **C (Stencil+Render Queue hair Redraw)** if possible since it will produce a perfect result, but the set up is much more complex.

A (semi-transparent hair):

Steps Overview:
Simply make the front hair become semi-transparent, to let the viewer see the color under the hair, which is the easiest method.
No Stencil / ZOffset / NiloToonRendererRedrawer needed

This is a classic way to let the viewer see the eyebrow under the hair, it is a method that exists in most of the toon shaders.

If the front hair material's Base Map has alpha for front hair already, then you don't need to do anything, you can simply use the following setting for front hair:

- **Surface Type = Opaque(Alpha)/Outline**

If the front hair material's basemap doesn't have alpha for front hair, you can use/create an extra map as the basemap's alpha by doing the following:

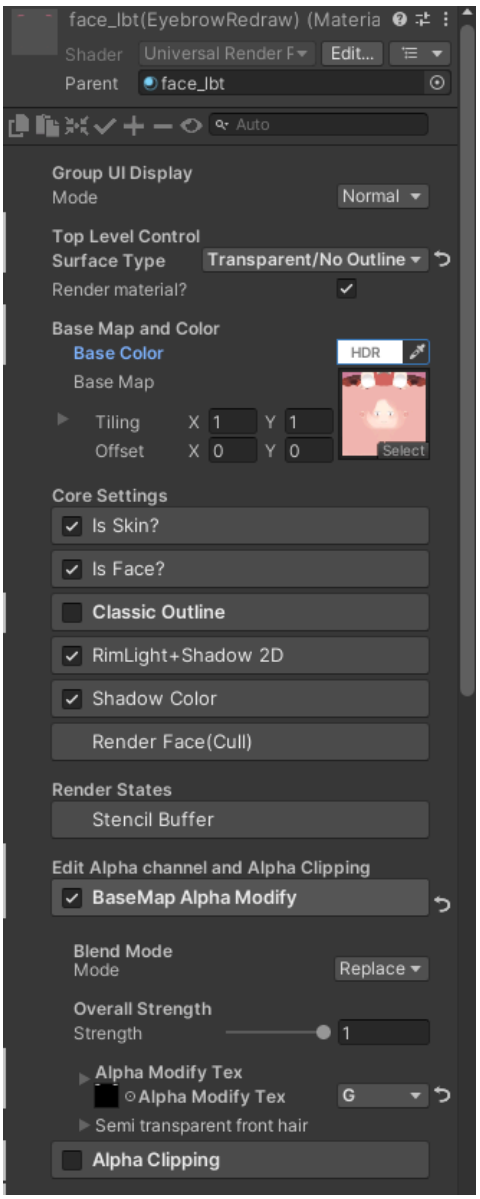
- **Surface Type = Opaque(Alpha)/Outline**
- Enable **BaseMap Alpha Modify** group, assign a texture to **Alpha Modify Tex**, set mode = **Multiply**, select the rgba channel that contains the alpha. This texture will modify BaseMap's alpha
- (optional) If you enabled **BaseMap Alpha Modify** group, set **Is front hair?** = 1, this will make the **BaseMap Alpha Modify** fadeout/cancel it's effect when the character is not facing the camera

Then you should see that your model's front hair becomes a little bit semi-transparent, which will let the viewer see through the front hair and see the eyebrow (also any other objects) under the hair.

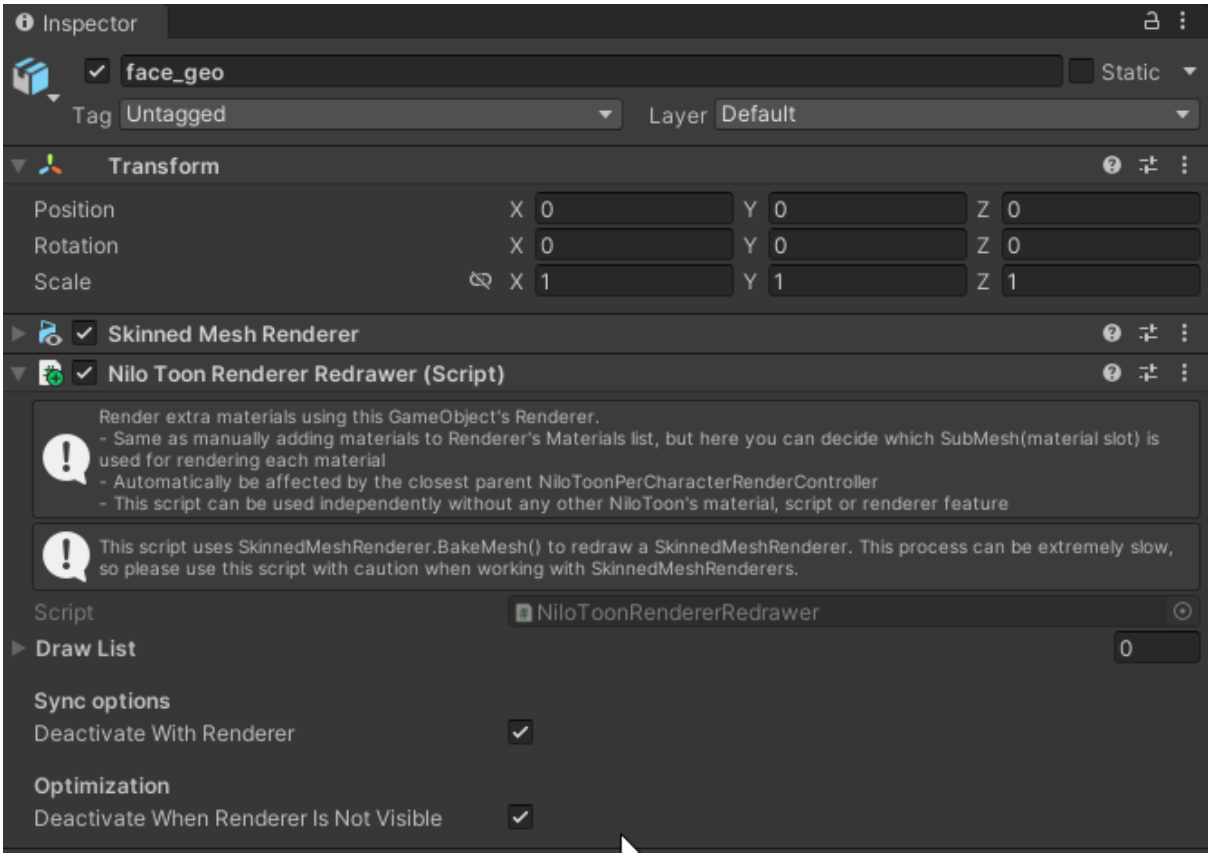
B (ZOffset eyebrow Redraw):

Steps Overview:
Draw a semi-transparent eyebrow on top of hair again, using a newly created **material variant** of the original eyebrow material's with **(Transparent/No Outline) & (ZOffset)** override, **without** relying on material's **Stencil** setting

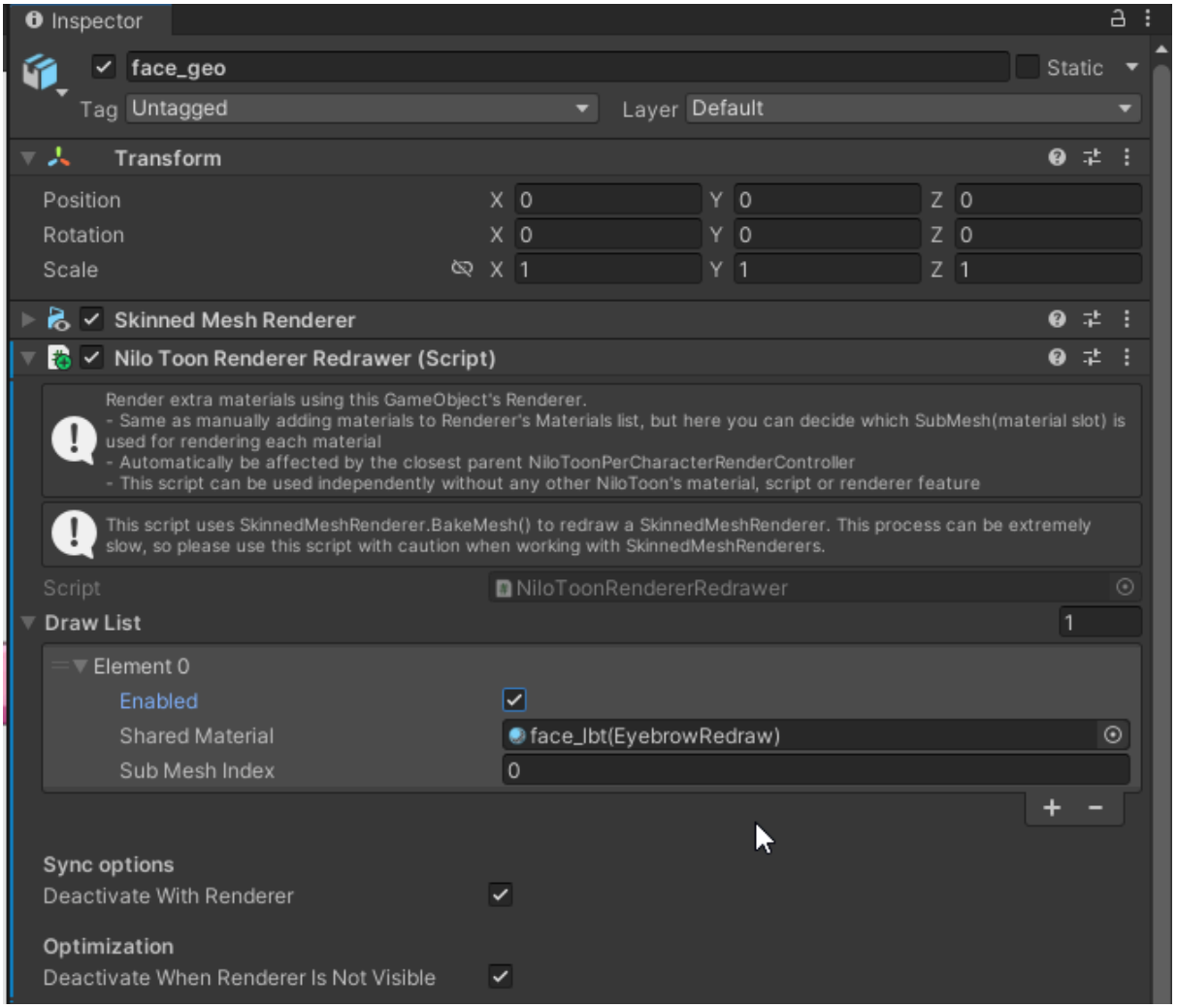
1) make a **material variant** of the original eyebrow material in your project window. Set **Surface Type = Transparent/No Outline**
(If it is a material that includes eyebrow and other parts, e.g., the whole face material, you can use the material variant's **BaseMap Alpha Modify**'s mask to remove all non-eyebrow part, you can optionally keep the upper part of eyelash also if the art style suits your needs)



2) Add **NiloToonRendererRedrawer** script to the eyebrow's renderer, this script can allow you to redraw a renderer's target sub mesh again using a material that you provided.
NiloToonRendererRedrawer is similar to adding an extra material to the end of a renderer's material list, but **NiloToonRendererRedrawer** allows you to choose the sub-mesh index.


Inspector
face_geo
Static
Tag Untagged
Layer Default
Transform
Position X 0 Y 0 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
Skinned Mesh Renderer
Nilo Toon Renderer Redrawer (Script)
Render extra materials using this GameObject's Renderer.
- Same as manually adding materials to Renderer's Materials list, but here you can decide which SubMesh(material slot) is used for rendering each material
- Automatically be affected by the closest parent NiloToonPerCharacterRenderController
- This script can be used independently without any other NiloToon's material, script or renderer feature
This script uses SkinnedMeshRenderer.BakeMesh() to redraw a SkinnedMeshRenderer. This process can be extremely slow, so please use this script with caution when working with SkinnedMeshRenderers.
Script NiloToonRendererRedrawer
Draw List 0
Sync options
Deactivate With Renderer
Optimization
Deactivate When Renderer Is Not Visible

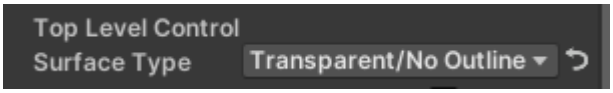
3) by clicking the **+** button of **NiloToonRendererRedrawer**, you will add a Redraw request. Assign the **material variant** in step(1) to **NiloToonRendererRedrawer,** set **Enabled** to **true**, and set the correct **Sub Mesh Index** that draws the eyebrow (it is the same index from Renderer's Materials list. Index number starts from 0, not 1).

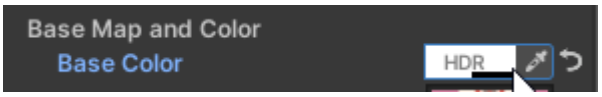


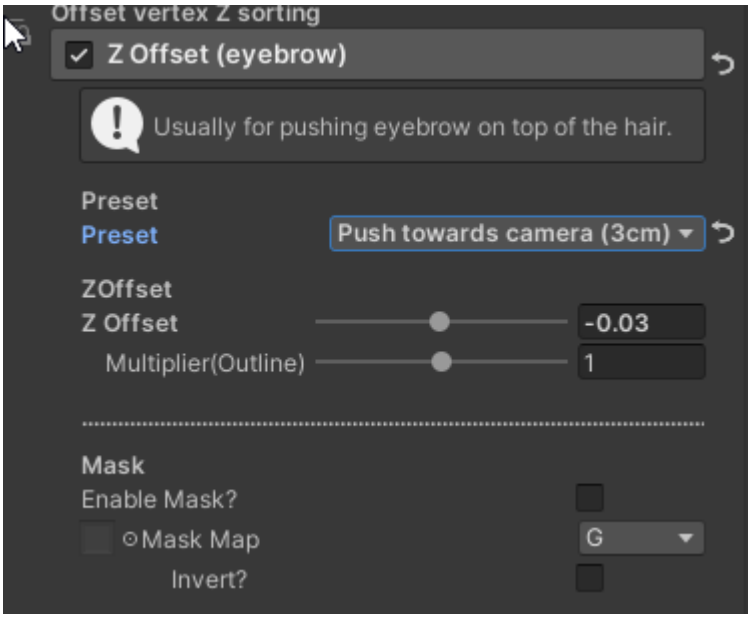
After the above steps, the render result will not change yet since the material variant is currently not having the correct setting to let it show on top of hair.

4) now edit the **material variant** to use the following settings

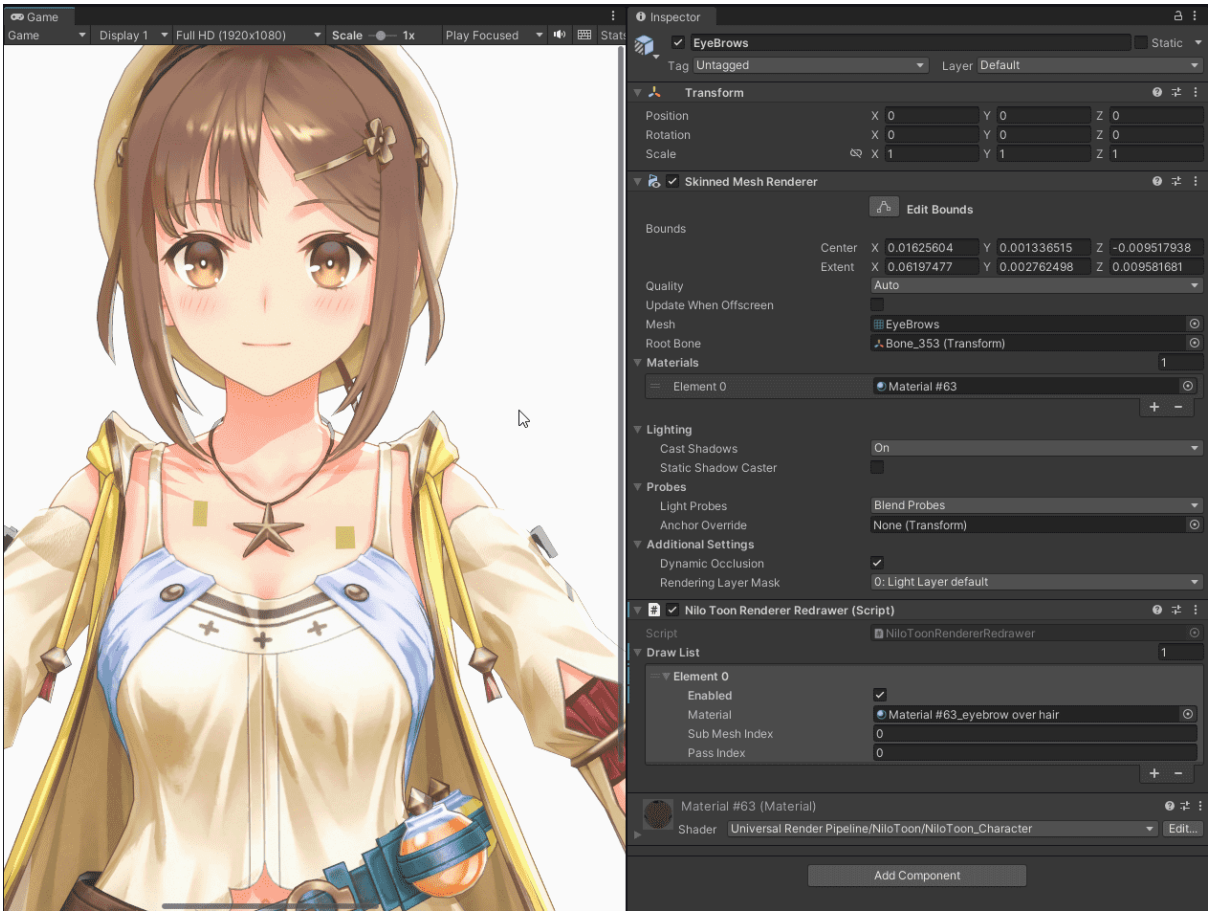
- **Surface Type = Transparent/No Outline**
- **BaseColor's alpha = < 1 (e.g., 0.5)**
- **ZOffset group = enable, and preset set to 3cm or larger**

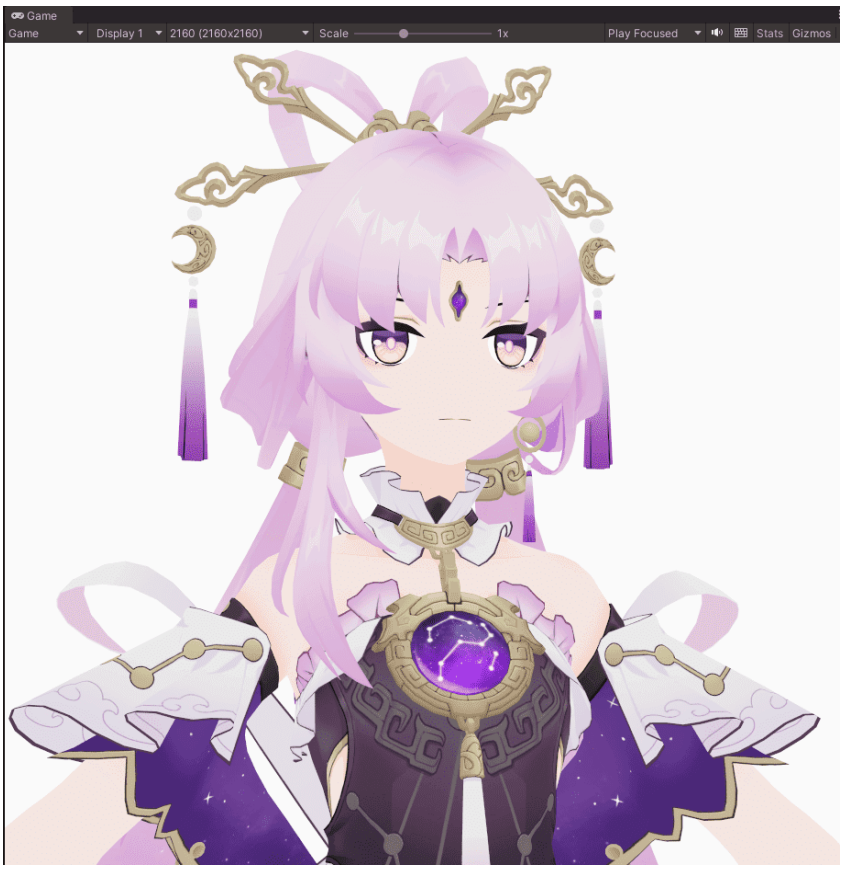


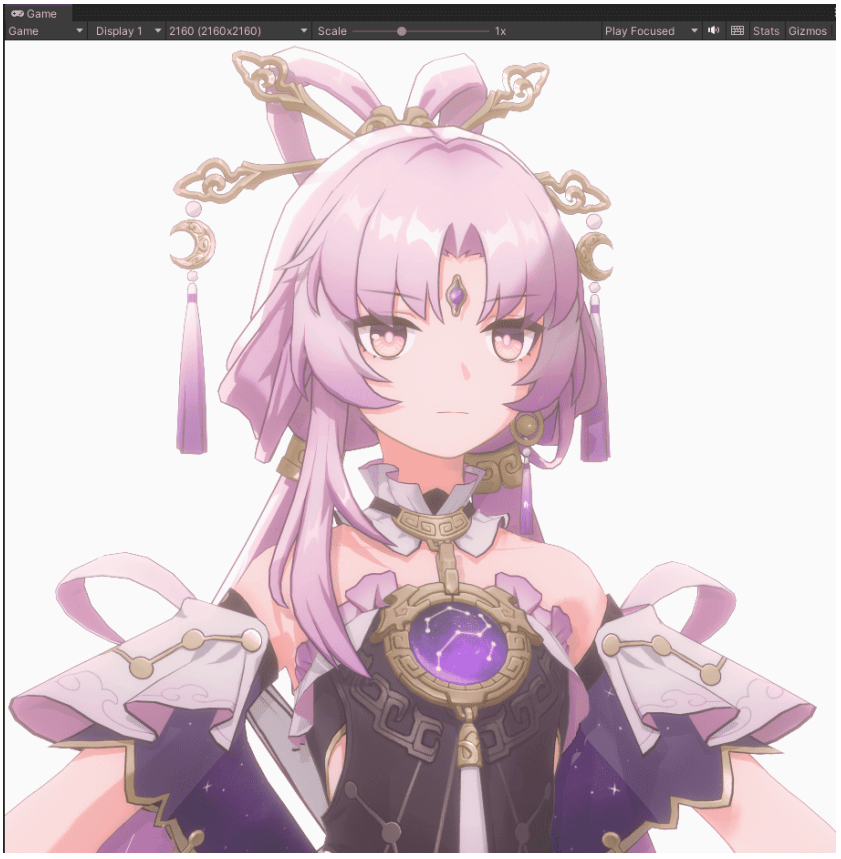




5) After step 1->4, you should now see the semi-transparent eyebrow material rendered on top of the hair. The eyebrow will block the hair outline also.
After adjusting the ZOffset value(e.g. **-0.025~-0.1**) to match your front hair 3D shape, you can check the eyebrow from any camera angle by rotating the camera/character in the scene window, the semi-transparent eyebrow should still look correct in any angle.


Game
Game
Display 1
Full HD (1920x1080)
Scale
1x
Play Focused
Inspector
EyeBrows
Static
Tag
Untagged
Layer
Default
Transform
Position
Rotation
Scale
Skinned Mesh Renderer
Edit Bounds
Bounds
Center
Extent
Quality
Auto
Update When Offscreen
Mesh
EyeBrows
Root Bone
Bone_353 (Transform)
Materials
Element 0
Material #63
Lighting
Cast Shadows
On
Static Shadow Caster
Probes
Light Probes
Blend Probes
Anchor Override
None (Transform)
Additional Settings
Dynamic Occlusion
Rendering Layer Mask
0: Light Layer default
Nilo Toon Renderer Redrawer (Script)
Script
NiloToonRendererRedrawer
Draw List
Element 0
Enabled
Material
Material #63_eyebrow over hair
Sub Mesh Index
Pass Index
Material #63 (Material)
Shader
Universal Render Pipeline/NiloToon/NiloToon_Character
Edit...
Add Component

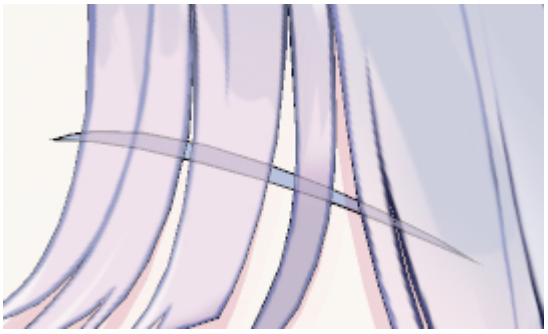




*Images above is another example of the just mentioned **Method B**, look at the **eyebrow and upper eyelash** area, a semi-transparent eye will draw **on top of hair** due to **NiloToonRendererRedrawer** and your new **material variant** of the original eyebrow

C (Stencil+Render Queue hair Redraw):

(This solution is usually visually perfect for showing eyebrows in a semi-transparent way, but it is **much more complex!** And will assume that you are already familiar with **NiloToonRendererRedrawer**. Also **Render Queue**, **Stencil** of NiloToon Character shader)



Let's see some example videos:

- NiloToon Semi-transparent hair for eye
- https://x.com/kim_raming/status/1908018945631858895
- [NiloToon Unity URP shader] Girls Frontline 2 Exilium - Sharkry rendering
- [NiloToon Unity URP shader] Girls Frontline 2 Exilium (2023 CBT3) - MP…

Steps Overview:

0.Requires the eyebrow being a separated material
1.Draw the face (without eyebrow) as usual without stencil settings
2.Draw the eye/eyebrow, with stencil write
3.Draw the original hair which discards the pixel on eyebrow pixels using the stencil buffer. (Imagine you draw the original hair material with a hole, and the hole only allows you to see the eyebrow)
4.(key step)Redraw the hair again in semi-transparent, which discards the pixels on non-eye pixels (Imagine you draw a small piece of semi-transparent hair just to filling the stencil hole of step 3)

顔パーツを髪の上に半透明で描画

顔パーツを髪の上に描画 に加えて、顔パーツの領域に半透明で髪を描画することで実現できます。

プロパティ	肌	顔パーツ	髪	髪（透過）
Ref	0	1	1	1
Comp	Always	Always	NotEqual	Equal
Pass	Keep	Replace	Keep	Keep
Render Queue	2450	2451	2452	2460

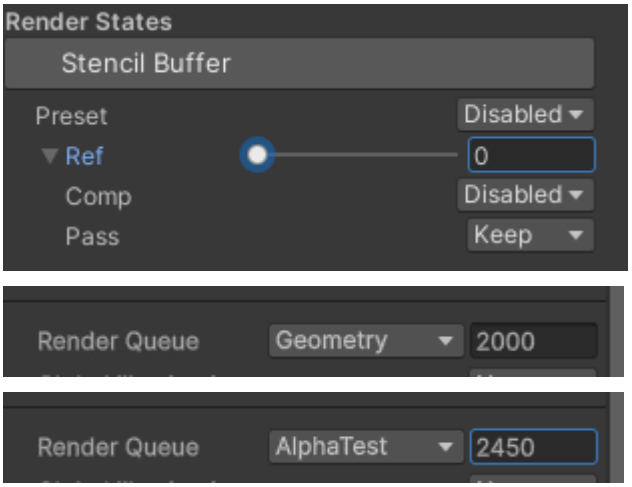
1. 肌が描画される
2. 顔パーツが描画され、その領域が `1` で置き換えられる
3. `1` ではない領域（顔パーツ以外のところ）に髪が描画される
4. `1` の領域（顔パーツのところ）に髪（透過）が描画される

*You must control **render queue** of each material to ensure steps(1~4) are draw in the correct order, wrong render queue order will produce wrong result

Detail steps:

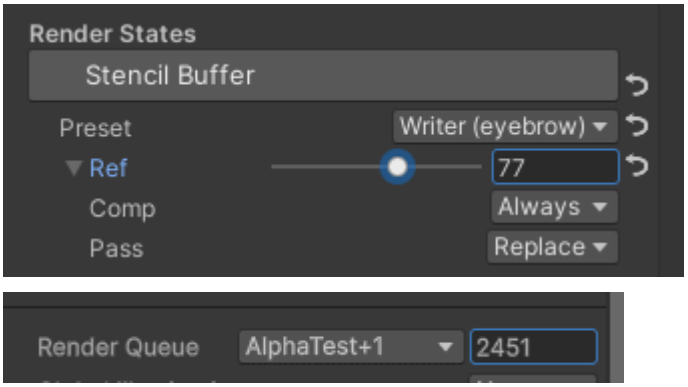
1.Draw all **face** materials (without eyebrow) normally first (**without stencil** settings), these face materials will only fill the depth buffer normally, without editing any stencil buffer.

(For example, these face material can use **render queue 2000~2450**)

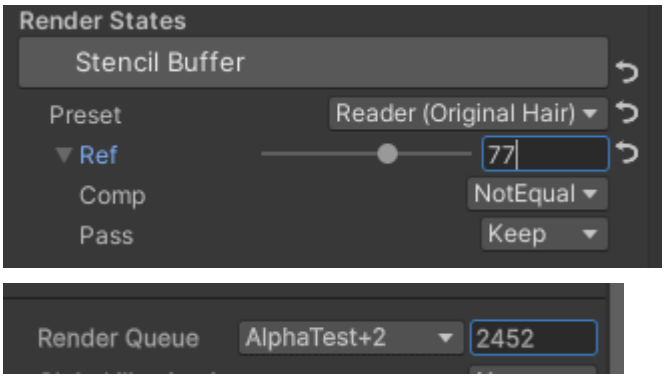


2.Draw **eye, eyelash** and **eyebrow** materials. In the **Stencil** section, use preset **Writer (eyebrow)**, which sets **Comp** = **Always**, **Pass** = **Replace.** You need to set a special **Ref**(e.g. **77**), so for every pixel of the eye that passed the depth test(not occluded by step1's face material), those eye pixels will **replace**(write) **Ref** ID into the stencil buffer.

(For example, these eye/eyebrow materials can use **render queue 2451**)

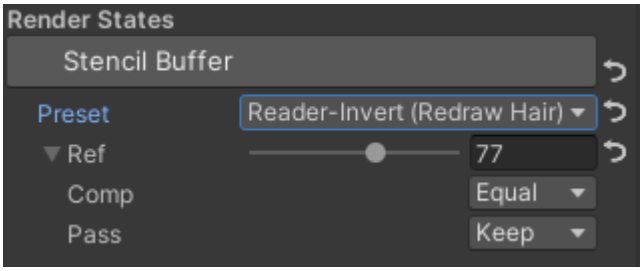


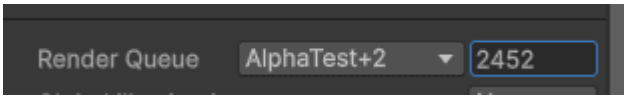
3.Draw **front hair** materials. In the **Stencil** section, use preset **Reader (Original Hair)**, which sets **Comp** = **NotEqual**, **Pass** = **Keep.** You need to set the **same Ref** in step2 (e.g. still 77). This hair material can't draw on the eye area anymore due to stencil, which means it will draw the hair normally but leaving a hair hole on the eye area.
(For example, these eye/eyebrow materials can use **render queue 2452**)



After finish step3, you should see the eyebrow become completely visible on top of the hair, without any semi-transparent hair alpha blending.

4.Finally, the key step, **create** a new **material variant** of step3's hair material in your project window, then use **NiloToonRendererRedrawer** or manual material append to **redraw it**. That **semi-transparent front hair material** should use preset **Reader-Invert (Redraw Hair)**, which sets stencil **Comp** = **Equal**, **Pass** = **Keep** with the same **Ref** ID in step2 (e.g. still 77). This will only fill the hair hole left out by step3.
(For example, these eye/eyebrow materials can use **render queue 2452**)



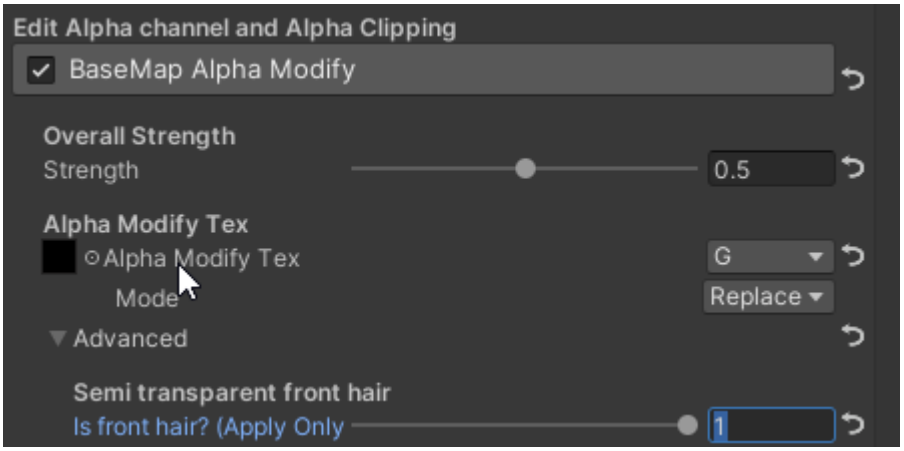


*The **Ref** ID **77** can be any number, recommended to be unique between characters!
*remember the draw order is 1 > 2 > (3+4), you **must** control the draw order using **render queue** of each material

After finish step4, you should see the whole hair become opaque again, it is a correct result before step5.

5.The last step is to make that **hair material variant** in step4 becomes semi-transparent. Apply these settings to that material variant:
- **Surface Type** = **Opaque(Alpha)/Outline** or **Opaque(Alpha)/No Outline**

- **BaseMap Alpha Modify** enabled**,** assign **PureBlack4x4** texture from the NiloToonURP folder.
    - Set the **strength** to a value you like, it controls how much visibility of eye over the hair, for example set **0.25** ~ **0.5**
    - Set Semi transparent front hair > **Is front hair?** = **1**

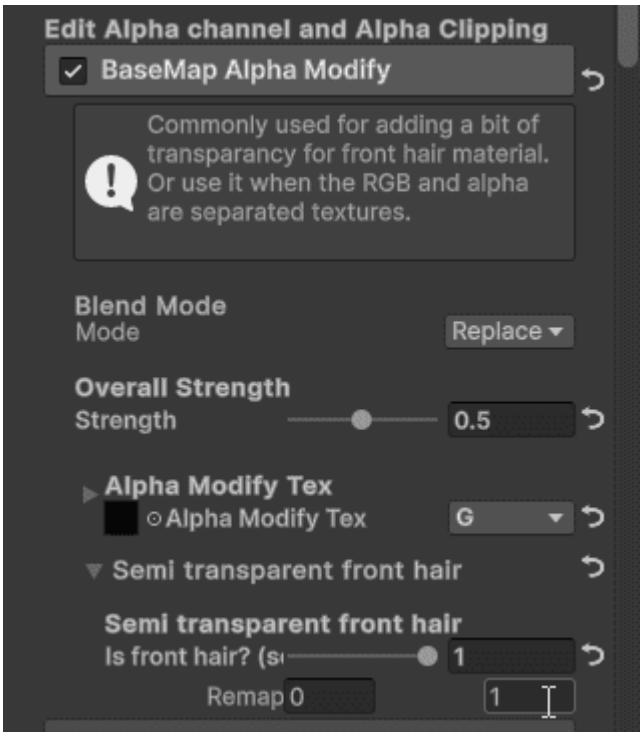


This is the end of the setup. With these settings, that new material variant of step4 will **fill the hair hole** of step3 by redrawing a semi-transparent hair only on top of the hair hole.
*No **Z Offset (eyebrow)** is needed for all steps 1-5.
*above pictures are settings of that new semi-transparent **hair material variant** in step4/5

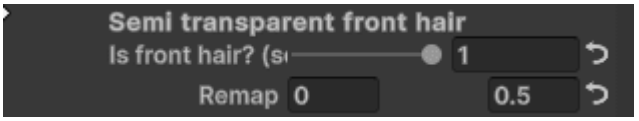
You should now get the “semi-transparent hair on eyebrow” result after doing all the above steps (look at the eyebrow area in the picture below). Rotate the character

prefab in 360 degrees to check if the eyebrow’s visual is good for all camera angles.

Due to setting Is front hair? to 1, the semi-transparent material will adjust the alpha effect according to camera angle. To control how the “semi-transparent hair effect” fadeout to camera angle, use the **Remap** section in the picture below



For example, if you want to let the semi-transparent effect show for more camera angle, lower the right side value. For example, set remap to [0,0.5]



Here is the expected result after all the setup, for your reference:

Although it is unlikely to have this problem, but If you still see the eyebrow appears in the back side of hair, it is likely that you have **separated renderer** for front hair and back hair, but they **shares the same hair material**,

To solve this problem, please read the **Backup plan**.

Backup plan:

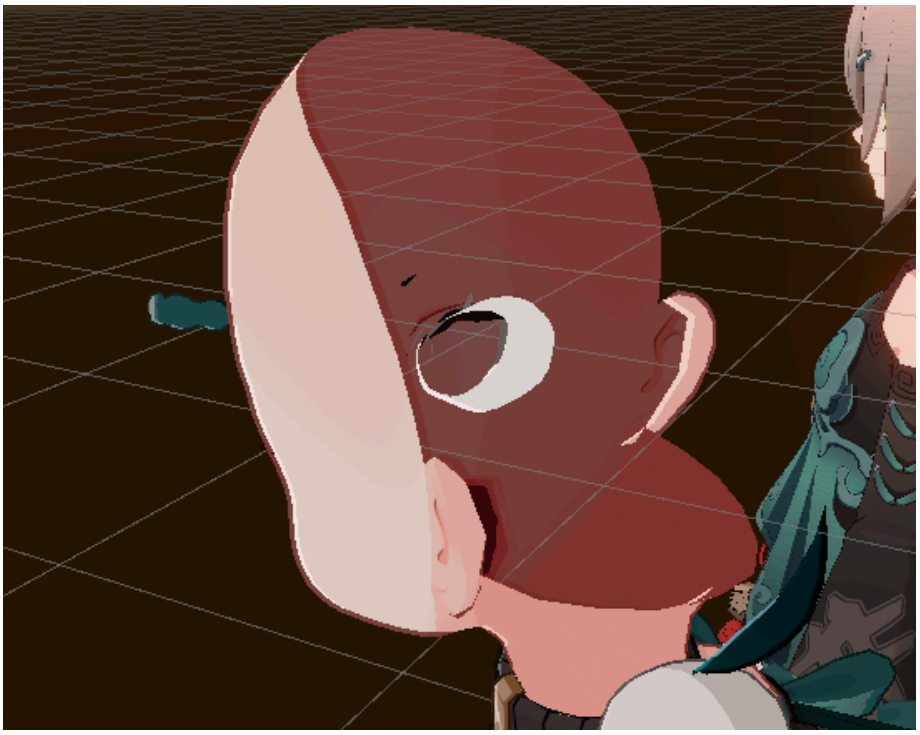
Method (A):

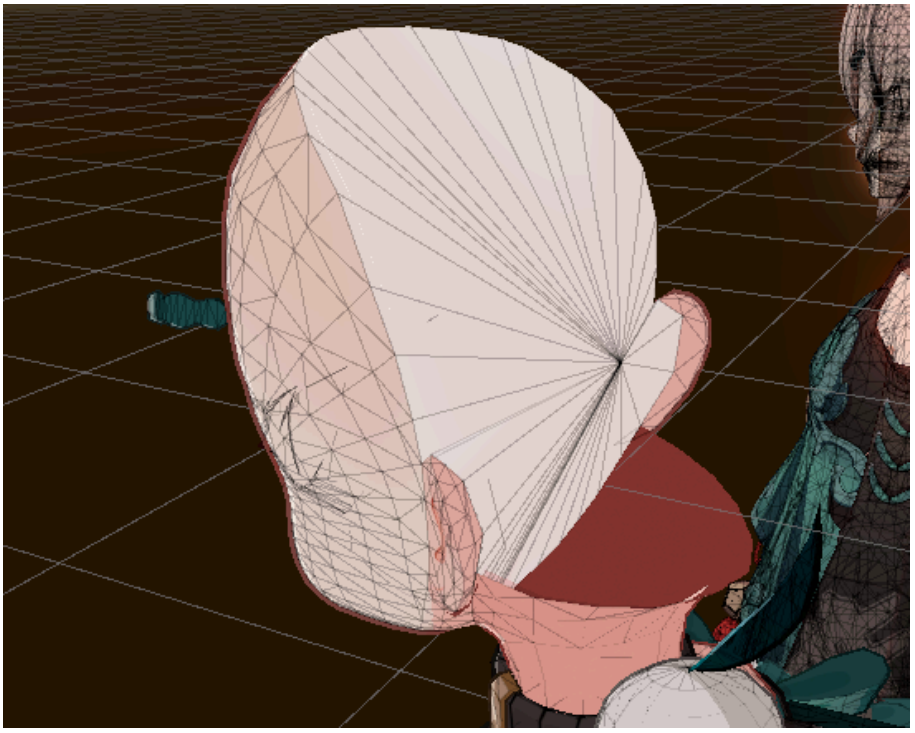
Copy the **NiloToonRendererRedrawer** from front hair's renderer, **paste as new** to other hair's renderer that is showing the eye wrongly(those renderers are expected to share the same material with the front hair renderer)

Method (B):

Switch the hair material that is showing the eye wrongly to another material that without stencil settings

Method (C):
Add a “**blocker mesh**(RenderFace = both)” inside the head to block the draw of eyebrow’s stencil, where this “blocker mesh”’s renderqueue is between step(1),step(2), this will block the step2 eyebrow’s stencil write due to “blocker mesh” filling depth buffer first, and step2 can’t pass the depth test when viewed in side/back camera angles. This will solve the problem by purely preventing unwanted Stencil write by the eye material in step(2).



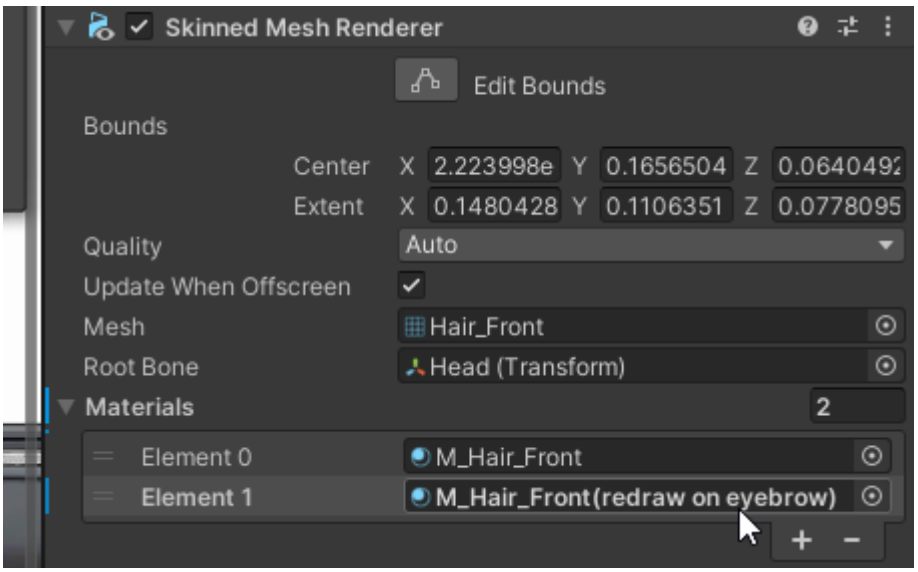


*If you want to know more methods of using the **Stencil** setting of material, you can view this list
lilToon's Stencil setting list

Since these stencil methods are quite complex and require a deep understanding of **Stencil, ZTest** and **Render Queue**, 1 small mistake in setting will result in wrong rendering. If you need any help on checking files/setting/understanding this, feel free to contact us.

NiloToonRendererRedrawer alternative

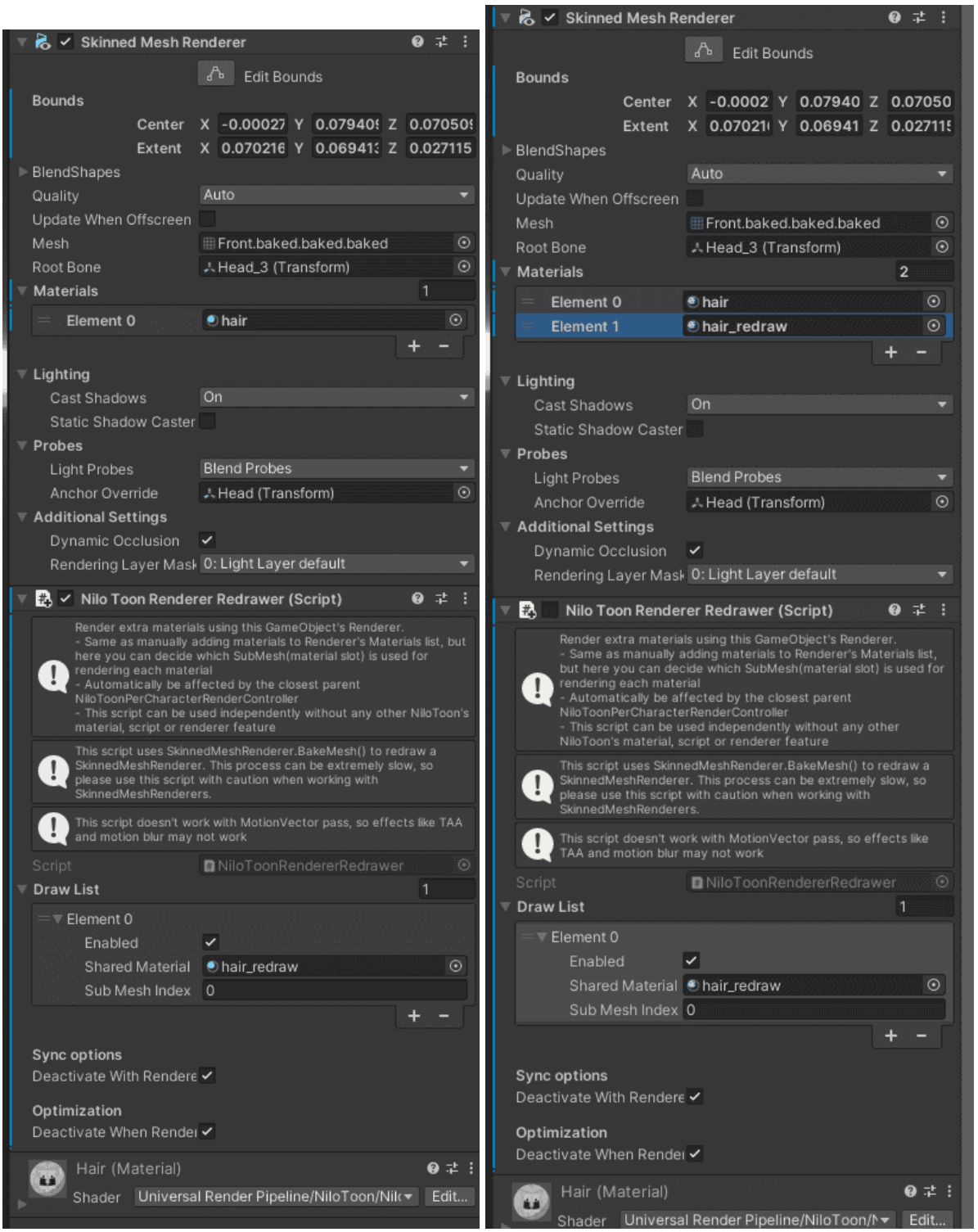
If you can't use **NiloToonRendererRedrawer** due to conflict with other tools that edit the renderer's mesh at the same time (e.g., **MagicaCloth2's bone cloth, rendering problem with NiloToon's volume**), as an alternative method you can redraw a renderer's **last submesh** by assigning(appending) extra materials to the end of renderer's **Materials** list, this doesn't requires **NiloToonRendererRedrawer**. So that allows you to redraw the last submesh, but you can not pick the submesh index the same as using the **NiloToonRendererRedrawer**.



This solution only works if your target submesh is the **last submesh** of a renderer.

Below is the before(left) and after(right):

- In the left image, it shows the result of using **NiloToonRendererRedrawer** as usual
- In the right image, it show the alternative method to redraw the hair, it doesn't use **NiloToonRendererRedrawer**, which should avoid most of the problem



If your target submesh is **not the last submesh** of a renderer,
to **reorder submeshes**, it will require you to **edit** the **fbx**.
The process depends on your 3D modeling software:

In **Blender**:
1. Select your mesh
2. Go to Edit Mode
3. Select the submeshes you want to reorder

4. Press 'P' to separate them into different objects
5. Rejoin them in the desired order using Ctrl+J (join command)
   - Join them one by one in the reverse order you want them to appear
   - The last object you select before joining will be the first submesh
   - Example for a,c,b order: Select b first, then c, then a last, then join

In **Maya**:
1. Select your mesh
2. Separate the submeshes into different objects
3. Combine them back in the desired order using Mesh > Combine
   - The order of selection determines the submesh order
   - For a,c,b order: Select objects in order a, c, b, then combine

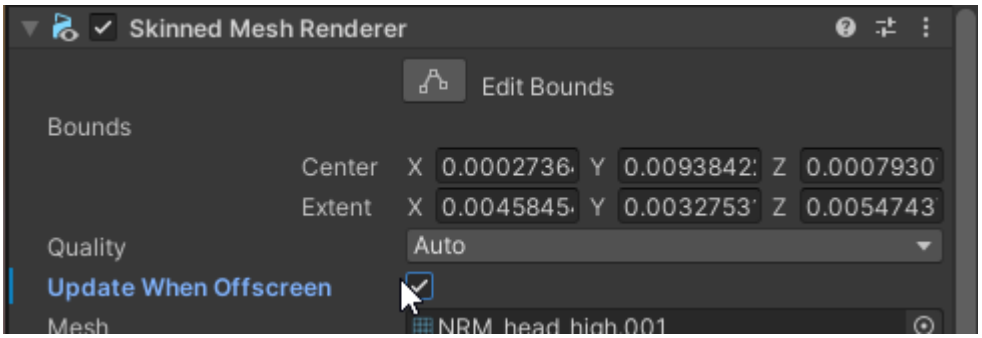
In **3ds Max**:
1. Select your mesh
2. Use Element mode to select submeshes
3. Detach them into separate objects
4. Attach them back in the desired order
   - The order of attachment determines the submesh order
   - Select the base object, then attach others in the desired order

After reordering, remember to:
1. Check your UV maps are preserved
2. Verify material assignments
3. Export as FBX
4. Test in Unity to confirm the new order

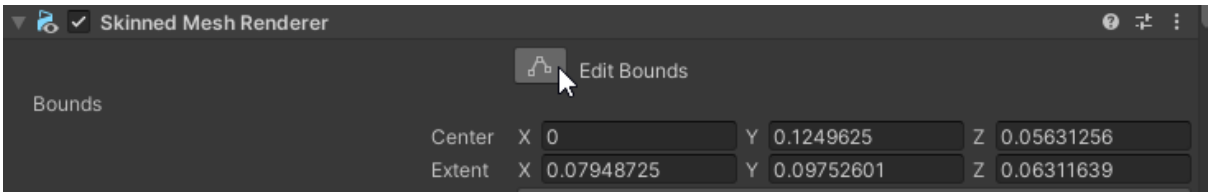
Perspective removal = wrong culling at the edge of the screen

If your character's renderer disappear when close to the edge of the screen, it maybe due to perspective removal.

First try to enable **Update When Offscreen** in all character's **Skinned Mesh Renderer**



If the above method doesn't work,
try making the **bound** of Skinned Mesh Renderer **bigger**, making the bound bigger can avoid Unity's culling system wrongly cull your renderer, because Unity's culling system always use non-perspective removal geometry for culling, and there is no way to edit Unity's culling system unless you have access to Unity's source code.

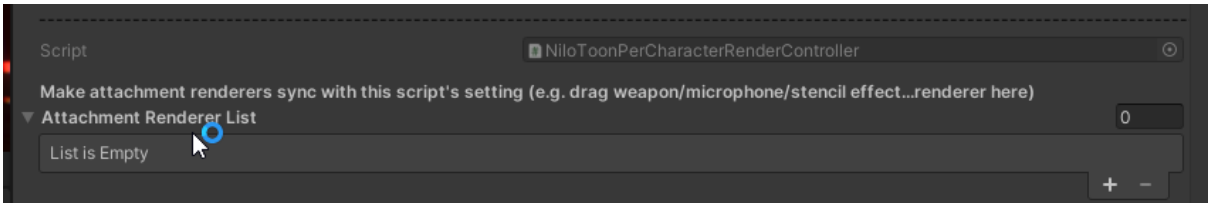


Perspective removal = objects attached to char not perfectly matched in 3D

Only NiloToon_Character shader can receive perspective removal, any other shader not supporting this feature will not be affected by perspective removal (e.g. hand can't grab a weapon/microphone perfectly correct)

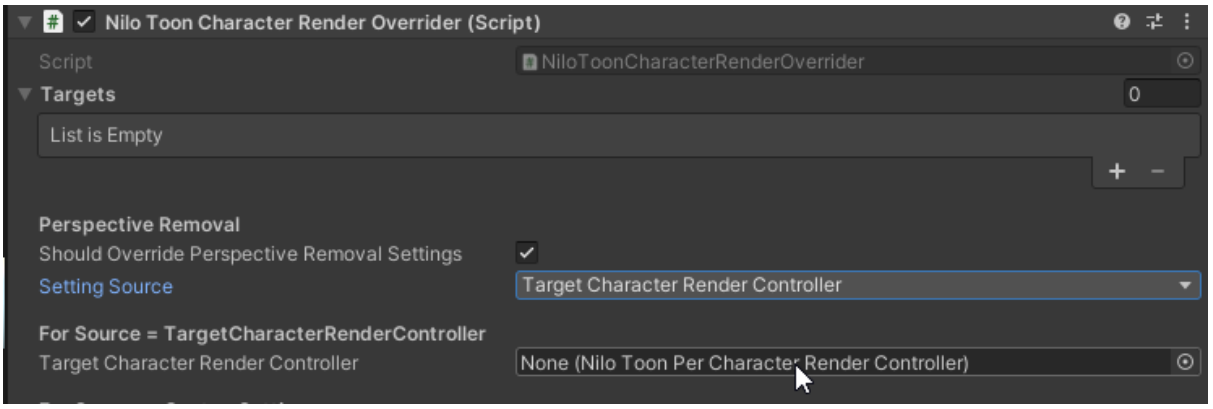
Easy method

1) Just drag target object's renderer to NiloToonPerCharacterRenderController script's attachmentRendererList (target object should be using NiloToon's character shader, and **disabled** it's **NiloToonPerCharacterRenderController**)



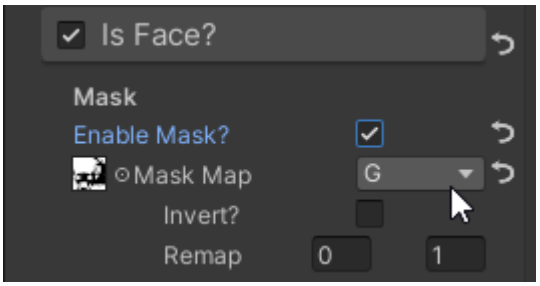
Complex method

1) Add a script NiloToonCharacterRenderOverrider to any GameObject
2) Drag the character's root script to TargetCharacterRenderController slot
3) Drag all attached objects(e.g. weapon) to Targets



If renderers inside **Targets** are all using NiloToon_Character shader, all Target's perspective removal result will be sync to that character(perspective removal result will be the same)

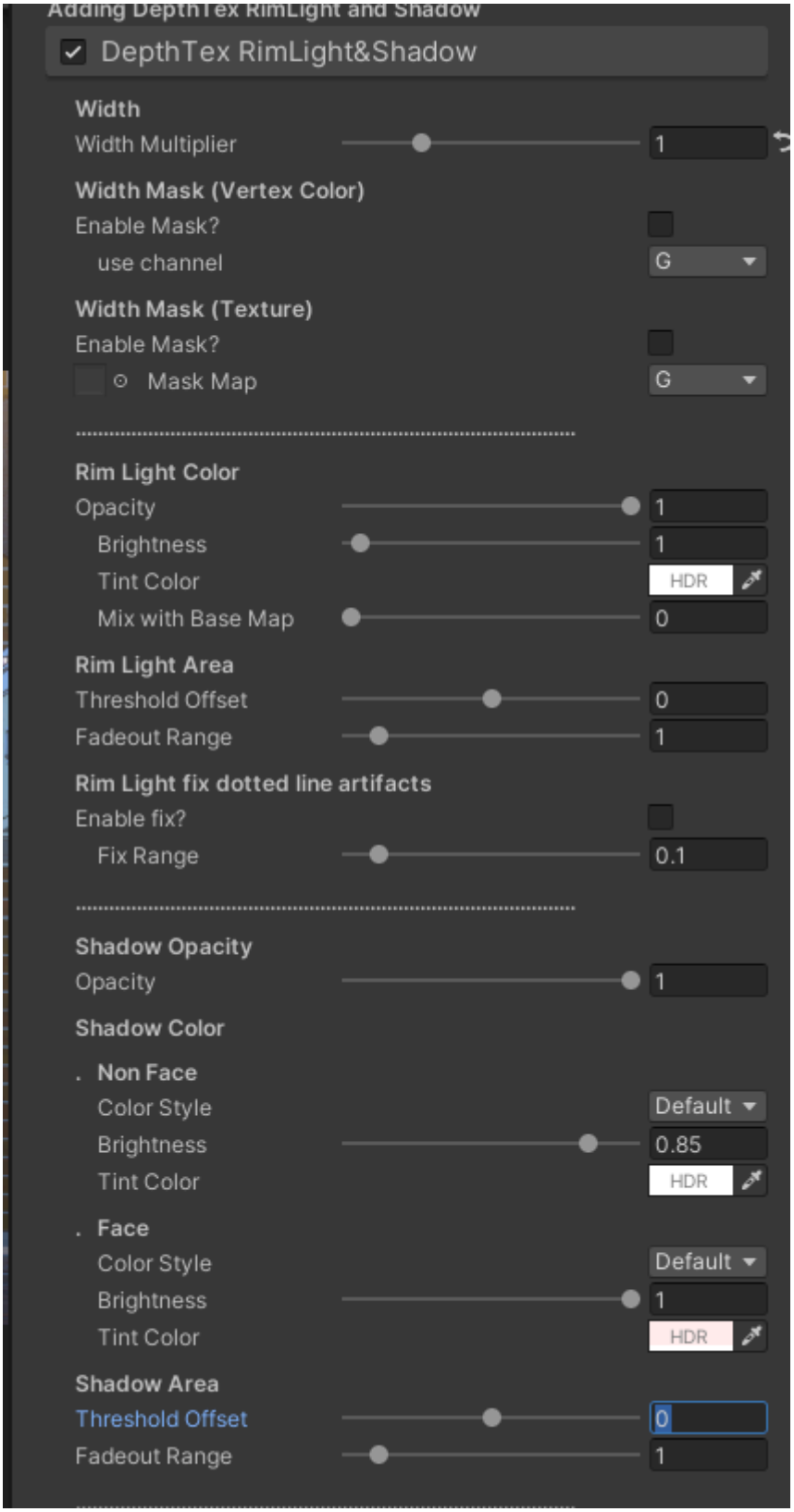
Masking IsFace? in mat



Use **“Is Face?”** section’s **“Mask Map”** to select valid face area, make sure to enable **“Enable Mask?”**

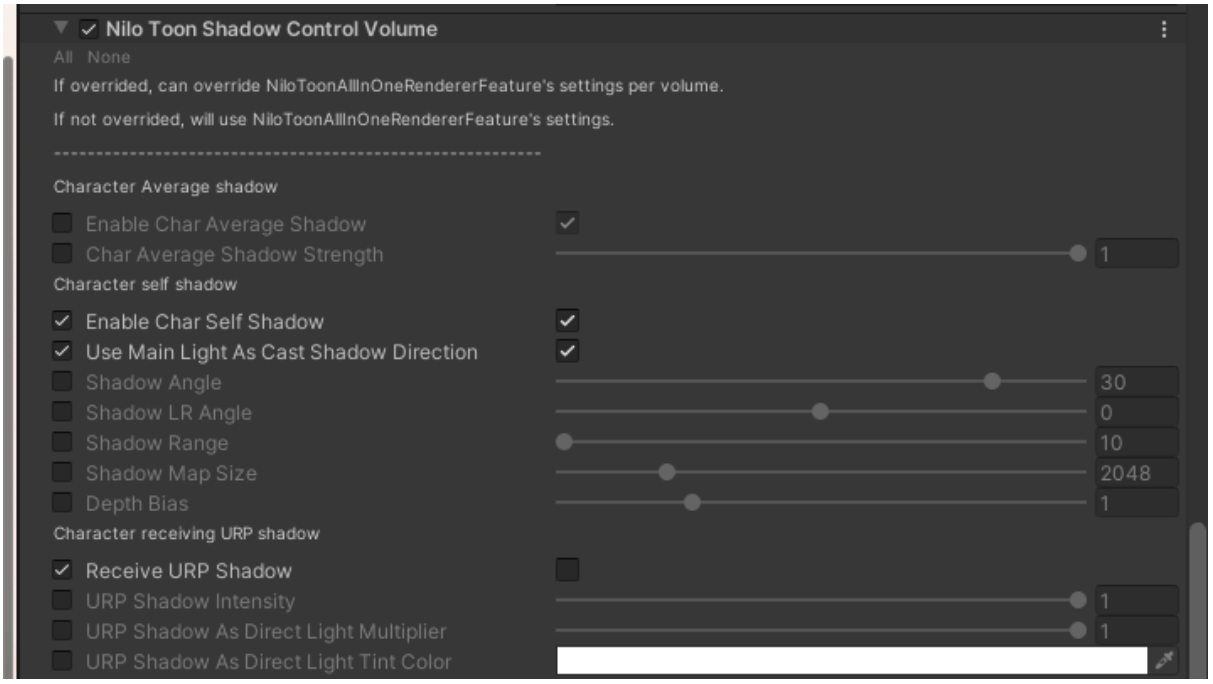
It is useful when you combined face and non-face parts into the same material

2D shadow visibility threshold



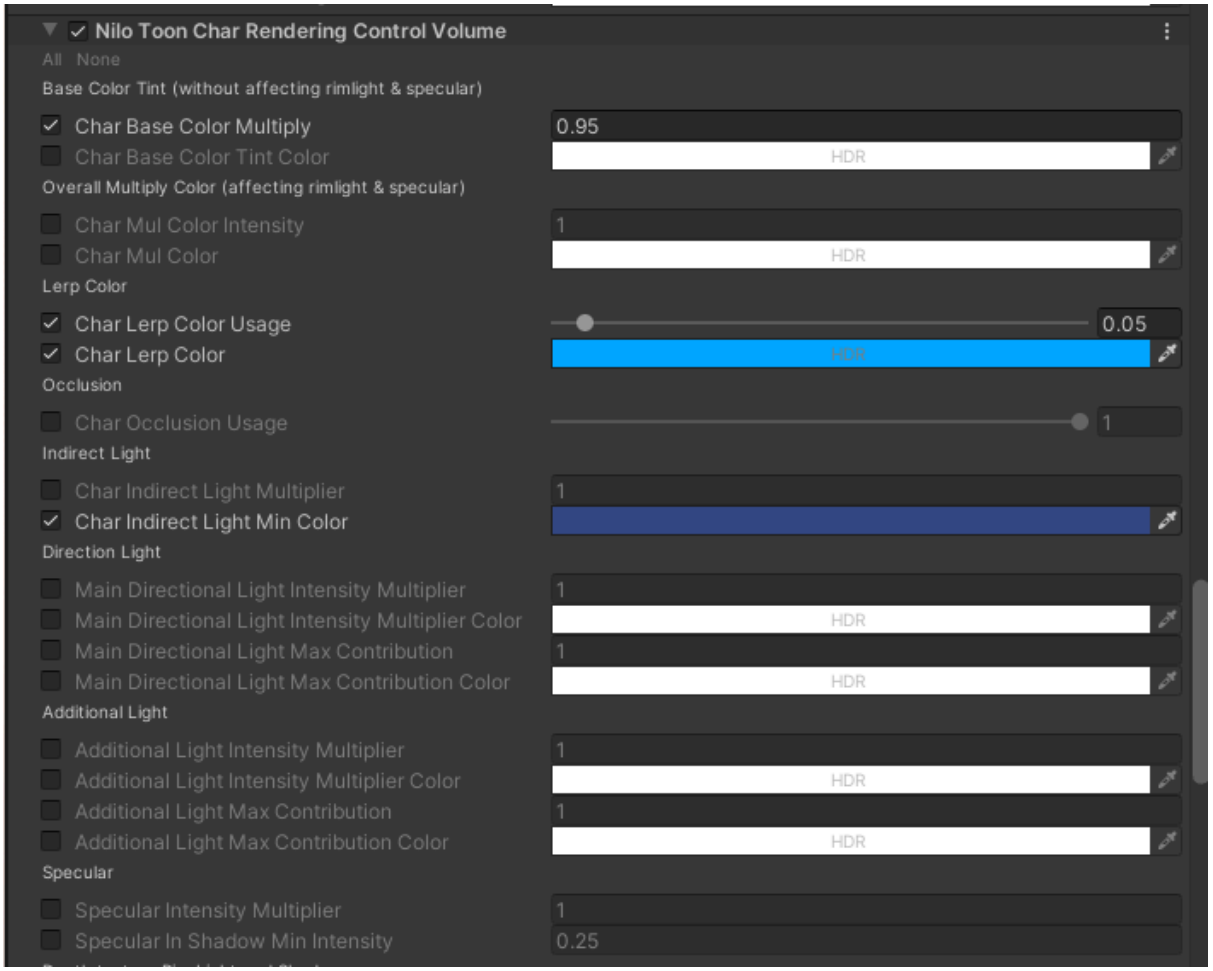
if 2D shadow is missing when shadow caster is very close to shadow receiving surface, you can tweak DepthTex RimLight&Shadow -> Shadow Area's **Threshold Offset** and **Fadeout Range**

Char shadow style (volume)



add **NiloToonShadowControlVolume** to a volume script in your scene if you didn't, then you can control the shadow's visual/color globally. For example, you can use the "Character receiving URP shadow" section, to change how much light should be blocked by URP's shadow system.

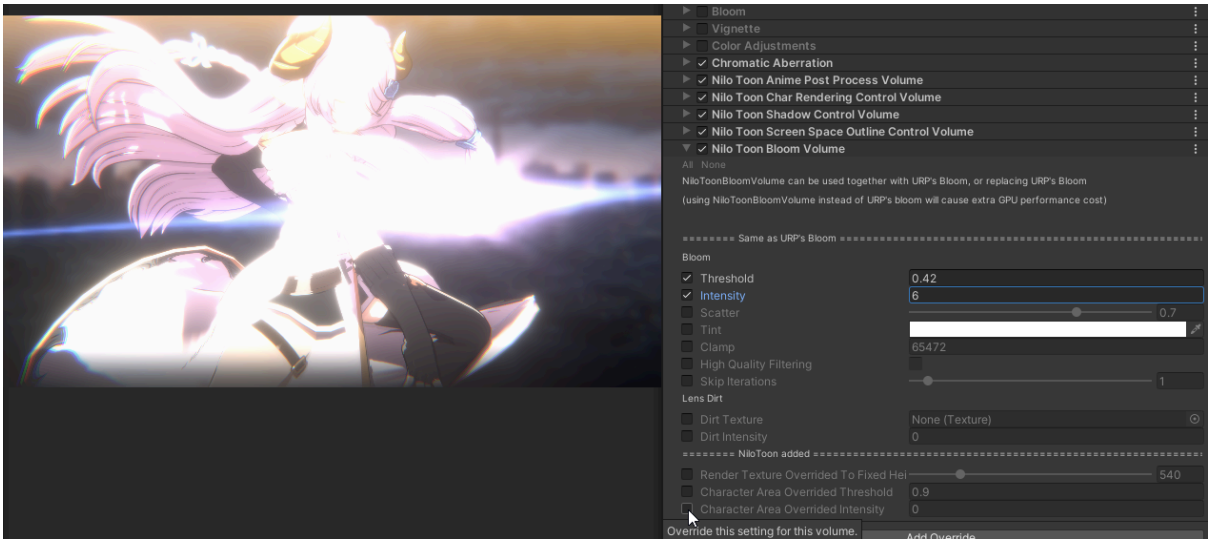
Char render style (volume)

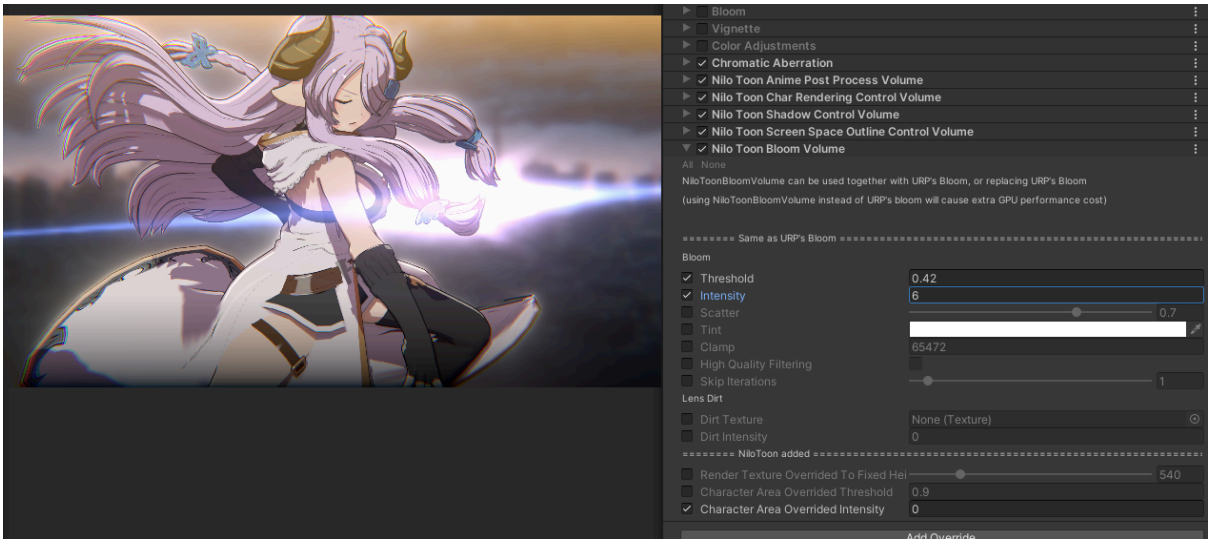


add **NiloToonCharRenderingControlVolume** to a volume script in your scene if you didn't, then you can control the character's visual/style/color globally.

Reduce bloom on character

You can try using NiloToon/Bloom instead of URP's Bloom, so you can have extra control on the character area's bloom result.





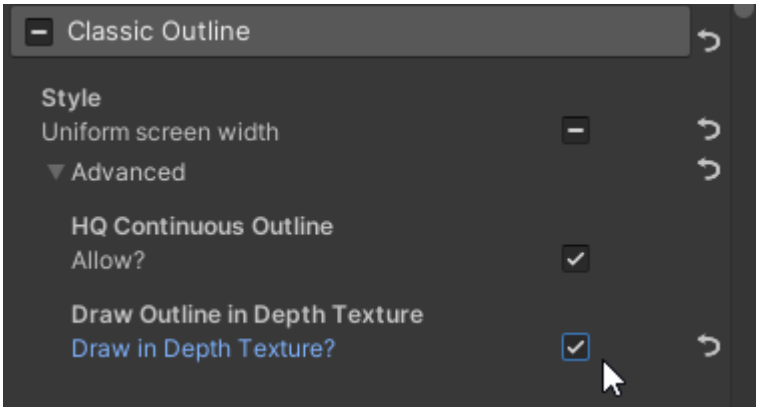
Outline blurred by Dof

(Not recommended anymore, this option will be removed and NiloToon will provide an auto fix in future version via RenderGraph)
Classic Outline blurred by **Depth of Field**

Classic Outline **NOT** blurred by **Depth of Field**

To make classic outline **NOT blurred by Depth of Field**, your NiloToon Character materials need to meet these conditions:

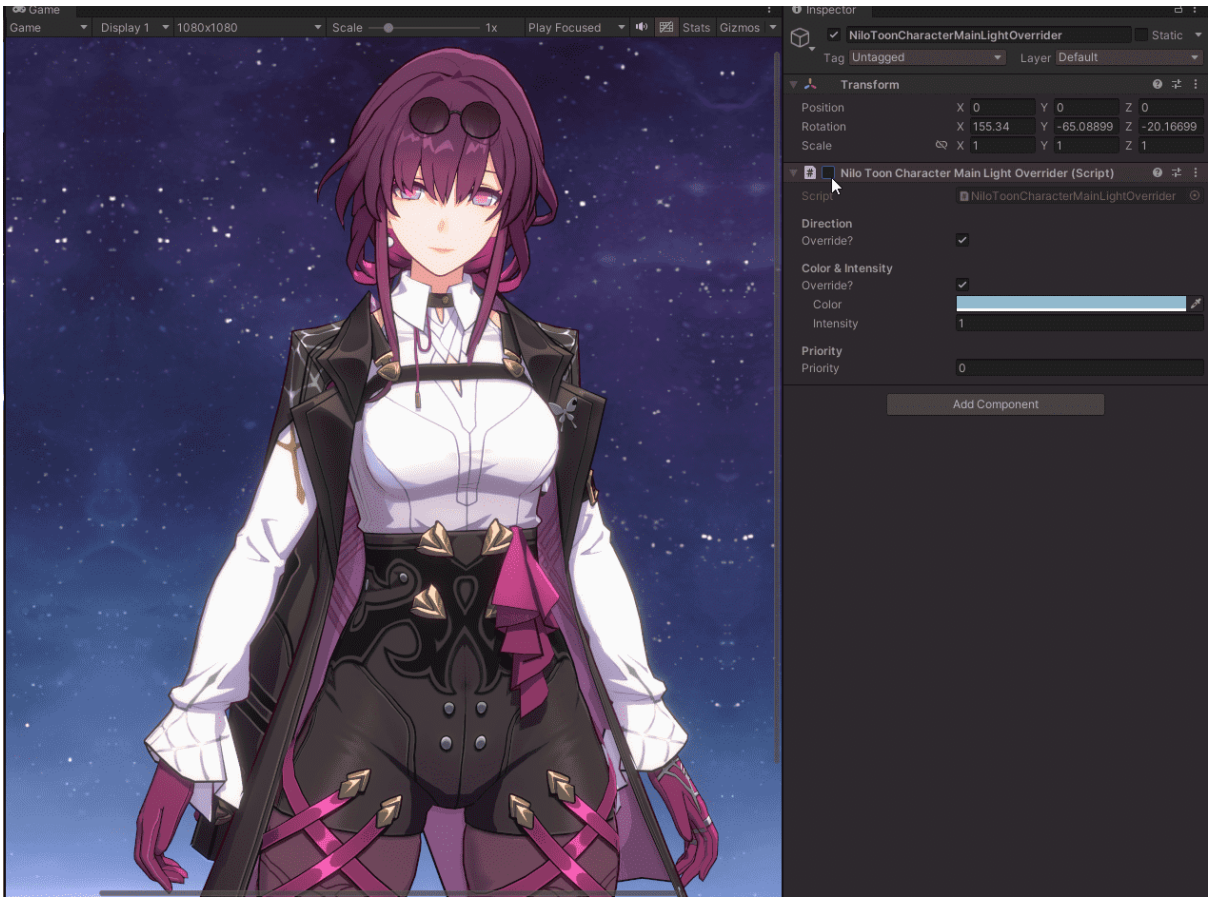
- the model is an .fbx, so uv8 smooth outline data can be generated
- HQ Continuous Outline is ON
- Draw in Depth Texture? is ON

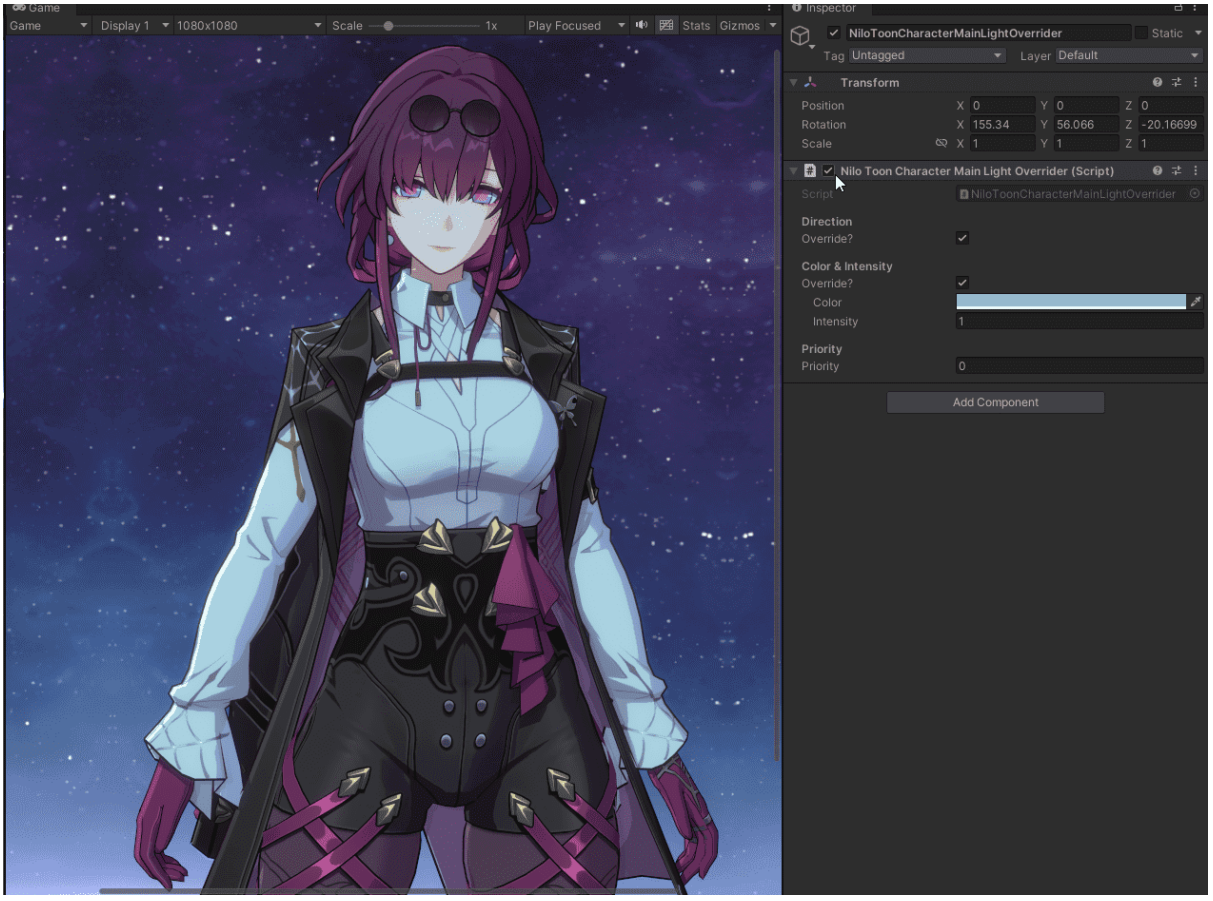


Enabling this **Draw in Depth Texture?** option will prevent NiloToon_Character's motion vector pass's render, so if you will use **TAA/DLSS/Motion blur**…, we highly **NOT** recommend enabling **Draw in Depth Texture?**. Due to this reason, the default value of **Draw in Depth Texture?** is off

Face SDF shadow (Genshin Impact/Honkai: Star Rail)

To create face shadow similar to these videos:
[NiloToonURP] +face shadow gradient texture feature
シュガーラッシュ / **miComet(official)**


Game
Display 1
1080x1080
Scale
1x
Play Focused
Stats
Gizmos
Inspector
NiloToonCharacterMainLightOverrider
Static
Tag
Untagged
Layer
Default
Transform
Position
X 0
Y 0
Z 0
Rotation
X 155.34
Y -65.08899
Z -20.16699
Scale
X 1
Y 1
Z 1
Nilo Toon Character Main Light Overrider (Script)
Script
NiloToonCharacterMainLightOverrider
Direction
Override?
Color & Intensity
Override?
Color
Intensity
1
Priority
Priority
0
Add Component

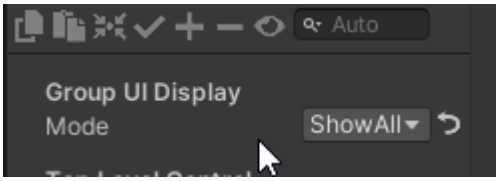

Game
Display 1
1080x1080
Scale
1x
Play Focused
Stats
Gizmos
Inspector
NiloToonCharacterMainLightOverrider
Static
Tag
Untagged
Layer
Default
Transform
Position
X 0
Y 0
Z 0
Rotation
X 155.34
Y 56.066
Z -20.16699
Scale
X 1
Y 1
Z 1
Nilo Toon Character Main Light Overrider (Script)
Script
NiloToonCharacterMainLightOverrider
Direction
Override?
Color & Intensity
Override?
Color
Intensity
1
Priority
Priority
0
Add Component

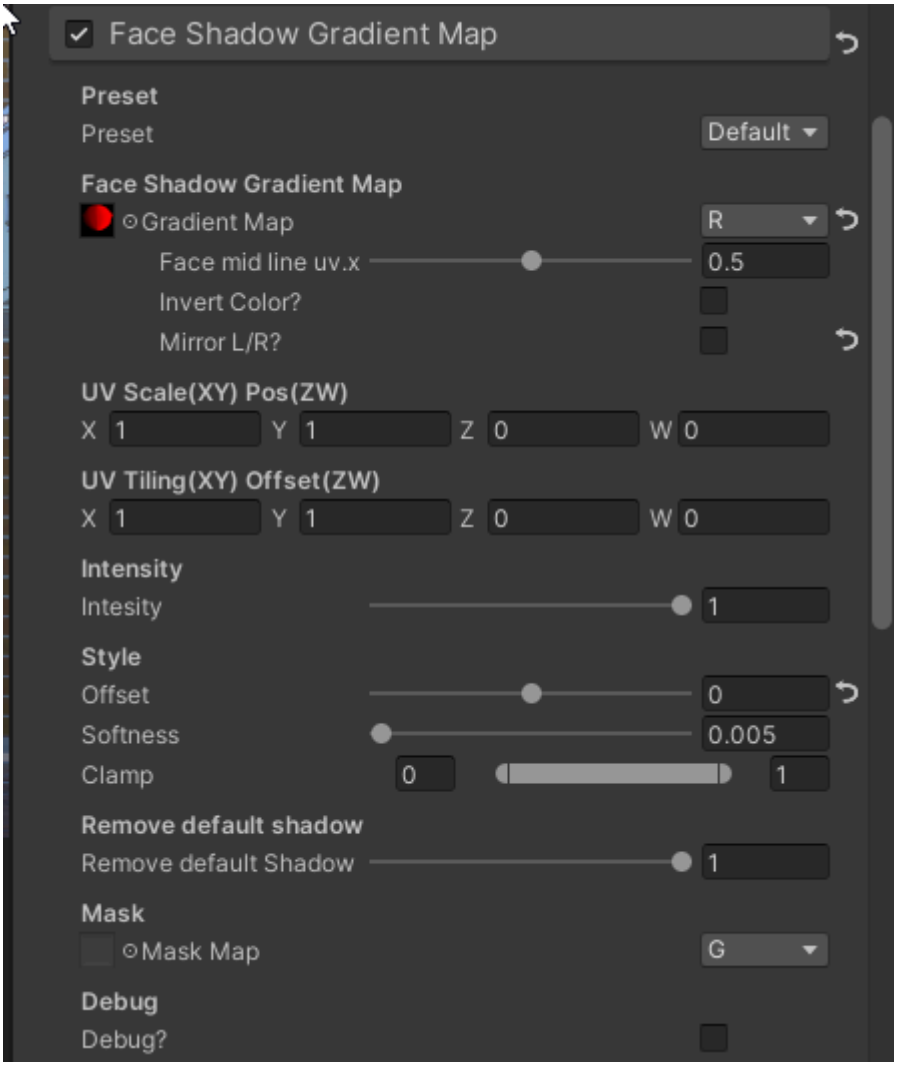
View an example

You can first open the demo project, drag
- NPC_Homeworld_Avatar_Loli_Catalyst_Klee_Edit (NiloToon).prefab
into the scene, and see Klee's face material **Avatar_Loli_Catalyst_Klee_Mat_Face.mat**.
*Rotate the main directional light and character prefab to see the face SDF shadow result

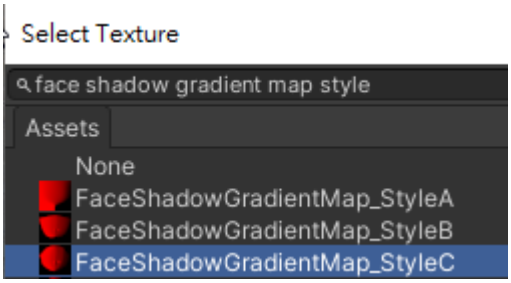
Apply to your model

To set up a similar shadow in your face material, **Face Shadow Gradient Map (SDF)** in the material inspector is the feature to render face shadow similar to Genshin Impact.
(To see this feature, you need to first set **Group UI Display Mode** to **ShowAll**)

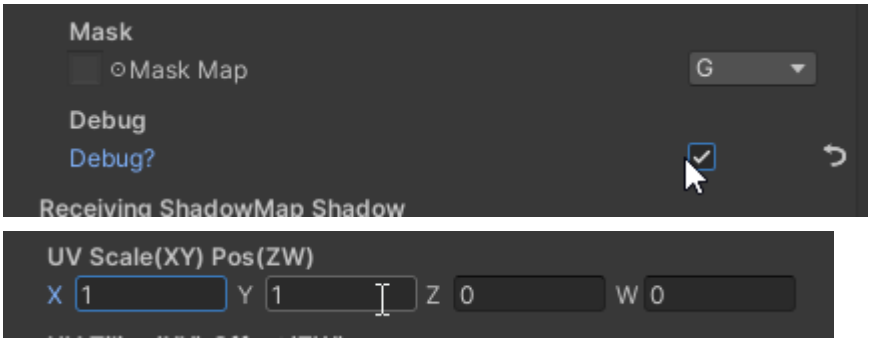




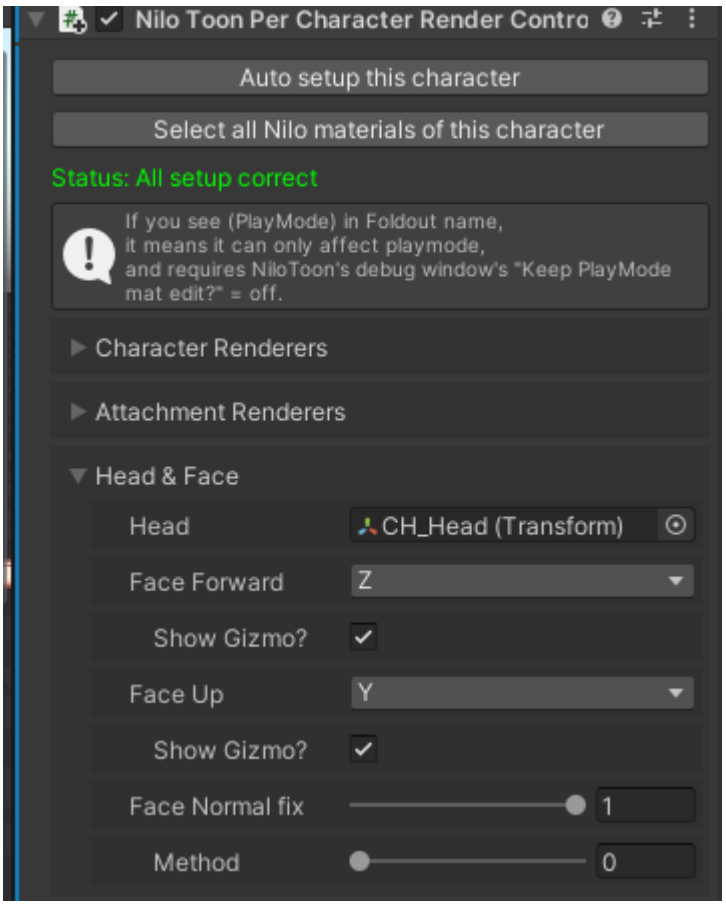
You will need to assign a **FaceShadowGradientMap** to the Gradient **Map** slot.
(you can search these example textures by searching **FaceShadowGradientMap**, they are included in NiloToonURP's .unitypackage already. We will recommend starting from **FaceShadowGradientMap_StyleA** first because it can fit to almost all art style of faces since it doesn't contain any nose shadow info)



In the **Face Shadow Gradient Map (SDF)** of your material, you can turn on **Debug?**, and edit **UV Scale(XY) Pos(ZW)** until the debug looks like this. Each character may have a different correct **UV Scale(XY) Pos(ZW)** value, depending on the face UV layout of that character.

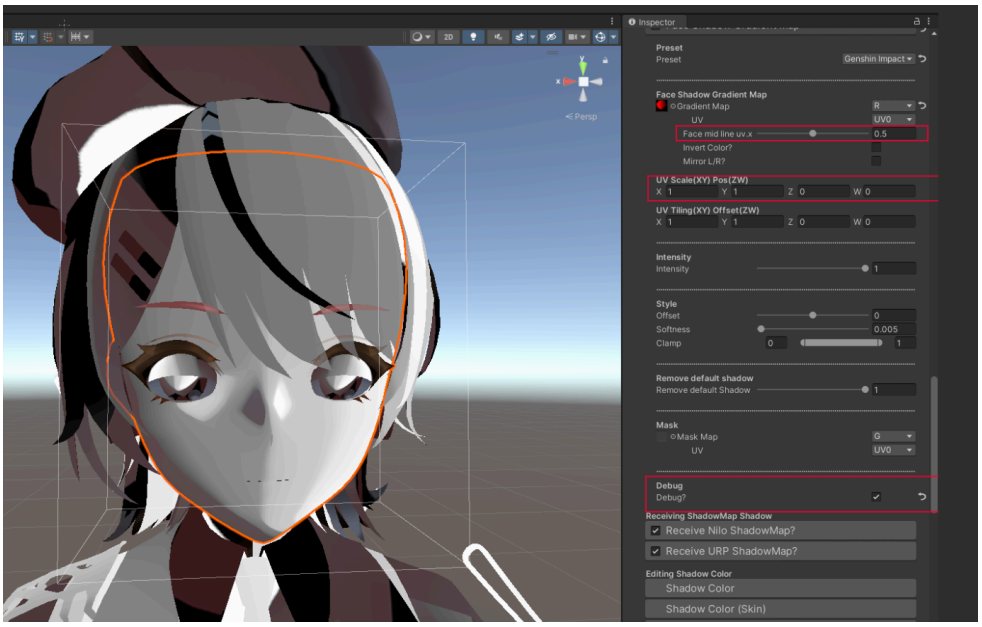


Also, make sure the NiloToon script attached to the character's root (NiloToonPerCharacterRenderController) 's **Head Bone Transform, Face Forward Direction, Face Up Direction** are set up correctly,
where the **green arrow** is pointing **up**, and the **blue arrow** is pointing to the face's **forward** when you turn on **Gizmo** in the scene window.



When your face uv didn't match the Gradient Map in shape, you will need to
1) turn on "Debug?"
2) adjust **UV Scale(XY) Pos(ZW)** to make the gradient map fit on your face uv
3) adjust "Face mid line uv.x" so it is a vertical line on your face uv's nose position

(e.g. when the nose is at the middle of face uv, "Face mid line uv.x" = 0.5)
(e.g. when the nose is at the left of face uv, "Face mid line uv.x" = 0~0.5)
(e.g. when the nose is at the right of face uv, "Face mid line uv.x" = 0.5~1)



Model requirement

Face SDF feature requires the model to have a **complete face uv** (**not half mirrored**).

- If the UV0 is suitable for the gradient map, that's good, you can use it directly
- If the UV0 is not suitable for the gradient map, it will require you to save the complete face uv in any of the UV1~UV3 channels, where UV0 is the UV for the BaseMap.

Map requirement

The SDF map should be:

- **Texture Compression** = **None**
- **sRGB** = **Off** (it is linear data map, not color)

Not working for eye/mouth

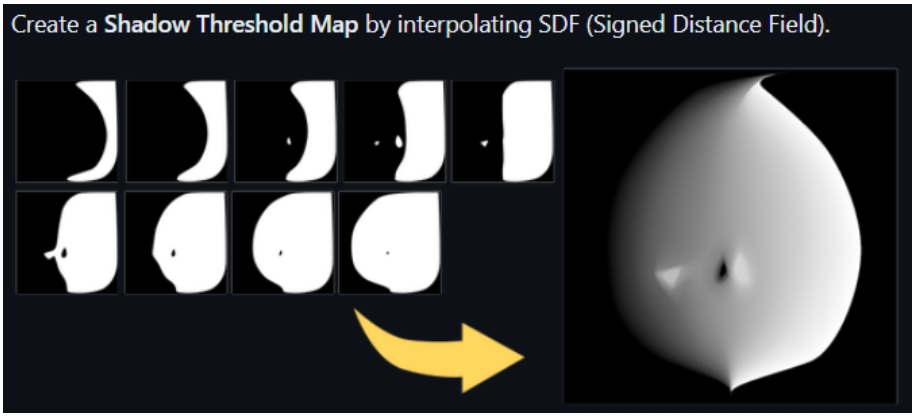
When this feature works for the face but looks wrong on the eye or mouth, you can consider:
A) use a mask map to mask face SDF feature, so it will NOT affect the eye or mouth area
or
B) provide a custom SDF gradient map for that character to support the shadowing gradient of the eye or mouth

External Tool/Ref

Here are some related websites showing more information about this feature, it may hint you how to create **custom Gradient Map** that fit your model's uv for this feature:

(GitHub - akasaki1211/sdf_shadow_threshold_map: Create a Shadow Threshold Map by interpolating SDF)
(https://x.com/akasaki1211/status/1806322840775585906?ref_src=twsrc%5Etfw%7Ctwcamp%5Etweetembed%7Ctwterm%5E1806322840775585906%7Ctwgr%5E9e666baec3282f51

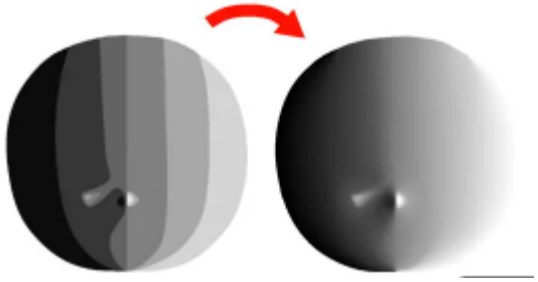
1cd4ca39a1c06c1a87ec09d7%7Ctwcon%5Es1_&ref_url=https%3A%2F%2Fwww.notion.so%2Fnilocat%2FCel-Note-c5963395359e4a0ea3ca66e266480d12)



(二次元角色卡通渲染—面部篇)
(GDC Vault - 3D Toon Rendering in 'Hi-Fi RUSH')

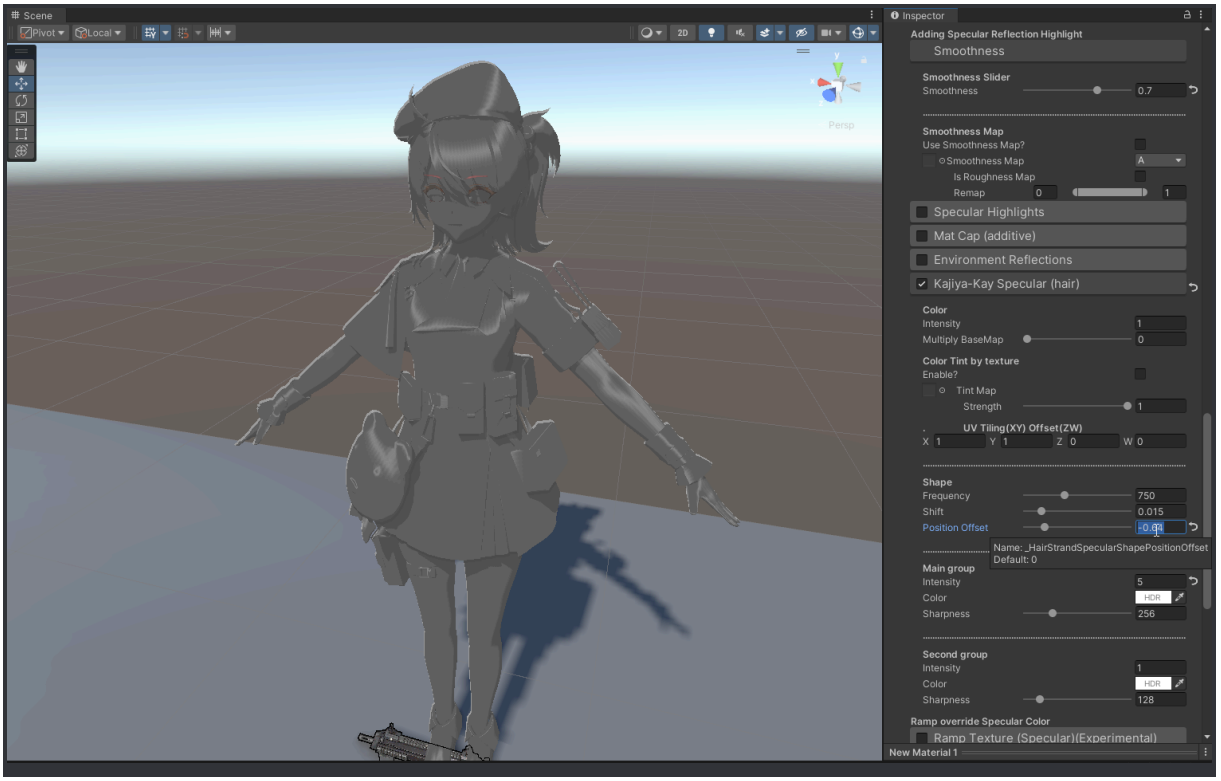
(原神模型脸部阴影渲染的实现方法#0)
(原神模型脸部阴影渲染的实现方法#1)
(【Genshin Shader Test】MMD/Genshin Impact :// Barbara's Solo☆MV (?)【CGSS】)
(神作面部阴影渲染还原)
(Signed Distance Field - 知乎)
(如何快速生成混合卡通光照图)
(卡通渲染之基于SDF生成面部阴影贴图的效果实现（URP）- 知乎)

(【【教程】使用csp等高线填充工具制作三渲二面部阴影贴图】)



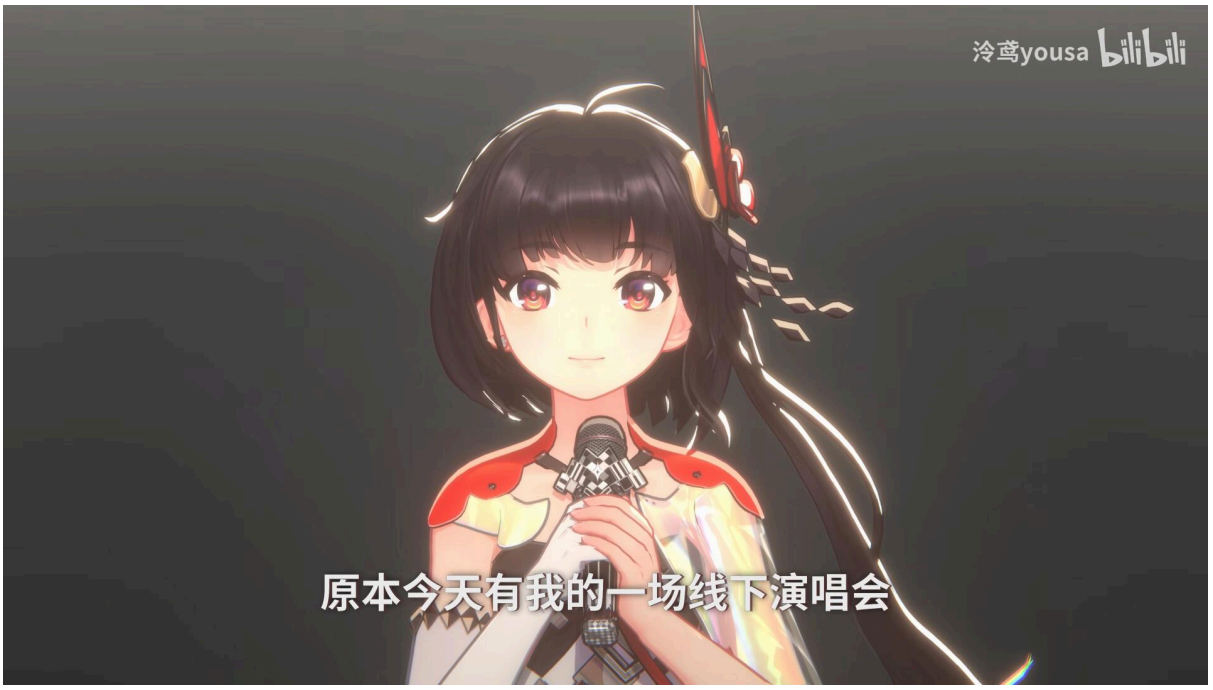
(卡通渲染流程【番外-使用SD批量生成SDF贴图】-简介有教程和工程文件！)
(https://github.com/ipud2/Unity-Basic-Shader/blob/master/YuanShen_Face.shader)
(https://www.bilibili.com/video/BV1sf4y1e7Kd?share_source=copy_web)

Kajiya-Kay Specular (hair)

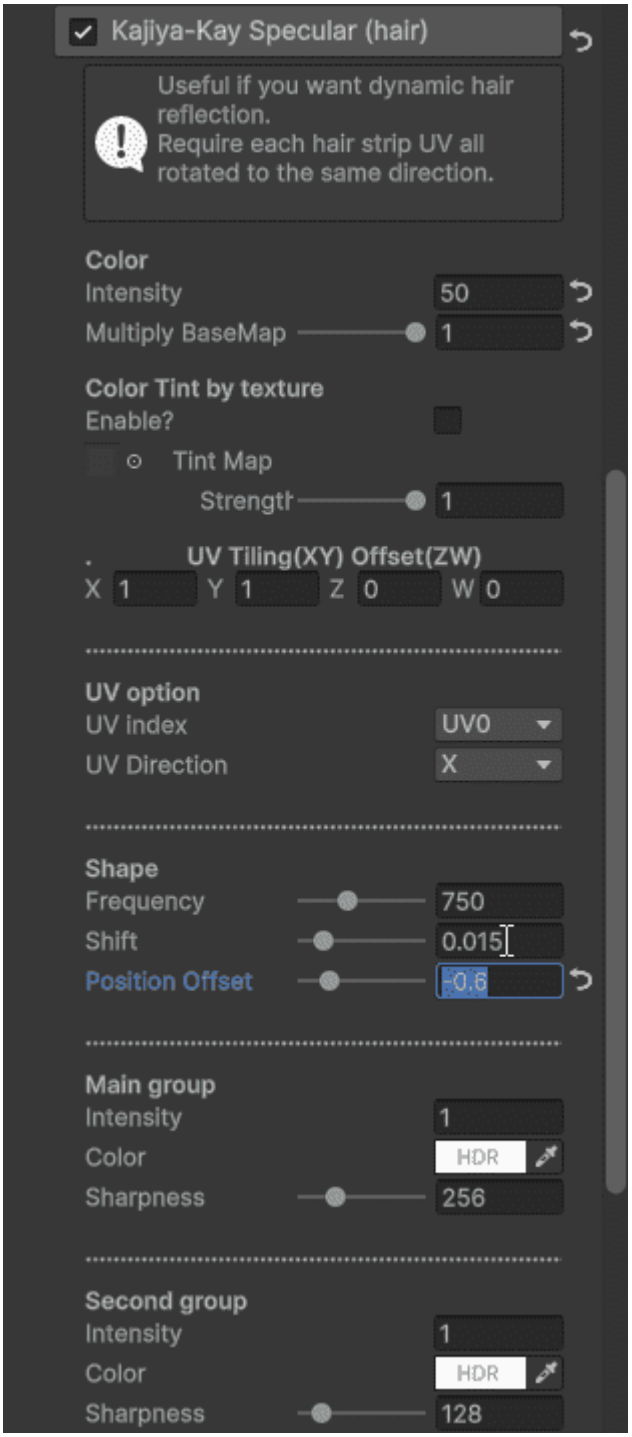


It is a very rarely used feature, and it is only for producing a dynamic hair "**angel ring**".

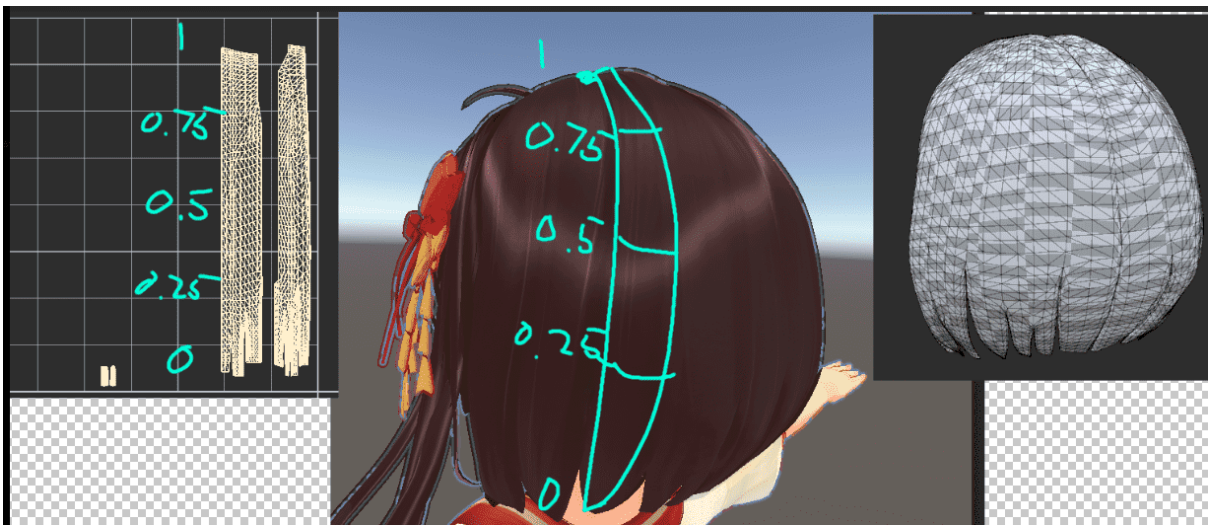
Here is an use-case that use this feature for a **dynamic hair angel ring**:
[泠鸢yousa + NiloToon]



To produce a better specular, **Position Offset** needs to be manually adjusted for each hair material.



We can't recommend using **Kajiya-Kay Specular (hair)**, because this function requires all uv island in **UV0**(first UV set/channel) line up vertically (see the image below: all hair has their root vertices at the top of the uv, and their tip at the bottom of the uv), most of the model's **UV0** won't have this special format uv, that why we can't recommend using it. But if your model already has this kind of vertically lined up uv in **UV0**, you can turn it on and have a look at the result, see if it is useful or not.

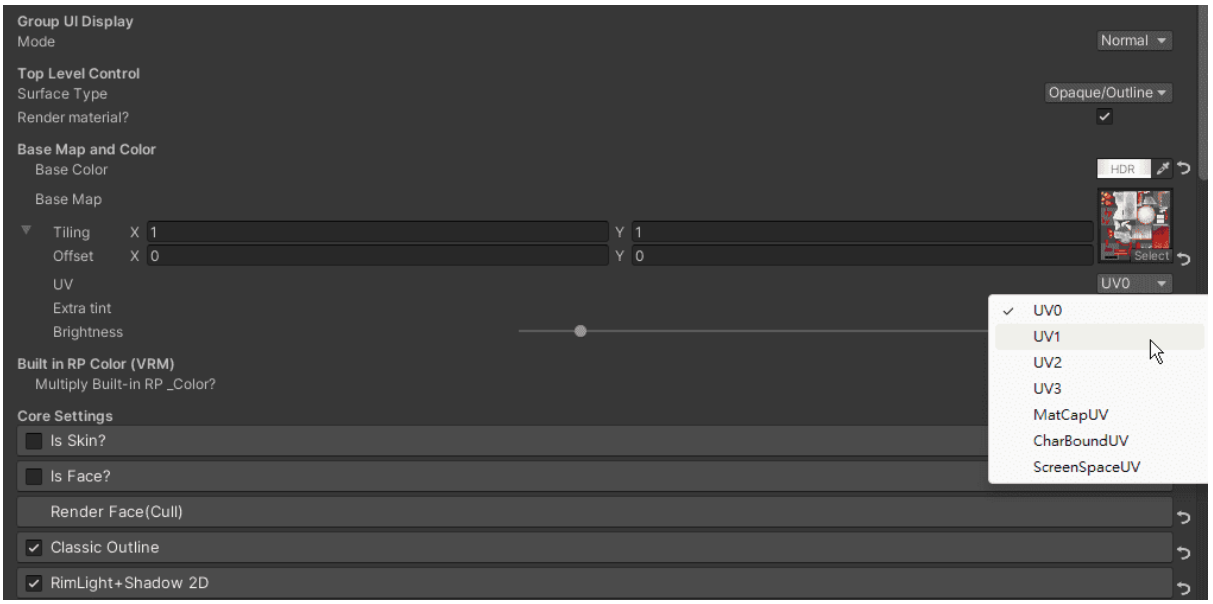


Only when **uv0** is set up like this, the auto-generated tangent using uv0 as input will generate the correct data for this function to calculate the correct hair specular.

If your model has a regular uv only, one trick to use this feature is to provide an additional uv in your model:

- UV0: the **special** vertically line up uv
- UV1: regular uv for basemap color

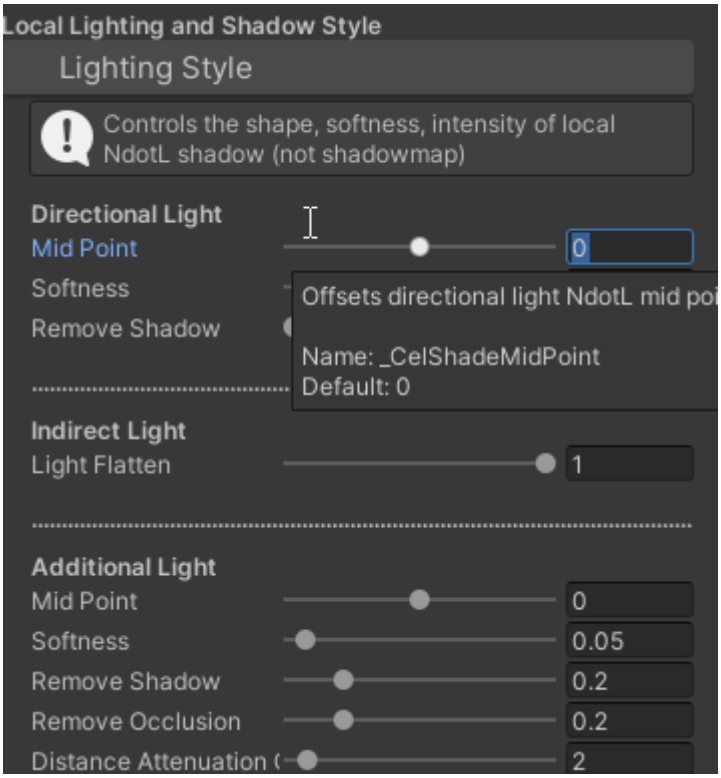
When a model has this set of data, the auto-generated tangent using UV0 will be correct for this function, and you can still sample the BaseMap color using UV1, see the setting below:

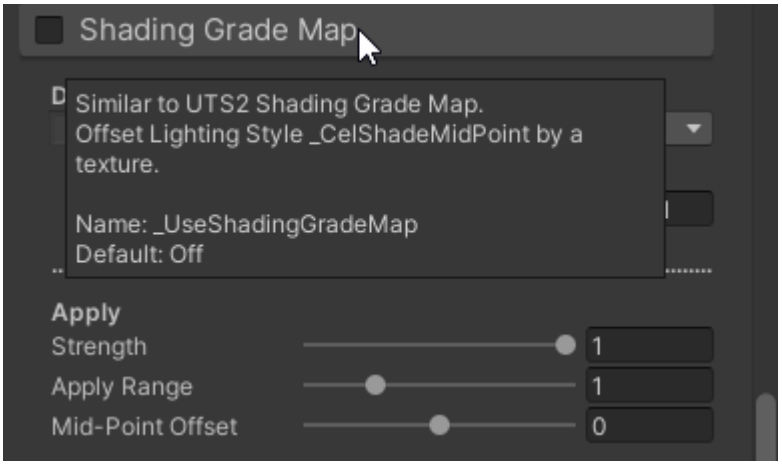


Shading Grade Map

It is a map similar to **occlusion map**, but not the same, the only purpose is to **offset** Lighting Style's "**Mid Point**" by the "Shading grade map".

- When shading grade map is gray(linear gray), nothing will happen
- when you paint an area of shading grade map towards black, it will offset the "Mid Point" so shadow will be harder to appear
- when you paint an area of shading grade map towards white, it will offset the "Mid Point" so shadow will be easier to appear, similar to an occlusion effect





Below is an example of what the Shading Grade Map may look like, please note that the default settings in NiloToon expect an inverted color map, you can freely control the **invert toggle** inside the material.

UnityのTimelineを活用したライブ制作のこだわり～「学園アイドルマスター」制作事例～

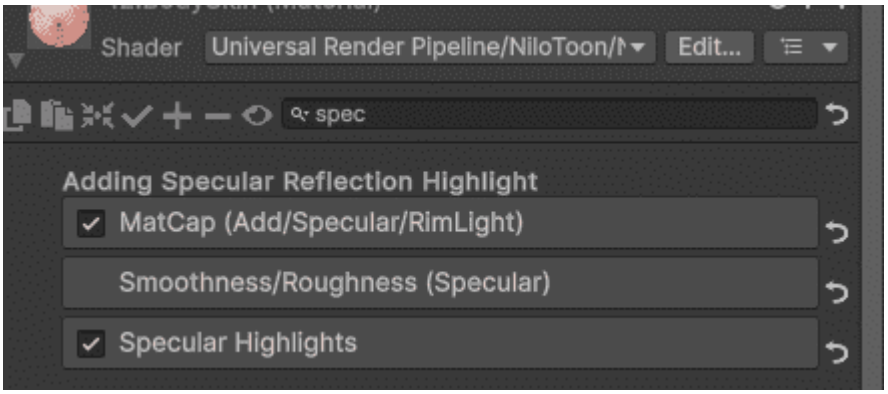
Ref1: 『学マス』ひとつのライブを制作するのに半年以上。アイドルたちの“らしさ”から振り付けを考え、汗の表現は5種類を使い分け。UnityのTimelineで実現する妥協なき精神【CEDEC2024】| ゲーム・エンタメ最新情報のファミ通.com

Ref2: 神は細部に宿る――「学園アイドルマスター」3Dキャラ・背景・ライブ演出はいかにこだわり抜かれ，生まれたのか？［CEDEC 2024］

Specular & Smoothness

The following groups:

- **Smoothness/Roughness (Specular)**
- **Specular highlights**



are usually **used together as a pair**, great for:

- realistic cloth/metal PBR reflection
- realistic skin reflection with sweat / oil, usually used for sexy models, or ero games (a.k.a H-games)

These functions will be improved a lot by providing the following maps, common from PBR materials:

- normal map
- detail map (normal)
- smoothness / roughness map

MatCap (Add/Specular/RimLight) is also often used together, it is the regular matcap specular option that is not PBR nor realistic, it doesn't care about the light direction or light color, but for some styles a stable specular from a matcap texture is helpful to add details to specular.

Let's see some example usage of all these 3 groups for:

Realistic clothing:

- [NiloToon Unity URP shader] Girls Frontline 2 Exilium (2023 CBT3) - MP7 Che…
- [NiloToon Unity URP shader] Girls Frontline 2 Exilium - Sharkry rendering
- [NiloToonURP] Girls Frontline 2 Exilium (2023 CBT3) - Vepley rendering
- ROSÉ & Bruno Mars - APT. || Cover by Bukuki & Mare Flos
- NiloToon Specular & Matcap(Additive) example

Realistic skin:

- NiloToon Realistic Skin specular example

Metal:

- 【粛聖!! ロリ神レクイエム☆】パトラちゃんに罵倒してもらうだけ #vtuber #周防パトラ #shorts - YouTube

- 粛聖!! ロリ神レクイエム☆ / 周防パトラ(9さい＋81××さい)cover

Eye-glass / semi-transparent plastic:

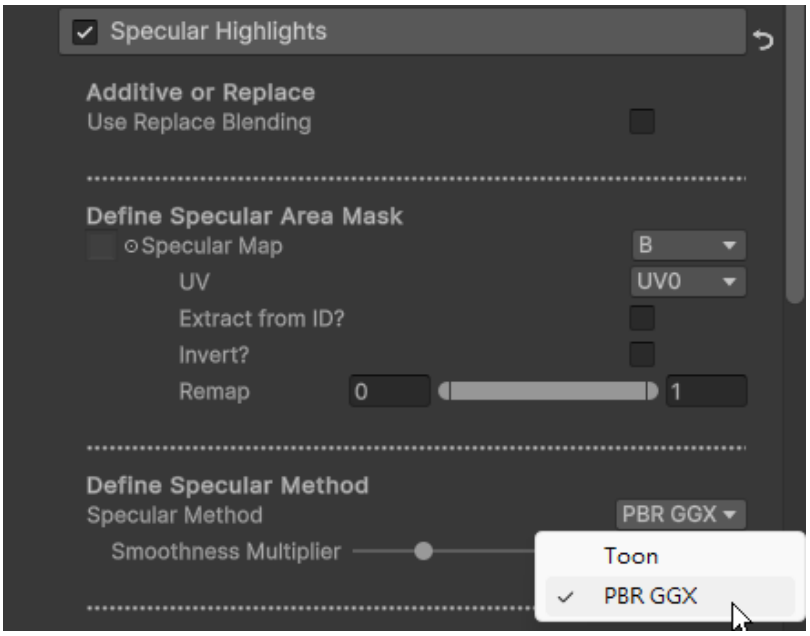
- [4K] 스텔라이브 아라하시 타비! 3D 데뷔!
- #胡桃のあ3Dライブ | ~星降る夜に願いを込めて~ 🌠

Subtle specular for plastic / metal / stocking:

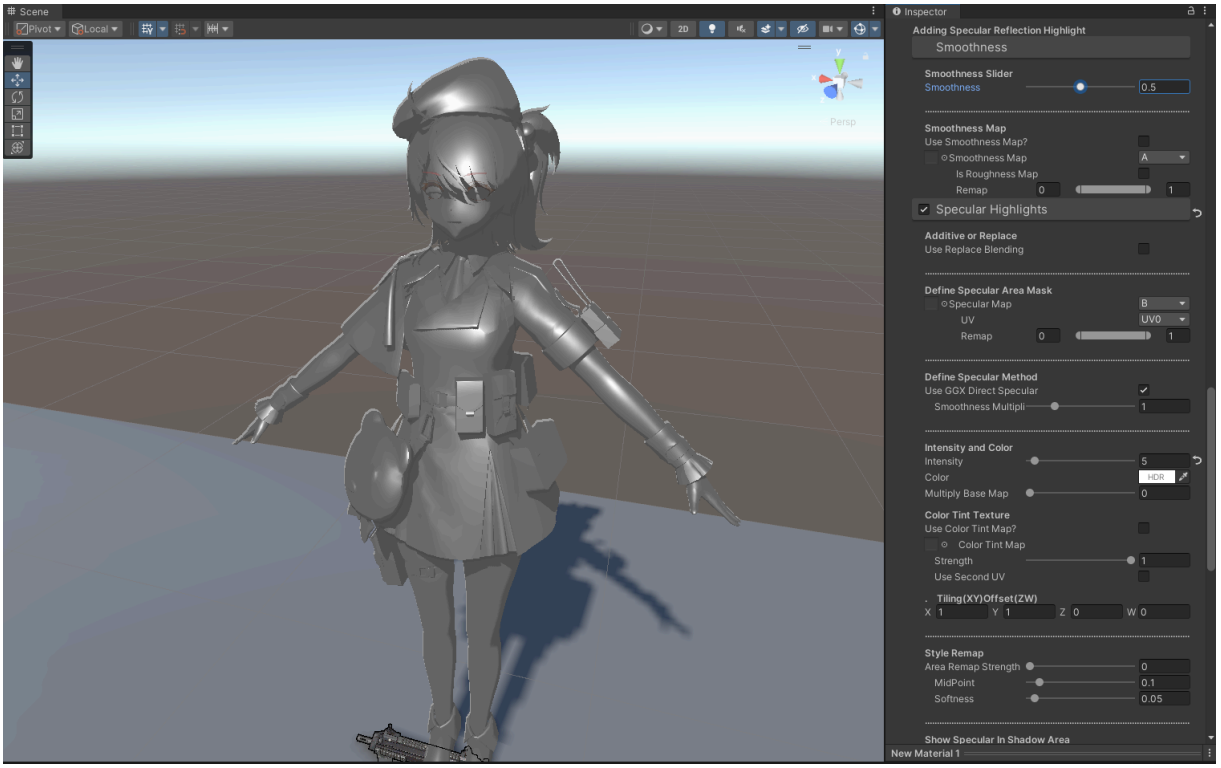
- Starseed (스타시드) - Character & Ultimate Skill Preview - Android on …

The default **specular method** is **PBR GGX**, it means the specular reflection will use the PBR specular algorithm, you will mainly control the shape of specular using the

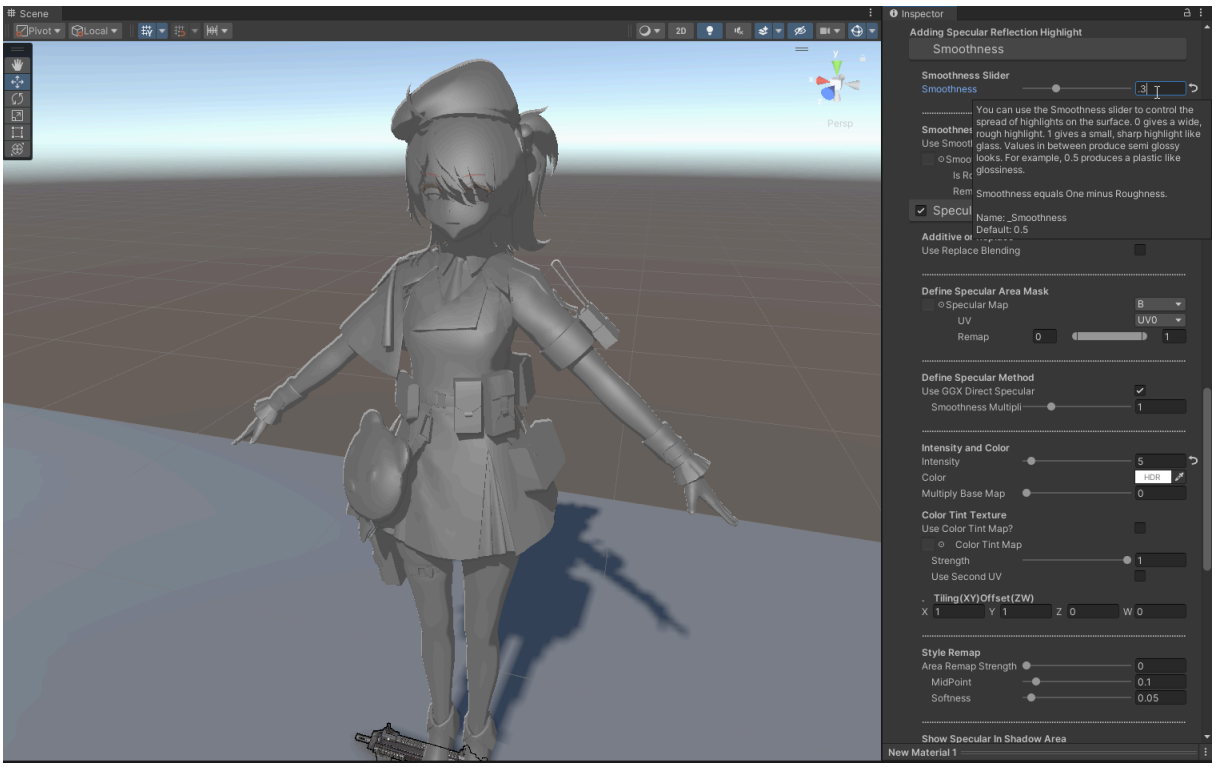
Smoothness/Roughness slider. PBR GGX is great for realistic reflection.



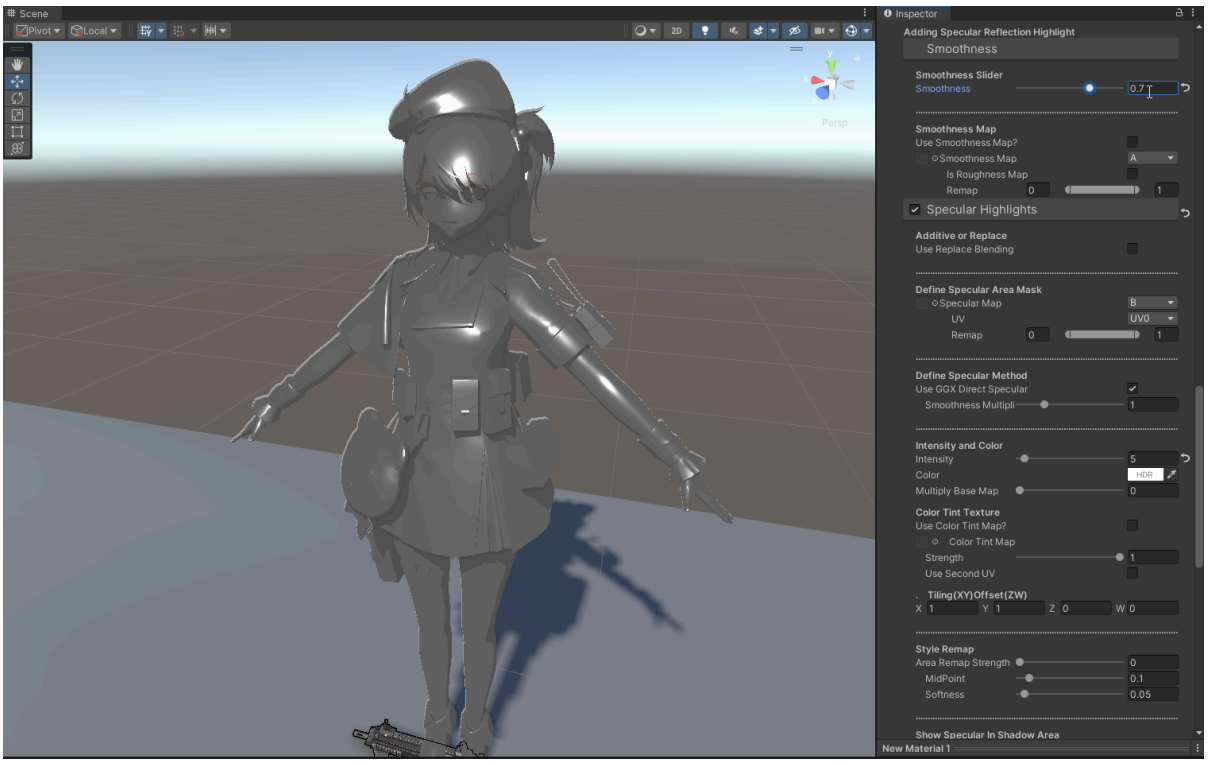
Below are examples of adjusting ‘Smoothness / Roughness” only:



(Image above: Smoothness = **0.5**)



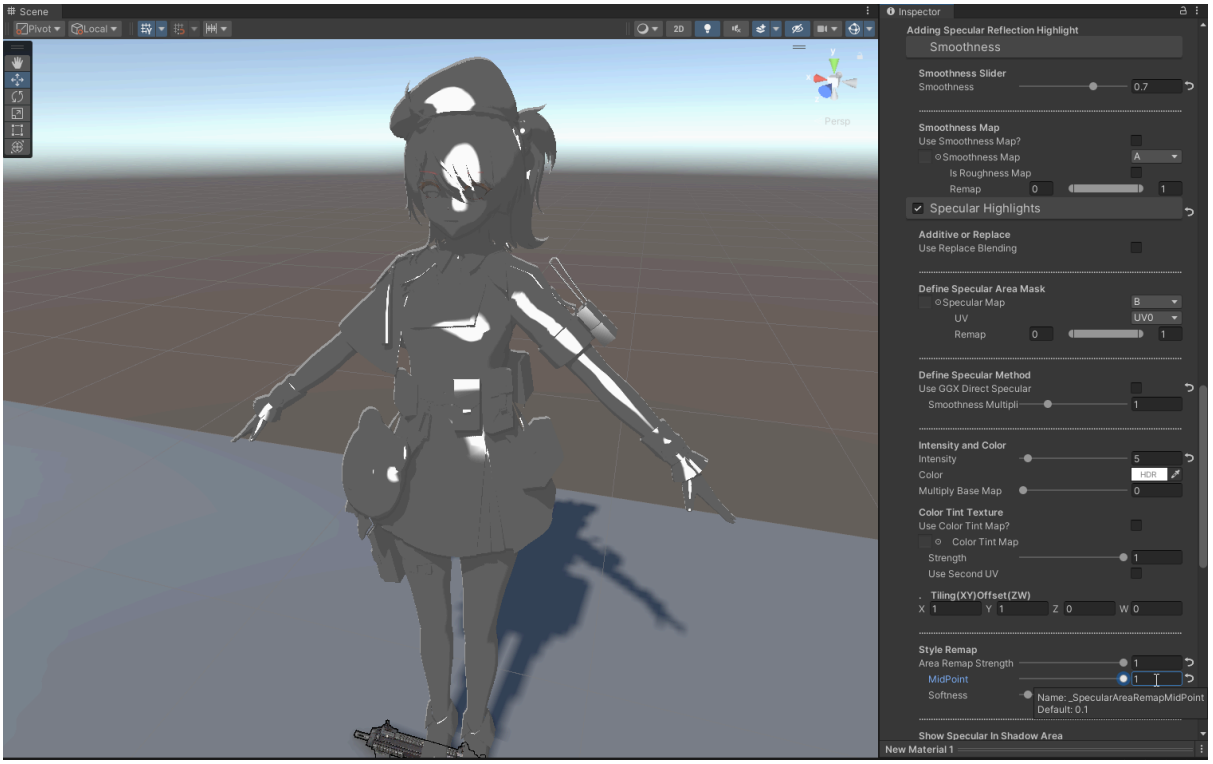
(Image above: Smoothness = **0.3**)

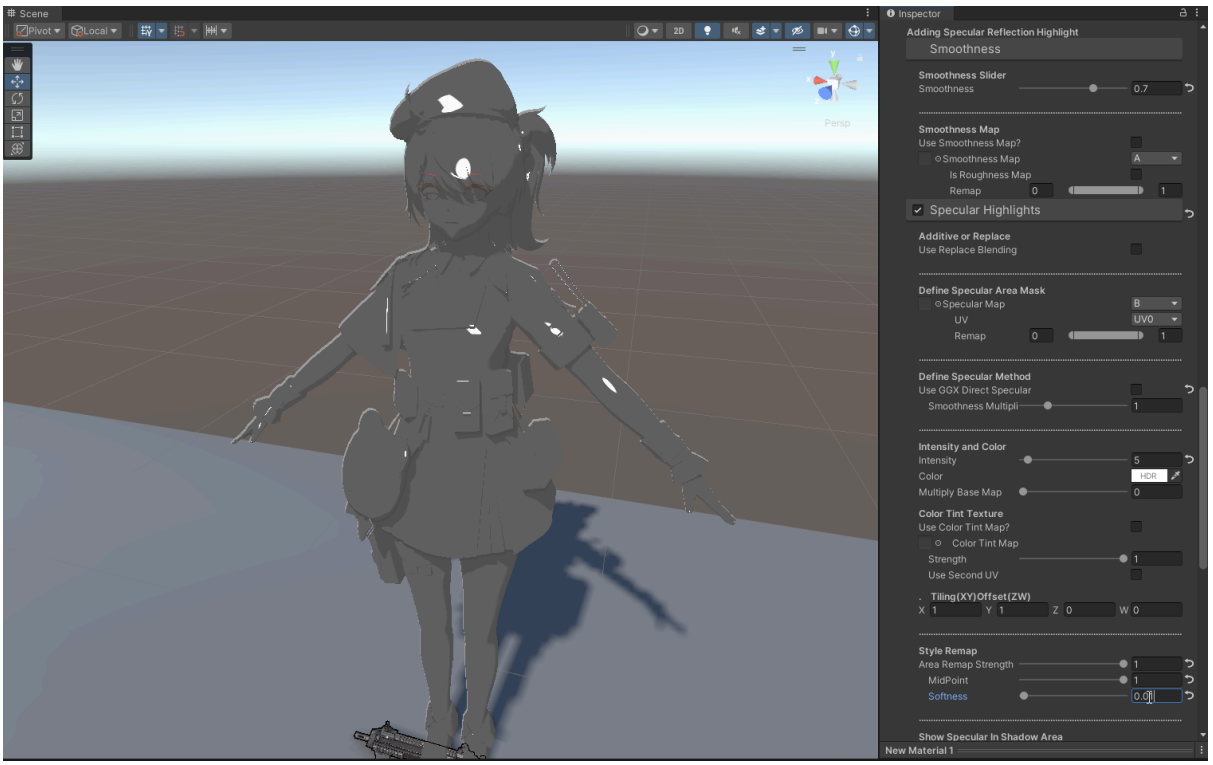


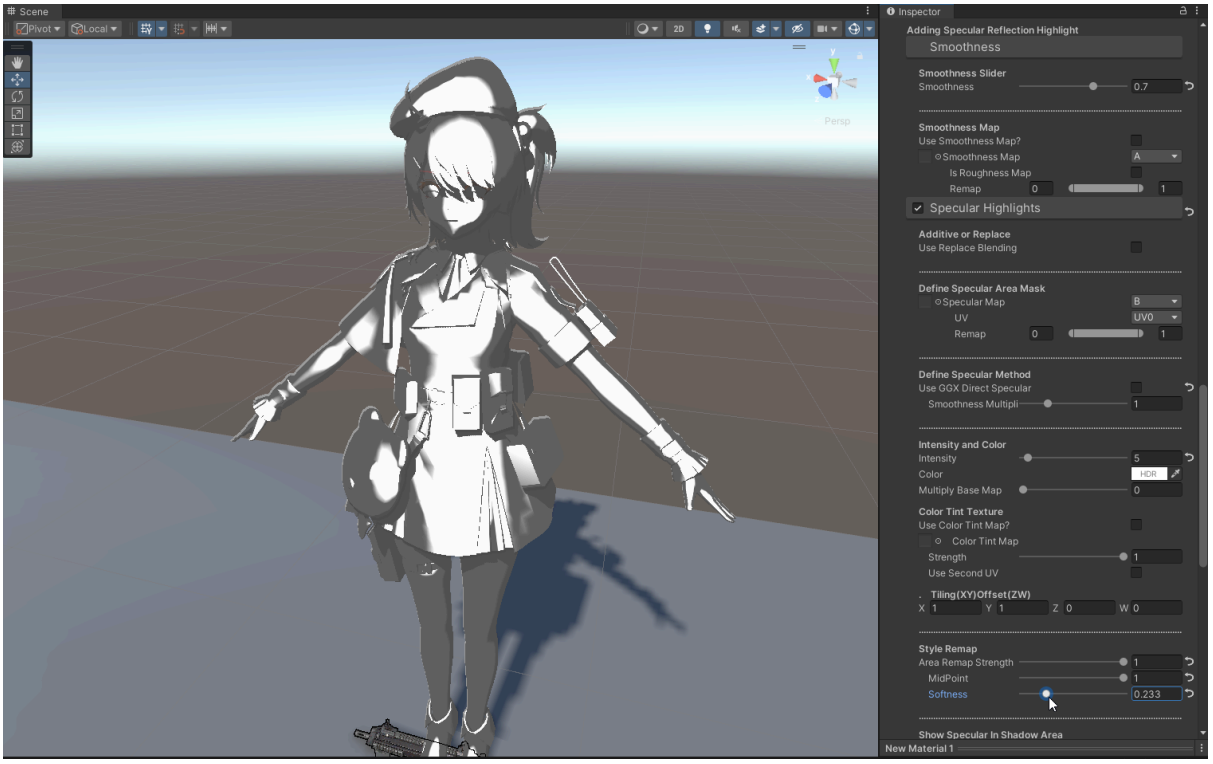
(Image above: Smoothness = **0.7**)

- when **specular method = Toon**, it means the reflection will use the classic NPR specular algorithm, not the PBR algorithm, so **smoothness is useless** and relies on your manual shape control

You will mainly control the shape of specular using the **style remap slider**, great for setting up hair specular with a reflection shape mask

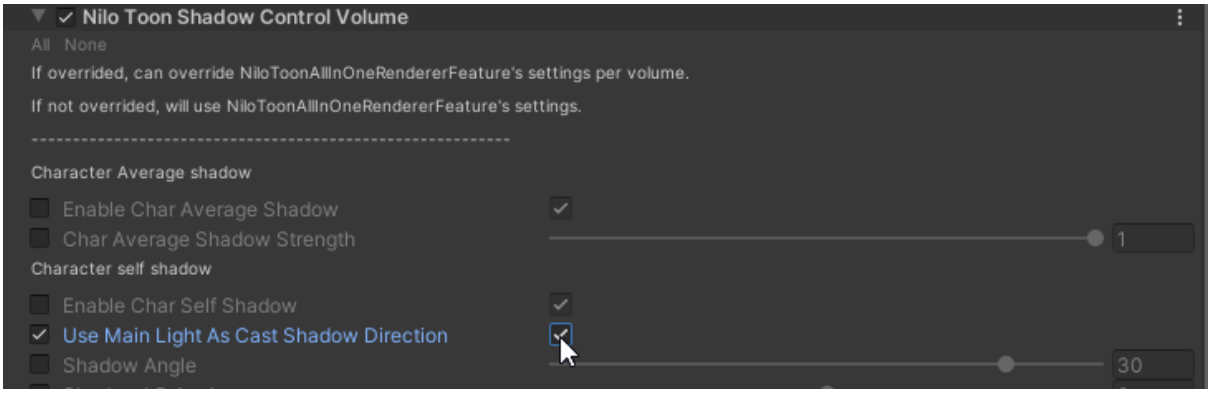



Scene
Pivot
Local
2D
Persp
Inspector
Adding Specular Reflection Highlight
Smoothness
Smoothness Slider
Smoothness
0.7
Smoothness Map
Use Smoothness Map?
Smoothness Map
A
Is Roughness Map
Remap
0
1
Specular Highlights
Additive or Replace
Use Replace Blending
Define Specular Area Mask
Specular Map
B
UV
UV0
Remap
0
1
Define Specular Method
Use GGX Direct Specular
Smoothness Multipli
1
Intensity and Color
Intensity
5
Color
HDR
Multiply Base Map
0
Color Tint Texture
Use Color Tint Map?
Color Tint Map
Strength
1
Use Second UV
Tiling(XY)Offset(ZW)
X 1
Y 1
Z 0
W 0
Style Remap
Area Remap Strength
1
MidPoint
1
Softness
0.0
Show Specular In Shadow Area
New Material 1


Scene
Pivot
Local
2D
Persp
Inspector
Adding Specular Reflection Highlight
Smoothness
Smoothness Slider
Smoothness
0.7
Smoothness Map
Use Smoothness Map?
Smoothness Map
A
Is Roughness Map
Remap
0
1
Specular Highlights
Additive or Replace
Use Replace Blending
Define Specular Area Mask
Specular Map
B
UV
UV0
Remap
0
1
Define Specular Method
Use GGX Direct Specular
Smoothness Multipli
1
Intensity and Color
Intensity
5
Color
HDR
Multiply Base Map
0
Color Tint Texture
Use Color Tint Map?
Color Tint Map
Strength
1
Use Second UV
Tiling(XY)Offset(ZW)
X 1
Y 1
Z 0
W 0
Style Remap
Area Remap Strength
1
MidPoint
1
Softness
0.233
Show Specular In Shadow Area
New Material 1

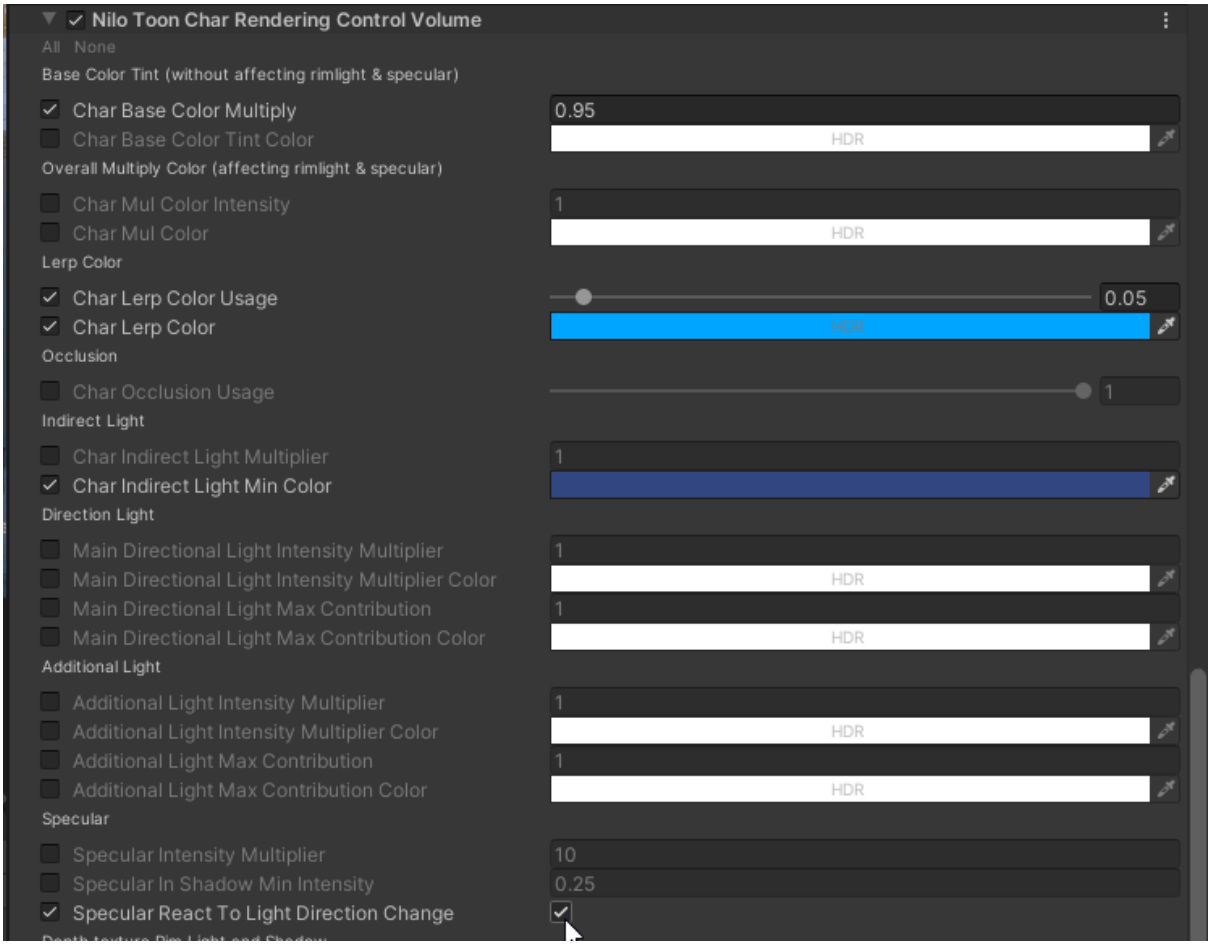
Self shadow map follow cam

NiloToonAllInOneRendererFeature and **NiloToonShadowControlVolume** has a toggle useMainLightAsCastShadowDirection, you can enable it if you want NiloToon's self shadow system use scene's MainLight direction to cast shadow(same shadow casting direction as regular URP main light shadow, so shadow result will NOT be affected by camera rotation/movement anymore)



Specular reacts to light dir

NiloToonCharRenderingControlVolume has a toggle specularReactToLightDirectionChange, you can enable it if you want NiloToon character shader's specular results react to the main light's direction change, similar to any regular specular reflection.



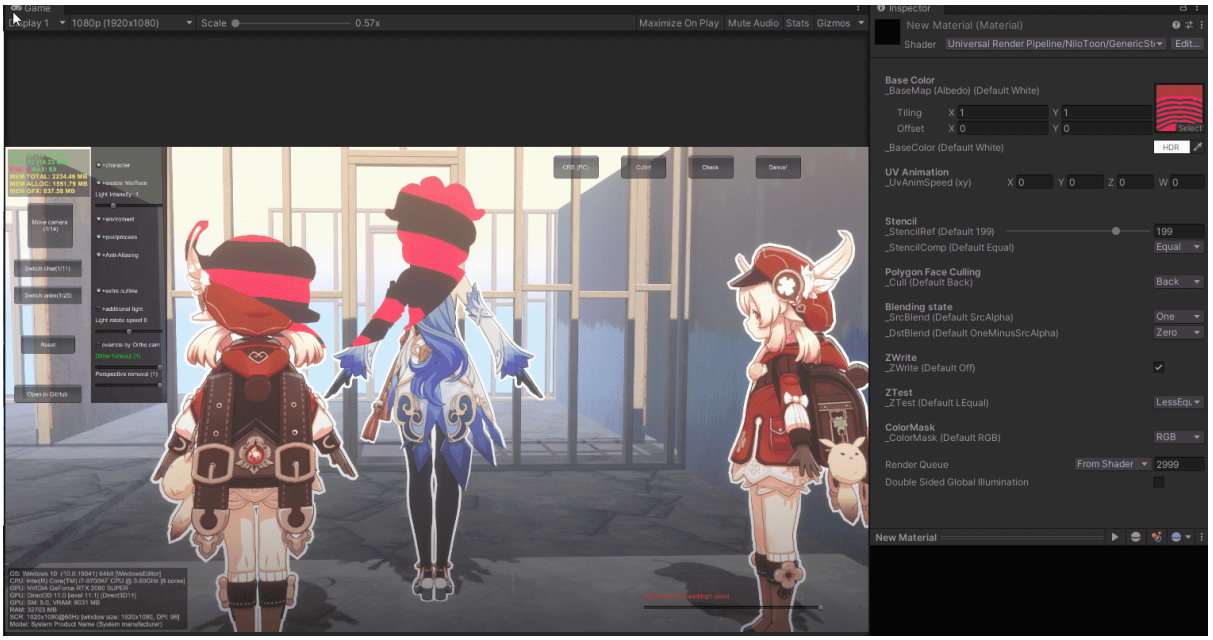
On the other hand, override it to false, will make the specular not react to light direction, which will produce a matcap like specular result

Stencil shader on characters

Use GenericStencilUnlit.shader as an example shader, useful if you want to apply stencil effects that use drawn character pixels as a stencil mask, or just need an example shader for reference.

*the leftmost bit of the stencil value will be set to 1 by NiloToon's pass to indicate it is a character pixel when the "extra thick outline" is enabled.
For example:

- material's stencil ref is 3 (00000011)
- the actual stencil ref after the special stencil pass is 10000011



Prevent write to RT's alpha

you can use the following setting in the material, set to a color mask that without A (e.g., RGB)

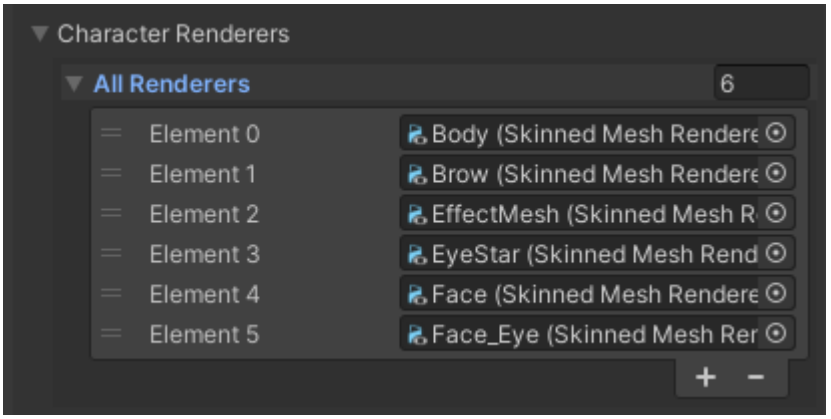
Temp .fbx on import

These are temporary .fbx files "(you can ignore or delete it safely)xxx.fbx" generated by NiloToonEditor_AssetLabelAssetPostProcessor.cs, you can always delete these .fbx safely anytime you see them, usually NiloToon will clean up them right after any import, so you will not see these temp files normally.

Switch char renderer in runtime

For example, switching hair or clothes.

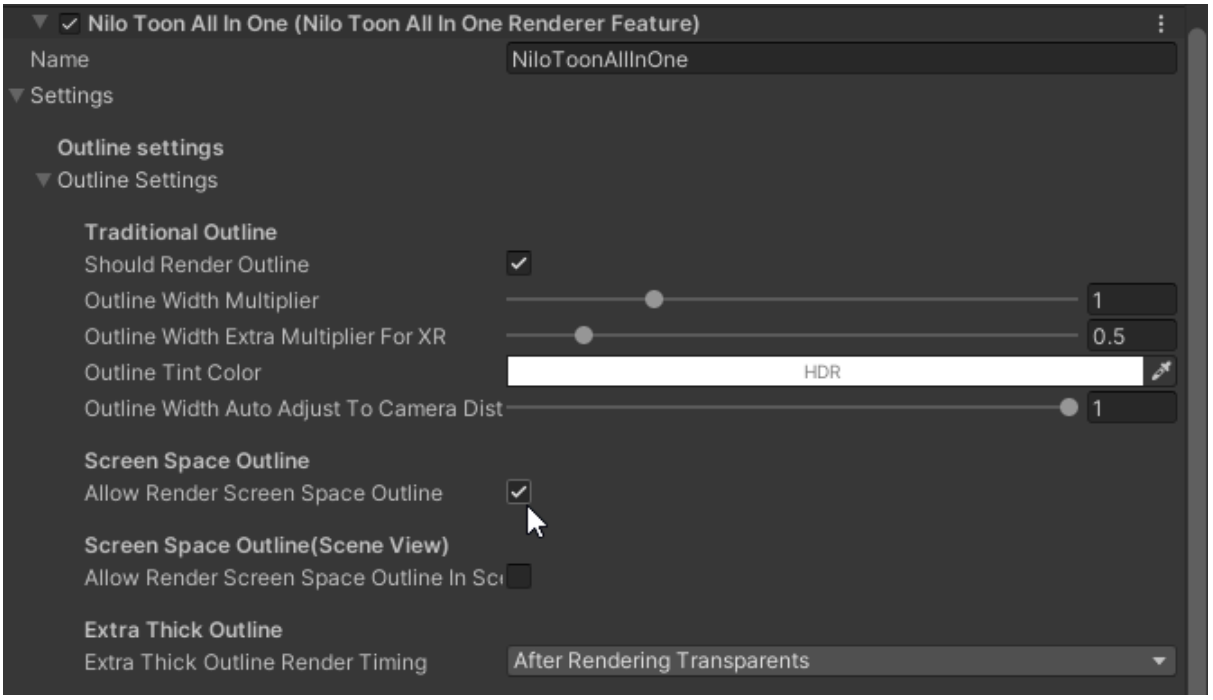
If any new renderer are not yet included in NiloToonPerCharacterRenderController's AllRenderers list,



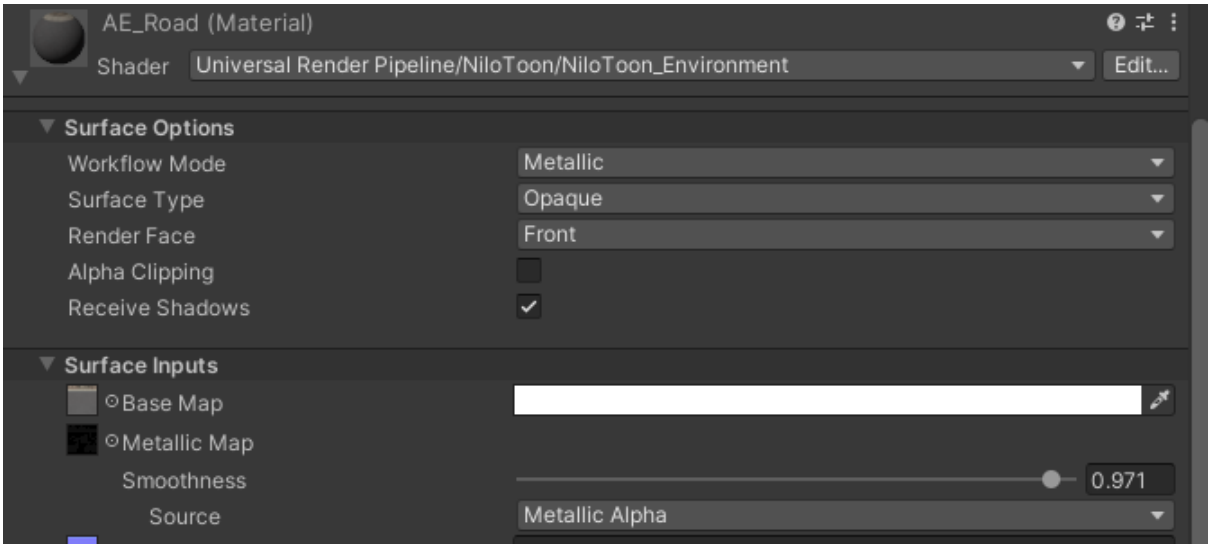
After switching renderers, set NiloToonPerCharacterRenderController's allRenderers count to 0, this will trigger the script to re-find all latest renderers under this script.

Use NiloToon's envi shader

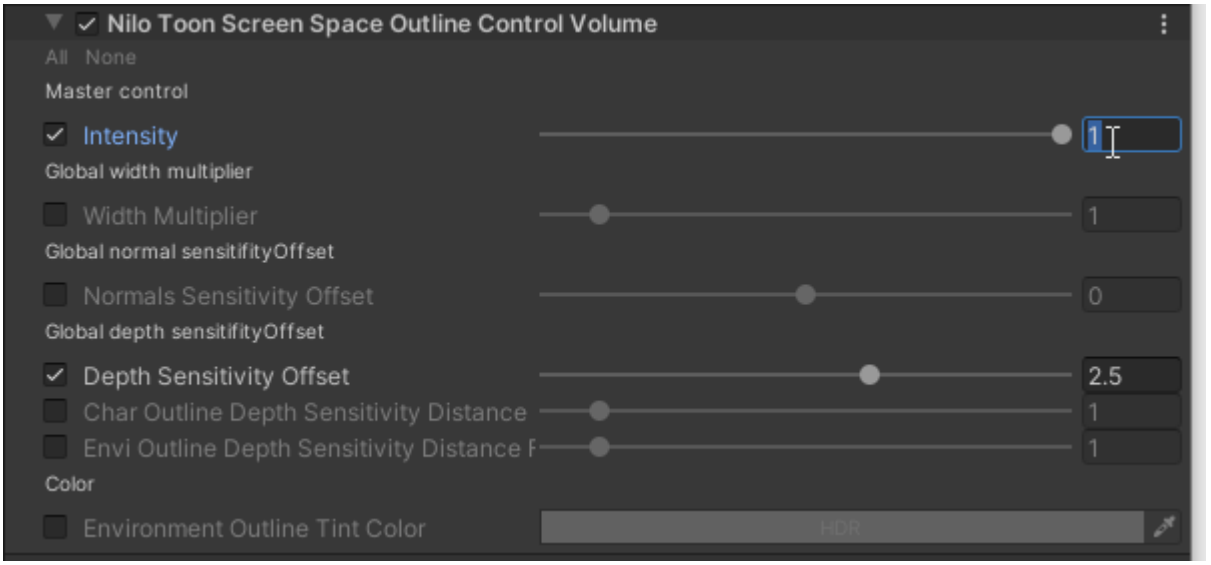
1)Click on your **NiloToonAllInOne renderer feature**, enable AllowRenderScreenSpaceOutline



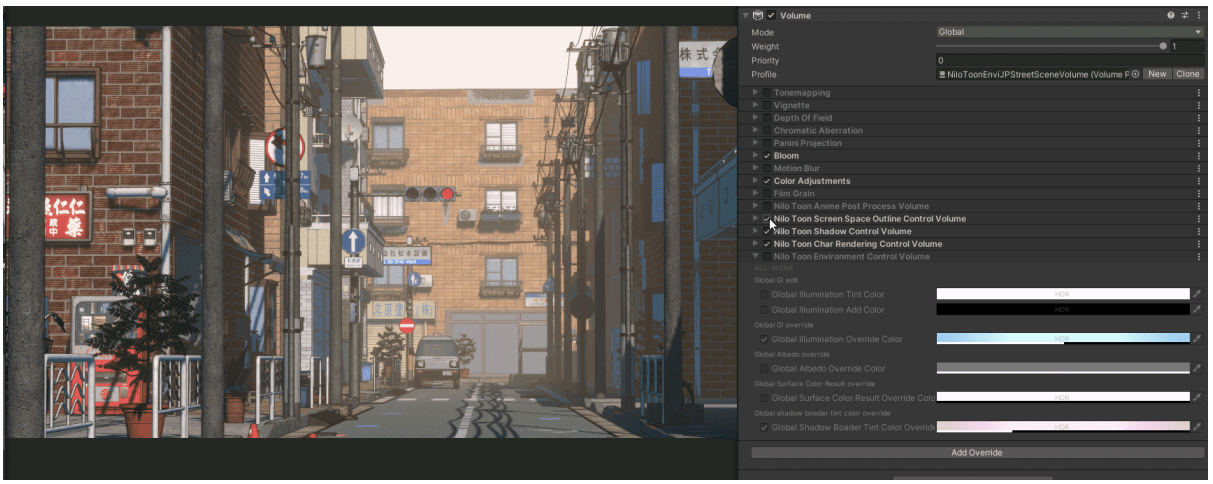
2)Switch any environment material's shader from URP's Lit to NiloToon_Environment



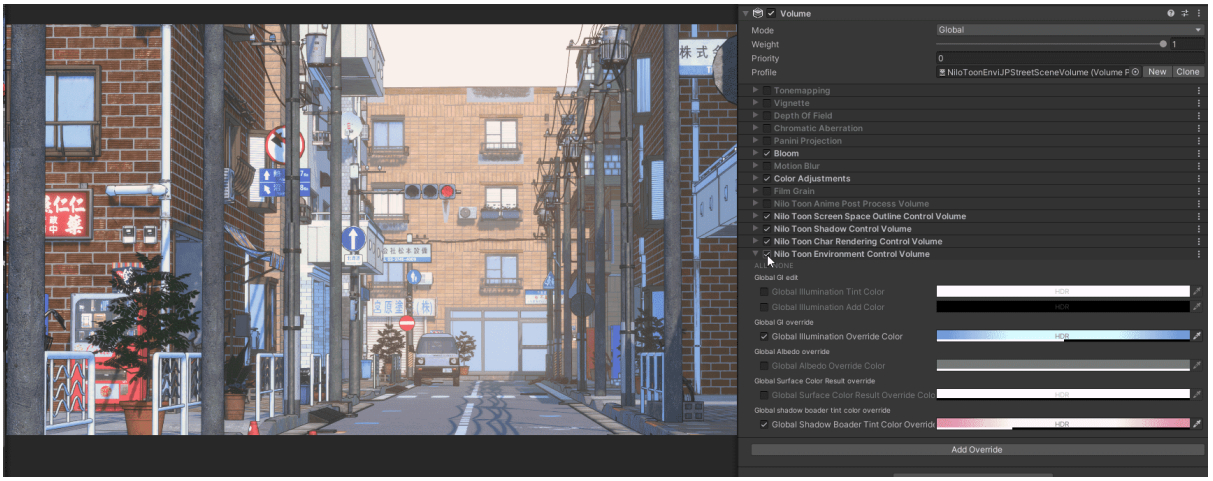
3)Add NiloToonScreenSpaceOutlineControlVolume to a volume component in scene, enable intensity and drag it to 1

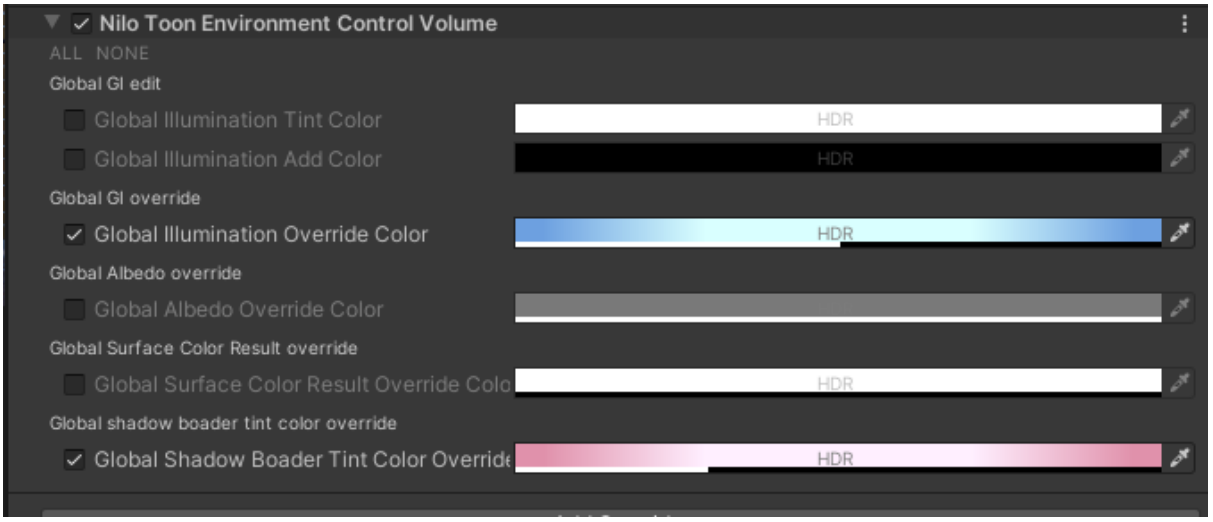


4)You should now see you environment renders the same as before, but having outline on them

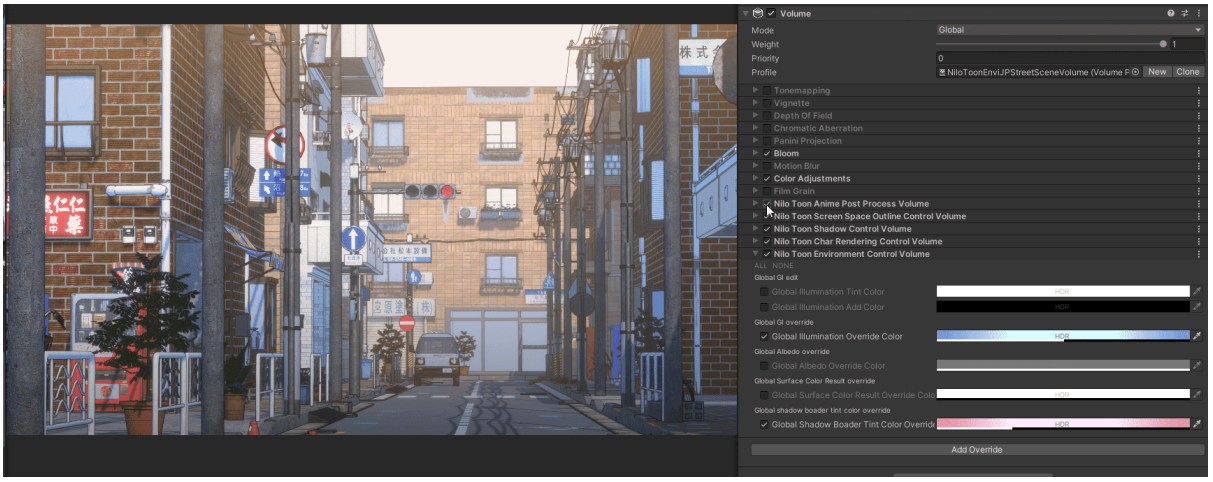


5)(optional) Add NiloToonEnvironmentControlVolume to a volume component in scene, edit the color rgb and alpha settings to control environment color result





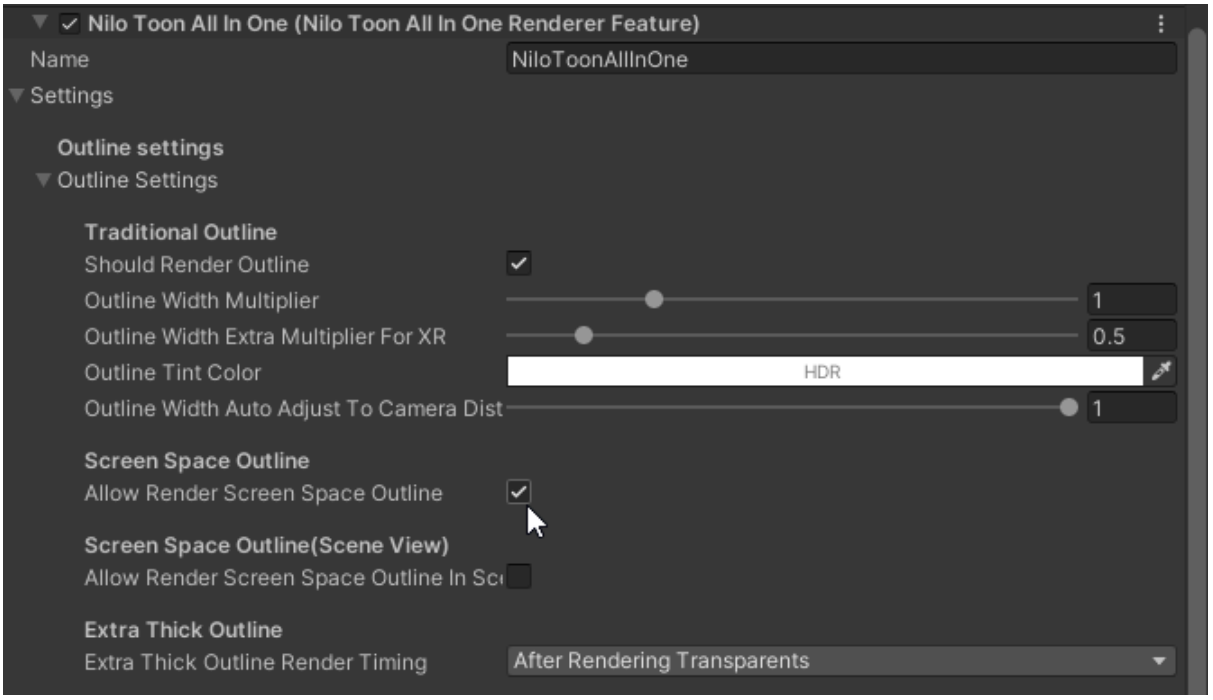
6)(optional) Add other volume to improve the result, for example, adding NiloToonAnimePostProcessVolume




Nilo Toon Anime Post Process Volume
ALL NONE
Master control
Intensity
1
Rotation
0
Draw Timing
Draw Before Post Process
Top Light Draw Setting
Top Light Effect Draw Height
0.5
Top Light Effect Intensity
0.5
Top Light Desaturate
0
Top Light Tint Color
HDR

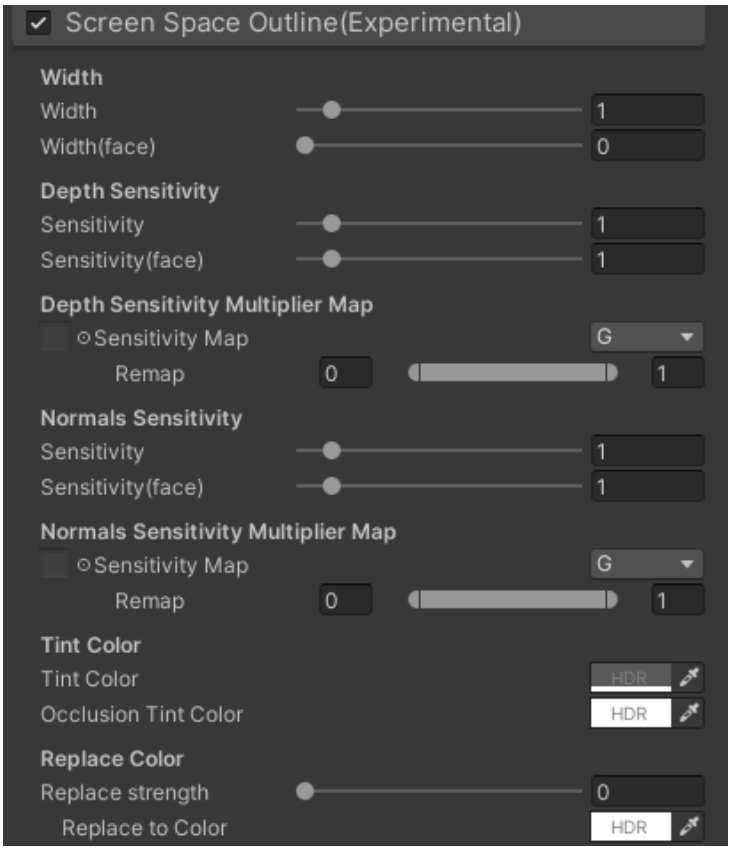
Enable screen space outline

1)Click on your active NiloToonAllInOne renderer feature, enable AllowRenderScreenSpaceOutline

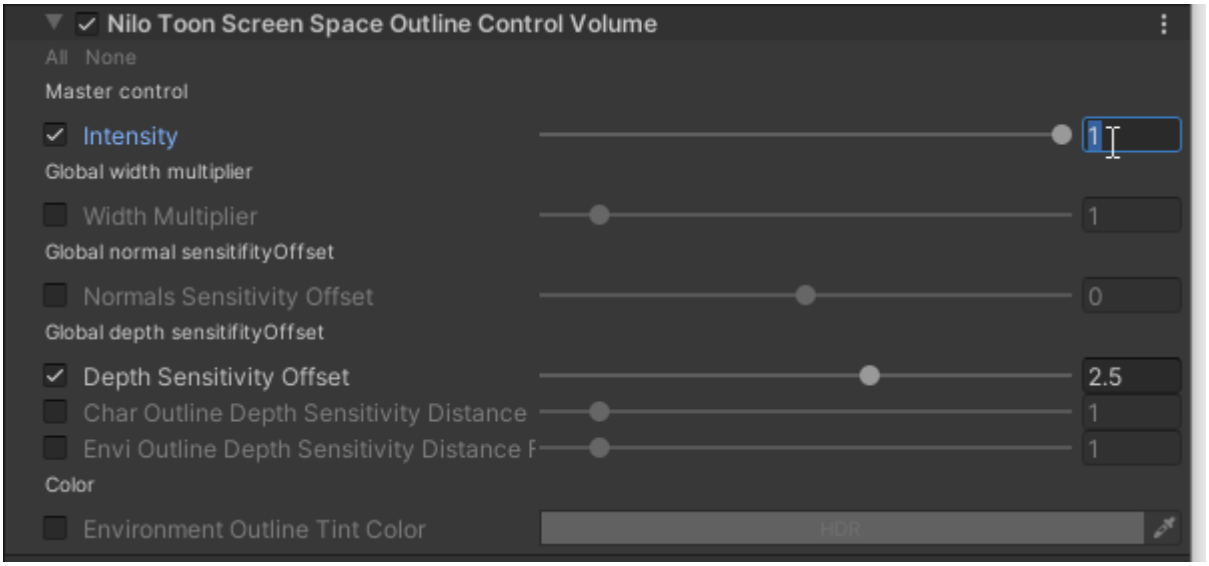


2)In your character material, enable the Screen Space Outline section.

For environment material, you can skip this step.



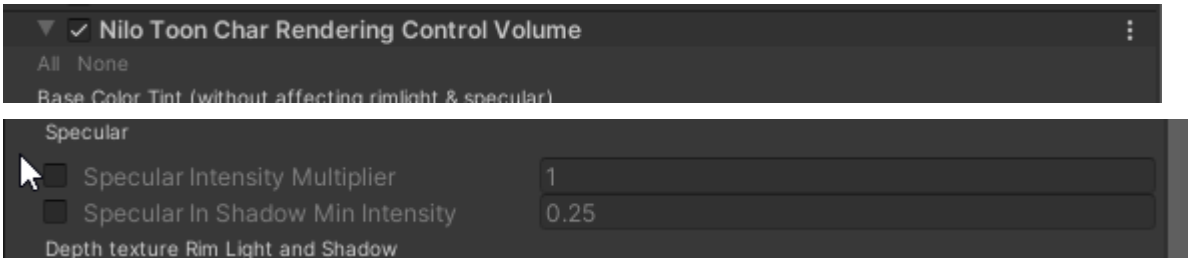
3)Add NiloToonScreenSpaceOutlineControlVolume to a volume component in scene, enable intensity and drag it to 1



4)You should now see the screen space outline affecting NiloToon's character and environment material in the Game view window.

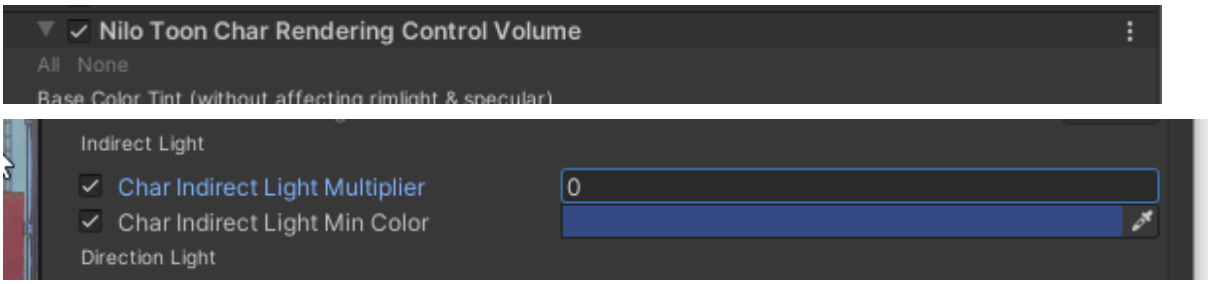
Specular intensity in shadow

In NiloToonCharRenderingControlVolume, control specularInShadowMinIntensity



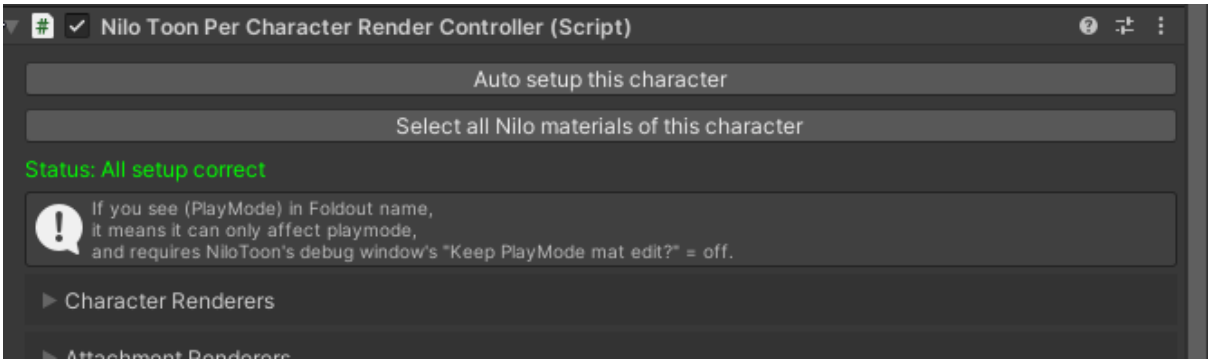
Make char ignore light probe

In NiloToonCharRenderingControlVolume, make:
- **charIndirectLightMultiplier** = 0
- **charIndirectLightMinColor** = a shadow color that looks good in your current scene

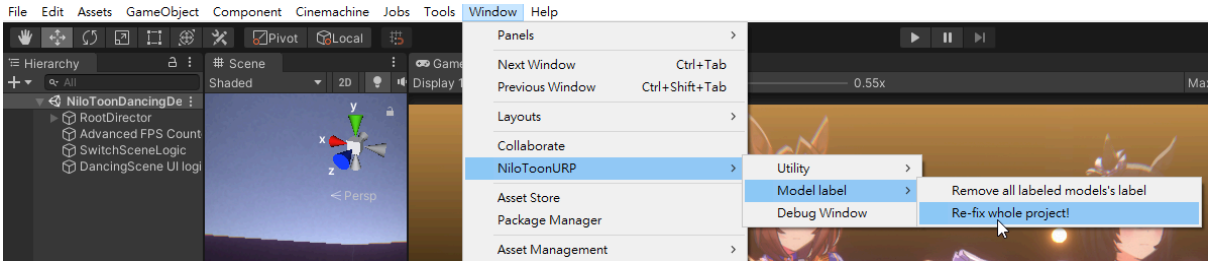


Outline lost on reimport or switching platform

You can try click on that character's NiloToonPerCharacterRenderController script to trigger a rebake,

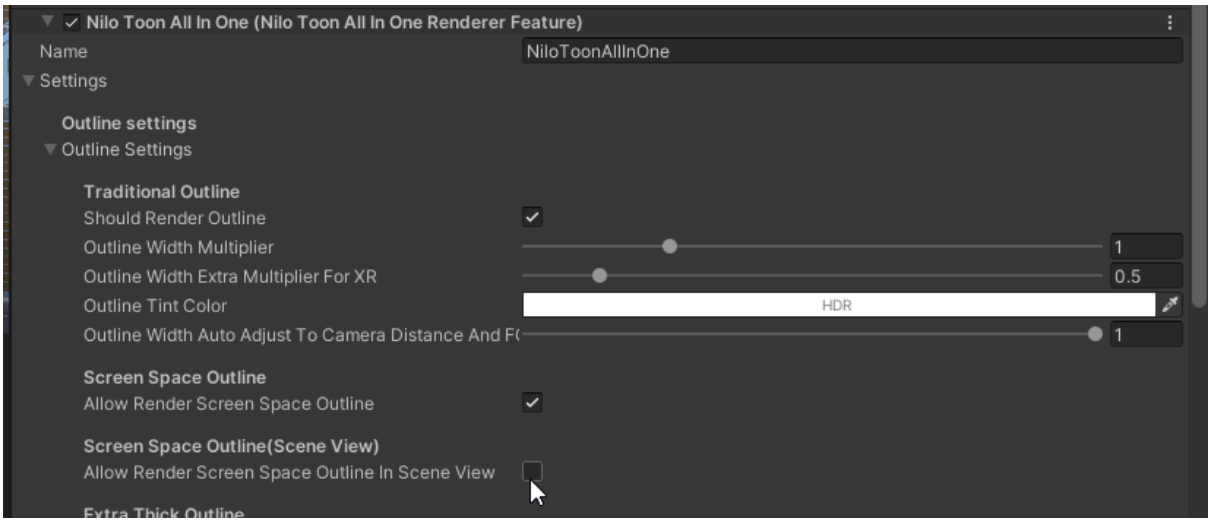


or click [Window/NiloToonURP/Model label/Re-fix whole project!] to rebake all character prefabs in the project(**it is very slow**), if you are using a build machine that never played the game, clicking this button before build may solve the “outline disappeared in build” problem.



Screen space outline in scene window is flicking

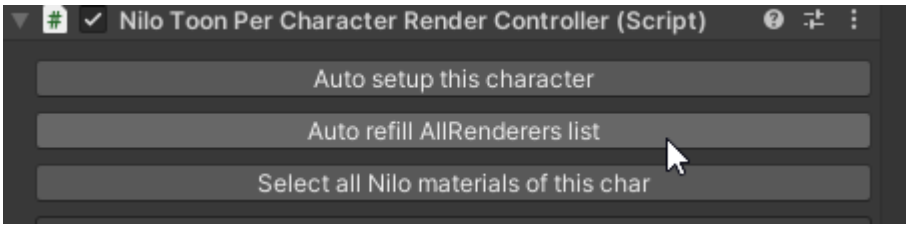
It is a known bug, when you move your mouse, which triggers GUI update, this will make the scene window's screen space outline flicker. You can try **disable Allow Render Screen Space Outline in Scene View**



Screen space outline affected by Game window size or RenderScale?

It is a known limitation of screen space outline method, we are looking for ways to avoid this behavior.

Update NiloToonPerCharacterRenderController's AllRenderers list automatically?

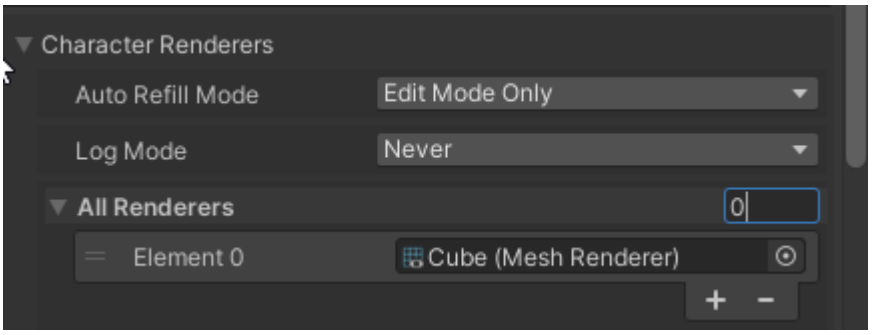


(method A)
click "**Auto setup this character**" button

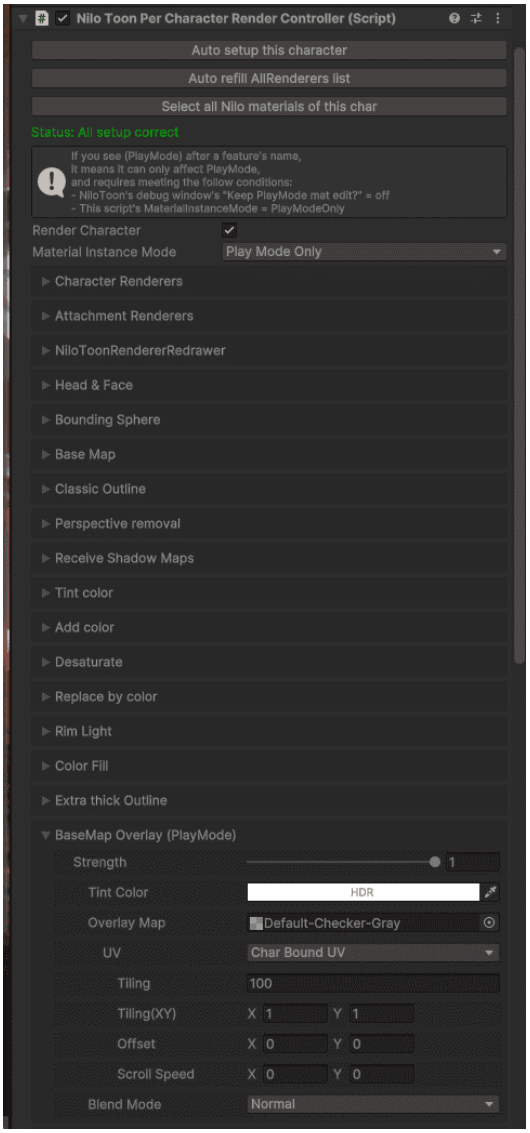
Doing this will trigger an auto update once, NiloToon will find all valid renderers in all child gameobjects of this script, and put them to **All Renderers** list.

(method B)
Or click the "**Auto refill AllRenderers list**" button

(method C)
Or you can **set All Renderers list's count to 0**, which will also find all renderers again automatically.



BaseMapOverride per chara

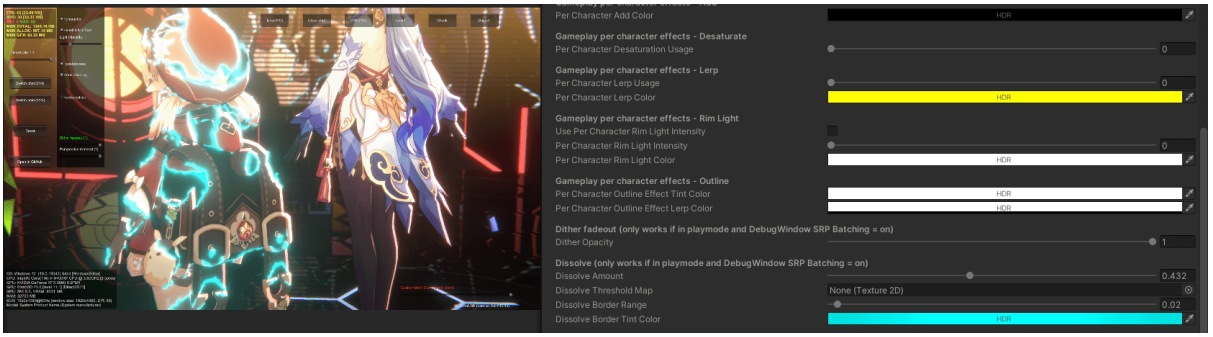


Just like applying a BaseMapStackingLayer to the whole character, common to control:

- Overlay Map
- UV
- Tiling

Dissolve per chara

In **play mode**, find NiloToonPerCharacterRenderController at the root of each character, drag Dissolve Amount from 0 to 1 and you should see the dissolve effect.



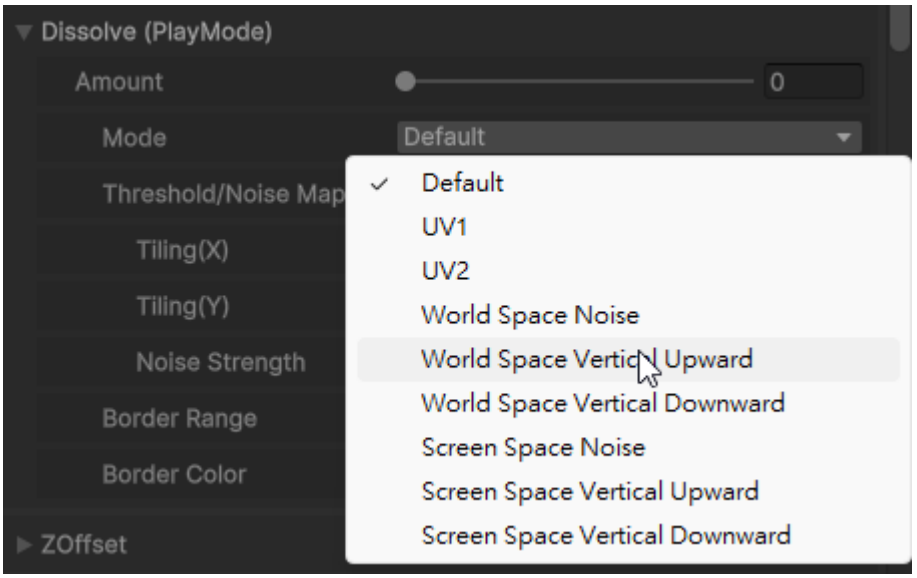
See examples of per character dissolve:

シュガーラッシュ / miComet(official)

[NiloToonURP] Per character render control demo (Little Witch Nobeta model)

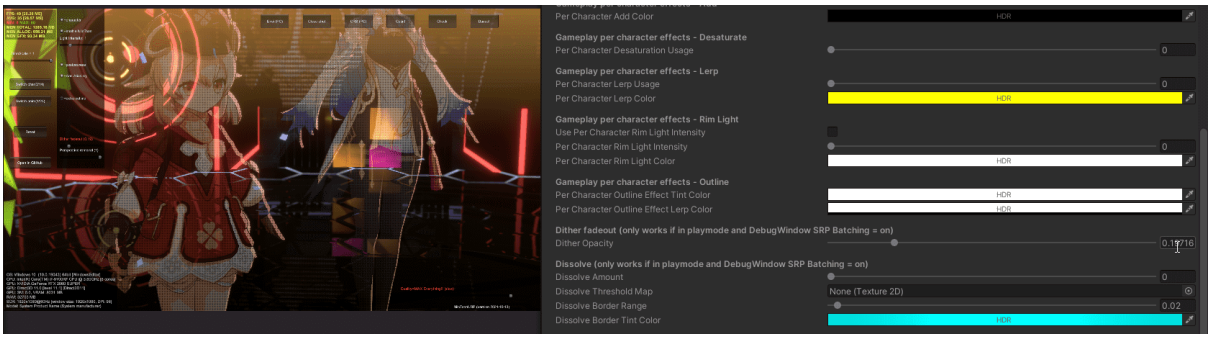
Highly recommend trying different **Mode** for finding a better dissolve style, for example, we may use the following **Mode** when introducing a new character on a 3DLive stage:

- **World Space Vertical Upward**
- **World Space Vertical Downward**



Dither fade per chara

In **play mode**, find NiloToonPerCharacterRenderController at the root of each character, drag Dither Fadeout's Opacity to 0 and you should see the dither transparent effect.



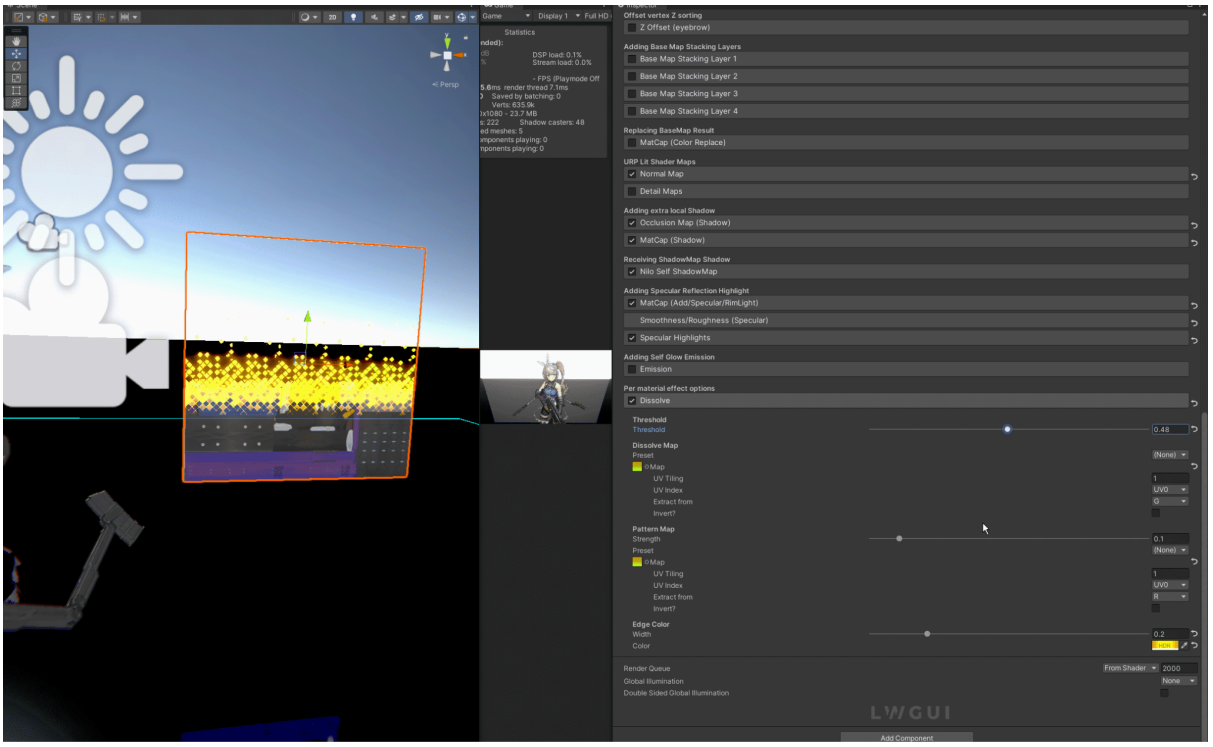
It required High **RenderScale** and **TAA/DLSS** to blur the holes of dither effect.

Dissolve/dither target renderers/materials only

If you need to dissolve per renderer / per material that is similar to the following video:

https://www.youtube.com/live/PlC7enNzbgE?si=sMBJEsOuQ2nfYXaz&t=226

First try using the **Dissolve** group inside the **material** instead, **without** using any settings from **NiloToonPerCharacterRenderController**.
It dissolves by a selected uv, it is easier to use, and can produce better per material animated results and dissolve edge patterns (you can freely use any pattern textures that you like).



This per material dissolve is great for dissolving cloth parts / attachments one by one when character is transforming to another set of cloth, or dissolving weapons in a game with a character holding a switchable weapon.

but if you need to control dissolve using **NiloToonPerCharacterRenderController**, see solutions below.

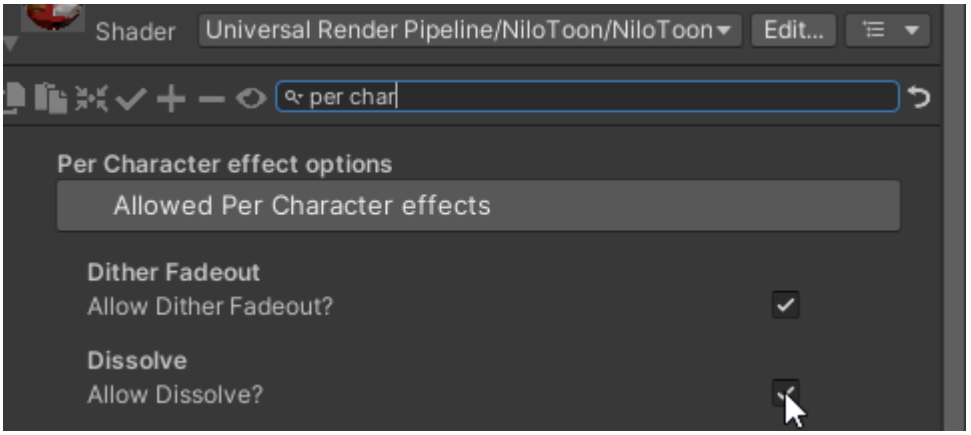
Solution A (Recommended):

Don't add extra NiloToonPerCharacterRenderController. Then use NiloToon character material's **Allow Dissolve?** as a mask to control which material can be dissolved, you can control this toggle(float) in runtime via your custom **C# script**.

For example, if you only want to dissolve a jacket material, you will need to do these via C# in play mode:

1. iterate all materials, set their **_AllowPerCharacterDissolve** to 0 (Material.SetFloat)
2. only set the jacket material's **_AllowPerCharacterDissolve** to 1 (Material.SetFloat)

3. Use NiloToonPerCharacterRenderController's dissolve slider as usual, but now only the jacket material can receive the per character dissolve effect due to step 1 & 2



This solution gives you per material control, so it is recommended.
If you only need a per renderer control, both Solution A/B are ok.

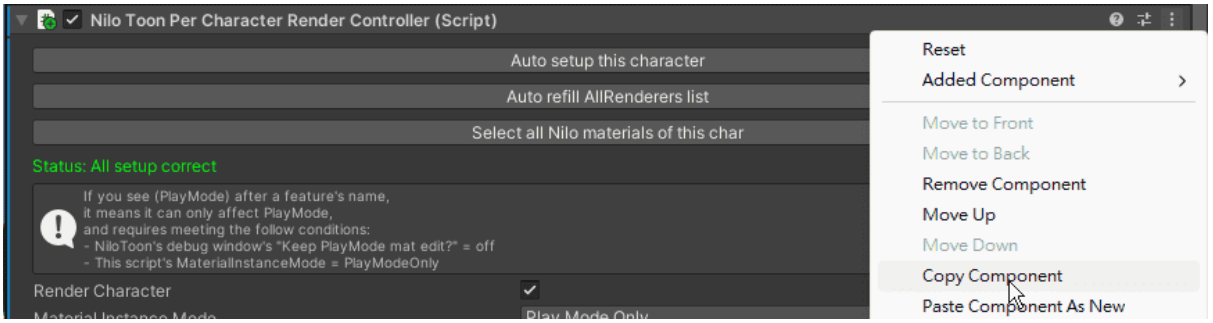
Below is a short example code that will only dissolve any materials that contains body in their name:

```
using System.Collections;
using NiloToon.NiloToonURP;
using UnityEngine;

public class DissolveTargetMatExample : MonoBehaviour
{
    public NiloToonPerCharacterRenderController charaScript;

    IEnumerator Start()
    {
        // wait 1 frame for NiloToon to init material instances, after that
you can safely call renderer.materials
        yield return null;

        // set _AllowPerCharacterDissolve on the next frame
        foreach (var renderer in charaScript.allRenderers)
        {
            foreach (var mat in renderer.materials)
            {
                if (mat.name.Contains("body"))
                {
                    mat.SetFloat("_AllowPerCharacterDissolve", 1);
                }
                else
                {
                    mat.SetFloat("_AllowPerCharacterDissolve", 0);
                }
            }
        }
    }

    void Update()
```

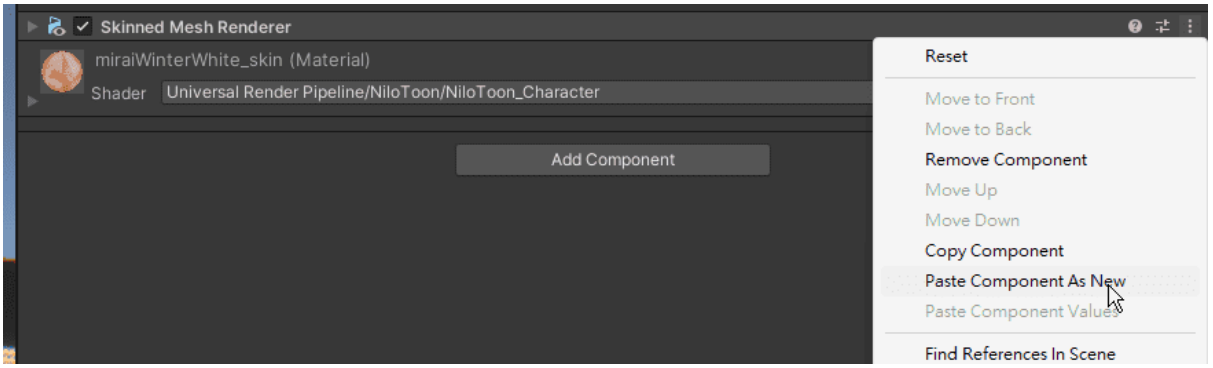
```
    {
        charaScript.dissolveAmount = Mathf.Sin(Time.time * 5f) * 0.5f + 0.5f;
    }
}
```

Solution B:

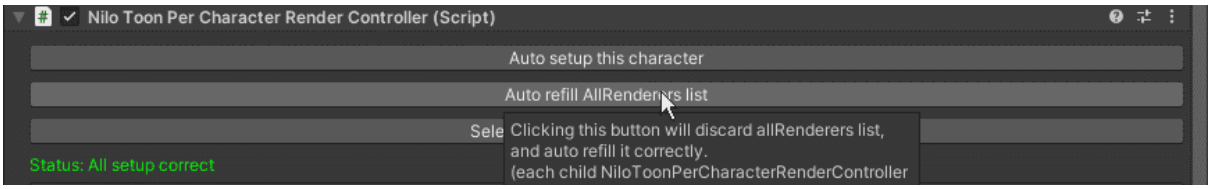
1) copy the **root** NiloToonPerCharacterRenderController script



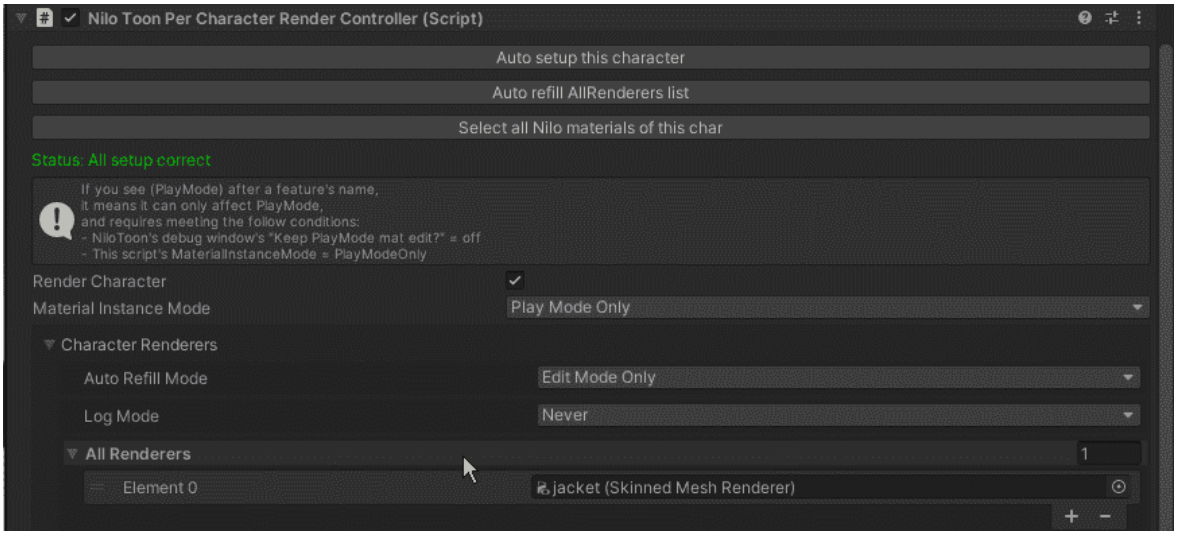
2) paste it to the target renderer (e.g., jacket renderer)



3) click 'Auto refill AllRenderers list' button on the **root** NiloToonPerCharacterRenderController



4) check the new pasted NiloToonPerCharacterRenderController, in Character Renderers > All Renderers, only the target renderer is in the list (e.g., jacket)

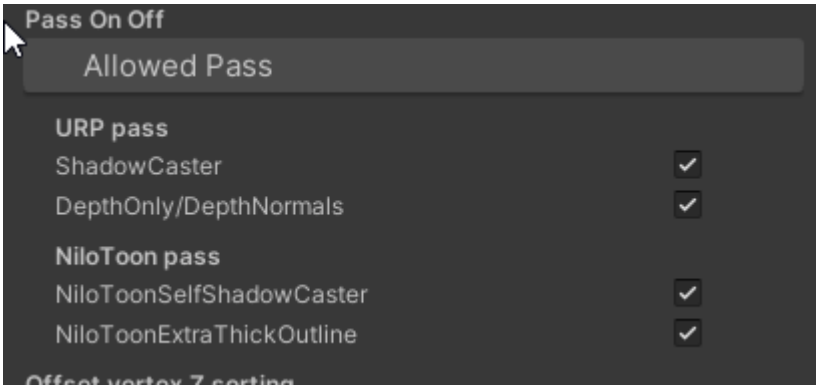


5) check the original root NiloToonPerCharacterRenderController, in Character Renderers > All Renderers, it includes all child renderer, except the jacket

6) Now enter play mode, you can apply controls to that specific renderer alone (e.g. dissolve/dither/color tint a jacket alone)

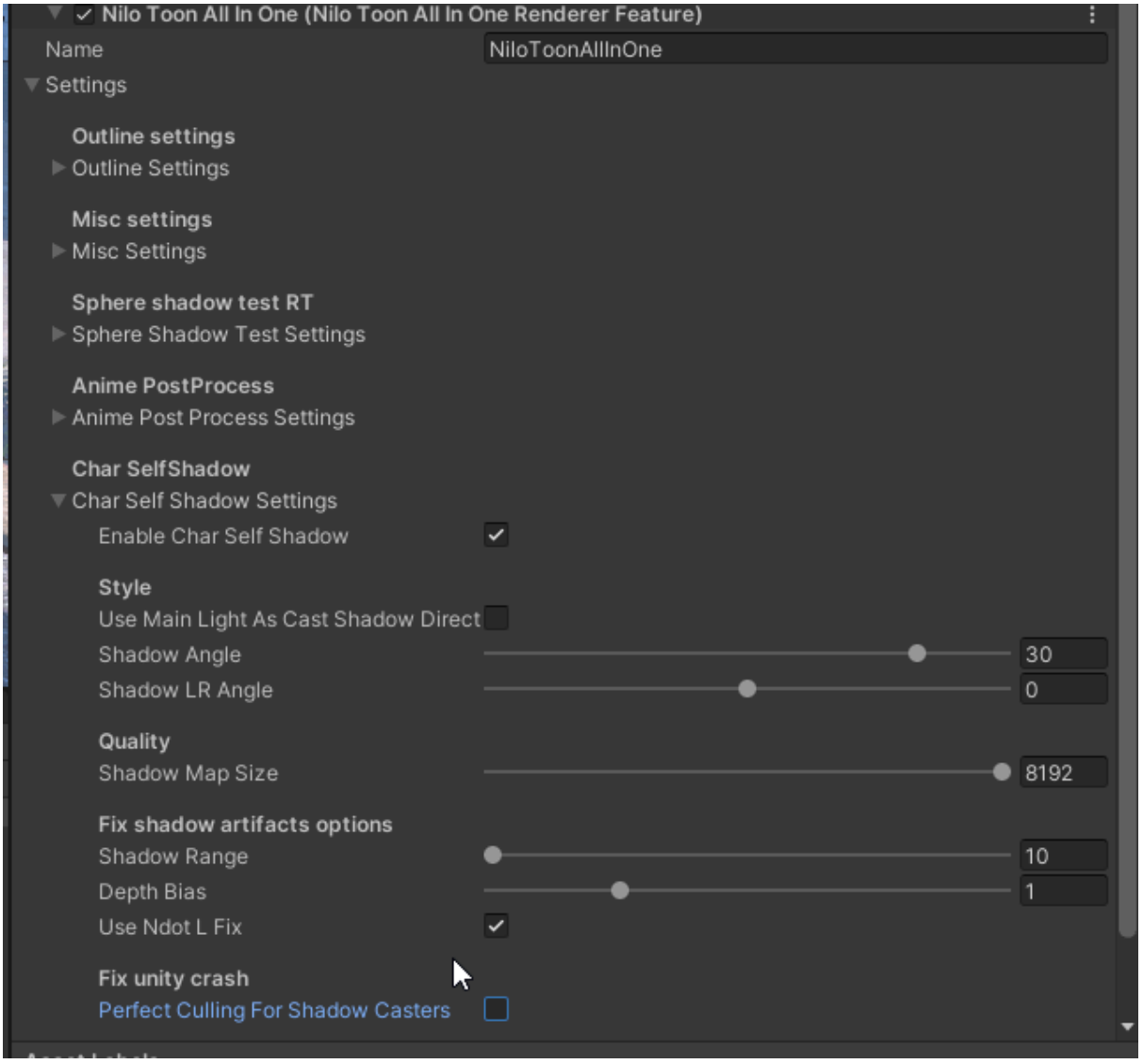
Please note that the controls of two NiloToonPerCharacterRenderController will be independent of each other. For example, after the above setup, a tint color control from the root NiloToonPerCharacterRenderController will not control that specific child renderer(e.g. jacket) anymore, be careful.

Disable pass per material



In material's Pass On/Off section, you can control the following per material:
- material cast URP's shadow?
- material cast NiloToon's shadow?
- material render extra thick outline?
- material draw into depth texture?

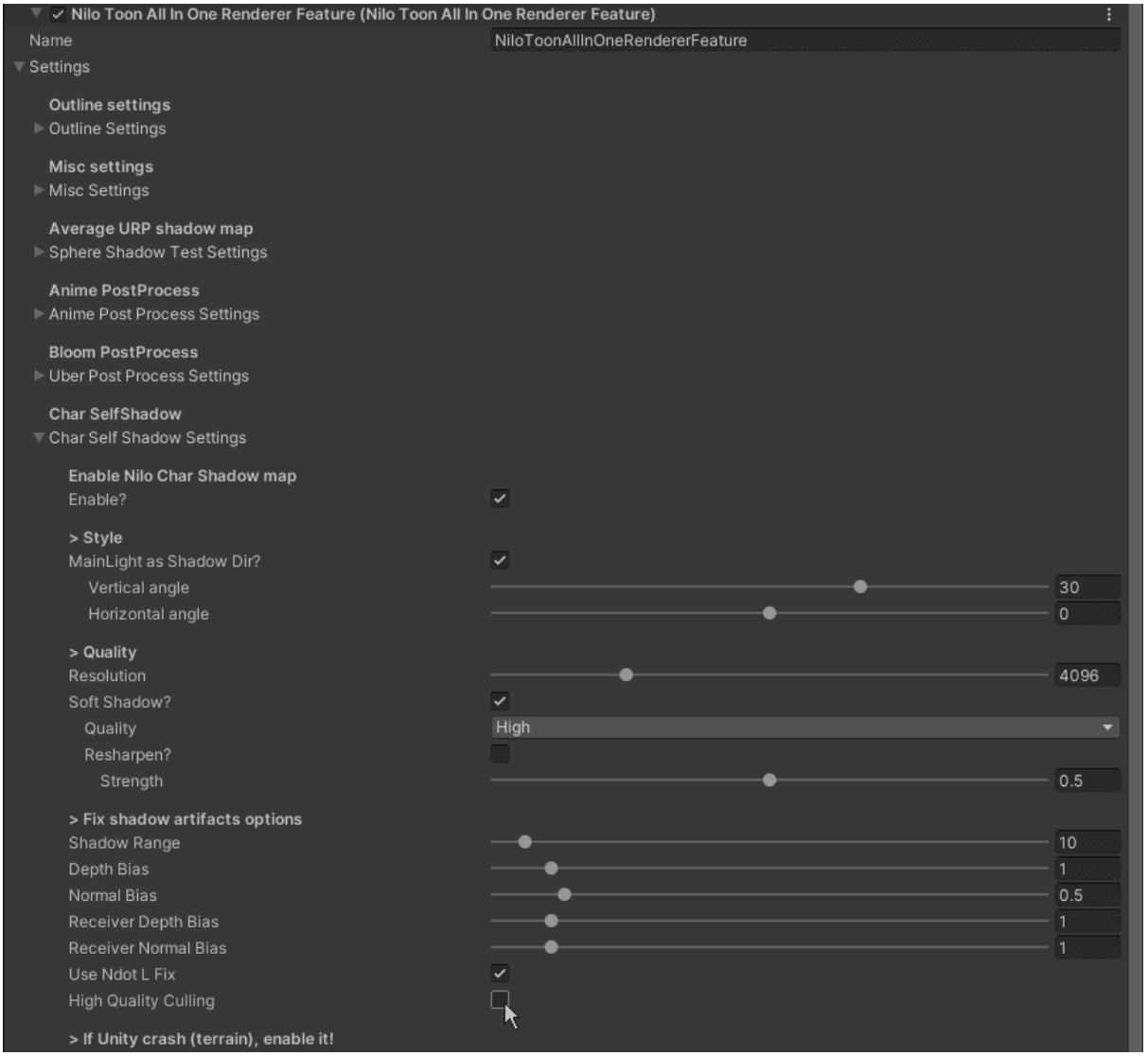
Terrain makes Unity crash



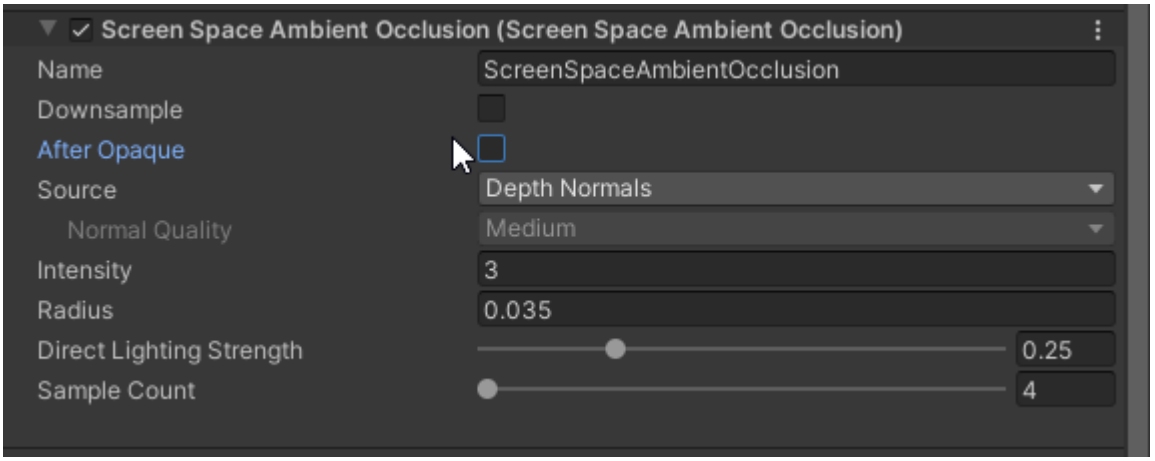
Disable Perfect Culling For Shadow Casters in NiloToonAllInOneRendererFeature, but disable it will make shadow culling not always correct if shadow caster is not visible on screen.

VLB SRP batcher bug

Disable NiloToon renderer feature's **High Quality Culling**, it conflicts with VLB if VLB's SRP Batcher mode is used. This is due to VLB using OnWillRenderObject() to change batching results, which can't handle custom camera culling.



Char darken by SSAO

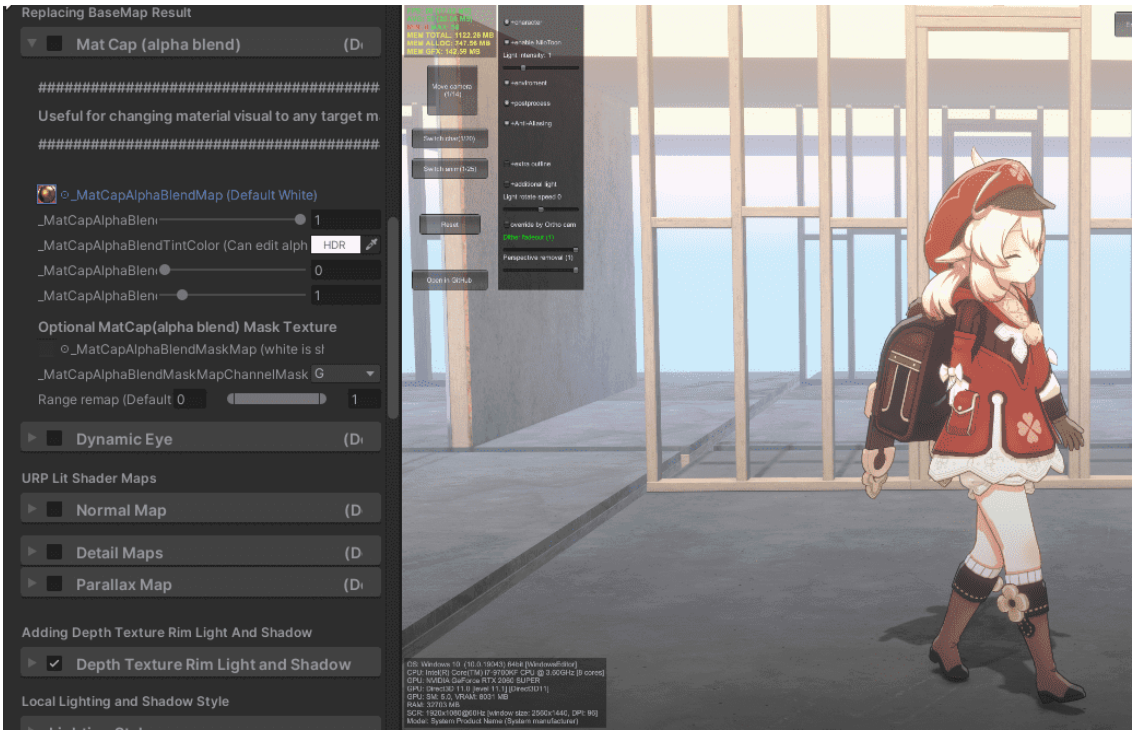


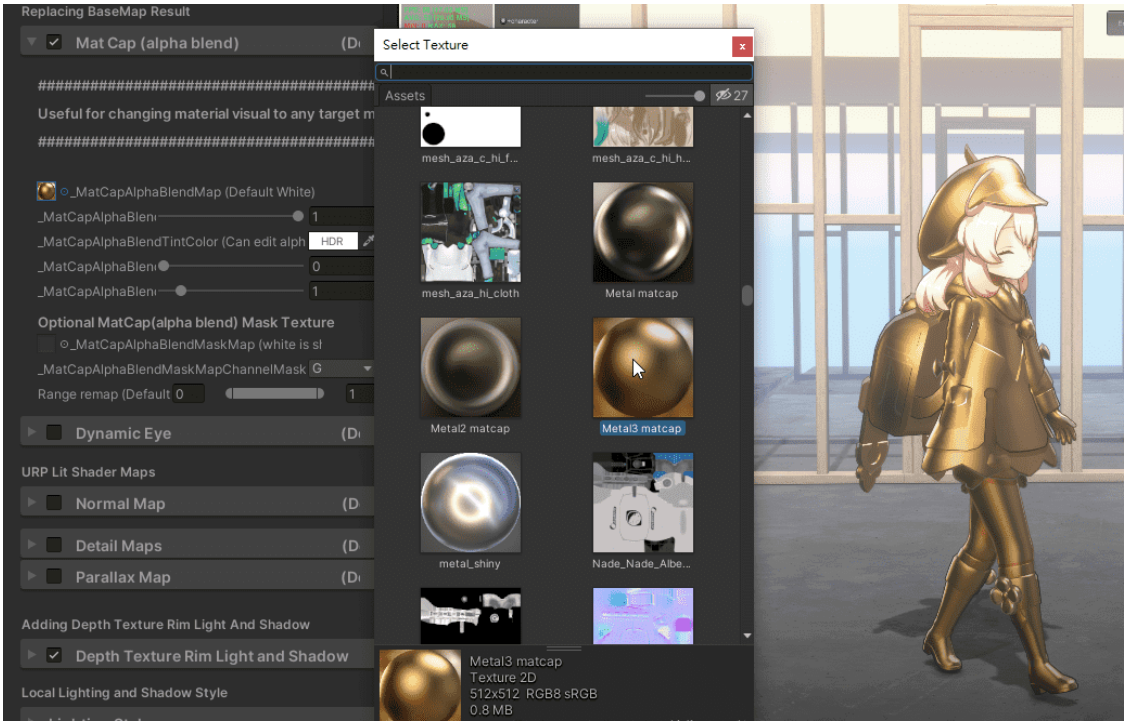
Try to disable SSAO's **After Opaque**.
When **After Opaque** is enabled, it will draw SSAO on top of Opaque character materials pixels, which is not desired for Non-photorealistic rendering (NPR) shaders since it feels dark and dirty.

Metal/reflect/rim/specular material

You can use these functions within the character material to produce metal material:

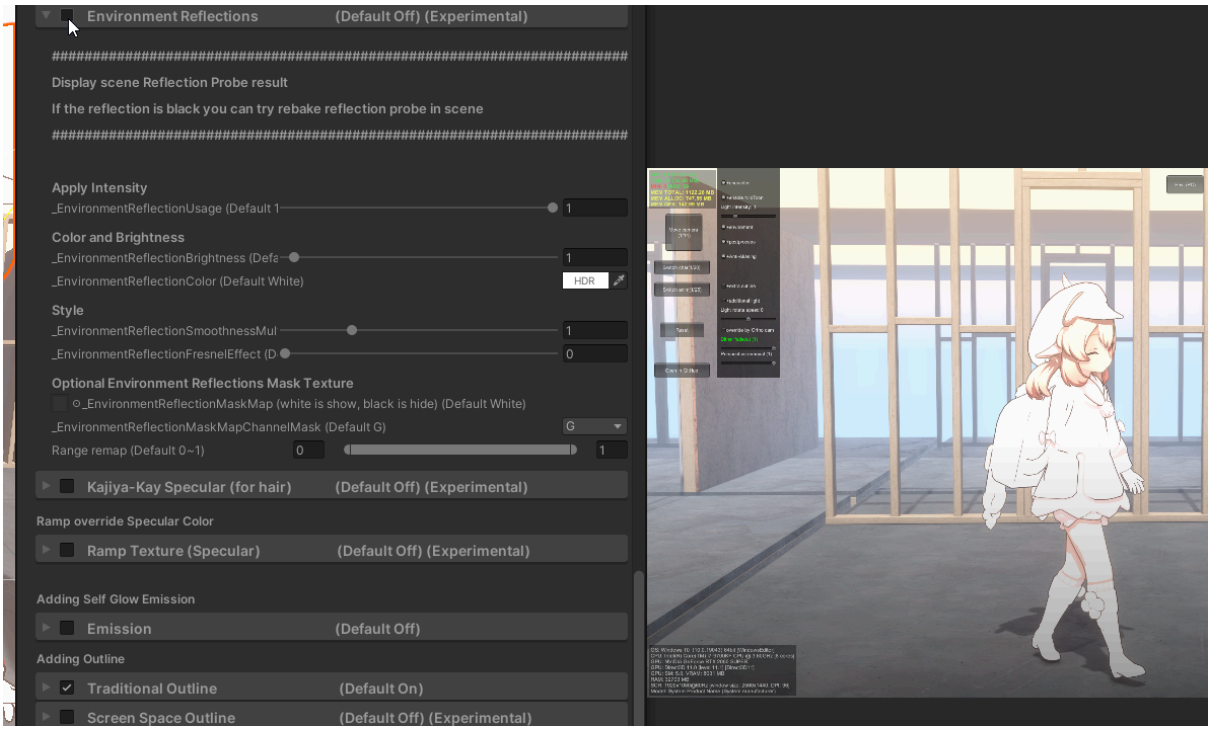
(1) Mat Cap(alpha blend)
you can download matcap textures from:
https://www.kchapelier.com/matcap-studio/#gold
https://github.com/nidorx/matcaps

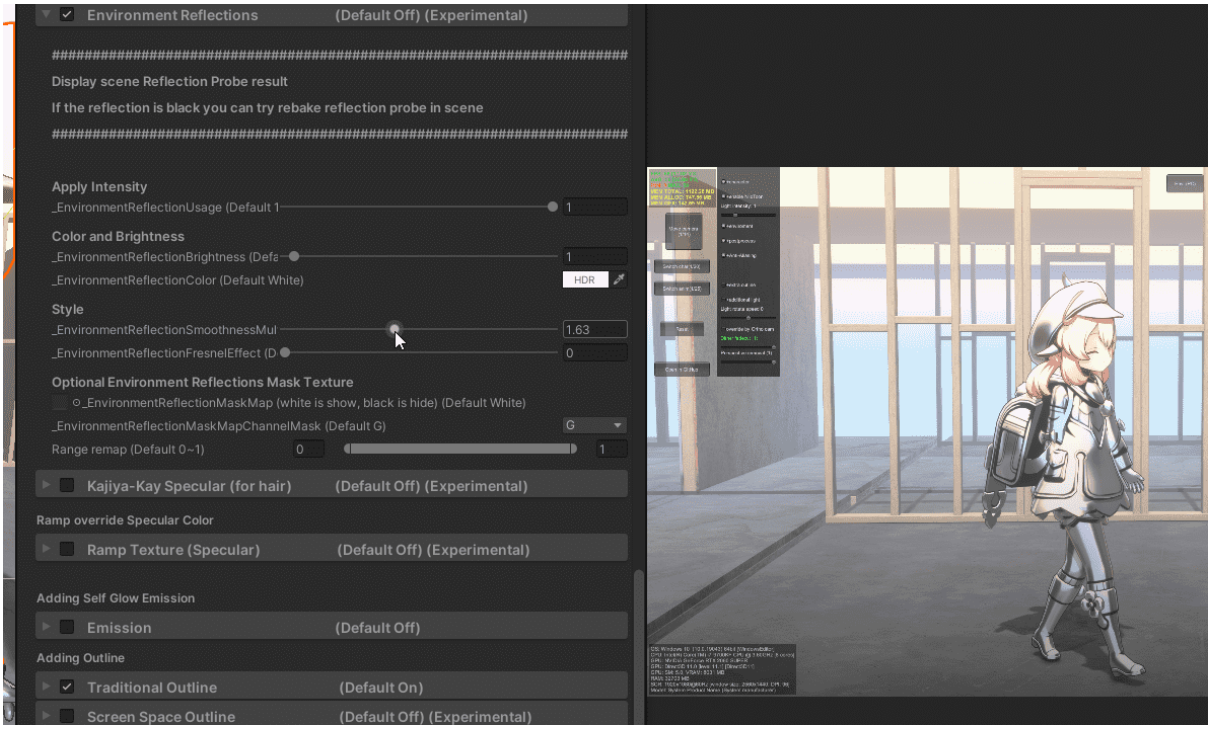




(2) Environment reflection

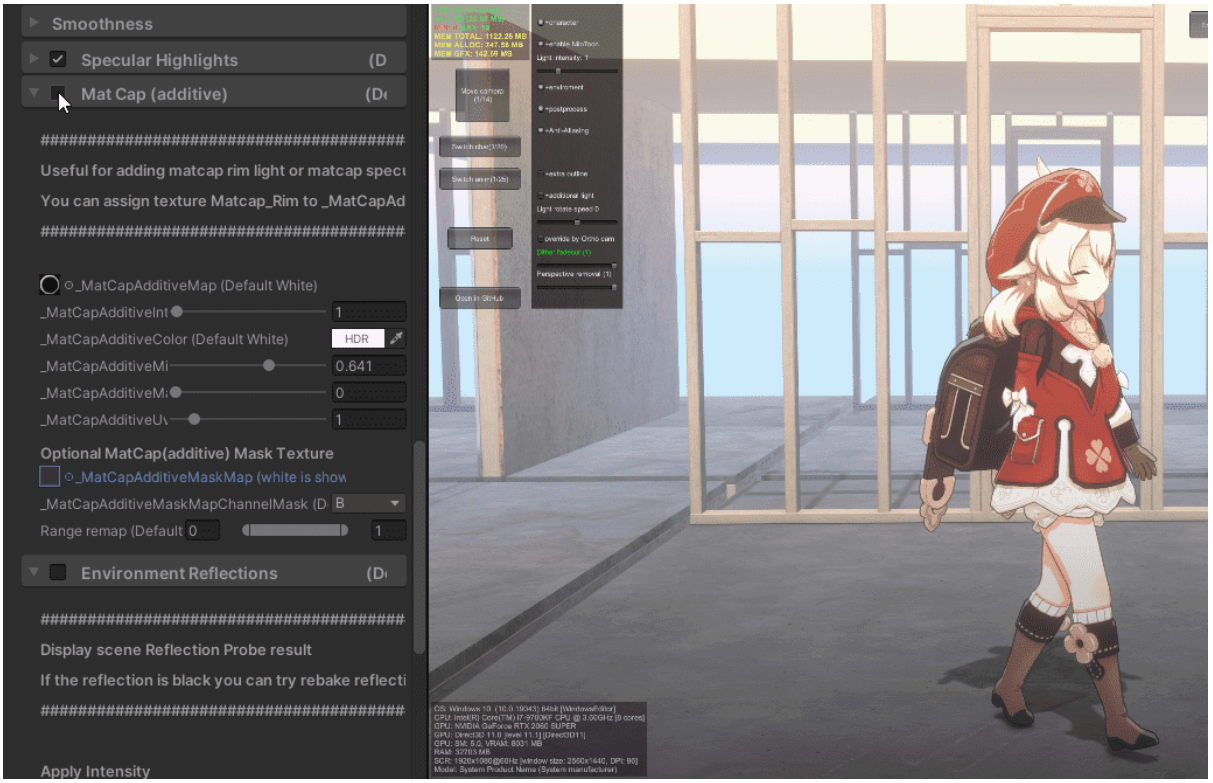
this feature will display scene's reflection probe data, so this feature will require you to bake reflection probe in scene

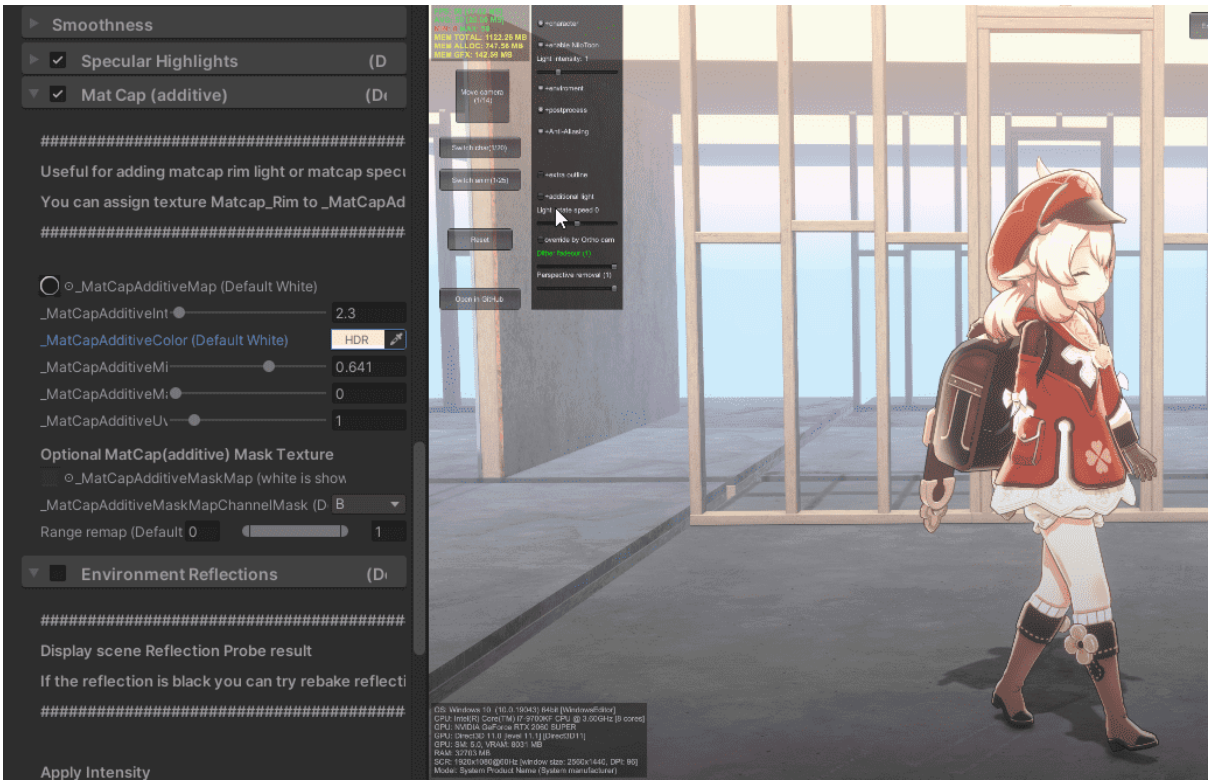




(3) Mat Cap(additive)

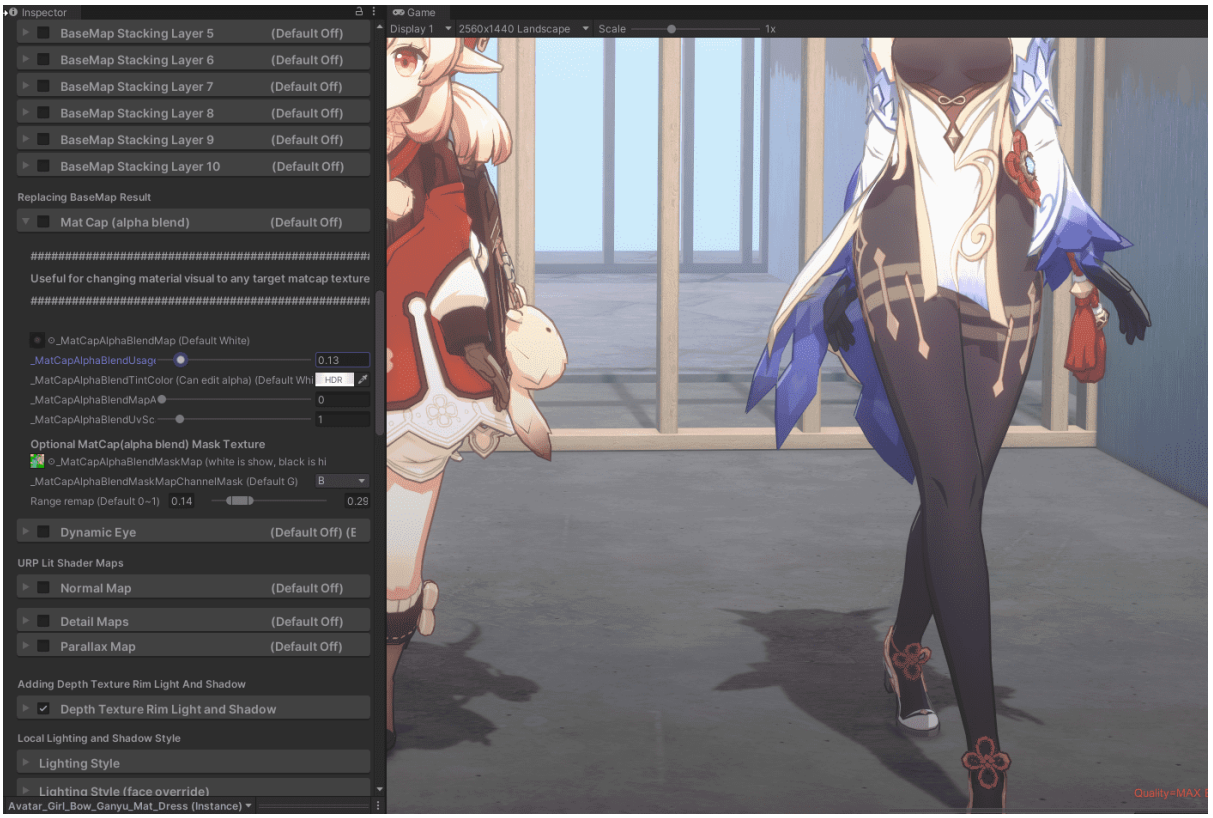
Ideal for adding rim light or high light using a matcap texture

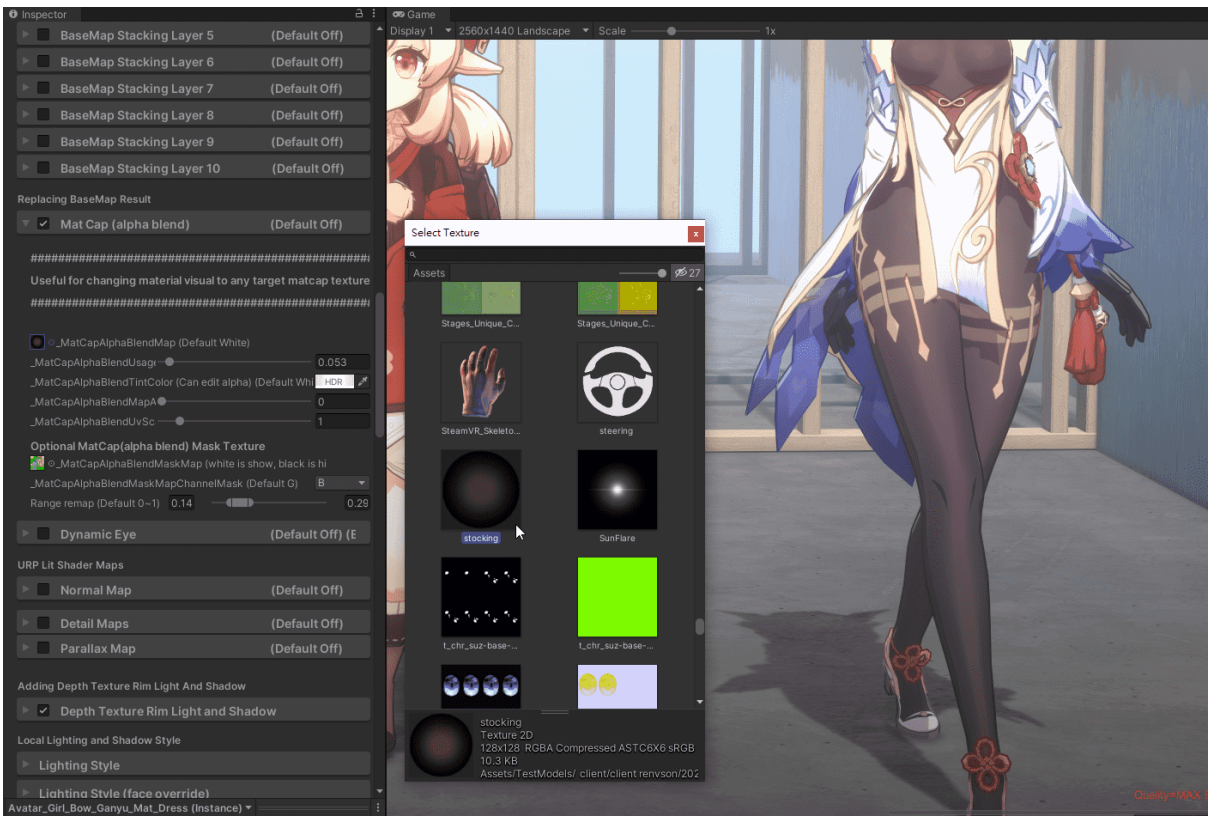




(4) Specular Highlights

Black/white stocking material





Solution A:

Use Basemap stacking Layer, set:

- multiply blending mode
- mask uv = matcapUV
- mask texture = matcap soft shadow
- tint color = a darker than white color

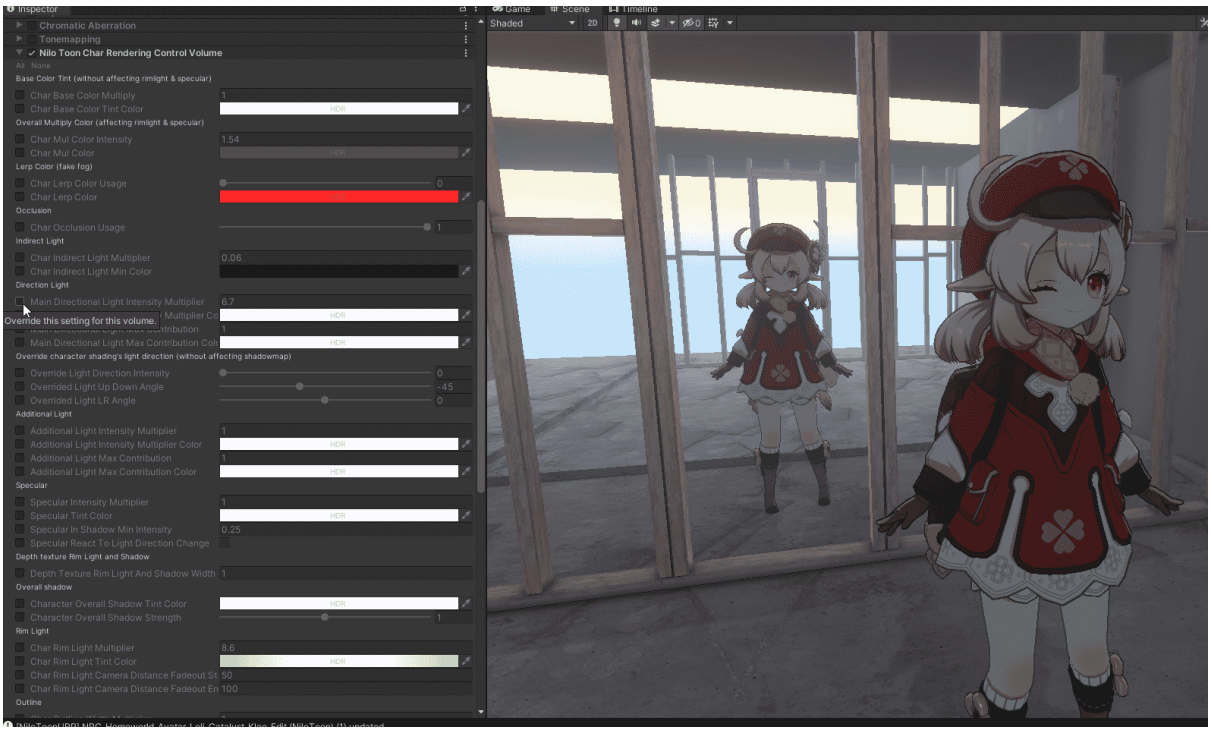
then you can tint the a surface only at glazing angle

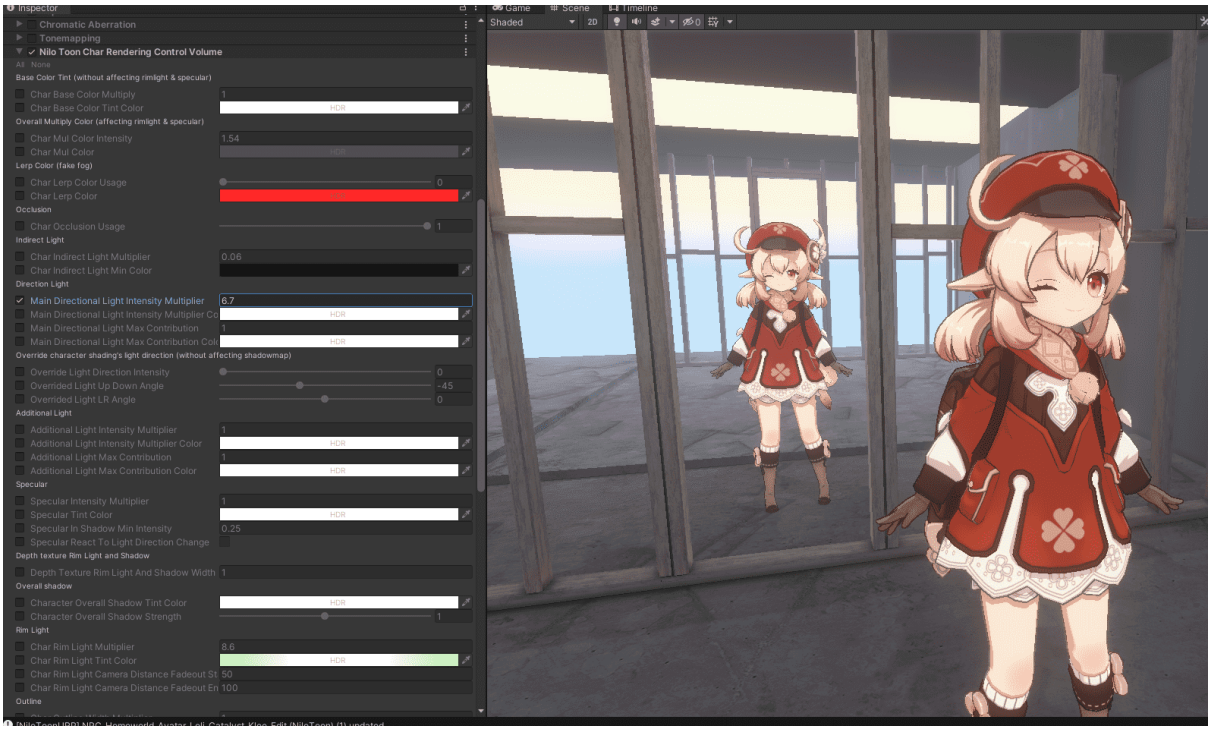
Solution B (not recommended):
use Matcap (alpha blend) with a stocking matcap texture

Brighten char in dark envi

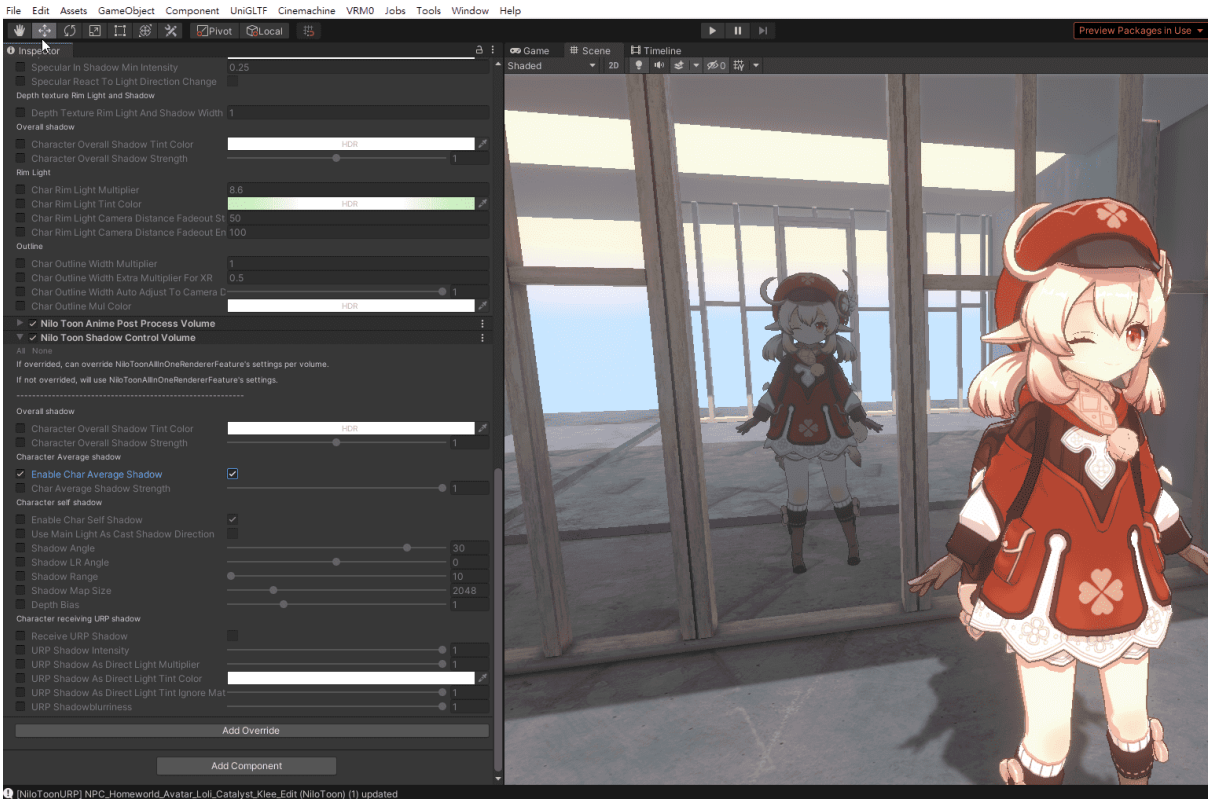
use NiloToonCharRenderingControlVolume to adjust

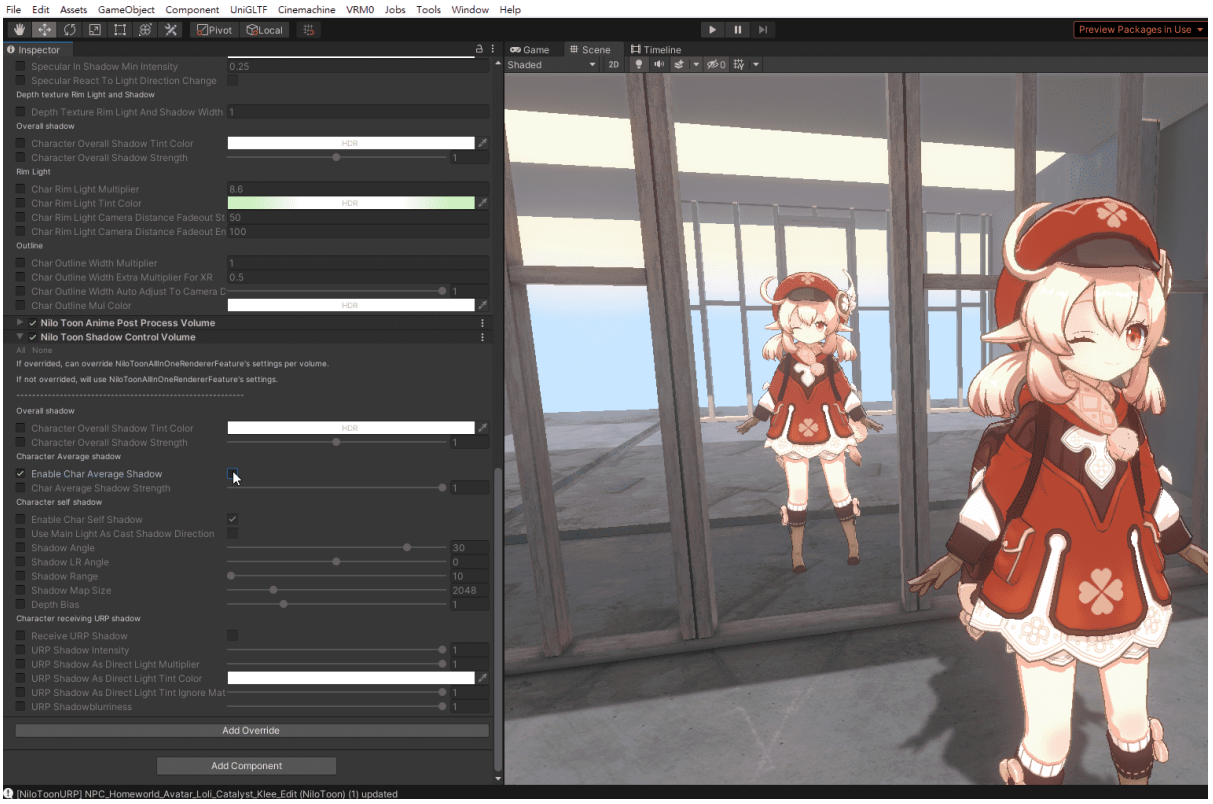
- **Char Indirect Light Multiplier**, set a higher value
- **Char Indirect Light Min Color**, set a brighter indirect color



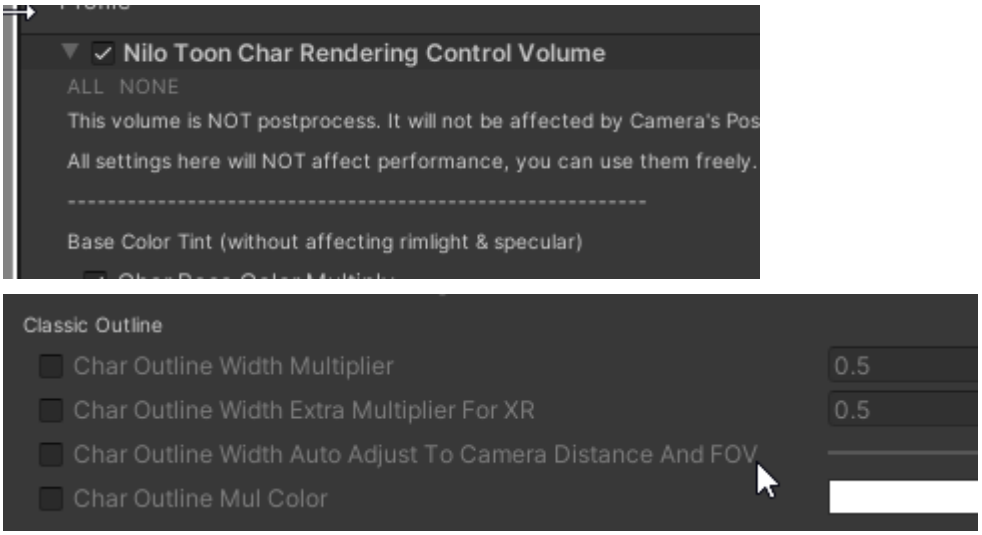


or NiloToonShadowControlVolume, just disable **average shadow** completely





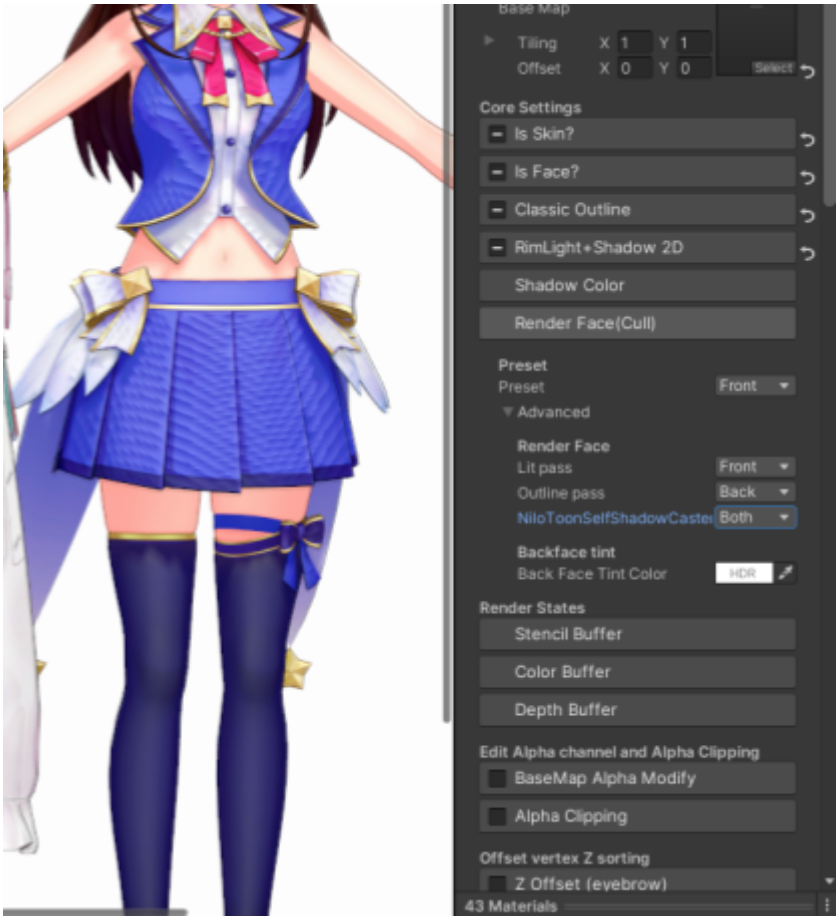
Outline fixed width WS(world space)



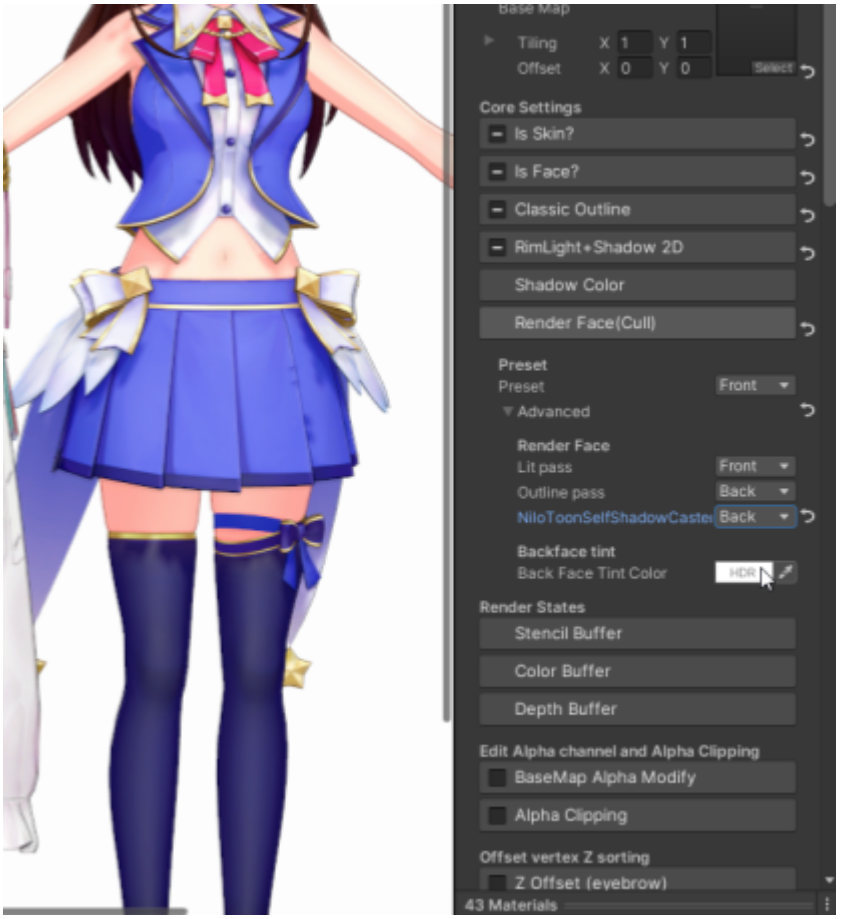
to make the outline never change width in world space, it will require you to use volume -> **NiloToonCharRenderingControlVolume** -> **Char Outline Width Auto Adjust To Camera Distance And FOV** , edit the value from 1 -> 0

Shadow acne - single face mesh

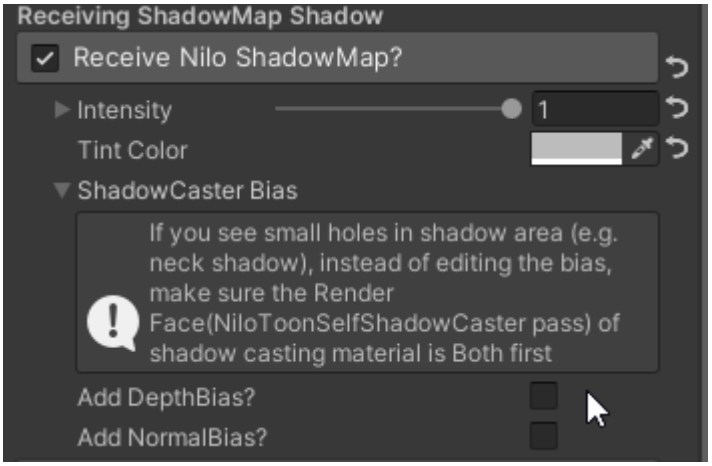
try editing the RenderFace of **NiloToonShadowCasterPass** from **front/both** -> **back**, it may solve the problem



After changing NiloToonSelfShadowCaster pass's Render Face from both -> Back, shadow acne disappeared

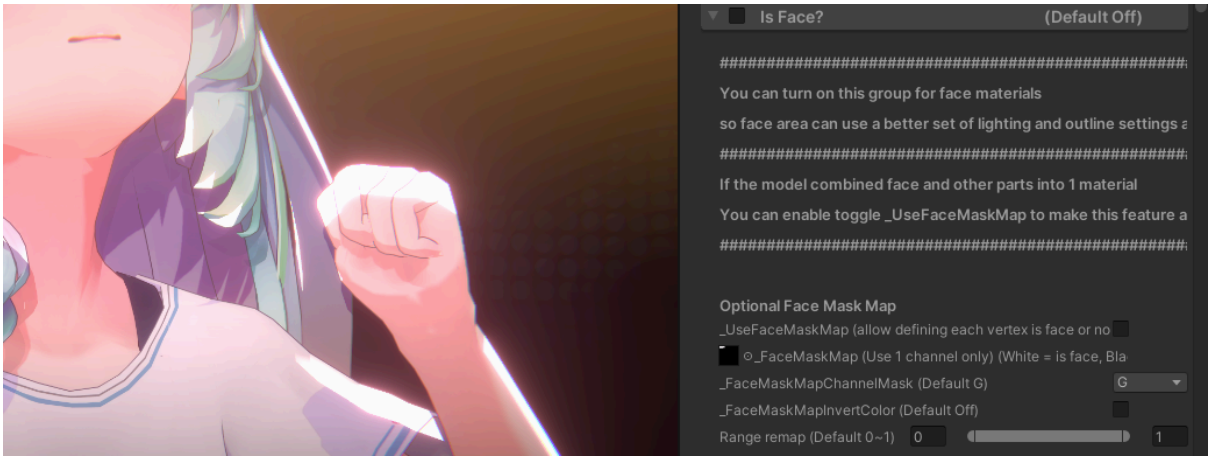


If the above method didn't solve the problem, you can try editing per material shadow bias

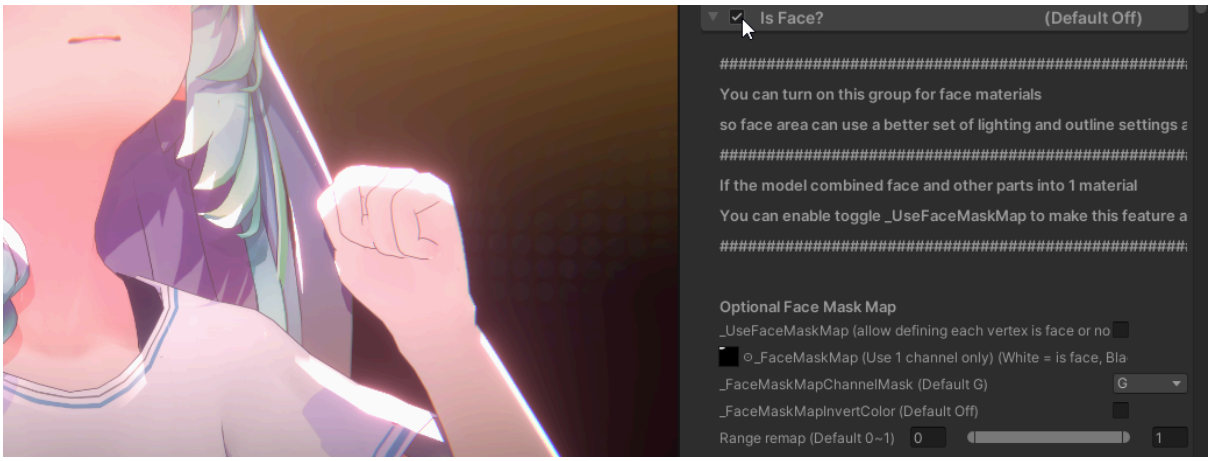


Hide “shadow split line” between face & body mat

1.if you turn on **Is Face** for face material, shadowing between face material and body material is different. In the image’s right side, it shows the body material setting, it doesn’t have **Is Face** turn on.

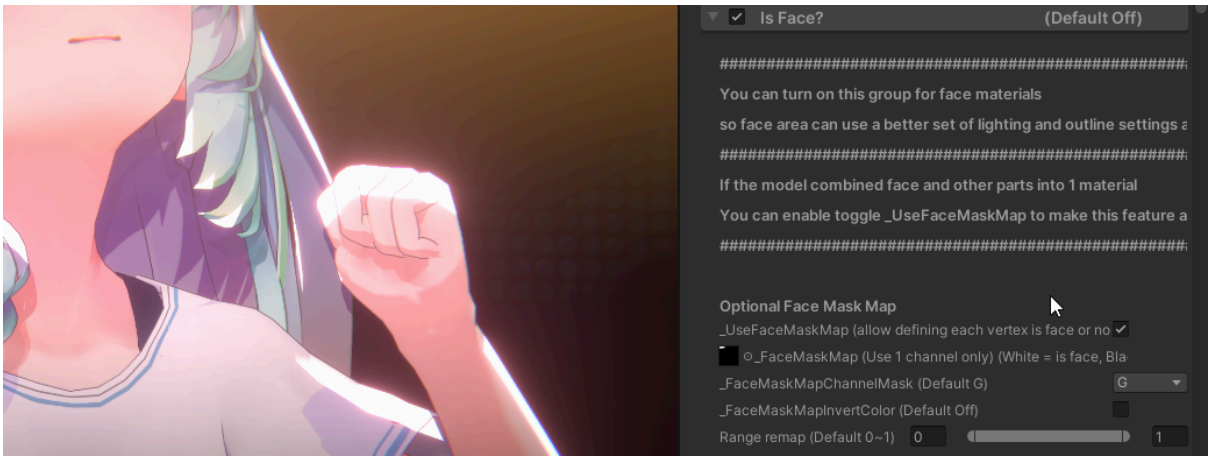


2.but if you turn on **Is Face** for body material, lighting for body material is not correct(e.g. hand)



3.The solution is to turn on **Is Face** for body material, and also turn on _UseFaceMaskMap, then draw a _FaceMaskMap to make only the upper part of the neck become **Is Face,** and the rest of the body material is not **Is Face**.Doing this will produce a soft gradient showing from **Is Face(upper part of the neck)** to not **Is Face(lower part of the neck)**, which looks better.

*You can also not editing the body material, and only edit the face material(by using _FaceMaskMap), either of these method will work also

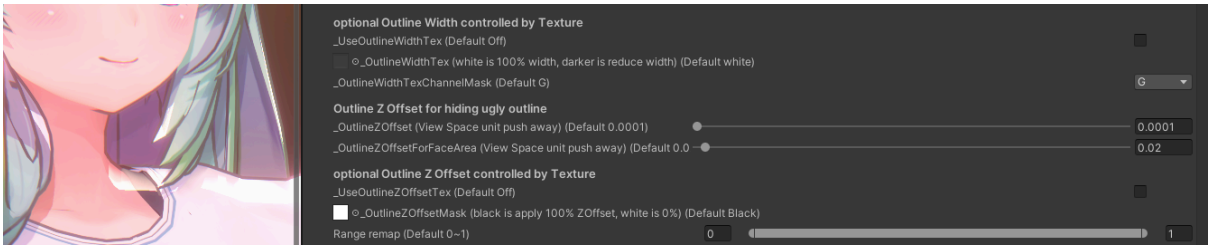


this image below shows the neck color texture

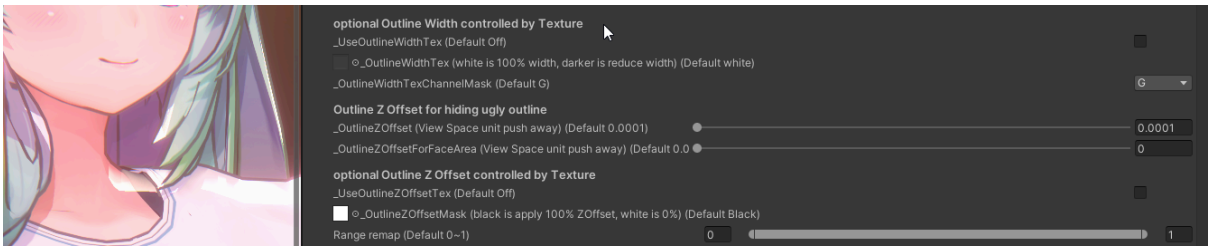
you will want to draw a blurry mask _FaceMaskMap like this (if you prefer a sharp line that matches the head's jaw geometry or matches shadow color drawn in BaseMap, it is also possible)

Improve face's Outline

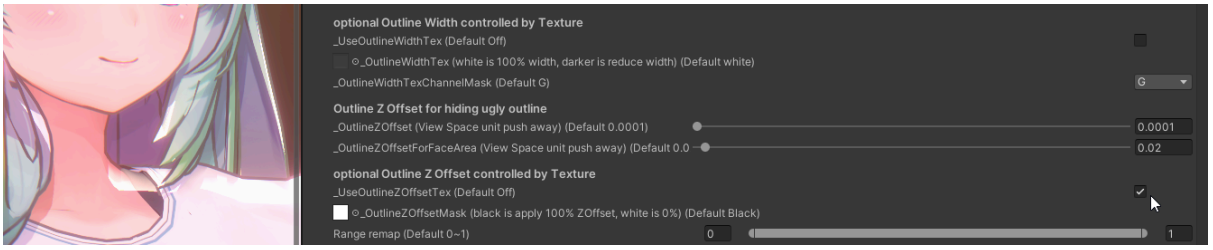
if you turn on **Is Face?** for face material, a stronger default outline ZOffset **_OutlineZOffsetForFaceArea**(0.02) is applied, it will remove most of the ugly outline around eye and mouth, but it may also remove the face shape outline. see the image below.



Solution A: make **_OutlineZOffsetForFaceArea** lower, but ugly outline may appear around eye and mouth again, you will need to fix them using your own OutlineWidth map or vertex color

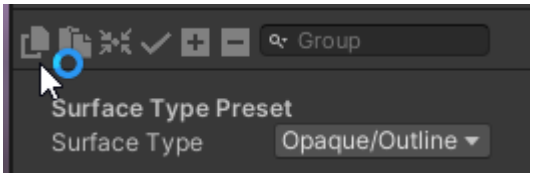


Solution B: keep **_OutlineZOffsetForFaceArea** high enough, turn on **_UseOutlineZOffsetTex**, and draw **_OutlineZOffestMask** to make **_OutlineZOffsetForFaceArea** only apply to ugly outline areas(eye/mouth), but not face shape area, so face shape outline will not be affected, while eye's outline will still be hidden use to outline ZOffset.

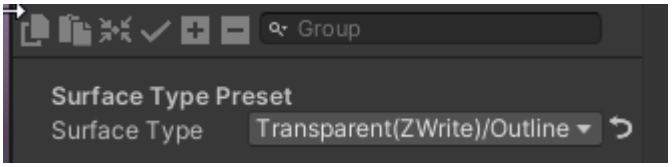


Transparent and outline

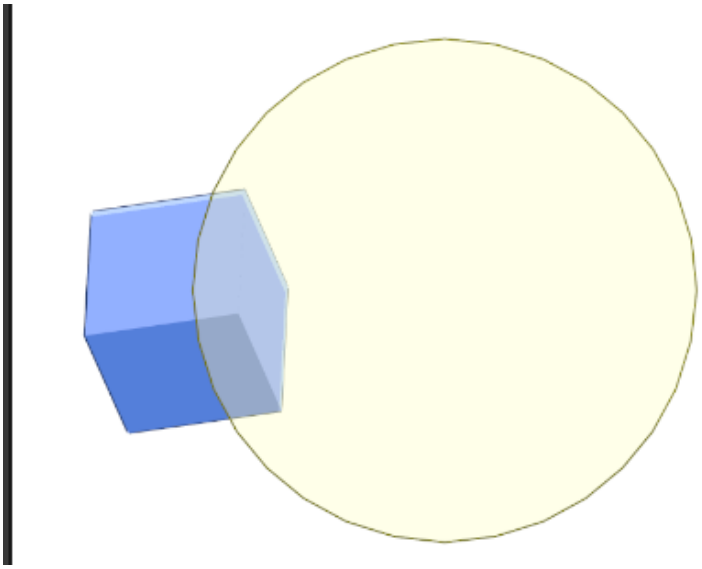
For the Opaque material, use surface type = **Opaque/Outline** or **Opaque(Alpha)/Outline**



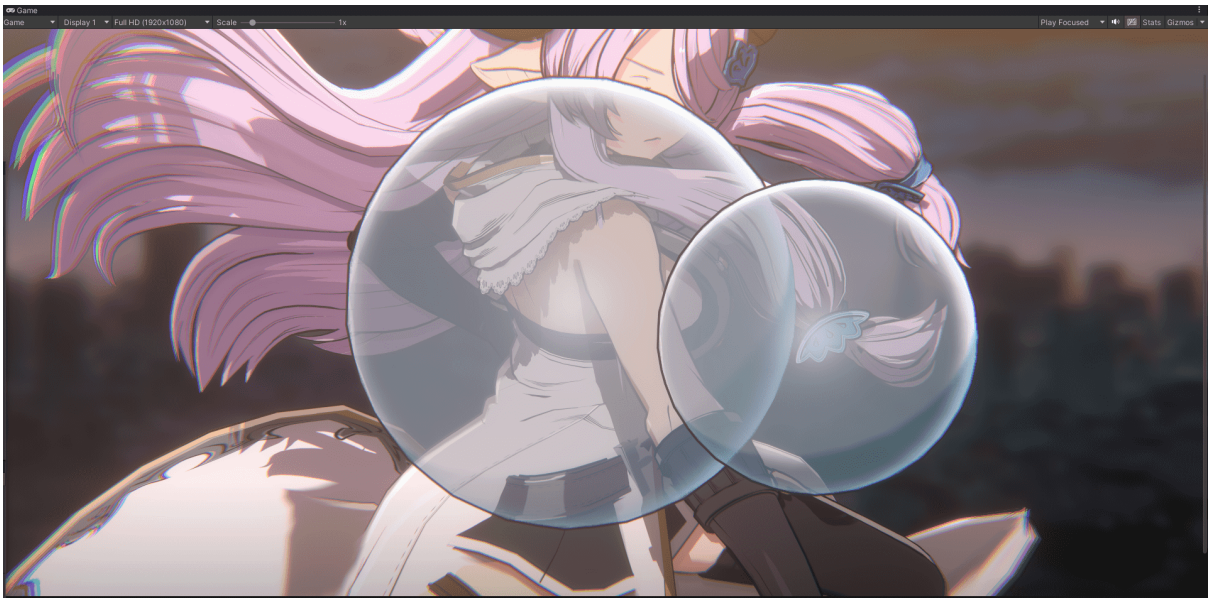
For the transparent material, use surface type = **Transparent(ZWrite)/Outline**

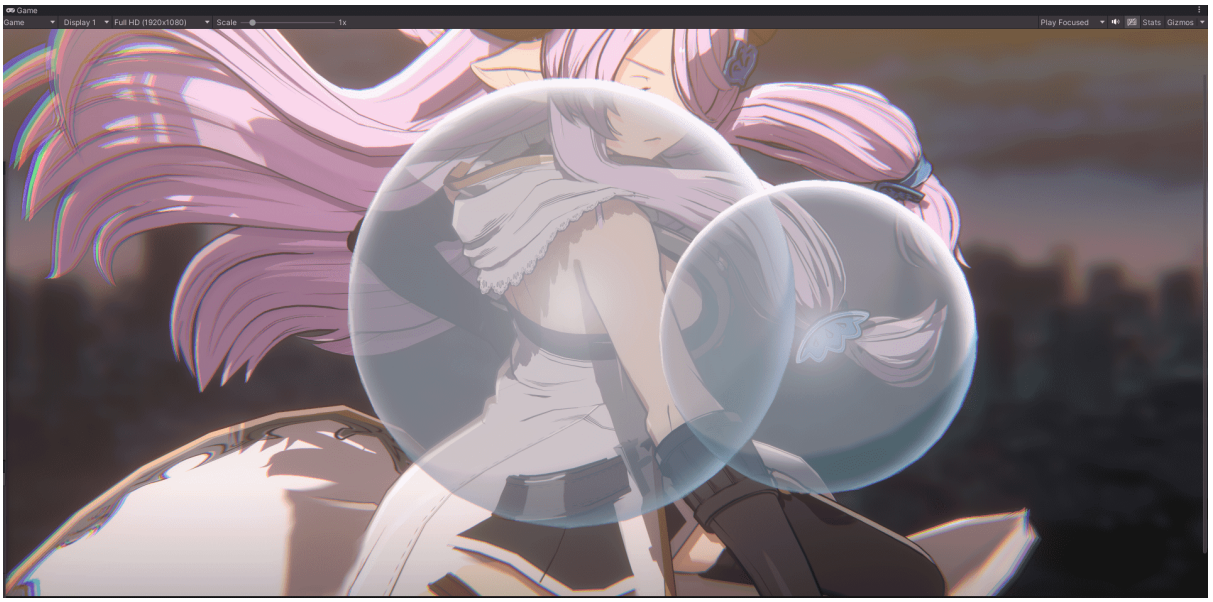


then their outline will work together




Game
Game
Display 1
Full HD (1920x1080)
Scale
1x
Play Focused
Stats
Gizmos


Game
Game
Display 1
Full HD (1920x1080)
Scale
1x
Play Focused
Stats
Gizmos


Game
Game
Display 1
Full HD (1920x1080)
Scale
1x
Play Focused
Stats
Gizmos

Outline ZFight in mobile build

In mobile/WebGL, especially OpenGLES,
Try to:

- lower the far plane of camera as much as possible (e.g. 1000 -> 100)
- increase the near plane of camera as much as possible (e.g. 0.03 -> 1)

it may solve the outline ZFighting problem.

Cam output char’s alpha

If you need to render the character's alpha channel for game UI, RenderTexture, png, mov, OBS, Spout..

You can read this section for solutions.

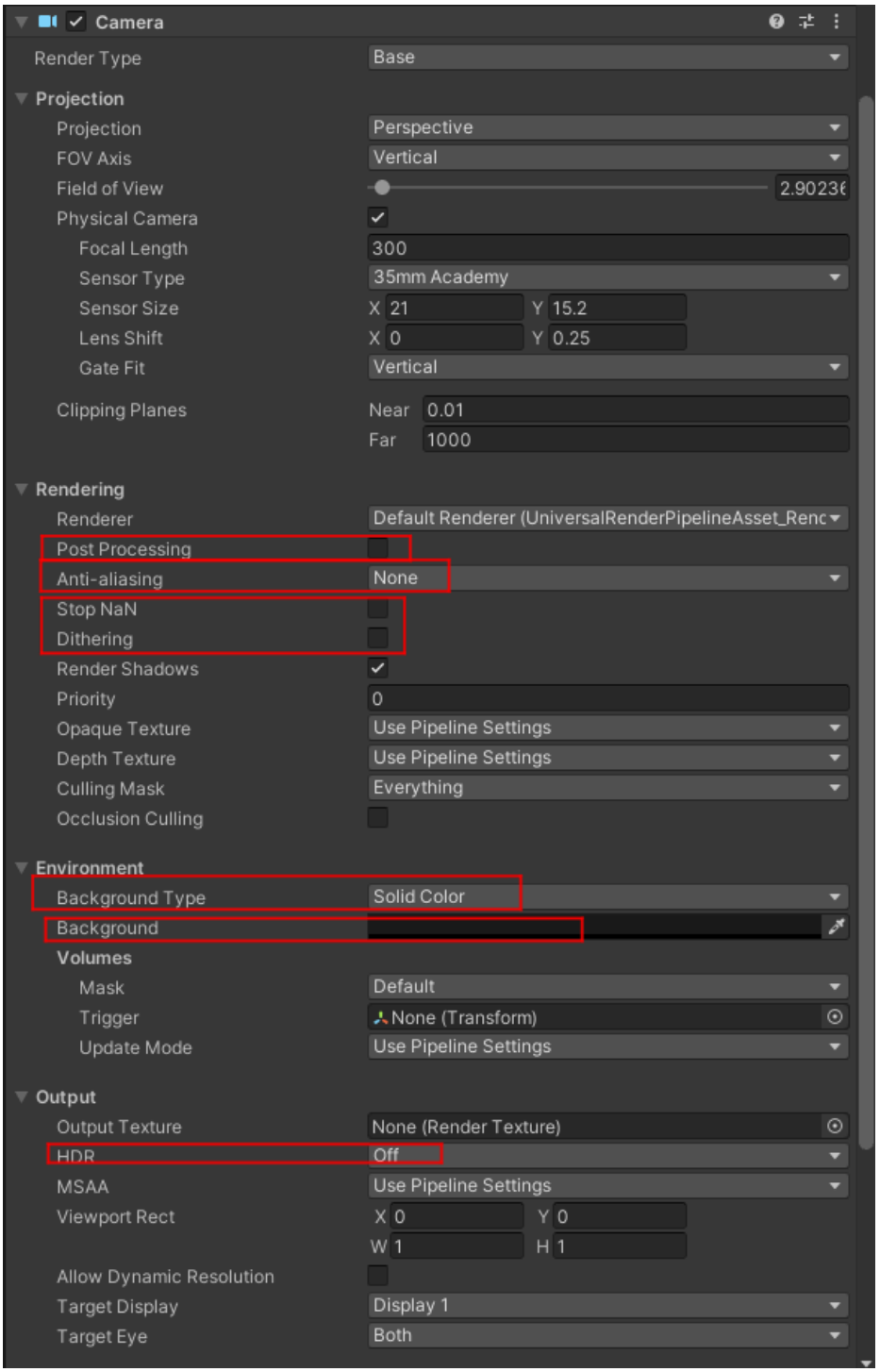
When is needed?

If you are in the following situations:

- You want to render character to a **Render Texture**, where **alpha** of the Render Texture is the character’s mask (e.g., for displaying 3D character using UGUI RawImage in your game’s UI)
- You want to use Recorder to generate **.png** or **.mov**, where **alpha** is the character’s mask
- You want to use **OBS** to capture Unity’s game window or a RenderTexture(**Spout**), where **alpha** of the rendering is the character’s mask, for example, for VTuber 3D streaming using Unity Editor as **OBS’s transparent overlay**

Camera settings for alpha

When using URP(Unity2022.3 or lower), your camera needs to use the following settings in order to output the character's mask to alpha.



URP’s current design will discard any alpha rendered if there is any active post process/HDR, the above camera setting is the only solution to output character’s alpha if you don’t edit URP’s source code.

*Make sure **Environment > background** color’s **alpha** is **0**, many users will miss this step!

*Please Note that effects like **bloom** are PostProcessing, so all those effects will be turned off after turning off the **Post Process** toggle in the camera(see above image).

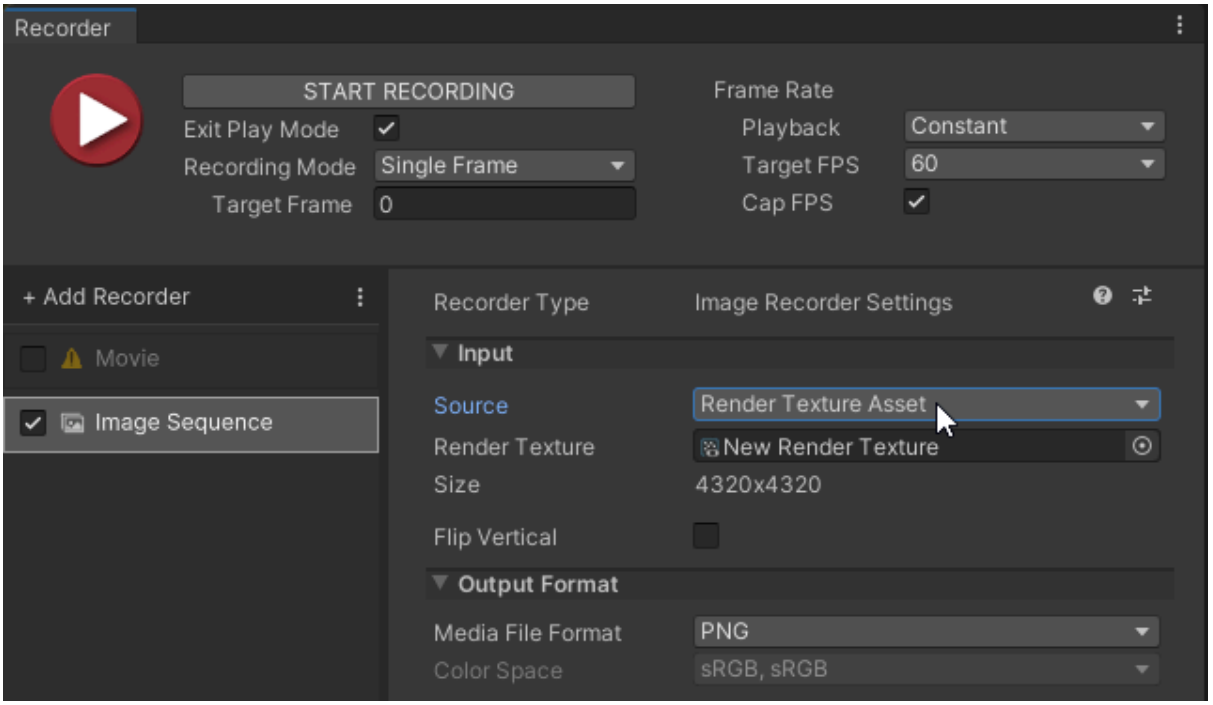
*In Unity6, a new **Alpha Preservation** for postprocess option was added, this may allow you to skip the above setup which disables postprocess.

Recorder Settings for alpha

When using **Recorder 3.x or 4.x (Unity 2021.3 or Unity 2022.3)**, you can only output alpha channel to .png or .mov using these modes:

- Source = **Game View** (Can't record alpha)
- Source = **Target Camera** (Can't record alpha)
- Source = **360 View** (Can't record alpha)

- Source = **Texture Sampling** **(Easy to use, but bad AA quality & limited to 4K)**
- Source = **Render Texture Asset** **(More complex setup, but best AA quality & allow 8K + MSAA. We(NiloToon's developers) always use this method)**

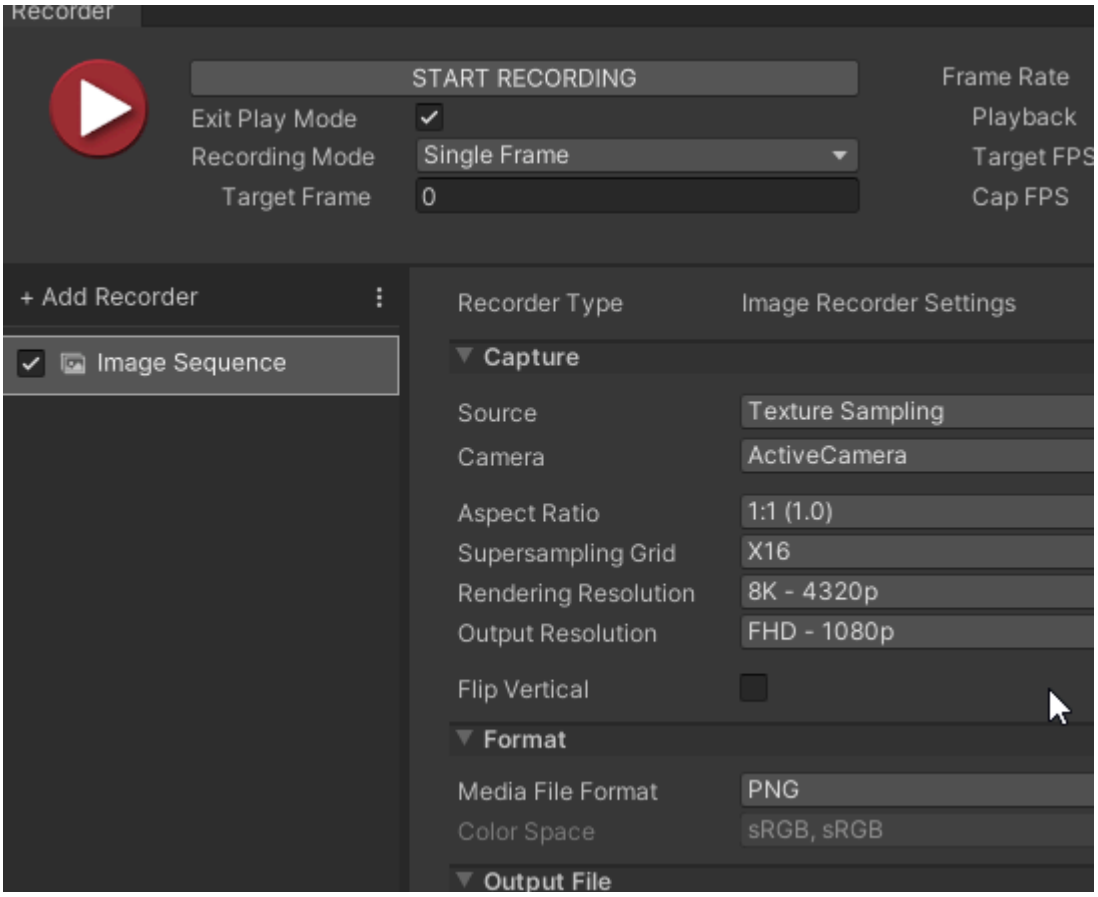


Output alpha .png

Method: Quick & Easy

In Recorder, use Source = **Texture Sampling** if you want the following pros & cons:

- **Quick and Easy to set up**
- **bad AA quality (e.g., bad classic outline)**
- **maximum 4K(2160p) png**



Using the above recorder settings, then click **START RECORDING**, and your alpha png should be produced.

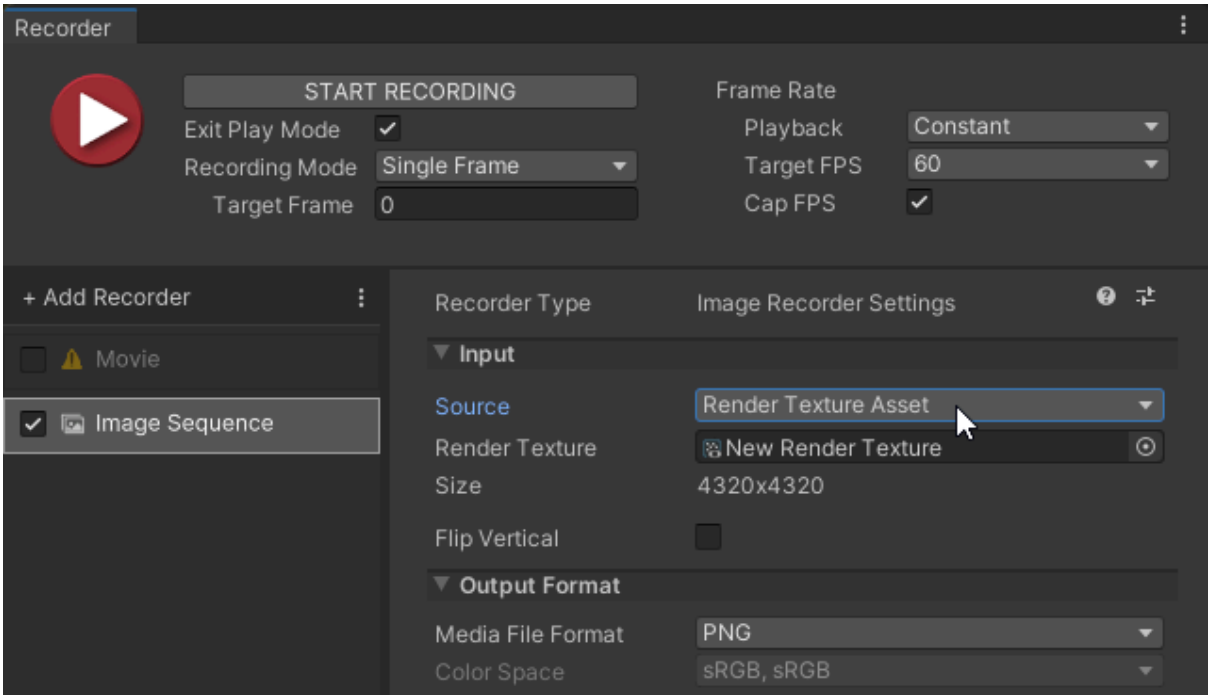
*You can select the desired **Output Resolution** (maximum 4K/2160p).
*If you need 8K/4320p, this method won't work, use Method:Best Quality instead

Method: Best Quality

This is the method we(NiloToon developers) always use to produce the best quality alpha png, it is a bit **complex** to set up, but if you aim for **best quality alpha png**, this is a practical solution.

In Recorder, use source = **Render Texture Asset** if you want the following pros & cons:

- **best MSAA quality (e.g., MSAA 4x or 8x)**
- **Unlimited resolution above 4K (e.g., 8K)**
- **More complex setup**

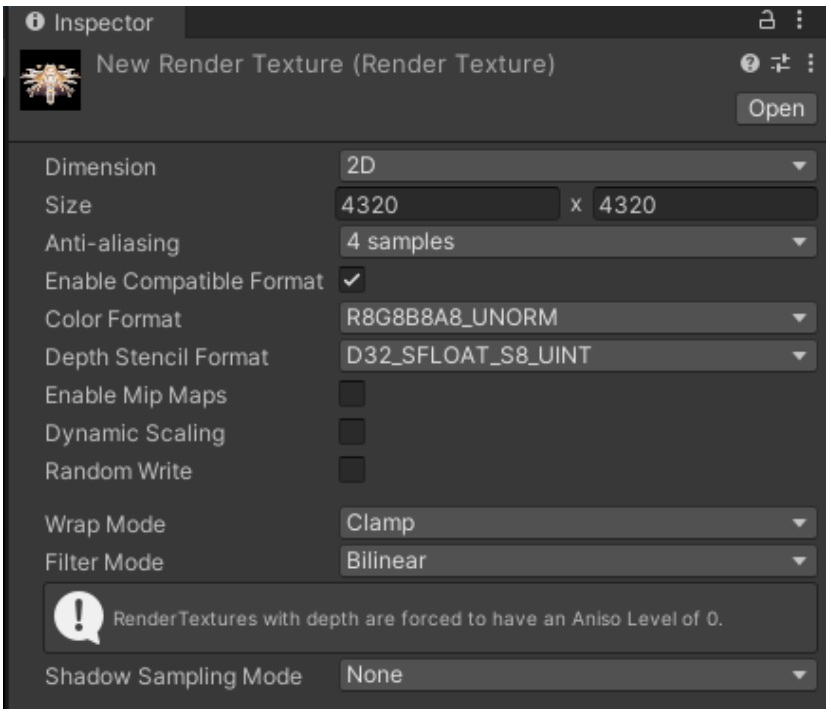


Set up

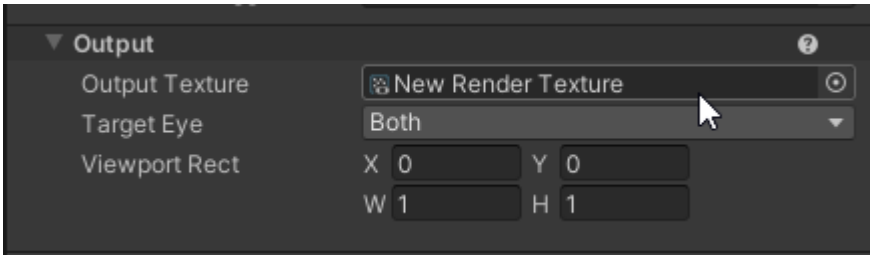
To produce the best alpha png, follow these steps:

1.Create a **new RenderTexture** in project window
2.Set RenderTexture's **Size(e.g., 4320x4320, 3240x6480)**, **Anti-aliasing (e.g, 4 samples MSAA)**. You can keep the RenderTexture's **other settings** such as **format** as **default**

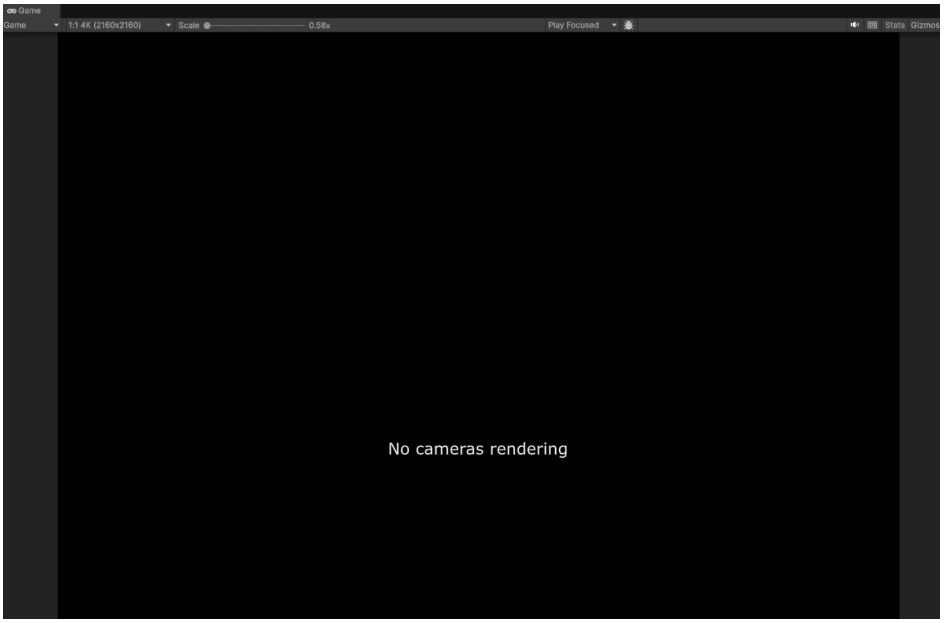
*It is recommended to pick a high resolution, because RenderScale will be ignored when using this method
*see the image below for example settings of RenderTexture



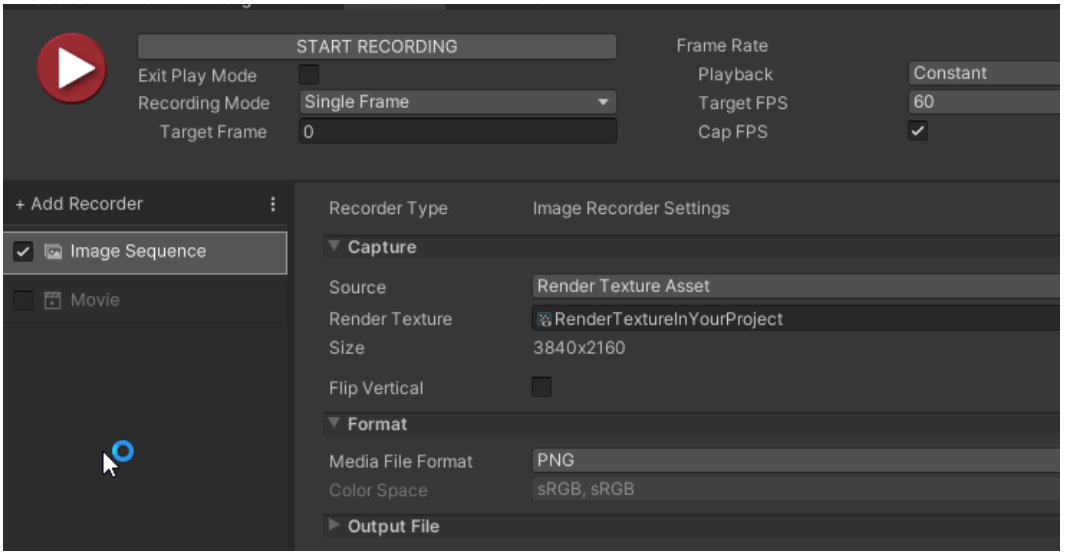
3.Assign this RenderTexture to your **active camera**'s **Output > Output Texture** slot.



It is normal that you can't see anything in Game window after this step, don't worry



4.Assign the same RenderTexture to **Recorder's RenderTexture** slot also, the **Capture > Source** should be **Render Texture Asset**



5. Make sure the **RenderTexture and URP asset** has **the same MSAA** setting (e.g., 4X MSAA), else your result png may display weird lighting/color.

Using the above settings, then click **START RECORDING** in the **Recorder** window, and your alpha png should be produced with good quality Classic Outline MSAA in a high resolution (e.g., 4320x4320).

Here is a summary of example setting:

Broken RenderTexture

Rendering **4320x4320 (8K) + MSAA 4x** will require a good GPU to support it, if the RenderTexture's result looks completely wrong when using such a high resolution and high MSAA, usually a **red error** will be produced in **editor's console**:

```
Assertion failed on expression: 'rs && rs->m_Texture && rs->m_SRView'
```

In this situation, you can:

- cut the **Size** in setting in the RenderTexture if there are many empty space in png wasted, which allows you to cut out unneeded empty area(e.g., use 2160x4320 for vertical character png, or use 4320x2160 for horizontal character png)
- lower **Anti-aliasing**
- lower **Size**

8K->4K in PS/CSP

You can downsample the big **8K(4320p) png** back to a **4K(2160p) png** in software like **Photoshop** or **Clip Studio Paint**, it will generate a much better 4K png than rendering the 4K png directly in Unity Editor.
*This step is similar to a manual SSAA(Super sampling), you can apply this to recording video also if your computer has a good enough GPU to handle this.

Postprocess(bloom)

If you need post process (e.g., character bloom) in the result png, you will need a “2png” method:

- **png-A**: a png without alpha, with all the postprocess as usual
- **png-B**: a png with alpha, but without postprocess
-

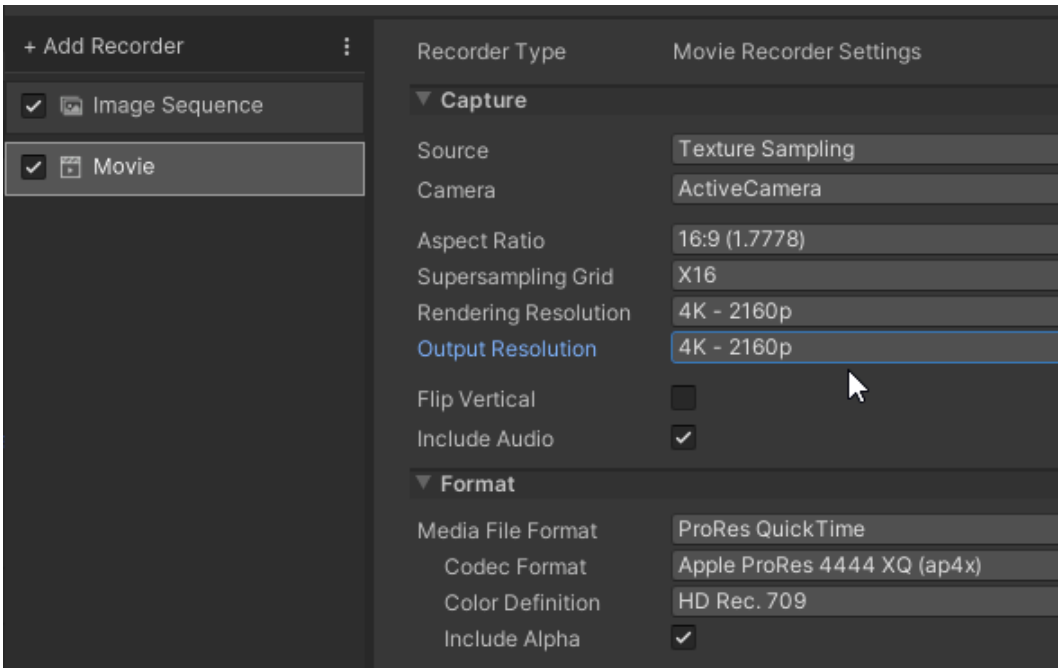
Then you can manually composite the final png using **png-A** as **rgb** color, and **png-B** as **alpha**, using tools like photoshop.

Output alpha .mov:

Everything is similar to the above Output alpha .png, so we will skip some details, you can read the Output alpha .png section, and apply the method to .mov, everything is the same.

Method: Quick & Easy

In Recorder, use source = **Texture Sampling**

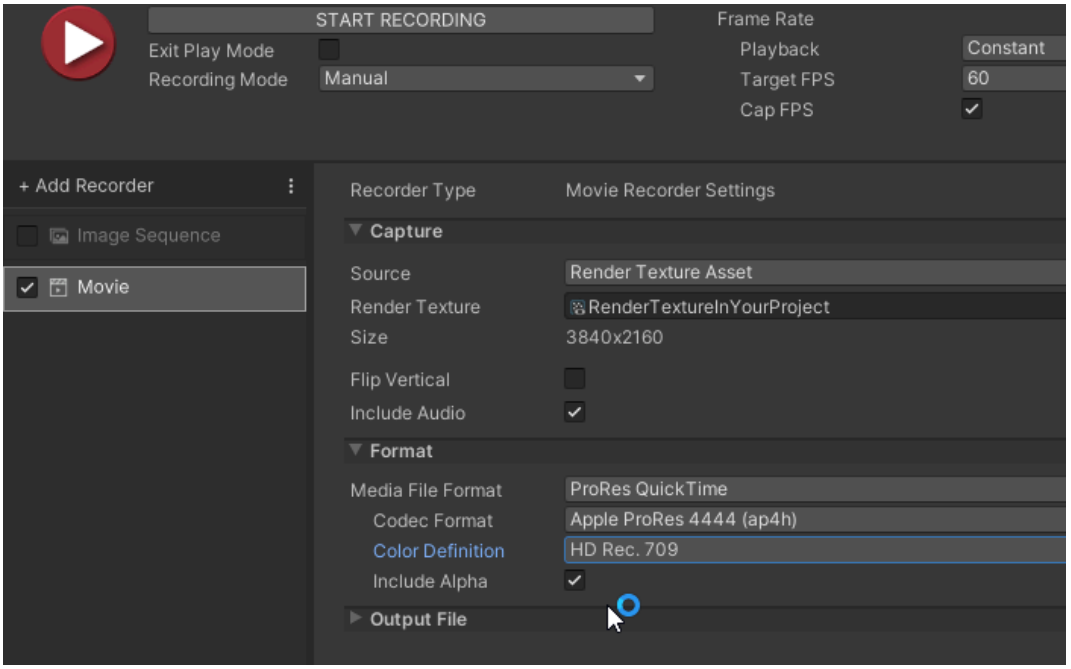


Method: Best Quality

In Recorder, use source = **Render Texture Asset**, you need to assign the same RenderTexture to your Camera's Output Texture also.

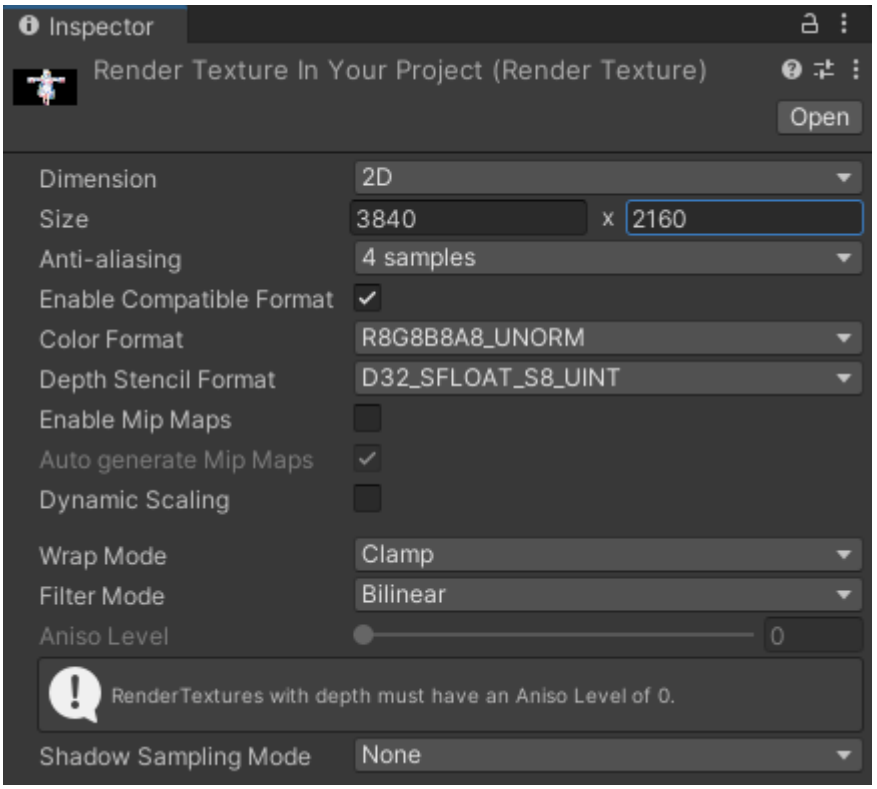
*Make sure **Codec Format = Apple ProRes 4444**, it is a video codec that can store alpha data

*Make sure to **enable Include Alpha**



Source=RenderTextureAsset

In Recorder, when using **Source = Render Texture Asset** mode, URP's rendering will not use any **RenderScale / MSAA** settings, so you should use a much higher RenderTexture **Size and Anti-aliasing** to output a high enough quality result. (for example, use at least **3840x2160+4xMSAA RenderTexture** for a **1920x1080** final display, this can produce a high enough quality image/videos)



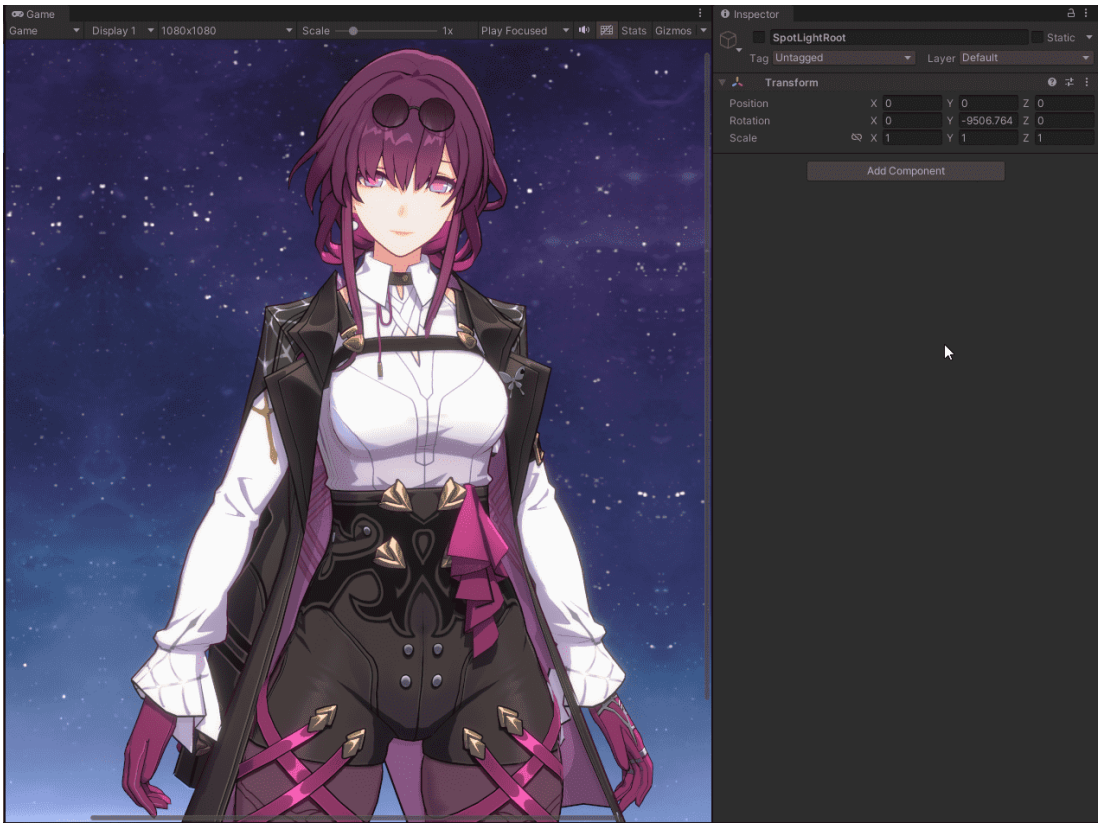
Spot/point as rim light

Try using **NiloCinematicRimLightVolume** in volume, which will turn all additional lights(spot light / point light / additional directional light) into **rim light** for **NiloToonCharacters** materials, and also unlock additional light's max contribution, so rim light can become very bright.
When combined with NiloToon's bloom, you can produce great rim light result using additional lights with different color, position and direction

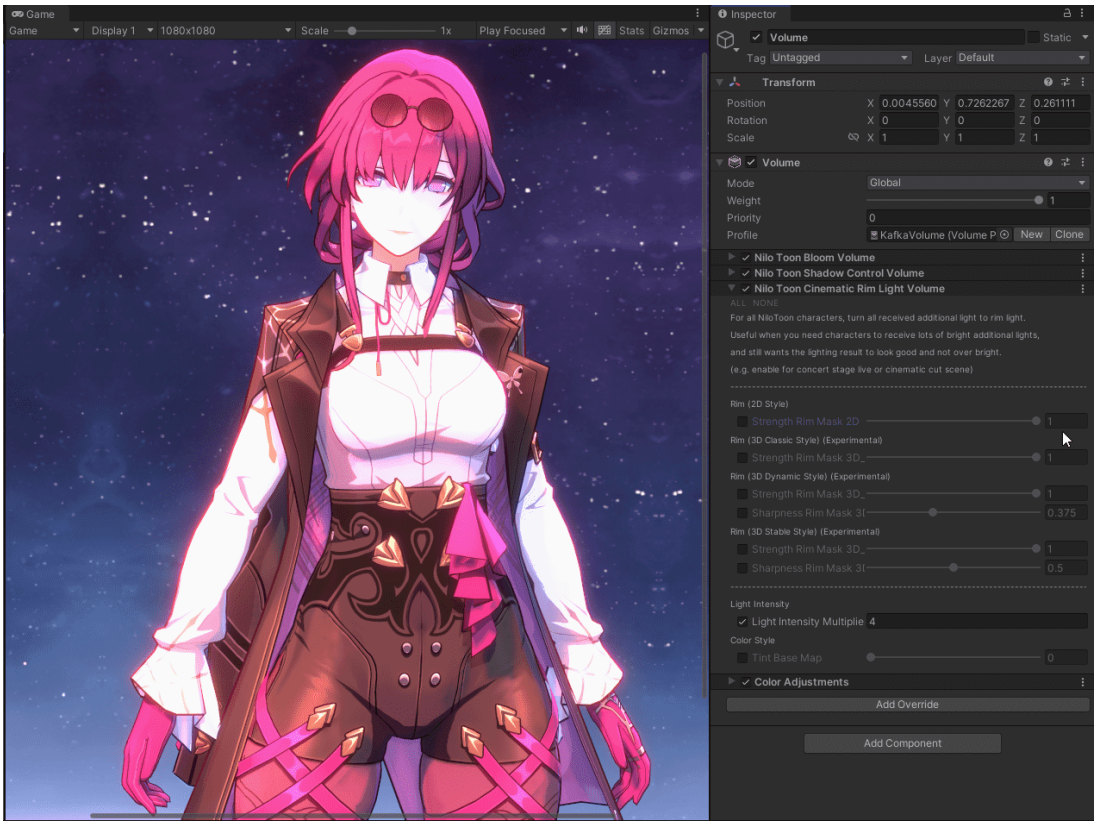
*when using multiple directional lights, only 1 of them will be treated as Main Light, and others will be treated as additional light by URP.
If the **sun light** is assigned, that light will be the main light,
else the **highest intensity** directional light will be treated as Main Light, other directional lights will be treated as additional light.

See Additional Light Settings for details and examples.

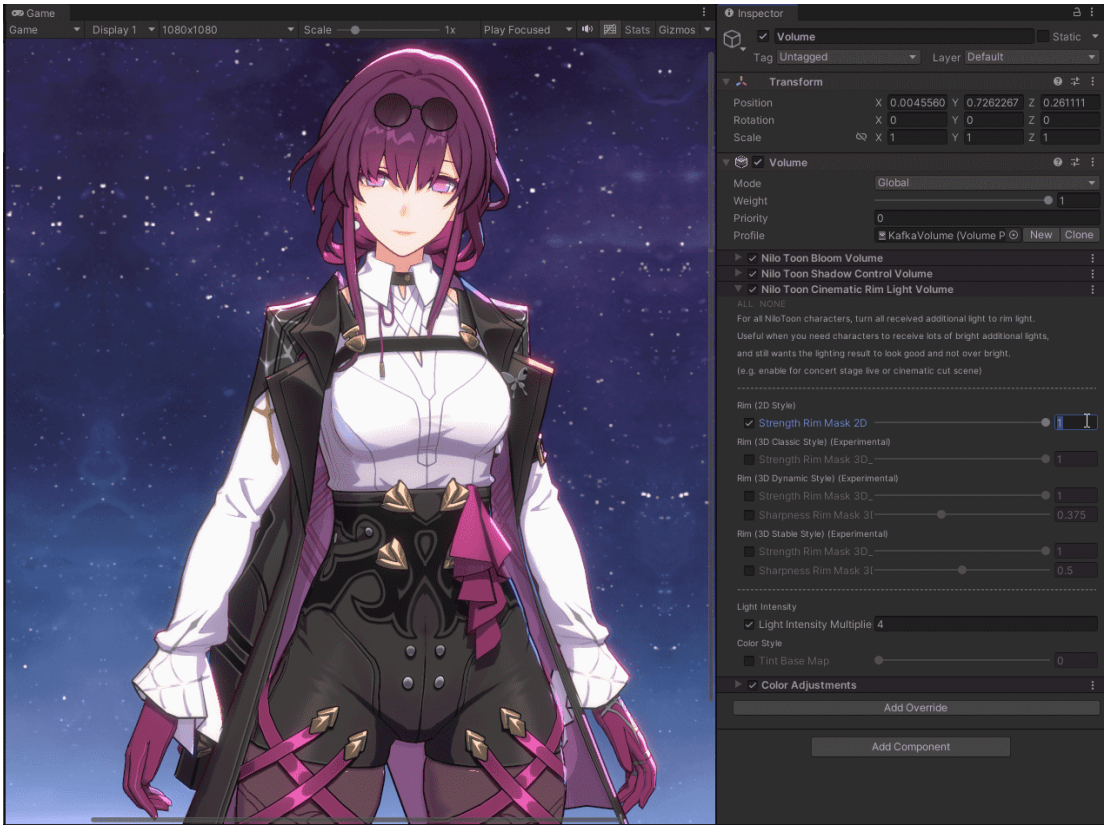
Before adding any additional light:

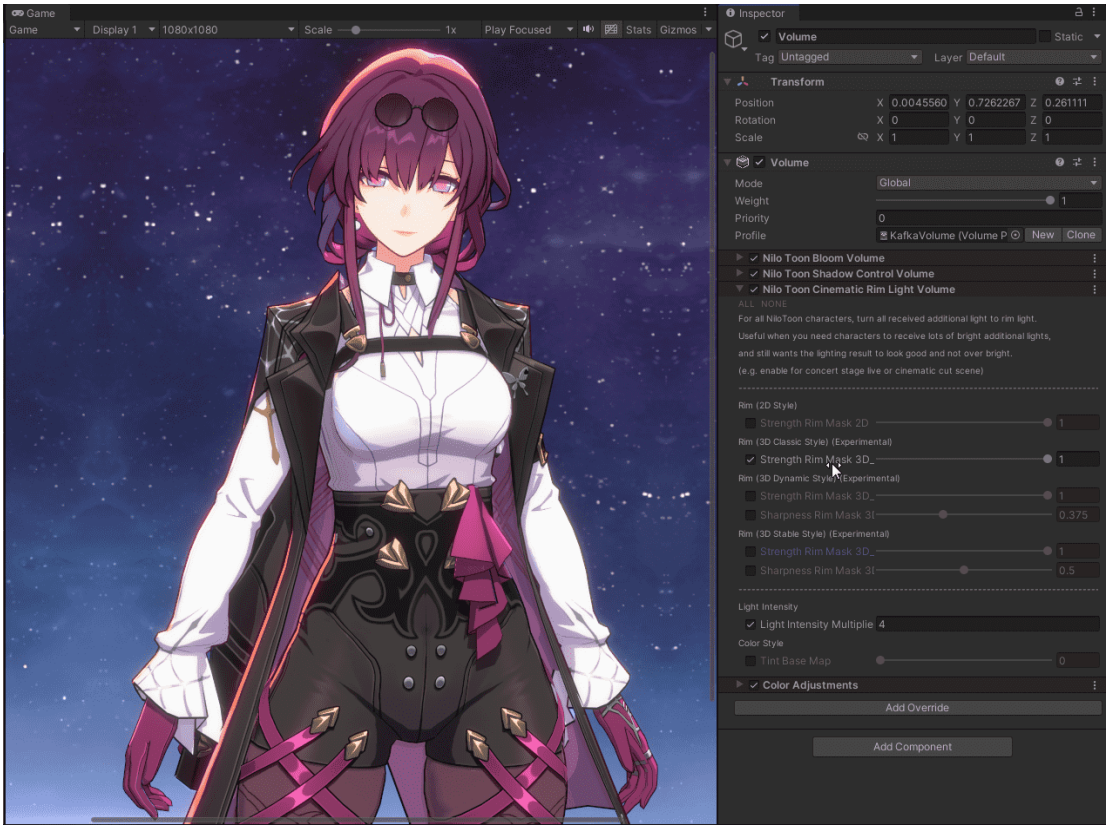


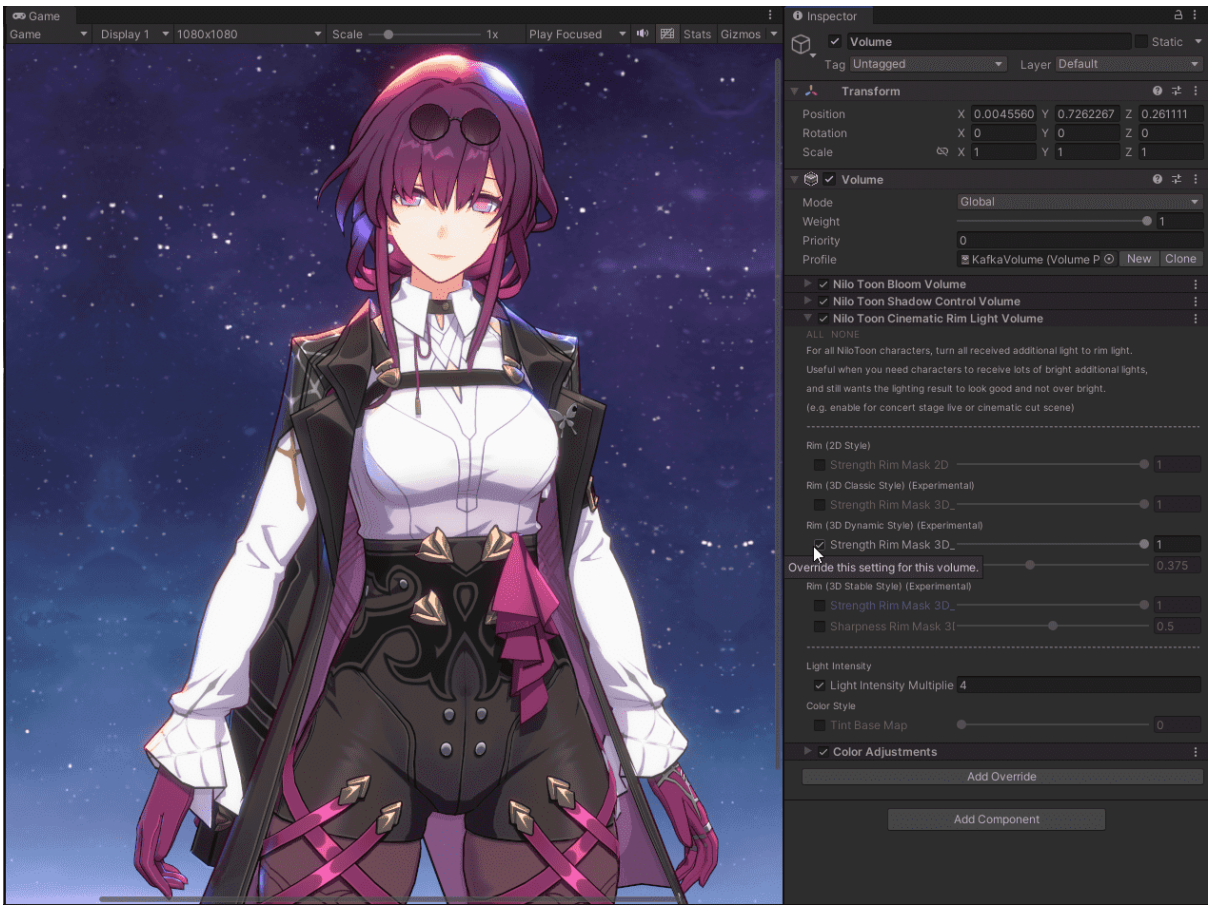
After adding some additional light, **without** **NiloCinematicRimLightVolume**, the result is too bright:

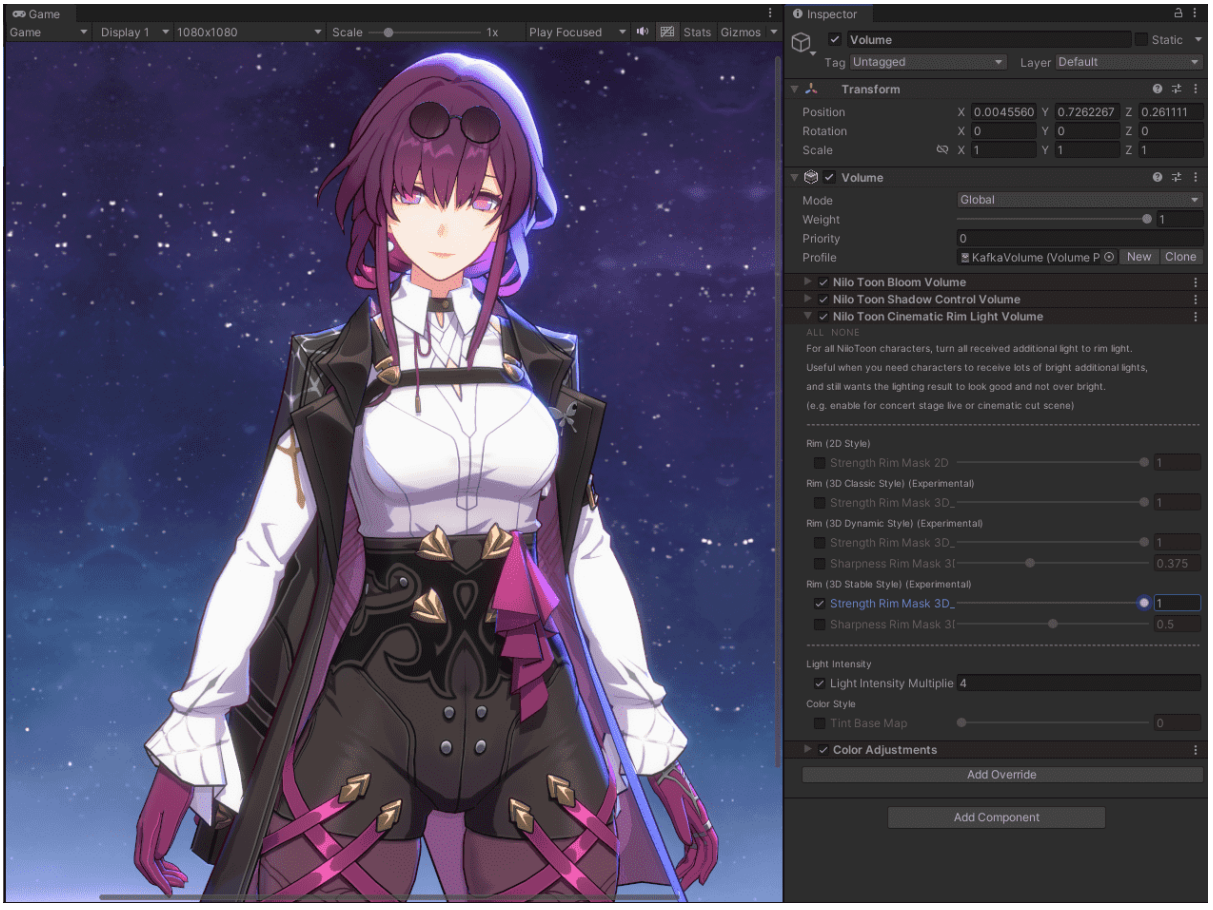


After adding **NiloCinematicRimLightVolume**, below images show different settings of this volume, notice that the over bright additional light is now become a bright rim light

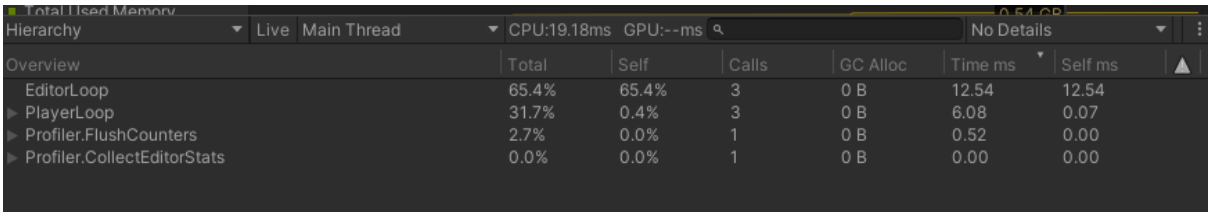





Game
Display 1
1080x1080
Scale
1x
Play Focused
Stats
Gizmos
Inspector
Volume
Static
Tag
Untagged
Layer
Default
Transform
Position
Rotation
Scale
Volume
Mode
Global
Weight
Priority
Profile
KafkaVolume (Volume P
New
Clone
Nilo Toon Bloom Volume
Nilo Toon Shadow Control Volume
Nilo Toon Cinematic Rim Light Volume
ALL NONE
For all NiloToon characters, turn all received additional light to rim light.
Useful when you need characters to receive lots of bright additional lights,
and still wants the lighting result to look good and not over bright.
(e.g. enable for concert stage live or cinematic cut scene)
Rim (2D Style)
Strength Rim Mask 2D
Rim (3D Classic Style) (Experimental)
Strength Rim Mask 3D_
Rim (3D Dynamic Style) (Experimental)
Strength Rim Mask 3D_
Override this setting for this volume.
Sharpness Rim Mask 3D_
Rim (3D Stable Style) (Experimental)
Strength Rim Mask 3D_
Sharpness Rim Mask 3D_
Light Intensity
Light Intensity Multiplie
Color Style
Tint Base Map
Color Adjustments
Add Override
Add Component

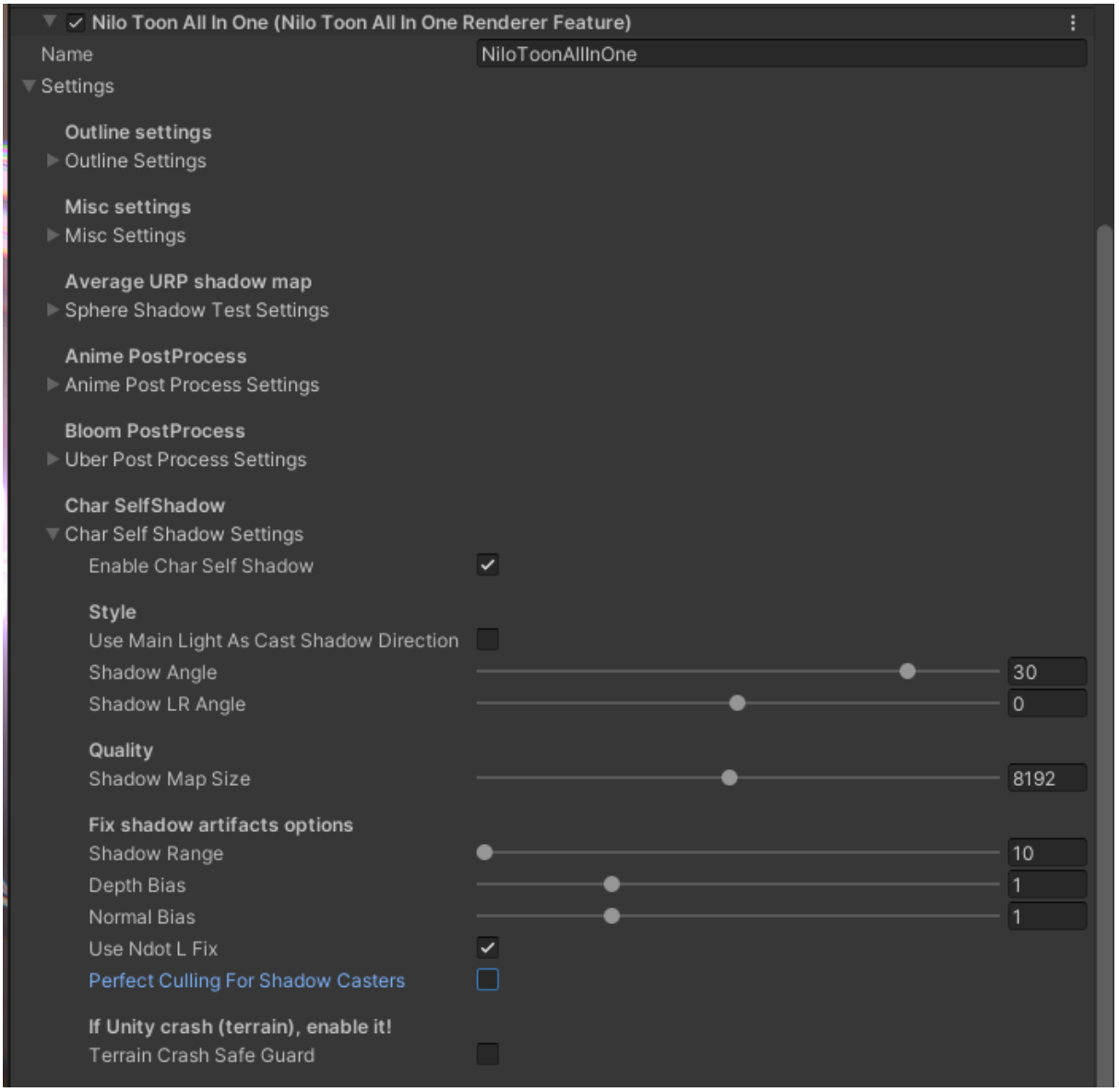

Game
Display 1
1080x1080
Scale
1x
Play Focused
Stats
Gizmos
Inspector
Volume
Static
Tag
Untagged
Layer
Default
Transform
Position
Rotation
Scale
Volume
Mode
Global
Weight
Priority
Profile
KafkaVolume (Volume P
New
Clone
Nilo Toon Bloom Volume
Nilo Toon Shadow Control Volume
Nilo Toon Cinematic Rim Light Volume
ALL NONE
For all NiloToon characters, turn all received additional light to rim light.
Useful when you need characters to receive lots of bright additional lights,
and still wants the lighting result to look good and not over bright.
(e.g. enable for concert stage live or cinematic cut scene)
Rim (2D Style)
Strength Rim Mask 2D
Rim (3D Classic Style) (Experimental)
Strength Rim Mask 3D_
Rim (3D Dynamic Style) (Experimental)
Strength Rim Mask 3D_
Sharpness Rim Mask 3D_
Rim (3D Stable Style) (Experimental)
Strength Rim Mask 3D_
Sharpness Rim Mask 3D_
Light Intensity
Light Intensity Multiplie
Color Style
Tint Base Map
Color Adjustments
Add Override
Add Component

Achieve 0 GC alloc



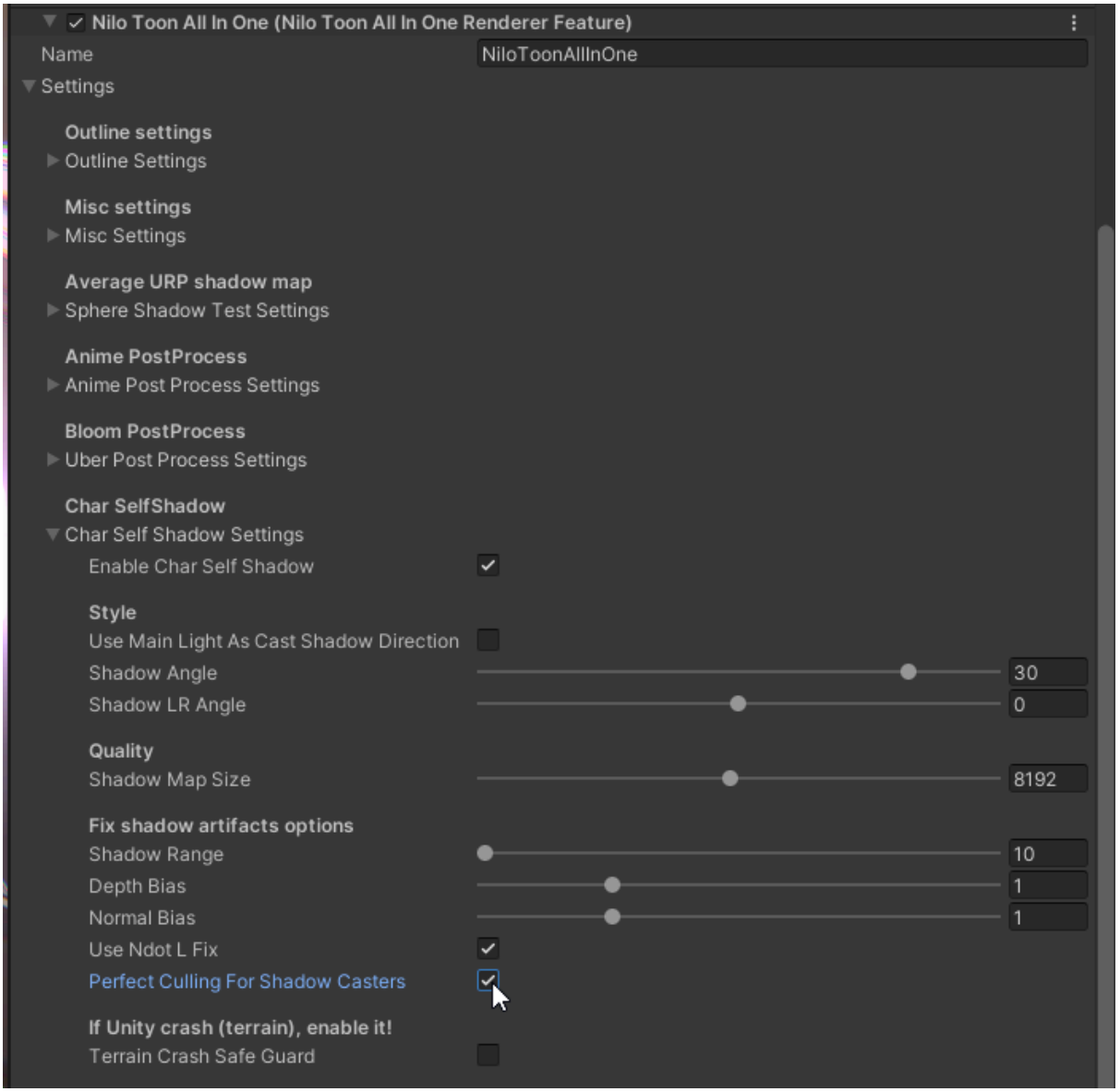
If your project uses **Terrain**, in all NiloToon's renderer feature,

- turn off **Perfect Culling For Shadow Casters**, this will prevent terrain crash
- turn off **Terrain Crash Safe Guard**, this will remove GC alloc



If your project don’t use **terrain**,

- turn on **Perfect Culling For Shadow Casters**, this improves shadow quality
- turn off **Terrain Crash Safe Guard**, this will remove GC alloc



VRM mat resets to MToon

After
- deleting the Library folder
- Reimport All
- export .unitypackage to a new project

All my vrm materials are reset to MToon original material(magenta error color in URP), How to stop it?

It is the **UnVRM(0.x)**'s design, there is no option to stop it.

Solution A:
Use Windows > NiloToon > Create prefab variant and materials
This will:
1) clone all materials from the selected prefab to a new folder
2) change these materials to NiloToon
3) create a prefab variant from the selected prefab
4) assign those cloned materials to your new prefab variant as an override.

This way, the original vrm materials are not being used in your prefab variant anymore, the override materials that you cloned will be used instead. So no matter how UniVRM edit / reimport / regenerate the original vrm materials, it won't affect your prefab variant anymore.

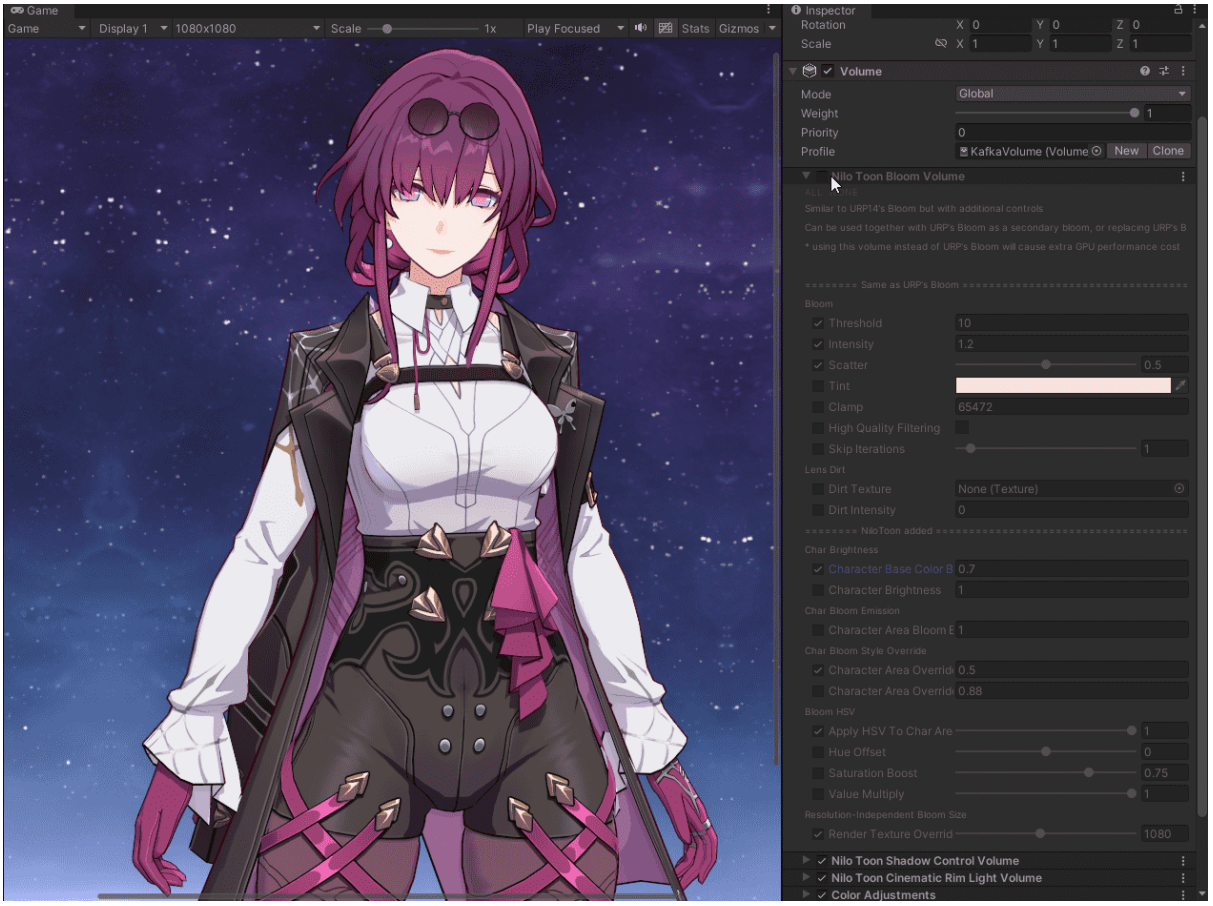
Solution B:
Just upgrade to UniVRM **1.0**, then you can use .vrm files just like any .fbx, and don't worry about the material reset to MToon again
https://github.com/vrm-c/UniVRM/releases

Solution C (Not recommended):
- rename the extension of .vrm to something else (e.g., character**.vrmDisabled** instead of **.vrm**)
- delete or not including the .vrm file on .unitypackage export
This way the UniVRM reimport code for .vrm file will not be triggered, since there are no .vrm files anymore

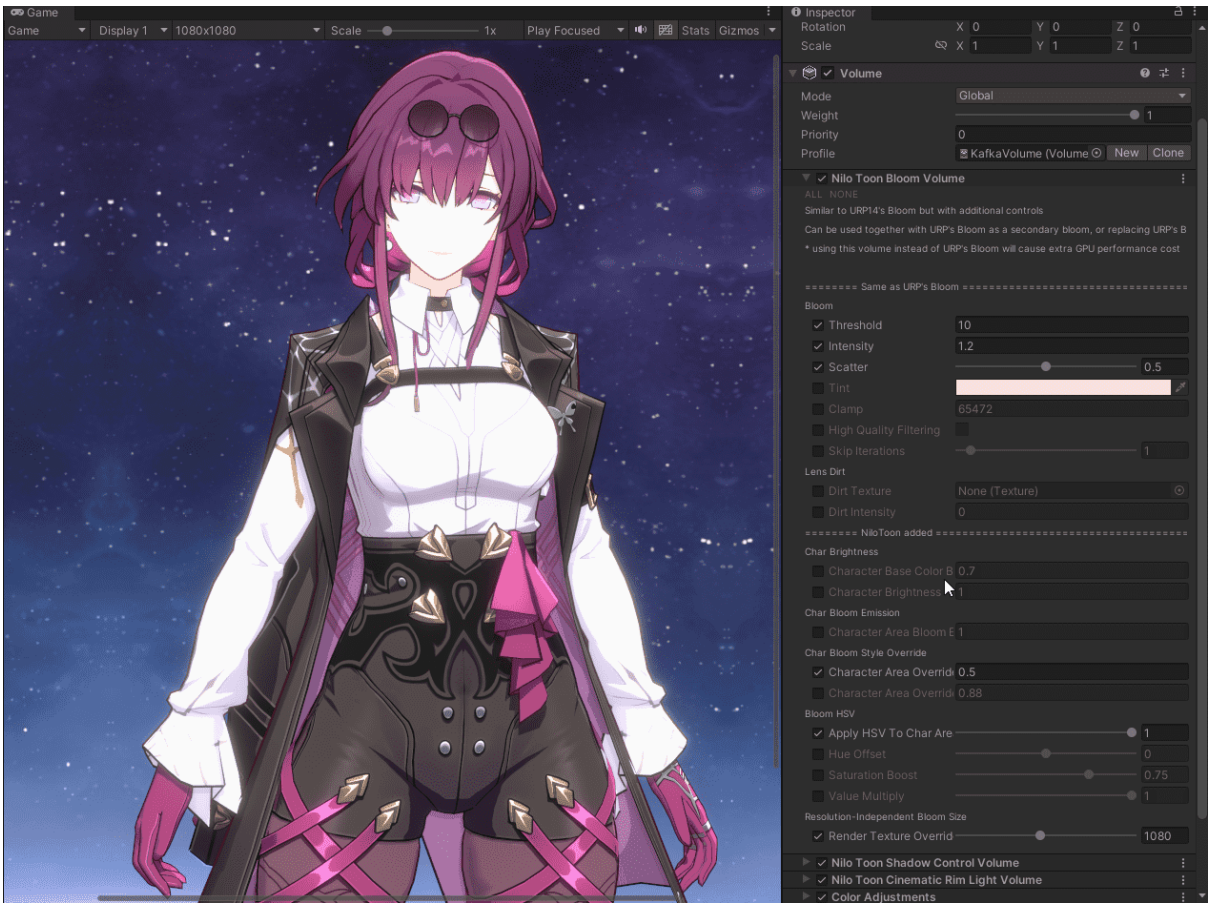
Solution D (Not recommended):
export the materials as a .unitypackage, reimport it again every time you see error shader

High saturation char bloom

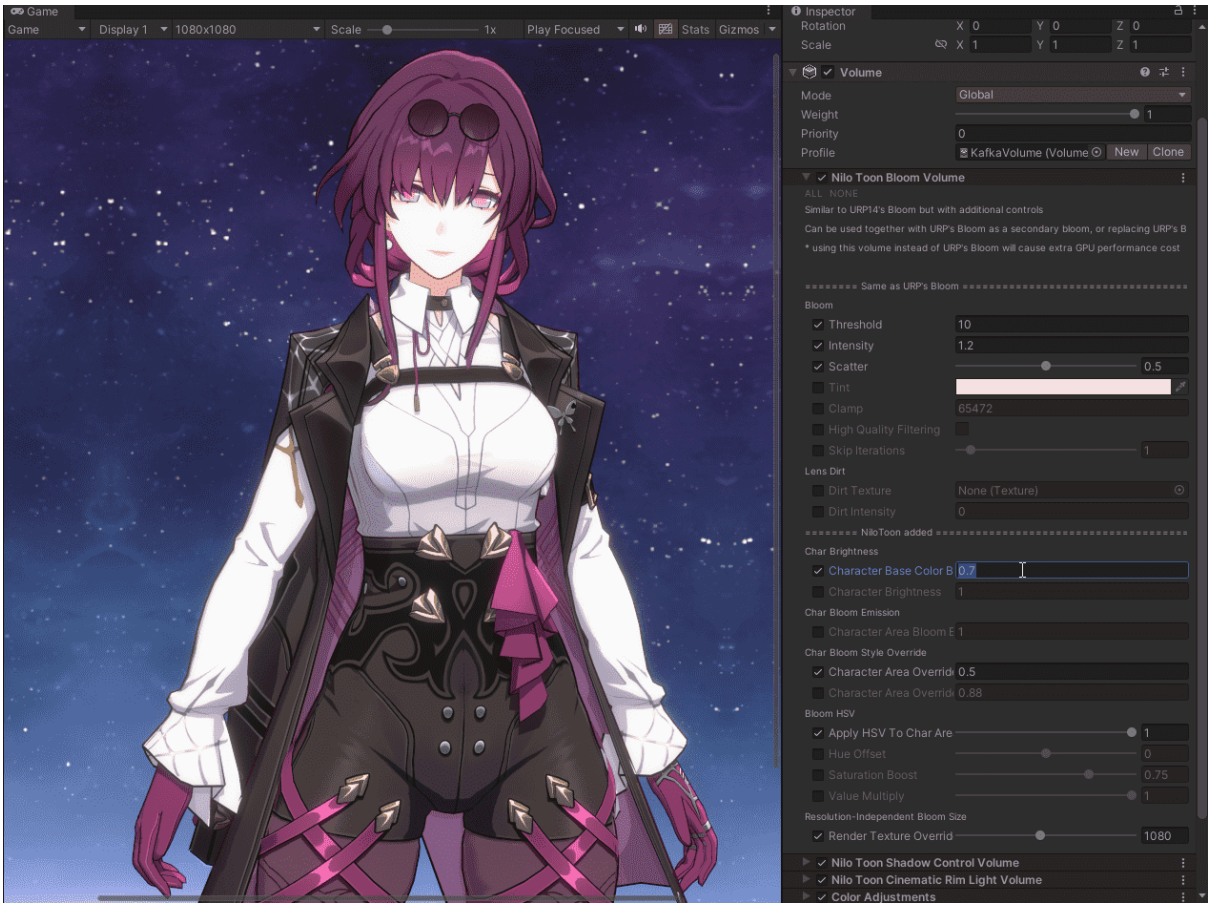
Use **NiloToon Bloom Volume**, set **saturation boost** higher, it will result in a more colorful bloom, instead of white bloom. It is critical to use this feature, if you want to create specific bloom color (skin with reddish bloom color)



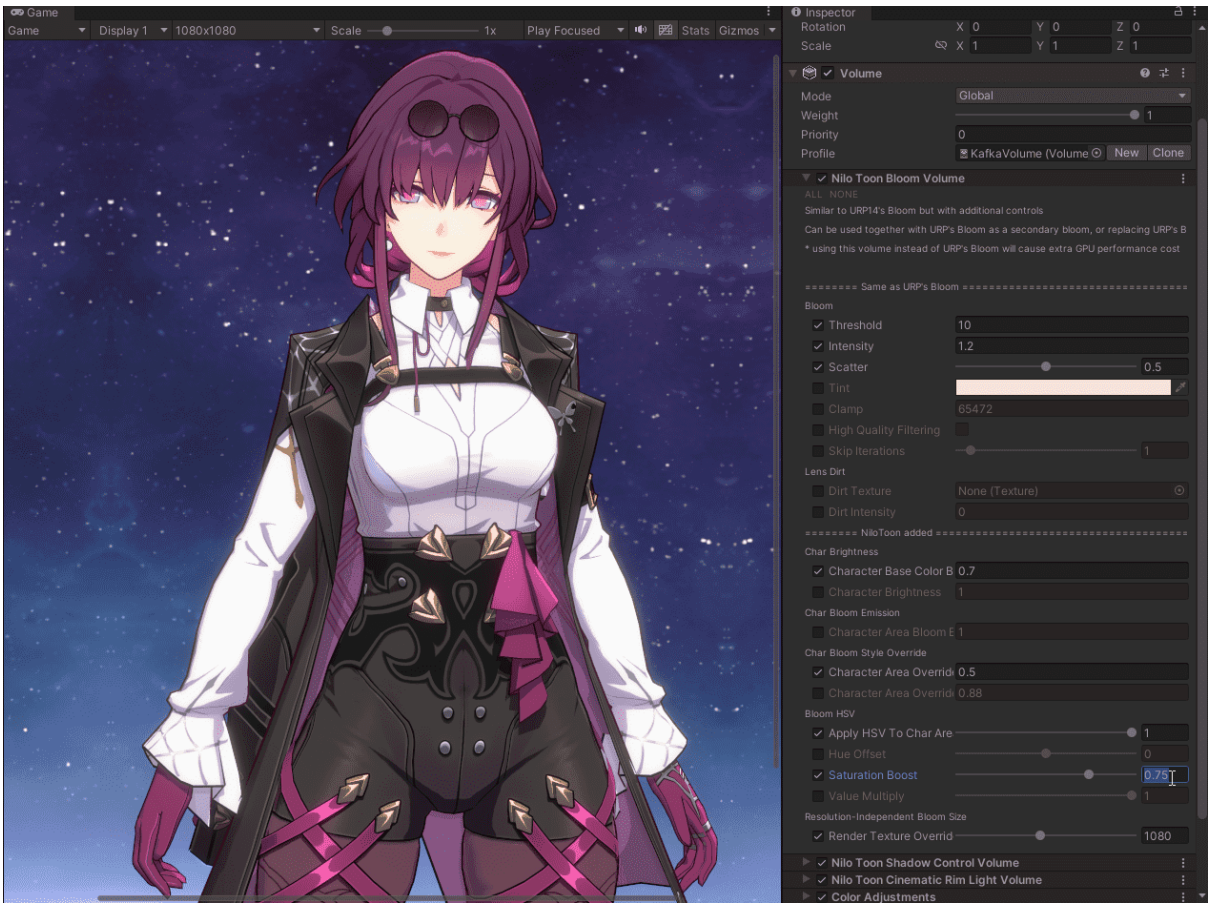
*Before enabling any NiloToonBloom



*After enabling NiloToonBloom, it is too bright due to Bloom is additive



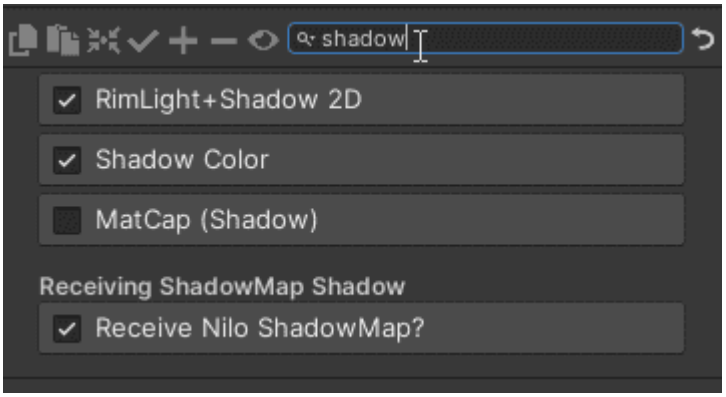
*Control the input brightness of character using **Character BaseColor Multiplier**, it will cancel out the additive brightness due to additive bloom, making the character having bloom but still not over bright



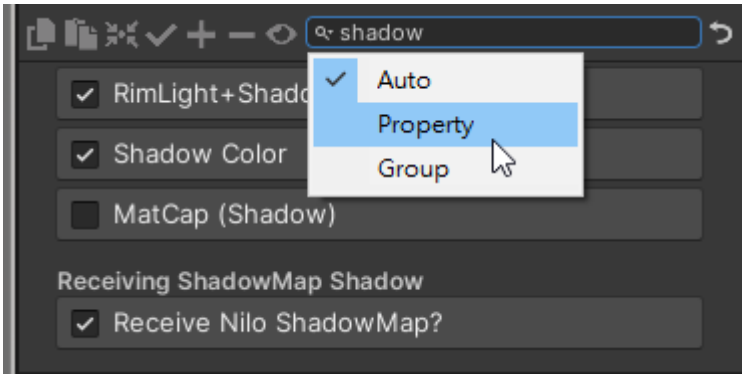
*Finally, increase the **saturate Boost** of Bloom, this will allow character skin(usually in pink~orange hue) to emit a reddish bloom color, instead of a white bloom color, in many situations a reddish bloom on skin will be much better than a white bloom on skin

Search/Filter in Material UI

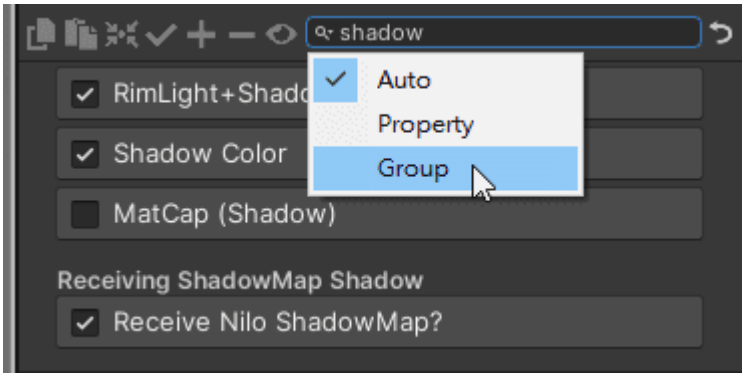
Use NiloToonCharacter material's **search bar**, type the group or property that you are looking for(e.g. ShadowColor, Outline, rim light, specular……), it will filter the group/properties for you, showing only those that you are looking for.



if you want to search a property, use **Property** search mode



if you want to search a group, use **Group** search mode

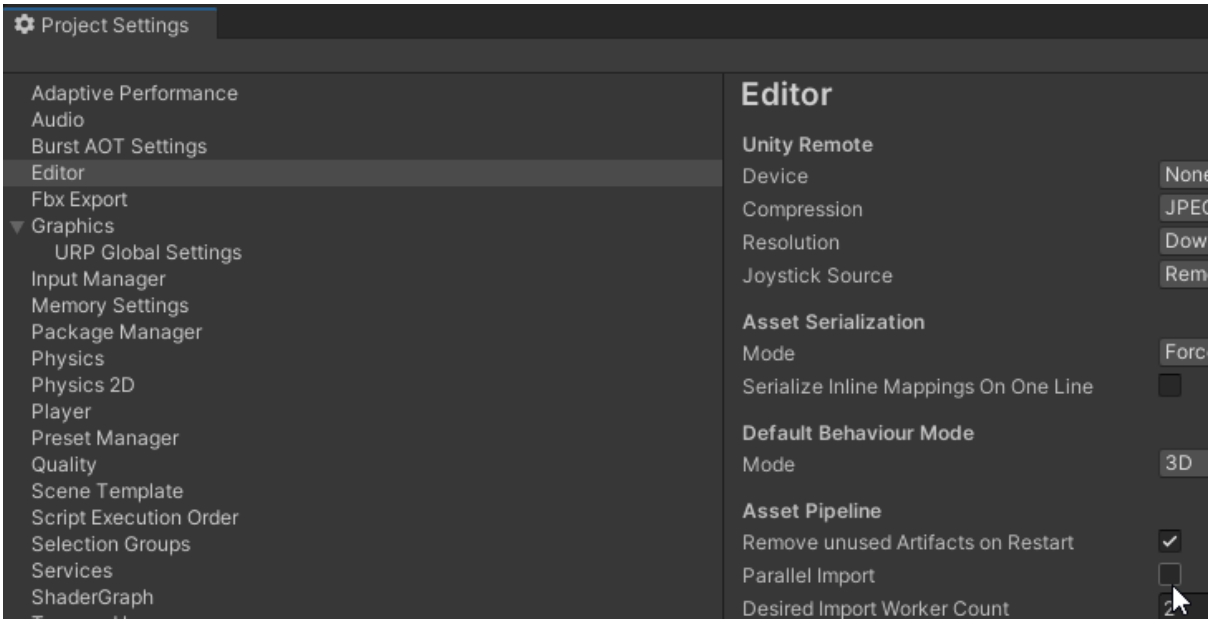


*By default it will search for group first, if no matches, it will search properties instead

Parallel import corrupts prefab

Parallel import introduction:
Unity - Manual: Importing assets simultaneously

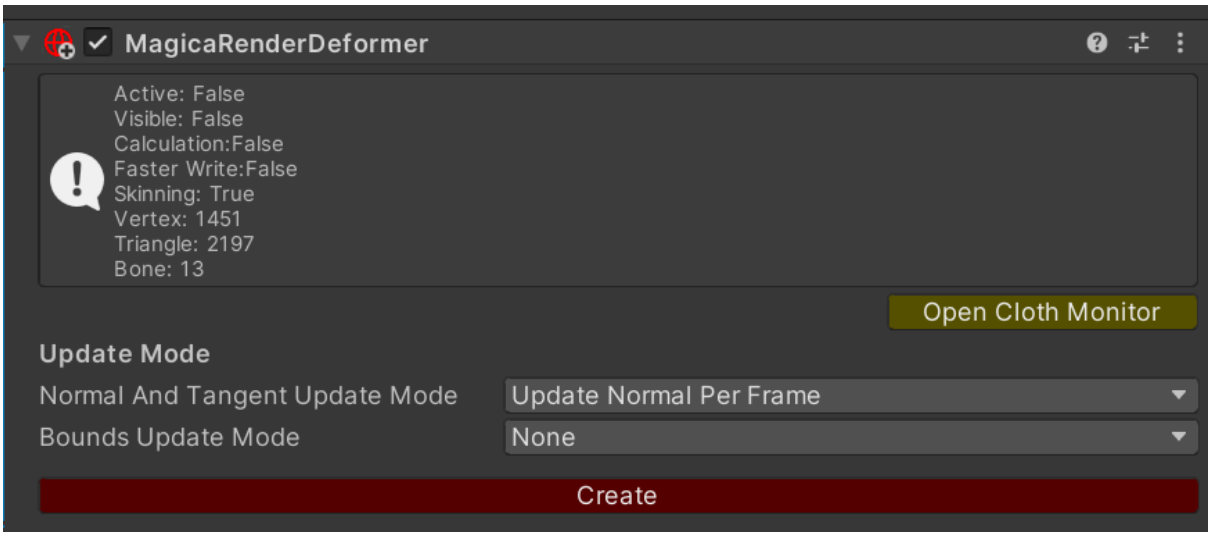
NiloToonURP's model(fbx) import requires **single-thread**(since it will create new temporary .fbx Assets for calculating smoothed normal for outline), so **ParallelImport** will not work. If enabling **Parallel Import** produced any error, or led to the prefab corruption or missing prefab, you need to **turn off** **Parallel Import** in **Project > Editor > Asset Pipeline > Parallel Import = off**



After turning **Parallel Import** off, you need to manual click **reimport** for the **fbx** again to fix the corrupted prefab

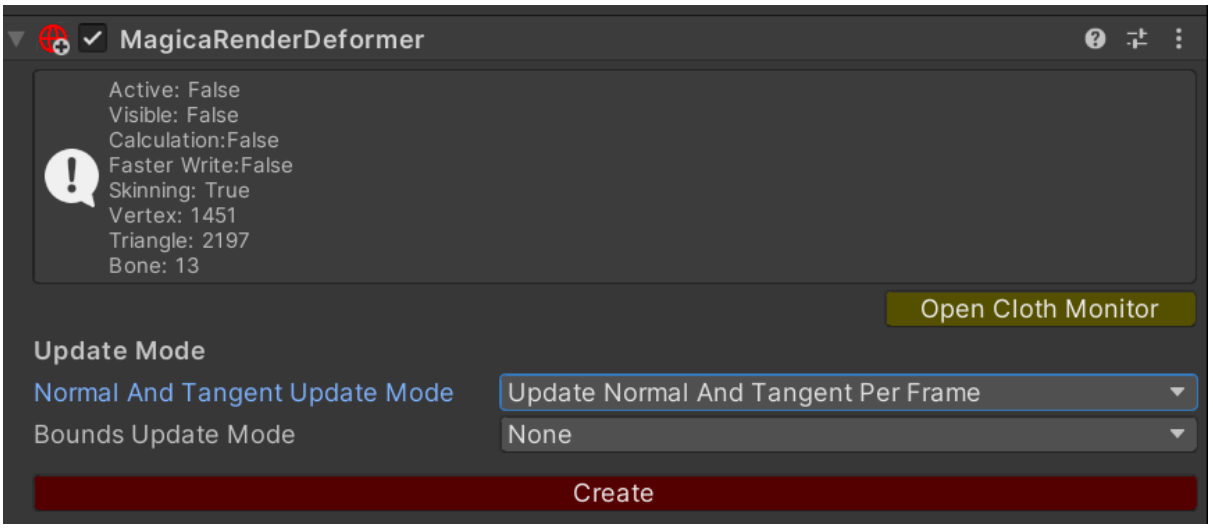
MagicaCloth broke outline

In MagicaCloth/MagicaCloth2, the default value for **Normal And Tangent Update Mode** is **Update Normal Per Frame**, where **tangent** is not updated, but NiloToon still tries to use the invalid tangent data for outline's calculation in the shader, so you need to override it to **Update Normal And Tangent Per Frame**.



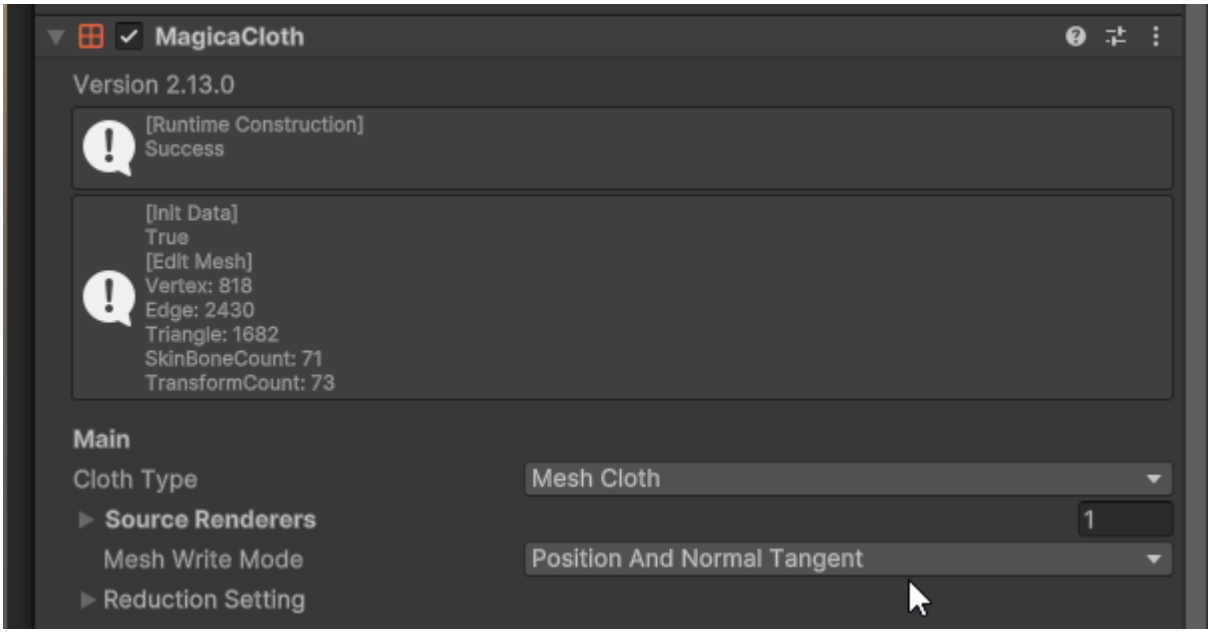
*2 images above show the **wrong** MagicaCloth settings, resulting **wrong outline**

If you change the deformer setting to **Update Normal And Tangent Per Frame**, both the Cloth simulation and NiloToon will work correctly together.



*2 images above show the good MagicaCloth settings, resulting good outline

For **Magica Cloth2**, the settings are here:



The problem occurred because the tangent was not cloned to the mesh for cloth simulation, and NiloToon's classic outline requires tangent.

In other words, for **SkinnedMesh** with **MagicaCloth** and NiloToon, **Update Normal And Tangent Per Frame** is always required.

Anti-aliasing (AA) guide

TL;DR - best AA?

Summary:

- For **RenderScale <= 1** and **NVIDIA RTX GPU** support only = **DLSS** + MSAA
- For **RenderScale <= 1** and **all GPUs** support(e.g. AMD / console) = **FSR3** + MSAA
- For **RenderScale > 1** and no unacceptable **TAA ghosting & blur** = **TAA**
- For **RenderScale > 1** and has unacceptable **TAA ghosting & blur** = MSAA*

Note: This list excluded Unity6 **STP** and Intel's **XeSS2**, not because they are bad (actually they should be similar or better than FSR3), just we didn't use them often to give a conclusion.

AA Quality for **overall graphics**:

- For **RenderScale <= 1**, [Best] **DLSS** >> **FSR3** > **TAA** >>> MSAA* [Worst]
- For **RenderScale > 1**, [Best] **TAA** >>> MSAA* [Worst]

AA Quality for **Classic Outline only**:

- For **RenderScale <= 1**, [Sharpest] MSAA* >> **DLSS** > **FSR3** >>>>> **TAA** [Blurriest]
- For **RenderScale > 1**, [Sharpest] MSAA* >>>>>>>> **TAA** [Blurriest]

Note: **DLSS** and **FSR3** are not usable when **RenderScale > 1**
Note: the above **MSAA*** means MSAA + FXAA/SMAA/Off

Our AA choice

In real projects (mainly VTuber 3D live PC projects for us), which focus on visual quality, running on high-end PC, we choose AA in the following order:

1. **MSAA = 4x/8x, RenderScale = 2, Camera AA = off** as our default AA choice first, due to MSAA never producing **ghosting / blurriness,** and MSAA doesn't require all opaque shaders having a correct **depth and motion vector pass**. We may also enable CameraAA = **FXAA/SMAA** if it helps, **FXAA/SMAA** may improve 2D rim light's AA when compared to **Camera AA = Off**.

2. The above solution(1) will likely solve all **Classic Outline**'s AA problem due to **MSAA**, but if NiloToon character's **rim light, screen space outline**, PBR **specular** or any small objects have **shading aliasing** due to low game window **resolution**, low **RenderScale** or **small renderer** on screen, we will switch to use **MSAA = off, RenderScale = 2, Camera AA = TAA** to stop all kind of aliasing. **TAA** will likely solve most of the aliasing problems, but TAA comes with disadvantages of **ghosting** and **blurriness** due to the nature of TAA (TAA is temporal AA, which mixes the new frame with history frames to produce AA).

3. At last, if the above solution(1) or solution(2) are both not producing good enough **graphic results**, or not producing good enough **performance(fps)** due to RenderScale = 2 when rendering 4K/8K, we will switch to use **DLSS** 's **Native AA (MSAA = off~8x, RenderScale = 1, Camera AA = off)**. This will likely solve all AA problems, but they will also lock your **RenderScale** to **<=**

1, so you can't use **RenderScale** = **2** to boost graphics quality / sharpness anymore.
*__DLSS__ is only suitable for **4K** or above Game Window
*For better Classic Outline quality, we may still enable **MSAA** together with **DLSS**

If you are making a 3D concert live or game, it is worth paying some time to experiment with the above 3 methods:

1. MSAA (+ FXAA / SMAA / Off), RenderScale = 2
2. TAA, RenderScale = 2
3. __DLSS__ 's **Native AA** (+ optional MSAA)

Try these 3 types of AA solutions and see which one is the best for your project.

If you don't want to try every AA, and just want to pick the best AA immediately for your project, follow the setting from Our AA choice or TLDR; Which is the best AA?

AA can improve the overall rendering massively if done right, so it is worth spending time experimenting with different AA. Make sure to validate your AA result in camera motion and with character animation playing, since DLSS will produce perfect results for static cameras and objects, it may give you the false conclusion that DLSS is always the best, although in most situations, Solution3(DLSS + MSAA) is actually the best for high fps real-time rendering 4K 3D live when target RenderScale is <= 1, giving you a great balance of graphics quality and performance.

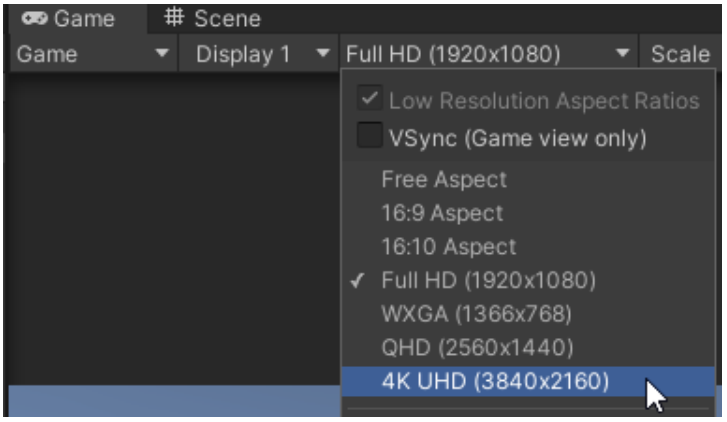
Settings that affects AA

To improve the **quality** and **anti-aliasing(AA)** of NiloToon shader's features, such as:

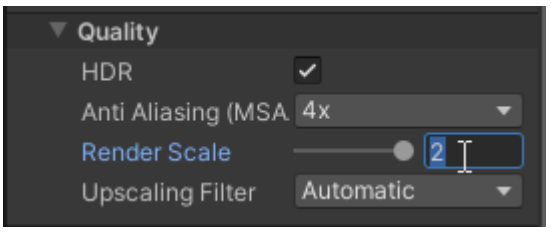
- **classic outline & screen space outline**
- **2D rim light**
- **self shadow map**
- **specular highlights**

you can set a higher value for the following settings to improve quality and anti-aliasing, but doing so will also slow down the GPU rendering performance:

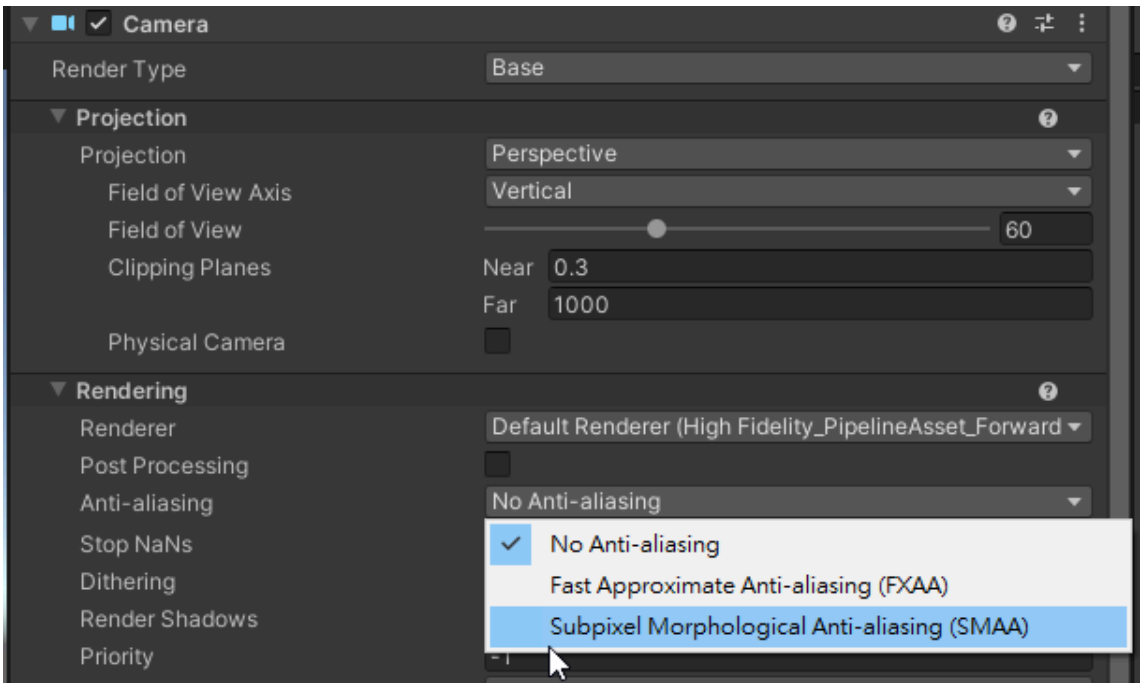
- **Game resolution** (affects **outline** and **rim light** quality)



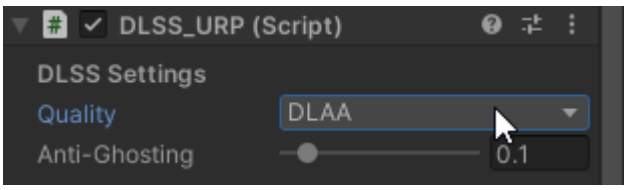
- **Render Scale** (affects **outline** and **rim light** quality)

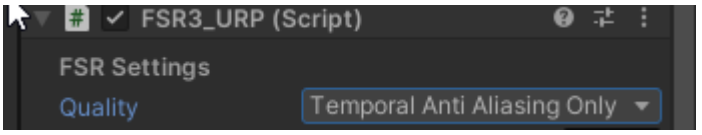


- **Camera's AA (FXAA/SMAA/TAA)** (affects **outline** and **rim light** quality)

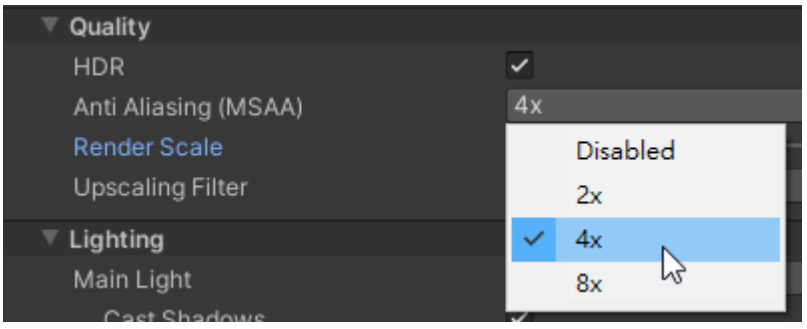


- DLSS / FSR3 **'s quality** (affects **outline** and **rim light** quality)

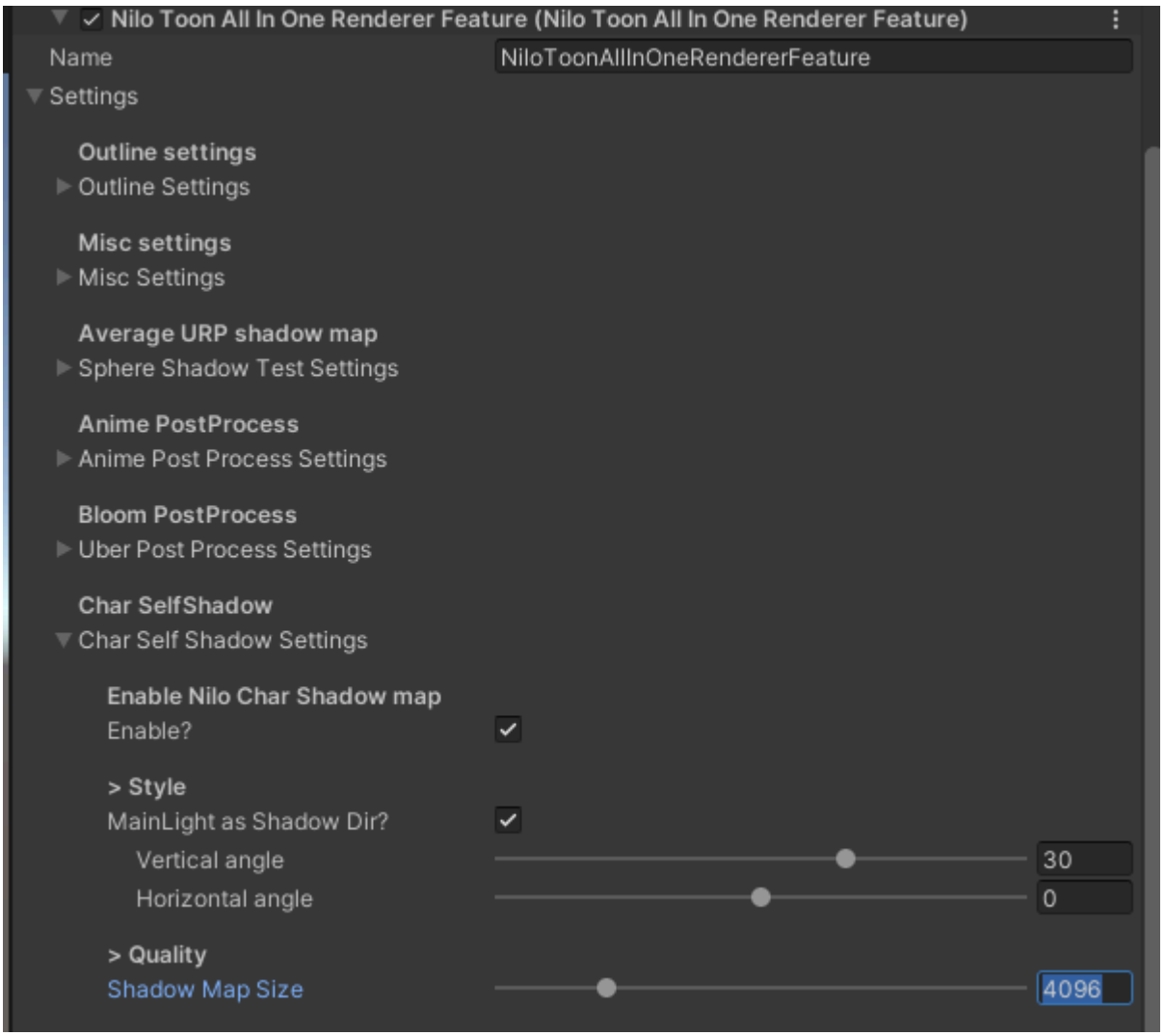




- **MSAA** (affects **classic outline** quality only, will not improve **screen space outline / rim light / specular aliasing etc**)



- **NiloToon's self shadow map resolution** & **soft shadow**(affects **NiloToon's self shadow map** quality only)



Unity's AA document

To have a basic understanding of Unity URP's MSAA/FXAA/SMAA/TAA, you can read this document: Anti-aliasing in the Universal Render Pipeline

External AA document

Tech Focus: Anti-Aliasing - What Is It And Why Do We Need It?

MSAA + FXAA/SMAA

In Unity URP, **FXAA/SMAA** can be used no matter **MSAA** is **on** or **off**, so you can create combinations of MSAA/FXAA/SMAA, for example:

- MSAA + FXAA
- MSAA + SMAA
- MSAA only
- FXAA only
- SMAA only
- no AA

Each AA has their strength and weakness:

- **MSAA** is the best AA for **geometry edge**(e.g., **Classic Outline**'s AA), also it will not produce any overall image blurriness, but it is extremely GPU resource intensive on high resolution like 4K or 8K, and can't improve the AA on any other aspect like **rim light**, **screen space outline, high tiling texture, specular highlight**
- **FXAA** is the **fastest AA**, but overall **blurrier** than SMAA/MSAA
- **SMAA** is **slower** than FXAA, but overall **sharper** than FXAA, although SMAA's anti-aliasing is not very helpful for outline

*If you mainly want to improve **Classic outline's AA**, within these 3 options (MSAA/FXAA/SMAA), only **MSAA** is useful, **FXAA/SMAA** are not helpful for Classic outline's AA

TAA vs MSAA

In Unity URP, **TAA** can only be used when **MSAA** is **off**, so you can only use either **TAA** or **MSAA**, but not both together.

When comparing **TAA** with **MSAA**,

- **Overall image AA**: **TAA** is better, since TAA can handle any kind of aliasing, while MSAA can only handle geometry aliasing (MSAA is only good at **geometry edge, classic outline**'s AA, but fails in all other types of aliasing problems)
- **Classic outline AA quality**: **MSAA** is the best, **TAA** fails in fast motion
- **Screen space outline AA quality**: **TAA** is better, since MSAA can't improve it
- **Rim light AA**: **TAA** is better, since MSAA can't improve it at all
- **Specular AA**: **TAA** is better, since MSAA can't improve it at all
- **Performance**: **TAA** is better (but in mobile, MSAA may be better due to hardware support)
- **Sharpness**: **MSAA** is better since it will not blur the result, while **TAA** is **very blurry** by design
- **Ghosting artifact**: **MSAA** is better since it will not produce ghosting artifact, while ghosting artifact may be produced by **TAA** on the motion trail of any fast moving object
- **Shadow AA**: **TAA**'s blur can slightly improve it. MSAA can't improve it at all
- **Project's shader requirement**: **MSAA** doesn't require **depth/motion vector pass**, while **TAA** requires every opaque materials having **depth and motion vector pass**, else a huge amount of **ghosting artifact** will be produced
- **Editor version requirement**: **MSAA** exists in all URP versions, while TAA only exists in **Unity2022** or later

In summary, if **DLSS/FSR3** is not an option for your project, and you are willing to accept **ghosting** and **blurriness** from **TAA**, **TAA** is a more visually stable option than **MSAA** in general.

How to use TAA

To select TAA for a Camera:

1. Select the Camera in the Scene view or Hierarchy window and view it in the Inspector.
2. Navigate to **Rendering** > **Anti-aliasing**, and select **Temporal Anti-aliasing (TAA)**.
3. Navigate to **Output** > **MSAA**, select **Off**

The following features cannot be used with TAA:

- Multisample anti-aliasing (MSAA)
- Camera Stacking
- Dynamic Resolution

For details about TAA, see: Anti-aliasing in the Universal Render Pipeline

TAA vs. MSAA benefits

Especially for rim light, specular and screen space outline, **TAA** can solve almost **any kind of aliasing** problems, while **MSAA** can only solve **Classic Outline's aliasing** problem.(e.g., 2D rim light aliasing is easily visible when using MSAA only if the character is small on screen and having motion)

TAA on mobile?

TAA is capable when rendering in extreme low resolution such as mobile(540p), outline is still visually stable although extremely blurry, while MSAA may start to fail visually when resolution is too low(e.g., lower than 720p).

* In a 1080p mobile screen, when RenderScale is 0.5, the actual rendering is 540p already
* For examples of mobile game using low resolution and temporal AA solutions similar to URP's TAA / FSR3, you can download 学園アイドルマスター, Genshin Impact, Wuthering Waves to a real phone and have a look, running these game on a phone allows you to render in an extremely low rendering resolution, so you can inspect the result of low resolution + temporal AA / FSR3.

TAA issue (ghosting)

Even when all opaque shaders having correct depth and motion vector pass rendered already, TAA ghosting can still exist in some situations, if you use TAA, please keep ghosting artifacts in mind.
The following example will show what TAA ghosting is, and when it will appear.

- when the background is displaying a high frequency detail with dense change of brightness(e.g., a big TV screen in a 3D live concert, bright screen with many black circles mask pattern), and the front character is moving in fast motion = TAA ghosting appears)

*In the above images, the left image shows TAA ghosting(right hand's ghost trail, visible on the back screen), but just by removing the back screen's black circle mask without any other edits, TAA ghosting will become hardly visible.

AA/Quality presets

According to your use case, different setting will be suitable for you, here are some common settings for your reference, you don't need to follow strictly:

Editor recorder (max)

Use this when recording an MV or trailer, especially when using Recorder!

- Game window or Recorder resolution = **2160p(4K UHD)**
- RenderScale = **2**
- Camera's AA = pick the one that looks the best, but likely it is not important
- MSAA = 4x~8x (When **RenderScale = 2**, if MSAA is too slow, use **2x or None**)
- NiloToon's self shadow resolution = 4096 or above
- NiloToon's self shadow soft shadow quality = high

*these settings assume a good GPU with enough VRAM (e.g. better than RTX2060 SUPER / RTX3060)

*If you prefer **TAA** over **MSAA**, override:

- Camera's AA = **TAA (very high)**
- MSAA = **None**

Due to disabling MSAA, VRAM requirement for this method will be lower

*If you prefer **DLAA**, override:

- Camera's AA = **None**
- Attach **DLSS_URP** script on camera, use **Native AA** mode
- RenderScale = **1** (automatically locked by DLSS script, you can't edit it)
- Game window or Recorder resolution = 4320p(8K UHD), not 4K

Note: After the recording, a huge 8K video is produced, using video encoding software(e.g. HandBrake) to down sample the video back to 2160p(4K UHD), this brute force super sampling (offline SSAA) method will merge 4 pixels into 1, and allows you to produce the **best quality 2160p(4K) video**, better than any other listed methods.

PC,Console (v.high)

- Game window resolution = **1440p(QHD)~2160p(4K UHD)**
- RenderScale = **1~2**
- Camera's AA = pick the one that looks the best
- MSAA = 4x (When **RenderScale >= 1.5**, if MSAA is too slow, use **2x or None**)
- NiloToon's self shadow resolution = 4096~8192
- NiloToon's self shadow soft shadow quality = medium / high

*If you prefer **TAA** over **MSAA**, override:

- Camera's AA = **TAA (high)**
- MSAA = **None**

*If you prefer **DLSS/FSR3**, override:

- Camera's AA = **None**
- Attach **DLSS_URP** / **FSR3_URP** script on camera, use **any** mode that fits your GPU budget
- RenderScale = **?** (automatically locked by DLSS/FSR3 script, you can't edit it)

PC,Console (high)

- Game window = **1080p~1440p**
- RenderScale = **1**
- Camera's AA = pick the one that looks the best
- MSAA = 4x
- NiloToon's self shadow resolution = 4096
- NiloToon's self shadow soft shadow quality = medium

*If you prefer **TAA** over **MSAA**, override:

- Camera's AA = **TAA (high)**
- MSAA = **None**

*If you prefer **DLSS/FSR3**, override:

- Camera's AA = **None**
- Attach **DLSS_URP** / **FSR3_URP** script on camera, use **any** mode that fits your GPU budget
- RenderScale = **?** (automatically locked by DLSS/FSR3 script, you can't edit it)

Mobile (v.high)

- Game window = **1080p** (use **Screen.SetResolution**) to ensure UI sharpness
- RenderScale = **0.8~0.9** (Target **864p~972p** actual 3D rendering)
- Camera's AA = None
- MSAA = 2x~4x
- NiloToon's self shadow resolution = 2048~4096, or turn it off
- NiloToon's self shadow soft shadow quality = low, or turn it off

*If you prefer **TAA** over **MSAA**, override:

- Camera's AA = **TAA (Low ~ Medium)**
- MSAA = **None**

*If you prefer **SGSR2**, override:

- Camera's AA = **None**
- Attach **SGSR2** script on camera, use **any** mode that fits your GPU budget

- RenderScale = **?** (automatically locked by **SGSR2** script, you can't edit it)

*__864p~972p__ actual 3D rendering is usually **not practical** on mobile for long play time, unless you have a really optimized rendering, or only target high-end phones

Mobile (high)

- Game window = **1080p** (use Screen.SetResolution) to ensure UI sharpness
- RenderScale = **0.7~0.8** (Target __756p~864p__ actual 3D rendering)
- Camera's AA = None
- MSAA = 2x~4x
- NiloToon's self shadow resolution = 2048, or turn it off
- NiloToon's self shadow soft shadow quality = low, or turn it off

*If you prefer **TAA** over **MSAA**, override:

- Camera's AA = **TAA (Low ~ Medium)**
- MSAA = **None**

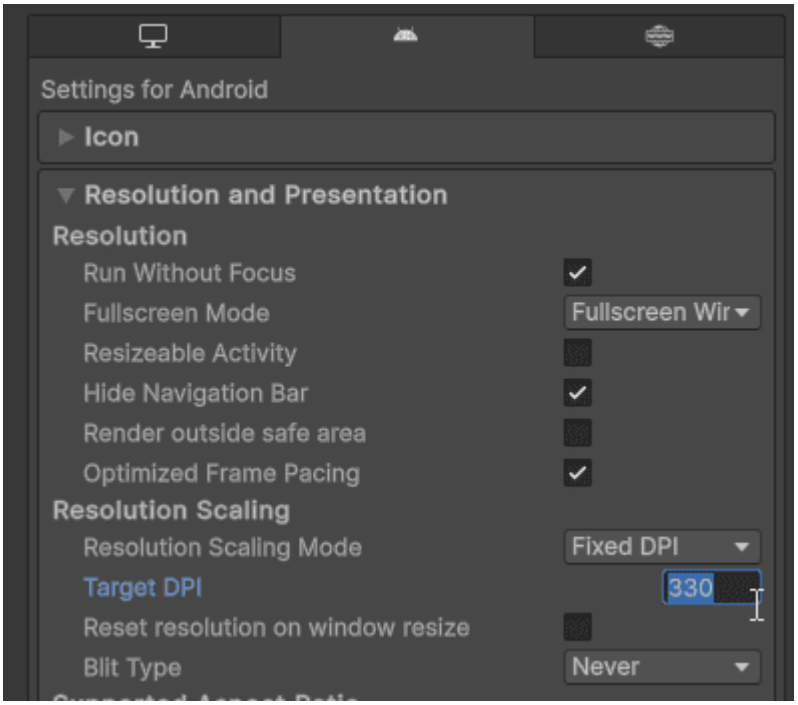
*If you prefer **SGSR2**, override:

- Camera's AA = **None**
- Attach **SGSR2** script on camera, use **any** mode that fits your GPU budget
- RenderScale = **?** (automatically locked by **SGSR2** script, you can't edit it)

*Most mobile games are limited to __720p~864p__ as the maximum actual 3D rendering resolution even if the player choose the highest graphics setting(e.g., GenShin Impact / Honkai StarRail / Wuthering Waves / Infinity Nikki), so we highly recommend you can use __720p~864p__ as a maximum 3D actual rendering resolution for mobile.

For why the suggested the maximum actual 3D render resolution is __720p~864p__, because this resolution range is the most common max resolution for highest setting in **2024Q4**, see: 手机游戏性能终极大横评！掉帧？发热？作弊？猫腻太多！

If you don't require the game window having at least 1080 resolution for UI sharpness, override **Resolution Scaling** to **Fixed DPI** **(target DPI = 330)** with a RenderScale close to 0.9~1 is usually rendering faster and easier to set up.
*Target DPI = **330**, mobile game resolution will become around **743p ~ 849p** in our test devices. You can lower the DPI to **300** for example to further reduce the GPU load but we don't suggest going lower than **300**, it will make your mobile game very blurry.



Mobile (medium)

- Game window = **960p** (use **Screen.SetResolution**)
- RenderScale = **0.68~0.78** (Target **648p~756p** actual 3D rendering)
- Camera's AA = None
- MSAA = 2x~4x
- NiloToon's self shadow resolution = 1024~2048, or turn it off
- NiloToon's self shadow soft shadow quality = low, or turn it off

*If you prefer **TAA** over **MSAA**, override:

- Camera's AA = **TAA (Very Low ~ Low)**
- MSAA = **None**

Mobile (low)

- Game window = **840p** (use **Screen.SetResolution**)
- RenderScale = **0.64~0.77** (Target **540p~648p** actual 3D rendering)
- Camera's AA = None
- MSAA = 2x~4x
- NiloToon's self shadow resolution = off
- NiloToon's self shadow soft shadow quality = off

*If you prefer **TAA** over **MSAA**, override:

- Camera's AA = **TAA (Very Low ~ Low)**
- MSAA = **None**

*If the game doesn't run 30fps in this low preset, you should consider:

- optimize the rendering/material setting
- Use NiloToon(Lite) shader
- give up some very low-end phone's support

Low = turn off AA and postprocess?

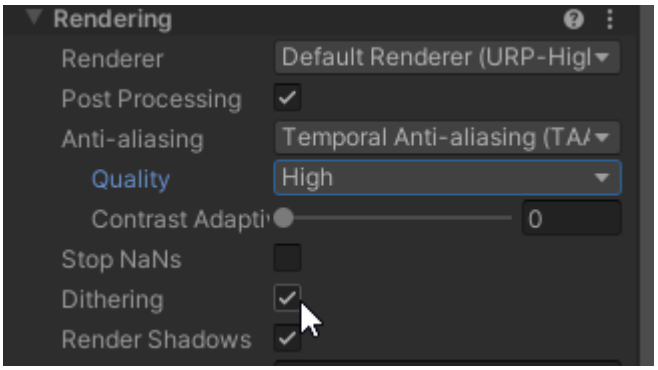
When working on optimizing the game, developer may allow player to control the on/off state of some graphics features in game's graphics setting UI, there are 2 important features that will greatly impact the character rendering if disabled:

- **AA** (e.g., MSAA / TAA / DLSS / FSR3 / XeSS2 / SGSR2)
- **bloom** (e.g., URP's bloom / NiloToon's bloom)

So we highly recommend at least keeping both **AA and bloom** on, even in the lowest graphics quality preset. Keeping AA and bloom on can preserve character's rendering quality (classic outline and overall color).

Color Banding

*If you see **Color Banding** in areas with color gradient (e.g., the color fadeout of volumetric light beam, common in 3DLive), try enable Camera's **Dither**, and not use **FXAA**



DLSS(better AA & fps)

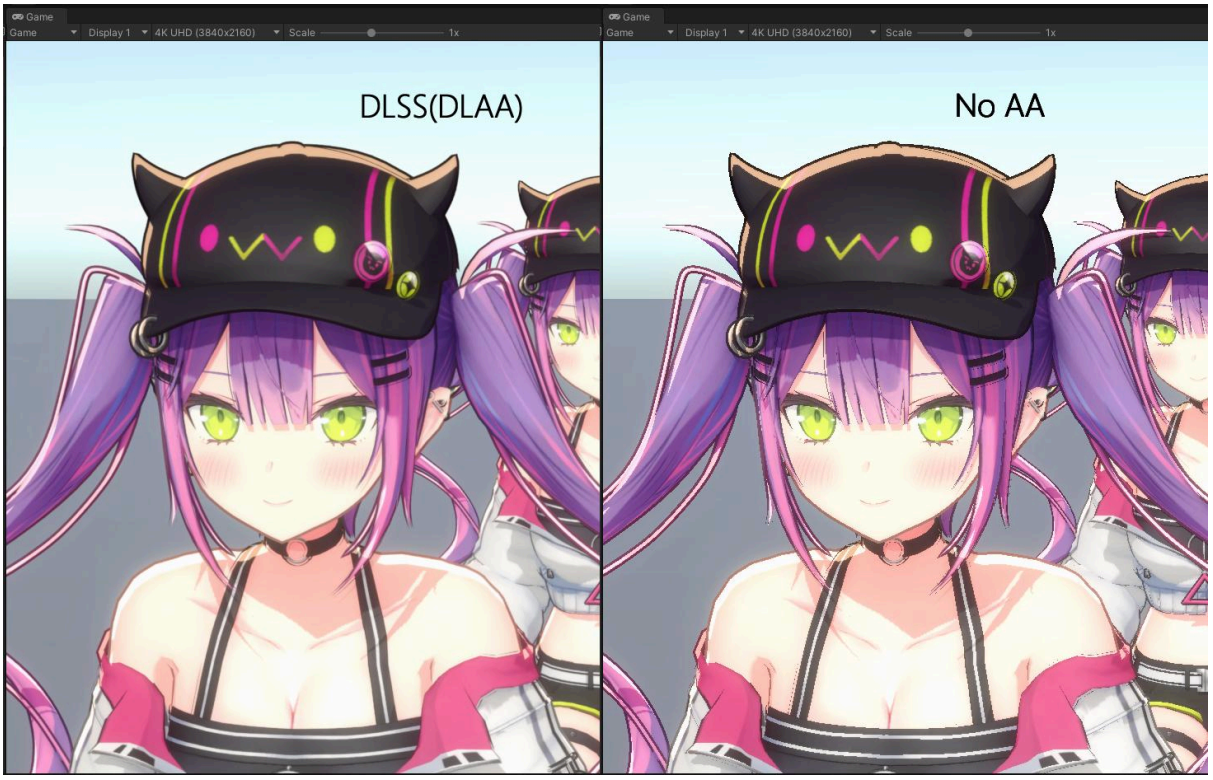
TL;DR - why use DLSS?

DLSS is a much better Anti-Aliasing alternative to URP's **TAA** when **RenderScale <= 1**.

Use **DLSS** If you are:
- **GPU bound** due to high resolution or RenderScale, for example, you find that lowering the **Game window size** or **RenderScale** increases your application's fps
- you want a much better AA to **replace URP TAA** due to **TAA's ghosting and blurriness** when **RenderScale <= 1**

You can purchase DLSS via the AssetStore:
- DLSS (For NVIDIA RTX 20xx GPU or above only)

In the above situations DLSS is very useful when all your shaders(atleast all the opaque character and opaque environment) are rendering depth texture and motion vectors correctly.



(In the above picture, you need to **zoom in / enlarge** the image to see the difference in AA, focus on the outline and rim light to see the difference in AA quality)

DLSS vs TAA

DLAA(DLSS's **NativeAA**) is "DLSS without upscaling", so it works like a better TAA.
DLAA and TAA both have ghosting problems and blurriness, but in our experience when the RenderScale is **1** in 4K:
- DLAA is so much **sharper** than TAA

- DLAA has way **less ghosting** than TAA
- DLAA's pixel jitter artifact is almost invisible, TAA has easily visible jitter pixels
- DLAA can be more stable when handling specular aliasing or high tiling texture effects
- DLAA is slower(GPU) than TAA when using the same RenderScale
- **DLAA is forced to use RenderScale <= 1, while TAA can use RenderScale > 1(max = 2) for better quality**

When to use DLSS

You should only use **DLSS** when targeting a **4K Game Window** or above.
For **1080p/1440p**, you should try RenderScale = 2 with TAA first instead of DLSS, see TL;DR - Best AA?. Switch to DLSS only if your GPU is too slow when using RenderScale = 2 with TAA

When **RenderScale <= 1**, **DLSS** always produce better quality than **TAA**, we use **DLSS's DLAA(NativeAA)** to replace Unity's URP **TAA** for improving character's rim light AA & outline AA, and to improve overall graphics stability since DLSS(Native AA) has way less jittering pixels than Unity's TAA.

Or when GPU bound, we use **DLSS**'s **UltraQuality / Quality** preset to produce a higher fps **4K/8K rendering** in a complex lighting scene with lots of realtime light, shadow and volumetric lighting/fog/particles, these non-NativeAA presets means rendering in a lower internal resolution using **RenderScale < 1** and let DLSS upscale back to full **4K/8K** resolution. When rendering **4K/8K**, the DLSS upscaler can be very important, if you are GPU bound it may give you a big boost to fps with an acceptable rendering artifact.

When to not use DLSS

In **1080p/1440p**, likely your GPU can afford to use RenderScale = 2 with TAA, this setting will produce better & sharper image quality than any DLAA, since RenderScale = 2 with TAA is a brute force Super Sampling AA(SSAA), it is closer to the ground truth quality than DLAA.

If you are working on a video production using Recorder, RenderScale = 2 with TAA is usually the best option, you don't need DLSS in any offline rendering since fps doesn't matter.

Setup and use DLSS

0. Purchase DLSS (For NVIDIA RTX 20xx GPU or above only)
1. Import DLSS (open package manager > My Assets > DLSS)
2. Adding the DLSS renderer feature to your renderer
3. Attach **DLSS_URP** script **only** to your **main camera**, not other cameras. Set Quality = **Native AA** for the best quality
4. **Turn off** that **main camera**'s AA (FXAA/SMAA/TAA)
5. Save scene
6. Install DLSS package using the install button on the DLSS_URP script
7. Close Unity
8. Replace the DLSS dll file to **DLSS4 TransformerMode**

9. Restart Unity
10. Enter playmode

Now DLSS should work in play mode. While in play mode, select the DLSS_URP script, try another **Quality** to confirm DLSS is working.

*All information in this section is also true for FSR3 / XeSS2.

*If you only need to support NVIDIA **RTX 20xx** cards or newer, always **pick DLSS instead of FSR3/XeSS2** since DLSS can produce a much better and stable result

*you must **turn off** all **camera AA**(FXAA/SMAA/TAA) when using DLSS / FSR3

*you may still enable **MSAA** if higher quality classic outline is needed, but DLSS can produce a good outline AA result already even without MSAA, so enabling MSAA may not gain much, it will increase your VRAM usage or may even make the result slightly blurry

DLSS4 TransformerMode

Assuming that you are already using DLSS in your unity project, the **nvngx_dlss.dll** your Unity editor is using may be very old (e.g., **nvngx_dlss.dll 3.8.x or lower**, which means the **old DLSS3 CNN model**, image quality is not the best).

After 2025/01/30, **DLSS4** was officially released via the NVIDIA app, you can improve Unity Editor's DLSS image quality massively by upgrading to **DLSS4**.
DLSS 4 Upscaling is Amazing (4K)

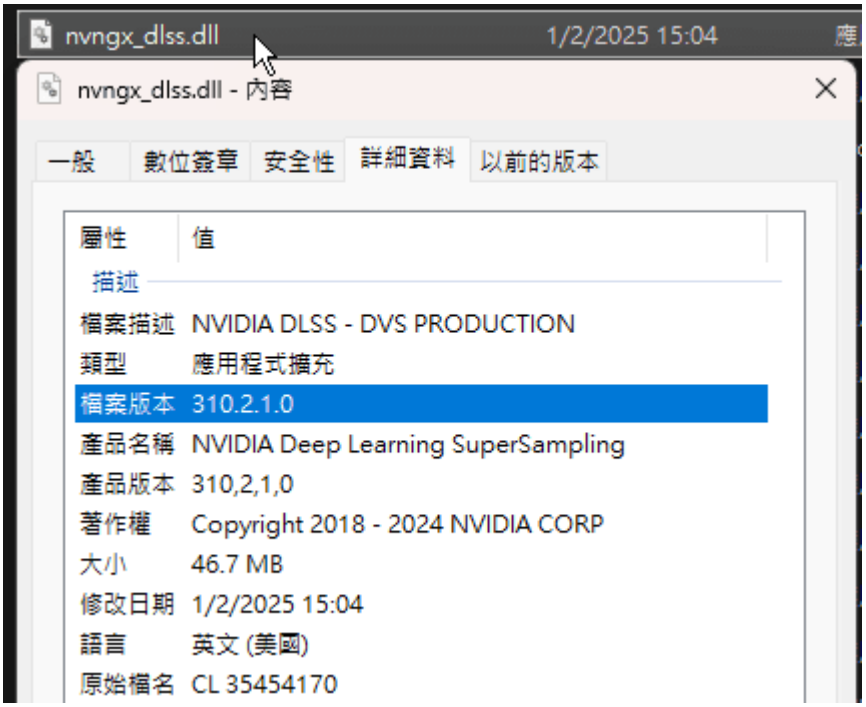
To upgrade, just replace a single file (**nvngx_dlss.dll**) to the latest **nvngx_dlss.dll** (e.g., **310.2.x or high**, which means the **new DLSS4 Transformer model**, image quality is better)

Supported GPU

All NVIDIA RTX cards will benefit from this **DLSS4** update (all RTX 20xx or higher), DLSS4 is not just limited to RTX 50xx cards.

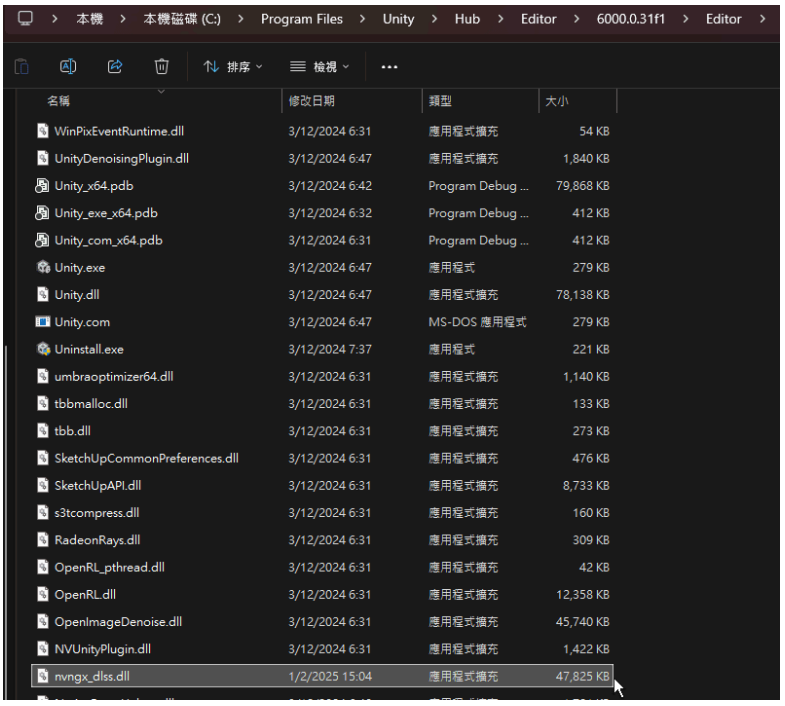
DLSS4 update guide

1.Download the latest official **nvngx_dlss.dll** here (download the version **310.2 or higher**):
NVIDIA DLSS DLL 310.2 Download | TechPowerUp



2.Close Unity, just replace this single file (**nvngx_dlss.dll**) in your Unity install folder, restart Unity, then DLSS4 should work already.

For example, replace this **nvngx_dlss.dll**:
(C:\Program Files\Unity\Hub\Editor\[Your Editor Version Here]\Editor**nvngx_dlss.dll**)



Preset J/K?

The latest DLSS4 **nvngx_dlss.dll** should pick their default Preset (**Preset J/K**) if the preset is not defined by the user. **Preset J/K** means the new **DLSS4 Transformer** model.

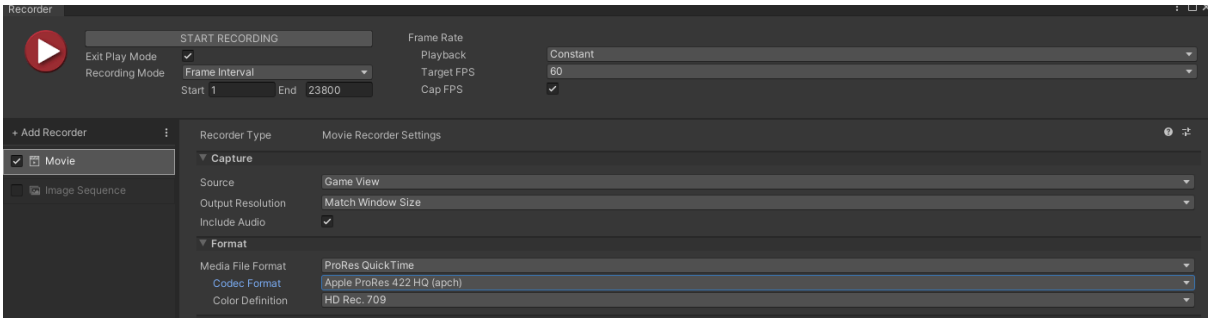
*Please correct us if we are wrong about the default **Preset J/K** behaviour!

Recorder - best settings

TL;DR - general solution

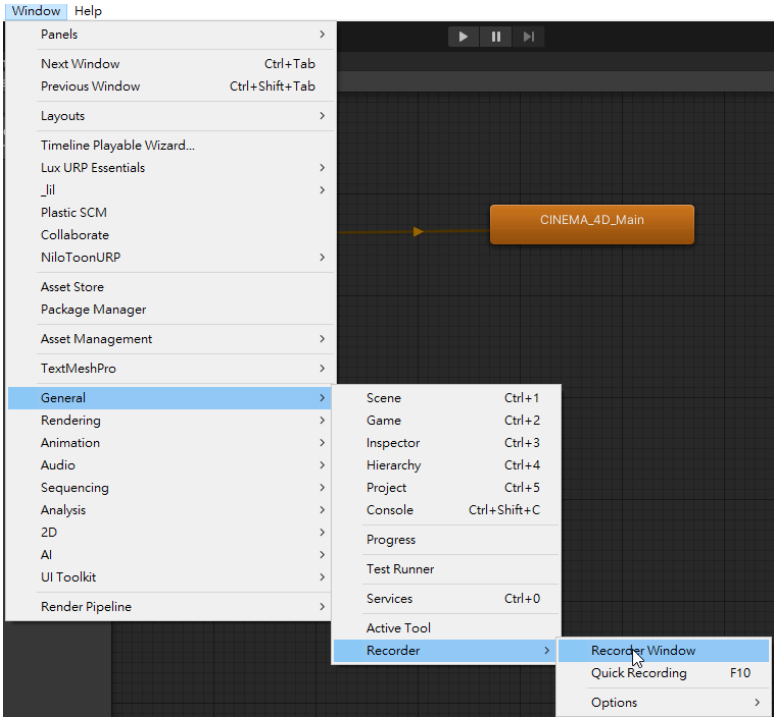
If you don’t want to read the document, just remember:

- use Codec **Apple ProRes 422 HQ (apch)**
- pick the fastest SSD as output(with enough space)



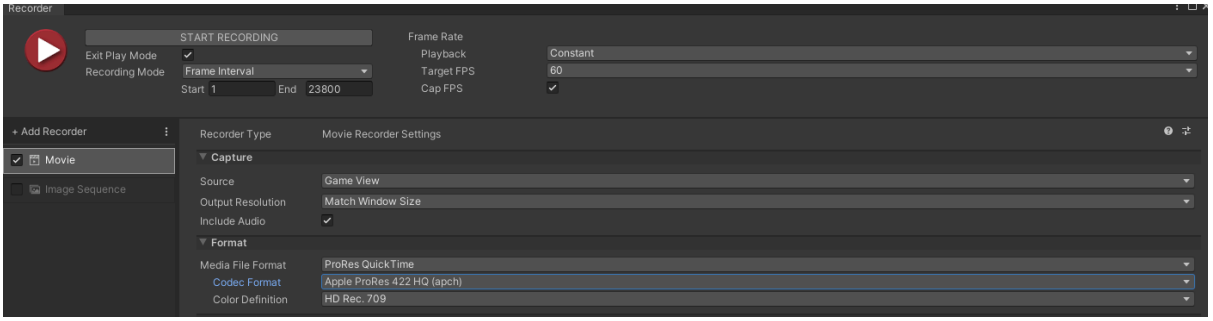
All possible solutions

When you want to record the play mode’s result as a video, you can use Unity’s **Recorder**. Install Unity’s **Recorder** in package manager, and open the **Recorder window**

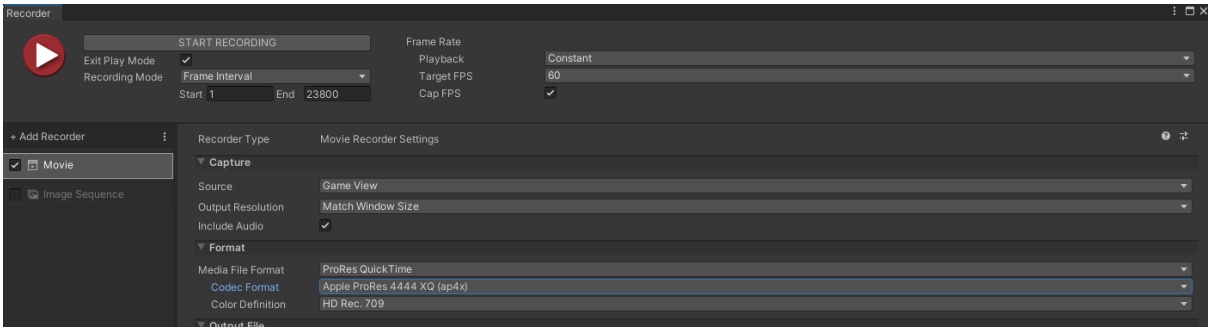


In **Recorder window**, always use **Apple ProRes 422 HQ (apch)** if you **don’t need video’s alpha channel**, it will offer the best balance between render speed, quality and file size (it is visually lossless, very hard to find any difference even when watching the recorded video replay frame by frame comparing to Unity’s game window)

For a 7 minutes 3840x2160 60fps recording, it will generate a ~100GB file



If you want to record the best quality video **with alpha channel**, or you just want the highest quality and don't care about file size, use **Apple ProRes 4444 XQ (ap4x)**.
For a 7 minutes 3840x2160 60fps recording, it will generate a ~200GB file



Although the **Codec format** has **Apple** in the name, it is very common to use it on **Windows** too.

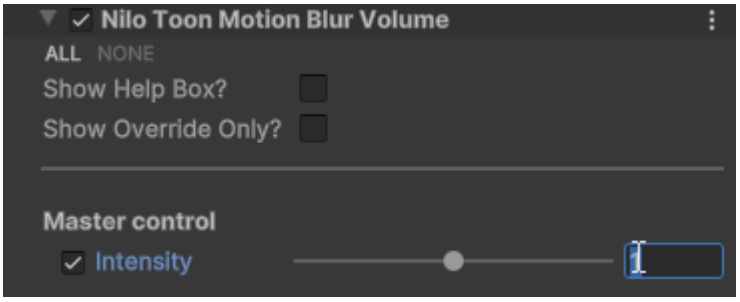
For more information about **ProRes**, see https://support.apple.com/en-us/HT202410
*For a GPU with **8GB or less VRAM** (e.g. RTX 2060 Super = 8GB), **4K** + **RenderScale=2** + **4xMSAA** will slow down the unity recording a lot due to VRAM limit, if that happens you can try to **disable MSAA and use TAA** if the recording speed is really too slow.

Before starting the ProRes recording, it is highly recommended to pick the **fastest SSD** drive of your PC to save the Output File, it can make the rendering much faster and avoid some video player sync problems, make sure you have **enough space** on that SSD.

NiloToon Motion blur tools

Motion blur is usually needed for low fps(e.g. 24/30/or even 60) to produce a sense of motion smoothness, NiloToon provides two types of tools to generate motion blur for real-time application and offline video production.

Real-time use



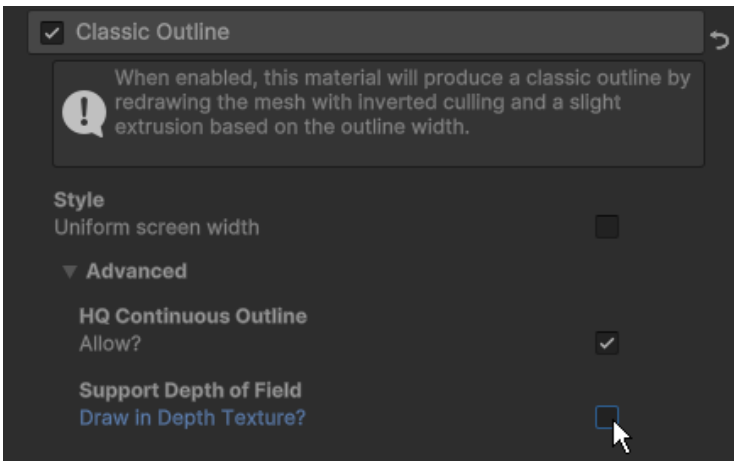
Simply add **NiloToonMotionBlurVolume** in volume profile, override **Intensity = 1**, adjust intensity if needed (common range = 0.5~1). When Intensity is 1, the shutter angle is **180 degrees of 24fps** (shutter speed = 1/48s), so **Intensity = 1** is usually a good default for cinematic motion blur.

Example (Recorder = **30fps**, **NiloToonMotionBlurVolume** = **0.5** intensity):

- Snow halation 이세계아이돌 COVER| 러브 라이브! OST

NiloToonMotionBlurVolume's Note:

- Common **Intensity** range is **0.5~1**
- Blur result is **fps independent**, in any fps, it will generate the same amount of blur on screen per frame
- Only supports **Unity2022.3** or later, works best in **Unity6**
- Using it alone can produce good real-time motion blur, it usually looks good enough in motion if you generated a correct motion vector texture, but if you pause the frame, the blur result is not good when compared to offline render quality
- For **NiloToon_Character** shader to generate the correct motion blur, you need to **disable Classic Outline > Draw in Depth Texture?** in material, because motion blur requires the non-outline depth in depth texture in order to generate the correct motion vector texture



- You can ignore Window > NiloToonURP > MotionBlurVideoBaker, it is not for real-time use

Offline use

TL;DR:
If you just want a summary of how we make 4K MV with offline baked motion blur, here are our standard steps:
1.Volume: NiloToonMotionBlurVolume>intensity = **0.5**
2.ProjectSettings>VFX: Fixed TimeStep = **1/960**
3.Recorder: **960fps**, **4K**, **ProRes422LT** (not ProRes422 nor ProRes422HQ)
4.Ensure **100GB per minute** free **SSD** disk space(e.g., 100x5=500GB for a 5 min recording)
5.Start Recording using Recorder (hide scene window to speedup)
6.Recording finished, a very big mov file is produced
7.NiloToonMotionBlurVideoBaker: use that mov as input video, bake to **60/30fps**, motion blur amount = **default(1)**
8.(optional) post-production: e.g., color correction in Affect Effect / DaVinci Resolve
9.(optional) Handbrake: 4k -> **8K** (preset = **ProductionStandard**). 8K just for trigger Youtube's AV1 encode
10.(optional) Upload Youtube

If you just need the settings reference, skip to 1.Settings

Goal:
Produce a **short video** with the ground truth, perfect offline render quality **cinematic motion blur** and anti-aliasing, where the motion blur quality is similar to offline rendering tools like the **Unreal Engine**'s **Movie Render Queue** (**Temporal sub-sample accumulation motion blur**):

- Enhanced Motion Blur with Movie Render Queue | Tips & Tricks | Unreal Engine
- Movie Render Queue Enhancements in 4.26 | Inside Unreal

^Image above = result of **NiloToonMotionBlurVolume + MotionBlurBaker**

How to:
NiloToon provides:

- **MotionBlurVideoBaker** in Window > NiloToonURP > MotionBlurVideoBaker
- **NiloToonMotionBlurVolume** in Volume

Using **both together** will let you produce offline rendering quality motion blur same as the image above.

Disadvantage:

- Although the result of Window > NiloToonURP > MotionBlurVideoBaker is perfect since it is the ground truth method to produce motion blur, it takes **a lot of time to render**! Using a high-end PC(2024), it still takes a few **hours** to produce a **5 minutes** 1440p MV and ~ **500GB temp ProRes 422 LT video data** generated by the **Recorder** on your SSD!
- It adds motion blur to the video, which may make your viewer **difficult** to take a **sharp screen shot / photo**, since anything moving will have motion blur now

Typical use case:
Usually used for producing a **<= 4k** video that is **<= 10 minutes**, for example:

- MV producion
- Concert/3D Live dancing video production (per song)
- Youtube dancing shorts production
- Trailer video production
- Any short videos with lots of character / camera / particle movement

Examples:

- (60fps result) NiloToon Concert demo
- (60fps result) 닐로툰 콘서트 데모 4K 60fps (nilotoon concert demo 01 4K 60fps)
- (24fps result) ROSÉ & Bruno Mars - APT. || Cover by Bukuki & Mare Flos

Follow the steps below to produce a similar or better motion blur same as the above videos and pictures.

1.Settings

Below are the recommended settings to produce the best motion blur, these settings will be used in 2.Unity Recorder steps.

For **1440p**:

- NiloToonMotionBlurVolume > intensity = **0.5** (for 960fps)
- Recorder = **960** fps (**ProRes 422 HQ**)

For **2160p(4K), max 3840x2160**:

- NiloToonMotionBlurVolume > intensity = **0.5** (for 960fps)
- Recorder = **960** fps (**ProRes 422 LT**)

For **above 4K**:

- NiloToonMotionBlurVolume > intensity = **0.5** (for 960fps)
- Recorder = **960** fps (**ProRes 422 Proxy**)

For **above 4K (if the above failed, fallback to this)**:

- NiloToonMotionBlurVolume > intensity = **1** (for 480fps)
- Recorder = **480** fps (**ProRes 422 Proxy**)

2.Unity Recorder steps

Let's use a **2160p(4K)** video production as an example:

1. Enable **NiloToonMotionBlurVolume** in volume, set **Intensity to 0.5**
2. If you use VFX graph, set Project Settings > VFX > Fixed Time Step **= 1/960**
3. Open Window > General > Recorder > Recorder Window
4. Record a **960** fps video using Recorder (use **ProRes 422 LT**)(not **HQ**), wait for a long time due to high fps recording

*For step1-4, you should change the settings in color, according the 1.Settings
*You should pick the **fastest SSD** that has **enough space** for the high fps Recorder recording result, having **300GB** is usually enough for:
- **1440p 960fps 5 minutes ProRes422HQ recording** using **Recorder**
- **2160p 960fps 5 minutes ProRes422LT recording** using **Recorder**

3.Nilo Baker steps

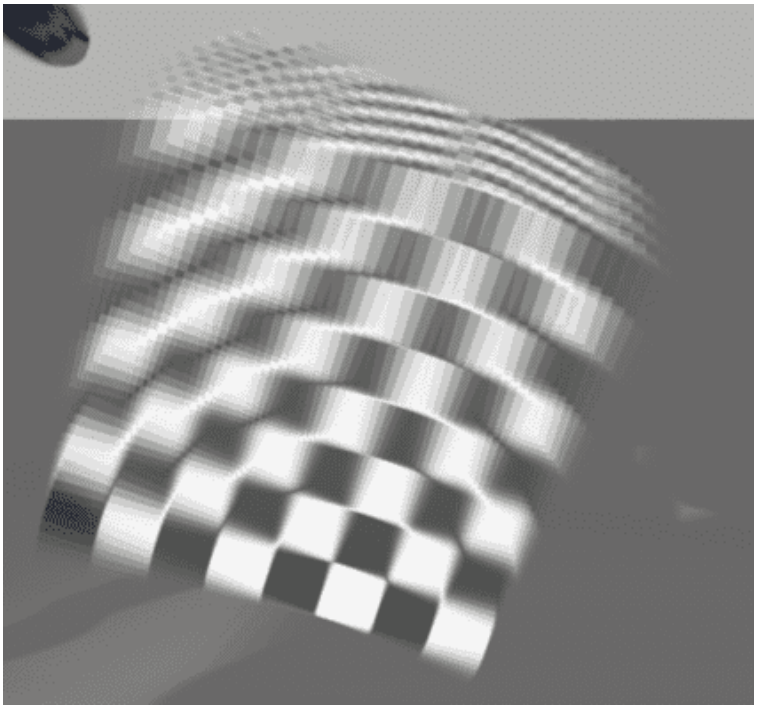
5. Open Window > NiloToonURP > MotionBlurVideoBaker
6. Locate your own **ffmpeg.exe** (download the latest from ffmpeg.org if you don't have it)
7. Import the result big ProRes **.mov** video from 2.Recorder steps as input video
8. Pick a final output fps (usually 24/30/60)
9. Adjust the amount of motion blur if needed (in most cases, **1** is the best already)
10. Click '**Bake Now!**', wait for some time (duration depends on your CPU performance)

You should now get a final video with perfect motion blur and anti-aliasing, similar to the above videos and pictures.

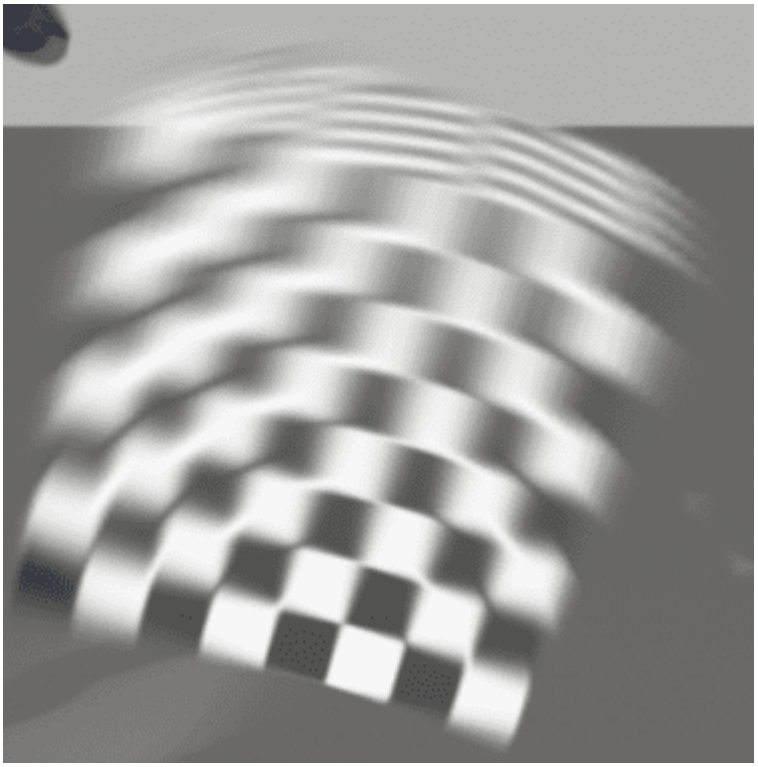
Blur quality example

To produce good motion blur results, **NiloToonMotionBlurVolume** is **needed**, it is important to use it to blur the gaps between subframes.

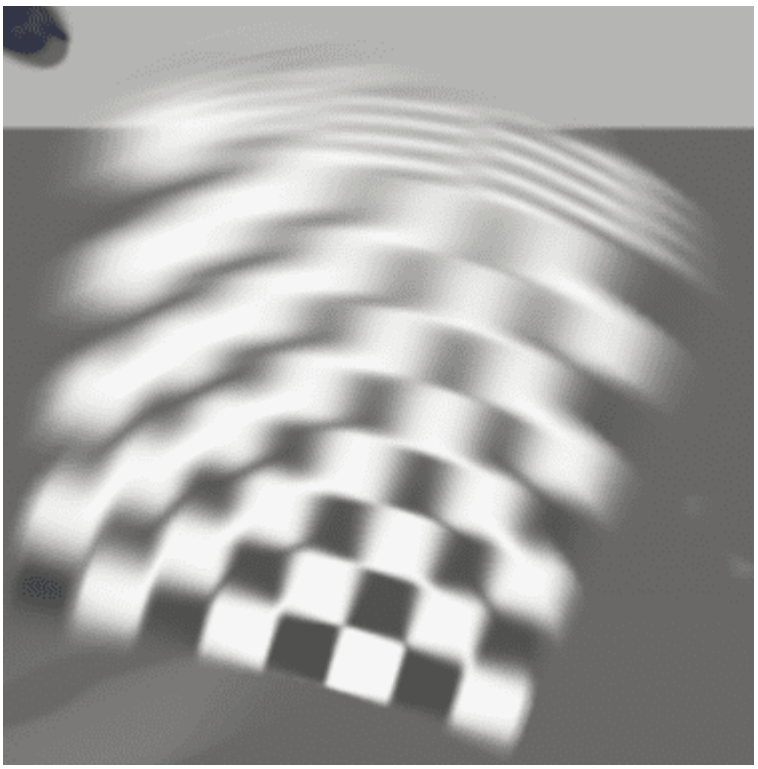
Using Window > NiloToonURP > MotionBlurVideoBaker alone without **NiloToonMotionBlurVolume** will produce subframes gaps, which doesn't look good, see the example images below:



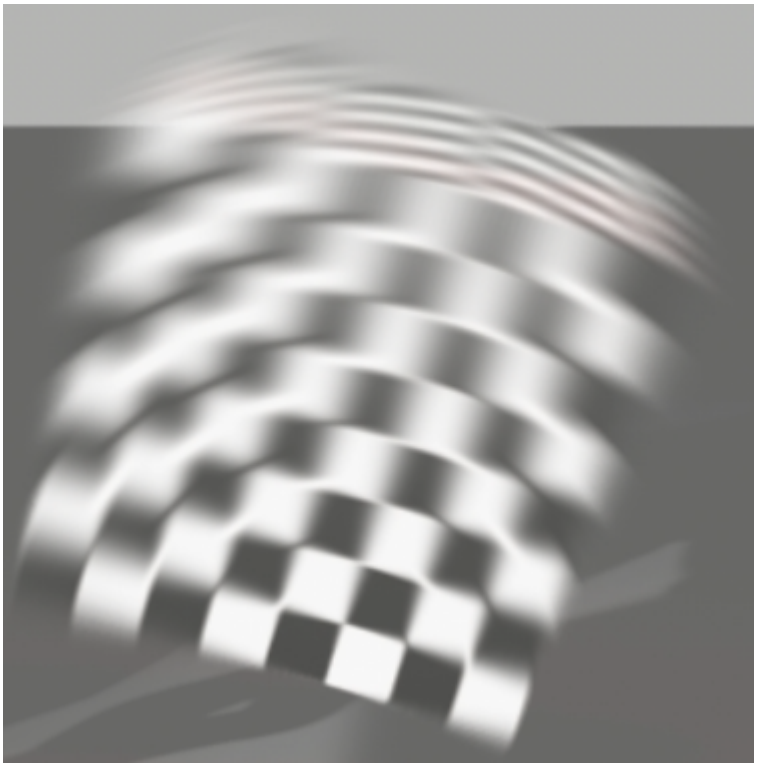
^(Case1) Above Image = Record **480 fps without NiloToonMotionBlurVolume**, then use **MotionBlurVideoBaker(Motion Blur = 1)**. Because **NiloToonMotionBlurVolume** is not used, the subframe gap is easily visible even if we don't zoom in the video for a close up check, it looks very bad, don't do this!



^(Case2) Above Image = Record **480 fps with NiloToonMotionBlurVolume(intensity = 0.5)**, then use **MotionBlurVideoBaker(Motion Blur = 1)**. The subframe gap is reduced due to enabling **NiloToonMotionBlurVolume**, but it is still not acceptable if you aim for high quality results on every static frame.



^(Case 3) Above Image = Record **480 fps with NiloToonMotionBlurVolume(intensity = 1)**, then use **MotionBlurVideoBaker(Motion Blur = 1),** Gap is greatly reduced due to increasing **NiloToonMotionBlurVolume** intensity, this is starting to be an acceptable quality result. If you want the best result, see (Case 4) below



^(Case 4) Above Image = Record **960 fps with NiloToonMotionBlurVolume(intensity = 0.5)**, then use **MotionBlurVideoBaker(Motion Blur = 1),** Gap is almost not visible due to a high amount of temporal samples from 960fps recording, this setting has the best quality result. **This is the setting that we always use for any production MV, so we recommend**

you use this when you need the best quality results.

Compress to smaller video

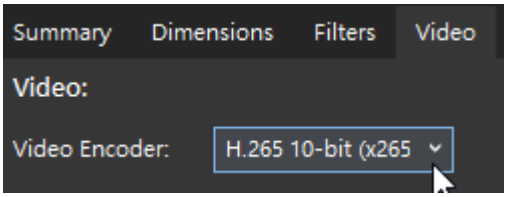
If a **ProRes mov** file is **too large** for upload, share, storage or even video editing, for example a single 4K60fps 7mins **ProRes 422 HQ** video is **~100GB** in size, you can download **HandBrake**, an easy to use encoding software that we always use to reduce video file size without producing visible visual quality loss.

TL;DR - general solution

If you don't want to read the document, just do this:

1. Open **HandBrake**
2. **Import** the big video
3. Pick **Production Standard** preset

4. Change Video Encoder from **H.264 (x264)** to **H.265 10-bit (x265)**, if H.265 is acceptable for your usage



5. Start **Encode**, the result video should be so much smaller, and you can't see any difference in quality unless you stop the video and compare frame by frame

Prerequisite

The document below will explain in depth about all the encoder settings and quality relationship, after reading you should be able to understand how to set up the best settings in HandBrake.

Before continuing, you should understand these concepts first:

- **Encoder difference**: **H.264** vs **H.265** vs **AV1** vs **ProRes**
  1. **H.264** is **universally supported**. **Quality** is **average** and **file size** is **large**
  2. **H.265** has **good compatibility**. **Quality** is **good** and **file size** is **Medium**
  3. **AV1** has **limited compatibility** in 2024. **Quality** is **very good** and **file size** is the **smallest**, but it is not a good format for use in video editing software. Due to **AV1**'s open source and royalty free nature, and it is backed and used by major tech companies, it is expected that **AV1** will slowly replace H.264/H.265/H.266. If you have a high-end computer for using **AV1** and want your compressed videos to be future proof in quality, you can try **AV1**
  4. **ProRes** is the result of your Unity Recorder ProRes recording, it is great for video editing due to its **visually lossless quality** and the ability to **fast decode**(it is an intra coding format), but the **file size is extremely large**, sometimes it is so large that you have no choice but are forced to compress it to a smaller file using other encoders

	Quality	Size	Compatibility
H.264	Average	Large	Universally supported
H.265	Good	Medium	Good
AV1	Very Good	Smallest	Limited
ProRes	Visually Lossless	Extremely Large	Good

- VMAF, a **0~100 score** developed by Netflix, where a score of **100** means **visually lossless** when comparing the compression result to the uncompressed source video. If you want the compressed video to be close to visually lossless, you should aim to reach VMAF = **99.9~100** if possible. To check your compressed video's VMAF score is high enough or not, we use this VMAF GUI tool

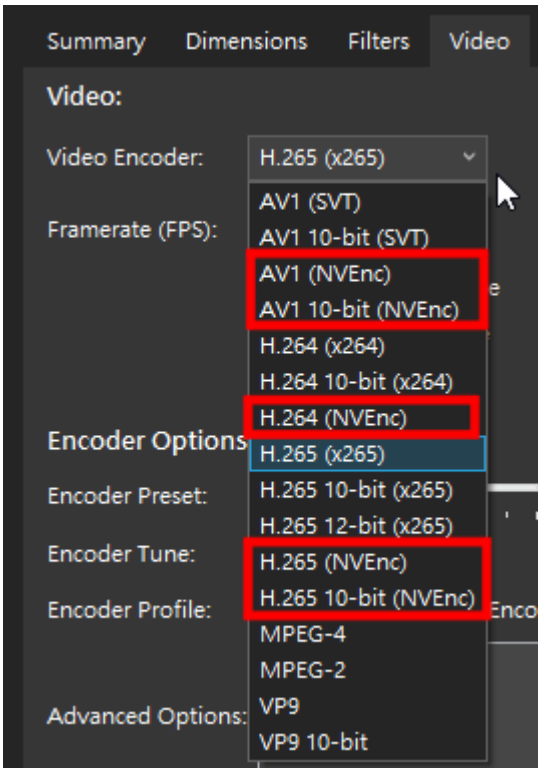
- RF/CRF (Rate Factor). It is the quality level in reverse, the lower the **RF** the better the quality, but bigger in file size. If you want to get a **VMAF** score of **99.9~100** (visually lossless), we recommend you using these **RF** as a start in HandBrake:
  - Use **RF = 10** for **H.264 (x264)**
  - Use **RF = 8** for **H.265 (x265)**
  - Use **RF = 10** for **H.265 10-bit (x265)**
  - Use **RF = 14** for **AV1 10-bit (SVT)**

HandBrake Guide

Assumption

Before we start, there are a few assumptions:

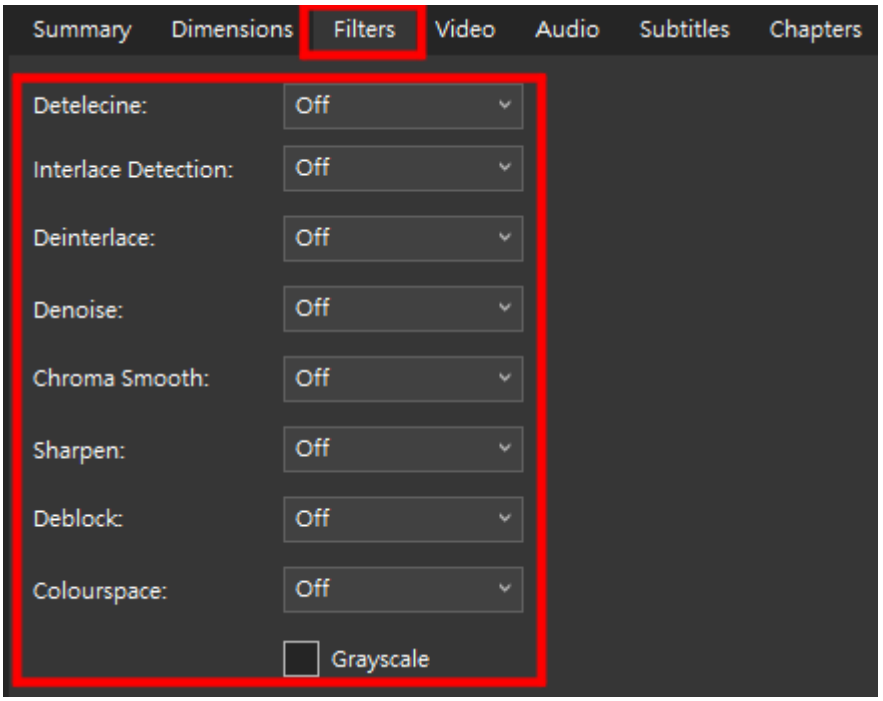
1. We will not use any **GPU encoders (e.g., NVEnc)**, since they are designed to sacrifice quality for encoding speed, which is usually for real-time streaming use. In this document we focus only on **CPU encoding** where encoding speed is not a concern, we care only about the final **quality** and **file size**.



*In the above image, we will **not** use any **(NVEnc)** GPU encoder (highlighted in red)

2. In HandBrake's **Filter** tab, you should always **turn off all filters** since we assume the input source is the **ProRes** recording from **Unity**, that means we don't need any filters to fix anything. In fact, enabling filters like **Interlace Detection** or **Deinterlace** will produce **weird high contrast dots/aliasing/pixel displacement** around eye, mouth and outline which looks wrong on a toon/anime style 3D character model.

So make sure you always set all filters = Off in the Filters tab!



Pick the best encoder

To use HandBrake to compress a video to a smaller size, the first step is to decide which Encoder (**H.264** vs **H.265** vs **AV1**) to use:
- If you aim for **maximum quality** = **AV1** > **H.265** > **H.264**
- If you aim for **smallest file size** = **AV1** > **H.265** > **H.264**
- if you aim for **maximum compatibility** = **H.264** > **H.265** > **AV1**
- if you aim for decode performance of **video editing software use** = **H.264** > **H.265** > **AV1**

AV1

Introduction

AV1 (AV1 10-bit (SVT)) has **limited compatibility** in 2024. **Quality** is **very good** and **file size** is the **smallest**, but it is not a good format for use in video editing software.

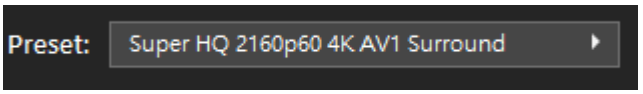
We recommend you use **AV1** only if you can **decode** **AV1** nicely in your video player(e.g., **PotPlayer**) and your video editing software(e.g., **Davinci Resolve**).

If you have difficulty playing(decode) **AV1** video in your video player or video editing software, then **AV1** is not a good option for your use, **skip** **AV1** and go read H.265 instead.

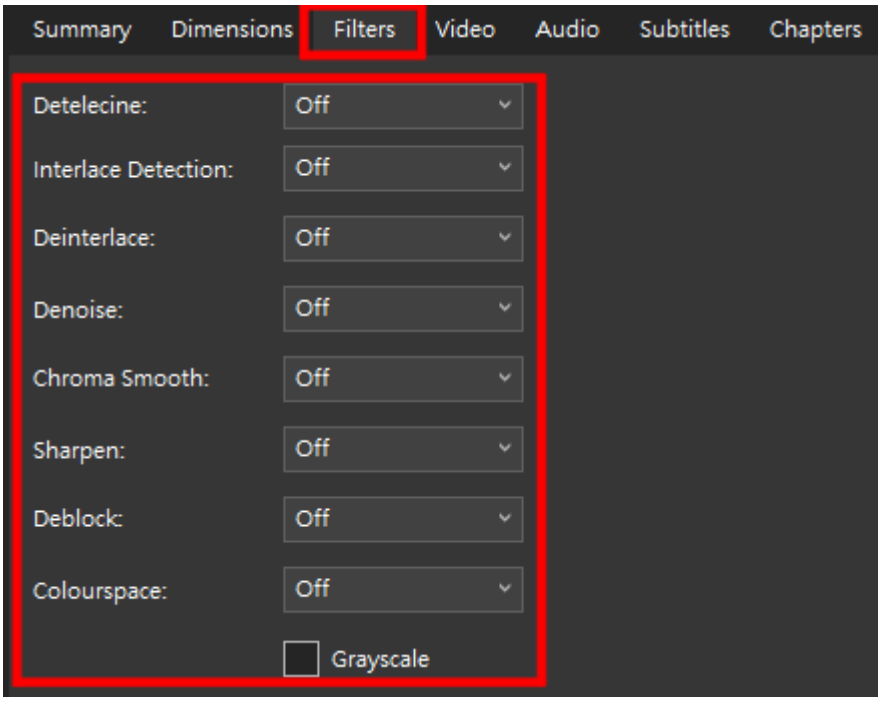
Encode in AV1

If you want to produce the **best quality(VMAF ~= 100)** **AV1** video, follow these steps:

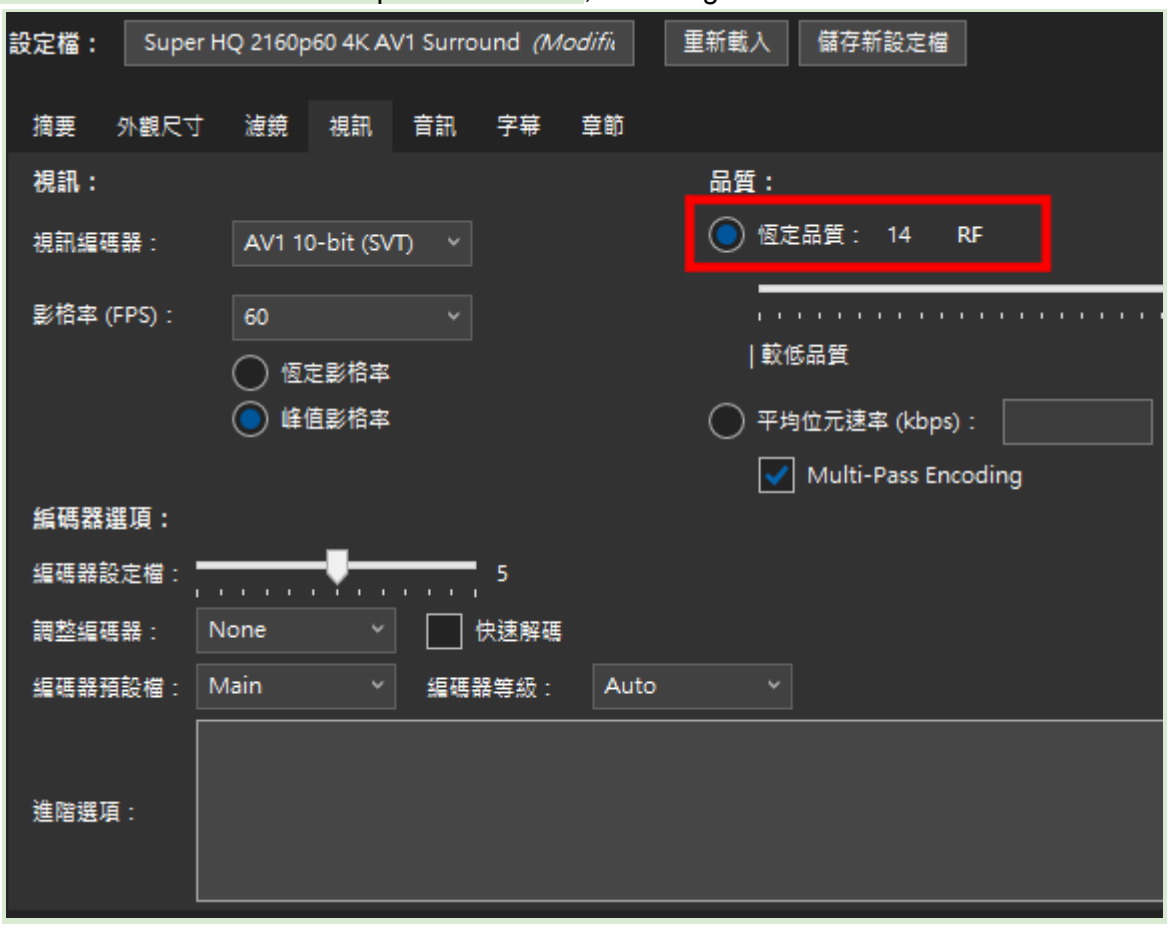
1. pick **Super HQ 2160p60 4K AV1** preset, it will reset all params to this preset



2. **disable all filters in the filter tab**.



3. Set **RF** to **14 or lower** to keep **VMAF ~= 100**, see image below

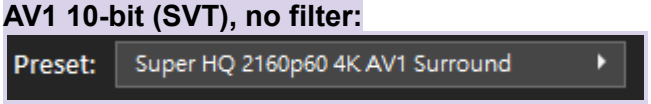


4. (optional) If you don't need the result to be visually lossless and prefer a smaller file(**VMAF = 99.9~100**), you can increase **RF** in the above image to **15~22**, doing this will generate a smaller file with minor visible compression artifact/noise
5. (optional) If you prefer a much smaller file (**VMAF = <99.9**), you can increase **RF** in the above image to **23~30**, doing this will generate a much smaller file but with clearly visible compression artifact/noise

AV1 RF sheet

Below is the compression result chart using an input video of 3384x1440, H.264, SDR, ~900Mbps bitrate, CRF0, 60fps, 04:09 length, NiloToon concert MV with fast character and camera motion, lots of ribbon VFX graph particles. Compression results are sorted by VMAF score, divided into VMAF color groups to help you decide the right **AV1 RF** for your HandBrake video compression.

AV1 10-bit (SVT), no filter:



Compression setting	VMAF (closer to 100 is better)	FileSize (smaller is better)
Uncompressed input	N/A	25.42GB

RF=**10**	100	2.46GB
RF=**12**	100	1.96GB
RF=**14**	100	1.65GB
RF=**15**	99.99	1.54GB
RF=**16**	99.99	1.43GB
RF=**18**	99.97	1.25GB
RF=**20**	99.95	1.12GB
RF=**22**	99.91	1.01GB
RF=**25**	99.82	0.86GB (860MB)
RF=**30**	99.39	0.63GB (630MB)
RF=**63**	71.07	0.07GB (70MB)

Usually you can start with **RF=14** for **AV1**, and keep increasing **RF** until you can see the compression artifacts in motion, or until you are satisfied with the file size(small enough).

AV1 10-bit (NVEnc)

It is the GPU **AV1** encoder for **NVIDIA RTX 40xx cards**, although the compression speed is very fast due to the encoding runs on the GPU, the result is not reliable, it may produce wrong results for some random frames. We don't use it and we don't recommend you using it in 2024, it still needs more time in development.

Summary

If you don't care about compatibility and video editing software use, and just want the **smallest file** with **very good quality** for playback/archive, **AV1** with **no filters** and **RF=14** is a very good start.

H.265

If **AV1** is not an option for you due to any compatibility problems, your next best option is **H.265 10-bit**.

Introduction

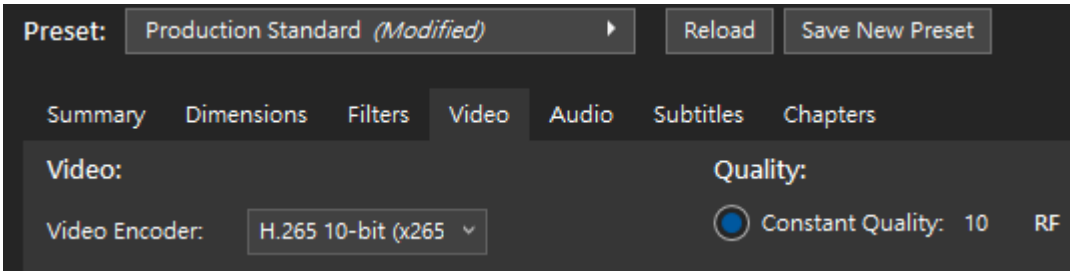
H.265 has **good compatibility**. **Quality** is **good** and **file size** is **Medium.**

Encode in H.265

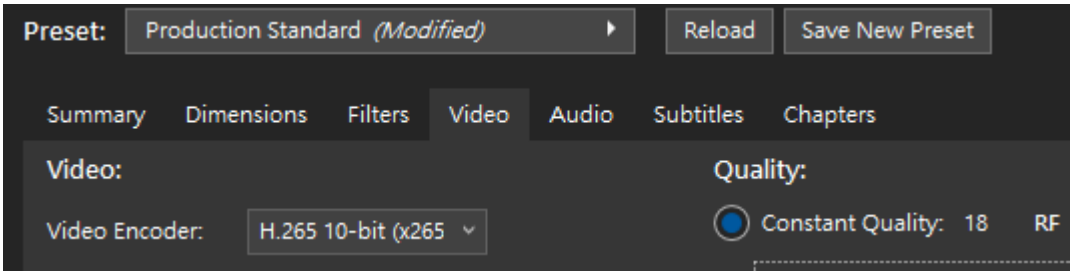
If you want to produce the **best quality(VMAF ~= 100) H.265** video, follow these steps:

A) If you need a **VMAF ~= 100 H.265** video with a **reasonable file size**:
   1. pick **Production Standard** preset, it will reset all params

2. click the **video** tab
3: change encoder from **H.264 (x264) -> H.265 10-bit (x265)** in the **video** tab
4a. keep **RF = 10**



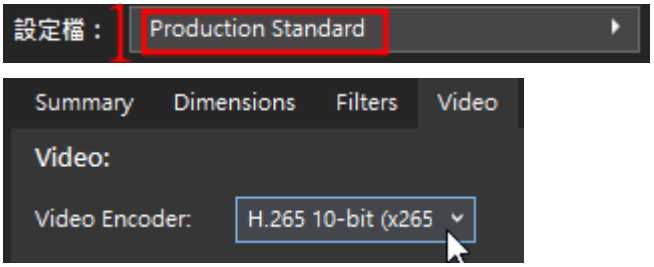
4b. (optional) change **RF** from **10** -> **11~18** according to how much quality loss you can accept



H.265 RF sheet

The following charts are the result of encoding with different encoder and RF settings; it can be used as an estimate to help you find the best **Encoder+RF** settings.
If you are not sure, you can simply start from this preset:
Production Standard, H.265 10-bit (x265), No Filters

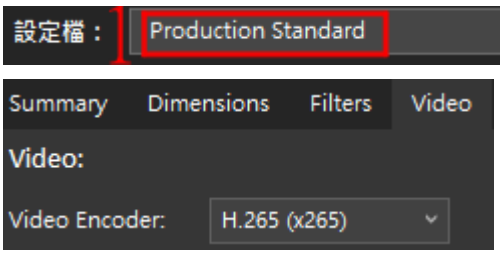
Production Standard, H.265 10-bit (x265), No Filters:



- recommend RF = **10~14** for quality

Compression setting difference	VMAF (closer to 100 is better)	FileSize (smaller is better)
Uncompressed input	N/A	25.42GB
RF=**10**	100	2.64GB
RF=**11**	99.99	2.34GB
RF=**12**	99.98	2.47GB (not typo)
RF=**14**	99.92	1.92GB
RF=**15**	99.77	1.41GB
RF=**16**	99.74	1.48GB (not typo)
RF=**18**	99.33	1.15GB
RF=**20**	98.56	0.89GB (891MB)
RF=**22**	97.33	0.69GB (694MB)

Production Standard, H.265 (x265), No Filters:

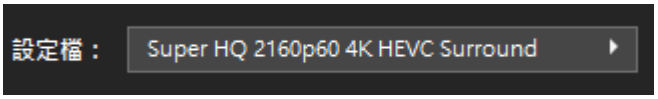


- recommend RF = **8~12** for quality

Compression setting difference	VMAF (closer to 100 is better)	FileSize (smaller is better)

Uncompressed input	N/A	25.42GB
RF=**8**	100	3.46GB
RF=**9**	99.99	3.05GB
RF=**10**	99.98	2.69GB
RF=**12**	99.92	2.09GB
RF=**13**	99.87	1.84GB
RF=**14**	99.79	1.62GB
RF=**15**	99.67	1.43GB
RF=**18**	98.88	0.96GB (985MB)
RF=**20**	97.92	0.77GB (772MB)
RF=**22**	96.54	0.61GB (611MB)

Super HQ 2160p60 4K HEVC Surround, No Filters:



- recommend RF = **10~14** for quality

Compression setting difference	VMAF (closer to 100 is better)	FileSize (smaller is better)
Uncompressed input	N/A	25.42GB
RF=**10**	100	3.38GB
RF=**11**	99.9	2.95GB
RF=**12**	99.99	2.58GB
RF=**14**	99.95	2.00GB
RF=**15**	99.9	1.76GB
RF=**16**	99.82	1.55GB
RF=**18**	99.5	1.20GB
RF=**20**	98.89	0.93GB (934MB)

RF=**22**	97.84	0.73GB (727MB)

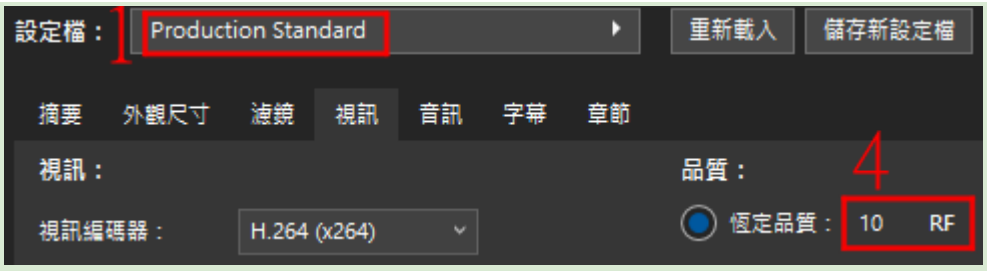
H.264

Introduction

We usually don't encode in **H.264**, unless we need to improve video editing software decode speed

Encode in H.264

If you need the **VMAF ~= 100** **H.264** video with a **large file size**: simply
1. pick **Production Standard** preset and **don't edit any settings**
2. keep **RF = 10**



It will give you a smaller video than the original ProRes, but still be able to use it in video editing software with fast decode time.

Production Standard setup

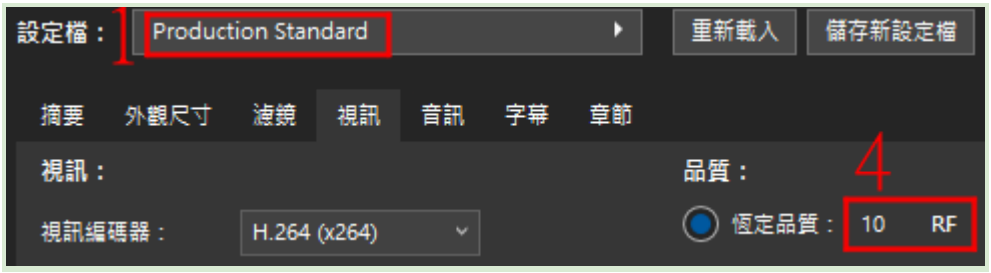
Production Standard preset is not 100% truly lossless since **H.264**/**H.265** are lossy compression methods, but it is visually lossless. In practice, the result is always good enough for any **video editing** (This preset uses **CRF=10** by default, which means extremely high quality and large file size)

For a **7 minutes 3840x2160 60fps ProRes 422 HQ Recorder recording (~100GB)**, **Production Standard** preset will convert it from **~100GB** to:

- **~15GB (H.264)**, **VMAF ~= 100**
- **~7GB(H.265)**, **VMAF ~= 99.98**

Best quality+support

If you want the **best quality** and **be able to play the video in any hardware & software**, simply use the **Production Standard** preset **without any change**, it will produce a **large H.264** video with the best quality. You can copy the setting in the image below:



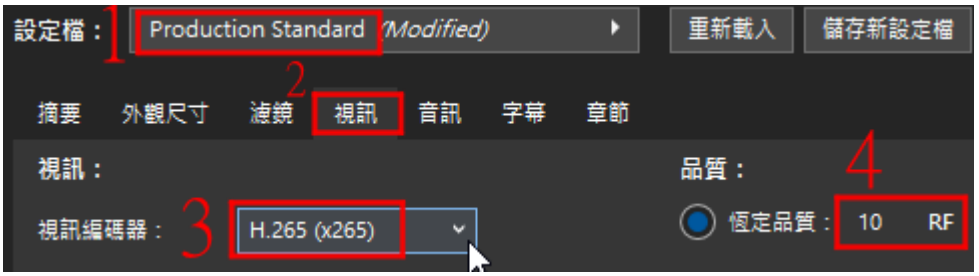
Pros: Visual Quality is **the best** for video editing software use, you can't see any compression artifacts (**VMAF ~= 100**). Also since it is **H.264**, it works in any video editing software, OS and hardware

Cons: The only disadvantage is a much **larger file size, usually ~2x in size when compared to H.265**

Reasonable quality

If you want to save disk space, and **H.265** + **a little bit of compression artifact** are acceptable**:**

1. pick **Production Standard** preset first (Click **1**)
2. only edit **H.264 -> H.265** in the **video encoder** section (Click **2**&**3**). Doing this will produce a file with **~50% smaller** size without noticeable quality loss
3. ensure **RF** is **10** or lower, by default it is **10** (Check **4**)



Pros: file size is **~50% smaller** due to H.265, when compared to H.264

Cons: Although this method's **Visual Quality** is still **high enough** for video editing software use (**VMAF ~= 99.98**), you can start to see compression artifacts if you check frame by

frame compared to the default H.264 **Production Standard**. Also since it is **H.265**, it is not always supported in all video editing software, OS and hardware

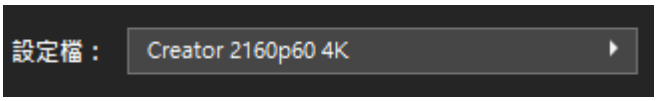
H.265(**1**->**2**->**3**->**4**) in the above image is the setting that we use most of the time. Our workstation PC and its software supports **H.265**, and we expect our clients can play/use **H.265** videos without any problem.

If **H.265** is a problem, then we fallback to the bigger file size **H.264** method

Other presets?

Creator 2160p60 4K?

It is actually the same as **Production Standard**, but with a higher default **RF** of **22**(lower quality). Due to the default low quality of this preset (**VMAF ~= 96.57**), and it uses **H.264** by default, it doesn't save that much space but hurts quality too much, so we **don't need to** use **Creator 2160p60 4K**.



Matroska?

For **H.265 MKV 2160p60 4K** or **AV1 2160p60 4K**, they are very similar to **Super HQ 2160p60 4K HEVC** or **Super HQ 2160p60 4K AV1**, but in a .**mkv** container

Hardware (NVENC)?

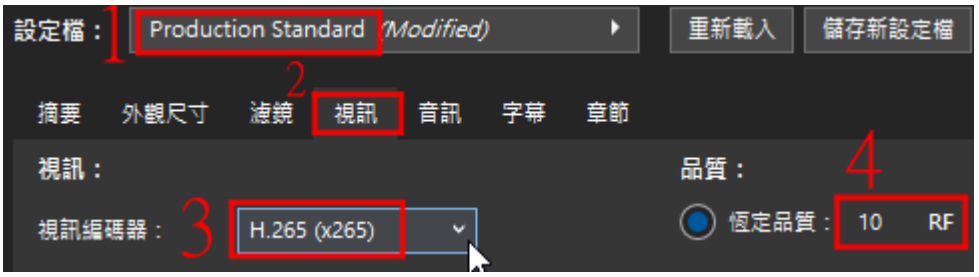
If you are using a **NVIDIA GPU**, and you don't want to spend time waiting for the CPU to encode, you can try the **H.265 NVENC 2160p 4K** option, it encodes using the GPU!
It will provide a huge encoding speed up (**~ 4x to 8x speed up**), but with a slightly lower VMAF score when compared to the **Production Standard** preset with **(H.265,RF18)**.

If you really need to use **H.265 NVENC 2160p 4K**, you should always use the following settings for the highest quality **H.265 NVENC 2160p 4K** result, the encoding speed is still very fast if you have a good GPU:

- **CQ = 0**
- **encoder preset = Slowest**

*We don't use **H.265 NVENC 2160p 4K** since it is designed to hurt/limit quality for a much faster encoding speed(**VMAF ~= 98.77** using the settings in the above image, where **Production Standard** preset with **(H.265,RF18)** can produce a better **VMAF ~=98.88** with a very similar file size). We never put encoding speed at a higher priority than the result visual quality, so we don't recommend you using this if you have a lot of time to encode

RF/CRF (Rate Factor)



(**RF/CRF** is located at the **position '4'** in the above image)

RF is similar to a **quality slider in reverse**, the lower the **RF**, the better in quality, but also the bigger the result video size, each HandBrake preset will use different default **RF**:

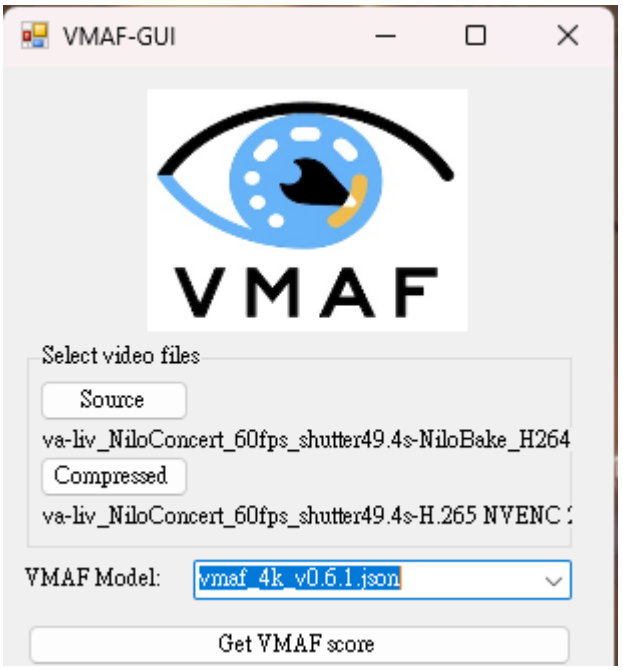
- CRF = **2** (**Production Max**) max quality, almost true lossless, but usually the quality is too high and overkill that makes the video too large without improving quality that much
- CRF = **10** (**Production Standard**) best quality, usually visually lossless. **RF 10** is highly recommended for production, due to its balance of quality and file size
- CRF = **15** (...) high quality, but start to show visible compression artifacts
- CRF = **20** (**Super HQ 2160p60 4K HEVC**) is ok quality, but with visible compression artifacts
- CRF = **22** (**Creator 2160p60 4K**) is low quality, with easily visible compression artifacts
- CRF = **24** (**Fast 2160p60 4K HEVC**) is bad quality, quite aggressive in compression, the video's visual quality will be damaged
- CRF = **26 or higher** is too aggressive in compression, the video's visual quality will be damaged badly and looks very bad, don't use it!

*Usually you don't need to set anything about **RF** manually, HandBrake **preset** will set the **RF** for you. If you want to have more control, the common range of RF is **10~22**

- **Set a lower RF(around 10~15)** if you want to improve quality
- **Set a higher RF(around 16~22)** if you want to make a smaller video

Check VMAF score

If you want to check your VMAF score locally after compressing a video (VMAF score shows how much quality is lost visually), you can use this VMAF-GUI, unzip it in your fastest SSD to improve score calculation speed since it needs to write a large amount of temp data into your disk



- Score of **100** is visually lossless, perfect for video editing software use
- Score of **99.9~100** is good enough for video editing software use
- Score of **99.0~99.9** is good enough as an end result video for playback
- Score of **<99.0** will show visible compression artifacts

Video Compare tool

We use **Video-Compare**, great for checking every possible small difference between 2 videos with the same resolution, especially good for checking compression differences.

GUI version (easier to use, recommended):

- GitHub - TetzkatLipHoka/VideoCompareGUI: GUI for the Video-Compare Tool

Core version (require cmd, but readme is good for control / document):

- GitHub - pixop/video-compare: Split screen video comparison tool using FFmpeg and SDL2
- Example tutorial video

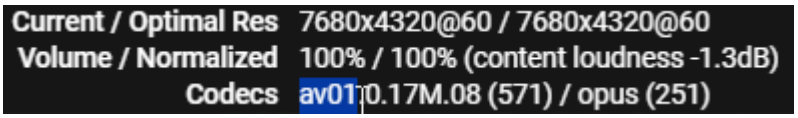
Upscale to 8K for Youtube

TL;DR - general solution

If you want to upload to youtube and produce the highest encode quality Youtube video:

- make sure the uploaded video has both **width >= 7680** and **height >= 4320,** this resolution(8K) force Youtube to use **AV1** encoder (**AV1** is the best encoder in youtube)

When done successfully, after 1-2 days, the Youtube video will be using **AV1**(**av01)** Codec, and **8K** option will be available, which means highest quality



If **8K** upload is not possible, at least upload as **4K (width >= 3840 and height >= 2160)**.

Introduction

When you upload a video to youtube, no matter how good the encode quality of your local video file is, if you upload it as a less than **8K(7680x4320)** video (e.g., 1080p/1440p), the result quality that YouTube displays will be worse than your local video file, especially for 1080p/1440p upload, since Youtube's **VP9 1080p/1440p** video encoding are intended to be low quality in order for YouTube to reduce operation cost and increase processing speed.

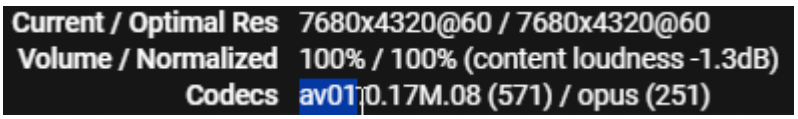
If you need to ensure the quality is as good as possible after uploading to Youtube, try the following "**Upscale to 8K"** solution.

Problem

- I uploaded a 1080p/1440p video to YouTube, the video file itself is very good in encode quality if I play it locally, but after the upload, the YouTube displayed result is very bad, looks like it was heavily re-compressed in a bad quality by YouTube

Solution

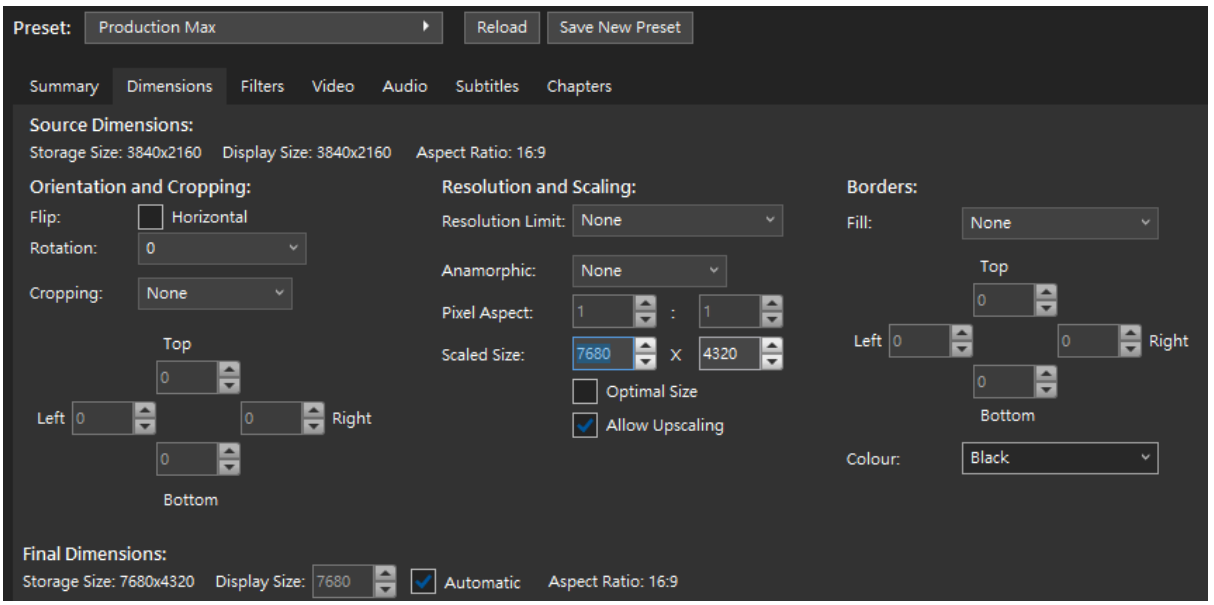
- **Upscale** to at least **8K(7680x4320)**, then upload to Youtube to trick Youtube using the best quality encoder **AV1 (av01)**, instead of **VP9 (vp09)**



Step by Step guide:

1. you first prepare a video file (aspect >= 16:9), intended to upload to youtube
2. open **Handbrake**, and select the video in step(1) as input
3. select **Production Max** preset
4. click **Dimensions** tab
5. disable **Optimal Size**
6. enable **Allow Upscaling**

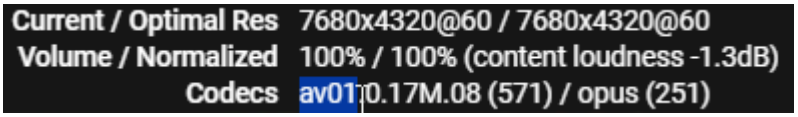
7. set **Scaled Size**'s **height = 4320**, also click on the **width** to let Handbrake auto calculate the width for you depending on your input video aspect
8. if all steps are correct, you will see the final dimensions = **width x 4320**, where width **>= 7680**

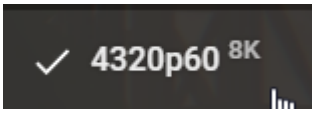


9. click **Start Encode**, the result video will be very large (~150GB for 4 minutes 60fps)
10. Upload the result of step(9) directly to youtube, we assume you have a good internet for uploading big video files

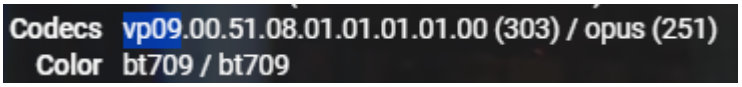
Successful 8K AV1?

- If successful, after 4 ~ 24 hours of the upload, the youtube video will be re-encoded again with Codecs **AV1**(**av01)**, and the **8K option** will be available for viewers also. Viewing in **4K/8K** will have less visual compression artifacts than any other lower resolution options like 1440p/1080p





- If not successful, after the upload, the youtube video will keep it's **VP9(vp09)** Codec forever, and the 8K option will not be available forever



Upload Resolutions

Here are some example upload resolution and there result, you need 8K to force Youtube using **AV1**:

- upload 3384x1440 = (bad) = VP9, 3384x1440, 4K (no 8K option)
- upload 6768x2880 = (bad) = VP9, 3840x1634, 4K (no 8K option)
- upload **7680x4320** = (good) = **AV1**, 7680x4320, **8K**
- upload 10152x4320 = (good) = **AV1**, 7680x3268, **8K**

8K AI Upscale tools

The “Handbrake upscale” method mentioned above is only upscaling the video without modification, so it will not add any new details, it is only for tricking Youtube to use a better AV1 encoder.

If you need a sharp result in 8K, you need an AI upscaler, you can try the follow tools:

- topazlabs - Video AI (can try for free with watermark on the output video)

but if you are uploading 8K for Youtube, likely these AI Upscale tools will not help much, since youtube compress your 8K video heavily, the sharpness and details gain from the AI upscaler will likely be removed visually.

OBS Youtube 4K streaming

When producing any live stream via youtube, setting the correct **OBS** settings will let your live stream looks much better than your competitors, below are some important settings in OBS that can improve live stream graphics quality massively:

PC requirement

This section is recorded using the following PC:

- GPU = **RTX4080** Super
- CPU = Intel(R) Core(TM) **i7-14700F** 2.10 GHz
- Internet = **100+ Mbps** upload speed internet

You may need a PC that is similar or better.

Best Resolution

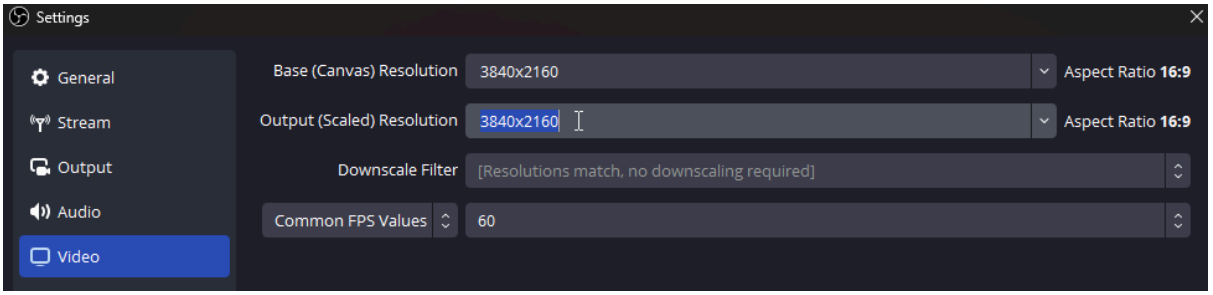
This is an important setting in OBS, make sure it is correct.

Youtube always transcodes(re-encode) your live streaming video to their VP9 WebM video format with different resolutions for viewers to choose from, so viewers will never see the source(original) quality.

To let Youtube produce the best VP9 transcode(re-encode) result in all resolutions, you should **always use Output Resolution = 3840x2160** in OBS's **Video** section, even if your **Base (Canvas) Resolution** is 1920x1080 / 2560x1440 / 3840x2160.

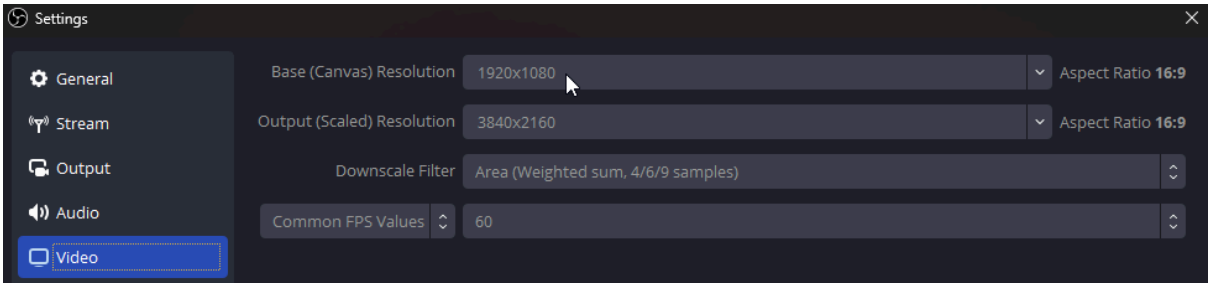
[Example settings A]

If you want to produce the **best graphics** in youtube, always use **3840x2160** for both **Output Resolution** and **Base (Canvas) Resolution**, **Downscale Filter** will be turned off automatically. (See the image below)



[Example settings B]

Let's say your OBS **Base (Canvas) Resolution**(e.g., 3DLive/Game window/Warudo window) is only 1920x1080, but you want youtube to produce 4K(3840x2160) quality transcoding to improve viewer's experience, use **Base (Canvas) Resolution** = **1920x1080** but **Output Resolution = 3840x2160**, OBS will output 3840x2160 and upload to youtube, and youtube will treat your live stream as 4K normally and transcodes in 4K quality instead of 1080p. **You don't need a 3840x2160 monitor to use this setup.** (See the image below)

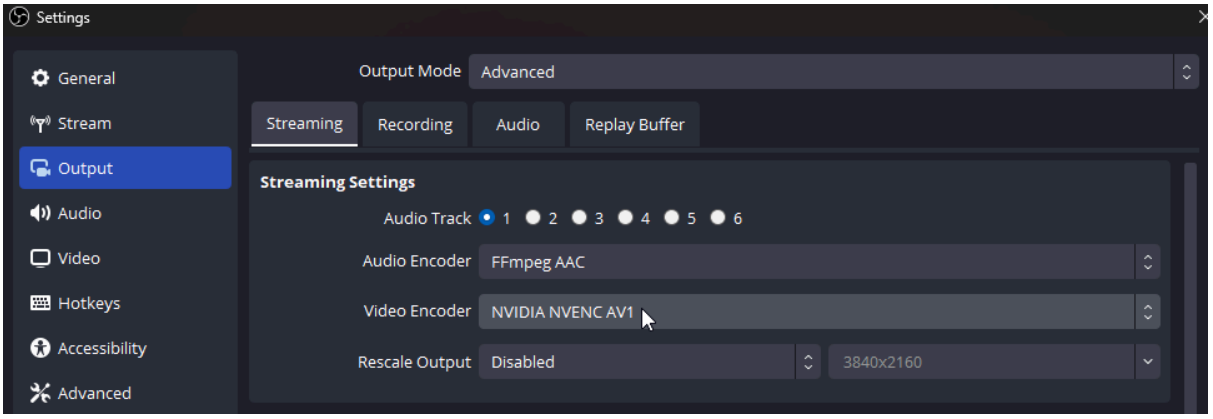


*In Youtube, when your viewer watches a 4K(3840x2160) video on their 1920x1080 monitor, a 4K(3840x2160) video will still produce a much better visual quality than a 1080p video, even if your **Base (Canvas) Resolution** is only 1920x1080.
So you should always live stream in **Output Resolution = 3840x2160** if you have a good PC and internet, there is no real disadvantage if your PC and internet can handle this output resolution. The only possible disadvantage is, when your viewer has bad internet, and they still manually pick 4K (higher than youtube's auto setting according to screen resolution and internet speed), which is beyond their internet speed, this will result in stutter streaming .

*On the other hand, if your **Output Resolution** is only **1920x1080**, no matter what OBS encoding settings you use, youtube will always destroy your live stream's visual quality due to their bad quality 1080p VP9/AVC transcoding, so you should at least set **Output Resolution** = **2560x1440** or **3840x2160** in most situations, never set **Output Resolution** = **1920x1080** unless your PC/internet is not good enough.

Best Encoder

This is an important setting in OBS, make sure it is correct.



Encoder will affect the encoding performance and visual quality greatly!
In practice, always use your GPU's Hardware encoder -> **NVIDIA / AMD / QuickSync(Intel)**

In the list below, you should pick the highest option if you see it in your OBS's Output > Streaming > **Video Encoder** drop menu:

- **NVIDIA NVENC AV1** (the most modern GPU encoder, great in performance and visual quality. **Always pick this if you see it**. Requires **RTX 40 series** or higher)
- **NVIDIA NVENC HEVC** (GPU **h.265**, will produce a slightly worse result than the above AV1, but overall still good enough. Requires **GTX 1650 rev 2** or higher)
- **NVIDIA NVENC H.264** (Visual quality is bad when compared to the above AV1 or HEVC(h.265) options, use this only if the above options are not available)
- **x264** (the old **CPU** encoder, bad quality, slow for 4K60fps**, don't use it**)
- **SVT-AV1** (**CPU** encoder, very high quality, but too slow for 4K60fps**, don't use it for streaming**)
- **AOM-AV1** (**CPU** encoder, best quality, but too slow for 4K60fps**, don't use it for streaming**)

RTX GPU Series	H.264	HEVC	AV1
RTX 20 Series	Yes	Yes	No
RTX 30 Series	Yes	Yes	No
RTX 40 Series	Yes	Yes	Yes

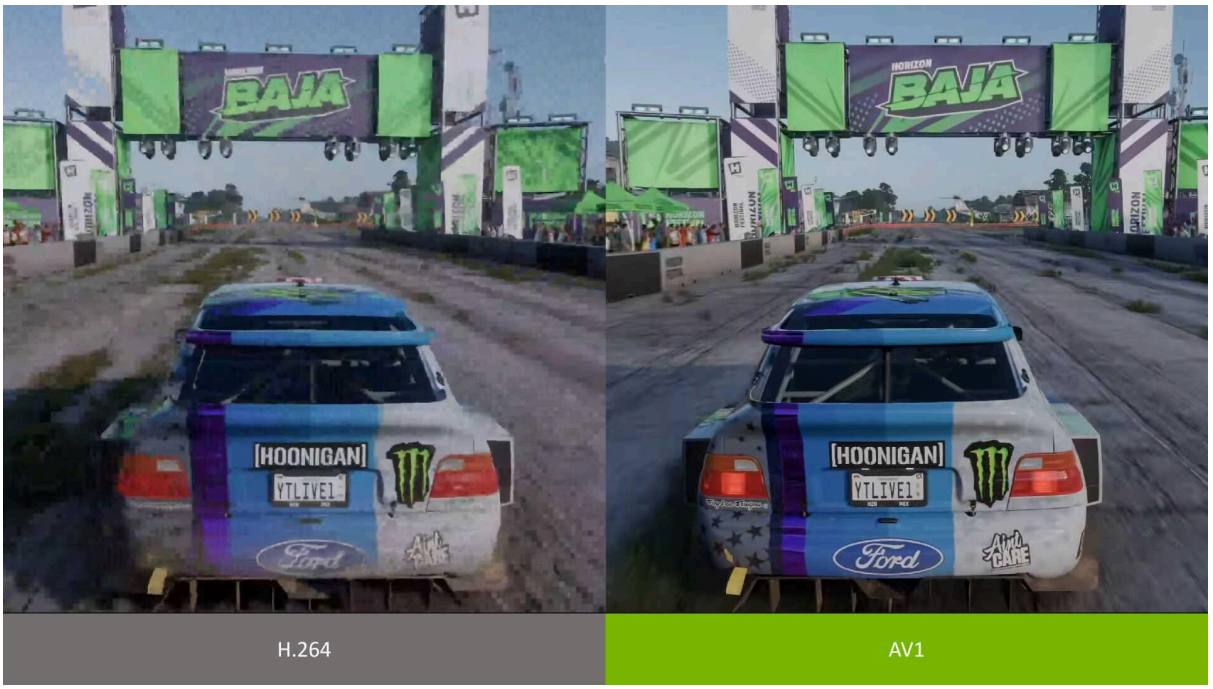
*When Comparing encoder's visual quality with the same **bitrate**(e.g., 6Mbps):
[best quality] AV1 > HEVC(H.265) >>>>>>>>>>>>>>>> AVC(H.264) **[worst quality]**
so you should always avoid any encoder with the name **264** (e.g. H.264/x264/AVC) for streaming if you have other better options like **AV1** or **HEVC(H.265)**.

It is expected that H.264, H.265, and even H.266 will become less relevant in the coming years for streaming, as they are likely to be increasingly replaced by AV1. This is due to several key advantages of **AV1**:

- **Better Quality at Low Bitrates**: AV1 offers significantly better quality, especially at low bitrates (e.g., less than 6Mbps), which is crucial for streaming services.
- **Wide Industry Support**: AV1 is developed and used by major tech companies such as Google (YouTube), Facebook and Netflix, soon other companies like Twitch will follow the new AV1 standard.
- **Open Source**: AV1 is open source, allowing any company to support it freely.
- **Royalty-Free**: AV1 is royalty-free, meaning companies can use it without incurring licensing fees. This is a significant advantage over **HEVC (H.265) / VVC (H.266)**, which has struggled with mass adoption due to its complex and costly licensing structure.

Relative Compression efficiency of codecs

H.264	HEVC	AV1
1.0x	1.15x	1.40x

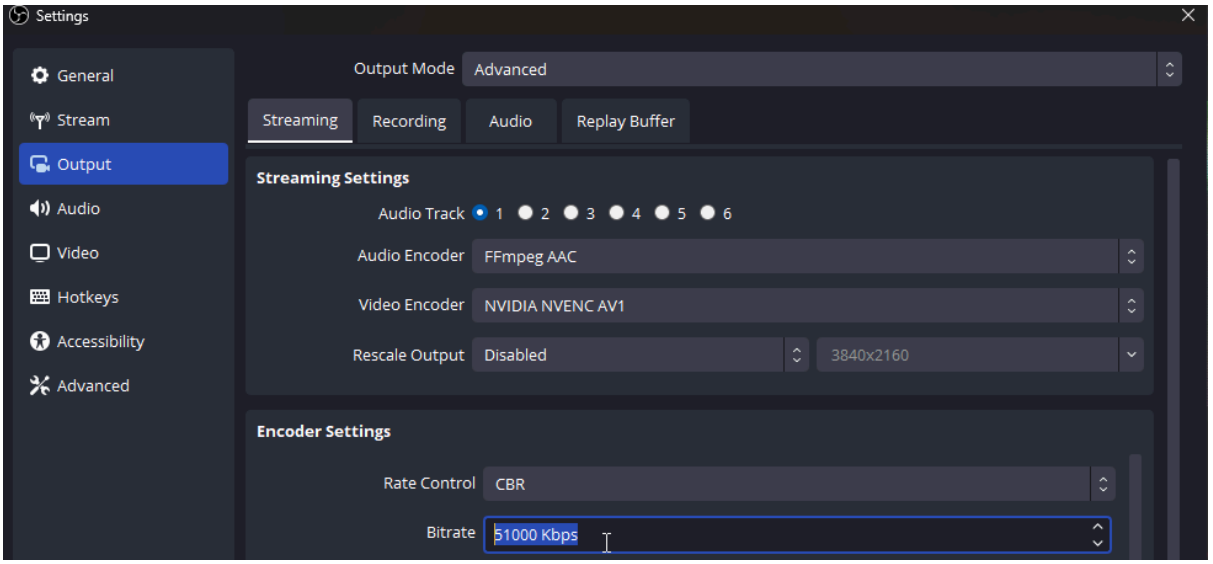


4K Video Quality comparison: **H.264** vs. **AV1** at 10Mbps
(**AV1** is great for fast motion graphics with limited bitrate, like 3DLive / game streaming, while **H.264** is the worst)

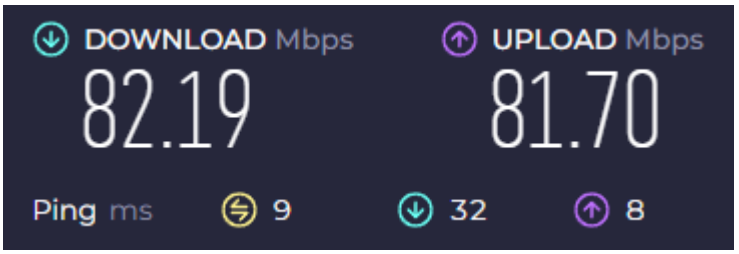
Best Bitrate

This is an important setting in OBS, make sure it is correct.

Youtube recommends never going over **51000 Kbps(= 51 Mbps)** when using OBS 4K60fps streaming, so you can use **~50000 Kbps** Bitrate for **4K60fps** OBS live streaming.



*To know what Bitrate is the best for you, first check your upload speed using SpeedTest. To stream at **50000 Kbps(= 50 Mbps)** Bitrate, it is recommended to have **75Mbps** (= **50 Mbps** * 150%) or higher **upload speed** for a stable and high quality youtube 4K60fps streaming, since your audio and other software like web browser, discord or your game will use some of your upload bandwidth.

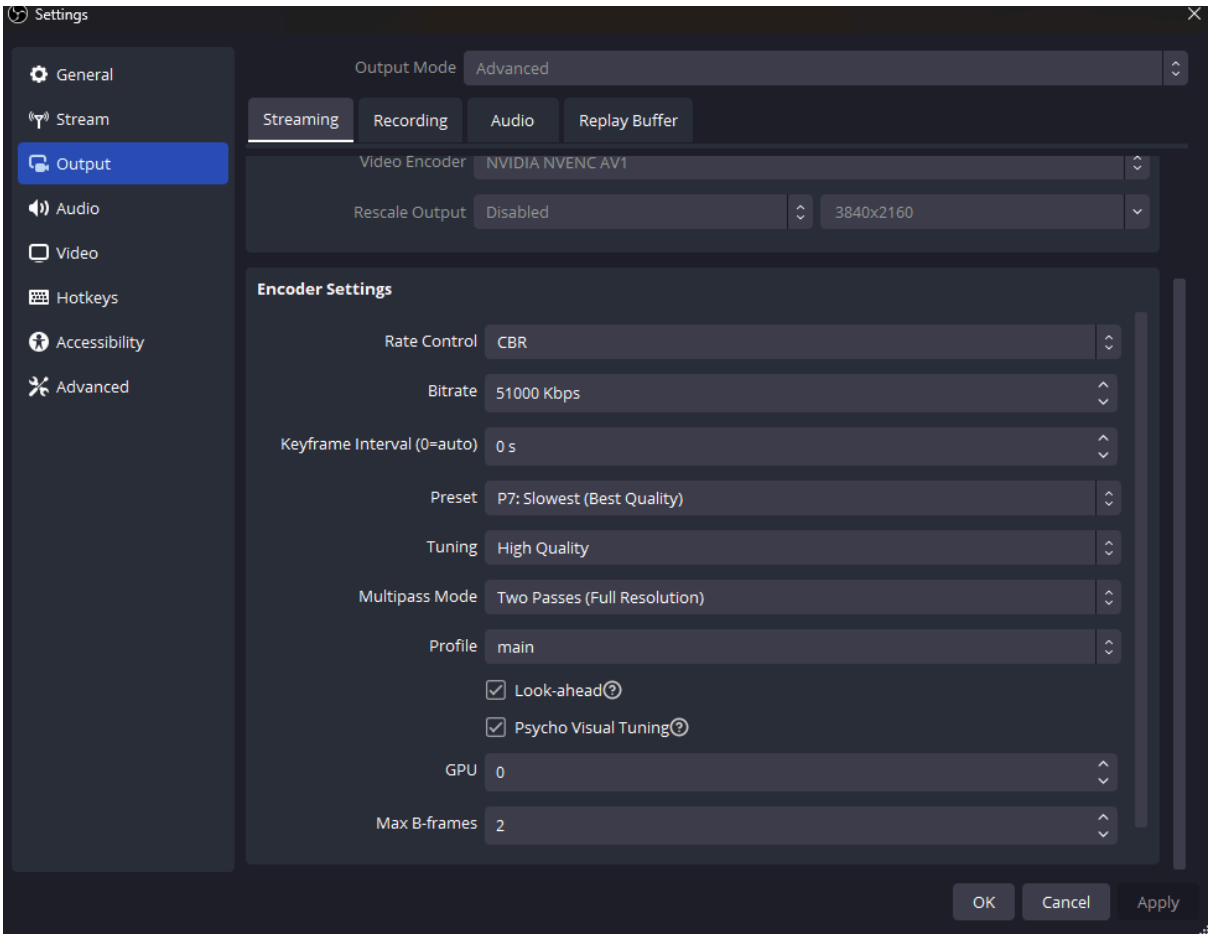


If your SpeedTest **upload speed** is far lower than the image above(e.g., only 40Mbps upload), you will need to use a lower **Output Resolution**, **Bitrate** or **fps** to produce a stable live stream.

To VTubers: Do not open SpeedTest while streaming, it will display your **IP** and **location** in the browser!

Best Encoder Settings

These are other settings that can also affect visual quality and performance, it is recommended to follow these settings, although it is not a must.



- **Rate Control** = CBR
- **Keyframe Interval** = 2
- **Preset** = try P7:Slowest (Best Quality) first, use lower(e.g., **P6~P5**) if limited by encoding performance. If you are using **NVIDIA NVENC HEVC**, likely you should always use **P6 ~P4** due to encoding performance cost, **NVIDIA NVENC HEVC + P7** may destroy your live stream's fps.
- **Tuning** = High Quality
- **Multipass Mode** = try Two Passes (Full Resolution) first, use Two Passes (Quarter Resolution) if limited by encoding performance
- **Profile** = main (use high when using h.264)
- **Look-ahead** = try On first, use Off if limited by encoding performance
- **Psycho Visual Tuning** = On
- **GPU** = 0 (set it according to your PC)
- **Max B-frames** = 4 (reduce to 2 if **Look-ahead** is off)

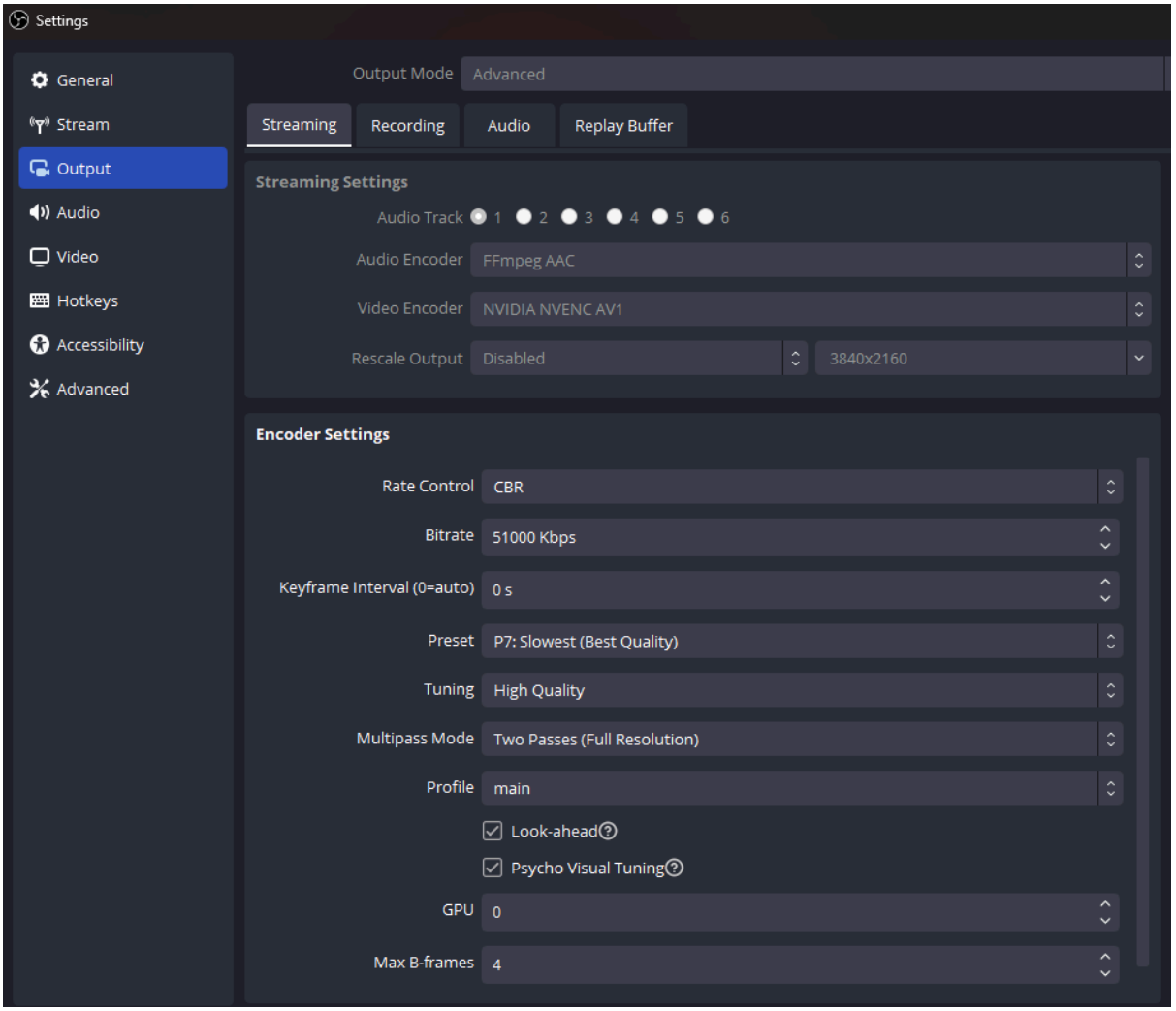
If you are **recording** instead of **streaming**, override these 3 settings:

- **Rate Control** = don't use CBR -> use **CQP** (VBR also ok)
- **CQ Level (CQ)**: 15 (you can **decrease** this number to get higher quality, but it will generate a larger file size).

- **Bitrate and Max Bitrate (VBR)**: 40,000 Bitrate; 60,000 Max bitrate. You can increase these to 100,000 and 200,000 (respectively) for higher quality.

Our OBS settings

Here is our usual OBS settings for 4K60fps Youtube 3DLive for your reference, it assumes a very good PC with **RTX40** GPU** since it uses **NVIDIA NVENC AV1** encoder, and assumes good internet with stable **80+ Mbps upload speed**.



If you have a good PC, copying the above OBS settings will let you live stream 4K60fps in Youtube with highest quality.

Nvidia's NVENC OBS guide

You will find a detailed explanation about why the settings above are the recommended settings for 4K60fps **Youtube streaming** for NVIDIA GPU. It also listed the recommended settings for 4K60fps **recording**.

- NVIDIA NVENC Obs Guide | GeForce News

Streaming Platform difference

The above settings are all targeted for **Youtube** 4K60fps live streaming only, those settings will not work for other streaming platforms!

(last update: 2024-06-20)

Youtube

- Max bitrate: **51Mbps**
- Encoder: **H.264, HEVC(H.265), AV1**
- Max output resolution: **4k60fps resolution**
- Always transcodes to lower bitrate VP9 WebM before delivery to viewers, so viewers will never see the source(original) quality. To produce the best quality, your job is to upload the highest bitrate streaming to Youtube and let Youtube handle the transcode.

Twitch

- Max bitrate: **6-8Mbps** (non Twitch-partner), where **6Mbps** is the guaranteed. If you go over **8Mbps**, your stream will be rejected by twitch (Higher bitrate is in beta)
- Encoder: **H.264** only (Twitch + AV1/HEVC is in beta).
- Max output resolution: **1080p60fps** (Twitch + 4K60fps is in beta). Due to such a low max bitrate limit and limited to H.264, you may want to try reducing to **720p~900p** for better viewing experience for games with fast motion like fps games
- Different from Youtube, viewers can choose **Source** quality (Quality settings not guaranteed to be available for non Twitch-partner), which means viewers will receive what you are streaming without any transcode to lower bitrate videos.

Bilibili

- (similar to Twitch)

AfreecaTV

- (similar to Twitch)

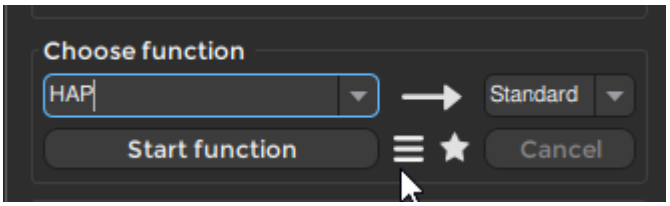
VideoPlayer&Recorder sync problem

If you find that the VideoPlayer's playback is not in sync when using Unity's recorder to record,
There are 2 solutions to make VideoPlayer's playback sync when using Unity's recorder.

KlakHap

(**Recommended**, works with Timeline Preview, we only use this)
Consider using **HAP codec .mov** videos with GitHub - keijiro/KlakHap: HAP video player plugin for Unity, **KlakHap** allows perfect recording with recorder, and perfect **preview playback using Timeline** in real time also, great for working on MV/cutscene in edit mode timeline(drag in timeline and preview instantly, boost productivity a lot).

- KlakHap only accepts **HAP codec .mov**, you can use Shutter Encoder to convert your video to **HAP codec .mov**, simply using the settings in the image below



VideoPlayer Nilo script

(**Not recommended**, doesn't work in Timeline Preview, not user friendly, we don't use it anymore)
1.Attach NiloToonURP folder's script **VideoPlayerForceSyncWithRecorder** on the **video player** component
2.Make sure the **fps** settings on the script is correctly matching your **recorder's fps** setting(e.g., 60)
3.Only enable **VideoPlayerForceSyncWithRecorder** when in recording, **don't enable it for normal gameplay**

*both Solution A/B's video audio will not output, you should play a separated audio in Timeline instead

Tools for 3D Live/MV

Here is a list of tools that we may use in this kind of 3D Live/MV project for your reference:

- NiloToon Unity6 Concert demo Project
- 닐로툰 콘서트 데모 4K 60fps (nilotoon concert demo 01 4K 60fps)

DLSS - AA / 4K upscaler

*__DLSS__ is highly recommended if you are using **NVIDIA RTX** card and targeting 4K/8K, it is the best upscale+anti aliasing tool when your target RenderScale is 0.333 ~ 1 for 4K/8K game window, replacing all URP's Camera AA(FXAA/SMAA/TAA):

- when used **without upscaling** (RenderScale = 1), **DLSS.NativeAA** can be used as a '**better TAA**' with **RenderScale locked to 1**
- when used **with upscaling** (RenderScale = 0.333 ~ 0.833), usually for a 4K/8K game window only, it can **boost GPU performance** due to lowering **RenderScale**. Will increase fps if you are **GPU bound**, for example, when you render 4K/8K with lots of volumetric fog and lights, with heavy postprocess.
- You can **use MSAA together** with these AA solutions to produce a **more stable Classic Outline**, but usually you don't need MSAA when using DLSS, since it will increase VRAM pressure a lot but only in return with a small gain in quality

Assets:

- DLSS (**best** graphics result, **best AA when RenderScale = 1**, but limited to **NVIDIA RTX GPU**) (We only use **DLSS** and ignore all other options below)
- XeSS2 (better visual than FSR3, works on **all GPU for windows x64, requires DX12**)
- FSR 3 (**ok** graphics result, works on **all GPU and platforms**, except VR & WebGL. Works on mobile too but it is too slow for mobile.)
- SGSR2 Mobile (A possible SR tool for mobile)
- SGSR1 Mobile (**Not recommended**: although GPU performance is very good on mobile, this asset produces **bad graphics results** for **Classic Outline**, you can compare SGSR1 Mobile with **TAA(Very Low)** or **MSAA 2x/4x** and see which one is a better option for your mobile game. We don't use this asset since the result looks like no AA with a bad sharpen filter, we would rather use **MSAA** or **TAA** due to these AA are both much better option for **Classic Outline**)

*The above upscale tools are sorted by their relative visual result quality
*For more info: Anti-aliasing (AA) Guide, DLSS | FSR3 (better fps & AA)

Planar Reflection

Usually for a 3D Live stage ground's sharp reflection rendering like a mirror.

Pros:

- Can produce perfectly sharp reflection, without any screen space reflection's artifact due to missing info on the screen

Cons:

- High **CPU** and **GPU** performance **cost** due to the necessity of an additional camera for each planar surface, each rendering the scene again into a RenderTexture

- blurring the reflection has extra **GPU** performance **cost**

Assets:

- PIDI : Planar Reflections 5 - URP Edition
- Mirrors and reflections (may produce better performance)
- kMirrors (free to use)

Screen Space Reflection

Usually for a 3D Live stage ground's rough/blurry reflection rendering, but it is not limited to a single planar surface, for example, you can also use it on any number of objects with any shape.

Pros:

- Low CPU performance cost
- Scalable GPU performance cost
- can produce blurry ground reflection

Cons:

- Will produce reflection artifacts when the screen has missing info, since it is a screen space method. (e.g., any background color blocked by a front object(e.g., character) will not be reflected and leaving a character shape hole in the reflection result)
- result can be quite noisy
- Rendering cost is higher in Forward/Forward+ when compared to Deferred

Assets:

- Shiny SSRR 2

Volumetric light & fog

If you need volumetric fog/light for 3DLive's spotlights, try these:

- Volumetric Light Beam (For adding simple spotlight cone effects mesh to spotlight, easy to control, fast to render, recommended)
- Volumetric Fog & Mist 2 inside Fog & Lighting bundle (Great for adding volumetric fog automatically to all spotlights after enabling Enable Native Lights option. Fog also has shadow automatically, which looks great when part of the spotlight is blocked by character)
- Volumetric Lights 2 inside Fog & Lighting bundle (Similar to Volumetric Light Beam, but with much more complex options)
- Dynamic Fog & Mist 2 inside Fog & Lighting bundle
- Lux URP Essentials (Fastest fog shapes shaders (box/sphere), but not many options, no shadows.) We use this for adding the far background's gradient fog/light

*Want to reduce the additive effect only for NiloToon character pixels on screen? see this

Vfx Graph - 6-way lighting

Extremely powerful tool for making smoke Vfx graphs that react to lighting, especially for concerts!

Realistic smoke lighting with 6-way lighting in VFX Graph

VFX Graph: Six-way lighting workflow | Unity at GDC 2023

Six-Way-Lighting-Textures

Light beam / laser performance system

A system to control spot light's rotation and light color animation in a timeline.

- GitHub - murasaqi/Unity_StageLightManeuver (free to use. We mainly use this as it is more powerful and complete than other options, but **CPU cost is heavy** when you need hundreds of spotlights due to StageLightManeuver's timeline track update)
- Unity_LightBeamPerformance (free to use)

Light's Timeline Track

If you only need to create a simple light color's timeline track, this official asset includes scripts for these tracks. It is also a great example code if you want to start writing your own timeline.

- Default Playables (free to use)

Material's Timeline Track

If you only need to create a material property(e.g., emission)'s timeline track, this asset includes scripts for these tracks.

- material-timeline (free to use)

LaserLightShader

Useful for 3DLive's stage lighting.

- LaserLightShader

Skylight Window LightShaft Shader

Useful for 3DLive's stage lighting.

- Skylight Window LightShaft Shader

Motion Blur

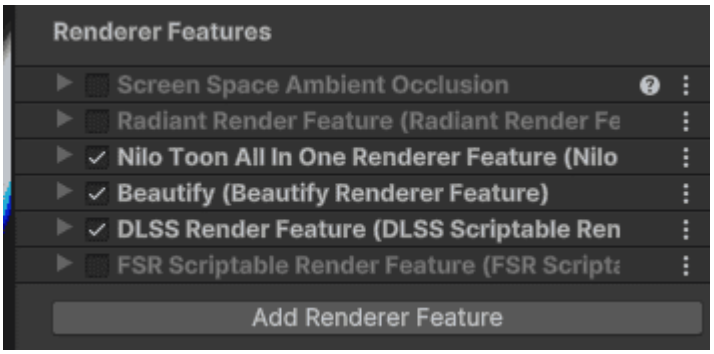
NiloToonMotionBlur volume is a higher quality alternative to Unity6's URP motion blur.

- For realtime use, **NiloToonMotionBlur** volume will produce better result, but with much higher GPU cost
- For rendering an offline video instead of realtime, also use **Windows/NiloToon/MotionBlurBaker** will produce a perfect quality motion blur

For details on how to use these tools, see NiloToon Motion blur tools

Utility PostProcess

- Beautify 3 (e.g., bloom of a horizontal flare shape)



(You should put **Beautify under** NiloToon's renderer feature, else NiloToon features like extra thick outline will not work since Beautify may clear the stencil buffer value)

- SC postprocess

Bloom

Bloom with Glare (e.g. X shape / Star shape), great for stylize light source bloom and specular reflection bloom

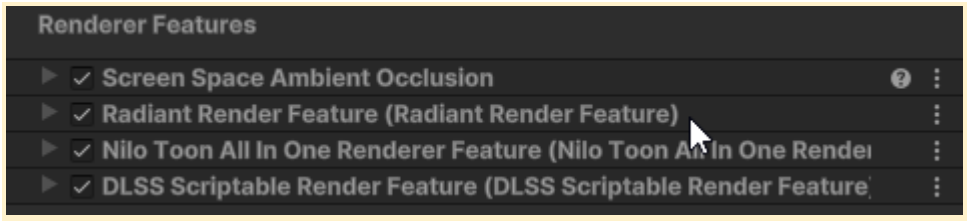
- MK glow

Screen Space GI

Can be used as an additional/alternative GI for scenes with mostly emissive material, no lights and lightmap baking is not allowed due to dynamic lighting/objects/materials.

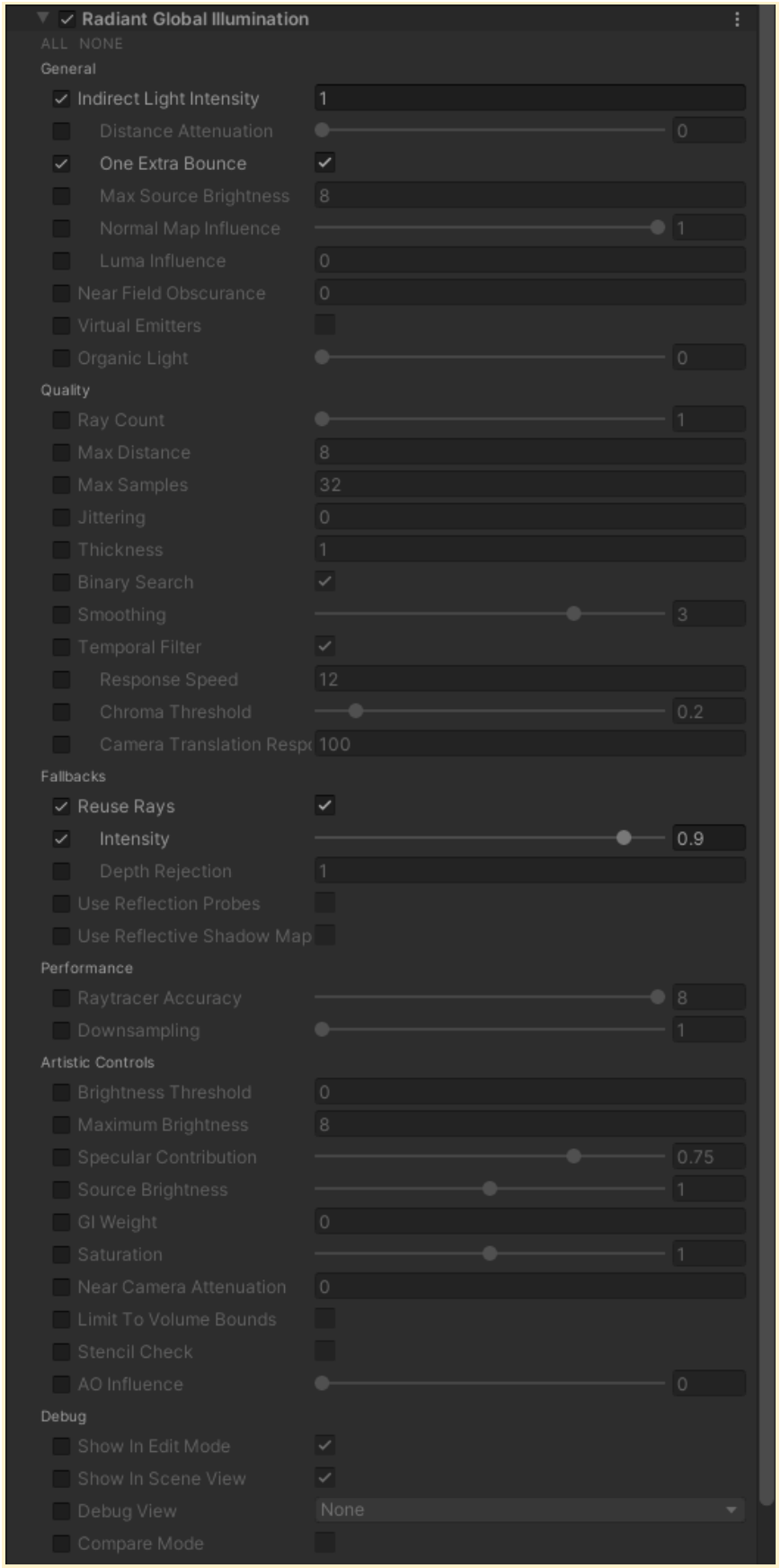
- Radiant Global Illumination

*Make sure the **Radiant Renderer Feature** is moved **on top** of **NiloToon All In One Renderer feature** to ensure NiloToonTonemap order correctness



*Want to reduce/remove the GI additive effect on character pixels? see this

We will usually start with these overrides, the default value is too noisy.



Ambient Occlusion

URP SSAO's alternative, slower but looks way better than URP SSAO.

- HTrace: Ambient Occlusion URP | Fullscreen & Camera Effects | Unity Asset Store
- Horizon Based Ambient Occlusion

Screen Space Cavity & Curvature

If you have a scene that looks too flat in shading, you can try this tool.
Can be useful for stylized scenes with lots of flat color.

- Screen Space Cavity & Curvature

Cloth,Hair,Breast Physics

Use **Magica Cloth2** if you can.

- Magica Cloth2 (Our solution for new projects)
- Magica Cloth (Our solution for old projects, Not recommended due to end of service)
- Dynamic Bone (Very old, but still usable)
- UnityChanSpringBone (Not recommended: development stopped for years)
- Cloth Dynamics (Not recommended: extremely complicated to correctly use it)

Mouth AIUEO animation

Generate AIUEO mouth animation from a song audio(bake) or microphone(realtime)

- uLipSync (free to use)
- Oculus LipSync (free to use)

*For more info: Song auto mouth AIUEO anim

Performant Video player

A video player that works with timeline preview + Recorder! Great for making concert MVs. Usually for playing a video on the big screen above the 3D live stage, which is behind the character.

Pros:

- works with timeline preview
- works with Recorder

Cons:

- Accept **HAP .mov** video only, HAP .mov is very big

Asset:

- KlakHap (free to use)

*For more info: VideoPlayer&Recorder sync problem

UniVRM

For importing .vrm as a character prefab, including tools like **VRM/MToon** shader and VRM **BlendShapeProxy** script. After the import, NiloToon can auto convert all **VRM/MToon** materials to NiloToon using NiloToon's auto setup character tool .

- UniVRM (free to use)

Chara Animation for testing

If you need some character animation for testing cloth physics or rendering(e.g. testing different Anti-aliasing in different game window resolution), you can use these assets:

- Anime Girl Idle Animations Free (free to use)
- Anime Girl Idle Animations
- UnityChan Candy Rock Star (free to use) (body and hand animation fbx in zip's **Assets/UnityChan/Animations/_.fbx**)
- UnityChan Unite In The Sky (free to use)

Lightmap baking

Increase project complexity due to baking lightmap, but it can produce better interior lightmap

- Bakery

Animation editor

If you need to create or adjust a character's 3d animation clip purely within Unity **without** using external tools like **Blender/Maya**, this tool may be useful. Usually to apply a quick fix to mocap animation.

- UMotion Pro - Animation Editor

IK animation

Great for generating new animation at runtime, for example, let the character grab a car wheel and only rotate the car wheel, then this tool will produce the character's arm animation automatically.

- Final IK

Stage object prefabs

Simple stage prefabs that may be useful if you want to make a concert stage from different small prefabs.

- Stage constructor
- PBR Stage Equipment
- GitHub - murasaqi/Unity_StageLightManeuve (free to use)
- LightBeamPerformance (free to use)

SkyBox for prototype

Use it when you just need some random Skybox to fill the sky and reflection probe for prototyping.

- Fantasy Skybox FREE (free to use)
- Fantasy Skybox
- Skybox Series Free (free to use)
- https://hdri-haven.com/ (free to use)

Dynamic procedural Skybox

If you don't want a static skybox, this tool maybe the best tool for a runtime dynamic skybox

- Enviro 3 - Sky and Weather

FPS Counter

For easily checking fps and system info in build.

- Advanced FPS Counter

Debugging

This is a great debugging tool for inspecting or editing any private field's value in runtime(including build) without using VisualStudio or Rider's breakpoint. Can boost productivity a lot when you are working on **programming debugging**. We rarely use it.

Used for:

- check any field value in runtime
- invoke any method in runtime
- compare memory snapshot to check memory leak(e.g. RenderTexture or material leak)

Assets:

- Debugging Essentials

Shader Graph Alternative

If there are some properties, nodes or settings that are not provided in **Shader Graph**, likely Amplify Shader Editor will have it.

- Amplify Shader Editor

Timeline Alternative

If Unity's **Timeline** is not powerful enough for you, you can try this, but usually you don't need it. We rarely use it.

- Timeflow

Compare prefab's difference

Port components from one character to another.

To ensure the components are the same between two characters.

This asset may be useful in these situations. We rarely use it.

- Inspector Compare - Component Differences

HDRP/BRP Material -> URP

If you have a scene/prefab with HDRP/BRP materials, and want to convert all materials to URP materials easier, you can try this tool. We rarely use it.

- SRP Material Converter

humanoid ⇆ generic format

Convert animation clips (*.anim) between all 3 animation types (humanoid ⇆ generic ⇆ legacy). We rarely use it.

- Animation Converter

Utility URP Shader

If you need a more complex shader than URP's Lit / ComplexLit shader, see if you find something useful from these assets. We rarely use it.

- Lux URP Esstenstails
- Better Lit Shader
- URP+ 2022 / URP+ 2023
-

Water Caustics Effect

Usually for 3D concerts with a stage in an aquarium theme. We rarely use it.

- Water Caustics Effect for URP v2

Character Auto LOD

If character vertices are causing performance problems, try this. We rarely use it.

- Poly Few
- Simplygon (Not recommended: Used to be the best tool, but now it is difficult to use due to new Simplygon **pricing**)

Anime style research data

We rarely use it. Likely you can ignore this, unless you are studying anime style postprocess graphics

- MIKONOTE Anime Toolbox Sample (free to download)

Hierarchy/Inspector UI

Useful if your scene/project is very complex, and need additional tools to help organize it.

- vHierarchy 2
- vInspector 2

MMD4Mecanim

Concert .pmx file to .fbx prefab in unity

- Stereoarts Homepage

Animate Volume

Animate Volume profile using animation. Without it you can only animate the Weight of volume.

- AnimatableVolumeComponent

Publisher

Maybe you will find some other useful assets from these publishers.

- Kronnect
- Amplify Creations
- noriben

Video Compare tools

- VMAF score (check if 2 videos are visually similar by a 0~100 score)
- Video Compare (check for visual differences between 2 videos)

Chara mask in any shader

This section requires basic knowledge about **hlsl / cg shader programming**.

There will be situations where you don't want additive effects from other assets apply on NiloToon's character pixels, since they will make the character over bright, which looks bad usually.

For example:

- You don't want **Volumetric light beam** to apply additive lighting on character pixels
- You don't want **Screen space GI** to apply additive lighting on character pixels
- You don't want **AmbientOcclusion** to apply color change on character pixels
- You want to control the apply strength of an effect on character pixels(0%~100%)

In this situation, you will need to modify the shader source code of that target asset(not NiloToon), to make them change how they apply their effect on NiloToon's character pixels.

Here are all the **hlsl** code you need to make it happen, add these code to your target asset's shader:

```
// anywhere above the target shader's fragment shader
TEXTURE2D_X(_NiloToonPrepassBufferTex);
```

```
// inside the target shader's fragment shader
// sample the NiloToon character mask
half characterArea = SAMPLE_TEXTURE2D_X(_NiloToonPrepassBufferTex, sampler_LinearClamp, uv).g; // NiloToon character area stored in g channel

// use the character mask to reduce/remove any unwanted effect on character pixels
half NiloToonCharacterAreaReceiveEffectStrength = 0; // 0~1 (0% ~ 100%)
anyAdditiveEffectThatYouWantToRemoveOnCharArea.rgb *= lerp(1,NiloToonCharacterAreaReceiveEffectStrength ,characterArea);
```

Common assets to edit

VolumetricLights

In **VolumetricLights\Resources\Shaders\VolumetricLightURP.shader**.hlsl > **frag** function

```
// Add this line anywhere above the target shader's fragment shader
TEXTURE2D_X(_NiloToonPrepassBufferTex);


// Add the following code to prevent VolumetricLights applying on NiloToon Character pixels, put this code right above the line "return color;"
half characterArea = SAMPLE_TEXTURE2D_X(_NiloToonPrepassBufferTex, sampler_LinearClamp, uv).g;
half characterAreaReceiveVolumetricLightStrength = 0; // 0~1 (0% ~ 100%)
color.rgb *= lerp(1,characterAreaReceiveVolumetricLightStrength , characterArea);
```

RadiantGI

In **RadiantGI_Upscale**.hlsl > **FragCompose** function

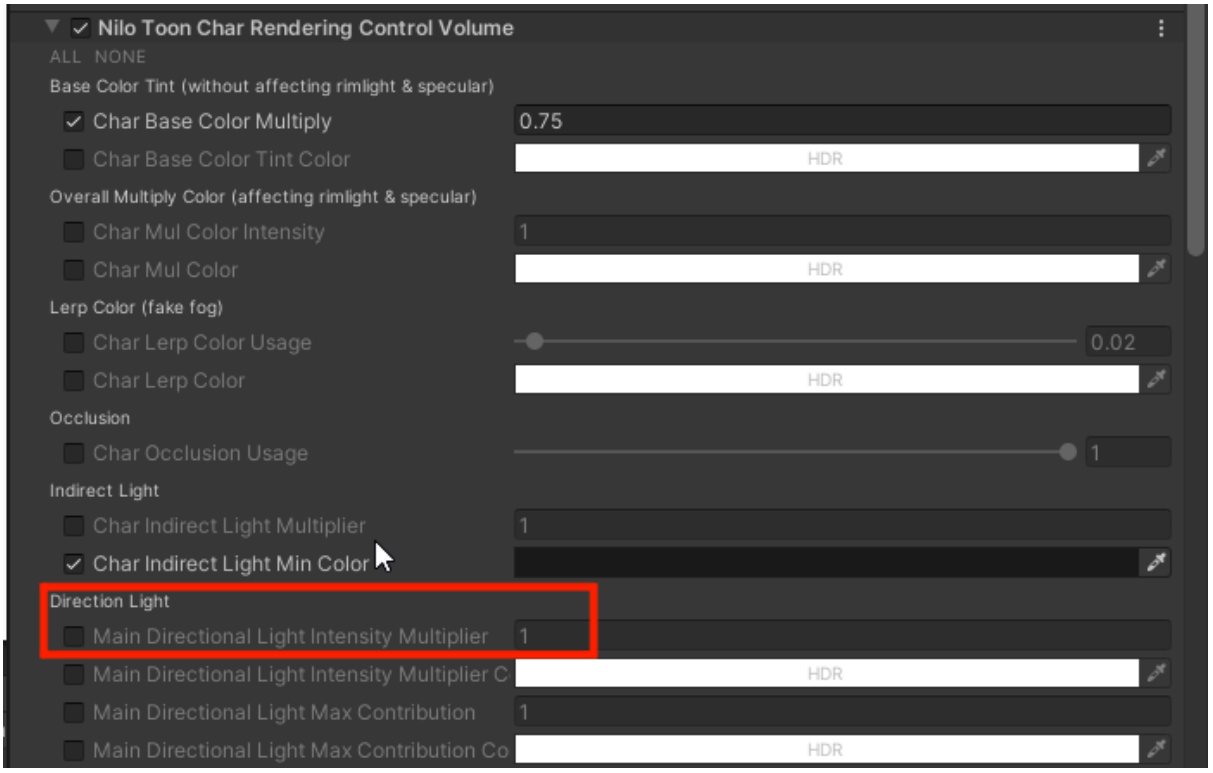
```
// Add this line anywhere above the target shader's fragment shader
TEXTURE2D_X(_NiloToonPrepassBufferTex);


// Add the following code to prevent RadiantGI applying on NiloToon
Character pixels, put this code right above the line
"// add GI to input image"
half characterArea = SAMPLE_TEXTURE2D_X(_NiloToonPrepassBufferTex,
sampler_LinearClamp, uv).g;
half characterAreaReceiveGIStrength = 0; // 0~1 (0% ~ 100%)
indirect.rgb *= lerp(1,characterAreaReceiveGIStrength ,characterArea);
```

Separate char/envi lighting

Due to URP's **Main Light** design, you will need to use **Main Light** for both character and environment together(don't use separated layers) in order to cast character shadow onto environment objects(e.g. floor).

In URP,
1.if SunLight(RenderSettings.sun) is present, it will be used as URP's **Main Light**
2.if SunLight(RenderSettings.sun) is not present, the brightest directional light will be treated as URP's **Main Light**, NiloToon requires URP's Main Light information to lit the character nicely.

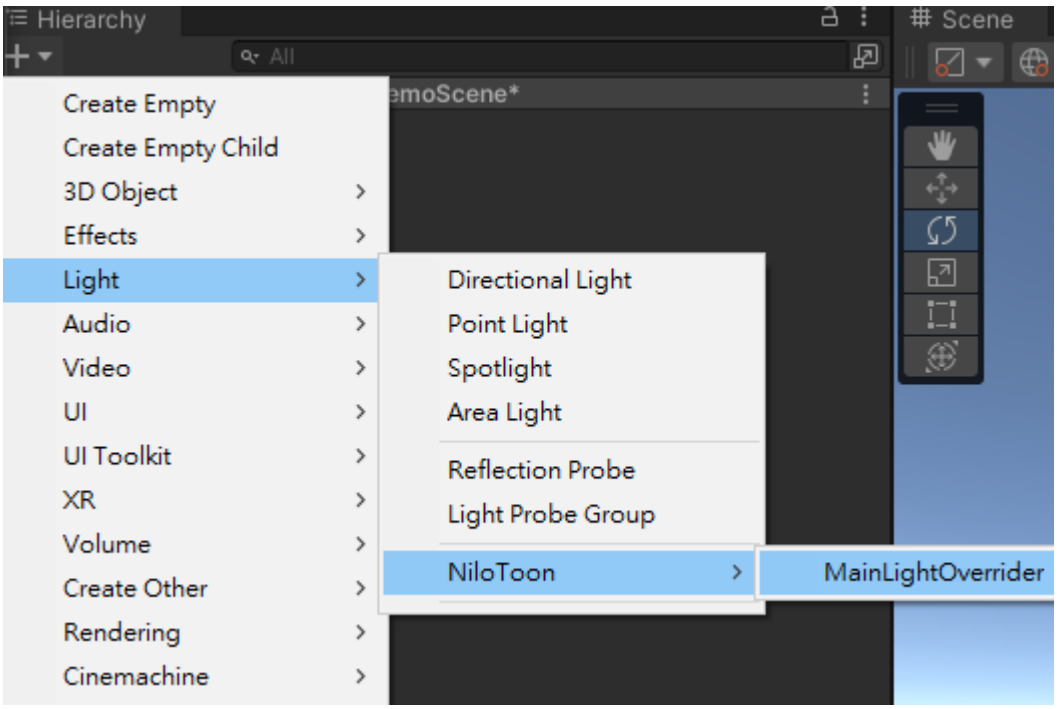
If you need extra **light intensity control on characters**, use **NiloToonCharRendereringControlVolume**'s **MainDirectionalLightIntensityMultiplier**

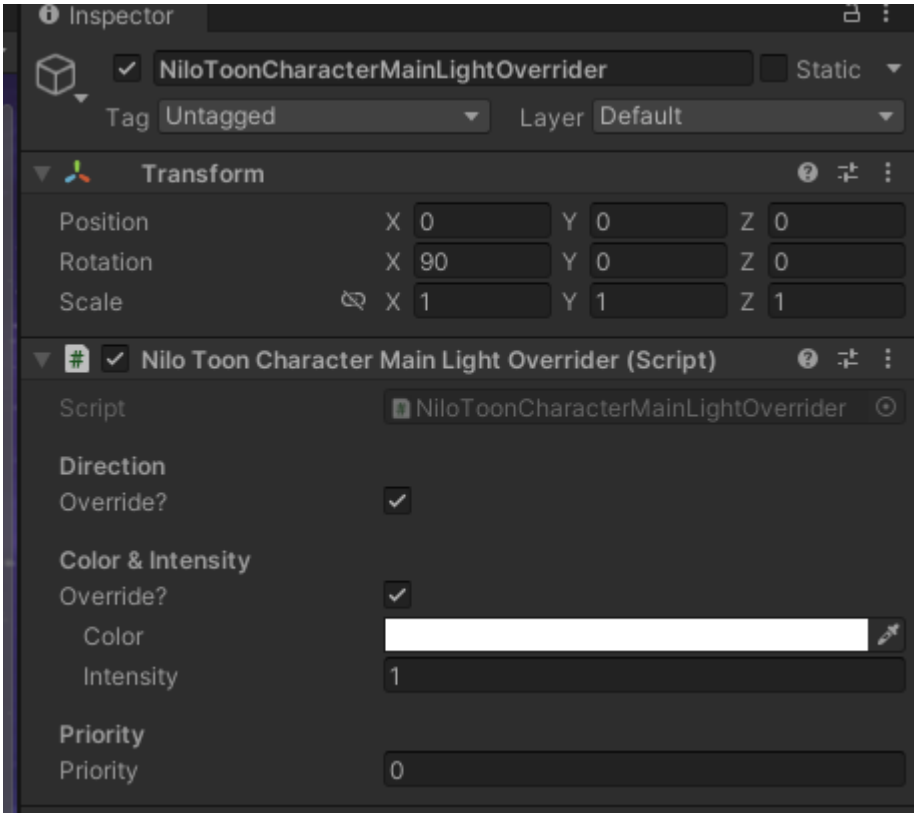


And if you need extra **light direction control on characters**, use NiloToonCharRendereringControlVolume's **OverrideLightDirectionIntensity**(e.g. set it to 1)

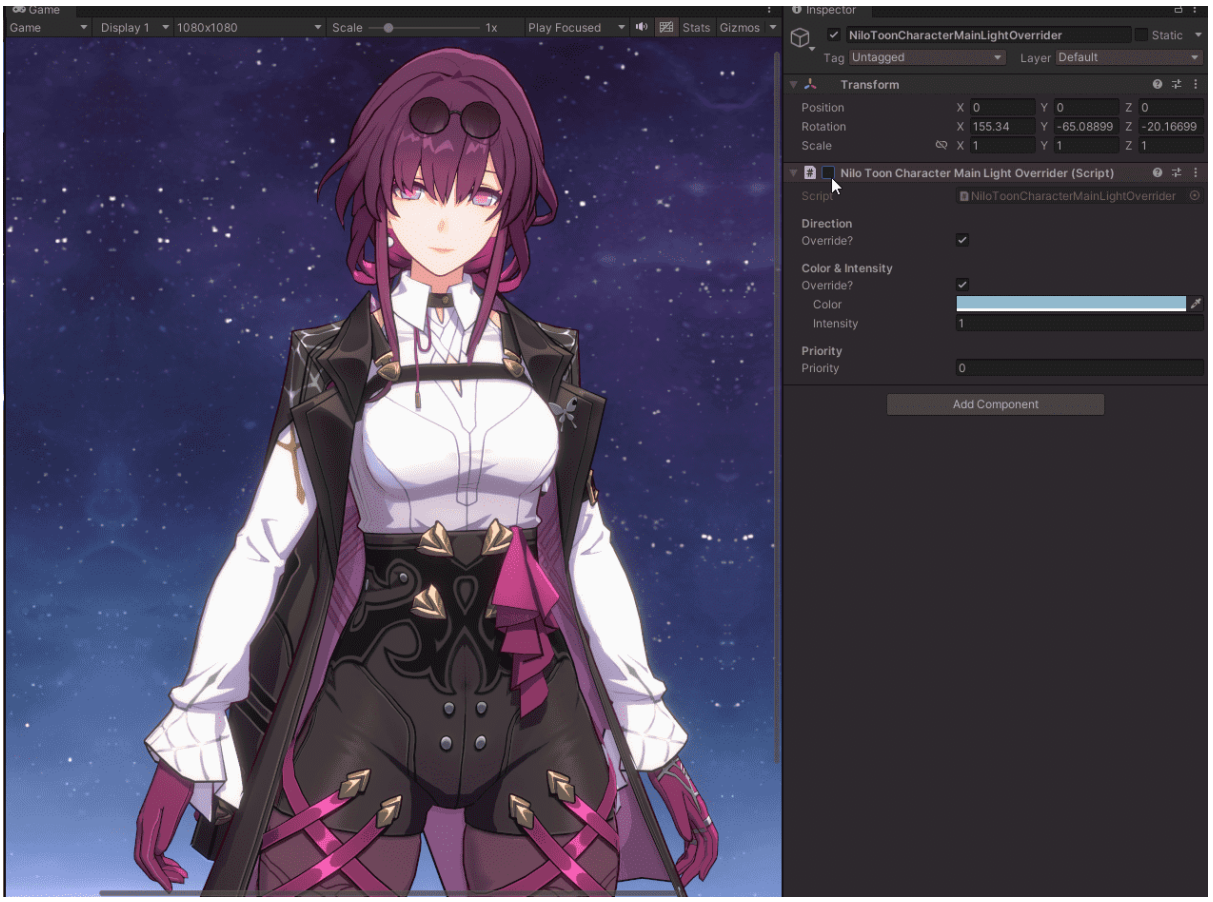
this will override the light direction for character only.

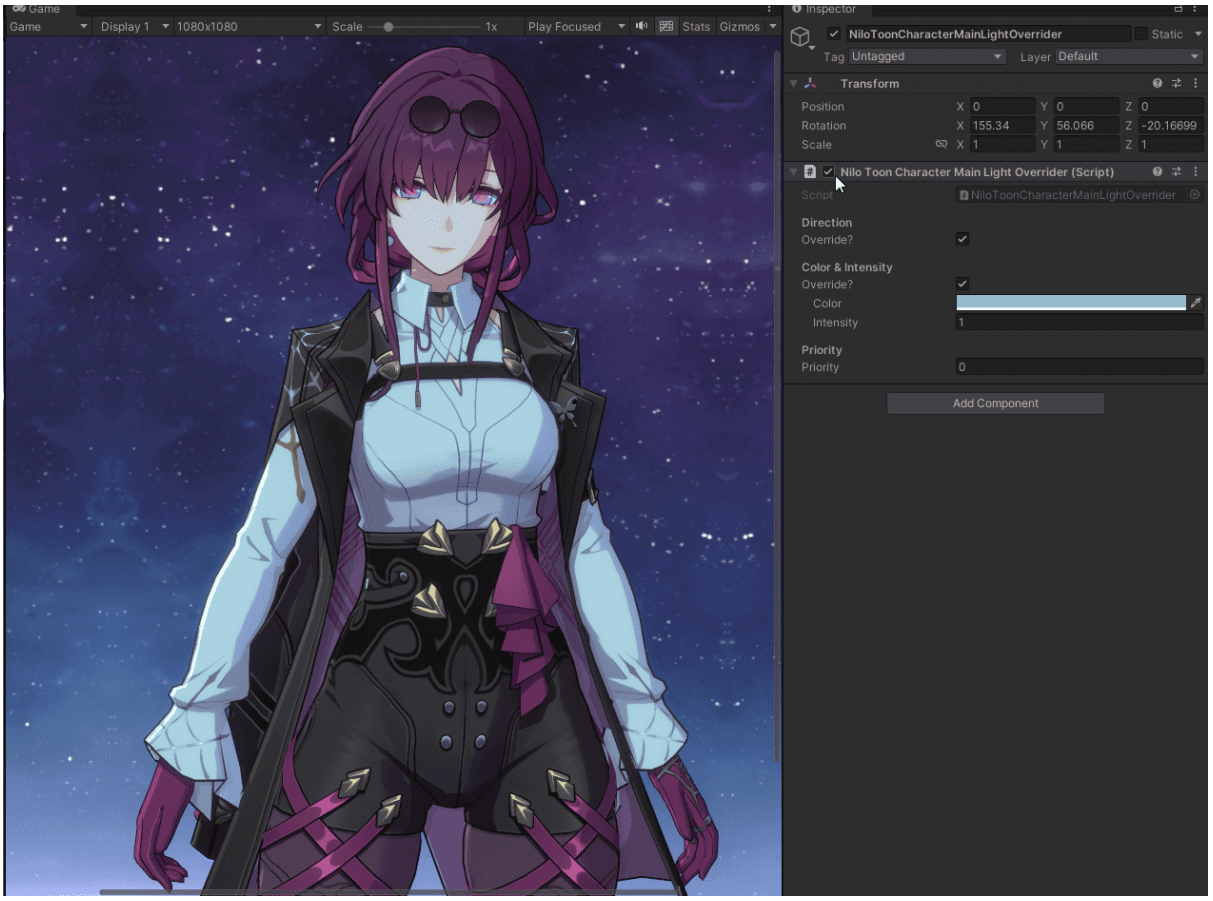
*In NiloToon 0.14.x or later, a new script NiloToonCharacterMainLightOverrider will let you override character's mainlight color and direction in a better way, without editing any URP's directional light. So you can set up main lighting for the environment first, and only override the character's lighting if it doesn't look good.





You can treat NiloToonCharacterMainLightOverrider as a Directional Light that affects NiloToon character only

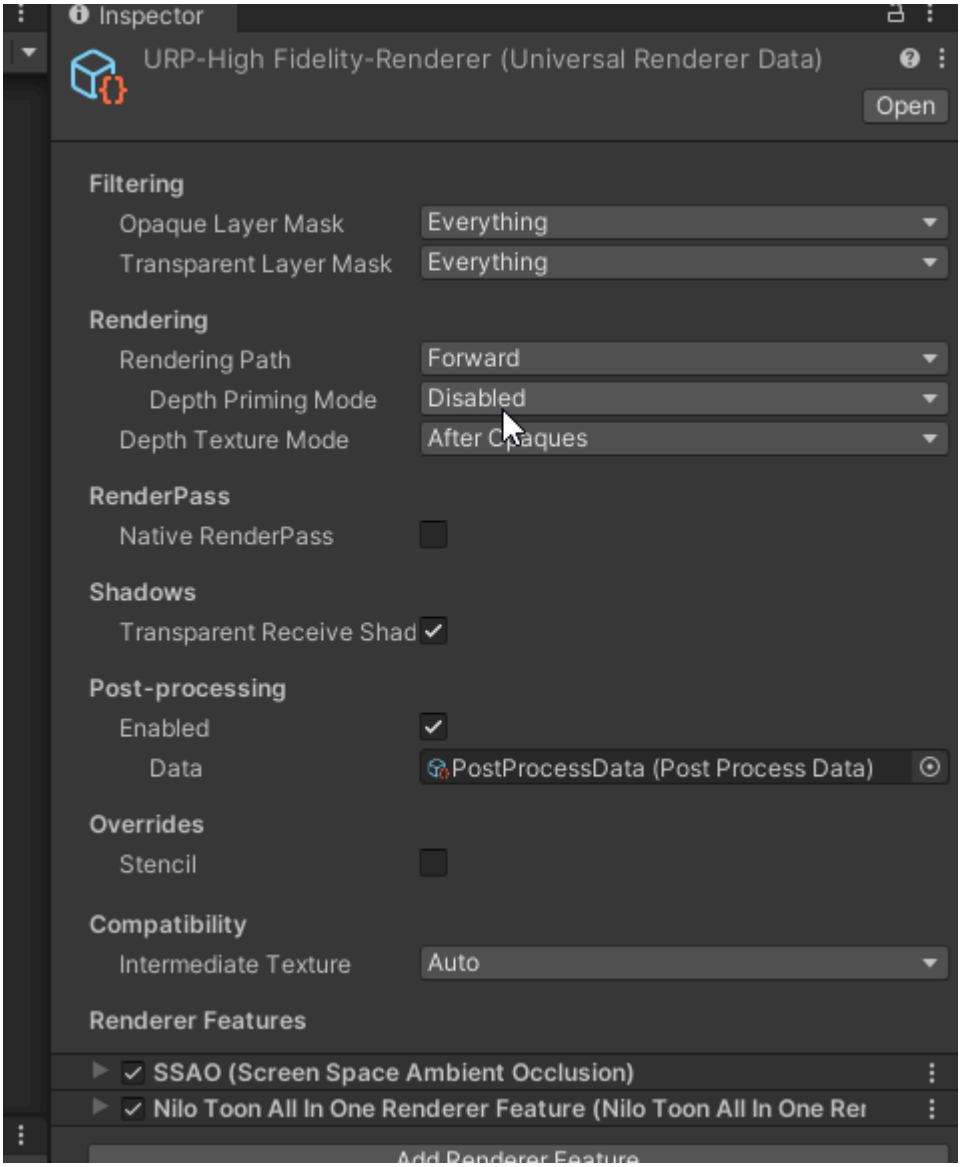

Game
Game
Display 1
1080x1080
Scale
1x
Play Focused
Stats
Gizmos
Inspector
NiloToonCharacterMainLightOverrider
Static
Tag
Untagged
Layer
Default
Transform
Position
X 0
Y 0
Z 0
Rotation
X 155.34
Y -65.08899
Z -20.16699
Scale
X 1
Y 1
Z 1
Nilo Toon Character Main Light Overrider (Script)
Script
NiloToonCharacterMainLightOverrider
Direction
Override?
Color & Intensity
Override?
Color
Intensity
1
Priority
Priority
0
Add Component


Game
Game
Display 1
1080x1080
Scale
1x
Play Focused
Stats
Gizmos
Inspector
NiloToonCharacterMainLightOverrider
Static
Tag
Untagged
Layer
Default
Transform
Position
X 0
Y 0
Z 0
Rotation
X 155.34
Y 56.066
Z -20.16699
Scale
X 1
Y 1
Z 1
Nilo Toon Character Main Light Overrider (Script)
Script
NiloToonCharacterMainLightOverrider
Direction
Override?
Color & Intensity
Override?
Color
Intensity
1
Priority
Priority
0
Add Component

*2 images showing **NiloToonCharacterMainLightOverrider** overriding light color and light direction of character only, and not affecting other environment

Face 2D shadow missing

Select all your URP **3D** Renderers (search t:UniversalRendererData in your project window), Set **Depth Priming Mode** to **Disabled**, else face material using NiloToon_Character shader can not render 2D shadow on the face.

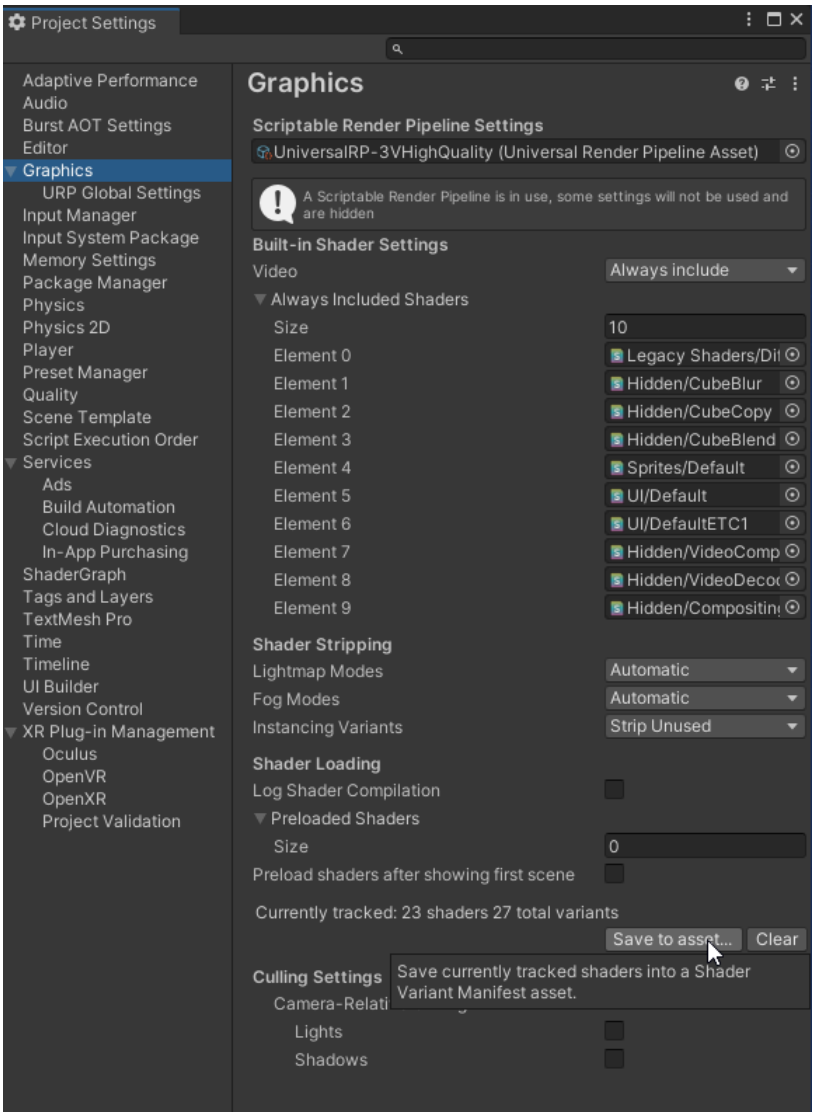


Asset Bundle miss variants of shader after build

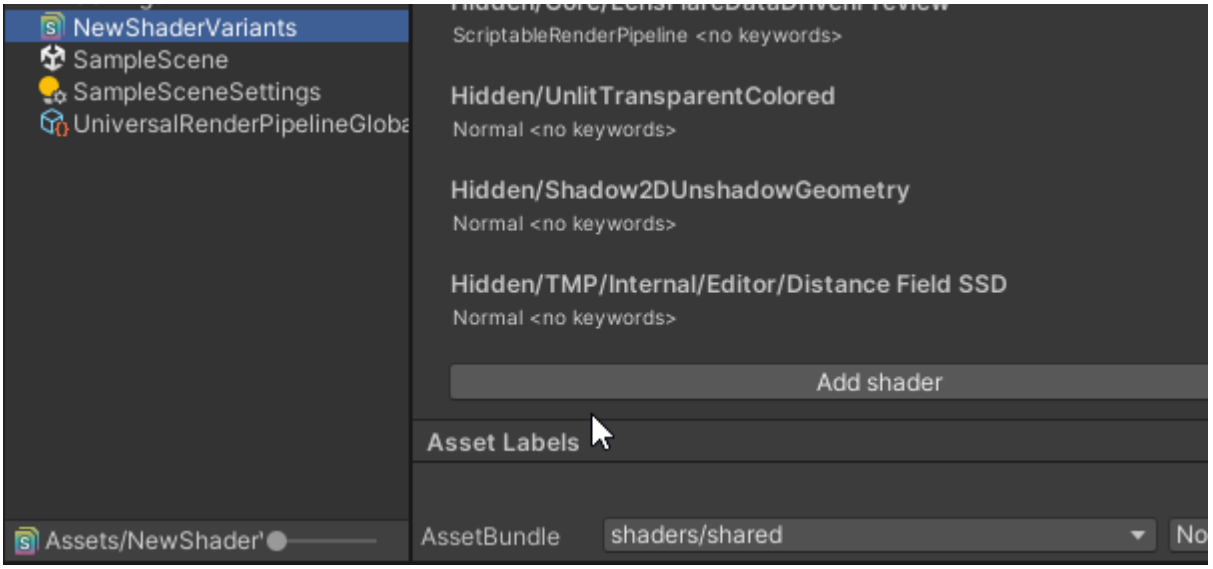
If the asset bundle that you built didn't include the required **shader variant** of the **NiloToonCharacter** materials, the rendering result will be wrong when you load that AssetBundle, since some shader variants are missing in the bundle.

If you **manually** add dummy materials(that have the target shader variant) to your shader asset bundle and rebuild the bundle, while it can work, it is impossible to do it perfectly when you have a large amount of characters, so we don't recommend doing it manually.

The better solution that we suggest is using "**Shader variant collection**",
step1) open your project that is used for building the asset bundles
step2) place all characters prefab that you need to build into a scene
step3) make sure the game camera can see all your characters
step4) enter playmode once
step5) due to step 2-4, all shader variants are now tracked by Unity Editor, now open ProjectSettings > Graphics > **Shader Loading** section > click **Save to asset**



step6) set the newly saved **Shader variant collection asset**'s AssetBundle tag to the **same tag of your NiloToonCharacter shader**, doing this will include the **Shader variant collection asset** into your NiloToon shader's asset bundle, which will hint Unity about what shader variants should be included when building the NiloToon shader's asset bundle



step7) rebuild all asset bundles again
step8) In your game, load the new assetbundle, all missing variant problem should be solved

If you want to check if you have any missing shader variants when using AssetBundle,
In Unity 2022.2 and above, you can force Unity to show a pink error shader during runtime, when a Material tries to use a missing shader variant.

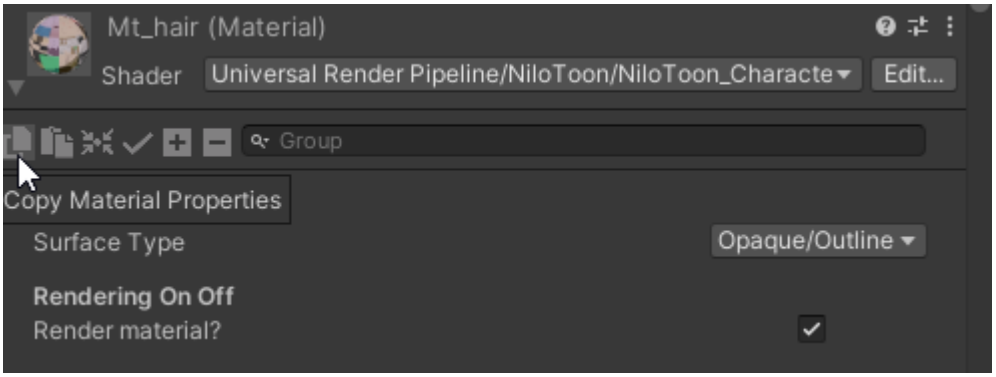
1. Go to **Edit > Project Settings > Player**.
2. Under **Other Settings**, in the **Script Compilation** section, select **Strict shader variant matching**.

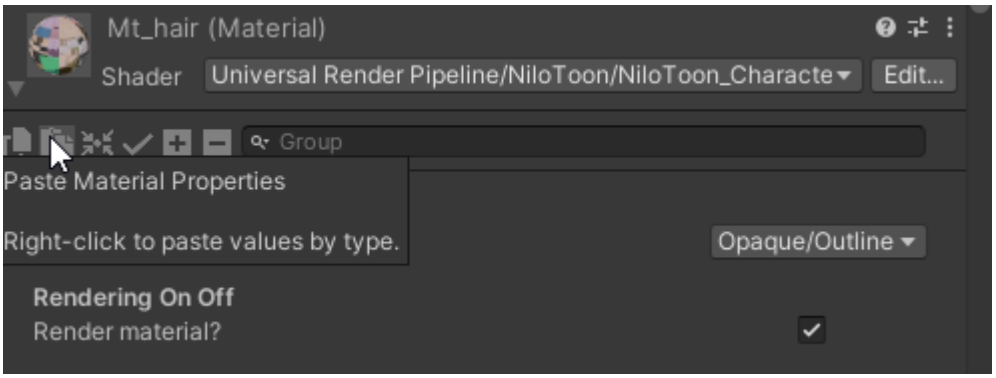
You can also enable this in C# using `strictShaderVariantMatching`.

When you do this, Unity shows a warning in the console with the missing variant and its keywords. You can use this during stripping to find any wrongly removed shader variants that your build needs.

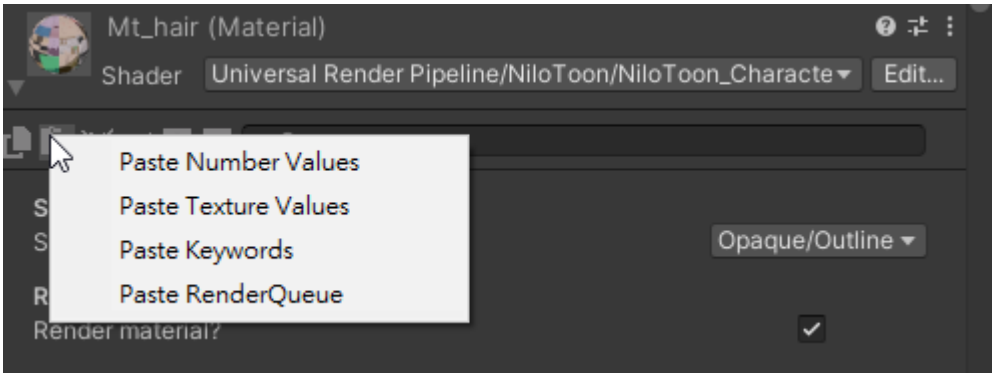
Copy properties from mat A to B, without editing textures

You can use NiloToonCharacter material's toolbar to paste part of the copied values

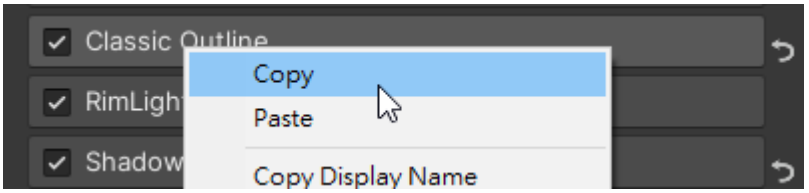




Right-click to paste every option, except texture

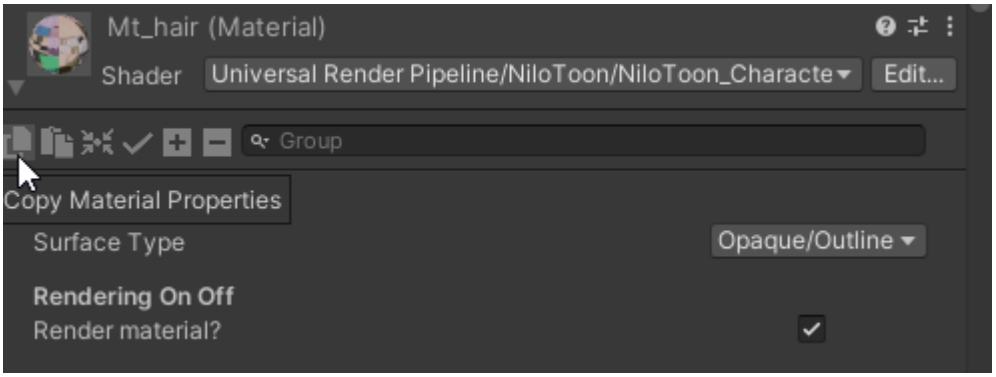


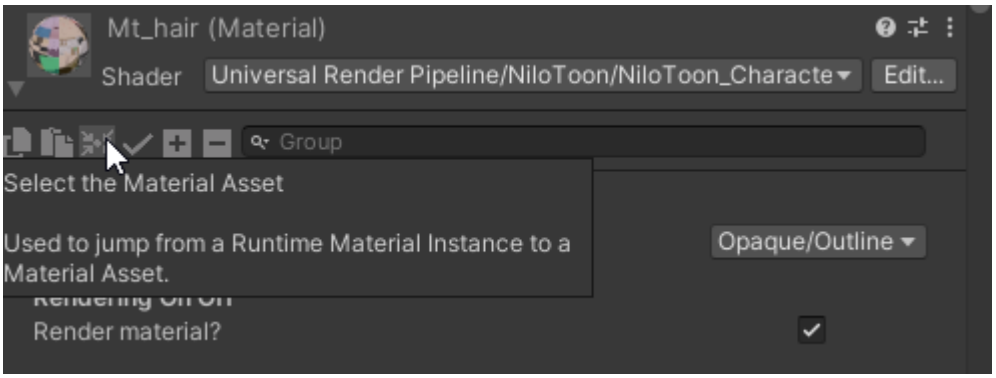
If you want to copy paste a specific group, you can right-click to copy and paste a group, it will be a full copy with texture

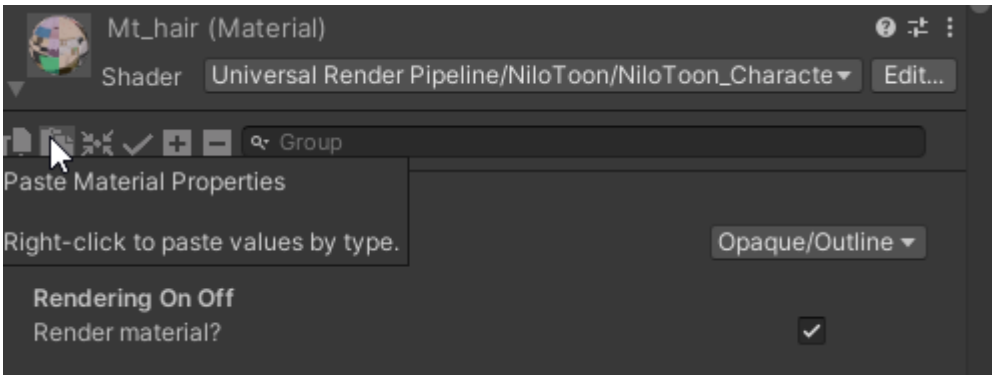


Keep playmode mat edit

Solution A: **copy** and **paste** back to the material asset in project window

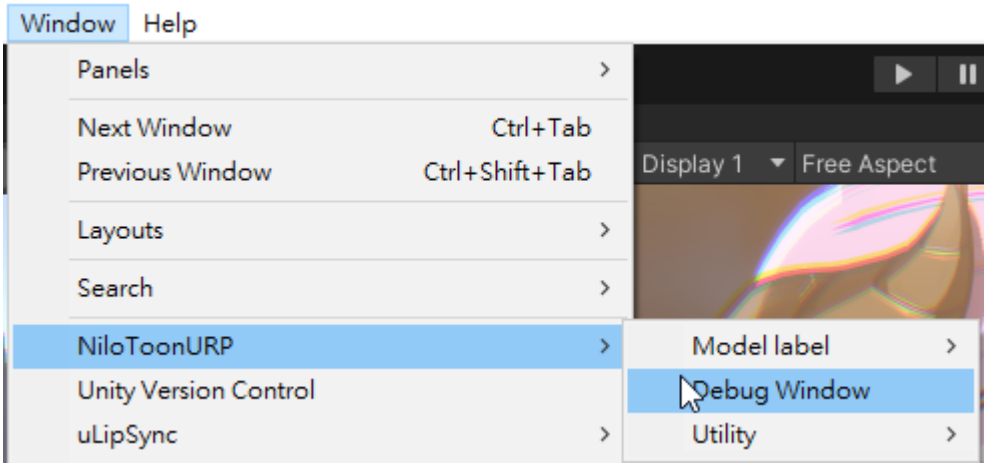


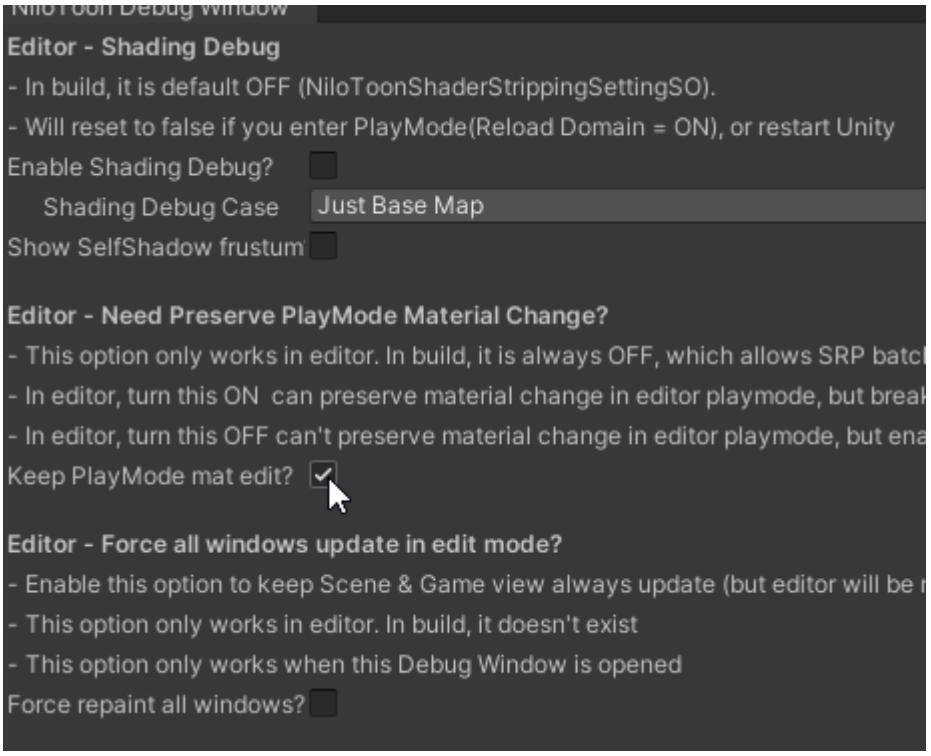




*you may copy unwanted properties that is set only in playmode, so we don't recommend this method, use **SolutionB** is usually better

SolutionB: enable **Keep PlayMode mat edit?** before entering play mode





When it is on,the material in playmode will not become a material instance, so you can edit the material as usual, change will be kept after exiting playmode. When enabled thius mode, SRP batching will not work anymore, so CPU rendering cost is higher

Auto mouth AIUEO anim

When creating MV videos, you can generate character's mouth animation using the audio as input.

1. Use AI to extract the "vocal only" as mp3. You can use tools like the following
- Vocal Remover
- Free online vocal remover tool powered by AI - Voice.ai
- uvr5
2. Use uLipSync to bake "vocal only" audio to ulipsync's mouth data
3. Play the mouth animation in your MV/Cutscene/Dialog, using ulipsync's monobehaviour & timeline track

Path too long(Windows 260 char limit) problem

If you are using Windows and you can't find a solution to solve the "File path too long" problem of your unity project, here are some solutions that you can consider.

Enable Windows long path

Objective:

To extend the maximum path length limit beyond the traditional 260 characters in Windows systems. This allows applications to access, read, and write files and directories with longer path names.

Steps to Enable:

1. Open Registry Editor:
   - Press `**Win + R**`, type `**regedit**`, and press Enter to open the Registry Editor.

2. Modify the Registry:
   - Navigate to the following key:
   ```
   HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\FileSystem
   ```
   - Locate the `**LongPathsEnabled**` value. If it does not exist, create a new DWORD (32-bit) value named `LongPathsEnabled`.
   - Set the value of `LongPathsEnabled` to `1` to enable long path support.

3. Restart the System:
   - Restart the computer to ensure that all system services and applications can recognize and utilize the new configuration.

Impact:

Enabling this setting allows all compatible applications on the system to handle file paths longer than 260 characters. This is particularly useful for large Unity projects involving complex and long directory(folder) structures and file paths.

Note:

- **Administrative privileges** are required to make changes to the Registry.

Enable Git long path

Open GitBash, type
```
git config --global core.longpaths true
```
hit enter, then you should be able to commit in a git repository containing long path files.

*if you are using tools like GitHub Desktop, this method should work also

If none of the solution works

Use the ‘easy’ solution, try to place the Unity project in a shorter path folder, for example:

- C:\MyUnityProjectName

Bug report & support

Contact:

- email - nilotoon@gmail.com
- Discord user ID - kuronekoshaderlab (+friend / invite to group&server)
- Slack - kuronekoshaderlab@gmail.com (invite to workspace)
- or alternatively, provide us with your QQ/WeChat ID
- skype - live:.cid.3b9d54776782296a (not supported anymore)

with any information about you(e.g. **purchaser name/invoice number/email**) that we can recognize you as a customer.

For bug report, include these information if possible:

- Unity version (e.g., Unity 2022.3.15f1)
- NiloToon version (e.g., NiloToon 0.16.11)
- What happens when the bug appears? (e.g., I see a red error message, the error message is _____)
- How to reproduce the problem? (e.g., when I enable _____ toggle, and build the game, the error message will appear)
- (optional) A minimum reproducible project zip **(bug report with reproducible files will be prioritized first)**

*We can communicate in text, but not audio:

- english
- 中文
- others (AI translated, e.g. 日本語, 한국어)

Change log

A copy of CHANGELOG.md in NiloToonURP folder

Changelog
All notable changes to NiloToonURP & demo project will be documented in this file.

The format is based on [Keep a Changelog](http://keepachangelog.com/en/1.0.0/)
and this project adheres to [Semantic Versioning](http://semver.org/spec/v2.0.0.html).

- (Core) or no tag means the change is part of NiloToonURP_[version].unitypackage (files inside NiloToonURP folder)
- (Demo) means the change is only changing the demo project, without changing NiloToonURP.unitypackage
- (Doc) means the change is only affecting any .pdf documents
- (InternalCore) means Core, but if you are only using NiloToonURP as a tool, and don't need to read/edit NiloToonURP's source code/files, you can ignore this change.

[0.17.2] - 2025-08-05

Added
- support forward depth priming via feature fallback (In old version, depth priming will remove the face rendering. In 0.17.2, NiloToon will render correctly by disabling face's 2D shadow if depth priming is detected)

Changed
- upgrade LWGUI from 1.25.0 to 1.27.0

Fixed
- Convert NiloAnimePostProcess from unsafe pass to raster render pass, so it works for render graph now
---
[0.17.1] - 2025-06-27

Added
- NiloToon Character shader: add 'Dissolve' group (controlled per Material, without rely on any NiloToonPerCharacterRenderController)

Fixed
- fixed RenderGraph record error when "NiloToonTonemapping is enabled, and NiloToonBloom is not enabled"
- fixed a bug when multiple directional light with NiloToonLightSourceModifier exist in scene (and those lights having the same intensity)
- support PIDI6 planar reflection

[0.17.0] - 2025-06-13

Added
- (Big change)All NiloToon passes fully support Render Graph!!!!!!!!!!! Users can enable RenderGraph now and should not see any visual difference.
- NiloToon Character shader: support Unity6.1 Forward+ / Deferred+ correctly (CLUSTER_LIGHT_LOOP)
- NiloToon Character shader: support Unity6.2 rendering layer
- NiloToonMaterialConvertor.cs: make method SetMaterialNiloToonSurfaceTypeAndProperties(...) public and static

Changed
- LWGUI upgrade from 1.21.2 -> 1.25.0

Fixed
- NiloToon Character shader: fix shader 6.1 unity version typo (#if UNITY_VERSION >= 60000100, corrected to #if UNITY_VERSION >= 60010000)
- NiloToon Environment shader: fix shader 6.1 unity version typo (#if UNITY_VERSION >= 60000100, corrected to #if UNITY_VERSION >= 60010000)
- NiloToonVolumePresetPicker: fix null error when build
- NiloToonCharacter_MatCapAdditivePreset_LWGUI_ShaderPropertyPreset: fix UV tiling (1->0.99)

[0.16.37] - 2025-03-08

Fixed
- NiloToonPerCharacterRenderController: fix a critical bug that adding new material to a renderer will produce a wrong lit material due to wrong Destory() code by NiloToonPerCharacterRenderController
- NiloToonPerCharacterRenderController: add Clean up all material instances when this script is destroyed
---
[0.16.36] - 2025-02-25

Fixed
- NiloToonCharRenderingControlVolume: fix 'Override Main Light Direction > LRAngle' direction flip bug
---
[0.16.35] - 2025-02-24

Breaking change
- NiloToon Character shader stripping: now NiloToon will by default strip all XR keywords since most user are not using NiloToon for XR (STEREO_INSTANCING_ON, STEREO_MULTIVIEW_ON, STEREO_CUBEMAP_RENDER_ON, UNITY_SINGLE_PASS_STEREO). User can re-enable them in NiloToonShaderStrippingSettingSO asset

Added
- NiloToon Character shader: add a new 'Decal' group, you can use it to control how URP Decal Projector affects the character
- NiloToon Character shader: +_MainLightSkinDiffuseNormalMapStrength, _MainLightNonSkinDiffuseNormalMapStrength. Usually for removing normalmap for skin diffuse, while keeping normalmap for skin specular
- NiloToon Character shader: +_FixFaceNormalAmountPerMaterial. Usually for eyeglasses enabling IsFace + Fix Face Normal = 0. This will cancel the 2D shadow on face due to eyeglasses

Changed
- NiloToon Character shader stripping: all fog variants will force using dynamic_branch for Unity6.1 or higher. This will reduce shader compile time and shader runtime memory usage to 50% or even 25% (= 2x~4x improvement) depending on how many type of fog was used (FOG_EXP,FOG_EXP2,FOG_LINEAR)

Fixed
- NiloToon Character shader: fix a bug that decal normal can't affect NiloToon_Character shader
- NiloToonBloom/NiloToonTonemapping: fix a NiloToonPrepassBuffer(character area) ZFighting problem when camera near plan is very small (<= 0.3)
- NiloToon Character shader: kajiya hair specular solved a NaN pixel bug. (convert half to float, use SafeSqrt & SafePositivePow_float)

[0.16.34] - 2025-02-16

Changed
- Nilo028-Concert005_StyleDark_ACES(NiloToonVolumePreset).asset: match settings closer to nilotoon concert demo project's volume

Fixed
- NiloToon character shader: fix a critical bug that normalWS become 0 when APV is used
- NiloToonSetToonParamPass.cs: fix a critical bug that rendering is wrong when mainlight is not active or intensity is 0

[0.16.33] - 2025-02-15

Changed
- NiloToon Volumes: improve display name and helpbox

Fixed
- NiloToon environment shader: fix a 'motion vector pass bug' that disabled shader's SRP batching
---
[0.16.32] - 2025-02-13

Breaking change
- (Big change)NiloToonPerCharacterRenderController: Remove "Allowed Passes" foldout completely, due to this foldout prevents a lot of material optimization. Now users should use material's "Allowed Passes" group, which works in edit mode, much more optimized, and allows controlling pass on/off per material instead of per character.
- (Big change)NiloToon Character shader: completely rewrite "Allowed pass" group using LWGUI's PassSwitch feature
- NiloToon Character shader: apply Stencil to DepthOnly, DepthNormalsOnly, MotionVectors, NiloToonPrepassBuffer passes
- NiloToon Character shader: lower _DepthTexRimLightAndShadowWidthMultiplier default 0.6->0.5

Added
- Added support for Deferred+ in Unity6.1

- Added RenderGraph support for NiloToonAnimePostProcessPass, NiloToonScreenSpaceOutlinePass, NiloToonExtraThickToonOutlinePass
- NiloToon Character shader: added _BlendOp, _SrcBlendAlpha, _DstBlendAlpha, _DepthTexRimLightAndShadowSafeViewDistance, _HairStrandSpecularUVIndex, _HairStrandSpecularUVDirection

Changed
- NiloToon Character shader: big refactor, sync all passes shader code to match Unity6.1 ComplexLit.shader structure
- NiloToon Environment shader: big refactor, sync all passes shader code to match Unity6.1 ComplexLit.shader structure
- LWGUI: update to latest 1.21.2
- NiloToonPerCharacterRenderController: add better foldout "VRM" + "Compatibility mode". Remove the old "Misc" foldout
- (InternalCore) NiloToonToonOutlinePass.cs: RenderGraph refactor

Fixed
- NiloToonAverageShadowTestRTPass.cs: fix a wrong blit uv and the missing on/off ability in RenderGraph mode
---
[0.16.31] - 2025-01-01

Fixed
- NiloToonMotionBlur: fix a hlsl bug that can't build using Unity6

[0.16.30] - 2024-12-31

Added
- Added Nilo030 & Nilo031 volume preset for Unity6 concert

Changed
- AutoEyeBlind: update the UI to allow selecting blendshape via a dropdown
- NiloToonCinematicRimLightVolume: Rim (3D BackLight only Style) remove (Experimental) tag
- NiloToonTonemappingVolume: Nilo NPR Neutral V1 & Nilo Hybird ACESremove (Experimental) tag

Fixed
- NiloToonSetToonParamPass: fix a critical bug "enabling/disabling additional lights may cause the rendering stop for 1 frame due to multiple hash"

[0.16.29] - 2024-12-17

Added
- Added AutoEyeBlink.cs, for doing simple eye blink blendshape anim
- NiloToonMotionBlurVideoBaker: add ProRes 422, now it is default due to the balance of it's quality and speed
- NiloToonMotionBlurVideoBaker: add a lot of fps options, now 60 is default
- NiloToonMotionBlurVideoBaker: add a 'Open Destination Folder' button

Changed
- Material Convertor: add 'me','ha' as exact target names for face & no outline
- Material Convertor: mouth target names add 'kounai', 'shita'
- UnlockFPS.cs: expose target fps = 60 as default

Fixed
- MToon(VRM0.x) -> NiloToon material convert: change the outline multiplier from 4->12 (4 is wrong for a long time). Now the convertion result's outline width will match MToon's original result much closer
- NiloToon MotionBlur: now motion blur will not apply to planar reflection cameras
- NiloToon MotionBlur: fix a bug that NiloToon still request MotionVectors when motion blur is not active
- NiloToon MotionBlur: fix a bug that NiloToon motion can't compile on build using Unity2021.3
- NiloToonLightSourceModifier: highly optimize the CPU cost when using lots of NiloToonLightSourceModifier with lots of lights in Forward+
- NiloToonCharacter shader: fix a UI bug about _ApplyAlphaOverrideOnlyWhenFaceForwardIsPointingToCameraRemapMinMaxSlider
- All NiloToon render passes: hide the RenderGraph not implented warning, because the warning log itself cause bad CPU performance problem.

[0.16.28] - 2024-12-04

Changed
- NiloToonMotionBlurVolume: Improve the quality and performance by rewriting it based on KinoMotion
---
[0.16.27] - 2024-12-03

Added
- Added NiloToonMotionBlurVolume in volume, it is an alternative to URP's motion blur

Changed
- MotionBlurVideoBaker: improve UI, adding document links
- LWGUI: upgrade to 1.20.2

Fixed
- NiloToon Character shader: fix a motion vector bug when _ALPHATEST_ON is used
- fix a bug that NiloToon can't import a mesh when MeshRenderer exists without MeshFilter
---
[0.16.26] - 2024-11-17

Developer Note
- This is the first version that makes NiloToon support RenderGraph in a very basic way. Note that NiloToon's self shadowmap pass and all NiloToon postprocess will be disabled when enabling RenderGraph, we are working on it, it will take some time. (Unity2021.3 is still supported, we are trying to delay the remove of Unity2021.3 support as much as possible, due to Warudo using Unity2021.3)

Breaking Change
- NiloToon Character shader: default _OutlineWidth changed from 0.6->0.5, since it is a more reasonable value for VTuber models

Added
- NiloToon Character shader: added 'XRMotionVectors' pass, for experimental support of XR Application SpaceWarp for Unity6
- NiloEditorMotionBlurVideoBaker: add ProRes,h265 option to a new codec option, default to ProRes now to improve speed and quality (but bigger file size)
- NiloToonAverageShadowTestRTPass: add experimental RenderGraph support
- NiloToonCharSelfShadowMapRTPass: add WIP RenderGraph support, now self shadow will always be disabled when RenderGraph is enabled, we will unlock this again after we finish the implementation in the future

Changed
- NiloToonToonOutlinePass.cs: improved RenderGraph support
- NiloToon Character shader: better 'MotionVectors' pass, it will now consider zoffset, perspective removal, dither, dissolve, enable renderings etc
- NiloEditorMotionBlurVideoBaker: improve log info, displaying important info like complete% and total duration, also requiring less memory now to reduce the chance of out of memory crash

Fixed
- NiloToon Bloom: Add a very small bias to NiloToon's prepass, to avoid prepass problems similar to shadow acne
---
[0.16.25] - 2024-11-13

Added
- NiloToon Character shader: +_ApplyAlphaOverrideOnlyWhenFaceForwardIsPointingToCameraRemapStart,_ApplyAlphaOverrideOnlyWhenFaceForwardIsPointingToCameraRemapEnd. For adding control to front hair semi-transparent fadeout
- NiloToonSetToonParamPass.cs: add experimental RenderGraph support
- NiloToonToonOutlinePass.cs: add experimental RenderGraph support

Changed
- NiloToon Character shader: sync motion vector pass to Unity6LTS
- LWGUI: upgrade to latest version

Fixed
- NiloEditorMotionBlurVideoBaker: support the latest ffmpeg, improve input video fps detection speed, improve UI info
---
[0.16.24] - 2024-11-02

Added
- NiloToon Character shader: Add Adaptive Probe Volumes(APV) support for Unity6LTS

[0.16.23] - 2024-10-17

Added

- Add menu item for "Assets/NiloToon/xxx", for easier menuitem usage by right clicking an asset within the project window, so user don't need to always click the "Window/NiloToon/xxx" menu item

Changed
- NiloToonMaterialConvertor.cs: lilToon convertor now consider ZWrite option to select the correct SurfaceType

Fixed
- NiloToonUberPostProcessPass.cs: remove the use of ShaderKeywordStrings.UseRGBM, it is a required fix to support Unity6 (2024/10/17 release version)
- NiloToonSetToonParamPass.cs: fix a bug when light number > max supported number of light
---
[0.16.22] - 2024-10-01

Added
- Improve and released "Window/NiloToon/MotionBlur Video Baker", now out of experimental
- Added NiloToonEditor_AutoGeneratePrefabVariantAndAssets.cs, now user can use "Window/NiloToon/Create Nilo Prefab Variant and Materials" to setup NiloToon without modifying any original material and prefab
- NiloToonVolumePresetPicker: added Nilo027~Nilo029 volume presets
- NiloToonTonemapping volume: +Nilo Hybird ACES (experimental)

Fixed
- NiloToonMaterialConvertor.cs: fixed name space problem

[0.16.21] - 2024-09-03

Added
- NiloToon_Character shader: To allow using reflection probe as 3D rim light, added -> _EnvironmentReflectionShouldApplyToFaceArea,_EnvironmentReflectionApplyReplaceBlending,_EnvironmentReflectionApplyAddBlending,_EnvironmentReflectionFresnelPower,_EnvironmentReflectionFresnelRemapStart,_EnvironmentReflectionMFresnelRemapEnd
- NiloToonUberPostProcessPass.cs: to allow better renderer feature sorting, added -> renderPassEventTimingOffset, default 0

Fixed
- NiloToon_Character shader: fix SRP batching not correct in Unity6 due to motion vector pass
- NiloToonCharacterSticker_Additive.shader: fix Float data mismatch bug, now corrected to Integer
- NiloToonCharacterSticker_Multiply.shader: fix Float data mismatch bug, now corrected to Integer
- GenericStencilUnlit.shader: fix Float data mismatch bug, now corrected to Integer
---
[0.16.20] - 2024-08-19

Added
- NiloToonTonemapping: + Khronos PBR Neutral, + Nilo True Color V1 (experimental)
- NiloToon_Character shader: add new group -> Face 3D rim light and shadow (experimental)
- Add Window/NiloToonURP/Experimental/MotionBlur Video Baker (experimental)

Fixed
- NiloToonPerCharacterRenderController: fix SetInt() data mismatch bug, now corrected to SetInteger() API
- NiloToon Debug Window: change normalTS debug to match URP's rendering debugger color format
---
[0.16.19] - 2024-08-07

Added
- NiloToon Character shader: +_AsUnlit, for multiply blending material like blush

Fixed
- NiloToon Character shader: change array index from Float to integer, to avoid graphics bug on Adreno740 GPU (https://issuetracker.unity3d.com/issues/android-shader-has-incorrect-uvs-when-built-on-android-devices-with-an-adreno-740-gpu)

[0.16.18] - 2024-07-30

Added
- NiloToon Character shader: +_PreMultiplyAlphaIntoRGBOutput, _ColorRenderStatesGroupPreset

Changed
- NiloToon Character shader: move "Color Buffer" group to Normal UI Display group (now show in default material setting)

[0.16.17] - 2024-07-28

Breaking Change
- NiloToon Character shader: reduce max char(maximum number of NiloToonPerCharacterRenderController) to improve mobile GPU performance, max char for opengles = 32->16, max char for vulkan/switch = 64->32, max char for PC = still 128(unchanged)
- NiloToon Character shader: apply basemapstacking layer to all passes, because basemapstacking layer will now affect alpha value of all passes

Changed
- NiloToon Character shader: _DepthTexRimLight3DRimMaskThreshold change default value from 0.075 -> 0.5, remove experimental tag

Added
- NiloToon Character shader: +_DepthTexRimLight3DFallbackMidPoint,_DepthTexRimLight3DFallbackSoftness, _DepthTexRimLight3DFallbackRemoveFlatPolygonRimLight, for letting user control 3D rim light fallback's visual, can be used as soft fresnel rim light
- NiloToon Character shader: + _AlphaOverrideTexUVIndex

Fixed
- NiloToon Character shader: average shadow now supports URP's screen space shadow renderer feature
- NiloToon Character shader: optimize GetUV() method GPU performance by "replacing dynamic array UV index access by if chains with const index"
- NiloToonPerCharacterRenderController: avoid calling Texture2D.whiteTexture, which will avoid triggering NDMF(Non-Destructive Modular Framework)'s crash bug
---
[0.16.16] - 2024-07-25

Fixed
- fix environment shader can't compile in Unity6(6000.0.11f1), by calling the correct OUTPUT_SH4 function overload
- fix average shadow not working correctly for multiple characters in Unity2022.3 or later
- (InternalCore) change GraphicsSettings.renderPipelineAsset -> GraphicsSettings.defaultRenderPipeline
---
[0.16.15] - 2024-07-05

Fixed
- Fix NiloToon renderer feature's runtime error in Unity2022.3, this error will trigger when PIDI 5 planar reflection is used together with NiloToon in Unity2022.3

[0.16.14] - 2024-07-03

Fixed
- error handle when bake smooth normal's mesh doesn't have tangent
- fix a basemap stacking layer NormalRGBA mode bug when the material is opaque cutout
---
[0.16.13] - 2024-06-21

Added
- NiloToon Character shader: +zoffset presets
- NiloToon Character shader: apply LWGUI ramp map to spceular ramp
- NiloToonVolumePicker: +ID 026 night soft bloom

Changed
- Auto setup: eye keyword + "sclera", "cornea", "limbus"
- Auto setup: ban "headset","phone" for face keyword
- NiloToon Character shader: specular ramp now affected by ramp alpha

Fixed
- fix reimport deadlloop when mesh has 0 vertex
- NiloToon Character shader: fix a bug where toon specular can't use _SpecularReactToLightDirMode
---
[0.16.12] - 2024-05-28

Breaking Changes
- NiloToonLightSourceModifier: remove penetration, add "back light occlusion(2D) & (3D)"

- NiloToonLightSourceModifier: when light list is empty, it will not be considered as a global modifier anymore
- NiloToonLightSourceModifier: now always auto re-fill empty light list, instead of auto fill in OnValidate() only
- Auto material convertor: don't force reset render queue anymore, unless the original material's render queue is out of expected render queue range. This change can help preserve the render queue from the original material (more correct draw order/ stencil order), while still not breaking NiloToon's material render queue requirements

Added
- NiloToon Character shader: +_MultiplyBRPColor, expose _Color
- NiloToon Character shader: Add "invert?" to many mask map -> _MatCapOcclusionMaskMapInvert, _MatCapAdditiveMaskMapInvertColor, _DepthTexRimLightMaskTexInvertColor, _OverrideShadowColorMaskMapInvertColor, _ZOffsetMaskMapInvertColor, _MatCapAlphaBlendMaskMapInvertColor, _DetailMaskInvertColor, _FaceShadowGradientMaskMapInvertColor, _EmissionMaskMapInvertColor, _EnvironmentReflectionMaskMapInvertColor
- NiloToon Character shader: +_DepthTexRimLight3DRimMaskEnable, _DepthTexRimLight3DRimMaskThreshold (experimental) for try fixing 2D rimlight problem on finger or small objects like ear ring
- NiloToon Character shader: +_SpecularReactToLightDirMode dropdown menu(Use settings from volume, Readct to light, Follow camera) for controlling how specular light react to light direction in material
- NiloToon Character shader: +RimHard(LightFromTop) matcap add preset
- NiloToonPerCharacterRenderController: +renderCharacter toggle, allow you to on/off the rendering of a character with just 1 click or 1 API call
- NiloToonPerCharacterRenderController: +MaterialInstanceMode toggle, for prevent generating material instance when required
- Auto material convert: Rewrite univrm0.x auto material convert, porting more properties from "VRM/MToon", now _Color in VRM blend shape proxy clips can be reused without editing any clips
- Auto setup character can now select multiple characters and process them together. For example, you can select multiple vrm character prefabs and process them together
- Added a new menu item 'Window > NiloToonURP > Convert Selected Materials to NiloToon_Character', so you can convert material to NiloToon without selecting any gameobject/prefab
- Auto material convert: + lilToon emission's alpha porting
- Auto material convert: +Stencil preset matching
- +Matcap_Upward_HardEdge.png
-(InternalCore) + NiloAAUtil.hlsl
-(InternalCore) + NiloToonMaterialConvertor.cs, extracted all NiloToonEditorPerCharacterRenderControllerCustomEditor's material auto convert logic into this script

Changed
- NiloToon Character shader: convert "Use GGX Direct Specular?" toggle to a "Specular Method" dropdown menu with 2 modes(Toon, PBR GGX)
- NiloToon Character shader: move 'UV Edit' and 'Lighting style' group
- Auto material convert: + 'karada' as skin material keyword, +'head' as face keyword, +'canthus' as eye keyword
- Auto material convert: auto disable '2D rimlight and shadow' if material renderqueue > 2500 (doesn't draw depth texture)
- Update LWGUI to the latest 2.x version (2024-05-29)
- Improve menuitem ordering in 'Window > NiloToon > ...'
- (Doc) Improve doc about improving rendering results in section 'AA(outline,rim light,shadow map)','DLSS | FSR3 (better fps & AA)','Cam output char’s alpha'

Fixed
- NiloToonVolumePresetPickerEditor: fix a critical bug that NiloToonVolumePresetPickerEditor's ID didn't save correctly when saving the scene, where the ID will become 0 when the scene reload
- NiloToonCharacter_LightingEquation: fix a critical rim light result breaking change bug due to wrong code implementation, where rim light artifact will appear in version 0.16.0~0.16.11
- NiloBlendEquationUtil.hlsl: fix a basemap stacking layer NormalRGBA mode rgb blending bug when alpha is 0 and the material is opaque
- NiloToonEditor_AssetLabelAssetPostProcessor: fix Model import bake UV8's 'HashMap is full' error log
- NiloToonPerCharacterRenderController: fix a dissolve bug where pixels are not dissolve correctly when the current dissolve texture pixel is pure white/black
---
[0.16.11] - 2024-04-11

Added
- NiloToonTonemappingVolume: add a new tonemap mode -> ACES (Custom SR)
- Added 5 Skin template materials
- NiloToonLightSourceModifier: add main light "Penetration" param, set it to 0 for blocking backlight(e.g., spot light)

[0.16.10] - 2024-03-29

Changed

- frame debugger and profiler: better display group & names for all NiloToon pass
- auto setup now treats material with name "matuge" as eye material now

Fixed
- NiloToonPerCharacterRenderController: fix a critical bug that makes FaceForward & FaceUp can't be edited (incorrectly locked to "auto value" every frame), if you saw that face shadow / face direction is incorrect in previous NiloToon version when character is playing animation(e.g., timeline), try to upgrade to this version.

[0.16.9] - 2024-03-23

Breaking Changes
- NiloToon Character shader: clamp ggx specular roughness's minimum to 0.04 instead of 0, to prevent bad bloom result when smoothness is 1(roughness is 0)

[0.16.8] - 2024-03-23

Breaking Changes
- NiloToon Character shader: _SelfShadowAreaSaturationBoost's default value changed from 0.5 -> 0.2, to reduce the default hue artifact due to texture compression

Added
- NiloToon Character shader: + WIP motion vector pass for URP16(Unity2023.2) or later
- NiloToonEditorPerCharacterRenderControllerCustomEditor: auto setup character now converts lilToon's _ShadowBorderMask -> NiloToon's _OcclusionMap
- NiloToonEditorPerCharacterRenderControllerCustomEditor: auto setup character now converts lilToon's parallax -> NiloToon
- NiloToonAnimePostProcessPass: add new toggle -> affectedByCameraPostprocessToggle
- Auto install renderer feature: now consider Graphics tab's URP asset also
- Add new Window>NiloToon>... [MenuItem] buttons: Update to latest version, Open document, Change log, contact support

Changed
- LWGUI: updated to latest 2.x
- Auto install renderer feature: if the setup is correct already and nothing to do, now also show a pop up window

Fixed
- NiloToon environment shader: fix a Unity6 compile error about SETUP_DEBUG_TEXTURE_DATA(...)
- NiloToonEditorPerCharacterRenderControllerCustomEditor: AutoSetupCharacterGameObject(...) prevent user calling from fbx generated prefab
- NiloToonPerCharacterRenderController: fix a null reference exception bug on OnDisable()
- NiloToonSetToonParamPass: fix a possible GPU global array wrong length when switching platform
- NiloToonEditorPerCharacterRenderControllerCustomEditor: fix lilToon->NiloToon "_BackFaceBaseMapReplaceColor" wrong port
- NiloToonEditorPerCharacterRenderControllerCustomEditor: fix lilToon->NiloToon "_Main2ndTexAngle" wrong port
- NiloToon Character shader: add [HideInInspector] _MainTex & _Color, to prevent vrm0.x output error
- NiloBlendEquationUtil.hlsl: fix NormalRGBA's 0 alpha bug on mobile
---
[0.16.7] - 2024-03-12

Added
- NiloToonSetToonParamPass: add new API -> public static bool ForceMinimumShader, to enable ForceMinimumShader using code
- NiloToonCharacter shader: add _SkinShadowBrightness, _FaceShadowBrightness

Changed
- NiloToonCharacter shader: _MultiplyBaseColorToSpecularColor's default value changed from 0 -> 0.5
- NiloToonCharacter shader: moved Parallax Map from "Show All" group to "Expert" group
- NiloToonPerCharacterRenderController: remove the (Experimental) tag of "Generate Smoothed normal?"
- NiloToonShaderStrippingSettingSO: better tooltip

Fixed
- Auto setup character: fix material wrongly set to renderQueue -1, now it is correctly set to 2000
- NiloDefineURPGlobalTextures.hlsl: fix _CameraDepthTexture_TexelSize macro condition bug, which makes it fail to compile in Unity2022.3.21f1
- NiloToonPerCharacterRenderController: fix a possible OnDisable() null reference exception when building the player
---
[0.16.6] - 2024-03-05

Changed
- Volume profile presets: specular react to light dir revert to default value(not override)
- Volume profile presets: outline width revert to default value(not override)

Fixed
- NiloToonCharSelfShadowMapRTPass: fix XR multi pass mode's rendering bug due to wrong view projection matrix
- NiloToon Character shader: fix debug shading can't compile when _PARALLAXMAP enabled
---
[0.16.5] - 2024-02-26

Fixed
- LWGUI: upgraded to the latest 2.x version, fixing ramp tex format ARGB32 error and helpbox width problem
- NiloToon Character shader: Fix Opaque SurfaceType not setting the correct render queue (-1) due to LWGUI, now Opaque SurfaceType preset set render queue to 2000 instead to prevent this problem
---
[0.16.4] - 2024-02-25

Breaking Change
- (InternalCore) VideoPlayerForceSyncWithRecorder: move from NO namespace to NiloToon.NiloToonURP.MiscUtil
- (InternalCore) DoFAutoFocusTransform.cs: move from namespace NiloToon.NiloToonURP to NiloToon.NiloToonURP.MiscUtil
- (InternalCore) rename global shader float _NiloToonGlobalSelfShadowDepthBias to _NiloToonGlobalSelfShadowCasterDepthBias, to make support of 0.13.8 shader possible
- (InternalCore) rename global shader float _NiloToonGlobalSelfShadowNormalBias to _NiloToonGlobalSelfShadowCasterNormalBias, to make support of 0.13.8 shader possible

Added
- NiloToonPerCharacterRenderController.cs: +compatibleWithNiloToon13Shader("Compatible with 0.13.8 shader") toggle in Misc group, for a special situation where NiloToon 0.16.x C# + 0.13.8 shader need to work together (e.g., to support old Warudo files/Asset bundles built with NiloToon 0.13.8 in the past in a NiloToon 0.16.x C# build)
- +SelfRotate.cs, for model 360 debug use
- +UnlockFPS.cs, for android build fps checking use

[0.16.3] - 2024-02-22

Added
- NiloToon Character shader: _DepthTexShadowFixedDirectionForFace, for a stable hair shadow on face

Changed
- Auto Setup Character now search 'bound center' bone in the following order: spine>hip>pelvis instead of hip>pelvis>spine
---
[0.16.2] - 2024-02-21

Fixed
- fix LWGUI null material problem (material inspector can't show, with error log)
- (InternalCore)fix shader compile warning about pow(f/e) potential negative f value
---
[0.16.1] - 2024-02-20

Breaking Change
- low-end OpenGL mobile limited maximum characters from 128 -> 32, due to shader compile limitation & performance
- mobile/OpenGL limited maximum characters from 128 -> 64, due to shader compile limitation & performance

Added
- NiloToon Character shader: +BaseMapStackingLayer 1-10 Mask's UVIndex, Remap, Extract from ID
- NiloToon Character shader: +MatCapp Add & MatCap Replace's new presets
- +Matcap_BlackSmoothSpecular01.png

Changed
- LWGUI upgraded from 1.16 to 2.0, you can use "Show Only Modified Properties by Group" option now, it is an important option for adjusting material easily and safely

Fixed
- fix all BaseMapStackingLayer's mask UV bug
- fix shader can't compile in OpenGL mode on a Window PC's editor due to "error C6020: Constant register limit exceeded at _____; more than 1024 registers needed to compile program", see [Breaking Change]

[0.16.0] - 2024-02-19

Version Highlights
In short, compared to NiloToon 0.15.15, NiloToon 0.16.0 introduces these significant changes:
- [breaking change] Rewrote the code for additional lights; all additional lights are now treated as the main light by default, allowing you to light a character nicely with just point/spot lights (great for interiors!)
- Added a new 'NiloToonCharacterLightController' script, which allows you to control how each character receives lighting(e.g, tint or add color to main light for a few target characters)
- Added a new 'NiloToonAdditionalLightStyleVolume' volume, which enables you to control how all lights affects all characters globally
- Added a new 'NiloToonLightSourceModifier' script, which enables you to control how each light affects all characters. For example, you can create a few point lights with different roles -> (1)a normal point light (2)a point light without rim light (3)a point light with only rim light (4)a point light that doesn't affect nilotoon characters...For (4), it is especially handy if you'd rather not use Unity URP's rendering masks, since URP's rendering layers mandate individual settings for each renderer's layer, this can be time-consuming and clutters your scene's game objects and prefab files.
- Many new options in the menu "GameObject > Create Other > NiloToon"
- Many new options in the menu "GameObject > Light > NiloToon"
- Introduced 'soft shadow' for shadow maps to reduce aliasing, enabled by default (medium quality)
- Updated the NiloToon Character shader: Basemap Stacking Layer 1-10 now includes +7 UVs (UV1-4, MatcapUV, CharBoundUV, ScreenSpaceUV) and 4 blend mode options (normal, add, screen, multiply). For example, you can create a basemap stacking layer using matcapUV with multiply or add blend methods.
- Updated NiloToonPerCharacterRenderController's' 'Basemap Override' group: +7 UVs (UV1-4, MatcapUV, CharBoundUV, ScreenSpaceUV), and 4 blend mode options (normal, add, screen, multiply)
- Added a new 'Color Fill' group to NiloToonPerCharacterRenderController, a utility function to redraw the character with different effects (e.g., x-ray vision, show characters that are behind a wall similar to CS2/Valorant)
- Added a new 'Generate Smoothed normal?' toogle to NiloToonPerCharacterRenderController, you can enable it for non-fbx prefabs(e.g., vrm character), so those non-fbx character prefabs can still have a similar outline quality as fbx characters
- Fixed all memory leaks and GC allocation issues that were found
- Significant CPU optimization when multiple characters are visible (approximately 1.72x CPU speed up for a test scene with 30 unique characters and 800 unique characters materials)
- HLSL code is now much easier to read in Rider

Note from the Developer (Unity6/RenderGraph/Warudo)
- NiloToon 0.16.x is confirmed to be the last version that still supports Unity2021.3.
- In NiloToon 0.17.0, we will completely remove support for Unity2021.3 to begin working on Unity6 (RenderGraph) support.
- After a few more 0.16.x releases, we will provide a stable NiloToon 0.16.x version to Warudo to be used as the official NiloToon version for Warudo(Unity2021.3).

Note from the Developer (Additional Light Rewrite)
Skip this message if you do not need to light characters using point/spot lights.
NiloToon 0.16.0 includes a major rewrite of the code for additional lights!

In the previous version, NiloToon 0.15.15, the treatment of additional lights was similar to Unity's method, where the lights' colors were additively blended onto the character. As a Unity URP shader, our original design aimed to align with Unity's URP additional light behavior as closely as possible, although we know that it is not the best for toon shader.

However, the additive light method only looked good under very carefully set up conditions, like this [video](https://youtu.be/ST9qNEfPrmY?si=98mqByySiRE7kcTO). In most regular use cases, NiloToon 0.15.15's point/spot lights could easily make the character appear overly bright or look strange.

Therefore, in NiloToon 0.16.0, we decided it was time to make a significant breaking change to ensure additional lights provide good results by default. Now, any additional lights are treated the same as the main light, so no matter how you light the character and regardless of the light type used, the lighting result will always maintain the same quality as the main directional light. This means scenes that mainly use point/spot lights, such as interiors, rooms, caves, etc., will now be much easier to light characters nicely and consistently with NiloToon 0.16.0.

*If you want to produce the old additional light result, you can:
- In Nilotoon Additional Light Style Volume: set color & direction = 0
- (optional)In Nilotoon Cinematic Rim Light Volume: disable “auto fix unsafe style”

Breaking Change
- NiloToon Character shader: "face normal fix" now only affects materials that are controlled by NiloToonPerCharacterRenderController
- NiloToon Character shader: fix a wrong MatCapUV & CharBoundUV code that wrongly flip these UV's x, now after the fix, MatCapUV & CharBoundUV will look the same as the Texture2D on a 3D sphere mesh correctly, not having the uv x flip

anymore. However, it is a bug fix that lead to a breaking change, so if you want to keep the old MatCapUV result where uv x is flipped, go to material's "UV Edit" group, set MatCapUV's tiling to (-1,1,_,_).
- NiloToon Character shader: _BackFaceTintColor will not affect outline pass anymore, since it will make the outline darken which is not correct
- NiloToon Character shader: normalize depth rim uv offset direction vector, to make rim light width more consistent under all light condition and direction
- NiloToon Character shader: disable NiloToonDepthOnlyOrDepthNormalPass's face normal edit, since we don't want to edit the face normal of URP's _CameraNormalTexture
- NiloToonPerCharacterRenderController: BaseMap Override group's UV tiling is now using center(0.5,0.5) as tiling center instead of (0,0)
- NiloToonPerCharacterRenderController: BaseMap Override's Map, when it is null, NiloToon will now use a default white opaque tex instead of gray & semi-transparent(50%) texture
- NiloToonPerCharacterRenderController: no longer force update/reimport all active characters in scene, since it is too slow when a scene contains too many characters
- NiloToonPerCharacterRenderController: make StencilFill, ExtraThickOutline, ColorFill pass only draw opaque queue (<2500) materials
- NiloToonCinematicRimLightVolume: +"Auto fix unsafe Style?" toggle, default enabled, which will force at least 1 cinematic rim light style on
- NiloToonCharRenderingControlVolume: specularReactToLightDirectionChange now default = true. If you want to reproduce the old result, you can override it and set it back to false
- All NiloToonVolumePreset assets: reset charOutlineWidthMultiplier to 1

Breaking Change(hlsl programming only)
- (InternalCore) NiloToon Character shader: rename ToonLightingData.PerPixelNormalizedPolygonNormalWS to normalWS_NoNormalMap
- (InternalCore) NiloToon Character shader: rename ToonLightingData.PolygonNdotV to NdotV_NoNormalMap
- (InternalCore) NiloToonCharacter_ExtendFunctionsForUserCustomLogic.shader: changed function param -> void ApplyCustomUserLogicToBaseColor(inout half4 baseColor, Varyings input, UVData uvData, float facing)
- (InternalCore) NiloToonCharacter_ExtendFunctionsForUserCustomLogic.shader: changed function param -> void ApplyCustomUserLogicBeforeFog(inout half3 color, ToonSurfaceData surfaceData, ToonLightingData lightingData, Varyings input, UVData uvData)
- (InternalCore) NiloToonCharacter_ExtendFunctionsForUserCustomLogic.shader: changed function param -> void ApplyCustomUserLogicAfterFog(inout half3 color, ToonSurfaceData surfaceData, ToonLightingData lightingData, Varyings input, UVData uvData)
- (InternalCore) NiloToon Character shader: move indirect light to main light injection phase

Added
- + new volume "NiloToonAdditionalLightStyleVolume"
- + new script "NiloToonCharacterLightController", with a create gameobject method in "GameObject > ... > NiloToon" menu
- + new script "NiloToonLightSourceModifier", with a few create gameobject methods in "GameObject > ... > NiloToon" menu
- NiloToonAllInOneRendererFeature: + soft shadow (default enable, medium quality)
- NiloToonAnimePostProcessVolume: + topLightMultiplyLightColor
- NiloToonCharacterMainLightOverrider: + overrideTiming
- NiloToonPerCharacterRenderController: + new "Color Fill" group
- NiloToonPerCharacterRenderController: + auto fill missing properties in LateUpdate
- NiloToonPerCharacterRenderController: +"Generate Smoothed normal?" toggle, great for non-fbx character to also generate a similar quality baked outline at runtime
- NiloToonPerCharacterRenderController: "Auto setup character" now support PhysicalMaterial3DsMax ArnoldStandardSurface's auto setup
- NiloToonPerCharacterRenderController: BaseMapOverride group + UVIndex {UV1-4, MatCapUV, CharBoundUV, ScreenSpaceUV}
- NiloToonPerCharacterRenderController: BaseMapOverride group + BlendMode {Normal,Add,Screen,Multiply}
- NiloToon Character shader: + _DepthTexRimLightIgnoreLightDir(360 rim) and _DepthTexShadowIgnoreLightDir(360 shadow)
- NiloToon Character shader: + _SelfShadowAreaHSVStrength
- NiloToon Character shader: + _SkinShadowTintColor2, _FaceShadowTintColor2
- NiloToon Character shader: "Normal map" group + UVIndex,UVScaleOffset,UVScrollSpeed
- NiloToon Character shader: + new "UV Edit" group, allow you can add tiling,rotation,animation.... to UV1~4 & MatCapUV
- NiloToon Character shader: all "Basemap stacking layers" group + master strength slider
- NiloToon Character shader: all "Basemap stacking layers" group + BlendMode {Normal,Add,Screen,Multiply}
- NiloToon Character shader: all "Basemap stacking layers" group + UVIndex {UV1-4, MatCapUV, CharBoundUV, ScreenSpaceUV}
- NiloToon Character shader: all "Basemap stacking layers" group + UVscale,position,angle,rotate options
- NiloToon Character shader: update basemap stacking layer 4-10, now they have the same quality & options as basemap stacking layer 1-3
- NiloToon Character shader: normal map will affect MatCapUV now

- NiloToon Character shader: +_BackFaceForceShadow, _BackFaceReplaceBaseMapColor (including liltoon auto setup convertion)
- NiloToon Character shader: +_ReceiveURPShadowMappingAmountForFace, _ReceiveURPShadowMappingAmountForNonFace
- NiloToon Character shader: + new group "URP Additional Light ShadowMap"
- NiloToon Character shader: + "Light Style" group's preset
- NiloToon Character shader: +_SkinMaskMapAsIDMap, _SkinMaskMapExtractFromID
- NiloToon Character shader: +_SpecularMapAsIDMap, _SpecularMapInvertColor
- NiloToon Character shader: +_DepthTexRimLightMaskTex
- + NormalmapSkin.png, NormalMap_MetalBump.png in Textures folder, as util textures for detailmap group
- more liltoon properties can port to NiloToon via "Auto setup character" (WIP)
- (InternalCore) Enhanced support for Rider. You can read and edit NiloToon source code in Rider, just like reading C#. Utilize Rider functions like 'Go To' (F12), 'Go Back' (Ctrl+-), and 'Find All Usages' (Shift+F12).

Changed
- LWGUI: upgrade 1.4.3 -> 1.6.0
- NiloToonPerCharacterRenderController: remove allowCacheSystem toggle, now NiloToon will auto detect material reference change and perform auto update (no API call due to material reference change is needed by user now, all update is automated)
- NiloToon Character shader: various UI inprovements like helpbox, tooltip, display names....
- Remove console Debug.Log(s) that are not too helpful, so NiloToon will try to keep the console window as clean as possible

Fixed
- NiloToonPerCharacterRenderController: Significant CPU optimization has been achieved when multiple characters use the NiloToon per character script. In tests with 30 characters and 800 materials, the time improved from 19ms to 11ms (speed up = x1.72). The speed up is due to NiloToon stopped a few per frame material updates via NiloToonPerCharacterRenderControllerm, which lead to SRP batcher using a fast path to upload data from CPU to GPU, since no material properties are edited per frame now
- NiloToonPerCharacterRenderController: fix all memory leaks due to material reference changed by user in playmode(e.g., due to playing an animator animation with any material reference key)
- NiloToonPerCharacterRenderController: fix a bug that clicking "don't edit material" in character auto-setup will wrongly skip updating the script's field also. Now after the fix, script's field will always update correctly
- NiloToonUberPostProcessPass: fix a GC alloc bug due to wrong shader name string
- NiloToonBloomVolume: fix bloom RT over max size due to fixedHeight option with a very wide screen, it now cap at the GPU allowed max size, usually it is 16384 width for PC
- package.json: changed "type" from "module" -> "tool", allow NiloToonURP to show up in project window even when installed via package manager(UPM)
- fix a few null reference exception found when entering/exiting playmode or when building the game

---
バージョンのハイライト
簡単に言うと、NiloToon 0.15.15に比べて、NiloToon 0.16.0は以下の重要な変更を導入しました：
- [破壊的変更] 追加ライトのコードを書き直しました。すべての追加ライトは、デフォルトでメインライトとして扱われるようになり、ポイントライト/スポットライトだけでキャラクターをきれいに照らすことができます（インテリアに最適！）
- 新しい「NiloToonCharacterLightController」スクリプトを追加しました。これにより、各キャラクターがライティングをどのように受け取るかを制御できます（例えば、いくつかのターゲットキャラクターに対してメインライトの色を調整したり、色を追加したりすることができます）
- 新しい「NiloToonAdditionalLightStyleVolume」ボリュームを追加しました。これにより、すべてのライトがすべてのキャラクターに対してグローバルにどのように影響するかを制御できます
- 新しい「NiloToonLightSourceModifier」スクリプトを追加しました。これにより、各ライトがすべてのキャラクターにどのように影響するかを制御できます。例えば、（1）通常のポイントライト（2）リムライトなしのポイントライト（3）リムライトのみのポイントライト（4）nilotoonキャラクターに影響を与えないポイントライト...（4）は特に便利です。Unity URPのレンダリングマスクを使用したくない場合に役立ちます。URPのレンダリングレイヤーは、各レンダラーのレイヤーに個別の設定が必要であり、これは時間がかかり、シーンのゲームオブジェクトやプレハブファイルを散らかす可能性があります。
- メニュー「GameObject > Create Other > NiloToon」に多くの新しいオプションを追加
- メニュー「GameObject > Light > NiloToon」に多くの新しいオプションを追加
- エイリアシングを減らすために「ソフトシャドウ」をシャドウマップに導入し、デフォルトで有効になっています（中品質）
- NiloToonキャラクターシェーダーを更新しました：Basemap Stacking Layer 1-10には+7 UVs（UV1-4、MatcapUV、CharBoundUV、ScreenSpaceUV）と4つのブレンドモードオプション（ノーマル、アド、スクリーン、マルチプライ）が含まれるようになりました。例えば、matcapUVを使ってmultiplyまたはaddブレンド方法でbasemap stacking layerを作成することができます。
- NiloToonPerCharacterRenderControllerの「Basemap Override」グループを更新：+7 UVs（UV1-4、MatcapUV、CharBoundUV、ScreenSpaceUV）、および4つのブレンドモードオプション（ノーマル、アド、スクリーン、マルチプライ）
- NiloToonPerCharacterRenderControllerに新しい「Color Fill」グループを追加しました。これは、異なる効果（例えば、X線ビジョン、壁の後ろにいるキャラクターをCS2/Valorantのように表示するなど）でキャラクターを再描画するためのユーティリティ機能です。
- NiloToonPerCharacterRenderControllerに「Generate Smoothed normal?」トグルを新たに追加しました。これを有効にすると、非fbxプレハブ（例えば、vrmキャラクター）でも、fbxキャラクターと同様のアウトライン品質を持つことができます。
- 発見されたすべてのメモリリークとGC割り当ての問題を修正しました。

- 複数のキャラクターが可視状態のときのCPU最適化が大幅に改善されました（約1.72倍のCPUスピードアップ、30個のユニークなキャラクターと800個のユニークなキャラクターマテリアルを含むテストシーンで）。
- HLSLコードがRiderで読みやすくなりました。

開発者からの注意（Unity6/RenderGraph/Warudo）
- NiloToon 0.16.xはUnity2021.3をサポートする最後のバージョンであることが確認されています。
- NiloToon 0.17.0では、Unity2021.3のサポートを完全に削除して、Unity6（RenderGraph）サポートに取り組むことになります。
- あと数回の0.16.xリリースの後、Warudo（Unity2021.3）で公式のNiloToonバージョンとして使用される安定したNiloToon 0.16.xバージョンを提供します。

開発者からの注意（追加ライトの書き換え）
ポイントライト/スポットライトを使用してキャラクターを照らす必要がない場合は、このメッセージをスキップしてください。
NiloToon 0.16.0には、追加ライトのためのコードの大幅な書き換えが含まれています！

前バージョンのNiloToon 0.15.15では、追加ライトの扱いはUnityの方法に似ており、ライトの色はキャラクターに加算的にブレンドされていました。Unity URPシェーダーとして、私たちの元のデザインはできるだけUnityのURP追加ライトの動作に沿うことを目指していましたが、トゥーンシェーダーには最適でないことは知っていました。

しかし、加算ライトの方法は、[このビデオ](https://youtu.be/ST9qNEfPrmY?si=98mqByySiRE7kcTO)のように非常に慎重にセットアップされた条件下でのみ見栄えが良かったです。ほとんどの通常の使用例では、NiloToon 0.15.15のポイントライト/スポットライトはキャラクターを過度に明るくしたり、奇妙に見せたりすることが容易でした。

したがって、NiloToon 0.16.0では、追加ライトがデフォルトで良い結果を提供するようにするための重要な変更を加える時が来たと判断しました。これで、追加ライトはメインライトと同じように扱われるので、キャラクターを照らす方法や使用するライトの種類に関係なく、照明結果は常にメインの方向光と同じ品質を維持します。つまり、インテリア、部屋、洞窟など、主にポイントライト/スポットライトを使用するシーンは、NiloToon 0.16.0を使ってキャラクターを簡単かつ一貫して綺麗に照らすことができるようになりました。

*以前の追加ライトの結果を出力したい場合は、以下の操作を行うことができます：
- NiloToon Additional Light Style Volumeで、色と方向を0に設定します。
- （オプション）NiloToon Cinematic Rim Light Volumeで、「auto fix unsafe style」を無効にします。

破壊的変更
- NiloToon Characterシェーダー：「face normal fix」は、NiloToonPerCharacterRenderControllerによって制御されるマテリアルのみに影響します。
- NiloToon Characterシェーダー：MatCapUVとCharBoundUVの間違ったコードを修正しました。これにより、これらのUVのxが間違って反転するのを修正しました。修正後、MatCapUVとCharBoundUVは、3D球メッシュのTexture2Dと同じように正しく見えるようになり、もうuv xが反転することはありません。ただし、これは破壊的変更をもたらすバグ修正ですので、古いMatCapUVの結果（uv xが反転している）を維持したい場合は、マテリアルの「UV Edit」グループで、MatCapUVのタイリングを(-1,1,_,_)に設定してください。
- NiloToon Characterシェーダー：_BackFaceTintColorはもうアウトラインパスには影響しません。なぜなら、アウトラインが暗くなるのは正しくないからです。
- NiloToon Characterシェーダー：リムライトのuvオフセット方向ベクトルを正規化しました。これにより、すべての照明条件と方向の下でリムライトの幅を一貫させます。
- NiloToon Characterシェーダー：NiloToonDepthOnlyOrDepthNormalPassの顔のノーマル編集を無効にしました。なぜなら、URPの_CameraNormalTextureの顔のノーマルを編集したくないからです。
- NiloToonPerCharacterRenderController：BaseMap OverrideグループのUVタイリングは、中心(0.5,0.5)をタイリングの中心として使用するようになりました。
- NiloToonPerCharacterRenderController：BaseMap Overrideのマップがnullの場合、NiloToonはデフォルトの白い不透明なテクスチャを使用するようになりました。以前は灰色の半透明（50%）テクスチャを使用していました。
- NiloToonPerCharacterRenderController：シーンに多くのキャラクターが含まれている場合、すべてのアクティブなキャラクターを強制的に更新/再インポートすることはなくなりました。なぜなら、それはシーンが多すぎる場合には遅すぎるからです。
- NiloToonPerCharacterRenderController：StencilFill、ExtraThickOutline、ColorFillパスは、不透明なキュー（<2500）のマテリアルのみを描画するようになりました。
- NiloToonCinematicRimLightVolume：「Auto fix unsafe Style?」トグルを追加しました。デフォルトで有効になっており、少なくとも1つのシネマティックリムライトスタイルが強制されます。
- NiloToonCharRenderingControlVolume：specularReactToLightDirectionChangeのデフォルト値をtrueに設定しました。古い結果を再現したい場合は、これを上書きしてfalseに戻すことができます。
- NiloToonVolumePresetアセットのすべて：charOutlineWidthMultiplierを1にリセットしました。

破壊的変更（HLSLプログラミングのみ）
- (InternalCore) NiloToon Characterシェーダー：ToonLightingData.PerPixelNormalizedPolygonNormalWSをnormalWS_NoNormalMapに変更しました。
- (InternalCore) NiloToon Characterシェーダー：ToonLightingData.PolygonNdotVをNdotV_NoNormalMapに変更しました。
- (InternalCore) NiloToonCharacter_ExtendFunctionsForUserCustomLogic.shader：関数のパラメータを変更しました -> void ApplyCustomUserLogicToBaseColor(inout half4 baseColor, Varyings input, UVData uvData, float facing)。

- (InternalCore) NiloToonCharacter_ExtendFunctionsForUserCustomLogic.shader：関数のパラメータを変更しました -> void ApplyCustomUserLogicBeforeFog(inout half3 color, ToonSurfaceData surfaceData, ToonLightingData lightingData, Varyings input, UVData uvData)。
- (InternalCore) NiloToonCharacter_ExtendFunctionsForUserCustomLogic.shader：関数のパラメータを変更しました -> void ApplyCustomUserLogicAfterFog(inout half3 color, ToonSurfaceData surfaceData, ToonLightingData lightingData, Varyings input, UVData uvData)。
- (InternalCore) NiloToon Characterシェーダー：間接照明をメインライト注入フェーズに移動しました。

追加された機能
- 新しいボリューム「NiloToonAdditionalLightStyleVolume」
-「GameObject > ... > NiloToon」メニューの「NiloToonCharacterLightController」新しいスクリプト、およびゲームオブジェクト作成方法を追加
-「GameObject > ... > NiloToon」メニューの「NiloToonLightSourceModifier」新しいスクリプト、およびいくつかのゲームオブジェクト作成方法を追加
- NiloToonAllInOneRendererFeature：ソフトシャドウを追加（デフォルト有効、中品質）
- NiloToonAnimePostProcessVolume：topLightMultiplyLightColorを追加
- NiloToonCharacterMainLightOverrider：overrideTimingを追加
- NiloToonPerCharacterRenderController：新しい「Color Fill」グループを追加
- NiloToonPerCharacterRenderController：LateUpdateで不足しているプロパティを自動的に埋める機能を追加
- NiloToonPerCharacterRenderController：「Generate Smoothed normal?」トグルを追加。非fbxキャラクターに対してもランタイムで同様の品質のアウトラインを生成できるようになりました。
- NiloToonPerCharacterRenderController：「Auto setup character」はPhysicalMaterial3DsMax ArnoldStandardSurfaceの自動設定もサポートするようになりました。
- NiloToonPerCharacterRenderController：BaseMapOverrideグループにUVIndex {UV1-4, MatCapUV, CharBoundUV, ScreenSpaceUV}を追加
- NiloToonPerCharacterRenderController：BaseMapOverrideグループにBlendMode {Normal,Add,Screen,Multiply}を追加
- NiloToon Characterシェーダー：_DepthTexRimLightIgnoreLightDir（360リム）と_DepthTexShadowIgnoreLightDir（360シャドウ）を追加しました。
- NiloToon Characterシェーダー：_SelfShadowAreaHSVStrength を追加しました。
- NiloToon Characterシェーダー：_SkinShadowTintColor2、_FaceShadowTintColor2 を追加しました。
- NiloToon Characterシェーダー：「ノーマルマップ」グループにUVIndex, UVScaleOffset, UVScrollSpeedを追加しました。
- NiloToon Characterシェーダー：新しい「UV Edit」グループを追加し、UV1～4 & MatCapUVにタイリング、回転、アニメーションなどを追加できるようにしました。
- NiloToon Characterシェーダー：すべての「Basemap stacking layers」グループにマスターストレングススライダーを追加しました。
- NiloToon Characterシェーダー：すべての「Basemap stacking layers」グループにBlendMode {Normal,Add,Screen,Multiply}を追加しました。
- NiloToon Characterシェーダー：すべての「Basemap stacking layers」グループにUVIndex {UV1-4, MatcapUV, CharBoundUV, ScreenSpaceUV}を追加しました。
- NiloToon Characterシェーダー：すべての「Basemap stacking layers」グループにUVscale, position, angle, rotateオプションを追加しました。
- NiloToon Characterシェーダー：ベースマップスタッキングレイヤー4-10を更新し、今ではベースマップスタッキングレイヤー1-3と同じ品質とオプションを持っています。
- NiloToon Characterシェーダー：ノーマルマップは、MatCapUVにも影響を与えるようになりました。
- NiloToon Characterシェーダー：_BackFaceForceShadow、_BackFaceReplaceBaseMapColor を追加しました（liltoon自動セットアップ変換も含む）。
- NiloToon Characterシェーダー：_ReceiveURPShadowMappingAmountForFace、_ReceiveURPShadowMappingAmountForNonFaceを追加しました。
- NiloToon Characterシェーダー：新しいグループ「URP Additional Light ShadowMap」を追加しました。
- NiloToon Characterシェーダー：「Light Style」グループのプリセットを追加しました。
- NiloToon Characterシェーダー：_SkinMaskMapAsIDMap, _SkinMaskMapExtractFromIDを追加しました。
- NiloToon Characterシェーダー：_SpecularMapAsIDMap, _SpecularMapInvertColorを追加しました。
- NiloToon Characterシェーダー：_DepthTexRimLightMaskTexを追加しました。
- TexturesフォルダーにNormalmap_Skin.png, NormalMap_MetalBump.pngを追加し、detailmapグループのユーティリティテクスチャとして使用できます。
-「Auto setup character」を介して、より多くのliltoonプロパティをNiloToonに移植できるようになりました（作業中）。
- (InternalCore) Riderのサポートを強化しました。RiderでNiloToonのソースコードをC#のように読んだり編集したりできます。'Go To' (F12)、'Go Back' (Ctrl+-)、'Find All Usages' (Shift+F12)などのRider機能を利用できます。

変更点
- LWGUI: 1.4.3から1.6.0へアップグレードしました。
- NiloToonPerCharacterRenderController：allowCacheSystemトグルを削除し、NiloToonはマテリアルの参照の変更を自動的に検出し、自動更新を実行するようになりました（ユーザーによるAPI呼び出しは不要で、すべての更新は自動化されています）。
- NiloToonキャラクターシェーダー：ヘルプボックス、ツールチップ、表示名などのUIの改善を行いました。
- あまり役に立たないコンソールのDebug.Log(s)を削除し、NiloToonができるだけコンソールウィンドウをクリーンに保つようにしました。

修正点
- NiloToonPerCharacterRenderController：複数のキャラクターがNiloToonのキャラクターごとのスクリプトを使用する場合、CPUの最適化が大幅に向上しました。30キャラクターと800マテリアルでのテストでは、19msから11msに改善しました（スピードアップ = x1.72）。スピードアップは、NiloToonがNiloToonPerCharacterRenderControllerを介してフレームごとのマテリアル更新をいくつか停止したためであり、これによりSRPバッチャーがCPUからGPUへのデータのアップロードを高速パスで使用するようになりました。
- NiloToonPerCharacterRenderController：ユーザーがプレイモードでマテリアルの参照を変更したために発生するメモリリークを修正しました（例：マテリアル参照キーを含むアニメーターアニメーションを再生した場合）。
- NiloToonPerCharacterRenderController：キャラクターの自動セットアップで「マテリアルを編集しない」をクリックすると、スクリプトのフィールドの更新も誤ってスキップされるバグを修正しました。修正後は、スクリプトのフィールドが常に正しく更新されるようになりました。
- NiloToonUberPostProcessPass：誤ったシェーダー名の文字列によるGC割り当てのバグを修正しました。
- NiloToonBloomVolume：非常に広い画面でfixedHeightオプションを使った場合にbloom RTが最大サイズを超えるバグを修正しました。現在はGPUが許可する最大サイズ（通常はPCで16384幅）でキャップされます。
- package.json：「type」を「module」から「tool」に変更し、パッケージマネージャ（UPM）を介してインストールされた場合でもNiloToonURPがプロジェクトウィンドウに表示されるようにしました。
- プレイモードの開始/終了時やゲームのビルド時に発生するいくつかのnull参照例外を修正しました。

版本亮点
简而言之，与 NiloToon 0.15.15 相比，NiloToon 0.16.0 引入了以下重大变化：
- [重大变更] 重写了额外灯光的代码；现在所有额外的灯光默认被当作主灯光处理，只需使用点光源/聚光灯就能很好地照亮角色（非常适合室内！）
- 添加了一个新的 `NiloToonCharacterLightController` 脚本，允许你控制每个角色接收光照的方式（例如，为几个目标角色给主光源着色或添加颜色）
- 添加了一个新的 `NiloToonAdditionalLightStyleVolume` 体积，使你能够全局控制所有灯光如何影响所有角色
- 添加了一个新的 `NiloToonLightSourceModifier` 脚本，使你能够控制每个灯光如何影响所有角色。例如，你可以创建几个具有不同角色的点光源 -> （1）一个普通点光源（2）一个没有边缘光的点光源（3）一个只有边缘光的点光源（4）一个不影响 nilotoon 角色的点光源...对于（4），如果你不愿意使用 Unity URP 的渲染遮罩，这特别方便，因为 URP 的渲染层要求为每个渲染器的层进行单独设置，这可能既费时又会使你的场景的游戏对象和预制文件变得杂乱无章。
- 菜单 "GameObject > Create Other > NiloToon" 中增加了许多新选项
- 菜单 "GameObject > Light > NiloToon" 中增加了许多新选项
- 引入了 'soft shadow'（柔和阴影）用于阴影图，以减少走样，默认启用（中等质量）
- 更新了 NiloToon 角色着色器：基础图堆叠层 1-10 现在包括 +7 UVs（UV1-4, MatcapUV, CharBoundUV, ScreenSpaceUV）和 4 种混合模式选项（正常，添加，屏幕，乘法）。例如，你可以使用 matcapUV 与乘法或添加混合方法创建基础图堆叠层。
- 更新了 NiloToonPerCharacterRenderController 的 'Basemap Override' 组：+7 UVs（UV1-4, MatcapUV, CharBoundUV, ScreenSpaceUV），和 4 种混合模式选项（正常，添加，屏幕，乘法）
- 在 NiloToonPerCharacterRenderController 中添加了新的 'Color Fill' 组，一个实用功能，用于用不同效果重新绘制角色（例如，X光视觉，显示墙后的角色，类似 CS2/Valorant）
- 在 NiloToonPerCharacterRenderController 中添加了新的 'Generate Smoothed normal?' 开关，你可以为非 fbx 预制件启用它（例如，vrm 角色），以便这些非 fbx 角色预制件仍然可以拥有与 fbx 角色相似的轮廓质量
- 修复了发现的所有内存泄漏和 GC 分配问题
- 当多个角色可见时进行了显著的 CPU 优化（在一个测试场景中，有 30 个独特角色和 800 个独特角色材质，大约提速了 1.72 倍）
- 现在在 Rider 中 HLSL 代码更易于阅读

开发者说明（Unity6/RenderGraph/Warudo）
- 确认 NiloToon 0.16.x 是最后一个仍然支持 Unity2021.3 的版本。
- 在 NiloToon 0.17.0 中，我们将完全移除对 Unity2021.3 的支持，以开始支持 Unity6（RenderGraph）。
- 在再发布几个 0.16.x 版本后，我们将提供一个稳定的 NiloToon 0.16.x 版本给 Warudo，作为 Warudo（Unity2021.3）的官方 NiloToon 版本。

开发者说明（附加灯光重写）
如果你不需要使用点光源/聚光灯来照亮角色，请跳过此消息。
NiloToon 0.16.0 包含对附加灯光代码的重大重写！

在之前的版本 NiloToon 0.15.15 中，附加灯光的处理方式类似于 Unity 的方法，灯光的颜色会加性地混合到角色上。作为一个 Unity URP 着色器，我们的原始设计旨在尽可能地与 Unity 的 URP 附加灯光行为保持一致，尽管我们知道这对卡通着色器来说并不是最好的。

然而，加性光方法只有在非常仔细设置的条件下才看起来不错，就像这个[视频](https://youtu.be/ST9qNEfPrmY?si=98mqByySiRE7kcTO)。在大多数常规用例中，NiloToon 0.15.15 的点光源/聚光灯可能会很容易使角色显得过分明亮或看起来奇怪。

因此，在 NiloToon 0.16.0 中，我们决定是时候做出重大的突破性变化，以确保附加灯光默认提供良好的结果。现在，任何附加灯光都被视为与主灯光相同，所以无论你如何照亮角色，无论使用哪种类型的灯光，照明结果始终会保持与主方向光相同的质量。这意味着主要使用点光源/聚光灯的场景，如室内、房间、洞穴等，现在使用 NiloToon 0.16.0 很容易且一致地很好地照亮角色。

*如果你想产生旧的附加光结果，你可以：
- 在 NiloToon Additional Light Style Volume 中：设置颜色和方向 = 0
-（可选）在 NiloToon Cinematic Rim Light Volume 中：禁用 “auto fix unsafe style”

破坏性变更
- NiloToon 角色着色器："face normal fix" 现在只影响由 NiloToonPerCharacterRenderController 控制的材料
- NiloToon 角色着色器：修复了错误的 MatCapUV 和 CharBoundUV 代码，使这些 UV 的 x 错误翻转，现在修复后，MatCapUV 和 CharBoundUV 将正确地与 3D 球体网格上的 Texture2D 看起来相同，不再有 uv x 翻转的问题。但这是一个导致破坏性变更的错误修复，所以如果你想保持旧的 MatCapUV 结果，其中 uv x 翻转了，请转到材料的 "UV Edit" 组，将 MatCapUV 的平铺设置为 (-1,1,_,_)。
- NiloToon 角色着色器：_BackFaceTintColor 不再影响轮廓通过，因为它会使轮廓变暗，这是不正确的
- NiloToon 角色着色器：标准化深度轮廓 uv 偏移方向向量，使得在所有光照条件和方向下边缘光宽度更一致
- NiloToon 角色着色器：禁用了 NiloToonDepthOnlyOrDepthNormalPass 的面法线编辑，因为我们不想编辑 URP 的 _CameraNormalTexture 的面法线
- NiloToonPerCharacterRenderController：BaseMap Override 组的 UV 平铺现在使用中心(0.5,0.5) 作为平铺中心而不是 (0,0)
- NiloToonPerCharacterRenderController：当 BaseMap Override 的 Map 为空时，NiloToon 现在将使用默认的白色不透明纹理而不是灰色和半透明(50%)的纹理
- NiloToonPerCharacterRenderController：不再强制更新/重新导入场景中的所有活动角色，因为当场景中包含太多角色时这会非常慢
- NiloToonPerCharacterRenderController：使得 StencilFill、ExtraThickOutline、ColorFill pass 仅绘制不透明队列（<2500）的材质
- NiloToonCinematicRimLightVolume：添加了 "Auto fix unsafe Style?" 开关，默认启用，这将强制至少开启一个电影边缘光风格
- NiloToonCharRenderingControlVolume：specularReactToLightDirectionChange 现在默认为 true。如果你想重现旧的结果，你可以覆盖它然后将其设置回 false
- 所有 NiloToonVolumePreset 资产：将 charOutlineWidthMultiplier 重置为 1

破坏性变更（仅限 HLSL 编程）
- (InternalCore) NiloToon 角色着色器：将 ToonLightingData.PerPixelNormalizedPolygonNormalWS 重命名为 normalWS_NoNormalMap
- (InternalCore) NiloToon 角色着色器：将 ToonLightingData.PolygonNdotV 重命名为 NdotV_NoNormalMap
- (InternalCore) NiloToonCharacter_ExtendFunctionsForUserCustomLogic.shader：更改了函数参数 -> void ApplyCustomUserLogicToBaseColor(inout half4 baseColor, Varyings input, UVData uvData, float facing)
- (InternalCore) NiloToonCharacter_ExtendFunctionsForUserCustomLogic.shader：更改了函数参数 -> void ApplyCustomUserLogicBeforeFog(inout half3 color, ToonSurfaceData surfaceData, ToonLightingData lightingData, Varyings input, UVData uvData)
- (InternalCore) NiloToonCharacter_ExtendFunctionsForUserCustomLogic.shader：更改了函数参数 -> void ApplyCustomUserLogicAfterFog(inout half3 color, ToonSurfaceData surfaceData, ToonLightingData lightingData, Varyings input, UVData uvData)
- (InternalCore) NiloToon 角色着色器：将间接光移至主光注入阶段

添加
- + 新的体积 "NiloToonAdditionalLightStyleVolume"
- + 新的脚本 "NiloToonCharacterLightController"，在 "GameObject > ... > NiloToon" 菜单中有创建游戏对象的方法
- + 新的脚本 "NiloToonLightSourceModifier"，在 "GameObject > ... > NiloToon" 菜单中有几种创建游戏对象的方法
- NiloToonAllInOneRendererFeature：+ 柔和阴影（默认启用，中等质量）
- NiloToonAnimePostProcessVolume：+ topLightMultiplyLightColor
- NiloToonCharacterMainLightOverrider：+ overrideTiming
- NiloToonPerCharacterRenderController：+ 新的 "Color Fill" 组
- NiloToonPerCharacterRenderController：+ 在 LateUpdate 中自动填充缺失属性
- NiloToonPerCharacterRenderController：+ "Generate Smoothed normal?" 开关，适用于非 fbx 角色，以便在运行时生成类似质量的轮廓
- NiloToonPerCharacterRenderController：现在支持 "Auto setup character" 自动设置 PhysicalMaterial3DsMax ArnoldStandardSurface
- NiloToonPerCharacterRenderController：BaseMapOverride 组 + UVIndex {UV1-4, MatCapUV, CharBoundUV, ScreenSpaceUV}
- NiloToonPerCharacterRenderController：BaseMapOverride 组 + BlendMode {Normal, Add, Screen, Multiply}
- NiloToon 角色着色器：+ _DepthTexRimLightIgnoreLightDir(360度边缘光) 和 _DepthTexShadowIgnoreLightDir(360度阴影)
- NiloToon 角色着色器：+ _SelfShadowAreaHSVStrength
- NiloToon 角色着色器：+ _SkinShadowTintColor2, _FaceShadowTintColor2
- NiloToon 角色着色器："Normal map" 组 + UVIndex, UVScaleOffset, UVScrollSpeed
- NiloToon 角色着色器：+ 新的 "UV Edit" 组，允许你为 UV1~4 & MatCapUV 添加平铺、旋转、动画.....
- NiloToon 角色着色器：所有 "Basemap stacking layers" 组 + 主强度滑块
- NiloToon 角色着色器：所有 "Basemap stacking layers" 组 + BlendMode {Normal, Add, Screen, Multiply}
- NiloToon 角色着色器：所有 "Basemap stacking layers" 组 + UVIndex {UV1-4, MatCapUV, CharBoundUV, ScreenSpaceUV}
- NiloToon 角色着色器：所有 "Basemap stacking layers" 组 + UVscale, position, angle, rotate 选项
- NiloToon 角色着色器：更新了 basemap stacking layer 4-10，现在它们具有与 basemap stacking layer 1-3 相同的质量和选项

- NiloToon 角色着色器：现在 normal map 会影响 MatCapUV
- NiloToon 角色着色器：+_BackFaceForceShadow, _BackFaceReplaceBaseMapColor（包括 liltoon 自动设置转换）
- NiloToon 角色着色器：+_ReceiveURPShadowMappingAmountForFace, _ReceiveURPShadowMappingAmountForNonFace
- NiloToon 角色着色器：+ 新的组 "URP Additional Light ShadowMap"
- NiloToon 角色着色器：+ "Light Style" 组的预设
- NiloToon 角色着色器：+_SkinMaskMapAsIDMap, _SkinMaskMapExtractFromID
- NiloToon 角色着色器：+_SpecularMapAsIDMap, _SpecularMapInvertColor
- NiloToon 角色着色器：+_DepthTexRimLightMaskTex
- + 在 Textures 文件夹中的 Normalmap_Skin.png, NormalMap_MetalBump.png 作为 detailmap 组的工具纹理
- 通过 "Auto setup character" 可以将更多 liltoon 属性移植到 NiloToon（正在进行中）
- (InternalCore) 增强了对 Rider 的支持。你现在可以在 Rider 中阅读和编辑 NiloToon 源码，就像阅读 C# 一样。利用 Rider 的功能，如 'Go To' (F12)，'Go Back' (Ctrl+-)，和 'Find All Usages' (Shift+F12)。

变更
- LWGUI：从 1.4.3 升级到 1.6.0
- NiloToonPerCharacterRenderController：移除了 allowCacheSystem 开关，现在 NiloToon 会自动检测材质引用变化并执行自动更新（用户现在不再需要因材质引用变化而进行 API 调用，所有更新都是自动的）
- NiloToon 角色着色器：各种 UI 改进，如帮助框、工具提示、显示名称.....
- 移除了那些没太大帮助的控制台 Debug.Log(s)，以便 NiloToon 尽可能保持控制台窗口的清洁

修复
- NiloToonPerCharacterRenderController：当多个角色使用 NiloToon 每个角色的脚本时，实现了显著的 CPU 优化。在测试中，包含 30 个角色和 800 个材料的场景，从 19ms 改善到 11ms（速度提升约为 1.72 倍）。这种提速是因为 NiloToon 停止了通过 NiloToonPerCharacterRenderController 进行的每帧材料更新，从而使得 SRP batcher 使用快速路径从 CPU 上传数据到 GPU，因为现在不再每帧编辑材料属性
- NiloToonPerCharacterRenderController：修复了用户在播放模式下更改材料引用（例如，播放包含任何材料引用键的动画器动画）时导致的所有内存泄漏
- NiloToonPerCharacterRenderController：修复了在角色自动设置中点击“不编辑材料”会错误地跳过更新脚本字段的错误。现在修复后，脚本字段将始终正确更新
- NiloToonUberPostProcessPass：由于错误的着色器名称字符串导致的 GC 分配错误已修复
- NiloToonBloomVolume：由于 fixedHeight 选项在非常宽的屏幕上导致 bloom RT 超过最大尺寸，现在在 GPU 允许的最大尺寸上限制，通常 PC 的宽度为 16384
- package.json：将“type”从“module”更改为“tool”，即使通过包管理器（UPM）安装，也允许 NiloToonURP 在项目窗口中显示
- 修复了进入/退出播放模式或构建游戏时发现的几个空引用异常

버전 하이라이트
간단히 말해서, NiloToon 0.15.15에 비해 NiloToon 0.16.0은 이러한 중요한 변경 사항을 소개합니다:
- [주요 변경] 추가 라이트 코드를 재작성했습니다; 이제 모든 추가 라이트는 기본적으로 메인 라이트로 처리되어, 점/스팟 라이트만으로도 캐릭터를 멋지게 조명할 수 있게 되었습니다(인테리어에 적합!).
- 새로운 'NiloToonCharacterLightController' 스크립트를 추가했습니다. 이 스크립트를 통해 각 캐릭터가 조명을 받는 방식을 제어할 수 있습니다(예: 몇몇 타겟 캐릭터들에게 메인 라이트 색상을 틴트하거나 추가하는 등).
- 새로운 'NiloToonAdditionalLightStyleVolume' 볼륨을 추가하여, 전체 캐릭터에 대한 모든 라이트의 영향을 전역적으로 제어할 수 있게 되었습니다.
- 새로운 'NiloToonLightSourceModifier' 스크립트를 추가하여, 각 라이트가 모든 캐릭터에 미치는 영향을 제어할 수 있게 되었습니다. 예를 들어, 다양한 역할을 가진 몇 개의 포인트 라이트를 생성할 수 있습니다 -> (1)일반 포인트 라이트 (2)림 라이트가 없는 포인트 라이트 (3)림 라이트만 있는 포인트 라이트 (4)nilotoon 캐릭터에 영향을 주지 않는 포인트 라이트... (4)는 특히 Unity URP의 렌더링 마스크를 사용하지 않으려는 경우 유용합니다. URP의 렌더링 레이어는 각 렌더러의 레이어에 대한 개별 설정을 요구하기 때문에, 이는 시간이 많이 걸리고 장면의 게임 오브젝트 및 프리팹 파일을 복잡하게 만들 수 있습니다.
- 메뉴 "GameObject > Create Other > NiloToon"에 많은 새로운 옵션을 추가했습니다.
- 메뉴 "GameObject > Light > NiloToon"에 많은 새로운 옵션을 추가했습니다.
- 앨리어싱을 줄이기 위해 'soft shadow'를 그림자 맵에 도입했으며, 기본적으로 활성화되어 있습니다(중간 품질).
- NiloToon 캐릭터 쉐이더를 업데이트했습니다: 베이스맵 스택 레이어 1-10에 이제 +7 UV(UV1-4, MatcapUV, CharBoundUV, ScreenSpaceUV)와 4가지 블렌드 모드 옵션(노멀, 애드, 스크린, 멀티플라이)이 포함되어 있습니다. 예를 들어, matcapUV를 사용하여 멀티플라이나 애드 블렌드 방식으로 베이스맵 스택 레이어를 만들 수 있습니다.
- NiloToonPerCharacterRenderController의 'Basemap Override' 그룹을 업데이트했습니다: +7 UV(UV1-4, MatcapUV, CharBoundUV, ScreenSpaceUV)와 4가지 블렌드 모드 옵션(노멀, 애드, 스크린, 멀티플라이).
- NiloToonPerCharacterRenderController에 새로운 'Color Fill' 그룹을 추가했습니다. 다른 효과로 캐릭터를 다시 그릴 수 있는 유틸리티 기능입니다(예: X레이 비전, CS2/Valorant와 유사하게 벽 뒤에 있는 캐릭터를 보여줌).
- NiloToonPerCharacterRenderController에 새로운 'Generate Smoothed normal?' 토글을 추가했습니다. 이를 비fbx 프리팹(예: vrm 캐릭터)에 활성화할 수 있어, 이러한 비fbx 캐릭터 프리팹도 fbx 캐릭터와 유사한 퀄리티의 윤곽을 유지할 수 있습니다.
- 발견된 모든 메모리 누수와 GC 할당 문제를 수정했습니다.
- 여러 캐릭터가 보일 때 상당한 CPU 최적화를 달성했습니다(30개의 고유 캐릭터와 800개의 고유 캐릭터 재질을 사용한 테스트 장면에서 약 1.72배의 CPU 속도 향상).
- 이제 HLSL 코드가 Rider에서 훨씬 쉽게 읽을 수 있습니다.

개발자의 메모 (Unity6/RenderGraph/Warudo)
- NiloToon 0.16.x는 여전히 Unity2021.3을 지원하는 마지막 버전임이 확인되었습니다.
- NiloToon 0.17.0에서는 Unity2021.3에 대한 지원을 완전히 중단하고 Unity6 (RenderGraph) 지원 작업을 시작합니다.
- 0.16.x 버전을 몇 개 더 출시한 후, Warudo에 사용될 공식 NiloToon 버전으로 안정적인 NiloToon 0.16.x 버전을 제공할 예정입니다(Unity2021.3용).

개발자의 메모 (추가 조명 재작성)
점/스팟 라이트를 사용하여 캐릭터에 조명을 제공할 필요가 없다면 이 메시지는 건너뛰세요.
NiloToon 0.16.0은 추가 라이트 코드에 대한 주요 재작성을 포함하고 있습니다!

이전 버전인 NiloToon 0.15.15에서는 추가 라이트의 처리가 Unity의 방식과 유사했으며, 라이트 색상이 캐릭터에 가산적으로 혼합되었습니다. Unity URP 쉐이더로서, 우리의 원래 디자인은 가능한 한 Unity의 URP 추가 조명 동작과 일치하려고 했지만, 툰 쉐이더에는 최적이 아니라는 것을 알고 있었습니다.

그러나 가산 조명 방법은 [이 비디오](https://youtu.be/ST9qNEfPrmY?si=98mqByySiRE7kcTO)와 같이 매우 신중하게 설정된 조건 하에서만 좋아 보였습니다. 대부분의 일반적인 사용 사례에서 NiloToon 0.15.15의 점/스팟 라이트는 쉽게 캐릭터를 과도하게 밝게 만들거나 이상하게 보이게 했습니다.

따라서 NiloToon 0.16.0에서는 추가 라이트가 기본적으로 좋은 결과를 제공하도록 중요한 변경을 결정할 때라고 생각했습니다. 이제 모든 추가 라이트는 주 라이트와 동일하게 처리되므로, 캐릭터에 어떻게 조명을 하든, 사용하는 라이트의 유형에 상관없이, 조명 결과는 항상 메인 방향성 라이트와 동일한 품질을 유지할 것입니다. 이것은 내부, 방, 동굴 등 주로 점/스팟 라이트를 사용하는 장면들이 이제 NiloToon 0.16.0으로 캐릭터를 아름답고 일관성 있게 조명하는 것이 훨씬 쉬워졌음을 의미합니다.

*예전의 추가 라이트 결과를 만들고 싶다면, 다음과 같이 할 수 있습니다:
- Nilotoon Additional Light Style Volume에서: 색상 & 방향 = 0으로 설정
- (선택사항) Nilotoon Cinematic Rim Light Volume에서: "auto fix unsafe style"을 비활성화

변경 사항
- NiloToon Character shader: "face normal fix"는 이제 NiloToonPerCharacterRenderController에 의해 제어되는 재질에만 영향을 줍니다.
- NiloToon Character shader: MatCapUV & CharBoundUV의 잘못된 코드를 수정하여 이 UV들의 x축을 잘못 뒤집는 문제를 해결했습니다. 이제 수정 후에 MatCapUV & CharBoundUV는 3D 구체 메쉬의 Texture2D와 정확히 같은 모습을 하게 되며, 더 이상 UV x축이 뒤집히지 않습니다. 그러나 이는 버그 수정으로 인한 주요 변경 사항이므로, uv x축이 뒤집힌 예전의 MatCapUV 결과를 유지하고 싶다면, 재질의 "UV Edit" 그룹으로 가서 MatCapUV의 타일링을 (-1,1,_,_)로 설정하세요.
- NiloToon Character shader: _BackFaceTintColor는 이제 윤곽선 패스에 영향을 주지 않으며, 이것은 윤곽선을 어둡게 만들어 올바르지 않기 때문입니다.
- NiloToon Character shader: 모든 조명 조건과 방향에서 림 라이트 너비를 일관되게 만들기 위해 깊이 림 UV 오프셋 방향 벡터를 정규화합니다.
- NiloToon Character shader: URP의 _CameraNormalTexture의 얼굴 법선을 편집하지 않으려고 하므로 NiloToonDepthOnlyOrDepthNormalPass의 얼굴 법선 편집을 비활성화합니다.
- NiloToonPerCharacterRenderController: BaseMap Override 그룹의 UV 타일링은 이제 (0,0) 대신 중심(0.5,0.5)을 타일링 센터로 사용합니다.
- NiloToonPerCharacterRenderController: BaseMap Override의 Map이 null일 때, NiloToon은 이제 회색 & 반투명(50%) 텍스처 대신 기본 흰색 불투명 텍스처를 사용합니다.
- NiloToonPerCharacterRenderController: 장면에 너무 많은 캐릭터가 포함되어 있을 때 너무 느려지기 때문에 더 이상 장면에 있는 모든 활성 캐릭터를 강제로 업데이트/재가져오기하지 않습니다.
- NiloToonPerCharacterRenderController: StencilFill, ExtraThickOutline, ColorFill 패스가 이제 불투명 큐(<2500) 재질만 그리도록 만듭니다.
- NiloToonCinematicRimLightVolume: "Auto fix unsafe Style?" 토글을 추가하여 기본적으로 활성화되며, 적어도 하나의 시네마틱 림 라이트 스타일을 강제로 적용합니다.
- NiloToonCharRenderingControlVolume: specularReactToLightDirectionChange의 기본값이 이제 true입니다. 옛 결과를 재현하고 싶다면 재정의하여 false로 다시 설정할 수 있습니다.
- 모든 NiloToonVolumePreset 자산: charOutlineWidthMultiplier을 1로 리셋

주요 변경 사항(HLSL 프로그래밍 전용)
- (InternalCore) NiloToon 캐릭터 쉐이더: ToonLightingData.PerPixelNormalizedPolygonNormalWS의 이름을 normalWS_NoNormalMap으로 변경
- (InternalCore) NiloToon 캐릭터 쉐이더: ToonLightingData.PolygonNdotV의 이름을 NdotV_NoNormalMap으로 변경
- (InternalCore) NiloToonCharacter_ExtendFunctionsForUserCustomLogic.shader: 함수 매개변수 변경 -> void ApplyCustomUserLogicToBaseColor(inout half4 baseColor, Varyings input, UVData uvData, float facing)
- (InternalCore) NiloToonCharacter_ExtendFunctionsForUserCustomLogic.shader: 함수 매개변수 변경 -> void ApplyCustomUserLogicBeforeFog(inout half3 color, ToonSurfaceData surfaceData, ToonLightingData lightingData, Varyings input, UVData uvData)

- (InternalCore) NiloToonCharacter_ExtendFunctionsForUserCustomLogic.shader: 함수 매개변수 변경 -> void ApplyCustomUserLogicAfterFog(inout half3 color, ToonSurfaceData surfaceData, ToonLightingData lightingData, Varyings input, UVData uvData)
- (InternalCore) NiloToon 캐릭터 쉐이더: 간접광을 메인 라이트 주입 단계로 이동

추가된 기능
- + 새로운 볼륨 "NiloToonAdditionalLightStyleVolume"
- + 새로운 스크립트 "NiloToonCharacterLightController"는 "GameObject > ... > NiloToon" 메뉴에서 게임 오브젝트를 생성하는 방법을 제공합니다.
- + 새로운 스크립트 "NiloToonLightSourceModifier"는 "GameObject > ... > NiloToon" 메뉴에서 몇 가지 게임 오브젝트를 생성하는 방법을 제공합니다.
- NiloToonAllInOneRendererFeature: + 소프트 쉐도우(기본 활성화, 중간 품질)
- NiloToonAnimePostProcessVolume: + topLightMultiplyLightColor
- NiloToonCharacterMainLightOverrider: + overrideTiming
- NiloToonPerCharacterRenderController: + 새로운 "Color Fill" 그룹
- NiloToonPerCharacterRenderController: + LateUpdate에서 누락된 속성을 자동으로 채움
- NiloToonPerCharacterRenderController: +"Generate Smoothed normal?" 토글, non-fbx 캐릭터에도 실행 시 유사한 품질의 베이크된 윤곽을 생성할 수 있습니다.
- NiloToonPerCharacterRenderController: "Auto setup character"는 이제 PhysicalMaterial3DsMax ArnoldStandardSurface의 자동 설정을 지원합니다.
- NiloToonPerCharacterRenderController: BaseMapOverride 그룹 + UVIndex {UV1-4, MatcapUV, CharBoundUV, ScreenSpaceUV}
- NiloToonPerCharacterRenderController: BaseMapOverride 그룹 + BlendMode {Normal, Add, Screen, Multiply}
- NiloToon 캐릭터 쉐이더: + _DepthTexRimLightIgnoreLightDir(360도 림) 및 _DepthTexShadowIgnoreLightDir(360도 쉐도우)
- NiloToon 캐릭터 쉐이더: + _SelfShadowAreaHSVStrength
- NiloToon 캐릭터 쉐이더: + _SkinShadowTintColor2, _FaceShadowTintColor2
- NiloToon 캐릭터 쉐이더: "Normal map" 그룹 + UVIndex, UVScaleOffset, UVScrollSpeed
- NiloToon 캐릭터 쉐이더: 새로운 "UV Edit" 그룹을 추가하여 UV1~4 및 MatcapUV에 타일링, 회전, 애니메이션 등을 추가할 수 있습니다.
- NiloToon 캐릭터 쉐이더: 모든 "Basemap stacking layers" 그룹에 주 강도 슬라이더를 추가
- NiloToon 캐릭터 쉐이더: 모든 "Basemap stacking layers" 그룹에 BlendMode {Normal,Add,Screen,Multiply}를 추가
- NiloToon 캐릭터 쉐이더: 모든 "Basemap stacking layers" 그룹에 UVIndex {UV1-4, MatcapUV, CharBoundUV, ScreenSpaceUV}를 추가
- NiloToon 캐릭터 쉐이더: 모든 "Basemap stacking layers" 그룹에 UVscale, position, angle, rotate 옵션을 추가
- NiloToon 캐릭터 쉐이더: basemap stacking layer 4-10을 업데이트하여 이제 basemap stacking layer 1-3과 동일한 품질 및 옵션을 가짐
- NiloToon 캐릭터 쉐이더: 이제 normal map은 MatCapUV에 영향을 줍니다.
- NiloToon 캐릭터 쉐이더: +_BackFaceForceShadow, _BackFaceReplaceBaseMapColor (liltoon 자동 설정 변환 포함)
- NiloToon 캐릭터 쉐이더: +_ReceiveURPShadowMappingAmountForFace, _ReceiveURPShadowMappingAmountForNonFace
- NiloToon 캐릭터 쉐이더: 새 그룹 "URP Additional Light ShadowMap" 추가
- NiloToon 캐릭터 쉐이더: "Light Style" 그룹의 프리셋 추가
- NiloToon 캐릭터 쉐이더: +_SkinMaskMapAsIDMap, _SkinMaskMapExtractFromID
- NiloToon 캐릭터 쉐이더: +_SpecularMapAsIDMap, _SpecularMapInvertColor
- NiloToon 캐릭터 쉐이더: +_DepthTexRimLightMaskTex
- Textures 폴더에 Normalmap_Skin.png, NormalMap_MetalBump.png를 유틸리티 텍스처로 추가하여 detailmap 그룹에 사용
- "Auto setup character"를 통해 더 많은 liltoon 속성을 NiloToon으로 이전할 수 있음 (작업 중)
- (InternalCore) Rider 지원 향상. 이제 Rider에서 NiloToon 소스 코드를 C#처럼 읽고 편집할 수 있습니다. 'Go To' (F12), 'Go Back' (Ctrl+-), 'Find All Usages' (Shift+F12) 같은 Rider 기능을 활용할 수 있습니다.

변경된 사항
- LWGUI: 버전 1.4.3에서 1.6.0으로 업그레이드
- NiloToonPerCharacterRenderController: allowCacheSystem 토글을 제거하고, 이제 NiloToon은 재질 참조 변경을 자동으로 감지하고 자동 업데이트를 수행합니다(사용자는 이제 재질 참조 변경으로 인한 API 호출이 필요 없으며, 모든 업데이트가 자동화됨).
- NiloToon 캐릭터 쉐이더: 도움말 상자, 툴팁, 표시 이름 등과 같은 다양한 UI 개선 사항.
- 유용하지 않은 콘솔 Debug.Log(s)를 제거하여 NiloToon이 콘솔 창을 가능한 한 깨끗하게 유지하려고 합니다.

수정된 사항
-NiloToonPerCharacterRenderController: 여러 캐릭터가 NiloToon 개별 캐릭터 스크립트를 사용할 때 상당한 CPU 최적화를 달성했습니다. 30개 캐릭터와 800개의 재질을 사용한 테스트에서 시간이 19ms에서 11ms로 개선되었습니다(속도 향상 = x1.72). 이 속도 향상은 NiloToonPerCharacterRenderController를 통한 몇몇 프레임당 재질 업데이트를 중단함으로써 이루어졌으며, 이는 SRP 배처가 CPU에서 GPU로 데이터를 업로드하는 빠른 경로를 사용하게 만들었기 때문입니다. 이제 재질 속성이 프레임당 수정되지 않습니다.
- NiloToonPerCharacterRenderController: 플레이 모드에서 사용자가 재질 참조를 변경함으로써 발생한 모든 메모리 누수를 수정했습니다(예: 재질 참조 키가 포함된 애니메이터 애니메이션을 재생하는 경우).

- NiloToonPerCharacterRenderController: 캐릭터 자동 설정에서 "재질을 편집하지 않음"을 클릭하면 스크립트 필드의 업데이트가 잘못 건너뛰어지는 버그를 수정했습니다. 이제 수정 후에는 스크립트 필드가 항상 올바르게 업데이트됩니다.
- NiloToonUberPostProcessPass: 잘못된 쉐이더 이름 문자열로 인해 발생한 GC 할당 버그를 수정했습니다.
- NiloToonBloomVolume: 매우 넓은 화면에서 fixedHeight 옵션으로 인해 블룸 RT가 최대 크기를 초과하는 문제를 수정했습니다. 이제 GPU가 허용하는 최대 크기에 맞춰져 있으며, 보통 PC의 경우 너비는 16384입니다.
- package.json: "type"을 "module"에서 "tool"로 변경하여, 패키지 관리자(UPM)를 통해 설치되었을 때에도 NiloToonURP가 프로젝트 창에 표시되도록 했습니다.
- 플레이 모드에 들어가거나 나올 때 또는 게임을 빌드할 때 발견된 몇 가지 null 참조 예외를 수정했습니다.

[0.15.15] - 2023-12-13

Fixed
- NiloToonUberPostProcessPass.cs: Unity 2022.3 or above's code compile error
---
[0.15.14] - 2023-12-13

Added
- Tonemapping: +GT Tonemap mode (GranTurismo Tonemap, usually a better mode for NPR than ACES/Neutral)
- Tonemapping: + more settings and improve volume UI
- Bloom: + Post-Bloom Result brightness control

Changed
- All script's with LateUpdate has a later ExecuteOrder than default
---
[0.15.13] - 2023-12-11

Note to user
- Changelog will try to be as short as possible from now on, to allow artist(non programmer) to read it easier

Breaking Change
- Character shader: reduce top rim light width fadeout range from 10% to 5%

Added
- VolumePresetPicker: +024,025 preset

Changed
- Character shader: move "Lighting Style" to normal UI group, so it can be edited using default UI
- LWGUI: upgrade from 1.13.6 -> 1.14.3
- Auto Install: hide DebugLog on recompile or editor start

Fixed
- Shaders & ShaderLibrary folders: + asmdef files, to allow .cs generate for IDE
- VolumePresetPicker: has it's own editor script now, to allow better asembly structure
---
[0.15.12] - 2023-12-06

Breaking Change
- (Core) NiloToonVolumePresetPicker: rewrite the whole script, delete all sub volume prefabs. Now NiloToonVolumePresetPicker.prefab is much simplier to understand, check volume values and use.
- (Core) changes NiloToonVolumePreset 002,004,006,009,010,011,012,018. Make them more conservative to fit user's usecase

Added
- (Core) NiloToonEditorPerCharacterRenderControllerCustomEditor.cs: Auto setup character button will now handle the material conversion of "Complex Lit", "Simple Lit", "Unlit" also
- (Core) + 019~023 NiloToonVolumePreset profile
- (InternalCore) +Runtime\Utility\NiloToonRenderingUtils.cs

Changed
- (Core) NiloToonCharacter.shader: rename title "Draw Outline in Depth Texture" -> "Support Depth of Field"

Fixed
- (Core) NiloToonUberPostProcessPass.cs: fix null exception on editor first startup or NiloToon's first install
- (Core) NiloToonAnimePostProcessPass.cs: fix enderingUtils.fullscreenMesh obsolete warning
---
[0.15.11] - 2023-11-24

Added
- (Core) NiloToonEditorPerCharacterRenderControllerCustomEditor.cs: +lilToon converter, when "auto setup character" from lilToon(1.4.1) characters, NiloToon will now try to port properties from lilToon material to NiloToon material (still WIP, can only port basic properties now)
- (Core) NiloToonEditorPerCharacterRenderControllerCustomEditor.cs: new menuitem "Window/NiloToonURP/Setup selected character GameObject", it is just adding a "NiloToonPerCharacterRenderController" script on that gameobject, and click the "auto setup" button for you
- (Core) NiloToonCharacter.shader: _AlphaOverrideTexInvertColor, _AlphaOverrideTexValueScale, _AlphaOverrideTexValueOffset
- (Core) NiloToonCharacter.shader: _AlphaOverrideMode added subtract type at index 3
- (Core) NiloToonAnimePostProcessPass.cs: expose _TopLightSunTintColor
- (InternalCore) +[DisableIf()] Attribute

Changed
- (Core) NiloToonEditorPerCharacterRenderControllerCustomEditor.cs: For Unity2022.3 or higher, auto setup character will not reimport inactive character in scene anymore, for setup speed optimization
- (Core) NiloToonCharacterMainLightOverrider.cs: disable property edit if the override toggle is off
- (Core) Auto setup character will now consider material with "body" as IsSkin material

Fixed
- (Core) NiloToonPerCharacterRenderController.cs: fix bound null exception, when character has null renderers
- (Core) NiloToonUberPostProcessPass.cs: fix NiloToonTonemapping bug -> when NiloToonTonemapping is off, the character becomes pure black when NiloToonBloom is on
- (Core) NiloToonUberPostProcessPass.cs: NiloToonTonemapping will try to run on HDR Grading mode also, even if the result is not be accurate
- (Core) NiloToonRendererFeatureAutoAdd.cs: fix null exception when project contains BIRP/HDRP RPasset
- (Core) NiloToonEditorPerCharacterRenderControllerCustomEditor.cs: fix null exception when auto setup a character with null renderer/mesh
- (Core) NiloToonCharacter_Shared.hlsl: fix "HQ continuous outline" toggle not working, fix "classic outline" disappear when extrathick outline is on
---
[0.15.10] - 2023-11-21

Added
- + NiloToonTonemappingVolume.cs, a new volume component that is similar to URP's tonemapping, but will not apply to character pixels. Using this instead of URP's tonemapping will keep the character area's color unaffected, which is usually good for NPR characters
- (Core) + many NiloToon volume presets (008~018)
- (Core) NiloToonRendererRedrawer.cs: +deactivateWhenRendererIsNotVisible (default true) to save CPU time from offscreen characters
- (Core) NiloToonCharacterMainLightOverrider.cs: +[CanEditMultipleObjects]
- (Core) NiloToonRendererRedrawer.cs: +[CanEditMultipleObjects]

Changed
- [MenuItem("Window/NiloToonURP/Utility/Select all Materials using NiloToon_Character")] to [MenuItem("Window/NiloToonURP/Select all NiloToon_Character Materials", priority = 20)]
- Shaders\NiloToonCharacter.shader: improve stencil UI, + preset
- Runtime\Volume\NiloToonCharRenderingControlVolume.cs: charIndirectLightMaxColor default white->gray
---
[0.15.9] - 2023-11-19

Breaking Change
- (Core) move VolumePrefabs folder into Runtime folder, so NiloToonURP can be installed from disk in package manager
- (Core) rename folder of VolumePrefabs to VolumePresetPicker
- (Core) NiloToon Character shader: +_UIDisplayMode (default Normal), it will hide uncommon groups from the material inspector, to help new user focus on important groups only. User can decide hiding how many groups. This option is at the top position in the material inspector.

Added
- (Core) NiloToon Character shader: _SkinFaceShadowColorPreset, _StencilPreset, _EnableShadowColor, _BaseMapBrightness

Changed
- (Core) NiloToonEditorCreateObjectMenu.cs: use full name of NiloToonCharacterMainLightOverrider in MenuItem to help user search it in editor

- (Core) NiloToonShaderStrippingSettingSO.cs: CreateAssetMenu use a shorter name = CreateNiloToonShaderStrippingSettingSO -> NiloToonShaderStrippingSetting
- (Core) NiloToonCharacter.shader: merge group "Shadow Color Map" into group "Shadow Color"
- (Core) upgrade LWGUI from 1.13.5 -> 1.13.6

[0.15.8] - 2023-11-15

Added
- (Core) All NiloToon volumes: + 'Show Help Box?', 'Hide non-override?' toggles
- (Core) All NiloToon volumes: some properties editor display name renamed to shorter names
- (Core) All NiloToon volumes: some properties + tooltips

Fixed
- (Core) Runtime\NiloToonRendererRedrawer.cs: fix a bug that the draw doesn't match MeshRenderer in scale
---
[0.15.7] - 2023-11-14

Breaking Change
- (Core) NiloToon will try to automatically add NiloToon's renderer features to all active renderers of user's project, if user allow it via a pop up window. If this new auto install pop up window is very annoying, let us know your use case, contact us (nilotoon@gmail.com)

Added
- (Core) + Editor\RendererFeatureAutoAdd\NiloToonRendererFeatureAutoAdd.cs, this script will handle all "auto install NiloToon renderer feature" logic
- (Core) RendererFeatureAutoAdd\NiloToonRendererFeatureAutoAdd.cs: allow user to add NiloToon's renderer feature via [MenuItem("Window/NiloToonURP/Auto Install RendererFeatures", priority = 10)]
- (Core) NiloToonCharRenderingControlVolume: + "Hide non-override?" toggle
- (Core) NiloToonCharRenderingControlVolume: + helpbox

Changed
- (Core) LWGUI upgraded from 1.13.5(dev) to 1.13.5(release)

Fixed
- (Core) Editor\NiloToonEditorShaderStripping.cs: fix ShaderVariantLogLevel compile error in Unity2022.3

[0.15.6] - 2023-11-13

Added
- (Core) + NiloToonVolumePresetPicker.cs and NiloToonVolumePresetPicker.prefab, allow user to use some volume profile preset provided by NiloToon
- (Core) + able to create NiloToonVolumePresetPicker.prefab from GameObject menu "GameObject/Volume/NiloToon/NiloToonVolumePresetPicker"
- (Core) + Nilo000, Nilo005, Nilo006 volume prefabs and their volume profiles
- (Core) + NiloToonVolumeComponentEditor.cs, allow better GUI for VolumeComponent
- (Core) + NiloToonCharRenderingControlVolumeEditor.cs, for improve GUI using NiloToonVolumeComponentEditor

Changed
- (Core) NiloToonCharRenderingControlVolume.cs: rename the display name and header of all properties, improve user experience a lot due to new display name

Fixed
- (Core) NiloToonRendererRedrawer.cs: fix rendering not sync with parent lossy scale bug
- (Core) NiloToonRendererRedrawer.cs: fix a bug that user can't switch to another material in runtime
- (InternalCore) NiloToon.NiloToonURP.Editor.asmdef: +Unity.RenderPipelines.Core.Runtime
---
[0.15.5] - 2023-11-12

Fixed
- (Core) NiloToonRendererRedrawer.cs: use SkinnedMeshRenderer.BakeMesh() to ensure Graphics.RenderMesh() 100% sync with SkinnedMeshRenderer
- (Core) NiloToonPerCharacterRenderController.cs: fix asset bundle load error due to generate material instance in OnValidate()

[0.15.4] - 2023-11-11

Breaking Change

- (Core) NiloToonRendererRedrawer.cs: Completely rewrite NiloToonRendererRedrawer! It is now a much more practical script, you can use it like an extension renderer on top of Unity's Renderer, allow you to add extra materials to a Unity's Renderer, and you can decide which SubMesh to draw. The material to draw will be the same as rendered by any Unity's Renderer (material's render queue, shader passes, lighting & shadow, batching... will all work as expected, due to the use of Graphics.RenderMesh() in LateUpdate() instead of cmd.DrawRenderer() in renderer feature). Now this script works in build & editor(edit mode, play mode, prefab mode).
- (InternalCore) Deprecate NiloToonRendererRedrawerPass.cs, due to NiloToonRendererRedrawer's complete rewrite

Added
- (InternalCore) + Shaders\NiloToonEnvironment_HLSL\NiloToonEnvironment_ExtendFunctionsForUserCustomLogic.hlsl, allow you to inject hlsl code into the nilotoon environment shader
- (Core) Added 2 more example NiloToon volume preset prefabs, for user copy or use it (NiloToonURP/ExampleAssets/VolumePrefabs)
- (Core) NiloToonEditorPerCharacterRenderControllerCustomEditor.cs: also find "spine" as extra alternative bone if no "customCharacterBoundCenter" found

Changed
- (InternalCore) NiloToonEnvironment_LitForwardPass.hlsl: apply NiloToonEnvironment_ExtendFunctionsForUserCustomLogic.hlsl
- (Core) NiloToonCinematicRimLightVolume.cs: strengthRimMask3D_ClassicStyle removed experimental tag
- (Core) NiloToonPerCharacterRenderController: OnValidate() always call LateUpdate() in playmode, to ensure game window correctness when user editing inspector value

Fixed
- (Core) NiloToonPerCharacterRenderController.cs: fix an important build crash bug when headBoneTransform is null
- (Core) NiloToonCharacterMainLightOverrider.cs: +[DisallowMultipleComponent]

[0.15.3] - 2023-11-06

Breaking Change
- (Core) Added new shadow receiver depth bias & norma bias, default ON. These new shadow receiver bias are not the same as the already existed shadow caster's bias. These new shadow receiver bias should solve most of the shadow acne problem, in any shadowmap size and zooming scale(shadow range).
- (Core) NiloToonPerCharacterRenderController.cs: Added support to VRMBlendShapeProxy's material control in playmode, but now NiloToonPerCharacterRenderController will generate material instances on OnEnabled() instead of LateUpdate(), to make sure material instances are generated before VRMBlendShapeProxy's Start().
- (Core) NiloToonPerCharacterRenderController.cs: + toggle keepMaterialInstanceNameSameAsOriginal (default ON) in a new "Misc" group, for supporting VRMBlendShapeProxy
- (Core) NiloToonCharacter_RenderFace_LWGUI_ShaderPropertyPreset: + control to property _CullNiloToonSelfShadowCaster. To avoid nilotoon character produce bad shadow on double face polygons, now the preset is "when rendering front face, only back face polygon will be renderered into NiloToon's character self shadowmap".

Added
- (Core) NiloToonAllInOne renderer feature: + receiverDepthBias,receiverNormalBias
- (Core) NiloToon Character shader: +_NiloToonSelfShadowIntensity
- (Core) NiloToonCharacter_ExtendFunctionsForUserCustomLogic.hlsl: added new method ApplyCustomUserLogicToVertexShaderOutputAtVertexShaderEnd(...)
- (Core) NiloToonCharacterMainLightOverrider.cs: added helpbox to hint user "main directional light's intensity must be > 0"

Changed
- (Core) NiloToonCharacter.shader: Apply more LWGUI new feature to UI, e.g.,"Receive Nilo ShadowMap?"/"Is Skin?"/"Is Face?" sections
- (Core) NiloToonCharSelfShadowMapRTPass.cs: default shadowMapSize 4096->8192
- (Core) NiloToonCharSelfShadowMapRTPass.cs: defualt normalBias 1->0.5

Fixed
- (Core) NiloToonEditorPerCharacterRenderControllerCustomEditor.cs: add more null material/renderer check when doing "Auto setup character"
- (Core) Update LWGUI from 1.13.4 to 1.13.5, which will fix more material UI null exception bugs

---

[0.15.2] - 2023-10-31

Added
- (Core) Added 1 more example NiloToon volume preset prefabs, for user copy or use it (NiloToonURP/ExampleAssets/VolumePrefabs)

Fixed
- (Core) Update LWGUI from 1.13.2 to 1.13.4, which will fix most of the material UI bugs
---
[0.15.1] - 2023-10-30

Added
- (Core) Added 2 example NiloToon volume preset prefabs, let user copy or use it (NiloToonURP/ExampleAssets/VolumePrefabs)

Changed
- (Core) Update LWGUI from 1.13.0 to 1.13.2, which will fix some material UI bugs
- (Core) NiloToonCharacter.shader: rename Face Shadow Gradient Map -> Face Shadow Gradient Map (SDF)
- (Core) NiloToonPerCharacterRenderController.cs: better log when NiloToon renderer feature is not yet setup
- (Core) NiloToon character shader: remove face additional light when cinematic rim light is on
- (Core) NiloToon character shader: add a fix to "2D rim light is weird at screen edge" problem (e.g., hair 2D rim light touching the top side of the screen)

Fixed
- (Core) NiloToonAllInOneRendererFeature: fix a possible missing/null pass
---
[0.15.0] - 2023-10-24

Version Highlight
- Started a UI rework on NiloToon character shader and NiloToonShaderStrippingSettingSO
- Tested with all 5 available Unity versions from the UnityHub(Unity2021.3.31f1 ~ Unity2023.3.0a10)
- Solved all GC Alloc and RTHandle performance problem
- Build is now faster, smaller, and has lower shader memory usage if user rely on the default stripping settings
- We expect NiloToon 0.15.x will become the next stable version, replacing 0.13.6 within the year 2023

Known Issue
- For Unity 2022.3.x, please use version 2022.3.11f1 or later to avoid the URP RenderTexture memory leak issue found in earlier versions (e.g., Unity 2022.3.9f1). Although this is not a NiloToon bug, the severity of this URP issue necessitates user notification.
- In Unity 2022.3 or later, the material UI for NiloToon_Character might disappear from the Material Inspector when two conditions are met: 1) in PlayMode, and 2) when the material becomes an instance due to the C# Renderer.materials call. This issue can be reproduced when viewing the material instance from a Renderer component. We are currently awaiting a resolution from LWGUI.

Breaking change
- (Core) NiloToonShaderStrippingSettingSO.cs: In previous versions, NiloToonShaderStrippingSettingSO attempted to include most of the possible shader keywords in the build to ensure correct keywords by default. However, this resulted in high default build time, size, and shader memory usage. To mitigate this, the default setting for NiloToon shader stripping will now be more aggressive, significantly reducing these metrics. To ensure all required keywords of your project are correctly included in build, you should configure NiloToonShaderStrippingSettingSO for each project to achieve the desired build stripping.
- (Core) NiloToonShaderStrippingSettingSO.cs: +WebGL stripping override (same default setting as mobile), it is default enabled so it is a breaking change
- (Core) NiloToonCharRenderingControlVolume: + charIndirectLightMaxColor, default clamp at white. If you have bright(intensity > 1) light probes baked, this change will affect you
- (Core) NiloToon Character shader: change the apply order of matcap (color replace), environment reflection. Now environment reflection is applied after matcap (color replace)
- (Core) NiloToon Character shader: fix the apply order of BackFaceColorTint and PerCharacterBaseMapOverride. Now PerCharacterBaseMapOverride is applied after BackFaceColorTint
- (InternalCore) NiloToonSetToonParamPass.cs: remove the public static access of EnableDepthTextureRimLigthAndShadow

Added
- (Core) NiloToon Character shader: +_OverrideShadowColorByTexMode,_OverrideShadowColorMaskMap
- (Core) NiloToon Character shader: +_EmissionMapUVScrollSpeed
- (Core) NiloToon Character shader: +_AlphaOverrideMode
- (Core) NiloToon Character shader: NiloToon self shadow map now supports XR
- (Core) + Textures\PureWhite4x4.png
- (Core) + more Tooltips to NiloToon renderer features properties
- (InternalCore) NiloToon Character shader: + WIP ScreenSpace Outline V2 code, it is still in a very early stage WIP, so user can't use it now.
- (InternalCore) NiloToon Character shader: + Rider resharper support, the support is WIP and not fully complete, but now you can see some nice hlsl code highlight and autopcomplete when reading the hlsl in Rider.

Changed
- (Core) NiloToonCharacter.shader: initiated a UI rework(without breaking change), with further improvements scheduled for release in subsequent 0.15.x versions. The overhaul will include features like "Group sorted based on importance", "Better Display Name", "Improved Property Folding", "Hiding Unused Groups and Properties by Default", "UI display mode(Simple/Advanced)" among others. The primary objective is to streamline the material UI, reducing its vertical UI length and learning curve by default. This way, users can work more efficiently without the need for excessive mouse wheel scrolling or group searching by text.
- (Core) Upgrade to LWGUI 1.13.0
- (Core) NiloToonShaderStrippingSettingSO.cs: rewrite the UI display
- (Core) NiloToonCharacter.shader: rename matcap (alpha blend) to (replace)
- (Core) NiloToonCharacter.shader: rename matcap (additive) to matcap(specular highlight/rim light)
- (Core) NiloToonCharacter.shader: _UnityCameraDepthTextureWriteOutlineExtrudedPosition is now default off, since it may produce 2D rim light artifact too often on some model
- (Core) NiloToonEditorPerCharacterRenderControllerCustomEditor.cs: limit MToon to NiloToon's converted outline width to a maximum of 0.6
- (InternalCore) NiloToonEditorPerCharacterRenderControllerCustomEditor.cs: ban "surface" for face's mat name keyword
- (InternalCore) NiloHSVRGBConvert.hlsl: rgb2hsv code now use SRP Core library's function, instead of custom code
- (InternalCore) any camera name containing 'reflect' will be treated as planar reflection camera now, before it is 'reflection' instead of 'reflect'
- (InternalCore) NiloToonPrepassBufferRTPass.cs: rename NiloToonPrepassBufferColor->_NiloToonPrepassBufferColor, and, NiloToonPrepassBufferDepth->_NiloToonPrepassBufferDepth

Fixed
- (Core) Correctly fix all GC/RTHandle problem in Unity2022.3 ~ Unity2023.3, pass and RTHandle dispose logic now follows URP 2022.3.11f1's RTHandle style
- (Core) NiloToon Character shader: fix bug -> OpenGLES's depth tex 2D rim light produced in opposite direction
- (Core) NiloToon Character shader: fix bug -> camera normal texture not having correct normal for Unity2021.3
- (Core) NiloToon Character shader: fix bug -> enableDepthTextureRimLightAndShadow always false when material is not controlled by pechar script
- (Core) NiloToon Character shader: EnableDepthTextureRimLigthAndShadow now set by renderer feature instead of per char script, to ensure value is always updated correctly every frame in edit mode
- (Core) NiloToonPerCharacterRenderController.cs: fix bug -> renderer's material lost reference when editing prefab in prefab mode while running in playmode
- (Core) NiloToonEditorPerCharacterRenderControllerCustomEditor.cs: 'auto setup this char' button now supports all 14 SurfaceType of NiloToon Character material
- (Core) NiloToonAverageShadowTestRTPass.cs: fix bug -> some samsung mobile device not supporting R16_UNorm
- (InternalCore) NiloScaledScreenParamUtil.hlsl: fix bug -> GetScaledScreenTexelSize() return incorrect result
---
[0.14.5] - 2023-08-26

Fixed
- (Core) NiloToonCharSelfShadowMapRTPass.cs: fix NiloToonCharacterMainLightOverrider didn't override direction of this self shadow pass
---
[0.14.4] - 2023-08-25

Fixed
- (Core) NiloToonAllInOneRendererFeature.cs: Correctly fix all RT memory leak by calling extra Dispose(), revert temp fix in 0.14.3

[0.14.3] - 2023-08-25

Known Issue
- (Core) NiloToonCharSelfShadowMapRTPass will have RT memory leak when user edit NiloToon's renderer feature(e.g. dragging the slider), now for Unity2022.3 or lower we converted back to old RenderTargetHandle system as a temp fix, but the memory leak will still exist for Unity2023.1 or higher

Changed
- (Core) rewrite NiloToonAllInOneRendererFeature's UI completely
- (Core) NiloToonCharacter.shader: _ReceiveSelfShadowMappingPosOffset's default value revert from 1 back to 0
- (Core) NiloToonAllInOneRendererFeature: URPShadowDepthBias's default value changed 0.1 -> 0, URPShadowNormalBiasMultiplier's default value changed 0 -> 1, so now the 'receive URP shadow map' section will match URP shadow's shadow result completely by default

Fixed

- (Core) NiloToonCharSelfShadowMapRTPass will have RT memory leak when user edit NiloToon's renderer feature(e.g. dragging the slider), now for Unity2022.3 or lower we converted back to old RenderTargetHandle system as a temp fix, but the memory leak will still exist for Unity2023.1 or higher
---
[0.14.2] - 2023-08-24

Fixed
- (Core) NiloToonEditorShaderStripping.cs: Stop stripping any URP keywords, this will make URP's stripping perform correctly, which produce correct stripping result of NiloToonCharacter shader (e.g. additional light and additional light shadow)
- (InternalCore) NiloToonCharSelfShadowMapRTPass.cs: fix validTRS matrix check bug
- (InternalCore) NiloToonEditor_AssetLabelAssetPostProcessor.cs: remove AssetDatabase.Refresh() when importing asset
---
[0.14.1] - 2023-08-23

Fixed
- (Core) NiloToonEditorPerCharacterRenderControllerCustomEditor: fix a bug that Unity version lower than Unity2021.3.29 using a non exist API FindObjectsByType. Now all Unity2021.3 will revert to use FindObjectsOfType(not FindObjectsByType)

[0.14.0] - 2023-08-23

Version Highlight (English)
This version is the largest update when compared to any previous update.
This version now supports Unity2021.3 to Unity2023.2, and supporting all URP's new features.
Feedback and bug report are appreciated! (email: nilotoon@gmail.com)(discord: kuronekoshaderlab)

[NiloToon now supports new URP features since Unity2021.3LTS]
A1. Point Light Shadows
A2. Deferred Rendering Support (NiloToon shaders will still render as Forward in Deferred Rendering with 8 light per renderer limitation)
A3. Decals
A4. Light Cookies
A5. Reflection Probe Blending
A6. Reflection Probe Box Projection

[NiloToon now supports new URP features since Unity2022.3LTS]
B1. Forward+ Support (Forward+ can render a maximum of 256 lights per camera, instead of Forward's maximum of 8 lights per renderer)
B2. Rendering Layer
B3. Light layer
B4. Decal layer
B5. LOD Crossfade (Only NiloToonEnvironment shader has this feature supported, since it is not very useful to apply LOD Crossfade to characters)
B6. TAA support (alternative of MSAA when MSAA is too slow in high resolution render)
B7. All types of light cookies

[NiloToon added feature]
C1. a new volume - NiloCinematicRimLightVolume, which will turn all additional lights into rim light for NiloToonCharacters materials
C2. a new MonoBehaviour - NiloToonCharacterMainLightOverrider, you can override character's mainlight color and direction, without editing any URP's directional light
C3. NiloToonCharacter material can now be used independently without a NiloToonPerCharacterRenderController script attached to the prefab(but when used without having a NiloToonPerCharacterRenderController to control that material, that NiloToonCharacter material's NiloToon self shadow map and average URP shadow will not be rendered)
C4. NiloToonCharacter material's Classic outline can now be rendered for transparent queue(2501-5000) materials. You can use "Transparent(ZWrite)/Outline" and "Opaque/Outline" surface type preset materials together now, outlines from both type of materials will be displayed correctly and not be blocked by each other anymore
C5. Rewrite NiloToon self shadow map algorithm, it will not show shadow artifact easily now when multiple characters are visible and far to each other.
C6. Auto update NiloToonPerCharacterRenderController's allRenderers, any child transform change will trigger an update in edit mode. The update will consider child NiloToonPerCharacterRenderController also
C7. NiloToonBloom added HSV control, high saturation bloom can produce another art style
C8. Rewrite NiloToonCharacter's surfaceType preset, allow you to setup a material more easily with a better dropmenu
C9. NiloToonCharacter material UI update (upgrade to LWGUI 1.11.1), including a new toolbar and group search mode

[NiloToon important bug fix]
D1. fix all possible crash that we know when building the player(compiling shader variants)

D2. now NiloToon will have 0 GC alloc per frame(unless using "NiloToonRendererRedrawer" or "Terrain Crash Safe Guard" enabled)

版本重點 (中文)
- 這版本(0.14.0)會是所有版本之中改動最大的版本
- 支援Unity2021.3~2023.2
- 歡迎意見和bug report (email: nilotoon@gmail.com)(discord: kuronekoshaderlab)(QQ/Wechat)

[支援所有URP(Unity2021.3)版本的新功能]
A1. Point Light Shadows
A2. Deferred Rendering Support (NiloToon shaders 在 Deferred Rendering 仍然會以 Forward 渲染(有8 light per renderer 的限制))
A3. Decals
A4. Light Cookies
A5. Reflection Probe Blending
A6. Reflection Probe Box Projection

[支援所有URP(Unity2022.3)版本的新功能]
B1. 支援Forward+，可以令角色收到256個additional light
B2. 支援URP rendering layer
B3. 支援light的rendering layer
B4. 支援decal的rendering layer
B5. LOD crossfade(只有NiloToonEnvironment shader有包括支持，因為角色的LOD crossfade似乎不太常用)
B6. 支援TAA (當你不想使用MSAA(例如在4K渲染時太慢)，可以試試TAA)
B7. 支援所有light cookie

[NiloToon重要新功能]
C1. 增加了一個「令additional light變rim light」的volume - NiloCinematicRimLightVolume，理論上角色可以隨便接受大量spotlight/point light但仍會好看不會過亮
C2. 增加「NiloToonCharacterMainLightOverrider」MonoBehaviour，可以使用新的gameobject來改變角色的main light燈光方向和顏色，而不需要改動URP的任何light
C3. NiloToonCharacter material可以獨立使用，不需要依賴nilotoon per character script (但會NiloToon的average shadow如selfshadowmap不會渲染)
C4. 內層Opauqe+Outline物體 和 外層半透明+Outline物體，都能同時出現描邊，不會互相檔住
C5. 重寫了NiloToon self shadow mapping，在多角色而且各自距離比較遠時都不會再出現難看的影子問題
C6. 自動設置NiloToonPerCharacterRenderController的allRenderers，在child transform有任何變動時會在edit mode自動更新，會考慮child NiloToonPerCharacterRenderController
C7. NiloToonBloom 增加 HSV 控制，例如高saturation的bloom可以做出新的bloom風格
C8. 重寫 NiloToonCharacter的surfaceType preset，新的dropmenu使用上會更直觀
C9. NiloToonCharacter material UI 更新 (使用 LWGUI 1.11.1), 增加新的 toolbar 和 search mode

[NiloToon重要修正]
D1. 修正所有build時會令unity crash的問題
D2. 修正所有C# GC，現在是0 GC Alloc per frame(除非使用 "NiloToonRendererRedrawer" 或 "Terrain Crash Safe Guard")

Known Issue
- (Core) NiloToonRendererRedrawer's rendering will not handle lighting correctly in build since it's first release(released in 0.13.1), we are fixing it and we do not recommend user using it right now other than "editor only rendering" purposes.
- (Core) When Deferred rendering + TAA + NiloToon’s character self shadowmap are all enabled, rendering will be wrong(random white flicker). Consider using Forward+ if possible

Breaking change
- (Core) NiloToonPerCharacterRenderController: add autoRefillRenderersMode and autoRefillRenderersLogMode, now NiloToonPerCharacterRenderController will auto refill allRenderers list if any child transform structure changed in editor. And when performing auto refill allRenderers, it will assign renderers correctly to all child NiloToonPerCharacterRenderController(s)'s allRenderers list
- (Core) NiloToonAnimePostProcessVolume: this volume will now be affected by Camera's PostProcessing toggle, since it should be treated as postprocess similar to Bloom/Vignette
- (Core) NiloToonBloomVolume: this volume will now be affected by Camera's PostProcessing toggle, since it should be treated as postprocess similar to URP's Bloom
- (Core) NiloToonStickerAdditive shader: this shader will now receive _GlobalVolumeMulColor from NiloToonCharRenderControlVolume
- (Core) NiloToonSetToonParamPass: EnableSkyboxDrawBeforeOpaque is now default true, it can solve Opaque queue(<2500) semi-transparent draw problems. Disable it if you don't need it can improve performance
- (Core) NiloToon Self shadow mapping: Rewrite depth bias and normal bias, it will match URP16's depth bias and normal bias design.

- (Core) NiloToon Self shadow mapping: Rewrite shadow caster CPU culling, now override main light dir from volume or script will also override NiloToon Self shadow's shadow casting direction
- (Core) NiloToon self shadow mapping: Rewrite shadow caster CPU culling, now shadowRange start from the closest visible character, not from camera, it can help low FOV camera with large distance between camera and character to stil render shadow map correctly
- (Core) NiloToon self shadow mapping: default shadowRange -> 10m
- (Core) NiloToon self shadow mapping: shadowRange clamp to 10m ~ 100m
- (Core) NiloToon self shadow mapping: fadeTotalDistance 2m->1m, same as URP's default default
- (Core) NiloToonCharacter.shader: +_EnableNiloToonSelfShadowMappingDepthBias,+_EnableNiloToonSelfShadowMappingNormalBias (default is false, which will disable all material depth and normal bias that was used in the past)
- (Core) NiloToonCharacter.shader: clamp emission light color mul, light received will not exceed intensity 1
- (Core) NiloToonCharacter.shader: remove min rimlight and specular, now rendering result between main light intensity 0 and 0.0001 is the same
- (Core) NiloToonCharacter.shader: emission, when tint by light, will receive URP shadowmap if allowed
- (Core) NiloToonCharacter.shader: make isFace's mask sampling using fragment shader quality instead of vertex
- (InternalCore)NiloToonPerCharacterRenderController.cs: In C#, we're changing shadowTestIndex and ExternalRenderOverrider from public to internal. This change ensures users won't modify these by accident.

Added
- (Core) Support all versions of Unity starting from Unity2021.3 to Unity 2023.2
- (Core) all shaders: support Forward+ (require Unity2022.2 or higher)
- (Core) all shaders: support RenderingLayer (e.g. Light and Decal 's rendering layer)
- (Core) all shaders: support all light's lightcookie (including directional light's light cookie)
- (Core) all shaders: support Unity2022.2's new depthnormal pass design
- (Core) +NiloToonCharacterMainLightOverrider.cs
- (Core) NiloToonCharacter shader: NiloToonCharacter material can now be used independently without a NiloToonPerCharacterRenderController script attached to the prefab(but when used without having a NiloToonPerCharacterRenderController to control that material, that NiloToon_Character material's self shadow map and average URP shadow will not be rendered)
- (Core) NiloToonCharacter shader: NiloToon_Character material's Classic outline can now be rendered for transparent queue(2501-5000) materials. You can use "Transparent(ZWrite)/Outline" and "Opaque/Outline" surface type preset materials together now, outlines from both type of materials will be displayed correctly and not be blocked by each other anymore.
- (Core) NiloToonCharacter shader: surface type preset + 4 new cases: Transparent(ZWrite)/Outline, Transparent(ZWrite)/No Outline, Transparent(ZWrite)/Outline (Cutout), Transparent(ZWrite)/No Outline(Cutout)
- (Core) NiloToonCharacter shader: +_FaceShadowGradientMapUVIndex,_FaceShadowGradientMaskMapUVIndex
- (Core) NiloToonCharacter shader: improve the auto check of ShouldOutlineUseBakedSmoothNormal() method, added unit vector length check, this will ensure outline is still correct when no smoothed normal baked data inside mesh's uv8
- (Core) NiloToonCharacter shader: +_OutlineBaseZOffset,_UseOutlineZOffsetMaskFromVertexColor,_OutlineZOffsetMaskFromVertexColor
- (Core) NiloToonPerCharacterRenderController: + "Auto refill AllRenderers list" button
- (Core) NiloToonPerCharacterRenderController: + RefillAllRenderers() API
- (Core) NiloToonPerCharacterRenderController: + ZOffset (similar to material's ZOffset, but this is per character)
- (Core) NiloToonPerCharacterRenderController: "Select all Nilo materials of this char" button will include materirals from AttachmentRendererList & NiloToonRendererRedrawerList also
- (Core) NiloToonAllInOneRendererFeature: + toggle "allowRenderNiloToonBloom"(default on), turn it off when you want to disable NiloToonBloom in low quality settings for performance
- (Core) NiloToonAllInOneRendererFeature: add warning console log when depth priming is enabled, since face materials need depth priming = off
- (Core) NiloToon/Bloom: +downscale,maxIterations (matching URP14's bloom design, require Unity2022.3)
- (Core) NiloToonEditorSelectAllMaterialsWithNiloToonShader.cs: + new API -> SelectAllMaterialsWithNiloToonShader() , GetAllNiloToonCharacterShaderMaterials()
- (Core) NiloToonBloomVolume:+ characterBaseColorBrightness,characterBrightness
- (Core) NiloToonDebugWindow: + Show SelfShadow frustum? toggle in debug window
- (Core) NiloToonShadowControlVolume.cs: + charSelfShadowStrength
- (Core) NiloToonCharRenderingControlVolume.cs: + addLightColor and desaturateLightColor
- (Core) NiloToonBloom: +HSV control
- (Core) NiloToonAllInOneRendererFeature: +auto check terrain crash fix (terrainCrashSafeGuard)
- (InternalCore) + Runtime\Utility\NiloToonPlanarReflectionHelper.cs
- (InternalCore) + NiloToonEditorCreateObjectMenu.cs
- (Doc) Improve the document pdf, adding more section about NiloToon 0.14.x's new feature

Changed
- (Core) NiloToonCharacter.shader: changed the default URP shadow depth bias to 1 for non-face also. Having a default 1m depth bias is because usually you do not want character to receive low resolution ugly URP shadow that was casted by self (e.g. hand,hair,hat), but still wants to receive URP shadow that was casted by far environment objects(e.g. tree, building).

- (Core) NiloToonAllInOneRendererFeature: added allowClassicOutlineInPlanarReflection (default false). When this is off, NiloToon will disable classic outline if any camera has "mirror"/"planar"/"reflection" in camera gameobject's name (ignore cases)
- (Core) NiloToonSetToonParamPass.cs: optimize C# GC to 0 by caching C# Reflection calls like GetProperty()
- (Core) NiloToonPerCharacterRenderController: Extra thick outline now works in edit mode
- (Core) stop rendering NiloToonExtraThickToonOutlinePass,NiloToonPrepassBufferRTPass,NiloToonRendererRedrawerPass,NiloToonAverageShadowTestRTPass in preview window
- (Core) NiloZOffsetUtil.hlsl: zoffset improve cam near plane clamp method, will avoid zfight when vertex is pushed beyound camera near plane
- (Core) NiloToonCharacter.shader: _OutlineWidth default 0.7->0.6, in many cases a smaller outline looks better
- (Core) NiloToonCharacter.shader: rename all surface preset to shorter names and put them into a folder structure drop menu
- (Core) Upgrade LWGUI to 1.11.1
- (Core) improve all NiloToon passes's frame debugger naming and foldout structure
- (Core) NiloToon_Character.shader: make _MainLightIgnoreCelShadeForFaceArea default 0, this will make fade and body skin brightness closer in shadow area
- (Core) NiloToon_Character.shader: rename Smoothness-> Smoothness/Roughness, to make it more easier for group search
- (Core) NiloToonEditorPerCharacterRenderControllerCustomEditor.cs: rewrite auto set up character, will improve the chance of setting the correct bone and set up the material correctly by identity material and bone names
- (Core) NiloToon_Character.shader: _MatCapAdditiveMixWithBaseMapColor,_HairStrandSpecularMixWithBaseMapColor default is now 0.5 (50%)
- (Internal)NiloToonEditorShaderStripping.cs: matching URP12.1.10's shader stripping more
- (Internal)NiloToonCharacter_Shared.hlsl: make shader structure matching URP14 LitForwardPass more(Remove viewDirectionWS in Varyings, recalculate in fragment shader)(fog always calculate in fragment shader)
- (InternalCore)For URP14 or later, nilotoon's all render passes and renderer feature are all converted to "RTHandle+RenderingUtils.ReAllocateIfNeeded+Dispose" style, the "RenderTargetHandle" or "RenderTargetIdentifier" style code will remain for Unity2021.3 only. We will need to make a big change again when Unity2023.4LTS's "render graph system + URP" is released
- (InternalCore) change NiloToonSetToonParamPass's render pass event to BeforeRenderingPrePasses
- (InternalCore) NiloToonEnvironment shader: make envi shader matching URP16.0.3 complexlit 100%(keep all NiloToon changes and 2021.3+2022.3 support), now only 1 SubShader will remain instead of 2

Fixed
- (Core) fix -> extra thick outline will not correctly render when "Keep PlayMode mat edit" is enabled
- (Core) NiloToonCharacter_Shared.hlsl: fix a bug about NiloToonCharacterAreaStencilBufferFillPass. The bug will draw StencillBuffer incorrectly on Classic outline area even Classic outline has been disabled
- (Core) NiloToonSetToonParamPass.cs: fix a possible null exception when getting NiloToonRenderingPerformanceControlVolume instance in constructor
- (Core) Runtime\RendererFeatures\NiloToonAllInOneRendererFeature.cs: turn char list to static list, this will solve all build crash
- (Core) Runtime\NiloToonPerCharacterRenderController.cs: enable faceNormalFix only if head bone exists
- (Core) fix -> In Unity2022.3 LTS, NiloToon/Bloom is not working
- (Core) fix -> In Unity2022.3 LTS, when NiloToon/AnimePostProcess is enabled and it's DrawBeforePostProcess = off, that anime postprocess effect will not work when camera is using fxaa
- (InternalCore) Singleton of NiloToonAllInOneRendererFeature.Instance will now set to the last Create() instance renderer's renderer feature
- (InternalCore) Runtime\RendererFeatures\Passes\NiloToonAnimePostProcessPass.cs: make anime postprocess from RenderPassEvent.AfterRendering + 2 to AfterRenderingPostProcessing
- (InternalCore) Editor\NiloToonEditorPerCharacterRenderControllerCustomEditor.cs: replace FindObjectsOfType to FindObjectsByType
- (InternalCore) ShaderLibrary\NiloUtilityHLSL\NiloHSVRGBConvert.hlsl: fix wrong function declare bug
- (InternalCore) remove NiloToon's _ADDITIONAL_LIGHT_SHADOWS stripping for Unity2022.2 or up, due to Forward+ this stripping is not safe to perform
- (InternalCore) ban NiloToon average shadow in WebGL, since the large iteration forloop in shader can't be compiled correctly in WebGL anyway
- (InternalCore) NiloToonEnvironmentShaderGUI.cs remove "MaterialChanged" override, now it use "ValidateMaterial" override

---

[0.13.5] - 2023-06-28

Known Issue
- (Core) NiloToonRendererRedrawer's rendering will not handle lighting correctly on build since it's first release(released in 0.13.1), we are fixing it and we do not recommend user using it right now other than "editor only rendering" purposes.
- (Core) In Unity2022.3 LTS, NiloToon/Bloom is not working
- (Core) In Unity2022.3 LTS, when NiloToon/AnimePostProcess is enabled and it's DrawBeforePostProcess = off, that anime postprocess effect will not work when camera is using fxaa

Added

- (Core) NiloToonBloomVolume.cs: +_NiloToonBloomCharacterAreaBloomEmitMultiplier and more toolip to other settings
- (Core) NiloToonSetToonParamPass.cs: +auto detect any possible planar reflection camera by name and handle these camera rendering using NiloToonPlanarReflectionHelper (for support PIDI: planar reflection 5 automatically)
- (Core) NiloToon_Character.shader: +_BumpMapApplytoFaces,_OcclusionMapApplytoFaces, _EnvironmentReflectionApplytoFaces

Changed
- (Core) NiloToonBloomVolume.cs: NiloToonBloom IsActive() logic is more strict for better optimization chance
- (Core) NiloToonBloomVolume.cs: default height of renderTextureOverridedToFixedHeight changed from 540->1080
- (Core) NiloToon_Character.shader: rename Mat Cap (occlusion)'s display name to a more intuitive name "Mat Cap (shadow)"

Fixed
- (Core) NiloToonPrepassBufferRTPass.cs: fix prepass RT depth and MSAA not match to cameraTargetDescriptor bug, this bug fix will make NiloToon/Bloom render 100% correct now.
- (Core) NiloToonCharacter_Shared.hlsl: fix an important bug of typo "NiloToonPrepassBuffer", the correct one is "NiloToonPrepassBufferPass", this will solve many NiloToon/Bloom artifact
---
[0.13.4] - 2023-06-26

Known Issue
- (Core) NiloToonRendererRedrawer's rendering will not handle lighting correctly on build since it's first release(released in 0.13.1), we are fixing it and we do not recommend user using it right now other than "editor only rendering" purposes.
- (Core) In Unity2022.3 LTS, NiloToon/Bloom is not working
- (Core) In Unity2022.3 LTS, when NiloToon/AnimePostProcess is enabled and it's DrawBeforePostProcess = off, that anime postprocess effect will not work when camera is using fxaa

Breaking Change
- (Core) NiloToonEditorPerCharacterRenderControllerCustomEditor.cs: Rewrite "Auto setup this character" button, it will now convert MToon10(VRM1 only)/UniUnlit/Lit material properties to NiloToon_Character material. If you need to convert VRM1 generated MToon10 materials to NiloToon shader, the "Auto setup this character" button is the best option now
- (Core) NiloToonCharacter_RenderFace_LWGUI_ShaderPropertyPreset: edit element order, new order doesn't not match to old material properties

Added
- (Core) NiloToonCharacter_SurfaceType_LWGUI_ShaderPropertyPreset: +CutoutTransparent(ZWrite)+Outline,CutoutTransparent,CutoutTransparent(ZWrite) surface type preset
- (Core) NiloToonEditorDebugWindow.cs: +more debug case (to a total of 32 cases)
- (Core) NiloToonCharacter.shader: +_CullNiloToonSelfShadowCaster
- (Core) NiloToonCharacter.shader: +_MultiplyLightColorToEmissionColor,+_EmissionMaskMap
- (Core) NiloToonCharacter.shader: _OutlineWidthExtraMultiplier slider range increase to 2048 for accepting properties from MToon10 shader

Changed
- (Core) LWGUI: upgraded 1.5.3 -> 1.7.0

Fixed
- (Core) NiloToonCharSelfShadowMapRTPass.cs: (VERY Important bug fix)fix nilotoon shadowmap flicker when more than 1 camera is active(including game+scene window's camera)
- (Core) NiloToonCharacter.shader: fix _StencilRef Int Slider
- (Core) NiloToonEditorPerCharacterRenderControllerCustomEditor.cs: fix a null reference exception related to headBoneTransform
- (Core) NiloToonCharacter.shader: fix debug case's alpha output
---
[0.13.3] - 2023-06-08

Known Issue
- (Core) NiloToonRendererRedrawer's rendering will not handle lighting correctly on build since it's first release(released in 0.13.1), we are fixing it and we do not recommend user using it right now other than "editor only rendering" purposes.
- (Core) In Unity2022.3 LTS, NiloToon/Bloom is not working
- (Core) In Unity2022.3 LTS, when NiloToon/AnimePostProcess is enabled and it's DrawBeforePostProcess = off, that anime postprocess effect will not work when camera is using fxaa
- (Core) NiloToonCharacter.shader: _StencilRef's IntRange is not active correctly, it is now a float slider instead of int slider

Breaking Change

- (Core) NiloToonCharacter.shader: +_DepthTexRimLightAndShadowReduceWidthWhenCameraIsClose, this will reduce the depth tex rimlight&shadow's width when camera distance is < 1, inorder to keep the width not too big when view closely. If you want the old result, you can edit this back to 0

Added
- (Core) added Textures\Matcap_Rainbow.png
- (Core) NiloToonCharacter.shader: +_AllowPerCharacterDissolve,_AllowPerCharacterDitherFadeout
- (Core) NiloToonCharacter.shader: +_CullOutline, _RenderFacePreset, Render Face group
- (Core) NiloToonCharacter.shader: +_DepthTexRimLightBlockByShadow, _DepthTexRimLightAndShadowWidthExtraMultiplier
- (Core) NiloToonPerCharacterRenderController.cs: +extraThickOutlineRendersVisibleArea
- (Core) NiloToonAllInOneRendererFeature.cs: +EnableSkyboxDrawBeforeOpaque, for solving semi-transparent material + Skybox flicker bug
- (Core) NiloToonEditorPerCharacterRenderControllerCustomEditor: auto disable depth tex shadow if it is an eye white material

Changed
- (Core) LWGUI: upgrade from 1.4.3 to 1.5.3
- (Core) NiloToonCharacter.shader: better matcap preset grouping
- (Core) NiloToonCharacter.shader: increase _FaceAreaCameraDepthTextureZWriteOffset's default value 0.03->0.04
- (Core) NiloToonCharacter.shader: reduce _DepthTexRimLightAndShadowWidthMultiplier's default value 0.7->0.6
- (Core) NiloToonCharacter.shader: move Render states settings into Collapsible Group (Color Buffer, Depth Buffer, Stencil, Render Face)
- (Core) NiloToonPerCharacterRenderController.cs: disable a few warning spam
- (Core) NiloToonPerCharacterRenderController.cs: fixFaceNormalUseFlattenOrProxySphereMethod default value 0->0.75f
- (Core) NiloToonEditorPerCharacterRenderControllerCustomEditor: Auto setup char button only change _SurfaceTypePreset if _SurfaceTypePreset is the default preset, to avoid editing "Already edited" materials
---
[0.13.2] - 2023-05-07

Known Issue
- (Core) NiloToonRendererRedrawer's rendering will not handle lighting correctly since it's inititial release(released in 0.13.1), we are fixing it and we do not recommend any user using it right now.

Breaking Change
- (Core) NiloToonCharacter_LightingEquation.hlsl: update the algorithm of calculating the width of depth texture rim light and shadow, it is different to the old algorithm, but it will produce a more "character relative" consistant width between different camera distance and fov.

Added
- (Core) NiloToonCharacter.shader: add 3 new SurfaceType preset: Transparent(ZWrite), CutoutOpaque+Outline, CutoutOpaque
- (Core) NiloToonCharacter.shader: add _MatCapOcclusionPreset, _MatCapAdditivePreset, _MatCapAdditiveExtractBrightArea
- (Core) Added a new Textures\Matcap_Glossy01.png texture

Changed
- (Core) NiloToonRendererRedrawerPass.cs: remove a useless ExecuteCommandBuffer() call
- (Core) NiloToonPrepassBufferRTPass.cs: Now this Prepass will show correctly as a group in Frame Debugger, due to an added ExecuteCommandBuffer() call
- (Core) NiloToonAllInOneRendererFeature.cs: optimize the renderPassEvent of all pass
- (Core) NiloToonEditorPerCharacterRenderControllerCustomEditor.cs: added "oral", "brow", "mayu", "express" to IsFaceTargetNames and IsNoOutlineNames array. So material that contains these keywords in their name, will enable "IsFace?" and disable "Classic Outline" when user click the "Auto setup this character" button
- (Core) NiloToonCharacter.shader: Update properies default value. _OutlineWidth 0.8->0.7, _ApplyDepthTexRimLightFixDottedLineArtifacts 0 -> 1, _DepthTexRimLightMixWithBaseMapColor 0 -> 0.5, _DepthTexRimLightIntensity 1 -> 1.5
- (Core) NiloToonCharacter_Shared.hlsl: improve matcap uv using bgolus's method
- (Core) In all NiloToon's .hlsl, turn all unity_OrthoParams.w to !IsPerspectiveProjection() or similar code, to improve readability

Fixed
- (Core) NiloToonEnvironment.shader: update NiloToonEnvironment.shader to match URP12.1.10, to display shadow mask baked lightmap correctly
- (Core) NiloToonCharacter.shader: CharacterAreaStencilBufferFill pass now defines NiloToonCharacterAreaStencilBufferFillPass instead of NiloToonSelfOutlinePass. This will help NiloToonCharacter_Shared.hlsl to solve a bug about "ExtraThick outline pass not correctly render"
- (Core) NiloToonCharacter_Shared.hlsl: fix "ExtraThick outline pass not correctly render" when Classic Outline is off
---
[0.13.1] - 2023-04-23

Breaking Change
- (Core) NiloToonPerCharacterRenderController.cs: remove extraThickOutlineStencilID, now extra thick outline only relies on the 1st bit of the stencil buffer. The 2nd-8th bit can be controlled by NiloToonCharacter.shader material's new _StencilRef,_StencilComp,_StencilPass settings freely.
- (Core) NiloToonPerCharacterRenderController.cs: now "Extra Thick Outline" will draw only the outline area when "Show Blocked" is enabled
- (Core) NiloToonCharacter_ExtendFunctionsForUserCustomLogic.hlsl: fix typo attrubute -> attribute

Added
- (Core) Added NiloToonRendererRedrawer.cs, this script can be attached to any NiloToonPerCharacterRenderController's child renderer, to redraw that renderer's target SubMesh again. Usually for drawing semi-transparent eye or hair again, using Stencil setting of the material.
- (Core) Added NiloToonRendererRedrawerPass.cs, this is the pass for drawing all NiloToonRendererRedrawer component that is enabled in scene
- (Core) NiloToonAllInOneRendererFeature.cs: enqueue NiloToonRendererRedrawerPass
- (Core) NiloToonPerCharacterRenderController.cs: added nilotoonRendererRedrawerList, to update the materials of every child NiloToonRendererRedrawer
- (Core) NiloToonCharacter.shader: add _StencilRef,_StencilComp,_StencilPass. It is usually for drawing eye on top of hair. User can control 2nd~8th bit of Stencil Read/Write freely per material, while the 1st bit is always reserved for NiloToon's ExtraThickOutline.
- (Core) NiloToonEditorPerCharacterRenderControllerCustomEditor.cs: More auto setting by material name. Now if material name contains "lash", then "Is Face?" will be turned on. If material name contains "body", then "Is Skin?" will be turned on. Now if material name contains "facial"/"eye"/"iris"/"pupil"/"mouth"/"teeth"/"tongue"/"lash", then "Classic Outline" will turned off, and "Surface Type Preset" will be set to "Opaque"
- (Core) add PureBlack4x4.png
- (Core) NiloToonCharacter.shader: add a "Require Depth Priming Mode equals Disabled" note to "Is Face?" section

Changed
- (Core) NiloToonEnvironment.shader: now SRP batching is working. Every pass will share the same CBUFFER_START(UnityPerMaterial).
- (Core) NiloToonEnvironment.shader: upgrade to use URP12.1.9 ComplexLit shader as base code, so ClearCoat can be used now, and you can expect the render result will be the same as URP12.1.9's ComplexLit shader but with all NiloToon functions applied to it.
- (Doc) rewrite "How to make eyebrows render on top of hair(in a semi transparent way)?" section, to use new feature(NiloToonRendererRedrawer) of this update

Fixed
- (Core) NiloToonCharacter.shader: fix a bug that uv4 is not using the 4th uv(TEXCOORD3), in the past it is using the 5th uv(TEXCOORD4)
- (Core) FoldoutEditor.cs: MeshFilter is not showing correctly due to this class, so we changed FoldoutEditor.cs to not affect classes other than 'NiloToonPerCharacterRendeController' now
- (Core) NiloToonPerCharacterRenderController.cs: fix Extra Thick Outline's ZOffset is not work bug
- (Core) NiloToonCharSelfShadowMapRTPass.cs: optimize NiloToonCharSelfShadowMapRTPass memory allocation, by calling to a non-allocate version of GeometryUtility.CalculateFrustumPlanes()

[0.12.3] - 2023-03-29

Known Issue
- (Core) NiloToonEnvironment.shader: will show error log -> "Instancing: Property 'unity_RenderingLayer' shares the same constant buffer offset with 'unity_LODFade'. Ignoring."

Changed
- (Demo) Upgrade demo project to Unity2021.3.20f1, remove demo project support for Unity2020.3LTS
- (Demo) CRS SampleScene URP (NiloToon control).asset: change charBaseColorMultiply from 1->0.95 and charBaseColorTintColor to a redish value
- (Demo) NiloToonSampleSceneProfile.asset: change charBaseColorMultiply from 1->0.95
- (Demo) UniversalRP-5HighestQuality.asset: enable m_ReflectionProbeBlending, m_ReflectionProbeBoxProjection, m_SupportsLightLayers
- (Demo) UniversalRP-5HighestQuality.asset: change m_MSAA from 8 to 4, to avoid performance problem in editor
- (Demo) UniversalRP-5HighestQuality.asset: enable m_RequireOpaqueTexture
- (Demo) ForwardRenderer 5HighestQuality.asset: remove an unused renderer feature "GTToneMapping"
- (Demo) ForwardRenderer 5HighestQuality.asset: adding a renderer feature "Decal", enabled "dbuffer" technique
- (Demo) pidi reflection upgrade from 4.0.7 to 4.2.0
- (Demo) Magica Cloth: replace NativeMultiHashMap to NativeParallelMultiHashMap
- (Demo) UnityPackageManager(UPM): remove jobs, install collections

- (Demo) ProjectSettings\ProjectSettings.asset: m_BuildTargetDefaultTextureCompressionFormat now use ASTC compression

Fixed
- (Core) Upgrade LWGUI to 1.4.3
- (Demo) fix advancedFPSCounter font bug for Unity2022.2
- (Demo) NiloToonDemoSampleScene_UI.cs: fix UI typo enviroment -> environment
---
[0.12.2] - 2023-03-19

Fixed
- (Core) temp fix a bug that LWGUI1.4.2 failed to show tooltip and helpbox inside a group.

[0.12.1] - 2023-03-19

Breaking change
- (Core) Since Unity will stop Unity2020.3 LTS support soon, NiloToon 0.12.1 removed the support of Unity2020.3, and merged URP12's support branch to NiloToon's main branch. Now Unity2021.3 is the minimum and recommended Unity version for NiloToon. If after upgrading to 0.12.1, you encountered Unity editor crash when building the game, please close Unity -> delete library folder -> restart unity to rebuild the shader cache.
- (Core) Removed NiloToonShaderPatch(for URP12 or newer).unitypackage. If you still see this package in your project, please delete it.
- (Core) NiloToonPerCharacterRenderController: run desaturate before tint color and add color, since this will allow more color combination results.

Added
- (Core) NiloToonUPMImporterCollections.cs: Added this [InitializeOnLoad] editor script to auto install required UPM package when user imported NiloToonURP's .unitypackage to a project (e.g. install Collections package), user don't need to open UPM and install anything manually now.
- (Core) NiloToonPerCharacterRenderController: added Dissolve Mode, noise TilingX&Y and noise Strength. This will allow user to create different style of dissolve.
- (Core) NiloToonEditorPerCharacterRenderControllerCustomEditor.cs: added "facial", "iris", "pupil", "tongue" to IsFaceTargetNames, material with these name will be auto processed when user clicked "Auto setup this character" button
- (Core) NiloToonCharacter.shader: added _FaceShadowGradientMapFaceMidPoint, this will allow FaceShadowGradientMap to support any non-horizontally mirrored texture. When using any non-horizontally mirrored texture user need to set _FaceShadowGradientMapFaceMidPoint to the mid point of their _FaceShadowGradientMap texture
- (Core) NiloToonCharacter.shader: "Environment Reflection" added Reflection probe blending & reflection probe box projection support
- (Core) Added Textures\Matcap_Metal4Gold.png

Changed
- (Core) Assembly definitions: Removed the requirement of Jobs package, now NiloToon only rely on Collections package
- (Core) NiloToonCharacter.shader: added _fragment & _vertex suffix to shader_feature & multi_compile, this will reduce build time, build size, and build's runtime shader memory usage, but user may need to delete their Library folder to rebuild the shader-cache, else crash may happen
- (Core) NiloToonCharacter.shader: edit the display name of _EnvironmentReflectionUsage from Intensity to Strength
- (Core) NiloToonCharacter.shader: DepthNormalsOnly pass added normal related shader_feature
- (Core) NiloToonCharacter.shader: re-enable _NILOTOON_DITHER_FADEOUT for DepthOnly and DepthNormalsOnly pass
- (Core) NiloToonCharacter.shader: enable _NILOTOON_DITHER_FADEOUT for prepass, prepass remove shader_feature _NORMALMAP
- (Core) NiloToonCharacter.shader: changed _OutlineTintColorSkinAreaOverride default color from (0.623,0.238,0.289,0) -> (0.4,0.2,0.2,1), which is enabled by default now
- (Core) NiloToonCharacter.shader: changed _IgnoreDefaultMainLightFaceShadow default value from 0 -> 1
- (Core) NiloToonCharacter.shader: removed _REFLECTION_PROBE_BLENDING/_REFLECTION_PROBE_BOX_PROJECTION for Outline pass, to reduce shader variant count
- (Core) NiloToonCharacter.shader: outline pass + _BASEMAP_STACKING_LAYER1-10, _MATCAP_BLEND shader_feature, to make outline pass's color matching lit pass
- (Core) NiloToonCharacter.shader: changed _FaceAreaCameraDepthTextureZWriteOffset default value from 0.04 -> 0.03, this will produce better Classic Outline for face
- (Core) NiloToonCharacter.shader: Adding separation line to shader GUI
- (Core) NiloToonCharacter.shader: improved shader GUI
- (Core) NiloToonCharacter.shader: _OutlineWidth's default value change from 1 -> 0.8, since most of the time a thinner outline is preferred
- (Core) NiloToonAllInOneRendererFeature.cs: edit prepass timing from AfterRenderingPrePasses to BeforeRenderingOpaques. since character shader's NiloToonPrepassBuffer pass needs _CameraDepthTexture. (AfterRenderingPrePasses is still too early because _CameraDepthTexture is not ready at this timing)

- (Core) NiloToonCharSelfShadowMapRTPass.cs & NiloToonShadowControlVolume.cs: useMainLightAsCastShadowDirection's default value changed from false->true, since many user expect it to be true by default.
- (InternalCore) remove all UNITY_2021_3_OR_NEWER macro in C#, since the minimum Unity version is 2021.3 already
- (InternalCore) remove all !UNITY_2021_3_OR_NEWER or below's macro and content in C#, since the minimum Unity version is 2021.3 already
- (InternalCore) NiloToonCharacter.shader: removed all URP10(Unity2020.3) support code
- (InternalCore) NiloToonCharacter.shader: In shader, convert all SHADER_LIBRARY_VERSION_MAJOR to UNITY_VERSION, since SHADER_LIBRARY_VERSION_MAJOR is deprecated for Unity2022.2 or later
- (InternalCore) NiloToonCharacter_Shared.hlsl: replace raw type to FRONT_FACE_TYPE / FRONT_FACE_SEMANTIC / IS_FRONT_VFACE(), this will let Unity handle the per platform conversion
- (InternalCore) NiloToonCharacter.shader: change "UniversalMaterialType" = "Lit" -> "ComplexLit"
- (InternalCore) Replace all UnsafeHashMap -> UnsafeParallelHashMap
- (InternalCore) Upgrade LWGUI to 1.4.2
- (Doc) improved NiloToonURP user document.pdf, adding jump links, removed old data and replace all pictures to an updated version.

Fixed
- (Core) NiloToonCharacter.shader: NiloToonPrepassBuffer now follow depth pass outline extrude logic, so NiloToon/Bloom's character prepass mask will have better accuracy
- (Core) NiloToonCharacter.shader: NiloToonIsAnyOutlinePass don't flip normal on back face anymore, this fix will make outline color matching the original surface color
- (Core) NiloToonCharacter.shader: _IgnoreDefaultMainLightFaceShadow is now only effective when _FACE_SHADOW_GRADIENTMAP is enabled
- (Core) NiloToonCharacter.shader: only enableDepthTextureRimLightAndShadow when _DitherFadeoutAmount == 0, since rendering enableDepthTextureRimLightAndShadow when character has dither holes will not produce good result
- (Core) NiloToonCharacter.shader: fix typo of _SkinMaskMapRemapStart, _SkinMaskMapRemapEnd, this function is not working for a long time, now it is working correctly.
- (Core) NiloToonToonOutlinePass.cs: fix outline pass flickering when mouse move across game view and material inspector
- (Core) NiloToonEditor_AssetLabelAssetPostProcessor.cs: fix multiple renderer with same name in same hierarchy = can't generate uv8 bug
- (InternalCore) NiloToonCharacter.shader: rename reflectionVectorVS to reflectionVectorWS, it is a typo.

[0.11.9] - 2023-02-01

Changed
- (Core) NiloToonCharacter.shader: _OverrideShadowColorTexTintColor + [HDR]
- (Core) VideoPlayerForceSyncWithRecorder.cs: update this utility script to the latest version
- (Doc) Improve document pdf
- (Demo) Update demo project to the latest version, adding more models, edit model materials using NiloToon 0.11.9. This maybe the last demo project version to support Unity2020.3. In the future only Unity2021.3 or above's demo project will be provided.

[0.11.8] - 2023-01-19

Fixed
- (Core) NiloToonUberPost.shader: compile support for Unity 2022.2(URP14), this version can now build correctly without compile error.
- (Doc) Improve document pdf

[0.11.7] - 2023-01-19

Fixed
- (Core) NiloToonCharacter_Shared.hlsl: compile support for Unity 2022.2(URP14)
- (Core) NiloToonAverageShadowTestRT.shader: compile support for Unity 2022.2(URP14)
- (Core) NiloToonBloom.shader: compile support for Unity 2022.2(URP14). Although the shader compiles, NiloToon's Bloom will not show in the game window in Unity2022.2.

[0.11.6] - 2023-01-19

Added
- (Core) NiloToonCharacter.shader: added sample UV channel and apply UV channel option to "Parallax Map"

Changed
- (InternalCore) Upgrade LWGUI to 1.4.0

Fixed

- (Core) NiloToonCharacter.shader: fix a bug that "Parallax code" can not compile on iOS platform / iOS Editor
---
[0.11.5] - 2022-12-28

Added
- (Core) NiloToonCharacter.shader: added additional light (point/spot)'s light layer support for URP12 or above

Changed
- (Core) NiloToonCharSelfShadowMapRTPass.cs: NiloToon's shadowmap default size changed from 2048->4096, max limit changed from 8192->16384
- (Core) NiloToonShadowControlVolume.cs: NiloToon's shadowmap default size changed from 2048->4096, max limit changed from 8192->16384

[0.11.4] - 2022-12-21

Added
- (Core) NiloToonCharacter.shader: added face shadow gradient map preset, _FaceShadowGradientMapUVCenterPivotScalePos
- (Core) NiloToonCharacter.shader: matcap additive added render face option
- (Core) NiloToonCharacter.shader: specular highlight added render face option
- (Core) NiloToonCharRenderingControlVolume.cs: + _GlobalAdditionalLightRimMaskSoftness

Fixed
- (Core) NiloToonCharacter.shader: temp disable soft shadow due to filter bug in 2021.3 or later
---
[0.11.3] - 2022-12-20

Breaking change
- (Core) NiloToonCharacter.shader: Now "Override Shadow Color" feature will accept BaseMap tint color from material, NiloToonPerCharacterRenderController and NiloToonCharRenderingControlVolume. By design the "Override Shadow Color" should not be affected by any tint color, but most of the NiloToon user will assume "Override Shadow Color" should be affected by all tint color, so we make this breaking change now to meet user's assumption. If anyone wants the old behavior back, tell us and we will add a toggle for that.

Changed
- (Core) NiloToonCharacter.shader: rewrite all material UI (except BaseMap stacking layer 4-10 are not yet finished), now all UI have shorter name, and some of them have tooltip also.

Added
- (Core) NiloToonCharacter.shader: some textures can select UV0~UV3 now.
- (Core) NiloToonCharacter.shader: add _BaseMapStackingLayer1TexIgnoreAlpha,_BaseMapStackingLayer1ApplytoFaces. (also for layer2,3)
- (Core) NiloToonPerCharacterRenderController.cs: add extraThickOutlineWriteIntoDepthTexture
---
[0.11.2] - 2022-12-06

Added
- (Core) added FoldoutAttribute.cs and FoldoutEditor.cs
- (Core) NiloToonPerCharacterRenderController.cs: now use FoldoutEditor, and added Foldout group to inspector UI. This will change the inspector UI greatly.
- (Doc) adding table of content in document
- (Doc) adding preview and download link in document

Changed
- (Core) NiloToonPerCharacterRenderController.cs: now 'Classic Outline' will always have effect in play mode or edit mode. In the past it requires in play mode.

[0.11.1] - 2022-12-05

Breaking change
- (Core) NiloToonPerCharacterRenderController.cs & NiloToonEditorPerCharacterRenderControllerCustomEditor.cs: completely rewrite the inspector UI, now use short names and adding helpbox and tooltip to explain what each setting does. Our goal is to make it easy to understand and use without the need to read a lengthy document.
- (Core) LWGUI: update to 1.3.0

- (Core) NiloToonCharacter.shader: rewriting the inspector UI (still WIP), now use short names and adding helpbox and tooltip to explain what each setting does. Part of the rewrite UI are now released in this version to gather feedback and bug report. Our goal is to make it easy to understand and use without the need to read a lengthy document.
- (Core) Completely removed Unity 2019.4 support, now the oldest Unity version that can run NiloToon is Unity 2020.3LTS
- (Core) NiloToonCharacter_Shared.hlsl: prepass pass now use a dynamic bias in fragment shader based on camera far plane value. (when camera far plane is higher than 1000, we want to increase the bias, else we will return wrong pixels as black, same as any shadow map artifact)
- (Core) NiloToonCharSelfShadowMapRTPass.cs: disable scene view nilotoon shadowmap pass's render if UseMainLightAsCastShadowDirection is false.
- (Core) NiloToonCharSelfShadowMapRTPass.cs: When rendererfeature's enableCharSelfShadow is false, no matter what override value on volume is, nilotoon shadowmap pass will not render. So user can control performance using renderer feature more easier.
- (Core) NiloToonAverageShadowTestRTPass.cs: When rendererfeature's enableAverageShadow is false, no matter what override value on volume is, nilotoon average shadow will not render. So user can control performance using renderer feature more easier.
- (Core) NiloToonCharacter.shader: change HSV's Saturation apply method, now extremely low saturation pixels will still be low saturation after HSV edit, it will avoid random hue pixels appears due to texture compression (e.g. ETC/ASTC)
- (Core) NiloToonCharacter.shader: _LowSaturationFallbackColor now default off(alpha = 0), because it is not needed after we change HSV 's S apply method

Added
- (Core) NiloToonCharacter.shader: +SurfaceType preset, now you can switch between Opaque+Outline/Opaque/Transparent+Outline/Transparent presets easily
- (Core) NiloToonCharacter.shader: +Dynamic Ramp Map, you can edit ramp map (lighting)'s ramp map inside material inspector in real-time
- (Core) NiloToonCharacter.shader: +_SkinMaskMapInvertColor
- (Core) NiloToonCharacter.shader: +_OutlineZOffsetMaskTexChannelMask, +_OutlineZOffsetMaskTexInvertColor
- (Core) NiloToonCharacter.shader: + auto ShouldOutlineUseBakedSmoothNormal() check in vertex shader, now .vrm character will auto NOT use baked smoothed outline automatically
- (Core) NiloToonCharacter.shader: +_OcclusionMapInvertColor, _OcclusionMapStylePreset
- (Core) NiloToonCharacter.shader: +_DepthTexShadowColorStyleForNonFacePreset, _DepthTexShadowColorStyleForFacePreset
- (Core) NiloToonCharacter.shader: +_ZTest
- (Core) NiloToonCharacter.shader: _ColorMask added A and None options
- (Core) added VideoPlayerForceSyncWithRecorder.cs and DoFAutoFocusTransform.cs
- (Core) NiloToonAllInOneRendererFeature.cs: added [DisallowMultipleRendererFeature]
- (core) NiloToonScreenSpaceOutlinePass.cs: +AllowRenderScreenSpaceOutlineInSceneView toggle
- (Core) NiloToonPerCharacterRenderController.cs: +useCustomCharacterBoundCenter toggle
- (Core) NiloToonPerCharacterRenderController.cs: +extraThickOutlineZWriteMode, to solve extra thick outline ZWrite problem
- (Core) NiloToonToonOutlinePass.cs: +extraThickOutlineRenderTiming

Change
- (Core) improve NiloToon's VolumeComponentMenu, now you can search all NiloToon's volume by typing "Nilo"
- (Core) NiloToonCharacter.shader: _DepthTexRimLightAndShadowWidthMultiplier default value 1->0.7, the default width is too large in the past
- (Core) NiloToonCharacter.shader: URP shadow depth bias(face) default 0.2->1
- (Core) Don't auto add NiloToon tag for .vrm and .asset mesh files anymore, which reduced reimport time for .vrm characters by a lot.

Fixed
- (Core) convert all VFACE to SV_IsFrontFace in shader, to make DX12 shader compile works correctly for XBox One GameCore
- (Core) NiloToonCharSelfShadowMapRTPass.cs: fixed a NiloToon shadowmap flicker bug when mouse is moving into unity editor's material inspector
- (Core) NiloToonScreenSpaceOutlinePass.cs.cs: fixed a NiloToon Screen space outline flicker bug when mouse is moving into unity editor's material inspector
- (Core) NiloToon.NiloToonURP.Runtime.asmdef now includes any platform
- (Core) NiloToonAnimePostProcessPass.cs: anime postprocess will skip drawing if shader is yet not ready, it will remove an error log after 'reimport all'.
- (Core) NiloToonEditorShaderStripping.cs: In Unity2021.3's build, NiloToon will incorrectly strip some keyword(e.g. _MATCAP_ADD _MATCAP_BLEND _RAMP_LIGHTING_SAMPLE_UVY_TEX _SKIN_MASK_ON _SPECULARHIGHLIGHTS), we now disable some stripping logic for "Unity 2021.3 or later" temporary until bug is fixed.
- (Core) NiloToonPerCharacterRenderController.cs: fix a bug that bounding sphere center pos per frame's calculation is wrong

---

[0.10.18] - 2022-10-24

Fixed
- (Core) NiloToonCharSelfShadowMapRTPass.cs: skip execute if it is processing a preview camera. Skip preview camera can solve a bug that cause shadowmap flicker when mouse is moving into unity editor's material inspector region
---
[0.10.17] - 2022-10-24

Breaking change
- (Core) NiloToonBloomVolume.cs: rename VolumeComponentMenu from "NiloToon/Bloom" to "NiloToon/Bloom(NiloToon)", to avoid having the same name as URP's Bloom inside volume's menu
- (Core) NiloToonCharSelfShadowMapRTPass.cs: add normalBias for nilotoon's self shadowmapping, with default value = 1, it is enabled by default, which will change the result of nilotoon's self shadowmapping. Keep depthBias and normalBias to 1 if you are not sure about the correct value
- (Core) NiloToonCharacter.shader: rewrite depthbias implementation method, added normalbias(enabled by default).

Added
- (Core) NiloToonCharSelfShadowMapRTPass.cs: add normalBias for nilotoon's self shadowmapping, with default value = 1
- (Core) NiloToonShadowControlVolume.cs: add normalBias, you can use it if you want to override it in volume
- (Core) NiloToonCharacter.shader: Expose NormalBias offset in nilotoon char material

changed
- (Core) NiloToonCharacter.shader: improve "Can Receive NiloToon ShadowMap?" material GUI's naming,tooltip,helpbox

Fixed
- (Core) NiloToonCharacter.shader: fix Unity2021.3(URP12) additional light shadowmap's sampling bug, now nilotoon's character shader should receive point light/spot light shadowmap similar to the result of URP's Lit shader shadowmap sampling
---
[0.10.16] - 2022-09-30

Added
- (Core) NiloToonPerCharacterRenderController.cs: added ditherNormalScaleFix, used to hide SSAO problem when dither opacity is close to 0. Default is 1(enabled)

changed
- (Core) LWGUI: upgrade to 1.2.5 (added search bar, tooltip, property default value)
- (Core) NiloToonEnvironment.shader: support more URP12 keywords (e.g. Decal)

Fixed
- (Core) NiloToonEnvironmentShaderGUI.cs: fix Unity2021.3 or above nilotoon environment shader GUI not always update bug
---
[0.10.15] - 2022-09-17

Added
- (Core) NiloToonAnimePostProcessVolume.cs: add rotation,topLightExtraRotation,bottomDarkenExtraRotation
- (Core) NiloToonCharRenderingControlVolume.cs: add additionalLightIntensityMultiplierForFaceArea,additionalLightIntensityMultiplierForOutlineArea,additionalLightApplyRimMask,additionalLightRimMaskPower

changed
- (Core) LWGUI: upgrade to 1.1.2

Fixed
- (Core) NiloToonCharacter.shader: optimize additional light code, now additional light code will only enable if keyword _ADDITIONAL_LIGHTS || _ADDITIONAL_LIGHTS_VERTEX is enabled
---
[0.10.14] - 2022-09-07

Fixed
- (Core) NiloToonCharacter.shader: fix _DBUFFER's tempMetallic typo in code, which makes decal section failed to compile
- (Core) NiloToonCharacter.shader: try to solve Unity2022.2 NiloToon self shadow (soft shadow) compile error
- (Core) NiloToonCharacter.shader: fix _DepthTexRimLightUsage & _DepthTexRimLightIntensity can't completely remove rim light when set to 0

[0.10.13] - 2022-09-04

Breaking Changes

- (Core) Replace all old LWGUI related script to the latest LWGUI 1.0.1, you MUST delete NiloToonURP/Editor/ShaderGUI folder before upgrade to NiloToonURP 0.10.13, or just delete NiloToonURP folder completely before import NiloToonURP 0.10.13
- (Core) NiloToonAnimePostProcess.shader: use ColorMask RGB instead of default RGBA, to keep RT's alpha

Added
- (Core) NiloToonCharacter.shader: Due to upgrade to LWGUI 1.0.1, now material property and group has a "reset/revert to default value" button at the right side of each property or group, it can be used to show all the edited values too

Changed
- (Core) NiloToonCharacter.shader: Due to upgrade to LWGUI 1.0.1, merge all texture channelMask option into the same line of that texture using [Tex(,)]
- (Core) NiloToonCharacter.shader: Due to upgrade to LWGUI 1.0.1, remove (Default XXX) in group name
- (Core) NiloToonCharacter.shader: Due to upgrade to LWGUI 1.0.1, rename "Calculate Shadow Color of BaseMap" to "Shadow Color"
- (InternalCore) NiloToonCharacter.shader: Due to upgrade to LWGUI 1.0.1, all [Header()] changed to [Title()]
- (InternalCore) NiloToonCharacter.shader: Due to upgrade to LWGUI 1.0.1, remove all [Space()]
- (InternalCore) NiloToonCharacter.shader: Due to upgrade to LWGUI 1.0.1, turn all [Main(x,_,2)] to [Main(x,_,off,off)]

Fixed
- (Core) NiloToonCharacter.shader: fix Unity2020.3 additional light shadow not appear in build, but this fix will increase NiloToon_Character shader memory usage
---
[0.10.12] - 2022-08-21

Fixed
- (Core) NiloToonSetToonParamPass.cs: fix a bug that is introduced in 0.10.11(refactor code)
- (Demo) vrm prefab will have their own material copy as override in prefab variant, these materials will not be overwrite by univrm after reimport all
- (Doc) added section "After deleting Library folder or Reimport All, all vrm material are reset to mtoon original material(purple in URP), How to stop it?"

[0.10.11] - 2022-08-20

Important! Future NiloToonURP version 0.11.x will use Unity2021.3 as the minimum version
- (About NiloToonURP 0.11.x's Unity2019.4 support) Since Unity has stopped Unity2019.4's support on 01 Jun 2022(https://endoflife.date/unity), now Unity2019.4.40f1 is the last Unity2019 version available, we will remove NiloToonURP 0.11.x's Unity2019.4 support completely. If possible, please upgrade your project to Unity2021.3 before you upgrade to future NiloToonURP 0.11.x versions
- (About NiloToonURP 0.11.x's Unity2020.3 support) Since Unity2021.3LTS has been released for a while already, our testing found that Unity2021.3 is a better version than Unity2020.3, we will soon use Unity2021.3.x as the minimum version for using NiloToonURP in the near future, you can expect NiloToonURP 0.11.x's Unity2020.3 support will be removed also.
- If you must stay on Unity2019.4 or Unity2020.3, you should keep using NiloToonURP 0.10.x and not upgrading to 0.11.x

Known Issue
- (Core) NiloToonEnvironment.shader: this shader is not SRP-batcher compatible since 0.8.20(0.8.19 is safe), we are fixing it
- (Core) For URP12(Unity2021.3) or newer, if depthPriming is enabled, character shader's face will not render correctly, we are fixing it
- (Core) For URP13(Unity2022.1) or newer, if depthPriming is enabled, character will not render correctly, we are fixing it
- (Core) For URP13(Unity2022.1) or newer, NiloToon/Bloom will not render correctly, we are fixing it

Added
- (Core) (Important) NiloToonChar.shader: added support for URP12(Unity2021.3LTS) Point and Spot light cookies, decal, reflection probe blending, reflection probe box projection, correct point light shadow
- (Core) NiloToonCharRenderingControlVolume.cs: added depthTexRimLightDepthDiffThresholdOffset
- (Core) NiloToonCharRenderingControlVolume.cs: added charRimLightMultiplierForOutlineArea && charRimLightTintColorForOutlineArea
- (Demo) Added more testing models
- (Demo) +Assets\ThirdParty(GitHub)
- (Demo) +Assets\ThirdParty(GitHub)\URP_BlitRenderFeature-master
- (Demo) +Assets\ThirdParty(GitHub)\GT-ToneMapping-main
- (Demo) Packages\manifest.json: added .abc support

Changed
- (Core) NiloToonCharRenderingControlVolume.cs: charRimLightCameraDistanceFadeoutStartDistance default value increased from 50 to 1000, because we want it to have no effect by default

- (Core) NiloToonCharRenderingControlVolume.cs: charRimLightCameraDistanceFadeoutEndDistance default value increased from 100 to 2000 because we want it to have no effect by default
- (Core) NiloToonShadowControlVolume.cs: changed depthBias's default value from 1 to 0.5
- (Core) NiloToonCharSelfShadowMapRTPass.cs: changed depthBias's default value from 1 to 0.5
- (InternalCore) NiloToonSetToonParamPass.cs: refactor code
- (Demo) update project from Unity2020.3.33f1 to Unity2020.3.38f1
- (Demo) NiloToonShaderStrippingSettingSO.asset: demo project's default stripping now match mobile setting, to reduce build time(shader compile) and runtime shader memory
- (Demo) update luxiya cloth material
- (Demo) update Klee face material
- (Demo) update Barbara all materials
- (Demo) update MaGirl2 materials
- (Demo) Project Settings/Editor -> EnterPlayModeOption changed from off to on, to allow faster playmode testing in editor
- (Demo) Player Settings -> use il2cpp for build instead of mono, to allow better CPU performance in build
- (Demo) ForwardRenderer 5HighestQuality: add GT-ToneMapping renderer feature(default inactive), for tonemapping style testing
- (Demo) Assets\ThirdParty(VRM): update univrm to 0.99
- (Demo) ForwardRenderer 5HighestQuality: shadow range 10->200
- (Demo) ForwardRenderer 4ExtremeHighQuality: shadow range 10->100
- (Demo) ForwardRenderer 3VHighQuality: shadow range 10->50
- (Demo) ForwardRenderer 2HighQuality: shadow range 10->25

Fixed
- (Core) NiloToonSetToonParamPass.cs: correctly support depth/normal texture request when deferred rendering is enabled. Now if you switch between ForwardRendering and DeferredRendering, NiloToon character shader's rendering result should be very similar
- (Core) NiloToonCharacter.shader: shader stripping will work correctly on URP12 also
- (Core) NiloToonEditor_AssetLabelAssetPostProcessor.cs: increase maxOverlapVertices from 10 to 100, this change will improve uv8's result (usually affect hair's tip outline)
- (Core) NiloToonCharacter_Shared.hlsl: now specular debug will correctly show masked specular area
- (Core) NiloToonCharacter_Shared.hlsl: optimize BaseColorAlphaClipTest_AndNiloToonPrepassBufferColorOutput clip() to return 0

[0.10.10] - 2022-06-28

Known Issue
- (Core) NiloToonEnvironment.shader: this shader is not SRP-batcher compatible since 0.8.20(0.8.19 is safe), we are fixing it
- (Core) For URP12(Unity2021.3) or newer, if depthPriming is enabled, character shader's face will not render correctly, we are fixing it
- (Core) For URP13(Unity2022.1) or newer, if deferred rendering is enabled, character shader will not render correctly, we are fixing it
- (Core) For URP13(Unity2022.1) or newer, NiloToon/Bloom will not render correctly, we are fixing it

Breaking Changes
- (Core) NiloToon/bloom: will now affected by camera's post processing toggle

Added
- (Core) NiloToonCharacter.shader: Added _UseHairStrandSpecularTintMap, _HairStrandSpecularTintMap, _HairStrandSpecularTintMapUsage, _HairStrandSpecularTintMapTilingXyOffsetZw

Changed
- (Core) NiloToonShadowControlVolume.cs & NiloToonCharSelfShadowMapRTPass.cs: increase nilotoon self shadow map default shadowRange from 10 to 100

Fixed
- (Core) NiloToonSetToonParamPass.cs: Make NiloToonURP running in Unity2022.1(URP13) possible

---

[0.10.9] - 2022-06-23

Known Issue
- (Core) NiloToonEnvironment.shader: this shader is not SRP-batcher compatible since 0.8.20(0.8.19 is safe), we are fixing it
- (Core) For URP12 or newer, if depthPriming is enabled, character shader's face will not render correctly, we are fixing it

Fixed
- (Core) NiloToonEditor_AssetLabelAssetPostProcessor.cs: Fixed a critical bug -> "if MeshRenderer/SkinnedMeshRenderer is attached on .fbx GameObject's root, uv8 smooth normal can not be generated"

[0.10.8] - 2022-06-20

Known Issue
- (Core) NiloToonEnvironment.shader: this shader is not SRP-batcher compatible since 0.8.20(0.8.19 is safe), we are fixing it
- (Core) For URP12 or newer, if depthPriming is enabled, character shader's face will not render correctly, we are fixing it

Breaking Changes
- (Core) NiloToonEditorShaderStripping .cs: rewrite shader stripping, now will correctly strip all possible keywords, result in much smaller shader memory if you use NiloToonShaderStrippingSettingSO. If you find that a feature is missing in build after this update, make sure to include that feature in your NiloToonShaderStrippingSettingSO
- (Core) NiloToonCharacter.shader: make "specular hightlight" and "matcap(additive)" add their additive Luminance to alpha value, to simulate correct reflection on "very low or zero alpha" semi-transparent material. If you find that after this update, some semi-transparent's reflection is too bright, you need to reduce their intensity to a normal level
- (Core) NiloToonCharacter.shader: _EditFinalOutputAlphaEnable & _ZOffsetEnable are now default off, since in most cases they are not needed
- (Core) NiloToonCharacter.shader: for all opauqe material(SrcBlend = One, DstBlend = Zero), shader will force outputAlpha = 1, to produce a correct alpha value on RenderTexture

Added
- (Core) NiloToonCharacter.shader: + _BaseColor2
- (Core) NiloToonCharacter.shader: + _SpecularColorTintMapTilingXyOffsetZw,_SpecularColorTintMapUseSecondUv
- (Core) NiloToonPerCharacterRenderController.cs: +receiveAverageURPShadowMap,receiveStandardURPShadowMap,receiveNiloToonSelfShadowMap
---
[0.10.7] - 2022-06-13

Known Issue
- (Core) NiloToonEnvironment.shader: this shader is not SRP-batcher compatible since 0.8.20(0.8.19 is safe), we are fixing it
- (Core) For URP12 or newer, if depthPriming is enabled, character shader's face will not render correctly, we are fixing it

Added
- (Core) NiloToonCharacter.shader: + _SpecularUseReplaceBlending

Changed
- (InternalCore) NiloToonCharacter.shader: move ZOffsetFinalSum != 0 check to NiloZOffsetUtil.hlsl

Fixed
- (Core) NiloToonCharacter.shader: Fixed "physical camera makes depth texture rim light in shader not correct" bug
- (Core) Fixed NiloToonCharacter.shader's shader warning

[0.10.6] - 2022-05-31

Known Issue
- (Core) NiloToonEnvironment.shader: this shader is not SRP-batcher compatible since 0.8.20(0.8.19 is safe), we are fixing it
- (Core) For URP12 or newer, if depthPriming is enabled, character shader's face will not render correctly, we are fixing it

Added
- (Core) NiloToonCharacter.shader: + _OutlineUniformLengthInViewSpace
- (Core) NiloToonCharacter.shader: + _SkinMaskMapRemapStart,_SkinMaskMapRemapEnd
- (Core) NiloToonCharacter.shader: + _EnvironmentReflectionTintAlbedo
- (Core) added some Metal & stocking matcap textures
- (Doc) + section "How to set metal/reflective/fresnel rim light/specular material?"
- (Doc) + section "How to set black/white stocking material?"

Changed
- (Core) NiloToonCharacter.shader: section "Environment Reflections" rewrite blending method, remove (Experimental tag)

[0.10.5] - 2022-05-26

Known Issue
- (Core) NiloToonEnvironment.shader: this shader is not SRP-batcher compatible since 0.8.20(0.8.19 is safe), we are fixing it
- (Core) For URP12 or newer, if depthPriming is enabled, character shader's face will not render correctly, we are fixing it

Added
- (Core) support URP12 deferred rendering path

- (Core) support URP12 depth priming mode, but face is not rendering correctly in this version
- (Core) added NiloToonShaderPatch(for URP12 or newer).unitypackage, user must import it when using URP12 or newer

Changed
- (Core) NiloToon debug window: default debug case is "JustBaseMap" now

Fixed
- (Core) NiloToonAverageShadowTestRT.shader: fix a bug that make this shader not able to compile in Unity2019.4(URP7)

[0.10.4] - 2022-05-23

Known Issue
- (Core) NiloToonEnvironment.shader: this shader is not SRP-batcher compatible since 0.8.20(0.8.19 is safe), we are fixing it

Added
- (Core) NiloToon debug window: added "JustBaseMap" case, which is different to albedo
- (Core) NiloToonCharacter.shader: smoothness section added note "Make sure _Smoothness is not 0 in order to see specular result"

Changes
- (Core) NiloToonShadowControlVolume.cs: increase shadowRange max value to 1000, from 100
- (Core) NiloToonCharacter.shader: _MatCapAlphaBlendTintColor added [HDR] tag

Fixed
- (Core) NiloToonCharacter.shader: fix LoadDepthTextureLinearDepthVS() not clamping out of range loadTexPos, character depth tex rim light will render correctly on edge of screen now
- (Core) NiloToonCharacter.shader: fix some harmless shader warning
---
[0.10.3] - 2022-05-16

Known Issue
- (Core) NiloToonEnvironment.shader: this shader is not SRP-batcher compatible since 0.8.20(0.8.19 is safe), we are fixing it

Breaking Changes
- (Core) NiloToonCharacter.shader: Rewrite Kajiya-Kay Specular (for hair) section, in old versions we hardcoded light direction as world space up, which is a wrong implementation, we now replace it with head up direction, we also rewrite NiloStrandSpecular.hlsl to use a better method. Old material settings of Kajiya-Kay Specular (for hair) section may need to adjust by user
- (Core) NiloToonCharacter.shader: _FaceShadowGradientMapChannel's default value change from g to r, because the recommend texture format is changed to R8 now, which use less memory
- (Core) FaceShadowGradientMaps change format from auto(rgba32) to r8
- (Demo) upgrade demo project to 2020.3.33f1

Added
- (Core) support Unity2022.1.0f1, added "NiloScaledScreenParamUtil.hlsl" for supporting Unity2022.1.0f1(URP13.x)
- (Core) NiloToonCharacter.shader: section "Depth Texture Rim Light and Shadow" added _ApplyDepthTexRimLightFixDottedLineArtifacts and _DepthTexRimLightFixDottedLineArtifactsExtendMultiplier, user can enable it to solve DepthTex rim light's weird white line/dot artifacts on uv seams
- (Core) NiloToonCharacter.shader: section "FaceShadowGradientMap" added note about using texture R8 format and None compression
- (Core) NiloToonCharacter.shader: added _NiloToonSelfShadowMappingTintColor
- (Core) NiloToonCharacter.shader: added _FaceShadowGradientThresholdMin,_FaceShadowGradientThresholdMax
- (Core) NiloToonCharacter.shader: added _FaceShadowGradientMapInvertColor
- (Core) NiloToonCharacter.shader: added _FaceShadowGradientMapUVxInvert
- (Core) NiloToonCharacter.shader: added _IgnoreDefaultMainLightFaceShadow
- (Core) NiloToonCharacter.shader: added _FaceMaskMapRemapMinMaxSlider
- (Core) NiloToonCharacter.shader: added _FaceMaskMapInvertColor
- (Core) NiloToonCharacter.shader: added _RampLightingNdotLRemapMinMaxSlider
- (Core) NiloToonCharacter.shader: added _RampLightingSampleUvYTexChannelMask
- (Core) NiloToonCharacter.shader: added _RampLightingSampleUvYTexInvertColor, _RampLightingUvYRemapMinMaxSlider
- (Core) NiloToonCharacter.shader: added _RampLightingFaceAreaRemoveEffect
- (Core) NiloToonCharacter.shader: added _BaseMapStackingLayer1MaskInvertColor
- (Core) NiloToonCharacter.shader: matcap sections added note about usable textures in NiloToonURP folder
- (Core) NiloToonCharacter.shader: Outline pass added multi_compile _ADDITIONAL_LIGHT_SHADOWS
- (Core) NiloToonEditorShaderStripping: added _ADDITIONAL_LIGHT_SHADOWS stripping
- (Demo) Support Unity2022.1.0f1

- (Demo) Added and updated a few new demo characters

Changes
- (Core) [IMPORTANT]NiloToonCharacter.shader: due to vulkan constant buffer size limitation, we disabled srp batching in vulkan, all shader features are now supported correctly on vulkan again, but batching performance is reduced
- (Core) update NiloToonAverageShadowTestRT.shader algorithm
- (InternalCore) apply "NiloScaledScreenParamUtil.hlsl"'s GetScaledScreenWidthHeight() to replace all _CameraDepthTexture_TexelSize.zw calls
- (InternalCore) NiloToonCharacter.shader Lit pass alpha blend write = One OneMinusSrcAlpha (alpha set to One OneMinusSrcAlpha, due to semi-transparent overwrite wrong value to RT.a, see https://twitter.com/MuRo_CG/status/1511543317883863045)
- (InternalCore)Outline pass always render as opaque colors, so we force output alpha always equals 1
- (Demo) PC quality settings + SSAO
- (Doc) updated document about how to install Jobs in package manager

Fixed
- (Core) NiloToonCharacter.shader: _RampLightingTex force LOD 0 sample in shader, to fix sampling mipmap artifact
- (Core) NiloToonCharacter.shader: write a correct cameraBackwardDir method for _ApplyAlphaOverrideOnlyWhenFaceForward
- (Core) NiloToonCharacter.shader: fix a bug that ramp lighting doesn't consider shading grade map
- (Core) NiloToonEditorPerCharacterRenderControllerCustomEditor: fix a bug that after clicking auto setup character, uv8 is not generated
- (Core) NiloStrandSpecular.hlsl: fix dirAtten missing multiply bug
- (Core) FaceShadowGradientMap textures use repeat instead of clamp
- (Demo) fix 4ExtremeHighQuality.asset use wrong rendererdata bug
- (Demo) jp envi scene disable character screen space outline

---

[0.10.2] - 2022-04-27

Known Issue
- (Core) NiloToonEnvironment.shader: this shader is not SRP-batcher compatible since 0.8.20(0.8.19 is safe), we are fixing it

Added
- (Core) NiloToonCharacter.shader: section "BaseMap Alpha Override" added _ApplyAlphaOverrideOnlyWhenFaceForwardIsPointingToCamera

Fixed
- (Core) revert this change due to invalid chinese char in .cs-> NiloToonEditorPerCharacterRenderControllerCustomEditor.cs: when user click "Auto setup this character" button, we added more material name keywords, to help NiloToonURP auto setup more materials to correct settings

---

[0.10.1] - 2022-04-25

Known Issue
- (Core) NiloToonEnvironment.shader: this shader is not SRP-batcher compatible since 0.8.20(0.8.19 is safe), we are fixing it

Added
- (Core) NiloToonPerCharacterRenderController.cs: added perCharacterRimLightSharpnessPower, it will let user control the sharpness of perCharacterRimLight
- (Core) NiloToonPerCharacterRenderController.cs: added "PerCharacter BaseMap override" section, user can use it to apply effect to a character(e.g. character convert to ice texture when freeze)
- (Core) NiloToonShaderStrippingSettingSO.cs: added include_NILOTOON_PERCHARACTER_BASEMAP_OVERRIDE
- (Core) NiloToonCharacter.shader: section "Screen Space Outline" added note about "Not supported on Vulkan"
- (Core) NiloToonCharacter.shader: section "Shading Grade Map" added _ShadingGradeMapInvertColor, _ShadingGradeMapApplyRange
- (Core) NiloToonCharacter.shader: section "Face Shadow Gradient Map" added _FaceShadowGradientResultSoftness, _FaceShadowGradientMapChannel
- (Core) NiloToonCharacter.shader: section "BaseMapAlphaOverride" added _AlphaOverrideStrength
- (Core) NiloToonCharacter.shader: section "Traditional Outline" added _OutlineTintColorSkinAreaOverride
- (Core) NiloToonShadowControlVolume.cs: added URPShadowblurriness,URPShadowAsDirectLightTintIgnoreMaterialURPUsageSetting
- (Core) added textures: FaceShadowGradientMap_StyleA,FaceShadowGradientMap_StyleB,FaceShadowGradientMap_StyleC
- (Core) added NiloToonEnvironment(for URP12.x).unitypackage, user can install it to support URP12
- (InternalCore) ShaderDrawer.cs: TitleDecorator added "SpaceLine" keyword, for supporting Rider
- (InternalCore) NiloToonCharacter.shader: now support Rider highlight (replace [Title(X,)] to keyword SpaceLine)

- (Demo) Added more test characters
- (Demo) Added new scene: full body shoot scene
- (Demo) upgrade Magica cloth to latest version, fix Magica cloth not working in 2021.2 bug

Breaking Change
- (Core) NiloToonCharacter.shader: reorder and rewrite UI, added header & space to group functions into categories. (This is a very big change to NiloToonCharacter.shader, so we consider it as Breaking Change even it should not break anything unless you edit NiloToonCharacter.shader's code before)

Changed
- (Core) NiloToonEditorPerCharacterRenderControllerCustomEditor.cs: when user click "Auto setup this character" button, it will now auto find the best faceUpDirection and faceForwardDirection, it is an important improvement because most user did not not setup faceUpDirection and faceForwardDirection correctly
- (Core) NiloToonEditorPerCharacterRenderControllerCustomEditor.cs: hen user click "Auto setup this character" button, we added more material name keywords, to help NiloToonURP auto setup more materials to correct settings
- (Core) NiloToonCharacter.shader: _HairStrandSpecularShapePositionOffset range increase from (-0.5,0.5) to (-1,1)
- (Core) NiloToonCharacter.shader: if _NILOTOON_DEBUG_SHADING is active, now we only render NiloToonForwardLitPass to show better debug result
- (Core) NiloToonCharacter.shader: Convert "Depth Tex Rim Light and Shadow" section to a toggle feature
- (Core) NiloToonPerCharacterRenderController.cs: convert GetFinalFaceDirectionWS(...) to a public function

Fixed
- (Core) NiloToonEditorPerCharacterRenderControllerCustomEditor.cs: fix a critical performance problem which NiloToonPerCharacterRenderController is extremely laggy when user focus it
- (Core) NiloToonCharacter.shader: section "Shading Grade Map" fixed _ShadingGradeMap's wrong description
- (Core) NiloToonCharacter.shader: fix a bug where _BackFaceTintColor is not working
- (Core) NiloToonCharacter.shader: _ZOffsetMultiplierForTraditionalOutlinePass fixed wrong default value in UI(0 is wrong, 1 is correct)
- (Core) NiloToonBloomVolume.cs: it will now correctly support URP12, user don't need to do anything
- (Core) NiloToonCharacter.shader: outline pass added shader_feature_local _RECEIVE_URP_SHADOW, else outline is not correctly darken when character is within URP shadow
- (Demo) Demo project will now strip include_NILOTOON_DISSOLVE and include_NILOTOON_PERCHARACTER_BASEMAP_OVERRIDE in NiloToonShaderStrippingSettingSO.asset
---
[0.9.1] - 2022-03-21

Known Issue
- (Core) NiloToonEnvironment.shader: this shader is not SRP-batcher compatible since 0.8.20(0.8.19 is safe), we are fixing it

Added
- (Core) NiloToonCharacter.shader: section "Face Shadow Gradient Map" added _FaceShadowGradientMaskMapChannel
- (Core) NiloToonCharacter.shader: section "Depth Texture Rim Light and Shadow" added _DepthTexRimLightIntensity,_DepthTexRimLightMixWithBaseMapColor
- (Core) NiloToonCharacter.shader: section "Mat Cap (additive)" added _MatCapAdditiveMixWithBaseMapColor
- (Core) NiloToonCharacter.shader: section "Kajiya-Kay Specular (for hair)" added _HairStrandSpecularMixWithBaseMapColor,_HairStrandSpecularOverallIntensity,_HairStrandSpecularMainIntensity,_HairStrandSpecularSecondIntensity,_HairStrandSpecularShapeFrequency,_HairStrandSpecularShapeShift,_HairStrandSpecularShapePositionOffset
- (Core) NiloToonCharRenderingControlVolume.cs: added characterOverallShadowTintColor
- (Core) NiloToonCharRenderingControlVolume.cs: added characterOverallShadowStrength
- (Core) NiloToonCharRenderingControlVolume.cs: added specularTintColor
- (Core) NiloToonCharRenderingControlVolume.cs: added charOutlineWidthAutoAdjustToCameraDistanceAndFOV
- (Core) NiloToonToonOutlinePass.cs: added charOutlineWidthAutoAdjustToCameraDistanceAndFOV (you can find it in NiloToonAllInOneRendererFeature's UI)
- (Demo) added new character prefab: CH0066_Mesh Variant.prefab, suz.prefab, yu.prefab, miku.prefab, Miko (NiloToon Variant).prefab

Breaking Change
- (Core) fix "NiloToonCharacter.shader not working on Vulkan platform(android Mali G-** GPU)", but now we disabled basemap stacking layer 3-10 & screen space outline on vulkan as a work around to avoid Vulkan platform(android Mali G-** GPU) constant buffer size limitation
- (Core) NiloToonCharacter.shader: base map stacking layer 3,5,7,9 will now use sampler_linear_clamp, to reduce sampler count
- (Core) NiloToonCharacter.shader: base map stacking layer 4,6,8,10 will now use sampler_linear_repeat, to reduce sampler count

- (Core) NiloToonCharacter.shader: 5 mask maps -> _MatCapAlphaBlendMaskMap,_MatCapAdditiveMaskMap,_MatCapOcclusionMaskMap,_SkinMaskMap,_EnvironmentReflectionMaskMap use sampler_BaseMap instead of their own texture sampler, in order to reduce sampler count
- (InternalCore) NiloOutlineUtil.hlsl->GetOutlineCameraFovAndDistanceFixMultiplier(...) added "applyPercentage" param, used by NiloToonCharRenderingControlVolume/NiloToonToonOutlinePass's charOutlineWidthAutoAdjustToCameraDistanceAndFOV
- (InternalCore) NiloStrandSpecular.hlsl->ShiftTangent(...) added frequency,shift,offset param
- (Demo) Demo project -> android player setting -> default Graphics API = Vulkan+OpenGLES now, instead of OpenGLES only

Changed
- (Core) better notes in character shader material UI
- (Core) remove _StencilRef and _StencilComp because they are not being used for a long time
- (Core) rename "Alpha Override" to "BaseMap Alpha Override" (affects UI display only)
- (Core) rename "KajiyaKaySpecular" to "Kajiya-Kay Specular (for hair)" (affects UI display only)
- (InternalCore) Improve NiloToonCharacter.shader comments in all related .hlsl files
- (InternalCore) NiloToonCharacter_Shared.hlsl: rename positionWS to positionWS_ZOffsetFinalSum because it is a packed float4(xyz = positionWS, w = ZOffsetFinalSum)
- (InternalCore) NiloToonCharacter_Shared.hlsl: change _GlobalSpecularIntensityMultiplier to _GlobalSpecularTintColor to combine specularTintColor in volume
- (Doc) Improve document pdf
- (Demo) Upgrade project to Unity2020.3.30f1
- (Demo) Optimize NiloToonDancingDemo.unity
- (Demo) Optimize NiloToon_CRS_SampleScene.unity
- (Demo) Optimize NiloToon_CloseShot_SampleScene.unity (remove rendering URP shadow and NiloToon's average shadow rendering)
- (Demo) Force NiloToon_CloseShot_SampleScene.unity's NiloToon shadowmap resolution = 2048, even for mobile quality
- (Demo) Optimize RenderScale and shadow resolution for mobile quality
- (Demo) Force android max screen resolution height = 900, instead of 1080
- (Demo) Highest quality turn on additional light shadow, and allow 4096 resolution
- (Demo) MaGirl materials allow Screen space outline now
- (Demo) Improve Sienna skin material setting
- (Demo) Improve Luxiya, Klee material setting
- (Demo) NiloToonDancingDemo.unity update camera default active state, user should be able to see character before entering play mode

Fixed
- (Core) NiloToonAverageShadowTestRTPass.cs: fix "camera anime postprocess average shadow not working"
- (Core) NiloToonAverageShadowTestRTPass.cs: fix "camera anime postprocess average shadow -> shadow test position out of camera frustrum bound" problem
- (Core) NiloToonCharacter.shader: fix shader compile warning
- (Doc) Fix typo

[0.8.35] - 2021-12-29

Breaking Change
- (Core) NiloToonCharacter.shader: set _ZOffsetMultiplierForTraditionalOutlinePass's default value to 1, which enable ZOffset for outline pass by default now (e.g. 3D eyebrow with ZOffset will show outline by default now)

Added
- (Core) NiloToonPerCharacterRenderController.cs: Add "Dissolve" section
- (Core) NiloToonCharacter.shader: Add "Pass On/Off" section, allow you to disable target pass(e.g. disable URP shadow caster) per material
- (InternalCore) Added "NiloToonURP7Support.hlsl"
- (Core) Added "NiloToonDefaultDissolvePerlinNoise" in Resources folder
- (Core) NiloToonShaderStrippingSettingSO.cs: + _NILOTOON_DISSOLVE stripping

Fixed
- (Core) NiloToonCharacter.shader: support 2019.4.33f1(URP7.7.1)
- (Core) NiloToonCharacter.shader: support OpenGLES3.0 & WebGL by removing #pragma target 4.5
- (Core) NiloToonCharacter.shader: fix debug shading can't compile for face area
- (Core) NiloToonCharacter.shader: try to fix a bug introduced in 0.8.22, which makes mobile (vivo Y73s) (TAS_AN00) not rendering corerctly
- (Core) NiloToonCharacter.shader: added if() check to make planar reflection correct by default, when ZOffsetFinalSum is 0
- (Core) NiloToonCharacter.shader: ZOffset will not affect depth texture rim light and shadow's result now. In the past, ZOffset will make material's depth texture rim light become white wrongly. (e.g. eyebrow with ZOffset)

[0.8.34] - 2021-12-14

Added
- (Core) NiloToonCharacter.shader: add "Parallax Map" feature, same as URP's Lit.shader
- (Core) NiloToonCharacter.shader: add "Shading Grade Map" feature, similar to UTS2's "Shading Grade Map"
- (Core) NiloToonCharacter.shader: add "Enable Rendering" toggle
- (Core) NiloToonCharacter.shader: add "Back face color tint"

Changed
- (Core) NiloToonCharacter.shader: improve additional light shading for face area
- (InternalCore) NiloToonCharacter.shader: viewDirectionWS is now calculate in vertex shader using GetWorldSpaceViewDir(), and will take care of orthographic camera
- (InternalCore) NiloToonCharacter.shader: change some NormalizeNormalPerPixel() to normalize() to ensure normalize in all platform
- (InternalCore) NiloToonCharacter.shader: refactor additional light code
- (InternalCore) NiloToonCharacter.shader: refactor "disable rendering" section

Fixed
- (Core) NiloToonCharacter.shader: remove wrong unity_WorldTransformParams.w multiply when calculating bitangent in fragment shader
- (Core) NiloToonCharacter.shader: fixed a bug that Perspective removal doesn't affect NiloToonPrepassBufferPass correctly
---
[0.8.33] - 2021-12-6

Added
- (Core) Added NiloToonDepthTexRimLightAndShadowGlobalOnOff.cs, and will now called by NiloToonPlanarReflectionHelper.cs
- (Core) NiloToonPlanarReflectionHelper.cs: will now call NiloToonDepthTexRimLightAndShadowGlobalOnOff's functions also
- (Core) NiloToonPlanarReflectionHelper.cs: added command buffer version functions, for user that renders planar reflection using renderer feature(command buffer)
- (Doc) Added "NiloToon’s character shader is broken in build, it only works correctly in editor"

Changed
- (Core) NiloToonCharacter.shader: now _BASEMAP_STACKING_LAYER2 will use it's own texture sampler sampler_BaseMapStackingLayer2Tex, to allow more wrap mode control from user
---
[0.8.32] - 2021-12-4

Fixed
- (Core) NiloToonEnvironment.shader: added _CASTING_PUNCTUAL_LIGHT_SHADOW multi_compile, which means support URP11 point light shadow
---
[0.8.31] - 2021-12-1

Added
- (Core) NiloToonEnvironment.shader: added "Screen Space Outline" section
---
[0.8.30] - 2021-12-1

Added
- (Core) NiloToonEnvironment.shader: "splatmap" section added PackedDataMap(b for height) and a height multiplier
---
[0.8.29] - 2021-11-30

Changed
- (Core) NiloToonCharacter.shader: BaseMap Stacking Layer 1's texture allow user to edit wrap mode
---
[0.8.28] - 2021-11-26

Fixed
- (Core) NiloToonEnvironment.shader: fix "splatmap normalmap not enabled" bug

Changed
- (Demo) Upgrade project to Unity2020.3.23f1

[0.8.27] - 2021-11-26

Fixed
- (Core) support URP7.6.0 (Unity2019.4.32f1)

[0.8.26] - 2021-11-25

Added
- (Core) NiloToonCharacter.shader: +_FixFaceNormalUseFlattenOrProxySphereMethod, to let user choose how to fix face normal (flat or round sphere)

Fixed
- (Core) NiloToonEnvironmentShaderGUI.cs: +temp support URP12

[0.8.25] - 2021-11-24

Fixed
- (Core) NiloToonEnvironment.shader: use a much better splat map blending method

Changed
- (InternalCore) NiloToonCharacter_Shared.hlsl: change Varying struct's data layout
- (InternalCore) NiloToonCharacter_Shared.hlsl: cleanup and refactor code, improve code notes
---
[0.8.24] - 2021-11-22

Fixed
- (Core) NiloToonCharacter.shader: fix main light culling mask bug when more than 1 directional light exist in scene
---
[0.8.23] - 2021-11-17

Added
- (Core) NiloToonEnvironment.shader: +splatmap's smoothness control
- (Core) NiloToonEnvironment.shader: +splatmap's normal intensity control

Changed
- (Core) NiloToonEnvironment.shader: improve splatmap's mateiral UI

Fixed
- (Core) NiloToonEnvironment.shader: fix blending softness's bug
---
[0.8.22] - 2021-11-14

Added
- (Core) NiloToonCharacter.shader: increase BaseMapStackingLayer count from 6 to 10, each layer added mask texture, uv scale offset, uv anim speed
- (Core) NiloToonCharacter.shader: "ZOffset" section now affect NiloToonSelfOutlinePass also
- (Core) NiloToonCharacter.shader: "ZOffset" section added _ZOffsetMultiplierForTraditionalOutlinePass

Changed
- (InternalCore) NiloToonCharacter.shader: BaseMapStackingLayer now use a shared sampler to prevent max 16 sampler limit
- (InternalCore) NiloToonCharacter.shader: unify all ZOffset to 1 NiloGetNewClipPosWithZOffsetVS() call
---
[0.8.21] - 2021-11-12

Changed
- (Core) NiloToonEnvironment.shader: splat map section now controlled by keyword shader_feature _SPLATMAP

[0.8.20] - 2021-11-11

Changed
- (Core) NiloToonCharacter.shader: "Face Shadow Gradient Map" section is not experimental now
- (Core) NiloToonEnvironment.shader: add working splat map section

Fixed
- (Core) NiloToonCharacter.shader: "Face Shadow Gradient Map" will only render on face area
- (Core) NiloToonCharacter.shader: "Face Shadow Gradient Map"'s mask texture uv will now use basemap's uv
---
[0.8.19] - 2021-11-10

Added
- (Core) NiloToonEnvironment.shader: added WIP splat map section, it has no effect in this version
- (Core) NiloToonCharRenderingControlVolume.cs: added overrideLightDirectionIntensity, overridedLightUpDownAngle, overridedLightLRAngle

Changed
- (Core) NiloToonCharacter.shader: _FaceShadowGradientOffset default value is now 0.1, instead of 0
- (InternalCore) NiloToonEnvironment.shader: now GUI is controlled by custom material UI code
---
[0.8.18] - 2021-11-04

Breaking Changes
- (Core) NiloToonCharacter.shader: much better additional light default value in "Lighting style" section

Fixed
- (Core) NiloToonCharacter.shader: fix point spot light's shader bug about URP's additional light shadow
---
[0.8.17] - 2021-11-03

Breaking Changes
- (Core) NiloToonCharacter.shader: better additional light default value in "Lighting style" section

Added
- (Core) NiloToonCharacter.shader: +_AdditionalLightIgnoreOcclusion,+_AdditionalLightDistanceAttenuationClamp

Fixed
- (Core) NiloToonCharacter.shader: fix point spot light's shader bug when shadow is off
---
[0.8.16] - 2021-11-02

Breaking Changes
- (Core) NiloToonCharacter.shader: completely rewrite additional light!!! it now support shadow map, fragment quality shading, adjustable cel shade param, but much slower.
- (Core) NiloToonCharacter.shader: add _AdditionalLightCelShadeMidPoint,_AdditionalLightCelShadeSoftness,_AdditionalLightIgnoreCelShade
- (Core) NiloToonCharacter.shader: add _OverrideAdditionalLightCelShadeParamForFaceArea,_AdditionalLightCelShadeMidPointForFaceArea,_AdditionalLightCelShadeSoftnessForFaceArea,_AdditionalLightIgnoreCelShadeForFaceArea
---
[0.8.15] - 2021-10-31

Breaking Changes
- (Core) NiloToonCharacter.shader: add "_SpecularShowInShadowArea" in "Specular Highlights" section, which is default 0. Before this version specular result will show in shadow area also. User will need to edit this slider if they want to show specular result in shadow area.
- (Core) NiloToonCharacter.shader: "Ramp Texture(Specular)" section affected by _SpecularMap mask now
---
[0.8.14] - 2021-10-29

Added
- (Core) NiloToonEditorDebugWindow.cs: added "Force all windows update in edit mode" section, to allow user force update all view(e.g. game view) in edit mode, but enable it will make editor slower due to extra repaint
- (Core) NiloToonCharacter.shader: add "Matcap (occlusion)" section
- (Core) NiloToonCharacter.shader: added _FaceShadowGradientIntensity & _FaceShadowGradientMaskMap for "Face Shadow Gradient Map" section

Fixed
- (Core) NiloToonEditorDebugWindow.cs: fix "editing debug shading param can not instantly see result in game view" bug, now we call UnityEditorInternal.InternalEditorUtility.RepaintAllViews() when changes are made by user
- (Core) NiloToonEditorDebugWindow.cs: fix "if there are more than 1 NiloToon renderer feature enabled, debug window is not working" bug, solved by using static var instead of Singleton Instance var in NiloToonSetToonParamPass.cs
---
[0.8.13] - 2021-10-15

Fixed

- (Core) NiloToonToonOutlinePass.cs: fix "when user edit NiloToon renderer feature's value, screen will flickering" bug
---
[0.8.12] - 2021-10-15

Added
- (Core) NiloToonCharacter.shader: add _FaceAreaCameraDepthTextureZWriteOffset, which replaced a hardcode const value
---
[0.8.11] - 2021-10-14

Fixed
- (Core) NiloToonSetToonParamPass.cs: fix "when user edit NiloToon renderer feature's value, screen will flickering" bug
---
[0.8.10] - 2021-10-13

Changed
- (Demo) manifest.json: remove com.unity.toolchain.win-x86_64-linux-x86_64

[0.8.9] - 2021-10-11

Added
- (Core) NiloToonCharacter.shader: added _FaceShadowGradientMapUVScaleOffset & _DebugFaceShadowGradientMap for "Face Shadow Gradient Map" section
---
[0.8.8] - 2021-10-10

Added
- (Core) NiloToonCharacter.shader: add more tips and information for "Face Shadow Gradient Map" section

Fixed
- (Core) NiloToonToonOutlinePass.cs: added isPreviewCamera check for screen space outline, to solve "material preview window makes outline flicker" bug
---
[0.8.7] - 2021-10-8

Breaking Changes
- (Core) NiloToonCharacter.shader: fix a bug "normalmap can not affect lighting result"

[0.8.6] - 2021-10-4

Breaking Changes
- (Core) NiloToonPerCharacterRenderController.cs: fix a gamma/linear color bug that cause play/edit mode color is not the same
---
[0.8.5] - 2021-9-29

Added
- (Core) NiloToonCharacter.shader: added _DepthTexRimLightAndShadowWidthTexChannelMask and _OutlineWidthTexChannelMask option

Fixed
- (Core) NiloToonEditorPerCharacterRenderControllerCustomEditor.cs: + null check
---
[0.8.4] - 2021-9-28

Added
- (Core) NiloToonPerCharacterRenderController.cs: added allowCacheSystem, to let user control CPU optimization option
- (Core) NiloToonPerCharacterRenderController.cs: added API RequestForceMaterialUpdateOnce(), for user to call after changing material in playmode
- (Core) NiloToonPerCharacterRenderController.cs: OnEnable will now call RequestForceMaterialUpdateOnce()

[0.8.3] - 2021-9-27

Breaking Changes
- (Core) NiloToonPerCharacterRenderController.cs: combine allowRenderDepthOnlyPass and allowRenderDepthNormalsPass into 1 public bool allowRenderDepthOnlyAndDepthNormalsPass

Added
- (Core) NiloToonCharacter.shader: BaseMap Alpha Blending Layer 1 added UVScaleOffset, UVAnimSpeed, and optional mask texture. If it is useful, we will add these feature to other layers

Changed
- (Core) remove keyword _NILOTOON_ENABLE_DEPTH_TEXTURE_RIMLIGHT_AND_SHADOW in C# and shader, replaced by a new material float _NiloToonEnableDepthTextureRimLightAndShadow. In order to trade a small amount of fragment performance for 50% lower shader memory usage
- (Core) NiloToonPerCharacterRenderController.cs: _NiloToonEnableDepthTextureRimLightAndShadow will not be enabled if allowRenderDepthOnlyAndDepthNormalsPass is false, to prevent wrong rim light result when user disabled allowRenderDepthOnlyAndDepthNormalsPass
- (Core) NiloToonPerCharacterRenderController.cs: auto set up button will not upgrade particle and sprite material anymore

Fixed
- (Core) NiloToonPerCharacterRenderController.cs: now script will not miss material update(shadowTestIndex_RequireMaterialSet) due to cache logic when setting up a new character
- (Core) fixed a bug that triggers useless Debug.warning when setting up a new character using auto button
---
[0.8.2] - 2021-9-23

Added
- (Core) NiloToonCharacter.shader: added _DepthTexRimLightThresholdOffset and _DepthTexRimLightFadeoutRange, to let user control rim light area
---
[0.8.1] - 2021-9-21

Added
- (Core) NiloToonCharacter.shader: added "Face shadow gradient map" section, allow user to control face shadow's visual by a special threshold gradient grayscale texture
- (Core) NiloToonPerCharacterRenderController.cs: added "Face Up Direction", for user to setup. (will affect material's "Face shadow gradient map" result)
- (Core) NiloToonEditorShaderStripping.cs will strip _FACE_SHADOW_GRADIENTMAP if _ISFACE is off also
- (Core) NiloToonPerCharacterRenderController: added allowRenderShadowCasterPass,allowRenderDepthOnlyPass,allowRenderDepthNormalsPass,allowRenderNiloToonSelfShadowCasterPass,allowRenderNiloToonPrepassBufferPass for user side optimization option
- (Demo) Update Ganyu.prefab and Klee.prefab's face rendering(using new "Face shadow gradient map" feature)

Changed
- (Demo) upgrade URP from 10.5.0 to 10.5.1
- (Demo) optimize close shot scene's camera near far

Fixed
- (Core) optimize NiloToonPerCharacterRenderController.LateUpdate()'s CPU time cost (only call material.SetXXX when value changed) and "IEnumerable<Renderer> AllRenderersIncludeAttachments()"'s GC
- (Demo) fix bug: "multiple incorrect renderer feature data in forward renderers"

[0.7.3] - 2021-9-12

Changed
- (Core) NiloToonPerCharacterRenderController will not clear material property block every frame now

Fixed
- (Core) correctly support 'ClothDynamics' asset

[0.7.2] - 2021-9-10

Added
- (Core) NiloToonCharacter.shader: Emission section added _EmissionMapUseSingleChannelOnly and _EmissionMapSingleChannelMask
- (Core) NiloToonCharacter.shader: added "Support 'ClothDynamics' asset" section(just an on/off toggle)
- (InternalCore) Added NiloToonFullscreen_Shared.hlsl, to provide FullscreenVert() function for fullscreen shaders for URP9 or lower

Changed
- (Core) When EnableDepthTextureRimLightAndShadow is off, low quality fallback rim light will NOT consider normalmap contribution anymore, which can produce rim light that is much more similar to depth texture rim light

Fixed
- (Core) Fix shader bug to correctly support Unity 2019

[0.7.1] - 2021-9-6

Added
- (Core) Added NiloToonBloomVolume.cs, a bloom volume that is similar to URP's offical Bloom, but with more controls
- (Core) Added NiloToonRenderingPerformanceControlVolume .cs, a control volume that let you control performance per volume
- (Core) NiloToonCharacter.shader: Added "Allow NiloToonBloom Override?" section, and all required functions
- (InternalCore) NiloToonCharacter_Shared.shader: Added "Allow NiloToonBloom Override?" section required MACRO, uniforms and functions
- (InternalCore) Added new files NiloToonPrepassBufferRTPass.cs, NiloToonUberPostProcessPass.cs, NiloToonBloom.shader & NiloToonUberPost.shader
- (InternalCore) NiloToonAllInOneRendererFeature.cs now enqueue NiloToonPrepassBufferRTPass and NiloToonUberPostProcessPass
- (InternalCore) NiloToonCharacter.shader: Added NiloToonPrepassBuffer pass

Changed
- (Core) NiloToonCharacter.shader: _CelShadeSoftness minimum value is now 0.001, instead of 0, to avoid lighting direction flipped.
- (Core) Remove large amount of useless Debug.Log of NiloToonEditor_AssetLabelAssetPostProcessor.cs
- (InternalCore) Delete files that are not currently using: MathUtility.cs and NiloToonShaderStrippingSettingSO.cs(a duplicated file)

Fixed
- (Core) all character shader: all _DitherOpacity are now correctly replaced by_DitherFadeoutAmount, shader can compile correctly(material not pink anymore)
- (Core) fix NiloToonEditor_AssetLabelAssetPostProcessor's memory leak, which will produce fatal error when importing large amount of character
- (InternalCore) add namespace to NiloToonEditor_EditorLoopCleanUpTempAssetsGenerated.cs and NiloToonEditorSelectAllMaterialsWithNiloToonShader.cs

TODO
- (Core) Optimize NiloToonPerCharacterRenderController.LateUpdate() (only call material.SetXXX when value changed)

[0.6.3] - 2021-8-27

Added
- (Core) NiloToonCharacter.shader: Added a "BaseMap Stacking Layer 4-6" section
---
[0.6.2] - 2021-8-25

Changed
- (Core) change _DitherOpacity to _DitherFadeoutAmount (not just rename, but with logic change). This will fix a bug -> preview window can't render nilotoon character shader
---
[0.6.1] - 2021-8-24

Added
- (Core) NiloToonPerCharacterRenderController: Added a "Select all NiloToon_Character materials of this character" button
- (Core) Added NiloToonShaderStrippingSettingSO.cs, a scriptable object storing per platform NiloToon shader stripping settings
- (Core) NiloToonAllInOneRendererFeature: Added shaderStrippingSettingSO slot, user can assign a NiloToonShaderStrippingSettingSO to control NiloToon's shader stripping per platform
- (Core) NiloToonCharacter.shader: Added BaseMapStackingLayer section (1-3)
- (Core) NiloToonCharRenderingControlVolume: Added bool specularReactToLightDirectionChange
- (Core) NiloToonCharacter.shader: Added _NiloToonSelfShadowIntensityMultiplierTex in "Can Receive NiloToon Shadow?" section, useful if you want to control nilo self shadow intensity by a texture
- (Demo) Added NiloToonShaderStrippingSettingSO to project, which reduce all platform's shader memory usage by 50%, and android/iOS shader memory usage by 75%, this change will let demo apk run on low memory phones(1-2GB).
- (Demo) Added "Close Shot" scene, showing a close shot male character.
- (DOC) Added "How to separate IsFace, because I combine everything into 1 renderer/material" section
- (DOC) Added "Some DepthTexture shadow is missing if shadow caster is very close to shadow receiving surface"
- (DOC) Added a few section's about how to use volume to edit visual globally

- (DOC) Added "How to make specular results react to light direction change?" section

Changed
- (Core) delete all NiloToonURPFirstTimeOpenProject folder, scripts and files
- (Doc) rewrite "NiloToon shader is using too much memory, how to reduce it?" section, to explain how to use NiloToonShaderStrippingSettingSO

Fixed
- (Core) Fixed a bug -> when dither fadeout is 0% (completely invisible), depth/depthNormal texture pass will not render
---
[0.5.1] - 2021-8-16

Added
- (Core) NiloToonPerCharacterRenderController's dither fade out will now affect URP's shadowmap, will rely on URP's soft shadow filter to improve the final result.

Changed
- (Demo) Now demo project will dynamic load character from Resources folder, instead of just placing every character in scene. This will reduce memory usage a lot, which helps android build(.apk) to prevent out of memory crash.
- (Demo) keepLoadedShadersAlive in PlayerSetting is now false, to save some memory usage on scene change

Fixed
- (Core) Fix an important bug which NiloToonPerCharacterRenderController wrongly edit URP's default Lit material's shader. If any user were affected by version 0.4.1, please update to 0.5.1 and delete URP's Lit.material in the project's Library folder, doing this will trigger URP's regenerate default asset, which reset URP's Lit material and fixes the problem.
- (Core) Environment shader will NOT receive screen space outline if SurfaceType is Transparent
---
[0.4.1] - 2021-8-10

Breaking Changes
- (Core) NiloToonPerCharacterRenderController.cs: rename extraThickOutlineMaxiumuFinalWidth -> extraThickOutlineMaximumFinalWidth

Added
- (Core) NiloToonPerCharacterRenderController: Added "Auto setup this character" button, user can use click this button to quickly setup any character that is not using nilotoon
- (Demo) Added 4 new demo model, set up using NiloToon
- (Doc) Added section about "Auto setup this character" button
---
[0.3.5] - 2021-8-04

Added
- (Core) NiloToonCharRenderingControlVolume: +rim light distance fadeout params, you can use it if you want to hide rim light flicker artifact due to not enough pixel count for character(resolution low / character far away from camera)

Fixed
- (Core) now NiloToon will NOT leave any generated character .fbx in project (in older versions, NiloToon will generate .fbx that has (you can ignore or delete safely) prefix, polluting version control history)

[0.3.4] - 2021-8-02

Added
- (Core) character shader: Occlusion section added _OcclusionStrengthIndirectMultiplier
- (Core) character shader: depthtex rim light and shadow width can now optionally controlled by texture and vertex color
- (Doc) added "How to enable character shader’s screen space outline? section

Changed
- (Core) Now FrameDebugger will show NiloToon's passes correctly (Profiling scope)
- (Core) Now screen space outline can run on mobile platforms(android/iOS)
- (Core) screen space outline now support OpenGLES

Fixed
- (Core) NiloToonPerCharacterRenderController will now auto re-find allRenderers, if any null renderer is detected in allRenderers list
- (Core) LWGUI will now correctly not saving _ keyword in material, which will also fix screen space outline constantly flicking in scene window unless user's mouse is moving on GUI

- (Core) fix _GlobalIndirectLightMinColor not applied correctly, which ignored occlusion map
---
[0.3.3] - 2021-7-19

Added
- (Core) Environment shader/volume: +_NiloToonGlobalEnviSurfaceColorResultOverrideColor
- (Core) provide a new option in NiloToonAllInOneRendererFeature, "Perfect Culling For Shadow Casters", to fix terrain crash Unity bug
- (Doc) added section for "Perfect Culling For Shadow Casters (how to prevent terrain crash)"

Changed
- (Core) character shader: rewrite "Ramp texture (specular)" section
- (Core) update SpecularRampTexForEroSkin.psd

[0.3.2] - 2021-7-11

Added
- (Core) Improve XR rendering a lot! (by adding correct depth texture rim light and shadow in XR)
- (Core) NiloToonScreenSpaceOutlineControlVolume: add an extra outline width multiplier for XR
- (Core) NiloToonScreenSpaceOutlineControlVolume: add separated control for environment and character
- (Core) NiloToonCharacter shader: _LowSaturationFallbackColor's alpha can now control the intensity of _LowSaturationFallbackColor
- (Core) NiloToonCharacter shader: rewrite specular ramp method
- (Core) NiloToonPerCharacterRenderController: + _PerCharacterOutlineColorLerp
- (Core) NiloToonEnvironmentControlVolume: +_NiloToonGlobalEnviGITintColor , _NiloToonGlobalEnviGIAddColor , _NiloToonGlobalEnviShadowBorderTintColor
- (Demo) NiloToonEnviJPStreetScene now support playing in XR
- (Demo) Some example character models + MatCap(Additive) for metal reflection
- (Demo) Add version number on OnGUI (bottom right of the screen when playing)
- (Demo) Add UI text background transparent black quad, to make OnGUI text easier to read

Fixed
- (Core) Fixed an environmment shader bug, NiloToonURP can support 2019.4 (URP 7.6.0) now
- (Core) Fixed a bug "After switching platform or reimport character model .fbx assets, baked smooth normal outline data in character model's uv8 will sometimes disappear"
- (Core) Fixed a bug "focusing on NiloToonPerCharacterRenderController is very slow due to wrongly call to AssetDataBase.Refresh()""
- (Core) Fixed a bug "NiloToonPerCharacterRenderController running very slow and allocate huge GC"

[0.3.1] - 2021-7-05

Breaking Changes
- (Core) NiloToonCharacter_Shared.hlsl: rename struct LigthingData to ToonLightingData, so now NiloToonURP can also run on Unity2021.1 (URP11) or Unity2021.2 (URP12)
- (Core) NiloToonPerCharacterRenderController.cs: remove useCustomCharacterBoundCenter, now assigning customCharacterBoundCenter transform will treat as enable
- (Core) All screen space outline related feature, will be auto-disabled if running on mobile platform(android / iOS) due to performance and memory impact is high

Known Issues
- (Core) when inspector is focusing on any NiloToon_Character materials, screen space outline in editor will always keep flickering in scene/game window
- [Done in 0.3.2] (Core) (this bug already exist in 0.2.4) After switching platform or reimport character model assets, baked smooth outline data in character model's uv8 will sometimes disappear. You will need to click on that character's NiloToonPerCharacterRenderController script to trigger a rebake, or click "Windows/NiloToonUrp/Model Label/Re-fix whole project!" button to rebake every character
- (Core) (this bug already exist in 0.2.4) Sometime (you can ignore or delete it safely)xxx.fbx will be generated in project, and not correctly deleted by NiloToonURP automatically

Added
- (Core) Added NiloToonEnvironment.shader (Universal Render Pipeline/NiloToon/NiloToon_Environment), you can switch any URP's Lit shader to this shader, they are early stage proof of concept toon shader for environment (Don't use it for production! still WIP/in proof of concept stage, will change a lot in future). Added these envi shader because lots of customers request to include it first(even the shader is far from complete)
- (Core) Added NiloToonEnvironmentControlVolume.cs, to provide extra per volume control for NiloToonEnvironment.shader

- (Core) Added NiloToonScreenSpaceOutlineControlVolume.cs, for controlling NiloToonCharacter.shader and NiloToonEnvironment.shader's screen space outline's settings
- (Core) NiloToonToonOutlinePass.cs added "AllowRenderScreenSpaceOutline" option, you can enable it in NiloToonAllInOne renderer feature if you need to render any screen space outline(default is off)
- (Core) Character shader: Added a WIP and experimental "Ramp Texture (Specular) section"
- (Demo) Add NiloToonEnviJPStreetScene.scene (and all related assets), it is a new scene to demo the new environment shader
- (Doc) Added CustomCharacterBoundCenter section
- (Doc) Added a simple environment shader's document

Changed
- (Core) now support URP11.0 and URP12.0, because all NiloToon shader renamed struct LightingData to ToonLightingData (solving Shader error in URP11 -> 'Universal Render Pipeline/NiloToon/NiloToon_Character': redefinition of 'LightingData')
- (Core) Character shader: screen space outline(experimental) section completely rewrite (new algorithm, adding depthSensitifity texture, normalsSensitifity, normalsSensitifity texture, extra control for face....)
- (Core) Character shader: screen space outline use multi_compile now(to allow on off by different quality setting using renderer feature)
- (Core) Character shader: rename Outline to Traditional Outline, to better separate it from Screen space outline
- (Core) make screen space outline not affected by RenderScale,screen resolution (scale outline size to match resoltion)
- (Core) global uniform "_GlobalAspectFix" now use camera RT width height as input, instead of screen width height
- (Core) extract screen space outline's code to a new .hlsl (now shared by character shader and environment shader)
- (Core) NiloToonPerCharacterRenderController: optimize shader memory and build time for mobile, but increase GPU shading pressure a bit
- (Core) NiloToonPerCharacterRenderController: optimize C# cpu time spent, and reduce GC
- (Core) NiloToonPerCharacterRenderController: better warning in Status section, if something not set up correctly
- (Core) NiloToonEditorShaderStripping.cs: screen space outline will get stripped when building for mobile(android / iOS)
- (Demo) upgrade demo project to Unity 2020.3.12f1
- (Demo) Upgrade demo project to URP 10.5

Fixed
- (Core) NiloToonEditor_AssetLabelAssetPostProcessor: handle KeyNotFoundException: The given key was not present in the dictionary. when reimporting a model. now will produce a warning message for tracking which model will produce this error

---

[0.2.4] - 2021-6-23

Added
- (Core)NiloToonPerCharacterRenderController: Add "attachmentRendererList" for user to attach any other renderer to the current character, these attachment renderers will use that NiloToonPerCharacterRenderController script's setting(e.g. sync weapon/microphone's perspective removal with a character)

Changed
- (Core)NiloToonPerCharacterRenderController: better ToolTip
- (Core)GenericStencilUnlit shader: rewrite to support attachmentRendererList and SRP batching

Fixed
- (Core)NiloToonPerCharacterRenderController: handle null renderer

---

[0.2.3] - 2021-6-21

Added
- (Core)Character shader: Add "Override Shadow Color by texture" section
- (Core)Character shader: Add "Override Outline Color by texture" section

Changed
- (Core)Character shader: better material GUI

Fixed
- (Core)Character shader: GGX specular use float instead of half to avoid precision problem on mobile platform
- GenericStencilUnlit.shader support SRP batching and VR

[0.2.2] - 2021-6-16

Changed
- (Core)Character shader: better material GUI's note

Fixed

- (Core)Character shader: fixed a bug which makes SRP batching not working correctly
- (Core)Character shader: fixed a bug which makes Detail albedo not used by specular amd emission

---
[0.2.1] - 2021-6-15

Breaking Changes
- (Core)(IMPORTANT)Character shader: not using all roughness related settings in "Specular" section anymore, add a new shared "Smoothness" section. This change is made to make NiloToon character shader matches URP Lit.shader's smoothness data convention, and now "Smoothness" section's data can be shared/used by multiple features such as "Environment reflection" and "Specular(GGX)". ***If you are using "Specular" section's roughness setting in your project's materials already, after this update you will have to set up it again in the new "Smoothness" section***
- (Core)(IMPORTANT)Character shader: "Matcap Mask" section merged into "MatCap(alpha blend)" and "MatCap(additive)" section. This change is made due to the old design will force user to combine 2 mask textures into 1 texture, which is not flexible enough. This change will break old materials's "Matcap Mask" section. ***If you are using "MatCap mask" section's settings in your project's materials already, you will have to set up it again in the "MatCap(alpha blend)" and "MatCap(additive)" section's Optional mask setting***
- (Core)rename class NiloToonPerCharacterRenderControllerOverrider to NiloToonCharacterRenderOverrider, to avoid confusion when adding a NiloToonCharacterRenderOverrider script to GameObject
- (Core)rename shader internal MACRO NiloToonIsOutlinePass to NiloToonIsAnyOutlinePass, you can ignore this change if you didn't edit NiloToon's shader code
- (Core)rename shader internal MACRO NiloToonIsColorPass to NiloToonIsAnyLitColorPass, you can ignore this change if you didn't edit NiloToon's shader code

Added
- (Core)(IMPORTANT): NiloToonAllInOneRendererFeature and NiloToonShadowControlVolume added a new toggle "useMainLightAsCastShadowDirection", you can enable it if you want NiloToon's self shadow system use scene's MainLight direction to cast shadow(same shadow casting direction as regular URP main light shadow, which means shadow result will NOT be affected by camera rotation/movement)
- (Core)Add GenericStencilUnlit.shader, useful if you want to apply stencil effects that uses drawn character pixels as a stencil mask
- (Core)Character shader: in "Specular" section, add _MultiplyBaseColorToSpecularColor slider, useful if you want to mix base color into specular result
- (Core)Character shader: in "Specular" section, add "Extra Tint by Texture" option, useful if you want to mix any texture into specular result
- (Core)Character shader: in "Emission" section, add _EmissionIntensity, _MultiplyBaseColorToEmissionColor, to allow more Emission color control
- (Core)Character shader: add ColorMask option (RGBA or RGB_), useful if you don't want to pollute RenderTexture's alpha channel for semi-transparent materials
- (Core)Character shader: MatCap(additive) add _MatCapAdditiveMaskMapChannelMask and _MatCapAdditiveMaskMap's remap minmax slider
- (Core)Character shader: MatCap(alpha blend) add _MatCapAlphaBlendMaskMapChannelMask and _MatCapAlphaBlendMaskMap's remap minmax slider
- (Core)Character shader: MatCap(alpha blend) add _MatCapAlphaBlendTintColor and _MatCapAlphaBlendMapAlphaAsMask
- (Core)Character shader: add a new "Environment Reflections" section
- (Core)Character shader: in "Lighting Style" section,added _IndirectLightFlatten, to allow more control on how to display lightprobe result
- (Core)Character shader: in "Outline" section, added _UnityCameraDepthTextureWriteOutlineExtrudedPosition, you can disable it to help removing weird 2D white line artifact on material
- (Core)Character shader: in "Can receive NiloToon Shadow?" section, add _NiloToonSelfShadowIntensityForNonFace(Default 1) and _NiloToonSelfShadowIntensityForFace(Default 0). Face can receive NiloToon's self shadow map now.
- (DEMO)Add MMD4Mecanim folder
- (DEMO)Add some MMD model for testing(only exist in project window)
- (DEMO)Bike Mb1MotorMd000001 add a prefab variant for showing "Environment Reflection" material

Changed
- (Core)Character shader: hide stencil option (stencil options are not being used in all versions)
- (Core)Sticker shader: use ColorMask RGB now, to not pollute RT's alpha channel
- (Core)NiloToonAnimePostProcessPass set ScriptableRenderPass.renderPassEvent = XXX directly, internally NiloToonAllInOneRendererFeature don't need to create 2 renderpass now
- (Core)Internal shader code big refactor without changing visual result, if you didn't edit NiloToon's shader source code, you can ignore this change
- (DEMO)Update model materials (IsSkin and Smoothness)
- (DEMO)CRS scene use main light as NiloToon self shadow casting direction (enable useMainLightAsCastShadowDirection in NiloToonSelfShadowVolume)

Fixed
- (Core)Character shader: fixed a bug where _OcclusionStrength and _GlobalOcclusionStrength is not applied in a correct order
- (Core)Debug window: Fix a bug where NiloToon Debug window always wrongly focus project/scene window
- (Core)Anime postProcess shader: Fix Hidden/NiloToon/AnimePostProcess produce "framgent shader output doesn't have enough component" error in Metal graphics API

[0.1.3] - 2021-5-19

Breaking Changes
- (Core)Change namespace of all NiloToonURP C# script to "using NiloToon.NiloToonURP;", please delete your old NiloToonURP folder first before importing the updated NiloToonURP.unitypackage
- (Core)Change shader path of Character shader to "Universal Render Pipeline/NiloToon/NiloToon_Character"
- (Core)Change shader path of Sticker shader to "Universal Render Pipeline/NiloToon/xxx"

Known Issues
- (Demo)Planar reflection in CRS scene (VR mode) is not correct

Added
- (Core)Character Shader: Add "IsSkin?" toggle in material, you can enable this toggle if a material is skin(hand/leg/body...), enable it will make the shader use an optional overrided shadow color for skin, optional skin mask can be enabled also if your material is a mix of skin and cloth
- (Core)Character Shader: Add _OverrideByFaceShadowTintColor,_OverrideBySkinShadowTintColor,_OutlineWidthExtraMultiplier
- (Core)Character Shader: Add _ZOffsetEnable to allow on off ZOffset by a toggle
- (Core)Character Shader: Add _EditFinalOutputAlphaEnable to allow on off EditFinalAlphaOuput by a toggle
- (Core)Character Shader: Add _EnableNiloToonSelfShadowMapping to allow on off NiloToonSelfShadowMapping by a toggle
- (Core)Character Shader: Add _NiloToonSelfShadowMappingDepthBias to allow edit NiloToonSelfShadowMapping's depth bias per material
- (Core)Character Shader: Add RGBAverage and RGBLuminance to RGBAChannelMaskToVec4Drawer, you will see it if you click the RGBA channel drop list in material UI
- (Core)Sticker Shader: Add override alpha by a texture (optional)
- (Core)Correctly support Unity 2019.4.0f1 or above (need to use URP 7.4.1 or above, NOT URP 7.3.1, you can upgrade URP to 7.4.1 in the package manager)
- (Core)Add NiloToonPlanarReflectionHelper.cs, for planar reflection camera support, you need to call NiloToonPlanarReflectionHelper.cs's function in C# when rendering your planar reflection camera (see MirrorReflection.cs in CRS demo scene), user document has a new section about it, see "When rendering NiloToon shader in planar reflection camera, some part of the model disappeared"
- (Core)Add NiloToonPerCharacterRenderControllerOverrider.cs, to sync perspective removal result from a source to a group of characters
- (Core)Add SimpleLit debug option in NiloToon debug window
- (Core)NiloToonAllInOneRendererFeature: Add useNdotLFix to hide self shadowmap artifact, you can turn it off if you don't like it
- (Core)NiloToonCharRenderingControlVolume: + depthTextureRimLightAndShadowWidthMultiplier
- (Core)NiloToonPerCharacterRenderController: + perCharacterOutlineWidthMultiply , perCharacterOutlineColorTint
- (Demo)Add CRS(Candy rock star) demo scene, with a planar reflection script(MirrorReflection.cs)
- (Demo)Add more demo models
- (Demo)Add .vrm file auto prefab generation editor script, inside ThirdParty(VRM) folder. You can drag .vrm files into demo project and prefab will be generated
- (Demo)Add 4ExtremeHighQuality and 5HighestQuality quality settings, PC build now default use 4ExtremeHighQuality
- (Doc)Add section for how to correctly use NiloToonURP in 2019.4 (need to enable URP's depth texture manually and install URP 7.4.1 or above)
- (Doc)Add section for setting up semi-transparent alpha blending material
- (Doc)Add section for changing VRM(MToon)/RealToon material to NiloToon material

Changed
- (Core)remove _vertex and _fragment suffixes in multi_compile and shader_compile, in order to support 2019.4 correctly
- (Core)Character Shader: Now the minimum supported OpenGLES version is 3.1, not 3.0, due to support SRP Batching correctly
- (Core)package.json: Now the minimum supported URP version is 7.4.1, not 7.6.0
- (Core)package.json: Now the minimum supported editor version is 2019.4.0f1, not 2019.4.25f1
- (Core)NiloToonPerCharacterRenderController.cs: now auto disable perspective removal in XR
- (Core)Smooth normal editor baking: Will skip baking if model don't have correct tangent, and better import error message if model don't have tangent data
- (Core)if _ZWrite = 0, disable all _CameraDepthTexture related effects (e.g.auto disable depth texture 2D rim light of a ZOffset enabled eyebrow material)

- (Core)now use global keyword SHOULD_STRIP_FORCE_MINIMUM_SHADER in demo's enable NiloToon toggle
- (Demo)change demo script to use namespace using NiloToon.NiloToonURP;
- (Demo)now limit Screen height to maximum 1080, to increase fps in 4k monitors
- (Demo)optimize some model's texture max resolution for android, to avoid using too much memory in .apk
- (Doc)improve user document with more FAQ

Fixed
- (Core)fix SRP batcher mode linear gamma bug
- (Core)fixed NiloToonCharacterSticker shaders return alpha not equals 1 problem, these shader will not pollute RT's alpha channel anymore (always return alpha == 1)
- (Core)fixed some shader and C# warning (no harm)
- (Core)fixed a bug that make SRP batcher not working (add _PerCharacterBaseColorTint in NiloToonCharacter.shader's Properties section)

[0.1.2] - 2021-5-3

Added
- (Core)Add NiloToonShadowControlVolume for Volume, you can use it to control and override shadow settings per volume instead of editing NiloToonAllInOneRendererFeature directly
- (Core)All shaders add basic XR support, but many shader features are now auto disabled in XR in this version temporarily, because we are fixing them
- (Core)In XR, due to high fov, outline width is default 50% (only in XR), you can change this number in NiloToonCharRenderingControlVolume or NiloToonAllInOneRendererFeature
- (Core)character shader: Add DepthNormal pass, URP's SSAO renderer feature(DepthNormals mode) will work correctly now
- (Core)character shader: Add "screen space outline" feature in material (experimental, need URP's _CameraDepthTexture enabled), enable it in material UI will add more detail outline to character, useful for alphaclip materials where the default outline looks bad
- (Core)character shader: Add "Dynamic eye" feature in material, for users who need circular dynamic eye pupil control
- (Core)character shader: Add VertExmotion support, you can enable it in NiloToonCharacter_ExtendDefinesForExternalAsset.hlsl
- (Core)character shader: Add _ZOffsetMaskMapChannelMask,_ExtraThickOutlineMaxFinalWidth,_DepthTexShadowThresholdOffset,_DepthTexShadowFadeoutRange,_GlobalMainLightURPShadowAsDirectResultTintColor in material
- (Core)NiloToonAnimePostProcessVolume: add anime postprocess effect draw height and draw timing control
- (Core)Add per character and per volume extra BaseColor tint
- (Core)Can install from PackageManager(install from disk) using NiloToonURP folder's package.json file
- (Core)Add auto reimport and message box when using NiloToonURP the first time
- (Demo)All demo shader add XR support
- (Demo)All scene add steam XR support, see user document pdf for instruction on how to try it in editor play mode (steam PCVR)
- (Demo)Add NiloToonDemoDanceScene_UI.cs, adding more user control in NiloToonDancingDemo.unity

Changed
- (Core)NiloToonAnimePostProcess: Reduce default top light intensity from 100% to 75%
- (Core)improve UI display and tooltips
- (Demo)player settings remove Vulkan API for Android, now always use OpenGLES3 to support testing on more devices
- (Demo)UnityChan edit skin material to have better shadow color(in 0.1.1 shadow color is too dirty and grey)
- (Doc)improve user document with more FAQ

Fixed
- (Core) Fixed bake smooth normal UV8 index out of bound exception if .fbx has no tangent, will now produce error log only
- (Core) Fixed NoV rimlight V matrix not corerct bug (now use UNITY_MATRIX_V instead of unity_CameraToWorld)
- (Core) Fixed a missing abs() bug in NiloZOffsetUtil.hlsl
- (Demo) Fixed multi AdvanceFPS script in scene bug

[0.1.1] - 2021-4-19

Added
- (Core)Add change log file (this file)
- (Core)character shader: Add "MatCap (blend,add and mask)" section, which mix MatCap textures to Base map
- (Core)character shader: Add "Ramp lighting texture" section, which override most lighting params by a ramp texture
- (Core)character shader: Add "Final output alpha" section, to allow user render opaque character to custom RT without alpha problem
- (Core)character shader: "Detail Maps" section add uv2 toggle and albedo texture white point slider
- (Core)character shader: "Calculate Shadow Color" section add _FaceShadowTintColor

- (Core)character shader: "Calculate Shadow Color" section add _LitToShadowTransitionAreaTintColor, LitToShadowTransitionArea HSV edit
- (Core)character shader: "Outline" section add _OutlineOcclusionAreaTintColor, _OutlineReplaceColor
- (Core)character shader: "Depth texture shadow" section add _DepthTexShadowUsage
- (Core)character shader: NiloToonCharacter_ExtendFunctionsForUserCustomLogic.hlsl add more empty functions for user
- (Core)character shader: Add support to VertExmotion asset (no change to NiloToon shader name) (add NiloToonCharacter_ExtendDefinesForExternalAsset.hlsl for user to enable VertExmotion support if they need it)
- (Core)volume: NiloToonCharRenderingControlVolume.cs add Directional Light, Additional Light, Specular's volume control to allow better control light intensity on character
- (Core)character root script: NiloToonPerCharacterRenderController.cs Add perCharacterDesaturation
- (Core)character root script: NiloToonPerCharacterRenderController.cs Add extra think outline view space pos offset, usually for stylized color 2D drop shadow
- (Demo)Add GBVS Narmaya model and setup her using NiloToon (face material not ready for 360 light rotation)
- (Demo)Add NiloToonCutinScene.unity to show GBVS Narmaya model
- (Demo)Add bike (using NiloToonURP) in NiloToonSampleScene.unity
- (Demo)Include XR related files, preparing for future XR support
- (Demo)Add 4HighestQuality, it will now ignore performance limit, using the best possible setting (only for PC build, WebGL and editor)

Changed
- (Core)NiloToonAllInOneRendererFeature: Unlock self shadow's shadow map resolution limit from 4096 to 8192
- (Core)NiloToonAverageShadowTestRT.shader: add sampling pos offset to prevent over generating average shadow due to near by objects/characters
- (Core)revert indirectLightMultiplier's default setting from 2 to 1
- (Demo)Upgraded dmeo project to 2020.3.4f1
- (Demo)Optimize 0~3 quality settings for mobile, while allow best quality(4) for PC/Editor/WebGL
- (Demo)Allow every scene to switch to another in play mode, no matter user start playing in which scene
- (Demo)Refactor and rename some files and folder

Fixed
- (Core)character root script: NiloToonPerCharacterRenderController.cs fix can't edit material in pause play mode bug

Know Issues
- (Core)Depth texture rim light is not correct in PCVR Editor
- (Core)Shader variant(memory usage) too much in mobile build
- (Core)Outline render incorrectly if camera is too close
- [DONE in 0.3.2] (Core)per character scirpt's editor script is too slow
- (Demo)Switching scene in editor play mode takes a very long time, due to editor script

TODO
- (Core)Finish detail document on material properties, NiloToon UI params
- (Core)Charcter shader can't receive point or spot light shadow (need to support URP11 first, since point light shadow only exist in URP11)
- [DONE in 0.1.2] (Core)depth texture sobel outline shader_feature
- [DONE in 0.1.2] (Core)control URP shadow intensity remove directional light contribution (as volume)
- (Demo)Add moving tree shadowmap demo scene
- (Demo)GBVS Narmaya's face material not yet setup correctly to support all light direction

---

[0.0.3] - 2021-4-06
Added
- (Demo)Add MaGirlHeadJsonTransformSetter.cs
- (Demo)Add random face & ear animation for MaGirl using MaGirlHeadJsonTransformSetter.cs

---

[0.0.2] - 2021-4-03
Added
- (Demo)Add MaGirl3.prefab and it's variants, setup her using NiloToon
- (Demo)Add MaGirl2.prefab and it's variants, setup her using NiloToon
- (Demo)Add NiloToonDancingDemo.unity scene

[0.0.1] - 2021-3-28
Added
- (Core)First version to record in change log.
- (Demo)First version to record in change log.