Sleek Render

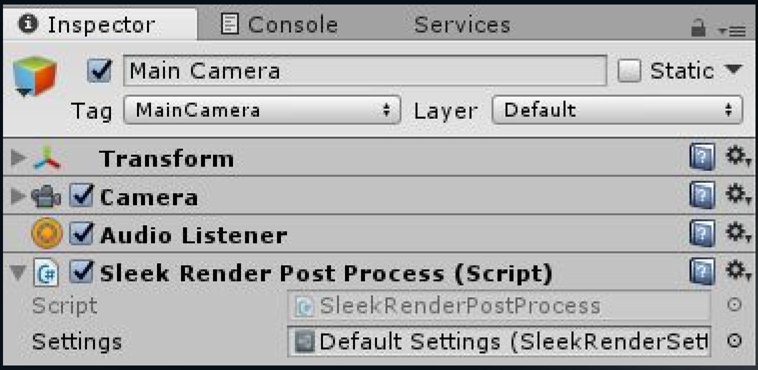
v 0.3

Thanks for using Sleek Render.
Please read the setup and usage part carefully.

1. Setup

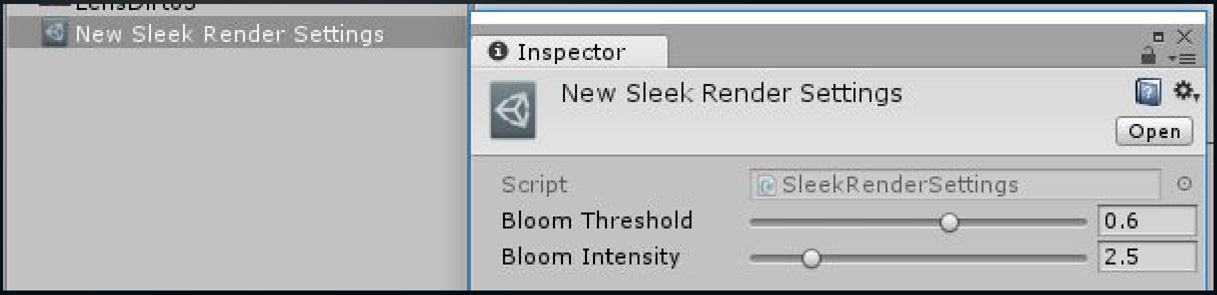
Steps are simple:

1. Add “*SleekRenderPostProcess*” component to your scene camera.



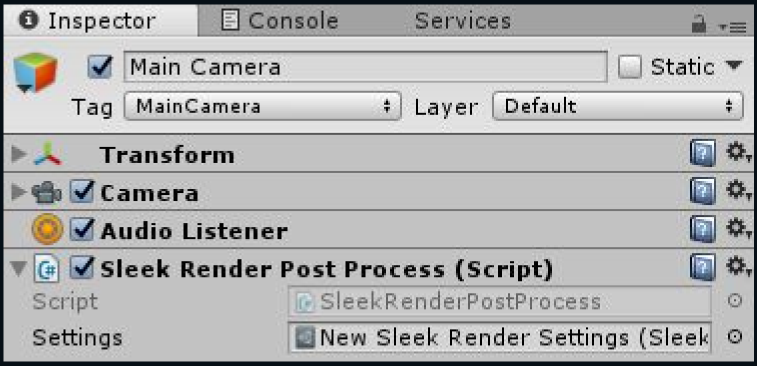
Main Camera
Tag: MainCamera
Layer: Default
Script: SleekRenderPostProcess
Settings: Default Settings (SleekRenderSett

2. Create a Sleek Render Settings asset in the Project View. To do it, right click anywhere inside the Project View in Unity, and choose Create -> Sleek Render Settings. This will create a settings asset where you can tweak the post process rendering parameters.



New Sleek Render Settings
Script: SleekRenderSettings
Bloom Threshold: 0.6
Bloom Intensity: 2.5

3. Link this newly created asset to a “*SleekRenderPostProcess*” component “*Settings*” field.



Main Camera
Tag: MainCamera
Layer: Default
Script: SleekRenderPostProcess
Settings: New Sleek Render Settings (Sleek

4. Done. You’re awesome. Now let’s tweak some parameters.

2. Bloom

Bloom settings include:
1. Bloom Threshold
2. Bloom Intensity

Bloom Threshold sets the brightpass parameters. It shows how bright or dim should be the light to be picked up by the bloom effect. Bright pass is not binary, that is - it's not just black-or-white after the step, it gradually decreases the bloom brightness based on the pixel luminance

Bloom Intensity sets the additive strength of the bloom. The higher the value, the more bright the bloom will appear.

3. HDR Compression

Bloom can be too bright sometimes, creating spots of plain white color. This package includes a simple HDR Compression solution that allows to compress those highlights a bit. The settings include:
1. Gamma Compression Power
2. Hdr Max Intensity

The formula used to calculate this is as follows:

$$V_{out} = AV_{in}^{\gamma}$$

Where A is Hdr Max Intensity and γ is Gamma Compression Power. Most likely you'll be good to go with the default values.

4. Color overlay

Sleek render has a very useful and versatile color grading tool which is essentially free to calculate. It's not based on LUTs, so color overlay will appear uniform over the whole image.
The Colorize parameter sets the Color Overlay (RGB) and it's Intensity (Alpha). The more alpha the color has, the more apparent the colorize effect becomes. Zero alpha means no color grading will be visible.

Common usage patterns of Color Overlay are:

1. **Desaturate**. Set color to WHITE and set Alpha to 1 (fully desaturated grayscale image) or somewhere in between (less alpha - more saturation).
2. **Colorize**. Set color to desired value and tweak Alpha to make it more or less apparent.
3. **Fade In / Fade Out**. Color Overlay effect can be used to greatly reduce common fillrate wasted on Fade Out - Fade In effects. Rather than making a fullscreen black UI Texture and tweening it's alpha, you can set color to BLACK and tweak or tween Alpha from 0 to 1 to Fade Out or from 1 to 0 to fade out. It will create the same fade effect at virtually zero cost without any wasting any additional fillrate.

4. Vignette

Vignette is an effect in photography or filmmaking where corners and edges of the image are slightly darkened or painted with some color. It can help to focus the viewer on important things in the scene and adds a bit of realism to the picture.

Vignette parameters are:

1. Vignette Begin Radius - distance on which the vignette begins (larger radius - vignette effect starts closer to edges/corners)
2. Vignette Expand Radius - vignette gradient shape and size. (larger radius - softer vignette gradient)
3. Vignette Color - pretty self-explanatory. Black color with full alpha will darken the vignetted parts of the image completely.

5. Script control

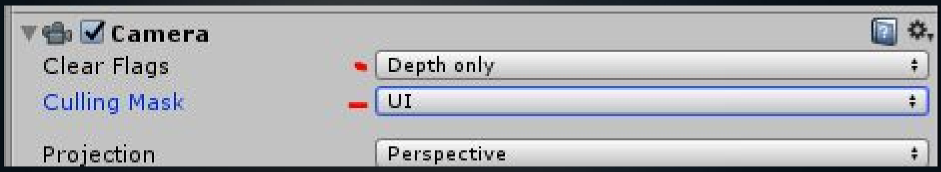
To edit post process parameters via scripts, add “*using SleekRender*” line at the top of your script, create a “*SleekRenderSettings*” public field. Then, link the same settings asset that is used to control the effects on your main camera. After that, changing parameters is done simply by accessing and changing public fields of the settings asset.

6. UI Canvas with *Screen Space - Camera* setting.

While uGUI works well with **Screen Space - Overlay** and **World Space**, Unity has some specific limitations for UI rendering with **Screen Space - Camera** mode if the Camera is the same as the one that is used for Sleek Render post-effects.
As for now, Unity doesn't allow custom UI rendering calls and thus UI will not be rendered in this case. The solution is:

1. Add another camera with the same FOV settings
2. Set it's rendering layers to UI
3. Set it's clear flags to **Depth Only**.

By default this camera will be rendered after the main one and the UI will be visible. Alternatively you can tweak the Depth property to move it "forward" or "backward".
This solution doesn't incur any noticeable performance penalties and is absolutely appropriate for mobile devices.



Camera
Clear Flags: Depth only
Culling Mask: UI
Projection: Perspective

7. Support

For any questions related to this package, feel free to write to:
- Unity forum thread https://goo.gl/pzgt3p
- support email support@nadezhdin.org