Pre-Integrated Skin Shader for Unity3D

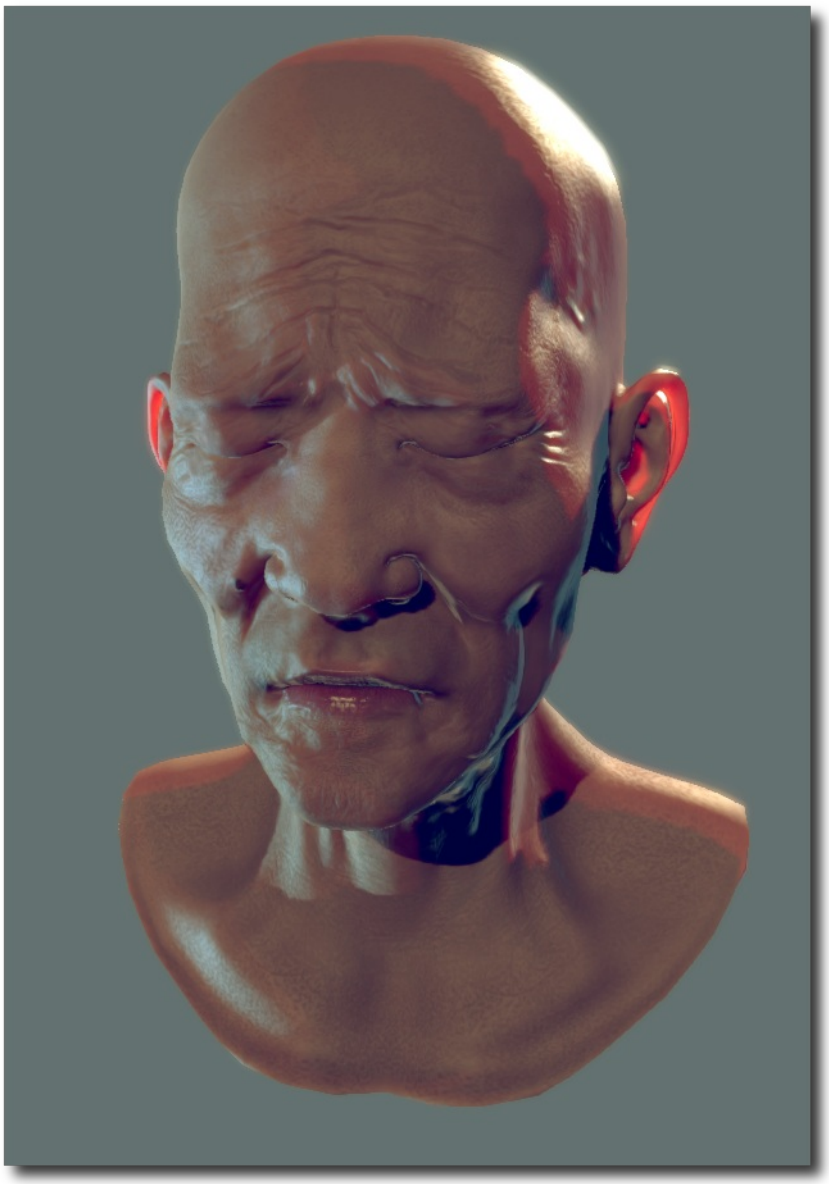
Version 2.0

Description

Realistic skin shader with pre-integrated physically based profiles.

Main features include: - Pre-integrated diffuse scattering using configurable diffusion profiles.
- Artist friendly setup and easy tweaking with sliders.
- Physically based specular lighting and Unity 5 reflection probes.
- Per pixel Sherical harmonics lighting with sub-surface-scattering. This includes light probes and Unity 5's real time Global Illumination and Image Based Lighting.
- Translucency/back-scattering.
- Shadow penumbra scattering.
- Ambient occlusion with sub-surface scattering.
- ~~Direct3D 11 tessellation with phong smoothing and/or displacement map (windows only due to Unity limitations).~~ DISABLED AS OF 2.0, PLANNED FOR 2.1
- Shader Model 2.0 fallback (limited, see the manual for details).
- Optimized and scalable.
- Full source code with comments. Per request access to git repository, online documentation and issue tracker.

Example scene included. See the Web player demo before deciding. TODO a webgl demo? or at least update this one.



Contents

Questions, Ideas & Support

Check out the official forum thread or contact me directly.

Licensing

This file is provided under standard Unity Asset Store EULA: http://unity3d.com/company/legal/as_terms See the FAQ for some tips.

Beta testing and git repository access

If you already bought the asset on the asset store and either want to participate in beta testing, or just follow the development more closely, you can get access to complete git repository and issue tracking here.

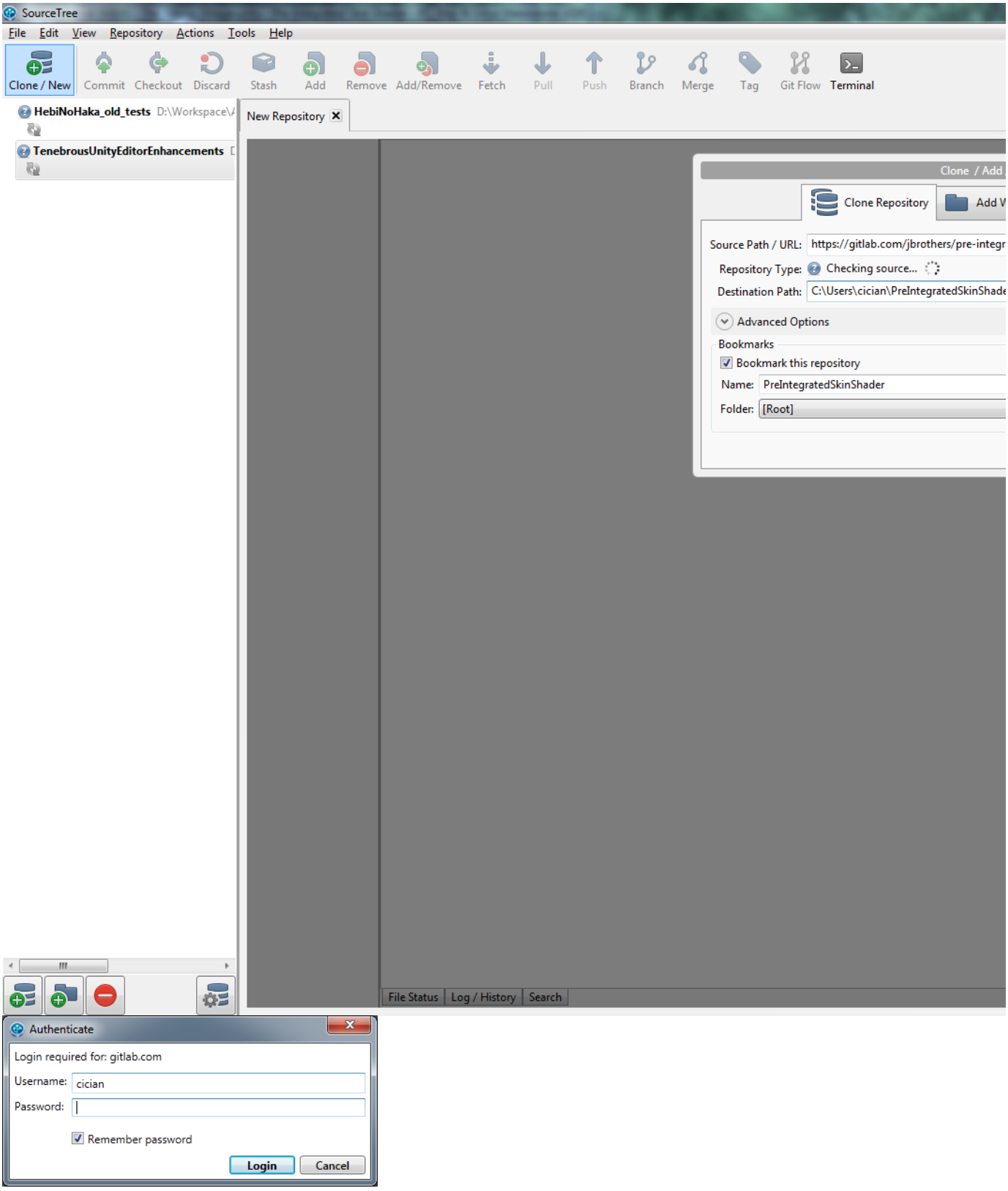
- First you have to register on gitlab.com, if you haven't done so already. It's totally free and similar to github or bitbucket.
- Then you can send me your gitlab username and Pre-Integrated Skin Shader invoice (either pdf or just the invoice number).
- Then I'll give you access to the repository, which you can access here.

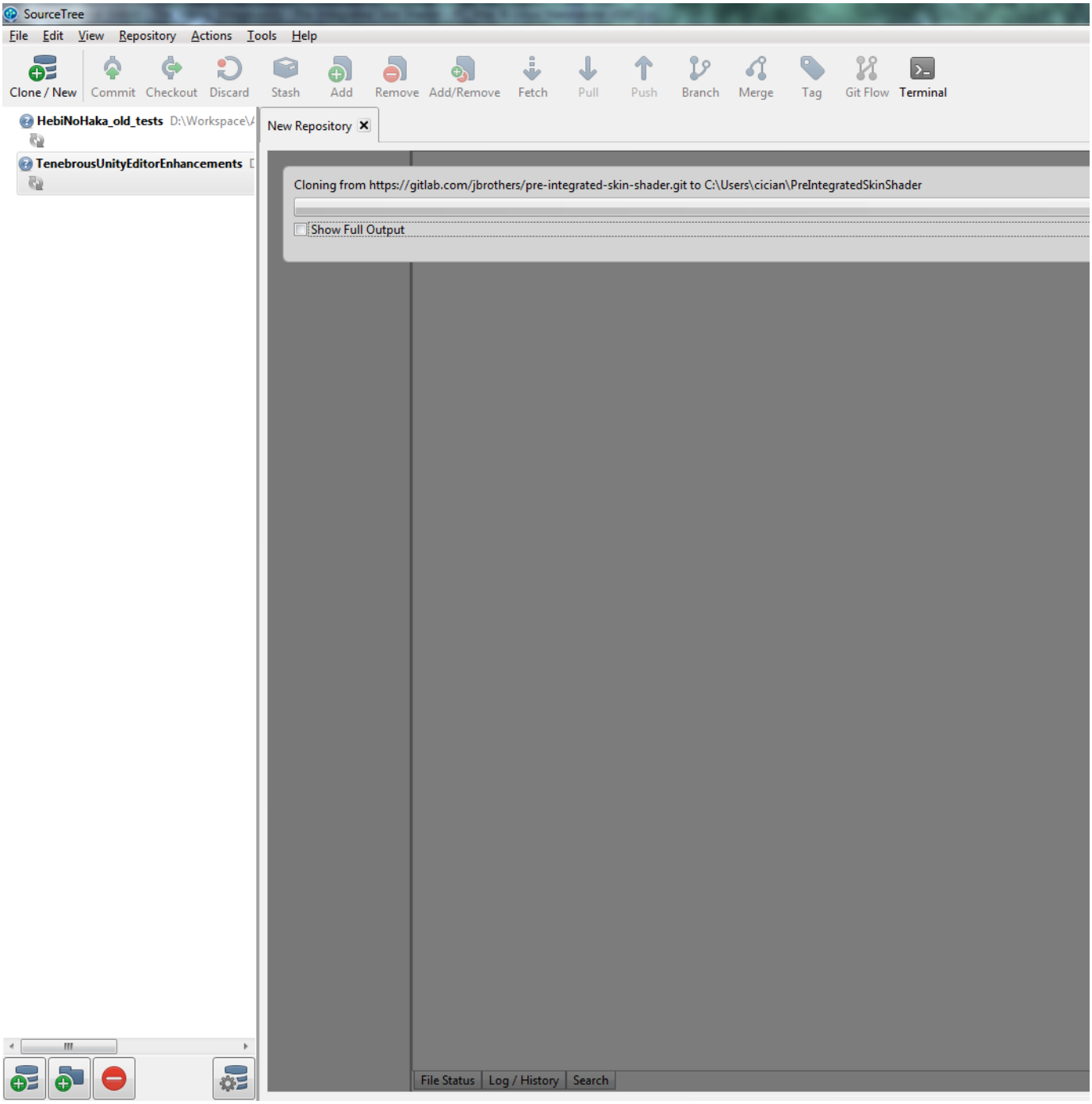
Disclaimer

When downloading from git repository keep in mind that it's the code I may be actively working on. Things may brake occasionally and features may get added or removed based on my mood swings.

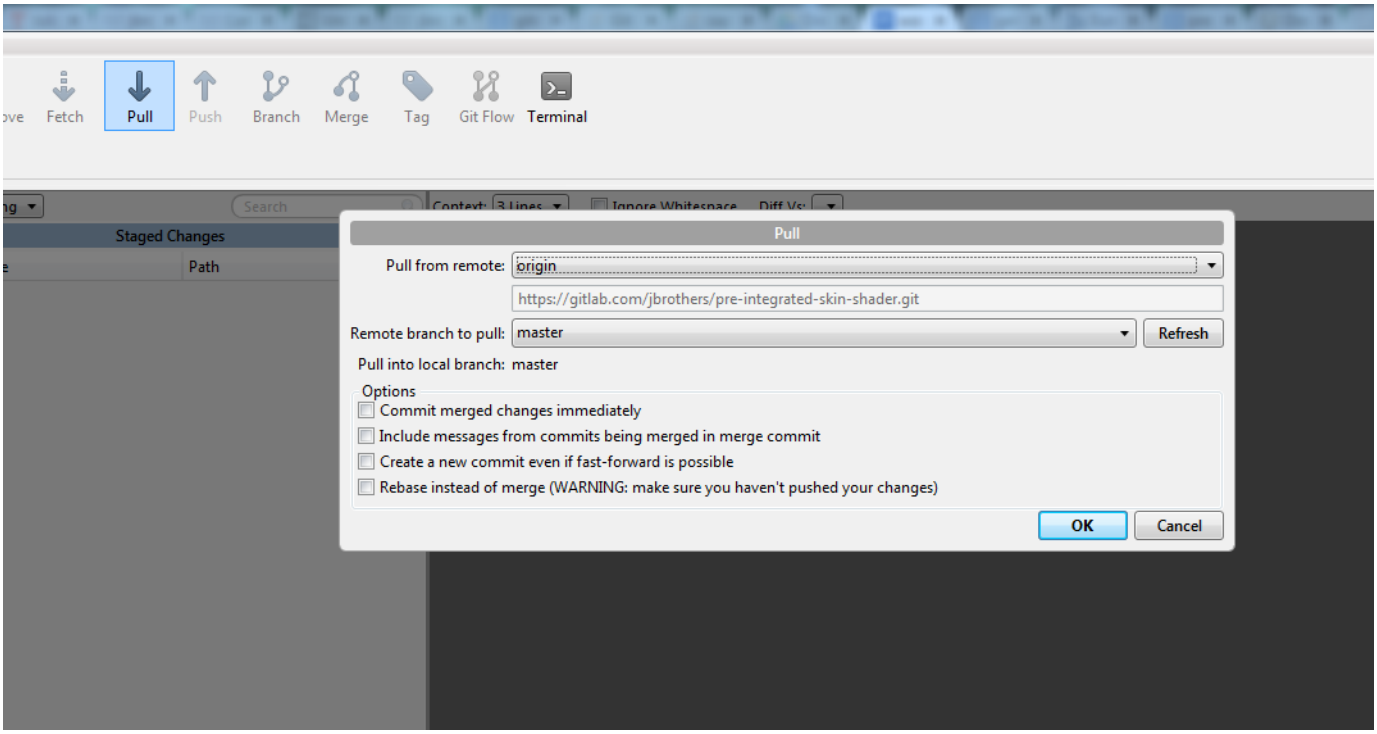
Downloading from git repository

If you're not a programmer or otherwise don't have any git experience, don't panic! All you need to do is to "clone" the repository. That just means downloading all the files together with the development history. First you need a git client. For example source tree. Install it, and then, perform a clone. You just need to provide the repository url, https://gitlab.com/jbrothers/pre-integrated-skin-shader.git, your gitlab account and password, and the local path to download. Then relax for a bit, since it can take a while.

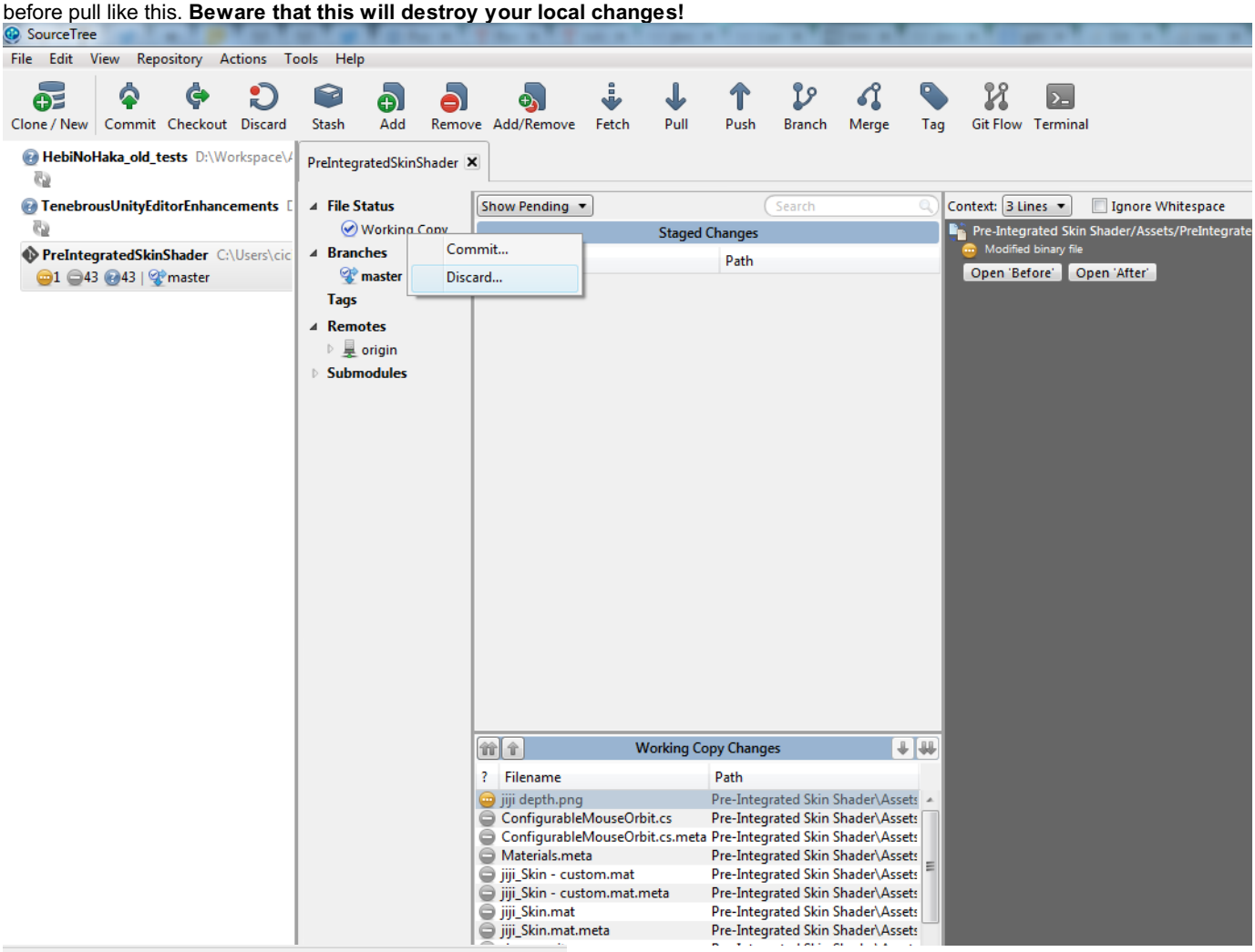


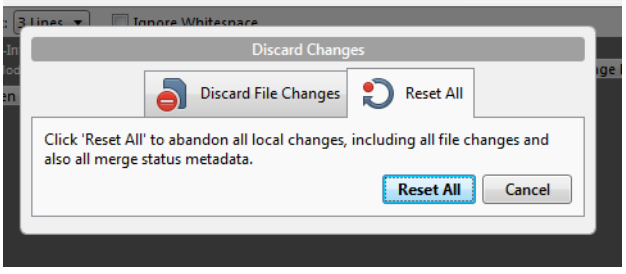


To download the latest changes you "pull" them.



If you don't have any local changes this should work flawlessly. Otherwise, inevitably you'll get conflicts and the pull command will fail. At this stage, you either deep dive into git and learn how to manage conflicts, or just hard-reset your local changes before pull like this. **Beware that this will destroy your local changes!**





If you have any problems, let me know so I can improve this little guide.

Reporting issues

If you're already registered on gitlab and granted "reporter" access to the repository you can fill an issue directly. Otherwise send me a mail or write on the official forum thread.

Frequently Asked Questions (or ¿THA FAQ?!)

I thought your shader would take care of rendering the photo-realistic skin, but I see all these empty texture slots and parameters. Do I need to create and set them all?

I'm sorry to inform you that this shader doesn't do magic and you have to create all these textures in traditional way. My sorcery level hasn't reached this level just yet ;)
See preparing necessary content here.

I just bought the shader, but it doesn't work!

Please provide some details about the error. Look in the console for relevenat errors, but skip the ones saying "Material doesn't have a color property.."
If you lost the message right click on the shader file and click reimport.
Then send me a mail or post on the official forum thread.

Can I publish a free or commercial game with this shader?

YES!

Can I publish a FLOSS game/program with this shader?

Nope, sorry.

Can I publish a character on the asset store with this shader bundled?

No, but we can make a deal. Otherwise you can provide the preconfigured material and link to the shader on asset store. Please vote here for this to be possible without cutting me out of direct sales.

Why did it take so long for 2.0 update?

There are a few reasons: It was almost a rewrite of 1.0, I have a day job to put food on the table and things happened in my life. But the truth is these are all excuses and I'm a big procrastinator :P

Does it work on console platforms?

I don't know. I really don't. I don't even have a way to test it, since the dev kits are way out of my reach. If someone have some info on how Unity handles shaders on consoles let me know.

Credits

This shader in major part is an implementation of techniques described in Eric Penner's publictly available Siggraph slides. Much theory and some code in earlier versions are taken from Nvidia's GPU Gems 3 (D'Eon et al.). These two were my bibles for a while.

A bit too late I discovered the *Photo - Realistic Real - time Face Rendering* paper by Daniel Chappuis, that describes much of what is in the shader and more. In particular the spherical harmonics work.

The shader incorporates code to calculate transmittance by Jorge Jimenez et al.

Lookup texture generation code was initially based on code published on Gamasutra by Jon Moore.

Thanks to those mentioned above and to my brothers Karol and Jerzy for providing me support, demo content, testing, help with releases and this manual. Thanks also to folks at Infinite Realities for the freely available head scan that served me as initial testing material and ten24.com for another free head scan.

Version change-log

1.0.0

Initial upload to the Asset Store.

1.0.1

Cleaned up the project and essentially recreated it in Unity3.5 to make it available for everyone (required 4.0 before).

1.1.0

- Shader Model 5 version now compiles on Unity3 and fallbacks to Shader Model 3 as expected. This way you can keep the file in the project and it gives you possibility to make a single shader with embedded subshaders.
- Changed tessellation mode to one more optimal for skin where the outline is more tessellated than interior of the mesh. This is implemented using face normal angle compared to view direction.
- Specular, Glossiness and depth maps are now separate textures. This adds a little overhead, but is easier to use and more flexible. This has potential to introduce problems on old hardware and mobile. For your final game it's recommended to create a shader with customized texture setup. See example customized shaders. Contact me if you want me to make one for you ;) Since this breaks existing materials I include the old version as well.
- Added optional Skin Mask texture. This allows to mask areas which should not render as skin, for example under hair. The masked-out areas still use the same specular model, but are not affected by SSS and rim. This should not be abused. If possible you should use a different material for non skin parts of your characters. This just allows blending in fine details.
- Internally added more preprocessor flags for customization. Abstracted texture names and channels for various functions with compiler defines to maximize flexibility.
- Internally exchanged specular and gloss field as it's not necessary to keep Unity's broken nomenclature when using a custom lighting model.
- New Shader Model 2 version that supports translucency and rims. It uses a second pass to surpass the SM2 instructions limit and is thus up to 100% slower. It's much better fallback than the single-pass SM2 version because it results in less visible differences. Some limitations in respect to SM3 still apply: no separate diffuse normals (bumpiness sliders); no full forward shadows (no shadows from point and spotlights, only from single directional light);
- Adjustable fresnel values for specular. One for skin and one for masked regions.
- Changed a bit how the translucency works, potentially changing the look of existing materials. Now the script uses a texture lookup for translucency. This was necessary to make it work on SM2.
- Added an optional translucency color texture and overall translucency color for tweaking it. This is NOT what you'd expect from a traditional subdermal map. It allows for subtle details like veins visible only on translucent areas.
- Fixed the texture lookup script(s) to import with relative path so the settings take effect immediately.
- Reduced specular slider range from 0-100 to 0-3 for better precision/strength trade-off.
- Changed bumpiness sliders to 0-2 range to allow boosting sharpness a little
- Added smoothness sliders for rims. They were too high contrast and reducing strength was only making them bland and not smooth.
- Avoid saturating specular for better HDR rendering

1.1.1

Mostly a [partial] hotfix for Unity 5:
- Fixed rendering artifacts caused by Unity 5 having changed surface shader lighting to world space.
- Adapted the demo scene's camera script for API changes in Unity 5. Removes ugly warnings.
- Added a workaround for Unity 4 bug causing flickering on some OpenGL platforms due to Z buffer imprecision.
- Lookup textures are now set by default for new materials.
- Bumped required Unity version to 4.3. Though it still works just fine in Unity 3.5, v2.0 won't. Nothing fundamental changed from version 1.1.0. This is a stop-gap solution before version 2.0, which brings proper Unity 5 support.

2.0.0

An almost rewrite in the form of a CG/HLSL shader instead of a Surface Shader to take advantage of some features not accessible otherwise and to better control and optimize at lower level.
- Proper support for Unity 5.
- Explicit penumbra scattering for a nice reddish gradient. Old version did some scattering with a cheap trick, but it was pretty bad. Penumbra now uses shadow penumbra separate from light attenuation at low level. It's not exact in any way and limited by Unity's shortcomings in the shadow map implementation, but still much better than before.
- Lighting is now done in world space. Shouldn't change anything for end users. This change is mostly for IBL and light probe ground work.
- Customizable sub-surface-scattering diffusion profiles. Other than fine tuning what was hard-coded in the shader before, this allows representing diverse materials, like aliens, monsters, zombies and even other non skin semi-translucent surfaces like marble.
- Lookup texture generation rewritten in shader form to take advantage of GPU parallelism. This was primarily for my own benefit during development.
- Lookup textures are now independent of diffusion profile so one can tweak it in real time.
- Per pixel spherical harmonics lighting. Both light probes and Unity 5's GI now have diffuse scattering, translucency, ambient occlusion and even rim lighting.
- "Non important" light over the limit in quality settings that Unity approximates to 4 point lights are now rendered as part of spherical harmonics, per fragment and with sub-surface scattering.
- Overall the shader is more scalable and allows more source-level options for personalized version (if you wan't to dive into the sources, that is).
- Ambient occlusion support. Unlike traditional Ambient Occlusion it contributes to diffuse scattering and is suppressed when light direction is close to surface normal to fake directionality (similar to bent normals). For spherical harmonics lighting Ambient Occlusion only affects the constant term, again giving a bit of directionality. In a nutshell it doesn't look like old-school AO that usually looks like burnt.

WARNING: significant changes compared to 1.1.
- Some parameters behave differently, thus old materials need to be hand-adjusted when used with 2.0. You can keep old shader in the project and upgrade only materials you want. New version has different names and GUIDs intentionally. Known incompatible changes that need to be adjusted:
- Translucency sliders have different response
- Bumpiness sliders are removed in favor of diffusion profiles. To tune the overall blurriness there's now a single blur range property, but the profile decides the the blur ratio of different light frequencies.
- Tessellation has been removed. Planned for 2.1 update in one form or another.
- Minimum Unity version required now is 5.0.

Preparing necessary content

If you strive for realism the process requires good normal, specular and roughness maps. The most common way of making them is sculpting a high poly mesh in ZBrush (or similar, I heard Blender has sculpting too) and reprojecting to low poly mesh you use in the engine. At the very least the normal maps are done this way. Ambient occlusion and depth/thickness map can be baked in most 3D software packages and utilities, including xnormal. For other textures you can do just fine in photoshop.

Below a quick cheat-sheet that should get you started. Written by myself, a programmer and not necessarily backed by real world experience or hard numbers. Feel free to discuss on the official forum thread where some nice people may help you or mail me with some feedback positive or otherwise, based on your experience.

Texture resolution and material division

Ff you want detail to include skin pores for close-ups things may get a bit tricky. In this case 4k textures are not enough for the whole body and you may want to split the head from the rest (and hide seams near clothing). While Unity supports 8k textures now, for high quality/main characters you may want to split body parts anyway for better control of blur levels etc. A future version may introduce detail maps which you can use to enhance visual quality considerably while keeping texture sizes low, but it's out of scope for 2.0.

UV mapping

The shader makes use of mip maps for blurring the normal and ambient occlusion maps for fine scale scattering. This implies that: 1. UV scale should be kept roughly uniform on the area covered by one material. Not strictly necessary, but on high detail areas the varying blur can be noticeable. This limits the uv compression strategies often used by artists and can expand necessary texture resolution by extension, so a compromise can be made. 2. With higher blur levels areas around the UVs may bleed-in resulting in artifacts/seams. You have to keep a bit of distance between elements and fill the empty areas with nearest color. See "heal selection" or "minimum" in gimp/photoshop. NB: if the seams are on border of the texture and the texture is repeat-adapted then it's not necessary, but such UV layout is likely to have issues described in point nr. 1.

Mesh density and normal map detail

The shader is somewhat geared towards a certain level of detail balance between mesh density and normal map. Something between 10k and 50k triangles per character. Lower than that some visual artifacts emerge, higher and some SSS begin to disappear. Basically it's adapt for medium to high quality assets (as writing, it's early 2015), but some benefits are gone with too high poly meshes, like raw 3D scans. For these, screen space scattering techniques are more adapt.

Normal Map

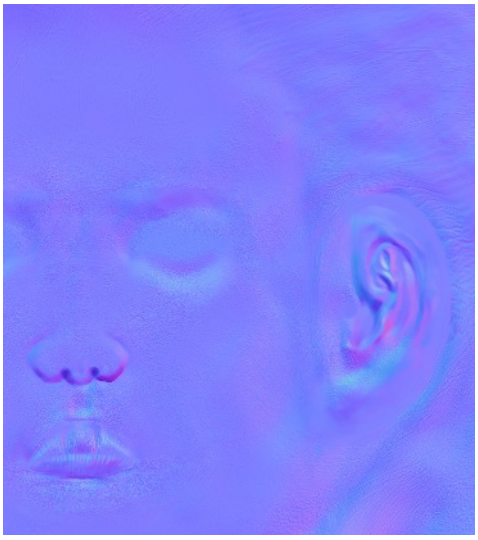
It's the most important texture for this shader. It should contain fine details of the skin along with larger features, down to the skin pores, if desired. Standard way of making such a texture is to generate it from high polygon mesh. ZBrush software is usually the tool of choice.

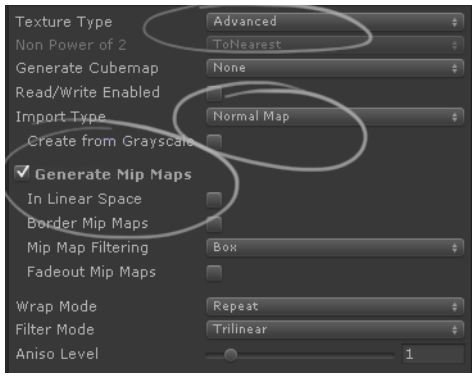
Make sure that the texture type is set to normal map and mip maps are enabled. Some features of the shader depend on them.

If you encounter visual artifacts, try turning off the texture compression or choosing other texture format. Sometimes it's enough to smooth out some parts of the texture to get rid of artifacts and keep texture compression on.

If you observe strong color bleeding on some areas either adjust blur level for whole material or smooth out buggy areas on texture.

To smooth out a part of normal map manually spray out a bit with (R:128,G:128,B:255) color with 30% or less opacity.



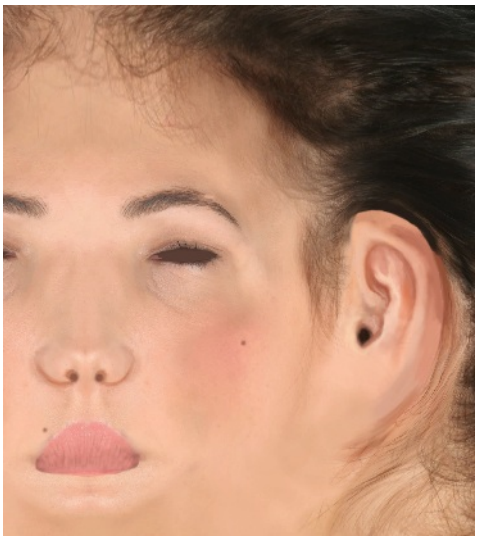


Diffuse Map

Defines albedo for diffuse diffuse lighting. It should contain overall color independent of light direction, possibly cleaned of specular reflection and shadows.

You should make it pretty smooth leaving finer details to other textures. Skin pores and fine wrinkles can be completely absent here. Skin pores - if any - should be present on normal map instead. Wrinkles should be on normal and ambient occlusion maps.

You can start with some photos as a reference, but try to reduce original lighting to a minimum. It should look as if lit from all directions.

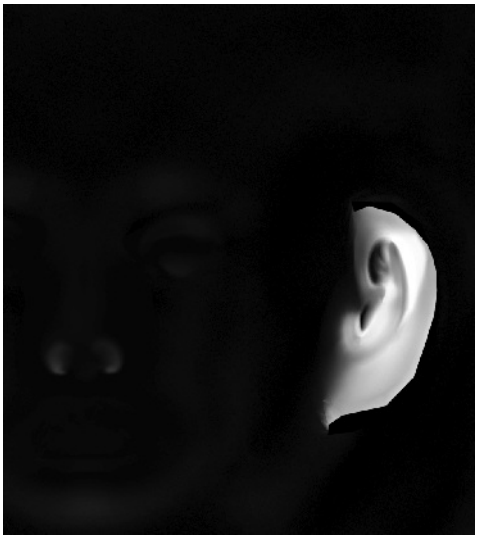


Depth Map

The depth map defines how much light can enter and scatter under the surface. It's also used for strength of translucency/back scattering effect. You can think of it as a thickness map. The areas with much soft tissue will scatter more. Areas with little tissue between skin and bone, like forehead will scatter less. Curved surfaces like nose scatter more, while flat surfaces scatter less.

Ears are particularly curved and thin so they scatter a lot of light. In particular the light penetrates easily from back to the front surface. This phenomenon is called back-scattering, but is referred to as translucency in the shader.

This texture can be generated in most 3D packages by baking ambient occlusion with inverted normals. You'll need to tweak it manually as well, but it's not difficult or at least not impossible, even for a beginner.
In VRay you can bake a dirt map. example
In Mental Ray you can bake an ambient occlusion map. example
Make sure to fill with pure black color parts you don't want to scatter, like hair, clothes or inside of the mouth. Good solution is to temporarily hide or remove polygons of inside elements of the object before baking. Alternatively you can provide a Skin Mask texture.



Skin Mask

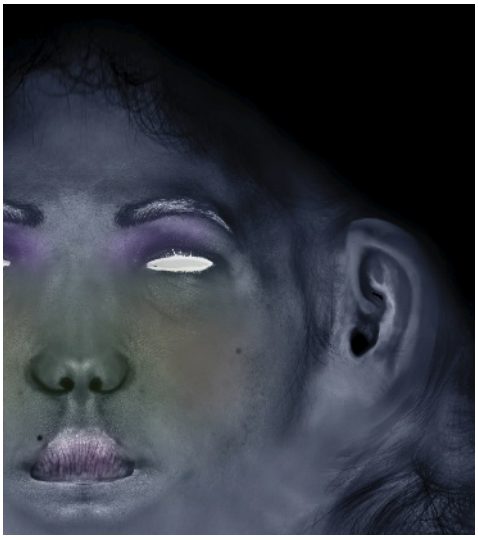
Skin mask texture allows to mask areas which should not render as skin. White will render as skin and black is masked. Masked-out areas still use the same specular model model, but are not affected by scattering, translucency, diffuse bumpiness and rim. It also blends between the two fresnel values. You can leave it empty to render all the surface as skin.

It's useful for elements such as hair, inside of the mouth, nostrils, or areas under the clothes. If possible you should use a different material for non skin parts of your characters. This just allows blending in fine details without additional polygons and materials.

Specular Map

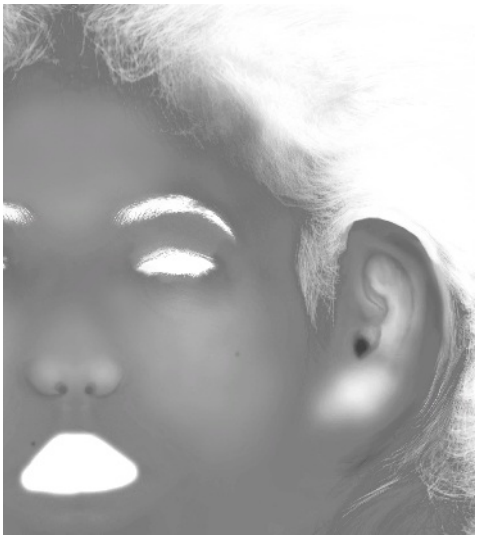
This is specular intensity and color map. The shader doesn't use metallic setup. The human skin and it's oily layer reflect light mostly as. That means the texture should generally be gray scale. The shader permits colorized texture so you can add things like make up or metallic elements. If you wan't you can colorize slightly to add a moce artistic touch. Just remember too saturated colors can unnaturally override the light color, defying physics.

TODO some tips on measured skin ares and how much detail should go in



Glossiness Map

Specular glossiness map. Defines the way light reflects off the surface due to surface roughness.



Translucency Map

Allows to add some detail to the translucency, like inner veins or cartilage that block the passing light or vary translucency color.

Currently details are barely visible, unless the texture is high contrast. As it's currently in experimental stage the behavior may change in future releases.

Tessellation Displacement Map

Vertex displacement (height) map for tessellation.

Pay attention to the seams of UV layout, to not create discontinuities, otherwise it will cause holes in the mesh. Also consider the tessellation may cause drastic performance drop if subdivision is too high. We suggest not to put too many details in the displacement map, since it can cause undesired spikes on your mesh if subdivision factor is too low. In linear rendering mode check bypass sRGB Sampling option.

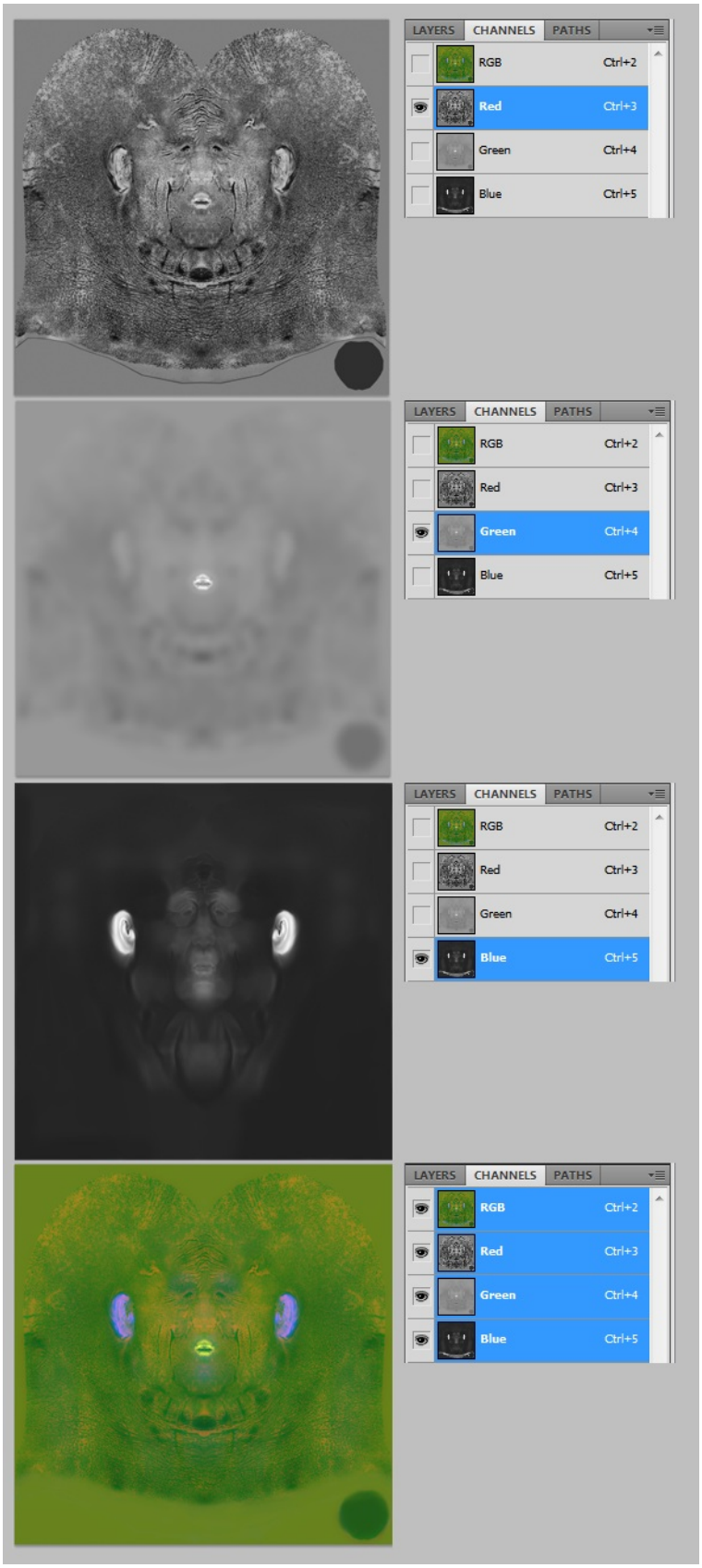
Combined Textures Variant

Included is a variant of the shader that uses combined textures for efficiency. This reduces texture memory and bandwidth, but you have to make the combined textures. Specular, Glossiness, Depth and Ambient occlusion maps are combined into one texture in red, green, blue and alpha channels respectibely. Skin mask is stored in alpha of diffuse texture. Normal map is separate for texture compression reasons and displacement map is left alone.

Some limitations apply: translucency color map is not supported and specular is grey scale only.

The recommended approach is to first prepare and test the individual textures with standard version of the shader, then combine them.

In Photoshop you can copy an image, paste it into a color channel. You can also edit the channels separately. In Gimp you can use the Decompose/Recompose feature.



Useful links and resources

Our video tutorials

Introduction

Premise

This manual presumes the reader has basic understanding of real time graphics and Unity3D software package. Terms such as normal maps, diffuse and specular lighting are used directly, without further explanations. Beware that to achieve high quality results with this shader you'll have to put a great amount of work in creating necessary assets, especially the textures. For best

workflow we recommended using ZBrush which is commercial software.

Requirements

Version 2.0 of the shader requires at least version 5.0 of Unity. Earlier version which you can still download from asset store or from source code history works in earlier versions, starting from 3.5.

It should work on large variety of graphics configurations, but features are limited by graphics card capabilities. A graphics card of Shader Model 3 level (Direct3D9.0**c** / OpenGL2.1) is recommended. The most basic variant works on OpenGL ES 2.0 devices, but mobile device support is not official yet. Future release may include official support (a mobile variant with limited feature-set). Consoles are not supported at the moment since I don't have any access to dev kits.

Limitations for Shader Model 2 / OpenGL ES 2.0 hardware

- Spherical Harminics lighting (light probes/GI/IBL) only per vertex and without subsurface scattering
- Translucency only for main directional light
- Shadows only for main directional light
- Ambient occlusion without directional suppression (can look burned)
- No reflection probe support
- Overall low quality compared to SM3/SM5
- Uses three passes per light so performance is roughly three times worse than SM3, despite less features.

Remarks

The shader is designed to be applied to dynamic objects only, thus it doesn't support lightmaps and doesn't implement the meta pass, used to emit GI (it receives GI, including scattering).

Diffusion Profiles

The editable values are parameters to six gaussians that make up the approximated diffusion profile for subsurface scattering of skin (or any multi-layer semi-opaque material).

Creating a new physically based profile may be pretty hard and I myself just bundle one profile based on values kindly offered by NVidia. See section "Diffusion Profiles" in http://http.developer.nvidia.com/GPUGems3/gpugems3_ch14.html for a thorough explanation. You may need a deep dive into the math and study the scientific papers involved world and learn to use Mathematica/Matlab/Octave.

Anyway feel free to tweak or edit the profile with intuitive artistic approach or just use the bundled ones. When creating a profile you can think of the Radial Distance preview as a laser dot pointed at skin in darkness. You can zero all values at the beginning and set one profile element, or "layer" at a time. Each element row correspond to a row in the inspecor and is made of a **color** **weight** and **variance**. Variance is the blur radius in millimeters. Note that while it's defined in millimeters in the profile, the actual size in world space depends on the model its UV layout. The minimum and maximum blur amount is adjustable in material inspector for quick tweaking. Variance is used by the shader to select the mip level for textures, in particular the normal map, which implies some requirements on content creation. The color weights are automatically normalized so the total sum is always white color of magnitude 1. This is to allow the diffuse texture to define the overall color like usual. The first profile element should be the least blurred with variance near zero. This is because the shader at low quality settings reuses the normal with this blur level for specular lighting etc. Others can come in any order you want.