PBR workflows in Cycles Render Engine

PBR workflows for realistic rendering in Cycles Render Engine

Joonas Sairiala

Bachelor's thesis
February 2015
Degree Programme
in Media

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Degree programme in Media

SAIRIALA, JOONAS
PBR Workflows in Cycles Render Engine

Bachelor's thesis 78 pages
2015

The Cycles render engine is capable of impressive renders, but is not one of the most usable render engines. Cycles renderer lacks masters shaders, and out of the box, it is not able to be utilised as part of a physically based production environment. Fortunately Cycles offers extremely powerful shading tools that allow for the creation of various types of master shaders. In this thesis I will examine the common physically based rendering paradigms and workflows from the artist's perspective and implement a physically based shader using Cycles' nodes. This thesis includes practical render examples where I have utilised the physically based shading methods in order to create realistic materials and textures.

Key words: rendering, cycles, blender, shading, texturing, 3D

CONTENTS

ABBREVIATIONS AND TERMS

t

HDRI	High Dynamic Range Image
IBL	Image based lighting
BRDF / BSDF	Shader describing light reflection, refraction and absorption at an object surface.

1 INTRODUCTION

The choice of a render engine for any 3D artist and studio can be a tough one. While most render engines offer up to date features and quality renders, their performance, licensing, and suitability for any given pipeline is what usually determines whether their chosen or not. For the individual freelance artist and small studios the costs of licensing and hardware requirements for any given render engine is often the primary concern when choosing an engine to use and greatly limits one's choice.

For students, small studios and freelance artists, especially in countries like Finland with a small VFX industry, the accessibility to tools like Cycles offers chances to grow. The industry is constantly pushing the boundaries of what a computer generated image can look like, setting the bar higher than what most small studios or independent artists can reach simply due to the assosiated costs or lack of capable tools.

One of the current trends in real-time and offline rendering is physically based rendering. In this thesis I will examine the conceptual frameworks and workflow guidelines surrounding physically based rendering and content creation including how they can be used with Cycles render engine.

In this thesis I will take a look at how Cycles in it's current state can function as a physically based production renderer. I will take an indepth look in to the general concepts of physically based rendering as well as go over how these concepts work within the Cycles render engine.
With the current implementation of Cycles lacking any kind of physically based master shader I chose to create one. In this thesis I will go over the implementation of a physically based shader utilizing Cycles shader nodes and the existing BRDF components. I have also included practical case studies done with Cycles that help to illustrate some of the key concepts of realistic rendering with Cycles.

2 The role of visual effects

In this chapter I examine the nature of modern visual effects and what the general requirements are for a visual effects production to work. While my thesis focuses on the use of realistic materials and lighting and the associated workflows it is important to get a general idea of how realism in general visual effects is used. Examining modern visual effects and the way they are used allows us to perceive the significance of light interaction and surface materials in synthetic imagery.

In cinema visual effects have always been an important consideration as they can arguably make the novel experience more impactful. Visual effects are used to aid the narrative through immersing the viewer in the world of the film.

From the very beginning of cinema film makers have used various types of special effects in order to portray various scenes and situations otherwise unfeasible to produce. Visual effects whether practical or digital have played a major role in creating the world, narrative and athmosphere of various iconic films and television series. The importance of various effect types have increased over the years allowing for flexible ways of production and an increased number of ways each production can be completed.

It is safe to assume most people equate visual effects in film and television with fantasical landscapes, creatures and events, but visual effects can also be used to create the subtle but necessary effects. The nature of visual effects expands far beyond what most people would think. The experience of watching a film or a television series has changed radically since the introduction of digital visual effects. We often cannot distinguish between what is a clear visual effect and what is not. Even the most mundane looking things shown on screen can be visual effects.

2.1. 3D rendering as part of a visual effect production

Depending on the production 3D-rendering can influence the final product in various ways. The use of 3D as part of a visual effect shot can be divided into two main categories. The use of 3D geometry to guide other aspects of the final synthetic image or as an additional element rendered from scratch.

Combining live action images, animation, stills and paintings in all three dimensions is the cornerstone of modern visual effects. The digital toolsets that allow for the image to be manipulated in limitless ways derive the produced image from real locations, real objects, but eventually synthesize new ones. (Prince, 2011. Digital Visual Effects in Cinema: The Seduction of Reality)

2.2. Digital rendering and realism

2.2.1 Defining the types of realism

Realism as a term has been used to describe various different things from philosophy to art, but for modern digital rendering realism is often assosiated with various ambiguous forms of image production. For the purpose of this thesis I will now outline exactly what I mean when referring to a specific type of realism and why the distinctions matter.

Much like realism, photorealism is used to refer to either an art movement or in computer graphics an un-biased form of rendering.

In this thesis I am examining ways to produce digital renderings that are in every sense of the word realistic in terms of the visual fidelity and accuracy of light interaction. It is important to make the distinction that realism does not in any way refer to reality.

As Stephen Prince points out in his book. The nature of realism in cinema and other digital medium is inherently limited as the display and capture devices used to produce the content are incapable of producing the amount of dynamic range in color and light as the human eye is. (Prince, S. 2011. Digital Visual Effects in Cinema: The Seduction of Reality, 192-194)

Therefore aesthetic realism or photorealism is not strictly speaking a representation of reality.

For the purposes of this thesis and for clarity I am referring to realism as the following.

Realism in this thesis refers explicitly to aesthetic forms of realism where the subject matter is represented in a way that is true to life and the renderings of objects are void of most artificial artistic conventions with moderate amounts of accuracy in the way the object interacts with artificial light.

2.2.2 Realism and immersion of visual effects in live action cinema

The nature of realism in live action cinema has changed dramatically over decades of filmmaking. We have shifted from an inherent expectation of on-screen depictions of reality to a way of cinema more focused on the notion of immersion and credibility.

When viewing a live action film or looking at a photograph there used to be an inherent expectation of reality, where the viewer would perceive the images portrayed as real. The inherent authority of analog media came from the fact that they were hard to manipulate and change. This made viewers far more likely to believe that what they saw was actually real. This has changed dramatically over the years. The inherent nature of digital media means that the authenticity of the images can no longer be trusted.

(Prince, S. 2011. Digital Visual Effects in Cinema: The Seduction of Reality, 148-150)

However this does not mean that the visual effects used should not strive for realism as it is ultimately the combination of narrative and the perceived realism of the film as a whole that dictates whether or not the viewer experiences the film as immersive and believable. (Prince, S. 2011. Digital Visual Effects in Cinema: The Seduction of Reality, 148-150)

What modern film goers experience as realistic is a combination of context sensitive artistic choices. An immersive film, where each part of the production is carefully considered, is experienced as “realistic”. The nature of the medium is inherently not mimetic of our own reality, but presents a different visual space with it's own subjective reality. For this separate imaginary world to sustain the illusion with the help of visual effects they have to strive for aesthetic realism.

In modern cinema the presentation of sound and moving images produce the depiction of imaginary worlds and characters. The more convincing these things are made to seem, the more immersive the film becomes.

(Prince, S. 2011. Digital Visual Effects in Cinema: The Seduction of Reality, 183-185)

2.2.3 Aesthetic realism in 3D animation and video games

Animations are often regarded as a more stylized medium with varying degrees of realism in their art style. For the most part considerations of mimicing reality are less prominent in stylized animation, but aesthetic realism in the 3D-renderings is not. The development of digital imaging tools that allow for a more realistic representation of materials, light and surfaces is directly responsible for making the perceived images more immersive.

For example in Toy Story the viewer is primarily responding to the characters, but the ability to create a more visually appealing and aesthetically realistic images strenghtens the immersion and response viewers have to the film. (Prince, S. 2011. Digital Visual Effects in Cinema: The Seduction of Reality, 190-192)

PICTURE 1. Toy Story 1 (Disney Pixar Animation Studios 1995)

PICTURE 2. Toy Story 3 (Disney Pixar Animation Studios 2010)

As evident from the images above the visual style of both Toy Story movies remains essentially the same but technological advancements have provided the later Toy Story film with far greater visual fidelity through increased contrast between the surface materials on the characters and greater details in light interaction. While the images are stylized the surface materials and rendering methods in Toy Story 3 are far more detailed and realistic, providing an image that pleasing to look at and all round more expressive.

For video games the same notion of increased immersion and emotional response rings true as well. The development of rendering technologies, especially physically based rendering methods, have brought real time rendering closer to the image quality of cinematic visual effects, which makes striving for realism in video games presentation more relevant.

2.3. Realism and Physically Based Rendering

Before moving on to the second chapter where I introduce the Cycles render engine it is worthy to take a quick look at what realism in physically based rendering is.

As I will describe in detail in section 4, physically based rendering is a rendering method that bases calculations of light interaction and material shaders in reality. This however does not mean that physically based rendering is explicitly used for art styles that attempt to replicate reality in one form or another. In a PBR environment the render engine utilises methods that can produce a wide range of visually interesting images through visually credible material shaders and lighting. Realism in a PBR environment is inherently implied through the way the systems work, but it is not necessarily the goal of the production's art direction.

As Hand states in his 2014 presentation about PBR in Unity:

Realism in physically based rendering is often a misunderstood term which fails to conceptually realize what a PBR system is capable of. A PBR system is capable of a wide range of material representations which in turn enables the production of a variety of visual styles. So for example a stylized production would benefit from a PBR system in increased options for visual contrast and a more accurate representation of light. (Hand, A. 2014. Unite 2014 – Best Practices For Physically Based Content Creation)

An art directable approach in a physically based environment is also possible as effectively illustrated by the Disney princibled BDRF.

The Disney princibled BRDF is a physically based shader model that utilises physically plausible input values to represent a variety of materials. While the physically based the shader model is not strictly physically correct, using more intuitive inputs makes authoring physically plausible but art directable content easier. (Burley, B. 2012. Physically-Based Shading at Disney, 12)

3 UNDERSTANDING CYCLES

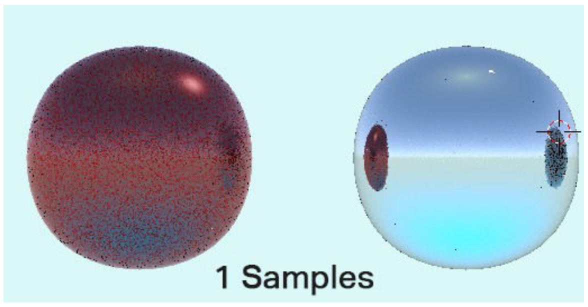
3.1. The basics

In order to effectively work with Cycles we first need to understand how it actually works. Understanding the basic principles of how Cycles renders images as a path tracer allows us to reduce noise, increase render quality and troubleshoot our renders without having to resort to pure trial and error. This chapter describe the basic features in Cycles and how they work.

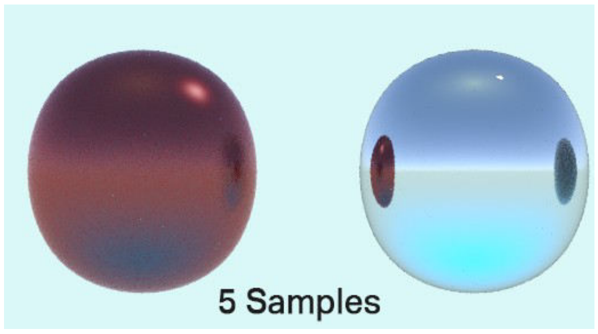
Cycles is an unbiased physically based render engine that uses path tracing in order to produce the rendered image by tracing rays through the scene, which then based on the surface function (BRDF / BSDF) the ray hits, returns a value to the camera (Blender Foundation, 2015. Integrator).

Unlike with a traditional scanline renderer, the quality and sharpness of an image rendered with a path tracer is in direct correlation with the amount of per pixel samples the renderer traces.

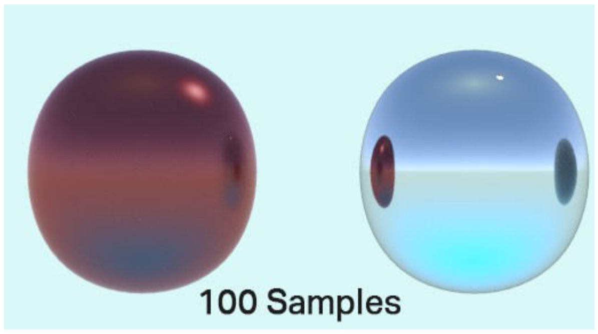
Below are examples of how drastic the difference between different sample amounts can be. We can see that starting from one samples onward even the sphere's edges start to get clearer and less jagged, which directly affects the images overall quality.



PICTURE 3. Object rendered to 1 samples (Joonas Sairiala 2015)



PICTURE 4. Object rendered to 5 samples (Joonas Sairiala 2015)



PICTURE 5. Object rendered to 100 samples (Joonas Sairiala 2015)

As described in the Blender Manual the ray types used by Cycles can be divided into four categories: camera, reflection, transmission and shadow. These types determine where the ray is generated from. Out of the rays reflection and transmission rays can have additional properties that describe the way the ray is traced. The additional properties are as follows: diffuse, glossy and singular. The diffuse ray is generated by a diffuse reflection or transmission, describing a surface that reflects light at many angles. A glossy ray in Cycles is generated by a glossy surface, which in turn, reflects light a fewer angles. (Blender Foundation, 2015. Light Paths)

3.2. Cycles Materials

Cycles uses physically based shader models that offer the ability to build advanced layered materials according to their real world counterparts (Blender Foundation 2015. Surface).

A complete Cycles material consists of many shader nodes. Even the very basic materials in Cycles are defined by combining multiple shader nodes together, as opposed to using a pre-built material with a large array of settings to tweak. A single shader material in Cycles often has no real world counterpart as all shader components in Cycles have a single function in terms of how light reflects off of that surface. For this reason building good materials, based on a solid understanding of how real world materials work, is paramount for a good Cycles render.

In Cycles the main shader type used for majority of materials is the surface shader, which defines all light interaction at an objects surface. The surface shader consists of one or more BSDF shader nodes which are used to control the light interaction. (Blender Foundation 2015. Surface).

The below diagram describes how a Cycles material is structured.

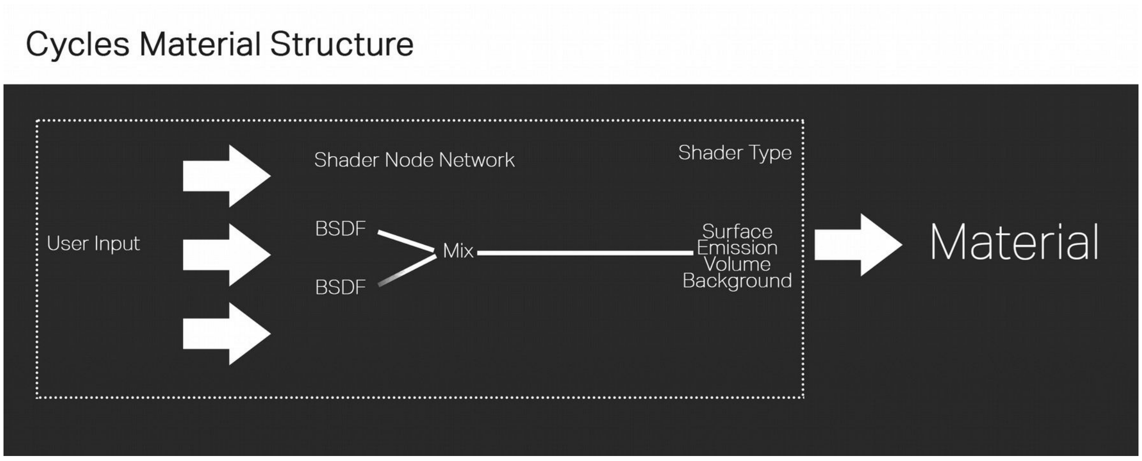


FIG 1. Cycles Material Structure (Joonas Sairiala 2015)

3.2.1 BSDF

Bidirection scattering distribution functions, or BSDF's, define how light reacts to a shader. Cycles offers three primary BSDF types: Reflection, Transmission and Refraction BSDF's which we can use to build any type of material with significant control. (Blender Foundation 2015. Surface).

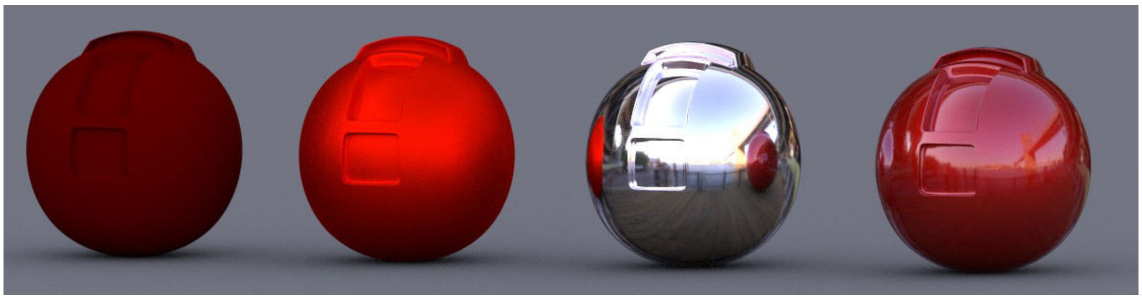
BSDFs in Cycles are technically shaders that each represent a single component of light interaction, for example diffused and glossy reflection. Each BSDF includes various important inputs from color of the surface, shading normal input to roughness of the reflection. Some BSDFs have additional properties and all glossy BSDFs have a setting for different microfacet distributions. (Blender Foundation 2015. Shader Nodes).

The BSDFs used in Cycles are the most essential building blocks of a material. Most materials consist of one or more BSDFs mixed together that each contribute a component in to the material. The most essential thing to understand that each BSDF outputs, not color, but a shader output. This means that we can build advanced materials with fine control as mixing them together results in a more advanced shader output.

The most clear example of multiple BSDFs and their role in a Cycles material would be for example a clear coated paint. A clear coat paint would have three distinct components that make up the final material. As paint is usually an insulator the material would be dielectric, with achromatic reflections. The base layer of the clear coat represents the diffuse reflection color of the paint and the top layer represents the clear coat. All of the layers are then rendered together as a complete clear coat shader.

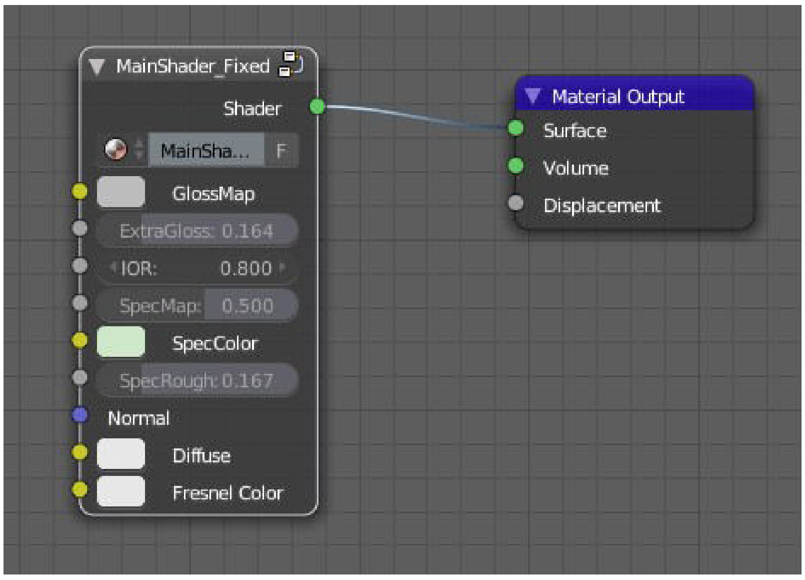
Understanding the way BSDFs work and contribute to a material allows us to build a solid workflow for creating advanced and accurate materials in Cycles. The below images show the clear coat shader and the different components rendered with an sIBL HDR environment from the sIBL archives.

PICTURE 6. Clear coat shader (Joonas Sairiala 2015)

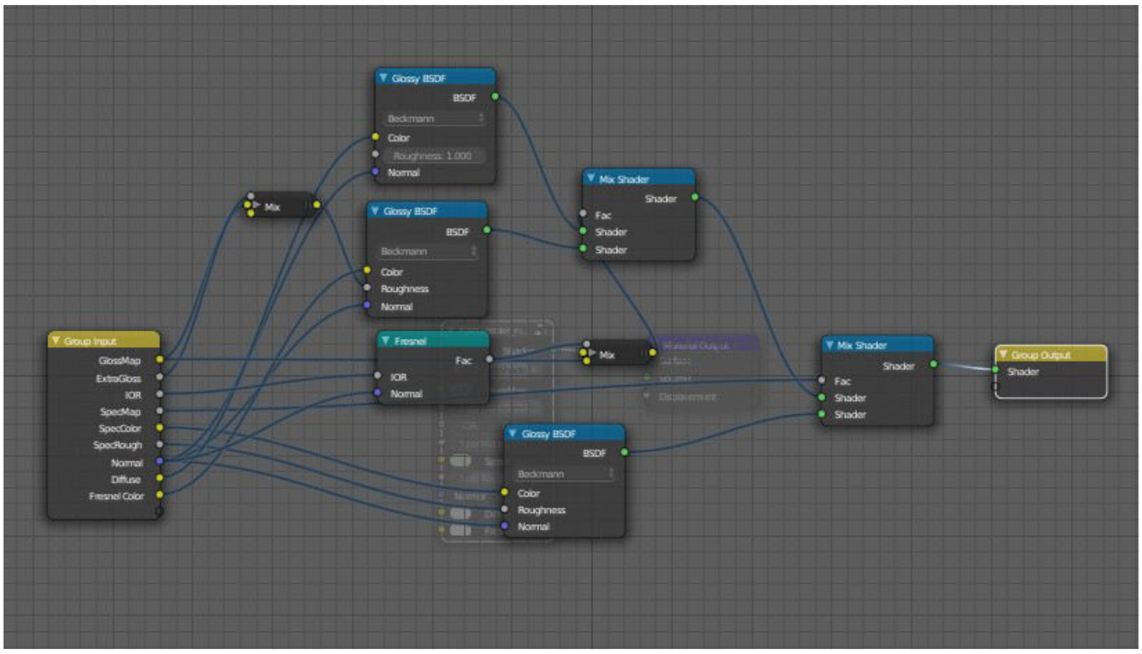


PICTURE 7. Different clear coat shader components. From left to right: Diffuse BSDF, Glossy BSDF, Glossy BSDF and the final shader. All glossy BSDFs use the GGX microfacet. (Joonas Sairiala 2015)

In my experience this type of workflow makes Cycles an extremely powerful render engine in terms of render quality and the ability for the user to create and freely edit all the components of any given material for the purposes of improving realism and efficient rendering.



PICTURE 8. Example of a Cycles master shader group built out of multiple shader components with multiple inputs for textures and settings. (Joonas Sairiala 2014)



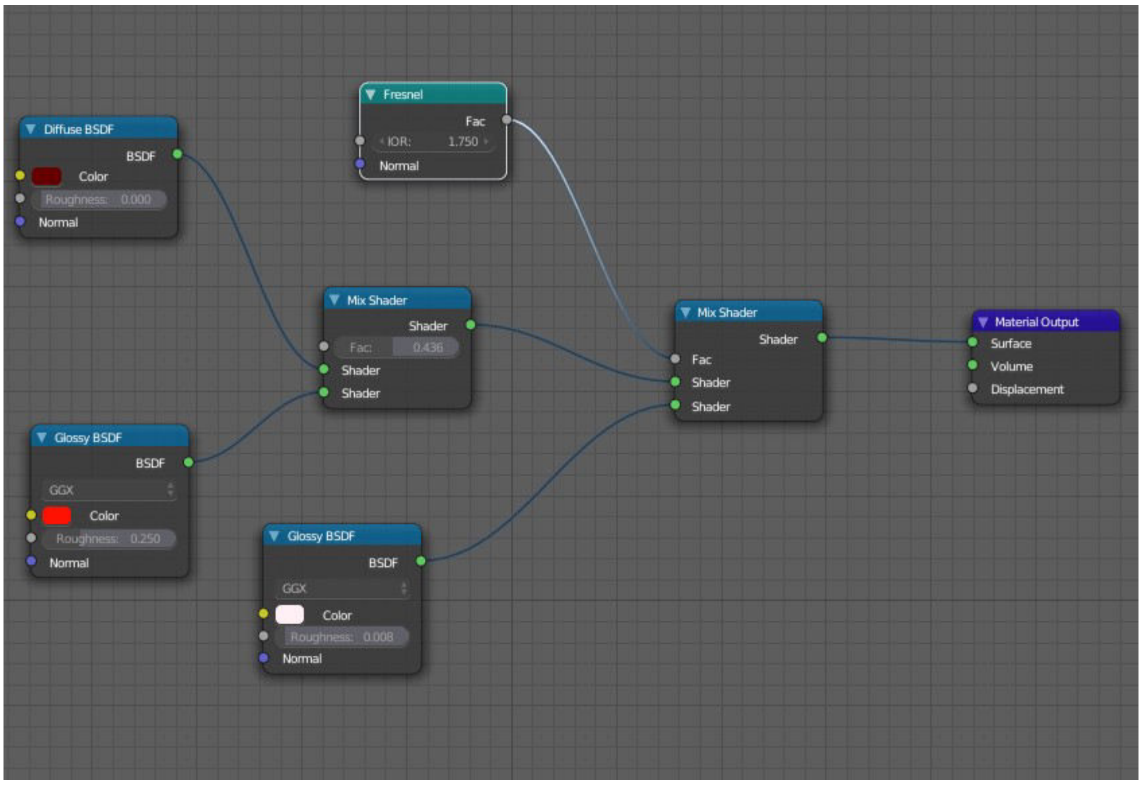
PICTURE 9. Above master shader group nodes. (Joonas Sairiala 2014)

3.2.2 Mixing shaders in Cyclers

Cycles BSDF's can be mixed together using the Add or Mix shader nodes, both of which output a mixed shader for the surface shader to use. Additionally the mix shader includes a factor value, which is used to blend between two shaders. The value range for the factor is 0-1. (Blender Foundation 2015. Shader Nodes)

In order to create a realistic material, use of the mix node is key. The mix node governs how two shaders are blended together and is essential in producing quality materials. The mix node's input value can come from anywhere from an image to a another shader node. For example a texture can be used to blend between two shaders.

In the case of the clear coat shader the mix shader node is used to control the amount of reflectivity on the base paint and the amount of clear coat on top of it. The clear coat on top of the base shader is mixed in using the fresnel node as a factor for the mix in order to achieve a realistic result.



Roughness: 0.000
IOR: 1.750
Fac: 0.436
GGX
Roughness: 0.250
GGX
Roughness: 0.008

PICTURE 10.Clear coat shader nodes (Joonas Sairiala 2014)

3.2.3 Open Shading Language

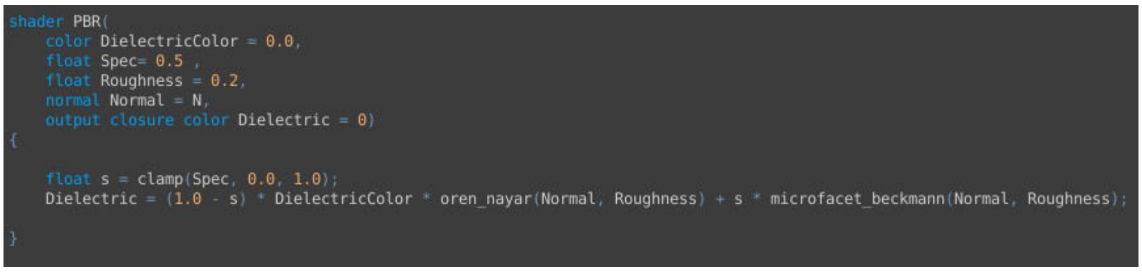
Open Shading Language is an open source language for shader writing developed by Sony Pictures Imageworks. OSL is built as a C++ library with a flexible programming interface for building production shaders for multiple different render engines. (Martinez, A. 2012. Physical Production Shaders with OSL. 2).

Blender's Cycles supports OSL as a way for building shaders and provides a solid alternative for Cycle's node based shading workflow. As OSL has access to all of Cycles' shader nodes this becomes a very powerful way of describing shaders.

OSL has a similar syntax to C, but differs from other shader languages in the fact that BSDFs and similar features are first-class concepts in OSL. OSL has been built to facilitate advanced shader algorithms with performance and ease of access in mind, which means that OSL computes a symbolic descrpition of light interaction referred to as a closure. OSL also does not feature any light loops as the standard BSDFs used with any OSL shader provide the light interaction calculations. (Sony Pictures Imageworks Inc. 2015. Open Shading Language 1.6. 1-4.).

Unlike the majority of shading languages, the way OSL works reduces the threshold of an artist without a background in programming entering the field of shader writing. OSL is effectively redefining the requirements for what is required of a 3D-artist.

As we can see from the piece of code below, describing simple shaders in OSL and Cycles is quite straightforward.



```
shader PBR(
    color DielectricColor = 0.0,
    float Spec= 0.5 ,
    float Roughness = 0.2,
    normal Normal = N,
    output closure color Dielectric = 0)
{

    float s = clamp(Spec, 0.0, 1.0);
    Dielectric = (1.0 - s) * DielectricColor * oren_nayar(Normal, Roughness) + s * microfacet_beckmann(Normal, Roughness);

}
```

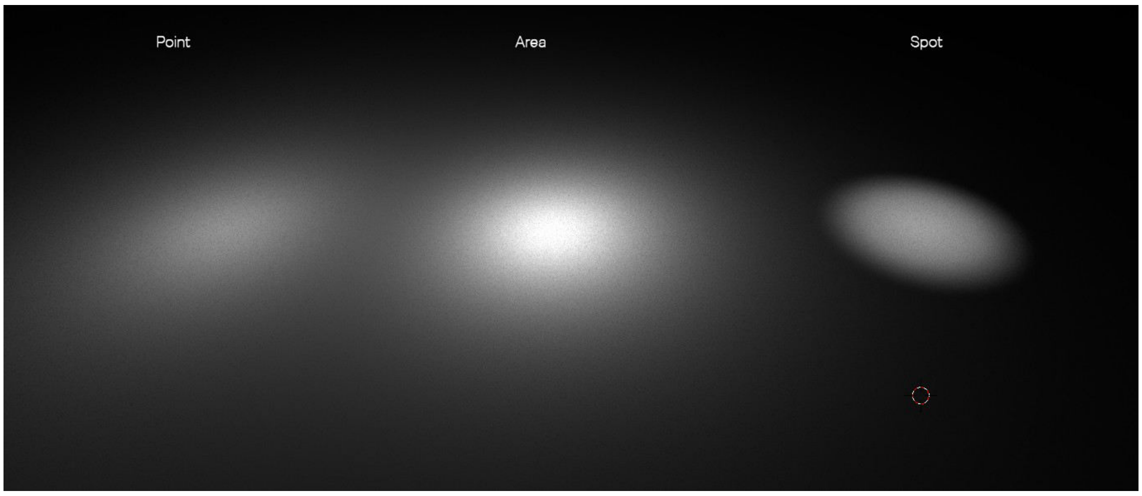
PICTURE 11. Dielectric PBR Shader component described in OSL (Joonas Sairiala 2014)

PICTURE 12. Rendered view of the OSL Shader component (Joonas Sairiala 2014)

3.3. Lighting in Cycles

As a pure path tracer Cycles uses light bounces originating from lamps to illuminate the scene, which provides fairly accurate global illumination without the need to manually fake it. (Iraci, 2013. Blender Cycles: Lighting and Rendering Cookbook 5)

Cycles has all the necessary lighting tools, including mesh lights, each editable through nodes, one would need to effectively light a scene. The primary lamp types are: area, point, spot and sun. Each lamp has a variety of options that affect their look, performance and behaviour. (Blender Foundation 2015. Lamps).



PICTURE 13. Examples of Cycles light types. Point ligh, area light and spot light. (Joonas Sairiala 2015)

An additional key component in Cycles lighting is the ability to use HDRi environment maps to produce lighting and reflections.

3.4. Cycles Render Passes

3.4.1 Cycles Render Passes

Aside from basic scene data passes, Cycles render is one of the very few progressive path tracers that are able to produce render passes that divide the image in to different parts. This is purely due to the nature of path tracing as a means of rendering.
Traditionally a scanline render engine would render the image in steps element after element, distinguishing between shadows, specular highlights and colors easily. With a path tracer this is not always possible as the image is rendered as a whole.

Cycles manages to divide the final image in to the following light passes, as described in the Blender Manual: three pass types Direct, Indirect and Color passes for Diffuse, Glossy, Transmission and Subsurface and a single pass for Emission, Environment, Shadow and Ambient Occlusion. Cycles defines direct light coming from lamps after a single reflection or transmission of a surface and indirect light after more than one reflection or transmission. (Blender Foundation, 2015. Light Paths)

This essentially means that the direct light pass has brighter highlights than the indirect pass, but both passes still have essential lighting information that cannot be excluded, unlike for example a legacy specular highlight pass. In addition to color and lighting types Cycles divides the image into passes by ray type, which produces different passes for diffuse and glossy surfaces. The division in passes between ray types is often quite alien for most artists, but provides a necessary tool for compositing the final image together and even with it's obvious limitations provides much needed control over the final image. While Cycles render passes, due to their nature, cannot be excluded from the final image per se, they can be manipulated in similar ways you would manipulate a legacy, more defined pass.

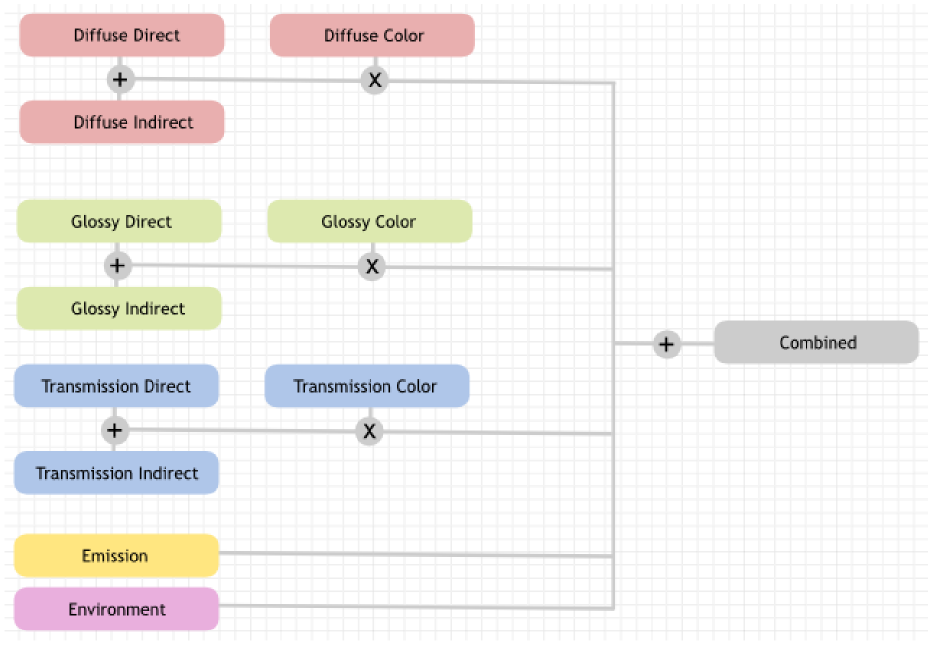
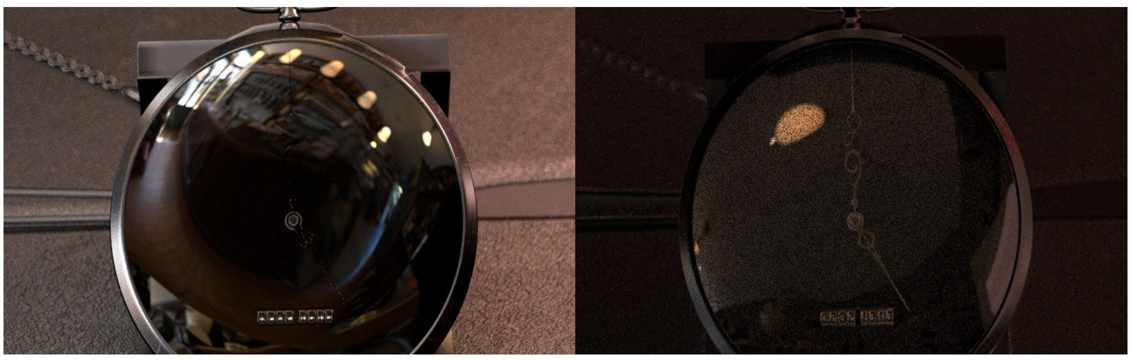


FIG 1. Diagram of the formula used to combine Cycles render passes (Blender Wiki)

For any 3D-render the compositing phase is extremely important for achieving a realistic result. In Cycles this is no different, but in addition to considerations of realism, Cycles render passes provide a necessary tool for denoising the renders. As you can see below in the render pass examples the indirect pass has a lot of noise, which can be reduced through various compositing methods.



PICTURE 14. Glossy Direct and Indirect render passes (Joonas Sairiala 2015)

For most pipelines Cycles render passes provide sufficient data for a successful composite, but the limitations mentioned before have to be kept in mind when compositing Cycles renders.

4 Physically Based Rendering

Physically based rendering, or PBR, is an umbrella term for shading and lighting models that accurately represent real world materials and phenomena. In a PBR environment legacy approximating techniques, like for example a specular highlight, have been disregarded. (Russell, PBR Theory)

4.1.1 Benefits of a physically based rendering environment

There are various ways that a physically based rendering environment can benefit a production. Especially for render engines like Cycles, where a singular workflow method is not strictly enforced, the established PBR paradigms can offer a variety of benefits. Adobting a set PBR workflow can help artists to author content in a more effective way compared to other types of workflows by allowing the use of a predictable shading and texturing pipeline.

Physically based shading systems offer a more detailed reasoning for the behaviour of light as legacy systems. The reasoning has basis in the real world, which makes the workflow for creating materials and shaders more logical and predictable. (Russell, PBR Theory.)

In a physically based environment the artist approaches the process of shading and texturing with a mindset that aims to render things as accurately as possible. In practice this means using measured values and creating materials that are consistent with a variety of lighting conditions. (Wilson, PBR in Practice)

Physically based rendering, PBR, as a term encompasses more than just the technical aspects of a shader that uses physically based algorithms to calculate light interaction. PBR introduces workflow guidelines that are about making the production pipeline more predictable and consistent.The mindset and workflow introduced by physically based rendering can mitigate production and content creation costs. (Hand, A. 2014. Unite 2014 – Best Practices For Physically Based Content Creation)

While the exact implementation of various PBR systems vary from each other the basic principles are the same. This means that most of the physically based rendering concepts and related workflows transfer easily from one render engine to another. (Wilson, PBR in Practice)

PBR is a new term in the world of real time rendering, physically based rendering has existed in the offline rendering world for quite a while. However the introduction of real time physically based rendering engines has provided the entire computer graphics industry with a collection of high level guidelines and content creation methods.

4.1.2 Physically based rendering and Cycles

Cycles calculates light interaction with a basis in reality, but by default Cycles does not work as a physically based rendering environment.

The Cycles BSDF implementation utilises lighting models (GGX, Ashikmin-Shirley, Beckmann) with a basis in reality, but it is not enough for a fully functioning PBR environment. In terms of workflows and efficient content creation there are some caveats when it comes to working with Cycles.

Physically based rendering is often associated with a set of common parameters and texture types for materials used by PBR authoring tools. Cycles does not have an enforced set of parameters or texture types, which makes for a variety of cases a very unintuitive and cumbersome shading workflow.

Additionally a PBR environment enforces plausible materials in various ways. In a PBR environment the creation of a non-physically based material has to be deliberate as the environment defaults to basis in reality. The power of node based shading in Cycles comes from the ability to create almost anything, but it also introduces a situation where it is incredibly easy to produce physically inplausible and unrealistic materials.

The ability to build various types of shaders means that Cycles is able to work as part of a PBR pipeline with an established tool chain that makes producing realistic content faster and more effective. We can utilise Cycles nodes to effectively create a PBR shader that enforces all the requirements of a PBR environment and enforces a physically based approach.

If we compare a standard way of producing content in Cycles to a workflow enabled by a PBR approach it is clear that for a variety of projects the PBR paradigms offer a much faster and effective way of production. The below chart represents a rough overview on the conceptual elements that go in to creating content with each workflow.

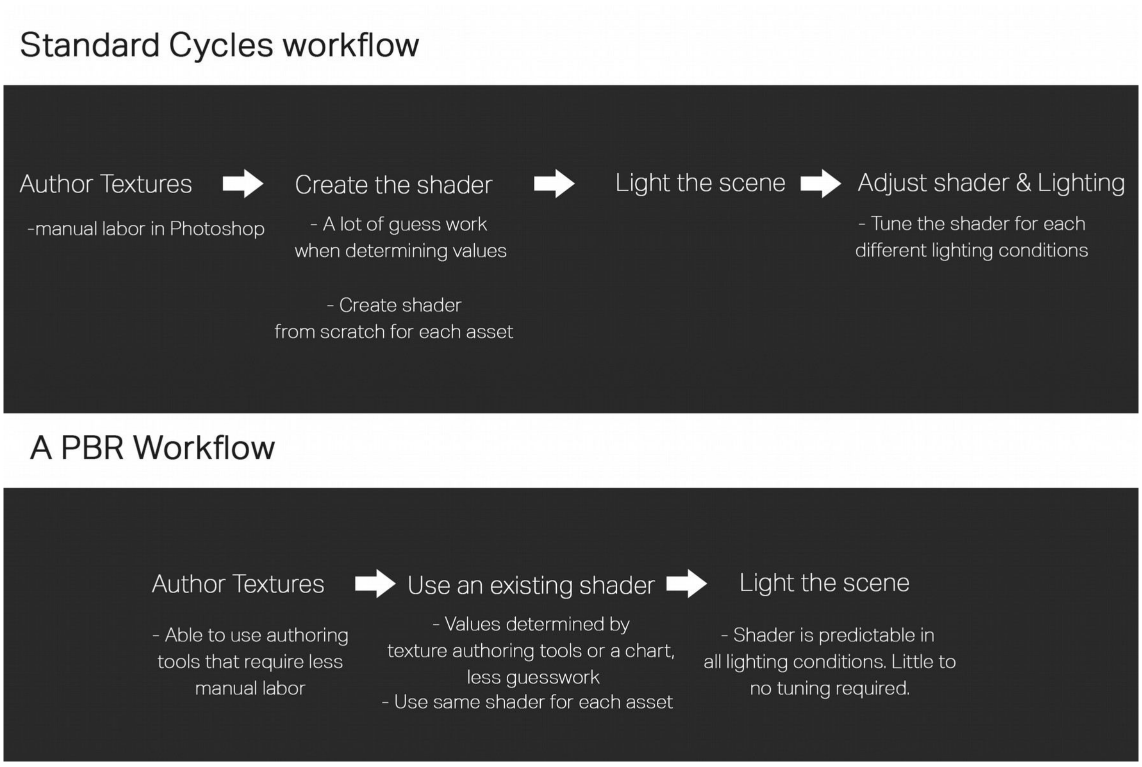

Standard Cycles workflow
Author Textures
-manual labor in Photoshop
Create the shader
- A lot of guess work when determining values
- Create shader from scratch for each asset
Light the scene
Adjust shader & Lighting
- Tune the shader for each different lighting conditions
A PBR Workflow
Author Textures
- Able to use authoring tools that require less manual labor
Use an existing shader
- Values determined by texture authoring tools or a chart, less guesswork
- Use same shader for each asset
Light the scene
- Shader is predictable in all lighting conditions. Little to no tuning required.

FIG 2. Workflow overview comparison (Joonas Sairiala 2015)

4.2. Physically Based Rendering concepts

4.2.1 Light interaction

For the artist creating materials for physically based environments looking at how real world objects react to light is a usefull way for producing a realistic material. When working in a PBR environment all decisions should ideally be based on observation of the real world.

How shaders react to light is one of the core elements of a good render engine and is one of the key aspects why physically based render engines can produce realistic results. Diffusion and reflection are two mutually exclusive terms desrcibing light interaction at an object surface. These physically distinct phenomena are the basis of any physically based material. (Russell, PBR Theory)

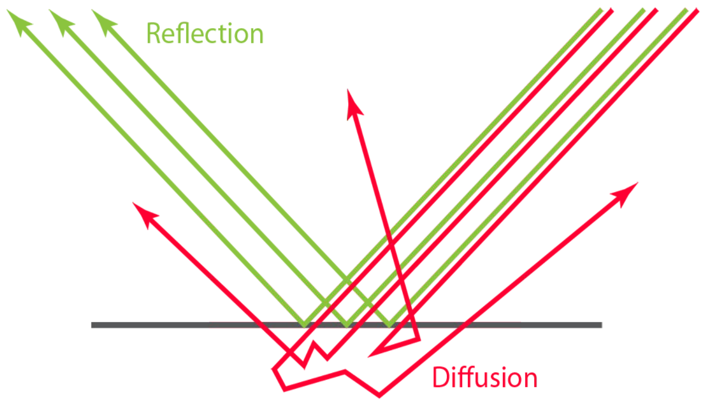


FIG 3. Light behaviour (Marmoset LLC 2015)

For physically based rendering environments understanding how to approach the very basis of a good material, which is the amount of light it either absorbs, diffuses or reflects, is key. While a good material is the sum of it's parts, finding the right balance between various light interactions for any given material is ultimately what dictates whether the material looks realistic to begin with.

In practice diffusion and reflection work differently in terms of how light is reflected back of a surface. A diffused surface scatters and absorbs light more than a reflective surface. The basic concepts of diffuse and reflection are fairly self explanatory, but in terms of creating good materials, the degree of reflection or diffusion has to be closely examined in order to achieve a realistic result.

4.2.2 Energy Conservation

One of the most important aspects of physically based rendering is energy conservation. Energy conservation means that a material cannot reflect more light than it receives and more often it reflects far less due to varying degrees of diffusion. In order for light to be diffused it must fail to reflect. (Russell, PBR Theory)

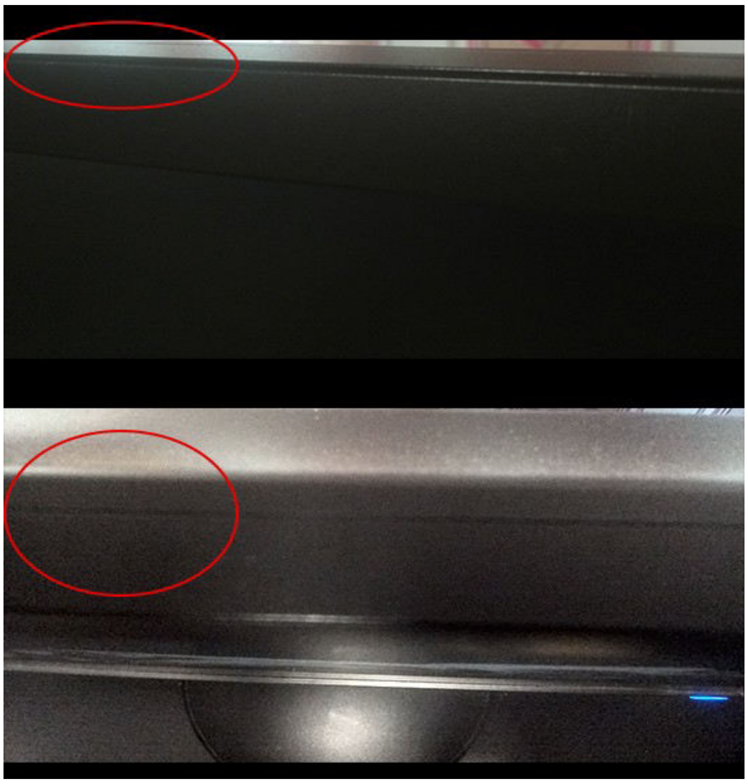
Energy conservation is characteristic of a physically based environment. Most legacy rendering systems use a specular shader for highlights, which without artist input, does not conserve energy.

In Cycles as long as all values the shader outputs are clamped to the 0-1 range the material is by default energy conserving. (Blender Foundation, 2015. Shader Introduction)

For Cycles the concept of energy conservation means that when we are building materials we do not go out of our way to multiply or add additional shader components in a way that goes beyond 1.

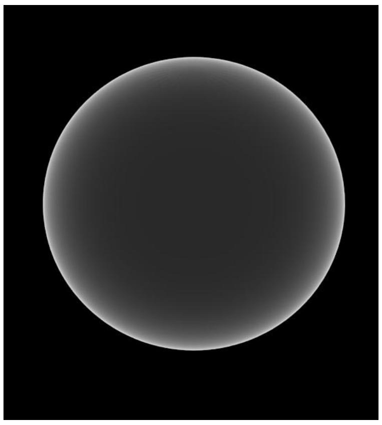
4.2.3 Fresnel

For computer graphics the term Fresnel refers to the physical phenomena of differing reflectivity that occurs at different angles. While this has been part of computer graphics for a long time, in a PBR environment the approach to fresnel is different to legacy renders. In a pbr environment the fresnel effect is ideally automated or atleast artists control over it is reduced when assuming realism is desired. (Russell, PBR Theory)



Picture 15. The Fresnel effect observable on a plastic lcd-screen border (Joonas Sairiala 2015)

In Cycles the Dielectric fresnel node calculates how much light is reflected off a layer and how much is allowed through. In terms of building shaders this means that for each layered material a fresnel effect can be used to mix the shaders together. The fresnel shader is based on the angle between the surface normal and viewing direction, meaning that the artist has control over how ligh reflects off of a surface at glazing angles and an actual refractive index value can be used to control it. (Blender Foundation, 2015. Input Nodes).



Picture 16. Cycles Fresnel Node Output (Joonas Sairiala 2015)

The Cycles' fresnel node is a separate input, use of which is entirely under the artist's discretion. Cycles does not by default enfroce the use of fresnel relfections, which results in often incorrect material behaviour. In chapter 7 I will outline in detail the use of the fresnel node in a PBR suitable Cycles shader.

4.2.4 Microfacets

On a large scale the relfection and diffusion of a material is supplied by the shape of the mesh it is applied to, but real surfaces exhibit microscopic surface detail that affects how reflective or diffused a surface is depending on its microscopic roughness. The term microfacet, micronormal or microsurface refers to the PBR shading systems way of simulating this, what is referred to as gloss, roughness or smoothness. (Russell, PBR Theory)

The microsurface roughness affects the apparent reflectivity of a surface. The relative intensity of reflection changes based on the values that are used for the roughness. (Russell, PBR Theory)

In Cycles the glossy shader node uses microfacet distribution models that produce either sharp reflections with the sharp algorithm or various degrees of sharp to blurry reflections with Beckmann, GGX or Ashikhim-Shirley. (Blender Foundation, 2015. Shader Nodes)

4.2.5 Material values

One of the key differences in working with PBR environments to legacy environments is the way we approach material values for color, reflectivity and roughness. (Wilson, PBR in Practice)

As all material behaviour in a PBR environment are based on the physical world we need to base the input values we use for the materials on the physical world as well. In practice this means taking a logical approach in determining the input values for any given material either by using pre-existing known values for things like reflectivity and surface roughness or determining the values ourselves by visual comparisons.

Using pre-existing values that are often scanned in a laboratory setting reduces the guesswork and provides a base material to work with that is physically accurate. The base values are not a strict rule, but rather a starting point for a good physically based material. A good example of a source for measured values is the Quixel Megascans library, which provides scanned surface values for a varity of materials. (Wilson, PBR in Practice)

4.2.6 Common PBR Texture inputs

One of the key things that makes a PBR production environment effective and intuitive for the artist to work in is a set of texture inputs for a shader that are common across all authoring tools designed for PBR.

Depending on the PBR engine in question the most common types of texture maps used with PBR systems are: albedo map, roughness map, metalness or reflectivity map, normal map and an ambient occlusion map. (Wilson, PBR in Practice)

Various PBR engines use either a specular map and gloss map to determine reflectivity values or a metalness and a roughness map. In the spec-gloss map style the specular map determines the color of the specular reflections and the gloss map determines the roughness. In a metallicness based input system the inputs for reflectivity are for metalness of the surface and a specular intensity for non-metals with a roughness texture for both. The metalness input style is also different in the sense that an albedo map is usually intrepreted differently by the shader based on what type of surface it is supposed to represent. (Hand, A. 2014. Unite 2014 – Best Practices For Physically Based Content Creation)

In Cycles the common input maps can be utilised by building a shader specifically for them, but there are a few considerations when using either style which I will outline in more detail in secion 5.2.2.

4.2.7 Requirements of a PBR system

For a PBR system to work in a render engine there are a few soft and hard requirements that need to be met. Most importantly in order for the shader to display correct values the system needs to work with a high dynamic range camera in linear space. This is due to the fact that output luminances on a PBR shader are far greater than with legacy shaders. A PBR system greatly benefits from tonemapping and post processing effects like bloom as well. (Hand, A. 2014. Unite 2014 – Best Practices For Physically Based Content Creation)

For Cycles the requirements of a PBR system are met by definition as it is a scene referred renderer where all color is referred to as scene linear. So by default Cycles works in linear space. (Blender Foundation 2015. Units).

Cycles also color manages all input textures in to scene linear space through an implementation of OpenColorIO, which removes the need for an artist to gamma correct input textures separately. (Blender Foundation 2015. Color Management)

4.2.8 Requirements of a PBR shader in Cycles

As previously mentioned Cycles is not in every sense of the word a physically based renderer, it merely has the ability to be used as such. The reason for this is that physically based rendering is more than just the technical implementation of lighting models, it is an entire production environment where material shaders behave a certain way.

For the purposes of this thesis I have outlined some hard requirements for the Cycles PBR shader below:

- Ability to use and display measured values for reflectivity and albedo color correctly
- Consistency across different lighting conditions
- The shader has to enforce a material behaviour that is accurate and physically based
- Input values should ideally be intuitive and match common PBR authoring tools

5 Workflows

5.1. Understanding physical material properties

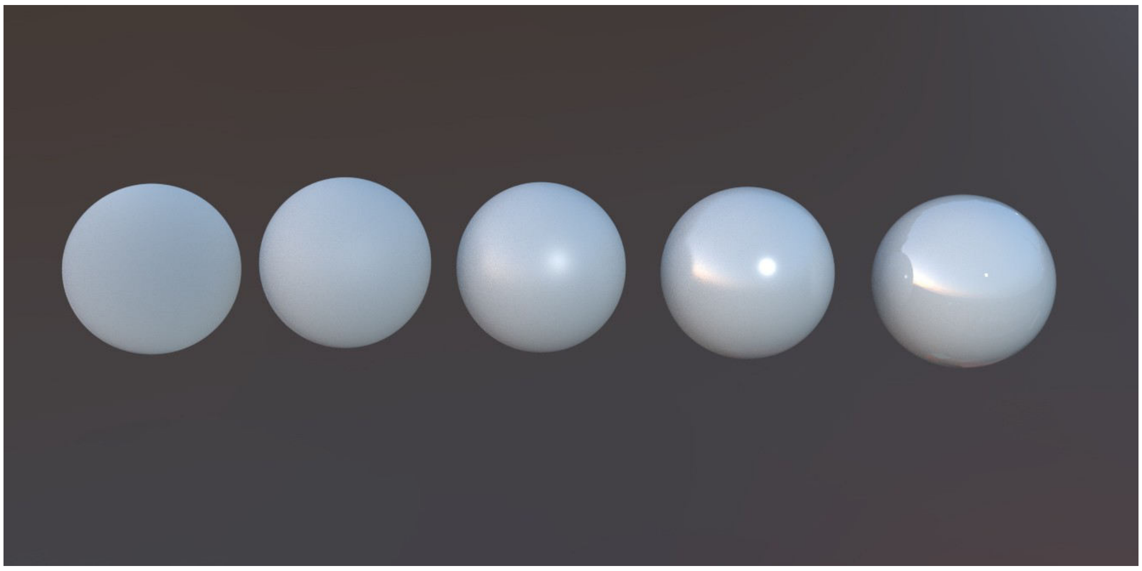
The realm of real world materials can be crudely divided in to two primary categories: dielectrics and metals.

5.1.1 Dielectrics

Dielectrics describe materials that are primarily non-conductive or conduct very little electricity. When handling dielectrics the shader needs to provide more diffuse reflection and absorption, including subsurface scattering, than when handling metals. The light that gets reflected of a pure dielectric material is always the same color. Additionally a dielectric material, when polished, exhibits greater degrees of fresnel reflections than metals. (Renaldas, Z. 2014. Unite – 2014 Mastering Physically Based Shading)

Most dielectrics absorb and let light in. For example a simple rock has degrees of subsurface scattering that affects it's appearance. (Renaldas, Z. 2014. Unite – 2014 Mastering Physically Based Shading)

When an artist is producing a dielectric material a simple diffuse shader with slight reflection might not be enough. We need to consider the microstructure of the material and whether or not it absorbs light in order to produce a realistic result. However in terms of realism it is up to the artists discretion whether or not it is usefull for any given project to produce a more advanced dielectric material.



PICTURE 17. Dielectric materials produced by the Cycles PBR Shader (Joonas Sairiala 2015)

The primary differences between pure dielectric materials comes from the surface microstructure and the diffuse reflection color. (Renaldas, Z. 2014. Unite – 2014 Mastering Physically Based Shading)

5.1.2 Metals

Metals are materials that conduct more electricity than dielectrics. These materials are very good mirrors due to a high amount of free electrons that prevent light from entering the material. Most metals are distinguishable from each other through their tinted reflections. (Renaldas, Z. 2014. Unite – 2014 Mastering Physically Based Shading)

5.1.3 Real world materials

While the division between material types seems quite simple we need to remember that a pure dielectric material or a pure metal is often hard to find in the real world. (Renaldas, Z. 2014. Unite – 2014 Mastering Physically Based Shading)

When observing the real world we can see that most materials have impurities and variations that affect how they reflect light. This means that in order to produce realistic materials we need to consider these variations and what they mean for any given material type.

Dielectrics can occasionally have properties that produce tinted reflection color and based on the microstructure of the surface the fresnel effect can be less than anticipated. Metals also exhibit, sometimes extreme, impurities like rust, oxidization and dirt. (Renaldas, Z. 2014. Unite – 2014 Mastering Physically Based Shading)

When creating materials understanding the division between metals and dielectrics is important, but we need to keep in mind that more often than not most materials exhibit qualities from both categories through various ways. The crude division and the rules that govern them are not strict, so ultimately obervation and careful consideration is key.

For example in a dirty metal material we would need to have qualities from both categories. The metal might have varying degrees of mircrosurface roughness through wear and tear and a colored reflection, but the dirt on top of it would absorb and diffuse light without any tint to the reflected light.

This principle of impurities in real world surfaces is what needs to be addressed in any production assuming realism in their materials as it affects directly on how an artist produces textures or shaders for said materials. If the material in question appears to be clean or pure it is likely to be affected by far more than we can actually perceive and take note of. Even if the effect of impurities is minimal, in order to produce a realistic material in a CG -environment we need to add dirt maps or texture maps, so that the rendered objects produces a more interesting result.

5.2. Building physically based materials

As stated previously the concept of PBR is far more than the underlying math behind the shader models. Physically based rendering introduces a wide variety of high level workflow guidelines that affect how the entire production works. These guidelines help to reduce the amount of guess work when it comes to producing physically based content allowing for what in my experience is a much faster per asset completion time than previously.

Over couple of years the Blender community has seen a number of various shaders created by the users for Cycles that are able to use common PBR inputs to an extent. The community has provided itself with a collection of shaders to choose from, but in terms of PBR workflows and how Cycles can fit in to any type of offline PBR pipeline it is important to understand how we can approach creating a physically based material.

Since physically based rendering workflows and paradigms for content creation are not enforced in Cycles it is easy to assume it requires another type of workflow, or that the existing user created Cycles PBR shaders are somehow inferior to PBR shaders in commercial engines. This however is not strictly true. As I have outlined previously a Cycles shader is as complex and intricate as you make it. While some limitations obviously exist, the ability for the artist to create realistic materials in Cycles is immense. I have outlined the process of building physically based materials below in detail.

5.2.1 Cycles Material Workflow

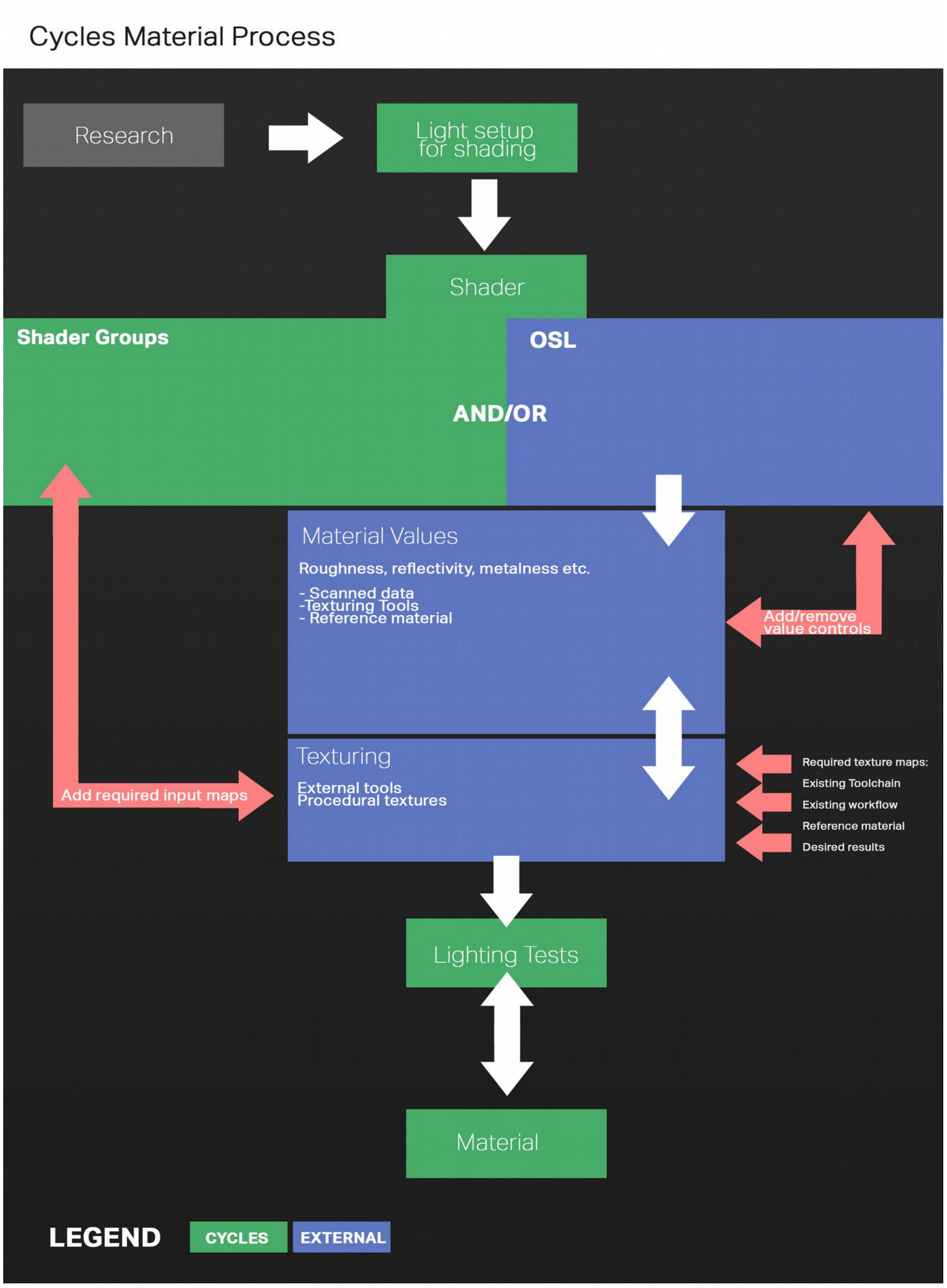


FIG 4. Physically based material building workflow (Joonas Sairiala 2015)

As the basic principle of physically based rendering revolves around observations of the real world, we can use that as the first step in creating realistic materials in Cycles. Based on the basic principles of PBR I have laid out a simple workflow diagram above on how I would approach creating physically plausible materials in a PBR environment.

This diagram provides a look in to the conceptual process of material creation in Cycles as well as in similar render engines.

As described in the diagram figure 4, the process of building a material begins by researching how the material should work. This is according to the previously outlined basic princibles of PBR and provides the decisions an artist makes a base in reality.

The material can be broken down in to pieces based on dielectric components and metals. By observing how the object reacts to light we can determine what type of BSDF components we need in our shader and how they should be mixed together in order to produce the material with Cycles.

The BSDFs and the values we give them are the components that do the heavy lifting in terms of calculating light interactions, all we need to concern ourselves with is which BSDFs to use and how to mix our components together so that the material remains physically based.

It is worthy of noting that most commercial PBR capable render engines provide a master material which determines the BSDFs and their mix for us and we are left only to research and determine the value inputs. Cycles currently does not do this so we are tasked with creating the master shader ourselves and even if it would provide a master shader it would be beneficial to create our own master shader based on the needs of any given production.

When building the shader we need to address what sort of inputs we need for it. In terms of a physically based pipeline the pre-existing toolchain or chosen work methods would dictate how we build our shader network. Determining inputs required for the user is dictated what the texturing process is like or vice versa. In terms of material building in the context of Cycles we can pre-determine how to control various aspects of our shader quite freely and even end up using custom input maps that determine how each BSDF get mixed together.

The workflow diagram serves as the basis for all materials I have used in my practical examples and applies to Cycles as well as other physically based rendering engines.

6 Cycles PBR Shader

6.1. Choosing parameters

For the purposes of creating a PBR shader suitable for Cycles I looked at how various different render engines and pipelines utilised a physically based workflow. As I have outlined previously the underlying concepts behind a physically based environment require the material shader to be able to produce predictable and consistently realistic results. This essentially meant that the PBR shader I created had to be able to utilise values and parameters that are intuitive and based in reality.

Insight in to an intuitive and physically based shader gave the Disney principled BRDF, which is structured in a way that provides control and predictable parameters.

The Disney principled BRDF approaches shading and PBR in a way that is art directable and not strictly physically correct, but still manages to produce realistic results with ease. The parameter range according to the principles of Disney's reflectance model should be 0 to 1 and all combinations of parameters should be as plausible as possible.

(Burley, B. 2012. Physically-Based Shading at Disney, 12)

While the technical aspects of a Cycles shader and the Disney BRDF are far apart, the conceptual principles behind the Disney BRDF are excellent guidelines for not only Cycles but a variety of different render engines utilizing the PBR paradigms. Due to the universal and intuitive PBR concepts of the Disney BRDF I chose to build the Cycles shader with a similar approach.

The Disney principled BRDF utilises the following input parameters: base color, subsurface, metallic, specular, specular tint, roughness, anisotropic, sheen, sheen tint, clearcoat and a clear coat gloss. (Burley, B. 2012. Physically-Based Shading at Disney, 12-13)

For the Cycles PBR master shader group I chose to use a similar set of parameters due to their ability to produce realistic results with little effort in addition to being physically plausible.

The parameters chosen for the shader were:

- Albedo - Albedo color dielectrics. For metals this affects the color of the reflection.
- Translucency - Only for dielectrics
- Metalness - 0 to 1 range between metals and dielectrics
- Specular - Reflectivity at normal incidence (f-0).
- Roughness - Controls microsurface roughness for both metals and dielectrics
- IOR - IOR value of both the dielectric materials and metals. This value is intended to be left alone, but for some cases it's use might be beneficial.
- Coat - Additional Glossy BSDF component on top
- Coat Roughness - Roughness of the Coat component

The workflow chosen for the Cycles PBR shader is a metalness based worklfow where all input maps except the albedo are black and white images ranging from 0 to 1. The choice of a metalness workflow with additional parameters was made purely out of personal preference and due to the fact that having greyscale images as inputs allows us to pack the images in to different channels of a single RGB image file.

It is worthy of note though that Cycles is also able to produce a shader utilizing the specular-gloss workflow.

6.2. Structure

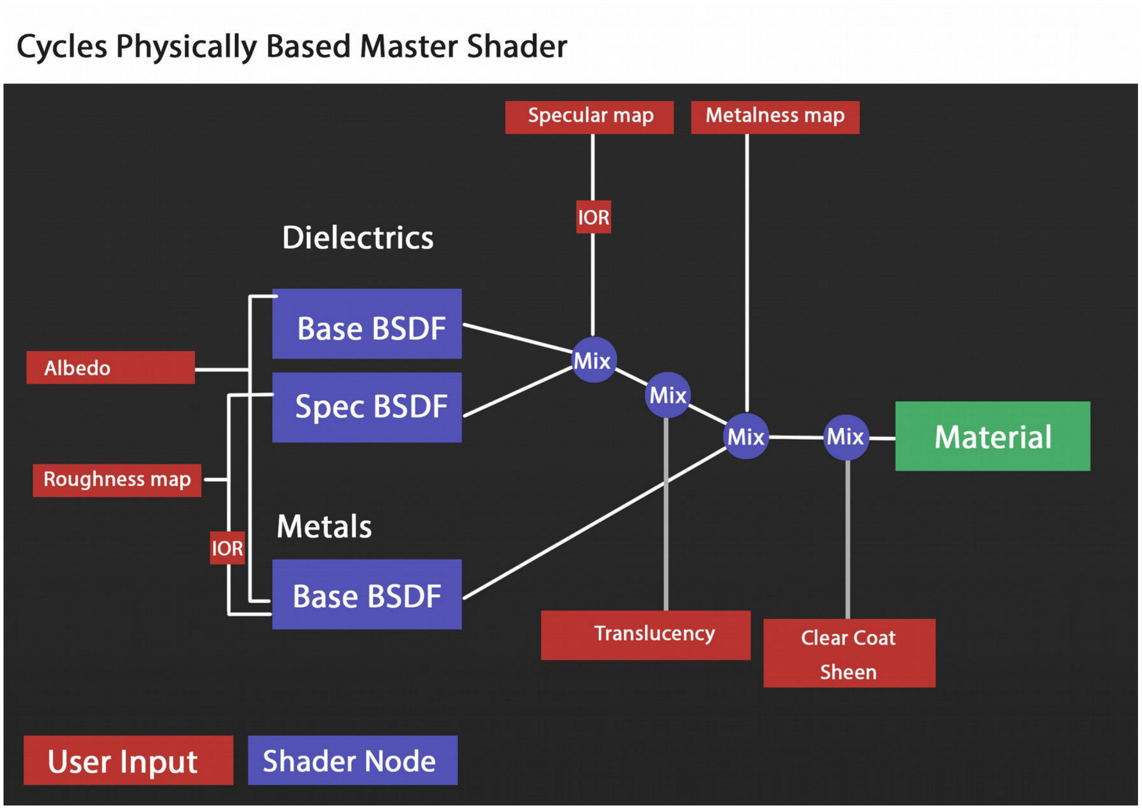


FIG 5. Cycles PBR Shader Structure (Joonas Sairiala 2015)

The shader is structured as describe in the above diagram. I chose to use the GGX distribution method for each glossy component as it seemed to provide the best results, based on visual comparisons of rendered metal and dielectric materials.

The components are as follows:

- Dielectrics are a combination of Diffuse, Glossy and Translucent BSDF
- Metals are a single Glossy BSDF
- Coat component is a Glossy BSDF
- Custom Fresnel node for mixing the BSDF together

The main PBR components of the shader are mixed together using either scalar parameters or an input map with a value range of 0-1. On top of the main components the shader mixes translucency for dielectrics and a coat layer for both material types.

6.2.1 Reflectivity

Due to the way Cycles' shaders work the combination of different BSDF dictate how the materials reflectivity is handled.

For the dielectrics I used a diffuse BSDF and a Glossy BSDF mix to produce the material.

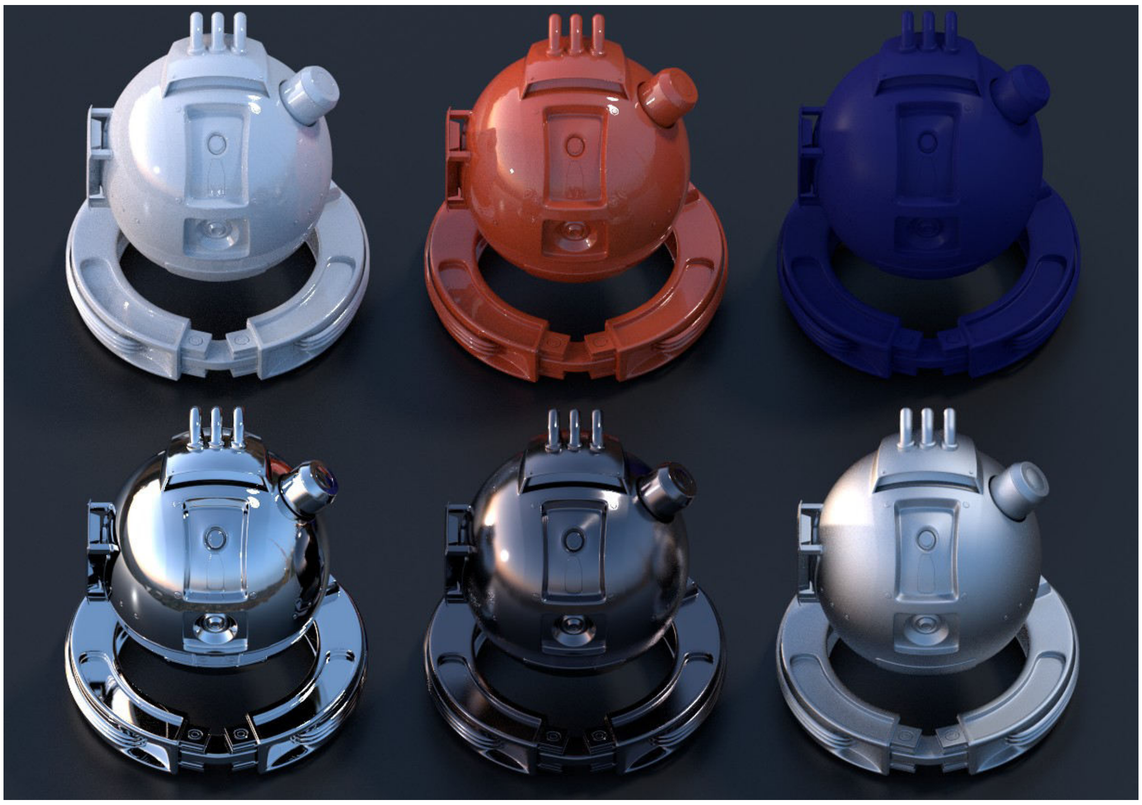
In order to utilise existing PBR tools I chose to use a specular map for the mix. The type of specular map for this shader was chosen according to the common metalness workflow, but a few additions were added in terms of functionality.

Specular maps for dielectrics should ideally provide a base level of reflectivity and the fresnel equation should handle the rest. This speeds up general content creation and ensures a physically based result. (Russell, PBR Theory)

In my Cycles PBR shader the specular map replaces the f0 values of the fresnel node's output.This means that the specular input only contributes a base level of reflectivity for dielectric materials and ensures a user has to deliberately provide incorrect IOR values for a material to not have physically based reflectivity as the fresnel node provides a physically based level of reflectivity around the edges regardless specular input.

As stated previously in section 5.1.1, in the real world dielectric materials have achromatic reflections and metals reflect back tinted light. Therefore the choice of a glossy BSDF with a white color input was used for dielectrics.

The metals in the PBR shader are handled by a single glossy BSDF. Therefore I chose to use the fresnel node group as a mix factor for the main metal color and an additional color input for the fresnel effect. The mixed colors are used as input for the glossy BSDF color, which results in a very metallic looking surface. A single Glossy BSDF was chosen for the metals as in the real world pure metallic surfaces have no diffuse component to them.


Picture 18. Various material types produced by the shader (Joonas Sairiala 2015)

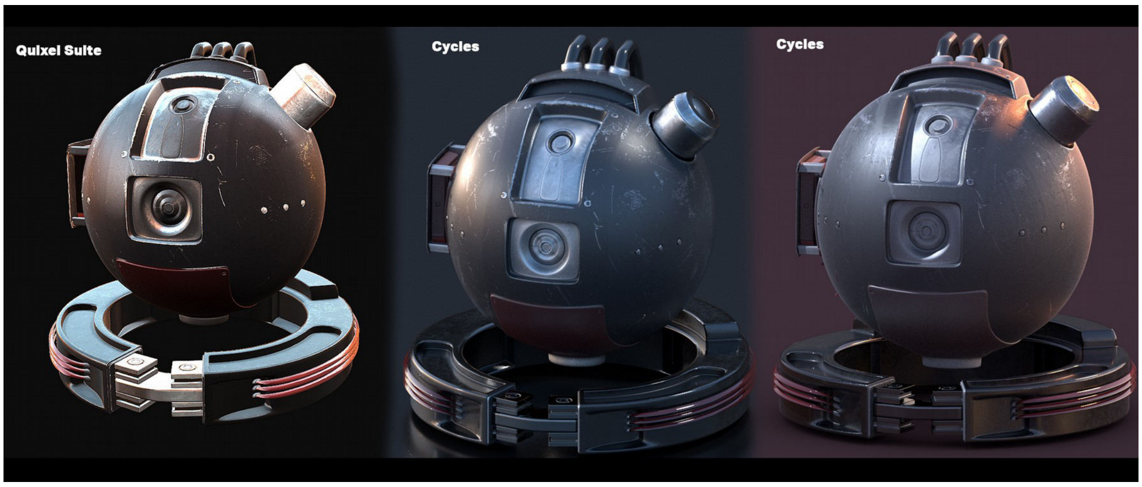
6.3. Remapping values and calibrating the shader

In order for the shader to work with common PBR texture creation tools I had to remap some of the input values. Most common PBR tools are able to export textures in to an “uncalibrated” form where all the values are represented in a linear fashion from 0 to 1.

All input values for the Cycles PBR shader have been clamped to the 0-1 range. However in order to remain physically based and utilise various scanned surface values from common PBR tools, the input values had to be remapped to a different range depending on the input. The value ranges and remappings were made to ensure that the materials created with the shader remained physically plausible.

When building the shader I utilised Quixel Suite as the texture authoring tool of choice which is able to produce physically plausible values and colors for various materials and includes an effective preview viewport that uses physically based rendering.

Comparing the Quixel Suite's viewport rendering to the final Cycles renderings allowed me to calibrate the shader and tune the value remappings in a way that would eventually produce as close a match a possible, making the authoring of textures for future projects with PBR tools more consistent.



PICTURE 19. Render comparisons from Quixel to Cycles (Joonas Sairiala 2015)

As apparent from the above image despite the changes in lighting conditions the material response is predictable across the board from the Quixel preview all the way to the two different lighting conditions in Cycles. I compared various combinations of lighting in Quixel and Cycles while tuning the remappings.

Below I have included an example prior to the remapping of the values. All of the shader components are in place, but lack any sort of calibration and remapping which produces unrealistic and useless results when using the PBR texture inputs. Essentially without value remapping the shader interprets the input textures incorrectly.

PICTURE 20. The shader before remapping (Joonas Sairiala 2015)

7 Cycles PBR Shader – Details

During my testing when comparing the render results to source material and real world observations it became apparent that in order for a Cycles shader to truly work in a PBR environment certain features had to be considered in detail.

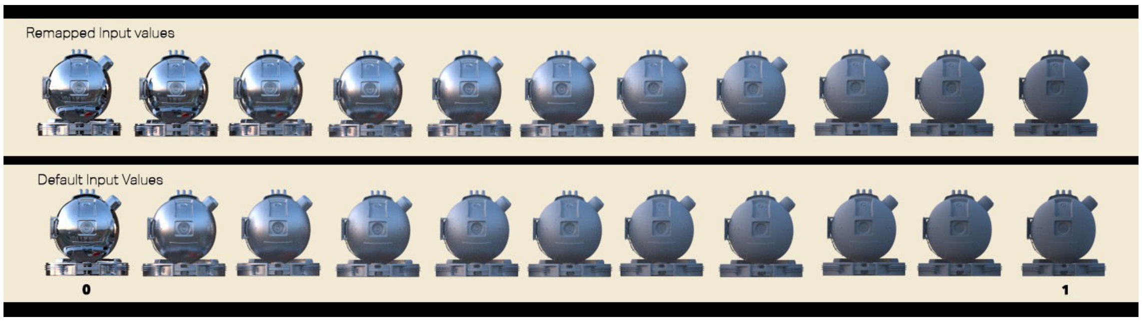
7.1. Physically based roughness

In the case of roughness it was clear from the beginning that the way Cycles handles roughness input in the GGX Glossy BSDF is not linear. The changes from 0 to 1 were non-linear, which produced some extremely low and hard to manipulate values. In addition the uncalibrated PBR export in most software assumed a linear value range.

Again going back to the Disney Princibled BRDF offered some valuable insight in to how this particular issue can be fixed.

In the Disney Princibled BRDF the roughness value is remapped to roughness^2, which produced a more perceptually linear result. (Burley, B. 2012. Physically-Based Shading at Disney, 15)

Utilizing the Disney method in roughness remapping made the distribution of roughness effectively more linear as apparent from the below image.



PICTURE 21. Roughness remapping of the GGX glossy shader in Cycles. (Joonas Sairiala 2015)

The roughness remapping done here was essential in ensuring that the shader works with the metalness workflow chosen.

Additionally in order to remain physically based the use of the fresnel node had to be approached in particular way. The fresnel node is a separate shader node and has no access or connection with the Cycles BRDF nodes used to calculate microsurface roughness and light distribution. This means that the amount of fresnel reflection is independent of the roughness value when mixing shader nodes, which is incorrect.

The fresnel effect that makes edges appear brighter and smoother diminishes as the surface microstructure gets rougher. The roughness of a surface directly affects the apparent reflectivity of a surface. (Russell, PBR Theory)

Ideally this sort of detailed phenomena should be hanlded at the BRDF level, but Cycles currently does not support the implementation of such effects, as I will outline in more detail in the conclusions chapter.

In order to achieve this effect of diminishing fresnel when the surface gets rougher I implemented a work around solution that recreates the effect sufficiently for a physically based environment by manipulating the normal input for the fresnel node.

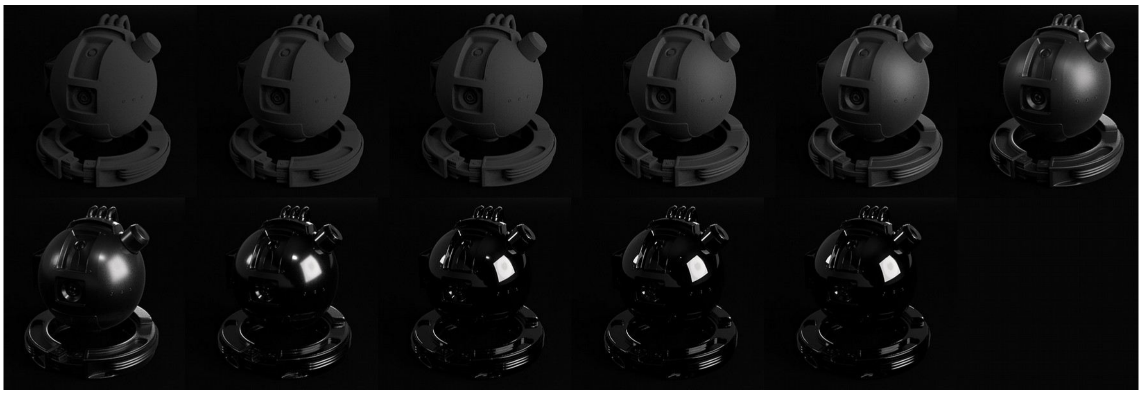
Cycles manual states two key things regarding the possible ways to achieve this.

Fresnel is dependent on the angle of the surface normal and the viewing direction.

(Blender Foundation 2015. Input Nodes)

The incoming vector input is the vector pointing towards the point the shading point is viewed from. (Blender Foundation 2015. Input Nodes)

Roughness input is used to mix between shader normal input and incoming light vector so that as roughness approaches 1 the incoming light vector takes precedence. That mix is then used as a normal input for a fresnel node which results in the fresnel node interpreting the surface normal based on the amount of roughness input, leaving only values at normal incidence when roughness is 1. This results in a situation where the fresnel node's output is tied to the roughness input in a physically based fashion.



PICTURE 22. Fresnel effect increases the smoother the surface gets (Joonas Sairiala 2015)

7.2. Base reflectivity

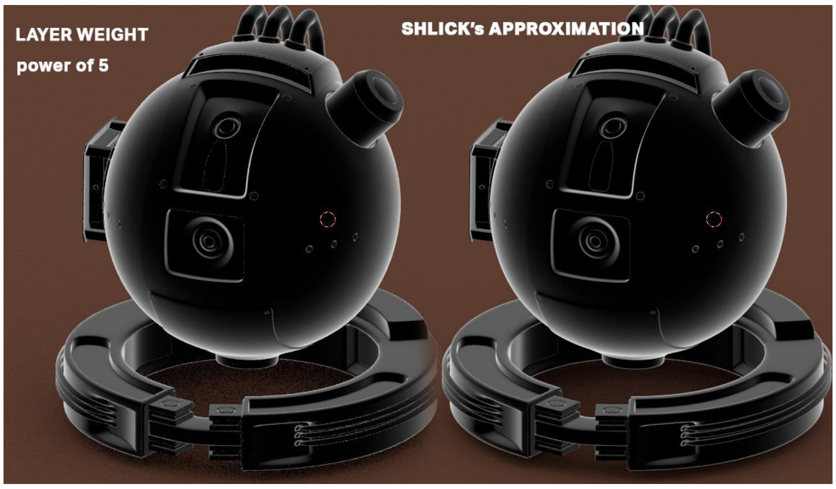
Simply adding the cycles fresnel node output on top of a f-0° base reflectivity map resulted in slightly overbright f-0° reflectivity and resulted in inherently incorrect materials as the f-0° value describes the reflectivity at normal incidence, which is already calculated by the fresnel node.
In order to remain physically plausible the use of a custom fresnel node group was necessary.

I wanted to keep the ability to utilise a base reflectivity value as for the artist it is an easy to understand value, but it cannot be used together with Cycle's default fresnel node. The fresnel node produces visually impressive results that appear accurate and realistic and the ability to use IOR input allows for physically correct materials.

Through my testing I found that implementing a f-0° reflectivity input is possible in two ways; through the use of the facing input produced by the layer weight node or by using two fresnel node inputs.

7.2.1 F0 with Layer Weight node method

The layer weight node is essentially the normalized dot product of the incoming light ray and surface normal inverted. The modified facing input of the layer weight node is almost identical to a simplified Schlick's approximation of the fresnel equation.



PICTURE 23. Layer Weight facing output compared to a simplified Schlick's approximation. (Joonas Sairiala 2015)

I implemented a simplification of the Schlicks approximation using math nodes, but as Cycles does not offer access to the micronormal half-vector produced by the BRDF it became apparent that implementing an error free Schlick's approximation in either OSL or shader nodes is not possible.

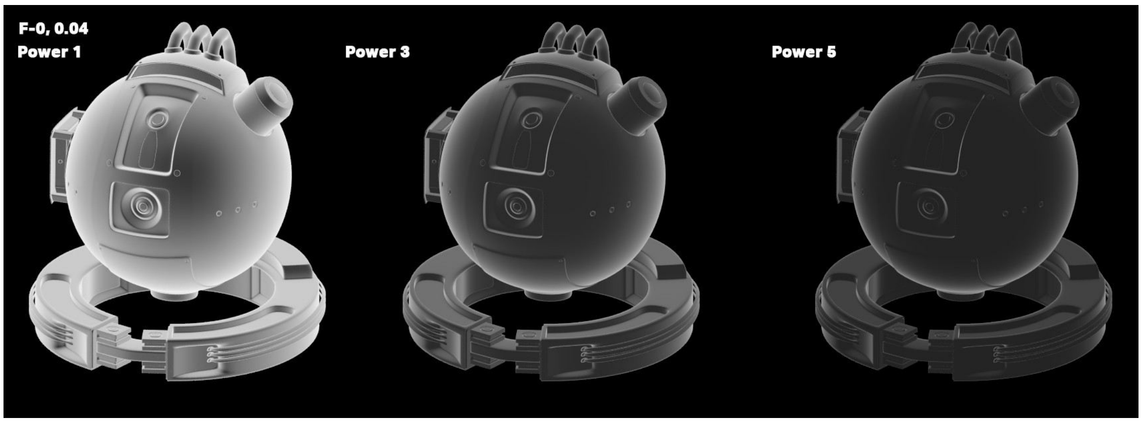
$$F_{\text{Schlick}} = F_0 + (1 - F_0)(1 - \cos\theta_d)^5$$

In the formula F_0 represents specular reflectance at normal indcidence and θ_d the angle between the light vector and the micronormal half-vector, not the angle of incidence with the surface normal. (Burley, B. 2012. Physically-Based Shading at Disney, 16)

As I will outline in the conclusions chapter Cycles uses the angle of incidence with the surface normal for fresnel calculations which is slightly incorrect.

As the layer weight node produces essentially the first part of the Schlick's equation, albeit without the half-vector, raising it to the power of 5 as the formula describes resulted in a usable fresnel effect.

The values at normal incidence are completely black so I used the color mix node to mix pure white with another value by the factor of the layer weight output. This means that whatever value from the 0-1 range is used results in reflectivity at the normal incidence. This produced a very useful fresnel effect with a lot of control available. For the layer weight node I also exposed the power value in order to manipulate the effect as needed. Manipulating the power value proved useful as it essentially modifies the perceived fresnel curve of the material.



PICTURE 24. Manipulating the power value produces various fresnel curves (Joonas Sairiala 2015)

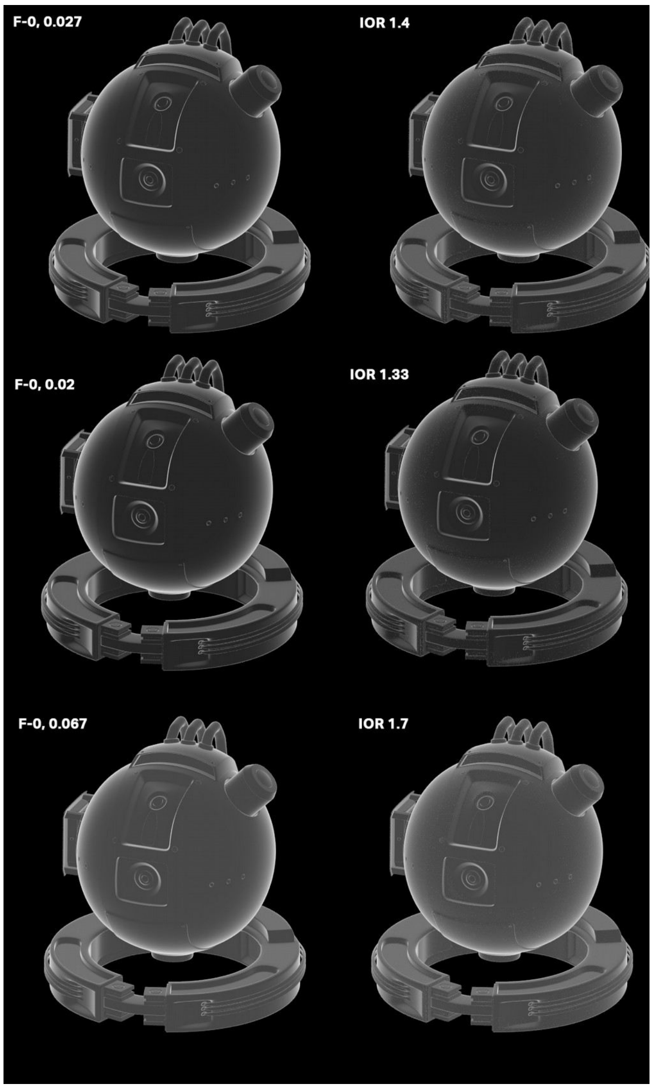
In order for the f-0° value input for the shader to be useful it has to produce physically plausible results. In the context of Cycles this meant that comparing the output of the fresnel node, which uses a more advanced fresnel equation, to the output of the layer weight node group was necessary.

I chose an IOR value and used the following formula, where n is IOR, to find the corresponding f-0° value:

$$F(0^\circ) = \frac{(n-1)^2}{(n+1)^2}$$

PICTURE 25. Formula for finding the index of refraction value (Lagarde, Feeding a Physically based Shading model 2011)

I then did s visual comparison of the two fresnel calculation methods and tuned the fresnel curve of the layer weight node to find a closer match. The comparisons were rendered to 40 samples.



PICTURE 26. Comparison between the layer weight method and the fresnel node (Joonas Sairiala 2015)

I compared the results to the default fresnel node as when working with a node based implementation the fresnel node's default output was the most accurate. As stated, the aim of this was to implement an accurate f-0° value control, which ment that the default fresnel node and the custom fresnel node group had to match closely.

From the comparison I came to the conclusion that the layer weight method matches closely with the fresnel node, but not entirely accurately.

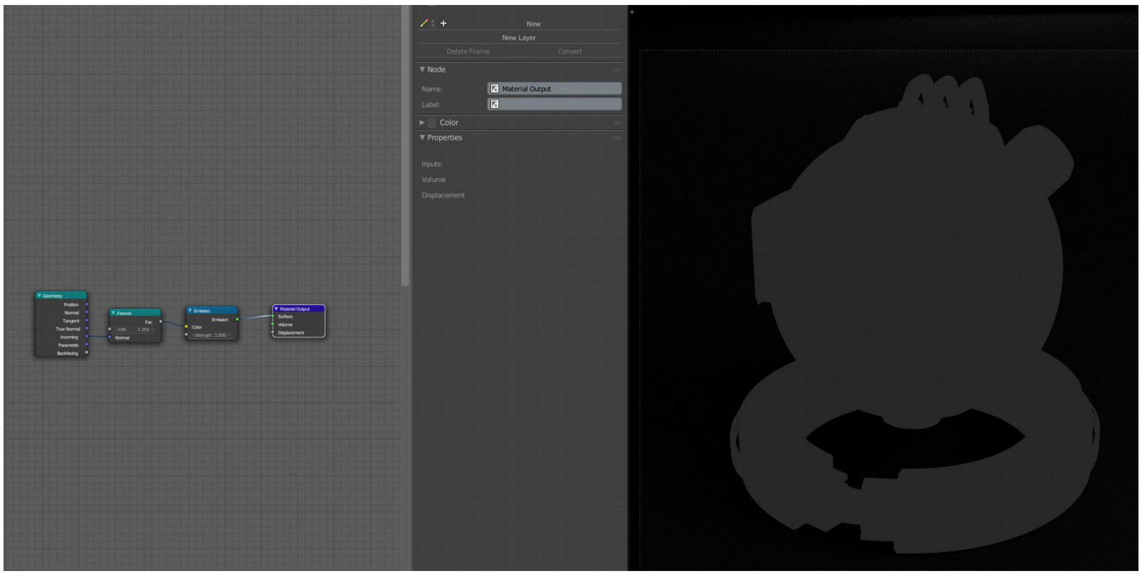
The most significant difference is that the fresnel node produces a smoother transition between dark and light values and the values at grazing angles seem to be slightly brighter for the layer weight node group. There are also some differences in the detail and visual contrast which can arguably show in certain materials. Additionally as apparent from the image the layer weight method is noise free at 40 samples while the fresnel node has some visible noise.

7.2.2 F0 with Fresnel node method

Based on my testing I chose to use the fresnel node due to it's inherent accuracy, which meant that I needed to remove the existing f-0° values using Cycles nodes. Similar to the previously described diminished fresnel as roughness increases, I isolated the f-0° values of the fresnel node by utilising the node's normal input.

When we input the incoming vector to the fresnel node shading normal input the result is the fresnel node considers the angle of the surface normal and viewing direction the same thing, outputting only the values at normal incidence.

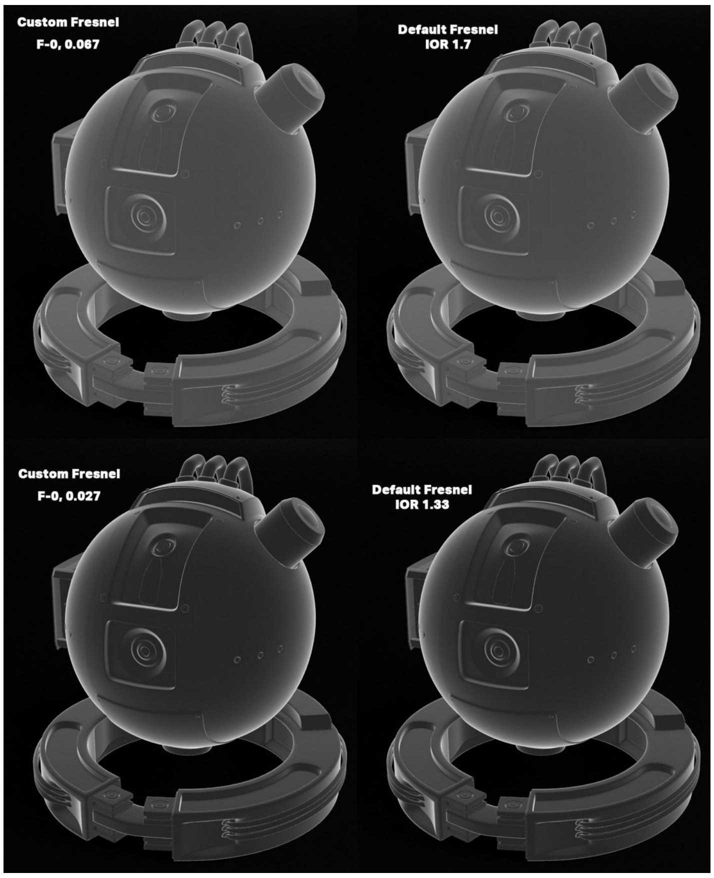
The second method of producing an f-0° input is otherwise identical to the layer weight node method, except in place of the layer weight node a second fresnel node is used. The second fresnel node is used to isolate the values at normal incidence and substracting them from the first fresnel node, which leaves only the values at angles.



Picture 27. Extracting the f-0° values (Joonas Sairiala 2015)

After the substraction the resulting fresnel node group is used to control a mix of f-0° and white to enable the use of f-0° input without compromising a physically based approach.

After creating the custom fresnel node group I compared the results to the default fresnel node's output similarly to the previous comparison.



PICTURE 28. Comparison between the custom fresnel method and the fresnel node (Joonas Sairiala 2015)

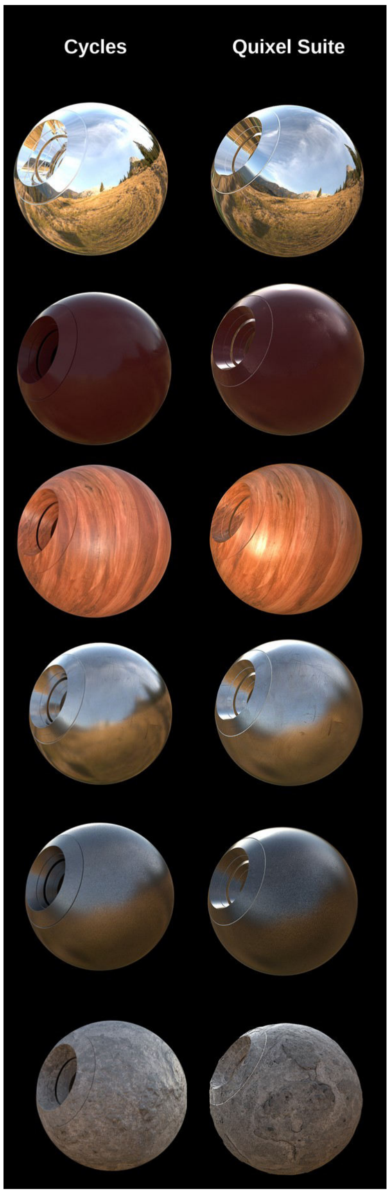
As apparent from the visual comparisons the results are identical, which means that the shader is able to utilise base reflectivity maps, which provide a far more intuitive and physically based way of authoring materials than using a fresnel node and an IOR input.

For the custom fresnel group I exposed the IOR value, which means that for additional control we can set an IOR value which affects the edge reflections only, but this value can be left alone at 1.5 for most common materials.

8 Cycles PBR Shader – Testing the shader response with Quixel Suite

For the primary testing of the shader I chose to use Quixel Suite as it is one of the most commonly used PBR authoring tools that offers scanned surface data.

I a variety of materials to test in both Quixel Suite's 3Do viewport and Cycles render engine. For both renders I used the same HDRI environment map to produce identical results.

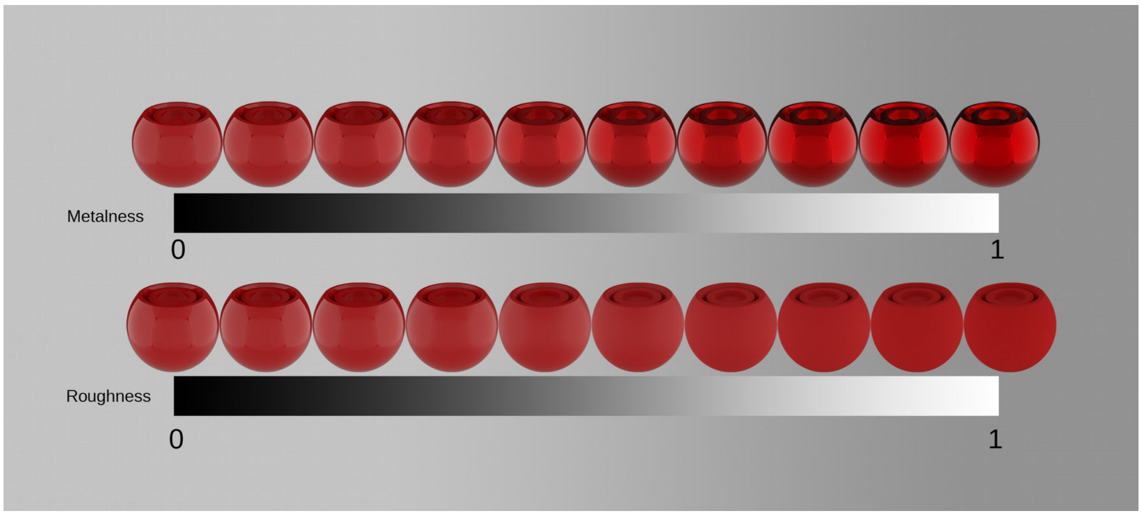


PICTURE 29.Comparing the shader output to Quixel Suite (Joonas Sairiala 2015)

From the comparison it is evident that shader is able to adequately reproduce most materials and as Cycles is a more accurate renderer it is able to produce   more detailed lighting, which in these examples resulted in less bright highlights. The most important criteria in this comparison was whether or not the base amounts of reflectivity, roughness and overall albedo color matched. Differences in the render are also introduced by not using a displacement map in Cycles.

8.1. Value Chart

Below is a value chart on how the shader handles metalness and roughness inputs. Out of the inputs I chose to only include the metalness and roughness as those two are what seem to be the most engine specific ones of the PBR inputs.



PICTURE 30. Value chart for roughness and metalness (Joonas Sairiala 2015)

9 CASE 1 Cycles PBR Shader – The first look in to PBR

9.1. The Pocket watch animation

The clock animation is a short animation clip of a pocket watch set on a watchmakers table. The purpose of this animation was to apply the principles of physically based rendering workflow in practice and produce a very specific type of material from scratch that would work for multiple lighting conditions and behave predictably during animation.

The pocket watch was modelled in Blender loosely based on various reference images of antique pocket watches.

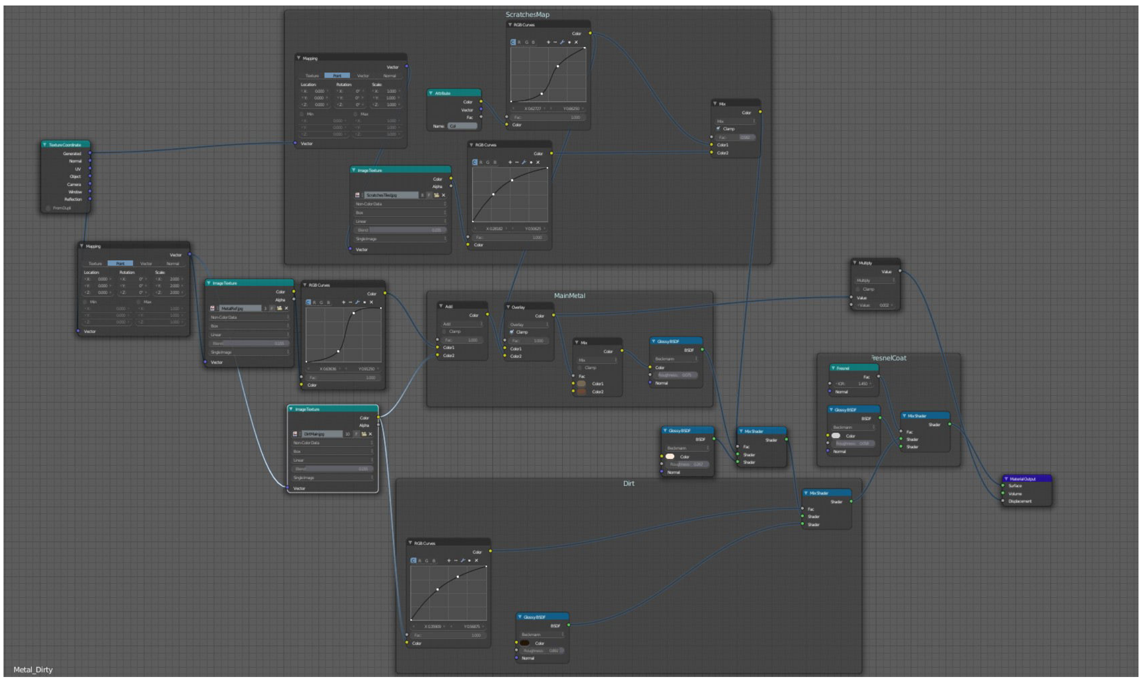
As per the basic principles of physically based rendering observation of the real world was used to determine how the material should react to light and how to approach it's creation. This was done by examining the reference images closely.

A pure PBR master shader was not used for this case study as I built the shader for the watch specifically from scratch utilizing a physically based material workflow in order to test out various BSDF combinations and input values. This was the first case study that laid out the base for my Cycles PBR shader and the following two case studies.

9.1.1 Shading & Texturing

The pocket watch was shaded and textured in a environment with a single studio HDR image providing lighting and reflections. This was due to the fact that the metallic surface exhibits a high degree of reflectivity so in order to see the results of a highly reflective shader the object needed something to reflect all around it. After the initial shading was done the watch was moved to it's final setting.

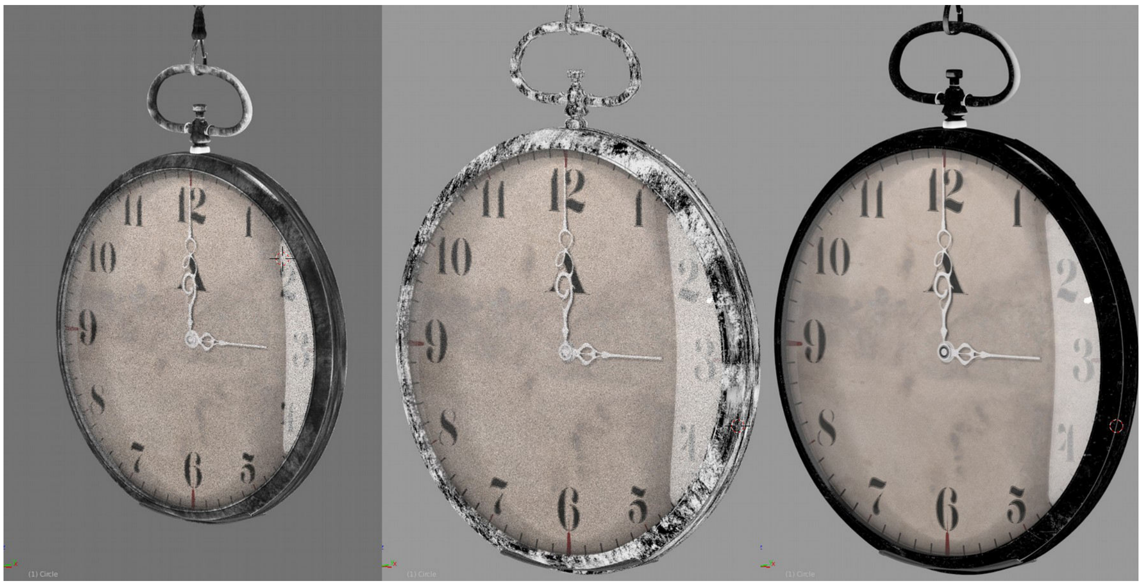
The shader is built as follows: The metal component is a mix of three instances of a glossy shader, one of which is responsible for producing the first layer of reflections coming from the base metal material and one which produces rough reflections based on a scratch texture map. The glossy component overlayed on top is mixed to the previous shaders based on the material's IOR value of 1.35 using the fresnel effect. In between them the surface shader has a diffuse shader with a texture control that adds dirt and grime to break up the smooth glossy surface and gives the shader an old, worn look. The approach taken here was slightly different to a standard PBR shader as every single part of the metal had it's own BSDF. In this case I chose this approach due to extreme close ups of the pocket watch and the nature of the surface, which appeared to have miscoloration, oxidization and wear with very diverse characteristics from each other.



PICTURE 31. Shader nodes for the antique metal (Joonas Sairiala 2015)

The base metal here receives it's color, including miscoloring, from a color mix node which mixes two colors together based on a black and white texture input. The colors chosen for this were eyeballed based on observation and reference images as opposed to picking them from a chart or a scanned material preset in an external software. The simple studio lighting HDR ensured that the lighting did not contribute any colored light to the shader and enabled me to adequatly judge whether the choice of color was right.

Each of the texture maps have been tuned for the Cycles render engine through the use of Cycles nodes before inputting the value into the main shader in order to avoid too high or too low values. Certain physically based values were used as a basis for the maps, but ultimately the texture maps were tweaked heavily in Cycles in order to achieve the desired results. This was simply due to the fact that the shader was built for this object alone and the values were adjusted on the fly as opposed to remapped inside the nodegroup.


PICTURE 32. Textures for Base color mix, Dirt & Miscoloration mix and Scratches mix (Joonas Sairiala 2015)

Every texture, except the clock face, on the watch is a simple tiling grunge texture, box mapped on to the model. The textures were created in Photoshop CC with a collection of grunge and scratch brushes.

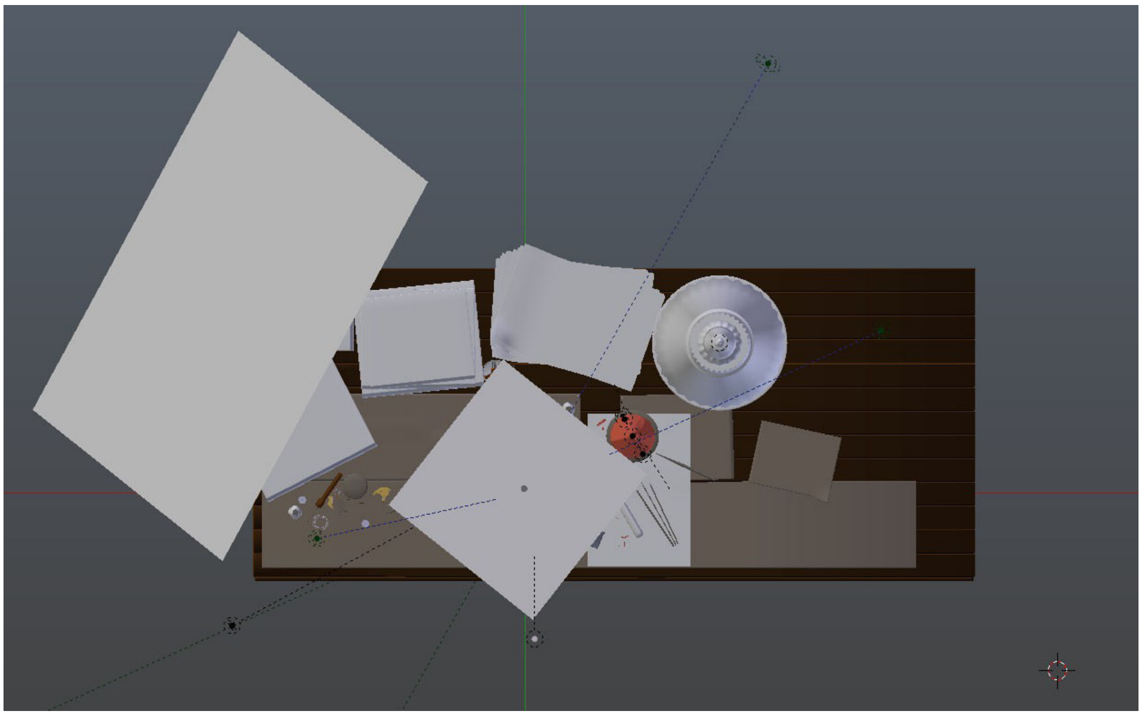
PICTURE 33. Different shader components before being mixed together.(Joonas Sairiala 2015)

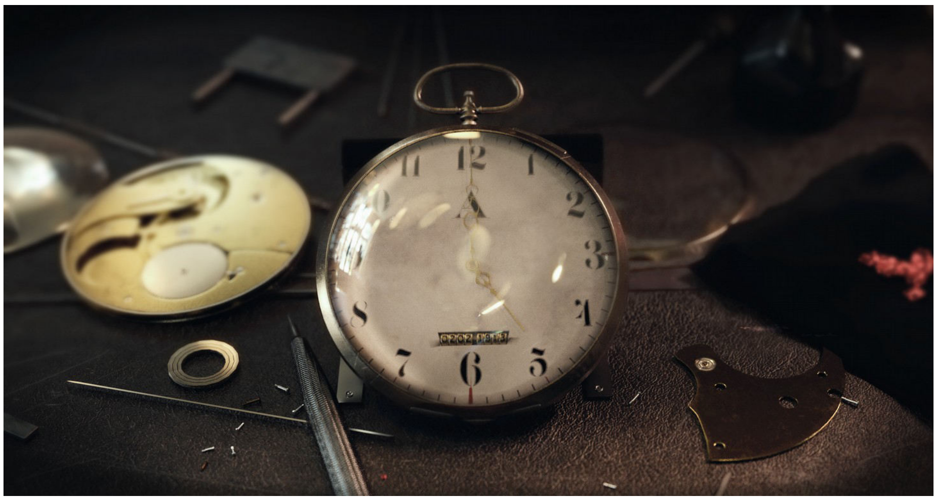
9.1.2 Lighting

For any PBR shader it is imperative that the lighting is adequate as reflections are in the case of PBR true rather than faked. For the materials to work a proper lighting setup needed to be used. A similar lighting setup was used for all of the case studies as well as render tests.
The final scene was lit with a combination of image based HDRI lighting additional area lights and image based mesh lights.

The primary concern for lighting this scene was getting enough reflections to show up on the watch in an otherwise quite dark and moody environment, which is why extra image based mesh lights were added in front of the watch and certain parts of the scene.

Below is a top down view of the 3D-Scene with the image based mesh lights visible. The key light is coming from the front of the clock, a fill light from both sides and additional complementary effect lights were added around the scene where needed.


PICTURE 34. Top down view of the 3D-Scene (Joonas Sairiala 2015)



PICTURE 35.The final clock render (Joonas Sairiala 2015)

10 CASE 2 Cycles PBR Shader – Texture Authoring with Quixel Suite

In this case study I will describe how the overall process behind authoring textures for the PBR shader works in Quixel Suite.
In this case I created a sci-fi character model with various different surfaces each with their distinct qualities that needed to be portrayed.

10.1. Preparing the model

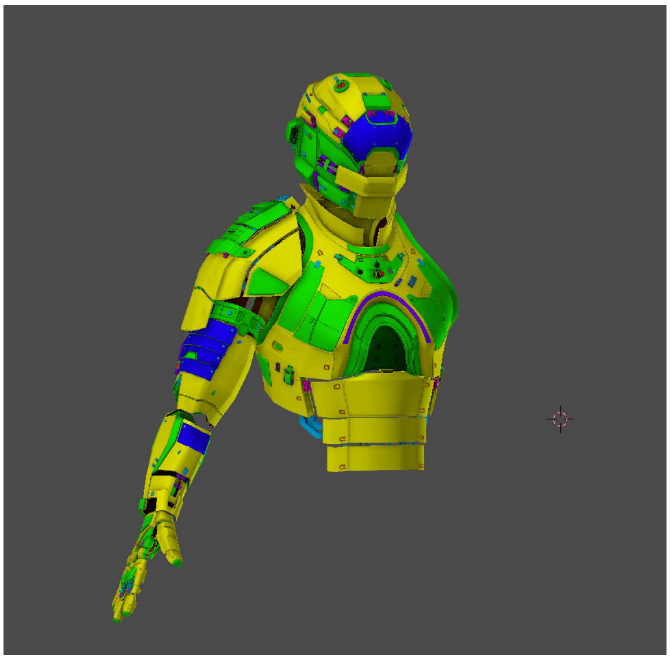
With the character model UV-unwrapped I started to develop the look of the model by applying materials with 3D -viewport colors and outlined which different materials were needed.

At this stage it was important to distinguish all the possible different materials as Quixel uses a baked color map to determine which part of the model any given material goes to. Addionally at this stage I baked curvature, object space normal and ambient occlusion maps for use with Quixel Suite.

(c) Joonas Sairiala

PICTURE 36. Look development (Joonas Sairiala 2015)

During the preparation I also took in to consideration the artistic aspects of the texturing and shading process. With this model I wanted to portray a decorated sci-fi soldier in an alternate future in a realistic manner so paying attention to how the character's armor would actually be structured was necessary as it helped in choosing a collection of plausible materials.

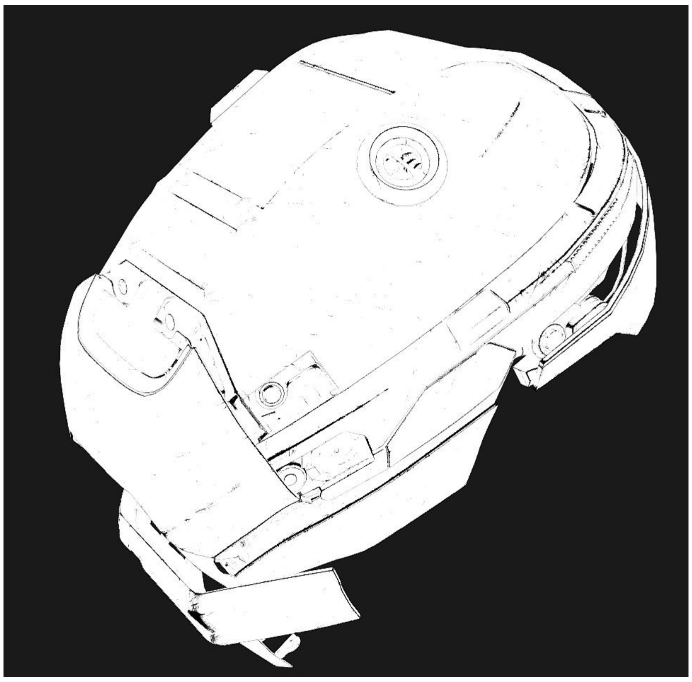


PICTURE 9. Material ID map (Joonas Sairiala 2015)

10.2. Texture creation process

For this case the texture creation process for fairly straightforward. In Quixel Suite I started building the materials piece by piece, starting with the base layer of each material using masks generated in Quixel Suite to transition between them.

For example in the case of the white painted steel armor started building the material from the steel preset provided by Quixel Suite. On top of the steel I added a paint material preset and used a generated mask with some hand painted details and fixes.



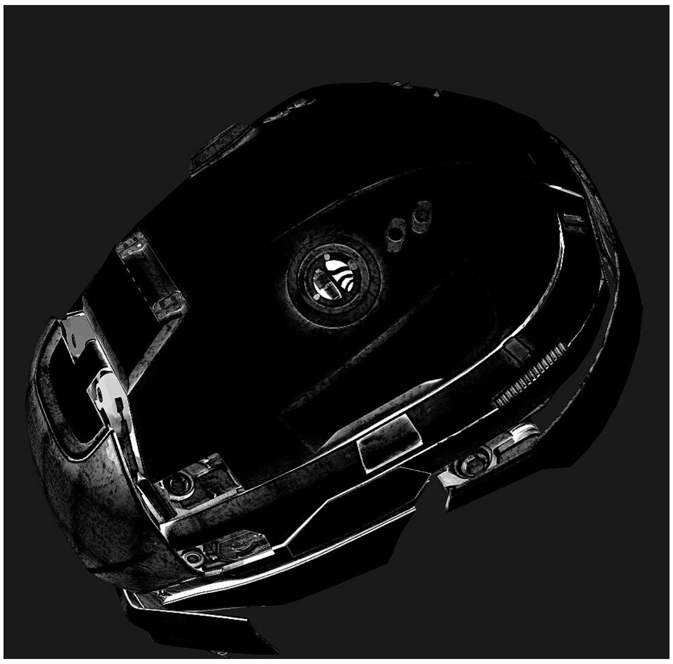
PICTURE 37. Mask between steel and paint where black values represent steel. (Joonas Sairiala 2015)

It is important to note that in the case of Quixel Suite the presets include measured material values for the various surfaces, which gave an excellent baseline to work from. This meant that for most materials I did not have to change the reflectance values or modify the roughness that much, which sped up the content creation process substantially.

During the creation process I also referred to a variety of reference images of real world counterparts for the materials to ensure that I understood how the material should be built. Below is an example reference image used to determine various ways a painted metal surface ages, scratches and miscolors.

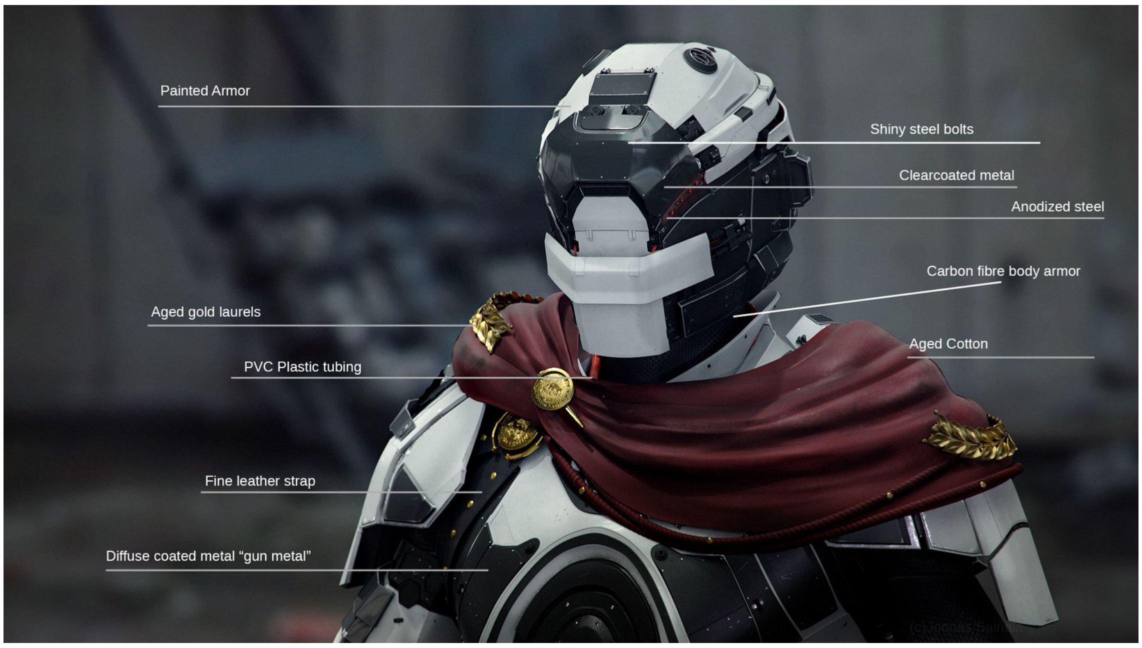
PICTURE 38. Reference image for white painted armor (Wikimedia Commons 2010)

The process was repeated for each material used in with the model. Each material ended up consisting of multiple components from the base layers of the material to an overall dirt layer. In short, each material was broken down in to their most basic elements and rebuilt from scratch in order to ensure the final result has plausible amounts of variation in reflectance and roughness values in addition to changes in albedo.

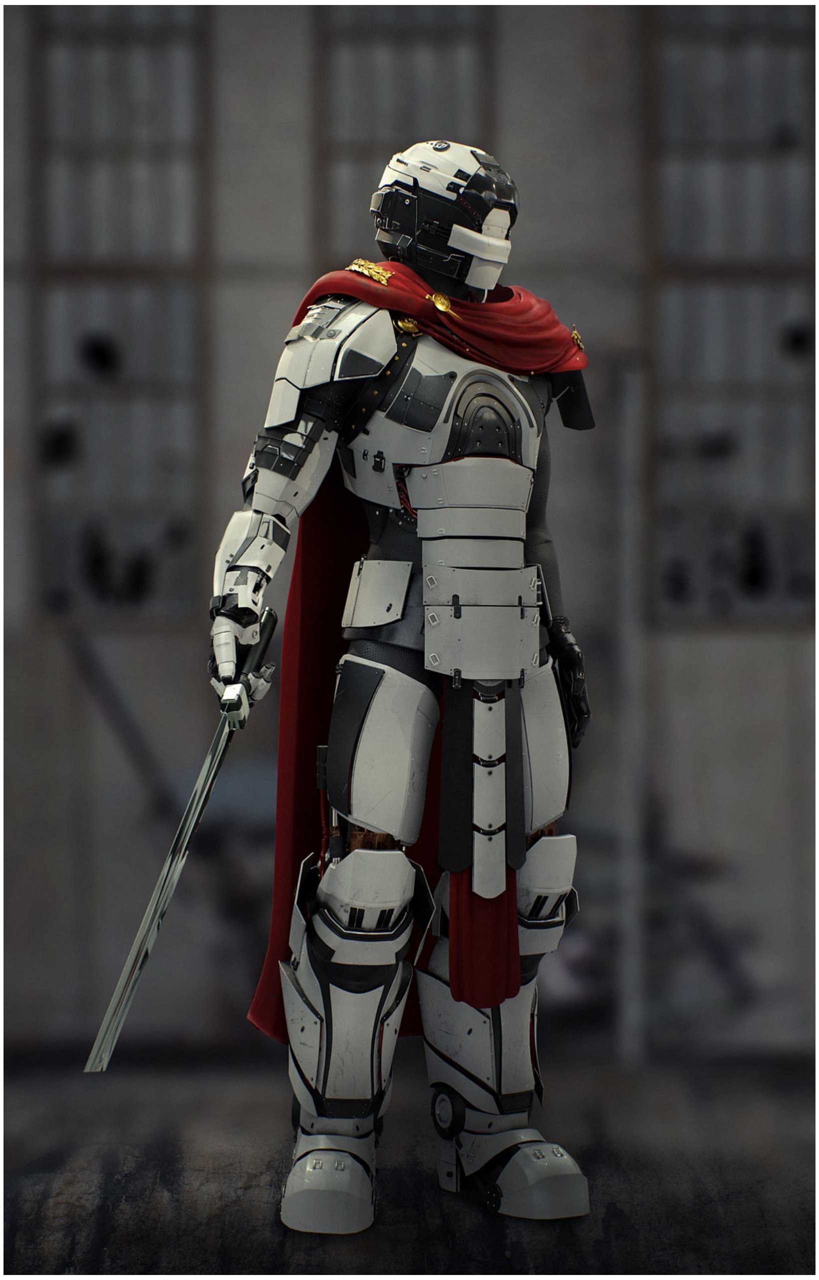


PICTURE 39. Dirt mask where white represents dirt. (Joonas Sairiala 2015)

Below is the final render with each different material superimposed on the image to illustrate the thought process behind choosing the materials.



PICTURE 40. Sci-fi character materials (Joonas Sairiala 2015)



PICTURE 41. Sci-fi character render (Joonas Sairiala 2015)

10.3. Problematic materials

For the sci-fi character render there were a couple of things the shader portrayed with a few issues. Primarily the biggest issues was the aged cotton which with the shader looked very plastic and hard and did not match the reference images that well. For improving the rendering of fabrics such as cotton the shader would need to have a separate node setup that would include more attributes such as translucency.

Additionally for the shader I had to tint the fresnel color of the aged golden laurels in order to get a closer result to the reference images, which ideally should be handled by the metallic component of the shader, but that is not currently possible with the shader.

11 CASE 3 Cycles PBR Shader – Texture Authoring with masks in Blender and tileable textures

The second case study I did was specifically geared toward not using a single texture map for the materials, but separate each material component in Blender utilizing multiple instances of my Cycles PBR shader and using masks with tileable textures to combine them.

This method allowed me to produce some fairly realistic results wihtout extensive usage of external tools and gave a lot of control over the materials at rendertime.

The 3D-model used in this case is a model of a large scale v-tail quadcopter with various metallic, leather and plastic surfaces.

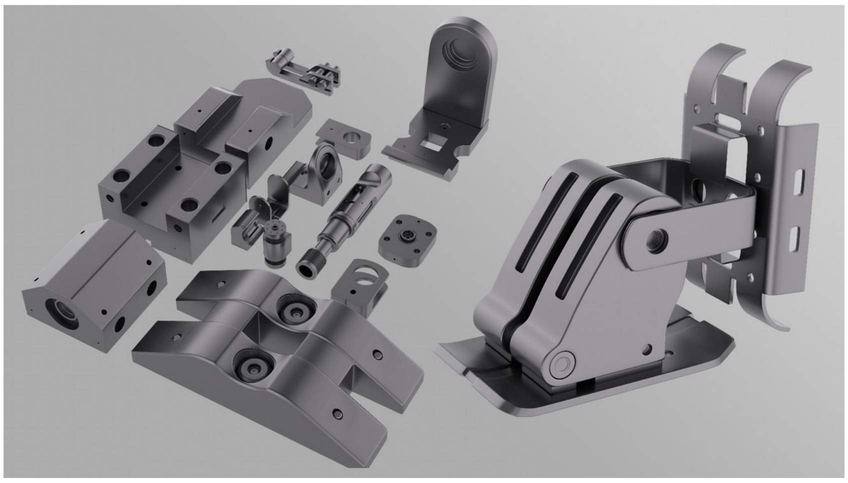
11.1. Creating base materials

The first part of this case was defining the materials and comparing them to reference images similar to the previous case studies.

All of the base shader inputs from color to base reflectivity were sourced from a Quixel megascans material and tweaked so that the apparent reflectivity and apparent color matched various reference images.

PICTURE 42. V-Tail base material values(Joonas Sairiala 2015)

For each material I used the PBR shader to create three distinct components: base material, possible miscoloration or scratches and a dirt layer. Each component was then mixed together using a combination of a tileable texture and a vertex color map adjusted by the RGB curves node.

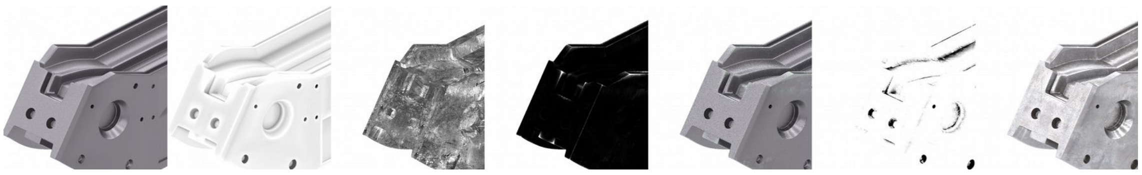


PICTURE 43. Base values for aluminium(Joonas Sairiala 2015)

PICTURE 44. Unmodified generated vertex colors(Joonas Sairiala 2015)

Each material uses only a single tileable texture input to modify the mask and only the leather uses a separate albedo texture. The rest of the materials get their surface miscoloration from the combination of the various material components, namely the dark dirt component.

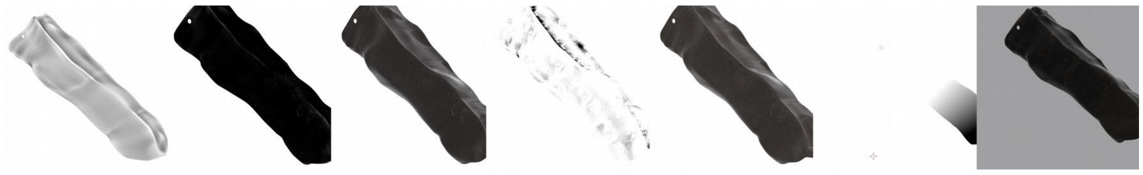
Below are example images that illustrate how each layered material is built. From left to right: base material, vertex colors, texture map, mask for wear around the edges, worn out material, dirt mask and finally the full material.



PICTURE 45. Aluminium material with surface detailes(Joonas Sairiala 2015)

While making the masks and different material components I consulted various reference images to produce a fairly realistic material. In the case of the aluminum for example as the fairly polished aluminum base was worn the reflections became more diffused and rough.

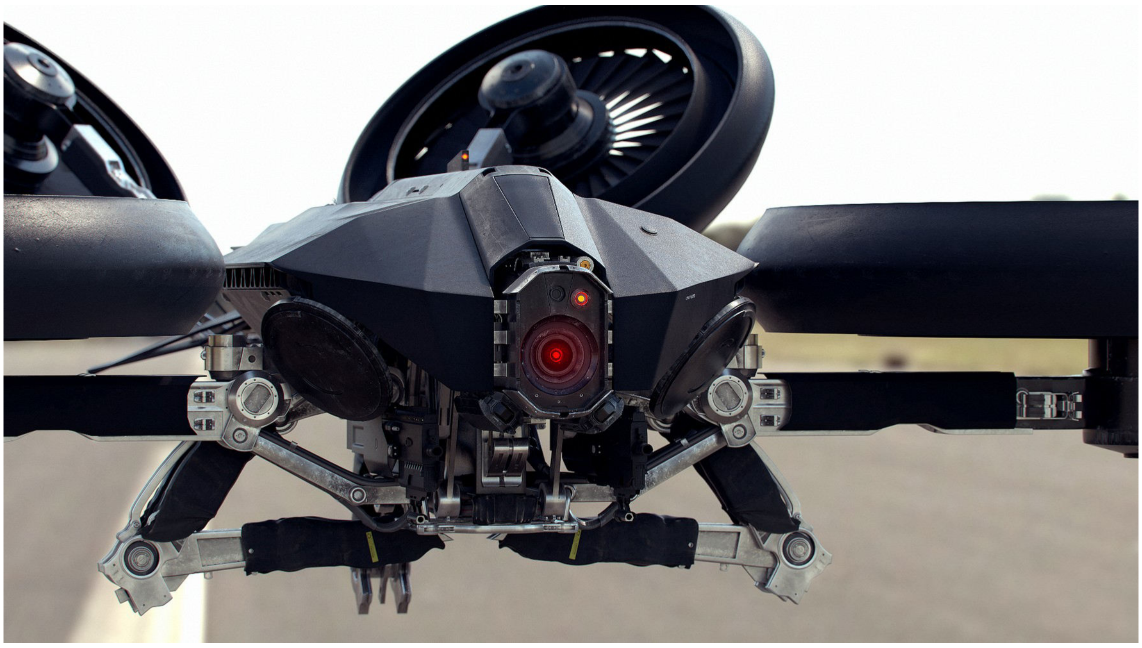
PICTURE 46. V-Tail materials with texture inputs(Joonas Sairiala 2015)



PICTURE 47. Leather material components with a gradient mask for directional dirt(Joonas Sairiala 2015)

When applying the materials to other parts of the model no adjustments were made as the whole setup worked simply by generating vertex colors for each separate object.

PICTURE 48. Close up of the dried leather material applied to another part of the model. (Joonas Sairiala 2015)


PICTURE 49. Final render of the V-tail (Joonas Sairiala 2015)

12 Conclusions

12.1. PBR Workflows and Cycles

12.1.1 PBR as a concept

At the beginning I was unsure whether or not PBR would turn out to be that effective of an approach with Cycles or other offline renderers as the reason it has become a popular term was it's introduction to real time renderers. However after studying the subject and impelemeting a PBR shader in Cycles I came to the conclusion that physically based rendering offers much more than it's technical aspects. For the artist PBR as a concept offers a wide range of effective workflow guidelines and a set of effective cross-platform tools that allow for the creation of proper textures efficiently.

12.1.2 Master shader

Implementing a PBR -capable shader with Cycles nodes was succesful. The shader responded to the texture inputs in a predictable and realistic manner and without trouble. Working with a master shader similar to the PBR -shader is certainly beneficial to any Cycles project as it eliminates the laborous process of individually shading each different material from scratch.

Cycles would definitely benefit from an official standard master shader implemented in code in addition to it's powerful node based system. This would allow for beginner artists to produce content with more intuitive inputs used throughout the industry.

A master shader like this would also enable Cycles to be used as part of a production where each tool used needs to work together seamlessly and predictably. Most studios working with Cycles undoubtedly have a master shader and develop their own inhouse master shader, but from the perspective of the individual Cycles user or from the perspective of someone who uses a different render engine Cycles can appear extremely complex and unintuitive when shading even the most simple objects. Therefore implementing an easy to use master shader would greatly benefit Cycles.

Based on my personal experience with render engines like V-Ray, Octane render and Keyshot it is evident that Cycles lacks usability above all else. Without an out of the box master shader it takes even an experienced artist much longer to produce a good shader in Cycles than it does in V-ray or Octane.

12.1.3 Workflows

The common PBR workflows provide the artist a much more intuitive and reasonable way of creating shaders and materials. In a PBR environment it is not relevant to work at the terms of the software, but rather at the terms of what you are creating. A more detailed reasoning behind why certain materials look apart in the render that has a basis in reality brings the authoring process closer to reason and enables artists to produce realistic content with less time spent on thinking about how the render is going to interpret their work and more time spent on thinking about the artwork itself.

Based on my experiences working with the case studies presented here I came to the conclusion that producing content for a PBR shader is much more intuitive and faster than other shader types. I was able to utilise various reference materials and an intuitive approach in creating texture content for the shader without having to constantly render and tweak the materials. I was able to work with Quixel Suite first and after moving on to render in Cycles the results were exactly as expected. The greates benefit of this is that working with a PBR shader allows the same texture content to be used in various different PBR capable renderers with little no tweaking and learning to texture for a any given PBR render engine translates to other PBR renders as well. This allows for the artist to concentrate on improving the quality of the textures through knowledge of real world materials and painting without the need to worry about a large amount of engine specific texturing techniques.

12.2. Problems and necessary improvements for the Cycles PBR shader

The primary problems with the shader implemented here was the lack of access to half-vector of the glossy BSDF. I was unable to find documentation or any reference material in regards to the lack of this input, which seems strange considering the most other PBR capable renderers take in to account the half-vector in fresnel calculations. Cycles is currently unable to take in to account the half-vector so a work around had to be developed. This also means that any fresnel effect in Cycles, while the fresnel node itself is accurate, is incorrect and a needless approximation.

Additionally I noticed that this implementation of a PBR shader is not strictly a metalness or specular workflow, but rather a mix between the two. Ideally in the metalness workflow the use of an extreme f-0 value would not make the dielectric material look metallic as it would be constrained by the shader. In the case of the Cycles PBR shader accidentally setting the f-0 value high makes the surface entirely reflective and it looks completely metallic. Also the way the shader components handle color input is different from most PBR engines and therefore an albedo map designed for metalness workflow is not suitable for the shader, but rather an albedo map designed for specular workflow should be used.

Some materials were also problematic for the shader as it would need node groups for translucent fabrics and glass, which are not implemented in the current version.

The next step in improving the implementation of this Cycles PBR shader would logically be to implement it in code in a custom Blender build with a more detailed approach to mixing the shader, taking in to account the various light interactions at BRDF level, thus eliminating any error that the node system would possibly produce.

REFERENCES

Blender Foundation. 2015 Blender Manual. Manual. Updated 14.3 2015. Read 22.3.2015
http://www.blender.org/manual/contents.html

Blender Foundation. 2015 Shader Nodes. Manual. Updated 14.3 2015. Read 22.3.2015
http://www.blender.org/manual/render/cycles/nodes/shaders.html

Blender Foundation. 2015 Shader Introduction. Manual. Updated 14.3 2015. Read 22.3.2015
http://www.blender.org/manual/render/cycles/nodes/introduction.html

Blender Foundation. 2015 Light Paths. Manual. Updated 14.3 2015. Read 22.3.2015
http://www.blender.org/manual/render/cycles/settings/light_paths.html

Blender Foundation. 2015 Surface. Manual. Updated 14.3 2015. Read 22.3.2015
http://www.blender.org/manual/render/cycles/materials/surface.html

Blender Foundation. 2015 Lamps Manual. Updated 14.3 2015. Read 22.3.2015
http://www.blender.org/manual/render/cycles/lamps.html

Blender Foundation. 2015 Input NodesManual. Updated 14.3 2015. Read 1.9.2015
https://www.blender.org/manual/render/cycles/nodes/input.html

Blender Foundation. 2015 Units. Developer Documentation. Updated 14.3 2015. Read 1.9.2015
http://wiki.blender.org/index.php/Dev:2.6/Source/Render/Cycles/Units

Blender Foundation. 2015 Color Management. Release Notes. Updated 14.3 2015. Read 1.9.2015
http://wiki.blender.org/index.php/Dev:Ref/Release_Notes/2.64/Color_Management

Hand, A. 2014. Unite 2014 – Best Practices for Physically Based Content Creation. Referred 24.3.2015
https://www.youtube.com/watch?v=OeEYEUCa4tI

Renaldas, Z. 2014. Unite 2014 – Mastering Physically Based Shading
https://www.youtube.com/watch?v=eoXb-f_pNag Referred 12.8.2015

Valenza, E. 2013. Blender 2.6 Cycles : Materials and Textures Cookbook. Olton, Birmingham, Packt Publishing.

Iraci, B. 2013. Blender Cycles : Lighting and Rendering Cookbook. Olton, Birmingham, Packt Publishing.

Prince, S. 2011. Digital Visual Effects in Cinema: The Seduction of Reality. Piscataway, Rutgers University Press.

Martinez, A. 2012. Siggraph 2012 Course Notes. Physical Production Shaders with OSL: Using Open Shading Language At Sony Pictures Imageworks. Culver City, CA, Sony Pictures Imageworks.
http://blog.selfshadow.com/publications/s2012-shading-course/martinez/s2012_pbs_osl_notes_v3.pdf

Sony Pictures Imageworks Inc. 2015. Language Specification. Open Shading Language 1.6. Read 25.3.2015
https://github.com/imageworks/OpenShadingLanguage/blob/master/src/doc/osl-languagespec.pdf

Burley, B. 2012. Course Notes. Physically-Based Shading at Disney. Read 25.3.2015
https://disney-animation.s3.amazonaws.com/library/s2012_pbs_disney_brdf_notes_v2.pdf

Lagarde, Sebastian. 2011. Article. Feeding a physically based lighting model. Read. 15.9.2015
https://seblagarde.wordpress.com/2011/08/17/feeding-a-physical-based-lighting-mode/

Lagarde, Sebastian. 2011. Article. Adopting a physically based shading model. Read. 15.9.2015
https://seblagarde.wordpress.com/2011/08/17/hello-world/