AUTODESK UNIVERSITY

DV15677

Procedural PBR Material Creation using Substance Designer for Visualization

Gensler

Learning Objectives

- Learn how to create some basic materials
- Learn how to get started in Substance Designer
- Learn how to export the materials created
- Learn how to apply the materials across various rendering engines

Description

Learn to procedurally create any kind of physically based rendering (PBR) material with Substance Designer. This will enable you to create physically based materials that not only look believable, but also materials that tile perfectly when rendered. In this class you'll learn how to create some basic materials, and how to translate them into the major rendering applications including: mental ray rendering machine, NVIDIA's Iray rendering engine, Chaos Group's V-Ray rendering engine, and A360 cloud-based collaboration service. We will also touch on creating material definition language (MDL) materials by using Iray directly inside Substance Designer. With Substance Designer at the center of your material pipeline, you'll be a0ble to create materials that will look relatively the same across any rendering engine you end up using for visualization. They will even translate into the major gaming engines, such as Stingray game engine, Unreal, and Unity. The industry use case for Substance Designer is unlimited. Anyone doing product design, automotive, architecture, film, and gaming can benefit from adding Substance Designer to his or her pipeline.

Your AU Expert(s)

Scott DeWoody has always had an affinity for art and technology. After seeing the animation being done through computers, he knew he could combine the two. In 2007, he graduated from The Art Institute of Houston with a bachelor's degree in media arts and animation. There, he focused on lighting and rendering techniques using 3ds Max software, V-Ray, Iray for 3ds Max, and Adobe Photoshop software. A day does not go by when he is not using one of these applications. Image quality and workflow are the top priorities in his work. He is constantly studying color theory, composition, and new ways to produce the most effective possible results. He has worked at M. Arthur Gensler Jr. & Associates (Gensler), for the past 9 years as a visualization artist and manager. He has worked for numerous clients, including NVIDIA Corporation, ExxonMobil, and so many more. Currently, he is exploring the new possibilities of architecture in the interactive space with gaming platforms, augmented reality, and virtual reality.

AUTODESK UNIVERSITY

Chapter 1: Color Management

When working with Materials, **Color** is one variable that needs to stay consistent through the entire Rendering Process. From initial conception of the material in regards to: references, material creation, material export/import, rendering, and post-production, keeping **Color** consistent, and accurate, is going to be a challenge.

Linear Workflow

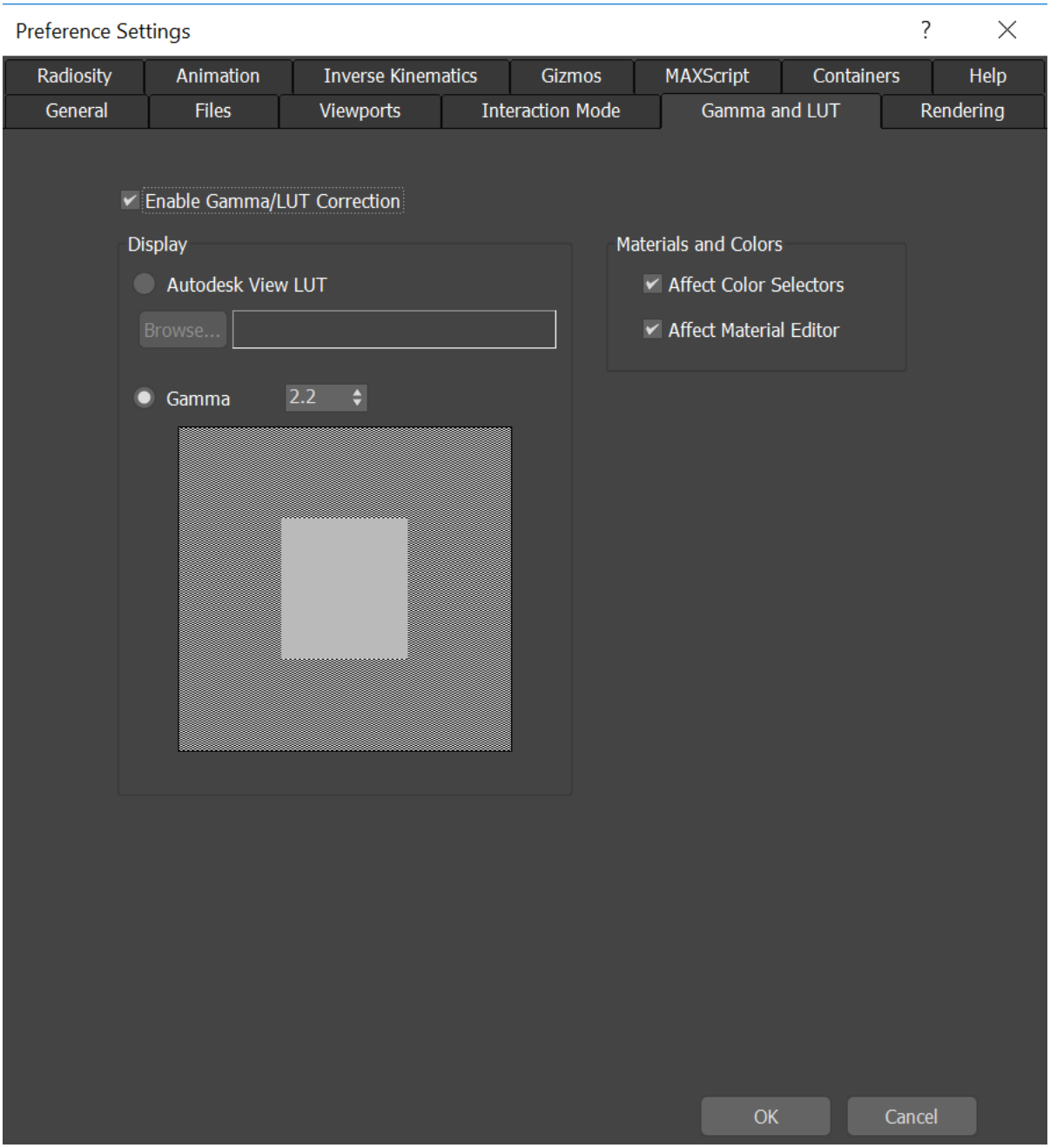
This section could have an entire course written over it, so we're just going to cover the few main key topics right here. There is a lot of information online about **Linear Workflows**, and there will be links included at the end of this document to those.

When rendering for **Computer Graphics**, we are rendering in **Linear Space**. However, our Monitors/Displays cannot show this properly. **EVERYTHING** needs to have a **2.2 Gamma Curved** applied to the Render**.** This starts to get a little more complex now when we start talking about **Bitmaps**, because **Bitmaps** already have a **2.2 Gamma Curve (sRGB)** applied when they are displayed on the screen. An **Inverse Gamma Curve (.4545)** needs to be applied to these bitmaps in order for them to be rendered in **Linear Space**, and then displayed back in **Gamma Corrected Space**. But that only applies to images that are in **Color**. Some **Bitmaps** we will be generating **need to stay in Linear Space**, as they are not being used for their **Color** but for the **Greyscale Data** values they represent. So the Rendering Engine needs to know to keep these images in **Linear Space**, and not correct back to **sRGB**. This is usually handled through the **Rendering Engine,** and/or the **DCC Host Application (DCC = Digital Content Creation)**. In the case of this course the **DCC is 3ds Max. (Maya is also considered a DCC.)**

3ds Max Setup

Setting up the **Linear Workflow** inside of **3ds Max** is extremely easy. And in **3ds Max 2017**, it should be on by default. But if not, follow these steps to enable it. Then set it and forget it!

1. Go to the top file menu inside **3ds Max** and hover over the **Rendering Menu**.
2. When the dropdown list appears, go all the way to the bottom to **Gamma/LUT Setup…**
3. A new window will appear with a few settings.
4. **Enable Gamma**
5. Set the **Gamma to 2.2**
6. Make sure that the checkboxes for: **Affect Color Swatches and Affect Bitmaps are checked.**
    a. For users using an older version of 3ds Max, make sure that **Input is set to 2.2 and Output is set to 2.2.**
    b. Set **Output to 1.0** if the saved images will be saved as **.EXRs.**
7. Click "OK" to save the **Gamma Settings.**



This is how the Gamma and LUT Settings should look like in 3ds Max.

In case the need arises, here are a few equations to help with the conversion between **sRGB, Gamma Correct sRGB, and Linear Space**. To make things easier, 3ds Max has a **Numerical Expression Evaluator** that will do all the heavy lifting. Just hit **Ctrl+N** in a **numerical input box in a Color Selector**. It will open up the **Numerical Expression Evaluator**, and the equation just needs to be typed into that. It will even put the answer to the equation into the numerical input for you!

sRGB to Gamma Corrected sRGB: (Ideal if converting RGB colors from Photoshop or Substance Designer to 3ds Max)

255*((x/255)^2.2) = Gamma Corrected sRGB

Gamma Corrected to sRGB: (3ds Max to Photoshop or Substance Designer)

((x*255)^.4545) = sRGB

sRGB to Linear

((x/255)^2.2) = Linear

Linear to sRGB:

(x^0.4545)* 255 = sRGB

Rendering Engines

Each rendering engine handles **Gamma** a little differently. For instance, both **NVIDIA Mental Ray** and **NVIDIA Iray** will **natively use the 3ds Max Settings**. But **V-Ray** has a section in the **Render Settings** called **Color Mapping**, and will also use the **3ds Max Settings**. By default, in **V-Ray 3.4**, the engine is already set up to work in **conjunction with 3ds Max's Gamma Settings**. So it is now in **Linear Space** right out of the box, and this not need to be adjusted by the user.

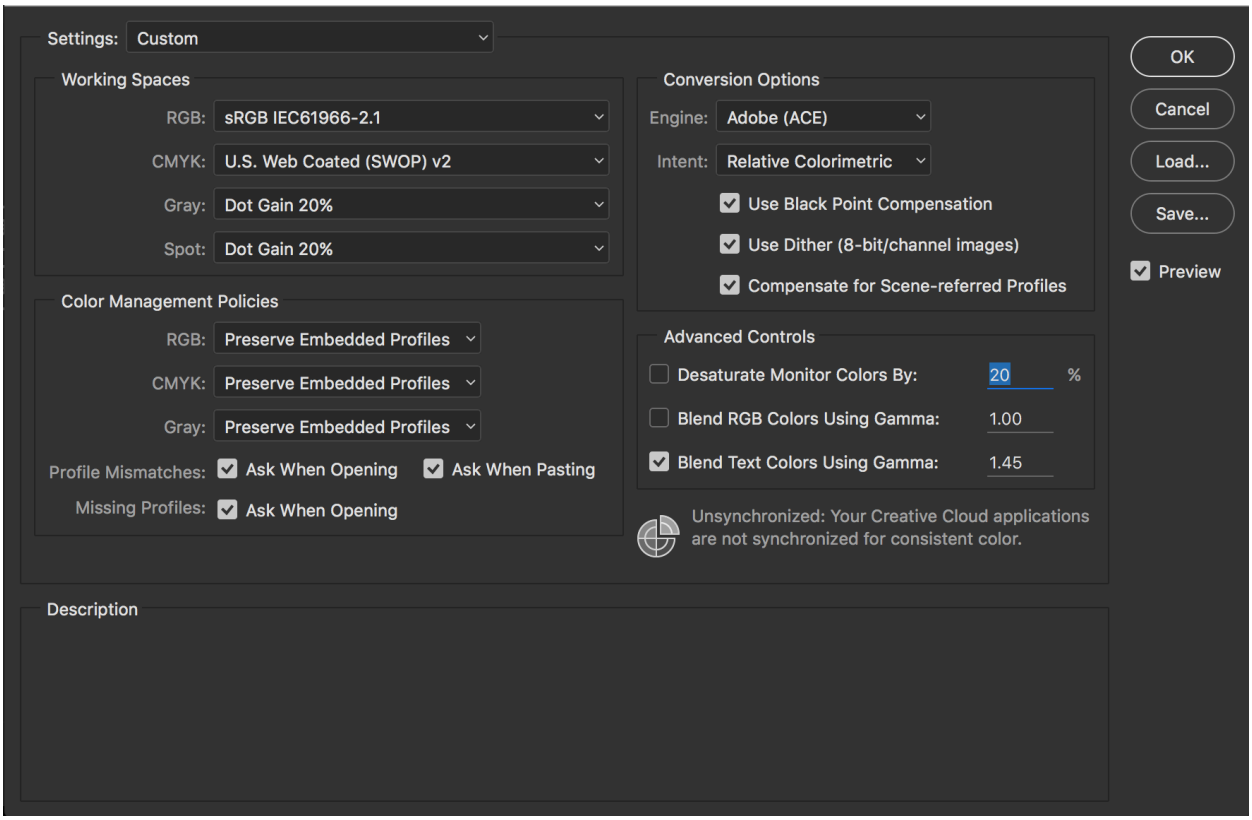
Changing the gamma values in the Render Engine is a process known as **Tone Mapping**. This is something I personally prefer to do in **Post Production** and **not the Render Engine**. So I highly recommend leaving the **Tone Mapping** in any Rendering Engine to be in **Linear Space** so the pipeline stays consistent all the way to **Post Production**.

Color Space: sRGB

When looking at working in certain colors spaces, it is very important when working in 3D to **keep everything in sRGB.** There are different **Color Spaces** out there, with **AdobeRGB** being one of them. However, a**ll 3d applications work completely in sRGB**, and **every output device (Displays, Phones, TV, Web Browsers) will only display images properly in sRGB**. So it is imperative that the entire pipeline stay within the **sRGB Color Space**. It will be easy to spot this issue as images will not display or

render properly inside of 3ds Max. Images also will not appear correct when viewed in the default photo applications, or web browsers, such as the **default photo app in Windows** or even **Google Chrome**.

When working in **Image Processing Applications**, such as **Photoshop**, make sure to have them working in **sRGB** as well. **Photoshop** for example will warn users when opening a file that it does not matching the **Working Color Space**, and will have a few options to pick from. This is good for two reasons. One, it allows users to know when they are working with something that **is not in sRGB**. And two, it will give the user the ability to easily **convert the image into sRGB**. Users will see this when opening their final rendered outputs from **3ds Max into Photoshop**. **3ds Max** will use the **OS's defined color profile**, the one that is generated with the color calibration device, but it **will not save** the actual profile information to the final output. **Photoshop** will open this image, and see that it has **no color profile assigned** to it, in which users will be able to select **"convert to working profile: sRGB"**. This will successfully convert what **3ds Max** renders into s**RGB Color Space**.



This is how the Color Settings should be set inside of Photoshop

Display Calibrations

Even with a professional display, they need to be calibrated about once every 2-3 months. This can easily be done with a $300 investment into a small piece of hardware that sits on top of the display and measures the color coming from it. It will then generate

a profile based off of its readings. This profile will be added to the OS of the system, and convert the colors over to proper values.

There are also more expensive options that will also calibrate projectors, printers, and scan color from materials, etc. Feel free to pick one of these up if that is required for your Pipeline. However, a majority of work stays digital, so the entry level devices are enough for just the Display Calibration.

Definitely make sure that all displays are calibrated in the work environment, especially displays that will be used for Design Reviews and Client Presentations. There is nothing worse than putting up an image on a poorly calibrated in front of a client, and having them complain that the physical materials in the room do not match the same color as the ones on the screen. That scenario can cause all kinds of issues, and I've been there personally many times! **Get. One. Of. These. Now.**

Macbeth Color Chart

When the need arises to take photograph references of materials, or photographs of anything really, use a **Macbeth Color Chart**. This color chart consists of **24 different color swatches**, which are all documented as known color values through the industry. By taking a photo of the chart in the lighting conditions of the photos that will need to be calibrated, it can be used to create a camera profile that will shift all the colors in the photograph to be the correct value!

Combining this with a good Professional Display that is Color Calibrated properly, photographic references will look correct across the entire pipeline. This becomes extremely important as we jump **into Substance Designer, 3ds Max, and the Rendering Engines.**

R - 115 G - 80 B - 64	R - 195 G - 151 B - 130	R - 94 G - 123 B - 156	R - 88 G - 108 B - 65	R - 130 G - 129 B - 177	R - 100 G - 190 B - 171
R - 217 G - 122 B - 37	R - 72 G - 91 B - 165	R - 194 G - 84 B - 98	R - 91 G - 59 B - 107	R - 160 G - 188 B - 60	R - 230 G - 163 B - 42
R - 46 G - 60 B - 153	R - 71 G - 150 B - 69	R - 177 G - 44 B - 56	R - 238 G - 200 B - 27	R - 187 G - 82 B - 148	R - 49 G - 135 B - 166
R - 243 G - 243 B - 243	R - 201 G - 201 B - 201	R - 161 G - 161 B - 161	R - 122 G - 122 B - 122	R - 83 G - 83 B - 83	R - 50 G - 50 B - 50

These are the known RGB Values for a MacBeth Color Chart

AUTODESK UNIVERSITY

Chapter 2: Material Basics Overview

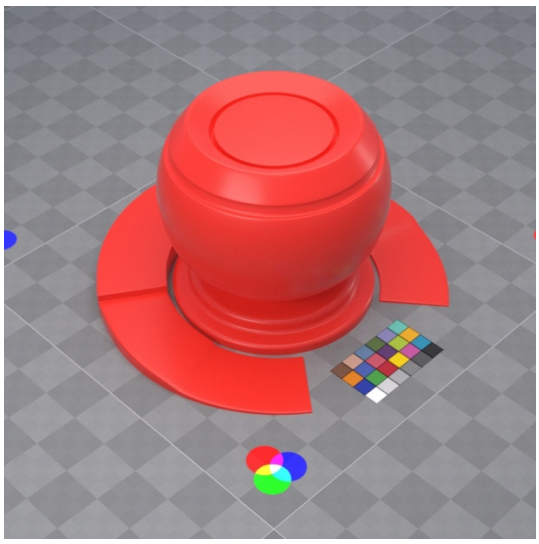
Types of Materials

There are **three material** types in the world: **Dielectrics, Metals, and Gemstones**. A majority of materials can fall into the **Dielectric or Metal** categories. And a small handful can be considered **Gemstones**. This course will focus mainly on **Dielectrics and Metals**.

Dielectrics (Non-metals)

Dielectrics are considered to be **poor conductors of energy**. When light hits the surface it is **both reflected and absorbed**. The absorbed light can refract and bounce around inside the surface until it re-emerges. Sometimes light can be completely absorbed by a material. And some light will only enter a portion of the surface but scatter greatly inside the surface. This is known as **sub surface scattering (SSS).** All of this will ensure that **Dielectric** materials will have a color in the **Diffuse/Albedo** channel.

The side effect from all of this is that **Dielectrics reflect smaller amounts of light compared to metals.** They are only typically reflecting **2-5%** of the spectrum, where **metals are reflecting 70-100%.**



Example of a Dielectric Material

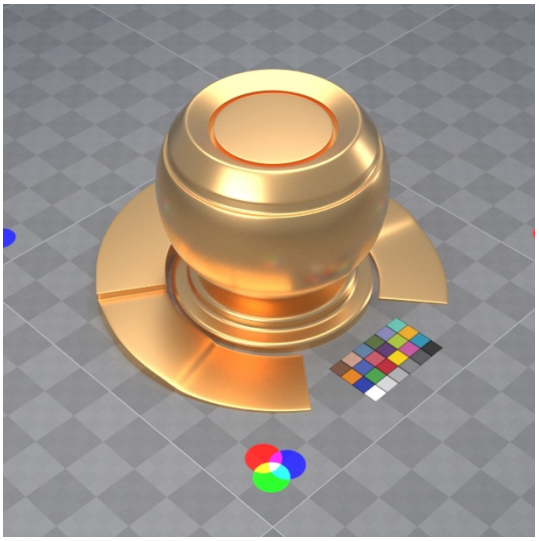
Metal (Conductors)

Since **Metals** are **high conductors of energy**, they will **absorb all of the refracted light, and only reflect a particular wavelength.** This is what allows **metal** to have reflective angles in the **70-100%** range, versus the **dielectrics** in the **2-5%** range.

Metals will **absorb light at different wavelengths**, so the **color of metal** actually comes from the **reflected light**. **This is why our diffuse color of metal is actually 100% Black, and our Reflection Channel should receive the color of the metal.**

Things to know about Metals:

- In a **Metal/Roughness workflow** the color of metal is included in the **BaseColor Map** for the material. This is due to the **Metallic Map** telling the engine what is metal, and what is not, via a black and white map.
    - The only current Raytrace engine that uses a **Metal/Roughness Workflow** is the **Autodesk Raytracer (ART).**
- When creating a material for **Metal**, the reflection should be determined if the surface is considered to be “Raw” metal. If the metal surface is painted, or rusted, the reflection properties of the top attribute should be the main source for reflectivity. For instance, if the metal is painted, you should use the reflectivity of the paint over the metal. However, if the paint is chipped, the exposed “raw” metal surfaces would have the reflectivity of the metal surface.
- **Metals will only look correct when set in an environment.** They are highly reflective and receive their visual properties from the environment around them.



Example of a Metal Material

Three Basic Material Properties

There are **three main properties** that every material has: **Diffuse/Albedo/BaseColor**, **Specular/Reflection, Glossiness/Roughness (aka Micro-facet).** All three of these properties are greatly tied together, because they all deal with how light is **absorbed, reflected, and refracted.** When creating materials, these are the **first three properties** that should always be considered. Different applications can use different names for these properties, but these are the most commonly referred to terms. However, every **Physically Based Rendering Engine** will have a spot for all three of these properties in their material Shader.

Diffuse/Albedo/BaseColor

This property describes the **over-all color** of a material. **Color** is determined by the surface absorbing lighting, and the **refracted wavelength** that escapes the surface becomes the **color we perceive**.

This is most commonly called a **Diffuse** in most engines, but in **gaming engines** this can be referred to as an **Albedo or Base Color**. The difference between the naming describes a little more about what the kind of map does. **Diffuse Maps** are usually color maps that include lighting information inside of them. With Gaming entering the world of **Physically Based Rendering (PBR),** it is **no longer required (and recommended)** to include lighting information in the **Diffuse Map**. This is because it breaks the realistic look and feel of the rendering, if light is already painted in. **PBR Rendering** takes lighting to a new level. So the **Albedo Map** is essentially the **Diffuse Map**, but without any of the lighting information baked into the map.

In pretty much **every Raytrace Engine**, the property will be labeled as **Diffuse**. And lighting information should **NEVER** be baked into the **Diffuse Map** when working with a **Raytrace Engine**.

Dielectrics
Light is absorbed by the material, and certain wavelengths escape the material. The wavelengths that escape make up the color of the material. Hence the **Diffuse/Albedo Map** containing **Color**.

Metals
Light is either completely absorbed or reflected. **The reflecting light makes up the color of Metals**, which is why in a **Diffuse/Albedo Map Metals are Black**. The **color for Metals** is authored in the **Specular/Reflection Map.** This is due to light waves being either completely absorbed (refracted) or completely reflected off the surface.

In a **BaseColor Map** however, the **Metal Color** is included due to how this map interacts with the Shader in a **Metal/Roughness Workflow**.

Diffuse/Albedo/BaseColor Best Practices

- The map should be **void of all extra lighting information**.
    - **No direct light or shadows** should appear on the map.

- It should appear if your map was lit from a bright light across the map 100% evenly
- Do not have every contrast looking maps
- Do not have “dark” looking maps, they should look evenly lit.

- All values stay between **sRGB 30-243**
  - **“White” is not 255, but actually 243**. Setting Diffuse colors to 255 will give overly bright results and not look realistic. The same goes for Black.
- **Pure Metal should be considered Pure Black** (sRGB 0,0,0) due to their reflective nature described earlier.

Left: Good Diffuse Map ***Right: Bad Diffuse Map***

Specular/Reflection

This property describes how **reflective the surface of a material can be**. Every material has **some level** of reflectivity, even if it’s not extremely noticeable by the human eye. This property is controlled via the greyscale value of **Black to White (Or a value of 0 to 1).**

Depending on the workflows, which will be covered later, **Black is usually considered to be zero reflection**. **White is usually considered to be 100% reflective.** This property also goes hand-in-hand with the **Index of Refraction (IOR)** of the material. It is also very closely tied to the next property, which is **Glossiness/Roughness.**

Most **Raytrace Engines** recommend that the **Reflection Color be set at pure White** a majority of the time. And as a result have the **IOR**, and the **Glossiness/Roughness** values, contribute to how the **reflection looks and falls off** the material. Only time to decrease the value from White would be to simulate dirt, or other substances, that would dilute the reflectivity of the material.

Fresnel Effect

"Everything has Fresnel" – Christopher Nichols

So **every single material** in the world has **Fresnel**, and every **Raytrace Engine** takes this into consideration. Most have the ability to **disable Fresnel,** and gives users the ability to create their own reflections, but that's **highly discouraged** in the making of **Physically Based Materials**.

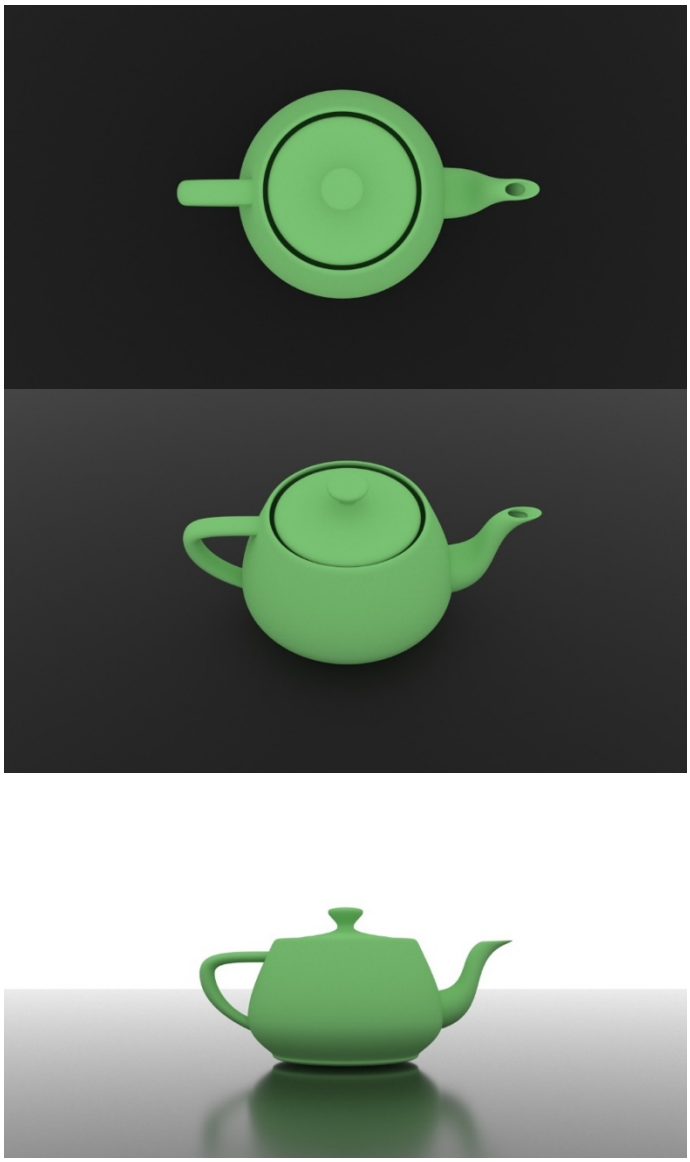
The **Fresnel Effect** states that the amount of light that we see reflected on a material depends on the angle at which it is viewed. The **Index of Refraction** will control the level of reflectivity when the angle of incidence is 0°, which is when someone looks directly straight at the surface of a material. From that point, towards a parallel viewing angle, the material becomes 100% reflective. However, how sharp that reflection is for the material depends on the **Glossiness/Roughness** of the surface.

Dielectrics

As stated previously about **Dielectrics**, they absorb a majority of the light and reflect only a small amount back. That amount that gets reflected back is roughly **2-5%** of the light that hits the surface of the material.

In **Raytrace Engines**, this is easy to accomplish as **IOR** values between **1.0 - 1.6** will achieve this look. Though a majority of materials can be set to **IOR 1.4** and appear accurate. The difference in reflectivity between these values can be minimal, so a value of **1.4 is a good baseline for all Dielectrics**. There are plenty of documented values online for **Dielectrics**, but the values to know are the following:

- **Water:** 1.33
- **Plastics:** 1.46
- **Glass:** 1.5-1.6
- **Glass with reflective coating:** 2.0.-2.5



Top: *Almost no reflection* ***Middle:*** *Very little Reflection* ***Bottom:*** *Almost 100% Reflective*

Metals

Calculating the **Index of Refraction for Metals** is a bit more difficult than it is for a D**ielectrics**. Since **Metals** either completely absorb light, or completely reflect it, they behave differently. Their reflectance range is anywhere from **70-100%**! This is again why Metals are represented in the **Diffuse Map as Black**, and their color is derived from the **Reflection Color**. The **Fresnel Effect** in this case is **much more complex** than a **Dielectric**. **Metals** take an additional property into consideration known as the **Extinction Coefficient (k).**

Most **Raytrace Engines do not take the Extinction Coefficient (k)** into account. In **3ds Max** the ability to **mimic this effect** can be done via a **FallOff Map**, but it is **tedious** to set up. Most **work arounds** have users setting the **color of the Metal in the Reflection Channel** and then cranking the **IOR up to 25-50**. Although this is **not "technically" accurate**, it reproduces results that **look accurate**.

NOTE: If using **V-Ray**, Vlado at **Chaos Group** has released an **Open Shader Language** file for this very thing. It can be downloaded for free online at the Chaos Group Documentation.

NOTE: Redshift does take the **Extinction Coefficient (k)** value into consideration if the **Fresnel** mode is changed to **Advanced** in the Shader.

Metal Colors
Below is a list of known sRGB Values for Metals.

*These colors are actual **presets** inside of **Substance Designer** when using **the BaseMaterial Node.**

Material	F0 (Linear)	F0 (sRGB)	Color
Gold	R = 1.00 G= .766 B= .334	R = 255 G = 226 B = 155	
Silver	R = .974 G = .957 B = .915	R = 252 G = 250 B = 245	
Aluminum	R =.915 G=.923 B=.923	R = 245 G = 246 B = 246	
Iron	R =.560 G=.579 B=.579	R = 196 G = 199 B = 199	
Copper	R =.957 G=.638 B=.535	R = 250 G = 208 B = 192	
Titanium	R =.541 G=.499 B=.447	R = 193 G = 186 B = 177	
Nickel	R =.659 G=.605 B=.523	R = 211 G = 203 B = 190	
Cobalt	R =.659 G=.652 B=.632	R = 211 G = 210 B = 207	
Platinum	R =.673 G=.638 B=.585	R = 213 G = 208 B = 200	

AUTODESK UNIVERSITY

F0

When viewing a material directly the **angle of incidence it is 0°,** which is known as **F0**. This is one way to figure out a materials **IOR**, but it is also a key value for **the Spec/Gloss Workflow**.

Dielectrics

Since there is only a **2-5%** reflectance with **Dielectrics**, this will result in a linear range of **.02-.08** and a **greyscale sRGB range of 43-80.**

Metals

Since the reflectance values of **Metals** are anywhere between **50-100%,** the linear values for metals are in the **.5-1.0** range and **sRGB range of 187-255.**

Color does come into play with **Metals,** but the **Value** set within the **RGB Scale** should be in this **greyscale range**.

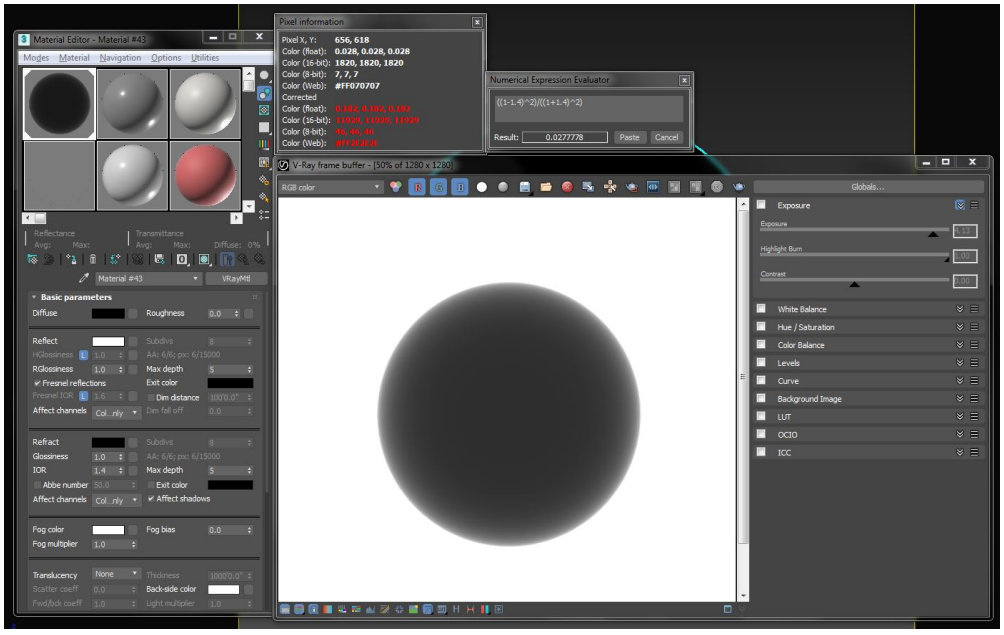
To convert IOR values into F0 values, use the following equations:

Linear Equation:

F0=((1-n)^2)/((1+n)^2)

Gamma Corrected Equation (sRGB):

F0=((((1-n)^2/(1+n)^2)^(1/2.2))*255



This example samples the middle of the sphere, and shows what an IOR of 1.4 looks like. Note the RGB Value is 42, which falls in the acceptable range, plus the float value is .02 which equals the F0 Value of 1.4. This is also shown in the IOR to F0 equation shown in the image as well!

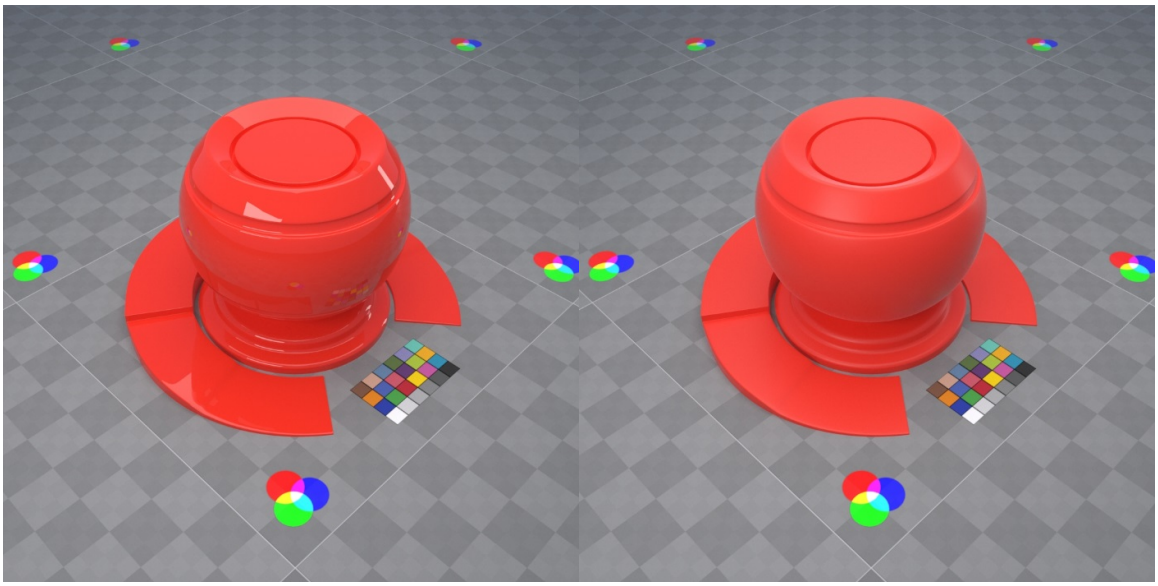
Glossiness/Roughness (Micro-facet)

This property describes the **micro level of surface irregularities** that make up the actual surface of a material. This is **not to be confused with Bump or Normal** detail, but detail that is so small that humans don't really perceive it with their eyes.

This channel is what gives the specular highlight either it's **sharp, or wide-spread, characteristics.** This is due to how light reflects off the micro uneven surface of the material.

In a perfect world a surface would be completely smooth, thus creating a perfect reflection. Very few materials have this, but most materials have microscopic grooves and niches that will angle light in a different reflecting direction. This creates a **"blurred"** or **"diffused"** reflection or specular highlight.

NOTE: If a Shader had a both a **Glossiness** and a **Roughness** Channel, this section is referring to the **Glossiness Channel** of the Shader. The **Roughness Channel** in this case does something different in the Shader.



Left: *Glossiness at 1.0* ***Right:*** *Glossiness at 0.6*

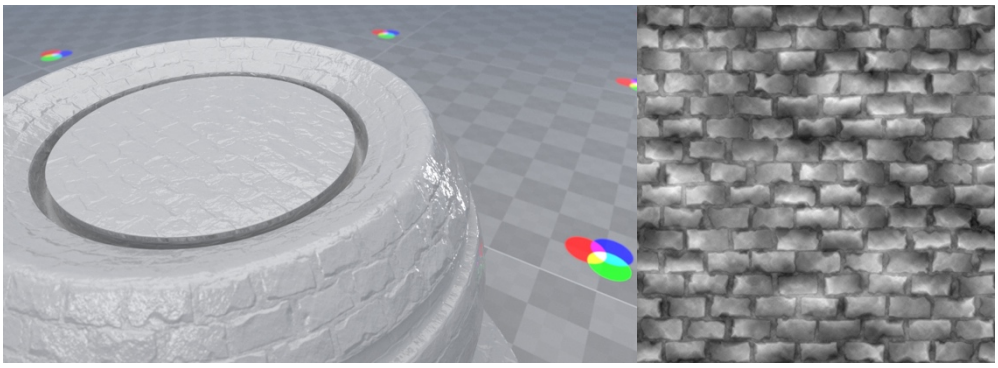
Other Material Properties

Some materials have **more than the three basic properties**, and these need to be taken into account for during the creation process. Some materials may have one or more of the following properties.

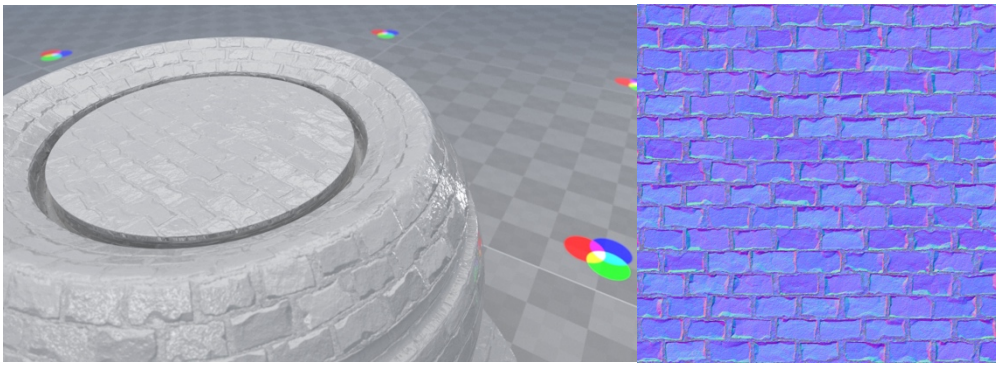
Bump/Normal

To **simulate extra detail** on a surface of a model, a **Bump or Normal Map** can be used. This is done via **a Black and White map (Bump)**, or a **colored map that specifies which direction the surface is pointing (Normal)**. This effect happens at render time, and alters the surface normals depending on how light hits the surface. **No actual geometry is effected**, so at certain angles, this can look **fake.**

*If possible **always use a Normal Map**, because these provide more detail than **Bump Maps**.



Example of a Bump Map



Example of a Normal Map

Refraction

Some materials will end up needing to be **transparent**, and this is where **Refraction** comes into play. The **IOR** comes into play here to help how light **transfers through the material**. **Refraction** is controlled through **Black and White** values. Like most channels. **Black results in a completely opaque material,** and **White will result in a purely Refractive material**. This property should **rarely be set to pure white**. Very few, if any, are ever purely refractive. The **IOR** should mimic the same as the Reflection IOR. Most engines by default have these two properties linked, but some do offer the ability to break the link between the two.

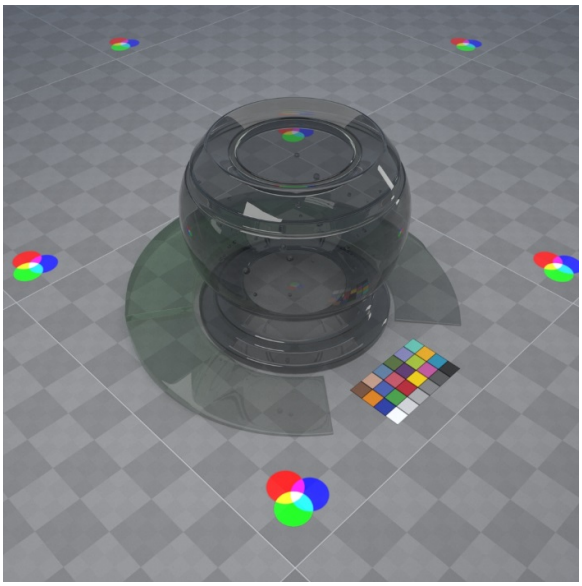
Glass

This is usually one of the few things that will be commonly made with the **Refraction Property**. **Visible Light Transmittance (VLT)** can be correlated with the Refraction property. Keep in mind that **50% VLT is not 128, but 187 (in Gamma Corrected Space)**. But also note that some glass pieces in the real world are **doubled paned glass**, and that should be taken into account for a more realistic rendering.

Glass is usually 99% of the time Black in the diffuse, however dark color values can be added to help simulate some color. Its recommended to use a property called **Tint, or Fog Color**, if the **Raytrace Engine** supports this. This property will give glass more of a color than its diffuse property, as glass gets its color from metals that get added into the creation process.

Some **highly reflective glass** will have **additional coats** on top of the actual glass, especially exterior curtain wall glass. Engines can simulate this in a few ways. The ideal way would to be able to **add a Coat Layer** to the material, if the engine supports this feature. Other ways are to **break the IOR from the Refraction and Reflection Channel**. Glass is usually around 1**.6 for the IOR**, and highly reflective glass can be **2.0-2.6** for the reflection IOR. Some engines however do not let users split the two values apart.

Refraction is something that is still a **little expensive to Gaming Engines**. **Unity and Unreal** do not support this feature out of the box, but their Shaders can be edited to support it. The same goes for **Substance Designer**. However, **Substance Designer** does fully support **Refraction** when the renderer is switched to **Iray and users work with MDLs.**



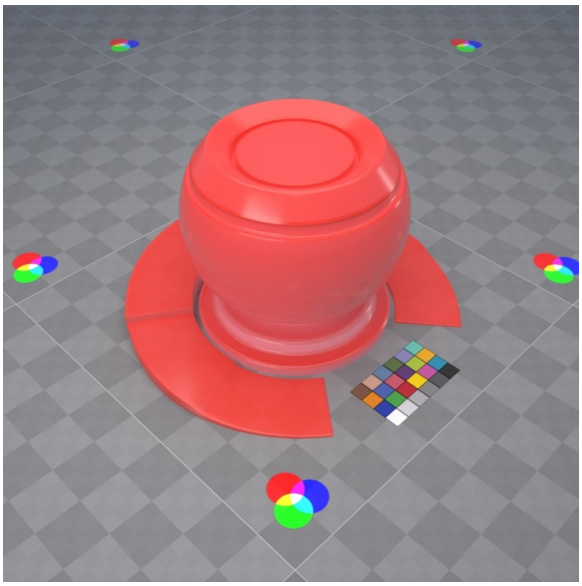
Example of Refraction with a slight green "Tint" or "Fog Color"

Translucency

This property was referenced a little earlier in the document under **Dielectrics**. **Translucency** is the result of light entering a material, and then **bouncing around inside the material until it decides to come back out.** Sometimes light may **never escapes** the material. The effect is known as **Sub Surface Scattering**, and is most commonly found when creating **Skin**. However, there are some other material types such as: **Fruit, Milk, Marble, Leaves, Grass, etc**.

Raytrace Engines pair **Translucency** with the **Refraction** property, as the **Refraction Property** is what allows light to pass through the surface of an object. **Low (Black) Values** are enough to start to simulate this effect. The **Translucency** will then control the scattering of the light inside of the surface of the material.

Some engines even have Shaders that are completely built around this effect, and are ideal to use in situations when **Translucency** is really required. This is usually when the artist is trying to make **Skin**, but it is not limited to that.



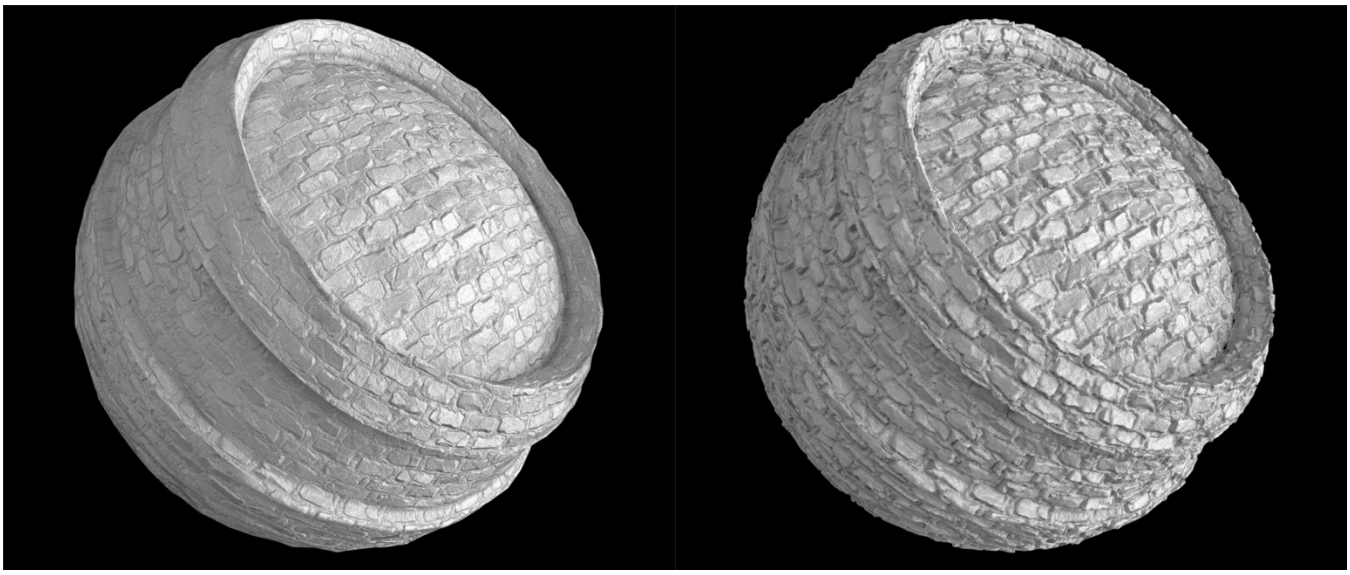
Example of Translucency added, with a light in the center of the model. Note the soft glow inside of the mesh itself, especially around the base.

Displacement (Height)

Unlike the **Bump and Normal Maps**, **the Displacement Map will actually add deformation and detail to the geometry of a model**. Displacement is controlled via a **Black and White map**. **Black** tells the geometry to **move down**, and **White** tells geometry to **move up**.

Displacement can be extremely costly at times, but it will yield an extremely better result over the **Normal and Bump Map**. It can also help keep models simple with adding the extra detail. Lighting and GI will be calculated properly as the **Displacement** is calculated just prior to the rendering process starting.

In **Gaming Engines**, **Displacement** is too costly for the most part. It's used extremely sparingly. Another workaround for this is with a Shader that supports **Parallax**. This Shader will give a better illusions of deformed geometry than a **Normal Map** will, but still cost less than using **Displacement**. In the **Gaming Engine**, this map is usually referred to as a **Height Map.**



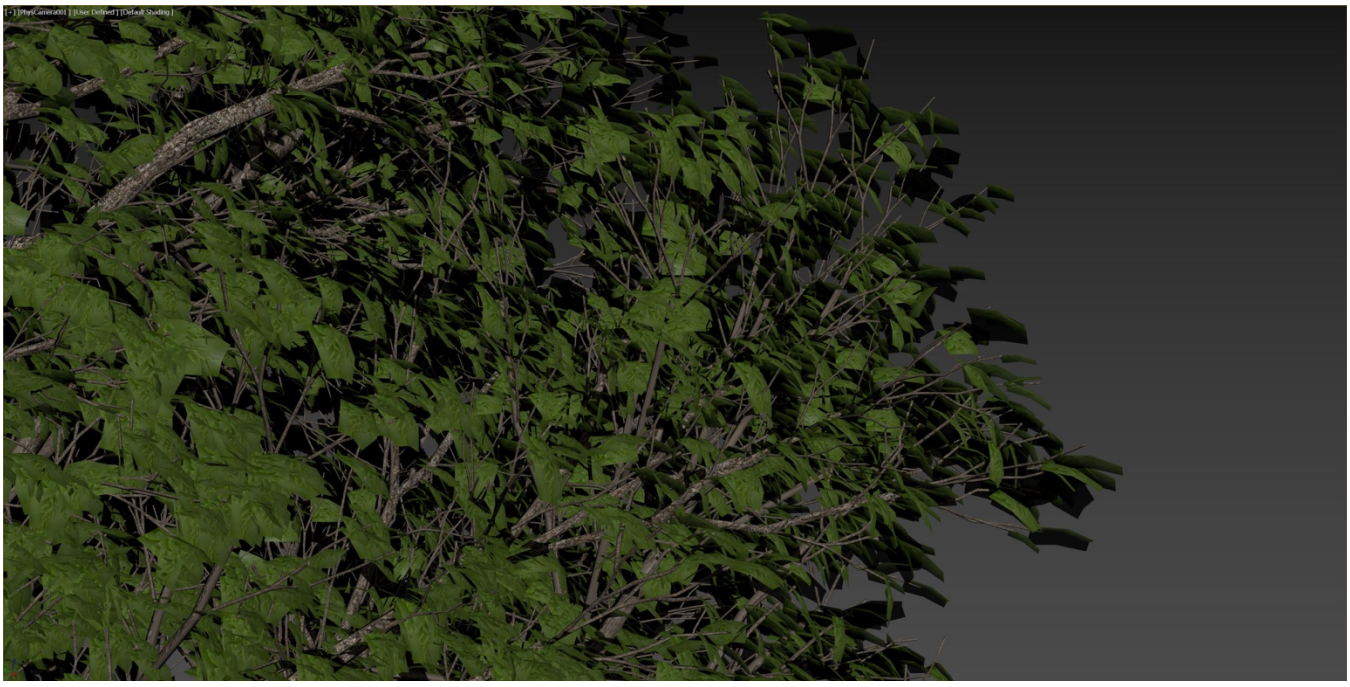
Left: No Displacement **Right:** Displacement

Opacity/Cut Out/Transparency

This property **is not to be confused with Refraction**. **Refraction** controls how light transmits through a material, while **Opacity cuts away geometry at render time.** This is the same concept as putting a **Mask on a layer in Photoshop**, and then painting away the areas that need to be transparent.

Opacity Maps are perfect **for faking geometry** that might have complex shapes and might take a while to model. It also helps keep poly-count low for Gaming Engines. **Opacity Maps** are most commonly used for **Leaves, Grass, Fences, and Decals.**

While it is always best to model this level of detail, **Opacity Maps** can do a really good job of faking the detail. Just don't get the rendering too close, or the camera parallel to an object using an **Opacity Map**. The illusion breaks pretty easily in these cases, and only real geometry will be able to fix it.



3ds Max Viewport prior to rendering Opacity Maps

Rendering with Opacity Maps

Chapter 3: Material Workflows

Each **Raytrace/Gaming Engine** has their own way of dealing with Materials from a User-Interface point of view, but the workflows behind the UI are pretty consistence. Some terminology will vary from engine to engine, but at the end of the day the process still encompasses everything covered earlier in the document

There are **three main Material Workflows** that engines adopt:

- **Reflection/Glossiness**
  - NVIDIA Mental Ray
  - NVIDIA Iray
  - Iray for 3ds Max
  - V-Ray
  - Corona
  - Redshift
  - F-Storm
- **Metal/Roughness**
  - Unity
  - Unreal
  - Stingray
  - Marmoset Toolbag
  - ART (Autodesk Raytracer)
  - RedShift
- **Spec/Gloss**
  - Unity
  - Marmoset Toolbag

Reflection/Glossiness Workflow

This is the **most common workflow** inside a majority of **Raytrace Engines**, and it has been around for the longest time. This model follows the Material Basics that were outline earlier in the document **extremely closely**.

- **Diffuse Map** should contain no lighting information
  - **Metals are Black in the Diffuse**
- **Reflection** should be set to **100% (White)**
  - **Metal color goes in this channel**
- **Fresnel** is turned **On**
- Set the **IOR**
  - **IOR 1.4 is a safe number for Dielectrics if the IOR unknown for the material.**
  - **IOR of 20-50 is good when creating materials, although not accurate.**
- **Glossiness** controls how light reflects off the surface
  - Use a **FallOff Map** in this Channel to help control the spread based on the viewing angle for additional realism. **(Not supported by all engines)**

Metal/Roughness

****We will be using this workflow inside of Substance Designer.**

This workflow is the **new kid on the block**, and has recently been introduced through **Gaming Engines** adopting **PBR** into their workflow. The basics of Materials are still here, but a few things get **flip-flopped when it comes to Dielectrics and Metals.**

- Unlike the other two workflows in this list, the **Metal/Roughness Workflow** controls the **IOR and Reflectivity** through a **Metallic Channel**. The **Metallic Channel** tells the engine if the Shader is either a **Metal or a Dielectric**. So the value of **0.0 (Black)** defines the material as a **Dielectric**, and a value of **1.0 (White)** defines the material as a **Metal.** This essentially sets either the **IOR to 1.4 for Dielectrics**, or something around **IOR 25+ for Metals**. There is the ability to move the value between that range, but there is not much need to do so. Materials are really one or the other, so setting it to **0 or 1** is ideal.
- The **Roughness Map** will control how light is **reflected off of the surface**. **Black Values (0.0)** will result in a **clean reflection**, while **White Values (1.0)** values will result in a much more **diffuse reflection.**
    - **This is the exact opposite of how Reflection/Glossiness works.**
- **BaseColor** should not contain any lighting information.
    - **Metal Colors do not go in the Metal Channel, but instead go into the Base Color Channel.**
- **Fresnel** is already enabled, and handled by the Shader. **It cannot be disabled.**

Specular/Glossiness

This **workflow mimics the Reflection/Glossiness Workflow** the most, and is the evolution from previous **Gaming Engine** workflows into **PBR**. This workflow combines the **IOR Values into the Specular Values**. So it does assume that the material is **always reflective**, but it's the value inputted into **the Specular Channel** that determines what kind of material the Shader becomes. This is equivalent to the **F0** value talked about earlier.

- **Diffuse Map** should not contain any lighting information.
  - **Metal is Black in the Diffuse Map.**
- Use the **F0 Values** for materials in the **Specular Channel**
  - If the need arises to accurately calculate the F0 value for a material, based off of its IOR, use the following equations: (n=IOR)
    - Linear Equation: **F0=((1-n)^2)/((1+n)^2)**
    - sRGB: **F0=((((1-n)^2/(1+n)^2)^(1/2.2))*255**
  - Here are a few guidelines when creating **F0 values**:
    - If all else fails, use a value of **.04 (linear) or 59 (sRGB).** Most Dielectrics tend to be around this value, so it is a safe bet.
    - **No material should go below .02 (linear) or 43 (sRGB).**
    - **Dielectrics tend to be in the range of .02-.08 (linear) or 43-80 (sRGB).**
    - **Metals tend to be in the range of .5-1.0 (linear) or 186-255(sRGB).**
      - **Color does come into play with Metals, but the Value set within the RGB Scale should be in this greyscale range.**
- The **Gloss Map** will control how light is reflected off of the surface. **High values** will result in a **clean reflection**, while **lower values** will result in a much more **diffuse reflection.**

Chapter 4: Substance Designer

All of the engines listed will have specific feature sets that are unique to them, but they all understand the Physically Based properties of Materials. So in theory, you will be able to create any kind of physically based material in any of these applications. Which is where **Substance Designer** comes into play.

The power of **Substance Designer** is its flexibility to generate any kind of physically based, realistic-looking, material in the world, and let the user export it to any **Render Engine** of their choice. It can become the center of your material creation workflow in the creative pipeline, giving designers a tool that will allow them to create whatever they need from scratch!

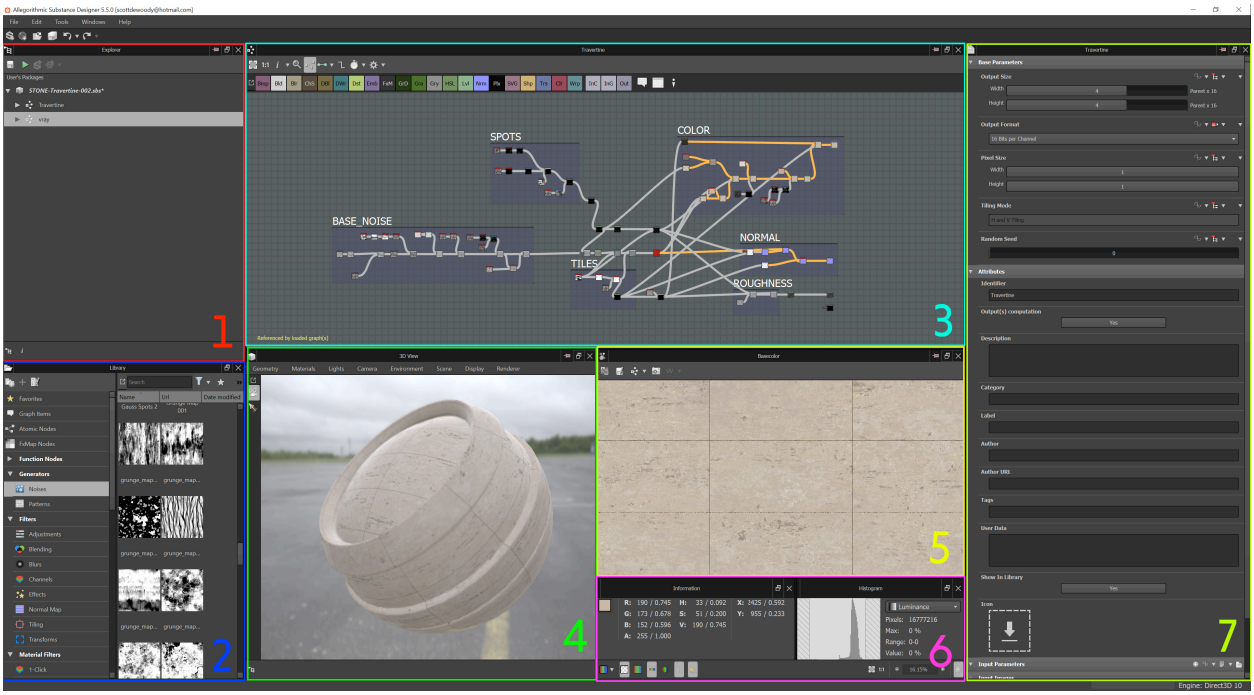
Substance Designer's unique node-based workflow creates endless possibilities and variations for materials that are generated inside of it. The learning curve for all of this is indeed extremely high, and may take some time wrapping your head around all of its features. And there are A LOT of features in this application. But when it is broken down and learned, the pay-off is substantial.

Putting **Substance Designer** at the center of the material pipeline will help create a Material Library that can be translated where ever users need it to go. It has the ability to plug into any of the key **Gaming Engines** directly, and can export texture maps to any **Raytrace Engine** users might be working with. It can even create **MDLs** now, which we'll cover a little more in-depth later.

If by this point in the course you have not tried out Substance Designer, I highly recommend going and downloading the trail from www.allegorithmic.com.

AUTODESK UNIVERSITY

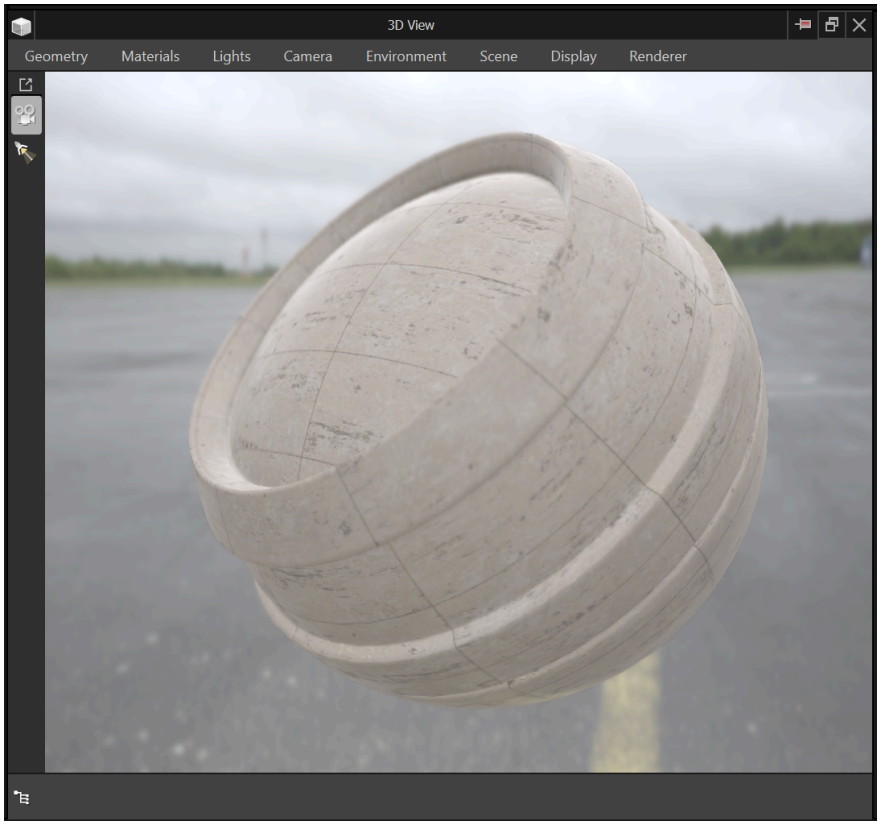
Overview of the User Interface



1. **Explorer**
   The Explorer will contain all open Packages that are currently being worked with.
2. **Library**
   This is where every node, function, hdr, tool, and asset can be found. Custom Libraries can be added to increase organization of custom assets that do not ship with Substance Designer.
3. **Graph**
   Where all the main work is done inside of Substance Designer.
4. **3D View**
   Different 3D models can be loaded here for preview. Custom FBXs can be dragged and dropped into the View Port to load specific models for texturing.
5. **2D View**
   This view will display the current Map associated with the node that is currently being worked with.
6. **Information and Histogram**
   These two windows will display current information being presented in the 2D View. This extremely helpful when trying to look at the color ranges and values of the maps being generated.
7. **Parameters**
   All the settings to control all of the Nodes and Graphs can be found here. This window will change depending on what is currently selected.

AUTODESK UNIVERSITY

3D View Setup

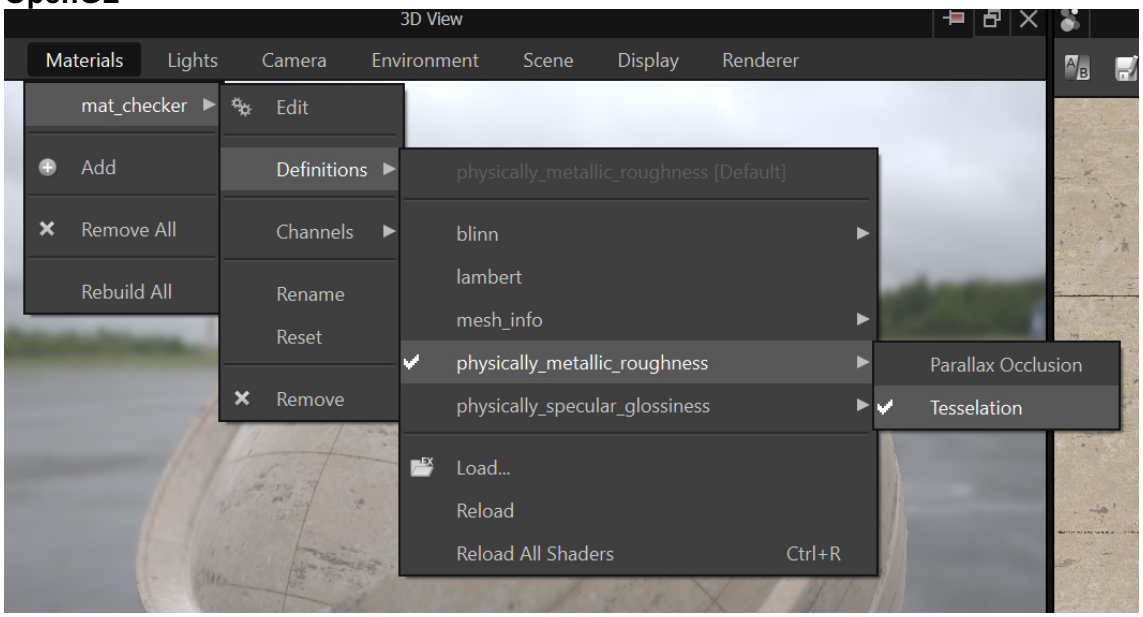


1. **Geometry**
   Substance Designer has a few options to pick from when it comes to Geometry. It's best to find the piece of geometry that works well for the Substance being created. My favorite two are the Plane (High-Resolution), and Cube Rounded Corners. I have a third favorite, which is a custom Sphere that can be downloaded from the Substance Share website.

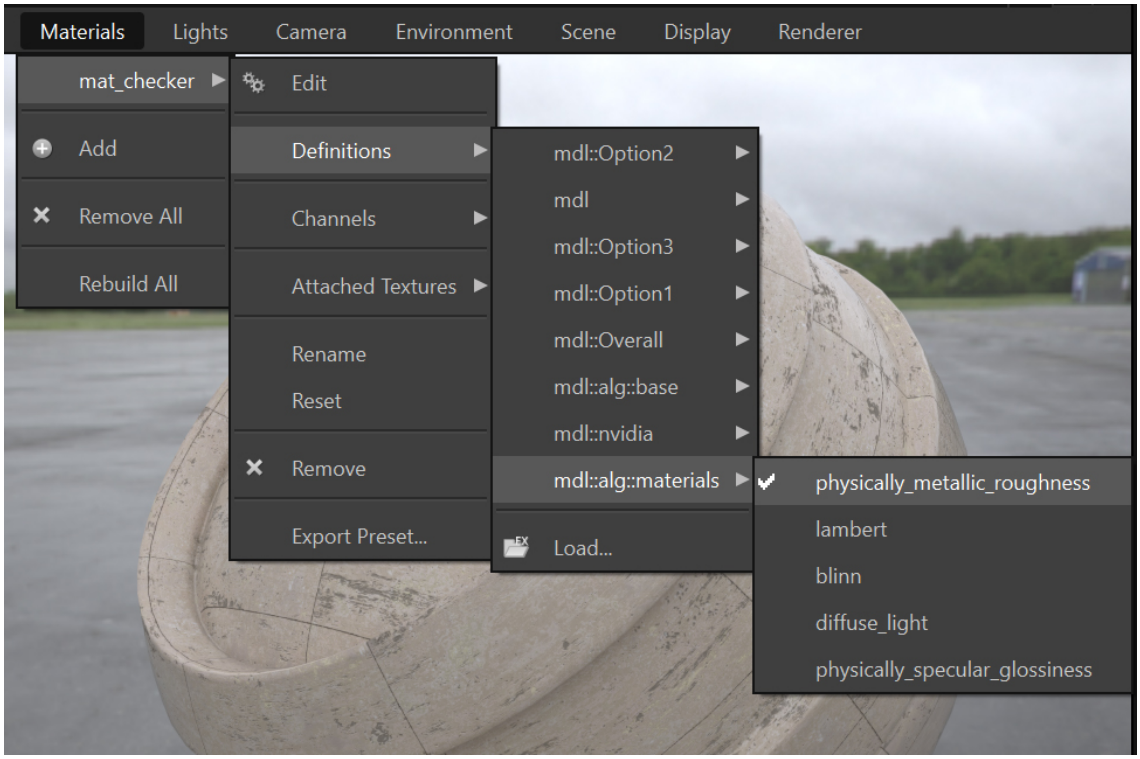
   There is also the ability to load any FBX file into Substance Designer. This way if a Substance is being designed for a specific object, it can be loaded into the viewport and worked on in real time.
2. **Materials**
   This window is extremely important as it allows the user to pick which Shader will be used for the viewport. This is extremely important to match the Shader with the Workflow/Output Nodes that will be constructed for the Substance.

   For this course, we're working with the **Metallic/Roughness Workflow**. We'll need to make sure that we're using the **physically_metallic_roughness** Shader set to **Tessellation.** This can be set by clicking on the **Materials Menu** and following the path from **mat_checker > Definitions > physically_metallic_roughness > Tessellation.**

OpenGL



When using **Iray**, we'll need to have the **physically_metallic_roughness** Shader selected. This can be selected by going to **Materials > mat_checker > Definitions > mdl::agl::materials > physically_metallic_roughness**

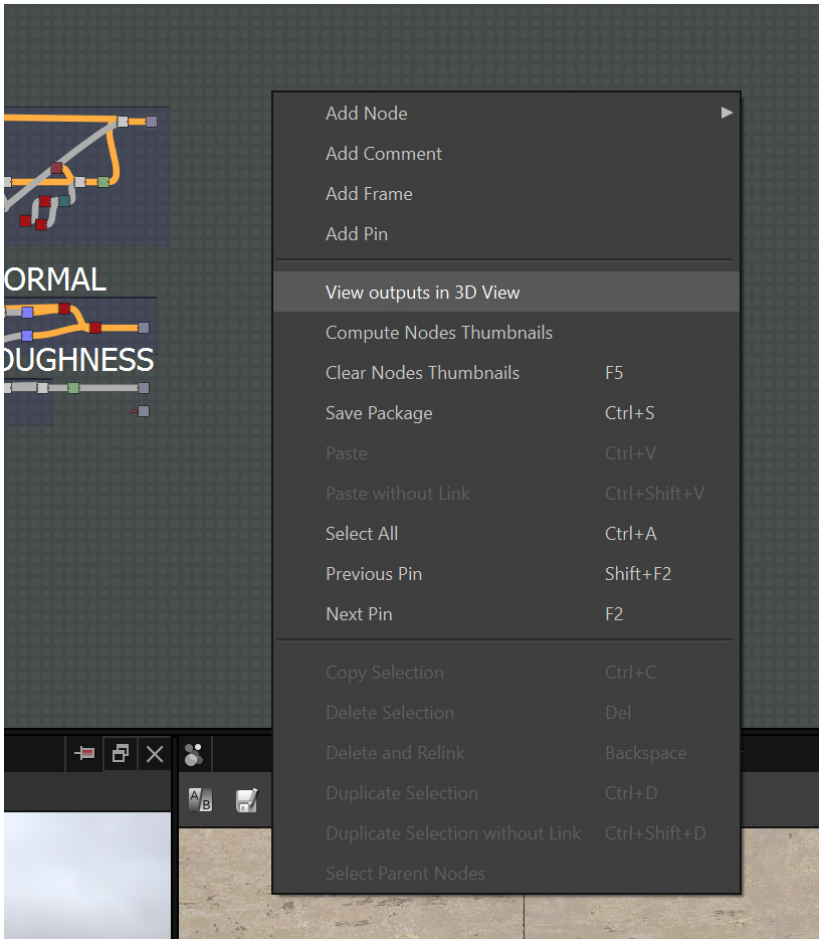


Other **MDLs** are there for use with Iray, but this course focuses on using the MDL **physically_metallic_roughness**.

****When importing MDLs into NVIDIA Mental Ray and NVIDIA Iray, this will need to be switched to Blinn.**

Materials also have some specific Properties that can be found, such as **Tessellation and Scale for Displacement**, and also a **Tiling Property** that will tile the Substance across the selected Geometry. This can be found by selecting the Shader needed for the Substance First, and then click on **Materials > mat_checker > Edit**. The Shader's Properties will show up in the **Properties Window.**

*****Tessellation is not available on OSX. Use the Parallax Occlusion instead.**
******If maps do not appear on the Geometry after selecting a Shader, right click in the Graph View and select "View Outputs in 3D View". Users also have the ability to right click on any Node and drag it into the 3D View.**



3. **Lights**
With only the Edit option in the menu, Lights will have series of Properties to edit

for the Viewport. Additional Point Lights can be added and moved around the object to help check the specular highlights and reflectivity of a Substance.

*I usually leave the lights off and just make my judgements off of the HDRIs in the Environment section. **This is personal preference.**

4. **Camera**
   Substance Designer gives a lot of options to edit for post processing the render in the 3D View. Through the Edit settings under Camera things like **Depth of Field, Glare, Lens Flares, Exposure, Tonemapping,** and so much more can be found. This way you can use just Substance Designer for the final rendering. The resolution can even be set in the **Camera Properties** in order to create an image at any resolution that is required. There is even a feature to publish the 3D View rendering straight to ArtStation!

   Custom Cameras can be imported through FBX as well! This is extremely helpful if moving a specific view or composition is required. Just remember to use the **3ds Max Standard Camera**, and **NOT** the new **Physical Camera** that can be found in **3ds Max 2016 and 2017.**

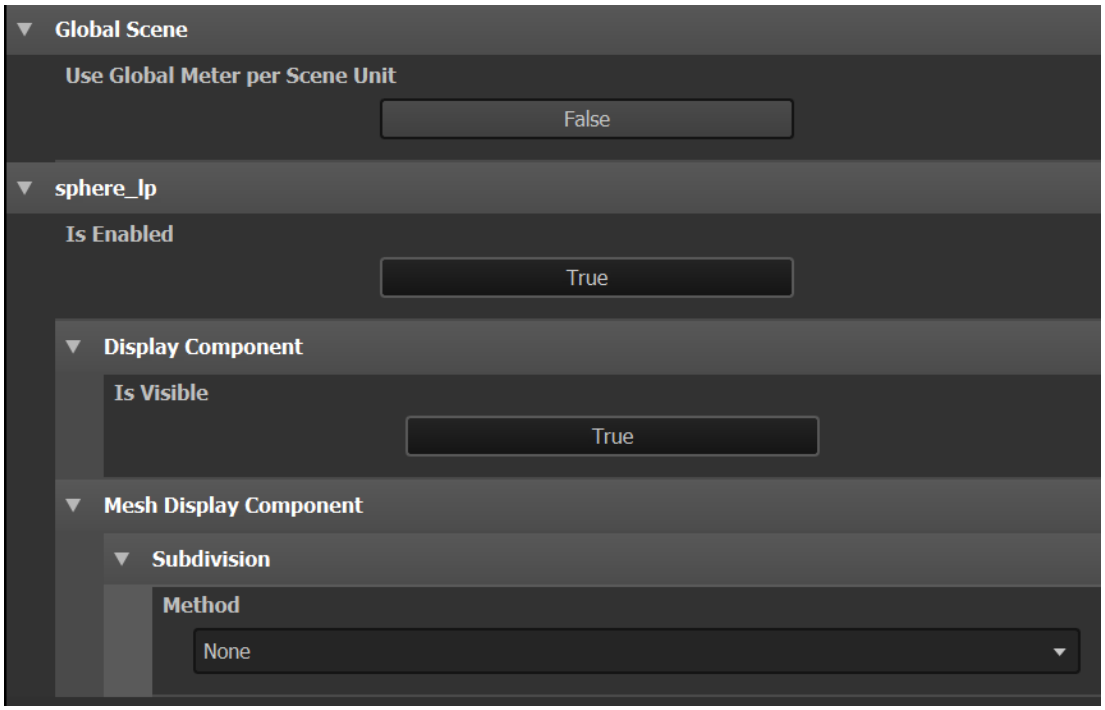
5. **Environment**
   This tab controls the **HDRI** that is loaded into the background. It can be disabled, which I personally like to do when working in Substance Designer. The Exposure can be increased, or decreased, and it can be rotated in any direction needed.

   When using **Iray**, there are a few additional options, such as being able to add a fake **"Ground Plane"** to the rendering. This is a good fake for some renderings, and can help ground an object into the Environment.

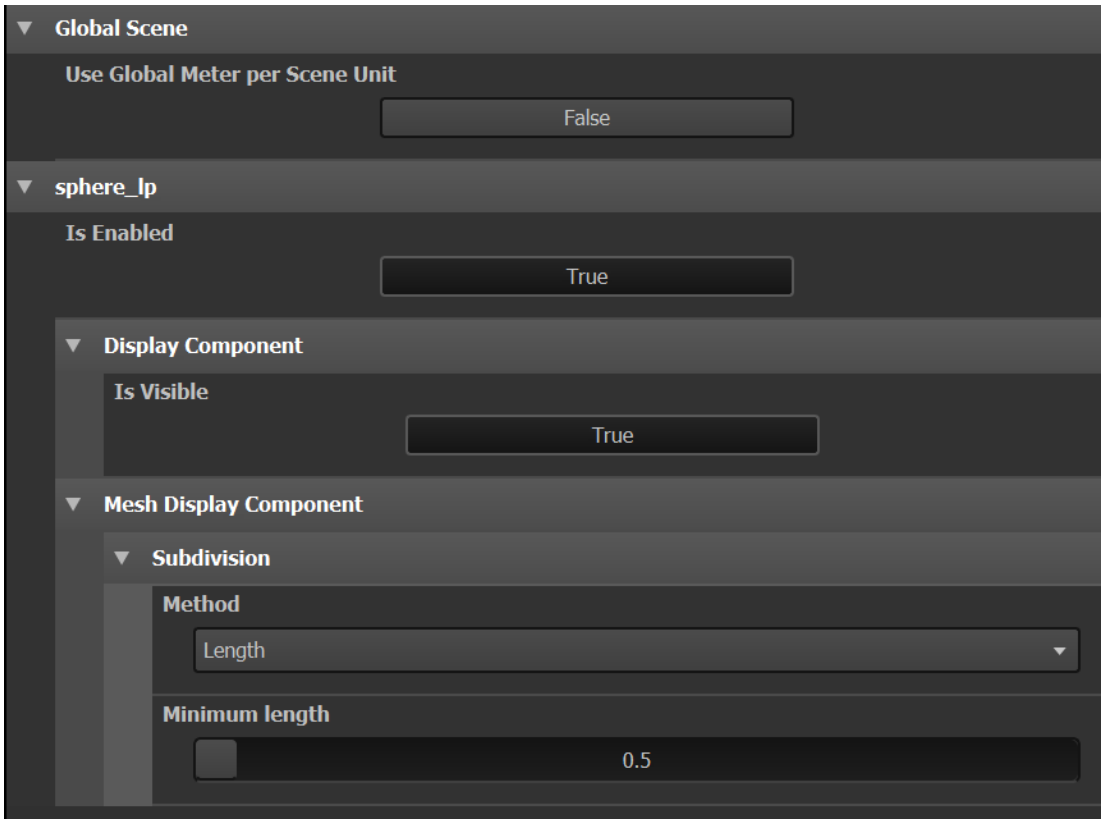
6. **Scene**
   There are a few basic settings to take note of here:

OpenGL
The only open here is to disable the geometry in the view

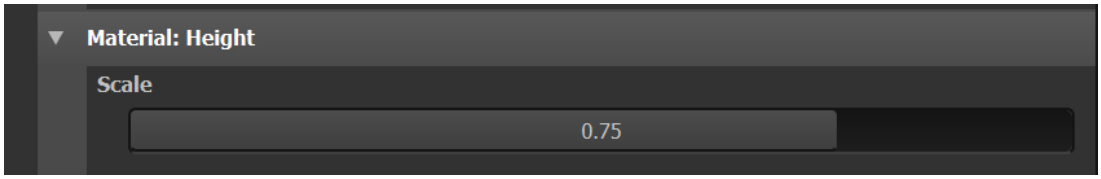
Iray
The additional properties in this window control Displacement in Iray. If a **Height Map** is in the Graph, Iray will be able to render the geometry with **Displacement** based off of this Map.

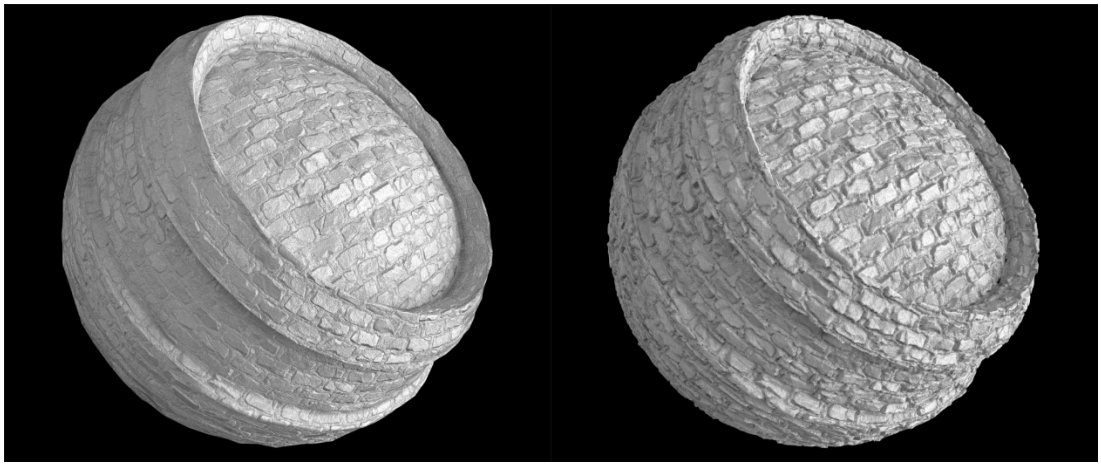


To **enable Displacement** in the **Scene Properties**, switch the **Subdivision Method from None to Length**. The **Minimum Length** will control the quality of the **Displacement**. Lower values will give a more accurate Displacement, but will take more computing power. Set this value from **.5-1.0 for really good results,** there might be some performance lag at this point.



At this point in the **Materials > mat_checker > Edit** there is a **Scale** property in the **Properties Window**. This will control the **intensity of the Displacement**.





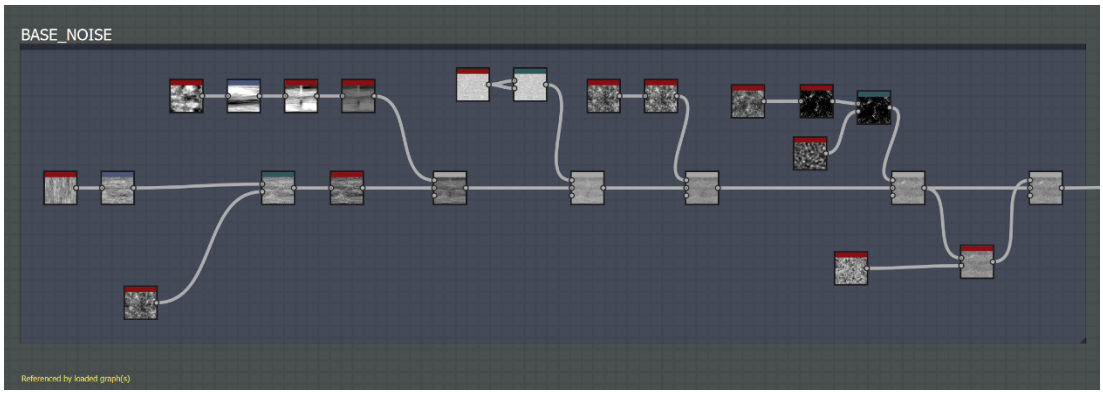
7. **Display**
   There are some simple options here to display additional items in the viewport including a **Grid** and the **Wireframe** of the mesh.
8. **Renderer**
   This section controls the toggle between **OpenGL and Iray**. Depending on which one is selected here, is what will render the 3D View. Each engine has some additional options under Edit. The defaults should be good here.

Atomic Nodes
Everything in **Substance Designer** is created with these specific nodes. They can be accessed a few ways: The Library, the top of the Graph View, and by hitting the **Space Bar in the Graph View**. These nodes are extremely accessible at all times because they will be the most commonly used nodes when working in **Substance Designer.**

I'm not going to cover every single node in this document, as I feel that the Official Documentation over at Allegorithmic's website does a really good job at that. But I will list the Nodes users will most likely use the most.

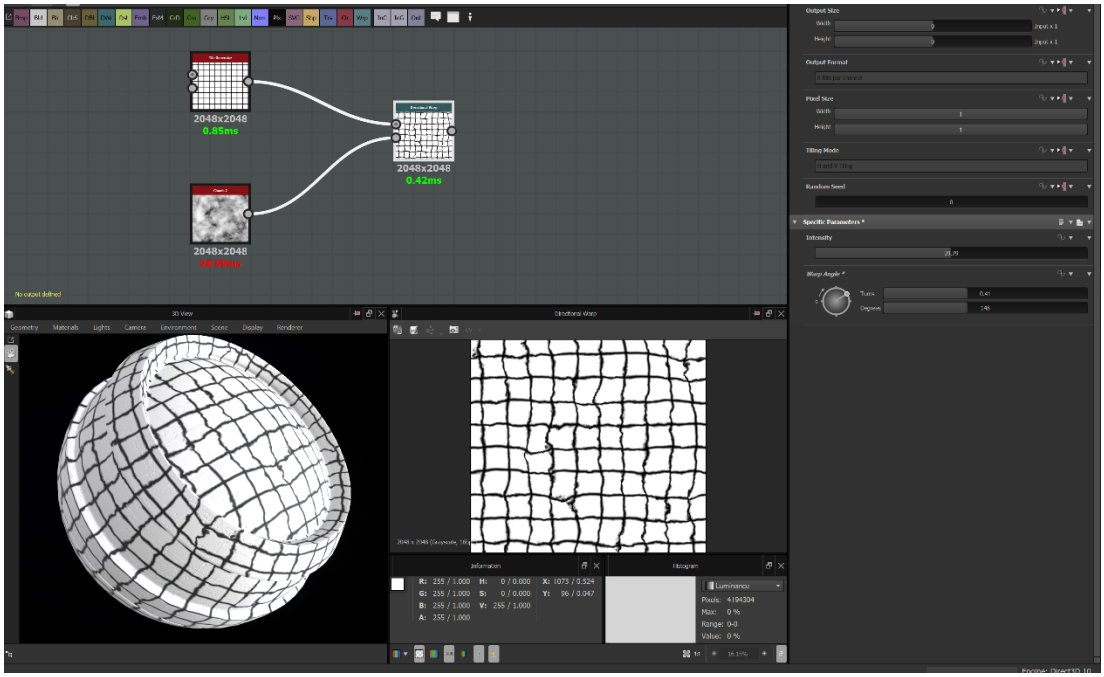
- **Blend**
  This can be considered the most used Node in Substance Designer. It is used to blend all kinds of different levels of detail and colors together. There are a ton of blending modes to choose from, and a section for a Mask to be added as well. It may surprise users how flexible this node can become during the course of production.

  Expect a LOT of happy accidents when using this node!



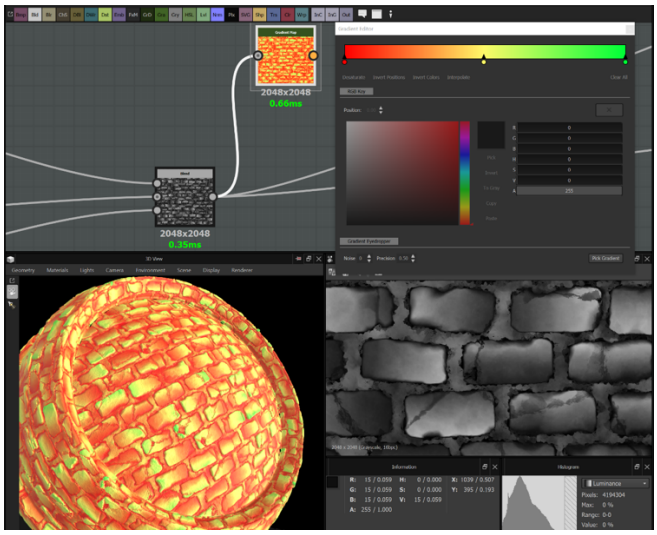
- **Directional Warp**
  This Node is extremely helpful when it comes to removing the Procedural look to a Substance. Plug some kind of grey scale node (preferable a Noise Node) into this node to help distort the Substance.

Large values can produce some interesting results, but sometimes a small value is all that is needed!



- **Gradient Map**
  This Node is **EXTREMELY** powerful, although it might not seem that way at first. Depending on the Node that is getting plugged into the **Gradient Map**, the **Gradient Map** will **assign a color based on the current values in the previous Node.** This means if a Red Color is set as the 0 Value in the Gradient Map, it will be colored Red for anything that has a Value of 0 in the node being plugged in.

  There is also an amazing tool in the **Gradient Editor** called **"Pick Gradient".** This will allow users to draw a line across any screen, or image, and pick all the colors that the user draws across. This is extremely helpful when needed to sample multiple colors at once time.



- **Levels**
  This Node works just like it does inside of **Photoshop**, and any other application that has a Levels feature. It is pretty straight forward on how it is used.

  Some users use it as a placeholder since it's computation time is really cheap. This works really well inside of Graphs that get extremely large and complicated.

- **HSL**
  Much like the Levels Node, the HSL Node is pretty straight forward. It gives the typical sliders for **Hue, Saturation, and Lightness**

- **Normal**
  To convert a greyscale image into a Normal Map, this Node must be used. Users will also be able to control the overall intensity of the Normal Map with this Node as well in the Properties Window

- **Transformation 2D**
  To manually move, rescale, rotate, or skew, the Transformation 2D Node will get the job done. It acts a lot like "Free Transform" inside of Photoshop.

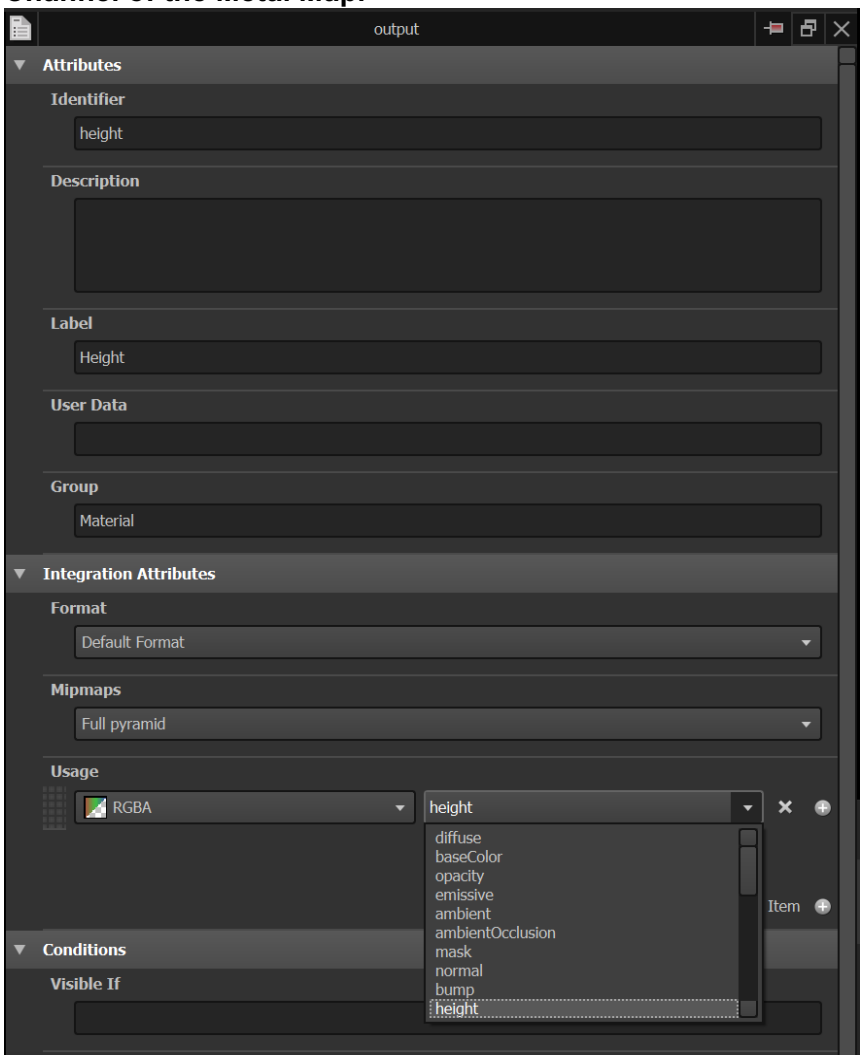
  When using this Node, pay attention to the tiling of the Substance. Seams will become very apparent when using this Node, so be very careful when using it. If the **Transformation 2D Node** causes tiling, the **Make It Tile Photo Node** will help clean up the seams.

- **Uniform Color**
  Probably the simplest Node available. The only thing this Node will do is create a solid color, either in **sRGB or Greyscale.**

- **Output**
  To create additional **Outputs**, such as **Height or Ambient Occlusion**, this node will be used. In the **Properties Window**, there are a variation of options for Formats, Mipmaps, and Usages.

  For this course we will be focusing on just the **Usage** section of this Node. By clicking the **Add Item +,** a list of options will then become available. Channels from **RGBA to separate channels can be selected**, and then given a **Map Type.**

  ***Don't forget to add an Identifier, Label, and Group name to the Output node!**
  **** There can be multiple Usages per Output Node. This is extremely helpful if Maps need different information on certain channels. This is common in Gaming Engines. For example, in Unity, the Roughness Map is the Alpha Channel of the Metal Map.**



Other Nodes

There are so many Nodes inside of **Substance Designer**, and each Node almost has endless possibilities of the things that it can make. It would take forever for me to cover every node, but I will cover some of my favorites.

These are the Nodes I use the most in Substance Designer:

- **Shape**
  Extremely straight forward with its Atlas Map of shapes to choose from. However, this will probably be Node that starts most Substances. Pretty much any shape in the world can be made with a combination of **Blend Nodes** and Shape Nodes!

- **Tile Generator**
  Don't' take the name of this Node too literally. Its good at making **Tiles**, but it's really good at taking the **Shape Node and making it Tile**. This Node is a good building block for anything that needs some repetition.

- **Edge Detect**
  The name is pretty self-explanatory on this one. **Edge Detect** will look at the **Parent Node** being plugged into it, and will **generate lines around contrasting values.** This Node is good at creating lines for **Tile Grout**, among other things.

- **Splatter**
  Most people mistake this Node for **"Scatter"**, because that's what it does really well. It can take a **Shape Node** and randomly scatter it around. This can be exceptionally helpful when trying to break the Procedural look of certain Nodes.

- **Clouds**
  One of the many **"Noise" Nodes**, but one of my favorites. There are three variations, all with a different look and feel. They however give a good organic feel to a lot of noise and distortion.

- **Perlin Noise**
  Another **Noise Node** that has a few variations. My favorite is the **Perlin Noise Zoom**, as it gives a little more control over how the Noise looks and acts.

- **Histogram Scan**
  One of my favorite Nodes to experiment with. It is my **Go-To** when I need to switch things up a little bit. Plug any **Grey Scale Node** into this guy, and it will slowly let you **pull values back** into the Node. This can give all kinds of crazy results that would be unexpected, and allow for small details to be added easily to the chain! Best way to understand the Histogram Scan is to start plugging other Nodes into and see what comes from it!

AUTODESK UNIVERSITY

Chapter 5: Creating a Material

*** A majority of this Chapter will be done via Live Demo in the Class. I also recommend checking out the resources attached with this course, as I have uploaded some of my materials that I have created. I just ask that you do not resell them! But feel free to dissect them for your own learning experience. I have learned a lot this way.**

**** Once this class goes Free for the public, I will create a video on my YouTube Channel for this portion. My YouTube can be found here for future reference: https://www.youtube.com/scottdewoody**

The key power in **Substance Designer** comes from its ability to generate anything with its Node-Based workflow. However, this can be extremely daunting at first due to the complexity of the application. But thankfully it is not nearly as scary at it seems when you break down the workflow in to a series of steps, or processes.

Substance Designer Workflow Guidelines:

- View everything has **SHAPES**.
  - Don't take Materials literally! Materials are just **SHAPES**. Everything can be broken down into **basic building blocks.**
- Think about **POSITIVE** and **NEGATIVE SPACE**
  - Don't just look at the **POSITIVE SPACE** the shape of the detail creates, but also the **NEGATIVE SPACE** the shape creates. Sometimes it is easier to generate the negative space versus the actual detail.
- Start **BIG Detail** and work to **SMALL Detail**.
  - Just like every other art form, working from **Big to Small** is always the key to success. And it's exactly the same when working in **Substance Designer**. The main step that will be repeated over and over is Blending nodes together in the end to combine details into one map. This will allow the material to be broken down easier, and combined together in the end.
- Work in **GREY SCALE** all the way until the **VERY END**.
  - Just like creating a photograph, or illustration, **if you cannot read the image well in Black & White, it won't read well when color is added to it.** This is very much in the mind set of getting the basics right at the start, so everything else comes together nicely. Getting the material to read right in **Black & White** in **Substance Designer** makes the rest of the process go extremely smoothly.
  - Also **Substance Designer** tends to take **longer to calculate** things in **Color**. Not that it is drastically longer than **Grey Scale**, but when you stack enough nodes down the chain, it can make all the difference. So keep everything **Grey Scale** up until the very end of the node chain.

Substance Designer Workflow Optimization Techniques

- Always create new Graphs with:
  - **Size Mode: Relative to Parent**
  - **Format: 16-Bit**
- Work with **Metal/Roughness Workflow** inside Substance Designer for the most flexibility to move between different Rendering Engines.
- Working at a **2k resolution is recommended**, but it is best to work at the target resolution. A 2k resolution just covers the best performance vs quality.
- For **Raytrace Engines**, export maps at the **highest resolution that can be supported.**
  - **Substance Designer can export out 8k Maps, but it can take a while sometimes depending on the complexity of the Substance.**
  - **I like to export 4k Maps, and only do 8K if I need the additional detail and resolution for the render. Close-Ups shots benefit from the additional resolution.**
- Set **Uniform Color Nodes** to **16px**.
  - This size is optimal for these kinds of nodes, and it can help save some computational time.
  - Does not need to be done while in the design process, but is a good idea to do this **prior to packaging** up the Substance.
- Use the **Base Material Node** as a bridge to creating the final **Outputs**. It will also help keep things grounded with **Physically Based Properties**.
- If possible, Import the model that will have the texture applied to it to help with scale and tiling issues.
  - **Make sure it is UV Unwrapped!**
- Only have the **Outputs you require** in the Graph.
  - **This is key when publishing a Substance. Everything in the Substance will be packaged up together.**
- **Expose Parameters** for easy adjustment and linking!
- **Create Tools** out of repetitive tasks!
  - Noises
  - Grunge Masks
  - Colorization Tools
  - Brick Pattern Tools
  - Crack Generators
- **SAVE HAPPY MISTAKES!**
  - It is really easy to create something that was unexpected. Save these for later! You never know when it will become useful.

Chapter 6: OpenGL VS Iray

***To use Iray on the GPU, (which it is design for) there needs to be an NVIDIA GPU that supports CUDA in the machine running Iray. Thankfully any NVIDIA GPU in the past few years does support CUDA. The issue is if the computer being used has an NVIDIA GPU. If not, Iray will default back to CPU Rendering. There can be some performance issues here depending on how fast the CPU is.**

****Recent Apple computers have moved from NVIDIA to AMD GPUs. So Iray will only run on CPU on these machines.**

*****OSX also does not support DirectX, so the Tessellation Shader is unavailable in OpenGL.**

Since **Substance Designer 5.3**, **NVIDIA's Iray** has been included with **Substance Designer**. This finally opened the application up to truly making it easy for creating materials for **Raytrace Engines**. I see a lot of people ask which render engine inside of **Substance Designer** they should use, and the answer is: **BOTH!**

Allegorithmic has done an amazing job getting both **OpenGL** and **Iray** working hand-in-hand together. So using the best of both worlds is ideal! However, working in one or the other is just fine too. This really comes down to personal preference. I'm one who likes to work in **OpenGL** a majority of the time. This helps me design the material I am wanting to make for **Gaming Engines**, and then when I'm ready I toggle **Iray** on to preview the material in a **Raytrace Engine.** There is very minimal difference between the two, so there isn't much tweaking after that. However, this is my own personal preference. If a user it only looking to target a **Raytrace Engine**, working in only **Iray** is completely acceptable.

If more complex materials are being generated with **Refraction** and **Translucency**, users will have an **easier time working with Iray**. It supports those features right out of the box, where **OpenGL** needs to have a Shader that support those features imported.

Chapter 7: Exporting from Substance to an Engine

Once a material in **Substance Designer** is ready, there are three ways to export the material out. The method will determine the target **Rendering Engine** that will be used for the final output.

.sbsar

Applications Supporting: Unity, Unreal, Stingray, Marmoset Toolbag

This is the **native format unique to Substance Designer**. When a Substance is "Complete", users can Publish the substance into **one complete .sbsar file**. This will contain everything needed to generate the Substance, plus it will also expose any parameters that the Publisher might have exposed for additional controls in a host application. **Any application that supports Substance out of the box only requires to import this file.**

***If the Substance is set to "Relative to Parent", the Host Application will be able to determine the size of the Substance. Users will be able to pick from any supported range of resolutions. In gaming engines, this property can actually be called on at run-time if needed!**

How to Publish a Substance:

1. Reset all Graphs back to **0x0 under the Output Size** in the Parameters Window.
    a. Only do this if the **Graph's Output Size** is set to **"Relative to Parent".**
2. Remove any Resources that are not required from all graphics and folders under the Package that will be exported. **Everything will be included into the .sbsar file, much like a .zip file works.**
3. Right Click on the Substance Package Name in the Explorer Window, and click **"Export .sbsar File".**
4. Select where to save the .sbsar file, and what the name will be.
5. Select which Parameters that need to be exposed to the end-user of the Substance.
6. Click "Ok".

Bitmaps

Applications Supporting: Everything!

The more standard approach, and will be the common approach for **Raytrace Engines**, will be to export out bitmaps from the **Output Nodes** that are specified in the Substance. Users will need to define the resolution in the Substance prior to exporting. For Raytrace Engines, **I recommend 2-4k bitmaps for exporting**. Going up to **8k can sometime be helpful,** especially when the material is going to be rendered up-close. However, 8k bitmaps in Substance Designer **can take a while to process** depending on the complexity of the Substance Graph

A list of all bitmaps that will be needed:

- Diffuse/Albedo (sRGB)
- BaseColor (sRGB)
- Reflection (sRGB)
- Glossiness (Linear)
- Roughness (Linear)
- Specular (sRGB)
- Metal (Linear)
- Bump/Normal (Linear)
- Height/Displacement (Linear)
- Translucency (Linear)
- IOR (Linear)
- Mask (Linear)

How to export Bitmaps:

***Only the Outputs in the current selected Graph will be available for export into a Bitmap.**

****Bitmaps will be exported at the resolution that is currently set in the Graph that is being exported.**

*****Make sure Outputs have been named with an Identifier in the Properties Window. This will be the name that shows up in the Export Images Window.**

1. Set the **Targeted Resolution** in the Properties Window of the Graph.
    a. Double Click the Grid in the background of the Graph Window for this to show up in the Properties Window.
2. Click the little **"Gear"** icon at the top of the Graph Window, and select **Export Outputs** from the dropdown list.
3. In the **Export Images Window**, select where to save the Bitmaps under the **Destination** section.
4. Select the targeted **File Type** with the **Format dropdown** list.
5. Set how the name will look under the **Pattern** section.
    a. The defaults will pull the name of the graph and the identifier of the **Output**. Additional variables and text can be added to this to match specific naming conventions
6. Select all the **Outputs** that are required for Export by **checking and unchecking** them in the Outputs section.
7. Click **"Save All"** to export the Bitmaps to disk.
    a. Substance Designer will go through each Output one by one. **There is no progress window to watch this work.** High resolution and complex File Types will determine the speed at which the Bitmaps are exported. This does not take a lot of time regardless.

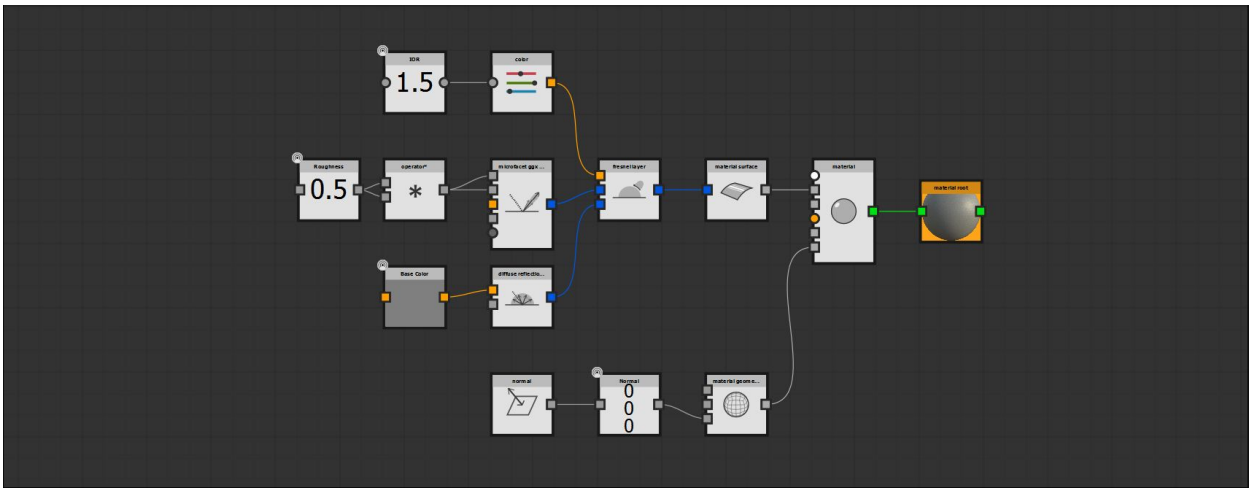
MDLs

Applications Supporting: NVIDIA Mental Ray, NVIDIA Iray, V-Ray (Coming Soon)

With the introduction of **Iray** into **Substance Designer 5.3**, **MDL** support was added as well. **MDL** stands for **Material Definition Language**, and is starting to be widely adopted by the industry. The beauty of **MDL** is that it separates the Material from the **Rendering Engine**, thus allowing the **MDL** to be translated **1:1** into any **Rendering Engine** that supports **MDL**. This means one single material can be generated inside of **Substance Designer**, and it will render the exactly same in any application. Currently the **Render Engines** supporting **MDLs** are **Iray** and **Mental Ray**. But **Chaos Group** has stated that **MDLs** are coming to **V-Ray** soon, even as early as the Winter 2016 Service Pack.

MDL Editor

In **Substance Designer 5.5**, Allegorithmic introduce the **first full MDL Editor** to the industry. **MDLs** can be written in just plane Notepad if needed, but Allegorithmic has taken their Node-Based workflow and created a **visual editor for authoring MDL.** Users are able to create an **MDL** with just about any property that is needed, and can even easily add “Coat” layers, and include actual Substances inside the **MDL** as well! Best of all they have worked on translation processes to move the **MDL generated into OpenGL**! There are going to be some discrepancies between the two, but it’s still pretty close to looking 1:1!

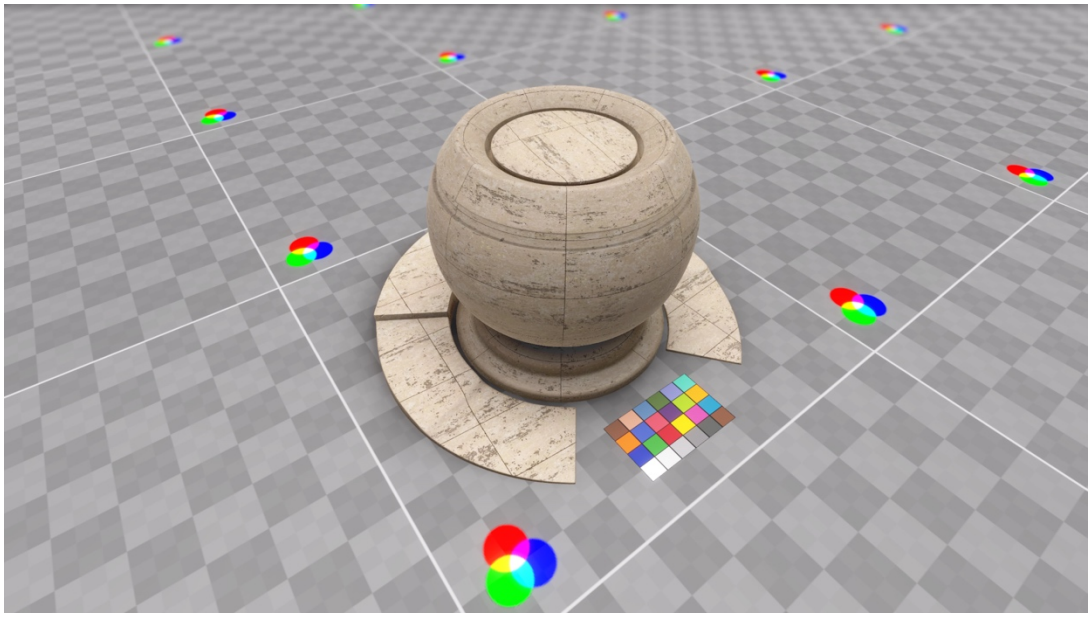


This is the default template for a Dielectric with IOR MDL

AUTODESK UNIVERSITY

Chapter 8: Moving from Substance to a Render Engine

Gaming Engines:



Unity:
For Unity, the only thing that needs to be done is to click and drag the Substance into the File Explorer. From there the Substance File contains a Material Object that can be dragged onto an object.

Unreal:
In order to bring a Substance into Unreal, the Substance Plug-in needs to be downloaded from the Unreal Asset Store, which is completely free. Once this is installed, the Substance File can be directly imported like any other asset into Unreal. Once the Substance is imported, there is a Material Node that can be applied to any object.

Stingray:
At the time of writing this paper, Stingray does not support Substance. However, by the time AU happens, it will support Substance directly! Sadly, I cannot comment directly on how the process will work. But talking with the developers over at Autodesk, they have assured me it is much like the other gaming engines. Substance files can be directly imported into the editor, and assigned to an object. It's just that easy!

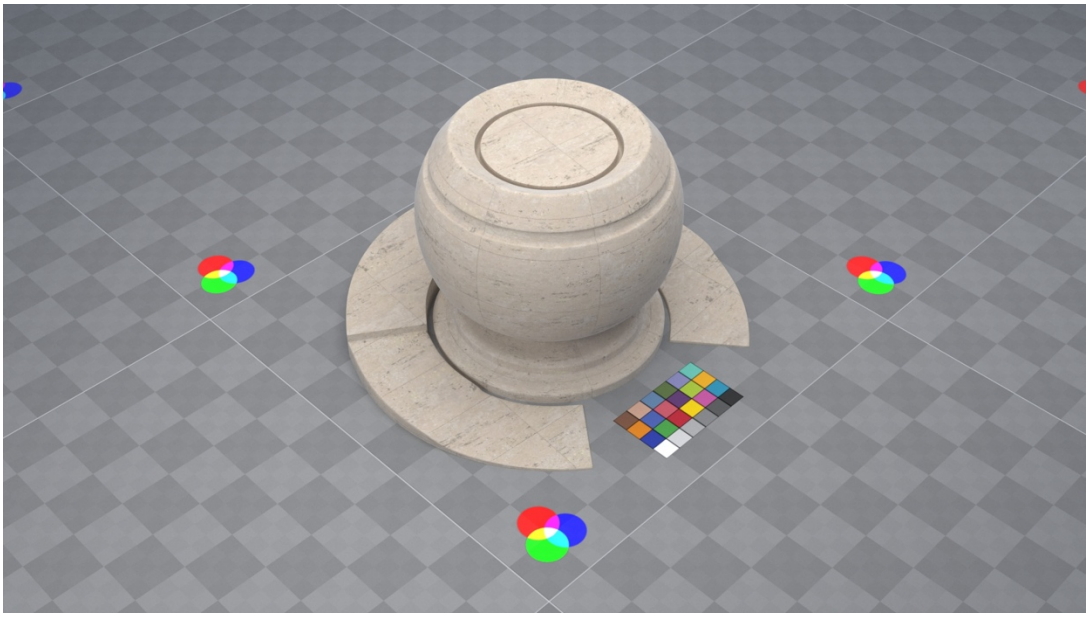
Marmoset Toolbag:
Substance Files can be directly imported in the Material Window by clicking the Import button. Imported Materials from FBX files can be converted to Substance Files under the "Extra" property of the Material

* I would like to note that during the time I was finalizing this document, Marmoset announced Toolbag 3. Since I'm not aware of the new improvements in version 3, I am assuming things are the same as version 2.

AUTODESK UNIVERSITY

Raytrace Engines:

V-Ray:

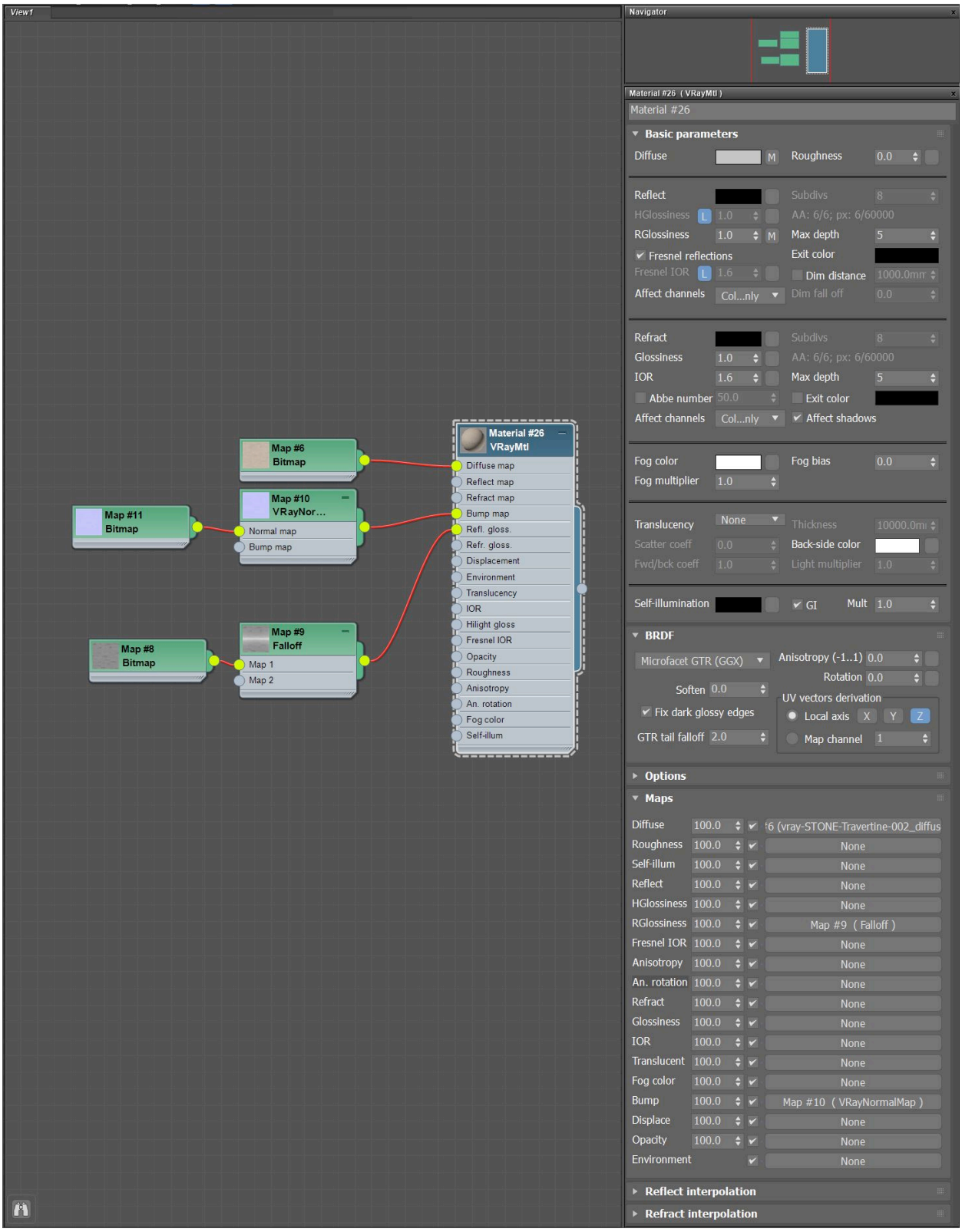


*Substance Designer has a Node for converting Substances generated in the Metal/Roughness Workflow to Outputs specific for V-Ray. This node is called the **BaseColor_Metalic_Roughness_Converter**. It can be found in the Library, and by hitting the Space Bar in the Graph Window and typing in the name.

1. Reset the Graph that needs to be converted to a **V-Ray Material** back to **0x0 in the Output Size in the Properties Window**.
    a. This only needs to be done if the Graph's Output Size is set to **"Relative to Parent".**
2. Create a new Graph either in the current Substance Package, or in a new one. Either way works, we just need a blank Graph to handle the conversion.
    a. Set the Template for the new Graph to **Empty**
    b. Set the Size Mode to **"Relative to Parent"**
    c. Set the format to **"16-bits per Channel"**
3. Click and Drag the Graph that will be converted into the blank Graph that was just created.
4. Bring the **BaseColor_Metalic_Roughness_Converter** node into the graph.
5. Double Click on the **Convert Node**
6. In the Properties Window, switch the **"Target" property to VRay**
7. Hit the **Number 3** on the keyboard to switch to the **Compact Material Link Mode**
8. Connect the Substance Graph to the **Convert Node**
9. Right click on the Convert Node and go to **Create > Output Nodes**
10. When prompted to **"Create Nodes for Hidden Outputs",** click **No**

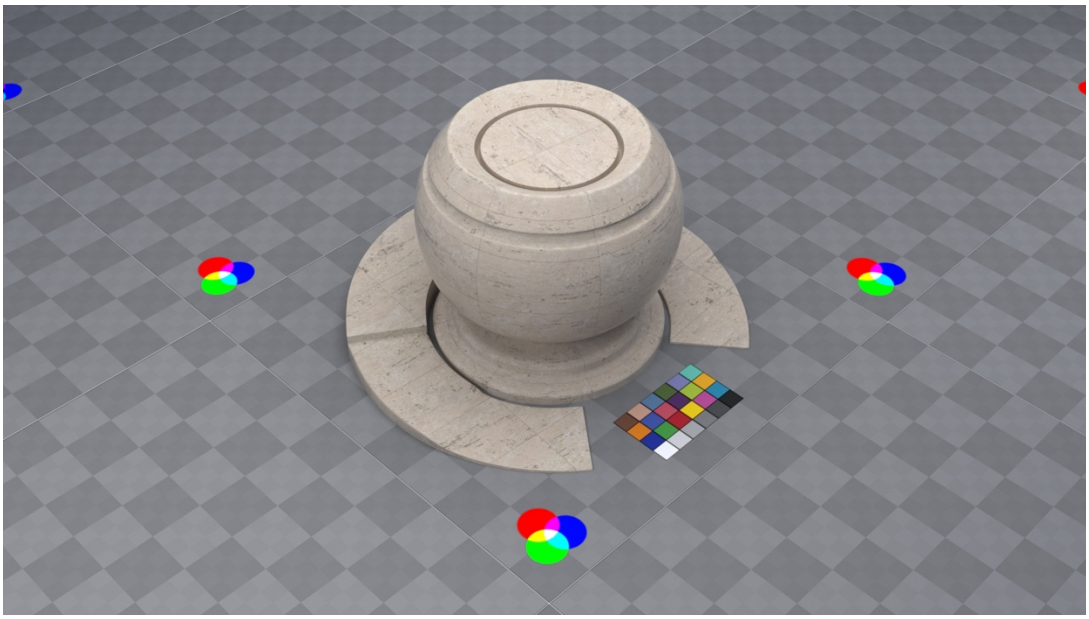
11. A **"Diffuse" "Reflection" "Glossiness" and "IOR"** Output will be created
12. Hit the **Number 1** on the keyboard to go back to the **Standard Link Mode**
13. Create Output Nodes for any additional Maps that the Convert Node does not make.
    a. **Bump/Normal**
    b. **Refraction**
    c. **Displacement**
    d. **Translucency**
14. Link the Outputs from the Substance Graph that was imported into this one in Step 2, to the newly created Output Nodes.
15. Export all the Output Nodes as Bitmaps
    a. See Exporting Bitmaps in Chapter 7
16. In the **VRayMtl**, plug in all the Output Bitmaps into the appropriate channels.
    a. Diffuse needs to be set to **"Automatic"** in the Bitmap Import options. This will import it with the **2.2 Gamma** that is set in the **3ds Max Settings**
    b. All other maps should be imported with **Gamma Override set to 1.0** in the Bitmap Import settings.
    c. If using a **Normal Map**, make sure to use the **VrayNormal Map** in the **bump map slot**. Load the **Normal Map as a Bitmap** as the **sub-map** set and have it imported at **Gamma Override 1.0.**
        i. The **Bump Map** value should be set to **100** in the **VRayMaterial**
    d. Set the **BRDF from Blinn to GGX**. This will give the material a more realistic result, especially for metals.

AUTODESK UNIVERSITY



Example showing the structure for a V-Ray Material

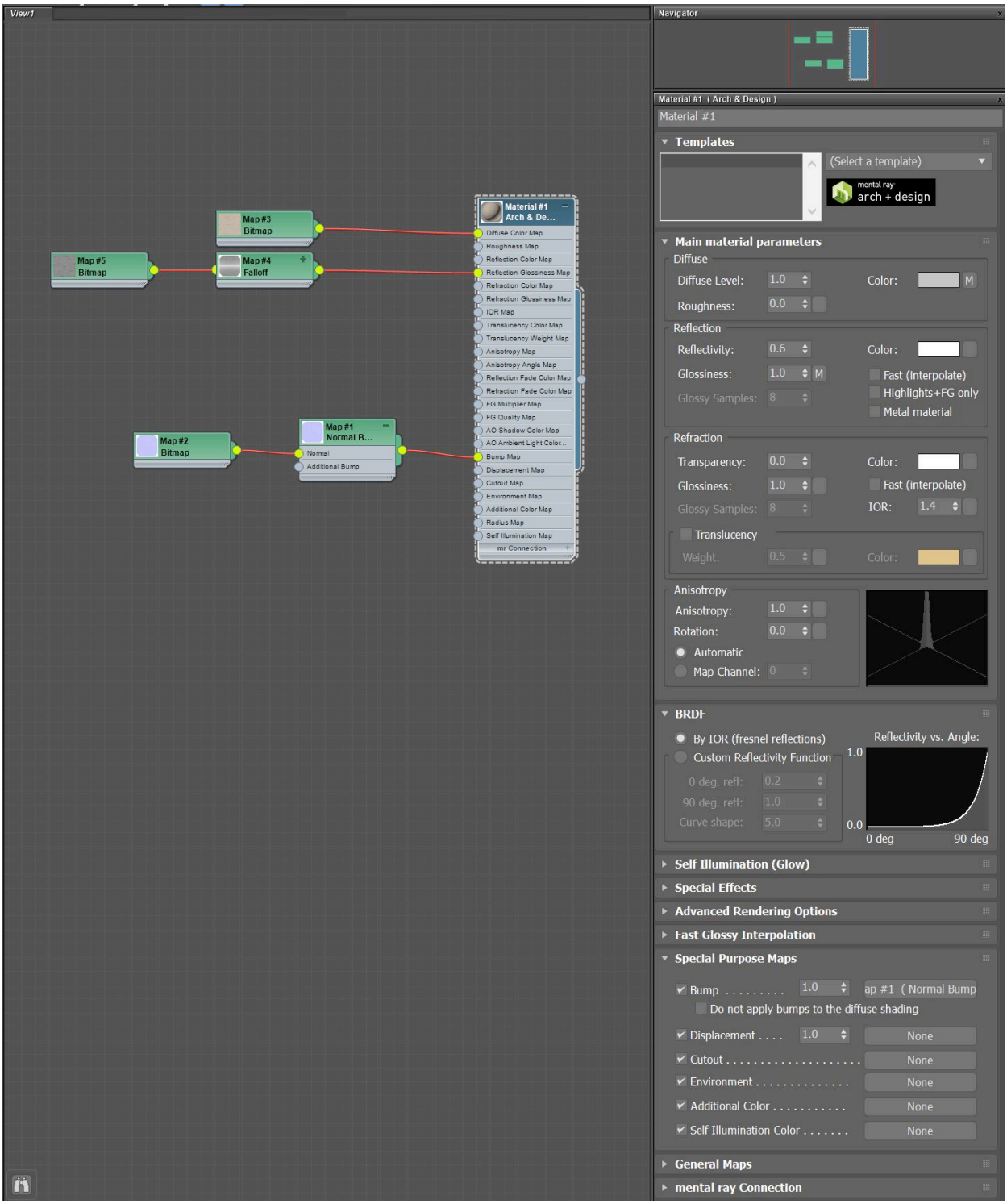
NVIDIA MentalRay and Iray



Sadly, there is no conversion preset in Substance Designer for **NVIDIA MentalRay and NVIDIA Iray**. However, the **V-Ray Preset** can be used in place, as the **Arch&Design Shader** resembles the **VRayMtl Shader** pretty closely. The outputs for the two are the same, but there are a few things to note:

- The **Reflection** section in the **Arch&Design Shader** has **two** controls for **Reflection**. It has a **Multiplier** that is a linear input from 0-1 and a **Color Input**. Its recommended to leave these at their defaults.
    - If **"Metal Material"** is checked, the Shader will **pull the color of the Metal from the Diffuse Property.**
- Under the **BRDF Section**, the option is set to **"Custom Reflectivity Function"**, this should be switched back to **"By IOR".**
- For **Normal Maps**, place a **Normal Bump Map** into the **Bump Map** channel. Then in the **Normal property**, select a **Bitmap** to add the **Normal Map** from Substance Designer.

AUTODESK UNIVERSITY



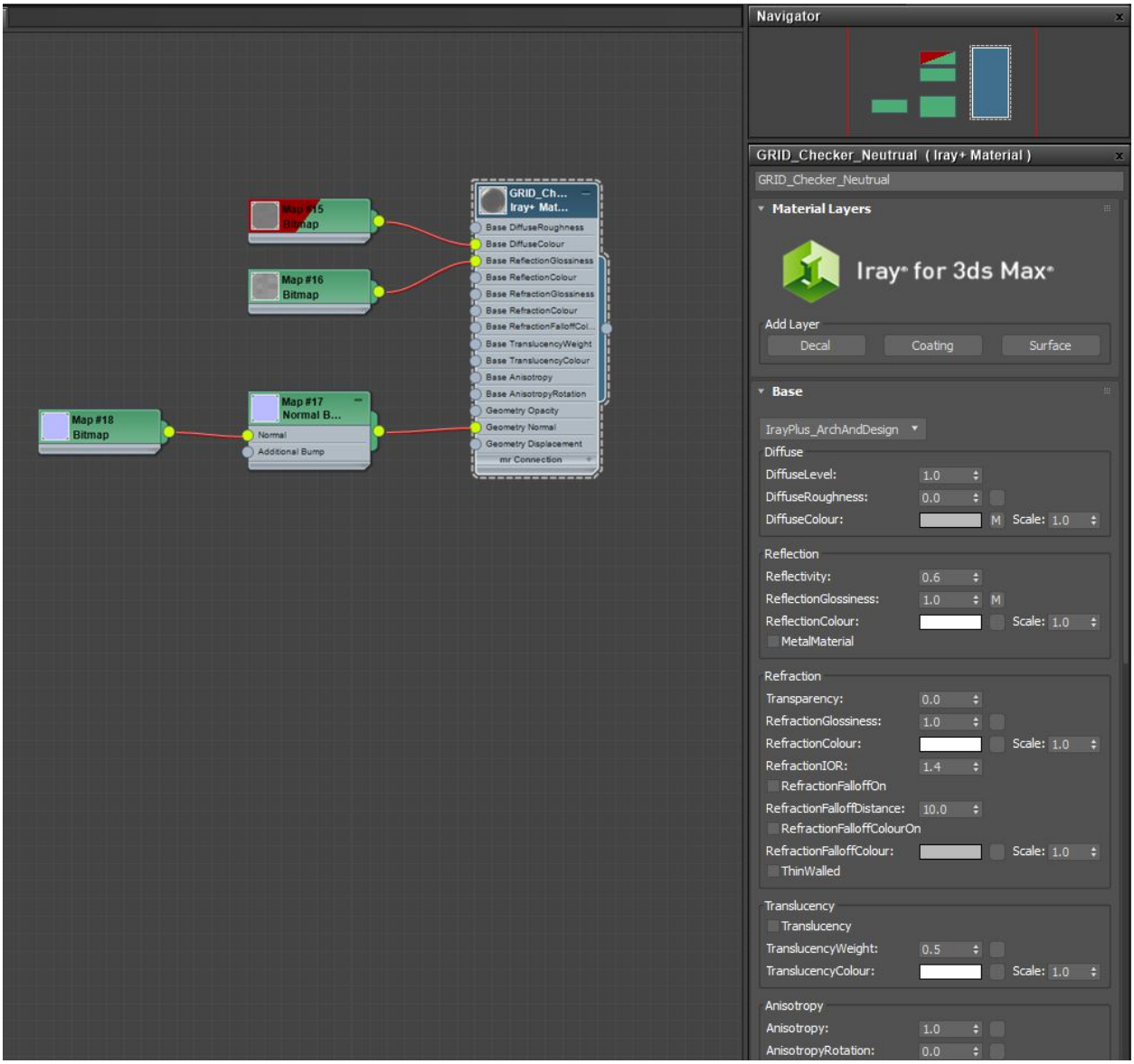
Example showing the structure for an Arch&Design Material (Mental Ray and Iray)

AUTODESK UNIVERSITY

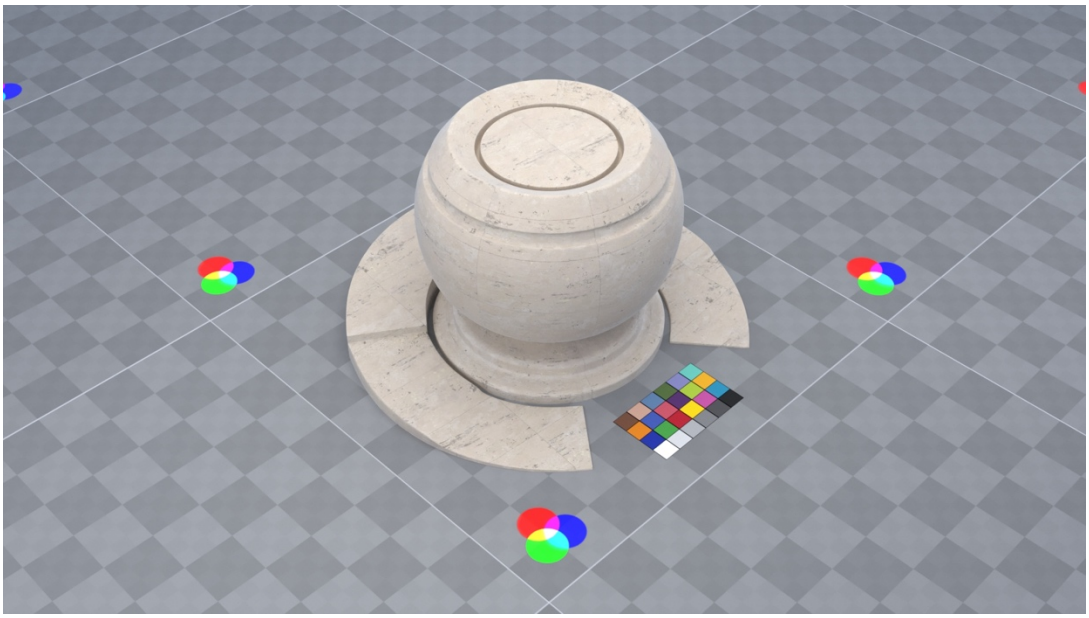
Iray for 3ds Max (Bitmaps)

NOTE: I HIGHLY recommend using MDLs with Iray for 3ds Max.

This process will resemble the **NVIDIA Mental Ray** and **NVIDIA Iray** workflow. I recommend setting the **Iray+ Material** to the **Arch&Design Mode**. This will turn the **Iray+ Shader** into a version of the **Arch&Design Shader** that ships with 3ds Max. One the Shader is set to this Mode, please refer back to the NVIDIA Mental Ray and NVIDIA Iray section, as the workflow is the same. For MDLs, that is further down in the document.



ART



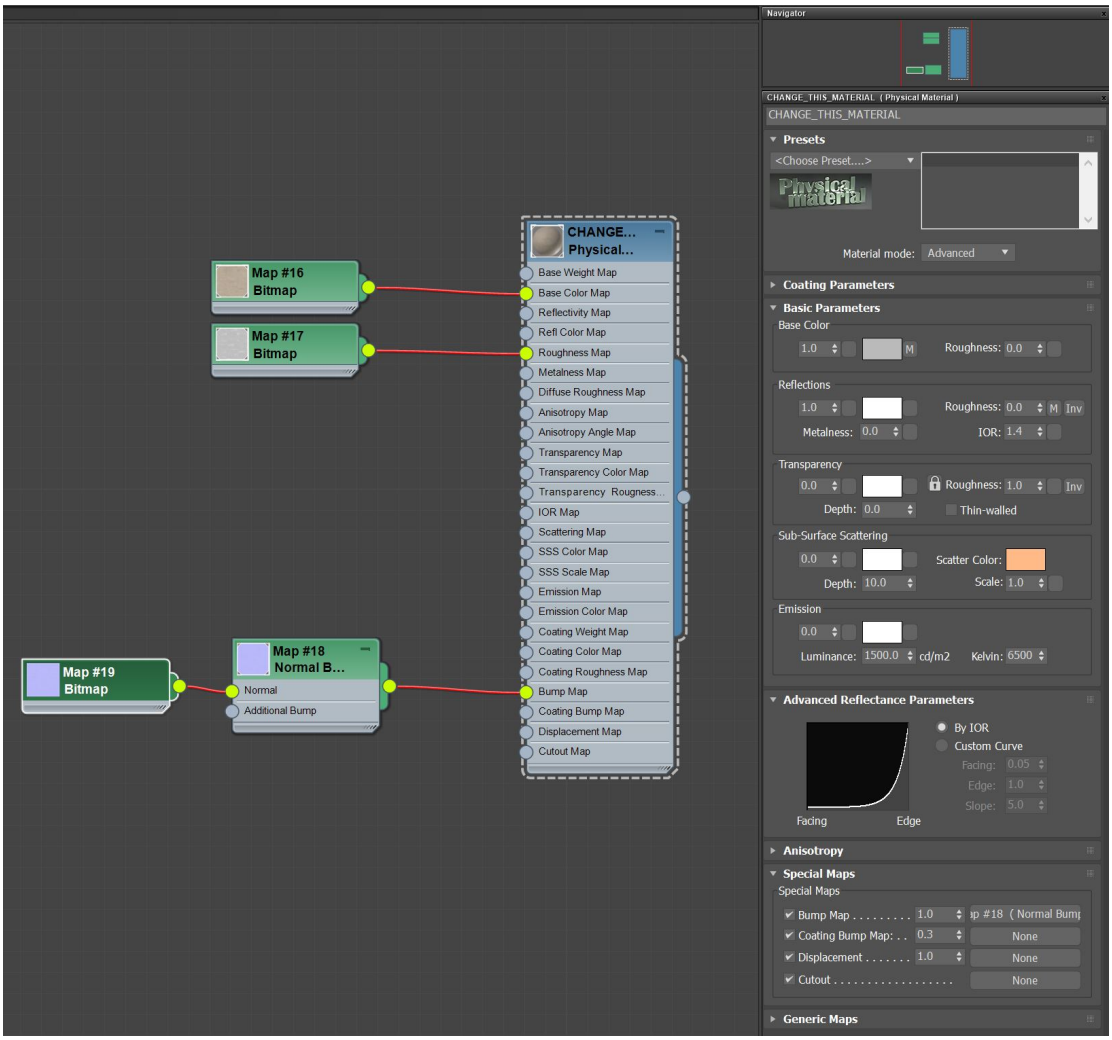
The **Autodesk Raytracer (ART)** is brand new this year to **3ds Max 2017**. With the inclusion of this new rendering engine comes a new Shader called the **Physical Material**. There are a few really exciting things about the **Physical Material**. First, it was designed with the mentality that **3rd Party Rendering Engines** could adopt the **Physical Material** as a supported Shader very easily. This would allow end-users to use which ever rendering engine they choose, but use **one single material inside of 3ds Max** between them. The second is this the first Raytrace Engine I've seen adopt the **Metal/Roughness Workflow** found in the Video Game Engines!

Since we are already working in the **Metal/Roughness Workflow** inside of **Substance Designer**, there is no need to do any conversions! All that needs to be done is to export all the **Outputs** being generated by **Substance Designer**.

There is one thing to note about the **Physical Material**. There is a button next to the **Roughness Property** that says **"Inv".** This stands for Invert, and it will invert the **Roughness Property** to act just like a **Glossiness Property** in the **Reflective/Glossiness Workflow**. This is there to help move older materials into the new **Physical Material**, and it is **ENABLED BY DEFAULT.** This will need to be disabled, **(The button is no longer Blue when it is disabled.)** because the maps being imported from **Substance Designer** are correctly based in the **Metallic/Roughness Workflow**.

1. In **Substance Designer**, set the **Graph** to the desired resolution for **Output**
    a. The resolution needs to be set only as high that is needed.
2. Export all required **Outputs** as **Bitmaps**

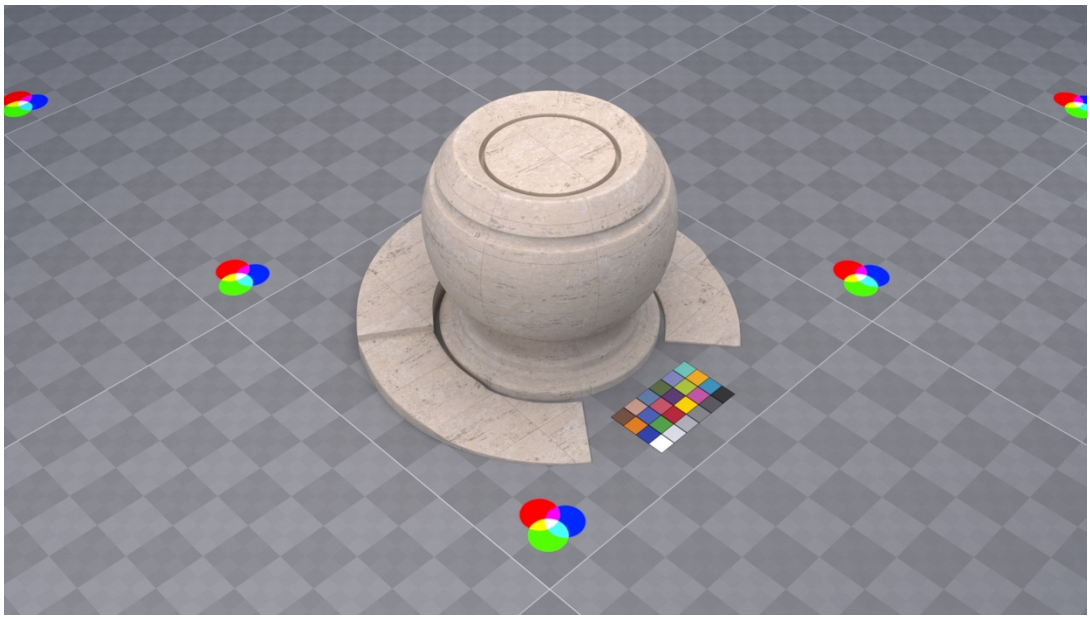
   a. **BaseColor** is used instead of **Diffuse** in this workflow
3. In the **Physical Material**, disable the **Invert Button** on the **Roughness Property**. This will allow the Shader to read the **Roughness Map** that is exported from **Substance Designer**
4. Import the **BaseColor Map** into the **BaseColor Property**
5. Set the **Metal Property** to either **0 or 1**, or import the **Metal Map**
   a. This needs to be imported with the **Gamma set to 1.0**
6. Set the **Roughness Property** to somewhere between **0 and 1**, or import the **Roughness Map**.
   a. This needs to be imported with the **Gamma set to 1.0**
7. Set the **Bump Map Value to 1.0**, and add a **NormalMap** to this channel. Then add the **Normal Map (from Substance Designer)** in the submap slot as a **Bitmap**.
   a. This needs to be imported with the **Gamma set to 1.0**



Example showing the structure for a Physical Material (ART)

AUTODESK UNIVERSITY

Corona

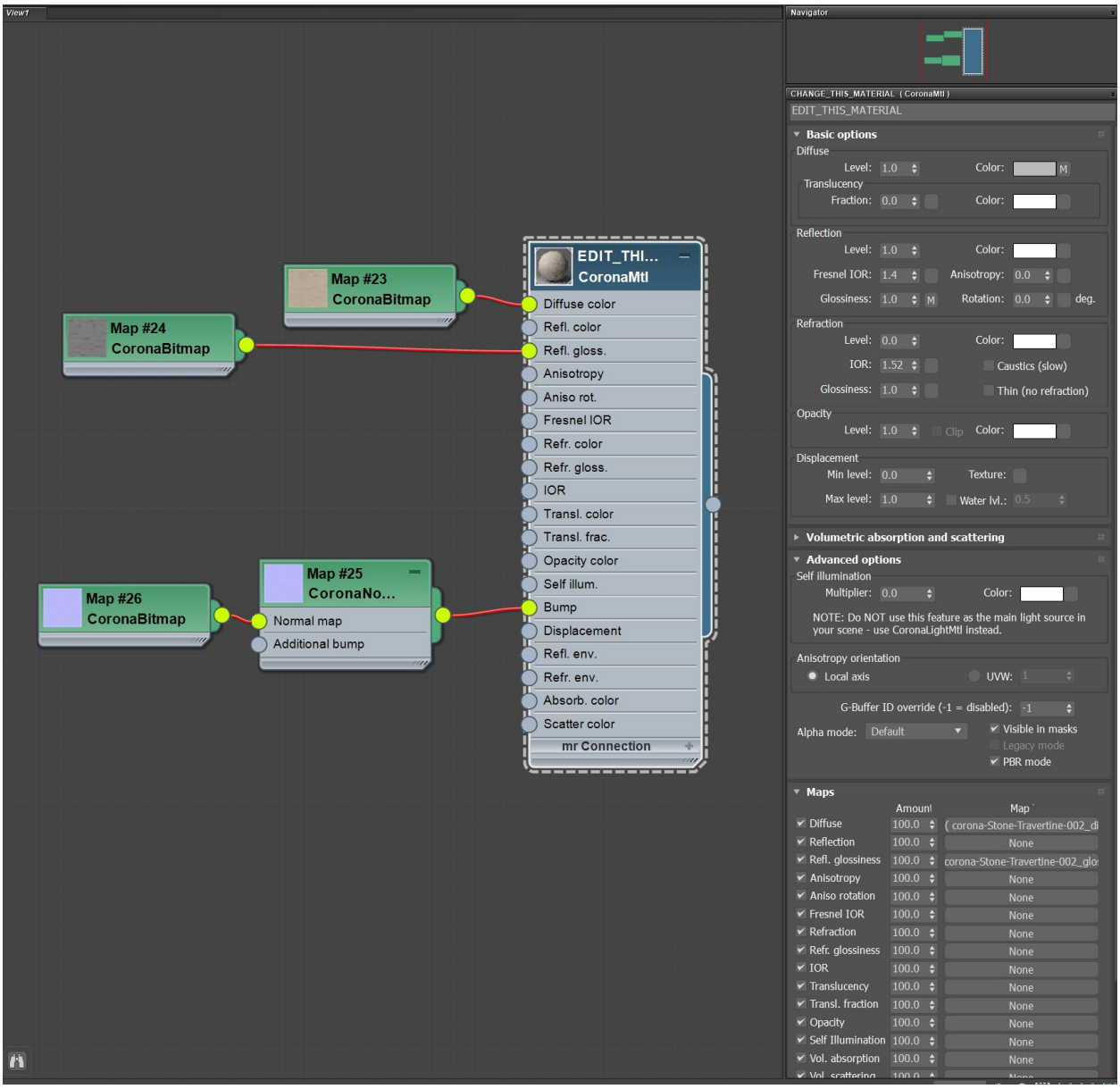


*Substance Designer has a Node for converting Substances generated in the Metal/Roughness Workflow to Outputs specific for Corona. This node is called the **BaseColor_Metalic_Roughness_Converter**. It can be found in the Library, and by hitting the Space Bar in the Graph Window and typing in the name.

1. Reset the Graph that needs to be converted to a Corona Material back to 0x0 in the Output Size in the Properties Window.
    a. This only needs to be done if the Graph's Output Size is set to **"Relative to Parent".**
2. Create a new Graph either in the current Substance Package, or in a new one. Either way works, we just need a blank Graph to handle the conversion.
    a. Set the Template for the new Graph to **Empty**
    b. Set the Size Mode to **"Relative to Parent"**
    c. Set the format to **"16-bits per Channel"**
3. Click and Drag the Graph that will be converted into the blank Graph that was just created.
4. Bring the **BaseColor_Metalic_Roughness_Converter** node into the graph.
5. Double Click on the Convert Node
6. In the Properties Window, switch the **"Target" property to Corona**
7. Hit the **Number 3** on the keyboard to switch to the **Compact Material Link Mode**
8. Connect the Substance Graph to the Convert Node
9. Right click on the Convert Node and go to **Create > Output Nodes**
10. When prompted to **"Create Nodes for Hidden Outputs"**, click **No**
11. A **"Diffuse" "Reflection" "Glossiness" and "IOR"** Output will be created

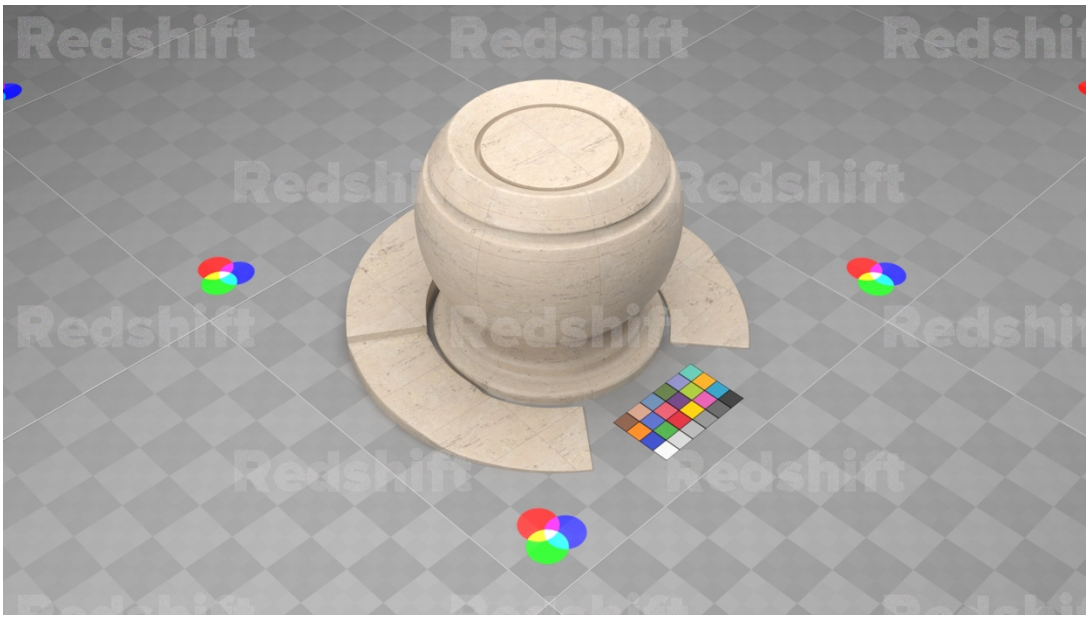
12. Hit the **Number 1** on the keyboard to go back to the **Standard Link Mode**
13. Create Output Nodes for any additional Maps that the Convert Node does not make.
    a. **Bump/Normal**
    b. **Refraction**
    c. **Displacement**
    d. **Translucency**
14. Link the Outputs from the Substance Graph that was imported into this one in Step 2, to the newly created Output Nodes.
15. Export all the Output Nodes as Bitmaps
    a. See Exporting Bitmaps in Chapter 7
16. In the **CoronaMtl**, plug in all the Output Bitmaps into the appropriate channels.
    a. **Diffuse** needs a **CoronaBitmap**, and needs to be set to **"Automatic" in the Bitmap Import options**. This will import it with the **2.2 Gamma** that is set in the **3ds Max Settings**
    b. All other maps should be imported as **CoronaBitmap** with **Gamma Override set to 1.0** in the **Bitmap Import settings**.
    c. If using a **Normal Map**, make sure to use the **CoronaNormal Map** in the bump map slot. Load the **Normal Map** as a **CoronaBitmap** as the **sub-map** set and have it imported at **Gamma Override 1.0**.
        i. The **Bump Map** value should be set to **1.0** in the **CoronaMtl.**

AUTODESK UNIVERSITY



Example showing the structure for a Cornoa Material

Redshift



This engine has taken an interesting approach to its shader. It gives users the ability to control the **Fresnel Effect** in multiple ways. It has the standard **IOR** value, an **Advanced IOR**, a **Color+Edge Tint**, and then a **Metalness** method. For this workflow, I recommend using the **Metalness** method in the shader, as it mimics the **Metallic/Roughness Workflow** and no conversion of the Material is required. However, in the **Metallic** method, it also gives the ability to define the **F0 value** of the material through a color channel. Since we're using the **Metallic/Roughness** workflow inside of **Substance Designer**, this output is not generated by default. Converting the IOR numbers to F0 values is an easy way to calculate this, and can be done through the following equation::

Linear Equation:

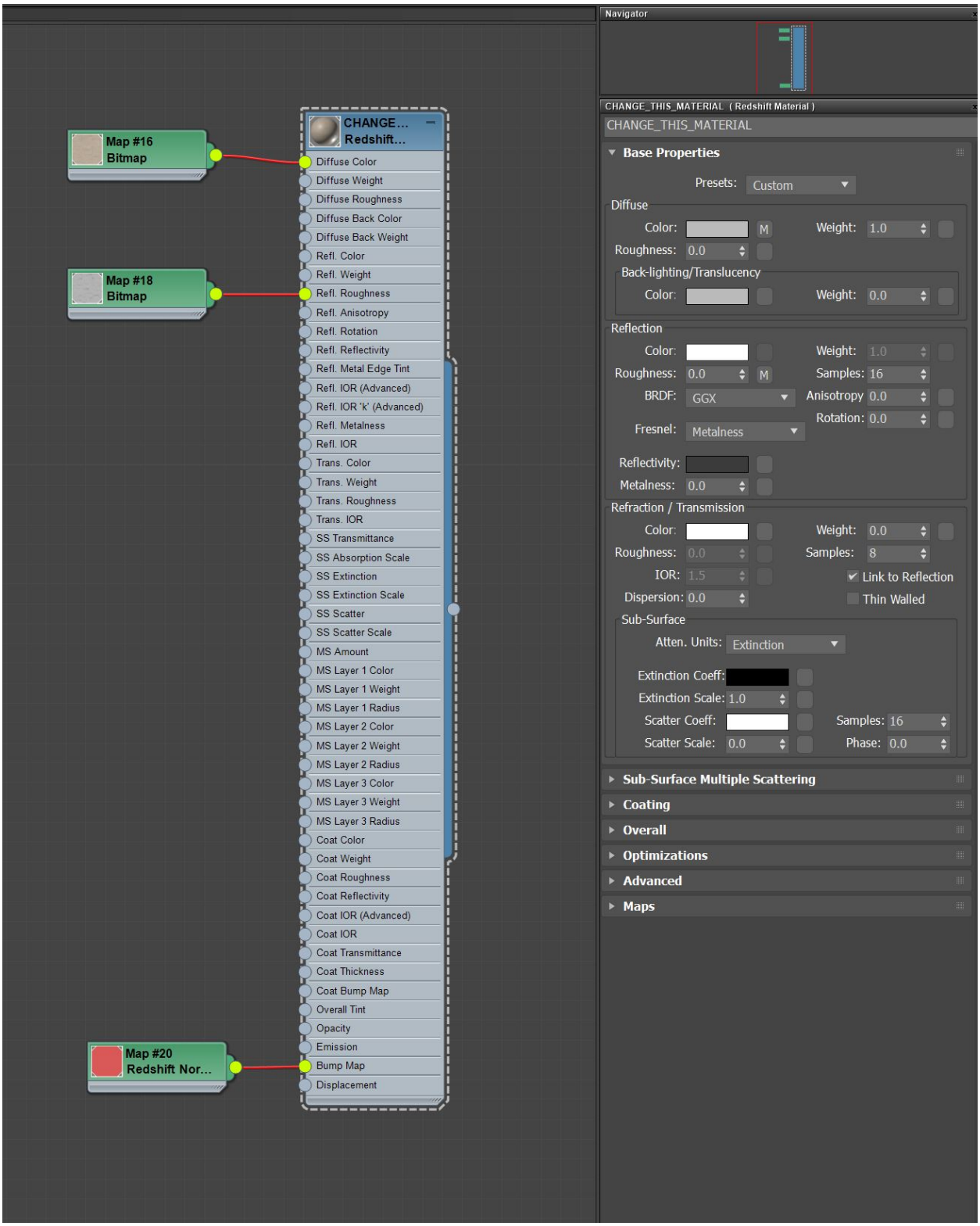
F0=((1-n)^2)/((1+n)^2)
Gamma Corrected Equation (sRGB):

F0=((((1-n)^2/(1+n)^2)^(1/2.2))*255

Switching the **Redshift Material** over to **Advanced IOR** will also enable users to introduce the **Extinction Coefficient (k)** into the shader as well. This will **disable the Metallic/Roughness Workflow**. So users will need to decide which way to go, as both are acceptable methods of created a physically based material. For this course, we're going to focus on the **Metallic/Roughness Workflow**, as it is the ideal way to move directly from **Substance Designer**.

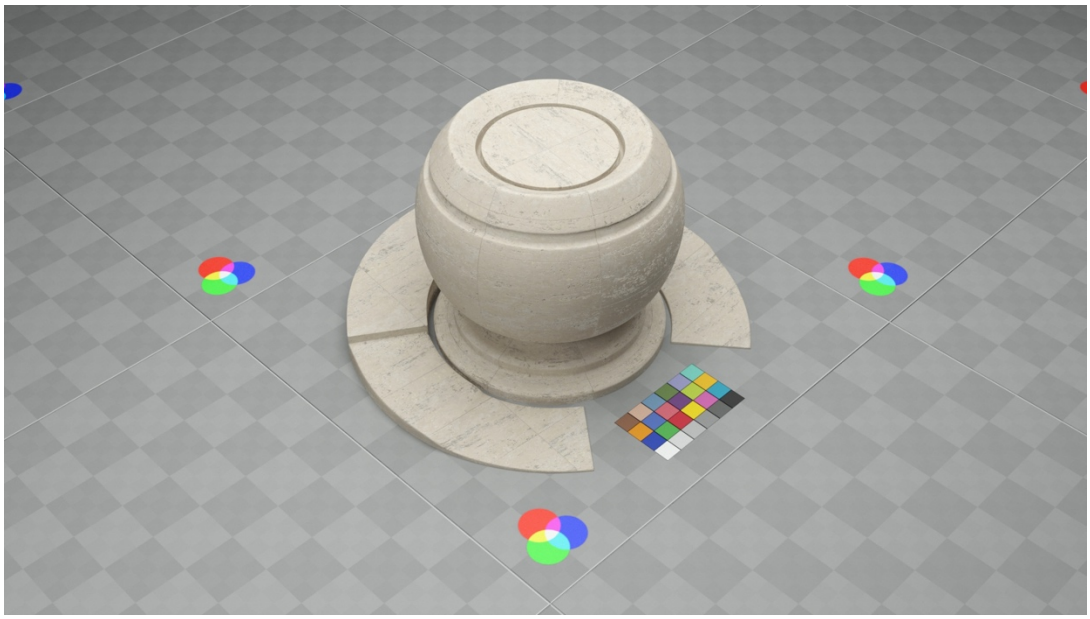
1. In **Substance Designer**, set the **Graph** to the desired resolution for **Output**
    a. The resolution needs to be set only as high that is needed.
2. Export all required **Outputs** as **Bitmaps**
    a. **BaseColor** is used instead of **Diffuse** in this workflow
3. In the **RedShift Material**, the **BaseColor Map** will plug into the **Color Map** under the **Diffuse** section of the material.
4. The **Roughness Map** will plug into the **Roughness Map Slot** in the **Reflection** section.
5. Set the **BRDF** to **GGX** under the **Reflection** section
6. Change the **Fresnel** mode to **Metalness**
7. Set the **Metalness** value to **0 or 1**, or plug in a **Metallic Map** from **Substance Designer**
8. Change the **Reflectivity Color** if needed, or plug in a Specular Map from Substance Designer.
    a. The **Specular Map** will need to be generated on the side from the typical **Metallic/Roughness Workflow**
9. In the **Bump Map Slot**, load a **Redshift Normal Map** in order to use a **Normal Map** generated from **Substance Designer**.
    a. In the **Redshift Normal Map**, check **“Flip Y”** in order for the **Normal Map** to render properly

AUTODESK UNIVERSITY



Example showing the structure for a Redshift Material

F-Storm

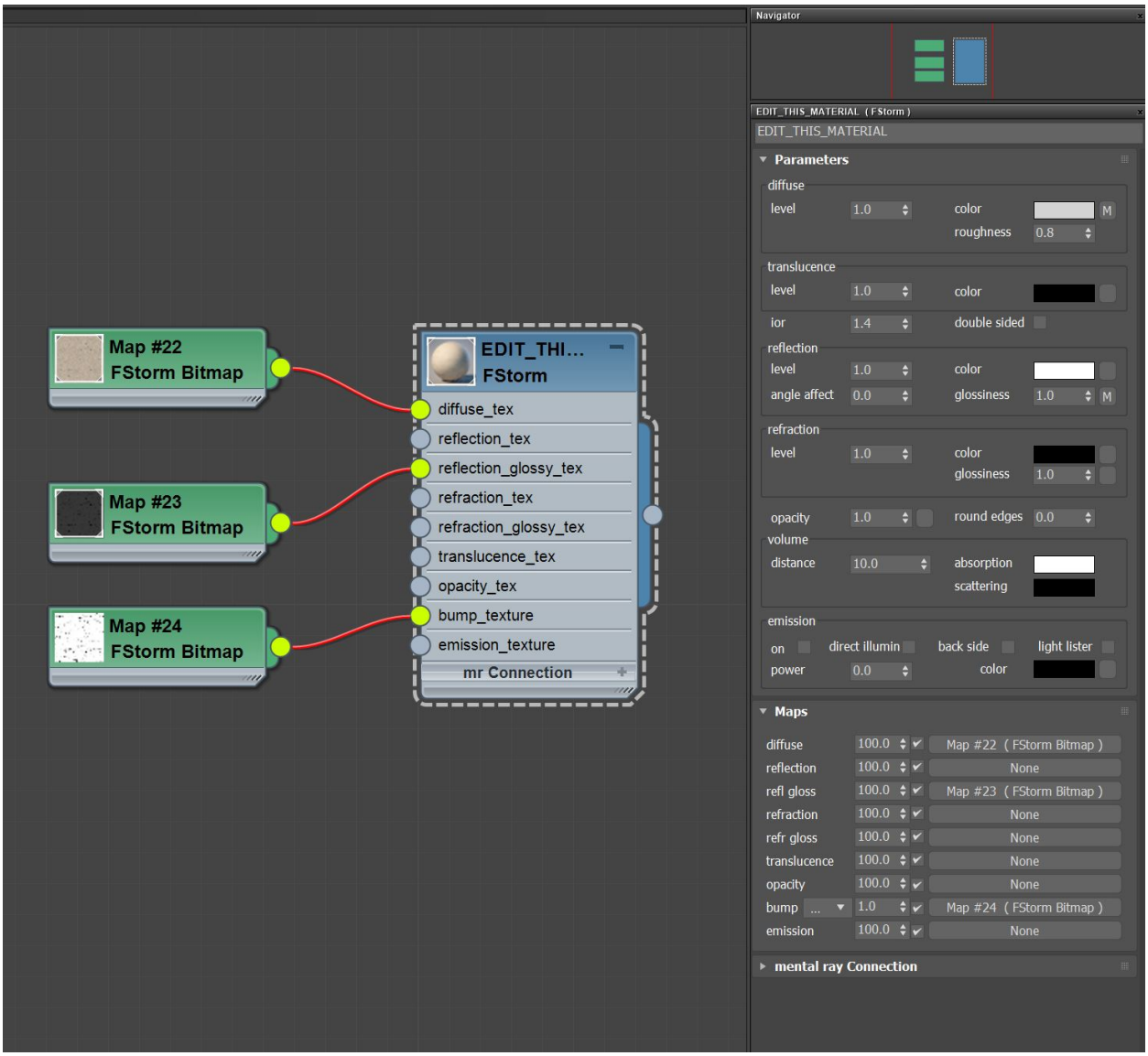


Even though **F-Storm** is still in **Alpha**, it's popularity is growing substantially. So I figured it would be good to include this engine as it is also currently free in its **Alpha** state. However, one feature it does not have (at the time of writing this document) is the support **of Normal Maps**. So in the **Substance Designer** process, a **Bump Map Output** will need to be generated. This is fairly easy, as **Normal Maps** in **Substance Designer** are generated from **Grey Scale Nodes**. These nodes can just plug into a **Bump Map Output Node**. From there, the process for exporting is the same as converting the **Graphs** for use with **V-Ray**.

1. Reset the Graph that needs to be converted to a **V-Ray Material** back to **0x0 in the Output Size in the Properties Window**.
    a. This only needs to be done if the Graph's Output Size is set to **"Relative to Parent".**
2. Create a new Graph either in the current Substance Package, or in a new one. Either way works, we just need a blank Graph to handle the conversion.
    a. Set the Template for the new Graph to **Empty**
    b. Set the Size Mode to **"Relative to Parent"**
    c. Set the format to **"16-bits per Channel"**
3. Click and Drag the Graph that will be converted into the blank Graph that was just created.
4. Bring the **BaseColor_Metalic_Roughness_Converter** node into the graph.
5. Double Click on the **Convert Node**
6. In the Properties Window, switch the **"Target" property to VRay**
7. Hit the **Number 3** on the keyboard to switch to the **Compact Material Link Mode**

8. Connect the Substance Graph to the **Convert Node**
9. Right click on the Convert Node and go to **Create > Output Nodes**
10. When prompted to **"Create Nodes for Hidden Outputs",** click **No**
11. A **"Diffuse" "Reflection" "Glossiness" and "IOR"** Output will be created
12. Hit the **Number 1** on the keyboard to go back to the **Standard Link Mode**
13. Create Output Nodes for any additional Maps that the Convert Node does not make.
    a. **Bump/Normal**
    b. **Refraction**
    c. **Displacement**
    d. **Translucency**
14. Link the **Outputs** from the **Substance Graph** that was imported into this one in Step 2, to the **newly created Output Nodes.**
15. **Export** all the **Output Nodes** as **Bitmaps**
    a. See Exporting Bitmaps in **Chapter 7**
16. When working with the **FStorm Material**, a **FStorm Bitmap** will need to be used to load in the **Diffuse** and **Glossiness Maps**
17. For the **Bump Map**, use the **FStorm Bitmap** in the **Bump Map Slot.**
    a. Leave the **Bump Map at 1.0**
    b. Set the **Bump Map Power** to **1/10^2** to have an appropriate level of **Bump**. Other values appear to be too strong or weak currently for what is produced with **Substance Designer.**

AUTODESK UNIVERSITY

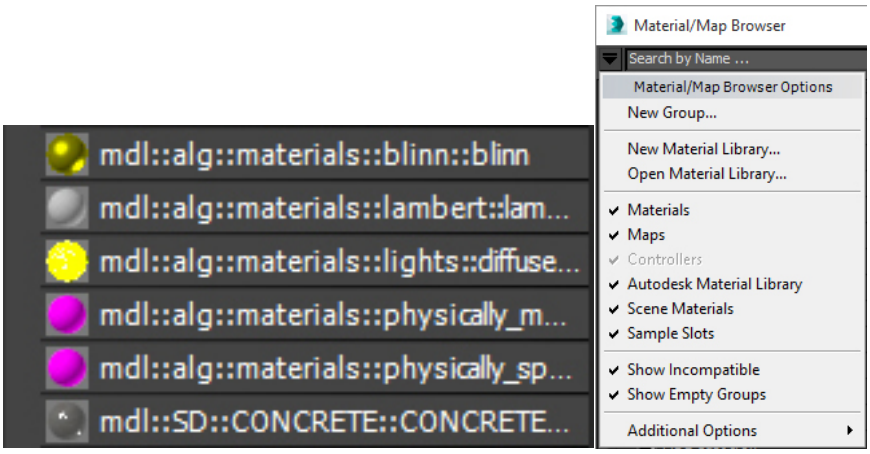


Example showing the structure for a FStorm Material

NVIDIA MentalRay and NVIDIA Iray (MDL)

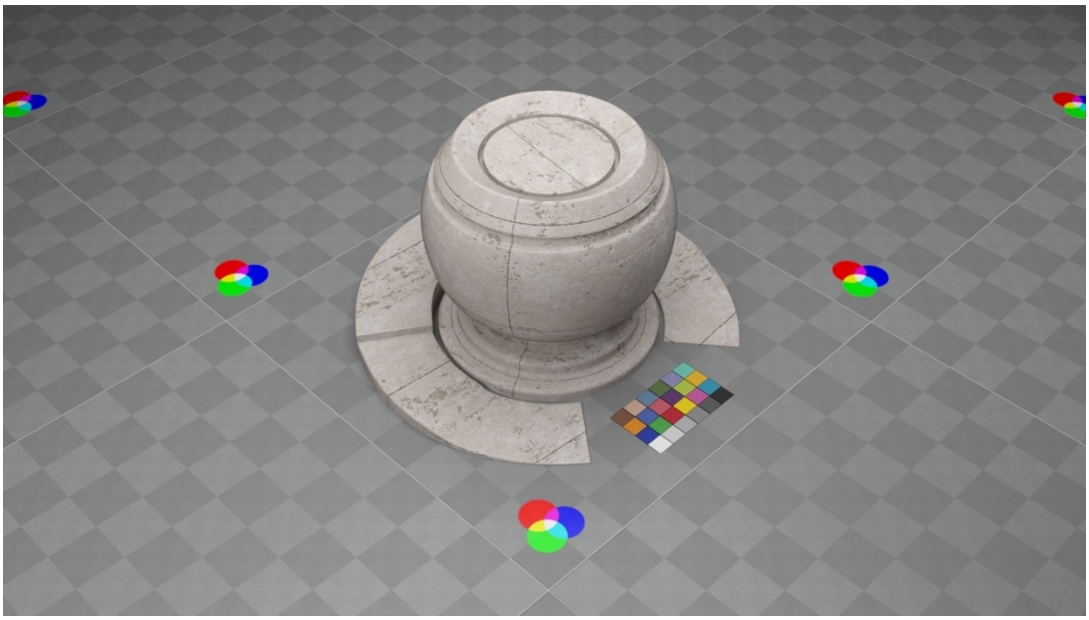
In order to import **MDLs** generated inside of **Substance Designer**, there are a few extra steps that are required. **3ds Max 2016 and 2017** needs a few extra support files, and the **MDLs** must be place in a certain location. These are the steps as of the time this paper was written.

1. Enable **MDLs** inside of **3ds Max 2016 and 2017** with the steps on this blog post from NVIDIA:
   http://blog.mentalray.com/2015/05/08/using-mdl-with-mental-ray-in-3ds-max-2016/
2. Copy the files in the following location:
   C:\Program Files\Allegorithmic\Substance Designer 5\resources\view3d\iray
3. To this location:
   C:\Program Files\Autodesk\3ds Max 2016\NVIDIA\shaders_3rdparty\mdl
4. In Substance Designer **enable Iray**
5. Change Material to the **Blinn shader** in the Viewport by going to:
   Materials > Default > Definitions > mdl::alg::materials > Blinn
6. Design the **MDL** using the nodes in the editor, and edit the Blinn properties by:
   Materials > Default > Edit
7. Export the **MDL** by going to:
   Materials > Default > Export Preset…
8. Name the **MDL** and select a location to export the **MDL** with its associated bitmaps
   - Don't use the " - " character for separation. For some reason the **MDL** will not export with that character in the name. Use the " _ " for separation.
9. Open the **MDL** file in a text editor and **edit Line 18**. Remove the " ***** " from the material name.
   Example: export material **NAME_OF_MDL(*) –** Remove the * to be:
   NAME_OF_MDL()
10. Copy the **MDL** and its associated bitmaps to a folder that can organize the **MDLs** here:
    C:\Program Files\Autodesk\3ds Max 2016\NVIDIA\shaders_3rdparty\mdl
11. Open 3ds Max 2016 and open the Material Editor
12. Open the Mental Ray Shader tab in the Material Browser and select the MDL!
    - If for some reason the **MDL** Is not listed with **NVIDIA Mental Ray or NVIDIA Iray** as the active Render Engine, enable the "**Show Incompatible"** option in the **Material Browser** via the drop down menu in the top left corner of the **Material Browser**. The **MDLs** are still compatible, there is just a graphics bug in **3ds Max** that may not display them.


Material/Map Browser
Search by Name ...
Material/Map Browser Options
New Group...
New Material Library...
Open Material Library...
Materials
Maps
Controllers
Autodesk Material Library
Scene Materials
Sample Slots
Show Incompatible
Show Empty Groups
Additional Options
mdl::alg::materials::blinn::blinn
mdl::alg::materials::lambert::lam...
mdl::alg::materials::lights::diffuse...
mdl::alg::materials::physically_m...
mdl::alg::materials::physically_sp...
mdl::SD::CONCRETE::CONCRETE...

AUTODESK UNIVERSITY

Iray for 3ds Max (MDL)



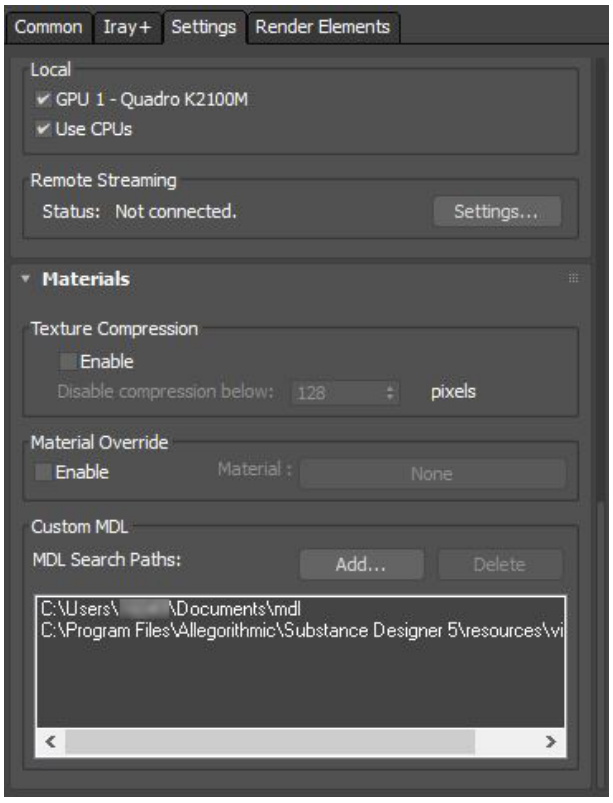
As stated earlier in the document, **MDL** is the **recommended workflow** when using **Iray for 3ds Max**. **MDLs** are completely native to **Iray**, and using **MDLs** from **Substance Designer** will result in the material looking **1:1 between Substance Designer and 3ds Max.**

NOTE: MDLs cannot have their bitmap Tile settings changed.

Here are the steps in order to use **MDL** with **Iray for 3ds Max**:

1. Set the current **Graph** to the **Target Resolution** for export
2. Make sure **Substance Designer's** rendering engine to **Iray**.
3. Under Materials, make sure the Shader is set to: **Materials > mat_checker > Definitions > mdl::agl::materials > physically_metallic_roughness**
4. Then go to **Materials > mat_checker > Edit**
5. Set all the additional settings in the **Properties Window** that are needed for the look and feel of the material
    a. This can include the **IOR, Emissive, Displacement, etc.**
6. When the **MDL** is ready to Export go to**: Materials > mat_checker > Export Preset…**
7. In the **Export Preset Window**, locate where the **MDL** should be exported
8. In the **Export Preset Window**, name the **MDL**
    a. **DO NOT USE** the **" – "** character in the naming of the **MDL**. It does not play well with **MDLs**, and will cause it not to export. There is no error dialog when this happens, so make sure to double check. Use the **" _ "** character in place of this.

9. Once exported, a **.mdl file** will be generated in the location that was specified, and bitmaps will be generated for every Output Node that is being associated with the **MDL**
10. With 3ds Max open, make sure **Iray+** is set as the current Rendering Engine.
11. Open the **Rendering Options**
12. Under the **Settings Tab**, scroll down to the **Custom MDL** section.
13. Click **Add…**
    a. Navigate to the following location: **C:\Program Files\Allegorithmic\Substance Designer 5\resources\view3d\iray**
    b. **THIS ONLY NEEDS TO BE DONE ONCE.** It allows Iray to read the MDLs generated with Substance Designer



14. Click **Add…** again
    a. Find the folder where the MDLs are stored on the computer or network. This should be a central repository, or library, for organization and easy access.
    b. T**HIS ONLY NEEDS TO BE DONE ONCE, but only if all the MDLs are in the same location.**
15. Open the **Material Editor**
16. Create a new **Iray+ Material**
17. Scroll to the **MDL** section at the bottom of the **Iray+ Material** settings
18. Click **Import…**

19. Select the **MDL** that needs to be loaded into this material from a location on the computer or network.
    a. If there is an error, double check to make sure that the correct file paths have been added to the **Iray+ Settings** in **Step 13 and 14.**
    b. If an error continues, download the **MDL Exchange Library**. This will add additional definitions to the installation, which should help.
        i. It's a good idea to download this regardless.
    c. The **MDL Exchange Library** is included with the **vMaterial** Install, which can be found **here.**
        i. Additional support can be found **here.**

AUTODESK UNIVERSITY

Resources and References:

Linear Workflows

Linear-Space Lighting (i.e. Gamma) – John Hable
http://filmicgames.com/archives/299

Gamma Correctionand Linear Colour Space – Maddieman
https://maddieman.wordpress.com/2009/06/23/gamma-correction-and-linear-colour-space-simplified/

X-Rite ColorChecker Passport Photo
http://xritephoto.com/colorchecker-passport/support

ColorChecker RGB Summaries – Bruce Lindbloom
http://www.brucelindbloom.com/index.html?ColorCheckerRGB.html

Material Basics

PBR Guidelines – Allegorithmic
https://www.allegorithmic.com/pbr-guide

Physically Based Rendering Encylopedia – Brian Yu
https://docs.google.com/document/d/1Fb9_KgCo0noxROKN4iT8ntTbx913e-t4Wc2nMRWPzNk/edit

Feeding a physically based shading model - Sébastien Lagarde
https://seblagarde.wordpress.com/2011/08/17/feeding-a-physical-based-lighting-mode/

DONTNOD specular and glossiness chart - Sébastien Lagarde
https://seblagarde.wordpress.com/2012/04/30/dontnod-specular-and-glossiness-chart/

DONTNOD Physically based rendering chart for Unreal Engine 4
https://seblagarde.wordpress.com/2014/04/14/dontnod-physically-based-rendering-chart-for-unreal-engine-4/

Physically-Based Shading at Disney – Brent Burley
http://disney-animation.s3.amazonaws.com/library/s2012_pbs_disney_brdf_notes_v2.pdf

Calibrating Lighting and Materials in Far Cry 3 – Stephen McAuley
http://blog.selfshadow.com/publications/s2012-shading-course/mcauley/s2012_pbs_farcry3_notes_v2.pdf

FXGuide - Game Environments – Part A: Rendering *Remember Me* – Mike Seymour
https://www.fxguide.com/featured/game-environments-parta-remember-me-rendering/

Refraction of Light – Rick Reed
http://interactagram.com/physics/optics/refraction/

Substance Designer

SUBSTANCE DESIGNER -TEXTURE CREATION WITH ROGELIO OLGUIN
https://www.thegnomonworkshop.com/tutorials/substance-designer

DEMYSTIFYING SUBSTANCE DESIGNER -AN ARTISTIC APPROACH WITH CHRISTOPHE DESSE
https://www.thegnomonworkshop.com/tutorials/demystifying-substance-designer

Kyle Horwood
https://gumroad.com/3dkyle

Jacob Norris
https://gumroad.com/purepolygons

Josh Lynch
https://gumroad.com/artofjoshlynch

Christophe Desse
https://gumroad.com/xtrm3d

Jacob Norris
https://gumroad.com/kazperstan

Rogelio Olguin
https://vimeo.com/user474658

Substance Days 2016
http://livestream.com/gnomon/substance-days-at-gnomon

Allegorithmic Youtube
https://www.youtube.com/user/Allegorithmic