PBR Guidelines for Sumerian

Confetti Interactive, Inc.

Overview

The aim of this document is to help users of the Sumerian Editor understand how to quickly author PBR materials that mimic the behaviour of their real-world counterparts. We'll introduce all the possible inputs that our PBR materials support, as well as pointing out some common values used to reproduce common materials, such as metals and plastics.

What is PBR?

PBR or Physically-Based Rendering is a set of rendering algorithms and techniques which are based on the same underlying physical principles that are involved when light bounces on a real-world surface. While not being a complete reproduction of real-world light phenomena, they closely matches its behaviour, which, in effect, results in realistic materials in a 3D scene. This makes PBR a great tool for development, as we can reproduce a wide array of different materials by just adjusting a few input for a given surface, which are:

* **Albedo:** defines the color of the surface if non-metallic, or base metallic reflectivity if metallic.
* **Roughness:** defines the microscopic irregularities on the surface and how light bounces from it.
* **Metallic:** defines how metallic a surface looks, which has to do with how much of the light reaching it gets absorbed.
* **Normal:** surface orientation.
* **Ambient Occlusion:** defines the amount of indirect lighting received on the surface.
* **Emission:** how much light is the surface radiating.

**Albedo**, **Roughness**, **Metallic** and **Normal** input are strictly necessary to be able to resolve the PBR rendering equation, and the user should **always** define them to have correct PBR results. On the other hand, **Ambient Occlusion** and **Emission Maps** (and also **Normal Maps** if the user wants to modify the surface orientation give by the used 3D model) are completely optional, and only add additional detail to PBR materials.

Albedo Map/Value

**Format:** 3-Channel (RGB) Texture Map **OR**

3-Channel (RGB) Color

The albedo property of a PBR material defines the color of its surface. It can be used as a single color for the whole surface, or loaded from a texture map, having each texel a different albedo value. It's important to note that, if a texture is loaded, it shouldn't contain any light information on it (slight shadowing, highlights, etc.), as this wouldn't suit the PBR formulas used in our shader, resulting in wrong results. Only surface color should be used for this texture.

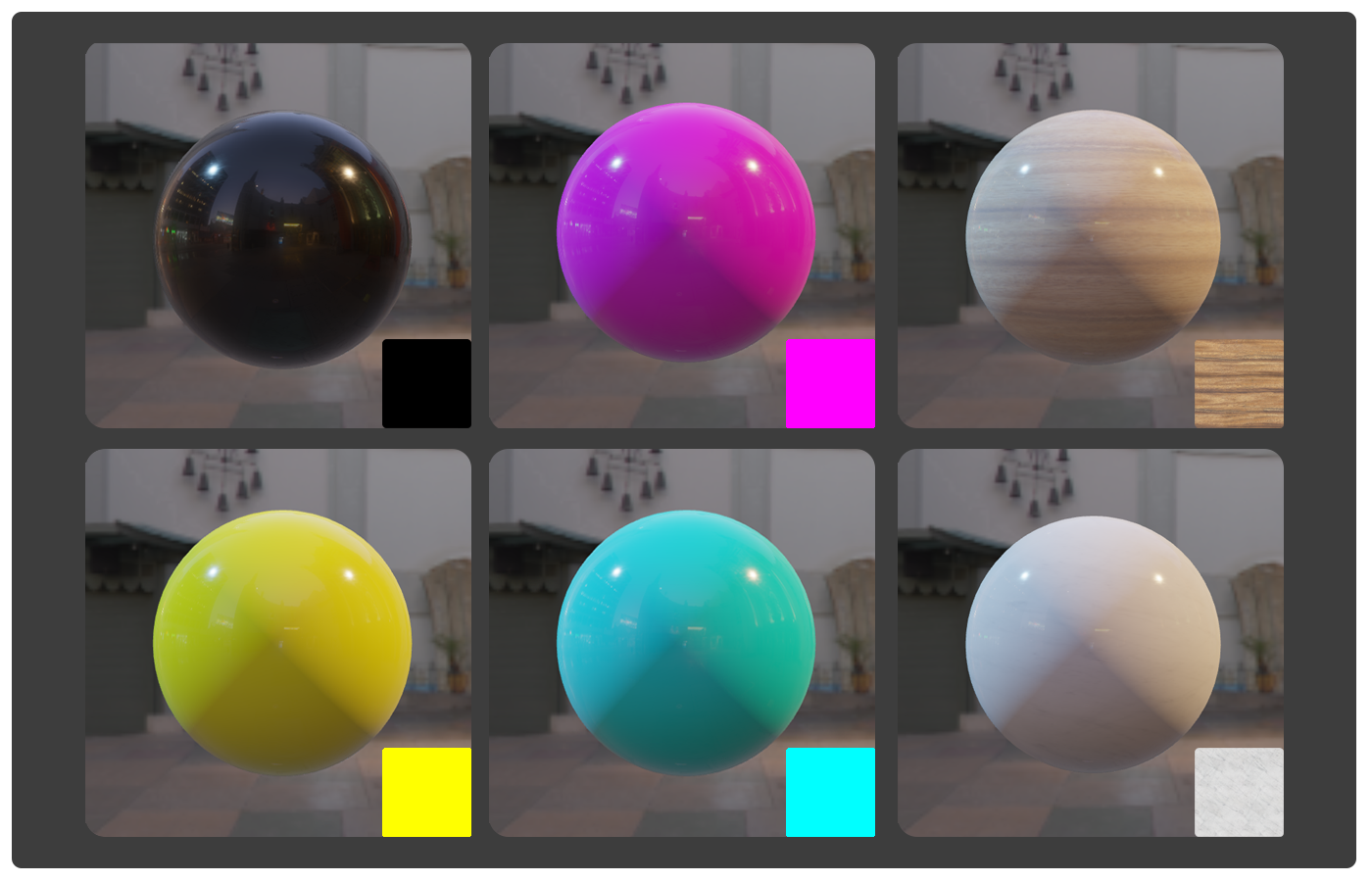


Figure 1: Comparison of different albedo inputs.

Another important detail regarding albedo is the fact that it behaves differently for metallic and nonmetallic surfaces. We follow what is known as a "Metallic PBR Workflow", which means that, instead of explicitly specifying the color of specular reflections (or what is known as the "Specular PBR Workflow"), this color is inferred from the incoming light and the surface's albedo depending on how metallic a surface is. Non-metallic surfaces will just bounce the environment's incoming light, but metallic surfaces will absorb part of this light, resulting in speculars tinted by the albedo color (also known as 'base metallic reflectivity' in this case).

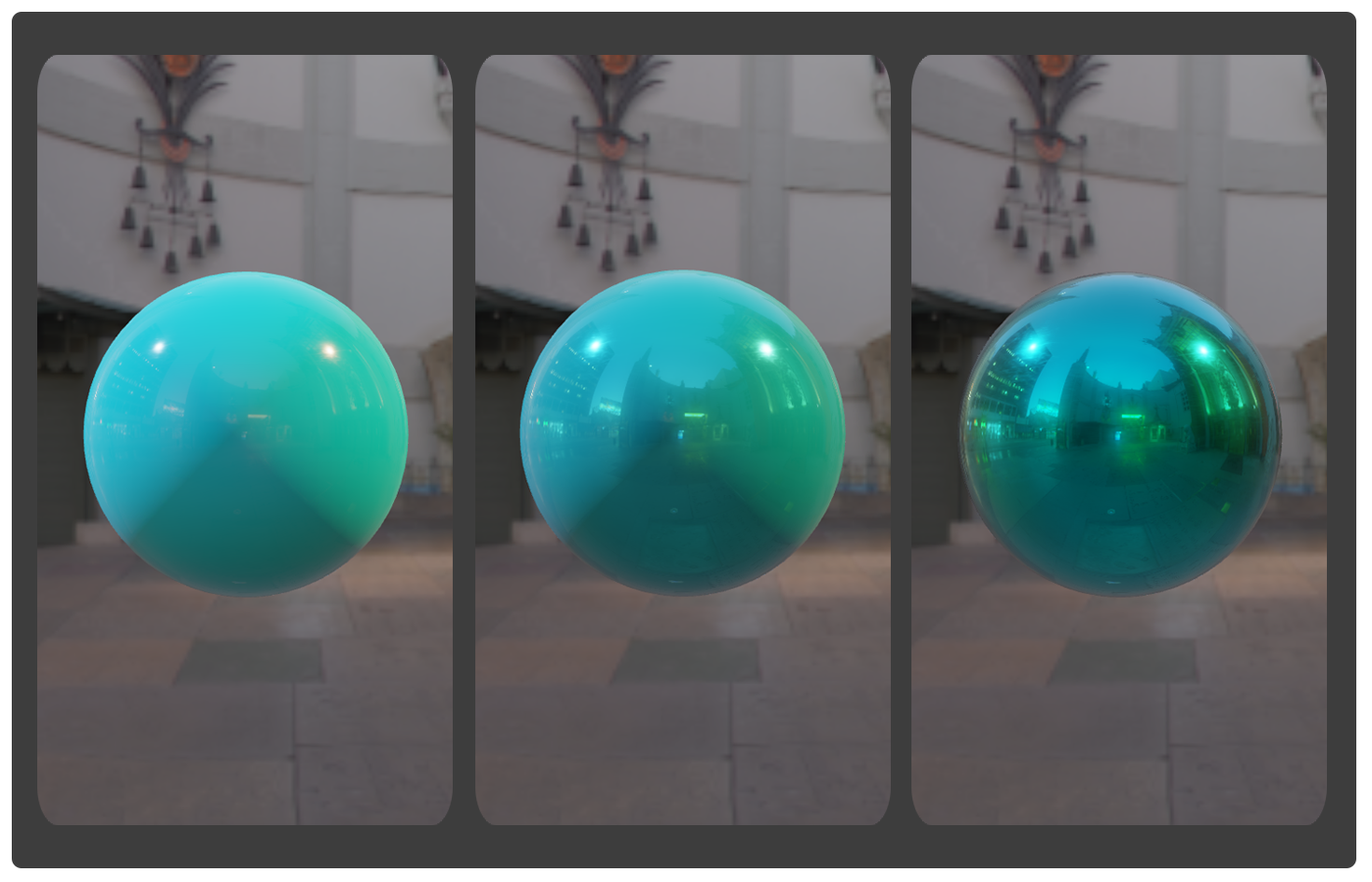


Figure 2: Comparison of different metallic (from left to right: 0.0, 0.5, 1.0) values using the same albedo input.

In the following table, you can observe common albedo values for different metallic and non-metallic materials which can help you reproduce the look you might be trying to achieve for a given object in a Sumerian scene.

|  |  |
| --- | --- |
| ***Material*** | ***Albedo (R,G,B)*** |
| **Charcoal** | (0.02,0.02,0.02) |
| **Fresh Asphalt** | (0.02,0.02,0.02) |
| **Green Grass** | (0.21,0.21,0.21) |
| **Plastic** | (0.24,0.24,0.24) |
| **Desert Sand** | (0.36,0.36,0.36) |
| **Fresh Snow** | (0.81,0.81,0.81) |
| **Iron** | (0.560, 0.570, 0.580) |
| **Silver** | (0.972, 0.960, 0.915) |
| **Aluminum** | (0.913, 0.921, 0.925) |
| **Gold** | (1.000, 0.766, 0.336) |
| **Copper** | (0.955, 0.637, 0.538) |
| **Nickel** | (0.660, 0.609, 0.526) |
| **Titanium** | (0.542, 0.497, 0.449) |

Table 1: List of different measured albedo values for real-world materials.

Metallic Map/Value

**Format:** Single-Channel (R) Texture Map **OR**

Floating-Point number (Range: [0...1])

The metallic input can be either a grayscale texture or a floating-point value that models how close to a metal a surface behaves. Pure metals would have a value of 1.0, pure non-metals (rock, plastic, etc) would have a value of 0.0, and hybrid materials would have a value in between 0.0 and 1.0.

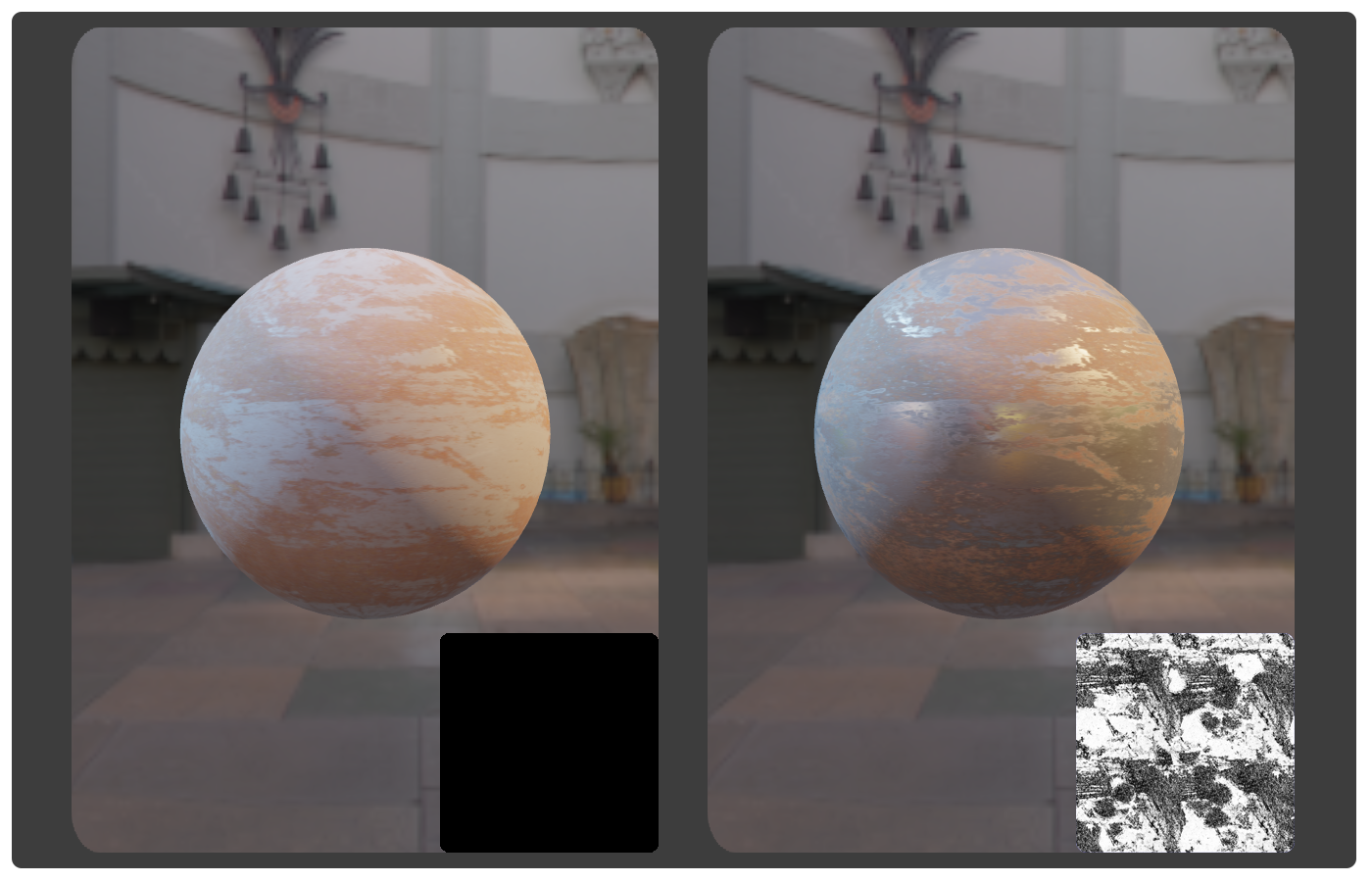


Figure 3: Comparison of standard metallic input (0.0) vs metallic texture map.

In a physical sense, what this really means is how much of the light hitting a surface escape from it. Metallic surfaces absorb all refracted light (light that, instead of bouncing on a surface, enters it, bounces around, and then leaves it), so only reflected light (light that just bounces on the surface) is emitted by them, while non-metallic emit both reflected and refracted light.

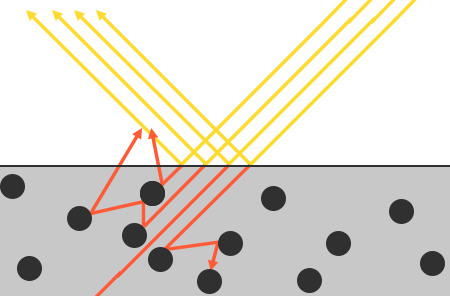


Figure 4: Diagram showcasing reflection (yellow rays) and refraction (orange rays).

Take into account that this texture shouldn't store the specular color of the surface, as we're following a 'Metallic PBR Workflow' (as explained on the albedo section, the specular color on metals is defined by the albedo texture map).

Roughness Map/Value

**Format:** Single-Channel (R) Texture Map **OR**

Floating-Point number (Range: [0...1])

The roughness input, as the metallic input, can be either a grayscale texture or a floating-point value. In this case, this parameter models how rough a surface is, which effectively changes how frequently the surface microfacets (the microscopic profile of the surface) change orientation across the surface. Rougher surfaces have wider and blurrier reflections, while smooth surfaces have focused and clear reflections.

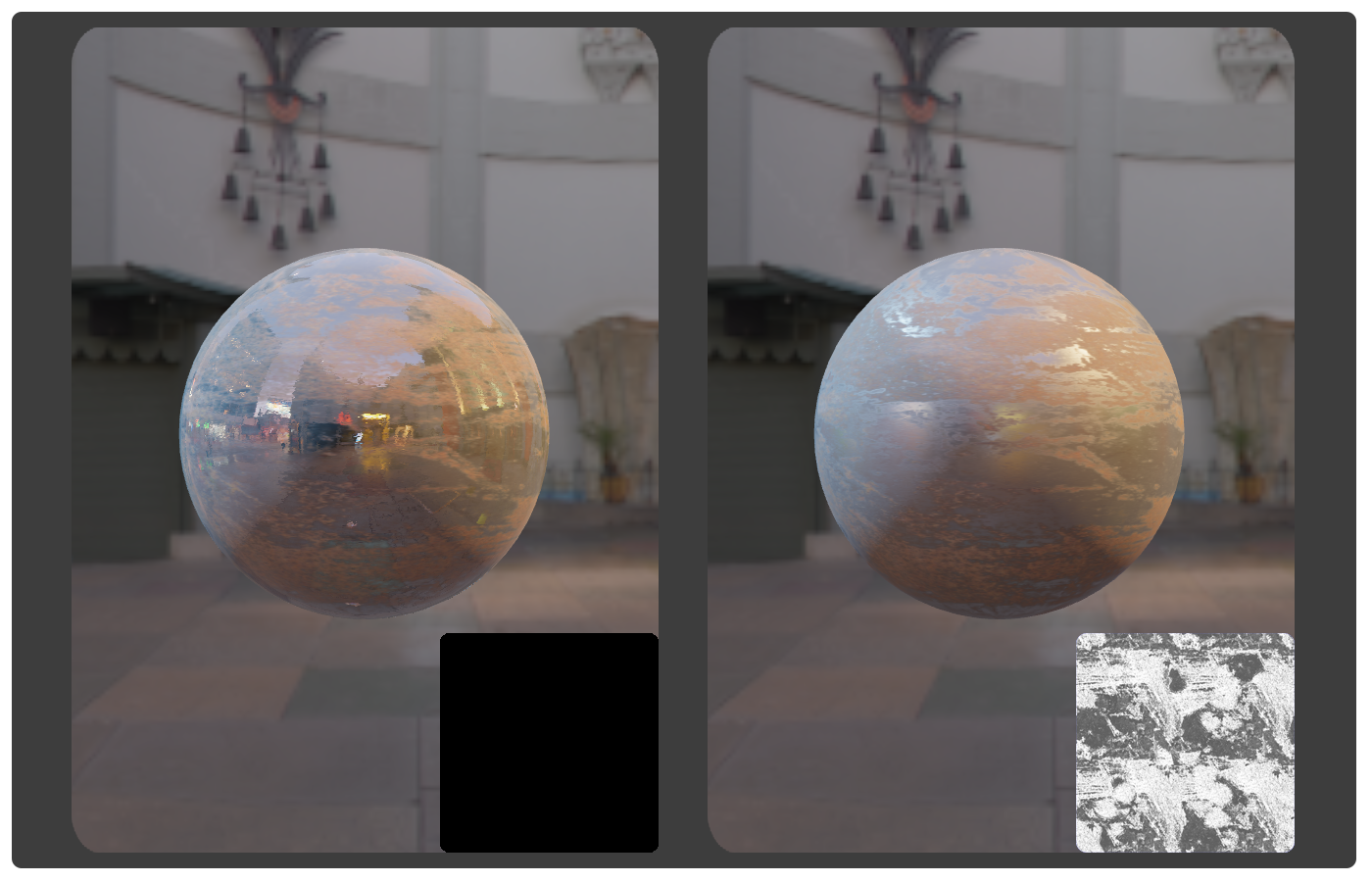


Figure 5: Comparison of standard roughness input (0.0) vs roughness texture map.

Normal Map [Optional]

**Format:** 3-Channel (RGB) Texture Map

The normal input of our PBR materials expects a texture map that defines how the surface orientation varies for each pixel on tangent-space (R-Channel stores the surface's tangent orientation, the G-Channel stores the surface's binormal orientation and the B-Channel stores the surface's normal orientation). This is pretty standard input for all 3D application, and it effectively modifies how the surface reacts to lighting.

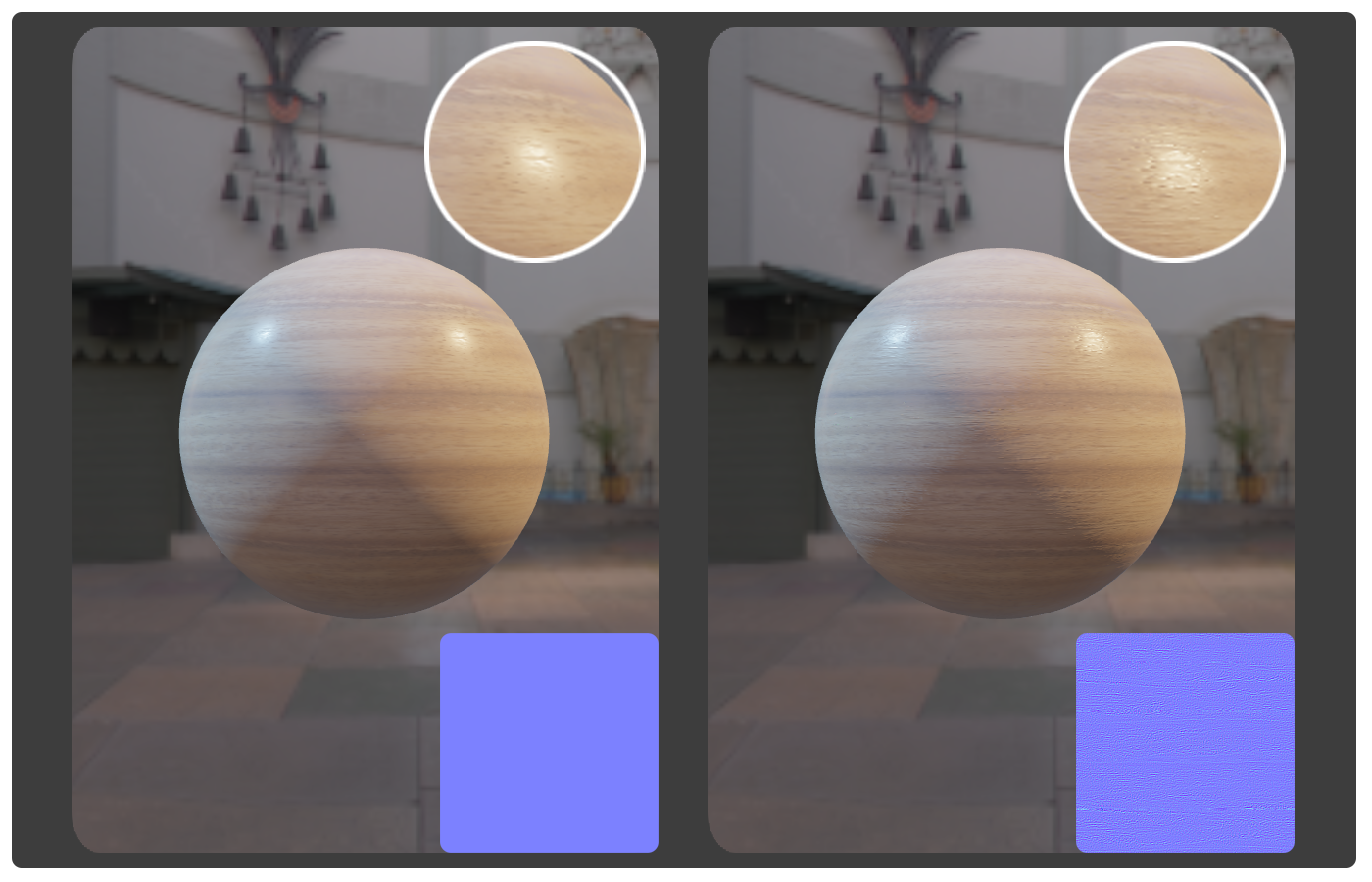


Figure 6: Comparison of flat normal map vs detailed normal map.

Ambient Occlusion Map [Optional]

**Format:** Single-Channel (R) Texture Map

The ambient occlusion input (or AO for short) is expected to be grayscale texture holding information on which areas of the surface should receive high or low indirect lighting. This causes extra shadowing to appear places such as small crevices, holes, etc, present on a material. This extra level of shadowing information can mark the difference between good and great visual quality in your materials, specially in those with high-frequency detail.

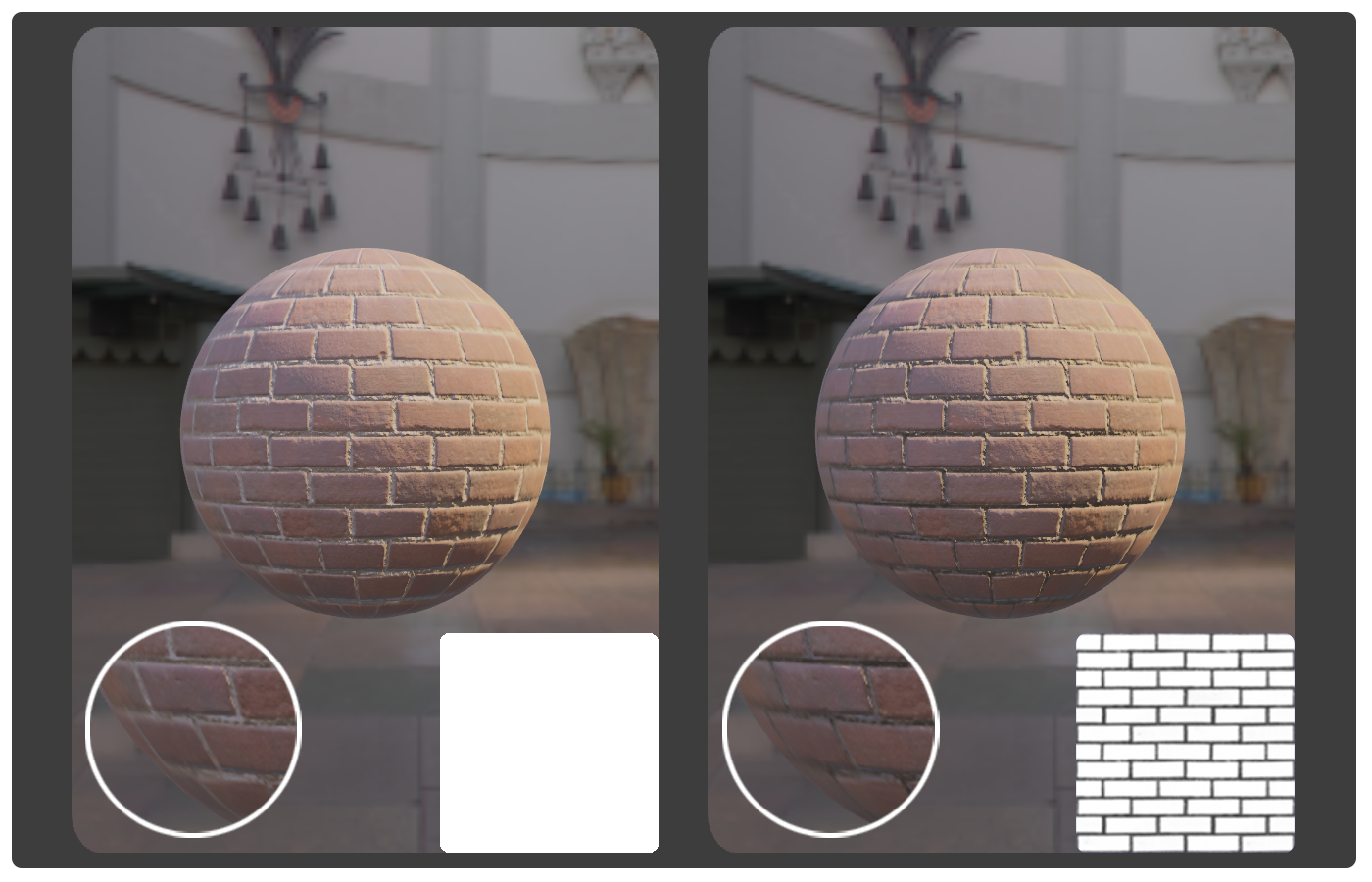


Figure 7: Comparison of no AO input vs AO texture map.

Emissive Map [Optional]

**Format:** 3-Channel (RGB) Texture Map **AND**

Floating-Point number (Range: [0...∞])

The emissive input is a combination of both an RGB texture and a multiplier factor (specified by a floating-point value) that are aimed to model light-emitting materials. Despite breaking one of the physical principles behind PBR (energy conservation, or how a surface cannot emit more light than it receives), is common on most PBR implementations, as it allow us to easily reproduce materials such as light-bulbs, lava, neon-bars, and other glowing materials within the same shading model.



Figure 8: Comparison of non-emissive input vs emissive texture map (and emissive multiplier of 2.0).