Bachelor Project

Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of Computer Graphics and Interaction

Materials appearance in Unity

Jakub Kyselka

Supervisor: doc. Ing. Jiří Bittner, Ph.D.
Field of study: Open Informatics
Subfield: Computer Games and Graphics
May 2022

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Kyselka** Jméno: **Jakub** Osobní číslo: **492053**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačové grafiky a interakce**

Studijní program: **Otevřená informatika**

Specializace: **Počítačové hry a grafika**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Vzhled materiálů v Unity

Název bakalářské práce anglicky:

Materials Appearance in Unity

Pokyny pro vypracování:

Zmapujte problematiku simulace vzhledu materiálů se zaměřením na fyzikálně založené modely. Popište podporu vzhledu materiálů v enginu Unity v zobrazovacích řetězcích BIRP, URP a HDRP.
Vytvořte vzorkovník základních často používaných materiálů (barevné plasty, různé typy dřeva, měď, zlato, stříbro, hliník, sklo, diamant, apod.). Vytvořte demonstrační aplikaci, která bude názorně ukazovat různé reálné materiály a rozdíly jejich vzhledu v různých zobrazovacích řetězcích.
Vymodelujte scénu, kde bude možné materiály ze vzorkovníku jednoduše přepínat a umožní studovat a měnit parametry materiálu. Vymodelujte rovněž komponovanou scénu, kde budou najednou využity všechny materiály ze vzorkovníku.
Realizujte základní výkonnostní test, který vyhodnotí rychlost zobrazování komponované scény v různých zobrazovacích řetězcích na nejméně dvou různých hardwarových konfiguracích.

Seznam doporučené literatury:

[1] Burley, Brent, and Walt Disney Animation Studios. Physically-based shading at Disney. ACM SIGGRAPH. Vol. 2012. vol. 2012, 2012.
[2] Schmidt, T. W., Pellacini, F., Nowrouzezahrai, D., Jarosz, W., & Dachsbacher, C.. State of the art in artistic editing of appearance, lighting and material. In Computer Graphics Forum (Vol. 35, No. 1, pp. 216-233), 2016.
[3] Heitz, Eric. "Understanding the masking-shadowing function in microfacet-based BRDFs." Journal of Computer Graphics Techniques 3.2 (2014): 32-91.
[4] Serrano, A., Gutierrez, D., Myszkowski, K., Seidel, H. P., & Masia, B. An intuitive control space for material appearance. arXiv preprint arXiv:1806.04950, 2018.
[5] Materials in Unity. Online: https://docs.unity3d.com/Manual/Materials.html
[6] Tomáš Cicvárek. Materials in Computer Graphics. Bakalářská práce, ČVUT FEL, 2021.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

doc. Ing. Jiří Bittner, Ph.D. **Katedra počítačové grafiky a interakce**

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **09.02.2022** Termín odevzdání bakalářské práce: **20.05.2022**

Platnost zadání bakalářské práce: **30.09.2023**

doc. Ing. Jiří Bittner, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. _____
Datum převzetí zadání

_____
Podpis studenta

Acknowledgements

I would like to thank doc. Ing. Jiří Bittner, Ph.D. for his supervision and my family and friends for their support.

Declaration

I declare that this thesis was developed independently and that I have cited all used sources properly. In Prague, 19. May 2022

Abstract

The main goal of this thesis is to describe the main BRDF functions, empirical and physical, used for calculating the amount of reflected light in materials and compare Unity's main graphical pipelines (Default, HDRP, and URP) and their main shaders. Part of this thesis is also an implementation of an application for the purpose of comparing Unity's pipelines and their shaders.

Keywords: materials, Unity, pipeline, shader, HDRP, URP

Supervisor: doc. Ing. Jiří Bittner, Ph.D.
Praha 2,
Karlovo náměstí 13,
E-421

Abstrakt

Hlavním cílem této práce je popsat základní BRDF funkce, empirické a fyzikální, používáné pro výpočet množství odraženého světla v materiálech a porovnat hlavní grafické pipeliny herního enginu Unity (Default, HDRP, URP) a jejich hlavní shadery. Součástí práce je také implementace aplikace v Unity pro porovnávání Unity pipeline a jejich shaderů.

Klíčová slova: materiály, Unity, pipeline, shader, HDRP, URP

Překlad názvu: Vzhled materiálů v Unity

Contents

Project Specification iii

1 Introduction 1

2 Materials and illumination models 3
2.1 Bidirectional reflectance function - BRDF 3
2.2 Local illumination model 4
2.3 Empirical illumination models . . . 4
2.3.1 Lambertian reflection model . . 4
2.3.2 Phong illumination model 5
2.3.3 Blinn-Phong illumination model 6
2.4 Physically based illumination models 7
2.4.1 Microfacet-based BRDFs 8

3 Unity and its render pipelines 11
3.1 Render pipeline 11
3.1.1 Shader 12
3.2 Default pipeline 12
3.2.1 Lighting in default pipeline . . 12
3.2.2 Shaders in the default pipeline 17
3.3 URP - Universal render pipeline 22
3.3.1 Lighting in URP 22
3.3.2 Materials in URP 22
3.4 HDRP High-definition render pipeline 23
3.4.1 Lighting in HDRP 24
3.4.2 Shaders in HDRP 26

4 Implementation 29
4.1 Default scene 30
4.1.1 Scene composition 30
4.2 Composite scene 32
4.3 Saving, loading, and pipeline switching 33
4.3.1 Shader properties 33
4.3.2 Shader keywords 34
4.3.3 Complications 35
4.4 Image view and image comparison modes 37

5 Comparison/analysis of pipelines illumination models 39
5.1 Empirical appearance analysis . . 39
5.2 Performance comparison 41

6 Conclusion 45

A Contents of electronic appendix 47

B User manual 49

Bibliography 51

Image sources 53

Figures

2.1 Bidirectional reflectance function model [1]. 3
2.2 Vectors used in Phong and Blinn-Phong illumination model [2]. 5
2.3 Components of Phong illumination model [3]. 6
2.4 Comparison of the Phong and Blinn-Phong illumination model [4]. 7
2.5 Directions of the vectors of reflected light becoming more uniform as the microfacet based surface gets smoother [10]. 8
2.6 On the left - light vectors unable to reach certain microfacets - shadowing. On the right - certain vectors of reflected light are blocked by other microfacets, and thus they cannot be seen - masking [11]. 9

3.1 An example of a rendering pipeline with parts that can be modified by shader highlighted [5]. 12
3.2 Area light and its range of influence [9]. 14
3.3 Three area lights illuminating a sphere showcasing the soft shading they produce [9]. 14
3.4 Properties in the light inspector. 16
3.5 Color computation in a BRDF function of the Standard shader, diffuse part in the first row, specular part in the second row, and global illumination in the third row. Functions or variables where metallic parameter contributes are highlighted blue (fresnel function, diff color), and where smoothness contributes are highlighted green. 18
3.6 Smoothness 0 on the left, smoothness 1 on the right (metallic 0). 19
3.7 Metallic 0 on the left, Metallic 1 on the right (Smoothness 0). 19
3.8 Image showcasing how the changes of metallic parameter influence the material appearance in the Standard shader. Smoothness = 0.8 [10]. . . . 20
3.9 Image showcasing the changes in specular reflections as the smoothness parameter changes in the Standard shader [10]. 20
3.10 Specular mode (left) of the Standard shader and metallic mode of the Standard shader (right) [6][7]. 21
3.11 Properties in the URP light inspector. 23
3.12 Material properties in URP Lit shader [8]. 24
3.13 Properties in the HDRP light inspector (without volumetric and shadow components). 26
3.14 Material properties in HDRP Lit shader (without emission, detail map, and advance options as they are very similar to the ones in the URP/Standard shader) [8]. 27

4.1 Material (left) and light (right) sliders. 30
4.2 Default composition in Default rendering pipeline. 31
4.3 Mitsuba composition in Default rendering pipeline. 31
4.4 Medusa composition in Default rendering pipeline. 32
4.5 Spikes composition in Default rendering pipeline. 32
4.6 View of the composite scene from camera 1. 33
4.7 View of the composite scene from camera 2 (without UI). 35
4.8 View of the composite scene from camera 3 (without UI). 36
4.9 View of the composite scene from camera 4 (without UI). 36
4.10 Comparison mode. 37
4.11 Image view mode. 37

5.1 Medusa model - Albedo: R-255, G-109, B-109, Smoothness 0 and Metallic 0, HDRP-Lit (left) , URP-Lit (middle), Standard shader (right). 39

Tables

5.2 Medusa model - Albedo: R-255, G-109, B-109, Smoothness 1 and Metallic 0, HDRP-Lit (left), URP-Lit (middle), Standard shader (right).. 40
5.3 Medusa model - Albedo: R-255, G-109, B-109, Smoothness 0 and Metallic 1, HDRP-Lit (left), URP-Lit (middle), Standard shader (right).. 40
5.4 Medusa model - Albedo: R-255, G-109, B-109, Smoothness 1 and Metallic 1, HDRP-Lit (left), URP-Lit (middle), Standard shader (right).. 41

4.1 Table of used assets. 29
4.2 Table of Standard shader properties and their equivalents in URP-Lit shader and HDRP-Lit shader that are converted when switching to a different pipeline or while loading/saving. 34

5.1 Table of average FPS (cumulative moving average) in various pipelines - PC 1. 42
5.2 Table of average FPS (exponential moving average) in various pipelines - PC 1. 42
5.3 Table of average FPS (cumulative moving average) in various pipelines - PC 2. 42
5.4 Table of average FPS (exponential moving average) in various pipelines - PC 2. 42

Chapter 1

Introduction

The believability of material appearance in computer graphics is crucial for creating real-looking environments that could be used in educational simulations, games, or movies. One such aspect which makes the material look believable is a proper simulation of light's behavior when interacting with the material. The function used for describing the behavior of the light on opaque materials is called the bi-directional reflectance function or BRDF.

BRDFs were first derived empirically and were widely used for their simplicity and low-performance cost when implemented while providing an acceptable appearance. Later physically-based BRDFs were developed, most of them based on microfacet theory. These physically-based BRDFs are now implemented in most rendering software or game engines like Unity or Unreal Engine.

Unity is a free game engine made for the development of both 3D and 2D games. It offers three pre-made graphical pipelines with differing features and capabilities. However, it is difficult to readily compare them in terms of graphical appearance as they all offer a multitude of shaders, each with different properties and purposes.

The purpose of this thesis is to describe the features and properties of the three main graphical pipelines (Default, URP, and HDRP) and develop an application that would make it easier to compare said pipelines.

Chapter 2

Materials and illumination models

Materials in computer graphics have a great impact on the appearance of three-dimensional objects, making them look more realistic or stylistically interesting. In applications such as Unity or Blender, materials are defined by various properties such as color, metallicity, smoothness or textures, and maps (bump, normal, height) [2, 3]. How each property of the material influences (or is influenced by) the variables of the illumination models depends on the illumination model used and its implementation.

2.1 Bidirectional reflectance function - BRDF

Bidirection reflectance functions defines the amount of reflected radiance on a certain point of an object.

BRDF operates according to a set of preconditions [1]:

- The reflection of the light ray is instantaneous
- A photon with a wave length of λ retains it is wave length upon reflection.
- A light ray arriving at a certain point x will also reflect from the same point x.

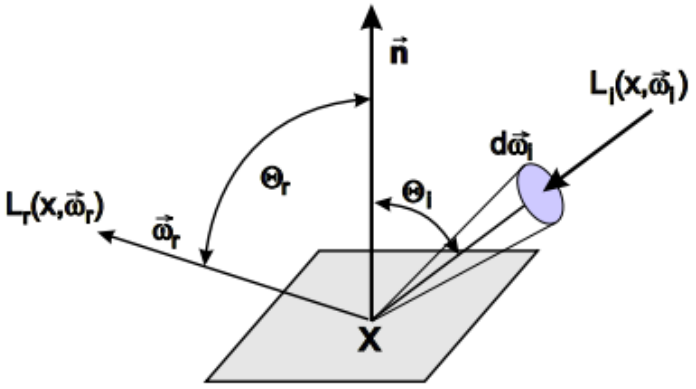

Figure 2.1: Bidirectional reflectance function model [1].

Let $\overrightarrow{\omega_i}$ be the direction of the incoming light, $\overrightarrow{\omega_r}$ the direction of the reflected light and x the point of reflection. BRDF is defined as a ratio of differential reflected radiance $d\mathbf{L}_r(x, \overrightarrow{\omega_r})$ to the differential incoming radiance $d\mathbf{L}_i(x, \overrightarrow{\omega_i})$ [1] :

$$\mathbf{f}_r(x, \overrightarrow{\omega_r}, \overrightarrow{\omega_i}) = \frac{d\mathbf{L}_r(x, \overrightarrow{\omega_r})}{d\mathbf{L}_i(x, \overrightarrow{\omega_i})(\overrightarrow{\omega_i} \cdot \overrightarrow{n})d\overrightarrow{\omega_i}}$$

2.2 Local illumination model

The local illumination model calculates the amount of radiance reflected off of a point on the object, taking into account all of the incoming radiance coming from all sources of light. In comparison with the global illumination model, it doesn't take into account the reflected light from other surfaces, only light from direct sources. Let $\overrightarrow{\omega_i}$ be the direction of the incoming light, $\overrightarrow{\omega_r}$ the direction of the reflected light and x the point of reflection, $\mathbf{f}_r(x, \overrightarrow{\omega_r}, \overrightarrow{\omega_i})$ be the BRDF and $\mathbf{L}_i(x, \overrightarrow{\omega_i})$ be the incoming radiance. The amount of reflected radiance $\mathbf{L}_r(x, \overrightarrow{\omega_r})$ is then defined as [1]:

$$\mathbf{L}_r(x, \overrightarrow{\omega_r}) = \int_\Omega \mathbf{f}_r(x, \overrightarrow{\omega_r}, \overrightarrow{\omega_i})\mathbf{L}_i(x, \overrightarrow{\omega_i})cos\theta d\overrightarrow{\omega_i}$$

2.3 Empirical illumination models

Empirical illumination models are based on empirical observations and approximations rather than being physically plausible. Empirical illumination models usually have a lesser impact on performance due to their simpler computational requirements. Some empirical models don't adhere to the preconditions outlined in chapter 2.1, which can result in an unrealistic appearance. If, for example, the law of energy conservation is not obeyed, it can reflect more energy than it received, resulting in excessive illumination [1, 4].

2.3.1 Lambertian reflection model

Lambertian reflection model only allows for physically plausible diffuse reflection omitting specular reflections of any kind entirely. Most other BRDFs use the Lambertian reflection model for the diffuse component or modified version of it [4]. In the Lambertian reflection model, the incoming light ray is scattered equally in all directions. The amount of reflected light is independent of the viewing angle and only dependent on the angle between the incoming light and the normal of the surface. Let $\mathbf{I}_d$ be the amount of reflected light, $\mathbf{I}_L$ represents the color of the incoming ray, $\mathbf{r}_d$ the color of the surface, $\overrightarrow{l}$ the light vector, $\overrightarrow{n}$ the normal of the surface. The amount of reflected light is computed as [1]:

$$\mathbf{I}_d = \mathbf{I}_L\mathbf{r}_d(\overrightarrow{l} \cdot \overrightarrow{n})$$

For the purpose of defining Lambertian BRDF, we also need to define reflectance. Since the range of the BRDF is not limited from above, which can be impractical, we can substitute the reflectance for BRDF. Reflectance ρ is limited to the values in range <0,1> and is defined as a ratio of the reflected radiosity at a point x - $d\mathbf{\Phi}_r(x)$ to the incoming radiosity at the point x - $d\mathbf{\Phi}_r(i)$:

$$\rho = \frac{d\mathbf{\Phi}_r(x)}{d\mathbf{\Phi}_i(x)}$$

The lambertian BRDF is then defined as reflectance divided by π [1]:

$$\mathbf{f}_d(x) = \frac{\rho_d}{\pi}$$

2.3.2 Phong illumination model

The Phong illumination model is an empirical illumination model widely used in many applications [4]. The amount of reflected lights depends on the viewing vector $\overrightarrow{v}$, normal vector at the point of reflection $\overrightarrow{n}$, light vector $\overrightarrow{l}$ and the reflection vector $\overrightarrow{r}$. The model recognizes three types of light [1]:

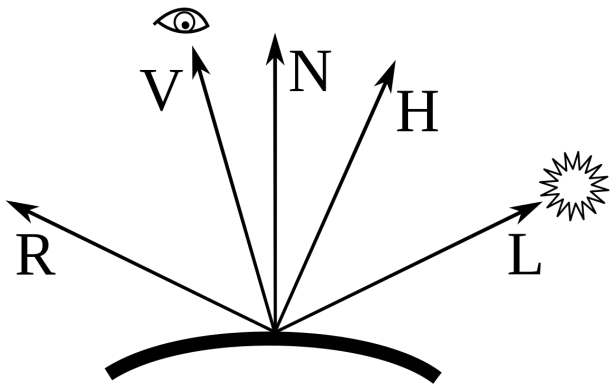

Figure 2.2: Vectors used in Phong and Blinn-Phong illumination model [2].

- Diffuse
- Specular
- Ambient

Diffuse component of Phong illumination model

The phong illumination model uses the Lambertian reflection model as its diffuse component [1, 5]:

$$\mathbf{I}_d = \mathbf{I}_L \mathbf{r}_d(\overrightarrow{l}.\overrightarrow{n})$$

Specular component of Phong illumination model

The specular component is similar to the diffuse one, but instead of depending on the cosine of the angle between the incoming light and the normal, it depends on the cosine of the angle between the reflected vector and the viewing vector powered to the order of coefficient h. Coefficient h dictates the sharpness of the mirror reflection. The higher the h, the smaller and more precise the specular reflection will be. Let $\overrightarrow{v}$ be the viewing vector, $\overrightarrow{r}$ the reflection vector, $\mathbf{I}_L$ the color of the light vector ,$\mathbf{r}_s$ the color of the surface and h the coefficient with its values in range <1,∞>. Computation [1, 5]:

$$\mathbf{I}_s = \mathbf{I}_L \mathbf{r}_s (\overrightarrow{v}.\overrightarrow{r})^h$$

Ambient component of Phong illumination model

The ambient component is light coming from an unspecified source at every point of the model. The amount of luminance is thus independent of any angle. Let $\mathbf{I}_A$ the color of the ambient light vector and $\mathbf{r}_s$ the color of the surface, the resulting vector is computed as [1, 5]:

$$\mathbf{I}_a = \mathbf{I}_A \mathbf{r}_s$$

Computation of the Phong illumination model

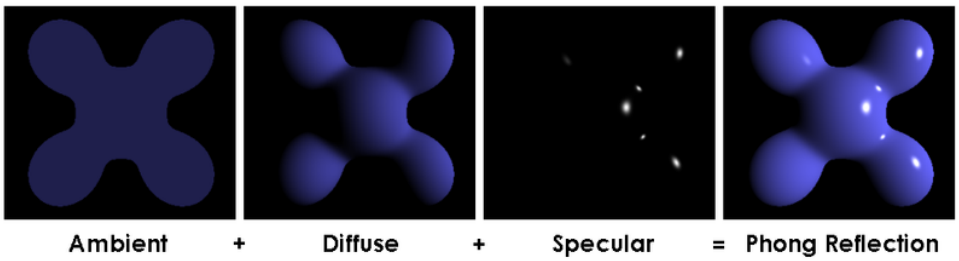

Figure 2.3: Components of Phong illumination model [3].

Let $\mathbf{I}_d$ be the diffuse component, $\mathbf{I}_s$ the specular component and $\mathbf{I}_a$ the ambient component. The combination of the three components give us the resulting luminance [1]:

$$\mathbf{I}_v = \mathbf{I}_s + \mathbf{I}_a + \mathbf{I}_d$$

2.3.3 Blinn-Phong illumination model

Blinn-Phong modifies and improves the Phong illumination model by replacing the computation of an angle between the reflection vector and the viewing vector and instead uses an angle between the half vector and normal vector. This results performance-wise in fewer calculations and graphically makes for a more physically accurate appearance. Half vector is defined as the

normalized sum of the light vector l and viewing vector v [6]:

$$h = \frac{l + v}{len(l + v)}$$

The modified specular component:

$$\mathbf{I}_s = \mathbf{I}_L \mathbf{r}_s (\overrightarrow{n} . \overrightarrow{h})^h$$

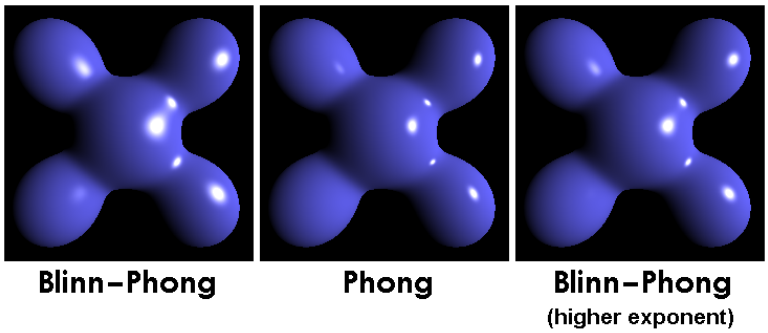

Figure 2.4: Comparison of the Phong and Blinn-Phong illumination model [4].

2.4 Physically based illumination models

For a BRDF to be physically plausible, it needs to adhere to these five preconditions [1]:

- Helmholtz reciprocity principle - This principle is states that at a given point the value of a BRDF stays the same if we interchange the reflection vector with the light vector:

$$\mathbf{f}_r(x, \overrightarrow{\omega_r}, \overrightarrow{\omega_i}) = \mathbf{f}_r(x, \overrightarrow{\omega_i}, \overrightarrow{\omega_r})$$

- Positivity - the value of BRDF is always bigger than or equal to zero.
- Anisotropy - the resulting value of the BRDF is also influenced by the rotation of the object the. light is reflecting on. This precondition is often omitten in popular physically based BRDFs [1].
- Law of energy conservation - Law of energy conservation in the context of BRDFs means that the amount of energy reflected will always be less than the energy recieved:

$$\int_\Omega \mathbf{f}_r(x, \overrightarrow{\omega_r}, \overrightarrow{\omega_i}) cos\theta_i d\overrightarrow{\omega_i} \leq 1$$

- Linearity states that the BRDF function of a light vector at a certain angle is not impacted by BRDF functions of light vectors at a different angle.

2.4.1 Microfacet-based BRDFs

Microfacet-based BRDF assumes the existence of tiny microfacets that comprise the surface of an object, forming a random set of V-shaped concavities, masking and shadowing each other, affecting the direction and visibility of the reflected vector. Microfacet-based BRDFs are generally defined as Lambertian diffuse + microfacet-based specular component [7, 1]. Torrance-Sparrow BRDF and Cook-Torrance BRDF are popular examples of a microfacet-based reflection function. Let $\mathbf{C}_d$ be the Lambertian diffuse function $\frac{\rho_d}{\pi}$, F the Fresnel function, D the distribution function and G the geometric function, $\overrightarrow{\omega_i}$ be the direction of the incoming light, $\overrightarrow{\omega_r}$ be the direction of the reflected light and ω_h cosine of the angle between the half vector and the normal vector. Cook-Torrance is defined as follows [4]:

$$\mathbf{f}_r(\overrightarrow{\omega_r}, \overrightarrow{\omega_i}) = \mathbf{C}_d + \frac{F}{\pi} \frac{D(\omega_h)G(\overrightarrow{\omega_r}, \overrightarrow{\omega_i})}{cos(\overrightarrow{\omega_r})cos(\overrightarrow{\omega_i})}$$

- D (ω_h) - microfacet distribution function which defines the amount of microfacets that are oriented in the direction of the half vector. Parameter m defines the roughness of the material, if the value of m is close to 0 then most microfacets are angled generally in the direction of the macronormal which is the normal of the idealized smooth surface, if, on the other hand, the value of m is close to 1 it results in an almost diffuse appearance as the microfacets are oriented in different directions scattering the reflected light. It is common to use either Gaussian distribution function or the Beckmann distribution function [4][18]. Let m be the roughness parameter, ω_h be the cosine of the angle between the normal of the macrosurface and the half vector of the microfacets. The Beckmann distribution function is:

$$D(\omega_h) = \frac{1}{m^2\omega_h^4} exp(\frac{\omega_h^2 - 1}{m^2\omega_h^2})$$

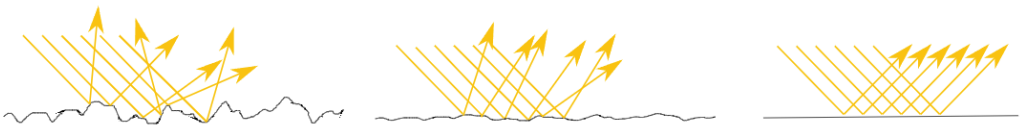

Figure 2.5: Directions of the vectors of reflected light becoming more uniform as the microfacet based surface gets smoother [10].

- F - Fresnel function increases the amount of light reflected at grazing angles, making it so the surface becomes more reflective if viewed at grazing angles (close to the surface or viewed from a great enough distance). One of the differences between Torrance-Sparrow and Cook-Torrance BRDF is that Cook-Torrance uses a more optimized version of the Fresnel function. Let κ be the absorption coefficient of the surface,

η be the index of refraction, u be the cosine of the angle between the light vector and the half vector, then the fresnel factor can be computed as follows:

$$F(\omega_r) = \frac{1}{2}\frac{(b-u)^2}{(b+u)^2}\{1 + \frac{[u(b+u)-1]^2}{[u(b-u)+1]^2}\}$$

where

$$b^2 = \eta^2 + u^2 - 1$$

It is common to use Schlick's approximation for the computation. Let u be the cosine of the angle between the light vector and the half vector, $\mathbf{f}_\lambda$ be the spectral distribution of the fresnel factor. The Schlick's fresnel approximation is defined as follows [4][11]:

$$F(\omega_r) = \mathbf{f}_\lambda + (1 - \mathbf{f}_\lambda)(1-u)^5$$

- G($\overrightarrow{\omega_r}, \overrightarrow{\omega_i}$) - Geometric function generally describes the amount of microfacets that will not be occluded by masking or shadowing. Let n be the normal vector, h the half vector then the geometric function can be computed as follows:

$$G(\overrightarrow{\omega_r}, \overrightarrow{\omega_i}) = min\{1, \frac{2(n\cdot h)(n\cdot \overrightarrow{\omega_r})}{(\overrightarrow{\omega_r}\cdot h)}, \frac{2(n\cdot h)(n\cdot \overrightarrow{\omega_i})}{(\overrightarrow{\omega_r}\cdot h)}\}$$

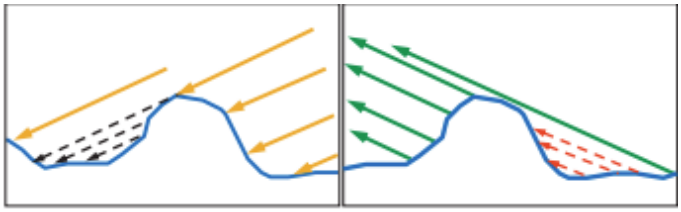

Figure 2.6: On the left - light vectors unable to reach certain microfacets - shadowing. On the right - certain vectors of reflected light are blocked by other microfacets, and thus they cannot be seen - masking [11].

Chapter 3

Unity and its render pipelines

Unity is a game engine that provides its user's lot of options and customizability when it comes to rendering and appearance. Users can choose between multiple pipelines (URP, HDRP, or default) or create a custom one. The same applies to shaders. Each of the three pipelines has different capabilities, and it depends on the user and a multitude of factors (platform, performance, familiarity, quality of graphics) which one should be used. For example, URP or HDRP are more scalable and customizable but are more demanding when it comes to performance and usability. Unity documentation divides the render pipeline into three stages [9]:

- Culling - culling objects that are not visible (occluded by other objects or are not in the viewing frustum).
- Rendering - drawing of the objects, light calculations.
- Post-processing - applying additional graphical effects - depth of field, bloom, or color grading.

3.1 Render pipeline

Render pipeline is a conceptual model that describes the process of putting 3D defined models on a 2D screen. As an input, it receives the geometry of the object, typically in the form of an array of triangles or similar simple geometric objects. The pipeline is generally defined in this order [8]:

- Camera projection - Model and camera transformation of the object - determining the position of the object relative to the camera.
- Viewing frustum clipping - Evaluating the visibility of the object from the camera. If it is not in the viewing frustum, it gets clipped.
- Projection - Perspective (far away objects get diminished in size) or orthogonal transformation.
- Viewport transformation - Transformation to the area of the screen.
- Local lighting - Calculating the color affected by lighting.

- Rasterization - Converting the object into rasterized form (the geometric primitives in the form of vertices get filled with pixels).
- Texture mapping - Mapping the 2d image (texture) onto each pixel.
- Pixel visibility checking - Determining which pixels of each object will be visible from the viewpoint of the camera. Calculation often involves a z-buffer - determining if a certain pixel is occluded by another.

This rendering pipeline is a simplification, and it is not always used in this exact form (for example, local lighting and rasterization could be exchanged and that would enable calculating the lighting more precisely per fragment) [8].

3.1.1 Shader

Shaders are closely connected to the graphical pipelines - they can be used for changing the way local lighting is calculated (BRDF) - creating lighting types (for example, point or directional), or defining the shading and color of the object. Shaders are often used for manipulating vertices (Vertex shader) and fragments (Fragment shader), but they can also be used to modify other parts of the pipeline. Shaders, as well as the whole graphical pipeline, are tied to the graphics card. To access them, one can make use of many available graphical APIs like Vulcan or OpenGL.

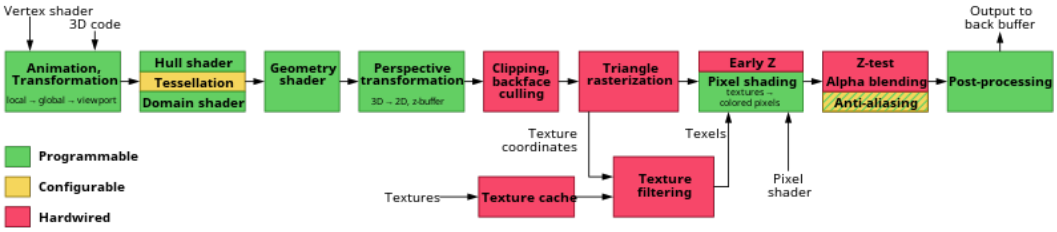

Figure 3.1: An example of a rendering pipeline with parts that can be modified by shader highlighted [5].

3.2 Default pipeline

Built-in render pipeline is the default pipeline used in Unity for rendering 3D spaces.

3.2.1 Lighting in default pipeline

Rendering paths

The default render pipeline provides a choice of a rendering path. It can be assigned in quality settings and can be overridden with each camera [9]:

- Forward rendering renders each object in one or more passes, depending on the number of lights that affect the objects [10]. The way in which light is calculated depends on the intensity and distance of the light, or they can be tagged 'Not important' (will be rendered per-vertex or SH) or 'Important' (then it will be rendered per-pixel). Forward rendering is recommended for scenes with a low number of lights and scene complexity.

- Deferred rendering renders every light per pixel, but the effect of the lights is applied to the pixels it illuminates on the screen, so the performance cost is not dependent on the number of objects the light illuminates, which in the case of forward rendering have to be then rendered multiple times. This can result in better performance when using a high amount of lights or creating a complex scene. It is not recommended to use deferred in the case of smaller scenes because it suffers from considerable overhead. It cannot handle semi-transparent objects and does not support antialiasing.

Types of global illumination

Default pipeline also provides multiple modes of global illumination [9]:

- Baked global illumination - Global illumination baked into a lightmap (a texture that mimicks shadowing).

- Real-time global illumination - The built-in render pipeline is the only of the three pipelines that provide real-time or mixed global illumination. Unity uses a middleware named Enlighten by Geometrics to calculate real-time global illumination.

- Mixed global illumination modes - Unity provides multiple modes of behavior for mixed lights [9]:

  - Baked Indirect - Combines real-time direct lighting with baked indirect lighting and real-time shadows. The result looks quite realistic and is moderately demanding resource-wise.
  - Shadowmask - Combines real-time direct lighting with baked indirect lighting and blends real-time shadows for objects near to the camera with baked shadows for objects far from it. It is the most realistic looking but also the most demanding out of the modes.
  - Subtractive - Provides baked for both direct and indirect lighting and real-time shadows only for one directional light. Suitable for low-end hardware because of its low-performance cost but using it causes the scenes to look less realistic and graphically impressive than the other two modes.

Types of lighting

Default pipeline provides four types of lighting [9].

- Point light - It sends out light in all directions from the center cut off at a specified range resulting in a sphere shape of influence. The decrease in intensity resulting from the increasing distance from the center is calculated using an inverse square law meaning that the intensity is inversely proportional to the square of the distance from the center of the light.
- Spot light - Similar to the point light, but it is limited to the angle resulting in the cone-shaped area of influence.
- Directional light - It sends out light in a specified direction everywhere in the scene from far away as it does not have a specified center. The intensity of the light does not diminish with range. It is often used as a source of sunlight in the scene.
- Area light - It can be defined as a disc or a rectangle and it sends out the light from one side of the shape. Due to the computational complexity needed, it is only available in a baked mode.

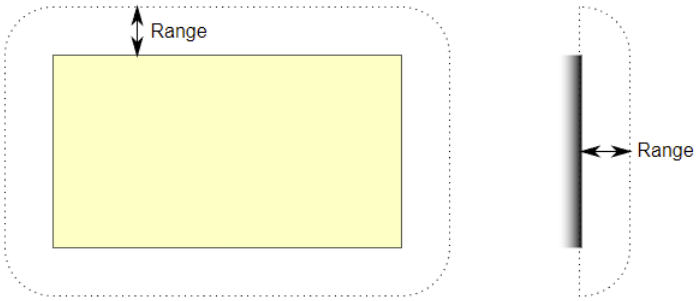

Figure 3.2: Area light and its range of influence [9].

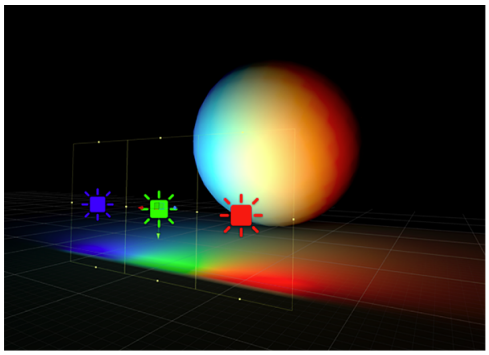

Figure 3.3: Three area lights illuminating a sphere showcasing the soft shading they produce [9].

Light modes

- Baked - Unity precomputes the lighting into lightmaps
- Real-time - Unity computes the lighting at runtime once per frame.
- Mixed - Its behavior is determined by the Mixed lighting mode.

Light properties

- Type - Directional, point, spot, and area.
- Range - Only for point and spot lights. Defines how far the light will travel from the center of the light before the intensity goes to zero.
- Spot Angle - Only for spot light. Defines the angle at the base of the spot light's cone.
- Color - Defines the color of the light.
- Mode - Baked, real-time and mixed.
- Intensity - Defines the brightness of the light.
- Indirect Multiplier - Useful only when using global illumination. Defines the intensity and behavior of the indirect light. If the value of the indirect multiplier is higher than 1 the intensity grows with every bounce. If it is lower than 1 the intensity diminishes with every bounce.

Shadow properties of the light

- Shadow type
  - Soft shadows - Shadows have smoothed edges producing more realistic-looking shadows. Soft shadows take more time to render than hard shadows.
  - Hard shadows - Shadows with sharp edges.
- Shadow settings for lights set to a mixed or baked lighting mode:
  - Baked shadow angle - Only for directional light with soft shadows. Softens the edges of the shadows.
  - Baked shadow radius - Only for point and spot lights. Softens the edges of the shadows.
- Realtime shadows - settings for lights with lighting mode set to realtime or baked lighting mode:
  - Strength - Controls the darkness of the shadows.
  - Resolution - Controls the resolution of the shadows.

- Bias - Controls the distance at which the shadows are pushed away from the light
- Normal bias - Controls how far along their normals shadow casting surfaces will be pushed.
- Near plane - Controls the values of the near clip plane.

Additional properties of the light

- Cookie - It is used to specify a texture mask to create silhouettes.
- Draw halo - Draws a halo around the light with its diameter defined by the range of the light.
- Render mode - Sets the rendering priority related to the rendering paths.
- Flare - Can be used to assign a lens flare.
- Culling mask - Uses Unity's layer system. It can be used to exclude objects from being affected by light.

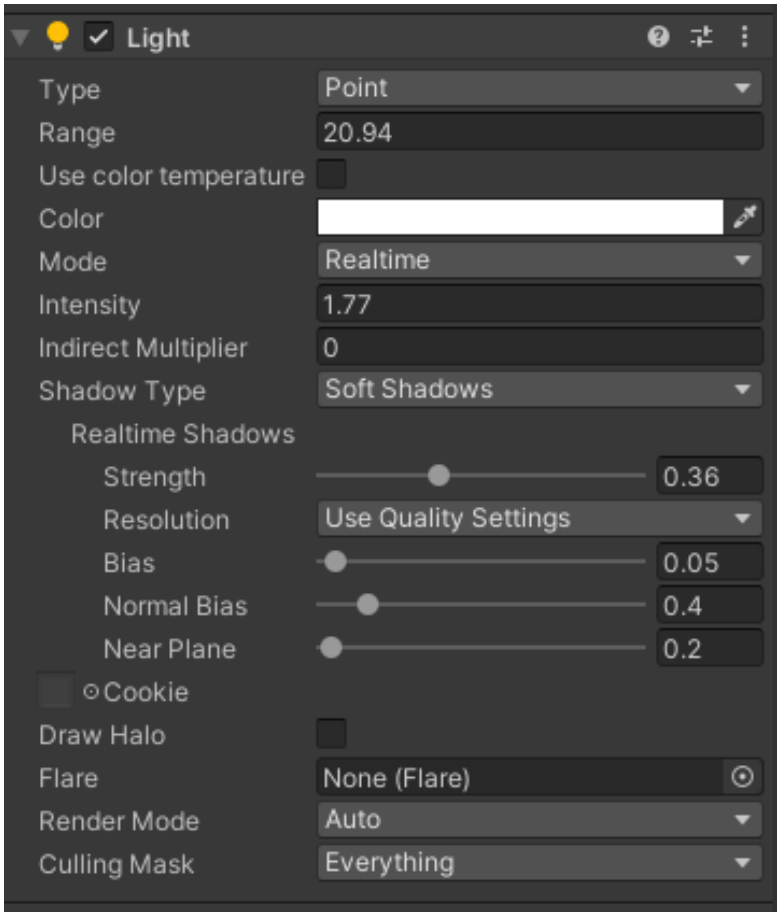

Figure 3.4: Properties in the light inspector.

3.2.2 Shaders in the default pipeline

There are 3 main shaders available in the default pipeline, excluding the legacy shaders kept in Unity because of backward compatibility [20]:

- Standard shader - Default shader when using the default pipeline.
- Standard particle shader - Used for rendering particle effects.
- Autodesk interactive shader - Used for materials with interactive PBS shaders used in Autodesk 3DsMax and Autodesk Maya. It uses identical properties to the PBS shader, but its appearance is not completely identical.

Standard shader

The Standard shader is the default shader for the build-in render pipeline. Shaders dictate the properties and appearance of material [10]. Standard shader incorporates physically based shading, illumination model of the Standard shader uses Disney model for diffuse component, GGX model for specular, with Smith Joint GGX visibility term and Schlick Fresnel approximation [21]:

$$\mathbf{f}_r(\overrightarrow{\omega_r}, \overrightarrow{\omega_i}) = \mathbf{D}_f + \frac{FDG}{4}$$

where $\mathbf{D}_f$ (diffuse component) is the Disney model for diffuse. This model chooses to add Fresnel factor to the diffuse model using Schlick-Fresnel approximation and slightly tweaking it. Let ω_v be the incidence angle of the viewing vector, ω_l be the incidence angle of the light vector, ω_d the angle between the light vector and half-vector, roughness the roughness parameter of the material, $\mathbf{C}_d$ the base color, the diffuse function is defined as [7]:

$$\mathbf{D}_f = \frac{\mathbf{C}_d}{\pi}(1 + (\mathbf{F}_{D90} - 1)(1 - cos\omega_l)^5)(1 + (\mathbf{F}_{D90} - 1)(1 - cos\omega_v)^5)$$

where

$$\mathbf{F}_{D90} = 0.5 + 2roughness\, cos^2\omega_d$$

Let u be the cosine of the angle between the light vector and the half vector, $\mathbf{f}_\lambda$ be the spectral distribution of the fresnel factor. The Schlick's fresnel approximation is defined as follows [4][11]:

$$F = \mathbf{f}_\lambda + (1 - \mathbf{f}_\lambda)(1 - u)^5$$

Let α_g be the roughness parameter, m a microsurface normal, n a macrosurface normal, $\chi^+(x)$ the positive characteristic function (equals one if x is greater than zero and equals zero if it equal or less than zero), the θ_m the angle between m and n, the GGX distribution is defined as [12]:

$$D = \frac{\alpha_g^2 \chi^+(m \cdot n)}{\pi cos^4\theta_m(\alpha_g^2 + tan^2\theta_m)^2}$$

G is the Smith Joint height-correlated masking-shadowing function. Let $\overrightarrow{\omega_o}$ be the direction of the reflected light, $\overrightarrow{\omega_i}$ be the direction of the incident light, $\overrightarrow{\omega_m}$ the microsurface normal, α be the roughness parameter, θ_o the angle of of the reflected light, θ_i the angle of incidence, λ be the directional correlation factor, the function is defined as [13]:

$$G = \mathbf{G}_2(\overrightarrow{\omega_o}, \overrightarrow{\omega_i}, \overrightarrow{\omega_m}) = \frac{\chi^+(\overrightarrow{\omega_o} \cdot \overrightarrow{\omega_m})\chi^+(\overrightarrow{\omega_i} \cdot \overrightarrow{\omega_m})}{1 + \Lambda(\overrightarrow{\omega_i}) + \Lambda(\overrightarrow{\omega_o})}$$

where

$$\Lambda(\overrightarrow{\omega_o}) = \frac{-1 + \sqrt{1 + \frac{1}{a^2}}}{2},$$

$$a = \frac{1}{\alpha tan\theta_o},$$

$$\Lambda(\overrightarrow{\omega_i}) = \frac{-1 + \sqrt{1 + \frac{1}{a^2}}}{2},$$

$$a = \frac{1}{\alpha tan\theta_i}$$

Smoothness and metallic parameter in the Standard shader

Standard shader adds to the model described above another component - global illumination, which includes its own fresnel term. The components of

```
half3 color =   diffColor * (gi.diffuse + light.color * diffuseTerm)
              + specularTerm * light.color * FresnelTerm (specColor, lh)
              + surfaceReduction * gi.specular * FresnelLerp (specColor, grazingTerm, nv);

return half4(color, 1);
```

Figure 3.5: Color computation in a BRDF function of the Standard shader, diffuse part in the first row, specular part in the second row, and global illumination in the third row. Functions or variables where metallic parameter contributes are highlighted blue (fresnel function, diff color), and where smoothness contributes are highlighted green.

the BRDF, as implemented in the Standard shader, use roughness instead of smoothness. Standard shader also distinguishes between perceptual roughness and roughness. Smoothness is the property that is presented to the player through material properties. The conversion formulas for perceptual roughness and roughness are [21]:

```
perceptual roughness = (1 - smoothness);
roughness = (perceptual roughness * perceptual roughness);
```

Perceptual roughness is used as the roughness parameter of the diffuse function, while roughness is used as the roughness parameter of the geometric and

distribution function. Thus, smoothness (as a Unity's material property) affects both the specular and diffuse components of the BRDF. Smoothness also affects the visibility of environmental reflection. Visually this means that at smoothness 0 there is no specular highlight and environmental reflection. At smoothness 1 there is a very focused specular highlight and mostly visible environmental reflection.

Metallic parameter is involved in the calculation of specular color and oneMinusReflectivity variable (albedo is the color of the material and unity_ColorSpaceDielectricSpec is a constant vec4):

```
oneMinusReflectivity = (unity_ColorSpaceDielectricSpec.a
 - metallic * unity_ColorSpaceDielectricSpec.a);

specColor = lerp (unity_ColorSpaceDielectricSpec.rgb, albedo, metallic);
```

Specular color is then used as a spectral distribution parameter of the fresnel function (fresnel function of the specular component and the global illumination component). OneMinusReflectivity multiplies diffuse color. This results in the diffuse component having no part in the color calculation if the metallic parameter is 1 as OneMinusReflectivity is then 0. The metallic parameter thus influences only the diffuse color and the effects of the fresnel function.

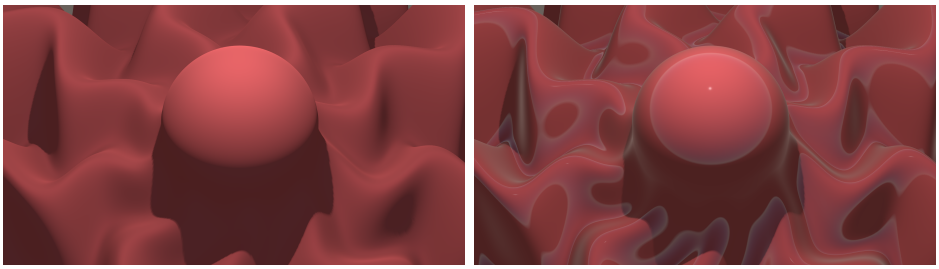

Figure 3.6: Smoothness 0 on the left, smoothness 1 on the right (metallic 0).

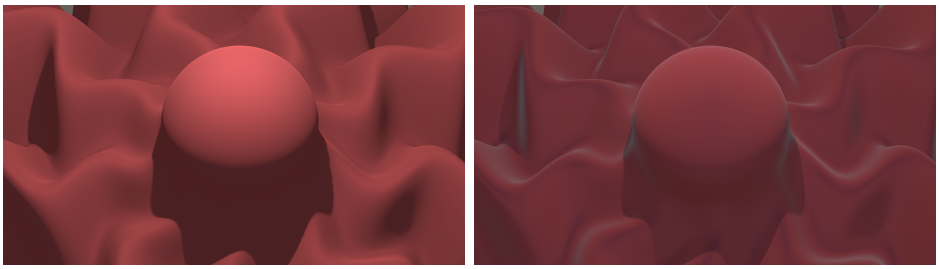

Figure 3.7: Metallic 0 on the left, Metallic 1 on the right (Smoothness 0).

Material properties in the Standard shader

These are the properties of the material that can be modified in the Standard shader [20]:

- Rendering mode - Unity provides 4 rendering modes that change the transparency of the object in various ways:
  - Opaque - Used for solid, non-transparent materials.
  - Cutout - Used for materials that have parts that are completely transparent - this mode does not allow for semi-transparency.
  - Transparent - Used for materials that have transparent or semi-transparent parts (for example, window texture)
  - Fade - This allows the material to completely fade out (and so make the object invisible), which is useful for animation.
- Albedo - Sets the color of the material either by changing the RGB values or using a texture.
- Metallic - values:[0-1] or a metallic map. The metallic surface reflects the environment more and loses its own albedo color. If the metallic parameter is closer to 0 material becomes less reflective, and its albedo color is more pronounced. The metallic map can be used to only define parts of a texture as metallic.

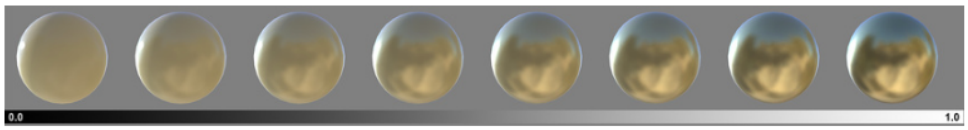

Figure 3.8: Image showcasing how the changes of metallic parameter influence the material appearance in the Standard shader. Smoothness = 0.8 [10].

- Specular - values: RGB color or a specular map. The brightness of the RGB color influences the strength of the specular reflection and the overall color of the tint.
- Smoothness - values:[0-1]. Smoothness controls the microsurface detail of the material. If the value of smoothness is close to one, the microsurface detail starts to resemble the smooth macrosurface making the light vectors reflect in a similar direction.

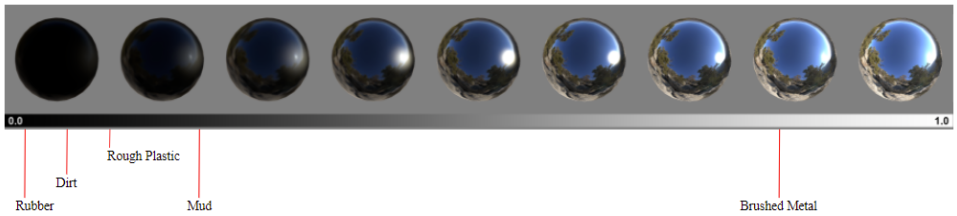

Figure 3.9: Image showcasing the changes in specular reflections as the smoothness parameter changes in the Standard shader [10].

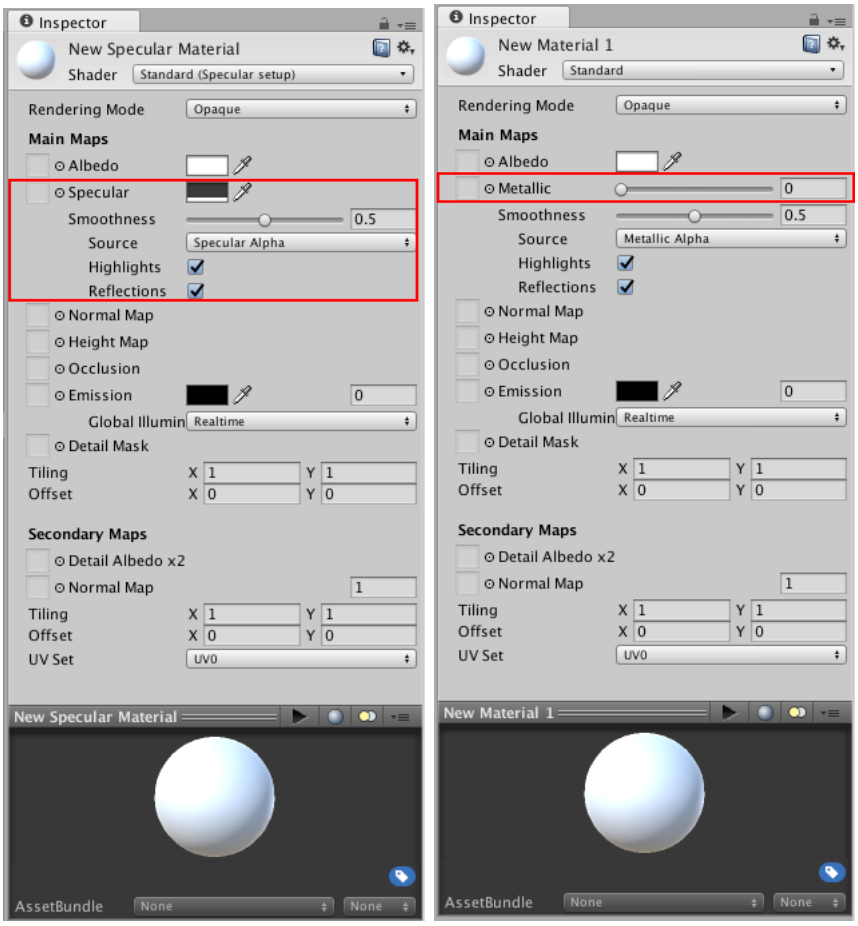

Figure 3.10: Specular mode (left) of the Standard shader and metallic mode of the Standard shader (right) [6][7].

- Source (of smoothness) - The source of the smoothness value from a texture. The source can be either the alpha channel of the specular/metallic map or the alpha channel of the albedo map.
- Highlights - Disable/enable the specular highlights.
- Reflections - Disable/enable reflections.
- Normal map - A texture that creates an illusion of small surface details by changing the way the light is reflected off of a surface.
- Height map - A texture that shifts the visible surface.
- Occlusion - Dictates which parts of the model should be more and which less indirectly lit
- Emission - Turning it on causes the material to emit light.
- Secondary map - Can be used to add a texture for details on top of the main texture, such as adding pores to the skin.

- Detail Mask - Can be used for excluding parts of the model from the effects of the secondary map.

Modes of the Standard shader

Materials in the Standard shader can be defined in two modes [20]:

- Specular mode - Changes the metallic slider to specular color [10].
- Metallic mode - It is the default mode of the Standard shader. Allows for changing the metallic parameter of the material.

3.3 URP - Universal render pipeline

URP is a prebuild scriptable pipeline and provides better customizability than default while being less demanding and available for most platforms [14].

3.3.1 Lighting in URP

URP provides two renderers:

- Universal renderer - the default renderer. Forward and deferred rendering modes are available.
- 2D renderer - applies 2D lighting to sprites.

The light properties are the same as in the Standard shader. URP also provides real-time global illumination using Enlighten as well as baked or mixed global illumination.

3.3.2 Materials in URP

URP shaders are:

- Lit - Default shader for URP - physically plausible.
- Simple Lit - Simplified Lit shader and is not physically plausible (doesn't check for energy conservation) [14].
- Bake Lit - It is not physically plausible and has no real-time lighting.
- Unlit - for object that don't need lighting.
- Particles Lit - For making almost photo-realistic looking particles.
- Particles Simple Lit.
- Particles Unlit.
- SpeedTree - For tree rendering.

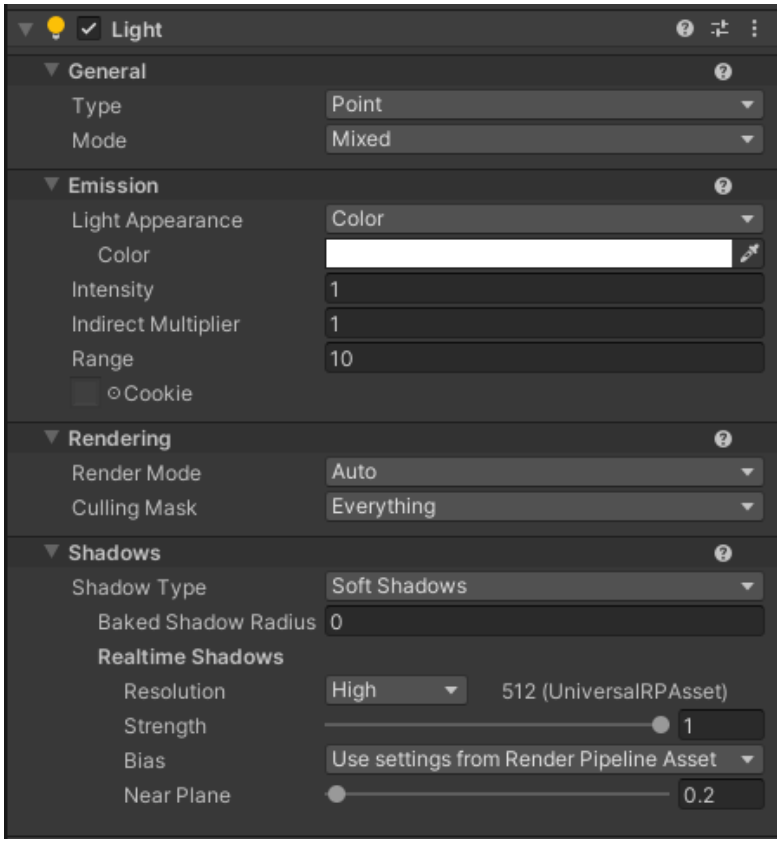

Figure 3.11: Properties in the URP light inspector.

- Autodesk shaders.

Similar to the Standard shader, Lit shader offers two modes: metallic and specular. Overall the options Lit shader provides are similar to the Standard shader, except lit shader adds a render face option which determines which side of the geometry to cull and which side to render. Only Lit and particles Lit shaders are physically plausible. Most of the other shaders are based on simpler illumination models (for example, Simple lit is based on the Blinn-Phong model). The illumination model of the Lit shader as described in the documentation - specular component based on GGX function seems very similar to the one used in Standard shader. Unfortunately, no details on how the diffuse component is calculated are provided [14].

3.4 HDRP High-definition render pipeline

High-definition render pipeline is the most demanding out of the three pipelines and is limited only to modern consoles and computers. It uses a configurable hybrid Tile/Cluster deferred/Forward lighting architecture[15].

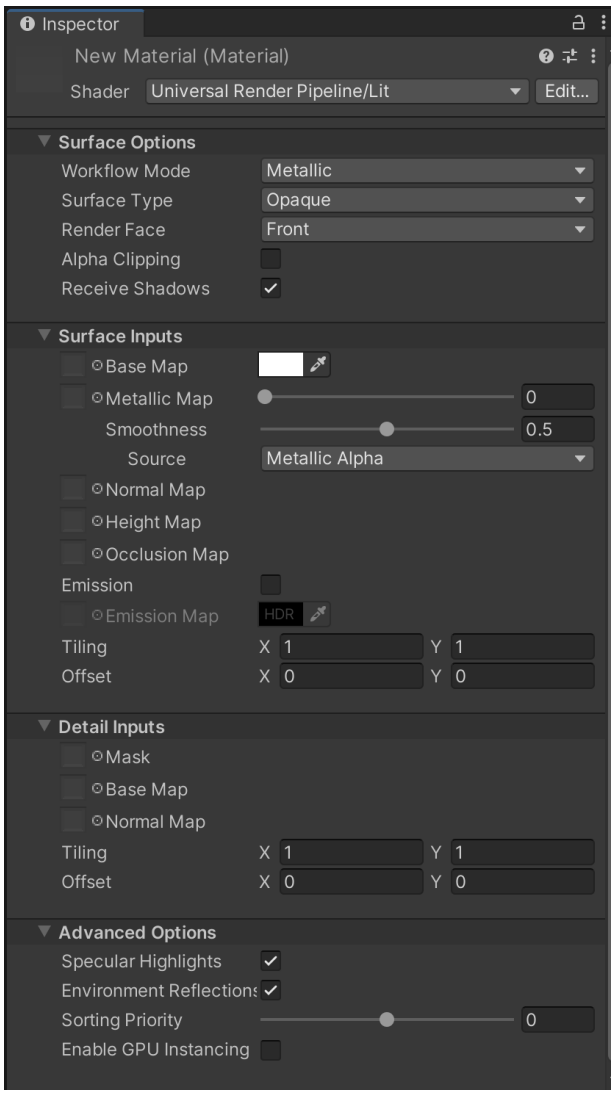

Figure 3.12: Material properties in URP Lit shader [8].

Volumes

HDRP uses volumes to determine how to render a scene environment. This includes post-processing, shadows, environmental effects like fog or sky, and exposure. Volumes with their volume profile can be used globally or assigned to a specific scene [15].

3.4.1 Lighting in HDRP

Global illumination in HDRP

HDRP provides baked and real-time global illumination. HDRP also supports raytracing.

Types and modes of light in HDRP

HDRP provides the same type of lights and modes as URP and default but with more options to modify their properties [15]:

- Spot light can be changed into three different shapes: cone, pyramid, and box.
- Directional light's angular diameter can be modified, which changes the size of the specular highlight and the softness of the shadows.

Light units in HDRP

- Candela - Unit of luminous intensity. Luminous intensity is a measure of the total amount of light the light source emits.
- Lumen - Unit of luminous flux. Luminous flux describes the intensity of the beam of light.
- Lux - Lumen per square meter.
- Nits - Candela per square meter.

Light properties in HDRP

Light properties are largely the same as in URP or default pipeline, but there are some additional properties [15]:

- Affect Diffuse - If enabled the light will affect the diffuse lighting of the material.
- Affect Specular - If enabled the light will affect the specular lighting of the material.
- Fade Distance.
- Range Attentuation - Enable or disable the light attentuation.
- Intensity Multiplier.
- Include For Raytracing.

Volumetrics and shadows

Volumetric options determine how the behavior of light is affected by atmospheric scattering (fog, clouds, or mist, for example).

- Enable.
- Dimmer - Dims the volumetric lighting effect of this light.
- Shadow Dimmer - Dims the volumetric fog of this light.

HDRP adds some new shadow properties to the ones in URP or default pipeline, such as dimmer, penumbra tint, or fade distance.

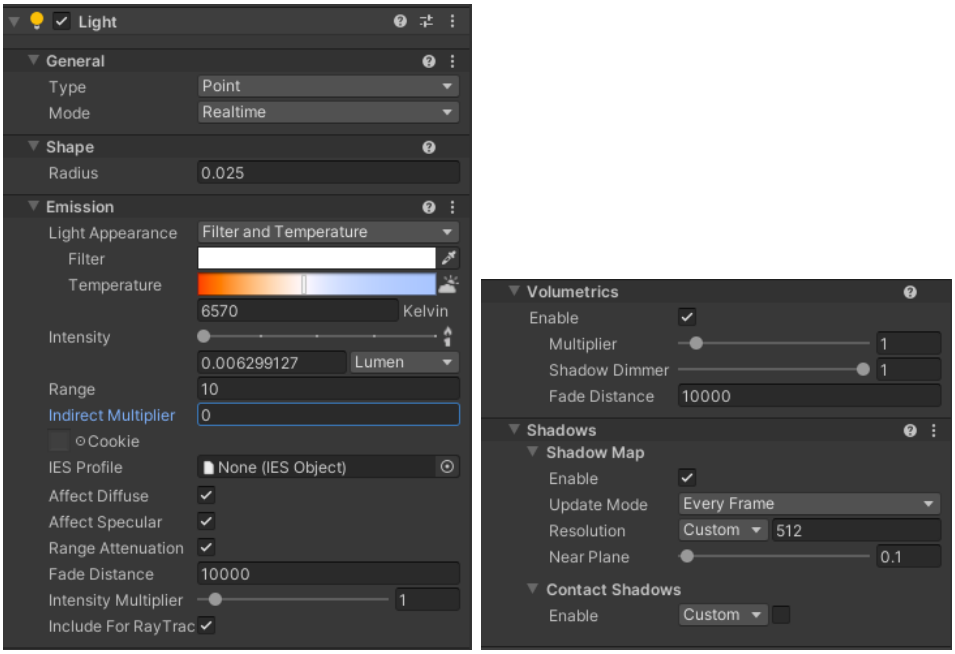

Figure 3.13: Properties in the HDRP light inspector (without volumetric and shadow components).

3.4.2 Shaders in HDRP

HDRP main shaders are:

- Lit Shader
- Layered Lit Shader

It also provides specialized shaders for specific objects/surfaces, for example, Eye shader, Hair shader, Silk shader, or Terrain lit shader. Lit shader for HDRP is physically based and seems like it is an upgraded version of the UPR Lit shader with additional features such as an option of turning on Anisotropy or Subsurface scattering.

Material properties in the Lit (HDRP) shader

Surface options of the material properties:

- Surface Type - Opaque or transparent.
- Alpha Clipping.
- Double-Sided - If enabled, both faces of the polygon are rendered.
- Material Type
  - Subsurface Scattering - Used for translucent objects. Simulates the light scattering occurring in translucent objects - blurring and scattering it.

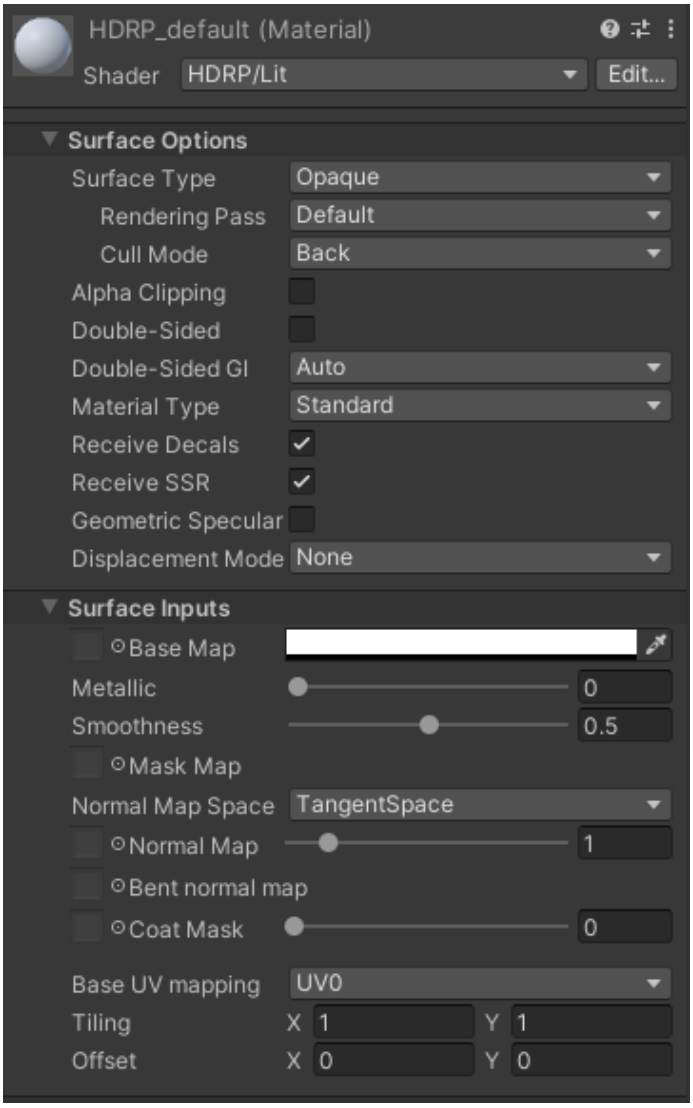

Figure 3.14: Material properties in HDRP Lit shader (without emission, detail map, and advance options as they are very similar to the ones in the URP/Standard shader) [8].

- Standard - The default mode.
- Anisotropy - The appearance of the highlights of anisotropic surfaces depends on the angle the material is looked at.
- Iridescence - Iridescence causes a change of color of the surface based on the viewing angle.
- Specular color - Similar to the specular mode of the Standard shader.
- Translucent - Similar to the subsurface scattering mode but does not blur the light.

- Receive Decals.

- Receive SSR (screen space reflections).
- Geometric Specular AA - Modifies the smoothness value to get rid of specular artifacts.
- Displacement Mode - Vertex or pixel. Heightmap either displaces mesh's vertices or pixels.

New or different maps to the ones available in the Standard shader:

- Mask Map - replaces the metallic, ambient occlusion, and smoothness map. Each component of the mask map stores a different map. Red color stores the metallic map, green the ambient occlusion map, blue the detail mask map, and alpha the smoothness map.
- Coat Mask - Simulates a clear coat effect.
- Bent Normal Map.

Chapter 4

Implementation

The main goal of the application is to provide an environment for comparing the appearance of materials between the main shaders (Default and Lit) of Unity's three pipelines (Default, URP, HDRP). The application was developed in Unity version 2021.2.6f1.
Packages used:

- High Definition RP
- Universal RP

Assets used in the application:

Asset name	**Author**
HDRP Furniture Pack	Tridify
Apartment kit	Brick project studio
Picture frames with photos	3Dfrk
Trash Bin	Rodolfo Rubens
Door Free Pack Aferar	Andrey Ferar
Pack Gesta Furniture 1	Gest
TV Furniture	Enozone
QA Books	QAtmo
Classic Interior Door Pack 1	Jan Fidler
Wooden PBR Table	Intercido
Kitchen Props Free	Jake Sullivan
Interior Props Pack Asset	reach the enD
Polygon dining room	alebrijes Studio
Free Stylized Skybox	Yuki2022

Table 4.1: Table of used assets.

The user can switch between two scenes - the default scene and the composite scene.

4.1 Default scene

The default scene contains four scene compositions - Default, Mitsuba, Medusa, and Spikes. On the left side of the screen, there are two columns of buttons. The first column enables the user to switch between pipelines and the second column between the scene compositions. There is also an option to hide the UI. Underneath, there is a checkbox to show FPS and a button to switch to the composite scene. In the bottom left part, there are buttons that uncover sliders to either modify the material properties or the light.
There are five sliders that change the material properties in all pipelines (range 0-1):

- Metallic
- Smoothness
- Red
- Green
- Blue

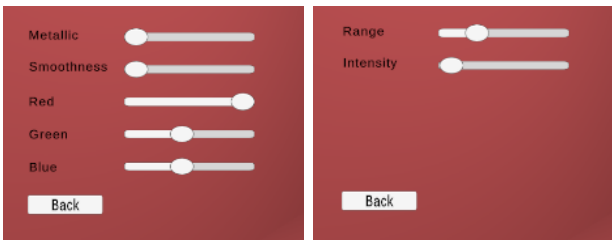

Figure 4.1: Material (left) and light (right) sliders.

There are two sliders that change the light properties in all pipelines :

- Range (range 0-40)
- Intensity (0-100 - in the case of HDRP multiplied by 10000)

On the right side of the screen, there are 12 material presets available - rubber, gold, silver, aluminium, plastic, withered wood, blue paint, fabric, marble, wood, concrete, and brick. They change the material of the object - in the default scene, it is the material of objects in all compositions except for the inner mitsuba ball and backgrounds. On the right side, there are also buttons to reset the scene, turn on image view and save the image of the scene or save the scene itself.

4.1.1 Scene composition

The default composition contains a sphere and a box surrounded by walls. The mitsuba composition contains a mitsuba knob standing on a plane. Medusa

and spikes compositions contain an optimized model for evaluating BRDFs, approximately emulating the proposed ideal conditions (50° angle and a distance of the camera between 0.3 and 0.8 meters) [16].

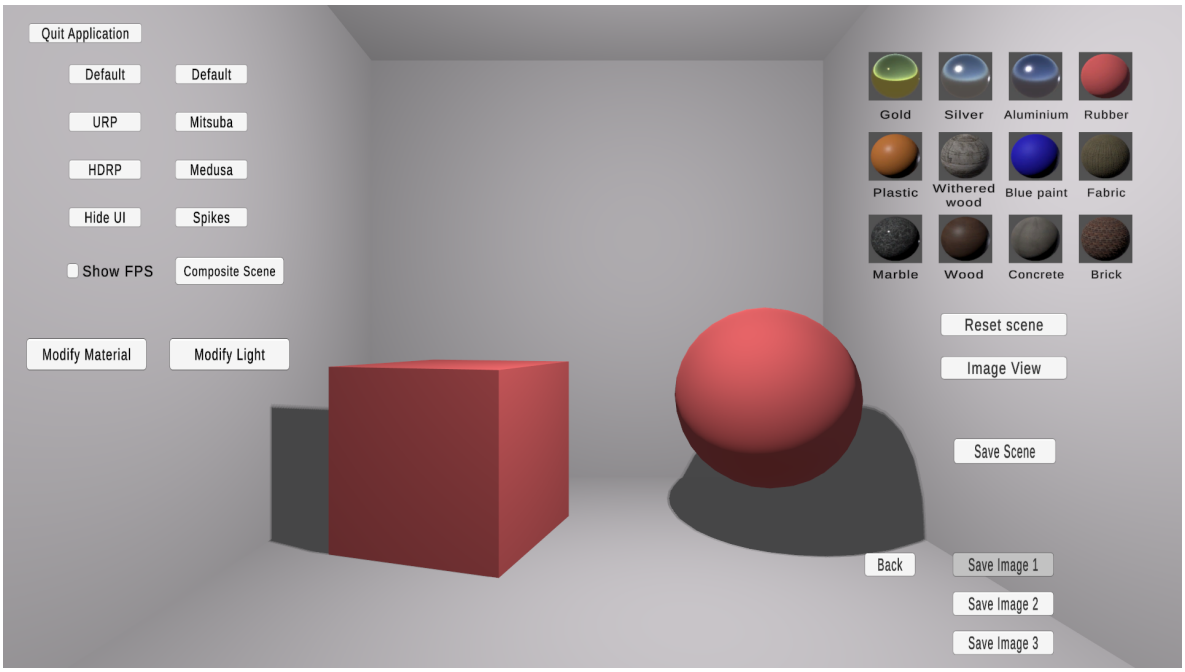

Figure 4.2: Default composition in Default rendering pipeline.

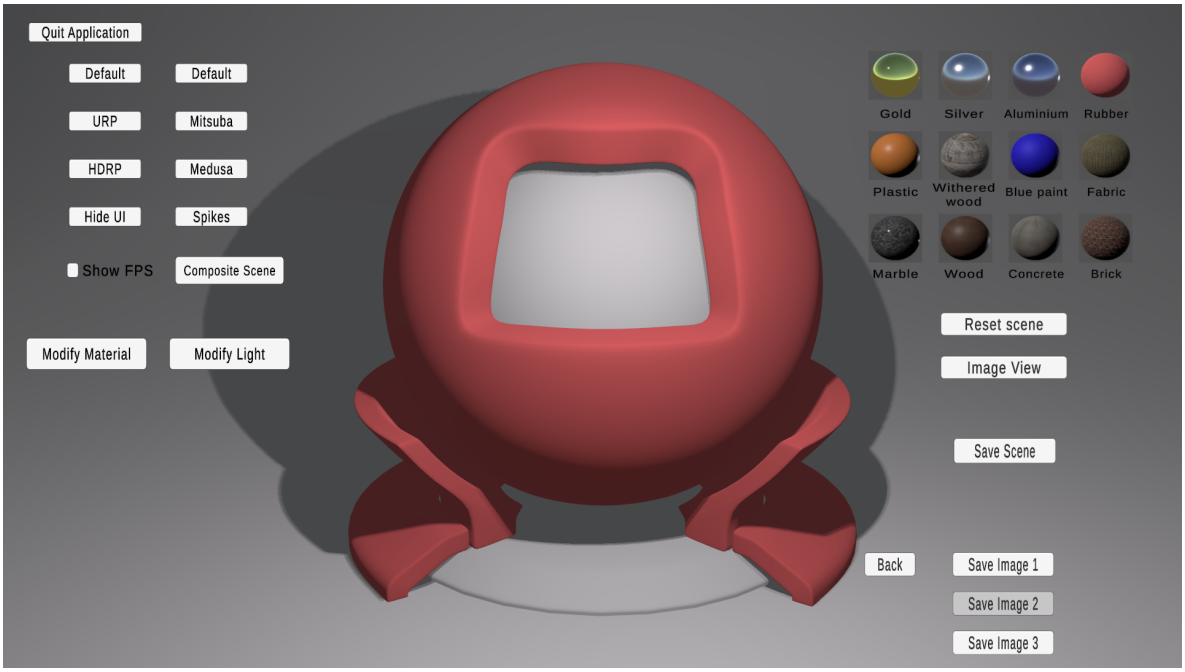

Figure 4.3: Mitsuba composition in Default rendering pipeline.

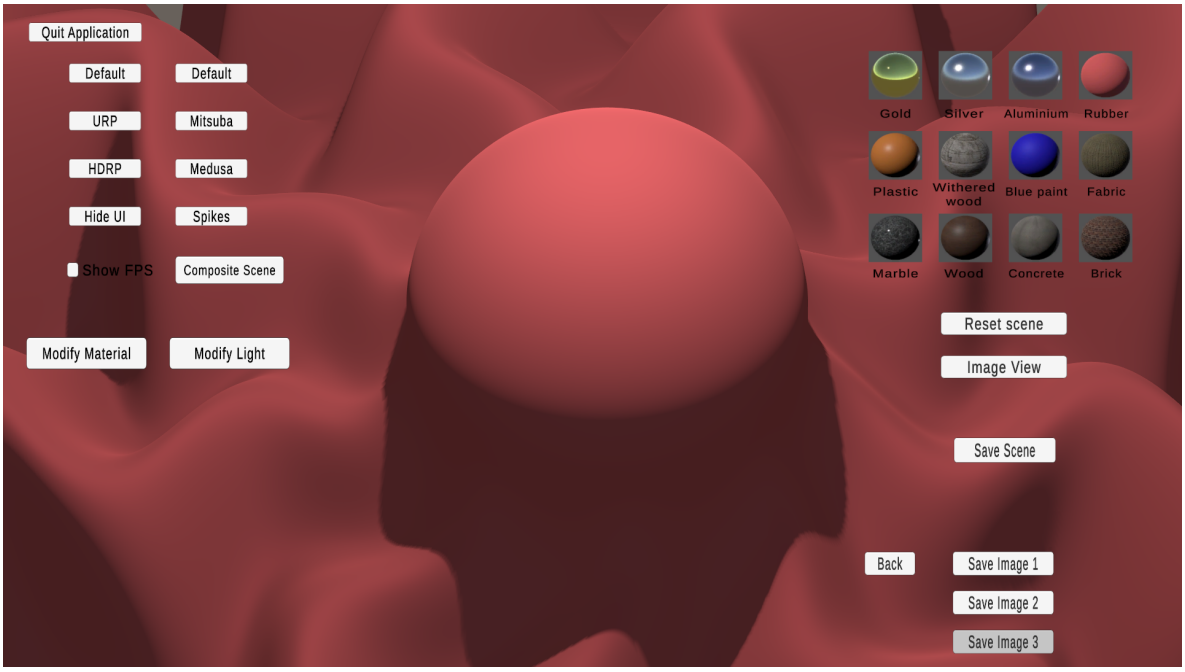

Figure 4.4: Medusa composition in Default rendering pipeline.

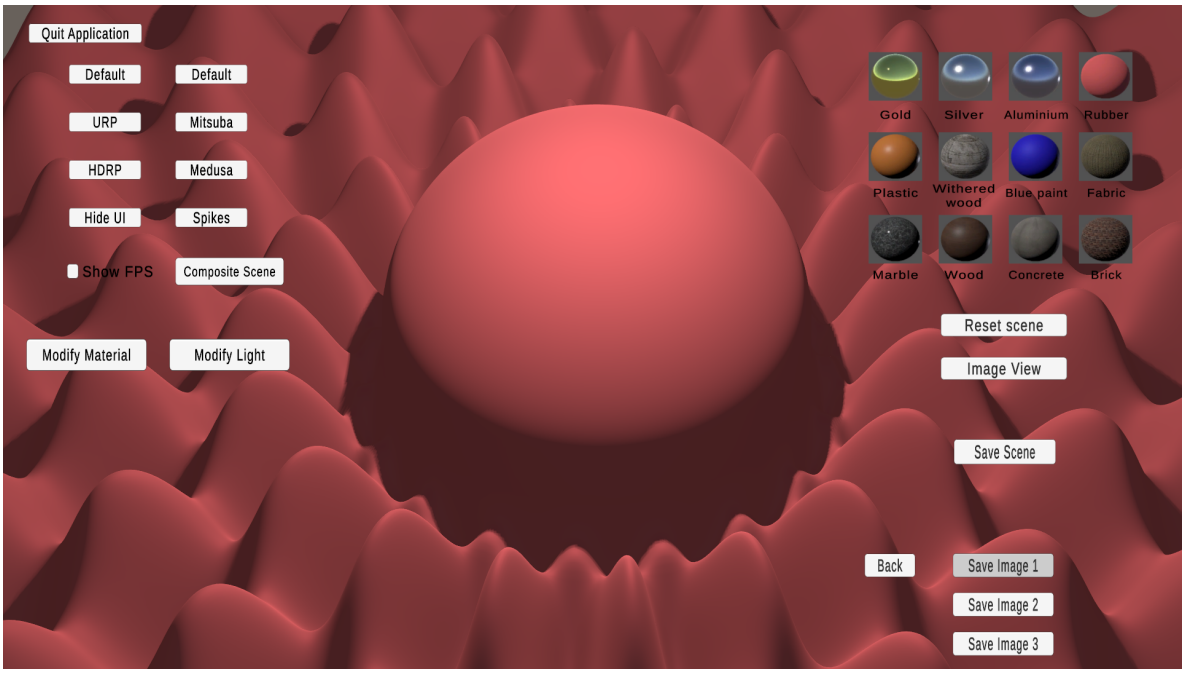

Figure 4.5: Spikes composition in Default rendering pipeline.

4.2 Composite scene

The composite scene contains an interior of a room from multiple angles and also an option to roam around it freely. Instead of switching between scene

compositions, there is switching between four different cameras placed around the room's interior. The material or light to be modified can be chosen by clicking on the pick an object or pick a light button and then hovering and clicking on an object or source of light (If the mouse hovers over an object in the picking mode, it turns orange. If the mouse hovers over a source of light in the picking mode a yellow sphere appears).

Figure 4.6: View of the composite scene from camera 1.

4.3 Saving, loading, and pipeline switching

Pipeline switching is done by assigning the respective pipeline asset to the quality and graphics settings and then changing the shader of every material to the main one used by the pipeline. In order not to have to create three types of every material by hand (Default, URP, and HDRP) for the composite scene, a script was written that takes the properties of the previous shader and assigns them to the new equivalent property in the new shader. Unity seems to ignore (and not transfer) shader properties if, upon changing the shader on the material, previous shader properties have different names than the new shader properties. Saving and loading functions save/load these shader properties to/from a JSON file located in the streaming assets folder.

4.3.1 Shader properties

Standard shader materials are the base that every material gets converted from. This means that exclusive properties to the HDRP-Lit shader or URP-lit shader are not taken into account. Properties that are not changed by

any of the materials used in the scene are not integrated into the conversion functions. These are specular highlights, glossy reflections, srcBlend, dstBlend, zWrite, emissionMap, detail maps (detail versions of - mask, albedo, normal map, and normal map scale), UV sec, glossMapScale, and smoothness texture channel. Table of the Standard shader properties and their counterparts in URP-Lit shader and HDRP-Lit shader:

Default	URP-Lit	HDRP-Lit
_Color	_BaseColor	_BaseColor
_MainTex	_BaseMap	_BaseColorMap
_Cutoff	_Cutoff	_AlphaCutoff
_Glossiness	_Smoothness	_Smoothness
_Metallic	_Metallic	_Metallic
_MetallicGlossMap	_MetallicGlossMap	_MaskMap - R component
_BumpScale	_BumpScale	_NormalScale
_BumpMap	_BumpMap	_NormalMap
_Parallax	_Parallax	No direct equivalent
_ParallaxMap	_ParallaxMap	_HeightMap
_OcclusionStrength	_OcclusionStrength	No direct equivalent
_OcclusionMap	_OcclusionMap	_MaskMap - G component
_EmissionColor	_EmissionColor	_EmissiveColor
_Mode	_Surface	_SurfaceType

Table 4.2: Table of Standard shader properties and their equivalents in URP-Lit shader and HDRP-Lit shader that are converted when switching to a different pipeline or while loading/saving.

4.3.2 Shader keywords

Shader keywords are used to switch between shader variants [22]. Shader variants are used because when building the project, Unity does not have to include the whole shader, just the parts that are used in the application. Subsequently, this creates a problem when the build application needs to switch between shader variants. This is resolved by creating a placeholder material that uses the needed material properties, which makes it, so the required shader variants are included in the build version. Keywords used in the application are:

- _NORMALMAP
- _ALPHATEST_ON - Cutout mode
- _EMISSION
- _PARALLAXMAP - Height map

- _METALLICGLOSSMAP - Metallic map

These keywords stay the same in URP-Lit, but in HDRP-Lit, there are a few changes -_PARALLAXMAP is _HEIGHTMAP, _METALLICGLOSSMAP is missing since there is no metallic map in HDRP, only mask map, and _EMISSION is also missing but can be turned on using a shader property instead.

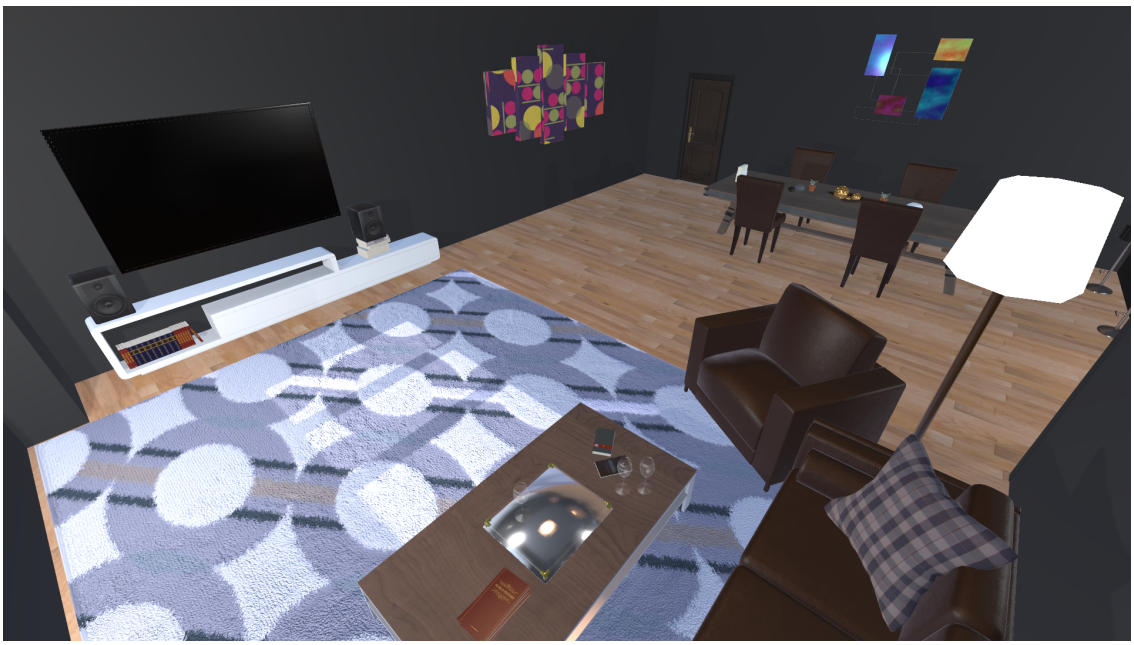

Figure 4.7: View of the composite scene from camera 2 (without UI).

4.3.3 Complications

- Due to the pipeline's differing lighting systems, it is difficult to set them up to be identical amongst them. Values of lights are thus only empirically configured to look similar.
- It is not possible to switch pipelines in the build version, only in the editor [17].
- The conversion of the metallic map and ambient occlusion map to a mask map was too slow since about 60 materials use either metallic map or ambient occlusion map. The conversion algorithm has to go through every pixel of the mask map to assign the proper value from the metallic map and occlusion map (most textures being 2048 * 2048), the algorithmic complexity being $O(n^2)$. For this reason, I have decided not to convert the occlusion map and metallic map to mask map resulting in HDRP-Lit materials being without these texture maps.
- Emission in HDRP pipeline seems not to work correctly - at least in Unity version 2021.2.6f1 - when assigning the emission color in script at

runtime. The new value is visible in the inspector but, unfortunately not in the scene.

- I was not successful at reliably turning on transparency at runtime when switching pipelines, so I instead used a prepared transparent material as a base to which then the properties are assigned through the converter.

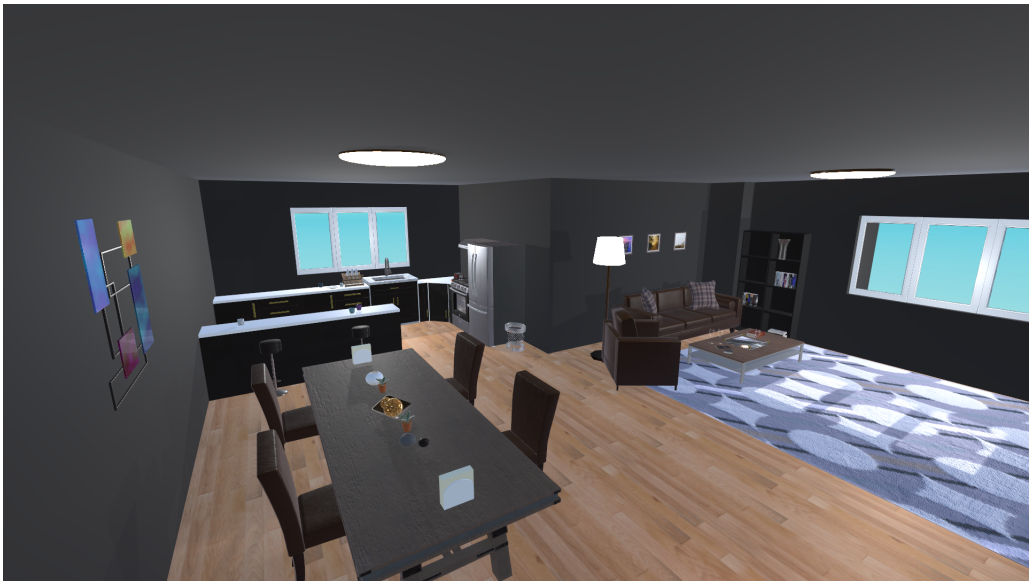

Figure 4.8: View of the composite scene from camera 3 (without UI).

Figure 4.9: View of the composite scene from camera 4 (without UI).

4.4 Image view and image comparison modes

The application also provides image view mode and image comparison mode. In image view mode, saved images of the scene can be displayed. In comparison mode, two images can be put over each other and compared by clicking on the image comparison button and moving with the comparison slider either right to reveal more of the first image or left to reveal more of the second image.

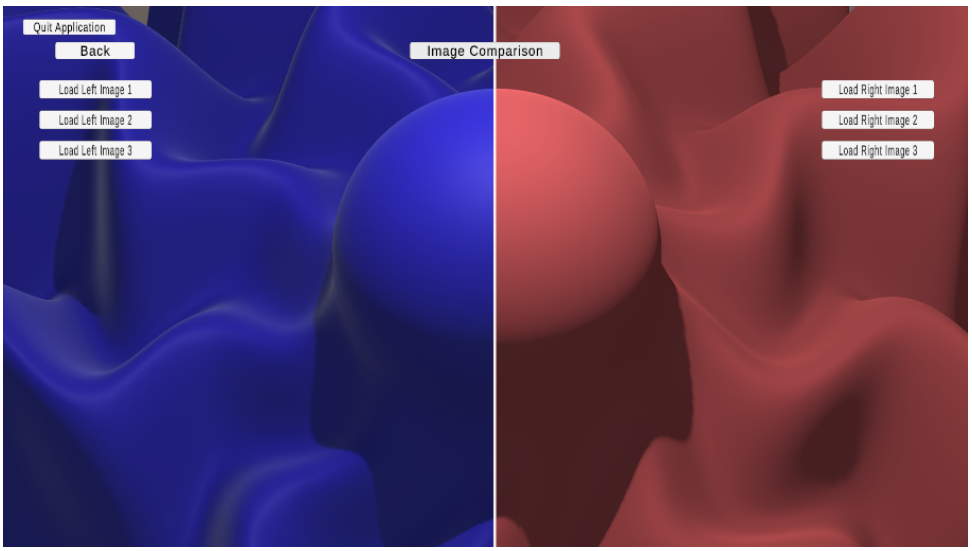

Figure 4.10: Comparison mode.

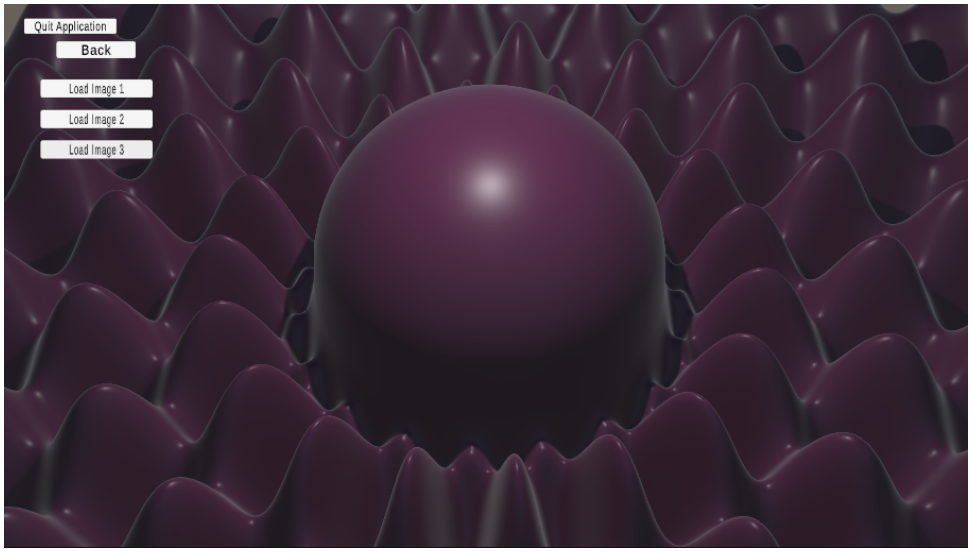

Figure 4.11: Image view mode.

Chapter 5

Comparison/analysis of pipelines illumination models

The appearance of the materials differs when using different pipelines and shaders. In this chapter, the appearance of materials will be compared between:

- Default pipeline - Standard shader
- URP - Lit shader
- HDRP - Lit shader

5.1 Empirical appearance analysis

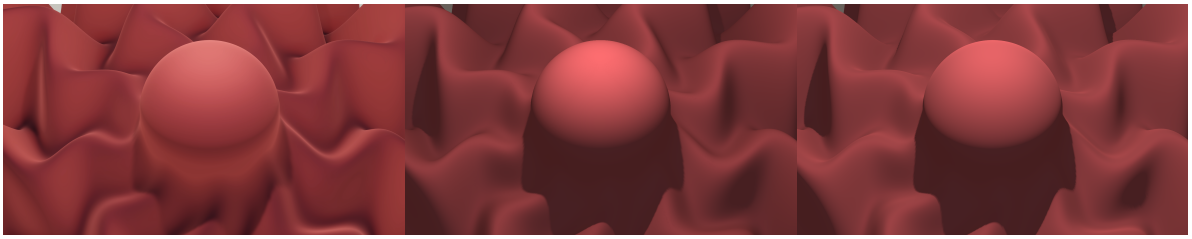

Figure 5.1: Medusa model - Albedo: R-255, G-109, B-109, Smoothness 0 and Metallic 0, HDRP-Lit (left) , URP-Lit (middle), Standard shader (right).

Due to the differing light settings in the pipelines, the intensity of the light varies. The intensity was corrected only empirically (For medusa composition, the light has an intensity of 3 in the Standard shader, 20 intensity in the URP, and 550000 in the HDRP). The color values of the materials are the same in all pipelines.

In Figure 5.1, the Medusa composition with material parameters smoothness 0 and metallic 0 in Unity's graphical pipelines is shown. The shadowing in the Standard shader looks much flatter and closer to pure Lambertian diffuse than in URP-Lit or especially in HDRP-Lit, where smoother light gradation is present. Even though the light gradation is smoother in the

URP-Lit shader than in the Standard shader, it seems to have starker shadow contrast. HDRP, on the other hand, seems to have the least stark shadows. All the materials are quite different color-wise despite having the same RGB values.

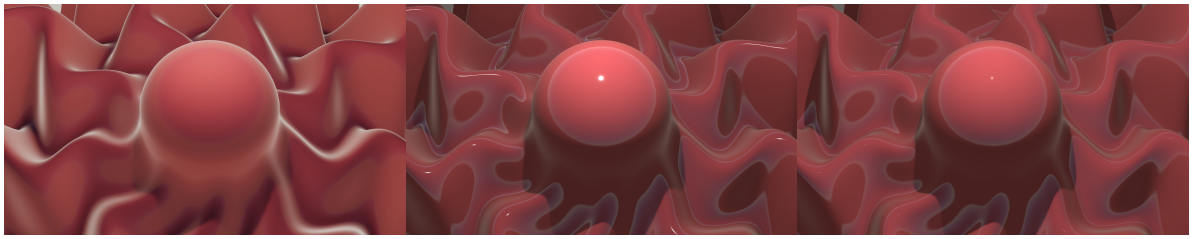

Figure 5.2: Medusa model - Albedo: R-255, G-109, B-109, Smoothness 1 and Metallic 0, HDRP-Lit (left), URP-Lit (middle), Standard shader (right).

In Figure 5.2, the Medusa composition with the material parameters smoothness 1 and metallic 0 in Unity's graphical pipelines is shown. Again Standard shader and URP-Lit shader are very similar in color and general appearance. The size of the specular highlight is the largest in the URP-Lit shader, followed by the Standard shader and the sharpest/smallest is in HDRP at the empirically similar light intensity. There is a big difference in the transparency of environmental reflections between the pipelines. At metallic 0 Standard shader seems to be the least transparent - environmental reflections being less prominent than in the URP-Lit shader or HDRP-Lit shader. This could be caused by HDRP-Lit and URP-Lit reducing the diffuse color more with increasing smoothness than the Standard shader. There can also be observed a much stronger fresnel effect (more light reflected when viewed from grazing angles) in HDRP-lit than in the other shaders.

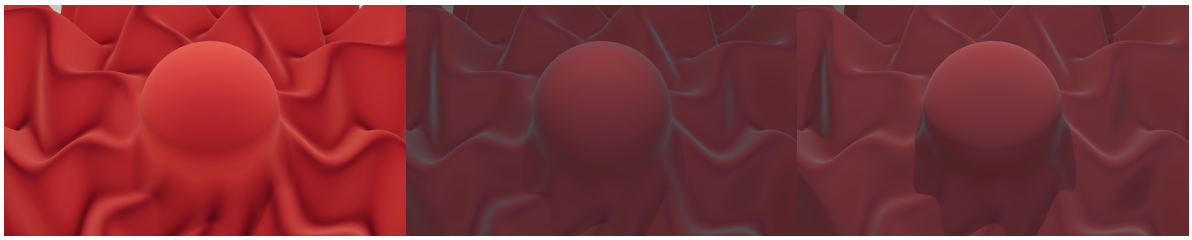

Figure 5.3: Medusa model - Albedo: R-255, G-109, B-109, Smoothness 0 and Metallic 1, HDRP-Lit (left), URP-Lit (middle), Standard shader (right).

In Figure 5.3, the Medusa composition with the material parameters smoothness 0 and metallic 1 in Unity's graphical pipelines is shown. The metallic parameter, as described before, influences the fresnel effect and the amount of diffuse color reflected in the Standard shader. URP-Lit shader's metallic parameter seems to have a very similar effect, only providing a slightly stronger fresnel effect. HDRP-Lit shader's metallic parameter does

not seem to provide any or provide at most a very minimal fresnel effect and only seems to influence the amount of color reflected from the environment.

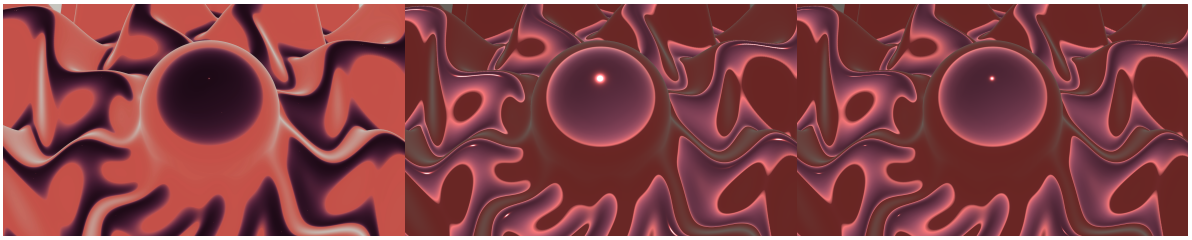

Figure 5.4: Medusa model - Albedo: R-255, G-109, B-109, Smoothness 1 and Metallic 1, HDRP-Lit (left), URP-Lit (middle), Standard shader (right).

In Figure 5.4, the Medusa composition with the material parameters smoothness 1 and metallic 1 in Unity's graphical pipelines is shown. This combination of smoothness and metallic parameters seems to result in a clear environmental reflection and a strong fresnel effect.

5.2 Performance comparison

Hardware used: PC 1

- Processor: Intel i5 - 8600k
- Graphics card: Nvidia GTX 2060S
- Ram: 16 GB
- OS: Windows 10

PC 2

- Processor: Intel i7 - 8550U
- Graphics card: Intel UHD Graphics 620
- Ram: 16 GB
- OS: Windows 10

The quality of textures and shadows in all pipelines is set to the highest possible option, and MSAA x8 is turned on. Resolution is 1920x1080. For the composite scene performance testing, the camera 3 view was used because it provides the view of the most objects in the scene. Rows are defined as pipelines and columns as model compositions. The values are the average FPS taken in an interval of ten seconds. The value of average FPS in the first table is calculated as a mean of five times recorded cumulative moving average in the interval of ten seconds. In the second table it is calculated as

Average FPS	Default	Mitsuba	Medusa	Spikes	Composite
Default	433	375	243	253	499
URP	456	427	335	316	501
HDRP	282	254	181	185	109

Table 5.1: Table of average FPS (cumulative moving average) in various pipelines - PC 1.

Average FPS	Default	Mitsuba	Medusa	Spikes	Composite
Default	435	372	241	255	501
URP	443	422	335	314	507
HDRP	282	253	181	185	107

Table 5.2: Table of average FPS (exponential moving average) in various pipelines - PC 1.

Average FPS	Default	Mitsuba	Medusa	Spikes	Composite
Default	16	16	11	11	29
URP	19	17	12	12	30
HDRP	10	10	6	6	4

Table 5.3: Table of average FPS (cumulative moving average) in various pipelines - PC 2.

Average FPS	Default	Mitsuba	Medusa	Spikes	Composite
Default	17	16	11	11	29
URP	19	17	12	12	30
HDRP	10	10	6	6	4

Table 5.4: Table of average FPS (exponential moving average) in various pipelines - PC 2.

a mean of five times recorded exponential moving average in the interval of ten seconds. The testing was done in the build versions.

There is an expected drop in performance between the compositions with simple models (mitsuba and default) and the compositions with more complex ones (medusa and spikes). The number of triangles displayed is around 6 800 for default composition, around 369 000 for mitsuba composition, and around 4 million for medusa and spikes compositions in the default pipeline. This triangle count difference is similar in all pipelines; for example, in URP and HDRP, it is 4 million triangles and 3.5 million respectively for the more complex compositions. Unexpected is the framerate difference between the composite scene and the default scene. The models used in the composite scene are not as complex as medusa or spikes models, but they are still more complex than default and mitsuba models. There are also more shadow

casters and lights in the composite scene.

URP seems to be slightly less demanding than the default pipeline in the build version. In the editor, however, the framerate is slightly worse when using the URP pipeline, which could mean a more demanding editor overhead in URP. Out of all the pipelines, HDRP seems to be the most demanding.

Chapter 6

Conclusion

In this thesis, various BRDF models (empirical and physically-based) were described as well as the BRDF used in the Standard shader in the default pipeline in Unity. Furthermore, the material properties of each pipeline's (Default, URP, and HDRP) main shader were described.

The application that would enable the comparison of material properties and lighting between pipelines was designed and implemented. Material and light properties can be modified at runtime, or the material of the object can be changed by choosing one of the pre-defined 12 materials. The application features two scenes, a default scene containing four compositions that can be used to compare the material appearance and the composite scene containing a complex room in which any object can be chosen and modified. Using this application, the appearance of the materials in each pipeline was empirically analysed and compared.

Unfortunately, it is not possible to switch between pipelines in the build version, so additionally, three versions of the application for each pipeline were developed that have the same features except for the pipeline switching. The performance was tested and analysed using these build versions.

The application could be expanded by making more material or light properties modifiable at runtime, such as tiling, offset, emission, or color of the light. The converter used for loading, saving, and pipeline switching could also include additional material properties such as the various detail maps. Since all materials get converted from default, the application could not showcase the features exclusive to the URP and HDRP pipelines, so specialized versions of the application could be developed to include them and the different shaders available besides the main ones.

Appendix A

Contents of electronic appendix

The latex source of the thesis is in the Latex folder (the name of the tex file with the thesis is thesis.tex), images used in the thesis are also in the latex folder, promotional images are in the Images folder, specification.pdf is the file with the specification.

Appendix B

User manual

The user starts in the default scene where he can choose between multiple scene compositions with the buttons on the left side of the screen (Default, Mitsuba, Spikes, and Medusa). Next to it are also buttons to change Unity's pipeline (only in the pipeline switching project). In the bottom right part of the screen user can save/load a scene or save an image and then view it in the Image view mode. In the image view mode choosing the comparison mode and then clicking on image comparison enables the user to control the slider. To stop controlling the slider press Esc.

To switch to the composite scene click on the composite scene button. In the composite scene user can choose between multiple camera angles or by clicking on the free view button roam around the room.

To control the camera use mouse and WASD, to get out of the free view mode click on any of the camera buttons.

In the composite scene user can choose to pick an object (the object with the mouse on it is highlighted orange) or a light (highlighted with a yellow sphere - only legible objects is the lamp and ceiling lights) then modify them by using the sliders or clicking on a material preset in the top right corner.

Bibliography

[1] J.Žára, B. Beneš, J. Sochor and P.Felkel., Moderní počítačová grafika., 2., pp. 319-336, přeprac. a rozš. vyd. Brno: Computer Press, 2005., ISBN: 80-251-0454-0.

[2] Unity technologies, Unity Documentation: Materials, Link: https://docs.unity3d.com/Manual/Materials.html, Last visit: May 2022 [Manual].

[3] Blender, Blender Documentation: Materials, Link: https://docs.blender.org/manual/en/latest/render/materials/index.html, Last visit: May 2022 [Manual].

[4] R.Montes and C. Ureña, An Overview of BRDF Models, Dept. Lenguajes y Sistemas Informáticos University of Granada, March 8. 2012.

[5] B.T. Phong, Illumination for Computer Generated Pictures, Communications of ACM 18 , no. 6, 311–317, 1975

[6] J. F. Blinn, Models of light reflection for computer synthesized pictures, Proc. 4th Annual Conference on Computer Graphics and Interactive Techniques, pp. 192–198., CiteSeerX 10.1.1.131.7741. doi:10.1145/563858.563893. S2CID 8043767.

[7] J. Burley, Brent, and Walt Disney Animation Studios. "Physically-based shading at Disney." ACM SIGGRAPH. Vol. 2012., 2012.

[8] J.Žára, B. Beneš, J. Sochor and P.Felkel., Moderní počítačová grafika., 2., pp. 312-330, přeprac. a rozš. vyd. Brno: Computer Press, 2005., ISBN: 80-251-0454-0.

[9] Unity technologies, Unity Documentation: Choosing and configuring a render pipeline and lighting solution, Link: https://docs.unity3d.com/2022.2/Documentation/Manual /BestPracticeLightingPipelines.html [Manual]., Last visit: May 2022

[10] Unity technologies, Unity Documentation: Forward Rendering path, https://docs.unity3d.com/Manual/RenderTech-ForwardRendering.html, Last visit: May 2022 [Manual].

[11] Ch. Schlick, An inexpensive BRDF Model for Physically-based Rendering, Computer Graphics Forum. 13 (3): pp. 233–246., 1994.

[12] B. Walter, S.R. Marchnes, H. Li, K. E. Torrance, Microfacet models for refraction through rough surfaces, Eurographics Symposium on Rendering, 2007.

[13] Heitz, Eric. Understanding the masking-shadowing function in microfacet-based BRDFs., Journal of Computer Graphics Techniques 3.2, pp. 32-91, 2014.

[14] Unity technologies, Unity Documentation - URP manual, link: https://docs.unity3d.com/Packages/com.unity.render-pipelines.universal@14.0/manual/index.html, Last visit: May 2022 [Manual].

[15] Unity technologies, Unity Documentation - HDRP manual: High Definition Render Pipeline overview, Link: https://docs.unity3d.com/Packages/com.unity.render-pipelines.high-definition@14.0/manual/index.html, Last visit: May 2022 [Manual].

[16] V. Havran, J. Filip and K. Myszkowski, Perceptually Motivated BRDF Comparison using Single Image, Computer Graphics Forum 35(3), pp. 1-12, 2016 (Proceedings of Eurographics Symposium on Rendering 2016).

[17] Unity Forum, Switching between pipelines - Default, UPR and HDRP (When build), January 2022, Available at: "https://forum.unity.com/threads/switching-between-pipelines-default-upr-and-hdrp-when-build.1219485/#post-7783473".

[18] R. Cook, K. Torrance: A Reflectance Model for Computer Graphics, ACM Transactions on Graphics, no.1 , 7-24, 1982

[19] Unity technologies, Unity Documentation: Lighting, Link: https://docs.unity3d.com/Manual/LightingOverview.html, Last visit: May 2022 [Manual].

[20] Unity technologies, Unity Documentation: Shaders, Link: https://docs.unity3d.com/Manual/Shaders.html, Last visit: May 2022 [Manual].

[21] Unity technologies, Standard shader, Available at: https://unity3d.com/unity/qa/lts-releases, 2021 [code].

[22] Unity technologies, Unity Documentation: Shader keywords, Link: https://docs.unity3d.com/2022.2/Documentation/Manual/shader-keywords.html, Last visit: May 2022 [Manual].

Image sources

[1] J.Žára, B. Beneš, J. Sochor and P.Felkel., Picture of BRDF, Moderní počítačová grafika., 2., pp. 327, přeprac. a rozš. vyd. Brno: Computer Press, 2005., ISBN: 80-251-0454-0.

[2] M. Kraus., Vectors for calculating Phong and Blinn–Phong shading, Wikipedia, June 17 2011,
Link: https://en.wikipedia.org/wiki/Blinn%E2%80%93Phong_reflection_model#/media/File:Blinn_Vectors.svg, Last visit: May 2022.

[3] B. Smith., Illustration of the components of the Phong reflection model (Ambient, Diffuse and Specular reflection)., Wikipedia, August 7 2006,
Link: https://en.wikipedia.org/wiki/Phong_reflection_model#/media/File:Phong_components_version_4.png, Last visit: May 2022.

[4] B. Smith.,Demonstration of Blinn-Phong versus Phong reflection models, Wikipedia, August 7 2006,
Link: https://en.wikipedia.org/wiki/Blinn%E2%80%93Phong_reflection_model#/media/File:Blinn_phong_comparison.png, Last visit: May 2022.

[5] M. Wantke.,Flow chart of a 3D graphics rendering pipeline, Wikipedia, June 1 2021,
Link: https://en.wikipedia.org/wiki/Graphics_pipeline#/media/File:3D-Pipeline.svg, Last visit: May 2022.

[6] Unity technologies, Unity Documentation: Specular mode: Specular parameter,
Link: https://docs.unity3d.com/Manual/StandardShaderMaterialParameterSpecular.html, Last visit: May 2022.

[7] Unity technologies, Unity Documentation: Metallic mode, Link: https://docs.unity3d.com/Manual/StandardShaderMaterialParameterMetallic.html , Last visit: May 2022.

[8] Unity technologies, Unity Documentation - URP manual: Lit Shader, Link: https://docs.unity3d.com/Packages/com.unity.render-pipelines.universal@7.1/manual/lit-shader.html, Last visit: May 2022.

[9] Unity technologies, Unity Documentation - Types of light - Point light, Link: https://docs.unity3d.com/Manual/Lighting.html, Last visit: May 2022.

[10] Unity technologies, Unity Documentation - Standard shader, Link: https://docs.unity3d.com/Manual/StandardShaderMaterialParameterSmoothness.html, Last visit: May 2022.

[11] N.Hoffmann, Physically-Based Shading Models in Film and Game Production, SIGGRAPH 2010 Course Notes, 2010.