Advanced Technologies: Clay Appearance

William Whitehouse (19019239)
University of the West of England

January 04, 2023

1 Introduction

In this report, we will explore the rendering equation and BRDFs, as well as the use of DirectX 11 in the rendering of physically-based clay materials in real time. By understanding these concepts and technologies, we can improve the realism and accuracy of material renderings, and better understand the underlying principles of computer graphics.

2 Related Work

To create realistic images, it is important to accurately simulate the way light interacts with surfaces. The way that light reflects off, absorbs into, or passes through a material is influenced by the physical properties of that material (Wynn, 2000). For example, a mirror and sandpaper will reflect light differently due to their different physical characteristics.

2.1 BRDFs & The Rendering Equation

A Bidirectional Reflectance Distribution Function (BRDF) is a model used to determine how much light, given an incoming direction ($\vec{\omega_i}$), is reflected from a point on the surface (x) along an outgoing direction ($\vec{\omega_r}$) (TUWien, 2015). It can be written formally as:

$$f_r(\vec{\omega_i}, x, \vec{\omega_r}) \tag{1}$$

BRDFs follow three important properties to be considered physically accurate (Duvenhage, Bouatouch, and Kourie, 2013):

- **Helmholtz-Reciprocity** - The direction of the ray can be reversed:

$$f_r(\vec{\omega_i}, x, \vec{\omega_r}) = f_r(\vec{\omega_r}, x, \vec{\omega_i})$$

- **Positivity** - It is impossible for an exit direction to have negative reflectance:

$$f_r(\vec{\omega_i}, x, \vec{\omega_r}) \geq 0$$

- **Energy Conservation** - A surface may reflect or absorb incoming light, but no more can exit than the incoming amount:

$$\int_\Omega f_r(\vec{\omega_i}, x, \vec{\omega_r})\ cos\theta\ d\vec{\omega_i} \leq 1$$

The symbol $\int_\Omega$ indicates an integral over a hemisphere of all directions.

Typically the function uses spherical coordinates rather than cartesian vectors (Wynn, 2000), and the direction parameters are a pair of angles. Each pair contains the azimuthal angle and normal angle respectively (Montes and Ureña, 2012). For simplicity, the function will use direction vectors instead ($\vec{\omega_i}$ and $\vec{\omega_r}$).

A materials brightness (radiance) at a point is made up of its emitted light plus the reflected incoming light, this is formally the rendering equation (TUWien, 2015 and Lafortune, 1996) which is written as:

$$L_o(x, \vec{\omega_r}) = \underbrace{L_e(x, \vec{\omega_r})}_{\textit{emitted light}} + \underbrace{\int_\Omega L_i(x, \vec{\omega_i}) f_r(\vec{\omega_i}, x, \vec{\omega_r})(\vec{\omega_i} \cdot \vec{n}) d\vec{\omega_i}}_{\textit{reflected incoming light}} \tag{2}$$

- $L_o(x, \vec{\omega_r})$ - Outgoing radiance at point x towards direction $\vec{\omega_r}$.
- $L_e(x, \vec{\omega_r})$ - Emitted radiance at point x towards direction $\vec{\omega_r}$.
- $\int_\Omega \dots d\vec{\omega_i}$ - Sum of all the scattered light over all possible incoming directions.
- $L_i(x, \vec{\omega_i})$ - Incoming radiance from direction $\vec{\omega_i}$ to point x.
- $f_r(\vec{\omega_i}, x, \vec{\omega_r})$ - BRDF.
- $(\vec{\omega_i} \cdot \vec{n})$ - The dot product of the incoming light direction ($\vec{\omega_i}$) and the surface normal at point x ($\vec{n}$). Sometimes written as $cos\theta$ - as seen in the energy conservation equation.

The rendering equation cannot be solved directly, because the radiance at point x depends on the radiance at all other points in the scene, which also depend on x (TUWien,

2015). Monte Carlo integration (path-tracing) is a method that can be used to accurately simulate the rendering equation (Harrison, 2010). However, this method is not viable for real-time applications, so the light is approximated instead.

Diffuse reflection (d) scatters light in all directions, resulting in a rough surface, while specular reflection (s) reflects light in a single direction, resulting in a glossy or shiny surface. In computer graphics, an ambient reflectance term (a) is often added to ensure that surfaces do not appear completely unlit - this is typically a constant value. The intensity of light on a surface (i) is equal to the sum of all these reflections:

$$i = a + d + s \tag{3}$$

2.2 Basic Reflectance Models

The Lambertian model is the simplest diffuse BRDF model (Koppal, 2020). In this model, the reflectance is equal in all directions and the intensity of the reflected light is determined by the angle between the surface and the light source. The Lambertian diffuse value is calculated as follows with k_d being a constant that controls the diffuse component and $\vec{l}$ being the light direction (Blinn, 1977):

$$d = \underbrace{f_r(\vec{\omega_i}, x, \vec{\omega_r}) = \frac{k_d}{\pi}}_{\textit{Lambertian BRDF}} \underbrace{(\vec{n} \cdot \vec{l})\vec{\omega_i}}_{\textit{intensity of light}} \tag{4}$$

Phong adds specular highlighting to make it more physically accurate (Phong, 1975). And an improvement by Blinn makes this model more efficient (Blinn, 1977).

The specular term is computed by a dot product of the surface normal ($\vec{n}$) and the halfway vector ($\vec{h}$) to some power (^{c}). The halfway vector is calculated using the light ($\vec{l}$) and view directions ($\vec{v}$) as seen in equation 5. On a perfect mirror surface, the light would only reach the viewer if the surface normal equals the halfway vector. On less reflective surfaces, the specular component falls off slowly as the normal direction moves away from the specular direction (Phong, 1975 and Blinn, 1977).

$$\vec{h} = \frac{\vec{l} + \vec{v}}{length(\vec{l} + \vec{v})} \qquad s = (\vec{n} \cdot \vec{h})^c \tag{5}$$

2.3 Microfacet Models

Microfacet models introduce a detailed microsurface (made up of many so-called microfacets) that replaces the simplified macrosurface to make the surface appear rougher; figure 1 shows a comparison between the two surfaces. The microsurface is not predefined or modelled and is too small to be seen directly but affects how light is reflected through a series of functions (Walter et al., 2007). α being the roughness parameter.

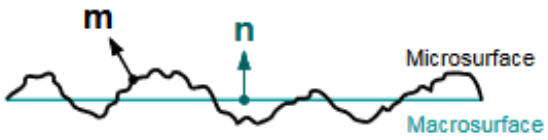


Figure 1: *Comparison of a micro and macro surface (Walter et al., 2007). $\vec{n}$ is the macrosurface normal, $\vec{m}$ is the normal of a microfacet.*

- **Distribution Function** $D(\vec{h}, \alpha)$ with $\int_\Omega D(\vec{h}, \alpha) = 1$
  Describes the distribution of the normals of the microsurface $\vec{m}$ that are aligned relative to vector $\vec{h}$ (Montes and Ureña, 2012).
- **Geometric Attenuation Factor** $G(\vec{v}, \vec{l}, \alpha) \in [0, 1]$
  Defines the ratio of visible facets on the surface due to shadowing or masking (Montes and Ureña, 2012). $\vec{v}$ being the view direction and $\vec{l}$ being the light direction.
- **Fresnel Reflection Function** $F(\vec{h}, \vec{v}, F_0) \in [0, 1]$
  Determines the fraction of light that is reflected and refracted at the interface between different surfaces (Walter et al., 2007 and Blinn, 1977). Where F_0 represents the base reflectivity of the surface (LearnOpenGL, n.d.).

A widely used Distribution Function is the Beckmann-Spizzichino model (Pharr, Jakob, and Humphreys, 2016):

$$D(\vec{h}, \alpha) = \frac{\exp(-\tan^2(\vec{n} \cdot \vec{h})/\alpha^2)}{\pi\alpha^2 \cos^4(\vec{n} \cdot \vec{h})} \tag{6}$$

Another commonly used Distribution Function is the Trowbridge and Reitz model (Trowbridge and Reitz, 1975 and Pharr, Jakob, and Humphreys, 2016) which is also known as the GXX model (Walter et al., 2007). Compared to the Beckmann Distribution, this model has a smoother falloff at grazing viewing angles, resulting in a more realistic rendering of certain surfaces.

$$D(\vec{h}, \alpha) = \frac{\alpha^2}{\pi((\vec{n} \cdot \vec{h})^2(\alpha^2 - 1) + 1)^2} \tag{7}$$

Heitz showed that the Geometric Function proposed by Smith is highly accurate (Heitz, 2014) as it accounts for both the geometry obstruction caused by the view direction and the geometry shadowing caused by the light direction (LearnOpenGL, n.d.). The function is as follows, with G_1 being set to the GGX formulation (Smith, 1967), i being the input parameter which is set to either $\vec{v}$ or $\vec{l}$:

$$G(\vec{v}, \vec{l}, \alpha) = G_1(\vec{v}, \alpha)G_1(\vec{l}, \alpha) \qquad G_1(i, \alpha) = \frac{2(\vec{n} \cdot i)}{\vec{n} \cdot i + \sqrt{\alpha^2 + (1 - \alpha^2)(\vec{n} \cdot i)^2}} \tag{8}$$

Finally, the Fresnel function is often approximated using the model proposed by Schlick (Schlick, 1994 and LearnOpenGL, n.d.):

$$F(\vec{h}, \vec{v}, F_0) = F_0 + (1 - F_0)(1 - (\vec{h} \cdot \vec{v}))^5 \tag{9}$$

The Torrance-Sparrow BRDF model uses V-shaped facets of equal length with random orientations to simulate specular reflections; the facet distribution is controlled k_s (Torrance and Sparrow, 1967 and Montes and Ureña, 2012). The model uses the Beckmann distribution, GGX geometry function and Schlick's Fresnel approximation.

$$f_r(\vec{\omega_i}, x, \vec{\omega_r}) = \frac{k_s}{4\pi(\vec{n} \cdot \vec{\omega_r})} D(\vec{h}, \alpha) G(\vec{v}, \vec{l}, \alpha) F(\vec{h}, \vec{v}, F_0) \tag{10}$$

This was later improved in the Cook-Torrance BRDF model which only considers the facets that are oriented towards the $\vec{h}$ as contributors to the final reflection (Cook and Torrance, 1982 and LearnOpenGL, n.d.). The model combines the D, G, and F functions of the Torrance-Sparrow model. The h vector simulates an imaginary surface that behaves as a perfect mirror, and the Fresnel term represents the reflection of polished surfaces (Montes and Ureña, 2012).

$$f_r(\vec{\omega_i}, x, \vec{\omega_r}) = \frac{D(\vec{h}, \alpha) G(\vec{v}, \vec{l}, \alpha) F(\vec{h}, \vec{v}, F_0)}{4(\vec{n} \cdot \vec{v})(\vec{n} \cdot \vec{l})} \tag{11}$$

The Oren-Nayar model improves on Lambertian diffuse reflections (Oren and Nayar, 1994 and Oren and Nayar, 1995). This model takes into account the angle between the incoming and outgoing light vectors, as well as the roughness of the surface. The model does not have a closed-form solution (Pharr, Jakob, and Humphreys, 2016), but this approximation fits well:

$$a = max(\vec{\omega_i}, \vec{\omega_r}) \qquad b = min(\vec{\omega_i}, \vec{\omega_r})$$
$$A = 1 - 0.5\frac{\alpha^2}{\alpha^2 + 0.33} \qquad B = 0.45\frac{\alpha^2}{\alpha^2 + 0.09}$$

$$f_r(\vec{\omega_i}, x, \vec{\omega_r}) = \frac{k_d}{\pi}(A + B \max(0, \cos(\phi_i - \phi_r)) \sin(a) \tan(b)) \tag{12}$$

3 Method

3.1 DirectX 11 Renderer

A small DirectX 11 renderer was written in C++ in order to simulate the appearance of clay in real time. Using the DirectX graphics API allows almost full control over the rendering pipeline while gaining low-level access to shaders and resources.

The program supports rendering a small scene where the user can fly around, which includes primitive objects loaded from OBJ files and a point light. These primitive objects each have a material, that with the inclusion of ImGUI, can be updated at run time to see live results.

3.2 Clay Appearance & Physically Based Rendering

Implementing Physically Based Rendering (PBR) into the renderer involves creating HLSL shaders that simulate light accurately by following a BRDF model. To help with understanding the PBR pipeline, the Blinn-Phong shading model was implemented first (see figure 2), along with a simple texture system to create some extra detail on the model.

Normal mapping was implemented to add extra detail that isn't modelled into the mesh (see figure 3) and was originally proposed by Blinn (1978). A normal map is a texture that defines a direction rather than a colour (Mikkelsen, 2020); its value is obtained from a texture lookup in the fragment shader. However, the value must be multiplied by a 3x3 Tangent ($\vec{T}$), Bitangent ($\vec{B}$), and Normal ($\vec{N}$) matrix ($TBN_{3\times3}$) calculated from the current vertex (Kilgard, 2000), see equation 13. The transformed normal, known as the perturbed normal (Blinn, 1978) replaces the vertex normal $\vec{N}$ as the surface normal $\vec{n}$ in the BRDF implementations.

$$TBN_{3\times3} = \begin{bmatrix} \vec{T}_1 & \vec{T}_2 & \vec{T}_3 \\ \vec{B}_1 & \vec{B}_2 & \vec{B}_3 \\ \vec{N}_1 & \vec{N}_2 & \vec{N}_3 \end{bmatrix} \tag{13}$$

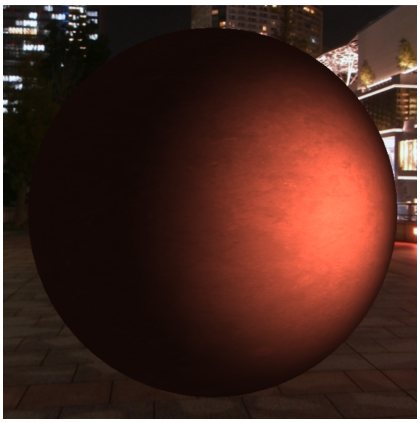


Figure 2: *First implementation of basic PBR using the Blinn-Phong lighting model.*

Figure 3: *A normal map texture is used when rendering the object with the Blinn-Phong lighting model.*

$\vec{N}$ is already known as part of the mesh, and $\vec{T}$ is pre-computed beforehand by equation 14. With $\vec{e}$ being the triangle edges calculated from each of the triangle's vertex positions ($\vec{p}$) and r being the reciprocal of the determinant of the UV differences.

$$\begin{aligned}\vec{e}_1 &= \vec{p}_1 - \vec{p}_0 \\ \vec{e}_2 &= \vec{p}_2 - \vec{p}_0 \\ \Delta u_1 &= u_1 - u_0 \\ \Delta u_2 &= u_2 - u_0 \\ \Delta v_1 &= v_1 - v_0 \\ \Delta v_2 &= v_2 - v_0\end{aligned} \tag{14}$$

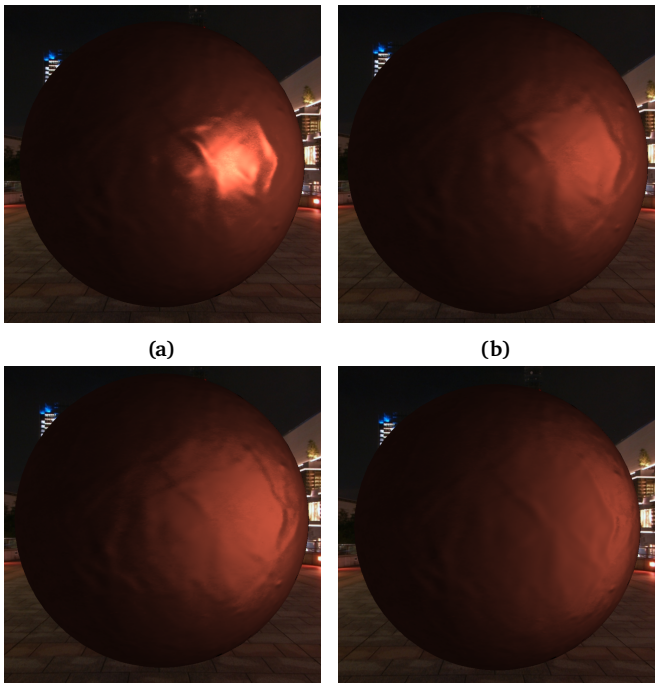
$$r = \frac{1}{\Delta u_1 \Delta v_2 - \Delta u_2 \Delta v_1}$$

$$\vec{T} = r(\vec{e}_1 \Delta v_2 - \vec{e}_2 \Delta v_1)$$

$\vec{B}$ is calculated as the cross between the tangent and normal at runtime in the fragment shader like so:

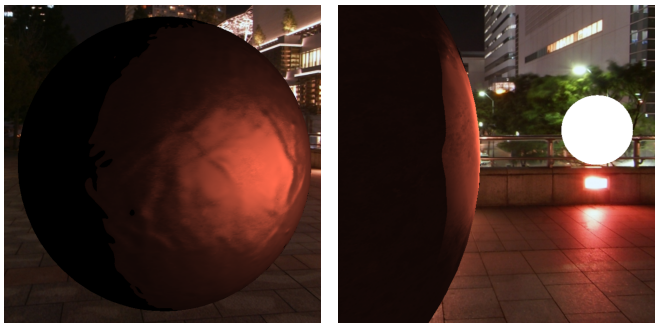
$$\vec{B} = \vec{N} \times \vec{T} \tag{15}$$

The specular component was improved to use the Cook-Torrance BRDF, figure 4 shows how different Fresnel and Specular roughness values affect the specular highlight on the object. Figure 4a shows a shinier specular highlight compared to 4d which is very rough.



(a) **(b)** **(c)** **(d)**

Figure 4: *Object rendered using the Cook-Torrance BRDF.*
4a Fresnel roughness of 0.1, Specular roughness of 0.3.
4b Fresnel roughness of 0.5, Specular roughness of 0.5.
4c Fresnel roughness of 0.2, Specular roughness of 0.7.
4d Fresnel roughness of 1.0, Specular roughness of 1.0.

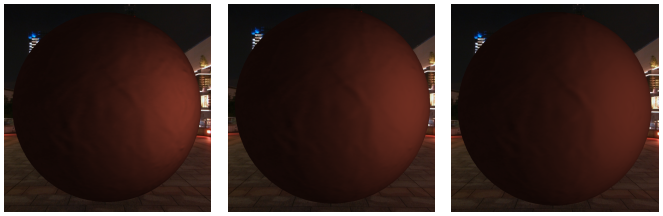


(a) **(b)**

Figure 5: *The specular component causing incorrect lighting.*

Implementing the Cook-Torrance with the normal mapping caused incorrect light calculations on any part of the object where $(\vec{n} \cdot \vec{l}) <= 0.0$ causing the object to be rendered completely black (see figure 5a). This was fixed by hard-coding the specular component to '0.0' when this occurs as proposed in the Unity forums (Wolfram, 2011). However, a harsh cutoff line still appears when viewing the object at certain angles (see figure 5b).

The Oren-Nayar BRDF was implemented to give roughness to the diffuse component. Figure 6 shows a comparison of the different roughness values when using the Oren-Nayar BRDF, to more easily compare the specular component has been disabled.



(a) *Roughness of 0.0.* **(b)** *Roughness of 0.5.* **(c)** *Roughness of 1.0.*

Figure 6: *Objects rendered using the Oren-Nayar BRDF.*

Further detail was added by including another normal map and blending between the two. The normal maps are blended together using the following equation, where v is a value retrieved from a normal map:

$$\vec{n} = (v_1 + v_2) \begin{bmatrix} 1.0 \\ 1.0 \\ 0.5 \\ 0.5 \end{bmatrix} \tag{16}$$

To imitate the effect of people touching modelling clay, a fingerprint normal map was used in order to replicate the surface texture (see figure 7).

There are also additional parameters to change the uv scaling and offsets of the normal textures, which simply multiply and add to the base uv respectfully.

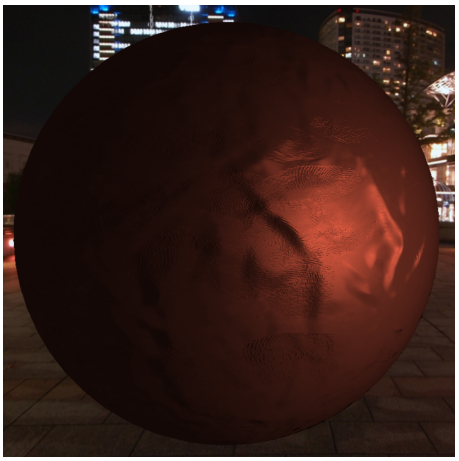


Figure 7: *The default normal texture with a "fingerprint" normal map applied for extra detail.*

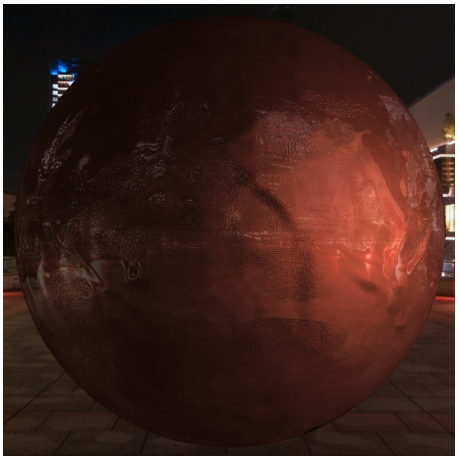
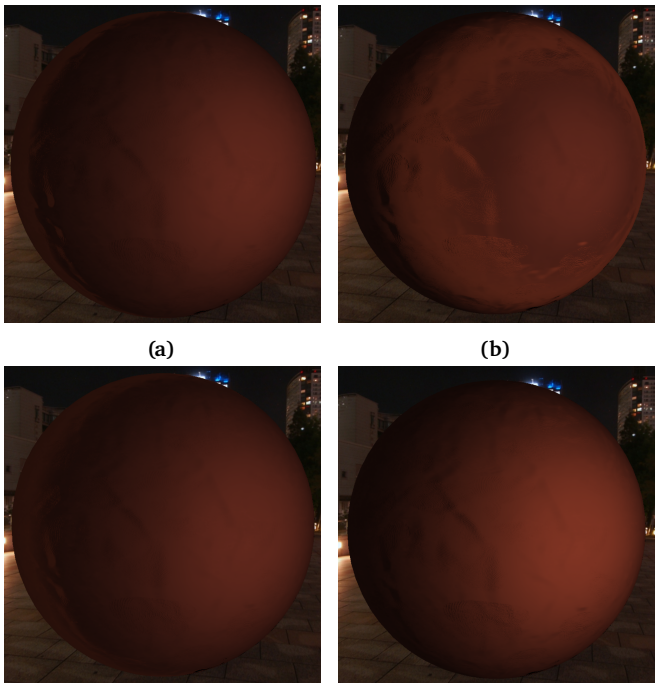


Figure 8: *Environment reflections.*

Environment reflections were also added, see figure 8. By including the skybox cubemap in the fragment shader it can be sampled by calculating the $\vec{uv}$ depending on the current view direction, as seen in equation 17. The reflections are controlled by a parameter and a noise texture to give them some amount of randomness.

$$\begin{aligned} \vec{uv} &= \vec{n}(2.0(\vec{n} \cdot \vec{v}) - \vec{v} \\ \hat{uv} &= \frac{\vec{uv}}{\|\vec{uv}\|} \end{aligned} \tag{17}$$

Subsurface Scattering (see figure 9) was achieved by using an efficient approximation named *wrap lighting* (Fernando et al., 2004). This modifies the diffuse function so the light wraps around the object where it would normally appear dark, this is when the normal is perpendicular to the light direction ($(\vec{n} \cdot \vec{l}) = 0.0$). This approximation is calculated by equation 18, with w_1 being a number between 0 and 1 that controls how far the light will wrap, w_2 being the width of the wrapping and s being the calculated scatter amount which is multiplied by a colour and an intensity before being added to the diffuse component.



(a) **(b)** **(c)** **(d)**

Figure 9: *Object rendered with subsurface scattering.*
9a Wrap amount of 0.5, Width of 0.2.
9b Wrap amount of 0.0, Width of 0.4.
9c Wrap amount of 0.4, Width of 0.2.
9d Wrap amount of 0.0, Width of 1.0.

$$\begin{aligned} x &= max(0.0, ((\vec{n} \cdot \vec{l}) + w_1)/(1.0 + w_1)) \\ s &= smoothstep(0.0, w_2, x)smoothstep(w_2 2.0, w_2, x); \end{aligned} \tag{18}$$

4 Future Work

One potential future direction would be to explore techniques such as path tracing and Monte Carlo integration to further improve the physical accuracy of rendering materials. It is unclear if the project would remain real-time as these techniques are commonly quite expensive.

The inclusion of a material system could allow for the quick iteration of different types of models. Typically, a renderer includes both metallic and dielectric BRDF materials. By adjusting the inputted material's "metallic" and "roughness" values, these BRDFs can be blended to create more lifelike materials. Additionally, incorporating a texture allows for distinct regions of the mesh to be rendered with different properties.

There are more advanced and complete BRDF shading models which were not discussed or implemented in this project; using these additional models could achieve a more realistic result. For example, the He-Torrance-Sillion-Greenberg BRDF considers the wave-like nature of light through diffraction and interference (Montes and Ureña, 2012).

5 Evaluation

Comparing the rendered clay with real-life modelling clay from the Chicken Run movie (see figure 10 and 11), it is clear that the implemented normal maps have successfully captured some of the fine-grain details of clay material. Specifically, Figures 11b and 11c have a close resemblance to the natural patterns found in clay as well as the fingerprints that can be seen on the hands in figure 10.

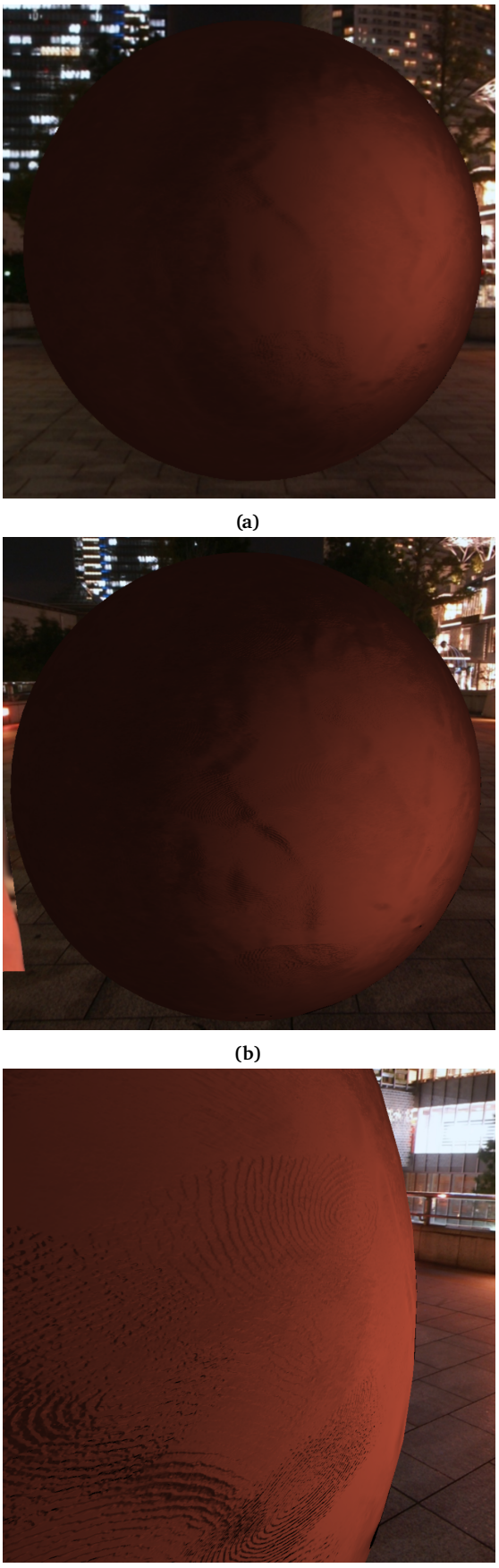
However, there are some differences between the rendered clay and the reference image. For example, the specular highlights on the beak areas in Figure 10 are noticeably stronger than those in the render. Although, it is important to note that these parameters can be adjusted at runtime to achieve the desired result.

The render accurately depicts the way light interacts with rough materials thanks to the use of the Cook-Torrance and Oren-Nayar BRDF models. This helps to create a more realistic and believable representation of clay material, enhancing the overall quality of the render.

6 Conclusion

In conclusion, the renderer created is capable of simulating clay materials that appear very close to reality. It incorporates some advanced BRDF models, normal mapping, environment reflections, and basic subsurface scattering, making it a promising start towards a PBR renderer. However, considering the extensive research on BRDFs, there is still plenty of room for improvement to achieve even greater physical accuracy.

Figure 10: *A screenshot captured from the Chicken Run movie (Aardman Animations, 2000), showing the main character made from modelling clay.*



(a) **(b)** **(c)**

Figure 11: *Final screenshots of clay renderer.*

References

Aardman Animations (2000). *Chicken Run*.

Blinn, James F (1977). "Models of light reflection for computer synthesized pictures". In: *Proceedings of the 4th annual conference on Computer graphics and interactive techniques*, pp. 192–198.

– (1978). "Simulation of wrinkled surfaces". In: *ACM SIGGRAPH computer graphics* 12.3, pp. 286–292.

Cook, Robert L and Kenneth E. Torrance (1982). "A reflectance model for computer graphics". In: *ACM Transactions on Graphics (ToG)* 1.1, pp. 7–24.

Duvenhage, Bernardt, Kadi Bouatouch, and Derrick G Kourie (2013). "Numerical verification of bidirectional reflectance distribution functions for physical plausibility". In: *Proceedings of the South African Institute for Computer Scientists and Information Technologists Conference*, pp. 200–208.

Fernando, Randima et al. (2004). *GPU gems: programming techniques, tips, and tricks for real-time graphics*. Vol. 590. Addison-Wesley Reading.

Harrison, Robert L (2010). "Introduction to monte carlo simulation". In: *AIP conference proceedings*. Vol. 1204. 1. American Institute of Physics, pp. 17–21.

Heitz, Eric (2014). "Understanding the masking-shadowing function in microfacet-based BRDFs". In: *Journal of Computer Graphics Techniques* 3.2, pp. 32–91.

Kilgard, Mark J (2000). "A practical and robust bump-mapping technique for today's GPUs". In: *Game Developers Conference 2000*, pp. 1–39.

Koppal, Sanjeev J (2020). "Lambertian reflectance". In: *Computer vision: a reference guide*, pp. 1–3.

Lafortune, Eric (1996). "Mathematical models and Monte Carlo algorithms for physically based rendering". In: *Department of Computer Science, Faculty of Engineering, Katholieke Universiteit Leuven* 20, pp. 74–79.

LearnOpenGL (n.d.). *PBR Theory - LearnOpenGL*. URL: https://learnopengl.com/PBR/Theory.

Mikkelsen, Morten S (2020). "Surface Gradient–Based Bump Mapping Framework". In: *Journal of Computer Graphics Techniques Vol* 9.3.

Montes, Rosana and Carlos Ureña (2012). "An overview of BRDF models". In: *University of Grenada, Technical Report LSI-2012-001*.

Oren, Michael and Shree K Nayar (1994). "Generalization of Lambert's reflectance model". In: *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pp. 239–246.

– (1995). "Visual appearance of matte surfaces". In: *Science* 267.5201, pp. 1153–1156.

Pharr, Matt, Wenzel Jakob, and Greg Humphreys (2016). *Physically based rendering: From theory to implementation*. Morgan Kaufmann.

Phong, Bui Tuong (1975). "Illumination for computer generated pictures". In: *Communications of the ACM* 18.6, pp. 311–317.

Schlick, Christophe (1994). "An inexpensive BRDF model for physically-based rendering". In: *Computer graphics forum*. Vol. 13. 3. Wiley Online Library, pp. 233–246.

Smith, Bruce (1967). "Geometrical shadowing of a random rough surface". In: *IEEE transactions on antennas and propagation* 15.5, pp. 668–671.

Torrance, Kenneth E and Ephraim M Sparrow (1967). "Theory for off-specular reflection from roughened surfaces". In: *Josa* 57.9, pp. 1105–1114.

Trowbridge, TS and Karl P Reitz (1975). "Average irregularity representation of a rough surface for ray reflection". In: *JOSA* 65.5, pp. 531–536.

TUWien (2015). *TU Wien rendering #3 - BRDF models, the rendering equation - youtube*. URL: https://www.youtube.com/watch?v=4gXPVoippTs.

Walter, Bruce et al. (2007). "Microfacet models for refraction through rough surfaces". In: *Proceedings of the 18th Eurographics conference on Rendering Techniques*, pp. 195–206.

Wolfram (2011). *BUG in all Specular shaders causing highlights from light sources behind objects - Unity Forums*. URL: https://forum.unity.com/threads/bug-in-all-specular-shaders-causing-highlights-from-light-sources-behind-objects.73125/#post-470278.

Wynn, Chris (2000). "An introduction to BRDF-based lighting". In: *Nvidia Corporation*.