Creating a Realistic Clay Shader for Digital Stop Motion

Zoe Rowbotham
18014180

University of the West of England

February 19, 2022

Learning Unity Shaders and Physically Based Rendering through creation of a realistic clay shader for use in stop motion animations.

1 Introduction

Some of the most famous stop motion animations and characters have been created by Aardman Animations, an animation studio created by Peter Lord and David Sproxton in 1972 (AardmanAnimations, 2021). Aardman Animations is the creator of Wallace and Gromit and the feature "The Wrong Trousers" which is one of the most successful animated films ever made (AardmanAnimations, 2021).

2007s Flushed Away marked Aardman Animations first computer generated feature (AardmanAnimations, 2021). With the improvement of digital technologies, the importance of a realistic clay material that appears and behaves as modelling clay would in real life has become vital for digital claymation.

2 Related Work

2.1 Clay Material Properties

Aardman Animations now use a combination of materials for their stop motion including modelling clay, silicon and hard polyurethane (Savage, 2018), however originally just used modelling clay for their stop motion characters.

The best kind of clay to use for a stop motion animation is a non-hardening, oil based clay, like Plasticine. Aardman Animations use their own clay mix called "Aardmix", which is theorised to be based off of the oil based clay "Newplast" with additional oils (IKITMOVI, 2018).

Figure 1: *Aardman Animations clay figures (Savage, 2018).*

Modelling clay is made from a lubricant, binder and base; recipes commonly use calcium carbonate, wax, petroleum and mineral oil (Romi, 2010).

Modelling clay is generally matte, with slight translucent properties; when wet or oily, modelling clay produces more specular light. It is very hard to get modelling clay completely smooth, even with moulds (Howell, Child, and Hall, 2014), and so there is always a roughness to the surface as well as folds, fingerprint indents and general bumps.

2.2 Physically Based Rendering

Physically Based Rendering (PBR), is an approach producing realistic interactions between light and a scene (Pharr, Jakob, and Humphreys, 2016). For a PBR model to be considered physically based, it has to meet three constraints: be based on the microfacet surface model, be energy conserving and use a physically based

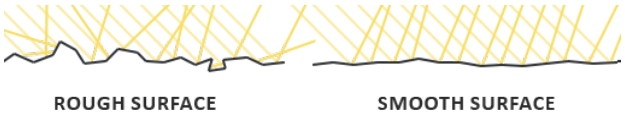

Figure 2: *Diagram showing microfacets and how they reflect light rays (Vries, 2016b).*

BRDF (Vries, 2016b).

2.2.1 Microfacet Reflection Model

The Cook-Torrance reflectance model introduced microfacet reflection models to computer graphics in 1981, allowing metallic surfaces to be rendered accurately (Pharr, Jakob, and Humphreys, 2016). The microfacet model states that any surface can be described on a microscopic scale as many reflective mirrors, the alignment of the mirrors (microfacets) determines the roughness, or smoothness, of a surface (Vries, 2016b). The way microfacets reflect light rays can be seen in Figure 2.

2.2.2 Energy Conservation

Energy conservation simply means the amount of light reflected from a surface cannot exceed the amount of incoming light to the surface. When light hits a surface it either reflects (specular) or refracts (diffuse) as seen in Figure 3.

To ensure the conservation of energy, the amount of specular can be calculated, which can then lead to the calculation of the diffuse amount by subtracting the specular from 1.

To calculate the amount of light at a position in a view direction, the Rendering Equation (created by Kajiya, 1986) can be used, found in Equation 1.

$$L_o(x,\omega_o) = L_e(x,\omega_o) + \int_\Omega f_r(x,\omega_i,\omega_o) L_i(x,\omega_i) \omega_i \cdot n d\omega_i$$

Where:
$L_o(x,\omega_o) = TotalOutgoingLight$
$x = Position$
$\omega_o = OutgoingLightDirection$
$L_e(x,\omega_o,\lambda,t) = EmittedLight$
$\Omega = UnitHemisphere$
$f_r(x,\omega_i,\omega_o) = BRDF$
$\omega_i = IncomingLightDirection$
$L_i(x,\omega_i) = IncomingLight$
$n = SurfaceNormal$
(1)

The rendering equation calculates the total amount of light at point x along ω_o by calculating the emitted light of the surface plus the sum of all the incoming light rays. $f_r(x,\omega_i,\omega_o)$ refers to a BRDF (Bidirectional Reflective Distribution Function) which is explained in the next section.

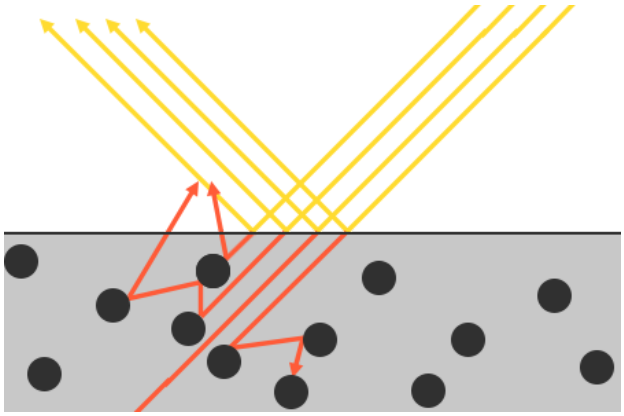

Figure 3: *How light reacts when it hits a surface, yellow lines reflect, red lines refract. (Vries, 2016b).*

2.2.3 Physically Based BRDF

A BRDF (Bidirectional Reflective Distribution Function) calculates the reflective lights of a material (Howell, Child, and Hall, 2014). A physically based BRDF must also adhere to the energy conservation principle (Vries, 2016b).

The Cook Torrance BRDF was developed in 1982 by Robert Cook and Kenneth Torrance (Cook and Torrance, 1982), the equation for the Cook-Torrance BRDF can be seen in Equation 2. Both Unreal Engine 4 and Disney use the COok-Torrance BRDF model in their implementations (Karis, 2013, Burley, 2012).

$$f_r(x,\omega_i,\omega_o) = k_d \frac{c}{\pi} + \frac{DFG}{4(\omega_o \cdot n)(\omega_i \cdot n)}$$

Where:
$k_d = RefractedLight$
$c = SurfaceColour$
$x = Position$
$\omega_i = IncomingLightDirection$
$\omega_o = OutgoingLightDirection$
$D = NormalDistributionFunction$
$F = FresnelEquation$
$G = GeometryFunction$
(2)

The Normal Distribution Function, or facet slope distribution function (Cook and Torrance, 1982), approximates the number of microfacets aligned to the halfway vector h (calculated in Equation 4. A Normal Distribution Function called the Trowbridge-Reitz GGX can be found in Equation 3. The halfway vector h found in the Trowbridge-Reitz GGX is calculated by Equation 4.

$$D_{TRGGX}(n, h, \alpha) = \frac{\alpha^2}{\pi((n \cdot h)^2(\alpha^2 - 1) + 1)^2}$$

Where:
n $= SurfaceNormal$
h $= HalfwayVector$
$\alpha = SurfaceRoughness$
(3)

$$h = \frac{l + v}{||l + v||}$$

Where:
l $= Light$
v $= ViewDirection$
(4)

The Fresnel Equation describes the ratio of reflected light, which can then be used to calculate the percentage of refracted light due to the energy conservation principle.

```
1 float specular = FresnelEquation(
    halfwayVector, viewDirection,
    baseReflectivity);
2 float diffuse = 1 - specular;
```

The Fresnel equation has been approximated by Schlick, 1994 producing the formula seen in Equation 5. Schlick, 1994 states this formula is inexpensive and well suited to computer graphics.

$$F_{Schlick}(h, v, F_0) = F_0 + (1 - F_0)(1 - (h \cdot v))^5$$

Where:
h $= HalfwayVector$
v $= ViewDirection$
$F_0 = BaseReflectively$
(5)

The Geometry Function, or the geometrical attenuation factor (Cook and Torrance, 1982), approximates the relative surface area where microfacets obstruct and shadow each other, causing light rays to be blocked. Schlick, 1994 states the original geometrical attenuation coefficient from Cook and Torrance, 1982 uses approximations about a surface and so seeks to improve upon it. The Schlick-GGX Geometry Function can be found in Equation 6. Where k is a manipulated roughness value; the way the roughness is manipulated is determined by the use of direct or IBL lighting (these calculations can be seen in Equation 7 and Equation 8.

$$G_{SchlickGGX}(n, v, k) = \frac{n \cdot v}{(n \cdot v)(1 - k) + k}$$

Where:
n $= SurfaceNormal$
v $= ViewDirection$
k $= RoughnessRemapping$
(6)

$$k_{direct} = \frac{(\alpha + 1)^2}{8}$$

Where:
$\alpha = Roughness$
(7)

$$k_{IBL} = \frac{\alpha^2}{2}$$

Where:
$\alpha = Roughness$
(8)

Howell, Child, and Hall, 2014 use the Cook-Torrance BDRF with come modifications to produce a more realistic model for rendering Plasticine. The improvements are made through modifications of the geometrical attenuation factor as well as the Fresnel equation. This project utilises the equations given by Howell, Child, and Hall, 2014 to produce a realistic clay material.

2.3 Layered BRDF

Weidlich and Wilkie, 2007 describe a layered BRDF that provides a way to layer two materials on top of each other. Figure 4 shows how the reflection of each layer is calculated. The reflection calculation of the bottom layer takes in the reflected incoming and outgoing light directions of the layer above it. The two layers are blended using the fresnel transmission ($T12$), internal reflection (t) and absorption (a) terms. Equation 9 shows how to calculate the total BRDF of two layers.

$$f_r = f_{r1}(\theta_i, \theta_r) + T12 f_{r2}(\theta_{i0}, \theta_{r0})at$$

Where:
$f_{r1}(\theta_i, \theta_r) = LayerOneBRDF$
$\theta_i = IncomingLightDirection$
$\theta_r = OutgoingLightDirection$
T12 $= FresnelTransmission$
$f_{r2}(\theta_{i0}, \theta_{r0}) = LayerTwoBRDF$
$\theta_{i0} = IncomingLightFromLayerOne$
$\theta_{r0} = OutgoingLightFromLayerTwo$
a $= AbsoprtionValue$

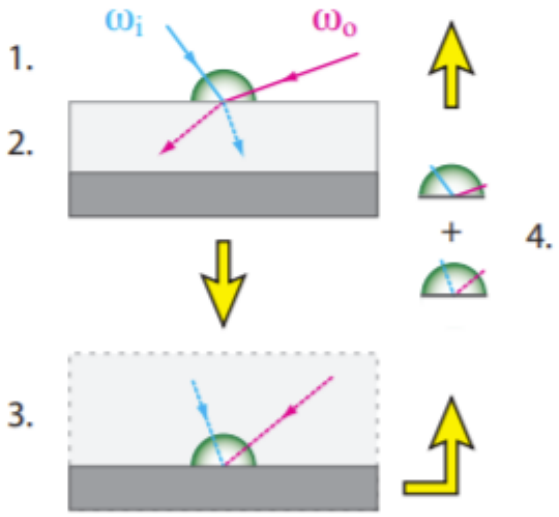

Figure 4: *Diagram of the layered BRDF calculated by Weidlich and Wilkie, 2007*

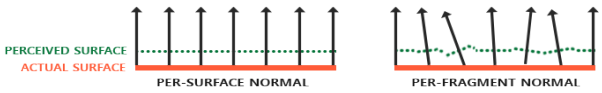

Figure 5: *Normal Mapping diagram from Vries, 2016a*

$$t = TotalReflectionValue \tag{9}$$

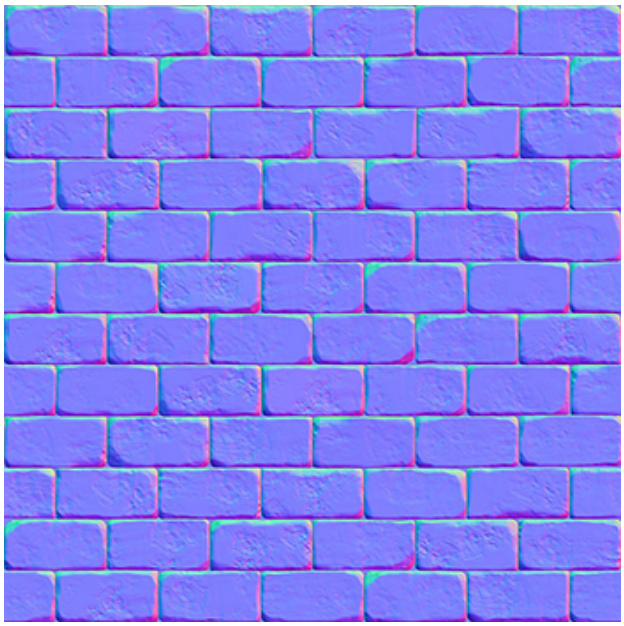

Figure 6: *Normal map example for a brick texture from Vries, 2016a*

2.4 Normal Mapping

Normal mapping refers to manipulating the normals of a surface to give the appearance of bumps, cracks and dents without changing the actual geometry of the surface.

Normal mapping requires a normal map which is a 2D texture with a mostly blue tint, representing a normal positive along the Z axis; pixels with a greenish or reddish tint represent a normal offset on the X and/or Y axis as seen in Figure 6. Using this information normals can be manipulated in the fragment shader.

3 Method

3.1 Initial Shadergraph Implementation

Creating materials in Shadergraph is designed to be quick and easy; creating an initial material in Shadergraph to achieve a desired look can help to write ShaderLab code.

Figure 7 shows the results of a simple clay shader made in Shadergraph. In figure 7 vertices are displaced by a noise value and a fingerprint texture that modifies the normal and the smoothness of a pixel.

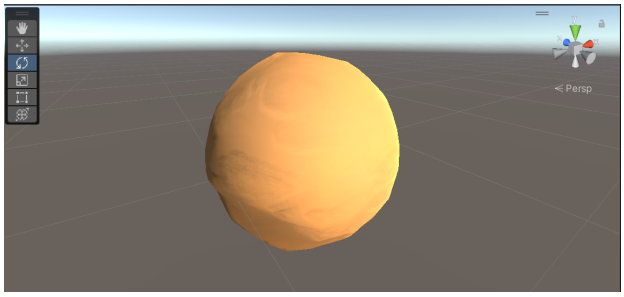

Figure 7: *Initial clay material made in Shader Graph*

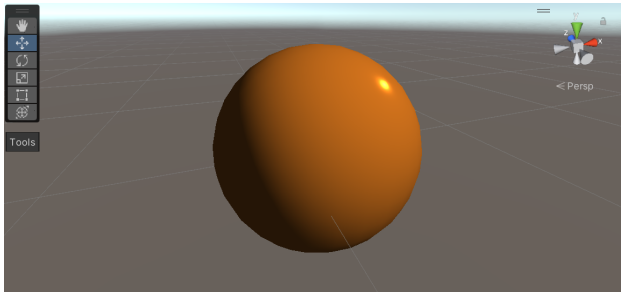

Figure 8: *Initial implementation of a physically based material.*

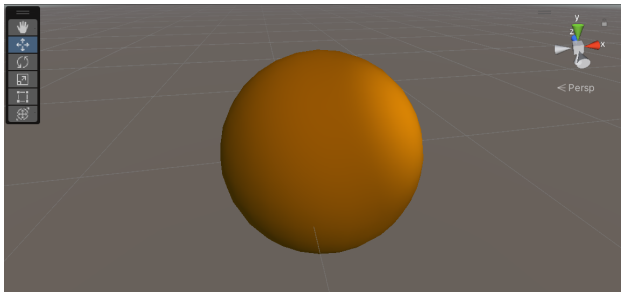

Figure 9: *Implementation of the layered BRDF described by Weidlich and Wilkie, 2007*

3.2 Creating a PBR Shader

A shader in Unity is made from a *.shader* file containing a shader object (an instance of the shader class). The shader object contains vertex and fragment functions can manipulate the vertex position and the pixel colour. These functions are defined by a *pragma* statement.

Implementing the formula shown in Equation 1 creates a physically based shader that interacts with the light in the scene, this can be seen in Figure 8.

3.3 Adding A Layer

An implementation of the layered BRDF equation derived by Weidlich and Wilkie, 2007 can be seen in figure 9 which has a better clay appearance compared to figure 8. The top layer can be used to simulate the reflections of water in the microscopic layers of clay.

The roughness values of each layer can be manipulated in the inspector as well as the base reflection value and the thickness of layer one (which affects the absorption value).

3.4 Normal Mapping

Figure 10 shows adding a normal map and a height map to the shader which adds more depth to the clay-like texture. The height map currently used gives some variation in the colour which gives the affect of different heights in the surface.

Both the height map and the normal map manipulate the normals of the surface before the lighting

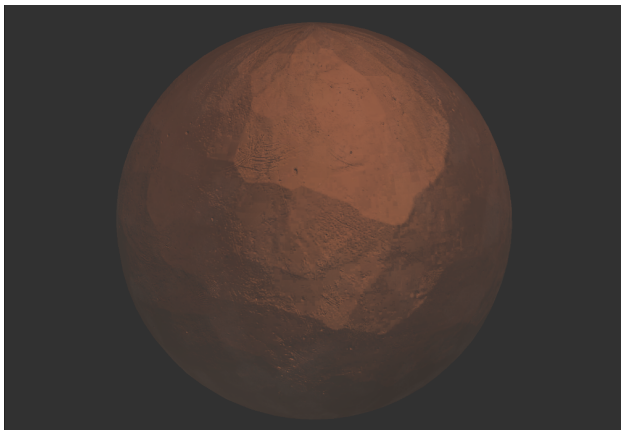

Figure 10: *Adding textures and normal mapping.*

calculation takes place, which affects how the surface is lit.

The code below shows how the height map is used to manipulate the normals in Unity's shaderlab. Adding a strength multiplier controls the intensity of the change in the normals.

```
1  // Normals Calculation
2  float3 normal;
3  normal.xy = tex2D(_BumpMap, input.UV
       ).wy * 2 - 1;
4  normal.xy *= _BumpStrength;
5  normal.z = sqrt(1 - saturate(dot(
       normal.xy, normal.xy)));
6
7  float3 binormal = cross(input.Normal
       , input.Tangent.xyz) *
8       (input.Tangent.w *
       unity_WorldTransformParams.w);
9
10 input.Normal = normalize(
11      normal.x * input.Tangent +
12      normal.y * binormal +
13      normal.z * input.Normal
14 );
```

3.5 Fingerprint Overlay

Modelling clay used for claymation will also have slight fingerprints from the modelling process, by using a fingerprint texture and adding the colour value to the final clay colour, fingerprints appear on the surface of the clay. Adding a scale factor means the intensity of the fingerprints can be controlled.

4 Evaluation

Figure 12 shows the final clay shader while Figure 13 shows a screenshot from the Aardman Animations and Greenpeace collaboration "Turtle Journey". Both figures have a slight specular to them due to the water

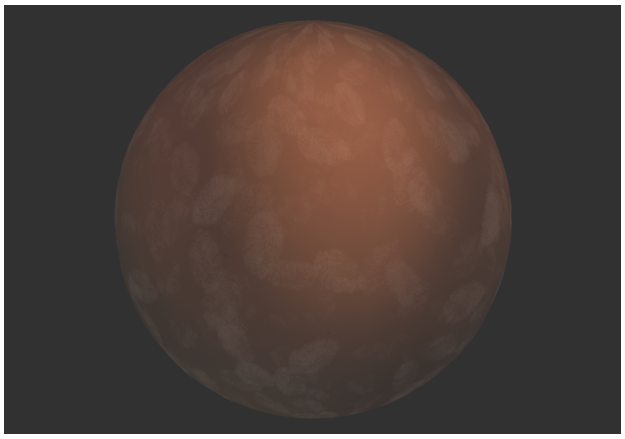

Figure 11: *The fingerprint texture addition with a high strength scale factor for visibility.*

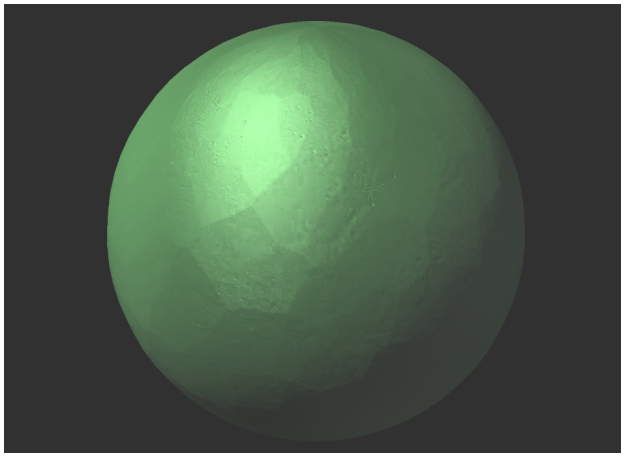

Figure 12: *Final clay shader.*

Figure 13: *Screenshot from Aardman Animations and Greenpeace Digital Claymation Film "Turtle Journey" (*Turtle Journey*)*

layers in clay, the specular in Figure 12 is diffused due to the layered BRDF creating a similar look to Figure 13. The fresnel falloff is steeper in Figure 12 producing darker edges than in the Aardman animations clay, which could be due to the parameters of the material or the microfacet model used (Torrance-Sparrow).

5 Conclusion

This project aimed to create a realistic clay shader for use in digital stop motion animations. The result is a clay shader that adheres to the constraints of physically based rendering, uses a layered BRDF and gives the appearance of clay. Designers can modify the roughness of each layer to achieve wet or dry looking clay as well as modifying the intensity of the fingerprints and normals. A greater understanding of shaders, materials and lighting was also achieved through this project.

Bibliography

AardmanAnimations (2021). *Aardman: A Little Bit of History*. URL: https://www.aardman.com/the-studio/history/ (visited on 12/02/2021).

Burley, Brent (2012). "Physically Based Shading at Disney". In: *SIGGRAPH 2012*. Walt Disney Animation Studios.

Cook, R. L. and K. E. Torrance (1982). "A Reflectance Model for Computer Graphics". In: *ACM Trans. Graph.* 1.1, 7–24. ISSN: 0730-0301. DOI: 10.1145/357290.357293. URL: https://doi-org.ezproxy.uwe.ac.uk/10.1145/357290.357293.

Howell, Lindsey, Philip Child, and Peter Hall (2014). "Plasticine Shading". In: *Proceedings of the 11th European Conference on Visual Media Production*. London, United Kingdom: Association for Computing Machinery. URL: https://doi-org.ezproxy.uwe.ac.uk/10.1145/2668904.2668933.

IKITMOVI (2018). *Best Clay to use for Claymation Animation*. URL: https://www.ikitmovie.com/best-clay-to-use-for-claymation-animation/ (visited on 12/02/2021).

Kajiya, James T. (1986). "The Rendering Equation". In: *SIGGRAPH Comput. Graph.* 20.4, 143–150. ISSN: 0097-8930. DOI: 10.1145/15886.15902. URL: https://doi-org.ezproxy.uwe.ac.uk/10.1145/15886.15902.

Karis, Brian (2013). "Real Shading in Unreal Engine 4". In: *SIGGRAPH 2013*. Epic Games.

Pharr, Matt, Wenzel Jakob, and Greg Humphreys (2016). *Physically Based Rendering: From Theory to Implementation*. Elsevier Science Technology.

Romi, Gomi (2010). *Homemade (Oil-based) Modelling Clay*. URL: https://www.instructables.com/Homemade-Oil-based-Modelling-Clay/ (visited on 12/02/2021).

Savage, Adam (2018). *The Clay in Stop-Motion Animation at Aardman Studios*. URL: https://www.youtube.com/watch?v=E5t_DXgZmFM&ab_channel=AdamSavage%E2%80%99sTested (visited on 12/02/2021).

Schlick, C (1994). "An Inexpensive BRDF Model for Physically-based Rendering". In: *Computer Graphics Forum* 12.3, pp. 233–246.

Turtle Journey.

Vries, Joey de (2016a). *Normal Mapping*. Last accessed 08 February 2022. URL: https://learnopengl.com/Advanced-Lighting/Normal-Mapping.

– (2016b). *PBR: Theory*. Last accessed 26 November 2021. URL: https://learnopengl.com/PBR/Theory.

Weidlich, Andrea and Alexander Wilkie (2007). "Arbitrarily Layered Micro-Facet Surfaces". In: *Proc. of the 5th International Conference on Computer Graphics and Interactive Techniques in Australasia and Southeast Asia*. Perth, Australia: ACM SIGGRAPH.